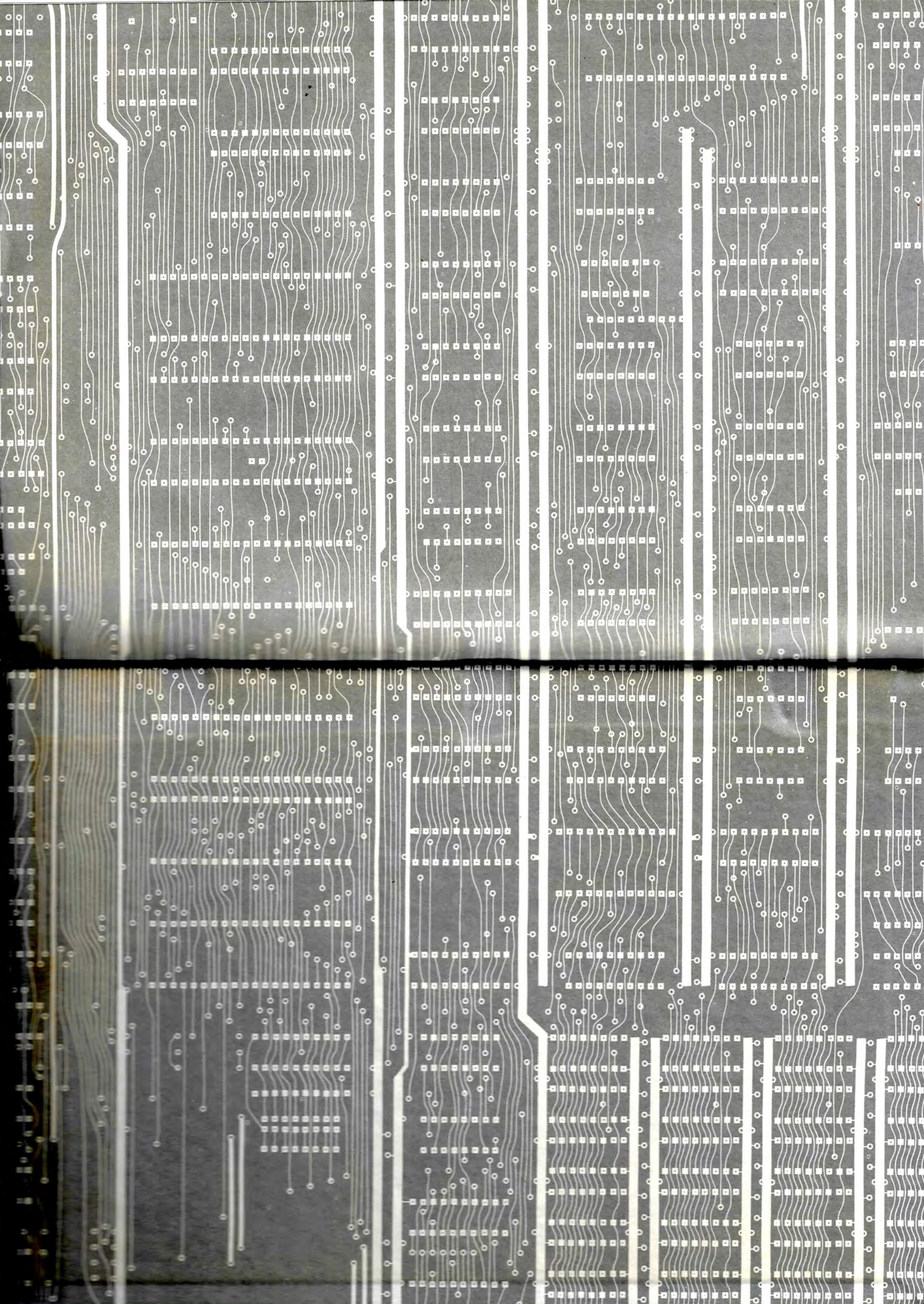
BASIC

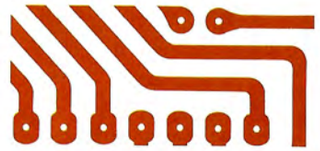
ENCICLOPEDIA DE LA INFORMATICA
MINIORDENADORES Y ORDENADORES PERSONALES

EDICIONES FORUM



BASIC

ENCICLOPEDIA DE LA INFORMATICA.
MINIORDENADORES Y ORDENADORES PERSONALES

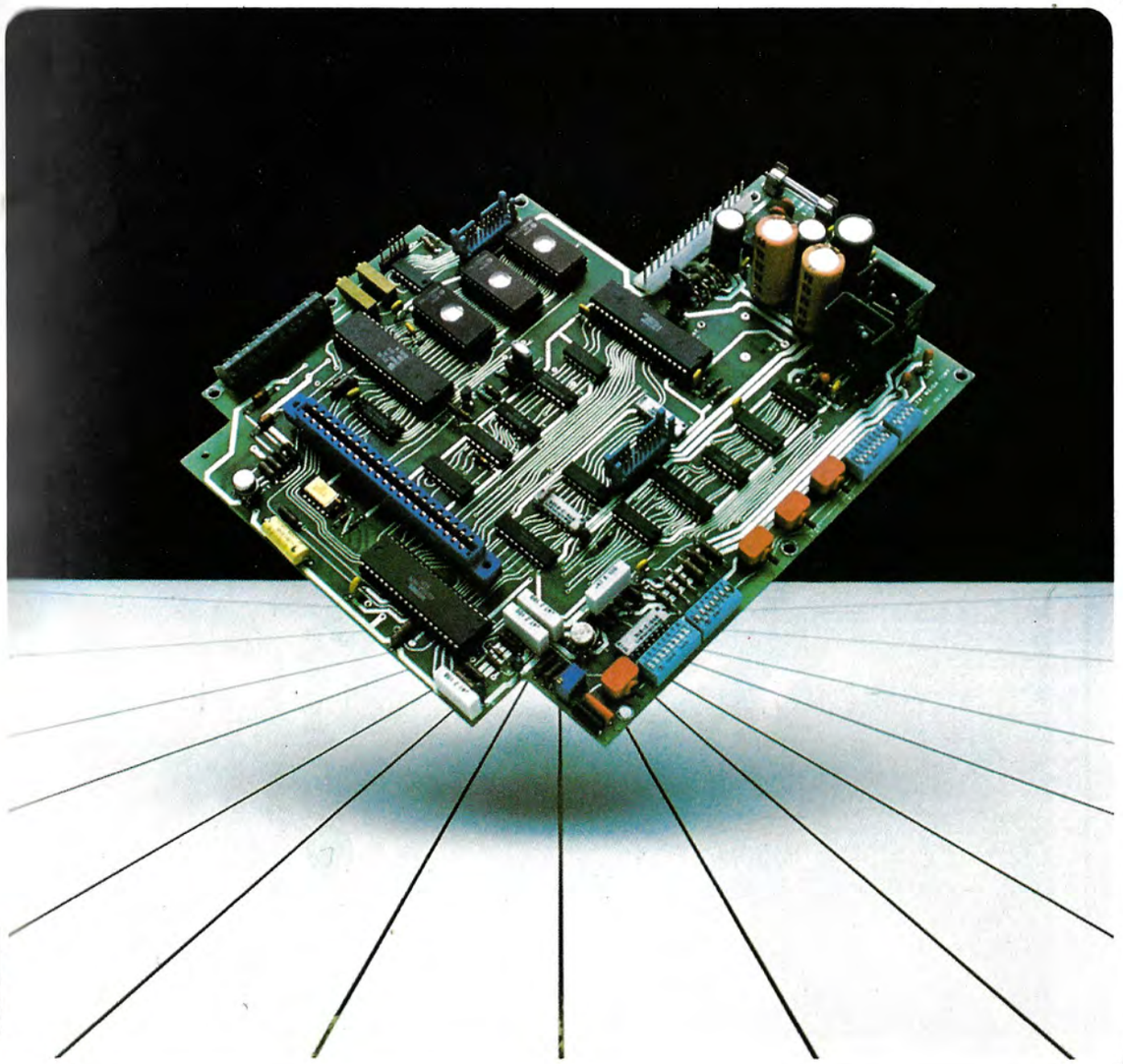


ENCICLOPEDIA DE LA INFORMATICA.
MINIORDENADORES
Y ORDENADORES PERSONALES

BASIC 3

Commodore

C. O'Rear/West Light-Grazia Neri



J. Kane/Meka

BASIC

ENCICLOPEDIA DE LA INFORMATICA.
MINIORDENADORES
Y ORDENADORES PERSONALES

Presidente
José Manuel Lara

Director Ejecutivo
Jesús Domingo

Dirección editorial
R.B.A., Proyectos Editoriales, S.A.

Dirección técnica
Sante Senni

Con el asesoramiento de la Sociedad **E.G.S.**

REDACCION

Dirección editorial: Gabriella Costarelli
Redactor jefe: Marcella Marcaccini
Secretaria de redacción: Giulia Abriani, Giovanna Aloisi
Revisión: Ugo Spezia
Corrección: Maria Albergo, Laura Salvini, Graziella Tassi
Recopilación material gráfico: Carla Bertini, Rossella Pozza
Producción: Piergiorgio Palma
Secretaria de producción: Maria Rita Ciucci
Diseño y dirección artística: Vittorio Antinori
Jefe estudio gráfico: Roberto Sed
Compaginación: Alberto Berni, Riccardo Catani, Patrizia Fazio
Dibujos: Renato Lazzarini, Gianni Mazzoleni, Rolando Mazzoni
Traducción: Carlo Frabetti
Diseño cubierta: Neslé Soulé
Jefe de Producción: Ricardo Prats

Composición electrónica: ITC-Fototipo. Barcelona-Madrid.
Imprime: CAYFOSA - Carretera de Caldas, Km. 3,7 - Santa Perpetua de Mogoda (Barcelona)
Depósito Legal: B. 37.099/83
ISBN (Obra completa): 84-7574-040-5
ISBN (Volumen III): 84-7574-168-1
ISBN (Fascículos): 84-7574-044-8
Impreso en Espa;a - Printed in Spain
Distribuye: Marco Ibérica, Distribuidora de Ediciones, S.A.
Carretera de Irún, Km. 13,350, variante de Fuencarral, MADRID-34

El editor agradece la colaboración de:
Alfa Romeo, Buffetti Data, Commodore Italiana, CPT Italia, Creazioni Walt Disney, Data General, Digital, Doxa, Elsag, Ericsson, Facit Data Products, Ferrari, FIAT, Harden Italia, Hengstler Italia, Hewlett Packard, IBM, Intema, IRET Informática, Italcable, Lancia, Litton BEI, MEE, MSI Data Italia, Olivetti, Perkin-Elmer, Plessey Trading, Prime Italia, Rank Xerox, Rhône-Poulenc Italia, Sanco Ibex Italia, Sarin, Selca Elettronica, Selenia, Sperry, SIP, Telespazio.

Gestión de la impresora

En las formas vistas, la instrucción LPRINT permite la impresión de los datos de acuerdo con una discreta cantidad de formatos. Esta todavía puede resultar insuficiente para cubrir todas las necesidades que se crean en la escritura de programas de aplicación.
Por ejemplo, puede ser necesario cambiar el tamaño de los caracteres o la distancia entre una línea y otra. Estas funciones sólo pueden obtenerse enviando a la impresora algunos códigos particulares de comando constituidos por determinados símbolos ASCII, que tienen para la impresora el significado de comandos y no de caracteres a imprimir. Estos códigos y su significado son expuestos por el fabricante de la máquina, que los indica en el manual en forma de números hexadecimales o decimales. Antes de su envío a la impresora, deben ser transformados en caracteres ASCII por el programa de aplicación. La instrucción que da un valor numérico (decimal) y determina la representación ASCII es CHR$(N). El envío de un dato de código, por ejemplo el código 12 (decimal), consiste en enviar a la impresora su representación en caracteres a través de la instrucción

```
LPRINT CHR$(12)
```

A este código le corresponde una determinada función, que será ejecutada por la impresora en el momento de recibir la instrucción. Los valores numéricos de los códigos y las correspondientes funciones dependen del tipo de impresora. Los siguientes ejemplos se refieren a una impresora PR 1471 Olivetti; para otros modelos u otros constructores, algunos códigos son numéricamente diferentes, pero las modalidades que corresponden a su uso no varían.

Estructura y características de una impresora de agujas

Antes de entrar en el detalle de la programación de las funciones de las impresoras es necesario examinar la estructura de estos periféricos. Todo lo que se expondrá a continuación se refiere a las impresoras de impacto de agujas. Las im-

La impresora de agujas Olivetti modelo PR 1450.

Olivetti

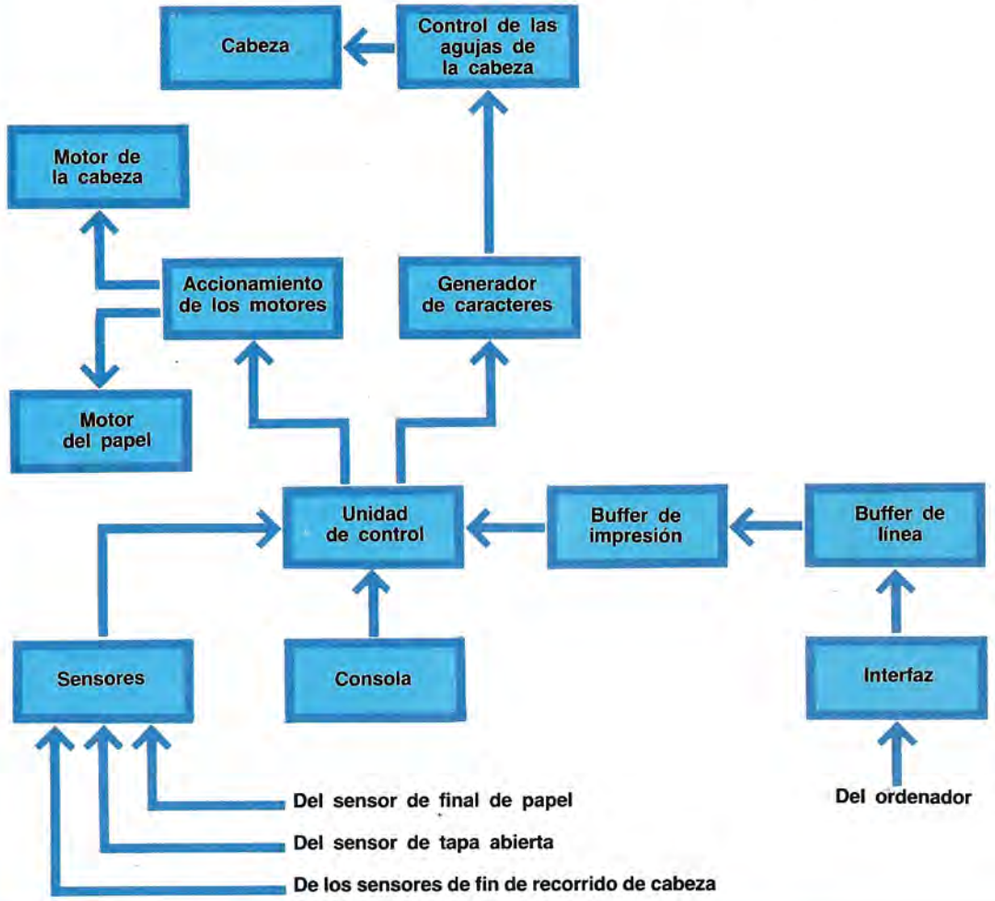
presoras de margarita no pueden programarse y los otros tipos, como por ejemplo las de chorro de tinta, todavía no se utilizan de modo extendido en los sistemas pequeños, sobre todo a causa de su costo. Abajo se ha indicado el sistema de bloques simplificado de una impresora. Los datos que llegan del ordenador se memorizan en el buffer de línea y se trasladan sucesivamente al buffer de impresión. De éste se toman en una segunda fase y, después de haber sido codificados en el generador de caracteres, se envían a controlar la cabeza de impresión.

Ahora describiremos las principales características de una impresora de agujas típica: la PR 1450 Olivetti. La impresión de la PR 1450 se obtiene mediante la acción de una cabeza de impresión de agujas de tipo balístico sobre la cinta entintada, posicionada entre la propia cabeza y el papel. El elemento de soporte del papel durante la impresión está constituido por un travesaño metálico. La cabeza de impresión va montada sobre un carro que se desplaza hacia delante y hacia atrás paralelamente al papel, accionado por un motor de corriente continua mediante una cinta dentada.

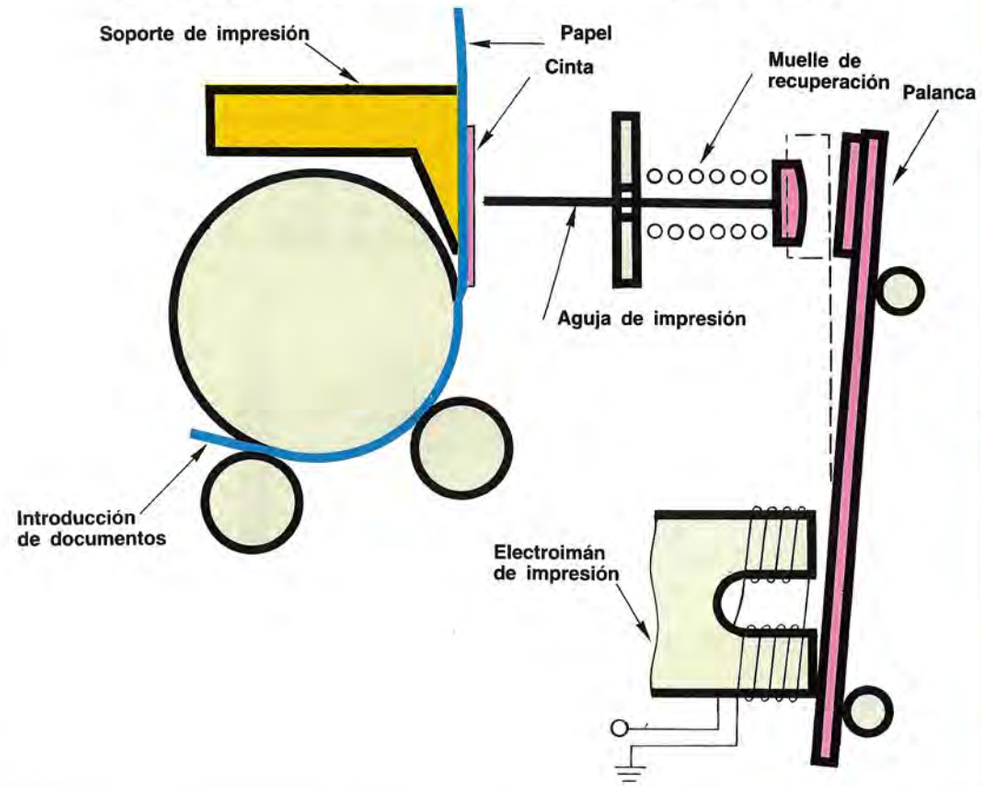
El carro portacabeza se desplaza sobre dos barras de acero de sección circular; una leva permite separarlo del travesaño de metal para la introducción del papel y acercarlo para la impresión. Las sucesivas acciones de la cabeza de impresión balística, en su movimiento horizontal, forman la línea de impresión.

En una cabeza de impresión de tipo balístico, las agujas no son comandadas directamente

ESQUEMA DE BLOQUES SIMPLIFICADO DE UNA IMPRESORA



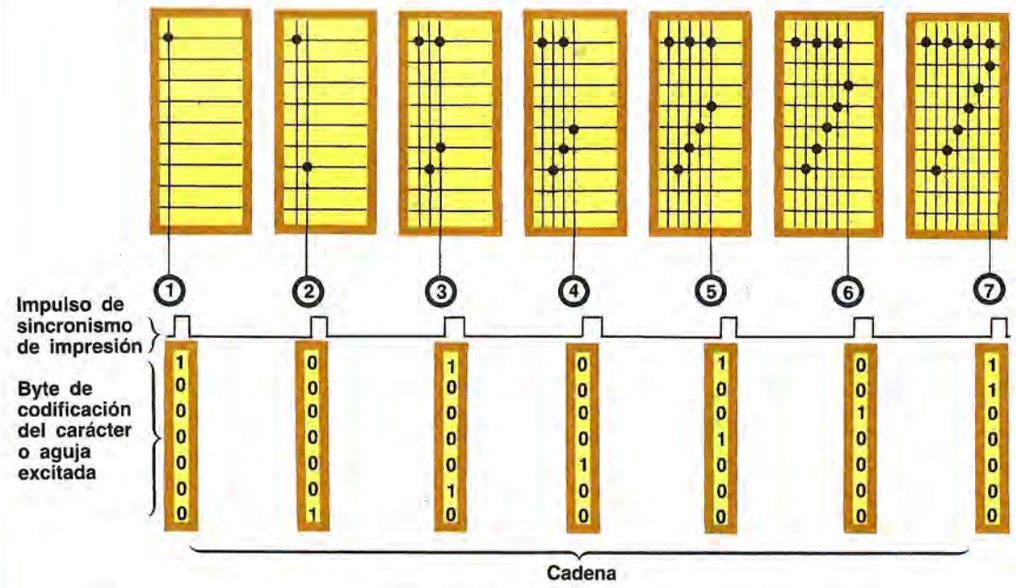
MECANISMO DE IMPRESION DE LA PR 1450 OLIVETTI



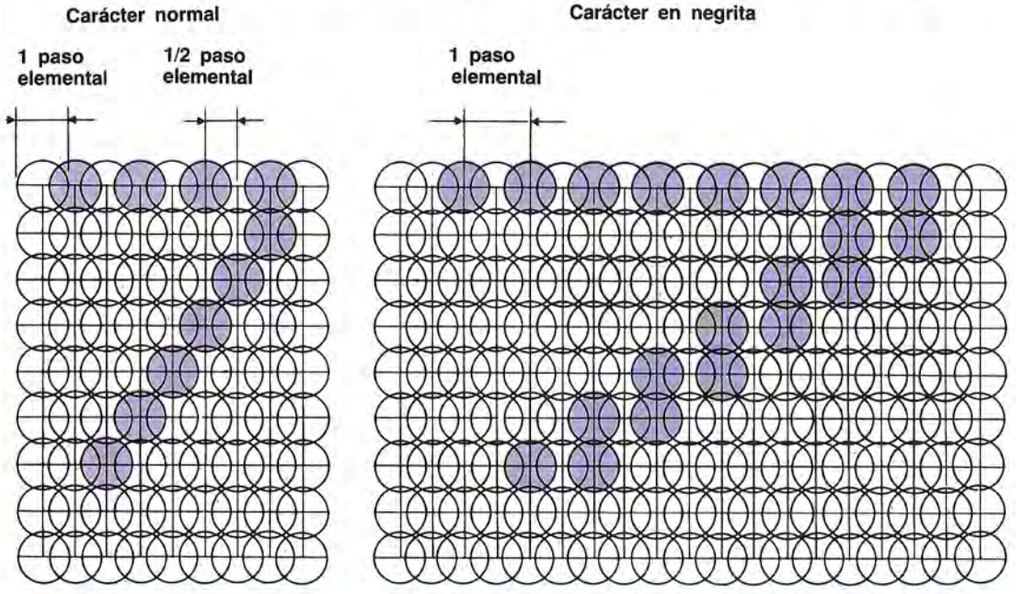
por los electroimanes de impresión (arrollamientos que generan un campo magnético al pasar corriente por ellos); estos últimos empujan hacia adelante una palanca que golpea contra la cabeza de la aguja impresora y se para cuando entra en contacto con las expansiones terminales del electroimán excitado. La aguja de impresión, que todavía posee energía suficiente, prosigue su movimiento hacia adelante golpeando la cinta entintada y el papel contra el soporte de impresión. La aguja vuelve hacia atrás y, con un muelle adecuado, vuelve a su posición inicial (figura de arriba). Los electroimanes son activados selectivamente, mientras la cabeza de impresión se mueve a lo largo del papel, para imprimir los puntos que forman cada carácter.

La impresión se obtiene mediante la activación selectiva de los 9 electroimanes que accionan las agujas de impresión. La secuencia exacta de activación de los electroimanes se controla mediante un adecuado microprograma y una tabla de traducción de códigos para la con-

IMPRESION DEL CARACTER 7 CON LA PR 1450 OLIVETTI



COMPOSICION DE LOS CARACTERES CON MATRIZ 9 × (4 + 3)



versión de los códigos ISO-ASCII en las correspondientes matrices de impresión (generador de caracteres). A la recepción de los impulsos, los electroimanes activados empujan las agujas contra la cinta entintada y el papel, tal como se ha descrito anteriormente; la presión de las agujas determina así puntos de impresión. Una adecuada combinación de puntos dispuestos sobre una matriz de filas y columnas, según una solución gráfica óptima, forma el carácter (ver

gráfico superior de la pág. 580).
Se utiliza una matriz de impresión 9 × (4 + 3), es decir, de 9 filas y 4 columnas con medio paso. La impresión de los caracteres está formada por 9 puntos verticales (max) para cada una de las 4 columnas, con posibilidad de imprimir en las posiciones intermedias entre columna y columna (3, medio paso). Así se tienen 63 (9 × 7) posiciones posibles, de las que sólo son imprimibles 36 (sobre la misma fila no pueden imprimirse dos puntos que sólo disten medio paso).
Para los caracteres en negrita, los puntos de impresión sólo se suceden a la distancia de un número entero de espacios de impresión, y cada columna en negrita se aprovecha de la columna del carácter normal (ver el gráfico de abajo de la pág. 580). Los juegos de caracteres previstos en la PR 1450 son:

— INTERNACIONAL: 96 caracteres
— USASCII: 96 caracteres
— ALEMANIA: 96 caracteres
— FRANCIA: 96 caracteres
— ITALIA: 96 caracteres
— GRAN BRETAÑA: 96 caracteres
— ESPAÑA: 96 caracteres
— PORTUGAL: 96 caracteres
— SUECIA-FINLANDIA: 96 caracteres
— DINAMARCA-NORUEGA: 96 caracteres
— SUIZA: 96 caracteres

Los juegos de caracteres pueden seleccionarse al nivel del operador mediante conmutadores, o bien por programa. Del juego gráfico del alfabeto ASCII no se imprime el espacio (Sp) y el carácter DEL (utilizado como comando de puesta a cero). Todos los caracteres no previstos en el juego de impresión son impresos con el signo gráfico "III", o con conmutador; la impresión de estos caracteres puede evitarse y el mensaje puede «compactarse».
La impresora PR 1450, como se ha dicho, hace posible imprimir caracteres en negrita, es decir, de anchura doble a la normal, como se indica en el ejemplo de impresión de abajo. La impresión en negrita se obtiene enviando el comando ESC 3 desde cualquier posición de la línea de impresión. La predisposición tiene efecto inmediato y es válida hasta el comando de anulación ESC 4. Un comando de variación de espaciado horizontal y la puesta a cero general vuelven a la escritura normal.
Después de la puesta en marcha de la PR 1450 no se tiene la impresión en negrita si no se envía por programa el comando específico.

Ejemplo de impresión

En la página 582 se indica el diagrama de flujo de un programa que tiene por objeto mostrar las posibilidades de una impresora de agujas típica para microordenador y ordenador personal.

EJEMPLO DE IMPRESION DE LA PR 1450

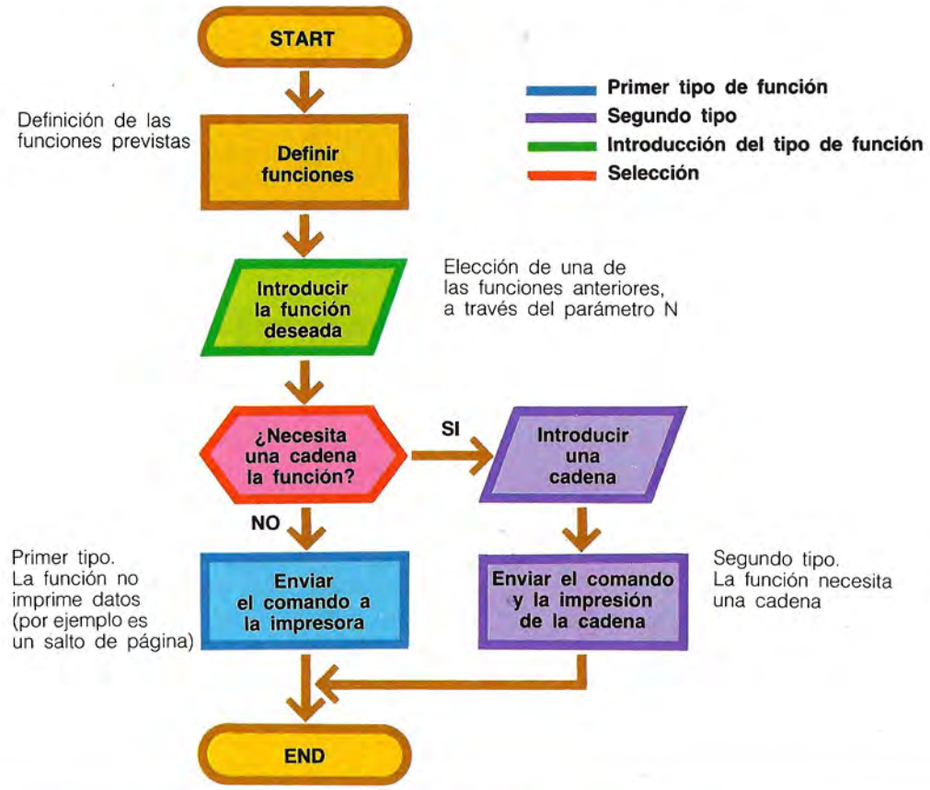
PRINTER CONTROL CODES

PRINT METHOD	Impact Dot Matrix
PRINTING SPEED	100 Char/Sec with fast carriage return
THROUGH PUT	50 lpm full line 86 lpm with 40 char
CHARACTERS COMPOSITION	9x7 (4+3) Dot Matrix
CHARACTERS PER LINE	1) 80 Char/line with 10 Char/inch 2) 132 Char/line with 16,6 Char/inch
LINE FEED	100 msec
COPIES	Up to 1 original and 2 copies

PRINTER CONTROL CODES

PRINT METHOD	Impact Dot Matrix
PRINTING SPEED	100 Char/Sec with fast carriage return
THROUGH PUT	50 lpm full line 86 lpm with 40 char
CHARACTERS COMPOSITION	9x7 (4+3) Dot Matrix
CHARACTERS PER LINE	1) 80 Char/line with 10 Char/inch 2) 132 Char/line with 16,6 Char/inch
LINE FEED	100 msec
COPIES	Up to 1 original and 2 copies

USO DE LOS CODIGOS DE COMANDO DE LA IMPRESORA



En el programa se hace referencia a la impresora PR 1471 Olivetti, que puede realizar impresiones en los siguientes formatos:

— doble anchura
— 132 caracteres por línea
— 159 caracteres por línea
— 220 caracteres por línea

El programa tiene el único objetivo de generar diversos ejemplos de impresión del mismo texto con diferentes formatos, seleccionados por el operador mediante la introducción por consola de los adecuados códigos. En el diagrama de flujo se ha omitido el bucle que permite obtener más fases de impresión sucesivas y el control de final de tarea, que se obtiene introduciendo el valor N = 0. El listado y las salidas del programa se indican en la pág. 583.
Los comandos previstos se dividen en dos grupos. El primer grupo corresponde solamente al avance del papel y no prevé la impresión de los datos; el segundo grupo corresponde a los caracteres de impresión, y, por tanto, necesita una cadena de impresión que debe ser introducida por el operador. Los códigos de comandos se obtienen definiendo algunas funciones (instrucciones DEF FN...) y se envían a su ejecución imprimiendo la función correspondiente.
Por ejemplo, el código correspondiente a un salto de línea es el valor numérico 10. La acción correspondiente se obtiene con la instrucción LPRINT CHR$(10), o bien definiendo la función DEF FNA$(N) = CHR$(N) y enviando a impresión esas funciones con N = 10.
En este caso, la serie completa de las instrucciones será la siguiente:

```
10 DEFINT A-Z
20 DEF FNA$(N) = CHR$(N)
30 N=10
40 LPRINT FNA$(N)
```


EJEMPLOS DE GESTION DE LA IMPRESORA

```
10 ' *** Ejemplos de gestión de la impresora ***
20 '   FILE : PRT
30 DEFINT A-Z
40 ' ***** FUNCIONES PREVISTAS *****
42 ' Avance de papel
44 '    CODIGO          FUNCION                 NOTAS
46 '      13        Carriage Return (CR)   Ordena el retorno al principio
47 '                                       de la cabeza
48 '      10          Line Feed (LF)       Salto de línea
50 '      12          Form Feed (FF)       Salto de página
52 '
54 ' Tipo de carácter
56 '    27+51          Carácter de doble anchura
58 '    27+52          Anula la doble anchura
60 '    27+60          Formato de 132 caracteres por línea
62 '    27+61          Formato de 159 caracteres por línea
64 '    27+62          Formato de 220 caracteres por línea
68 '
70 '
90 DEF FNA$(N)=CHR$(N)    ' PRIMER TIPO (avance papel) ,N=10,12,13
95 DEF FNB$(N)=CHR$(27)+CHR$(N) 'SEGUNDO TIPO (compuestos)
100 '
110 INPUT "Introducir el código de la función deseada" ;N
120 IF N=0 GOTO 300
130 IF N>62 THEN PRINT "*** ERROR ***":GOTO 110
140 IF N<=13 GOTO 260       ' PRIMER TIPO
150 IF N<51 THEN PRINT" *** ERROR ***":GOTO 110
160 IF N>52 AND N<60 THEN PRINT" *** ERROR ***":GOTO 110
170 IF N>62 THEN PRINT" *** ERROR ***":GOTO 110
180 ' *  SEGUNDO TIPO, DESPUES DE LOS CONTROLES DE LAS LINEAS 150, 160, 170
190 '    EL PROGRAMA PUEDE LLEGAR A ESTE PUNTO SOLO SI HAY
200 '    UNO DE LOS VALORES PREVISTOS PARA EL SEGUNDO TIPO DE FUNCION
210 INPUT"Cadena a imprimir";S$
220 LPRINT FNB$(N) ;S$
230 LPRINT "LINEA IMPRESA CON LA FUNCION: ";N
240 GOTO 110                  NUEVA SELECCION
250 ' * PRIMER TIPO
260 IF N<10 OR N=11 THEN PRINT" ** ERROR **":GOTO 110
270'LPRINT FNA$(N) ;"FUNCION : ";N;" NO TIENE TEXTO "
280 GOTO 110
290 '
300 ' **** FINAL ****
310 LPRINT FNB$(51);" ********* RUN END *********"
320 END
```

```
FUNCION : 13 NO TIENE TEXTO

FUNCION : 10 NO TIENE TEXTO
DOBLE ANCHURA
LINEA IMPRESA CON LA FUNCION: 51
ANULA DOBLE ANCHURA
LINEA IMPRESA CON LA FUNCION: 52
132 CARACTERES POR LINEA
LINEA IMPRESA CON LA FUNCION: 60
159 CARACTERES POR LINEA
LINEA IMPRESA CON LA FUNCION: 61
220 CARACTERES POR LINEA
LINEA IMPRESA CON LA FUNCION: 62
  ********** RUN END **********
```

Ordenador y psique

En este artículo, B. Meltzer, que se ocupa actualmente del Proyecto de Inteligencia Artificial en curso en el Politécnico de Milán, expone algunas interesantes consideraciones sobre el grado de similitud que existe entre los procesos cognoscitivos propios de la psicología humana y las posibilidades de los procesadores. Todos los programas destinados a desarrollar determinadas funciones se caracterizan hasta un cierto nivel de «cognoscitividad»; es decir, contienen una masa más o menos grande de conocimientos interconectados, según la complejidad de las funciones que deben desarrollar.

La potencia de elaboración de los calculadores de la última generación ha permitido la puesta a punto de programas muy complejos y, por tanto, cada vez más «cognoscitivos».

En general, los de nivel más alto son clasificados como de inteligencia artificial. Esto es lo que sucede cuando un programador introduce cada vez más conocimientos en su programa (el conocimiento no es sólo de hechos, sino también de procedimientos) y cuando transfiere una parte siempre mayor de su pensamiento al programa, en una forma de representación.

En otros términos, el material de la psicología cognoscitiva se introduce en los programas con una representación más o menos explícita.

Muchos estudiantes contemporáneos de psicología, filosofía y otras materias sostienen que los programas de los procesadores no pueden representar, o presentar, fenómenos mentales similares al humano, por toda una serie de razones. Vamos a exponer algunas de ellas.

Una de las objeciones más comunes es que la mente humana funciona sobre la base de sustratos biológicos constituidos por material neurónico, bien diferente de los transistores, microcircuitos, películas magnéticas, etc., y que por esto, el modo en que se realiza la función cognoscitiva debe ser del todo diferente al de los programas de los procesadores. Es probable que sólo pocas de las personas que se ocupan de los procesadores se dejen impresionar por este argumento, porque para ellas es sabido que un mismo programa que corre en dos máquinas diferentes, con dispositivos y arquitectura completamente diferentes, consigue sustancialmente de la misma forma el objetivo buscado.

Otra objeción muy corriente se hace a un nivel que está por encima del hardware. Esta se refiere al hecho de que los programas «cognoscitivos» de la inteligencia artificial desarrollan tareas intelectuales esencialmente a través de la búsqueda exhaustiva, y por tanto son profundamente diferentes del método humano que utiliza atajos, intuiciones, imaginación, «inspiración» y así sucesivamente.

Como todos saben, existen muchos programas de juegos que el procesador realiza contra jugadores humanos con diversos niveles de éxito. Por ejemplo, el programa de ajedrez 4,5 de Slate y Atkin ha ganado el campeonato open *de Minnesota, en Estados Unidos. Los programas de ajedrez de mayor éxito se basan sustancialmente en el examen detallado de posibles secuencias de jugadas a varios niveles de profundidad, pero no son los análisis «ciegos» y exhaustivos imaginados por algunos críticos sino el tipo de análisis de las posibilidades realizado por jugadores humanos, sean campeones o no; en este ámbito, a menudo los programas de ajedrez podrían jugar mejor que el jugador humano. Pero quizá es más interesante el caso del* backgammon *(en castellano, «chaquete» o «tablas reales»).*

En julio de 1979, en Montecarlo, el campeón mundial de backgammon, el italiano Luigi Villa, fue batido por 7 a 1 en un encuentro con el programa de Hans Berliner BKG 9.8 (Backgammon 9.8). Berliner, que es un investigador de inteligencia artificial en la Carnegie-Mellon University de Pittsburg, estableció el programa sobre la base de una brillante extensión del principio utilizado en uno de los primeros programas de juego, el de Samuel para las damas. En este enfoque se utiliza muy poca búsqueda sistemática; en su lugar, las jugadas se basan en una evaluación general, mediante un número finito de elementos característicos del juego. Después de analizar la secuencia del encuentro con Villa, Berliner llegó a estas conclusiones: «No hay duda de que el BKG 9.8 ha jugado bien. Villa ha realizado un juego técnicamente correcto en casi toda la partida, mientras que el programa no ha hecho el mejor juego en 8 situaciones sobre 73. Sólo uno de estos errores ha puesto en dificultades al programa. Sin embargo, el BKG 9.8 ha hecho saltar chispas cuando se ha tratado de utilizar la imaginación. Uno que no supiese quién era el Negro y el Blanco, habría podido pensar que era el hombre el que hacía el juego brillante y la máquina el normal».

El lenguaje se ha definido como la ventana de la mente, y se han escrito muchos programas utilizando diferentes enfoques, para comprender, parafrasear, traducir y resumir textos de lenguaje natural. Para dar una idea de lo que se ha hecho hasta ahora, tomaré como ejemplo un programa que selecciona noticias de periódicos, buscando cosas que interesen y dando un breve resumen de los hechos en un idioma cualquiera como inglés, ruso o español.

Su nombre es FRUMP (Fast Reading and Understanding Memory Program) *que fue desarrollado por el grupo para la comprensión del lenguaje natural de Roger Schank en la Universidad de Yale. He aquí un ejemplo de la capacidad del FRUMP, aplicado a una noticia tomada de un periódico norteamericano:*

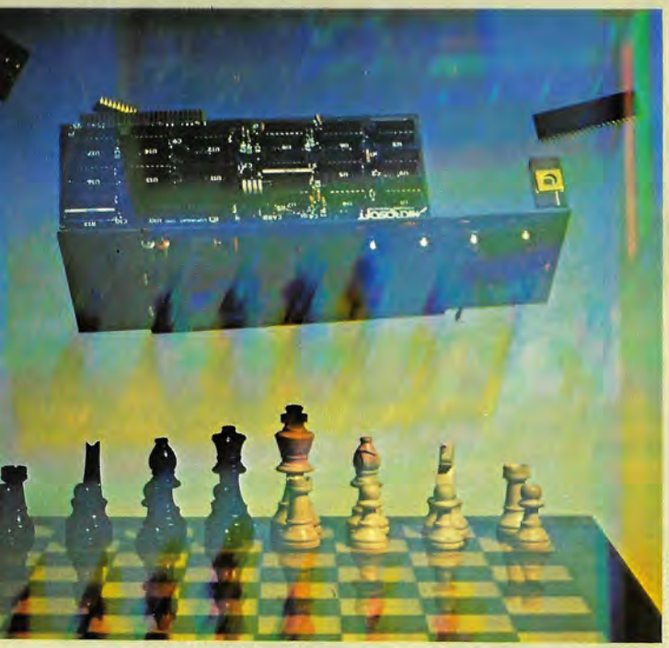
«Un violento terremoto ha sacudido Italia septentrional ayer noche, haciendo temblar barrios enteros en una ciudad al NE de Venecia, cerca de la frontera yugoslava, matando al menos a 95 personas e hiriendo al menos a 1.000, según lo referido por el Ministerio de Asuntos Interiores italiano. Un portavoz gubernamental ha dicho que, sólo en la ciudad de Udine, se teme que haya por lo menos 200 muertos debajo de los escombros. La ciudad, en la línea ferroviaria principal entre Roma y Viena, tiene una población de cerca de 90.000 habitantes. El portavoz de los carabineros, la policía nacional paramilitar, ha dicho que se tienen noticias de graves daños en media docena de ciudades al pie de los Alpes, con familias enteras sepultadas por los escombros de los edificios. Las comunicaciones con otros centros de la zona todavía están interrumpidas. El terremoto ha sido de 6.3 grados de la escala Richter, que mide los movimientos telúricos. En zonas habitadas, una sacudida de 4° grado puede provocar daños moderados, una de 6° puede ser grave, una de 7° indica un terremoto con efectos desastrosos».

He aquí el resumen del FRUMP: «95 personas muertas y 1.000 heridas en un terremoto que ha sacudido Italia. La intensidad registrada es de 6.3 grados de la escala Richter».

Mientras los programas del grupo de Schank se centran sobre conceptos de la vida diaria, el que sigue se ocupa en cambio de conceptos matemáticos y, además, genera por sí mismo otros conceptos y hace de ellos conjeturas más o menos interesantes. El programa ha sido escrito por el joven norteamericano Douglas Lenat y descrito en su tesis presentada en 1976 en la Stanford University. Lenat se ha concentrado en la investigación de los modelos relativos al descubrimiento de conceptos y proposiciones matemáticas interesantes. Efectivamente, su programa tenía bien pocas capacidades deductivas, en el sentido estricto de la demostración de teoremas. La actividad del programa consistía en intentar seguir una secuencia de cálculos según su «grado de interés», abandonando un cálculo apenas los resortes de elaboración asignados se hacían insuficientes. Si bien la mayor parte de los descubrimientos hechos correspondían a los números naturales, el programa AM no contenía estos conocimientos de base (probablemente, el nombre AM viene de «Matemático-Automático»). El programa los descubrió por su cuenta, aunque dejó de descubrir conceptos como los números reales y las fracciones.

Su base de conocimientos de partida consistía en cerca de 100 conceptos y cerca de 250 reglas eurísticas cuya acción se emprendía según las circunstancias. Los conceptos incluían objetos como conjunto, listados (el programa estaba escrito en lenguaje LISP, orientado a los listados), tablas de verdad, relaciones como pertenencia e igualdad, operaciones como inversión, intersección y composición. Lenat sostenía, basándose en referencias al trabajo de los psicólogos Piaget y Copeland, que su colección distribuida de conceptos intentaba ser la de un niño de 4 años, pero no sabía hasta qué punto era verdad. El programa encontró bien pronto, con una exploración empírica, que ningún número tenía cero divisores y que sólo uno tenía uno solo, pero encontró también ejemplos de números con sólo dos divisores (es decir, 1 y el propio número). Así, a su colección de conceptos adjuntó el del número que sólo tiene dos divisores (concepto de número primo).

Incidentalmente, es muy interesante el hecho de que el AM hubiese generado otro concepto de teoría de los números, que era completamente nuevo para Lenat (y para mí), pero que había sido estudiado en los primeros años del siglo por el joven genio indio Ramanujan. Es el concepto de los números «compuestos máximamente», que son una especie de opuesto del número primo. De hecho son aquellos números que tienen un número mayor de divisores que sus inferiores. Así, los números primos compuestos máximamente son 1, porque sólo tienen un divisor, 2 porque tienen dos, 4 porque es



Mallio Falcioni

Los programas para el juego de ajedrez son la punta de diamante de la inteligencia artificial.

el primero que tiene tres, 6 porque es el primero que tiene cuatro, y así sucesivamente. El AM no sólo descubrió los números naturales y los primos (en más de un modo), sino que también elaboró el teorema de factorización única, que dice que cada número natural es descomponible en un producto de números primos de una sóla manera; además, descubrió el enunciado de Goldbach (hasta ahora no demostrado), según el cual cada número par es la suma de dos números primos.

A éxitos notables similares corresponden fallos también interesantes. Ya he indicado el hecho de que el programa no supo descubrir las fracciones. Tampoco hizo grandes progresos en la teoría de conjuntos ni supo descubrir las relaciones más sencillas como las de De Morgan. En cambio siguió recorridos inútiles, ocupándose por ejemplo del concepto de números que pueden ser representados como la suma de dos primos únicamente de un solo modo. En su tesis, Lenat estimaba que el AM había descubierto cerca de 25 conceptos buenos («vencedores»), 100 aceptables y 60 «perdedores».

He presentado tres ejemplos de programas relativamente recientes para demostrar que no era infundada mi afirmación de que los programas en el nivel superior del espectro que he presentado anteriormente, muestran procesos que no son «mecánicos» en el sentido de menoscabo del término, sino que imitan aquellas características de creatividad, intuición, inteligencia, etc. que pertenecen a la psique humana. Además de estos se pueden encontrar ejemplos también convincentes en otros sectores, como por ejemplo el diagnóstico médico, la percepción visible, la prueba de propiedad de los programas, pruebas de coeficientes de inteligencia, aprendizaje, etc. Así, la ciencia de los procesadores nos ha familiarizado con los lenguajes que tienen como objeto no comunicar, sino representar y elaborar, por ejemplo lenguajes ensambladores y lenguajes máquina. Recientemente, un estimulante análisis filosófico de Aaron Sloma sugería que esta es efectivamente la principal función del lenguaje, siendo la función de comunicación sólo un «producto derivado». Puede decirse que los animales y nuestros lejanos antepasados humanos ya habían operado dentro de ellos lenguajes de representación y elaboración, los cuales, en el transcurso de la evolución, se abrieron, por así decirlo, a través de la invención de los signos externos, en forma de sonidos y de imágenes.

Para volver a la psicología humana, incluso si se acepta la existencia de un lenguaje interno de elaboración que se limita a acontecimientos del sistema nervioso, ¿no sería más seguro y más de acuerdo con lo que conocemos, trazar una distinción neta entre aquellos lenguajes internos y lo que, por el momento, queremos indicar como los lenguajes externos que hablamos, ya sea el inglés o el español? Yo creo que no, ya sea por razones de evidencia, ya sea por el potencial de esclarecimiento implícito en una visión unitaria del fenómeno.

Me refiero a la sugestiva evidencia que se deriva del aprendizaje humano, especialmente cuando se complica con el uso de simbolismos más bien formales. Tomemos como ejemplo de la estrecha asociación y continuidad entre el lenguaje interno y el externo, los cambios que se producen en nuestro aprendizaje de la aritmética elemental durante la infancia. Cuando se nos enseña por primera vez la suma, la seguimos en términos de instrucción del maestro, utilizando conscientemente palabras y frases del lenguaje natural como «llevo» y «2 por 3 son 6», pero cuando ya somos hábiles y realizamos las sumas del modo usual, automático, «sin pensar», prácticamente no queda en nuestra consciencia ninguna de aquellas instrucciones y expresiones. Pero nadie puede dudar, aunque no seamos conscientes, que se ha realizado una elaboración simbólica: la actividad ha pasado del lenguaje externo al interno.

Obsérvese cuán fácil y «natural» es la transición entre los dos lenguajes: si, cuando ya somos hábiles, nos encontramos frente a una suma particularmente larga o difícil, es posible que hagamos la transición a la inversa, volviendo a la parte externa del lenguaje, quizá diciéndonos a nosotros mismos: «Debo recordar que llevo dos cuando haya terminado de sumar esta columna». Por tanto, la hipótesis que quiero hacer es la siguiente: «Todos los procesos mentales en cualquier sistema biológico también son transformaciones que tienen lugar en el interior del lenguaje propio del sistema, y este lenguaje puede ser activado en cada nivel».

La idea de nivel es bien conocida y clara en un sistema de elaboración. Un programa escrito en el lenguaje fuente al más alto nivel, para ser ejecutado, primero debe ser convertido en el lenguaje ensamblador de la máquina, y el resultado convertido a continuación en código máquina. Un esquema jerárquico estratificado de este tipo es el modo normal en que se ejecutan los programas. Sin embargo, son posibles amplias variaciones; por ejemplo, algunos sistemas ofrecen la posibilidad (generalmente por razones de eficiencia) de interponer trozos de programas en código máquina en el programa escrito en lenguaje al más alto nivel. Cuando se hacen cosas de este tipo, o si sólo se piensa en la posibilidad de hacerlas, entonces la noción de estratos del lenguaje diferentes en la máquina empieza a perder sentido, y es más útil pensar en el programa como escrito en un lenguaje único (de los que el fuente, el ensamblador, y el código máquina, así como las transformaciones entre ellos, son los constituyentes).

Esto puede ser el análogo del lenguaje unitario de los organismos biológicos que he postulado. Y cuando hablaba de «niveles» a los que puede ser activado, me refería puramente a las posiciones en este complejo sistema lingüístico. El hecho de que, por ejemplo, podamos imaginar objetos sin verlos, y no solamente oyendo su nombre, necesita una explicación. Este hecho resulta explicable bajo la hipótesis de que el funcionamiento de nuestras diferentes modalidades cognoscitivas y sensoriales tengan un lenguaje común, mediante el cual pueden alcanzarse las mismas representaciones a través de diferentes procesos de elaboración lingüística. Esto podría ser un ejemplo de activación del lenguaje unitario a diferentes «niveles», de los que he hablado anteriormente. La poesía es particularmente interesante para tener en cuenta, ya que mientras utiliza exclusivamente el lenguaje natural, parece que intenta evocar «niveles más profundos» de la parte interna del lenguaje. La poesía activa resonancias que el uso normal del lenguaje no parece poder suscitar.

Tampoco la música utiliza el lenguaje discursivo y, por tanto, al no poder actuar a través de dichos canales, parece muy adecuada para suscitar una evocación de nuestro lenguaje interno más directa, o sea más potente.

En lo que se refiere a las artes figurativas, según esta hipótesis, probablemente nuestros antiguos antepasados transferían en las paredes de sus cavernas partes seleccionadas de su representación interna del mundo; así lo han hecho desde entonces todos los artistas. Es significativo que en el trabajo de David Marr sobre modelos computacionales de la percepción visible de los objetos, una de las etapas de elaboración es la «stick figures», característica de uno de los estilos de Picasso.

Puesto que hemos visto que los métodos de la inteligencia artificial se adaptan bien a la representación de objetivos, expectativas e interpretaciones, no debería ser demasiado difícil determinar eficaces modelos de las emociones.

El clásico trabajo pionero de Freud sobre los procesos inconscientes que intervienen en los sueños, en las bromas y en las paráfrasis, para un profano, como el que escribe, parece haber tenido bastante menos influencia científica de lo que podría haberse esperado.

Las transformaciones que he indicado, que tienen lugar en los sueños y en las bromas, podrían colocarse, con una formulación igualmente rigurosa, en el ámbito de las transformaciones informáticas ya indicadas, como son la interpretación, la compilación y otras también empleadas en la modelística computacional. Entonces no debería ser demasiado difícil escribir, por ejemplo, un programa que con un determinado cuerpo de conocimientos ¡pudiese generar chistes óptimos!

Extraído de «Informática y psicología», de B. Meltzer, en QUADERNI DI INFORMATICA, año VIII, n.° 2, 1981, Honeywell Information Systems Italia.

Generación de diagramas de barras

En la preparación de cualquier trabajo, científico o de gestión, es muy útil disponer de una representación gráfica de los datos. Normalmente, los valores que constituyen el objeto de un informe o de una relación se presentan en forma tabular para poderlos utilizar en el análisis de detalle; para una rápida visión de conjunto, o para observar la marcha general de un fenómeno, conviene utilizar una representación gráfica. El tema se profundizará a continuación, pero ahora expondremos los métodos para preparar histogramas con la impresora. Un **histograma** es la representación gráfica de una tabla. Así, al realizar una investigación estadística sobre la facturación de un número de vendedores, puede formarse la tabla:

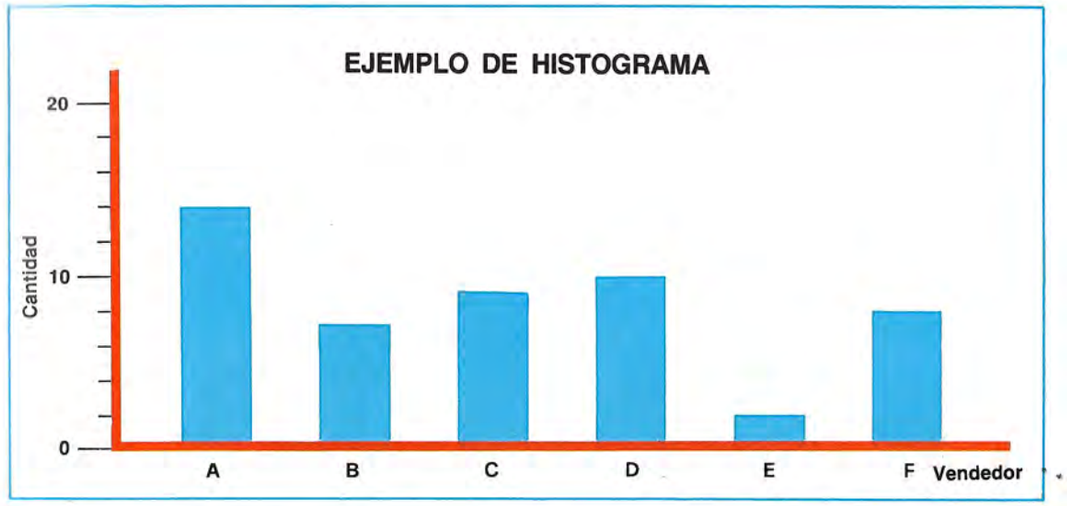
Vendedor	Cantidad vendida
A	12
B	7
C	9
D	10
E	2
F	8

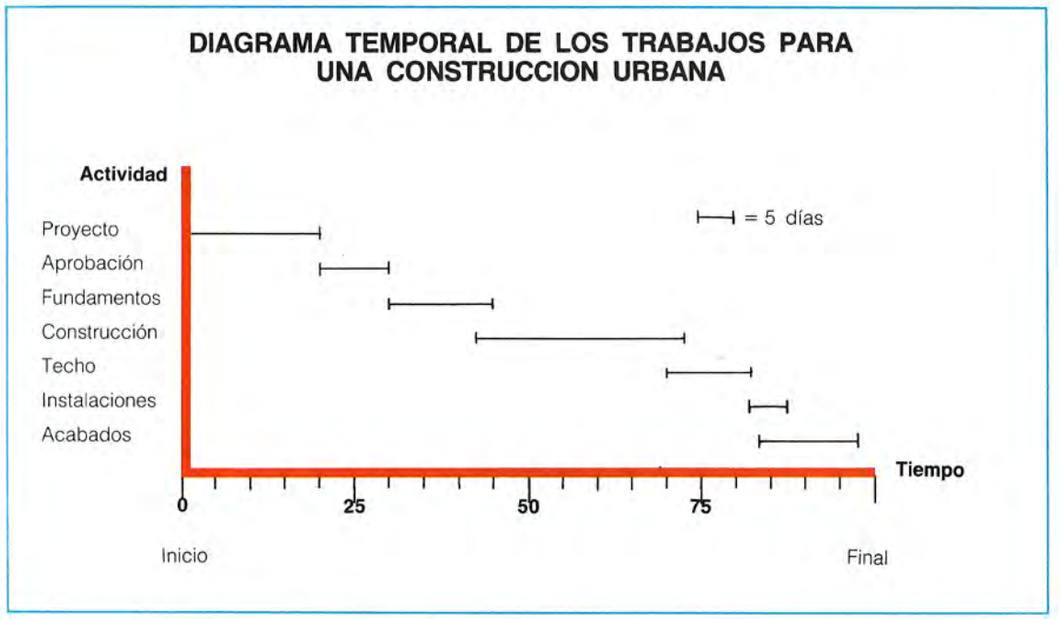
Cada vendedor tiene su propia facturación; la marcha global se ve transformando la tabla en un gráfico como el de la pág. 589, donde cada letra representa un vendedor. La cantidad de estos últimos es discreta; entre un punto y el otro (por ejemplo, entre A y B) no existen otros valores. En estos casos, la representación gráfica da lugar a un histograma del todo diferente a la representación gráfica de una curva.

La diferencia entre los dos tipos de representación gráfica reside en el «campo de definición». Para una curva existen, entre dos puntos, todos los puntos intermedios (excepto casos particulares); para los histogramas, la cantidad de valores posibles es limitada. Los **diagramas de barras** son análogos a los histogramas. En la fase inicial de cualquier proyecto industrial debe efectuarse el análisis de los tiempos de actuación. El proyecto se realiza a través de una serie de sucesos, dependientes entre sí, o con su propia duración. El tiempo total de actuación depende de los tiempos de cada suceso. En la fase de estudio del proyecto, el análisis de tiempos se realiza indicando en un gráfico las duraciones sencillas. Supongamos que debe analizarse el proceso temporal de la construcción de una casa. Las principales fases del trabajo son así intercambiables:

Impresora Buffetti con la tapa abierta.

V. Pirozzi/Archivo Curcio





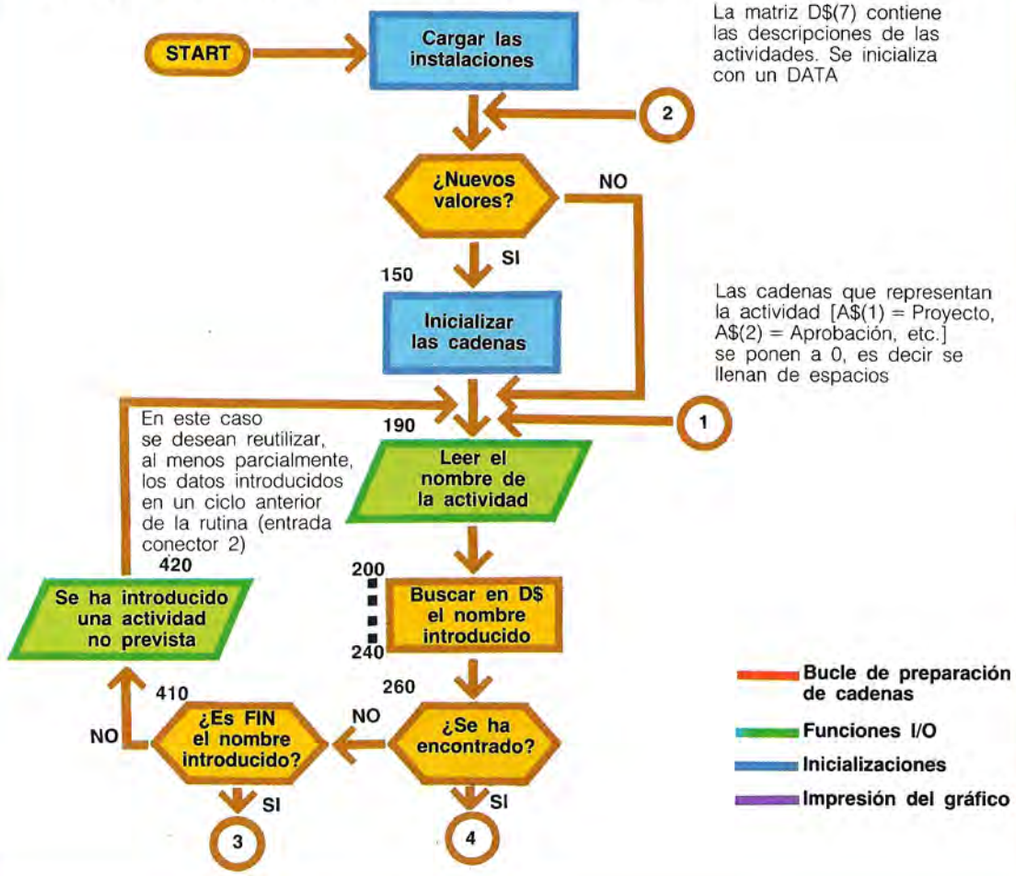
Actividad	Duración
Proyecto	20 días
Aprobación	10 días
Fundamentos	15 días
Construcción	30 días
Techo	12 días
Instalaciones	8 días
Acabados	16 días

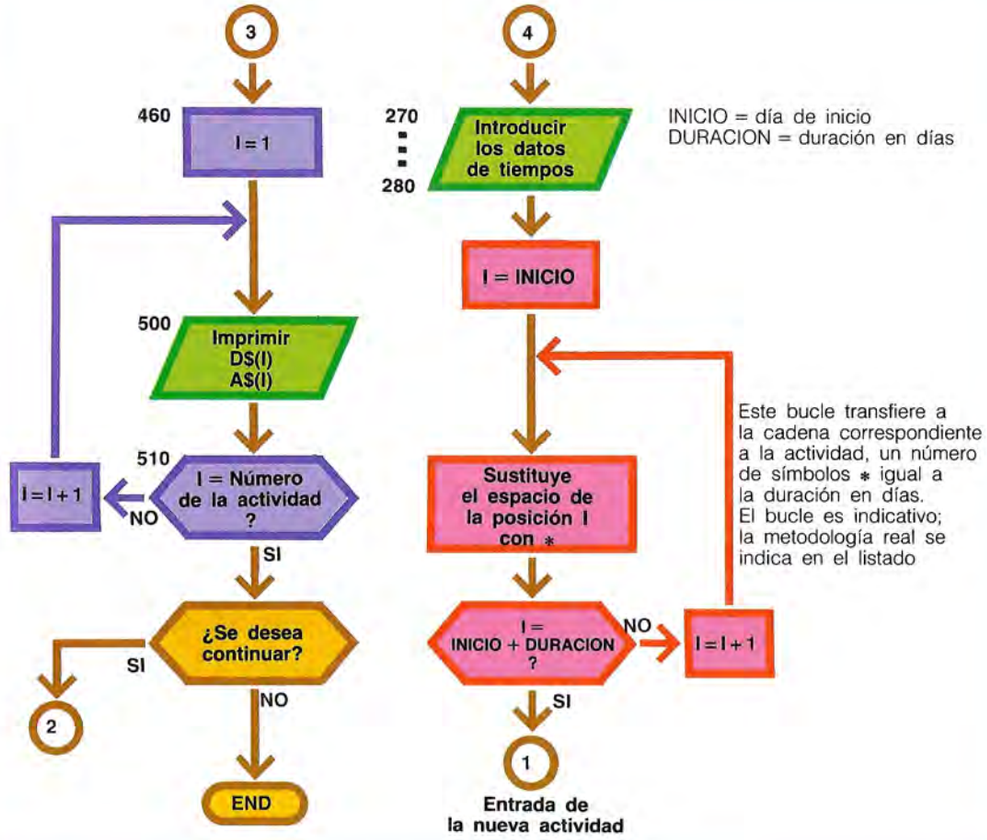
Llevando a un gráfico las diversas denominaciones y sus duraciones, se tiene el diagrama de barras del proyecto. Arriba aparece el diagrama que representa los valores tabulares. Como se ve, algunas actividades sólo pueden iniciarse si se han terminado las anteriores, y otras pueden iniciarse solapadas; por tanto, la duración total del proyecto no es la suma de las duraciones sencillas, sino el valor final que se obtiene leyendo el diagrama completo.

En la realidad industrial, el análisis de este tipo de diagramas es muy importante, por lo que para un ordenador de una cierta dimensión existen complejos programas para la gestión y la planificación de las actividades. Con estos programas es posible estudiar cómo repercute sobre todo el proyecto un eventual retraso sobre una de las

actividades previstas, o cuál debe ser en el tiempo la mano de obra disponible. Simulando las causas de retraso pueden estudiarse los efectos. Los diagramas de barras son muy sencillos de realizar con el ordenador. El método consiste en preparar, para cada una de las actividades previstas, una cadena de longitud de caracteres al menos igual a la duración del proyecto (en una adecuada escala). Las cadenas inicialmente se llenan con espacios; en el transcurso del programa se inserta un símbolo cualquiera para cada unidad de tiempo empleada en la correspondiente actividad.
Por ejemplo en el gráfico de la página 589 se prevén 7 actividades: por tanto, son necesarias 7 cadenas. La longitud inicial puede ser igual para todas e igual a 130 (casi igual al número máximo de caracteres que pueden imprimirse en una línea). Una vez preparadas las cadenas,

DIAGRAMA DEL PROGRAMA DE IMPRESION DE DIAGRAMAS DE BARRAS





debe llevarse a cada una de ellas un número de símbolos (por ejemplo asteriscos) proporcional a la duración que se prevé para cada fase. Los símbolos deberán ocupar algunos de los 130 caracteres disponibles; así es como se posiciona la actividad en el tiempo. Refiriéndose al mismo gráfico, la primera cadena (Proyecto) deberá contener 20 asteriscos (duración 20 días) posicionados a partir del principio, mientras que la última (Acabados) deberá contener 18 asteriscos a partir de, por ejemplo, la posición 82 (se supone que la fase de acabado pueda iniciarse a partir del día 82). Al lado y arriba se indica el diagrama de flujo del programa que genera un diagrama de barras similar al indicado en la pág. 589; el correspondiente listado y la fase de output aparecen en las págs. 592-593.
Al final del capítulo dedicado a la gestión de la unidad de vídeo pueden verse otras formas de representación gráfica.

PROGRAMA DE IMPRESION DE DIAGRAMAS DE BARRAS

```
10 ' **** PROGRAMA DE IMPRESION DE DIAGRAMAS DE BARRAS
20 '     FILE : BARRAS
25 OPTION BASE 1
30 DIM D$(7) ,A$(7)
35 DEFINT I-N
40 FOR I=1 TO 7:READ D$(I) :NEXT I
50 DATA" PROYECTO      "
60 DATA" REVISION      "
70 DATA" CALCULOS      "
80 DATA" EXCAVACION    "
90 DATA" FUNDAMENTOS   "
100 DATA" ESTRUCTURA   "
110 DATA" PAVIMENTADO  "
115 INPUT "INTRODUCIR COMENTARIO ",COM$
120 PRINT " ¿Van a variar todos los valores ?"
130 INPUT " SI/NO ";RESP$
140 IF RESP$<>"SI" GOTO 190
150 ' *** INICIALIZA LAS CADENAS
160 FOR I=1 TO 7
170 A$=" "          ' CADENA NULA
180 NEXT I
190 INPUT" Introducir el nombre de la actividad ";C$
200 ' *** BUSQUEDA DE ACTIVIDAD ENTRE LAS PREVISTAS (LINEAS 50-110)
201 ' LAS DESCRIPCIONES SON DE 14 CARACTERES DE LARGO MIENTRAS EN C$ (DATO
202 ' ENTRADO) NO HAY LOS ESPACIOS. ANTES DE PROCEDER AL COTEJADO
203 ' DEBE PREPARAR EL MISMO FORMATO
204 C$=" "+C$      ' Los datos empiezan con un solo espacio (ver líneas DATA)
205 N=14-LEN(C$)   'Número de espacios que faltan hasta 14
206 IF N<0 GOTO 420    ' ERROR, LA ACTIVIDAD TIENE DEMASIADOS CARACTERES
207 IF N=0 GOTO 210    ' LA ACTIVIDAD TIENE 14 CARACTERES
208 C$=C$+SPACE$(N)
210 K=0          ' K = flag si es de valor cero, la actividad no está prevista
220 FOR I=1 TO 7     ' Bucle de selección de una de las 7 actividades
230 IF C$=D$ (I) THEN K=I    '¿Existe en D$ la actividad introducida (C$) ?
240 NEXT I
250 '
260 IF K=0 GOTO 400          ' NO ** ERROR ** NUEVA INTRODUCCION
270 INPUT " DIA DE INICIO (Progresivo desde inicio trabajos) "; INICIO
280 INPUT " DURACION ACTIVIDAD En días                      "; DURACION
290 S$="*"           ' * = Símbolo usado para los programas (puede
295                        sustituirse por cualquier otro)
300 N=DURACION
310 T$=STRING$ (N, S$)        ' Crea una cadena (T$) de N caracteres
320                           ' igual al símbolo elegido
330                           '
340 ' ** POSICIONA T$ EN LA CADENA DE IMPRESION A$ (I)
350 I=K                       ' Actividad seleccionada (líneas 210-240)
360 M=INICIO-1                ' Posición de inicio en el tiempo
370 '
380 A$ (I)=SPACE$ (M)+T$
390 GOTO 190
400 '
410 'IF C$=" FIN                " GOTO 460
420 PRINT " **  ERROR, LA ACTIVIDAD INTRODUCIDA NO ESTA PREVISTA **"
430 PRINT " INTRODUCIR UN CARACTER PARA CONTINUAR "
440 S$=INPUT$ (1)
450 GOTO 190
460 '
470 ' *** INSTRUCCIONES DE IMPRESION ***
480 LPRINT COM$:LPRINT:LPRINT
490 'FOR I=I TO 7
500 'LPRINT D$(I), A$(I)
510 NEXT I
511 ' ** Escribe la escala de tiempos
512 LPRINT SPACE$(14);
520 '
530 '
540 INPUT " CONTINUA (SI/NO) ";RESP$
550 IF RESP$="SI" GOTO 115
570 END
```



```
DIAGRAMA DE BARRAS DE PRUEBA N.: 1

PROYECTO                **********
REVISION                     *****
CALCULOS                    ********
EXCAVACION                       *****
FUNDAMENTOS                          **********
ESTRUCTURA                                *****
PAVIMENTADO                                    **********


                DIAGRAMA DE BARRAS DE PRUEBA N.:2

PROYECTO                **********
REVISION                     *****
CALCULOS                    ********
EXCAVACION                       *****
FUNDAMENTOS                          **********
ESTRUCTURA                                **********
PAVIMENTADO                                    *****
```

TEST 17

1 / Una impresora tiene las siguientes características:
- cabeza de agujas
- bidireccional
- 132 columnas
- 120 cps
- buffer de 4k

¿Qué significan los datos citados?

2 / Decir qué caracteres serán escritos enviando a la impresora los siguientes códigos:

CHR$(33) CHR$(110) CHR$(27) CHR$(10)

3 / ¿Cuáles son las salidas del siguiente programa?

```
10 A$ = "Prueba":B = 1270:C = 10
20 PRINT A$,C
30 PRINT USING " # # # # . # "; A$;B
```

4 / La matriz numérica A(5) se ha impreso con las siguientes instrucciones:

```
10 FOR I = 1 TO 5 : LPRINT A(I) : NEXT I
20 FOR I = 1 TO 5 : LPRINT A(I) ; : NEXT I
30 FOR I = 1 TO 5 : LPRINT A(I) , TAB(10) ; NEXT I
```

¿Cuáles son las respectivas salidas?

Las soluciones, en la pág. 599.

Gestión de la consola de vídeo

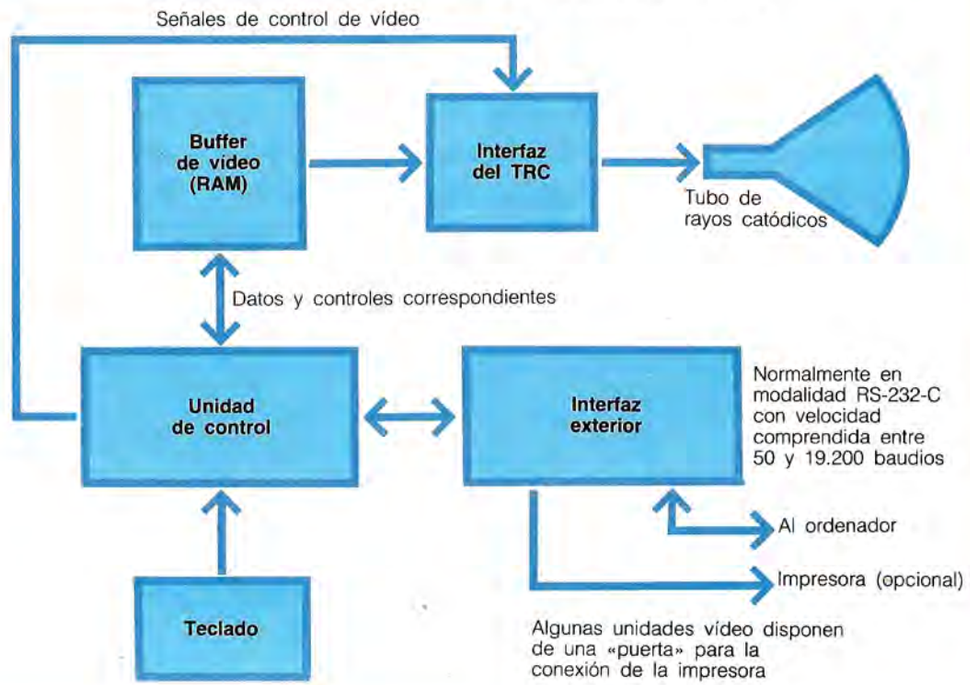
La configuración estándar de los microordenadores y ordenadores personales prevé siempre la presencia de una consola de vídeo como periférico de entrada/salida. La comodidad y racionalidad de empleo de este tipo de aparatos ha sido reconocida, y también los minicalculadores y grandes ordenadores están acoplados con el usuario a través de terminales de vídeo.
La eficaz gestión de la unidad de vídeo amplía la capacidad de presentación de un sistema de proceso, confiriéndole características «conversacionales» que son muy útiles para simplificar el diálogo con el elemento humano. Sin embargo, la gestión de la unidad de vídeo puede ser conducida por el programador, que deberá aprovechar al máximo las posibilidades de control ofrecidas por el hardware particular.
Las posibilidades de gestión completa de la unidad de vídeo son insustituibles en los trabajos de programación más complejos, mientras que es necesario disponer de rutinas adecuadas predispuestas para el control de las fases de presentación y de petición de los datos.

Estructura y características de una consola de vídeo

El esquema funcional de una unidad de vídeo típica se indica abajo. Los bloques principales son los que se detallan a continuación, seguidos de una breve descripción de las funciones que realiza cada uno de ellos.

- **teclado (keyboard)**: es el órgano de entrada. En él, mediante la presión de una tecla, se genera el código ASCII correspondiente que se envía a la unidad de control
- **unidad de control**: rige las demás unidades. Interpreta y ejecuta los comandos y las comunicaciones con el ordenador. Tiene misiones similares a las de una CPU
- **interfaz exterior**: controla las comunicaciones con el ordenador y con la eventual impresora. Algunas unidades de vídeo disponen de **puerta** para la conexión de una impresora directamente al terminal; las más sofisticadas pueden volver a enviar a la impresora todo el contenido de la pantalla de vídeo sin hacer intervenir el ordenador. Esta técnica de impresión se llama **hard copy**

ESQUEMA FUNCIONAL DE LA CONSOLA VIDEO



V. Pirozzi/Archivo Curcio

Detalle de una impresora Buffetti.

- **buffer vídeo:** es una memoria (RAM) que contiene todas las informaciones a visualizar. Su capacidad debe ser igual al número de caracteres que pueden contenerse en una pantalla de vídeo. Así, para llenar una pantalla de 80 columnas × 24 líneas se necesitan 1920 caracteres (80 × 24); por tanto, la RAM de un vídeo de estas dimensiones debe disponer de 1920 posiciones.

Para indicar la capacidad de esta memoria se ha utilizado el término **posiciones**, en lugar del byte usual, ya que para esta particular aplicación, una posición de memoria no corresponde a un byte. Por cada carácter a enviar al vídeo deben memorizarse bien el código ASCII (8 bits), bien los **atributos** del carácter, es decir, los códigos suplementarios que indican la modalidad según la cual debe escribirse el carácter (normal, invertido, etc.). Para esta función normalmente se necesitan otros 4 bits.
La longitud de una posición de memoria depende del tipo de vídeo y del tipo de ordenador.
Los valores indicados son los más utilizados, pero pueden existir diferencias incluso notables entre los diversos tipos de terminales existentes (todo lo expuesto no puede aplicarse a los terminales gráficos, que serán descritos aparte).

- **interfaz TRC**: genera las señales vídeo necesarias para visualizar los caracteres y las envía al mismo tiempo al monitor.

La unidad de vídeo tiene diversas modalidades de funcionamiento, que dependen de las particulares aplicaciones. Las principales son las que se detallan a continuación.

CONVERSACIONAL. Cada comando es enviado al host computer (ordenador huésped) normalmente con el protocolo RS-232 (serie asíncrono). En esta modalidad hay dos estados:

half-duplex: El terminal realiza todos los comandos, ya sean los generados en teclado (y enviados a la línea) ya sean los procedentes de la línea del ordenador huésped.

full duplex: Los comandos generados en teclado no se ejecutan directamente, sino que sólo se ponen en línea. Y viceversa, los comandos procedentes del ordenador (que recorren la línea en el otro sentido) se ejecutan. El terminal está bajo el total control del ordenador.

PAGE. En esta modalidad, utilizada sólo en aplicaciones muy particulares, pueden escribirse todos los caracteres previstos en una pantalla de vídeo e iniciar la transmisión hacia el ordenador huésped sólo con la página terminada. Las funciones de entrada de datos previstas en Basic (y en general en todos los lenguajes de alto nivel) no prevén el uso de esta modalidad operativa. El uso del estado PAGE sólo es posible escribiendo rutinas adecuadas (generalmente en lenguaje Assembler), que permiten regir este particular modo de transmisión.

SCROLL. Es similar al anterior, pero con la diferencia de que si se continúan introduciendo caracteres con la pantalla llena, se produce el desplazamiento (SCROLL) de la página hacia lo alto con la consiguiente pérdida de las primeras líneas. (En la modalidad PAGE con pantalla llena, el cursor no avanza).

PROTECT. Permite la protección de los caracteres con un particular atributo. El operador no puede modificar estos caracteres (el cursor no se para sobre los caracteres definidos protegidos). El modo de funcionamiento del vídeo puede variarse desde el programa con el envío de los oportunos códigos. Un código consiste en una secuencia de dos o más caracteres en los

que el primero es normalmente ESC [Escape = CHR$(27)] e indica al periférico que los siguientes caracteres deben interpretarse como controles y no como datos a escribir.
Por ejemplo, el código utilizado para pasar a modo conversacional es, para algunas estaciones de vídeo, el carácter C. Para transferir este carácter con el significado de comando que debe ejecutarse es necesario enviar la secuencia CHR$(27) + "C". El primer dato [CHR$(27)] genera el carácter correspondiente al código ASCII 27 (Escape) y permite interpretar el siguiente carácter (letra C) como un comando.

Gestión del cursor

La forma del cursor y el tipo de escritura también pueden variarse con los adecuados comandos. Los códigos citados a continuación son específicos de una máquina determinada (Buffetti B 801 con un terminal Seletron 3410); pero en la mayoría de los casos, la lógica y la modalidad de utilización permanecen invariables. Sólo en algunos casos se encuentran diferencias de importancia, que no obstante tienen un carácter exclusivamente formal (estructura de la instrucción). El cursor está constituido por una matriz de 6 puntos horizontales y 11 verticales que, en la puesta en marcha del vídeo, se presenta como un rectángulo luminoso fijo. Los comandos para su control permiten variar tanto el estado como la forma.
Los posibles estados son:

— cursor fijo
— cursor no visible
— cursor parpadeante (en algunas máquinas se puede variar la frecuencia de parpadeo)

La forma del cursor puede variarse especificando cuáles de las 6 filas y 11 columnas deben activarse. Los comandos deben proporcionarse en la siguiente forma:

ESC + "parámetros"

donde ESC es el carácter Escape [CHR$(27)] y "parámetros" indica una cadena de la que se escriben los códigos correspondientes a la función deseada. Por ejemplo, en la máquina examinada, el estado de cursor fijo tiene el código "c0", mientras que para hacer parpadear el cursor con frecuencia elevada, debe introducirse el código "c3". Enviando la instrucción

PRINT CHR$ (27) + "c0"

se tiene el cursor fijo para toda la escritura que sigue a la instrucción, hasta el siguiente comando

PRINT CHR$ (27) + "c3"

Después de la introducción de este comando, y hasta una nueva introducción de control, el cursor parpadea a alta frecuencia. La forma del cursor puede variarse con los códigos

PRINT CHR$(27) + "c4SE"

donde S es un carácter que indica en qué fila de la matriz de puntos reservada para ello debe ser activado el cursor, y E es un segundo carácter que especifica la columna de la matriz.
En la pág. 597 hay el listado de un programa que ilustra algunas funciones gráficas. Las funciones gráficas se controlan con el carácter ESC [CHR$(27)] más uno o dos caracteres de control (líneas 44 a 58). En la línea 100 se define una función (FNC$) constituida por el carácter Escape y una cadena [F$(N)] en la que hay escritos los códigos previstos mediante la instrucción DATA (líneas 140 a 210).
La función gráfica deseada se obtiene enviando a escritura la FNC$(N) con el adecuado valor de N. Por ejemplo, poniendo N = 3 se utiliza la cadena F$(3) = "G1" que tiene el efecto de producir el subrayado. De este modo, las instrucciones que permiten obtener el subrayado de una escritura son las siguientes:

N = 3
PRINT FNC$(N); "Línea de prueba"

En el programa se indica también la función de direccionamiento del cursor (línea 240) que permite posicionar una escritura en un punto cualquiera de la pantalla. Así, las instrucciones:

10 A$ = "Prueba"
20 X = 5:Y = 10
30 PRINT FNP$(X,Y); A$

posicionan la escritura "Prueba" a partir de la columna 5 (X = 5) de la línea 10 (Y = 10).

La máscara vídeo

La gestión de la unidad vídeo permite construir determinados programas que son utilizables

EJEMPLOS DE GESTION DEL CURSOR

```
10 ' *** Ejemplos de gestión del cursor
20 '  FILE : SVDU
25 OPTION BASE 1
30 DEFINT A-Z
42 '    COMANDO                              FUNCION
44 '        CHR$ (27)+")"      Escritura con luminosidad 1/2
46 '        CHR$ (27)+"("      Luminosidad normal: desactiva la anterior
48 '        CHR$ (27)+"G1"     Subrayado
50 '        CHR$ (27)+"G2"     Parpadeante
52 '        CHR$ (27)+"G4"     Invertido (negro sobre pantalla blanca)
54 '        CHR$ (27)+"G5"     Subrayado e invertido
56 '        CHR$ (27)+"G6"     Invertido y parpadeante
58 '        CHR$ (27)+"G0"     Anula las predisposiciones particulares
60 '
70 ' * FORMA DEL CURSOR : CHR$(27);"c4"+"S"+E"
80 '
90 '
100 DEF FNC$ (N)=CHR$(27)+F$(N)       'Función del modo (líneas 44-58)
110 DEF FNF$(S$,E$)=CHR$(27)+"c4+S$+E$    ' Forma (línea 70)
120 ' VALORES DE LAS CADENAS F$ QUE DEFINEN LA FUNCION
125 RESTORE 140
130 FOR I=1 TO 8 : READ F$(I):NEXT I
140 DATA")"
150 DATA"("
160 DATA"G1"
170 DATA"G2"
180 DATA"G4"
190 DATA"G5"
200 DATA"G6"
210 DATA"G0"
220 '
230    *** POSICION DEL CURSOR : X=COLUMNA  Y=LINEA
240 DEF FNP$ (X,Y)=CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X)
250 '
260 '  ** FUNCIONES PARTICULARES
270 '
280 BLK$=CHR$(27)+"+"              ' Anula toda la pantalla
290 BEL$=CHR$(7)                   ' Campana
300 '
310 ' ** ELECCION DEL CURSOR
320 '
325 X=2 Y=1:PRINT FNP$ (X,Y);
330 INPUT "Forma del cursor (S,E) ";S$,E$
360 PRINT FNF$(S$,E$)      ' Impone la forma del cursor
370 '
380 ' * TIPO DE ESCRITURA
390 PRINT FNC$(8)          ' RESTABLECE EL MODO NORMAL
400 INPUT"  TIPO DE ESCRITURA (de 1 a 8) ";N    'Ver las líneas DATA
410 IF N>=1.AND N<=8 GOTO 500              ' Modificado
420 ' ** ERROR **
430 '
440 X=35:Y=12:PRINT FNP$(X,Y)              ' Cursor al centro de la pantalla
450 N=6:PRINT FNC$(N);" ** ERROR **"
460 FOR I=1 TO 800:NEXT I                  ' Bucle de espera para la lectura
461 S$=INPUT$(1)
470 '
480 GOTO 260
510
520 INPUT" Introducir la cadena a escribir ";A$
530 INPUT" Introducir la posición (columna, línea) ";X,Y
540 K=LEN(A$)              ' Longitud de la cadena a escribir
550 M=80-K                 ' Controla si la cadena entra en pantalla
560 IF X<0 OR X>M THEN PRINT " ** ERROR ** ": GOTO 520
570 IF Y<0 OR Y>24 THEN PRINT " ** ERROR ** ": GOTO 520
580 '
590 ' **** ESCRITURA ****
610 PRINT FNP$(X,Y);FNC$(N);A$
640 INPUT" CONTINUA "; RESP$
650 IF RESP$="SI" GOTO 390
660 END
```

EJEMPLO DE GESTION DEL VIDEO

En esta página se han indicado algunas salidas en el vídeo obtenidas con el programa de la página anterior.

```
Forma del cursor (S,E)? A,9

TIPO DE ESCRITURA (de 1 a 8)? 3
Introducir la cadena a escribir? BASIC
Introducir la posición (columna, línea)? 10,8

        BASIC
CONTINUA ? SI

TIPO DE ESCRITURA (de 1 a 8)? 5
Introducir la cadena a escribir? BASIC
Introducir la posición (columna, línea)?  14, 15

              BASIC
CONTINUA ? SI

TIPO DE ESCRITURA (de 1 a 8)? 6
Introducir la cadena a escribir? BASIC
Introducir la posición (columna, línea)? 18,22

                    BASIC
CONTINUA ? _
```

La petición de los datos que definen la forma del cursor se activa con la ejecución de la línea 330. El operador ha insertado las cifras hexadecimales A y 9, que imponen la forma rectangular.

En la línea 400, el programa pide la introducción del código que representa el tipo de escritura. El valor introducido (3) corresponde al modo subrayado (línea 48).

La línea 520 del programa pide la introducción de la cadena a escribir en pantalla. La cadena BASIC está asignada a la variable A$.

En esta fase, el programa (línea 530) pide al operador las coordenadas del punto de la pantalla a partir de las cuales deberá aparecer la cadena A$. El operador elige el punto que se encuentra en el cruce de la columna 10 con la línea 8.

Se ejecuta la instrucción PRINT de la línea 610, que produce la aparición de la cadena introducida anteriormente (BASIC) escrita en el modo preseleccionado (3 = subrayado).

Por petición del programa (línea 640), el operador indica la intención de continuar. El control vuelve a la línea 390.

Las dos fases que siguen son otros dos ejemplos de escritura de la misma cadena: el primero en el modo invertido y el segundo en modo subrayado e invertido.

para la introducción de datos llamados **máscara vídeo**.
Utilizando las máscaras vídeo puede efectuarse una serie de controles que de otra manera serían irrealizables y, sobre todo, el usuario puede aportar correcciones en cualquier momento, incluso sobre los datos introducidos anteriormente. Si se utilizase la instrucción normal INPUT esto no sería posible, ya que los datos son adquiridos en el momento de la introducción y se transfieren inmediatamente al programa. La escritura de los programas para las máscaras vídeo comporta algunas dificultades; por tanto, el tema se afronta por grados. Para iniciarlo nos ocuparemos de una subrutina para la introducción y para el control de una fecha. En muchos programas de aplicación a veces debe introducirse la fecha como simple referencia para los tabulados, y alguna otra vez por exigencias fiscales. En ambos casos debe controlarse la validez del dato introducido. Podemos suponer que este control se reduce a verificar que el mes existe y que el día no es superior al valor máximo correspondiente al mes. En la pág. 600 se ha incluido el diagrama de flujo de la subrutina de control. El uso de esta subrutina requiere, en el main que la llama, dos matrices unidimensionales con las siglas del mes y con los días para cada mes. La lógica de control es la siguiente.
Llamando la rutina se deben preparar tres cadenas que contienen la fecha: DIA$ de 2 caracteres, MES$ de 2 caracteres y AÑO$ de 2 caracteres. En el interior de la rutina se confrontan las siglas contenidas en la variable MES$ con las previstas; si no las encuentra, el error está en el mes. Si las siglas se encuentran, su posición indica el número del mes (enero = 1, febrero = 2, etc.). El control de la validez del día consiste en verificar que el valor introducido (transformado de ASCII en numérico) sea menor o igual que el número de los días contenidos en el mes declarado (y verificado como válido). La única excepción a este procedimiento está prevista para el 29 de febrero; en este caso debe verificarse si el año es bisiesto o no. El control se desarrolla en una subrutina dedicada que verifica la divisibi-

SOLUCIONES DEL TEST 17

1 / La impresora construye cada carácter utilizando una «matriz» de puntos (agujas); la impresión se realiza por «impacto» mientras el carro se mueve en ambas direcciones. Pueden imprimirse 132 caracteres en la misma línea a la velocidad de 120 caracteres por segundo (cerca de una línea por segundo). La impresora posee un buffer en el que hay acumulados los caracteres procedentes del calculador, el cual puede contener hasta 4096 bytes.

2 / Los códigos CHR$(33) y CHR$(110) representan respectivamente los caracteres ! y n, mientras que CHR$(27) y CHR$(100) son códigos especiales. CHR$(27) es el código ESCAPE, que predispone la impresora para aceptar órdenes sucesivas; el código CHR$(10) es el LINE FEED, que ordena el salto de una línea.

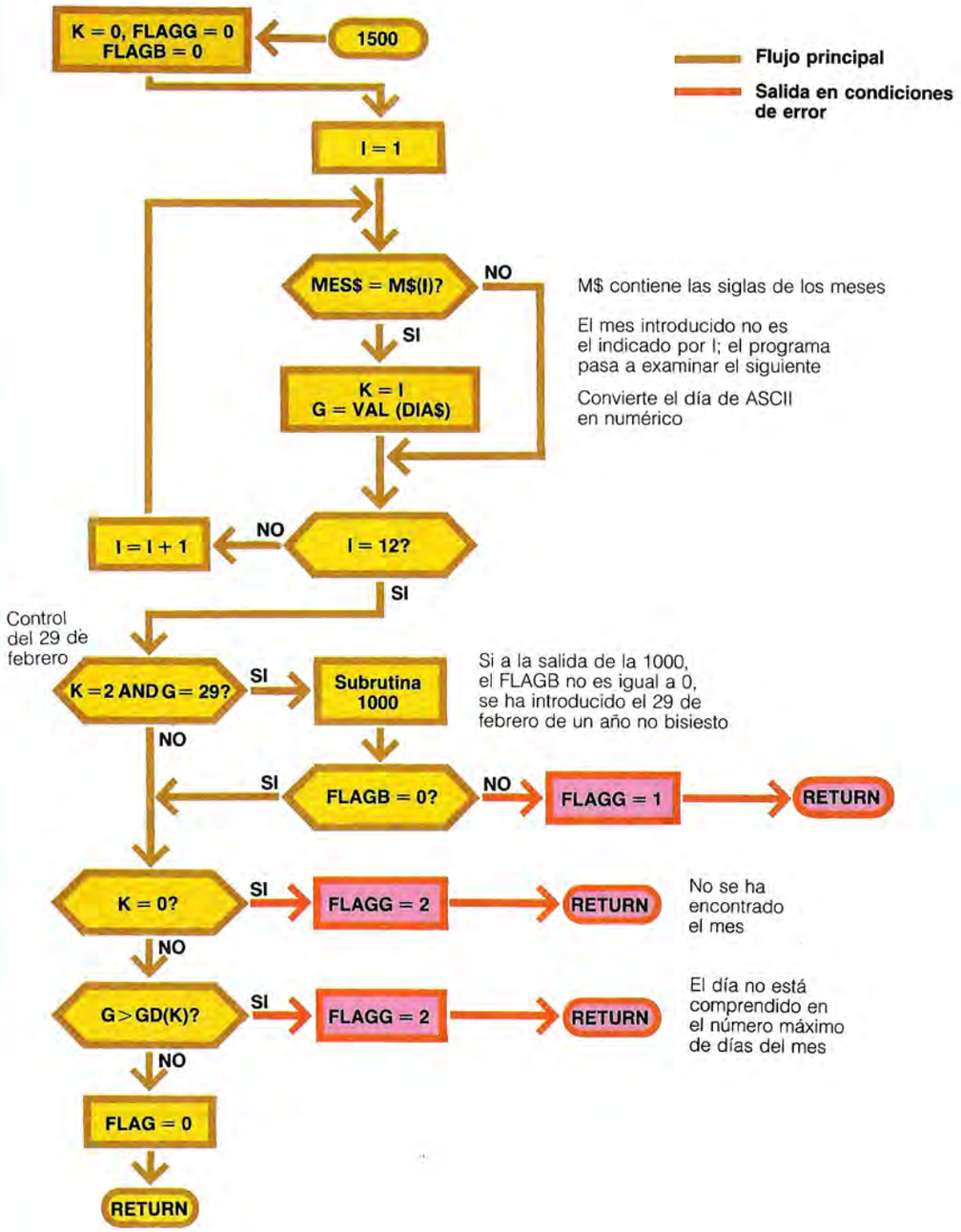
3 / La línea 20 imprime el contenido de la cadena A$ y de la variable C. La línea 30 contiene un error, ya que el formato utilizado sólo se refiere a valores numéricos, mientras que también se pide la impresión de una cadena. Una forma exacta de la instrucción es:

```
30 PRINT A$ ; : PRINT USING " # # # # . # " ; B
```

4 / Línea 10: los cinco valores se escriben uno por cada línea.
Línea 20: los cinco valores se escriben todos en la misma línea (símbolo ;).
Línea 30: Es un error, ya que TAB(10) seguida del símbolo ; pide la impresión de los cinco valores sobre la misma línea a partir de la misma columna (10). Por tanto, se tendrían cinco impresiones superpuestas.

SUBRUTINA DE CONTROL MES-DIA

Entradas: DIA$ = cadena que contiene el día (caracteres numéricos del 1 al 31)
MES$ = cadena que contiene el mes (siglas de tres letras)
AÑO$ = cadena que contiene el año (control de bisiestos)
Salida: FLAGG>0 si hay error, FLAGG = 0 para fecha correcta



lidad del año por 4 (ver gráfico inferior). Téngase presente que los años seculares no son bisiestos si no son divisibles por 400. En estas subrutinas se han previsto flags que indican, en caso de error, el tipo de error cometido.
En la pág. 602 aparece el diagrama de flujo de una subrutina que lee una fecha por el vídeo y la controla utilizando las subrutinas ilustradas antes. Esta subrutina es un ejemplo de introducción de datos con máscara vídeo.
La forma más sencilla e inmediata para comunicar al programa los valores de entrada es la instrucción INPUT. Por ejemplo, escribiendo:

```
INPUT DIA$, MES$, AÑO$
```

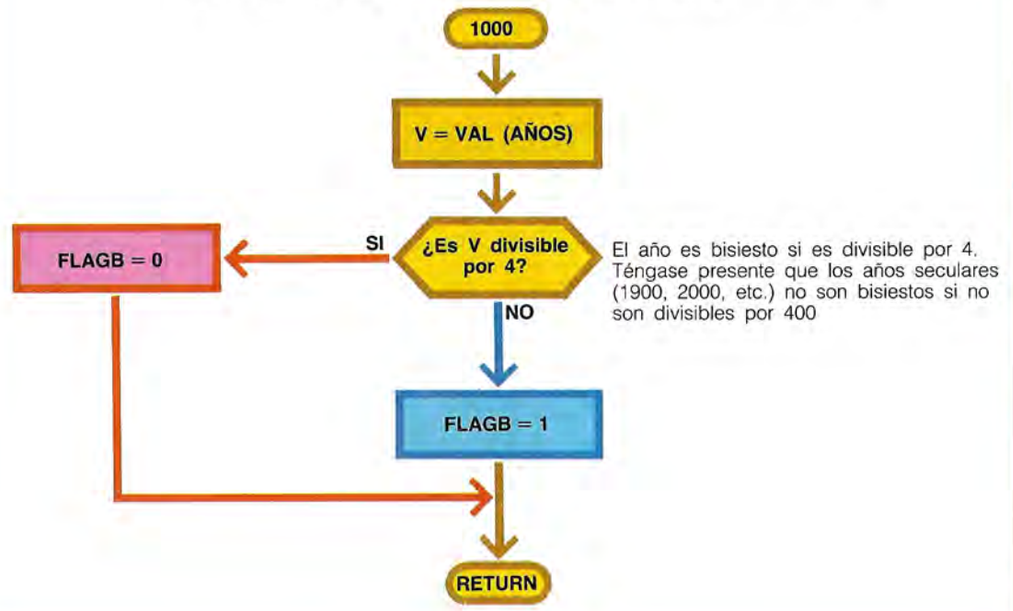
pueden adquirirse los tres valores (cadenas) que constituyen la fecha.
Esta forma de introducción tiene dos defectos principales. El usuario no tiene una guía inmediata acerca del formato pedido para los datos; no puede saber, por ejemplo, con cuántos caracteres debe indicar el mes. La posición de la línea de introducción, además, está determinada por el sistema y, en el caso de introducciones sucesivas, puede cancelarse.
Utilizando el procedimiento sugerido en el gráfico de la pág. 602 (ver el listado en las págs. 604-605), en pantalla aparece la escritura

```
../../..
```

con el cursor posicionado al principio del primer campo.
La introducción queda entonces completamente guiada. El usuario ve constantemente la longitud de los campos, y los separadores entre los diferentes datos (símbolos /) siempre están presentes. La escritura puede posicionarse en un punto cualquiera de la pantalla, donde permanece hasta que el programa no la cancela voluntariamente con una adecuada instrucción. Eventualmente, en otros puntos diferentes de la pantalla aparecen otras introducciones, las cuales no cubren la anterior. La subrutina se completa con instrucciones que permiten escribir eventuales valores erróneos con vídeo invertido, para llamar la atención del operador.

SUBRUTINA AÑO BISIESTO

Entrada : AÑO$ = cadena que representa el año
Salida : FLAGB = 0 si el año es bisiesto, FLAGB = 1 si no lo es
Algoritmos usados : el año es bisiesto si es divisible por 4

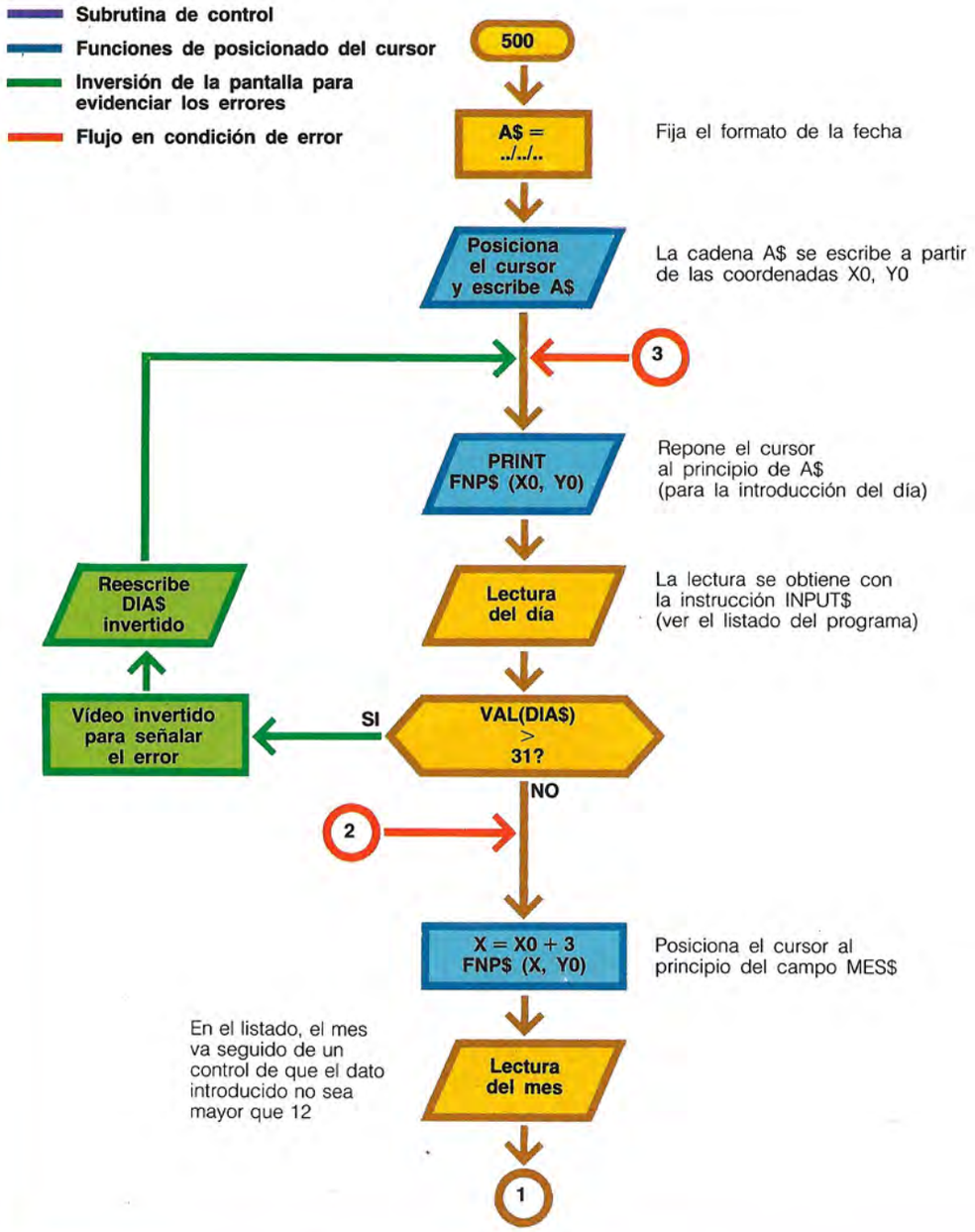


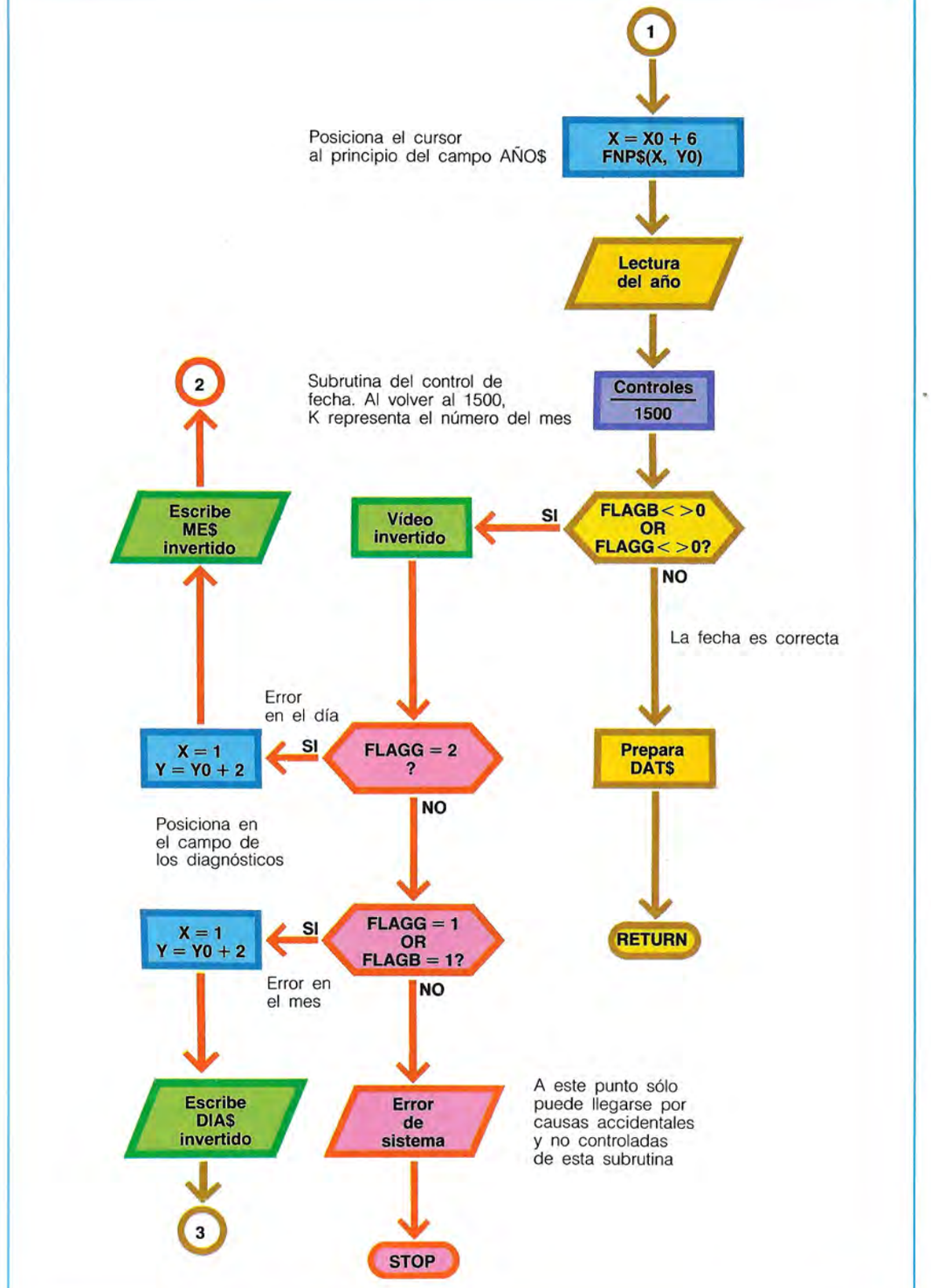
SUBRUTINA DE INTRODUCCION Y CONTROL DE FECHAS

La subrutina necesita dos matrices inicializadas con las instrucciones FECHA:
la primera [MS (12)] debe contener los números progresivos de los meses (1, 2, 3, etc.);
la segunda [GD (12)], el número de días correspondientes (31, 28, 31, etc.)

Entradas: X0, Y0 = coordenadas del punto de la pantalla en que debe empezar la fecha

Salida: FEC$ = cadena que contiene el día, el mes (numérico), año (las últimas 2 cifras)





PROGRAMA DE INTRODUCCION Y CONTROL DE FECHAS

```
10 '  **** MAIN DF PRUEBA ****
15 '  FILE : DATO
20 DEFINT A-Z
30 DEF FNP$(X,Y)=CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X)
40 DEF FNN$=CHR$(27)+"G0"
50 DEF FNI$=CHR$(27)+"G4"
60 DIM M$(12), GD(12)
62 FOR I=1 TO 12:READ M$(I):NEXT I
64 FECHA"01"
66 FECHA"02"
68 FECHA"03"
70 FECHA"04"
72 FECHA"05"
74 FECHA"06"
76 FECHA"07"
78 FECHA"08"
80 FECHA"09"
82 FECHA"10"
84 FECHA"11"
86 FECHA"12"
87 X0=1:Y0=10
88 FOR I=1 TO 12:READ GD(I):NEXT I
90 FECHA 31,28,31,30,31,30,31,31,30,31,30,31
91 PRINT CHR$(27)+"+"            ' BORRAR EL VIDEO
92 GOSUB 500
94 PRINT "La fecha ";FEC$;" es correcta"
100 INPUT" CONTINUA (SI/NO) ";RESP$
105 IF RESP$="SI" THEN Y0=Y0+1
110 IF RESP$="SI" THEN Y0=Y0+5: GOTO 92
120 END
500 ' * Subrutina de introducción fechas
501 X=X0:Y=Y0-2:PRINT FNP$(X,Y);
505 PRINT "INTRODUCIR LA FECHA CON EL FORMATO ../../.."
510 A$="../../.."
520 PRINT FNP$(X0,Y0);A$
530 PRINT FNN$                  ' VIDEO NORMAL
540 PRINT FNP$(X0,Y0);            ' POSICIONA
550 C$=INPUT$(1):PRINT C$;      ' PRIMER CARACTER
560 D$=INPUT$(1):PRINT D$       ' SEGUNDO CARACTER
570 DIA$=C$+D$
580 IF VAL (DIA$) <=31 GOTO 640       ' PROSIGUE SI EL DIA NO ES > QUE 31
590 ' **** ERROR DIA > 31 ****
600 PRINT FNI$                  ' INVERTIDO
610 PRINT FNP$(X0,Y0);          ' POSICIONA AL PRINCIPIO DEL CAMPO DIA$
620 PRINT DIA$                  ' REESCRIBE INVERTIDO
630 GOTO 530
640 ' EL DIA ES VALIDO
650 X=X0+3
655 PRINT FNP$(X,Y0);
660 C$=INPUT$(1):PRINT C$;
670 D$=INPUT$(1):PRINT D$
680 MES$=C$+D$:IF VAL (MES$)>12 GOTO 655
690 X=X0+6:PRINT FNP$(X,Y0);    ' CAMPO AÑO$
700 C$=INPUT$(1):PRINT C$;
710 D$=INPUT$(1):PRINT D$
720 AÑO$=C$+D$
730 ' ** CONTROLES **
740 GOSUB 1500
750 IF FLAGB<>0 OR FLAGG<>0 GOTO 790
760 ' FECHA CORRECTA
770 FEC$=DIA$+" "+MES$+" "+AÑO$
780 RETURN
790 ' **** ERROR ****
800 PRINT FNI$
```



```
810 IF FLAGG=2 THEN X=1:Y=Y0+2:PRINT FNP$(X,Y);:PRINT MES$:PRINT FNN$:GOTO 650
820 IF FLAGG=1 OR FLAGB=1 GOTO 880
830 '
840 ' **** ERROR EN LOS VALORES DE LOS FLAGS ****
850 '
860 PRINT " ******** ERROR DE SISTEMA *********"
870 STOP
880 '* PREPARA PARA CORRECCION
890 '
900 Y=Y0+2:X=1:PRINT FNI$:PRINT FNP$(X,Y);
910 PRINT DIA$
920 GOTO 530
930 '
940 ' ****
1000 ' * DETERMINA SI UN AÑO ES BISIESTO
1010 ' ENTRADA : AÑO$
1020 ' SALIDA  : FLAGB = 0 SI BISIESTO
1030 '
1040 V=VAL(AÑO$)
1050 FLAGB=V MOD 4:IF FLAGB>0 THEN FLAGB=1
1060 RETURN
1070 '
1500 '
1510 ' * CONTROL CONGRUENCIA DIA MES
1515 K=0:FLAGG=0:FLAGB=0
1520 FOR I=1 TO 12
1530 IF MES$=M$(I) THEN K=I
1540 G=VAL (DIA$)
1545 NEXT I
1550 IF K=2 AND G=29 THEN GOSUB 1000     ' CONTROLA SI EL AÑO ES BISIESTO
1560 IF K=2 AND G=29 AND FLAGB<>0 GOTO 1610 ' ** ERROR **
1585 IF K=2 AND G=29 GOTO 1600    ' AÑO BISIESTO DIA = 29 ES VALIDO
1590 IF K=0 THEN FLAGG=2:RETURN
1592 IF G>GD(K) THEN FLAGG=2:RETURN
1600 FLAGG=0:RETURN
1610 ' *** ERROR ***
1620 FLAGG=1
1630 RETURN
```

El método de invertir el vídeo en los datos erróneos es muy útil en los programas que necesitan numerosas introducciones, ya que permite identificar inmediatamente el error.

Las teclas funcionales

En muchos ordenadores existen algunas teclas cuyo código puede programarse. Estas teclas se llaman **teclas funcionales**, ya que se utilizan para desarrollar funciones programadas.
La pulsación de una tecla cualquiera de la consola genera el correspondiente código ASCII: la tecla de la letra A corresponde al valor 65 (decimal), a la tecla de la letra B el valor 66, etc. Las teclas funcionales no tienen asignados a priori ningún valor, y el programador puede asociar a cada una de ellas un código por medio de adecuadas instrucciones. El empleo de las teclas funcionales tiene la finalidad de obtener códigos (no previstos o no utilizados en la tabla ASCII) que tienen el significado de órdenes.

INTRODUCCION Y CONTROL DE FECHAS

En esta página se presentan algunas salidas en pantalla del programa representado en la pág. 604.


```
INTRODUCIR LA FECHA CON EL FORMATO ../../..

    29/02/84
La fecha 29 02 84 está corregida
 CONTINUA (SI/NO) ? SI

INTRODUCIR LA FECHA CON EL FORMATO ../../..

    9/02/83

29
```

La escritura de la cabecera se envía a impresión en la línea 505, y sirve exclusivamente como anotación para el operador.

La ejecución de la línea 520 produce la escritura de la máscara vídeo ../../.. La fotografía se ha tomado cuando el operador ya había insertado la fecha en la máscara.

A la introducción le sigue la fase de control de validez del dato introducido, realizada por programa. La fecha 29/02/84 ha resultado ser correcta (el 1984 es bisiesto), y el programa advierte al operador (línea 94).

Ante la petición del programa (línea 100), el operador indica la intención de continuar la introducción.

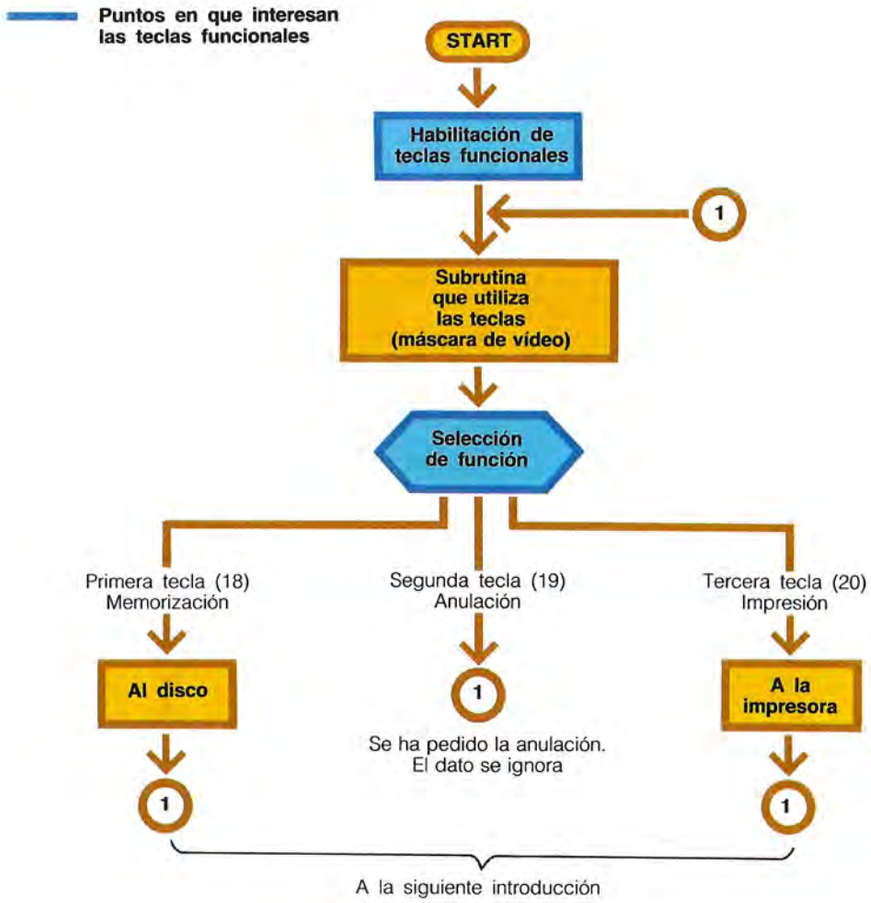
Esta vez se ha introducido una fecha errónea: el 29 de febrero de 1983 no puede existir, puesto que 1983 no es un año bisiesto.

El control de congruencia detecta el error, reescribe el día con vídeo invertido y posiciona el cursor al principio del campo día (líneas 600, 610 y 620).

Por ejemplo, asociando a una tecla funcional el valor numérico 18, que en ASCII no se utiliza ni para letras ni para números, el programa puede reconocer este código particular e interrumpir la función que estaba desarrollando para realizar la nueva orden. El código 18 podría hacerse corresponder a la función de memorización en disco, al código 19 de la impresora, etc. Abajo se ve el diagrama de flujo que muestra el uso de esta lógica. En este ejemplo, las teclas habilitadas son tres, y tienen los valores 18, 19 y 20.
La asociación de cada tecla con un valor numérico (código) debe implantarse por programa antes de la utilización de las teclas. En general, por tanto, en cada programa que haga uso de las teclas funcionales debe incluirse una rutina de iniciación de los códigos. La asociación entre una tecla funcional y un valor numérico, una vez implantada, permanece válida hasta el apagado de la máquina o hasta una nueva asignación para la misma tecla. La indicación de las teclas funcionales puede seguir criterios diferentes en las diversas máquinas.
En la máquina particular a la que se hace referencia (Buffetti B801), las teclas están marcadas con letras minúsculas*. La instrucción de inicialización para esta máquina está constituida por la siguiente serie de caracteres:

* La letra que designa una determinada tecla funcional también puede variar según la forma en que se hayan realizado las conexiones eléctricas del teclado.

LOGICA DE UTILIZACION DE LAS TECLAS FUNCIONALES



— Código de Escape = CHR$(27)
— Código que indica la función de inicialización = CHR$(46)
— Identificación de la tecla (p. e. c; en ASCII c = 99) = CHR$(99)
— Código a asociar a la tecla dividido en sus cifras componentes

A$ = CHR$(27) + CHR$(46) + CHR$(99) + CHR$(52) + CHR$(49)
Escape — Código de implantación — Identificación de la tecla (c) — Caracteres que representan el código a asignar (A)

Por ejemplo, para asociar la tarea A a la tecla funcional c, debe enviarse el código ASCII de la letra A (41 hexadecimal), descompuesto en los dos caracteres 4 y 1, cada uno en su propia representación ASCII. El código ASCII de 4 es 52 y el de 1 es 49; por tanto, la letra A está representada por dos caracteres CHR$(52) y CHR$(49).
La instrucción de inicialización completa viene dada por la siguiente cadena:

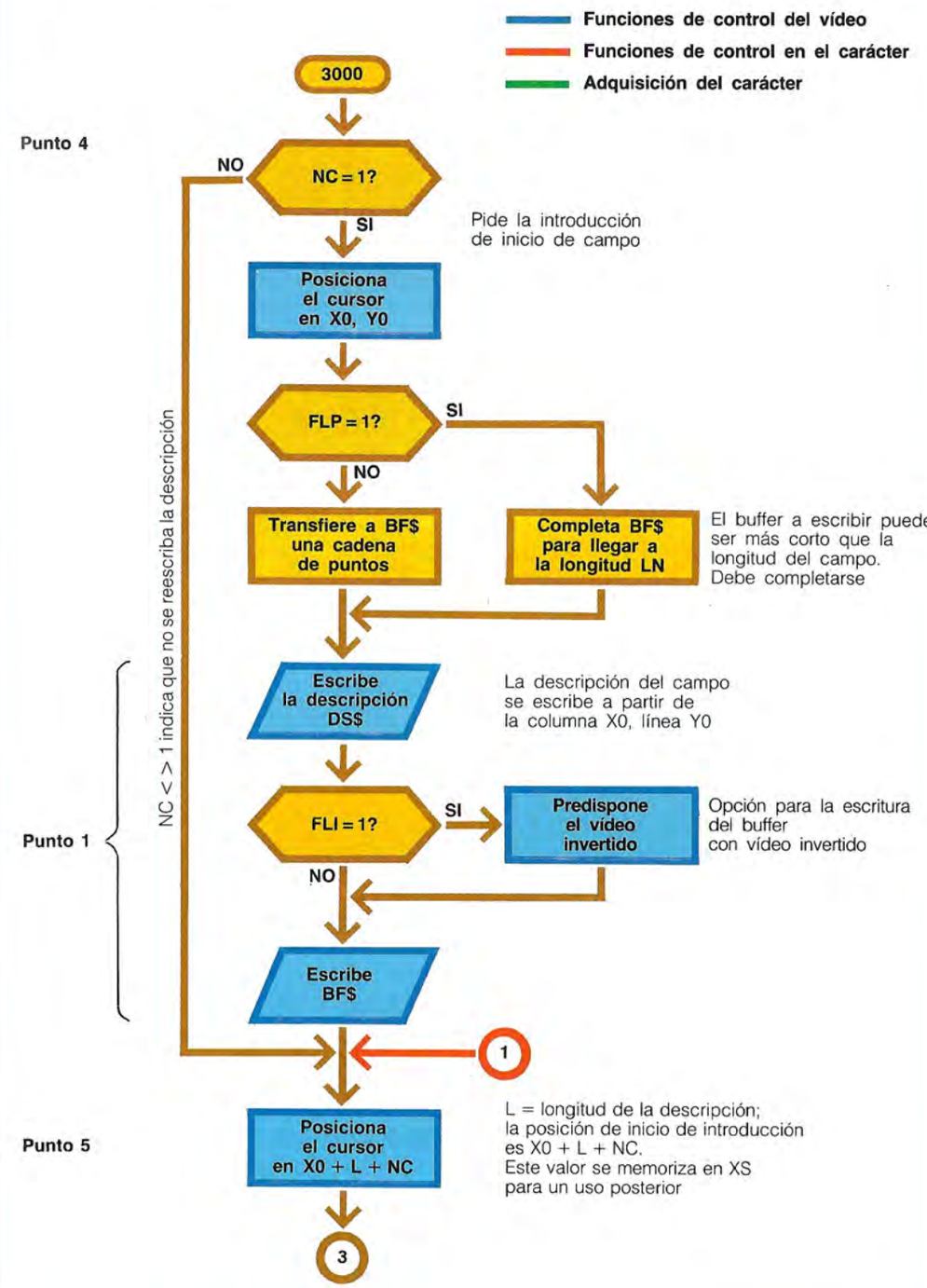
La habilitación de la tecla se obtiene simplemente enviando a escritura la cadena A$ con la instrucción PRINT A$.
Abajo se ha listado una subrutina de inicialización que asigna los valores 17, 18 y 19 decimales a otras tantas teclas funcionales. Una aplicación inmediata de éstas se tiene en los programas de introducción de datos con máscara vídeo. En las págs. 609 y 610 se ve el diagrama de flujo de primer nivel de una subrutina

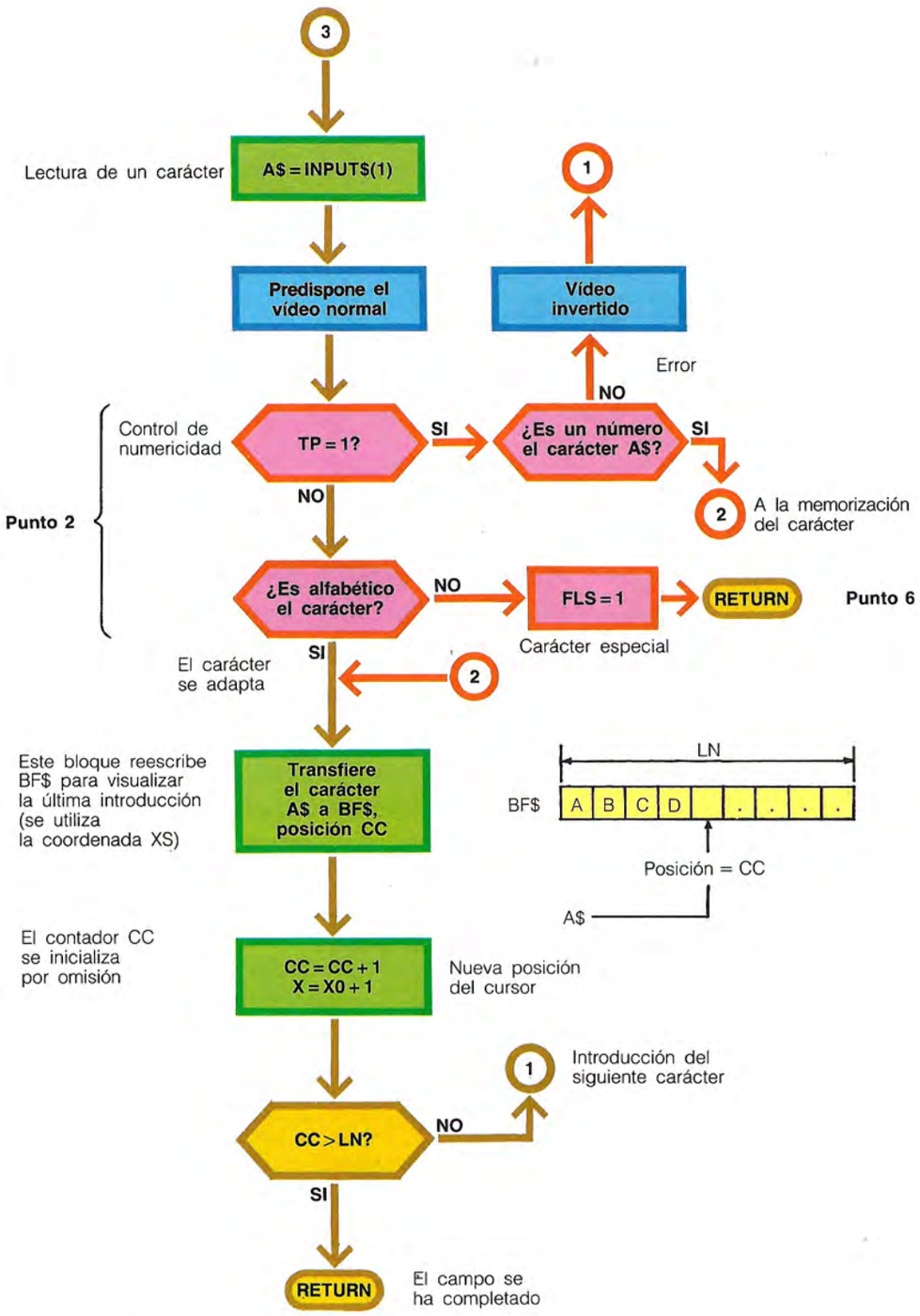
EJEMPLO DE ACTIVACION DE LAS TECLAS FUNCIONALES

```
10 ' ** EJEMPLO DE ACTIVACION DE LAS TECLAS FUNCIONALES
15 '   FILE : TFUN
30 DEFINT A-Z
40 A$=CHR$(27)+CHR$(46)                 ' PARTE COMUN DEL COMANDO
50 B3$=CHR$(99)                         ' TECLA N.2 = c
60 B4$=CHR$(100)                        ' TECLA N.3 = d
70 B5$=CHR$(101)                        ' TECLA N.4 = e
71 ' ** ASIGNACION VALORES A LAS TECLAS FUNCIONALES **
72 '    Los valores deben expresarse en hexadecimal
73 '    por ejemplo, el valor 11 (hexadecimal) equivale al
74 '    17 decimal. Asignando este valor a una tecla,
75 '    la función ASC(*) restituye 17
80 C1$=CHR$(49)+CHR$(49)    ' TECLA N.2
90 C2$=CHR$(49)+CHR$(50)    ' TECLA N.3
92 C3$=CHR$(49)+CHR$(51)    ' TECLA N.4
110 PRINT A$+B3$+C1$              ' ACTIVA LA TECLA 2
120 PRINT A$+B4$+C2$              ' ACTIVA LA TECLA 3
130 PRINT A$+B5$+C3$              ' ACTIVA LA TECLA 4
140 '
150 PRINT " Pulsar una tecla cualquierá"
160 I$=INPUT$(1)
162 ' Para pedir el valor asociado a la tecla pulsada
163 ' se utiliza la función ASC(*) que restituye el
164 ' valor decimal, por tanto, si la tecla tiene asociado 11,
165 ' el valor restituido es 17 (17 decimal=11 hexadecimal)
170 IF ASC(I$) <17 OR ASC(I$)>19 GOTO 260 ' Los valores asociados a
172 la tecla son:                        ' 17,18,19
180 K=ASC(I$)-16     ' K varía entre 1 y 3 en función de la tecla pulsada
190 ON K GOTO 200,220,240
200 NU=2:GOTO 250
210 GOTO 150
220 NU=3:GOTO 250
240 NU=4
250 PRINT " SE HA PULSADO LA TECLA FUNCIONAL N.: ";NU
255 GOTO 150
260 PRINT" LA TECLA PULSADA NO ES FUNCIONAL"
270 INPUT " CONTINUA ";RESP$
280 IF RESP$="SI" GOTO 150
300 END
```


SUBRUTINA DE INTRODUCCION DE DATOS. DIAGRAMA INDICATIVO

En esta subrutina falta la fase de alineado del dato (Punto 3) que conviene desarrollar por separado





El problema de la reserva

Entre las muchas expresiones que ha difundido la jerga de la era del calculador electrónico figura la de banco de datos. Los bancos de datos constituyen actualmente parte del cerebro gestor de organismos gubernativos, sociales o privados, y en todos los países industrialmente avanzados cada ciudadano es objeto de registro al menos en un archivo electrónico. Este sistema de registro no significa necesariamente que existan siniestras implicaciones; sin él, las administraciones estatales modernas no podrían ser eficientes, los servicios sanitarios no podrían llevar a cabo su labor y no habría modo de hacer respetar las leyes referentes a la motorización y el tráfico. Sin un censo nacional, la planificación de impuestos, la construcción de obras civiles y el desarrollo urbanístico quedarían sobre el papel. Pero hay también algunas consecuencias alarmantes. En los Estados Unidos, más de dos mil agencias para la concesión de créditos registran de este modo muchas informaciones confidenciales. Como ejemplo, baste decir que una de estas compañías cuenta en sus archivos con 100 millones de dossiers. Lo que realmente está cambiando no es tanto la cantidad de las informaciones recogidas como los métodos de registro de tales informaciones. Desde hace mucho tiempo se disponía ya de datos similares, amontonados en interminables hileras de estantes en los archivos tradicionales. Pero hoy, la capacidad de almacenamiento de los calculadores electrónicos, enormemente superior, junto con la gran facilidad de acceso, incluso a través de terminales que proporcionan datos del otro extremo del mundo, ha cambiado el proceso para hallar los datos.

Sin controles internacionales eficientes, que, lentamente, se está procurando introducir aunque con dificultad, una sola persona no tiene prácticamente posibilidad de conocer lo que de ella se ha registrado en los bancos de datos, como tampoco a quien se han pasado las informaciones. Los valedores de las libertades civiles se cuentan entre los muchos que han intentado establecer el derecho a la reserva del ciudadano, concepto que se ha definido como «el derecho del individuo al control sobre la difusión de las informaciones que le afectan».

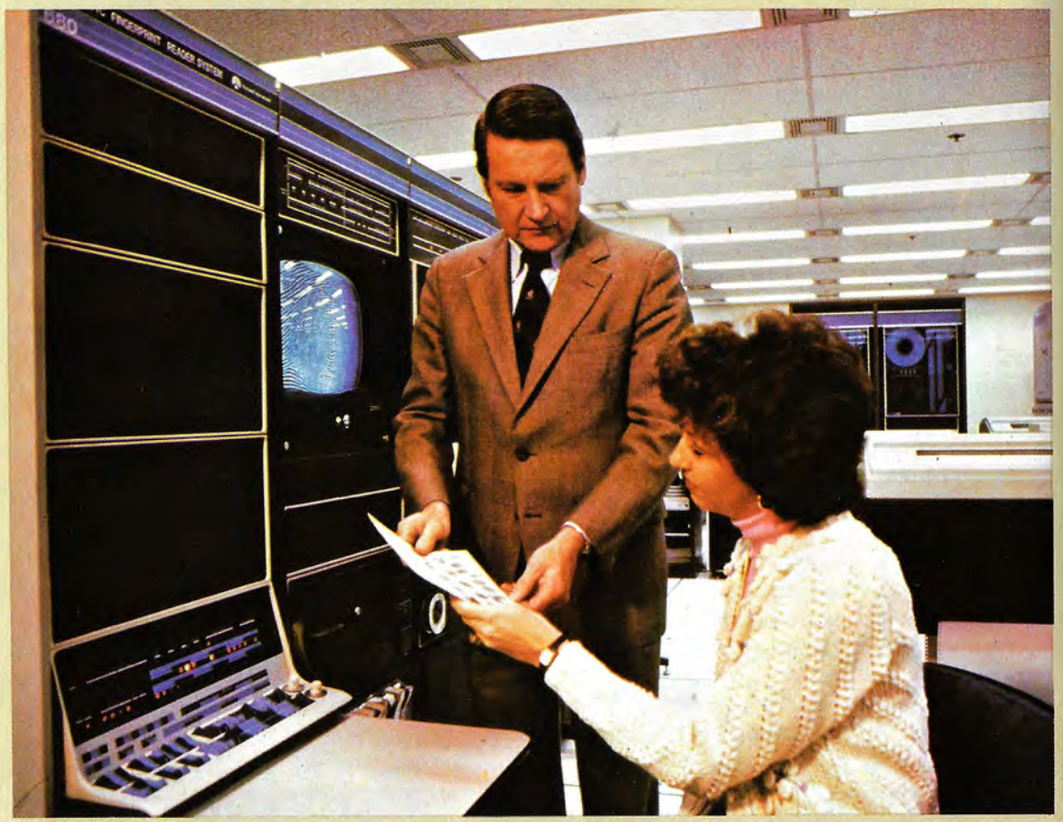
Todos los ciudadanos están obligados necesariamente a dar algunas informaciones personales a ciertas grandes administraciones, que las introducen en sus bancos de datos. Sin este requisito, no sería posible abrir una cuenta en un banco o conducir un automóvil. Sin embargo, deberían existir unos límites tanto sobre la cantidad de informaciones guardadas en los bancos de datos como sobre su accesibilidad. En realidad, las informaciones podrían ser erróneas, incompletas, inactuales o inútiles. Y, lo que es aún peor, una información facilitada para una finalidad determinada podría ser transmitida a otra administración y utilizada para un fin completamente distinto. También es posible que las informaciones se hayan recogido clandestinamente o con métodos ilegales, sin que el ciudadano al que conciernen lo haya sabido.

La cantidad de informaciones introducidas ya en los bancos de datos de los cuerpos de policía es impresionante. Por más democrático que sea un Estado, el ordenador de la policía contendrá, probablemente, una vasta gama de informaciones que va desde las referentes a los propietarios de automóviles, las listas de vehículos robados y de automovilistas indisciplinados, hasta las noticias sobre personas desaparecidas o buscadas, pasando por las características somáticas y huellas dactilares de los criminales. Pueden almacenarse también datos relativos a personas sospechosas de terrorismo o de actividades políticas clandestinas.

Puede ocurrir que una información facilitada a una autoridad por deber civil sea transmitida a otra que le dará un uso totalmente distinto. Por otra parte, las informaciones que posee la policía están sujetas a difusión, ya que son muchas las agencias de investigación privada, propiedad muchas veces de expolicías, que se valen en ciertas ocasiones de sistemas ilícitos para acceder al procesador y leer lo que contiene un determinado archivo. Asimismo, algunos delincuentes, conocedores de un determinado número de teléfono reservado, han podido obtener información haciéndose pasar por agentes de policía.

Hay grandes dificultades para asegurar la reserva de las informaciones contenidas en los bancos gestionados mediante los procesadores electrónicos. Con los viejos sistemas de archivo, era necesario que una persona los conociera materialmente para extraer la información deseada, y siempre era posible equipar los archivos con adecuadas cerraduras de seguridad. Si alguien conseguía llegar ilícitamente hasta los documentos, siempre tenía un límite físico en

cuanto a la cantidad de noticias que podía copiar, fotografiar o llevarse consigo.
Sin embargo, los archivos electrónicos han hecho más difícil para el buscón medio la obtención de informaciones, ya que los programas de los ordenadores requieren unos códigos de acceso que hacen que la información sólo esté a disposición de personas autorizadas. Pocas personas no pertenecientes a la industria de los procesadores tienen la experiencia necesaria para «descerrajar» estos códigos, pero si un ladrón decidido y competente quiere leer el contenido de un banco de datos, no hay procesador que se lo pueda impedir. Y aún más: puede conseguir una gran cantidad de información de una sola vez.
Pocas son las manipulaciones fraudulentas de procesadores llegadas a conocimiento del público, pero cuando así ha sido, el hecho ha adquirido gran resonancia. Algunos estudiantes del Massachusetts Institute of Technology consiguieron violar un procesador que contenía informaciones consideradas de la máxima importancia para la defensa nacional. Estudiantes de la Universidad de Michigan han descifrado los códigos del ordenador de la propia escuela y borrado cierto número de archivos. Un apasionado de los procesadores, que acabó en la cárcel, consiguió obtener noticias sobre personas sospechosas o detenidas de los mismos archivos del FBI.
Como los procesadores electrónicos están al servicio de terminales distantes, o son servidos por ellos, las posibilidades de error aumentan. Un acceso casual, debido a una interferencia entre líneas telefónicas, puede ya representar una violación del secreto de una noticia o, incluso, su borrado. Se plantea aquí la siguiente pregunta: ¿qué puede hacerse para salvaguardar la vida privada del ciudadano de forma compatible con las necesidades del gobierno y de otras administraciones de registrar ciertas informaciones para bien del propio ciudadano? Las fuerzas de la policía, por ejemplo, reciben sobre los ciudadanos una cantidad de informaciones bien distintas de las referentes a las características de los automóviles que poseen. En Gran Bretaña, los sospechosos arrestados de acuerdo con las disposiciones adoptadas contra el terrorismo son automáticamente fotografiados y deben marcar sus huellas dactilares. Incluso si luego no son procesados (y muchos no lo son) las huellas dactilares quedan en los archivos. Las personas que se hallan bajo vigilancia porque son sospechosas de preparar una acción criminal, pueden ser registradas en los archivos, como también pueden serlo todas aquellas asociadas de cualquier manera a aspectos delictivos. Todo esto constituye una acción de policía preventiva que tiene la legítima finalidad de prevenir los delitos antes de que ocurran.
Algunas agencias especializadas disponen de «registros de crédito» que ponen a disposición de las sociedades que desean tener informes sobre la situación financiera de sus potenciales clientes. El mayor banco de datos del mundo pertenece a una de estas agencias, la TRW Credit Data, que posee informes personales de más de 50 millones de ciudadanos americanos.
En Suecia, los principales bancos de descuento han creado recientemente un instituto, la Agencia de Informaciones para el Crédito, que facilita nombre, dirección, ingresos, estado civil, actividades sujetas a impuestos, contratos de arrendamiento o de compra, situación penal de todos los ciudadanos suecos mayores de 16 años. De cualquier modo, se entrega al interesado una copia de cada información que el instituto da a terceros. No obstante, es frecuente que las agencias de crédito no controlen la exactitud de las noticias que adquieren. Su papel es el de limitarse a recoger y distribuir informes sobre deudas y créditos inexigibles, y las organizaciones que los piden pueden llevar a cabo las investigaciones pertinentes antes de conceder cualquier crédito.
Si bien los registros de crédito son un medio de defensa útil y legítimo contra endeudamientos peligrosos, existen, sin embargo, grandes posibilidades de error. He aquí lo que se cuenta de un hombre de negocios, propietario de unos bienes muy saneados, que había pedido un crédito para ampliar sus actividades: su petición fue rechazada. Desorientado, ordenó una investigación sobre la situación de su propia sociedad y ¡se le aconsejó no mantener tratos con ella! Profundizó entonces en las indagaciones hasta que consiguió descubrir que un homónimo suyo había sido recientemente denunciado al juzgado.
Los servicios sociales, que también son tan importantes para la sociedad, son los que ofrecen mayores posibilidades de violación de la libertad individual. Muy a menudo, los asistentes sociales están al corriente de detalles de la máxima reserva que algunas personas les han confiado en momentos de gran angustia.
Puede ocurrir que el material recogido en un banco de datos se base en el juicio subjetivo de un funcionario, expresado en una jerga seudocientífica que le da un sello de credibilidad. «Paranoico» o «neurótico» son términos que aparecen con frecuencia en los informes de los asistentes sociales sobre personas que buscan ayuda; efectivamente, muchas de estas personas no solicitarían estas ayudas si no se hallasen momentáneamente en un estado de tensión nerviosa. Pero cuando una información así es codificada e introducida en el ordenador, tiende a adquirir autenticidad por el mero hecho de haber sido registrada. Incluso los tribunales aceptan, generalmente, las comunicaciones de los asistentes sociales como procedentes de expertos testigos, por lo cual un juicio casual puede transformarse rápidamente en una etiqueta que queda fijada al individuo para toda la vida.
Las informaciones sobre ciudadanos recogidas por las autoridades locales quedan articuladas en un gran número de secciones de acuerdo con los diversos departamentos: educación, vivienda, trabajo, hacienda y servicios sociales. Las informaciones de carácter financiero son abundantes en los archivos de las administraciones locales que están en posesión de los registros contables. Las autoridades locales recogen, además, informaciones financieras sobre cada individuo como elementos para desglosar casi cuarenta tipos de asistencia.
Actualmente, existe una creciente tendencia a introducir en el procesador toda la información recogida por las administraciones. Aunque es teóricamente posible codificar las informaciones de manera que cada oficina sólo tenga acceso a aquéllas que son de su competencia, no hay norma legal alguna que prohíba el intercambio de información entre distintas oficinas o, lo que es más, con organizaciones exteriores, como la policía. Verdaderamente, para que una persona pueda recibir los beneficios de la asistencia social es posible que su dossier tenga que pasar necesariamente por numerosas y distintas oficinas. El asistente social, por ejemplo, puede no estar en condiciones de ayudar a una persona si no recurre a otros profesionales, como el médico de la familia, el personal del hospital, los maestros, algunos funcionarios de policía o de los tribunales, o el encargado de la vigilancia en los casos de libertad condicional. Como las personas objeto de tales informaciones no están autorizadas a examinar su propio dossier personal, las informaciones que hay sobre ellas –verdaderas o falsas, en cualquier caso opinables– pueden salir en el momento menos adecuado, cuando es imposible responder de eventuales errores o solicitar pruebas que justifiquen juicios poco favorables. Se han mencionado casos en los que el informe del asistente social sobre un asistido –informe con la opinión personal del que lo escribió sobre ciertas debilidades del cliente– ha sido leído en voz alta durante la audiencia para la concesión de la ayuda. En otro caso, un hombre que vivía en una residencia para jubilados tuvo que escuchar cómo, en una reunión que tuvo lugar en la misma residencia, un asistente social leía en voz alta un escrito en el que se le acusaba de querer asesinar al doctor. Luego se descubrió que el contenido del escrito no correspondía a la verdad.
En muchos países se está pensando en la necesidad de promulgar leyes para la protección de la vida privada, y algunos de ellos están ya llevándolo a cabo. En Estados Unidos existe una abundante legislación que da libre acceso a las notas personales guardadas en las oficinas federales, e incluso está reconocido el derecho de impugnarlas y hacerlas corregir.
Suecia, con su sistema de registro de datos personales centralizado y altamente automatizado, posee asimismo una tradición para el acceso personal y una legislación sobre la concesión de permisos e inspecciones a toda organización que desee gestionar un banco de datos. Las leyes de la República Federal de Alemania establecen que todos los datos personales introducidos en los procesadores han de destruirse o ser definitivamente arrinconados después de un período de cinco años. Francia y Holanda poseen leyes que prohíben la recogida de datos no autorizada y Francia, además, tiene una ley que prohíbe la computerización de las informaciones sobre origen étnico de cualquier persona, así como sobre sus creencias políticas y religiosas. Dinamarca posee también una ley que prohíbe la difusión no autorizada de información concerniente a los asuntos privados de los ciudadanos.
La opinión sobre la existencia de controles legales y el modo en que debería recogerse la información está enormemente dividida; en cualquier caso, debería estarse siempre dentro de unos límites discretos y reducir aquella información al mínimo indispensable. Por otra parte, los



J. Pickerell/Marka

Los bancos de datos del FBI (Federal Bureau of Investigation) son gestionados por el ordenador. En la fotografía, un detalle del archivo de las huellas digitales.

informes dados para un fin determinado no deberían utilizarse para otro fin sin el consentimiento de la persona a la cual se refieren. Los sistemas de registro de los datos personales no tendrían que estar ligados entre sí, y las personas encargadas de la recogida y conservación de las informaciones tendrían que respetar un código de comportamiento que estuviera bajo la vigilancia de una autoridad especialmente designada para ello.

En Gran Bretaña se han corregido algunas imperfecciones a raíz de la promulgación de la Consumer Credit Act. Actualmente, a todo aquel que solicite la concesión de un crédito se le hace saber la agencia a la que se consultará en demanda de informes, y el interesado puede pedir la aportación de eventuales correcciones. Sin embargo, no se prevé aún la posibilidad de que surjan ciertos inconvenientes; por ejemplo, no se obliga a la agencia a facilitar las necesarias correcciones a otras firmas a las que podrían llegar peticiones de informes sobre el mismo cliente, y no se pone límite alguno a los medios a los que la agencia puede recurrir para la obtención de informaciones. Tampoco se ponen obstáculos a las agencias que recogen o venden información no pertinente, como la referente a las ideas políticas o a la vida sexual de las personas objeto de la investigación. Por último, tampoco está limitada la duración de la conservación de los informes, que en algunos casos podrían referirse a una deuda ya extinguida. Aunque existe una gran afinidad de ideas en cuanto a una mayor protección de la reserva personal, es verdaderamente difícil prever el modo en que estas ideas pueden ser puestas en práctica.

Y entre los problemas que van surgiendo, no es el último la rapidez con la cual se desarrolla la industria de los calculadores, frente a la cual las legislaciones quedan siempre inevitablemente atrasadas.

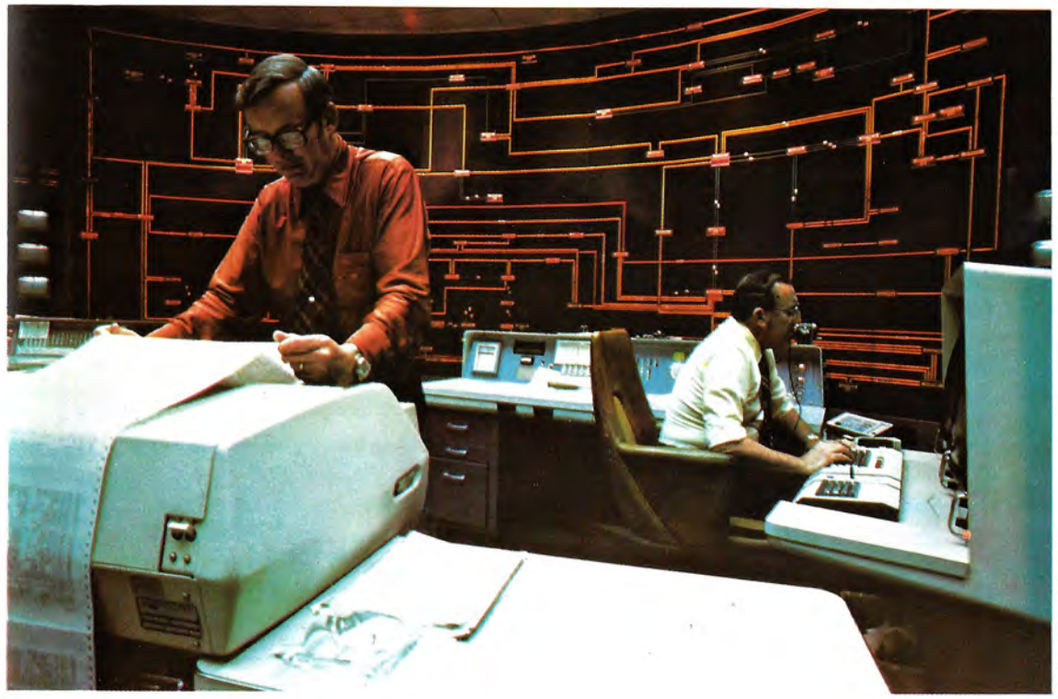
generalizada para la introducción de un dato. Esta subrutina se explicitará más adelante. Por lo que interesa aquí, la finalidad es la de realizar una rutina que realice estas funciones:

1 / Escritura en un punto cualquiera de la pantalla de la descripción del campo a introducir, con presentación de la longitud, mediante tantos puntitos como caracteres hay previstos en el campo.
2 / Control del tipo en cada carácter introducido (numérico, alfanumérico, especial). En caso de error, el cursor no avanza; el carácter equivocado queda escrito invertido y el programa espera la corrección.
3 / Si el campo no está completamente lleno, el dato debe escribirse alineado a la derecha si es numérico y a la izquierda si es alfanumérico.
4 / La subrutina debe trabajar sobre un buffer de tránsito en el que puedan apoyarse los datos introducidos, y debe prever su escritura para un determinado valor de flag. De este modo, llamando la rutina es posible bien escribir un dato a completar, bien indicar sólo el punteado que indica una introducción completa.
5 / La posición inicial del cursor (punto a partir del cual empezará la introducción) deberá establecerse por el programa que llama, de manera que se utilice la rutina también para una introducción parcial.
6 / En el momento de la introducción de un carácter no reconocido (es decir, no perteneciente ni a un número ni a una letra), el equivalente numérico del carácter se transfiere a una variable numérica y la rutina se abandona. Esta función se utiliza directamente en la gestión de las teclas programables.

La utilidad de las funciones descritas en los puntos 4 y 5 se aclarará cuando se hable más extensamente de las máscaras vídeo. Las variables utilizadas en el programa son:

BF$ = buffer de apoyo en el que se depositan los datos introducidos
LN = longitud del campo (en número de caracteres)
TP = tipo de campo (1 = numérico, 2 = alfanumérico)
DS$ = cadena con la descripción del campo
X0,Y0 = coordenadas de posicionado del campo sobre la pantalla del vídeo

La sala de control de una moderna central eléctrica.



K. Reese/Marka

NC = número de caracteres con que debe empezar la introducción
CC = contador del número de caracteres introducidos
FLS = flag de carácter especial: a la entrada de un carácter especial se pone FLS = 1
FLP = flag de escritura; si FLP = 1, entonces el contenido de BF$ debe escribirse en el vídeo
FLI = flag de inversión; llamando la rutina con FLI = 1, la escritura se invierte.

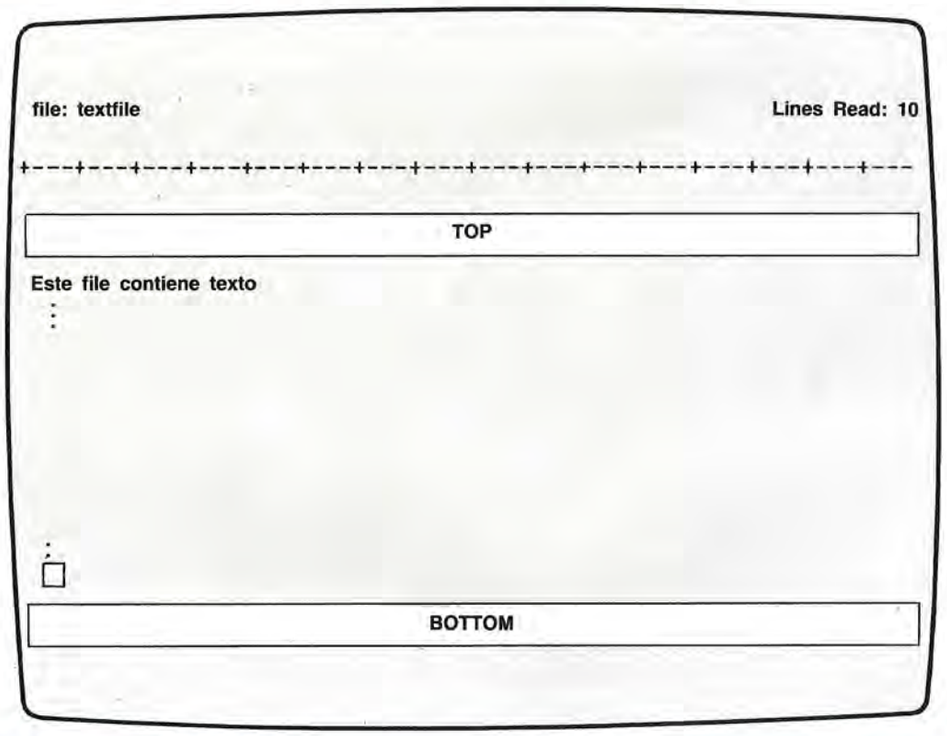
En esta subrutina existen dos salidas. La primera es activada cuando el dato se ha completado, es decir, cuando el correspondiente campo se ha llenado en toda su longitud; la segunda es activada con el reconocimiento de un código que no pertenece a las letras o a los números (punto 6). El valor de este código puede ser elaborado por el programa que llama, que puede utilizarlo para seleccionar una determinada función a desarrollar.

El Vídeo File Editor Olivetti

La gestión completa de la unidad vídeo permite poner a punto y utilizar determinados programas para la preparación de textos con el calculador o para la representación gráfica de los datos. El primer ejemplo a considerar es un programa de aplicación de Olivetti, el **Vídeo File Editor***, que permite notables posibilidades en fase de escritura y la modificación de un file, ya sea un programa (escrito en Basic y memorizado en formato ASCII) o un file de datos.
Se describirá cualitativamente, a título de ejemplo, las funciones previstas de este programa.
El Vídeo File Editor permite crear y modificar un **file de texto**, nombre con el cual se denomina un file de records compuesto de caracteres ASCII que pueden imprimirse, separados por caracteres CR/LF (Carriage Return/Line Feed) o por caracteres RS (Record Separator).
Cuando se entra en el Vídeo File Editor (mediante el comando EDIT del PCOS), si el file especifi-

* Vídeo File Editor es una denominación registrada.

MONITOR DEL VIDEO FILE EDITOR OLIVETTI



cado todavía no existe, lo crea y lo vuelve a llamar de la memoria del disco.
El Vídeo File Editor siempre visualiza una «ventana» de 24 líneas de 80 columnas cada una (21 líneas reservadas al file y las demás de servicio). Esta ventana, que permite visualizar cualquier parte del file, sólo muestra inicialmente la primera parte. El usuario puede operar, a través de las funciones y los comandos del Vídeo File Editor, para modificar en la memoria las líneas del file de visualización. Cuando se sale del Vídeo File Editor pueden registrarse en el disco las modificaciones realizadas en el file. Apenas se llama Vídeo File Editor, aparece la imagen que se ve en la pág. 616. La primera línea presenta el nombre del file y el mensaje existente. La segunda, está inicialmente vacía. A continuación puede utilizarse para la búsqueda de cadenas o para los comandos.
La tercera línea es la línea de escala, que presenta las posiciones de columna y las zonas de la pantalla. La pantalla está dividida en zonas, compuestas por cuatro caracteres. Las restantes 21 líneas, de la cuarta a la veinticuatroava, presentan la ventana de texto (text window). La veinticincoava línea no se emplea.
El principio y la línea del file siempre se enmarcan con dos líneas «virtuales» (llamadas TOP y BOTTOM) que no pertenecen al texto, pero que sirven como referencia y contienen las palabras TOP y BOTTOM, escritas en negro sobre fondo blanco. Cuando se entra en el Vídeo File Editor (mediante el comando EDIT), la línea TOP aparece inmediatamente antes de la primera línea del file y el cursor se posiciona en la línea TOP; la línea BOTTOM sólo aparece después de la última línea de texto. Un file recién creado está inicialmente vacío y sólo presenta en pantalla las líneas TOP y BOTTOM.
El cursor cambia de forma cuando se pasa al Estado de Inserción: el trazo debajo del carácter se convierte en un pequeño triángulo a la izquierda del carácter. En este caso, el cursor es intermitente.
Cuando se entra en el Vídeo File Editor, el teclado funciona de modo diferente, de manera que pueden activarse varias funciones de edición a través de una serie de teclas de función que se cargan automáticamente. En la pág. 618 se ha representado una imagen del teclado, y de la pág. 619 a la pág. 623 se han ilustrado las principales funciones del Vídeo File Editor.
La segunda línea de la pantalla (sobre la línea de escala) se llama **línea de los comandos** y se utiliza para la búsqueda de cadenas y para implantar comandos de alto nivel.
Para presentar caracteres en esta línea, el usuario debe implantar antes que nada la función COMMAND MODE (COMMAND + 1) que coloca el cursor al principio de esta línea.
Todas las operaciones de edición de línea como INSERT MODE, BACKSPACE, DELETE CHAR, RECALL LINE, etc. se aplican en este punto a la línea de los comandos. La función RECALL LINE, utilizada en Estado de Comandos, restituye el contenido anterior de la línea de los comandos. Todas las demás funciones pueden aplicarse al file de texto. En Estado de Comando, la tecla ↵ tiene la función de EXECUTE COMMAND y no la de INSERT LINE.
Para llevar el cursor al file de texto y reemprender las operaciones de edición en el file, es necesario implantar un nuevo COMMAND MODE.

Las funciones de edición y de búsqueda de las cadenas. De la pág. 619 a la pág. 623 se presenta el funcionamiento de los comandos de edición en el ambiente del Vídeo File Editor, con el efecto producido por las diversas funciones sobre un texto cualquiera.
La funcionalidad de **búsqueda de las cadenas** permite explorar el file para buscar una determinada secuencia de caracteres.
Para poder activar la búsqueda, el usuario debe implantar primero la función COMMAND MODE y después la cadena que se busca, que aparecerá en la segunda línea de la pantalla. Tras introducir la cadena, el usuario implantará la función SEARCH DOWN o SEARCH UP según desee explorar el file hacia adelante o hacia atrás con respecto a la posición en que se encuentra el cursor. Como se ha dicho, la entrada al Vídeo File Editor se realiza mediante el comando EDIT, a partir del ambiente PCOS.
Para volver a PCOS, el usuario debe implantar EXIT AND SAVE (CTRL + 6), que también registra en el disco el texto actualizado y, por tanto CR, o bien ABORT (COMMAND + 6) y CR.
En cambio, si el usuario implanta SAVE TEXT (CTRL + 5) el file de texto actualizado se registra en el disco, pero se permanece en el ambiente Vídeo File Editor.

Los menús

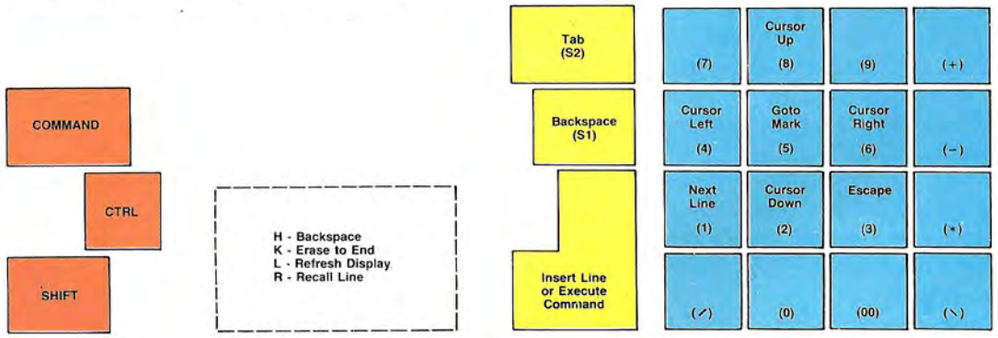
En cada procedimiento, incluso sencillo, puede identificarse un cierto número de bloques fun-

TECLAS DE FUNCION DEL VIDEO FILE EDITOR

Teclas activables pulsando al mismo tiempo la tecla COMMAND

Command Mode (1)	Delete Line (2)	Split Line (3)	Start of Line (4)	Restore Lines (5)	Abort (6)	Insert Mark (7)	Search Up (8)	Line Up (9)	Half Screen Up (0)	Full Screen Up (−)	Top (^)
Insert Mode (1)	Delete Char (2)	Join Lines (3)	End of Line (4)	Save Text (5)	Exit and Save (6)	Reverse Tab (7)	Search Down (8)	Line Down (9)	Half Screen Down (0)	Full Screen Down (−)	Bottom (^)

Teclas activables pulsando al mismo tiempo la tecla CTRL



Claves alfabéticas activables pulsando al mismo tiempo la tecla CTRL

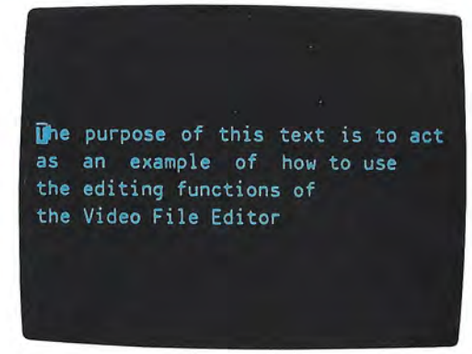
Teclas activables pulsando al mismo tiempo la tecla SHIFT

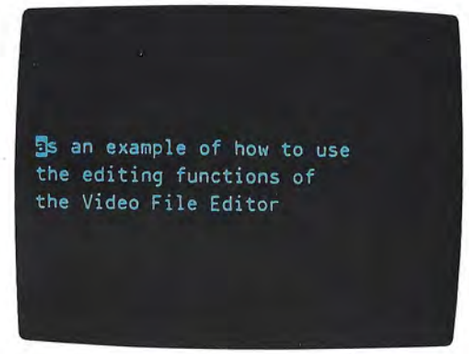
Los símbolos entre paréntesis son los rotulados en las teclas del Teclado USA-ASCII.

FUNCIONES DE EDICION Y DE BUSQUEDA DE LAS CADENAS DEL VIDEO FILE EDITOR OLIVETTI

En esta página y las siguientes se han presentado como ejemplo las funciones de edición y de búsqueda de las cadenas del Vídeo File Editor. Al lado se ha presentado la parte del monitor que contiene el file-texto a modificar. En lo sucesivo, en la columna de la izquierda se ilustra la función aplicada y, en la columna de la derecha, su efecto sobre el texto.

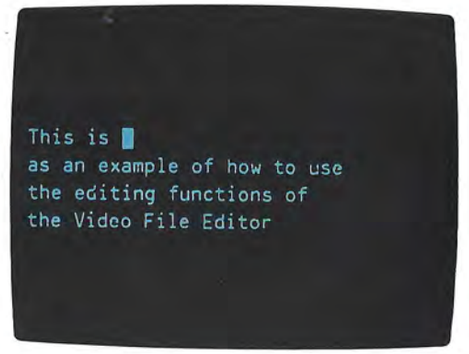
DELETE LINE
Cancela la línea indicada por el cursor





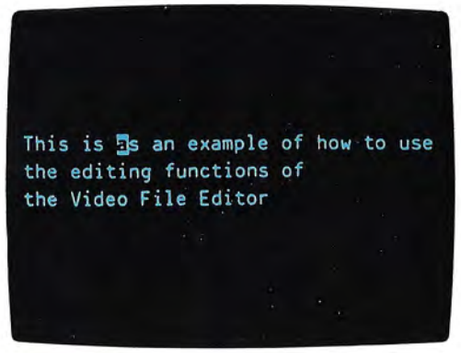
CURSOR UP
Coloca el cursor por encima de una línea

INSERT MODE This is
Inserta la cadena "This is" a partir del carácter indicado por el cursor



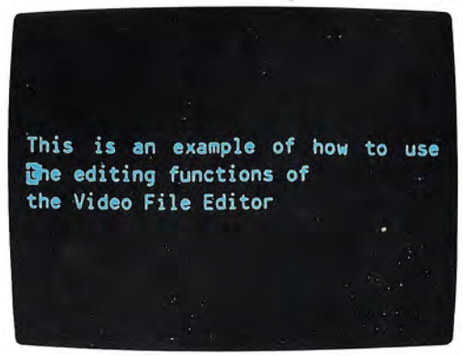
JOIN LINES
Une la línea indicada por el cursor
con la siguiente línea

CURSOR RIGHT
Desplaza el cursor una posición
hacia la derecha

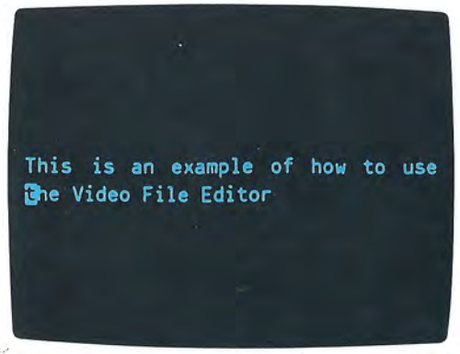


DELETE CHAR
Cancela el carácter que hay en la posición
del cursor y desplaza los restantes caracteres
de la línea una posición hacia la izquierda.
Aquí se ha utilizado tres veces

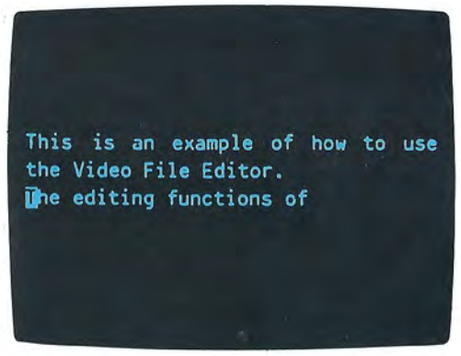
NEXT LINE
Desplaza el texto una línea hacia arriba
y el cursor al principio de la línea
siguiente a la que se encontraba inicialmente



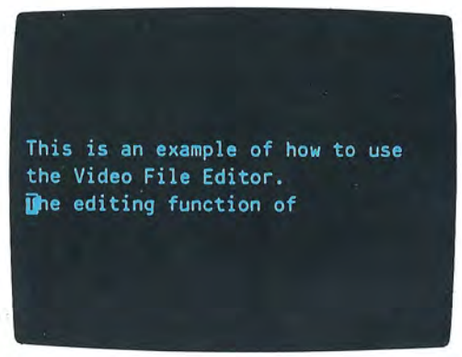
DELETE LINE
Cancela la línea indicada por el cursor



RESTORE LINES
NEXT LINE
Reclama en pantalla la siguiente
línea del file-texto



INSERT MODE
Pasa del estado de modificación al
estado de introducción



END OF LINE
Coloca el cursor en la posición que sigue
al último carácter de la línea que no sea
un espacio

BACKSPACE
Desplaza el cursor una posición hacia la izquierda y cancela el carácter en la posición del cursor. Aquí se ha utilizado dos veces

```
This is an example of how to use
the Video File Editor.
The editing functions
```

RECALL LINE
Repone el contenido anterior de la línea. El cursor permanece parado en el punto en que estaba colocado anteriormente

```
This is an example of how to use
the Video File Editor.
the editing functions of
```

SPLIT LINE
Envía al principio de la línea siguiente la parte de la línea a la izquierda del cursor (incluido el carácter de debajo del cursor)

```
This is an example of how to use
the Video File Editor.
the editing functions
of
```

CURSOR UP
Coloca el cursor por encima de una línea

```
This is an example of how to use
the Video File Editor
the editing functions
of
```


INSERT LINE
Inserta una línea vacía inmediatamente debajo de la que está el cursor y coloca este último al principio.

```
This is an example of how to use
the Video File Editor.

the editing functions
of
```

En los dos ejemplos que siguen se ilustran las funciones de búsqueda de cadenas. En el texto que aparece al lado, primero se buscará la cadena "func" y después la cadena "e of"

```
This is an example of how to use
the search function keys of the
Video File Editor to find a
particular combination of charac-
ters
```

Si se escribe func en la segunda línea de la pantalla y se aplica SEARCH DOWN, el Vídeo File Editor busca la cadena "func" a partir de la posición ocupada por el cursor avanzando hacia el final del texto.

```
This is an example of how to use
the search function keys of the
Video File Editor to find a
particular combination of charac-
ters
```

Si se escribe e of en la segunda línea de la pantalla y se aplica SEARCH UP, el Vídeo File Editor busca la cadena "e of" a partir de la posición ocupada por el cursor avanzando hacia el principio del texto.

```
This is an example of how to use
the search function keys of the
Video File Editor to find a
particular combination of charac-
ters
```

cionales que constituyen entidades residentes en el mismo. Los varios bloques actúan todos sobre los mismos datos y concurren a la solución del mismo problema, aunque cada uno trata un aspecto o una particularidad del conjunto.
La utilización de este procedimiento consiste en enviar a ejecución las diversas partes que lo constituyen, de acuerdo con la necesidad específica o debido a prioridades temporales establecidas por el programa. En el primer caso, la ejecución de una parte del procedimiento no está subordinada a la ejecución de las otras partes, y el usuario tiene libre acceso a la secuencia sin tener que respetar condiciones anteriores. En el segundo caso, necesidades particulares de elaboración exigen respetar precisas reglas de secuenciación; por tanto, la elección de las funciones debe ser guiada por el sistema.
En cada caso, el usuario debe tener a su disposición un medio para que pueda

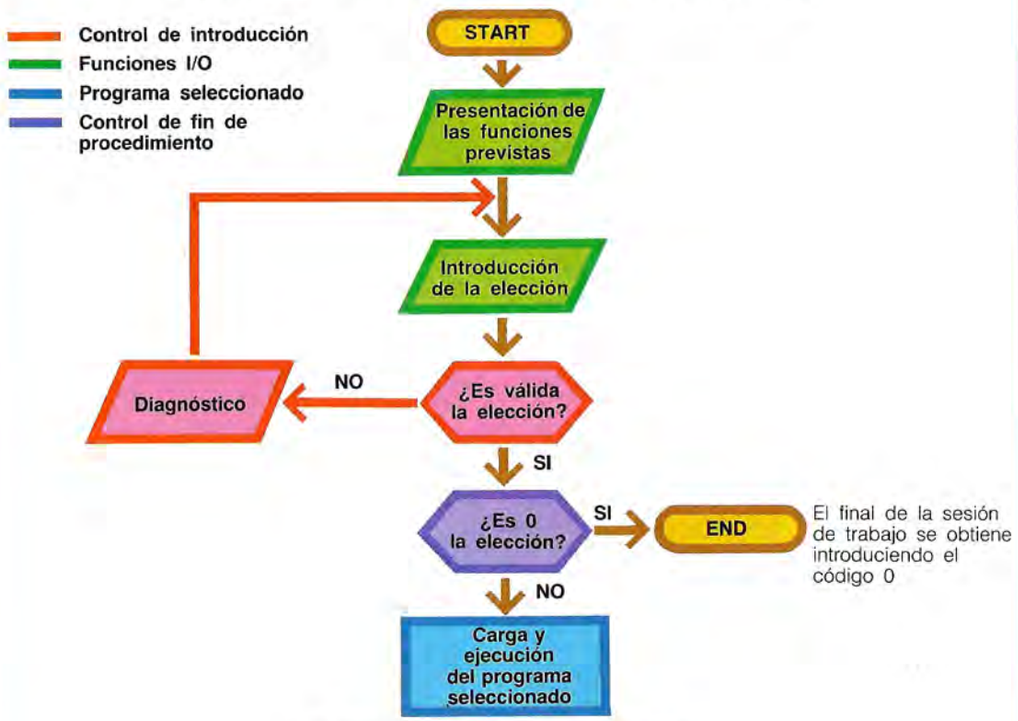
- presentar la lista de las funciones previstas en el procedimiento
- aceptar y controlar la elección del usuario
- presentar, bajo demanda, las principales características del propio procedimiento o de los programas que lo acompañan

Una estructuración similar también es útil para el programador, ya que facilita la división de todo el software en partes más pequeñas, simplificando la escritura y minimizando eventuales problemas de ocupación de memoria.
El software que cubre las necesidades mencionadas, y en consecuencia las funciones de guía para la elección que se realiza, normalmente se llama **menú**.
Estructuralmente, un menú es muy sencillo: consiste en la presentación de las posibles funciones y en la petición de introducción del código de la función deseada en aquel momento.
Según la respuesta, el programa de gestión del menú procede a cargar y a enviar a ejecución la oportuna parte del procedimiento.
Abajo aparece el diagrama de flujo de un menú genérico. En base a la elección hecha por el usuario se carga el oportuno programa, que ocupa también el área de memoria en la que estaba cargado el menú. Al final de las funciones previstas, a su vez el programa llamado debe recargar en la memoria el menú para permitir una elección sucesiva. La instrucción que permite realizar estas cargas es:

RUN "X:NOMBRE"

X indica la unidad de disco en la que reside el programa a llamar (A/B, 0/1); NOMBRE es el nombre con el que el programa que debe enviarse a ejecución se identifica en el disco.
Los menús pueden llamarse en cascada para multiplicar la posibilidad de elección y facilitar la búsqueda de la función deseada.
Por ejemplo, un menú principal puede llamar un cierto número de menús secundarios, cada uno dedicado a una cierta categoría de funciones; a su vez, los menús secundarios llaman los programas específicos. Al final de cada programa específico deberá reclamarse el menú secundario desde el que ha sido llamado y el menú secundario deberá reclamar el principal. De este modo se pueden realizar elecciones en los dos sentidos: del general subir al detallado y, viceversa, de la aplicación particular volver a salir a la general. En esta página y en la siguiente se ha indicado el listado de un menú principal que llama otros menús. Como esta rutina constituye el punto de entrada de todo el procedimiento que la utiliza, en ella deben haberse previsto las funciones comunes a todos los programas, como la inicialización de las teclas funcionales o la fijación de una forma particular del cursor. Obsérvese que, en esta etapa, las teclas funcionales sólo se inicializan, es decir, se les asocia un código numérico; la función que se desarrollará bajo este código se establece en cada programa sencillo. Por ejemplo, en el menú principal puede asignarse a una cierta tecla el código numérico 2: en un programa, este código podrá activar las funciones de impresión,

DIAGRAMA DE FLUJO DE UN MENU GENERICO



El final de la sesión de trabajo se obtiene introduciendo el código 0

El programa llamado al final debe volver a cargar el menú para la siguiente selección

MENU PRINCIPAL

```
4540 ' FILE = MENU
4541 '   ***** FILES *****
4542 '
4543 ' NOMBRE      UNID        CONTENIDO          BYTES     CAMPOS
4544 '
4545 ' DACLI         A       DATOS CLIENTES        128         7
4546 ' DAPRO         A       DATOS PROVEEDORES     128         7
4547 ' ALMSE         A       ALMACEN SEMANAL        91         9
4548 ' ALMAC         A       ACUMULADO ALMACEN     100        10
4549 ' FACTS         B       FACTURAS               70         9
4550 ' COBAC         B       COBROS ACUMULADOS      70         9
4551 ' PAGAC         B       PAGOS ACUMULADOS       70         9
4552 ' NOTAB         B       NOTAS DE ABONO         70         9
4553 ' ITE           A       I.T.E.                128         7
4554 ' ITEAB         A       I.T.E. ABONOS         128         7
4555 '
4556 LPRINT CHR$(27)+CHR$(62)     ' 220 CAR/PULGADA
4560 GOSUB 6040    ' HABILITACION TECLAS FUNCIONALES
4580 BI$=CHR$ (27)+"+"+CHR$(7)
4600 GOSUB 4960
4620 ON V GOTO 4640,4660,4680,4700,4720,4740,4760,4780,4800,4820
4640 RUN"A:MENU1"          ' ALMACEN
4660 RUN"A:MENU2"          ' DATOS CLIENTES Y PROVEEDORES
4680 RUN"A:MENU3"          ' COBROS
4700 RUN"A:MENU4"          ' FACTURACION
4720 RUN"A:MENU5"          ' CARGO MERCADERIAS
4740 RUN"A:MENU6"          ' NOTAS DE ABONO
4760 RUN"A:CACLI"          ' CARGO CLIENTES
4780 RUN"A:CAPRO"          ' CARGO PROVEEDORES
4800 RUN"A:HLP1"
4820 PRINT   BI$
4840 CK=5:X=30:Y=10:GOSUB 5700
4860 PRINT CHR$(27)+"G1"+" FIN TAREA"
4880 PRINT CHR$(27)+"G0"
4900 STOP
4920 END
```

```
4940 ' **********************
4960 PRINT BI$
4980 CK=5:X=2:Y=1:GOSUB 5700:PRINT"E.G.S. Contabilidad general"
5000 CK=5:X=15:Y=6:GOSUB 5700
5020 PRINT " 1 - ALMACEN"
5040 Y=Y+1:GOSUB 5700
5060 PRINT " 2 - DATOS CLIENTES Y PROVEEDORES" "
5080 Y=Y+1:GOSUB 5700
5100 PRINT " 3 - COBROS Y PAGOS"
5120 Y=Y+1:GOSUB 5700
5140 PRINT " 4 - FACTURACION"
5160 Y=Y+1:GOSUB 5700
5180 PRINT " 5 - CARGO MERCADERIAS"
5200 Y=Y+1:GOSUB 5700
5220 PRINT " 6 - NOTAS DE ABONO"
5240 Y=Y+1:GOSUB 5700
5260 PRINT " 7 - CARGO CLIENTES"
5280 Y=Y+1:GOSUB 5700
5300 PRINT " 8 - CARGO PROVEEDORES"
5320 Y=Y+1:GOSUB 5700
5340 PRINT " 9 - Explicaciones sobre el uso del Programa"
5360 Y=Y+1:GOSUB 5700
5380 PRINT " 0 - FIN TAREA"
5400 X=1:Y=Y+2:GOSUB 5700
5420 PRINT "INTRODUCIR OPCION ELEGIDA"
5440 X=26:GOSUB 5700:A$=INPUT$(1):PRINT A$
5460 V=VAL(A$)
5480 IF V>9 GOTO 5600
5500 PRINT BI$
5520 X=5:Y=6:GOSUB 5700
5540 PRINT "ESPERAR CARGA PROGRAMA"
5560 IF V=0 THEN V=10
5580 RETURN
5600 PRINT CHR$(7),CHR$(7)
5620 X=42:GOSUB 5700
5640 PRINT "NO VALIDO"
5660 GOTO 5440
5680 '
5700 '      DESPLAZAMIENTO CURSOR
5720 '     FILE CURS-X/Y
5740 ON CK GOTO 5760,5800,5880,5920,5960
5760 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
5780 RETURN
5800 PRINT CHR$(27)+"c"+"1"
5820 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
5840 PRINT CHR$(27)+"c"+"3"
5860 RETURN
5880 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
5900 RETURN
5920 PRINT CHR$(27)+CHR$(61)+CHR$(L+31)+CHR$(71);
5940 RETURN
5960 PRINT CHR$(27)+CHR$(61)+CHR$(Y+31)+CHR$(X+31);
5980 RETURN
6000 ' ****
6020 ' ****
6040       TECLAS FUNCIONALES
6060 '
6080 '       FILE : TASTI
6100 PRINT CHR$(27)+"c4AA"        FORMA DEL CURSOR
6120 A$=CHR$(27)+CHR$(46)
6140 B=98:I1=49:I2=48
6160 FOR I=1 TO 8
6180 PRINT A$+CHR$(B)+CHR$(I1)+CHR$(I2);
6200 K=(I1-48)*16+(I2-48):N=B-97
6220 B=B+1:I2=I2+1
6240 NEXT I
6260 B=106:I1=48:I2=52
6280 FOR I=1 TO 4
6300 PRINT A$+CHR$(B)+CHR$(I1)+CHR$(I2);
6320 K=(I1-48)*16+(I2-48):N=B-97
6340 B=B+1:I2=I2+1
6360 NEXT I
6380 RETURN
```


en otro podrá servir como clave de salida de la elaboración, etc. La lista indicada contiene las instrucciones necesarias para la gestión del cursor y de las teclas funcionales de una determinada máquina; si se utiliza un sistema diferente debe variarse tales instrucciones, dado que, tal como se ha dicho, los códigos utilizables son específicos de cada máquina particular.

Para facilitar el desarrollo del software presentado, las funciones particulares ligadas estrictamente al hardware se describen aparte en subrutinas adecuadas.

Refiriéndonos siempre al listado de la página 625 (menú principal) se observa una parte inicial que describe los files de datos utilizados en el procedimiento (líneas 4545 ÷ 4555). En ella se indican los nombres de los files, la unidad de disco sobre la que deben montarse y los discos que la contienen (columna DVR), la descripción del contenido, la longitud del record (en bytes) y el número de campos en que está dividido el record. Esta parte tiene el único objeto de documentar el programa y resulta muy útil, como memoria previa, cuando debe modificarse.

La línea 4560 llama la subrutina 6040 que inicializa las teclas funcionales y fija la forma del cursor. La cadena BI$ (línea 4580) contiene, además, el carácter Escape [CHR$(27)], los caracteres + y CHR$(7). El primero, para la particular unidad de vídeo utilizada, tiene la función de anular el contenido de toda la pantalla, el segundo provoca la emisión de un sonido con el fin de reclamar la atención del operador. Este sonido se llama «beep» y en algunas máquinas está controlado por una instrucción adecuada (BEEP). Obsérvese que el código CHR$(7) para obtener el beep está en las tablas ASCII y, por tanto, está presente en casi todas las máquinas.

La presentación de las diversas descripciones se ha hecho en la subrutina 4960, que se inicia cancelando el vídeo (PRINT BI$) y presentando las líneas de cabecera 4980.

La posición de las diversas líneas escritas en la pantalla es parametrizada.

Observando la línea 4980 puede verse cómo se introducen tres parámetros (CK, X y Y); el parámetro CK no se utiliza y está predispuesto para las implantaciones futuras. El valor de X expresa la columna en la que debe iniciarse la escritura, el valor de Y la línea. Para posicionar la escritura a partir de las coordenadas X, Y (columna, línea) debe situarse el cursor en esta posición antes de emitir la escritura. El posicionado se

El ordenador personal puede simplificar notablemente la gestión de un almacén

Olivetti

MENU DEL ALMACEN

```
4500 '
4505 ' ** MENU DEL ALMACEN **
4510 '   FILE : MENU1
4512 '
4514 '  VERS. 2/12/83
4515 'LAS TECLAS FUNCIONALES SON HABILITADAS POR EL MENU PRINCIPAL - FILE :MENU
4520 BI$=CHR$(27)+"+"+CHR$(7)
4525 GOSUB 4615
4530 ON V GOTO 4535,4540,4545,4550,4555,4560,4565,4570,4575,4580
4535 RUN"A:GENALM"       ' GENERACION DENOMINACIONES
4540 RUN"A:MOVSEM"       ' MOVIMIENTOS SEMANALES
4545 RUN"A:PRISEM"       ' IMPRIME MOV. SEM.
4550 RUN"A:TRAAR"        ' TRANSFERENCIA AL ARCHIVO
4555 RUN"A:PRIARC"       ' IMPRIME ARCHIVO
4560 GOTO 4510
4565 GOTO 4510
4570 GOTO 4510
4575 RUN"A:HLP2"
4580 PRINT BI$
4585 CK=5:X=30:Y=10:GOSUB 4800
4590 PRINT CHR$(27)+"G1"
4595 PRINT CHR$(27)+"G0"
4600 RUN"A:MENU"
4605 END
4610 '**********************
4615 PRINT BI$
4620 CK=5:X=2:Y=1:GOSUB 4800:PRINT"E.G.S. Gestión Almacén"
4625 CK=5:X=15:Y=6:GOSUB 4800
4630 PRINT "1- Generación y Variación denominaciones de ALMACEN"
4635 Y=Y+1:GOSUB 4800
4640 PRINT "2- Movimientos Cargo/Descargo SEMANALES"
4645 Y=Y+1:GOSUB 4800
4650 PRINT "3- Imprime Movimientos SEMANALES con VALORACION"
4655 Y=Y+1:GOSUB 4800
4660 PRINT "4- Transferencia Movimientos SEMANALES al ARCHIVO ACUMULADO"
4665 Y=Y+1:GOSUB 4800
4670 PRINT "5- Imprime con VALORACION del ARCHIVO ACUMULADO"
4675 Y=Y+1:GOSUB 4800
4680 PRINT "6- ..............."
4685 Y=Y+1:GOSUB 4800
4690 PRINT "7- ..............."
4695 Y=Y+1:GOSUB 4800
4700 PRINT "8- ..............."
4705 Y=Y+1:GOSUB 4800
4710 PRINT "9- Explicaciones sobre el uso del Programa"
4715 Y=Y+1:GOSUB 4800
4720 PRINT "0-FIN TAREA"
4725 X=1:Y=Y+2:GOSUB 4800
4730 PRINT "INTRODUCIR OPCION ELEGIDA"
4735 X=26:GOSUB 4800:A$=INPUT$(1):PRINT A$
4740 V=VAL(A$)
4745 IF V>9 GOTO 4775
4750 PRINT BI$
4755 X=5:Y=6:GOSUB 4800
4760 PRINT "ESPERAR CARGA PROGRAMA"
4765 IF V=0 THEN V=10
4770 RETURN
4775 PRINT CHR$(7),CHR$(7)
4780 X=42:GOSUB 4800
4785 PRINT "NO VALIDO"
4790 GOTO 4735
4795 '
4800 '      DESPLAZAMIENTO CURSOR
4805 '     FILE CURS-X/Y
4810 ON CK GOTO 4815,4825,4845,4855,4865
4815 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
4820 RETURN
4825 PRINT CHR$(27)+"c"+"1"
4830 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
4835 PRINT CHR$(27)+"c"+"3"
```



```
4840 RETURN
4845 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
4850 RETURN
4855 PRINT CHR$(27)+CHR$(61)+CHR$(L+31)+CHR$(71);
4860 RETURN
4865 PRINT CHR$(27)+CHR$(61)+CHR$(Y+31)+CHR$(X+31);
4870 RETURN
4875 ' ****
4880 ' ****
```

Menú principal: según la denominación seleccionada, se carga y presenta uno de los menús secundarios.

Se ha seleccionado la denominación 1 (ALMACEN). La siguiente selección permite el acceso a los procedimientos.

obtiene en la subrutina 5700; la escritura se emite con la instrucción PRINT que sigue a la llamada de la subrutina de posicionado del cursor.
El posicionado del cursor se obtiene simplemente enviando a impresión en el monitor la cadena de comando (línea 5760) terminada con el símbolo ; que sirve para impedir el retorno al principio (se verán los formatos de impresión).
En la línea 5440 hay la instrucción A$ = INPUT$(1) que permite la lectura de un solo carácter (la elección puede hacerse sólo entre 0 y 9); el carácter obtenido así se transforma en numérico (línea 5460) y se controla (línea 5480). Si el valor queda comprendido entre los previstos, se realiza la cancelación del vídeo (línea 5500) y se informa al usuario que debe esperar (línea 5540) ; el programa vuelve a realizar las instrucciones del main que siguen a la llamada a esta subrutina (línea 4620).
En este punto se realiza la selección del programa que hay que cargar de acuerdo con la elección efectuada por el operador.
Por ejemplo, si la respuesta es 1 (V = 1) se carga y se envía a ejecución el programa MENU1.
Este programa (cuyo listado se ha indicado al lado y en la parte superior) permite realizar elecciones posteriores (líneas 4530 ÷ 4555) y, al final, reclama el anterior (línea 4600); por tanto, se trata de un menú secundario.
En éste no hay presentes las instrucciones para obtener una particular forma del cursor, ni la inicialización de las teclas funcionales (operaciones ya realizadas en el menú principal), mientras que siempre debe haber la subrutina de posicionado del cursor (líneas 4800 ÷ 4840).
Obsérvese que los dos menús (principal y secundario) tienen la misma numeración, ya que el uno recubre el otro y no hay por qué utilizar numeraciones diferentes. Por el contrario, como todos los menús parten del mismo número de línea, se facilita la búsqueda de los errores y la realización de variaciones en el programa.

Generación de histogramas

Utilizando las funciones de direccionamiento del cursor también pueden construirse histogramas no gráficos en la pantalla de vídeo. El procedimiento consiste en escribir en el vídeo una serie de símbolos en números proporcionales (según un factor de escala prefijado) a la magnitud a representar. Por ejemplo, para representar el valor 150 puede elegirse un factor de escala 10, y el número 150 será representado por una columna de 15 símbolos (150/10).

Para obtener la representación en el vídeo deberá escribirse 15 veces el mismo símbolo, variando cada vez en una unidad el número de línea y manteniendo fijo el número de columna. De este modo se obtiene una pequeña columna de altura proporcional al valor numérico que hay que representar.

Aquí debajo se indica el esquema de principio de la lógica de representación.

El posicionado del cursor puede obtenerse definiendo una función que contenga los códigos de direccionamiento. Por ejemplo, la función

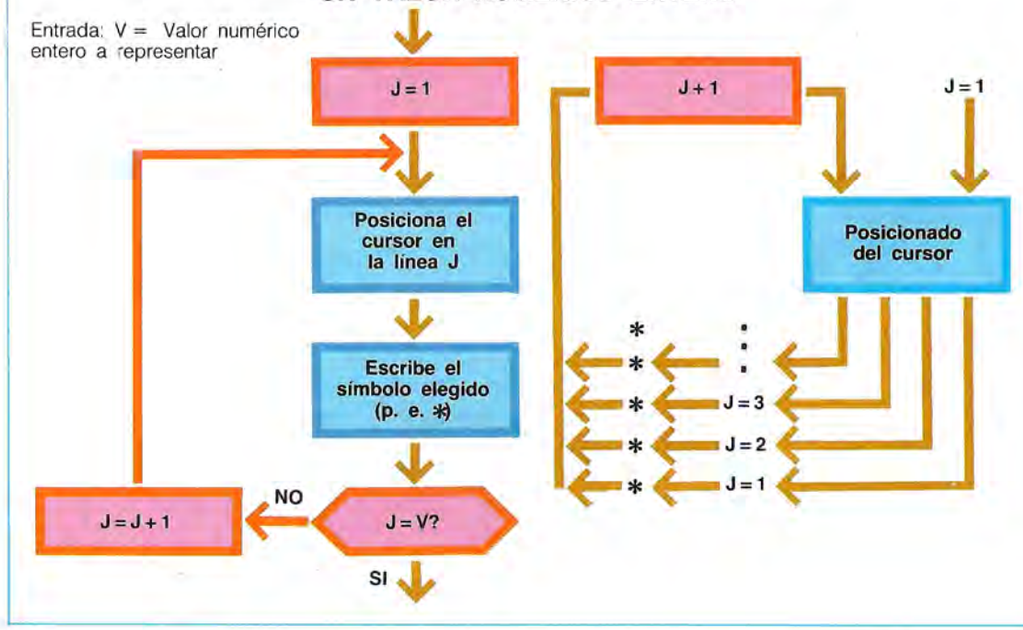
```
FNP$(A,B) = CHR$(27) + CHR$(61) +
            CHR$(31 + B) + CHR$(31 + A)
```

posiciona el cursor en la línea B, columna A. Como la pantalla de vídeo posee normalmente 80 columnas y 24 líneas, los valores máximos que pueden obtenerse son A = 80 y B = 24, mientras que los posicionados mínimos son para el eje X (indicado con I) de un carácter y para el eje Y (indicado con J) de una línea.

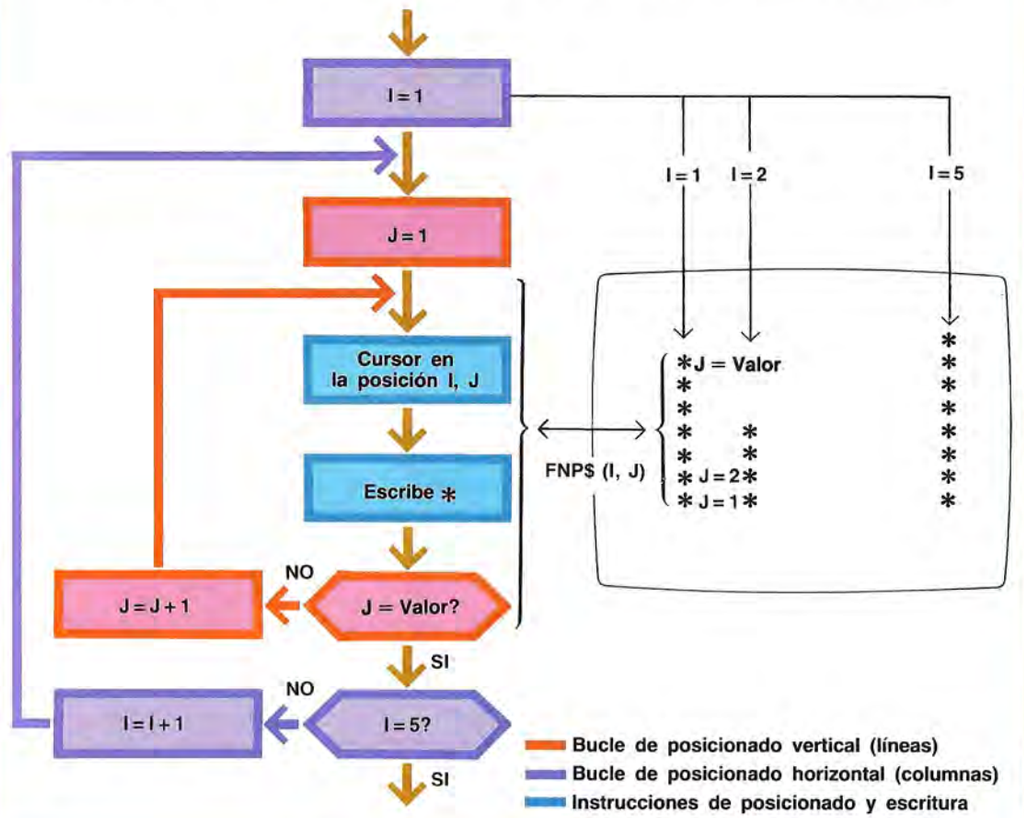
En otras palabras, no pueden obtenerse gráficos con símbolos más juntos que la distancia que existe ya sea entre dos líneas (eje Y) o ya sea entre dos caracteres (eje X), porque **para un vídeo no gráfico, la posibilidad de direccionamiento es para números enteros de línea y de columna (caracteres); no existen posiciones intermedias.**

Refiriéndonos al esquema de la pág. 631, la columna del histograma que representa el valor 150 se obtendrá direccionando el cursor sobre 15 líneas sucesivas (1, 2, 3, etc.) y escribiendo el símbolo elegido, mientras que el valor de la columna (I) permanece invariable. Para representar un nuevo valor al lado del anterior debe variarse la abscisa (columna I) y repetir el bucle en la línea (J). Uniendo los dos bucles, es decir insertando los dos bucles uno dentro de otro, se pueden obtener histogramas que comprendan hasta 80 columnas una al lado de otra, y que corresponderán a una para cada columna de la pantalla del vídeo.

LOGICA DE REPRESENTACION GRAFICA DE UN VALOR NUMERICO ENTERO



LOGICA DE REPRESENTACION GRAFICA SOBRE MAS COLUMNAS

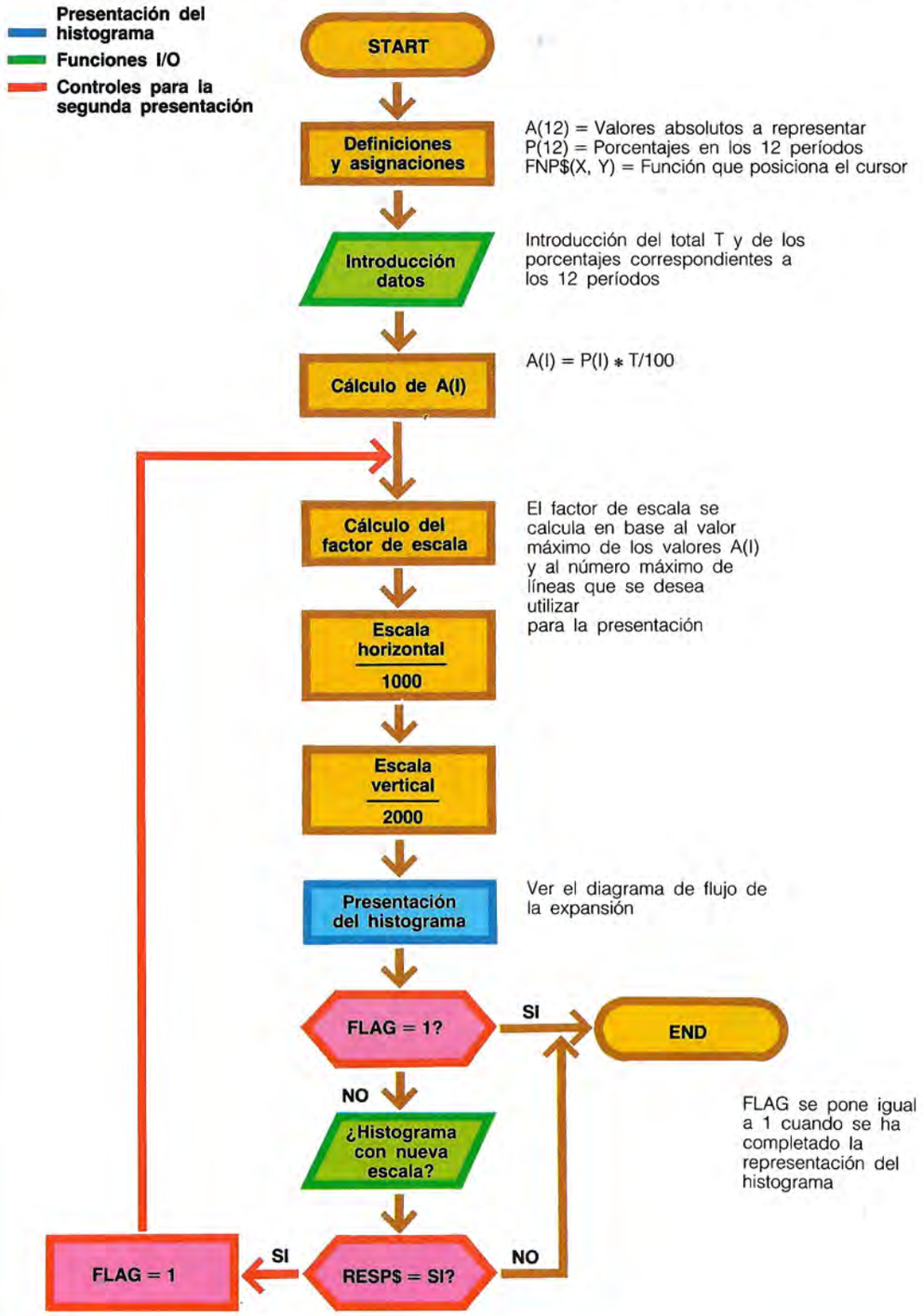


En el esquema se han indicado las coordenadas X, Y con los símbolos I, J para distinguir este caso particular de **direccionamiento discreto** del caso general de **direccionamiento por coordenadas**.

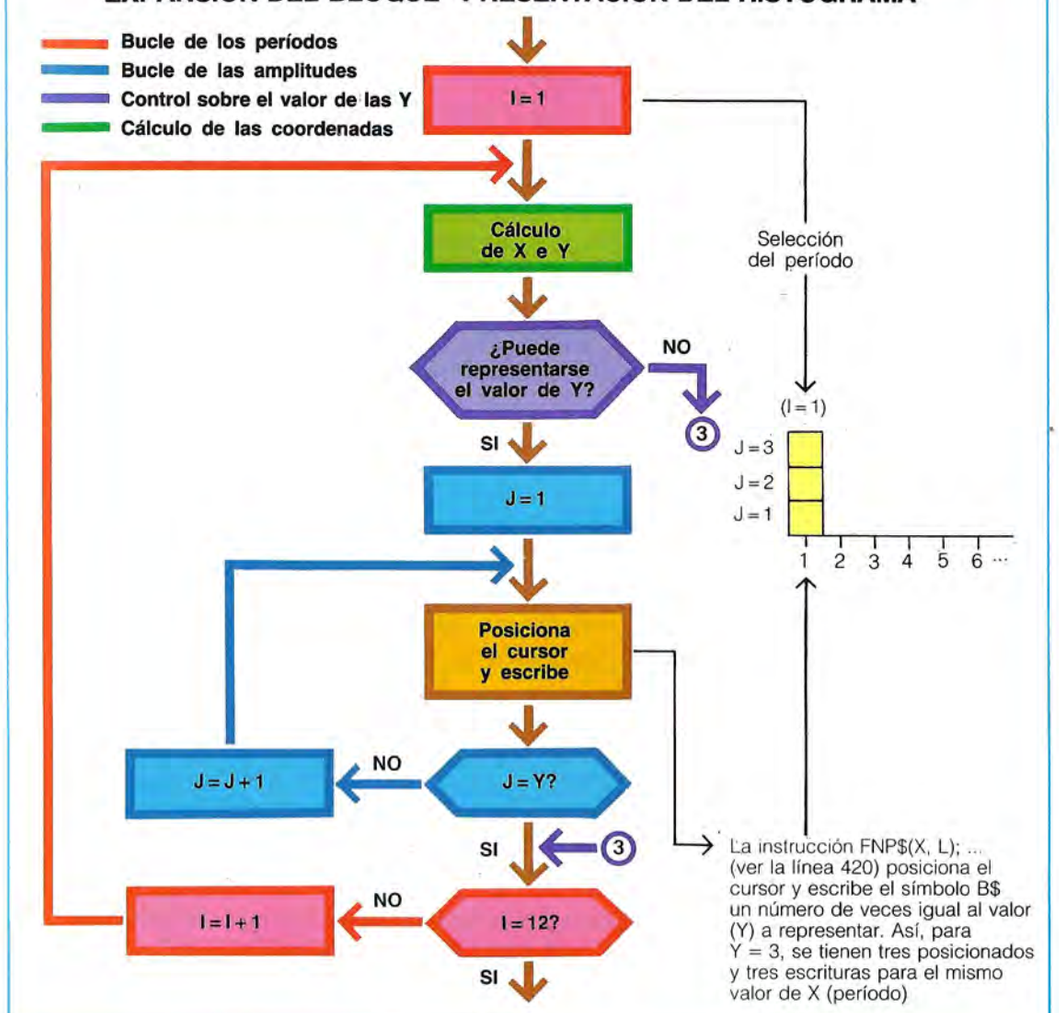
En las pantallas no gráficas, los valores que pueden asumir las coordenadas sólo son números enteros, ya que representan filas y columnas efectivas de la pantalla; en otros términos, no se puede posicionar el cursor en el punto 2.75 (líneas): o se elige la posición (línea) 2 o la posición (línea) 3. En este caso, el direccionamiento se llama discreto (es decir, limitado a un cierto número de valores).

En cambio, en las pantallas gráficas ya no se está vinculado a las líneas y a las columnas; se puede hacer referencia a los **puntos de pantalla**, que pueden ser definidos con un par de coordenadas parecidas a los puntos de un papel milimetrado. Tampoco en este caso pueden

PROGRAMA PARA LA GENERACION DE HISTOGRAMAS



EXPANSION DEL BLOQUE "PRESENTACION DEL HISTOGRAMA"



representarse todos los posibles valores de X y de Y, ya que el número de puntos de la pantalla, aunque muy grande, es limitado. En la página 631 se ha representado el esquema lógico para la preparación de un histograma que representa (uno al lado del otro) cinco valores numéricos.
El primer bucle, el más exterior, selecciona la columna del vídeo sobre la que se construirá la representación gráfica del valor. El bucle más interno determina la altura de cada una de las cinco columnitas, seleccionando un número de líneas proporcional al valor a representar y escribiendo para cada línea el símbolo elegido (en el ejemplo, el símbolo *).
Al lado y arriba se ha indicado el diagrama de flujo de un programa para la preparación de histogramas con doce valores numéricos. El ejemplo es generalizado y puede aplicarse a casos prácticos para la representación gráfica de la distribución en 12 períodos de cualquier magnitud. Puede utilizarse, por ejemplo, para representar previsiones de gastos (presupuestos), cargas de trabajo, etc.; el número de períodos (12) permite implantar una representación mensual en el marco de un año. Los datos que debe proporcionar el programa indicado en las páginas 634 y 635 son el total (cantidad gastada, horas de trabajo, etc.) y los porcentajes previstos en cada uno de los períodos. El programa calcula el valor absoluto correspondiente a cada

EJEMPLO DE PREPARACION DE HISTOGRAMAS

```
30 '      FILE : HIST
40 DIM A(12), P(12)
50 DEF FNP$(X,Y)=CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X)
60 C$=CHR$(27)+"+"        ' CLEAR VIDEO
70 B$=" "              ' SIMBOLO USADO PARA EL HISTOGRAMA
80 FI$=CHR$(27)+"G4"    ' FORMATO (VIDEO INVERTIDO)
90 FV$=CHR$(27)+"G0"    ' FORMATO (VIDEO NORMAL)
100 INPUT" CIFRA TOTAL ";T
110 PRINT " INTRODUCIR EL PORCENTAJE POR CADA PERIODO"
120 FOR I=1 TO 12
130 PRINT " Período ";I
140 INPUT " Porcentaje ";P(I)
150 NEXT I
160 SO=5     ' FACTOR DE ESCALA HORIZONTAL 1 PERIODO = 5 POSICIONES
170 ' * CALCULO DE LAS CANTIDADES EN CADA PERIODO SENCILLO
180 FOR I=1 TO 12
190 A(I)=P(I)*T/100
200 NEXT I
205 PRINT C$    ' CLEAR VIDEO
210 '
220 ' ** BUSQUEDA DEL MAXIMO ENTRE LOS VALORES A (I)
230 '
240 MX=0
250 FOR I=1 TO 12
260 IF A(I) >=MX THEN MX=A(I)
270 NEXT I
272 IF FLAG=1 GOTO 276    ' SALTA LA ELECCION DE ESCALA SI ES EL SEGUNDO GIRO
275 GOSUB 3000   '  ELECCION DEL TIPO DE ESCALA VERTICAL
276 PRINT C$      ' CLEAR VIDEO
280 '
290 ' ** FACTOR DE ESCALA VERTICAL = NUMERO DE FILAS/MAXIMO
300 '
310 SV=20/MX
320 '
325 GOSUB 1000          ' ESCALA HORIZONTAL
327 GOSUB 2000          ' ESCALA VERTICAL
330 '  *** BUCLE DE PRESENTACION HISTOGRAMA ***
350 FOR I=1 TO 12
360 X=I*SO
370 Y=A(I)*SV
375 IF Y<2 GOTO 460    ' SALTA LA ESCRITURA SI EL VALOR NO PUEDE OBTENERSE
380 FOR J=1 TO Y
390 L=22-J    ' *****     LA POSICION Y=1 ESTA ARRIBA A LA IZQUIERDA
400       .   '           PARA INICIAR DESDE ABAJO DEBE POSICIONARSE EN 22-J
410           '           (22 SON LAS LINEAS DE LA PANTALLA UTILIZADA)
420 PRINT FNP$(X,L);FI$;B$;FV$
430 NEXT J
450 X1=X-1:PRINT FNP$(X1,L);P(I)
460 NEXT I
470 X=40:Y=1:PRINT FNP$(X,Y) ;    ' POSICIONA EL CURSOR EN 40,1
475 PRINT" PARA CONTINUAR, INTRODUCIR UN CARACTER":S$=INPUT$(1)
480 PRINT C$
485 IF FLAG=1 GOTO 510    ' SI SE HA COMPLETADO LA SEGUNDA VUELTA TERMINA
490 INPUT" Si quiere una salida con escala relativa (SI/NO)";RESP$
500 IF RESP$="SI" THEN FLAG=1:GOTO 240
510 END
1000 ' ** ESCRITURA ESCALA HORIZONTAL 12 PERIODOS
1005 NL=SO*12          ' LONGITUD ESCALA HORIZONTAL
1006 G$=STRING$(NL,"-")
1007 X=1:Y=22:PRINT FNP$(X,Y) ;G$
1010 FOR I=1 TO 12
1020 X=I*SO-1
1030 Y=23
1040 PRINT FNP$(X,Y);I;
1050 NEXT I
1055 PRINT              ' RESET DEL CURSOR
1060 RETURN
2000 '   **   ESCRITURA ESCALA VERTICAL
2005 X=1
2010 FOR I=1 TO 20      ' 20 = MAXIMA ALTURA HISTOGRAMA
2020 L=22-I
2030 PRINT FNP$(X,L);"I"
2040 NEXT I
2045 PRINT FNP$(X,1);"Total = ";T
2050 RETURN
3000 ' **  ESCALA VERTICAL  **
3010 '
3020 '
3030 INPUT "Se quiere como valor de escala el máximo relativo (SI/NO)";RESP$
3040 IF RESP$="SI" THEN RETURN
3050 PRINT " Se ha elegido la escala vertical igual al 100% "
3060 INPUT " CONTINUA (SI/NO)";RESP$
3070 IF RESP$="SI" THEN MX=T:RETURN
3080 GOTO 3030
```

período (líneas 170 ÷ 200), busca, además, entre estos valores el máximo (líneas 240 ÷ 270) y, en base a esto, determina el factor de escala vertical. En el programa se han previsto tres subrutinas. La 3000 permite (a título de demostración, a fin de poner en evidencia el efecto del cambio de escala) la elección entre dos tipos de representación.

En el primer caso, el eje Y (vertical) se utiliza enteramente para representar el valor máximo; por tanto se tiene una escala expandida. En el segundo caso se utiliza una escala vertical que llena con el valor máximo solamente una parte de la altura de la columna.

Por ejemplo, suponiendo que entre todos los valores de los 12 períodos el valor máximo sea igual a 10 (el 20% del total), en la primera escala se representará una columnita vertical con la altura de todo el eje Y, y los demás valores serán representados proporcionalmente; el valor absoluto 5 se representará con media altura. En el segundo caso (escala en porcentajes) el valor 10 se representará con una altura igual a 2/10 de la columna correspondiente (2/10 representa el 20%) y el valor absoluto 5 con 1/10 de la columna. En el programa se ha insertado un flag (FLAG) que permite la presentación sucesiva en pantalla de los dos tipos de representación partiendo de la porcentual.

Las líneas 470 y 475 constituyen un ejemplo típico de mensajes con espera de respuesta. Para proseguir la elaboración al término de una representación, debe esperarse a que el usuario haya leído los resultados indicados en la pantalla. La práctica más generalizada consiste en reservar dos o tres líneas para el usuario con el fin de presentar en una de ellas el mensaje de espera (línea 475). La espera se realiza con la instrucción S$ = INPUT$(1), la cual suspende la ejecución del programa hasta que no se introduce un carácter cualquiera. La instrucción suprime el eco, por lo que el carácter no aparece en pantalla, y la cadena S$ únicamente sirve para realizar la espera. El programa presentado utiliza como símbolo para los histogramas un espacio invertido (aparece un rectángulo luminoso); el símbolo está contenido en la variable B$ (línea 70) y para sustituirlo es suficiente con cambiar la asignación.

El posicionado del cursor y la escritura del símbolo se obtienen con la instrucción 420. En ella, la FNP$(X,L) posiciona el cursor en la columna X, línea L; la línea viene dada por la cantidad 22-Y, para obtener el inicio del histograma a partir de la línea 22, en la parte baja de la pantalla. La variable FI$ contiene los códigos de comando para la inversión del vídeo (línea 80); B$ es el símbolo a escribir, mientras que FV$ contiene los códigos que devuelven el vídeo a la escritura normal.

Al final de la columnita que representa un cierto valor hay escrito el correspondiente porcentaje P(I) (línea 450). La escritura de este valor se obtiene volviendo a posicionar el cursor en la línea L de salida del bucle, pero desplazado una columna hacia la izquierda (X1 = X-1) para tener la primera cifra del número centrada con respecto a la columna del histograma.

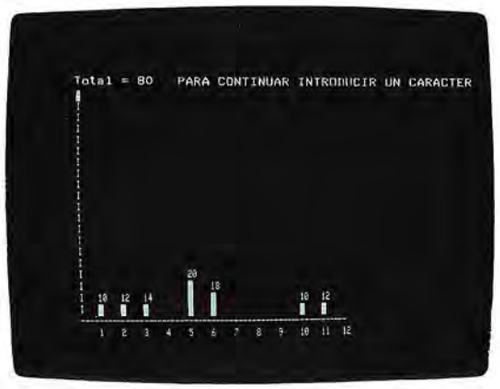
Los códigos indicados se refieren a una máquina específica; para otros modelos a utilizar deben sustituirse simplemente las líneas 50, 60, 80 y 90 por los oportunos valores indicados en el manual del usuario.

Las subrutinas 1000 y 2000 sirven para trazar las escalas horizontal y vertical. En la línea 1006 se utiliza la instrucción STRING$(NL, "–") para crear la cadena G$ de longitud NL que contiene todos los símbolos –; la escritura de esta cadena produce una línea de trazos horizontal. En la escala horizontal además se han indicado los valores de los intervalos (de 1 a 12) con las instrucciones 1010 ÷ 1050.

GENERACION DE HISTOGRAMAS EN LA PANTALLA

En esta página se han representado las imágenes de vídeo de la fase de introducción de datos (al lado) y de presentación de los histogramas (abajo) referidos al programa indicado en la página 634. La fase de introducción está guiada por el propio programa, que a cada vuelta pide los datos necesarios (líneas 100 ÷ 150). Después de la fase de cálculo, el control pasa a la subrutina 3000, que establece la escala vertical a adoptar.

En el primer caso se ha elegido la escala relativa: cada dato numérico está representado por una columnita de altura igual al porcentaje de la máxima altura posible indicada por el mismo dato.



En el segundo caso se ha elegido la plena escala: esta vez, el valor máximo introducido está representado con una columnita de altura igual a la máxima disponible. Los otros datos están representados proporcionalmente.

TEST 18

1 / ¿Qué son los atributos de un carácter?

2 / ¿Cuáles de las siguientes instrucciones son erróneas?

```
10 PRINT A,B,C
20 FOR I = 1 TO 20 : PRINT "PRUEBA" ; : NEXT I
30 PRINT A, TAB(60),B,TAB(100),C
```

3 / ¿Cuál es el efecto de las siguientes instrucciones?

```
10 FOR I =1 TO 30
20 PRINT I, "DATOS",A$
30 NEXT I
```

4 / Suponiendo que la función FNP$(X,Y) posicione el cursor en la columna X y la línea Y, ¿cuáles son las instrucciones que permiten escribir en un punto cualquiera de la pantalla una cadena cualquiera?
Las coordenadas (línea-columna) y la cadena deben introducirse por teclado y el programa debe terminar introduciendo columna = 0 o bien línea = 0.

5 / Suponiendo que la cadena INV$ contenga los códigos para la generación de escritura con vídeo invertido y que NOR$ contenga los códigos de funcionamiento normal, ¿cómo debe modificarse la instrucción de escritura del punto anterior para obtener la impresión de la cadena invertida?

Las soluciones, en las págs. 644-645.

Gestión de los archivos

Los files reconocidos del Basic son de dos tipos: SECUENCIALES y DIRECTOS (RANDOM). En los files secuenciales, los datos se escriben (y leen) uno a continuación del otro (no se puede acceder a un dato sin haber pasado antes por todos los anteriores). Un uso típico de estos files es el apoyo para datos en la entrada por teclado. En cambio, los files directos permiten la lectura y la escritura de los datos direccionando directamente el record que interesa y sólo aquél. La lógica de acceso a los datos de un file es la misma en ambos casos, y comprende las siguientes secuencias de funciones:

- Apertura del file
- Lectura y/o escritura de los datos
- Cierre del file (al término de su utilización)

Cada función tiene una sintaxis diferente según si se trata de files secuenciales o directos. Por tanto, se examinarán separadamente haciendo referencia a los dos casos indicados.

Funciones de acceso a los files secuenciales

Un file secuencial puede ser de dos tipos: **de salida**, indicado con las letras "O" (Output) o **de entrada**, indicado con la letra "I" (Input). El tipo de file (O/I) debe especificarse en la instrucción de apertura.
Esta instrucción, que debe introducirse antes de intentar el acceso a los datos, tiene la forma

OPEN "TIPO",N,"NOMBRE"

donde
TIPO puede asumir uno de los valores O/I
N es un valor numérico asociado al file,

con el cual este último deberá indicarse en las sucesivas operaciones de acceso a los datos

NOMBRE es el nombre con el que se ha creado el file

Por ejemplo, la línea

```
OPEN "I",2,"A:PRUEBA"
```

abre el file de nombre PRUEBA residente en la primera unidad de disco (A:) en la modalidad input (I) y le asocia el número 2.
En las operaciones de lectura de datos este file deberá indicarse con el número 2.
En algunos sistemas, el número del file debe ir precedido por el símbolo #. En este caso, la instrucción de apertura se modifica como sigue:

```
OPEN "I", #2, "A:PRUEBA"
```

En lo sucesivo, el símbolo # no se empleará en las instrucciones OPEN pero podrá ser obligatorio, según la versión del Basic.
La apertura del mismo file en modalidad output se obtiene con la línea

```
OPEN "O",2,"A:PRUEBA"
```

La modalidad con la que debe abrirse el file depende de las sucesivas funciones que se desea realizar en él. Para adquirir datos del file debe abrirse en modalidad input; de hecho se trata de una introducción de datos en el programa. Por el contrario, para escribir en el file debe abrirse en modalidad output ya que el file, visto desde el programa, es equivalente a un dispositivo de salida.
Al final de las operaciones de acceso a los datos, el file debe cerrarse. La instrucción que debe introducirse es

```
CLOSE N
```

donde N es el mismo valor numérico asignado al file en la fase de apertura. Por ejemplo, para cerrar el file anterior, la instrucción que debe introducirse es CLOSE 2.
Entre las instrucciones OPEN y CLOSE pueden utilizarse todas las funciones previstas para el acceso a los datos o para el control del estado del file. Las instrucciones utilizables con files secuenciales son las siguientes:

```
INPUT
LINE INPUT
WRITE
PRINT
PRINT USING
EOF
LOC
KILL
NAME
```

A continuación se describe y comenta cada una de estas instrucciones.

INPUT. Esta función, que tiene la sintaxis

```
INPUT # N, VARIABLE
```

adquiere los datos del file N y los transfiere a la variable especificada. Introduciendo la línea

```
INPUT # 1,B$
```

el contenido de un record del file 1 (el record «en curso»: se recuerda que, en los files secuenciales, la lectura y la escritura empiezan siempre por el record 1 y prosiguen en serie) se lee y se transfiere a B$. En esta instrucción, como en las instrucciones de acceso a los datos, el símbolo # no puede omitirse.

LINE INPUT. Su sintaxis completa es

```
LINE INPUT # N, VARIABLE
```

Adquiere una línea entera de caracteres (hasta 254) como la instrucción análoga para el teclado. Una utilización particular de esta instrucción se tiene en la lectura de un programa memorizado en disco en forma ASCII como parte de otro programa. El segundo programa ve el primero (el del disco) como un file de datos normal, y por tanto puede leerlo. El único obstáculo es la longitud variable del record (lógico). Cada línea de un programa memorizado en ASCII contiene un número de caracteres que depende de la instrucción que hay escrita; el único modo para reconocer el punto en que termina la línea es leer secuencialmente los records para encontrar el código CR (Carriage Return) que es el que termina la línea Basic.

WRITE. La función WRITE, que tiene la sintaxis

```
WRITE # N, VARIABLE
```

escribe en el file identificado con el número N el valor contenido en la variable especificada.
Para usar la instrucción, el file debe abrirse en modalidad "O" (es una salida del programa).

La instrucción WRITE inserta automáticamente una coma entre los valores si las variables a transferir son más de una.

PRINT y PRINT USING. Su sintaxis es:

```
PRINT # N, VARIABLE
PRINT # N, USING...
```

Tienen el mismo significado que las funciones de impresión. Con estas instrucciones, el file N se trata como si fuese un periférico cualquiera.

EOF(N). Es una función correlacionada al file (secuencial) número N que restituye el valor – 1 (verdadero) cuando en la lectura del file se encuentra el código particular que señala el fin del propio file. El código EOF se deriva de las iniciales de End OF File (fin del file). Esta función, usada en el bucle de lectura de cualquier file secuencial, permite desarrollar el bucle adquiriendo todos los valores escritos sin conocer antes la longitud del file. Así, el programa

```
10 OPEN "I",1,"A:PRUEBA"
20 INPUT # 1,A$,B$,C$
30 IF EOF(1) GOTO 100
40 GOTO 20
100 PRINT "Fin de lectura del file"
110 END
```

realiza el bucle de las instrucciones 20, 30, 40 hasta encontrar el código del End Of File. Aquí la función EOF(1) es verdadera (vuelve el valor – 1 = verdadero*) y el programa salta a la instrucción 100. Si se omitiese la línea 30 no se tendría ningún control y se intentaría leer más records de los que contiene el file, con el consiguiente error y la detención del programa.

LOC(N). Restituye el número de sectores leídos o escritos en el file N desde el momento de su apertura (recordemos que un sector está constituido por 128 o 256 bytes).

KILL. La línea KILL "NOMBRE" anula el file NOMBRE. El file a cancelar debe cerrarse previamente. La instrucción KILL puede utilizarse en cualquier tipo de file (secuencial, directo o de programa).

NAME. Esta función se utiliza para cambiar el nombre de un file. La línea genérica

```
NAME "NOMBRE 1" AS "NOMBRE 2"
```

sustituye al antiguo nombre NOMBRE 1 por el nuevo nombre NOMBRE 2. Por ejemplo, la línea

```
NAME "OLDFIL" AS "NUEVO"
```

sustituye el nombre OLDFIL por el nombre NUEVO. Ahora, en el disco seleccionado deberá existir un file de nombre NUEVO, mientras que ya no deberá haber el nombre OLDFIL.
En la pág. 640 se ha representado el diagrama de flujo de un programa que utiliza las instrucciones descritas para escribir en un file secuencial una serie de datos introducidos por teclado. Al final de la introducción, el programa da la longitud del file (en sectores) y procede a la relectura y a la impresión de los datos introducidos. El diagrama de flujo es mucho menos detallado que los que se han presentado hasta ahora y respeta la forma empleada en la práctica.
Los diagramas de flujo de un programa deben sintetizar las acciones que debe desarrollar el mismo, aunque sin entrar en el detalle de cómo obtener el resultado: este nivel de detalle podrá conseguirse en la fase de escritura de las instrucciones. Por ejemplo, en el bloque 500 del diagrama de flujo de la pág. 640 se ha indicado un control sobre los primeros cuatro caracteres de la cadena introducida. La función a desarrollar sólo se ha indicado con el símbolo de decisión mientras pide estas fases de elaboración:

1 / Controla que la cadena de entrada tenga al menos cuatro caracteres
2 / En caso afirmativo, los primeros cuatro caracteres se transfieren a otra cadena
3 / Siempre en caso afirmativo, esta última cadena se confronta con la cadena FIN

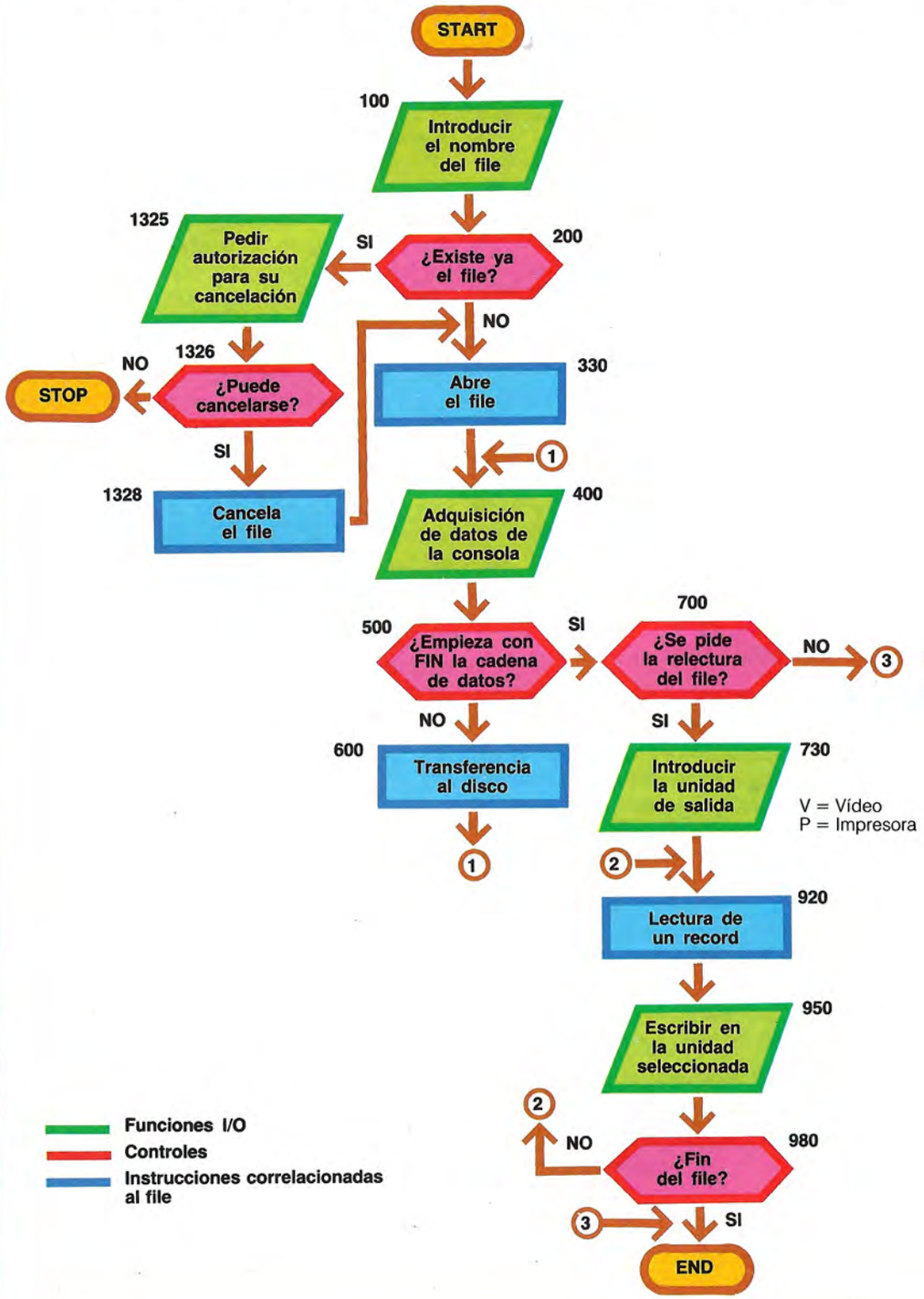
El listado del programa se indica en la pág. 641. El diagrama de flujo de la pág. 640, que no entra en los detalles, también deja para éste la máxima libertad de elección sobre cómo codificar el programa. El listado de la pág. 641 es una de las posibles soluciones, detallada y simplificada a voluntad. Sin embargo, pueden obtenerse los mismos resultados con programas más compactos, aunque menos comprensibles.

Adición de datos a un file secuencial

Los records de un file secuencial no pueden direccionarse. Por tanto, no pueden añadírsele

* En algunas máquinas la condición tiene signo opuesto: + 1 en lugar de – 1.

EJEMPLO DE LECTURA Y ESCRITURA DE UN FILE SECUENCIAL



EJEMPLO DE LECTURA Y ESCRITURA DE UN FILE SECUENCIAL

```
10     ** EJEMPLO DE LECTURA Y ESCRITURA DE UN FILE SECUENCIAL
20      FILE  :  SEQ
30  ON ERROR GOTO 1000
100 INPUT ' NOMBRE DEL FILE ";NM$
200 " * CONTROL EXISTENCIA
205 OPEN "I", 1,NM$       AL FILE HAY ASOCIADO EL NUMERO 1
210 ' * SI EL FILE NO EXISTE, LA INSTRUCCION ANTERIOR GENERA UN ERROR
215    QUE PASA EL CONTROL A LA INSTRUCCION 1000 (VER: ON ERROR...)
300 '
305 '
310 '
320 '
322 '  EL FILE EXISTE
324 PRINT " EL FILE YA ESTA PRESENTE EN EL DISCO"
325 INPUT " DEBE CANCELARSE (SI/NO) "; RESP$
326 IF RESP$ <> "SI" GOTO 2030
328 CLOSE 1: KILL NM$
329 '  PUNTO DE LLEGADA DE VUELTA DE LA 1000 Y SIGUIENTES
330 OPEN  "O",1,NM$            'EL FILE ES DECLARADO DE OUTPUT
331 PRINT
332 PRINT   "LAS INTRODUCCIONES TERMINAN CON LA PALABRA (FIN)"
335 PRINT   "SE HA CREADO UN FILE TIPO (O) OUTPUT"
338 PRINT
400 INPUT' DATOS (4 NUMEROS Y 1 CADENA) ",N1,N2,N3,N4,A$
500 IF A$="FINE" GOTO 700
600 WRITE £1,N1,N2,N3,N4,A$
610 GOTO 400
700   "* FINAL INTRODUCCIONES
710 INPUT" SE DESEA LA RELECTURA DEL FILE (SI/NO) ";RESP$
720 IF RESP$="NO" GOTO 2030
730 PRINT "¿EN QUE PERIFERICO SE DESEA EL LISTADO DEL FILE?"
740 INPUT"  V=VIDEO  P=IMPRESORA ";P$
750 IF P$<>"V" AND P$<>"P" THEN PRINT "** ERROR **" : GOTO 740
900 CLOSE 1:OPEN "I" 1,NM$     ' EL FILE SE CIERRA Y SE VUELVE A ABRIR TIPO "I"
910 K=0       ' INICIALIZA EL CONTADOR DE RECORDS
920 INPUT £1,N1,N2,N3,N4,A$     ' LECTURA DE UN RECORD
930 K=K+1   ' CONTADOR DE RECORDS
940 PRINT "RECORD N.: ";K
950 IF P$="V" THEN PRINT N1,N2,N3,N4,A$
960 IF P$="P" THEN LPRINT "RECORD N.: "; K:LPRINT N1,N2,N3,N4,A$
980 IF EOF (1)   THEN 2000    '   THEN 2000 EQUIVALE A LA (GOTO 2000)
995 GOTO 920
1000 ' ******  GESTION ERROR     ******
1010 PRINT " EL FILE NO EXISTE, DEBE CREARSE "
1020 RESUME 329
2000 '
2020 PRINT " LEIDOS : ";K;" RECORDS"
2030 PRINT "    ******   RUN   END     ******"
2040 END
```

```
RECORD N.:   1
 12              45         78          23              JUAN
RECORD N.:   2
 55              78          5         789              LUISA
RECORD N.:   3
 11              22         33          44              JORGE
RECORD N.:   4
 99              88         77          66              TERESA
```

datos escribiendo directamente al final del file. En vez de esto debe realizarse una copia del file que debe modificarse añadiendo, en la fase de escritura de la copia, los nuevos records. El procedimiento detallado a seguir es el siguiente:

1 / Apertura del file original en modalidad "I" (para el programa constituye una entrada)
2 / Apertura de un segundo file en la modalidad "O"; el segundo file deberá contener los datos del primero más los records a añadir
3 / Transferencia del file original al segundo; así se crea una copia de los datos en el segundo file, que está en curso de elaboración
4 / Añadir nuevos records al segundo file
5 / El primer file, al no ser ya necesario, primero se cierra y luego se cancela
6 / El último paso consiste en atribuir el nombre del primer file al segundo; así se obtiene un file que tiene el mismo nombre que el original, pero de mayor longitud.

Funciones de acceso a los files directos

Los files directos son los más utilizados, ya que permiten el acceso directo al record que interesa. Además, se memorizan en forma binaria y en el disco ocupan menos espacio que los files secuenciales, que se memorizan en ASCII.
Para los files directos no debe declararse el tipo ("I", "O"), ya que pueden utilizarse indiferentemente en lectura o en escritura.
Para el intercambio de datos entre estos files, el sistema operativo utiliza un buffer (recordemos que con este término se entiende un área de memoria utilizada como apoyo durante las funciones particulares, como por ejemplo el intercambio de datos con los periféricos).
Por tanto, para realizar la operación de escritura en disco, los valores a transferir deben pasar primero por el buffer, que se descarga sucesivamente en el disco. En cambio, en lectura, los datos se toman del disco y se depositan en el buffer; de éste, con las oportunas instrucciones, se pasan al programa que los pide.
Es importante observar que el buffer, cuyo nombre está definido en el programa de aplicación, no puede utilizarse si no es para intercambiar datos con el disco. Antes de cualquier otro uso, los datos contenidos en éste deben transferirse a otra zona de memoria identificada con un nombre diferente.
Las funciones y las instrucciones utilizadas en la gestión de los files directos son las siguientes:

OPEN
FIELD
CLOSE
PUT
GET
LOC
KILL
NAME

A estas se añaden todas las funciones de cadenas necesarias para dar el formato exacto a los datos. A continuación examinaremos con detalle la sintaxis y las funciones de cada instrucción. Para las funciones KILL y NAME se remite a lo que se ha dicho para los files secuenciales.

OPEN. La sintaxis es la siguiente:

```
OPEN "R",#N,"NOMBRE",L
```

donde

R — indica que se trata de un file directo
N — es el número con que el file se indica en el programa (en esta instrucción el símbolo # no siempre es necesario)
NOMBRE — es el nombre del file e indica a la vez la unidad de disco sobre la que reside (por ejemplo A:PRUEBA); si la unidad de disco no se especifica, el sistema asume la última unidad de disco (driver) seleccionada
L — expresa la longitud en bytes del record lógico

Por ejemplo, el comando

```
OPEN "R",3,"B:TEST",64
```

abre un file directo con el nombre TEST en la unidad B. El file tiene el número 3 asociado, y sus records tienen una longitud igual a 64 bytes.
El comando OPEN puede servir también para la **creación de files**. En el ejemplo anterior, si el file TEST existe, simplemente se abre y su contenido queda accesible; si el file no existe, el sistema procede a crearlo y a continuación lo abre con la misma instrucción.
Esta lógica puede crear alguna dificultad. Los programas más complejos necesitan a menudo trabajar sobre files contenidos en discos diferentes. Si el usuario olvida sustituir el disco, la instrucción OPEN, que sobre el disco adecuado sólo habría abierto el file, en realidad crea uno nuevo con el mismo nombre sobre el disco equivocado. Para evitar este inconveniente, primero debe abrirse el file como file secuencial en

la modalidad input (tipo "I"). Si el file no está, este comando genera un mensaje de diagnóstico y detiene la ejecución del programa. En cambio, si el file existe, el sistema no señala error. Entonces debe cerrarse el file, anulando la apertura en modo secuencial realizada sólo para control, y volverlo a abrir en modalidad "R" directa (random). Un programa que ilustra esta técnica puede estar estructurado así:

```
10 ' Secuencia de apertura de un file directo
20 ON ERROR GOTO 100
30 OPEN "I",1,"PRUEBA"
40 CLOSE 1
50 PRINT "El file existe"
60 OPEN "R",1,"PRUEBA",64
70 '
80 ' Sigue el programa de aplicación
90 '
100 ' ERROR
110 PRINT "El file no existe"
120 INPUT "Debe crearse"; R$
130 IF R$ <> "SI" THEN STOP
140 RESUME 60
```

El programa sólo es indicativo, ya que el uso que se hace en la línea 20 de la instrucción ON ERROR produce la creación del file, cualquiera que sea el origen del error. En los programas de aplicación deben condicionarse las instrucciones inherentes al file (líneas 110, 120, 130, 140) al tipo de error. Al verificarse un error de sistema, en la variable ERR se escribe el código del error y en la ERL el número de línea que lo ha generado (ERR y ERL son nombres reservados al sistema). Controlando estos valores puede establecerse si la causa corresponde a la ausencia del file o si es otra.
Así, suponiendo que el sistema operativo responde con ERR = 53 (es el caso del CP/M), el programa anterior debe modificarse como sigue:

```
100 ' ERROR
110 IF ERR = 53 AND ERL = 30 GOTO 130
120 ON ERROR GOTO 0
130 ' Siguen las mismas instrucciones
140 ' de las líneas 110, 120, 130, 140
150 ' del programa considerado anteriormente
```

La línea 110 controla el tipo y la procedencia del error; si los valores no son reconocidos como generados por la ausencia del file, el control pasa a la línea 120, que reactiva el mecanismo de gestión de errores del sistema; se emite el diagnóstico normal y el programa se detiene.

FIELD. Asigna a un file (que se ha abierto anteriormente) el buffer a utilizar en las operaciones de lectura y escritura. La sintaxis es:

```
FIELD 1,64 AS B$
```

La línea asigna a todo el record (64 bytes) el buffer B$.
El buffer así definido es el «vehículo» de intercambio de datos con el disco; no puede utilizarse para otras finalidades y, en particular, no puede incluirse en eventuales instrucciones de INPUT y de asignación. La siguiente transferencia de los datos del buffer al programa se obtiene con las instrucciones LSET o RSET.
Todo el record de datos puede subdividirse en más de un buffer, especificando para cada uno la longitud en bytes. Por ejemplo, la asignación antes considerada (64 AS B$) puede subdividirse en cuatro buffers del siguiente modo:

```
FIELD 1,12 AS N$,30 AS C$,20 AS V$,2 AS F$
```

El record (64 caracteres) está dividido en cuatro buffers: N$ de 12 caracteres, C$ de 30 caracteres, V$ de 20 caracteres y F$ de 2 caracteres.

CLOSE. El funcionamiento es idéntico al caso de los files secuenciales: insertando la línea CLOSE # N, el file número N (definido anteriormente con una OPEN) se declara cerrado, y su contenido ya no es accesible (hasta una nueva apertura del mismo file). La instrucción CLOSE libera los buffers de apoyo asignados al file cerrado, que así pueden utilizarse para otro file. Naturalmente, este file debe abrirse después de haber cerrado el anterior.

PUT. Es la instrucción con la que el contenido del buffer (o de los buffers) se transfiere al disco. En la instrucción debe declararse en qué file se quiere escribir, especificando su número (definido en la OPEN), y en qué posición, es decir, para qué número del record lógico. Por ejemplo, la instrucción

```
PUT # 1,75
```

escribe en el record 75 del file 1 (en algunos casos el símbolo # puede omitirse).
La longitud de un file directo no debe definirse

SOLUCIONES DEL TEST 18

1 / Los atributos son indicadores, memorizados junto con el carácter, que definen las particularidades. Por este motivo, la memoria del vídeo posee más posiciones que las estrictamente necesarias para memorizar sólo el código ASCII del carácter.

2 / Las líneas 20 y 30. La 20 escribe 20 veces una cadena de 6 caracteres de longitud (PRUEBA) para un total de 100 caracteres en la misma línea del vídeo. Normalmente, este último puede alojar 80 caracteres por línea. La línea 30 contiene el mismo tipo de error: TAB(100) posiciona la escritura de la variable C en la columna 100, que para el vídeo no existe (en cambio, la instrucción puede ser válida para la impresora sustituyendo PRINT por LPRINT).

3 / El bucle escribe una serie de datos sobre cada una de las líneas del vídeo, hasta la línea 30. Como el vídeo suele poseer 24 líneas, los primeros valores se pierden, ya que deben «correr» hacia la parte alta para dejar sitio a los últimos.

4 / A continuación se indica el listado del programa

```
10 'Solución punto 4
20 'Lectura de datos de vídeo
30 INPUT "Cadena a escribir" ; A$
40 INPUT "En qué columna" ; X
50 IF X = 0 GOTO 160
60 INPUT "En qué línea" ; Y
70 IF Y = 0 GOTO 160
80 '
90 'Controles
100 '
110 K = LEN(A$) 'Longitud de la cadena a escribir
120 IF (K + X) >80 THEN PRINT "Error" : GOTO 40
130 '
140 PRINT FNP$(X,Y);A$
150 GOTO 30
160 END
```

5 / La instrucción 140 se convierte en

```
140 PRINT   FNP$(X,Y);   INV$;    A$;     NOR$
            ( Posición )  ( Vídeo  ( Dato )  ( Vídeo
                          invertido)         normal )
```

La instrucción también puede escribirse sobre más líneas utilizando el símbolo ; que elimina el retorno al principio. Así, la línea anterior puede dividirse como sigue:

```
140 PRINT FNP$(X,Y) ; 'Posición
142 PRINT INV$ ; 'Vídeo invertido
144 PRINT A$ ; 'Cadena a escribir (dato)
146 PRINT NOR$ 'Vídeo normal
```


De este modo se pueden condicionar algunas líneas de impresión al valor de un flag y, por tanto, obtener formas de presentación diferentes según el valor del flag.
Por ejemplo, las siguientes instrucciones generan la escritura con vídeo invertido sólo si el FLAG vale 1:

```
140 PRINT FNP$(X,Y) ;
142 IF FLAG = 1 THEN PRINT INV$ ;
144 PRINT A$ ;
146 IF FLAG = 1 THEN PRINT NOR$
148 IF FLAG < > THEN PRINT
```

La línea 146 devuelve el vídeo al funcionamiento normal y realiza el retorno al principio (CR) en el caso de FLAG = 1. En el caso opuesto (FLAG < > 1) se produce una línea con sólo el PRINT para realizar el CR que se había suprimido en la línea 144.

previamente; a medida que se producen nuevos records, el sistema los tiene en cuenta y actualiza los punteros del file (éstos y otras informaciones están contenidas en el directorio).
En general, este procedimiento no produce inconvenientes, excepto en el caso en el que se varíen más veces las longitudes de más files. Inicialmente, los files, por ejemplo en número de tres, se abren y ocupan físicamente tres zonas contiguas del disco. Si el primero debe alargarse más allá de un cierto límite, los nuevos records no podrían acomodarse físicamente en un lugar contiguo al precedente, ya que antes o después se encontrarían los records del segundo file. Por tanto, la extensión del primer file se posiciona por separado.
Lo mismo puede suceder para el segundo y para el tercer file (este último encuentra el espacio ocupado por las extensiones de los anteriores). Las diversas partes de cada file sólo pueden volverse a encontrar gracias a una compleja tabla de direcciones (tabla de las extensiones). Incluso si esta última la gestiona interiormente el sistema operativo, la ejecución del programa también se complica y enlentece. Las soluciones pueden ser dos:

- copiar periódicamente el disco que contiene los files sobre uno nuevo. Durante la transferencia, el sistema procede a colocar físicamente contiguos todos los files y sus extensiones
- copiar desde el principio los files con longitudes iguales a las máximas previstas, operación que comporta la escritura de un valor cualquiera en el último record previsto.

Proyecto de un circuito impreso.

Marka

GET. La sintaxis de esta instrucción es:

GET # N,R

Lee los datos del record R del file N y los transfiere a los buffers anteriormente asociados al file N. Por ejemplo,

GET # 1,75

lee los datos que están contenidos en el record 75 del file 1.

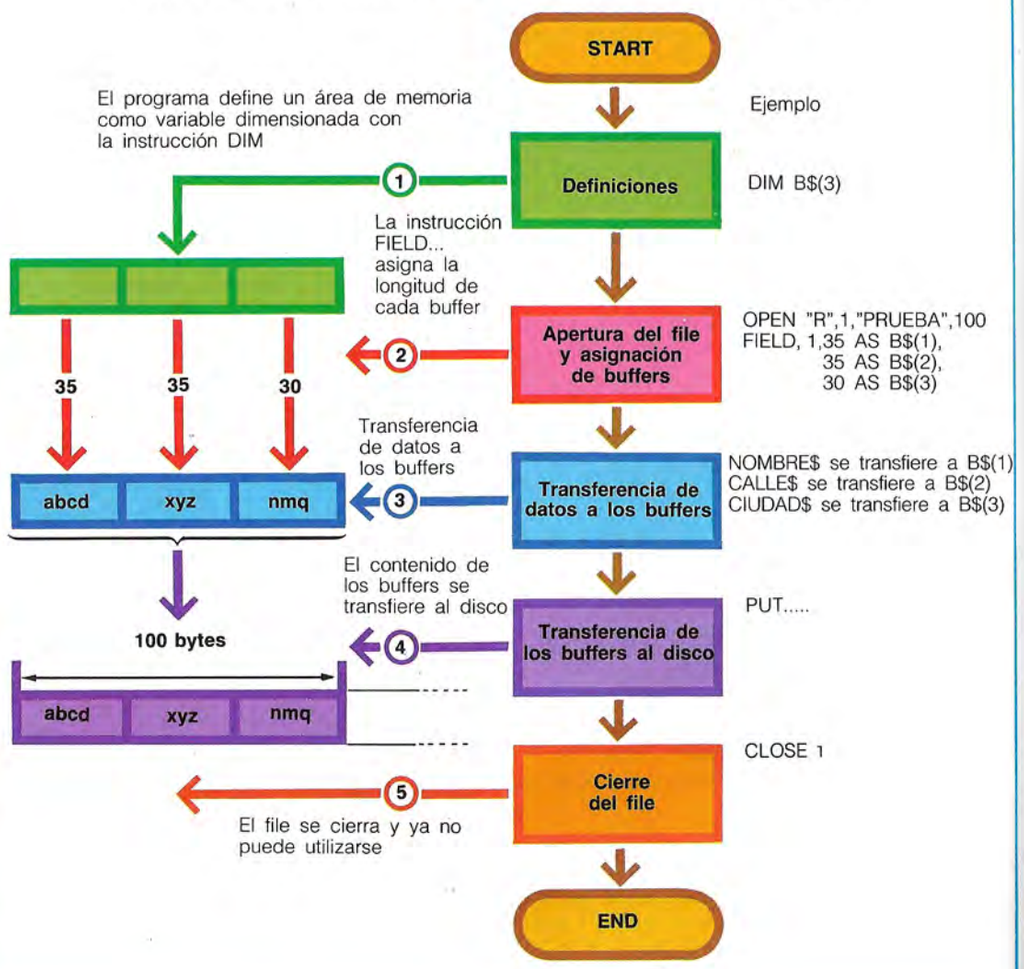
LOC(N). La función LOC(N), con N igual al número del file al que se refiere, restituye el valor «que pasa» del puntero al record.
El valor que pasa es el número del último record utilizado en una instrucción PUT o GET más 1. Por ejemplo, las instrucciones

10 GET 1,7
20 PUT 1,9
30 A = LOC(1)

proporcionan A = 10, ya que el último número de record utilizado es 9 (instrucción 20).

Abajo se ha representado un esquema de transferencia de datos al disco para la memorización de un listín de direcciones.

ESQUEMA DE INTERCAMBIO DE DATOS CON EL DISCO



PROGRAMA QUE ESCRIBE EN UN FILE SECUENCIAL

```
10  ' ** PROGRAMA QUE ESCRIBE EN UN FILE SECUENCIAL **
20  '                FILE = SCFLSQ
30  INPUT "NOMBRE DEL FILE ";NM$
40  NN$="A: "+NM$     ' EN LA VARIABLE (NN$) ESTA MEMORIZADA LA UNIDAD (A:)
50  '                   EN QUE RESIDE EL FILE Y EL NOMBRE DE ESTE ULTIMO
60  OPEN "O",1,NM$    ' ABRE UN FILE TIPO (OUTPUT) NUMERO (1) CUYO NOMBRE
70  '                   RESIDE EN LA VARIABLE(NM$)
80  PRINT "LA INTRODUCCION TERMINA INTRODUCIENDO (0)"
90  PRINT
100  INPUT "VALOR NUMERICO A ESCRIBIR"; V
105  IF V=0 GOTO 150
110  INPUT "CADENA A ESCRIBIR "; A$
120  PRINT #1,V,A$ ' ESCRIBE EN EL FILE 1 LAS VARIABLES V,A$
130  GOTO 80       ' INICIA UN NUEVO CICLO DE INTRODUCCIONES
140 '
150 PRINT "***** FIN INTRODUCCION *****"
160 CLOSE 1         ' CIERRA EL FILE 1. LA LINEA SIGUIENTE LO VUELVE A ABRIR DE
170 OPEN "I", 1,NM$ ' TIPO INPUT (HASTA QUE EL PROGRAMA PUEDA IR A
180 '                 LEER.
190 INPUT #1,V,A$   ' LEE LOS DATOS DEL FILE Y LOS TRANSFIERE A LAS VARIABLES
200 '                 V,A$
208 LPRINT
210 LPRINT "VALOR NUMERICO LEIDO=";V
212 LPRINT
220 LPRINT "CADENA LEIDA=";A$
230 IF EOF(1) GOTO 300 ' SI EL FILE SE HA TERMINADO, SALTA A LA LINEA 300
240 GOTO 190           ' SI NO, INICIA UN NUEVO CICLO DE LECTURAS
250 '
260 '
300 PRINT "***** RUN END *****"
310 END
```

```
VALOR NUMERICO LEIDO = 1965
CADENA LEIDA = MARIA
VALOR NUMERICO LEIDO = 1959
CADENA LEIDA = LAURA
VALOR NUMERICO LEIDO = 1925
CADENA LEIDA = ANA
```

Los campos previstos son los siguientes:

apellido y nombre = 20 + 15 caracteres
calle y número = 30 + 5 caracteres
ciudad y D.P. = 30 caracteres

Los buffers necesarios son tres, uno para cada campo de datos, y pueden llamarse con tres nombres de cadena cualesquiera.
En lugar de esto conviene utilizar el mismo nombre colectivo, que deberá diferenciarse con un índice; en otras palabras, puede utilizarse una variable de cadena dimensionada (en este caso específico de tres valores), de manera que se tenga un solo nombre reservado a la función de buffer en las operaciones de entrada y salida.

Formatos y codificación de los datos

En los files secuenciales, los datos se memorizan en formato ASCII, pero pueden proporcionarse a la rutina de adquisición también como valores numéricos. Arriba se ha representado el listado de un programa que escribe en un file secuencial (abierto en la línea 60 con el número 1) un cierto número de datos introducidos por teclado y constituidos, alternativamente, por un valor numérico (línea 100) y por una cadena (línea 110). La introducción termina introduciendo

Algunos kbytes para jugar

Al principio de los años 70, en el período en que más extendida estaba la psicosis de los OVNI y tanto se hablaba de los hombrecillos verdes, de sus avistamientos y de los encuentros en la tercera fase, el primer alienígena verdadero llegaba a la Tierra. Se llamaba «Pong» y era un videojuego que, a través del uso de un pequeño procesador, simulaba por primera vez el juego del ping-pong, representándolo sobre un monitor del todo similar a una pantalla de televisión.
En apariencia no poseía características antropomórficas, pero con los habitantes de nuestro Planeta se entendió inmediatamente. En primer lugar catalizó la atención de aquella variopinta fauna humana que frecuenta las salas de juego (en Estados Unidos se llaman «arcades»), ya que sus máquinas tragaperras están saturadas por un flujo imparable de monedas: constituyó un éxito increíble y su progenie se propagó de Norteamérica a todo el mundo.
El primero en acogerse a la explosión de este fenómeno fue el propio inventor del Pong: Nolan Bushnell. No mucho después, y empujado por un auténtico genio emprendedor, fundó ATARI, la primera gran firma para la construcción a escala industrial de los videojuegos.
La extraordinaria difusión de los juegos electrónicos con monedas destinado a las arcades convenció a los dirigentes de ATARI para poner en el mercado una nueva versión del Pong destinada a las familias, que utilizaba como monitor el difundido televisor doméstico.
Era el año 1975, y el Pong se convirtió en el primer videojuego doméstico de la historia.
El efecto inmediato de esta nueva presencia en casa fue el trastorno de la relación que mediaba entre el espectador y la televisión: por primera vez en una pantalla ya no aparecieron únicamente las sencillas reproducciones de imágenes reales e intangibles.
Desde aquel momento, el imaginario e impalpable hilo que enlaza la emisora de televisión con cada aparato de TV pudo cortarse. El espectador mediante el uso del videojuego, puede abrir de par en par la ventana de la pantalla para asomarse a una nueva realidad.
El jugador interacciona con el videojuego-procesador, intercambia con él informaciones de forma continua y, aunque a través de una representación ficticia, prueba emociones reales: se convierte él mismo en el protagonista de lo que sucede en el vídeo doméstico.
En aquel período se estableció por primera vez la estrecha relación funcional que existe entre dos de los bienes de consumo que más han caracterizado nuestra época: la televisión y el microprocesador. En aquellos años, éste daba sus primeros e inciertos pasos hacia un mercado todavía virgen, pero indudablemente rico en desarrollos en las más variadas aplicaciones, entre ellas el videojuego. Este término (neologismo formado con la unión de las dos palabras vídeo y juego del vocablo inglés «video game») indica un dispositivo electrónico apto para reproducir en una pantalla el ambiente y las circunstancias típicas de un juego específico. El objetivo principal es el de conseguir la realización de un juego mediante la descripción de hechos que generan en tiempo real respuestas e instrucciones del jugador hacia el procesador y viceversa.
En el videojuego encontramos una fiel réplica de las características funcionales típicas de un procesador electrónico interactivo.
Físicamente está constituido por una consola, por una o dos palancas de control y por un aparato de televisión normal funcionando como pantalla. Sabemos que el procesador y el jugador intercambian continuamente informaciones. En base a la lógica de este intercambio pueden describirse las características constructivas y funcionales de los diversos componentes del videojuego. Una vez realizadas las conexiones eléctricas, la consola envía al jugador el primer mensaje informativo. De forma parecida a como sucede en los dibujos animados, en la pantalla de TV se genera un paisaje de fondo sobre el cual se mueven los operadores activos (o personajes) del juego: el conjunto de estos dos elementos constituye la imagen de televisión.
Reaccionando ante este estímulo, el jugador impone sus propias decisiones más o menos rápidamente, de acuerdo con las necesidades de la dinámica del juego, y las envía al procesador a través de los mecanismos de control (joystick, track-ball), igual que un automovilista gira el volante para girar o pisa el pedal para acelerar o frenar. Los comandos, impartidos por el jugador en forma de desplazamientos mecánicos, se transforman en el dispositivo de comando en impulsos eléctricos fácilmente interpretables por el procesador. Éste, en una fracción de segundo, los interpreta, controla que sean compatibles con el correcto desarrollo del juego y, en base a las instrucciones que encuentra en el programa de gestión que tiene escrito en memoria, propone al jugador una imagen televisiva en la que está representada una nueva situación del ambiente. En una continua sucesión de acciones y reacciones, el videojuego varía las posiciones de los operadores activos, describe nuevos fondos y paisajes, cambia los colores y emite sonidos o música con el fin de hacer más sugestiva la simulación. En base al tipo de imagen de televisión propuesta existen dos categorías diferentes de juegos. Volviendo al ejemplo del automovilista, podemos decir que el juego en que el mecanismo de control no mueve operadores activos sino toda la panorámica de la imagen de televisión, igual que si el jugador estuviese a bordo de un coche o de un avión (es famosa la simulación de vuelo), se define como subjetivo; *en cambio,* objetivo *es aquel en que el jugador está representado en la pantalla por un operador activo, con el que se identifica haciéndolo mover de acuerdo con los propios comandos.*
Cada procesador que compone el videojuego puede gestionar las dos categorías a través de la sencilla sustitución del «cartridge», o cartucho, en el que está memorizado el programa que genera el juego. El cartucho ha sido, para el mercado de los videojuegos, el elemento revolucionario que, en gran parte, ha decretado su éxito comercial. Mediante su empleo, el usuario se libera de la esclavitud de tener que sustituir la consola para variar el juego: en el caso del Pong, todos los datos del programa estaban escritos en una memoria soldada en el interior de la consola. En los videojuegos de las siguientes generaciones (el VCS 2600 de ATARI, por ejemplo), el mismo programa está escrito en una memoria desconectable de la consola: mediante un conector (slot) puede sustituirse por otro cartucho para generar otros tantos juegos diferentes. Esta versatilidad, que recuerda muy de cerca la relación análoga que existe entre el grabador y la cassette o entre el tocadiscos y el disco, contribuyó a poner en marcha un nuevo mercado, el de la producción de cartuchos para videojuegos, al cual han tenido acceso las empresas que no construyen consolas. Fue el nacimiento y la prosperidad de nuevas industrias que hoy facturan anualmente del orden de centenares de millones de dólares; a esta crónica debe añadirse la nueva figura profesional del «game designer»: el creador de videojuegos.
Junto con sus colegas artistas y músicos, este personaje firma las propias creaciones, participa en los beneficios de las ventas, posee un público de fans y, en sustancia, es la nueva estrella de los años 80. Para él, transformar un buen tema en videojuego puede representar un asunto extremadamente lucrativo, y para la casa productora del cartucho un negocio de millones de dólares: ¡el nuevo Eldorado!
Pero los caminos que llevan a esta mítica tierra empezaron a congestionarse en 1983. El apetito, aunque sólo fuese por una pequeña tajada de los 1,5 billones de ptas. que representan el giro de negocios que gravitan alrededor de los videojuegos, empujó a una multitud de pequeñas y grandes firmas a incorporarse a esta aventura.
Inevitablemente surgió el conflicto. Lanzadas a una frenética carrera por la conquista de un mercado considerado fácil y prometedor, pero que pronto demostró ser más comprometido de lo previsto, muchas firmas se encontraron en un escenario digno del mejor de los «war games».
A principios del año 83, el advenimiento de la tercera generación de videojuegos produjo una drástica reconfiguración de las cuotas de mercado, haciendo difícil la supervivencia de aquellos que no ofrecían un producto tecnológicamente puesto al día. Paralelamente, en el campo de los videojuegos se desarrolló una abierta polémica con respecto a su presunta peligrosidad en el ámbito sicológico-social. Canibalístamente, muchos mass-media se dedicaron a una campaña de efecto escandaloso para desacreditar a los ojos del público este moderno fetiche que sólo pocos meses antes había sido objeto de tantas atenciones favorables. Así se ha consumado una típica farsa de la cultura de los excesos, en la que las elecciones no se hacen siempre ni constantemente mediante la inteligencia y la capacidad cognoscitiva del pensamiento humano, sino que son generadas por un sentimiento de angustia colectiva del que se huye recurriendo al uso y consumo continuo de artificios fascinantes, coloreados, cautivadores pero tremendamente efímeros. Durante el año 1983, en los locales de la Bolsa de Nueva York se respiró durante varios días un aire de tormentas: algunas grandes firmas vacilaban bajo los golpes producidos por la pérdida de centenares de millones de dólares, otras se iban a pique seguidas de los fúnebres tañidos de la campana del Lloyds. En 1983, el videojuego se repre-

sentó a sí mismo en este difícil paisaje, y en la pantalla muchos operadores activos se desintegraron; pero la partida no se ha terminado «gamer over». Las ventas de consolas y de cartuchos han registrado una cierta flexión sin más indicativo que una mayor atención en las compras: el público no ha desertado en masa ante este económico dispensador de emociones. En Estados Unidos, los ingresos de las arcadas han superado los 5.000 millones de dólares, superando el giro de los negocios de las industrias cinematográfica y discográfica.

Las aventuras que se presentan al jugador son prácticamente casi infinitas y satisfacen todos los gustos. Las estadísticas afirman que, en Estados Unidos, la relación entre la venta de cartuchos y de consolas, en el usuario medio, es de cerca de 7 a 1: 7 juegos de vídeo diferentes por cada jugador. Es posible elegir divertirse entre guerras estelares (la aplicación típica del concepto de dispara y huye), distraerse en complicados y laberínticos rompecabezas, huir de animales y fantasmas enfurecidos o salvar a la Tierra de invasores intergalácticos. El jugador pue-

Una imagen de la manifestación «Computer play 83» organizada en Milán por AICA.

R. Lonardi/A.I.C.A.

de acomodarse en la cabina de pilotaje de un B17 y desde la propia butaca bombardear tranquilamente la Alemania nazi. Bien, tranquilamente no es exacto, dado que un simulador de voces humanas y un generador de sonidos reproducen del mejor modo posible las sensaciones y los miedos que hay en el interior de un bombardero durante una misión.

Actualmente se está desarrollando la investigación y la experimentación de nuevos videojuegos en el sentido de una implicación psicofísico-sensorial siempre mayor. Hoy en día hay disponibles, también en forma de productos comerciales, sofisticadas tecnologías para la reproducción y manipulación de las imágenes de televisión, derivadas de la unión de la microelectrónica con la técnica del láser. Mediante la aplicación de dispositivos electrónicos en los que se ha hecho un amplio uso de componentes especiales (microprocesadores, etc.), la imagen de televisión se digitaliza y se hace inteligible para un procesador que se encarga de transcribirla, a través de un rayo láser, sobre un adecuado disco de metal y plástico. De este modo, sobre una cara del disco pueden quedar disponibles cerca de 250.000 imágenes de televisión, todas diferentes entre sí. A diferencia de las cintas de registro, que proporcionan las imágenes sólo en un orden secuencial, con el disco es posible, en un tiempo brevísimo, «leer» dos imágenes diferentes y muy lejanas en la superficie del propio disco.

Volvamos al ejemplo del bombardero. Dado que las imágenes producidas por el videojuego son muy rústicas, para hacer más verosímil la simulación podemos pensar en proyectar en la pantalla de TV las imágenes extraidas de un filme de guerra en el que se ha representado una o más escenas de bombardeo. Actuando sobre los comandos del videojuego, podremos tener la sensación de estar a bordo de un B17, pero no podremos guiarlo, ya que entre el comando y el film no se puede crear una relación de causa-efecto (o mejor orden-movimiento): se haga lo que se haga, la película del filme actúa siempre por su cuenta.

En cambio, con el uso del videodisco, el videojuego (que es controlado por un microprocesador) escoge, entre muchas, la única imagen de televisión pertinente con las maniobras y las decisiones del jugador piloto. Así se crean recorridos lógicos fundados sobre una verdadera relación acción-reacción. Volviendo a la ca-

Imprimatur

Jugar con la ayuda del ordenador también significa aprender.

bina del B17 y moviendo las palancas de mando tendré como respuesta una serie de imágenes de TV congruentes con mi estilo de pilotaje: si viro a la izquierda veré desaparecer por mi derecha el campo, las casas, el horizonte (descritos con precisión cinematográfica). Podré disparar a los enemigos, mirar siguiendo la traza de mis proyectiles y verlos explotar exactamente como si en realidad estuviese a los mandos de un avión de guerra o en el lugar del protagonista de un filme.

Se ha realizado el viejo sueño de los instructores de vuelo de poseer un simulador perfecto.

Este es el videojuego del futuro que está en puertas; algunos ejemplares de este tipo ya están en uso en Estados Unidos.

«Dragon's Lair» es el título del videojuego fundador de esta nueva ola. Realizado con los fotogramas de un dibujo animado estilo W. Disney, propone al jugador, mediante el uso de unos dibujos absolutamente maravillosos, 42 escenarios diferentes en los que se desarrolla una acción de la mejor épica medieval: salvar a la bella princesa prisionera del dragón; 200 decisiones a tomar correctamente o la muerte. Menos caballeresco, pero bastante más dramático, es el argumento propuesto a algunos estudiantes de medicina norteamericanos.

En lugar de la bella princesa hay un adiposo paciente afectado de peritonitis al que debe operarse con urgencia. Un escenario real, una máxima atención, decisiones rápidas y una absoluta verosimilitud, permiten a los noveles doctores hacer prácticas sin derramamientos de sangre. La primera aplicación de este nuevo videojuego es didáctica, pero es probable que lo utilicemos pronto en nuestras pantallas.

Es de observar que Italia, invirtiendo su tradicional relación comercial con Estados Unidos, se ha unido con retraso a la expansión interna de la venta de videojuegos; pero particularmente para las de salas de juego y de bares, cierra en activo la balanza comercial de este producto. El mérito va unido a la típica y notable capacidad emprendedora de nuestra pequeña industria. Multitud de pequeñas empresas han desarrollado conocimientos válidos en microelectrónica, y hoy pueden suministrar a la exportación un conjunto de productos de interesante calidad y a precios competitivos.

También se ha orientado la atención editorial hacia los videojuegos, y el año 1983 ha visto un pequeño boom de iniciativas correspondientes a este sector: el nacimiento de una nueva revista completamente dedicada a este fenómeno («Videogiochi»), la apertura de espacios de redacción y la aparición de firmas especializadas en numerosas cabeceras.

Es una pacífica invasión nacida bajo el signo de una apertura al mundo del juego y su cultura. Es una lástima que, como de costumbre, haya sido un estímulo procedente del otro lado del océano el que ha generado este interés; y más considerando que nuestra Italia es la depositaria de una antiquísima tradición de juegos.

En el año 1984 se imponen decisiones valientes, y hoy están tomando forma las bases y los caracteres de un fenómeno que, en los años venideros, marcarán por siempre más el empleo de nuestro tiempo libre.

No hay que hacerse ilusiones sobre la presunta independencia del videojuego de los modelos mentales típicos de la sociedad contemporánea: es su fiel reproducción. El ambiente cultural genera juegos y videojuegos. Mientras en éstos se lean únicamente mensajes de agresividad y de guerra, el mundo continuará viviendo el peligro de la autodestrucción. Para comprender cuán difícil es todavía hoy la supervivencia del hombre, basta con ir a África, donde hay personas que para morir de hambre ¡ciertamente no tienen necesidad de la simulación de un videojuego! Por esto debe crearse una alternativa a la masiva difusión de los videojuegos que simulan actos de hostilidad (Guerras Estelares y Co.) proponiendo una nueva filosofía basada en principios morales más válidos. El videojuego corre el riesgo de convertirse en un killer de la actividad pensante: esto sucede debido a modelos mentales utilizados sin criterio y vueltos a proponer continuamente al jugador en forma obsesiva y no sensible, ya que están enmascarados por los aspectos exteriores y más cautivantes (colores, música, sonidos) del juego. No olvidemos que los niños y los jóvenes son los grandes usuarios de esta diversión; es posible que en su personalidad pueda introducirse subrepticiamente alguna distorsión peligrosa. En algunos países anglosajones, la venta de los juguetes está subordinada al hecho de que éstos deben tener bien indicados el modo de uso y la referencia precisa de la edad de los bebés o niños a los que van destinados. Es una forma de salvaguardia y tutela de los más débiles muy difundida en algunas naciones socialmente en vanguardia. Para el videojuego también debería proponerse alguna cosa similar con respecto a las indicaciones sobre el contenido pedagógico y sobre el tipo de público a que va destinado.

El videojuego es, tiene una razón de ser y puede desarrollar un positivo papel social, sobre todo en la medida en que favorece al jugador al hacer activa su propia creatividad.

«Aprendemos más cuando tenemos que inventar», decía Jean Piaget, y también en este sentido el videojuego y el ordenador personal pueden ser dos instrumentos de soporte válidos para el aprendizaje.

Las diferencias que hay entre el videojuego y el ordenador personal no son abismales; este último es una evolución natural del juego hacia aplicaciones más universales del procesador en el ámbito doméstico. El procesador doméstico posee la misma estructura básica de la consola de un videojuego, pero ampliada con la introducción de otros dispositivos. El teclado, una mayor área de memoria y el grabador para la cassette de los programas sirven para hacerlo extremadamente versátil en su uso, sin necesidad de un adiestramiento particular del usuario. También el ordenador personal puede conectarse fácilmente a la pantalla de televisión y está dotado de muchas opciones: unidad de disco, impresora, interfaz para la conexión telefónica, lápiz óptico.

En el año 1983 se ha producido una verdadera invasión de procesadores domésticos en Italia. Estimaciones dignas de consideración informan que sólo en la campaña de Navidad se vendieron más de ciento veinte mil ejemplares: un éxito sin precedentes. Entre los programas dedicados a los ordenadores personales es posible realizar una clasificación de los más solicitados; con ello también se ponen en evidencia los criterios de aplicación del procesador establecidos por el usuario en el ámbito familia-casa. Siempre predominan los juegos: los juegos de entretenimiento y educativos llegan a cubrir el 54% de las ventas, inmediatamente seguidos por los programas para el análisis del balance doméstico con el 17%, por los de gestión de negocios, también el 17%; los programas para el tratamiento de textos (word processing) suponen el 3,5% de las ventas. Resulta evidente que el Homo ludens *utiliza el procesador doméstico principalmente en su tiempo libre.*

La importancia de este fenómeno no debe desdeñarse: el juego es una elección persistente también cuando el jugador puede producir programas de forma autónoma. Esto significa que estamos en presencia de estímulos fundamentales, de deseos profundos que se expresan en el hombre y que los quiere satisfacer como sea.

Para desarrollar un profundo examen sobre este tema, los días 2 y 3 de diciembre de 1983 se celebró en Milán el primer Congreso Muestra sobre el juego informático: COMPUTER PLAY 83. La manifestación nació en el seno de la sección estudiantil de la AICA (Associazione Italiana per l'Informatica ed il Calcolo Automatico) en el curso de licenciatura en Ciencias de la Información de Milán, precisamente porque un poco entre todos (profesores, investigadores, estudiantes) se había difundido la exigencia de saber alguna cosa más sobre la cultura del juego y sobre sus valores de aplicación.

La iniciativa, propuesta por G. Degli Antoni (director del curso de licenciatura), por G. Occhini (presidente de la sección de Milán de la AICA) y por mí, como responsable de la sección universitaria de la AICA, ha querido estimular el debate sobre la problemática correspondiente a la difusión de los videojuegos y con el intento de proponer, por parte de expertos cualificados, un análisis de los numerosos interrogantes que ha planteado la aparición de los juegos informáticos: qué es un juego, cómo se proyecta, cómo se evalúa, cuáles son los aspectos educativos del juego informático, qué mercado tiene, para qué juegos y con qué metodologías puede introducirse el juego en la didáctica, qué juegos para qué edad. En el congreso, los participantes han afrontado estos temas partiendo de diferentes bases de análisis; en cualquier caso se ha puesto de manifiesto la tendencia hacia unas aplicaciones cada vez mayores de los juegos educativos y de alto contenido conceptual, además de las típicas aplicaciones específicas de la didáctica.

Se trata de una nueva cultura del juego que se está haciendo lugar entre los gustos de la gente, y merece la debida atención.

Uno de los intentos de los promotores ha sido el de verificar sobre el terreno qué se estaba haciendo, a nivel de iniciativa privada, en el sector de los juegos para ordenador.

En Italia hay muchísimos fabricantes independientes (es decir, no ligados a casas productoras de hardware o software) de software-juegos;

R. Lonardi/AICA

El ordenador personal más sofisticado tampoco desdeña la aplicación al juego.

COMPUTER PLAY 83 puede adjudicarse el innegable mérito de haber conseguido, por primera vez en nuestro país, llamar la atención. Mediante el lanzamiento del CALL FOR GAMES se convocó un concurso al que se adhirieron más de 100 programadores y se tuvo un conocimiento público de sus capacidades creativas que han resultado ser de óptimo nivel.

Estos game designers *nuestros se han revelado a la altura de sus colegas del otro lado del océano. Quizá para ellos no habrá tanta fama ni tanto dinero, pero sin embargo han demostrado que también nuestro país es rico en energías intelectuales que deben cuidarse y desarrollarse.*

En el futuro, la difusión de los ordenadores personales irá sin duda ligada a la capacidad que tengan estas energías para hacer inteligibles al gran público las nuevas y cada vez más interesantes aplicaciones del ingenio humano.

(Antonio Verga, Associazione Italiana per l'Informatica ed il Calcolo Automatico)

un 0 como valor numérico (línea 105), después de lo cual, el programa pasa a la relectura de lo que se ha escrito en el disco. Para leer el file primero debe cerrarse (para eliminar su definición como file «0») y a continuación volverlo a abrir en modalidad «I».

Obsérvese que las instrucciones PRINT e INPUT de las líneas 120 y 190, debido a que se refieren a un file, incluyen el símbolo #; en algunas impresoras el símbolo # se escribe como £ y este es el caso que se ha presentado.

En los files directos, los datos se memorizan en binario, pero deben suministrarse siempre en ASCII, ya que se utiliza un buffer de apoyo del tipo de cadena.

Esta es la única dificultad que se encuentra en las operaciones de I/O con este tipo de files, por otra parte fácilmente superable utilizando las funciones intrínsecas del Basic.

Las principales funciones que pueden utilizarse para la transformación de los datos son:

MKI$, MKS$, MKD$ en escritura (convierten de numérico a cadena)

CVI, CVS, CVD en lectura (convierten de cadena a numérico)

El uso de estas funciones es inmediato y no presenta ninguna dificultad especial.

A la derecha se ha representado el listado de un programa que utiliza las funciones indicadas para la escritura y la lectura de datos numéricos en un file directo.

La conversión de los datos también puede obtenerse utilizando las funciones STR$(R) y VAL(A$). En la pág. 656 se ha representado el diagrama de flujo de una subrutina que, utilizando estas funciones, prepara el record y lo escribe en el disco. En la pág. 657 se ha representado el diagrama de flujo de la rutina que realiza la función inversa (de lectura). Los listados de las dos rutinas pueden verse en las págs. 657-658 junto a un main de prueba que las utiliza.

Para facilitar la comprensión de la subrutina 2000, en la pág. 659 se han representado los esquemas de la lógica utilizada en la extracción de los datos.

Selección de los datos

No es frecuente que en la gestión de los archivos deban elaborarse todos los records contenidos. Normalmente, las actualizaciones o las otras elaboraciones sólo interesan algunos records que tienen determinados valores en algunos campos. Estos campos constituyen la clave de acceso al record y, cuando el sistema lo permite, conviene adoptar la estructura de file con índice. De este modo, toda la gestión de los punteros la realiza el sistema y el usuario sólo debe implantar por software unas pocas instrucciones que especifican qué claves se van a utilizar para seleccionar el record.

Un ordenador ADM trabajando en una oficina de Falcon Jet Company.

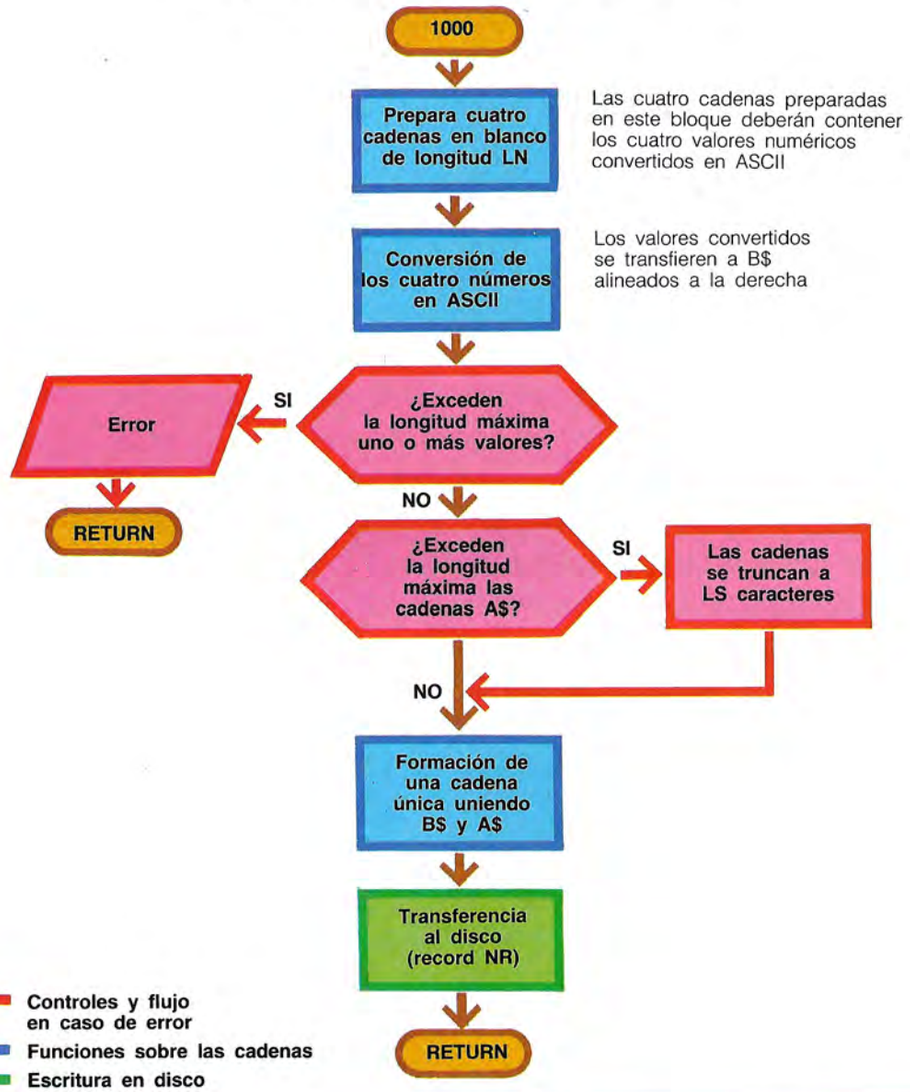
Leo de Wys/Marka

PROGRAMA QUE ESCRIBE EN UN FILE DIRECTO

```
5 ' ** PROGRAMA QUE ESCRIBE EN UN FILE DIRECTO **
10 ' EJEMPLOS DE TRANSFORMACION DE LOS VALORES NUMERICOS EN LAS OPERACIONES
20 '   DE (I/O) EN DISCO (FILES DIRECTOS)
30 '                                FILE = TRABAJ
40 OPTION BASE 1
50 DEFINT I      ' LOS NOMBRES QUE EMPIEZAN CON (I) SE REFIEREN A VARIABLES
60 '               ENTERAS
70 DEFSNG R      ' LOS NOMBRES QUE EMPIEZAN CON (R) SE REFIEREN A VARIABLES
80 '               DE SIMPLE PRECISION
90 DEFDBL D      ' LOS NOMBRES QUE EMPIEZAN CON (D) SE REFIEREN A VARIABLES
100 '              DE DOBLE PRECISION
110  DIM BF$(3)    ' BUFFER I/O (UNO POR CADA TIPO DE VARIABLE)
120 OPEN "R",1,"PRUEBA",30   '  LA LONGITUD DEL RECORD ES DE 30 BYTES
130 FIELD 1,10 AS BF$(1),10 AS BF$ (2), 10 AS BF$ (3)
140 ' * LAS LONGITUDES DE CADA BUFFER SON MAYORES DE LO NECESARIO
150 '
160 '   ** LECTURA VALORES NUMFRICOS A ESCRIBIR CON DISCO
170 '
180 INPUT "VALOR ENTERO"; ENTERO
190 INPUT "VALOR SIMPLE PRECISION ";REAL
200 INPUT "VALOR DOBLE PRECISION "; DOBLE
210 '
220 ' ** LOS NOMBRES ENTERO, REAL, DOBLE SON LAS VARIABLES DE ENTRADA
230 '    A CONVERTIR EN ASCII
240 '
250 A1$=MKI$(ENTERO)     ' LAS FUNCIONES (MKI$, MKS$, MKD$) TRANSFORMAN UN
260 A2$=MKS$(REAL)        ' VALOR NUMERICO EN EL EQUIVALENTE VALOR ASCII
270 A3$=MKD$(DOBLE)
280 '
290 '   INTRODUCCION DEL NUMERO DEL RECORD DONDE QUIERE ESCRIBIRSE
300 INPUT "EN QUE RECORD QUIERE ESCRIBIRSE  ";IR
310 ' * EL NUMERO DEL RECORD DEBE SER ENTERO
320 '
330 ' * TRANSFERENCIA A LOS BUFFERS (DE APOYO)
340
350 BF$(1)=A1$
360 BF$(2)=A2$
370 BF$(3)=A3$
380 '
390 ' * ESCRITURA EN EL FILE RANDOM
400 ' LA LINEA SIGUIENTE TRANSFIERE AL DISCO EL CONTENIDO DE LOS BUFFERS
405 ' (BF$), DEL FILE 1 AL RECORD (IR).
410 PUT £1,IR
420 INPUT  "CONTINUA (SI/NO)"; RESP$
430 IF RESP$="SI" GOTO 180    ' NUEVA ESCRITURA
440 '
450 '   * RELECTURA DATOS ESCRITOS
460 '
470 INPUT "QUE RECORD QUIERE LFERSE ",IR
480 GET £1,IR    'LEE LOS DATOS DEL RECORD (IR) Y LOS TRANSFIERE A LOS
485 '              BUFFERS DE APOYO ASOCIADOS AL FILE (BF$)
490 A1$=BF$(1)
500 A2$=BF$(2)
510 A3$=BF$(3)
520 ENTERO=CVI (A1$)    ' LAS FUNCIONES (CVI,CVS,CVD) TRANSFORMAN UN VALOR
530 REAL=CVS(A2$)        ' ASCII EN EL EQUIVALENTE VALOR NUMERICO
540 DOBLE=CVD(A3$)
550 LPRINT "DATOS LEIDOS:    "; ENTERO, REAL,DOBLE
560 INPUT "CONTINUA (SI/NO)   ";RESP$
570 IF RESP$="SI" GOTO 470
575 CLOSE £1
580 END
```

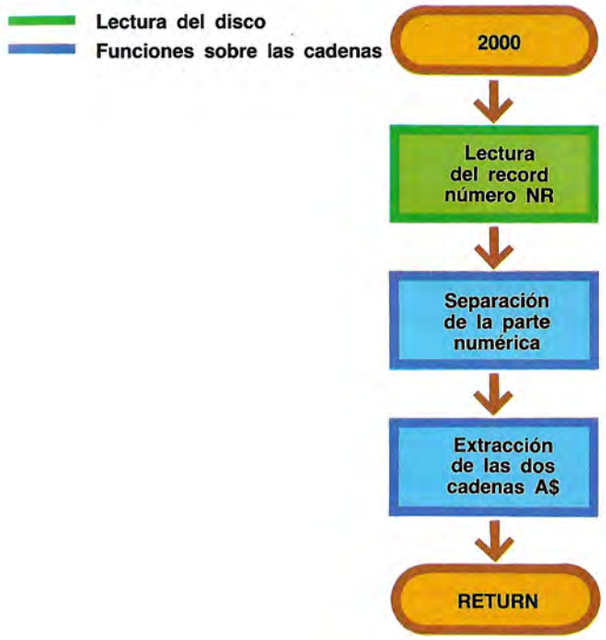
PREPARACION Y ESCRITURA DE UN RECORD

Entradas a la rutina: NR = Número del record en escritura
V(4) = Valores numéricos a escribir en el record
LN = Longitud máxima (en número de cifras) de los valores numéricos
A$(2) = Cadenas a escribir en el record junto con los valores numéricos
LS = Longitud máxima en caracteres de cada cadena



El software de base que permite este tipo de gestión no está en todas las máquinas, y muy a menudo es necesario escribir la subrutina de comparación. Una subrutina de este tipo no presenta particulares dificultades: sólo se trata de utilizar correctamente los operadores lógicos. La única limitación que deberá atenderse es la de alinear todos los campos del mismo modo. En general, durante la escritura en disco, los campos no se llenan completamente con caracteres y, por tanto, en cada campo quedan espacios en blanco. En la fase de relectura, en la memoria se cargan también los espacios. Al comparar un campo obtenido de esta forma con

LECTURA DE UN RECORD Y PREPARACION DE LOS VALORES NUMERICOS



PREPARACION, ESCRITURA Y LECTURA DE UN RECORD

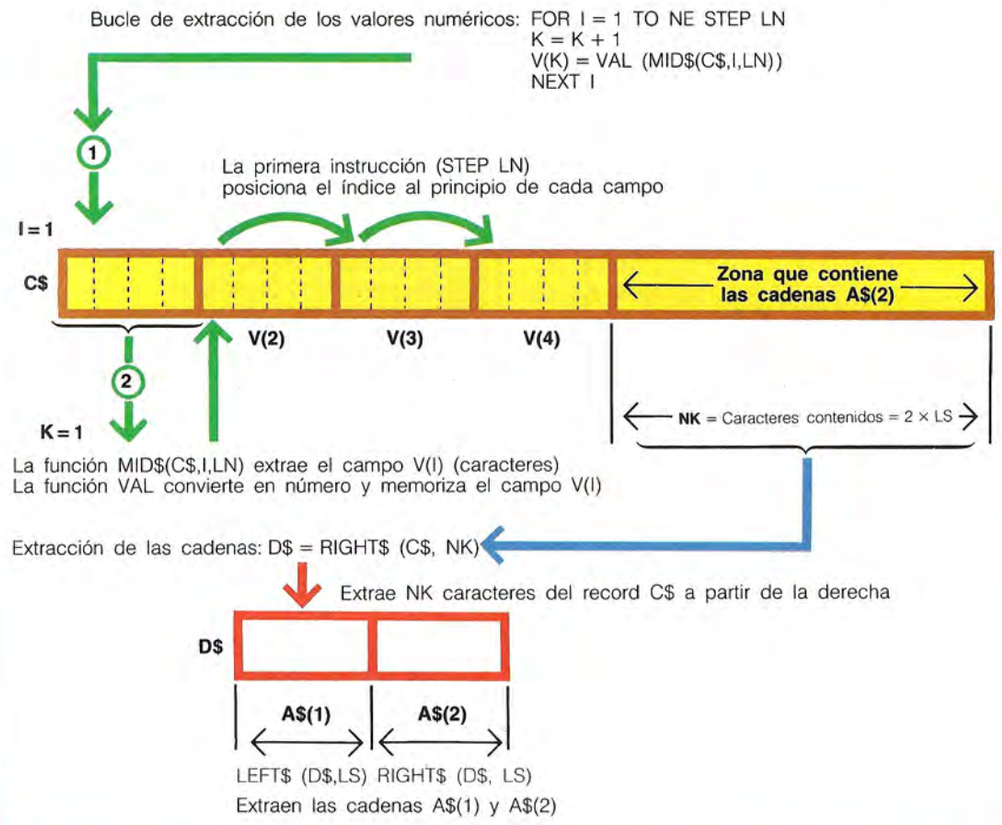
```
10 '
20 '
30 '            FILE = SCRREC
35 DEFINT I-N
40 '  *  ENTRADAS A LA RUTINA:
50 '     NR = NUMERO DEL RECORD EN QUE SE ESCRIBE
60 '   V(4) = VALORES NUMERICOS A ESCRIBIR EN EL RECORD
70 '     LN = LONGITUD MAX., EN EL NUMERO DE CIFRAS, DE LOS VALORES NUMERICOS
80 '  A$(2) = CADENAS A ESCRIBIR EN EL RECORD JUNTO CON LOS
90 '          VALORES NUMERICOS
100 '    LS = LONGITUD MAXIMA DE LAS CADENAS EN CARACTERES
110 DIM B$(4), A$(2)
112 OPEN "R", 1, "PRUEBA2", 50
114 FIELD f1,50 AS BUF$
120 LS=10
130 LN=5
140 INPUT "NUMERO DEL RECORD A ESCRIBIR"; NR
150 FOR I=1 TO 4
160 PRINT "DATO NUMERICO" ; I
170 INPUT "VALOR (MAX. 4 CIFRA) ";V(I)
180 NEXT I
190 INPUT "PRIMERA CADENA : ";D$
200 INPUT "SEGUNDA CADENA : ";E$
202 A$(1)=SPACE$(LS):A$(2)=SPACE$(LS)
204 LSET A$ (1)=D$
206 LSET A$ (2)=E$
210 '
220 GOSUB 1000
230 '
240 A$(1)="":A$(2)=""
250 FOR I=1 TO 5: V(I)=0:NEXT I
```

```
260 '
270 GOSUB 2000
280 '
290 LPRINT "VALORES LEIDOS: "
300 LPRINT "NUMERICOS: "
310 FOR I=1 TO 4
320 LPRINT V(I)
330 NEXT I
340 LPRINT
350 LPRINT "CADENAS: "
360 LPRINT A$(1)
370 LPRINT A$(2)
380 '
390 END
1000 ' **SUBRUTINA ESCRITURA **
1010 IF NR=0 THEN PRINT "ERROR EN EL NUMERO DE RECORD": RETURN
1020 FOR I=1 TO 4
1030 B$(I)=SPACE$(LN)
1040 NEXT I
1050 FOR I=1 TO 4
1060 C$=STR$ (V(I))          ' CONVERSION EN CADENA
1070 IF LEN(C$)>LN THEN PRINT "ERROR EN LONGITUD": RETURN
1080 RSET B$(I)=C$    ' ALINEADO
1090 NEXT I
1100 C1$=A$(1)        ' MEMORIZA LA PRIMERA CADENA
1110 IF LEN(A$(1))>LS THEN C1$=LEFT$(A$(1),LS)
1120 C2$=A$(2)        ' MEMORIZA LA SEGUNDA CADENA
1130 IF LEN (A$(2))>LS THEN C2$=LEFT$(A$(2),LS)
1140 ' ** C1$ Y C2$ CONTIENEN LOS MISMOS DATOS DE A$(1) y A$(2)
1150 '    EVENTUALMENTE TRUNCADOS A LA LONGITUD MAX. PREVISTA
1160 ' ***:PREPARACION CADENA TOTAL
1170 'T$=""
1180 FOR I=1 TO 4
1190 T$=T$+B$(I)          ' EN T$ SE HAN ACUMULADO LAS CADENAS B$(I)
1200 NEXT I
1210 T$=T$+C1$+C2$     ' SE HAN SUMADO LAS DOS ULTIMAS CADENAS A LOS DATOS
1212 PRINT     "LOS DATOS SE HAN TRANSFERIDO AL DISCO EN LA FORMA: "
1214 PRINT T$
1220  BUF$=T$              ' EL BUFFER HACIA EL DISCO ES BUF$
1230 PUT 1, NR
1240 RETURN
2000 ' ** SUBRUTINA LECTURA DATOS **
2010 GET 1,NR
2020  C$=BUF$     ' C$ ES' UN BUFFER INTEMEDIO QUE ALBERGA LOS DATOS
2025 '             TAL COMO SON LEIDOS.
2030 PRINT  "C$= ";C$
2040 NE=4*LN              ' LONGITUD EN BYTES DE LA PARTE OCUPADA POR LOS
2050 '                     DATOS NUMERICOS = 4 CAMPOS DE LONGITUD LN
2060 K=0                  ' PUNTERO A LAS MATRICES NUMERICAS V(4)
2070 FOR I=1 TO NE STEP LN       ' INICIO DE CADA NUMERICO
2080 K=K+1            ' CONTADOR PARA LA MATRIZ V(4)
2090 V(K)=VAL(MID$(C$,I,LN))
2100 NEXT I
2110 NK=2*LS              'NK = ESPACIO TOTAL OCUPADO POR LAS 2 CADENAS A$(2)
2120 D$=RIGHT$(C$,NK)  ' D$ CONTIENE LA ULTIMA PARTE DEL RECORD, O SEA
2130 '                     LOS CARACTERES CORRESPONDIENTES A LAS DOS CADENAS
2140 A$(1)=LEFT$(D$,LS)
2150 A$(2)=RIGHT$(D$,LS)
2160 RETURN


VALORES LEIDOS:
NUMERICOS:
 1111
 2222
 3333
 4444

CADENAS:
 AAAAAAAAA
 BBBBBBBBB
```


ESQUEMAS LOGICOS DE EXTRACCION DE LOS DATOS



otro introducido por teclado podrán existir desajustes y, por tanto, los dos campos podrán resultar diferentes a pesar de contener los mismos valores. Supongamos por ejemplo que en un file del disco se ha previsto un campo para la introducción de un nombre de ciudad de 10 caracteres de longitud; en la fase de introducción, el dato introducido ha sido M. En este caso, el campo contiene, además de los seis caracteres de un dato, cuatro espacios. Para buscar todos los records que contienen el dato M debe prepararse una cadena de comparación con las mismas características: el dato debe alinearse a la izquierda y los espacios restantes deben dejarse en blanco.

La elección del alineado a la izquierda (o a la derecha) depende del alineado utilizado en la fase de escritura en el disco. La práctica normal consiste en alinear las cadenas de caracteres a la izquierda y las cadenas que representan valores numéricos a la derecha, pero nada impide adoptar una convención diferente, siempre que se respete en todas las operaciones en el file.

En el ejemplo anterior, cada ambigüedad puede resolverse preparando una cadena de longitud igual y transfiriendo, adecuadamente alineado, el dato leído por teclado. El campo así formado podrá utilizarse como término de comparación para realizar la elección de los records. En la pág. 660 se ha representado el diagrama de flujo de uso general que desarrolla las funciones descritas.

Comandos y funciones particulares

Los sistemas operativos prevén un límite máximo del número de files que pueden abrirse al mismo tiempo (por ejemplo para el CP/M son 3); las eventuales variaciones de dicho límite deben comunicarse al sistema en el momento de la carga del intérprete Basic.

En este caso, normalmente, el formato del comando de carga del intérprete tiene la sintaxis:

MBASIC / F:N

El símbolo / indica que lo que sigue es una opción; la letra F indica que se desea abrir un número de files superior al límite normal y el valor numérico N indica la cantidad.
Por ejemplo, el comando

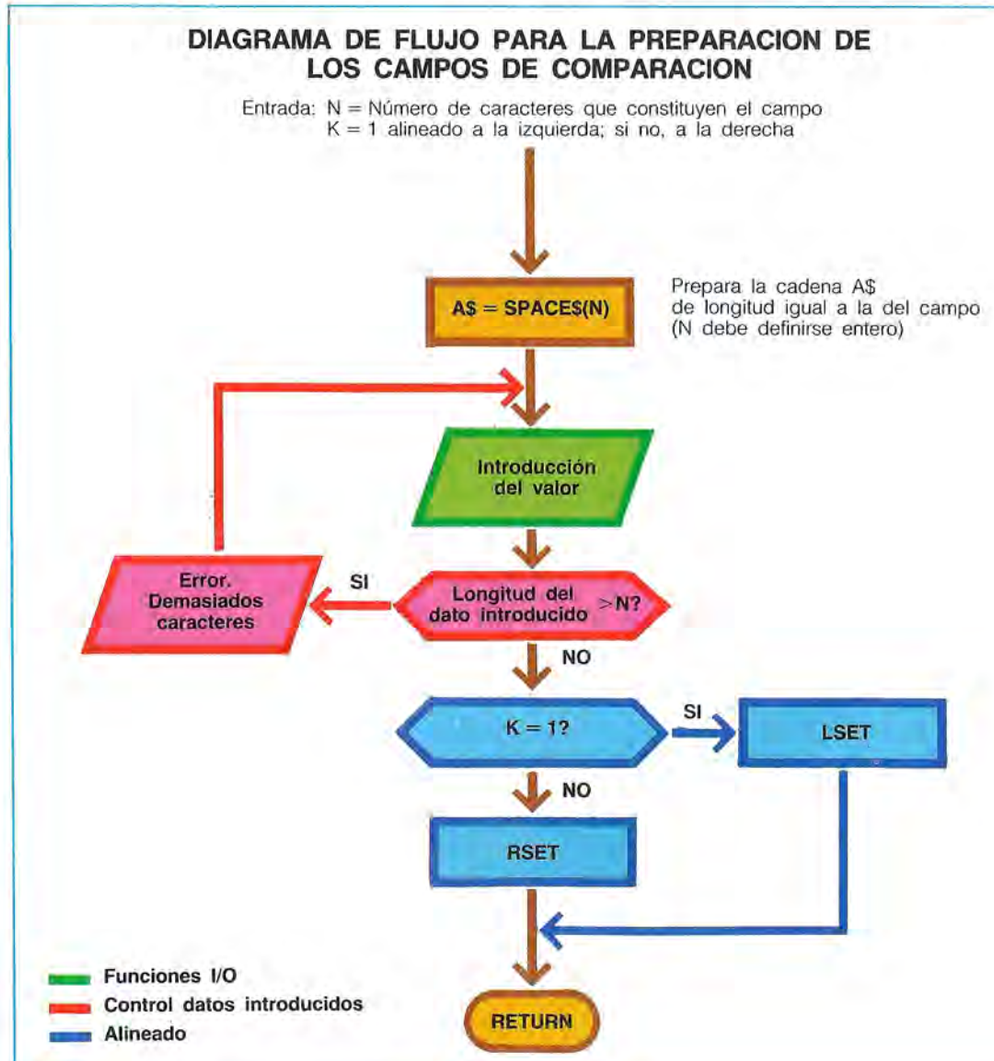
MBASIC / F:5

predispone el intérprete para utilizar 5 files al mismo tiempo. Se recuerda que para cada file hay reservada un área de memoria que contiene su descripción.
En el sistema operativo CP/M, esta área tiene una extensión de 166 bytes para cada file. En el caso anteriormente considerado se tiene una ocupación total de 5 × 166 = 830 bytes reservada a los bloques de descripción de los files*.
La otra opción correspondiente a los files es la siguiente:

/ S:N

* Recordamos al lector que los bloques de descripción de los files normalmente se indican con las siglas FDB (File Description Block).

DIAGRAMA DE FLUJO PARA LA PREPARACION DE LOS CAMPOS DE COMPARACION



donde el valor N indica el valor de la máxima longitud de los records expresada en bytes.
Si no se especifica, se asume la misma longitud por omisión igual a 128 bytes (especificar una longitud máxima no limita el uso de los valores inferiores de la instrucción FIELD).
Finalmente recordamos que en el sistema operativo CP/M, el número máximo de records lógicos que puede utilizarse es de 32767, salvo en el caso en que se defina una longitud del record igual a 256 bytes utilizando la opción S: en tal caso, el direccionamiento máximo se convierte en 8 Mbytes.
En algunos intérpretes Basic se ha previsto una función particular para la lectura de las longitudes de los files (en sectores). En el Basic 80, esta función es

LOF(N)

y restituye el número de records de la última extensión del file identificado con el número N. Si el file no tiene más de una extensión, este número coincide con la longitud real del file.

Los archivos multivolumen

La capacidad de memorización de los diskettes (floppy-disks) varía aproximadamente entre 160.000 y 1.000.000 de bytes, según las dimensiones y el tipo. En algunas aplicaciones puede ser necesario disponer de una mayor capacidad que la que ofrece el diskette.
La mejor solución consiste entonces en utilizar componentes hardware más adecuados, como por ejemplo los discos fijos. En algunas máquinas, los discos fijos permiten memorizar más de 20 Mbytes, aunque los tipos más normales tienen capacidades de 5 y 10 Mbytes. Además, los discos fijos ofrecen la ventaja de una mayor velocidad en las funciones I/O*.
Sin embargo, esta solución, óptima desde el punto de vista técnico, no siempre puede adoptarse, y en estos casos debe suplirse con software mediante los adecuados programas, la carencia del hardware. El método más utilizado consiste en dividir el archivo en varias partes, de manera que cada una pueda estar contenida sin dificultad en un diskette. El programa de elaboración de los datos evidentemente deberá tener en cuenta esta subdivisión, de modo que pueda guiar al usuario en la elección del diskette adecuado. En la fase de elaboración, el programa deberá establecer sobre qué diskette se encuentran los datos pedidos en aquel momento, así como informar al usuario y, finalmente, esperar que se complete el montaje del diskette deseado.
Un archivo de este tipo se llama **archivo multivolumen** por la característica de estar dividido en varias partes, cada una de las cuales se denomina **volumen**. Esta subdivisión de los datos, que en los ordenadores personales y microordenadores suple a menudo la capacidad de memorización, también puede justificarse en base a una elección lógica. No siempre es conveniente memorizar los datos en un file único; a veces puede ser ventajoso dividir el archivo en varios volúmenes, incluso si el disco puede contenerlo totalmente.
En estos casos se realiza una división de tipo lógico: se tiene una separación en volúmenes, pero físicamente todos los volúmenes residen en el mismo disco. Ordenando el archivo de esta manera, el programa que lo gestiona no debe prever las esperas para el montaje de los discos necesarios.
La gestión de los archivos multivolumen (con más volúmenes **lógicos** o **físicos**) se realiza memorizando para cada volumen cuáles son los datos que residen en él; por tanto, desde el valor del dato puede irse al número (o nombre) del volumen.
Un programa de gestión estructurado así puede convertirse en extremadamente complejo en el momento de trabajar con files índice, mientras que es relativamente sencillo en el caso de los files directos. La lógica de gestión más inmediata de un file directo prevé la utilización del número de record como clave de acceso a los datos: el primer dato introducido tendrá un número de record 1 y una clave de acceso 1, el segundo 2, y así sucesivamente. Sin embargo, se recuerda que hablar de un número del record como clave de acceso es una terminología impropia; la clave de cada record normalmente se obtiene sometiendo los datos a una elaboración también compleja. La gestión de un archivo multivolumen se reduce a memorizar en un file separado el nombre de los volúmenes (es decir, de los diskettes) y los extremos de los records contenidos en ellos. Este file, que conserva en memoria el desarrollo del archivo, constituye

* El intercambio de datos con los discos fijos se realiza en DMA y, por tanto, con velocidad muy elevada; en cambio, para los diskettes, esta técnica normalmente no se utiliza.

una especie de directorio, y normalmente está compuesto de pocos records (uno para cada diskette) que deben tenerse constantemente actualizados.

Gestión de los archivos en los grandes sistemas

En los sistemas más grandes, a partir de la gama de los minicalculadores, existen programas de utilización general para la gestión de las bases de datos que evitan al programador gran parte del trabajo de detalle. Por ejemplo, estos programas gestionan automáticamente las cadenas de punteros que definen la estructura de los datos y contienen las rutinas de sort.
Los programas de gestión pueden dividirse en dos categorías: los propiamente fijos y los ligados a un lenguaje huésped.
Los programas de gestión de datos propiamente fijos constituyen un lenguaje verdadero y particular de alto nivel, caracterizado por instrucciones similares a las típicas de los lenguajes de programación y, además, con instrucciones específicamente dedicadas a la gestión de los archivos. Los programas ligados a un lenguaje huésped contienen una serie de instrucciones que implantan las posibilidades de este último. En los grandes sistemas, los principales lenguajes huésped son el Cobol y el Fortran, y las instrucciones inherentes a la gestión de los archivos se traducen primero a la forma prevista por el lenguaje huésped y ejecutadas así.
Las principales funciones previstas en estos programas son las siguientes:

STORE	escribe un record y crea todas las eventuales cadenas de punteros para otros datos
FIND	busca un grupo de datos en base a los datos proporcionados por el usuario
MODIFY	modifica uno o más datos
ERASE	cancela un record y, en consecuencia, modifica las cadenas de los punteros que apuntan al mismo

Además de estas instrucciones principales (cuya sintaxis puede variar en función del tipo de máquina) hay previstas normalmente las frases condicionales como IF, los operadores lógicos (AND, OR, etc.) y los operadores aritméticos (+, –, /, *), con los mismos significados que tienen en el lenguaje Basic. Muchos lenguajes orientados a la gestión de las bases de datos contienen «subsistemas» para la interrogación que permiten la búsqueda de los datos incluso a personas que no tienen ningún conocimiento de programación. Las instrucciones que deben suministrarse son muy sencillas y utilizan frases que tienen un significado específico. Por ejemplo, la fase de interrogación de un almacén utilizando el sistema MDQS (Management Data Query System) puede tener la siguiente forma:

```
DISPLAY MERCADERIA TIPO CANTIDAD
IF CODIGO = "A"
LET CANTIDAD = ENTRADA – SALIDA
END
```

El empleo de este tipo de lenguajes sólo requiere el conocimiento de pocas palabras en el idioma inglés, muy fáciles de recordar porque activan la función indicada por el significado literal de la palabra: DISPLAY = presentación, visualización; IF = si, etc. En los microordenadores y ordenadores personales ya empiezan a estar disponibles sistemas análogos y, dada la importancia que tienen en los sistemas de automatización del trabajo de taller, se tratarán por separado más adelante.

La gestión de un almacén

La gestión de un almacén es una de las aplicaciones más típicas del calculador, ya que las funciones a realizar tienen carácter repetitivo e implican una elevada cantidad de datos.
La preparación de los programas de gestión puede presentar notables dificultades debido a la necesidad de correlacionar los datos con otros procedimientos y de desarrollar funciones particulares de carácter económico o fiscal.
En las aplicaciones de una cierta complejidad no resulta ventajoso proceder a la escritura de un programa dedicado; en cambio, conviene utilizar uno de los procedimientos existentes, personalizándolo eventualmente según las propias necesidades.
Dada la complejidad del tema, a continuación se exponen únicamente las principales problemáticas relacionadas con la gestión de un almacén. Los principales problemas que deben resolverse con un procedimiento de gestión pueden clasificarse así:

– Revaluación de las existencias e inventario
– Desglose arbóreo
– Simulación
– Enlace con otros procedimientos

Revaluación de las existencias e inventario. Las operaciones de carga de mercancías pueden producirse en tiempos sucesivos y, por tanto, con costos diferentes para la misma mercancía en función de la fecha de carga. En el momento de la salida de la mercancía debe encontrarse un método de atribución del costo que, por ejemplo, puede ayudarse con la fecha de entrada. Así, un cierto artículo tendrá un dato de costo y, por tanto, un determinado precio en función de la fecha de entrada. Sin embargo, esto repercute en un consistente aumento del espacio de memoria necesario. Además, en al-

TEST 19

1 / ¿Cuáles de las siguientes aseveraciones son verdaderas y cuáles falsas?

Un file secuencial:
a) no permite añadir records
b) tiene una velocidad de acceso elevada
c) ocupa menor espacio que un file directo
Un file directo:
d) no permite extensiones
e) puede utilizarse indiferentemente tanto en lectura como en escritura
f) utiliza la función EOF(N)

2 / ¿Qué sucede al introducir las siguientes instrucciones si en la unidad de disco A no existe el file PRUEBA?

a) OPEN "I",1,"A:PRUEBA"
b) OPEN "R",1,"A:PRUEBA", 120

3 / ¿Con qué instrucción puede determinarse el número del último record utilizado durante la ejecución de una instrucción PUT o GET?

4 / Escribir una subrutina para la introducción de los siguientes datos en un archivo:

C% = código numérico entero comprendido entre 1 y 99
D$ = descripción de 20 caracteres
QT% = cantidad, número entero
C = costo

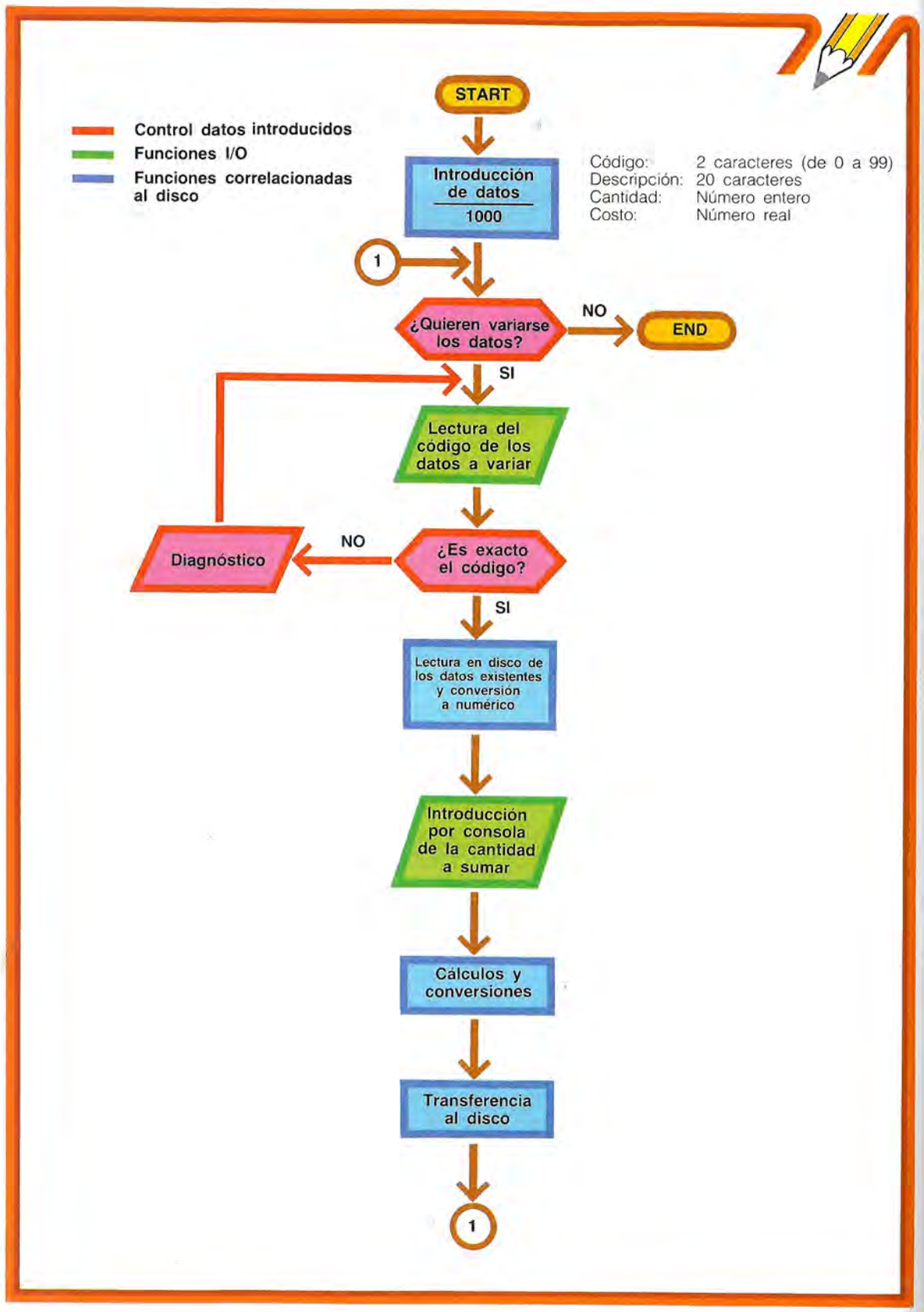
El file tiene el nombre PRUEBA y reside en el disco montado en la unidad A.
La rutina debe empezar con el número de línea 1000.

5 / Completar la rutina obtenida en el punto anterior añadiendo las instrucciones para

– releer un dato cualquiera introducido previamente
– añadir a la cantidad QT% preexistente una nueva cantidad introducida por consola
– memorizar las variaciones en el disco (en la misma posición)

El diagrama de flujo del procedimiento puede verse en la pág. 664

Las soluciones, en la pág. 671.



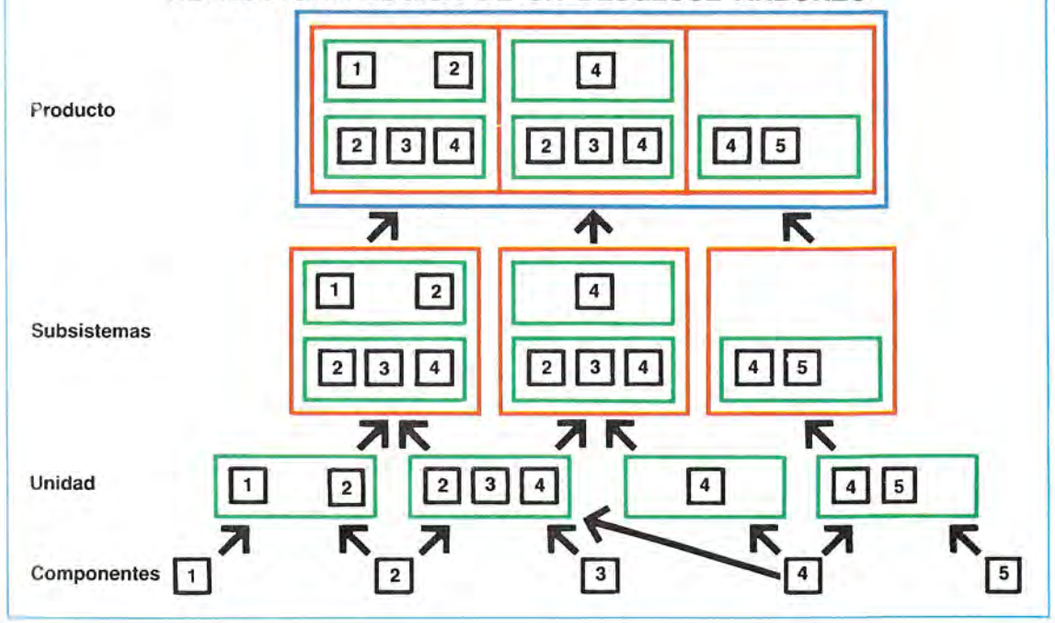
gunos casos en que el almacén debe estar sujeto a normas de carácter fiscal, la lógica de atribución del valor de las mercancías no puede ser arbitraria.

Desglose arbóreo. En las industrias manufactureras, la gestión del almacén está estrechamente relacionada con el **desglose arbóreo**, es decir, la relación de componentes necesarios para la construcción del producto.
En general, un producto está constituido por diversos subsistemas, cada subsistema por un cierto número de unidades y la unidad por un cierto número de componentes. Así se crea una escala jerárquica en la que se especifican por cada producto el número y el tipo de los elementos necesarios en el nivel inferior y, viceversa, dado un componente debe poderse volver a subir, a través de la escala jerárquica, a todos los niveles superiores en los que es necesario el componente.
Los diversos lazos de dependencia que se forman, como puede comprenderse fácilmente, generan notables dificultades en la escritura de estos programas. Abajo se ha representado una estructura de cuatro niveles. Siguiendo las coloraciones puede tenerse una idea de la complejidad de los punteros de gestión para mantener eficiente la estructura.

Simulación. Disponiendo de un desglose arbóreo pueden desarrollarse elaboraciones especiales que simulan el efecto de las variaciones de los costos o de la falta de existencias de uno o más componentes. Siguiendo las cadenas lógicas de dependencia, el calculador puede determinar sobre qué productos y en qué medida repercuten en ellos las diversas causas.
La simulación de determinados sucesos y el estudio de sus efectos en la programación del interior de una estructura constituyen un medio acelerado para la elección de las políticas de gestión empresarial. Esta metodología reviste tal importancia, que ha inducido a varias empresas de software a producir programas adaptados a esta finalidad, de empleo generalizado y alojables en microordenadores y ordenadores personales.

Enlace con otros procedimientos. La gestión administrativa de una empresa, incluso de modestas dimensiones, implica un denso intercambio de datos entre varios departamentos. Cuanto más automático y rápido es este intercambio, más ágil es la gestión de todo el conjunto. Inmediatamente se intuye la forma en que la gestión del almacén influirá en la determinación de los costos, en los plazos de entrega y en los balances. En otras palabras, los diversos departa-

ESTRUCTURA LOGICA DE UN DESGLOSE ARBOREO



mentos deben utilizar datos comunes. En las empresas de mayores dimensiones, algunos aspectos del problema se resuelven con la instalación de un gran ordenador que contiene los archivos centralizados. Sin embargo, incluso en este caso, no se resuelve totalmente el problema, ya que no siempre el usuario final tiene una formación especializada que le permita un fácil acceso al procedimiento generalizado y que, muchas veces, debe desarrollar determinadas elaboraciones particulares del departamento y no previstas en los programas generales.
La consecuencia de esta dificultad es el mantenimiento de la elaboración manual de las informaciones a nivel de cada departamento y del intercambio de datos sobre soporte de papel, voluminoso y poco rápido.
La introducción de los microordenadores y ordenadores personales puede resolver ágilmente el problema ofreciendo una solución de bajo costo y, de otro lado, fácilmente integrable al trabajo de oficina.
El uso de estas máquinas como usuario final no requiere conocimientos particulares y puede aprenderse en pocos días. Además, con las oportunas redes de conexión, las diversas unidades pueden dialogar entre sí o con el ordenador central e intercambiar información en tiempo real, o sea en el mismo momento en que se produce.
La evolución natural del programador de estas máquinas será, en el futuro inmediato, la de especialista de sistemas que, conociendo las capacidades del hardware y las posibilidades de los diversos paquetes de aplicación, podrá ayudar a los usuarios en las elecciones y en las instalaciones.

Unidad de disco en un centro de procesos.

J. Pickerell/Marka

Ejemplo de aplicación: la gestión de una nómina

Para resumir los conceptos expuestos acerca de las instrucciones de gestión de los files, queremos ahora exponer el método para la creación de un programa de gestión de una nómina. El tema se trata del modo más general para permitir el uso del programa también para otras aplicaciones.
Sea cual sea el archivo a gestionar y sea cual sea la aplicación, las funciones a realizar son las siguientes:

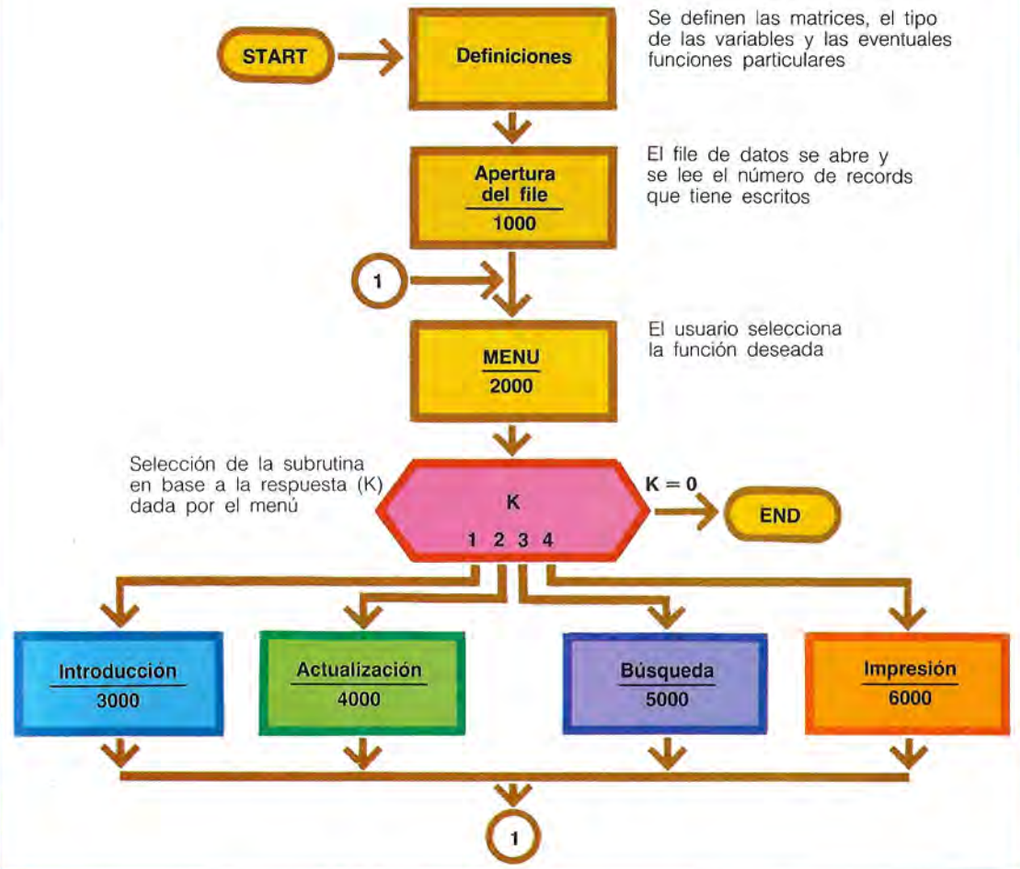
- Introducción de nuevos datos
- Corrección de los datos existentes (actualización)
- Búsqueda de un dato en base a algunos parámetros
- Impresión

Las cuatro funciones deben ser previstas en el menú a través del que se gestionará el archivo y se desarrollarán otras tantas subrutinas.
En la página de enfrente se ha representado el diagrama de flujo de primer nivel del programa: el main sólo contiene las definiciones (matrices, constantes, etc.) y las llamadas a las subrutinas; todas las funciones del programa se desarrollan en las diversas subrutinas.

Apertura del file (subrutina 1000)

Esta subrutina abre el file, asigna los campos y lee el número del último record ocupado; el número del primer record disponible viene dado por este valor más 1. La mejor técnica para gestionar el puntero al primer record disponible en escritura consiste en crear un file que tenga la máxima longitud prevista y utilizar el último record del file para memorizar el número de datos contenidos en el propio file. El último record deberá actualizarse a medida que se introduzcan

GESTION DE DATOS EN EL DISCO. DIAGRAMA DE FLUJO DE PRIMER NIVEL



los datos. En la pág. 668 se ha representado el esquema lógico del método.
En esta aplicación, el file deberá contener los campos que a continuación se mencionan:

Apellido	= 20 caracteres
Calle	= 20 caracteres
Teléfono	= 12 caracteres
Ciudad	= 20 caracteres
Longitud del record	= 72 caracteres

La creación del file puede obtenerse de forma inmediata introduciendo por teclado la instrucción:

```
OPEN "R", "1", "A:DATOS",72
```

Al ejecutarla, el sistema escribe el nombre del file en el directorio y le asigna una longitud inicial. Haciendo igual a 100 el número máximo de los grupos de datos a memorizar, debe escribirse en el record 101 (el último) el valor 0 ya que, inicialmente, el file no contiene ningún dato y el primer record disponible es el número 1 (0 + 1 = 1).
La creación del file y la inicialización del último record pueden obtenerse con un **programa de utilidad** de uso general, cuyo listado puede verse en la pág. 668, abajo. Enviado a ejecución más veces este programa también es posible variar la longitud del file.
Una vez preparado el procedimiento de creación y de inicialización del file puede pasarse a la escritura de la subrutina 1000, que deberá

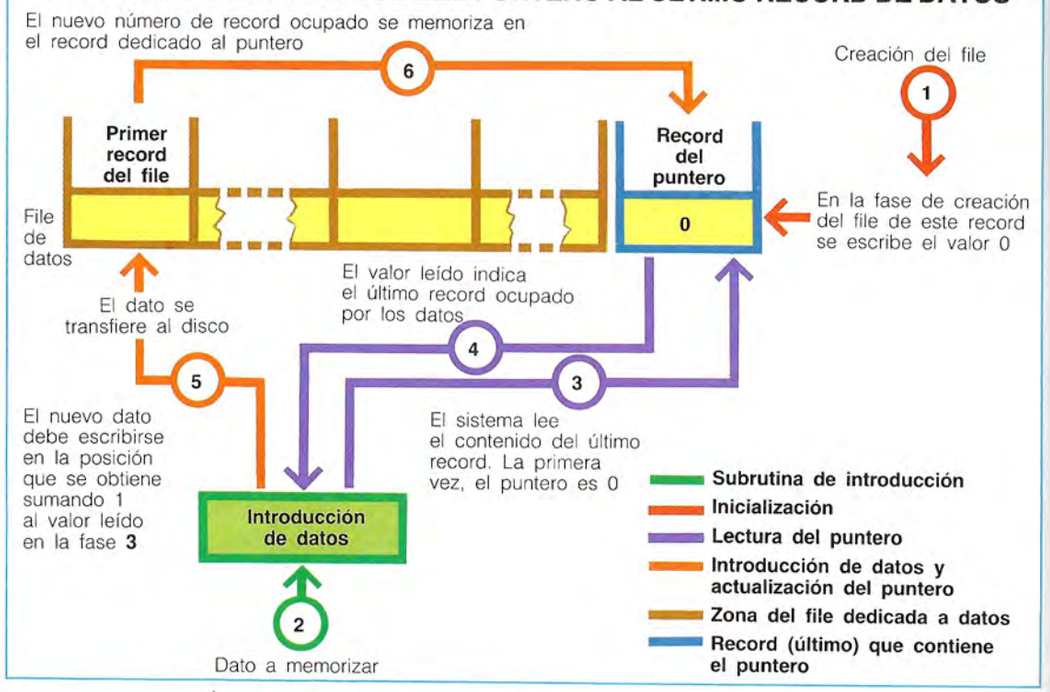
desarrollar las siguientes funciones:

– Apertura del file
– Lectura del último record

La estructura de esta subrutina es muy sencilla; el correspondiente listado se ha representado al lado. Como puede verse, la subrutina es parametrizada, y todos los parámetros (nombre del file, longitud, etc.) se los entrega el main. De esta manera puede utilizarse para cualquier aplicación cambiando simplemente los valores de los parámetros.

Debe indicarse que la instrucción 1090 (apertura en modalidad "I" y cierre del file) sólo tiene la

GESTION DE LOS FILES. USO DEL PUNTERO AL ULTIMO RECORD DE DATOS



CREACION DEL FILE E INICIALIZACION DEL ULTIMO RECORD

```
10 '  ** PROGRAMA DE UTILIDAD PARA LA CREACION DEL FILE
20 '  Y PARA LA INICIALIZACION DEL ULTIMO RECORD ***
30 '
40 '                          FILE = CRESCR
50 INPUT "NOMBRE DEL FILE"; NOMBRE$
60 INPUT "EN QUE UNIDAD DE DISCO (A:/B:) ";D$
70 IF D$<>"A:" AND D$<>"B:"GOTO 60
80 NM$=D$+NOMBRE$
90 INPUT "LONGITUD DEL RECORD (N. DE CARACTERES)";L
100 IF L>128 GOTO 90
110 OPEN "R",1,NM$,L
120 FIELD 1,L AS A$
130 INPUT "CUANTOS RECORDS DEBE CONTENER ";NR
140 MX=NR+1
150 B$="0"  C$=CADEN$ (L,B$)      PREPARA C$ = "00000....0"
160 LSET A$=C$          ESCRIBE TODOS LOS SIMBOLOS 0
170 PUT 1,MX
180 PRINT "EL FILE ESTA PREPARADO"
190 PRINT "LONGITUD:";NR
200 PRINT "PUNTERO EN POSICION: ";MX
210 END
```


SUBRUTINA DE APERTURA DEL FILE

```
1000 ' ** SUBRUTINA DE APERTURA DEL FILE **
1010 '
1020 '           FILE = APEFIL
1030 ' ENTRADAS
1040 '              NM$ = UNIDAD DE NOMBRE DEL FILE, EJEMPLO A:DATOS
1050 '                L = LONGITUD DEL RECORD (EN CARACTERES)
1060 '               NR = LONGITUD DEL FILE (EN RECORDS)
1070 '              BF$ = BUFFER ASOCIADO AL FILE
1080 '  **********
1090 OPEN "1",1,NM$:CLOSE 1
1100 OPEN "R",1,NM$,L
1110 FIELD 1,L AS BF$
1120 MX=NR+1
1130 GET 1, MX    ' LECTURA DEL RECORD QUE CONTIENE EL PUNTERO
1140 LSET A$=BF$ ' EL VALOR SE TRANSFIERE A A$
1150 B$=LEFT$(A$,3)  ' SOLO SE UTILIZAN LOS 3 PRIMEROS CARACTERES
1160 MAX=VAL (B$)        ' EL VALOR MAXIMO QUE PUEDE UTILIZARSE ES 999
1070 '
1180 ' ** EN LA SALIDA MAX  CONTIENE EL VALOR ESCRITO EN EL RECORD DEL
1190 '    PUNTERO, O SEA EL NUMERO ACUMULADO DEL ULTIMO RECORD
1200 '    OCUPADO. EL PRIMERO A DISPOSICION LO DA ESTE VALOR + 1
1210 RETURN
```

SUBRUTINA DE INTRODUCCION DE DATOS

```
3000 ' **INTRODUCCION **
3010 '   ENTRADAS:
3020 '                MAX = ULTIMO RECORD OCUPADO
3030 '                 NR = LONGITUD FILE
3040 ' LA PARTE DE INTRODUCCION DE DATOS ES ESPECIFICA DE LA APLICACION
3042 ' PARTICULAR Y SE SUSTITUIRA EN FUNCION DE LAS DIVERSAS
3044 ' EXIGENCIAS
3050 ' ** LECTURA DE DATOS POR VIDEO **
3060 INPUT "APELLIDO";C$
3070 INPUT "CALLE"; V$
3080 INPUT "TELEFONO "; T$
3090 INPUT "CIUDAD";Y$
3100 '  **FORMA EL RECORD **
3110 A1$=SPACE$(20)           ' PRIMER CAMPO  : APELLIDO
3120 A2$=SPACE$(20)           ' SEGUNDO CAMPO : CALLE
3130 A3$=SPACE$(12)           ' TERCER CAMPO  : TELEFONO
3140 A4$=SPACE$(20)           ' CUARTO CAMPO  : CIUDAD
3150 LSET A1$=C$              ' COPIA LA CADENA DE LA VARIABLE DE LA DERECHA
3160 LSET A2$=V$              ' EN LA DE LA IZQUIERDA PREVIO ALINEADO
3170 LSET A3$=T$              ' DE LA MISMA CADENA DE LA IZQUIERDA
3180 LSET A4$=Y$
3190 ' *** LA INSTRUCCION (LSET) TRUNCA LOS CARACTERES QUE SOBRAN
3192 '     DEL CAMPO DE LLEGADA
3200 'A$=A1$+A2$+A3$+A4$
3210 'LSET BF$=A$
3220 M=MAX+1
3230 PUT 1,M                  ' ESCRIBE LOS DATOS
3240 IF KS=1 THEN RETURN      ' ESTA INSTRUCCION SE EXPLICA EN LA
3250 '                        SUBRUTINA (4000)
3260 MAX=MAX+1
3270 A$=SPACE$(3)             ' EN EL ULTIMO RECORD, LOS TRES PRIMEROS CARAC-
3280 '                        TERES ESTAN RESERVADOS AL PUNTERO.
3290 B$=STR$(MAX)
3300 RSET A$=B$
3310 A$=A$+SPACE$(69)         ' COMPLETA EL RECORD CON ESPACIOS EN BLANCO
3320 LSET BF$=A$
3330 N=NR+1                   ' N = RECORD DEL PUNTERO
3340 PUT 1,N
3350 RETURN
```

finalidad de controlar que el file exista: si se abriese en modalidad "R" un file no creado anteriormente, se crearía uno nuevo sin inicializar.

Menú (subrutina 2000)

Para esta subrutina puede hacerse referencia a una de las subrutinas ya presentadas cuando se trató el tema del menú desde el punto de vista general. La única variación consiste en la escritura que debe preverse (1-Introducción, 2-Actualización, etc.).

Introducción (subrutina 3000)

Las funciones que realiza son las siguientes:

- Lectura de los datos por consola
- Su escritura en el primer record libre
- Actualización del último record para tener en cuenta el nuevo dato introducido

El listado de la subrutina se ha representado en la pág. 669 (abajo).

Actualización (subrutina 4000)

Debe realizar las siguientes funciones:

- Lectura por vídeo del número del record a actualizar
- Lectura de datos del disco
- Presentación de los datos existentes
- Lectura de los nuevos valores (variación en los datos existentes)
- Memorización de las modificaciones en el mismo record

El listado de la subrutina puede verse aquí abajo. Para escribirla, puede utilizarse parcialmente la subrutina de introducción.
La diferencia existente entre las fases de introducción y de actualización se debe a las diferentes posiciones en que se escriben los datos: en la fase de introducción, cada dato se posiciona en un nuevo record, mientras que en la fase de actualización, el dato actualizado debe escribirse en el mismo record que ha sido leído, al tiempo que el puntero a los datos no debe ser incrementado.
Estas funciones particulares pueden obtenerse de la subrutina 4000 alzando un flag (KS en el listado) que indica a la 3000 que no actualice el puntero a los datos.
En este caso particular, el uso del flag para modificar las funciones realizadas por la subrutina 3000 no aporta ninguna ventaja significativa puesto que las instrucciones necesarias podrían volverse a escribir directamente en la subrutina 4000 (de la línea 3050 a la 3230). El aumento de ocupación de memoria no tiene importancia y se cuenta con la ventaja de una mayor claridad del programa.
El uso del flag sólo se ha propuesto para mostrar una posible metodología.

SUBRUTINA DE ACTUALIZACION

```
4000 ' ** ACTUALIZACION **
4010 INPUT "NUMERO DEL RECORD A ACTUALIZAR ";N
4020 IF N>NR GOTO 4010
4030 GET 1,N
4040 LSET A$=BF$
4050 ' **SEPARACION DE LOS CAMPOS
4060 C$=LEFT$(A$,20)
4070 V$=MID$(A$,21,20)
4080 T$=MID$(A$,40,12)
4090 Y$=MID$(A$,53,20)
4100 PRINT "LOS DATOS CONTENIDOS EN EL RECORD ";N;"SON:"
4110 PRINT C$: PRINT V$:PRINT T$: PRINT Y$
4120 ' ** PARA LA INTRODUCCION DE LAS VARIACIONES EN LOS DATOS
4130 '    PUEDE UTILIZARSE PARTE DE LA SUBRUTINA
4140 '    (3000) LLAMANDOLA CON KS=1 PARA EVITAR QUE
4150 '    LAS CORRECCIONES SE CONSIDEREN COMO NUEVOS DATOS
4160 MS=MAX          ' MEMORIZA (MAX) EN LA VARIABLE (MS)
4170 MAX=N           ' LO UTILIZA PARA COMUNICAR CON
4180 '                 LA SUBRUTINA (3000)
4190 KS=1            ' FLAG PARA LA (3000)
4200 GOSUB 3000
4210 MAX=MS          ' RESTABLECE EL NUEVO VALOR DE (MAX)
4220 RETURN
```


SOLUCIONES DEL TEST 19

1 / a) es verdadera con la siguiente precisión: las adiciones son posibles utilizando un file de apoyo, y no directamente.
b), c), d) son falsas; e) es verdadera; f) es falsa: la función EOF (N) se utiliza para indicar el final de un file secuencial.

2 / a) genera un error y detiene la ejecución del programa.
b) crea un file.

3 / Debe utilizarse la función LOC(N).

4-5 / El listado del programa que comprende las soluciones de los puntos 4 y 5 se presenta aquí abajo; sólo se trata de uno de los posibles modos de realizar las funciones requeridas. La fase de realización del menú (escritura de las instrucciones) de un programa, en general es muy subjetiva. La forma de los programas y las instrucciones o las funciones utilizadas varían mucho según los hábitos del programador.

```
10 ' **SOLUCION AL TEST
20 '    FILE : TEST1
100 ' ** MAIN **
105 OPTION BASE 1
110 PRINT" Seleccionar la función deseada"
120 PRINT" 1 = Introducción    2 = Lectura    3 = Variaciones"
130 INPUT K
140 IF K<1 OR K>3 THEN GOSUB 2000
150 IF KR<> 0 GOTO 110 ' NUEVA SELECCION A CONTINUACION DEL ERROR
160 GOSUB 1000
165 MODO=0
170 INPUT" Continua (SI/NO) ";RESP$
180 IF RESP$="SI" THEN PRINT CHR$(27)+"+":GOTO 110
190 END
1000 ' ** SUBRUTINA DE GESTION DE DATOS **
1010 ON K GOSUB 1020,1280,1440
1015 RETURN
1020 '
1025 IF MODO=1 GOTO 1220
1030 INPUT" Código :";C %
1040 IF C %<1 OR C %>99 THEN GOSUB 2000
1050 IF KR <> 0 GOTO 1030
1060 '
1070 INPUT" Descripción";D$
1080 IF LEN (D$)20 THEN GOSUB 2000    ' ERROR EN LA LONGITUD
1090 IF KR <>0 GOTO 1030
1100 IF LEN (D$)=20 GOTO 1140
1110 N % =20-LEN(D$)                  ' Completa la cadena si se han
1120                                  ' introducido menos de 20 caracteres
1130 A$=SPACE$(N % )
1140 INPUT " Cantidad  :  ";QT %
1150 INPUT "Costo      : ";C
1160 '
1210 '
1220 OPEN "R",1,"PRUEBA",28      ' EL FILE TIENE EL NOMBRE   PRUEBA
1230 FIELD 1,2 AS W1$,2 AS W2$,4 AS W3$,20 AS W4$
1232 LSET W1$=MKI$(C%)
1234 LSET W2$=MKI$(QT%)
1236 LSET W3$=MKS$(C)
1238 LSET W4$=D$
1250 PUT 1, C %
1260 CLOSE 1
```

```
1270 RETURN
1280 '
1290 PRINT "Lectura datos"
1300 INPUT "Código artículo ";C %
1310 IF C % <1 OR C % >99 THEN GOSUB 2000
1320 IF KR <> 0  GOTO 1300
1330 OPEN "R",1 "PRUEBA",28
1335 FIELD 1,2 AS W1$,2 AS W2$,4 AS W3$,20 AS W4$
1340 GET 1,C %
1342 C %=CVI (W1$)
1344 QT %=CVI (W2$)
1346 C=CVS (W3$)
1348 D$=W4$
1370 PRINT "   ** DATOS LEIDOS ** "
1380 PRINT " CODIGO      = ";C %
1385 PRINT " DESCRIPCION = ";D$
1390 PRINT " CANTIDAD    = ";QT %
1400 PRINT " COSTO       = ";C
1410 CLOSE 1
1420 RETURN
1430 '
1440 ' LECTURA DATOS EXISTENTES
1445 MODO=1
1450 GOSUB 1280
1460 '
1470 ' LECTURA NUEVOS VALORES
1480 INPUT "Cantidad a añadir ";CA %
1490 QT%=QT % +CA %
1495 ' ESCRITURA NUEVO VALOR
1500 GOSUB 1020
1510 RETURN
2000 ' ** SUBRUTINA DE ERROR **
2010 '
2020 KR=1                ' Flag de error
2030 PRINT " **** ERROR ****"
2040 PRINT "Para continuar, introducir un carácter cualquiera": S$=INPUT$(1)
2050 PRINT CHR$(27)+"+"
2060 RETURN
```

Búsqueda (subrutina 5000)

En general, la escritura de las subrutinas de búsqueda depende estrechamente de la de los datos y de su cantidad. Si el archivo (file de datos) es de grandes dimensiones, en primer lugar deberá ordenarse de acuerdo con la clave con la que se va a realizar la búsqueda y después desarrollar una búsqueda con ruptura de código (siempre suponiendo que no exista un índice que apunta directamente a los datos). En las aplicaciones en microordenador y ordenador personal, el número de los datos nunca es particularmente importante; en este caso conviene examinar todos los files sin imponer a priori ninguna ordenación y extraer los records que contienen los valores a medida que se presentan. Con esta técnica se emplea más tiempo del necesario, pero la diferencia en términos de tiempo no justifica las complicaciones que se tendrían en la escritura del programa. Para tener un orden de magnitud, considérese que en pocos minutos pueden leerse varios centenares de records, mientras que el eventual ahorro de tiempo es del orden de minutos. La lógica y, por tanto, la complejidad de las rutinas de búsqueda dependen de los **grados de libertad** que quieran imponerse, o sea del número de parámetros que pueden utilizarse como elementos de selección. Por ejemplo, en el file examinado puede utilizarse como elemento de selección sólo el apellido, en cuyo caso deberán confrontarse únicamente los primeros 20 bits del record (el apellido es el primer campo), o bien también puede considerarse la selección por ciudad: en este caso también deberán considerarse los últimos 20 bytes. De este modo deberá escribirse una rutina para cualquier posible lógica de selección.

Esta conclusión muestra claramente la necesidad de parametrizar la rutina de búsqueda. Puede conseguirse un buen nivel de parametrización estructurando el programa de manera que éste presente al usuario el índice de los campos que constituyen el record.

A su vez, el usuario podrá seleccionar uno de los campos como clave de búsqueda, introduciendo el dato a buscar.

La rutina 5000 constituye un ejemplo de utilización de esta lógica. En este caso (ver el diagrama de flujo de abajo), la subrutina presenta al operador una petición de introducción en todos los campos previstos. Éste podrá utilizar el campo particular que se le propone introduciendo simplemente el valor a buscar. Alternativamente puede excluir aquel campo como clave de búsqueda introduciendo el carácter *.

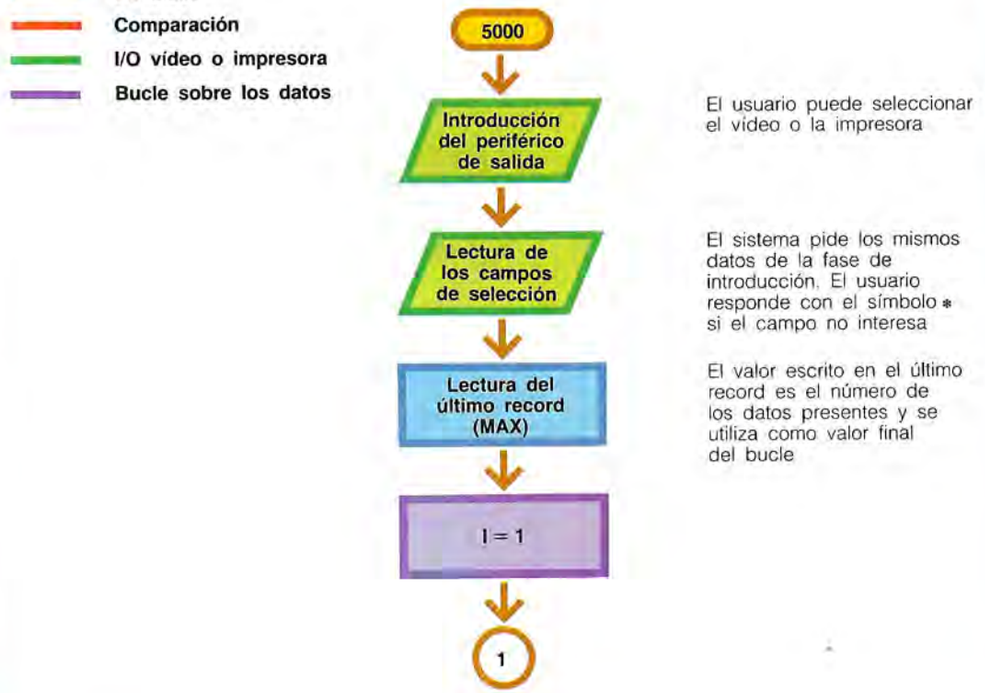
Impresión (subrutina 6000)

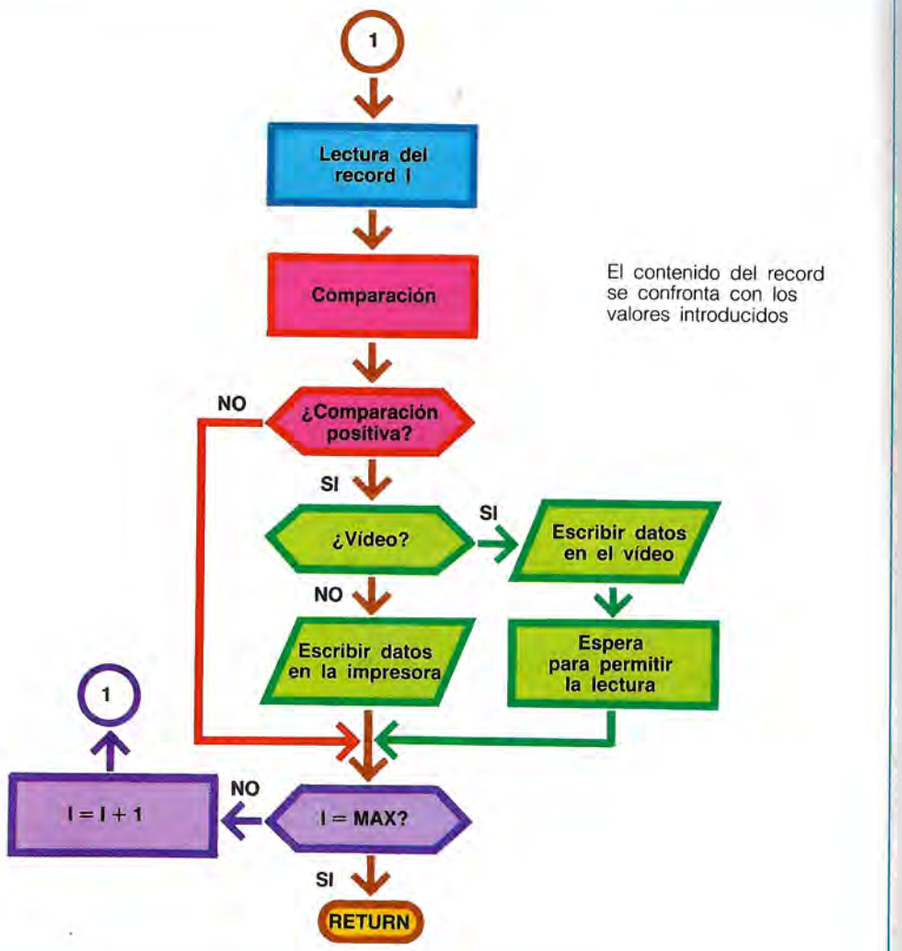
La subrutina de impresión debe poder imprimir sobre papel todo el contenido del file. Por tanto, estará constituida por un bucle articulado desde el número del primer record hasta el número del último record utilizado anteriormente en escritura, que está memorizado en el último record del file. Dado que el procedimiento no presenta particulares dificultades, se ha omitido el diagrama de flujo.

Funciones de impresión particulares

El programa descrito puede modificarse para que se adapte a una gran variedad de casos modificando sencillamente las longitudes y los significados de los campos. Sin embargo, en algunas aplicaciones son necesarias modificaciones de la subrutina de impresión. Por ejemplo, si el archivo se refiere a una dirección para el envío de material publicitario el programa debe poder proporcionar las salidas (es decir, las direcciones impresas) sobre un soporte utilizable

SUBRUTINA DE BUSQUEDA





El contenido del record se confronta con los valores introducidos

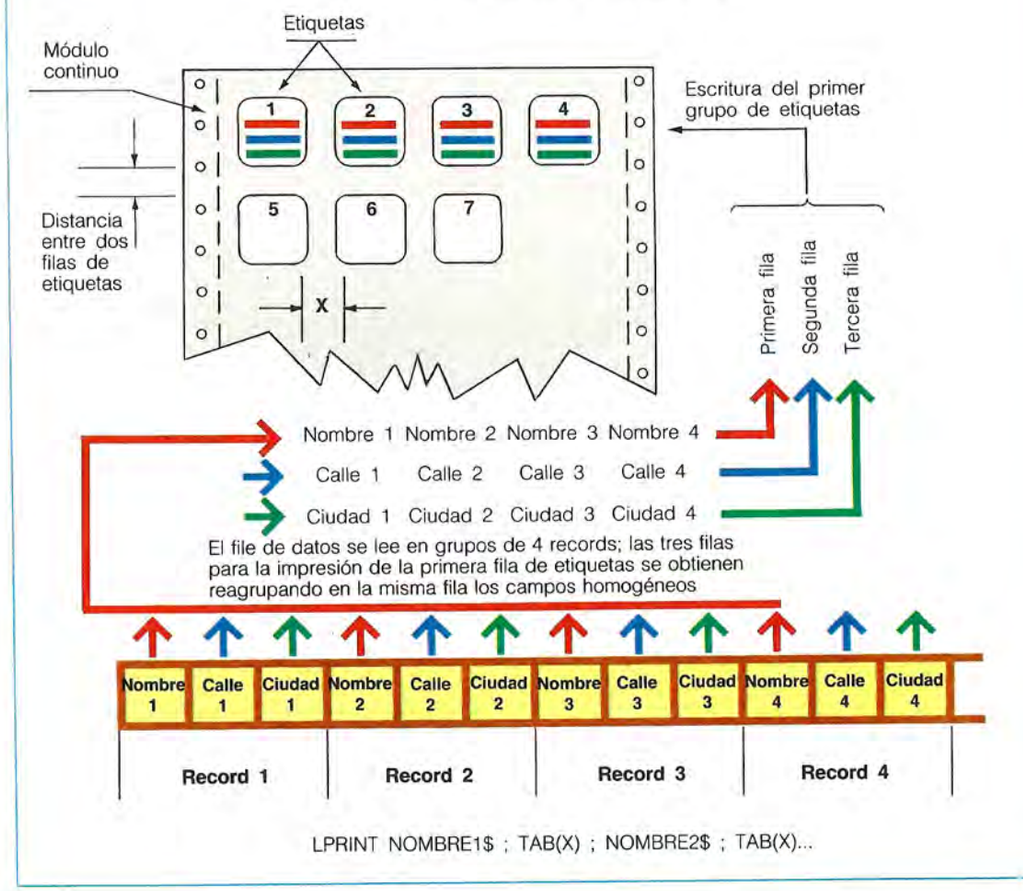
directamente, como las etiquetas autoadhesivas. Con este fin, en el comercio existen tiras continuas de papel encerado, adaptadas al sistema de arrastre de las impresoras, sobre las que hay filas de etiquetas adhesivas.
El programa de gestión del archivo puede adaptarse para tener la escritura de las direcciones directamente sobre las etiquetas. Al final de la fase de impresión bastará con desprender las etiquetas impresas del soporte provisional para transferirlas al definitivo.
La impresión sobre etiquetas se diferencia de la normal por la necesidad de estructurar la salida imponiendo en primer lugar la escritura de los nombres sobre una misma línea (y sobre diferentes etiquetas), después la escritura de las direcciones en la siguiente línea y luego la escritura de la ciudad en una tercera línea. Por tanto, la rutina de impresión debe estructurarse en función del número de direcciones que se desea escribir en la misma línea, que será igual al número de etiquetas que se encuentra en la misma fila en el módulo. La rutina estará constituida por dos bucles: el primero, más exterior, servirá para tomar los datos del disco; el segundo reagrupará los datos para formar las diversas líneas de impresión.
El esquema lógico de una fase de impresión sobre una fila de cuatro etiquetas se ha representado aquí al lado. En este caso, el bucle más exterior tiene un paso 4 y el más interior forma las tres líneas que contienen respectivamente el apellido y nombre, la calle y la ciudad. Las instrucciones de impresión deben prever entre un

campo y el otro las adecuadas tabulaciones para el correcto posicionado de las direcciones en las etiquetas.

El ordenamiento de los datos

En la gestión de los archivos es indispensable disponer de rutinas que puedan ordenar los datos con respecto a uno cualquiera de los campos que componen los records.
Estas rutinas de empleo generalizado, llamadas **sort**, suelen proporcionarlas el fabricante de la máquina, o pueden encontrarse fácilmente. Sin embargo, es muy útil disponer de rutinas propias, que pueden adaptarse mejor al caso específico, en particular si el número de datos no es demasiado elevado. En muchas aplicaciones es necesario ordenar tablas que están contenidas en la memoria; el software generalizado trabaja sobre files, o sea sobre la memoria masiva y, en estos casos, su utilización produce más dificultades que beneficios. Sin embargo, debe tenerse en cuenta que la escritura de rutinas de ordenación presupone la existencia del Compilador Basic. Empleando el Basic interpretado puede desarrollarse el software, pero debido a la baja velocidad de ejecución, un programa interpretado no sería prácticamente utilizable. Por tanto, es necesario compilar. Como alternativa puede escribirse el software de ordenamiento en lenguaje Assembler, que si por un lado constituye la mejor solución, por otro requiere una notable experiencia y un mayor empleo de tiempo para la escritura y para las pruebas. A continuación se presentarán dos rutinas de ordenamiento. La primera trabaja sobre la memoria y puede utilizarse para los datos en el disco únicamente a condición de que todo el file que se examina pueda estar contenido en la memoria

IMPRESION SOBRE ETIQUETAS



LPRINT NOMBRE1$; TAB(X) ; NOMBRE2$; TAB(X)...

de la máquina. La segunda trabaja en files memorizados en disco y utiliza la técnica de búsqueda binaria. Esta rutina ofrece elevadas prestaciones y puede utilizarse en la gestión de archivos de dimensiones medianas.

Ordenamiento en memoria

En la página siguiente se ha representado el diagrama de flujo de un programa de demostración que ordena en valor creciente el contenido de la matriz A(20).

El ordenamiento se obtiene confrontando cada valor contenido en la matriz con el que le sigue inmediatamente (hacia los índices mayores): si el primero es mayor, los dos elementos se intercambian. El ordenamiento termina cuando una exploración de todos los datos no ha dado lugar a ninguna inversión; esta condición la controla el flag K, que se pone a 1 cuando es necesaria una inversión. Al final de un bucle sobre los datos, basta con comprobar el valor de K para saber si se ha realizado o no una inversión. En caso afirmativo es necesario realizar un nuevo bucle sobre todos los datos. Por el contrario, si el valor de flag es cero, no se ha producido inversión; los datos están ordenados (cada uno de ellos es menor que el que le sigue) y la rutina puede terminar. El programa que se describe (el listado aparece en las págs. 678 y 679) tiene una finalidad de demostración; puede escribirse de forma más compacta aprovechando mejor la función realizada por el flag K.

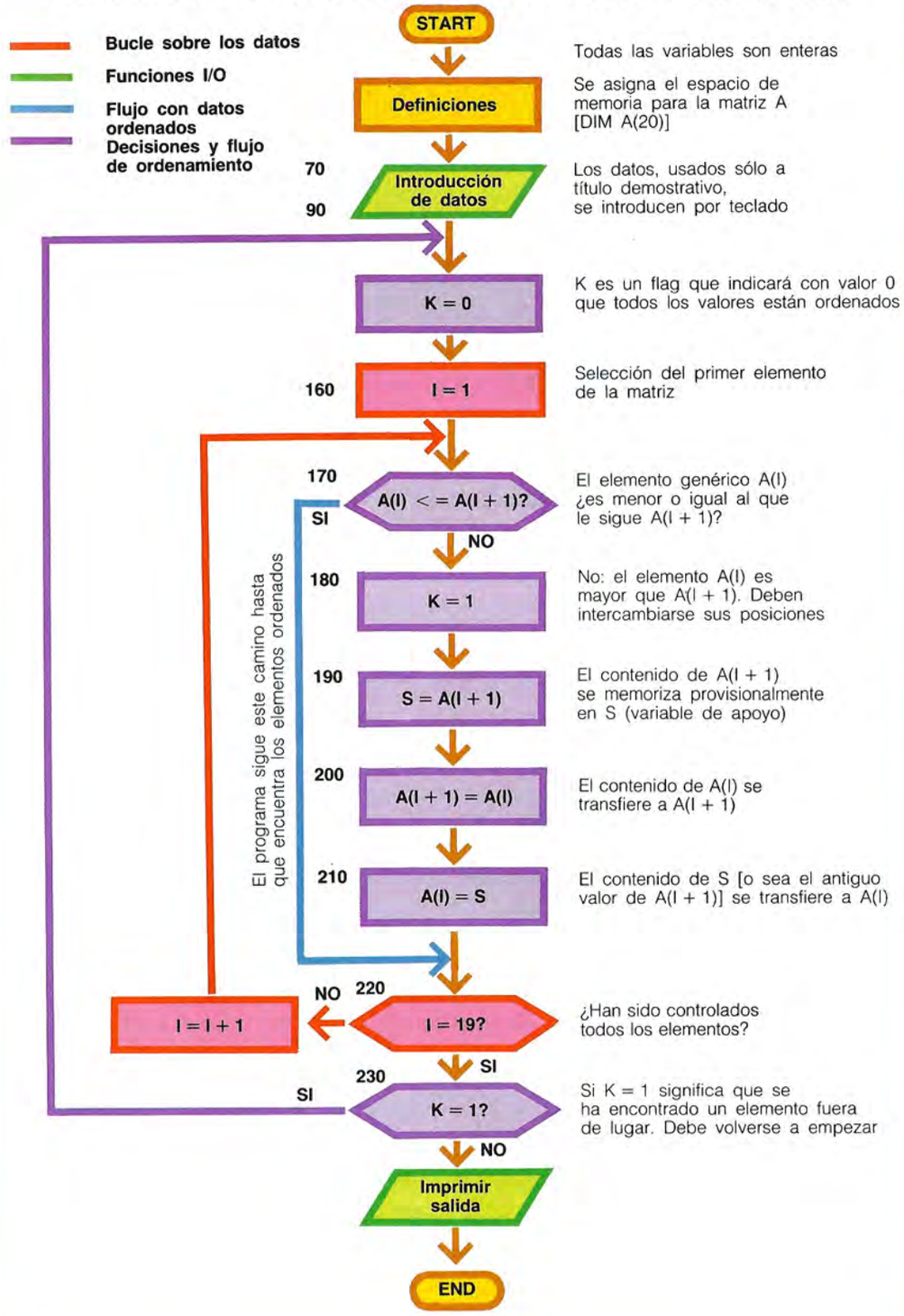
Este flag es un indicador que sólo puede asumir dos valores: 1 y 0, o sea verdadero o falso. Uno de los dos valores expresa que una cierta condición (ordenamiento de los datos) no se ha verificado; el valor opuesto indica la verificación de dicha condición. El bucle de ordenamiento de datos debe reemprenderse en tanto que la condición sea falsa (no verificada) y constituye un clásico ejemplo de aplicación de la instrucción WHILE... WEND. En la pág. 679, abajo, se ha representado el mismo programa reescrito utilizando esta última instrucción. En el correspondiente listado también se ha introducido otra variante: el empleo de la instrucción SWAP. En el programa anterior, para intercambiar el contenido de las memorias A(I) y A(I + 1), se utilizaba una memoria de apoyo (S); con la instrucción SWAP, la gestión de la memoria de apoyo está a cargo del sistema, con lo que el programa queda muy simplificado.

El intercambio del contenido de dos variables requiere el empleo de una variable de apoyo donde memorizar, provisionalmente, el valor que debe pasar de una a otra. Desarrollando la

La sala de control de un muelle ferroviario.

K. Reese/Marka

DIAGRAMA DE FLUJO PARA EL ORDENAMIENTO DE UNA MATRIZ



ORDENAMIENTO DE UNA MATRIZ (PRIMERA VERSION)

```
10 ' **FILE ORDENA
20 OPTION BASE 1
30 DIM A(20)
40 ' ** LECTURA DATOS A ORDENAR
50 '
60 FOR I=1 TO 20
70 PRINT " Dato N.:";I
80 INPUT "VALOR      ";A(I)
90 NEXT I
100 '
110 ' ** IMPRESION DATOS LEIDOS
120 '
130 LPRINT " VALORES INTRODUCIDOS "
140 LPRINT
150 FOR I=1 TO 20
160 LPRINT " I = ";I," VALOR = ";A(I)
170 NEXT I
180 '
190 ' *** ORDENAMIENTO
200 '
210 K=0
220 FOR I=1 TO 19
230 IF A(I) <=(I+1) GOTO 270
240 K=1
250 S=A(I+1)
255 A(I+1)=A(I)
260 A(I)=S
270 NEXT I
280 IF K=1 GOTO 210
290 '
300 ' ** IMPRESION DATOS ORDENADOS
310 LPRINT " VALORES ORDENADOS "
320 LPRINT
300 FOR I=1 TO 20
340 LPRINT " I = ";I, "VALOR = ";A(I)
350 NEXT I
360 END
```

```
 VALORES INTRODUCIDOS

I=  1           Valor=  32
I=  2           Valor=  25
I=  3           Valor=  18
I=  4           Valor=  0
I=  5           Valor=  5
I=  6           Valor=  48
I=  7           Valor=  98
I=  8           Valor=  74
I=  9           Valor=  125
I=  10          Valor=  2
I=  11          Valor=  56
I=  12          Valor=  36
I=  13          Valor=  14
I=  14          Valor=  25
I=  15          Valor=  12
I=  16          Valor=  10
I=  17          Valor=  91
I=  18          Valor=  58
I=  19          Valor=  77
I=  20          Valor=  61
```



```
VALORES ORDENADOS

I=  1           Valor=  0
I=  2           Valor=  2
I=  3           Valor=  5
I=  4           Valor=  10
I=  5           Valor=  12
I=  6           Valor=  14
I=  7           Valor=  18
I=  8           Valor=  25
I=  9           Valor=  25
I=  10          Valor=  32
I=  11          Valor=  36
I=  12          Valor=  48
I=  13          Valor=  56
I=  14          Valor=  58
I=  15          Valor=  61
I=  16          Valor=  74
I=  17          Valor=  77
I=  18          Valor=  91
I=  19          Valor=  98
I=  20          Valor=  125
```

ORDENAMIENTO DE UNA MATRIZ (SEGUNDA VERSION)

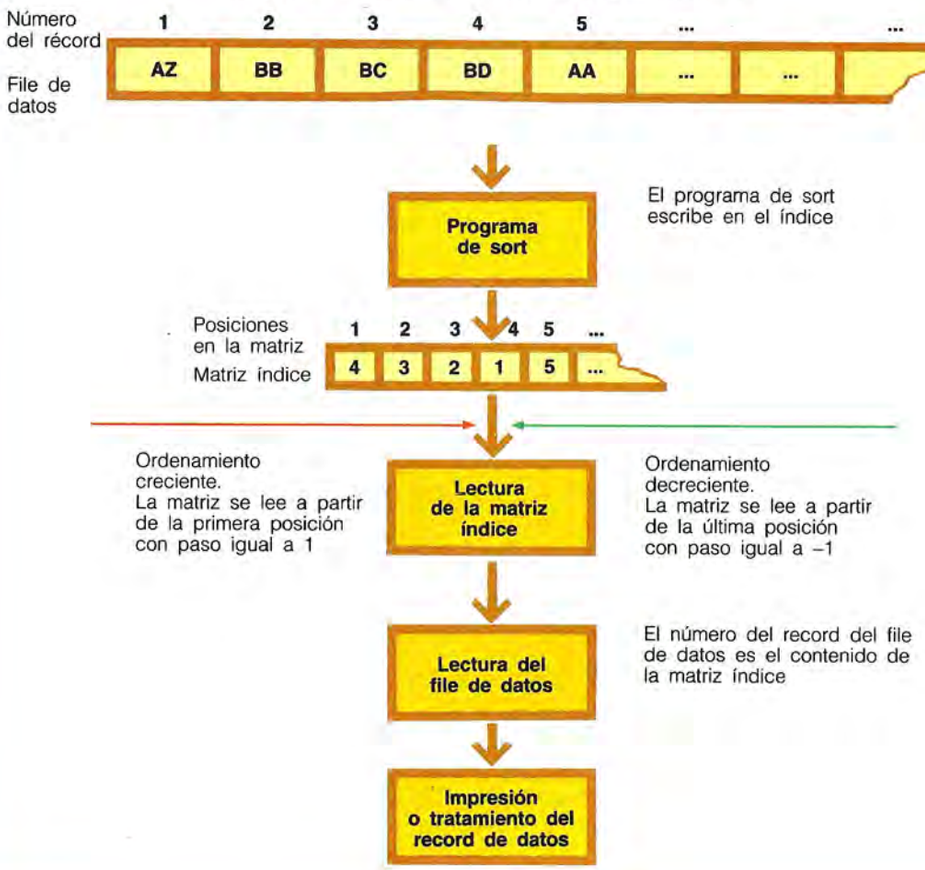
```
10 ' **FILE ORDENA
20 OPTION BASE 1
30 DIM A (20)
40 ' ** LECTURA DATOS A ORDENAR
50 '
60 FOR I=1 TO 20
70 PRINT " Dato N.: ";I
80 INPUT " VALOR     ";A(I)
90 NEXT I
100 '
110 ' ** IMPRESION DATOS LEIDOS
120 '
130 LPRINT " VALORES INTRODUCIDOS "
140 LPRINT
150 FOR I=1 TO 20
160 LPRINT " I = ";I,"VALOR = ";A (I)
170 NEXT I
180 '
190 ' *** ORDENAMIENTO
200 '
210 K=1     ' K se pone inicialmente = 1 para forzar el primer bucle
220 WHILE K
230 K=0
240 FOR I=1 TO 19
250 IF A(I)>A(I+1) THEN SWAP A(I),A(I+1):K=1
260 NEXT I
270 WEND
290 '
300 ' ** IMPRESION DATOS ORDENADOS
310 LPRINT "VALORES ORDENADOS "
320 LPRINT
330 FOR I=1 TO 20
340 LPRINT " I =";I, "VALOR = ";A(I)
350 NEXT I
360 END
```

misma función con la instrucción SWAP se activa una función que consiste únicamente en el intercambio de nombres de las variables. Así, la memoria de ida toma el nombre de la de llegada y viceversa: a efectos del programa, esto es lo mismo que el paso de los datos de una a otra, mientras que el sistema sólo ha cambiado las direcciones con una velocidad de ejecución mucho mayor. Para generalizar el programa, basta con sustituir en las líneas 30, 150, 240 y 330 los valores 20 y 19 por los nuevos valores. Finalmente, debe tenerse en cuenta que el programa, tal como está, únicamente trabaja con valores numéricos (la matriz A(20) está compuesta de números enteros); para utilizarlo con las cadenas es suficiente añadir el símbolo de cadena a la matriz [A$(N)], así como a la variable de apoyo [S$].

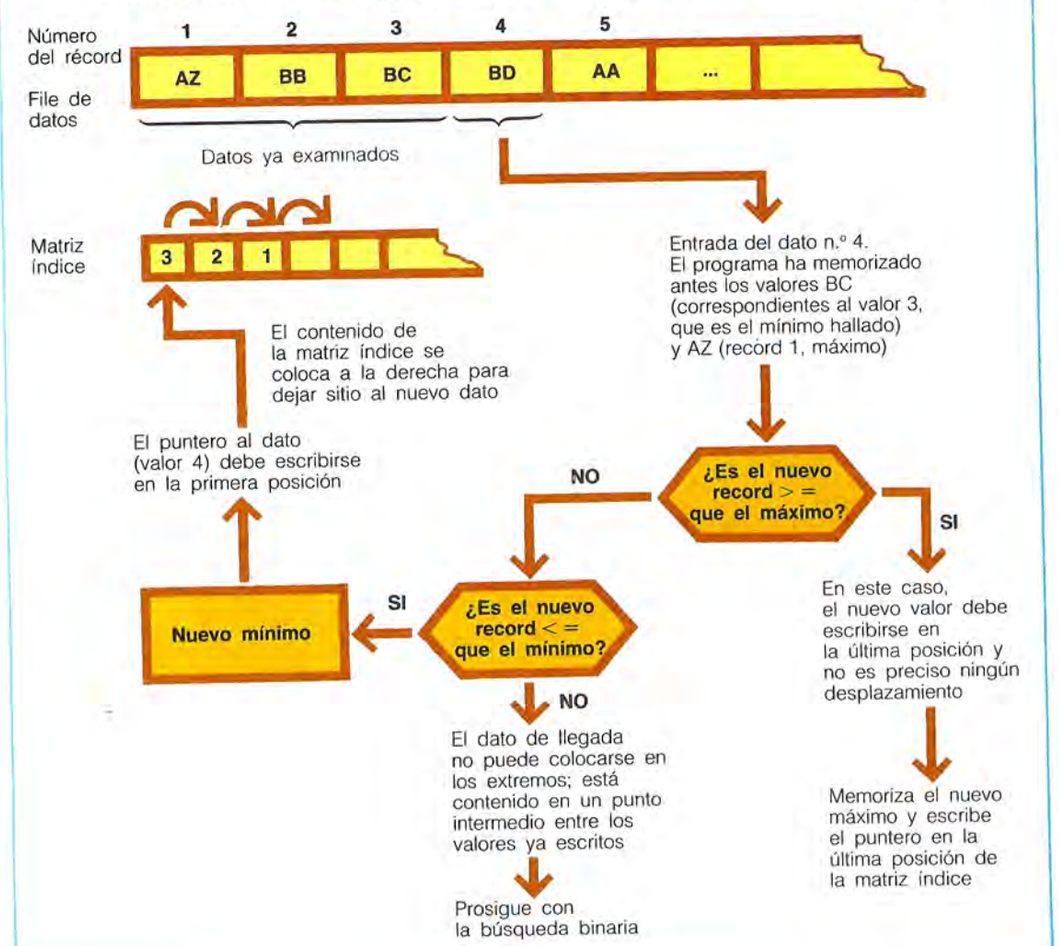
Ordenamiento en disco

La rutina de ejemplo que presentamos utiliza una memoria de apoyo para las direcciones de los records, que debe dimensionarse con el mismo número que los records del file. En esta memoria se depositan las direcciones de los records a medida que son seleccionados según el orden alfabético (Z = menor, A = mayor) del contenido. Por ejemplo, si los datos del record número 15 son menores que los contenidos en los records 1 y 2, en la **matriz índice** (memoria de apoyo de las direcciones), primero se escribirá el valor 15 y luego los valores 1 y 2. Así, utilizando la matriz como puntero a los datos (el file se trata como si fuese un file de índice), pueden leerse los datos en orden creciente o decreciente, iniciando la exploración de la matriz índice desde la primera posición o desde la última.

SORT DE UN FILE CON MATRIZ INDICE



MECANISMO DE ELECCION DE LOS VALORES MAXIMO Y MINIMO



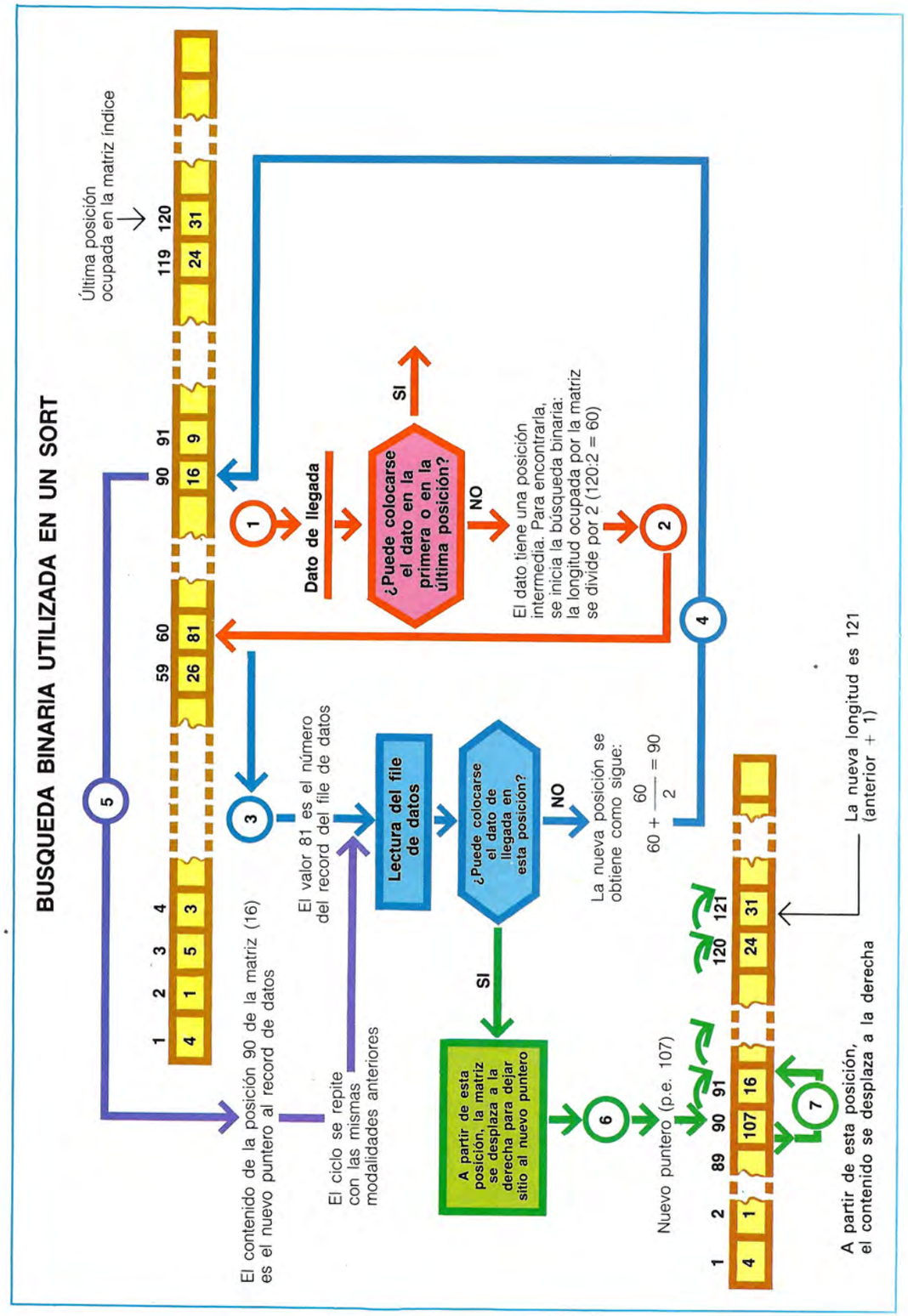
En la pág. 680 se presenta la lógica descrita. El mecanismo de preparación de la matriz índice puede esquematizarse de este modo.

Supongamos que ya se tienen situados algunos datos, por ejemplo los primeros tres records del file: la matriz índice contendrá entonces tres valores, que son los números de los tres primeros records del file ordenados en valor creciente del contenido. Así, el primer elemento de la matriz índice contendrá el número de records que corresponde al dato de menor valor entre los tres situados; el siguiente elemento contendrá el número de records del dato inmediatamente superior, etc. En este punto se toma el nuevo dato a situar (record ñúmero 4); para atribuir al número del record la colocación exacta en la matriz índice, debe releerse el contenido de los records examinados antes, confrontar cada uno con el nuevo dato numérico y decidir su posición.

Para acelerar esta búsqueda pueden adoptarse dos técnicas (conjuntas, no alternativas). La primera consiste en memorizar el valor mínimo y el valor máximo encontrados en los datos anteriores. A la llegada de un nuevo record para situar, sobre estos dos valores se aplica el primer control (ver el gráfico superior).

Si el nuevo dato es menor o igual, como mínimo deberá colocarse en la primera posición, y si es mayor o igual, como máximo en la última.

En el caso de que estos controles no hayan resultado positivos, el dato de llegada deberá situarse en un punto intermedio. Para colocarlo en



Un ordenador personal para comunicarse

Además de un medio de comunicación, el ordenador personal puede ser un eficaz instrumento de rehabilitación motriz para personas que sufren de graves carencias de movimiento o de articulación del lenguaje. Esto lo sugiere la experiencia en curso en el Centro de Educación Motriz de la XVI Unidad Sanitaria Local de Génova Levante. El Centro se dedica a la reeducación funcional motriz y se orienta principalmente a la patología neuromotriz tanto en la edad menor como adulta.

En este centro, un grupo de disminuidos incapaces de expresarse con la voz ha aprendido a utilizar un ordenador personal para escribir mensajes en la pantalla de vídeo. Como no pueden realizar los movimientos necesarios para utilizar el teclado, se valen de un programa especial, denominado LOGOS 4, que permite componer palabras y frases eligiendo las letras con la leve presión de una gran tecla.

El programa ha sido realizado para el ordenador personal IBM por la sociedad productora de software A & B de Génova, basándose en prototipos experimentales desarrollados en el Centro de Educación Motriz, con la colaboración del Instituto de Electrónica de la Universidad de Estudios de Génova, en el ámbito del proyecto Tecnologías Biomédicas del CNR.

Para construir el programa se han memorizado datos estadísticos correspondientes al idioma italiano, con particular referencia a textos escritos por niños de escuela elemental o destinados a ellos. A partir de estos datos y de los caracteres ya seleccionados, el procesador elige cada vez las letras que podrían seguir y las presenta en pantalla en orden decreciente de probabilidad. El minusválido puede seleccionarlas apretando un solo pulsador (las letras se suceden automáticamente a velocidad regulable), o dos pulsadores, uno de aceptación y otro de rechazo.

El LOGOS 4 presenta los caracteres a seleccionar en la pantalla en una sucesión diferente cada vez, basándose en las propiedades estadísticas del léxico italiano, que hacen más probable la presencia de ciertas letras en lugar de otras en la formación de las palabras. Por ejemplo, se ha seleccionado el grupo de consonantes «ch», que aparece a menudo en pantalla después de la «e» y la «i». De esta manera no es necesario recorrer todo el alfabeto cada vez, y una vez adquirida destreza con el nuevo medio, es posible comunicarse de manera relativamente rápida.

Otros dos pulsadores sirven para invertir la sucesión de los caracteres y para pasar de la exploración automática a la manual. Las letras se presentan en varios colores y con una particular forma gráfica «en embudo» que, aprovechando la visión periférica del ojo, permite que el inválido prepare con tiempo su elección.

Este tipo de presentación, asociado a la amplificación sonora, también permite, en los casos de notable dificultad de movimientos, realizar igualmente la función de selección utilizando tiempos de exploración más dilatados.

La diferenciación cromática para las letras que se seleccionan se ha introducido para facilitar la identificación, especialmente por parte de los usuarios con problemas de visión.

Las letras seleccionadas, que aparecen a la izquierda, constituyen provisionalmente un soporte para la construcción de las palabras y una ayuda para eventuales correcciones. Además de ser en colores, son realmente de dimensiones mayores que las de debajo (es decir, las del texto propiamente dicho).

Se han memorizado datos estadísticos relativos al idioma italiano, en particular a los vocablos contenidos en los textos de las escuelas elementales. La investigación estadística ha permitido predeterminar el orden de frecuencia en que se presentan las letras después del último grupo escrito.

Además de las 26 letras del alfabeto italiano, hay cinco signos convencionales que permiten activar las funciones de espaciado, retroceso, aparte, corrección de error de selección y del texto acabado de escribir (y no memorizado en la máquina) y retorno a las funciones iniciales del programa.

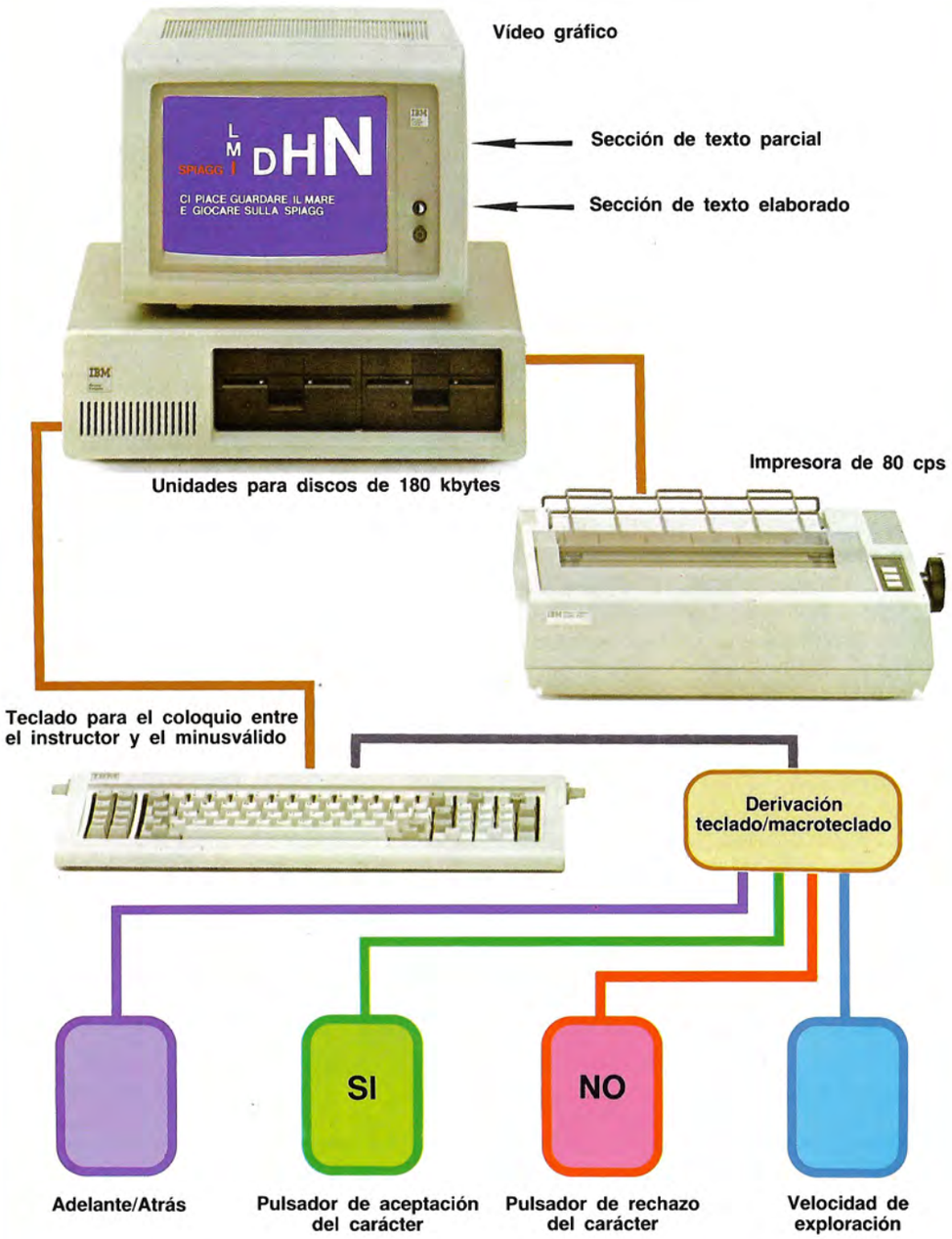
El LOGOS 4, como se ha dicho, es un programa implantado en el ordenador personal IBM.

La configuración mínima para la aplicación es la siguiente:

Unidad central (128 K de RAM) + tarjeta + adaptador de juegos
1 Unidad de lectura de discos de 180 K
Teclado
Teclado para minusválidos
Impresora (80 cps)
Vídeo gráfico

Dada la estructura del programa, la gestión del LOGOS 4 también es posible sin teclado ni impresora. Es evidente que con la configuración reducida, las funciones de diálogo y de impresión del texto no son posibles.
El vídeo gráfico también puede eliminarse y sustituirse por un aparato de TV realizando las adecuadas adaptaciones (modulador), el cual se utilizará como monitor.
En la hipótesis de futuros desarrollos (como, por ejemplo, la actualización dinámica de la estadística u otro), el PC IBM con la configuración mínima podrá implantarse añadiendo otra unidad, o disco de 180 K, o sustituyendo la anterior por una unidad de lectura de discos de 360 K.
Las posibilidades de «personalizar» el empleo del sistema LOGOS 4 son notables.
El programa prevé la eventualidad de poder proporcionar a cada usuario su propio disco operativo, programado en base a las capacidades motrices individuales. Efectivamente, existen dos «máscaras» que proponen algunas opciones relativas a las características de la pantalla y de la impresora.
La «máscara» propone algunas variables relativas a las modalidades de control del movimiento y a las propuestas visuales.
La selección de la letra puede efectuarse de dos maneras: en la exploración automática, las letras se desplazan automáticamente de derecha a izquierda a una velocidad que también es programable (de 1 a 5 segundos), mientras que con la exploración manual, las letras en el embudo están fijas y el usuario selecciona la letra deseada accionando los pulsadores de aceptación (sí) o de rechazo (no).
Para esto, el minusválido debe poder accionar estos dos pulsadores. El programa de exploración automática permite al inválido realizar el movimiento necesario para la selección de la letra con un esfuerzo mínimo (puede ser suficiente incluso un solo movimiento).
El de exploración manual, en cambio, permite una mayor velocidad y una gestión directa de la máquina. Para facilitar la adquisición de las informaciones visuales es posible modificar el color del fondo (hay disponibles 16 colores), de las letras en el embudo (4 colores), de las que están en fase de selección (4 colores) y de las adquiridas (4 colores). Estas modificaciones permiten adquirir las informaciones cromáticas de acuerdo con las preferencias o los condicionamientos de la vista del usuario.
Utilizando las variaciones cromáticas, además, puede regularse el grado de dificultad de las propuestas visibles en base a las capacidades perceptivas y de aprendizaje del usuario. Además, hay la posibilidad, siempre utilizando variaciones de color, de eliminar las letras del embudo o las letras adquiridas. En lo referente a las características de la impresora, es posible seleccionar el número de caracteres por línea, programando también el tamaño de las letras, la intensidad de impresión y los espacios.
El programa se inicia proponiendo al usuario siete funciones que permiten una gestión autónoma del propio programa.
El minusválido puede iniciar un nuevo texto, memorizarlo si lo desea, continuar el trabajo en diferentes momentos, reclamar al disco el texto memorizado e imprimir todo el texto procesado.
Además, como primera operación de control-ambiente se ha introducido la «llamada». Seleccionando esta función se activa una señal acústica intermitente que indica las eventuales necesidades de ayuda por parte del minusválido.
También se ha introducido la posibilidad de establecer un coloquio entre el teclado normal y el interface accionado por el minusválido.
Esto permite la formulación de peticiones escritas en el vídeo por parte del operador, a las cuales el usuario puede responder, facilitando así el aprendizaje del funcionamiento del programa por parte del inválido. Como se ha proyectado para responder a las exigencias de minusválidos con graves dificultades motrices, el LOGOS 4 hace posible la escritura utilizando también un solo pulsador o sensor.
Este pulsador permite la aceptación de la letra en la selección. Siempre que las capacidades del minusválido lo permitan, la velocidad operativa puede aumantarse introduciendo el uso de un segundo pulsador que pueda invertir la presentación de las letras del embudo, permitiendo así la recuperación de una letra anteriormente «perdida» o que casualmente se encuentra al final de la cadena.
Accionando un tercer pulsador, es posible pasar de la representación de las letras con exploración automática a la de exploración manual; en el caso de selección manual, el minusválido puede hacer desfilar las letras mediante dos pulsadores, uno para rechazar la letra y otro para aceptarla.
El uso combinado de cuatro pulsadores es una ulterior adaptación de la instrumentación a las

CONFIGURACION HARDWARE PARA LA IMPLANTACION DEL PROGRAMA LOGOS4



Macroteclado u otro periférico adaptado a las posibilidades del minusválido

IBM

Una imagen del programa LOGOS4 en funcionamiento. Las posibilidades de empleo del ordenador personal como ayuda para los minusválidos se van concretando gradualmente. La iniciativa común emprendida por la XVI USL de Génova Levante, IBM y A & B constituye un nuevo paso adelante hacia el derribo de la barrera de incomunicación que aísla a los minusválidos neuromotrices.

dificultades motrices del sujeto minusválido. Finalmente, se prevén proyectos de utilización de la máquina que inicialmente requieren modalidades más sencillas (p.e. el uso de un solo pulsador) hasta llegar a estrategias más complejas enfocadas a la utilización de los cuatro pulsadores para optimizar el tiempo de composición del texto o para llegar, hasta donde sea posible, al empleo de instrumentos de uso más corriente (p.e., con adaptaciones especiales, la máquina de escribir): de esta forma, la ayuda proporcionada por el programa tendrá también una función de reeducación.

En las estrategias más sencillas –siempre en presencia de una invalidez grave– se prevé la sustitución del pulsador mecánico por un sensor más sofisticado.

Las perspectivas para el futuro desarrollo del sistema LOGOS 4 son esperanzadoras. Un proyecto ya definido contempla la ampliación de la utilización en función de diferentes edades y preparaciones profesionales, realizando una estadística personalizada basada en textos escritos por el propio sujeto. Un segundo objetivo es el desarrollo de un programa que puede leer de forma completa una palabra abreviada o una frase sumarizada o sistemas particulares de comunicación simbólica no verbales (p.e., el lenguaje BLISS).

Un eventual programa de descodificación permitiría reducir de forma notable los tiempos de escritura ampliando considerablemente las capacidades expresivas del minusválido.

Mediante las funciones implantadas en el LOGOS 4, el minusválido puede memorizar el texto escrito, imprimirlo, reclamarlo en el vídeo, activar una señal acústica para pedir ayuda y establecer un coloquio con otras personas que igualmente utilizan el teclado del sistema con esta finalidad.

La variabilidad de los comandos, de los colores, de la modalidad de uso y de la velocidad permite adaptar el medio a las posibilidades del minusválido y reducir el estado de ansiedad que a veces se produce en su primer contacto con la máquina.

Además, en el transcurso de la actividad de reeducación, en algunos casos puede preverse un paso gradual de las modalidades de uso más sencillas a las más complejas, hasta la utilización de instrumentos de uso más corriente, como la máquina de escribir.

La versión actual del programa se ha estudiado para sujetos en edad escolar o postescolar, pero su empleo también podrá extenderse a minusválidos de diferentes preparaciones y edades con el desarrollo de estadísticas «personalizadas», o sea basadas en textos escritos por el propio usuario.

El empleo del ordenador personal, además de permitir el diálogo entre inválidos y preceptores, también se ha revelado como un estímulo tanto para el aprendizaje cuanto para la recuperación del control motriz.

el sitio justo, como ya se ha dicho, deberán leerse todos los datos ya situados, confrontarlos con el de llegada y, entonces, podrá determinarse su posición. Para hacer más rápida esta fase de búsqueda se adopta el método de la división por la mitad (búsqueda binaria).
En este caso se toma el valor central de la matriz índice (en el ejemplo indicado en la pág. 681, el número 2) y se lee el dato correspondiente (BD); el nuevo valor se confronta con éste y, a continuación del resultado de la confrontación, se decide si se prosigue hacia la derecha o hacia la izquierda en la matriz índice.
En la pág. 682 se ha representado el esquema completo de la rutina. A título de ejemplo se ha incluido la descripción del programa de utilidad OLISORT* de Olivetti.

El programa de utilidad OLISORT

El Olisort es un paquete sort/merge versátil de altas prestaciones, proyectado para el ordenador personal Olivetti M20. Sus prestaciones lo hace útil en casi todos los tipos de aplicación de gestión. El Olisort permite:

* OLISORT es una denominación registrada.

— realizar el sort de un file
— realizar el merge de dos files
— seleccionar un record de un file
— añadir un file al final de otro
— realizar un sort y seleccionar al mismo tiempo

El Olisort está comandado por parámetros que se pasan a una cadena Basic. Toda la cadena de parámetros puede pasarse al Olisort a través de una cadena de comandos escrita en Basic, o a través de una corta cadena de comandos que hace referencia a un file de parámetros. Un file de parámetros o una cadena de comandos puede crearse de forma interactiva mediante los programas de utilidad Olisort. Esto significa que pueden crearse fácilmente largas cadenas de parámetros reduciendo apreciablemente la posibilidad de errores de implantación.
El Olisort puede hacerse funcionar reclamándolo con un programa Basic o utilizando el programa de utilidad Olisortx. Esto permite construir más de un sort en una sucesión de programas que actúan como un programa único, o tomar un determinado file y realizar enseguida el sort.
Las prestaciones internas más importantes del Olisort son las siguientes:

Sistema de cálculo MOT 53385 de IBM.

EL PROGRAMA OLISORT

El programa Olisort puede ponerse en ejecución insertando una cadena adecuada de comando en el interior de cualquier programa Basic, o bien utilizando el programa de utilidad Olisortx. En esta página se ilustra la imagen de vídeo que aparece al llamar este último programa.


```
OLISORTX 1.1 - OLISORT UTILITY PROGRAM
Copyright (c) 1982  OLIVETTI


SELECT FUNCTION
---------------
 S = Sort Existing Data File
 B = Build and Display Command String
 C = Create Command String File
 E = END

Enter desired selection :BI
```

Las primeras dos líneas de la escritura que aparecen en el monitor informan al usuario que el control se ha pasado al programa Olisort (en particular al Olisortx, versión 1.1).
El programa Olisortx presenta en el vídeo el menú principal del Olisort y relaciona las funciones que son accesibles inmediatamente.

Ordenamiento de un file de datos existente antes: selección S.

Construcción y visualización de una cadena de comandos: selección B.

Creación de un file que contiene cadenas de comandos: selección C.

Salida del Olisortx (para salir del Olisort): selección E.

El programa pide al operador que introduzca las siglas correspondientes a la función deseada. En el ejemplo se ha seleccionado la opción B: la función permitirá introducir una cadena de comandos que contiene los parámetros y los valores necesarios para realizar un sort en un file de datos.

— pueden especificarse hasta 10 claves de sort por record
— selección o exclusión de record permitida mediante claves Select/Exclude (hasta 4)
— las claves Select/Exclude pueden combinarse juntas a través de operadores lógicos AND u OR
— para cada clave sencilla Select/Exclude pueden especificarse los extremos de comparación: menor, igual o mayor
— para cada clave sencilla Select/Exclude pueden utilizarse los caracteres * o ?: el primero iguala todos los caracteres sucesivos, y el segundo, sólo el primer carácter sucesivo
— con clave de sort y Select/Exclude puede elegirse la atribución del mismo valor alfabético para letras minúsculas y mayúsculas
— los discos utilizados para los files de trabajo Olisort y para los files de salida no pueden ser cambiados; el Olisort gestiona automáticamente las señalizaciones del cambio en el momento oportuno
— pueden saltarse hasta 32767 records iniciales de un file antes de iniciar el sort o el merge
— hay prestaciones para el tratamiento de los errores; el código de estado siempre es devuelto al programa o al programa de utilidad que llama el Olisort
— la introducción del Olisort en programas Basic existentes es muy sencilla y no necesita ningún conocimiento de interface Assembler

El Olisort se ha proyectado específicamente para trabajar con el intérprete Basic M20 y soporta records estándar de longitud fija con campos también de longitud fija.

Prestaciones. Debido a que no pueden procesarse files multivolumen, la máxima dimensión del file que puede introducirse en el Olisort está limitada por el espacio disponible en el disco. **El factor limitador es la dimensión del file de trabajo del sort, que puede ser hasta ~ 1,7 veces mayor que la del file de entrada.** La máxima longitud del record es de 256 bytes (valor por omisión). En el caso de files con records excepcionalmente largos, este valor puede modificarse mediante un comando CPOS.
Para mayor sencillez, a continuación haremos referencia a la máxima longitud del record de 256 bytes. Todos los files deben tener los records de longitud fija.

Con las claves de sort y Select/Exclude pueden gestionarse los siguientes tipos de datos:

— cadenas alfanuméricas
— campos en formato hexadecimal
— números enteros
— números en simple precisión
— números en doble precisión

Especificando una cadena alfanumérica con una clave de sort o Select/Exclude, pueden atribuirse los mismos valores a los caracteres mayúsculos o minúsculos.
Especificando campos hexadecimales con teclas sort o Select/Exclude, se pide al Olisort que trate los caracteres como códigos hexadecimales. Esto significa que las mismas letras tienen valor diferente en mayúscula y en minúscula.
Realizando el sort (definiendo hasta diez claves) se tendrá el ordenamiento ascendente o descendente según el parámetro: A (ascending) o D (descending). Por ejemplo, puede realizarse el sort de un file en orden decreciente de saldos atrasados, en orden creciente de estado, para producir una relación como la presentada en la siguiente tabla:

Relación de saldos atrasados

Estado	Cliente	Saldo ($)
Nevada	M & M Ltd.	2596,00
	Zilch & Sons	1976,55
	ABC Co.	568,88
New Jersey	NXR Ltd.	8870,77
	Wyatt & Hyatt	8345,98
	Acme Ltd.	5689,00
	Olive Oil Co.	67,43

Puede realizarse un sort para obtener un file de salida ordenado o, si se desea, también puede sobreponerse al mismo file de entrada el file de salida ordenado, prestación útil si el espacio en el disco es mínimo y no se desea cambiar el disco durante el sort. En cualquier caso, si se produce una avería mecánica durante la fase sort, puede perderse el file de entrada. Si se decide escribir encima del file de entrada para protegerse contra eventuales averías de la máquina, será conveniente realizar el sort sobre una copia del disco que contiene el file de entrada.
Con el Olisort puede generarse un file de salida que sólo contenga los records seleccionados del file de entrada. Puede realizarse una selec-

ción/exclusión al mismo tiempo que la ejecución de un sort. Así, si se decide realizar una selección/exclusión de un sort en el file de clientes considerado antes, eligiendo la exclusión de clientes cuyo saldo es inferior a $ 1000, obtendremos un listado como el que sigue:

Relación de saldos con selección/exclusión

Estado	Cliente	Saldo ($)
Nevada	M & M Ltd.	2596,00
	Zilch & Sons	1976,55
New Jersey	NXR Ltd.	8870,77
	Wyatt & Hyatt	8345,98
	Acme Ltd.	5689,00

Los records pueden seleccionarse o excluirse especificando hasta cuatro claves de selección/exclusión. Estas claves son completamente independientes de las claves de sort, que pueden definirse al mismo tiempo. Esto significa que es posible seleccionar o excluir en campos diferentes de aquellos que se están ordenando. Si sólo se especifican las claves de selección/exclusión y no las claves de sort, el Olisort genera un file de salida que contiene los records seleccionados, pero en el mismo orden del file de entrada.
Si se especifican cadenas de texto alfanuméricas dentro de una clave de selección/exclusión, el Olisort controla que el contenido de la clave y el campo dentro del record sean de la misma longitud antes de confrontarlos.
La lógica de este procedimiento es la siguiente:

1 / El Olisort verifica si es más larga la clave de selección/exclusión o si es más largo el campo con el que debe confrontarse
2 / A efectos de la confrontación, el más largo se trunca a la longitud del más corto
3 / Luego se realiza la confrontación, carácter por carácter, y si todos los caracteres son iguales, se tiene una respuesta positiva

Si se tiene un campo en un record del tipo

Tubo de plomo 12 mm × 3 m

la clave de selección podrá ser

Tubo de plomo

En la clave de selección/exclusión pueden insertarse los llamados **caracteres de confrontación**:

< significa que el carácter de la clave se considerará menor que el correspondiente carácter del campo confrontado
> significa que el carácter de la clave se considerará mayor que el correspondiente carácter del campo confrontado
= significa que el carácter de la clave se considerará igual que el correspondiente carácter del campo confrontado

El uso de los caracteres de confrontación está ligado a la particular definición de la notación **leg** para las claves de selección/exclusión utilizadas en el Olisort. Mediante la leg, puede definirse la actuación de la selección/exclusión si la correspondiente clave es «menor que», «igual a» o «mayor que» el campo confrontado. Empleando los caracteres de confrontación pueden realizarse confrontaciones parciales o con claves genéricas.
Por ejemplo, supongamos que se tiene un file de componentes que contiene códigos del tipo ilustrado en la tabla de más abajo. Se desea efectuar una selección parcial de los componentes que pertenecen a la clase A, considerando solamente aquellos que tienen el código de montaje igual a 80. Entonces puede utilizarse una clave de selección/exclusión similar a la que sigue:

80 = = = = = = A

Extracto del file de componentes

Código de montaje	Código componente	Clase componente
94	543678	B
67	553478	A
80	887690	A
88	113425	D
80	984400	C
80	000789	A
56	722990	B

Utilizando la clave de selección/exclusión indicada anteriormente, de los códigos de componentes quedarán seleccionados los siguientes:

80-887690-A
80-000789-A

La adopción de más claves de selección/exclusión requiere el uso de elementos de conexión como AND u OR. Para que la selección tenga lugar, a los datos deben corresponder dos claves de selección/exclusión conectadas con un AND. En cambio, si están conectadas con un OR, basta con que corresponda una de ellas para que se realice la selección.
Por ejemplo, si al especificar la selección de los componentes indicados en la tabla anterior se utilizara combinación de claves:

80 = = = = = = A OR 80 = = = = = = C

se seleccionarían los siguientes códigos de componente:

80-887690-A
80-984400-C
80-000789-A

en cambio, si se utilizase la siguiente combinación de claves sobre los códigos de componentes de dicha tabla:

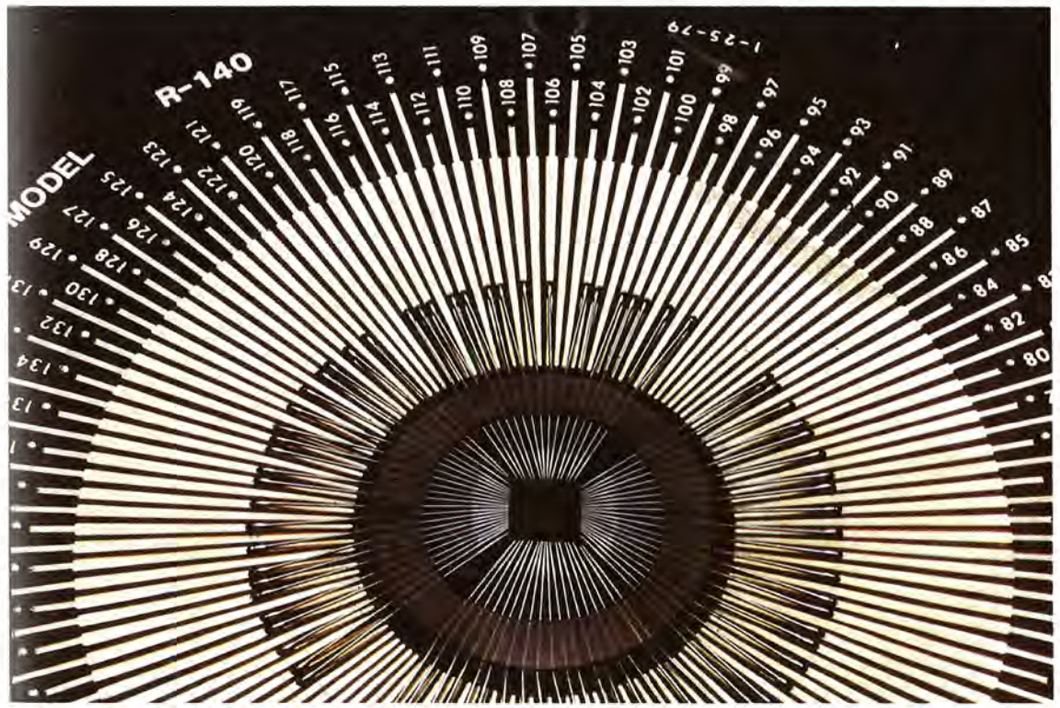
80 = = = = = = A AND 56 = = = = = = B

no se seleccionaría ningún código de componente.

El Olisort permite realizar el merge de dos files en entrada ordenados en un file en salida, o realizar un append de un file de entrada a continuación de otro, obteniendo así en la salida un file que contiene al mismo tiempo los records del primer y del segundo file en entrada.
Para realizar el merge de dos files de entrada deben especificarse las mismas claves de sort que se utilizarían en el caso de ordenar los dos files.
Para encadenar dos files de entrada no deben especificarse claves de sort. El segundo file de entrada se añade automáticamente a la cola del primero. El funcionamiento del Olisort permite utilizar en conjunto hasta cinco files, de los cuales están reservados los tres siguientes:

— INPUT 1: file indispensable para el uso del paquete
— INPUT 2: usado cuando se quiere utilizar la prestación merge y/o append
— OUTPUT: es un file de salida del sort, merge, append y selección/exclusión. Si el nombre del output fuese el mismo que el del input, la función de escritura se realizaría en el file de entrada.

Estrella de prueba para circuitos integrados.



Modalidad de ejecución. Para enviar a ejecución el Olisort hay dos maneras:
— incorporar los parámetros en una cadena de comandos de un programa Basic
— utilizar el programa de utilidad Olisortx para la creación interactiva de los parámetros y, por tanto, hacer trabajar el Olisort.

En ambos casos, los parámetros pasados tienen el mismo formato. Los parámetros pueden pasarse al Olisort:
— a través de un file de parámetros salvaguardado en disco (útil cuando se quiere hacer trabajar el mismo sort más de una vez). Puede crearse un file de parámetros utilizando el programa de utilidad Olisortp
— a través de una cadena de comandos que es una variable Basic con el nombre OLISORT.CMD$. Puede crearse una cadena de comandos de dos modos: utilizando el intérprete Basic, o bien utilizando el Olisortx; con este último método, la cadena de comandos puede salvaguardarse en disco

El uso de un file de parámetros no excluye una corta cadena de comandos en el programa Basic. Ésta sólo contiene pocos parámetros y una referencia al file de parámetros que contiene la mayor parte de ellos. Uno de los parámetros a pasar al Olisort especifica el modo de funcionamiento deseado.
Hay cuatro Modos Olisort.

El **Modo 0** permite el sort de un file, la selección o la exclusión de records de un file o el sort y la selección/exclusión al mismo tiempo. Los parámetros para el sort, para la selección o para el sort y la selección/exclusión ya deben existir en un file de parámetros del disco.
Por ejemplo, si se tiene un file principal que contiene artículos de almacén ordenados por códigos y dos files de transacción que contienen nuevas piezas de dos fuentes diferentes, el sort de los files transaccionales puede hacerse así:

1 / creación de un file de parámetros mediante el Olisortp para realizar el sort de los files transaccionales
2 / escritura de un programa Basic que llame dos veces el Olisort (utilizando el mismo file de parámetros cada vez) para realizar el sort de los dos files transaccionales. Esto comporta la preparación de una cadena de comandos y la definición del Modo 0. Esta cadena de comandos será muy corta, porque la mayor parte de los parámetros estará contenida en el file de parámetros del disco.

El **Modo 1** realiza el merge de dos files ordenados en uno solo, o añade un file al final de otro. En el disco también debe haberse creado un file de parámetros. Si los dos files se han ordenado y la información de clave de sort está especificada en el file de parámetros, se realizará un merge. Si en el file de parámetros no hay especificada la información clave de sort, los dos files se encadenarán. Por ejemplo, haciendo referencia a los files transaccionales y al file principal de los que ya se ha hablado, la actualización del file principal puede hacerse así:

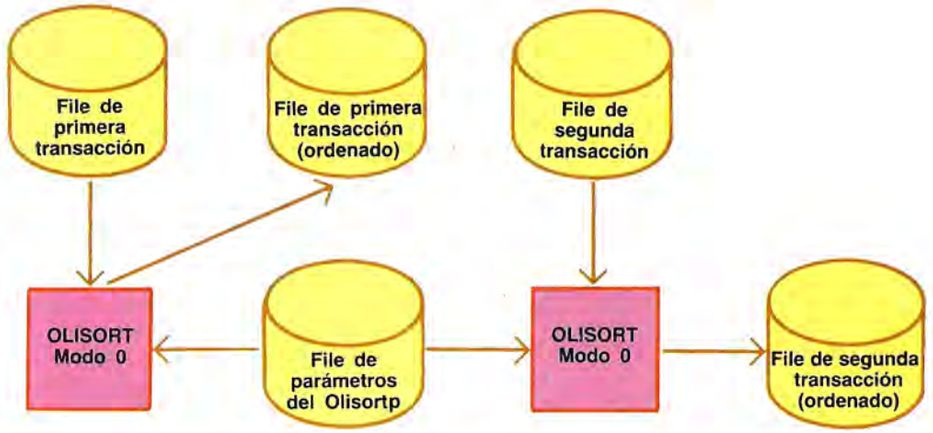
1 / usando el mismo file de parámetros creado para realizar el sort de los dos files transaccionales, se escribe un programa Basic que contenga una cadena de comandos que especifique el Modo 1. Los dos files de entrada se especifican como los dos files transaccionales ordenados
2 / una vez pasado el programa Basic, los dos files transaccionales quedarán fundidos (merge) en un file transaccional
3 / utilizando el mismo fiie de parámetros, se escribe un programa Basic que contenga una cadena de comandos que especifique el Modo 1. Los files de entrada son las transacciones fundidas (merge) y el file principal
4 / una vez pasado el programa Basic, se producirá un file principal actualizado

El **Modo 2** realiza las funciones del Modo 0, pero todos los parámetros se pasan a una cadena de comandos escrita en un programa Basic.
Por ejemplo, uno de los files transaccionales de arriba puede ordenarse de la siguiente manera:

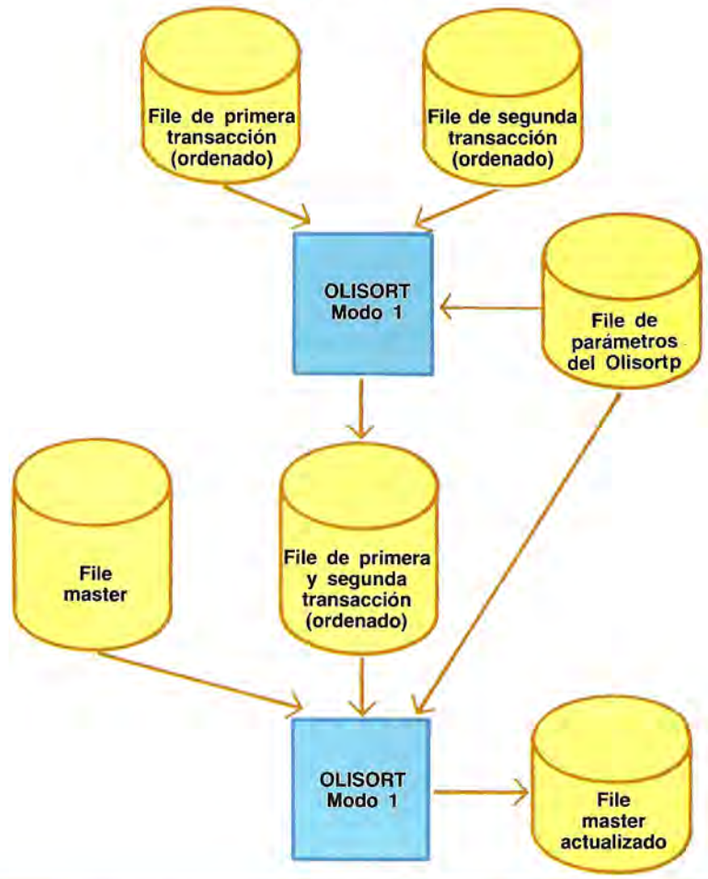
1 / escribiendo un programa Basic que contenga una cadena de comandos que especifique el Modo 2
2 / después de la ejecución del programa Basic se produce un file transaccional ordenado

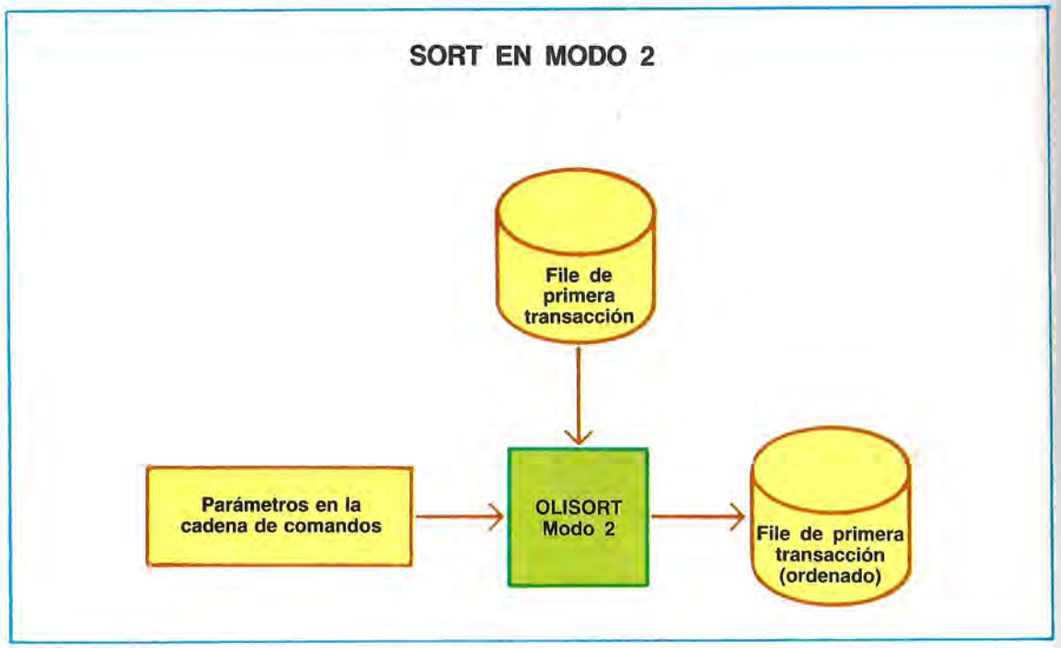
El **Modo 3** es idéntico al Modo 2, pero en el disco se crea un file de los parámetros establecidos en las cadenas de comandos. Esta prestación es útil si en un programa Basic se ha escrito una cadena de comandos y se quiere preparar con ésta un file de parámetros.
Por ejemplo, el sort de los dos files transaccionales puede realizarse de la siguiente manera:

SORT EN MODO 0



MERGE EN MODO 1





1 / se escribe un programa Basic que contenga una cadena de comandos que especifique el Mode 3. El file de entrada para este sort es el primer file transaccional
2 / con la ejecución del programa Basic del punto 1 se ordena el primer file transaccional y se genera un file de parámetros
3 / entonces se escribe otro programa Basic que contenga una cadena de comandos que especifique el Modo 0 y que utilice el file de parámetros producido en el último paso. Esta vez, el file de entrada es el segundo file transaccional
4 / con la ejecución del programa se ordena el segundo file transaccional

Desarrollo de un procedimiento de gestión de datos

El empleo usual del calculador contempla la memorización y el tratamiento de grandes cantidades de datos. En las máquinas más evolucionadas existen programas dedicados a la gestión de los datos con los que pueden generarse y gestionarse archivos incluso muy complejos.
En el microordenador y el ordenador personal, el software ya preparado tiene formas menos generalizadas. En estos casos es conveniente la escritura de un programa orientado a la aplicación especificada. Si se ha optado por esta última solución, puede encontrarse un compromiso que evite la necesidad de volver a escribir completamente el software para cada nueva aplicación: la solución consiste en parametrizar las subrutinas principales. De esta forma, para cada aplicación será suficiente con variar sólo algunos parámetros. En la mayor parte de los casos, las funciones que debe realizar un programa pueden clasificarse así:

— Introducción de datos
— Tratamiento
— Memorización en disco

Preparando una subrutina parametrizada para cada función puede formarse un programa de aplicación cualquiera encadenando adecuadamente las rutinas correspondientes a las funciones a realizar. El primer paso consistirá en definir el tipo y la longitud de cada uno de los datos que deberán tratarse.
Estos parámetros servirán para escribir la rutina de introducción y la de memorización.

La fase de introducción

Se quiere preparar un record que contenga los siguientes campos:

Nombre		9 caracteres
Código		4 caracteres
Fecha	Día	2 caracteres
	Mes	2 caracteres
	Año	2 caracteres

Para la introducción de los datos deberá construirse una máscara vídeo que prevea para cada campo el número exacto de caracteres y el control del tipo (las fechas, por ejemplo, deberán ser numéricas). Si en el transcurso del programa hubiese que introducir otros datos con formato diferente, debería reescribirse una rutina de introducción que prevea una máscara adecuada. Esta eventualidad puede eliminarse escribiendo la primera rutina del modo más generalizado posible (o sea parametrizada).

Preparación de la máscara vídeo. La estructura de una máscara vídeo puede definirse a través de un cierto número de parámetros:

— Fila y columna de principio (posicionado sobre la pantalla)
— Número de líneas que debe contener (cada línea es un campo)
— Descripción de cada línea
— Longitud en caracteres de cada campo
— Tipo del campo

En el ejemplo anterior tendremos:

— Posición de principio:	línea 4, columna 6 (son valores arbitrarios)
— Líneas contenidas:	5
— Descripciones:	1 - Nombre 2 - Código 3 - Día 4 - Mes 5 - Año
— Longitud de cada campo:	Campo 1 = 9 caracteres Campo 2 = 4 caracteres etc.
— Tipo de cada campo:	Campo 1 = alfabético Campo 2 = numérico etc.

Para presentar la máscara en el vídeo, el cursor deberá llevarse a la columna de principio (6 y 4), visualizar las descripciones y, junto a cada una de éstas, escribir un número de puntos iguales a la longitud de cada campo.
Las descripciones pueden memorizarse en una variable dimensionada, como por ejemplo DS$(5); escribiendo DS$(1) se obtendrá la descripción del primer campo, con DS$(2) la del segundo, y así sucesivamente. La puntuación puede obtenerse escribiendo, después de cada descripción, una cadena que contenga un número de puntos igual al número de caracteres previstos en el campo.
Por ejemplo, escribiendo A$ = "." y B$ = STRING$(9,A$) se genera la cadena B$ que contiene nueve puntos. La primera línea de la máscara puede prepararse con la instrucción

```
PRINT DS$(1)+ " " +B$
```

La variable DS$(1) contiene la descripción; B$ contiene la puntuación y el espacio insertado sirve para separar la parte descriptiva de la propia máscara. La forma más sencilla de proporcionar los datos a la rutina de preparación de las máscaras consiste en utilizar una serie de DATA y de READ. Por ejemplo, si se quieren transferir las descripciones consideradas anteriormente en la variable (matriz) DS$, las instrucciones a introducir son las siguientes:

```
10 RESTORE 30
20 FOR I = 1 TO 5: READ DS$(I):NEXT I
30 DATA "1 — Nombre"
40 DATA "2 — Código"
50 DATA "3 — Día"
60 DATA "4 — Mes"
70 DATA "5 — Año"
```

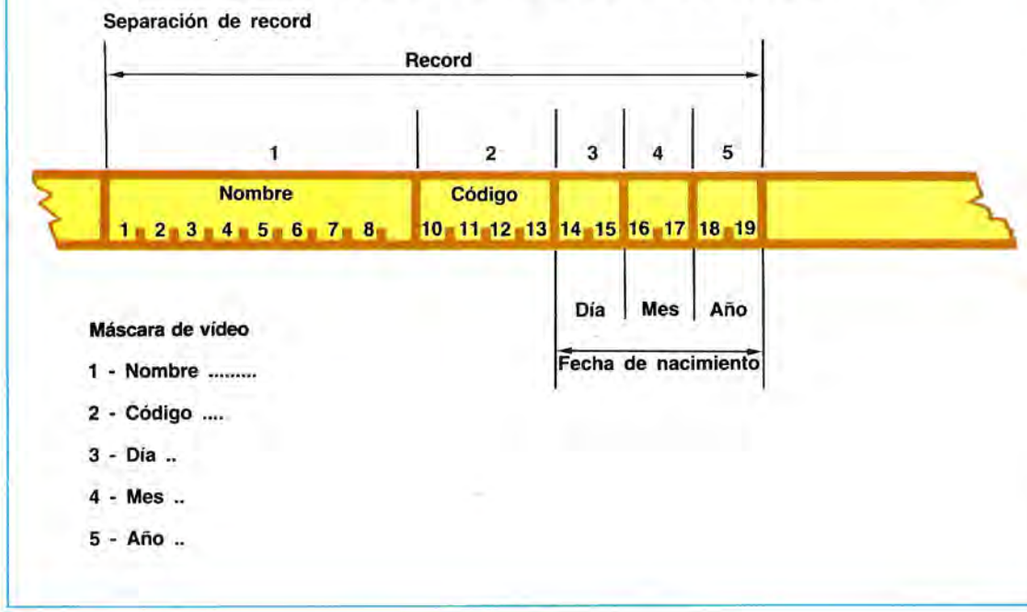
En la subrutina de memorización en disco para cada campo, además de la longitud, deberá definirse la posición: así, el campo Nombre (ver el gráfico de abajo) ocupa desde el byte 1 al 9, el campo Código del byte 10 al 13, etc. Estos valores también pueden asignarse con los DATA. En los listados que siguen se han adoptado los siguientes nombres:

PD(I) = Byte de principio del campo I
PA(I) = Byte de final del campo I
TP(I) = Tipo del campo I

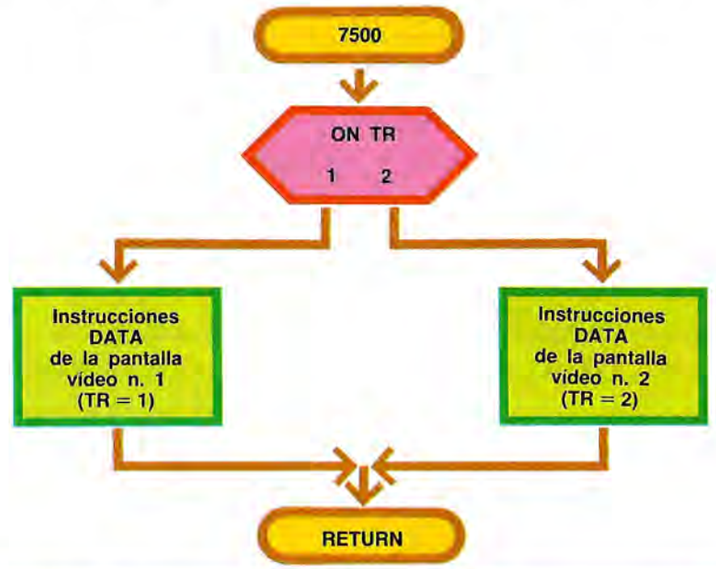
Para definir la máscara que se examina deben asignarse los valores

PD(1) = 1 PA(1) = 9 (primer campo, de 1 a 9)
PD(2) = 10 PA(2) = 13 (segundo campo, de 10 a 13)
PD(3)=14 PA(3)=15 (tercer campo, de 14 a 15)

MEMORIZACION DE UNA AGENDA Y MASCARA DE VIDEO



SUBRUTINA DE PREPARACION DESCRIPCION Y LONGITUDES DE CAMPOS



De forma análoga puede definirse el tipo de cada campo. Adoptando la convención

TP(I) = 1 si el campo I es numérico
TP(I) = 2 si el campo es alfanumérico
TP(I) = 3 si el campo es de sólo presentación

para el primer campo se tendrá el tipo 2 y, para todos los demás, el tipo 1, o sea TP(1) = 2 y TP(2) = TP(3) = TP(4) = TP(5) = 1. El código 3 puede ser útil para definir un campo que sólo deba presentarse y en el que el operador no podrá aportar ninguna variación.
La asignación de los valores se obtiene a través de una serie de instrucciones READ y DATA sobre cada una de las variables a inicializar. Es posible que en un programa complejo sea necesario disponer de diversas máscaras a presentar en tiempos sucesivos. Las variables utilizadas para una deben ser las mismas que las utilizadas para las otras; por tanto, para cada presentación deben activarse las oportunas DATA para asignar a las variables los valores específicos de la máscara examinada. Esta función puede obtenerse en una rutina en la que el número de la máscara a presentar se ha pasado a la variable TR. Según el valor de TR, la rutina selecciona el grupo DATA y transfiere los adecuados valores a las variables de presentación.

En esta rutina también puede preverse el nombre del file en el que irán a memorizarse los datos (variable NM$) y su número (NF).
Al hacer la llamada, el único parámetro será el número de la máscara; en la salida se restituirán todas las variables que podrán utilizarse en las otras subrutinas, así como el nombre y el número del file en que se memorizarán los datos.
En esta página se ha representado el diagrama de flujo de una rutina que da dos máscaras (TR puede asumir los valores 1 y 2), pero que puede ampliarse con nuevos bloques DATA. El correspondiente listado está en la pág. 698.
Cada DATA inherente a una nueva máscara vídeo deberá tener una estructura similar al bloque comprendido entre las líneas 7574 y 7620. La parte inicial del bloque (línea 7576) define la posición de la máscara que hay en presentación (KC = 2, KR = 3), el número de líneas que la componen (10) y el número del file (NF = 1); la parte central (líneas 7578 a 7612) asigna los valores a las variables que caracterizan la máscara. En este ejemplo, el record tiene una longitud de 100 bytes (ver la línea 7612), y esta longitud se asigna en línea 7614 [en la variable LN = PA(N)] junto al nombre del file (en la variable NM$). Téngase en cuenta que el nombre del file también puede contener la especificación de la unidad de disco. En el listado, el file de la prime-

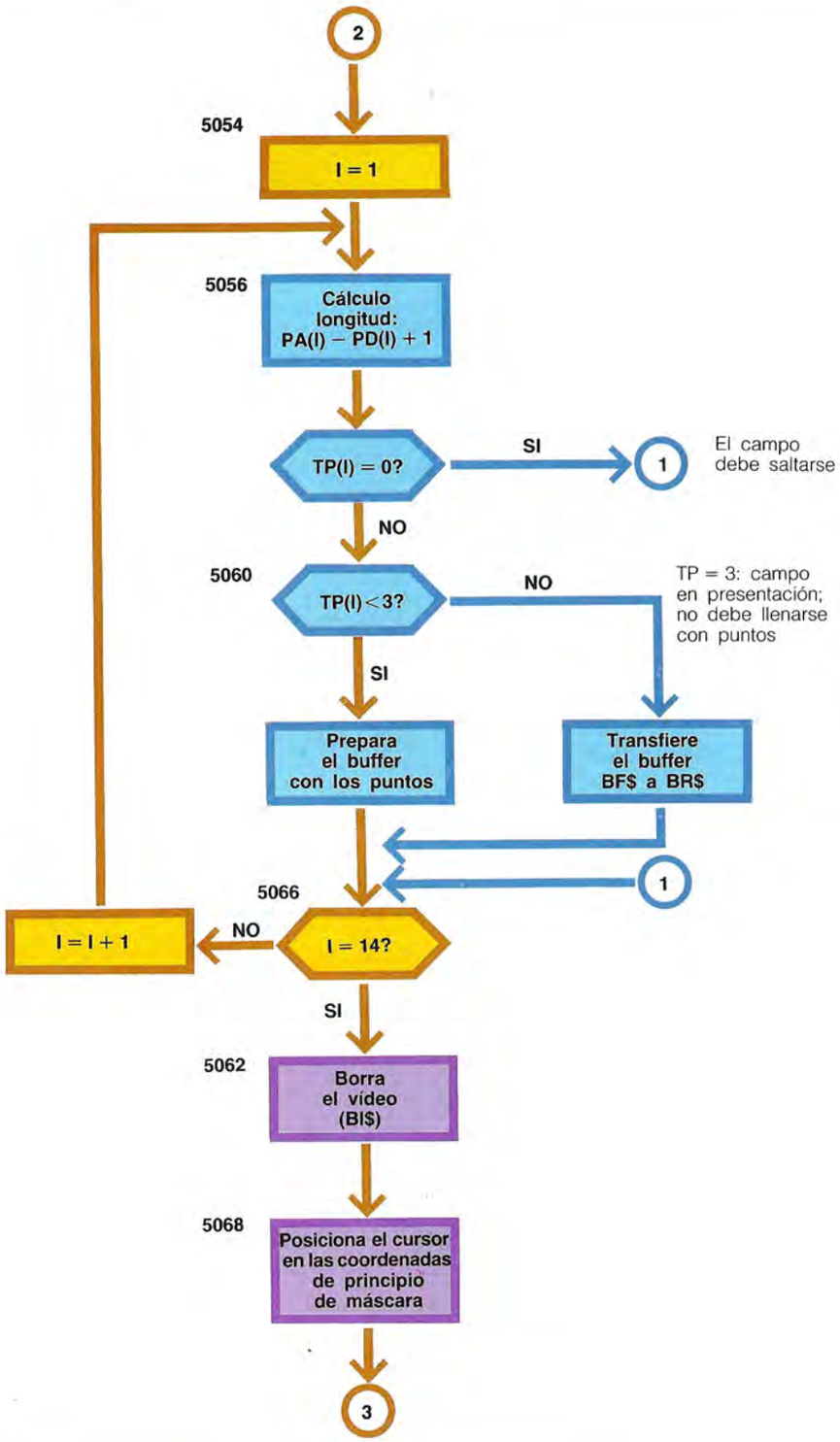
PREPARACION DE DESCRIPCIONES Y LONGITUDES DE CAMPOS

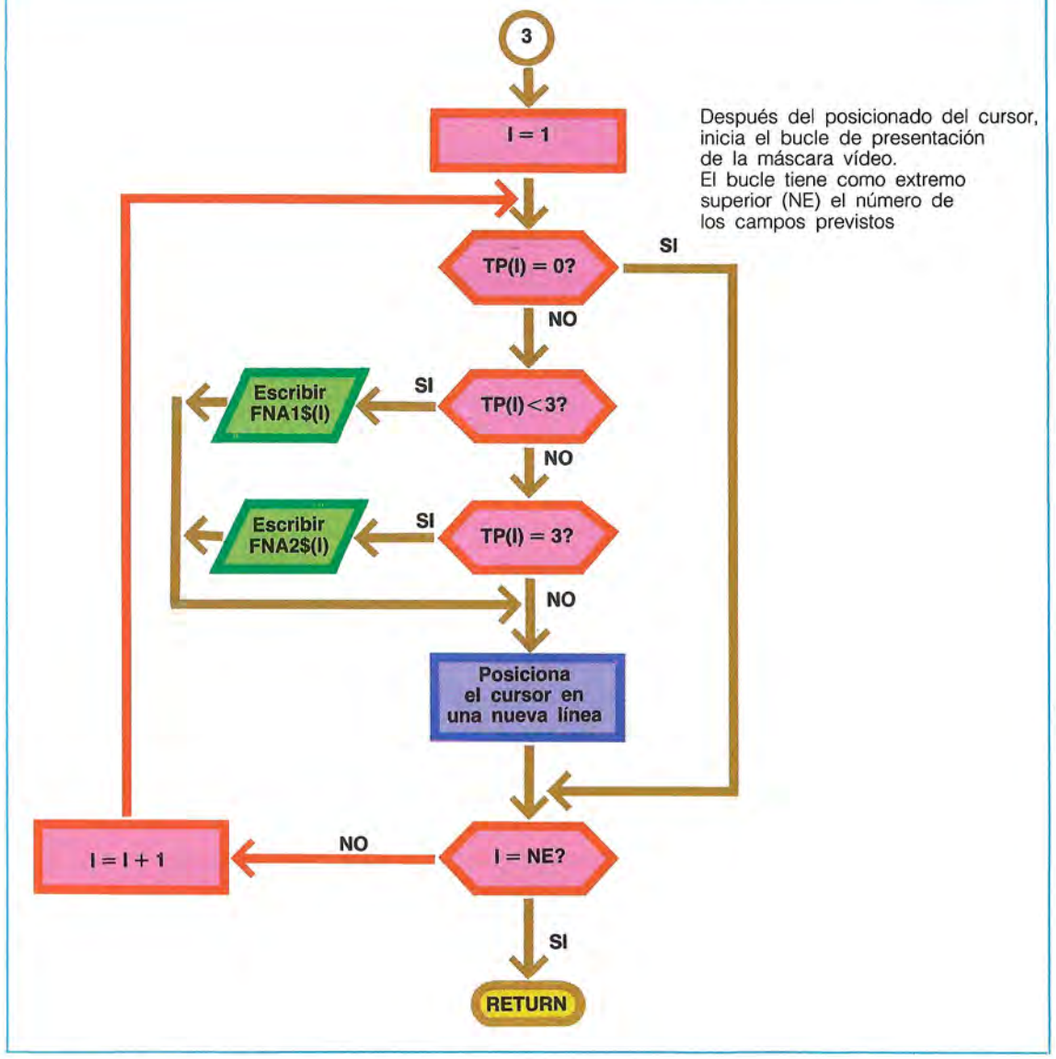
```
7500 '
7501 '   FILE : DATA
7502 '   VERS. ENCICLOPEDIA
7503 '   **DATA**
7504 '   ENTRADA     TR=NUMERO MASCARA
7506 '
7508 '   SALIDAS
7510 '               KC=POSICION PRIMERA COLUMNA
7512 '               KR=POSICION PRIMERA LINEA EN PANTALLA VIDEO
7514 '             NR(*)=NUMERO DE LINEAS
7516 '             NF=TR=NUMERO DEL FILE
7518 '           DS$(*)=DESCRIPCIONES DE LAS LINEAS
7520 '             TP(*)=TIPO DE CAMPO VER : VIDE - 5000
7522 '             PD(*)=PUNTERO PRINCIPIO CAMPO
7524 '             PA(*)=PUNTERO FINAL CAMPO
7526 '                LN=NUMERO DE CARACTERES
7528 '              NM$=NOMBRE DEL FILE
7534 ON TR GOTO 7576,7621    '   SELECCION DEL FORMATO DESEADO
7570 '
7572 '
7574 '  TR=1
7576 KC=2:KR=3:NR(TR)=10:N=NR(TR):NF=1
7578 RESTORE 7582
7580 FOR I=1 TO N:READ DS$(I): NEXT I
7582 DATA " 1 - Descripción Mercaderias"
7584 DATA " 2 - Porcentaje I.T.E."
7586 DATA " 3 - Unidad de Medida"
7588 DATA " 4 - Costo"
7590 DATA " 5 - Precio"
7592 DATA " 6 - Cargo Acumulado"
7594 DATA " 7 - Valor Acumulado"
7596 DATA " 8 - Descarga Acumulada"
7598 DATA " 9 - Valor Acum. Descarga"
7600 DATA "10 - ....................."
7602 FOR I=1 TO N:READ TP(I):NEXT I
7604 DATA 2!,1!,2!,2!,2!,3!,3!,3!,3!,3!
7606 FOR I=1 TO N:READ PD(I):NEXT I
7608 DATA 1!,21!,23!,31!,38!,45!,53!,64!,72!,82!
7610 FOR I=1 TO N:READ PA(I):NEXT I
7612 DATA 20!,22!,30!,37!,44!,52!,62!,71!,81!,100!
7614 LN=PA(N):NM$="A:MAANA"
7616 '
7618 N=N+1:FOR I=N TO 14 : DS$(I)=" ":TP (I)=0: PD(I)=0:PA(I)=0:NEXT I
7620 RETURN
7621 ' TR=2
7622 KC=2:KR=3:NR(TR)=9:N=NR(TR):NF=2
7624 RESTORE 7628
7626 FOR I=1 TO N:RFAD DS$(I): NEXT I
7628 DATA" 1 - CODIGO MERCADERIAS"
7630 DATA" 2 - FECHA MOVIMIENTO"
7632 DATA" 3 - CANTIDAD EN CARGO"
7634 DATA" 4 - COSTO"
7636 DATA" 5 - CANTIDAD DESCARGADA"
7638 DATA" 6 - PRECIO"
7640 DATA" 7 - ....................."
7648 DATA" 8 - ....................."
7644 DATA" 9 - Movimiento transf."
7646 FOR I=1 TO N:READ TP(I):NEXT I
7648 DATA 1!,2!,2!,2!,2!,2!,2!,2!,3!
7650 FOR I=1 TO N:READ PD(I):NEXT I
7652 DATA 1!,4!,9!,17!,24!,32!,39!,65!,91!
7654 FOR I=1 TO N:READ PA(I):NEXT I
7656 DATA 3!,8!,16!,23!,31!,38!,64!,90!,91!
7658 LN=PA(N):NM$="A:MASET"  'NM$ = NOMBRE DEL FILE EN QUE SE MEMORIZARAN
7659 '                              LOS DATOS DE LA PANTALLA 2
7660 '
7662 N=N+1 : FOR I=N TO 14:DS$ (I)=" ":TP (I)=0:PD(I)=0:PA(I)=0:NEXT I
7664 RETURN
```


ra máscara se llama MAANA y reside en el disco A. La última línea (7618) sirve para poner a cero los elementos inicializados de las matrices. En esta máscara se han utilizado 10 campos, mientras que en el programa se prevén 14; las variables que sobran se ponen a cero (si son numéricas) o se anulan (si son cadenas). La puesta a cero se obtiene con un bucle que empieza por N + 1, siendo N el número de elementos utilizados para cada matriz (línea 7576).

Presentación de la máscara vídeo. Después de llamar la rutina de preparación de las variables que caracterizan la máscara, ésta debe visualizarse. Este paso encuentra dos dificultades: el alineamiento de los campos y la presencia de campos del tipo 3 (sólo de presentación). Las descripciones de cada campo pueden tener diferente longitud y, escribiendo inmediatamente después la descripción del punteado representativo del campo introducido, se obtendrá una máscara no alineada. Para eliminar el efecto debido a la diversidad de longitudes de las descripciones, entre la descripción y el punteado debe insertarse un número de espacios en blanco suficiente para producir el columnado de todas las puntuaciones. El número de espacios a insertar puede determinarse extrayendo la longitud máxima de las descripciones e insertando a la derecha de cada una de ellas un número de espacios igual a la diferencia entre la longitud de la descripción más larga y la descripción considerada. Así, si la descripción más larga ocupa 12 caracteres, si se escribe una de 8 caracteres, la puntuación deberá empezar tras 4 espacios. En realidad se considerará siempre un espacio de más, ya que para la cadena más larga, el punteado quedaría demasia-

SUBRUTINA DE PRESENTACION DE MASCARAS VIDEO





do cerca de la escritura. En las págs. 699, 700 y 701 aparece el diagrama de flujo de la subrutina de presentación de la máscara vídeo, y en la 702, el listado. En esta subrutina se utilizan algunas variables inicializadas en el programa principal, en particular:

BL$ = cadena de espacios en blanco
BB$ = cadena de puntos

La búsqueda de la longitud máxima de las descripciones se realiza en las líneas 5028 a 5034, mientras que en las líneas 5038 a 5044 se preparan las variables de la cadena AT$, que contienen tantos espacios en blanco como requiere la descripción en examen. La generación de una cadena AT$ se obtiene tomando N caracteres de la BL$ (instrucción 5042), aunque podría obtenerse con la función SPACE$(N). La primera forma se ha elegido porque permite variar el símbolo en cada rutina, sustituyendo el contenido de BL$ en el main. La rutina prevé dos formas de presentación: para los campos en introducción y para los que están en presentación. La selección de una u otra se hace en las líneas 5082 y 5084. Para los campos en introducción, el tipo (TP) es 1 o 2 (o sea inferior a 3); las instrucciones anteriores han preparado en la función FNA1$ las descripciones más el punteado. En cambio, si es del tipo 3, las instrucciones an-

PRESENTACION DE LAS MASCARAS VIDEO

```
5000 '
5002 ' **** FORMATOS VIDEO ****
5004 ' FILE = VIDE
5005 ' VERS. ENCICLOPEDIA
5006 ' ------------
5008 ' TIP =
5010 '         1 NUMERICO
5012 '         2 ALFANUMERICO
5014 '         3 SOLO EN PRESENTACION (Introducción Impedida)
5020 '   ***  ENTRADA TR=Tipo de Record
5024 '
5026 P=TR
5028 MD=0
5030 FOR I=1 TO 14:N=LEN(DS$(I))
5032 IF N>MD THEN MD=N
5034 NEXT I
5036 '
5038 FOR I=1 TO 14:N=LEN(DS$(I)):M=MD-N+1
5040 IF M=0 THEN AT$(I)=" ": GOTO 5044
5042 AT$(I)=LEFT$(BL$,M)
5044 NEXT I
5046 FOR I=1 TO 14:IF TP (I)=0 GOTO 5052
5048 IF TP(I)=3   GOTO 5052
5050 BF$(I)=BL$
5052 NEXT I
5054 FOR I=1 TO 14:N=PA(I)-PD(I)+1
5056 IF TP(I)=0 GOTO 5066
5058 LD (I)=N
5060 IF TP (I)<3 THEN B$(I)=LEFT$(BB$,N):BF$(I)=LEFT$(BL$,N):GOTO 5066
5062 ' ** CAMPOS RENOVADOS **
5064 BR$(I)=LEFT$(BF$(I),N)
5066 NEXT I
5068 PRINT BI$
5072 X=KC:Y=KR:GOSUB 6900
5074 NE=NR(TR)
5078 FOR I=1 TO NE
5080 IF TP(I)=0 GOTO 5088
5082 IF TP(I)<3 THEN PRINT FNA1$(I)
5084 IF TP(I)=3  THEN PRINT FNA2$(I)
5086 Y=Y+1:GOSUB 6900
5088 NEXT I
5090 RETURN
5092
```

teriores han preparado FNA2$, que contiene las descripciones y los valores a presentar (contenidos en los buffers de datos BF$). Para obtener la presentación en pantalla de una máscara, debe llamarse la subrutina 7500, que contiene las DATA, y la 5000. Así, las llamadas para obtener la primera pantalla son:

```
TR = 1
GOSUB 7500
GOSUB 5000
```

En la 7500 (ver pág. 698) se han previsto, para el formato 1 (TR = 1), cinco campos de sólo presentación (los últimos, línea 7604). La llamada anterior no les asigna ningún valor y los campos, o no son presentados, o tienen un contenido erróneo, resto de anteriores tratamientos. Para obtener la correcta presentación debe inicializarse antes de llamar la 5000. Los campos en cuestión son los números 6, 7, 8, 9 y 10 (con los significados indicados en las descripciones, líneas 7592 a 7600) y tienen estas longitudes:

campo 6	del 45 al 52	= 8 caracteres
campo 7	del 53 al 62	= 10 caracteres
campo 8	del 64 al 71	= 8 caracteres
campo 9	del 72 al 81	= 10 caracteres
campo 10	del 82 al 100	= 19 caracteres

Poniendo en los correspondientes buffers una serie de cadenas, se tendrá la presentación sobre la máscara. Por ejemplo, la llamada

```
TR = 1
GOSUB 7500
BF$(9) = "0123456789"
GOSUB 5000
```


genera la presentación de la cadena "0123456789" en la línea 9 de la máscara (cada buffer tiene como índice el número de línea en la que se posiciona).

Introducción de los datos. Después de haber llamado la subrutina 5000 (presentación de las máscaras vídeo) debe pasarse a la fase de introducción. Las funciones a desarrollar (el diagrama de flujo está en la págs. 704 y 705 y el listado en las págs. 706, 707 y 708) son:

— Posicionado del cursor
— Lectura de un carácter (sin eco)
— Control sobre el carácter leído

El éxito del control sobre el carácter leído determina las siguientes acciones. Puede haber cuatro casos:

1 - el carácter es un código de control
2 - el carácter no es homogéneo con el campo
3 - el carácter es aceptado
4 - el carácter no es reconocido

En los casos 2 y 4 debe volver a posicionarse el cursor en el mismo punto y pedir una nueva introducción. En el caso 3, el carácter se transfiere al buffer correspondiente; por ejemplo, si el cursor se encuentra sobre la línea 4, el carácter introducido se inserta en el buffer 4 [BF$(4)]. Finalmente, en el caso 1, deben desarrollarse las funciones asociadas al código reconocido. En la gestión de la máscara debe preverse la posibilidad de efectuar correcciones; por tanto, debe poderse saltar con el cursor de una línea a otra o de un carácter a otro de una misma línea. Estos desplazamientos están asociados a algunas teclas funcionales determinadas, cuya activación determina un código numérico. Todos los códigos previstos en la subrutina se memorizan en la variable dimensionada TF (inicializada en el main), por lo que para determinar si la tecla activa un control, basta con verificar que el valor numérico generado está comprendido entre los previstos (líneas 6078 a 6086). En caso afirmativo se salta en función del valor (líneas 6190 a 6200) a la adecuada zona del programa. En el ejemplo, los códigos activos son 10, 8, 12, 11, 16, 17 y 13.
Las teclas habilitadas (a las cuales les corresponde un código reconocido) están divididas en dos tipos. Al primero pertenecen las que desarrollan funciones internas de la máscara y, al segundo, las teclas que causan la salida de la subrutina. El mecanismo de gestión de las teclas que desarrollan funciones internas presenta alguna dificultad. Para ilustrar sus aspectos principales consideraremos las dos teclas de desplazamiento vertical. Las funciones que generan los respectivos códigos son:

— Desplazamiento de una línea hacia arriba, código 11 (línea 6240)
— Desplazamiento de una línea hacia abajo, código 10 (línea 6210)

La lógica seguida bajo la influencia de una de estas teclas puede resumirse así:

— Incremento (o decremento, según el desplazamiento) del contador de línea RR
— Incremento (o decremento) de la posición vertical (Y)
— Puesta a cero del contador de caracteres (RC)
— Posicionado horizontal al principio del campo (X = X0)

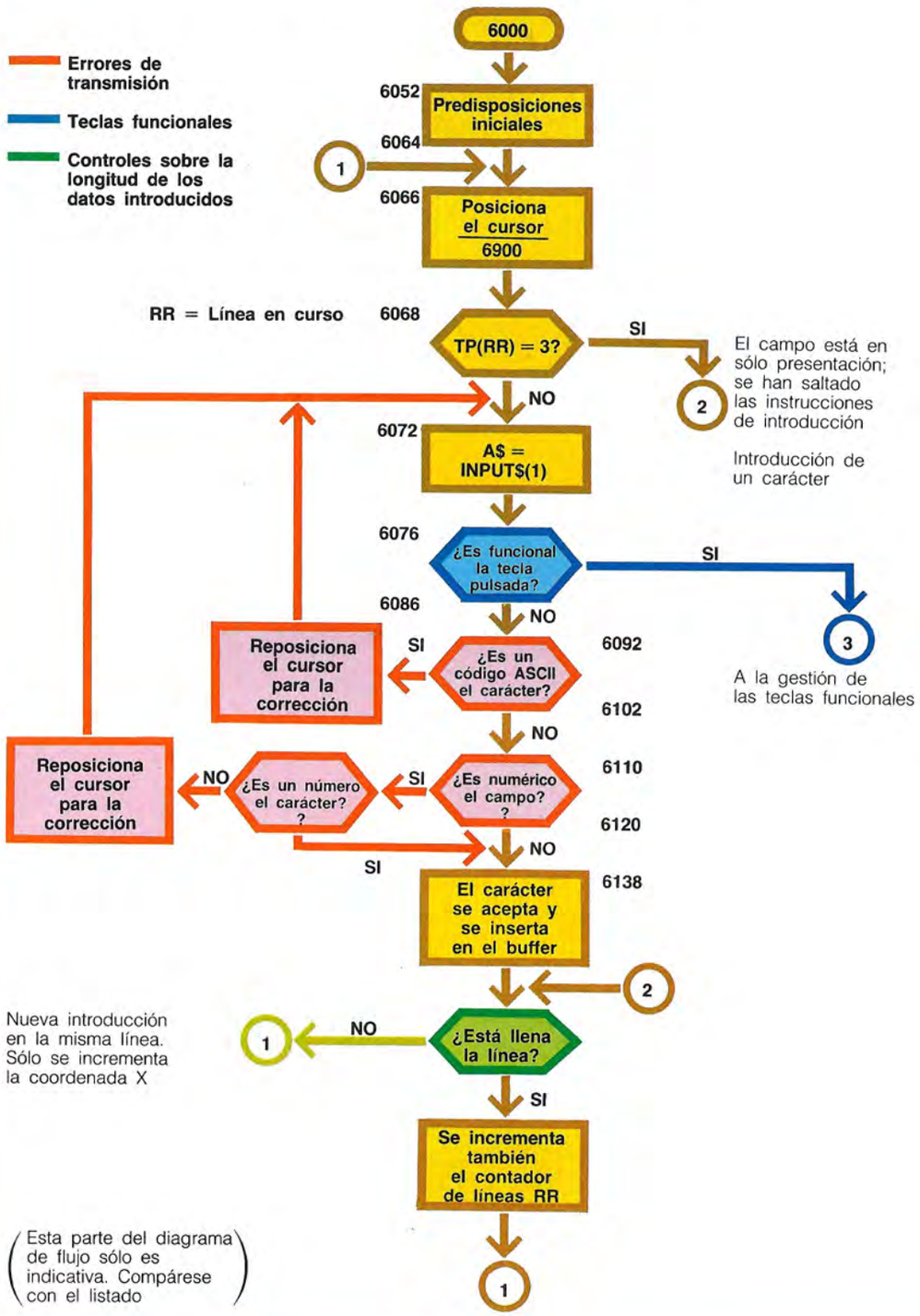
Supongamos, por ejemplo, que se está en la línea 3 (RR = 3). Los datos que van a introducirse pertenecen al buffer número 3 [BF$(3)]. Activando el código 11, se tendrá

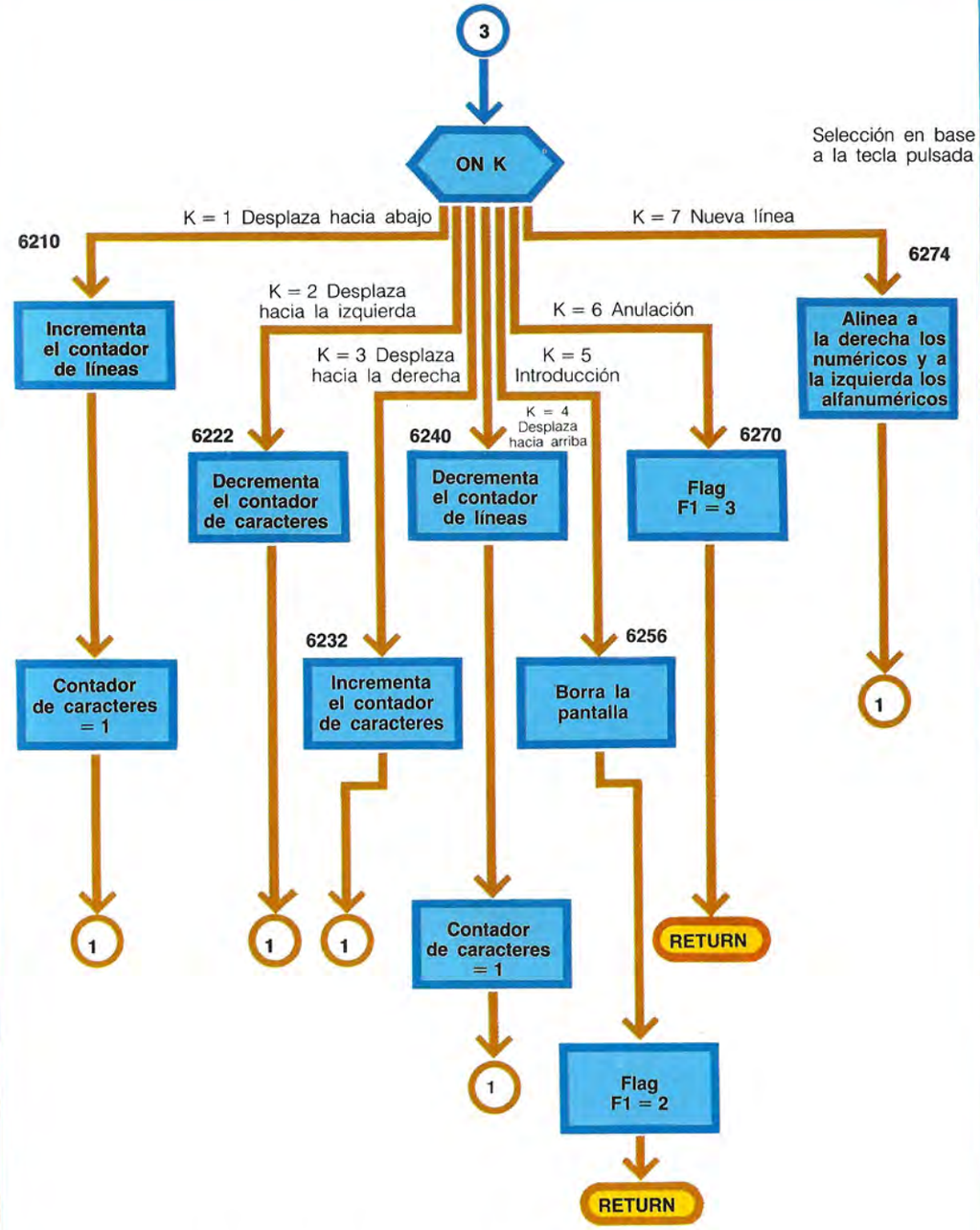
— Decremento del contador de líneas (RR = 3 – 1 = 2) para que apunte al buffer número 2
— Decremento de la posición (Y = Y – 1) para llevar el cursor a la línea precedente
— Puesta a cero del contador de caracteres (RC = 0), puesto que en el buffer 2 todavía no se ha introducido ningún carácter, e igualación de la posición horizonal del cursor (X) a la del principio del área de datos (X0)

Estas funciones se realizan en las líneas 6240 a 6252, que también efectúan algunos controles. Por ejemplo, la línea 6242 impide un desplazamiento hacia arriba si el cursor ya se encuentra en la primera posición. Las otras teclas funcionales internas desarrollan estas funciones:

— Desplazamiento a la derecha de una posición, código 12, línea 6232
— Desplazamiento a la izquierda de una posición, código 8, línea 6222
— Return, código 13, línea 6274 (tecla RT)

SUBRUTINA DE GESTION MASCARAS VIDEO





Obsérvese que de la rutina sólo puede salirse pulsando las teclas funcionales que corresponden a la introducción o a la anulación. Según la tecla, el flag se pone igual a 2 o a 3. El main controla este valor y prosigue con la introducción (flag = 2) o anula los datos y vuelve a una nueva introducción (flag = 3).

GESTION MASCARAS DE VIDEO

```
6000 '
6002 ' ** GESTION MASCARA **
6004 '
6005 ' ------------------------------------------------------------------------
6006 ' FILE : CURS
6007 ' VERS. ENCICLOPEDIA
6008 ' GESTION DE CAMPOS   RESULTADO
6010 ' ----------------
6012 '    TECLA     CODIGO     FLAG     FUNCION
6014 '      1         16        2      INTRODUCCION
6016 '      2         17        3      ANULACION
6038 '   VALORES EN TF(12)
6040 '  MD = Máxima longitud Descripciones
6042 '  TR = P DATA ENTRY EN CURSO
6044 '
6046 ' COORDENADAS EN CURSO
6048 '                          RC = Columna
6050 '                          RR = Línea
6052 XO=KC+MD+1:YO=KR
6054 '
6056 '
6060 P=TR
6062 RC=0:RR=1:F1=0      ' RC  RR  Apuntan al Buffer de Datos
6064 X=XO:Y=YO
6066 GOSUB 6900
6068 IF TP(RR)=3 GOTO 6174  ' Salta línea por ser campo protegido
6072 A$=INPUT$(1)
6074 GOSUB 6900
6076 V=ASC(A$)
6078 K=0
6080 FOR I=1 TO 20:IF TF(I)=0 GOTO 6084
6082 IF V=TF(I) THEN K=I
6084 NEXT I
6086 IF K<>0 GOTO 6190        '  *** Gestión Ordenes
6088 PRINT A$;
6090 '
6092 ' ** Controles sobre datos ASCII
6094 '
6096 IF TP(1)=0 THEN GOSUB 6368         ' Control primera línea
6098 IF TP(1)=0 GOTO 6066
6100 IF TP(RR)=0 THEN ER=2:PV=3:RETURN
6102 IF V>31 AND V<123 GOTO 6110        Ch. es ASCII : a los controles
6104 ' **Carácter no reconocido
6106 CK=3:GOSUB 6900:PRINT CHR$(7):GOTO 6072
6108 '
6110 ' ** CONTROL NUMERICIDAD
6112 '
6114 N=-1
6116 IF V>47 AND V<58 THEN N=1 ' **n=1 el carácter es numérico
6118 IF TP(RR)<>1 GOTO 6138          ' No es numérico
6120 IF N=1 AND TP (RR)=1 GOTO 6138 ' Control núm. O.K.
6122
6124 ' ***** Los campos renovados no tienen control
6126 '
6128 '
6130 ' **ERROR carácter no numérico
6132 '
6134 '
6136 PRINT CHR$(7):GOSUB 6900:GOTO 6072
6138 '
6140 ' Carácter aceptado
6142 RC=RC+1
6144 N=LD(RR):N8=LEN(BF$(RR))
6146 IF N8<N THEN N8=N
```



```
6148 M=RC-1
6150 BV$=LEFT$(BF$(RR),M)+A$
6152 IF RC=1 THEN BV$=A$
6154 N2=N-RC:IF N2<=0 THEN BR$=" ":GOTO 6158
6156 BR$=RIGHT$(BF$(RR),N2)
6158 BF$(RR)=BV$+BR$
6162 IF RC=N THEN RR=RR+1:RC=0:Y=Y+1:X=XO-1
6164 IF RR>NR(P) THEN RR=1 : Y=YO
6166 X=X+1
6168 IF TP (RR)=0 THEN GOSUB 6358
6170 IF RF=1 GOTO 6334
6172 GOTO 6066
6174 IF V=158 THEN RR=RR-1:Y=Y-1
6176 IF RR=0 THEN RR=1: Y=YO
6178 IF V=158 GOTO 6066
6180 RR=RR+1: Y=Y+1
6182 IF RR>NR(P) THEN RR=1: RC=0:X=XO: Y=YO
6184 IF TP (RR)=0 THEN GOSUB 6358
6186 GOTO 6066
6188 '
6190 ' *** GESTION ORDENES ***
6192 '
6194 ' K = Puntero a la orden en TF(12)
6195 ' CODIGOS: 10,8,12,11,16,17,13
6196 '             1    2    3    4    5    6    7
6198 ON K GOTO 6210,6222,6232,6240,6256,6270,6274
6200 ' ** ERROR
6202 '
6204 '
6206 '
6208 ER=5:PV=5:RETURN
6210 ' *** V=10 abajo
6212 '
6214 RR=RR+1:RC=0:X=XO:Y=Y+1
6216 IF RR>NR(P) THEN RR=1:Y=YO
6218 IF TP (RR)=0 THEN GOSUB 6368
6220 GOTO 6066
6222 ' *** V=8 a la izquierda
6224 IF RC=0 GOTO 6072
6226 RC=RC-1:X=X-1
6228 IF TP(RR)=0 THEN GOSUB 6368
6230 GOTO 6066
6232 ' *** V=12 a la derecha
6234 IF RC=LD(RR) GOTO 6068
6236 RC=RC+1:X=X+1
6238 GOTO 6066
6240 ' *** V=11 arriba
6242 IF RR=1 GOTO 6068
6244 RR=RR-1:Y=Y-1:RC=0:X=XO
6246 IF TP (RR)<>0 GOTO 6066
6248 RR=RR-1:RC=0
6250 IF RR=0 THEN RR=1
6252 GOTO 6066
6254 '
6256 ' V = 16 PRIMERA TECLA FUNCIONAL = INTRODUCCION
6258 PRINT BI$
6262 F1=2
6264 RETURN
6266 GOTO 6274
6268 '
6270 ' V =17 SEGUNDA TECLA FUNCIONAL = ANULACION
6272 F1=3:RETURN
6274 ' V=13 NUEVA LINEA
6276 IF TP(RR)=1 GOTO 6290       ' ALINEA LOS NUMERICOS A LA DERECHA
```

```
6278 AS$=LEFT$(BF$(RR),RC):BF$(RR)=" "
6280 N=LD (RR)-RC:B1$=LEFT$(BL$,N):PRINT B1$;:BF$(RR)=AS$+B1$
6282 RR=RR+1:RC=0;X=X0:Y=Y+1:IF RR>NR(P) THEN RR=1: Y=Y0
6284 IF RR>NR(P) THEN RR=1:Y=Y0
6286 IF TP (RR)=0 THEN GOSUB 6368
6288 GOTO 6066
6290 ' ** numéricos **
6291 'PRINT FNPS$;"6291 RC=";RC;"RR=";RR;"X=";X;"Y=";Y:S$=INPUT$(1)
6292 A$=""
6294 N=LD(RR)-RC:IF N=0 GOTO 6282
6296 FOR I=1 TO N:A$=A$+"0":NEXT I
6298 IF RC=0 THEN X=X0: GOSUB 6900: PRINT A$;:BF$(RR)=A$: GOTO 6282
6300 M=RC
6302 A$=A$+LEFT$(BF$(RR), M):X=X0:GOSUB 6900:PRINT A$;:BF$(RR)=A$
6304 GOTO 6282
6306 '
6308 '
6364 STOP
6366 ' **** GESTION CAMPOS SALTADOS ****
6368 ' ----------
6370 '
6372 RR=RR+1:RC=0
6374 IF RR>NR(P) THEN RR=1
6376 RETURN
6900 '        DESPLAZAMIENTO CURSOR
6910 '      FILE CURS-X/Y
6930 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);
6940 RETURN
7050 ' ****
7060 ' ****
```

Las teclas de desplazamiento horizontal son tratadas de forma similar a las anteriores (en este caso varía el contador de caracteres y no el de líneas), mientras que la tecla RT opera según una lógica diferente, puesto que se utiliza para alinear los campos. En la fase de introducción de datos, el cursor se posiciona al principio de cada campo, desplazándose una posición hacia la derecha por cada carácter introducido; el resultado es una escritura que empieza por la izquierda. Todos los campos, independientemente del tipo, quedarán así agrupados a la izquierda. Sin embargo, los campos numéricos deben quedar a la derecha. Esta función la desarrollan las instrucciones 6274 a 6288, que son activadas por la tecla RT (la línea 6276 determina si el campo es numérico o alfabético controlando la matriz TP).

Una última particularidad de la rutina es la función de posicionado del cursor obtenida mediante una adecuada subrutina. El contenido de esta subrutina (una sola línea) podría escribirse directamente en los puntos en que debe ejecutarse la función: volver a escribir la línea de direccionado o insertar en su lugar la llamada a la subrutina es equivalente, tanto en términos de ocupación de memoria como de tiempo. Sin embargo, se ha preferido la subrutina porque, en el caso de cambio de máquina la zona del programa que debe corregirse está localizada.

El segundo tipo de tecla funcional permite salir de la rutina con el flag F1 puesto igual a 2 (tecla 1) y con F1 = 3 (tecla 2). De esta forma puede establecerse en el main si los datos introducidos son válidos (F1 = 2), o si el operador ha decidido anularlo todo (F1 = 3). Para ampliar las funciones desarrolladas (hasta un máximo de 8, ya que las teclas funcionales habilitadas son 8, ver subrutina 9000) basta con insertar los correspondientes códigos en la matriz TF y prever las necesarias instrucciones en las subrutinas.

La fase de memorización en el disco

Bajo la condición F1 = 2 (datos válidos) deben memorizarse los datos en el disco. En este caso también puede escribirse una subrutina parametrizada de uso general. Las funciones que pueden desarrollarse sobre un file son:

— Apertura y asignación del buffer I/O
— Lectura de datos
— Escritura de datos
— Cierre del file

El cierre del file se obtiene insertando una sola instrucción (CLOSE n) y, por tanto, es más cómodo no incluirla en la subrutina, sino utilizarla localmente donde sea necesario. Las otras tres funciones, en cambio, son parametrizadas. En las págs. 710 y 711 se ha representado el diagrama de flujo y, en la pág. 712, el correspondiente listado.

La subrutina 1200 debe llamarse definiendo los siguientes parámetros:

MH = flag que indica la función deseada (apertura, lectura, escritura)
NM$ = nombre del file en el que se quiere operar
NF = número del file
RE = record a leer o escribir
LN = longitud del record

Además es necesaria una matriz (de cadena, DX$) que actúe como buffer I/O, con cada elemento asociado a un file. La asociación parametrizada se obtiene en la línea 1226.

Las otras funciones realizadas por la subrutina son las de lectura y escritura, que presentan alguna complejidad y, por tanto, necesitan un mayor detalle en su exposición.

La función de escritura (instrucciones 1242 a 1272). En el primer record de cada file hay memorizado el número del último record escrito; por tanto, el record 1 no es accesible a los datos. Es decir, el dato número 1 se escribe en el record 2, el dato 2 en el record 3, etc.

Esta desviación puede producir errores si se gestiona desde el exterior; el programador debe acordarse de añadir un 1 para leer o escribir los datos, aparte de que la operación no debe realizarse si se aplica al primer record. Esta lógica puede hacerse automática en el interior de la rutina sumando 1 en cada caso, con la convención de indicar el primer record con el número 0. De esta manera, si se quiere acceder al record 1, se indica RE = 0. Naturalmente, la rutina, antes de transferir de nuevo el control, debe restablecer el valor que tenía RE en el momento de la llamada restando 1 (línea 1271).

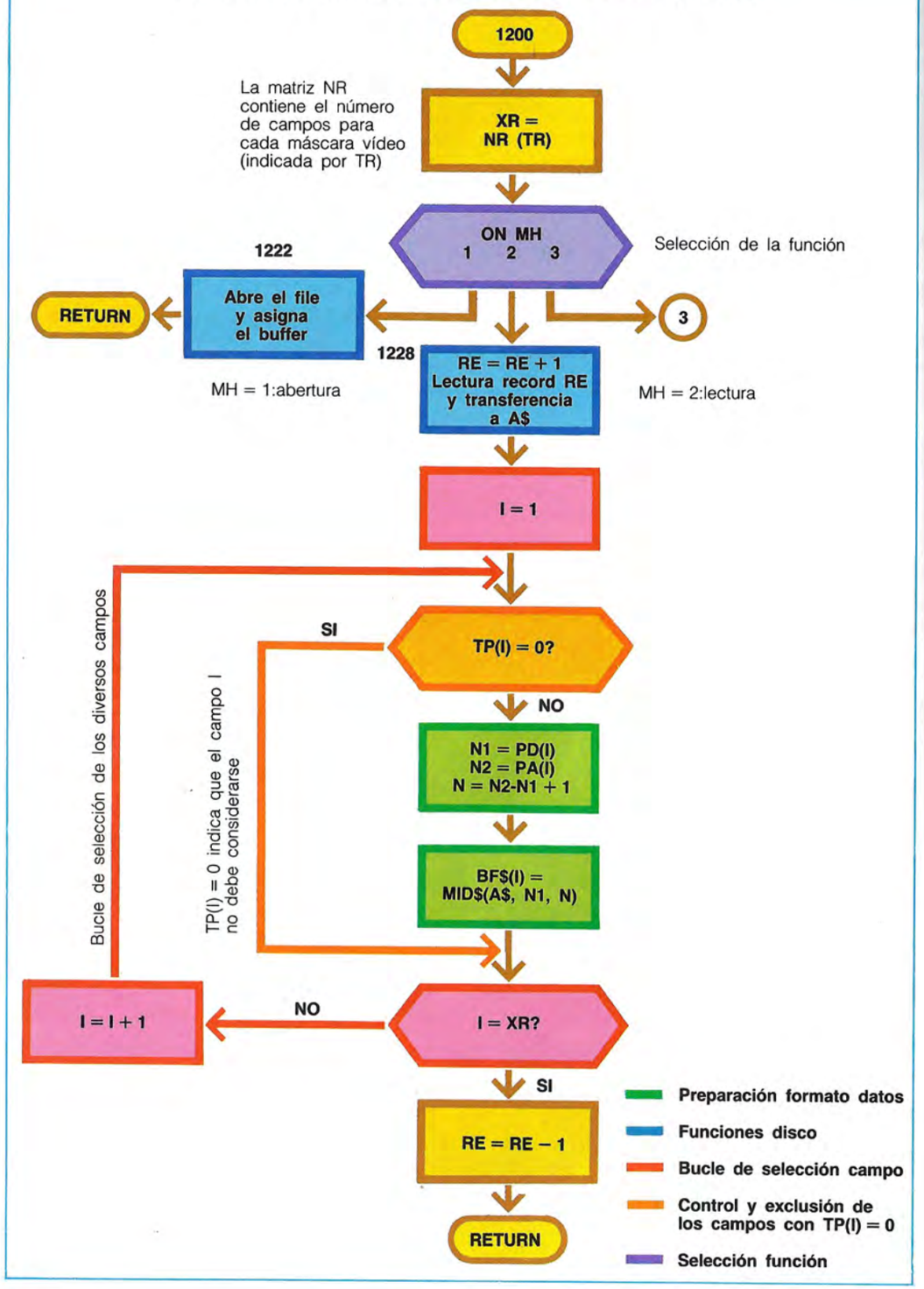
Antes de escribir un record de datos, deben unirse entre sí los buffers BF$ que contienen las introducciones hechas sobre la máscara.

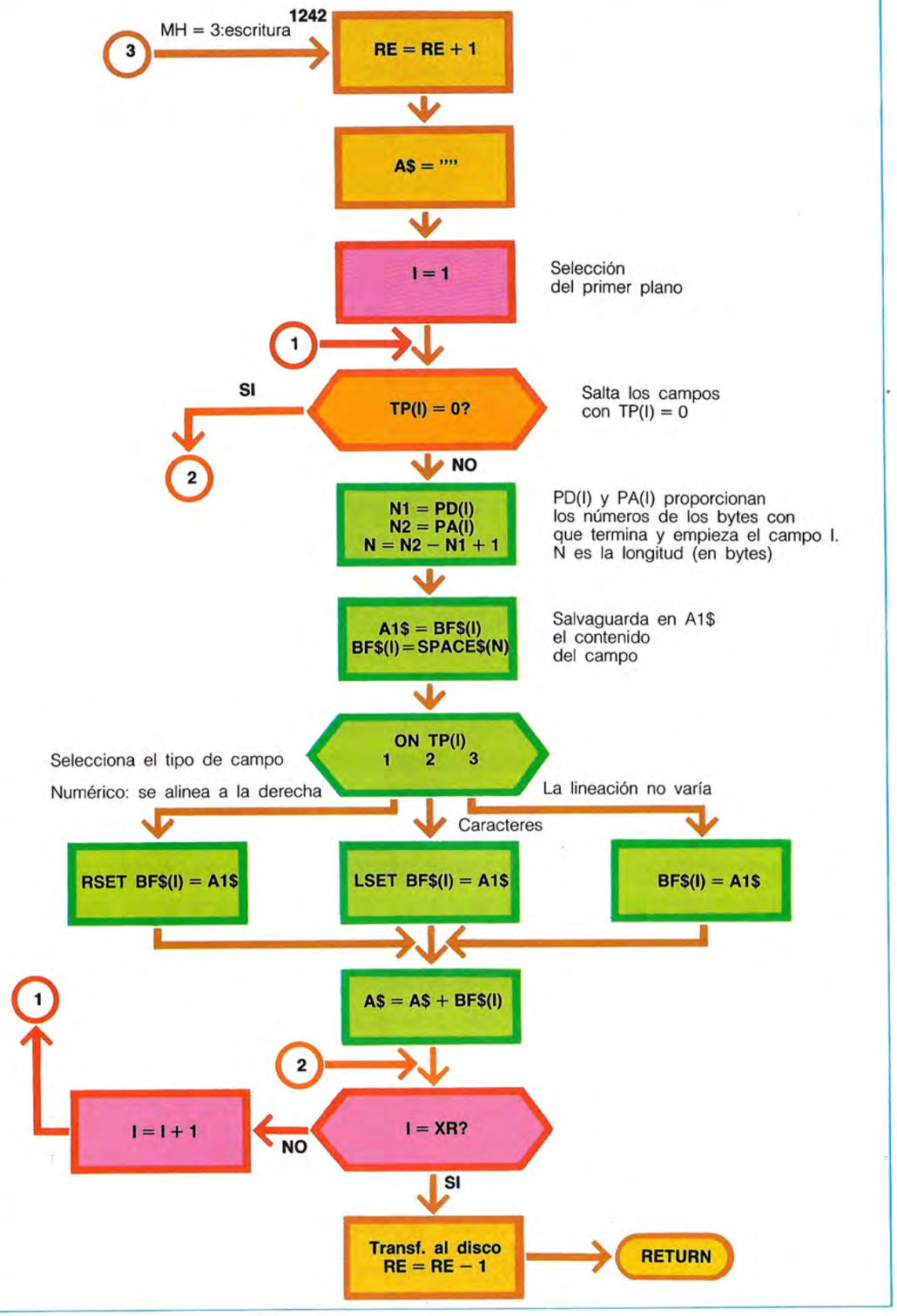
La lógica de formación del record de datos es la siguiente: el contenido de cada buffer de datos se coloca detrás (después del alineado del contenido) del precedente, de manera que formen la cadena A$ como la representada en la pági-

Una fase del trabajo de verificación funcional en un sistema de proceso.

K. Reese/Marka

SUBRUTINA DE GESTION DISCO PARAMETRIZADA





GESTION DISCO PARAMETRIZADA

```
1200 '
1201 ' VERS. ENCICLOPEDIA
1202 ' ** LECTURA/ESCRITURA DISCO **
1203 ' FILE : GDAT
1204 ' Record = RE
1206 '
1208 ' NF = Número   File = Número del Buffer
1212 ' MH =
1214 '       1 Apertura
1216 '       2 Lectura
1218 '       3 Escritura
1219 XR=NR(TR)       ' NUMERO DE CAMPOS (7500)
1222 ON MH GOTO 1224,1228,1242
1224 ' APERTURA
1226 OPEN "r", NF,NM$,LN:FIELD NF,LN AS DX$(NF) : RETURN
1228 ' LECTURA
1229 RE=RE+1    ' EL NUMERO DE RECORD SE INCREMENTA EN 1
1230 GET NF,RE:A$=DX$(NF)
1232 FOR I=1 TO XR:IF TP (I)=0 GOTO 1236
1234 N1=PD (I):N2=PA(I):N=N2-N1+1:BF$(I)=MID$(A$,N1,N)
1236 NEXT I
1237 RE=RE-1    ' RESTABLECE EL NUMERO DE RECORD AL VALOR INICIAL
1238 RETURN
1240 '
1242 ' ESCRITURA
1243 RE=RE+1
1244 A$=""
1246 FOR I=1. TO XR:IF TP(I)=0 GOTO 1264
1248 N1=PD(I):N2=PA(I):N=N2-N1+1
1250 A1$=BF$(I):BF$(I)=""
1252 BF$(I)=SPACE$(N)
1254 ON TP(I) GOTO 1256,1258,1260
1256 RSET BF$(I)=A1$:GOTO 1262
1258 LSET BF$(I)=A1$:GOTO 1262
1260 BF$(I)=A1$
1262 A$=A$+BF$(I)
1263 '
1264 NEXT I
1265 PRINT"A=";A$:S$=INPUT$(1)
1268 LSET DX$(NF)=A$
1270 PUT NF,RE
1271 RE=RE-1
1272 RETURN
```

na siguiente arriba (instrucciones 1246 a 1264). A continuación, la cadena A$ (encadenamiento de todos los datos) se transfiere al buffer asociado al file (línea 1268) y, por tanto, se escribe en el disco (línea 1270).

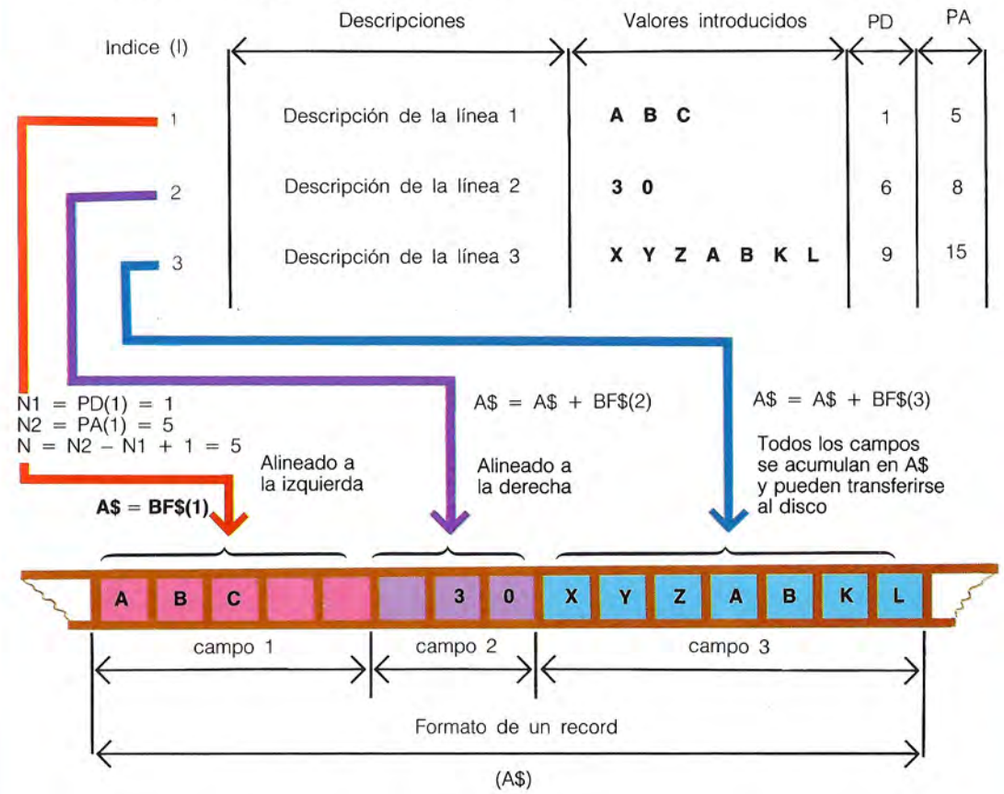
La función de lectura (instrucciones 1228 a 1238). El mecanismo es el opuesto al anterior. En el disco se lee el record deseado (siempre sumando 1), los datos se transfieren del buffer del file a la cadena A$, y de ésta se distribuyen a los buffers de datos sencillos BF$. La lógica se ve en la página siguiente, abajo.

El programa principal

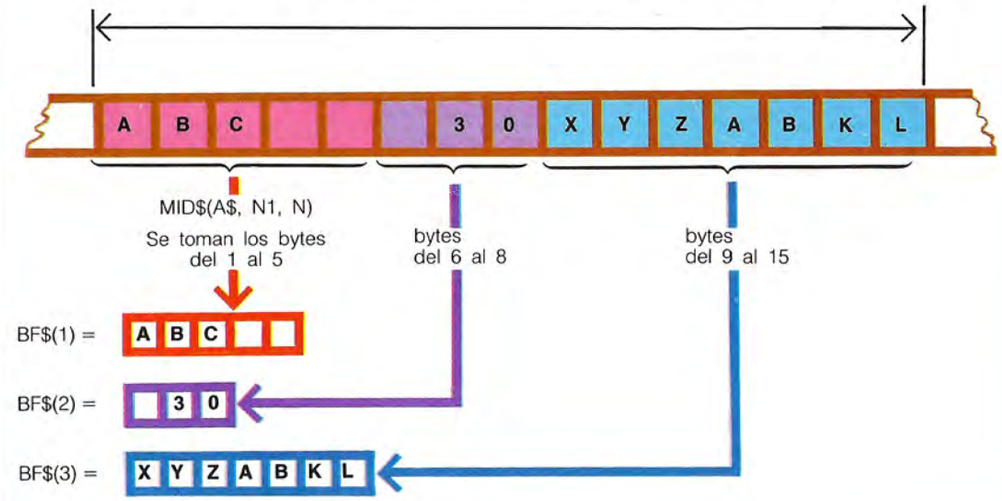
En la pág. 714 se ha representado el diagrama de flujo de un main de prueba. El listado está en las págs. 715 y 716. En este listado hay incluida la subrutina de habilitación de las teclas funcionales (9000) previstas en las máscaras vídeo; la inicialización de los correspondientes códigos se encuentra en las líneas 240 y 250 del programa principal (se han previsto 20 códigos para futuras implantaciones).

ESQUEMA DE LA FORMACION DE UN RECORD

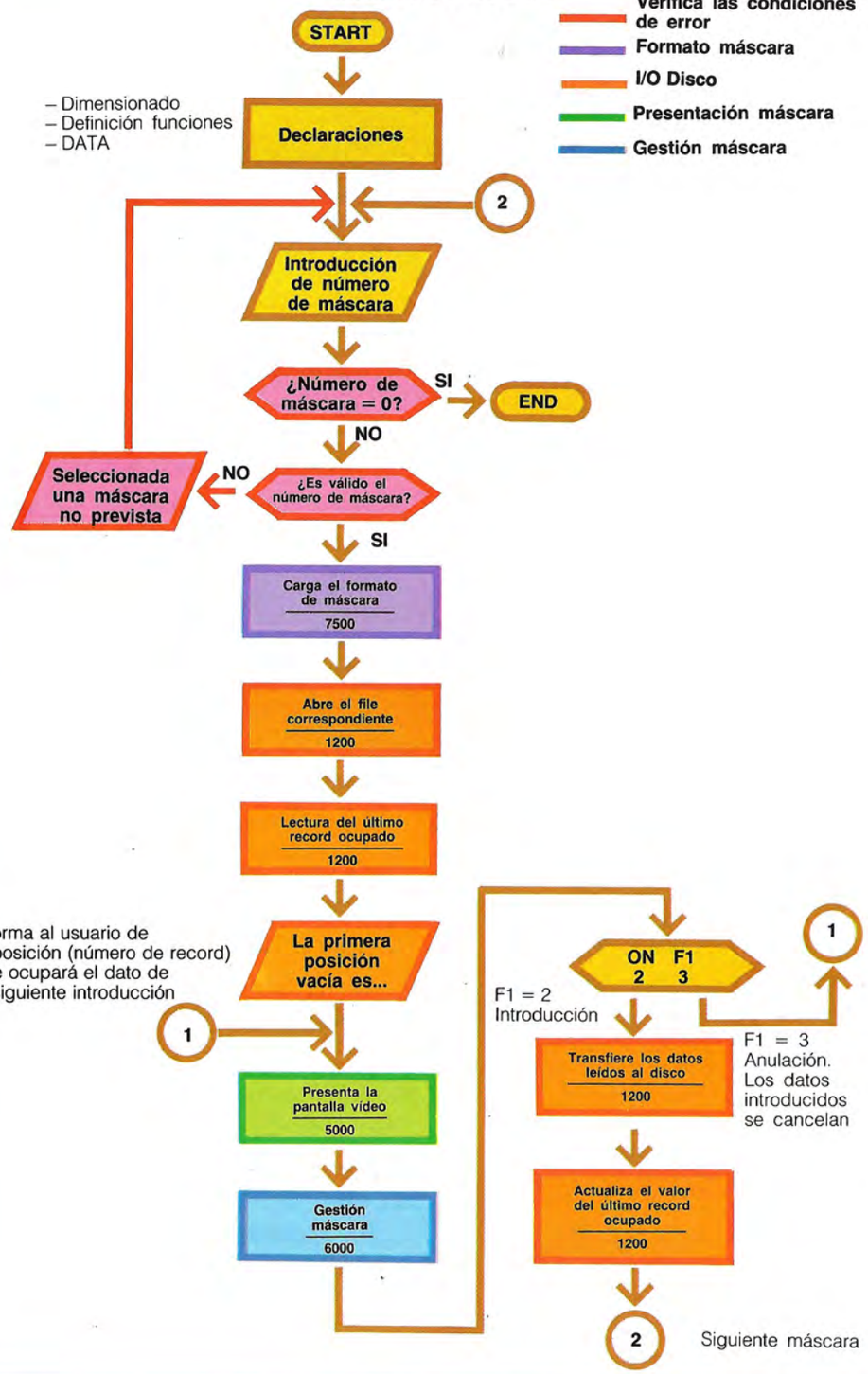
El record se constituye «sumando» los diversos campos con la instrucción A$ = A$ + BF$(I)



ESQUEMA DE LA SEPARACION DE LOS CAMPOS DE LECTURA



MAIN DE PRUEBA



MAIN DE PRUEBA

```
10 OPTION BASE 1
20 '   FILE : MAIN
30 '
40 ' VERS. ENCICLOPEDIA
50 ' ***** DEFINICIONES *****
60 '  TF(*)    = CODIGOS TECLAS FUNCIONALES
70 '  DX$(*)   = BUFFER I/O DISCO
80 '  TP(*)    = TIPO DE CADA CAMPO PRESENTADO POR LAS MASCARAS VIDEO
90 '  PA(*)    = PUNTERO FINAL CAMPO
100 ' PD(*)    = PUNTERO PRINCIPIO CAMPO
110 ' DS$(*)   = DESCRIPCIONES DE LOS DIVERSOS CAMPOS
120 ' B$(*)    = BUFFER DE PUESTA A CERO DATOS EN INTRODUCCION
130 ' BF$(*)   = BUFFER ENTRADA DATOS
140 ' AT$(*)   = BUFFER DE SEPARACION ENTRE DESCRIPCIONES AREAS DATOS EN LAS
150 '            MASCARAS VIDEO
160 ' BR$(*)  = BUFFER DATOS RENOVADOS
170 '
180 '
190 '
200 '
210 DEFINT A-V
220 DEFDBL Z
230 DIM TF(20)                    '  MATRIZ MONODIMENSIONAL
240 FOR I=1 TO 20 : READ TF(1):NEXT 1
250 DATA 10!,8!,12!,11!,16!,17!,13!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!,0!
260 DIM DX$(4)
270 DIM TP(14),PA(14),PD(14),LD(14),NR(5)
280 DIM DS$(14), B$(14),BF$(14),AT$(14),BR$(14),CN$(14)
290 C1$= "  FILE DATOS"
300 '
310 '
320 '
330 '     *  FUNCIONES  *
340 '
350 BL$="                                "
360 BB$="  ........................."
370 BI$=CHR$(27)+"+"+CHR$(7)
380 DEF FNPS$=CHR$(27)+CHR$(61)+CHR$(31+20)+CHR$(31+2)  ' POS. POR ERROR
390 DEF FNA1$(I)=DS$(I)+AT$(I)+B$(I)   ' VER SUB.5000

400 DEF FNA2$(I)=DS$(I)+AT$(I)+BR$(I)  ' VER SUB.5000
405 GOSUB 9000   ' HABILITA LAS TECLAS FUNCIONALES
410 PRINT BI$    ' BORRA EL VIDEO Y EMITE UN BIP
500 ' ** SELECCION MASCARA **
510 INPUT "INTRODUCIR EL NUMERO DE MASCARA DESEADO ",N
520 IF N=0 GOTO 900    ' FIN
530 IF N<1 OR N>2 THEN PRINT "ERROR MASCARA NO PREVISTA"
540 IF N<1 OR N>2 THEN S$=INPUT$(1):GOTO 510
550 ' ** FORMATO MASCARA **
560 TR=N   ' LA 7500 QUIERE EL NUMERO DE MASCARA EN TR
570 GOSUB 7500 ' CARGA LOS DATA
580 ' **APERTURA DEL FILE **
590 MH=1:GOSUB 1200
600 ' LECTURA DEL RECORD 1
610 RE=0:MH=2
620 GOSUB 1200
630 ME=VAL(BF$(1))     ' ULTIMO RECORD OCUPADO
640 RE=ME+1            ' SIGUIENTE POSICION VACIA
650 PRINT"EL SIGUIENTE NO ESTA EN LA POSICION: ";RE
660 PRINT "PULSAR UNA TECLA PARA CONTINUAR":S$=INPUT$(1)
670 '  ** PREPARA Y PRESENTA LA PANTALLA VIDEO **
680 GOSUB 5000
690 ' ** ADQUISICION **
700 GOSUB 6000
```

```
710 ' ** SELECCION EN BASE AL FLAG F1
720 K=F1-1    ' F1 TIENE VALOR 2 o 3 Y DEBE LLEVARSE A 1 o 2
730 ON K GOTO 740,880
740 ' * F1 = 2 * INTRODUCCION *
750 MH=3        ' MODO = ESCRITURA
760 GOSUB 1200  ' ESCRIBE LOS DATOS EN EL RECORD RE
770 ME=RE       ' SALVAGUARDA EL NUMERO DE RECORD
780 ' ** ACTUALIZACION DEL CONTENIDO DEL PRIMER RECORD
790 RE=0    ' EL PRIMER RECORD ESTA INDICADO CON 0
800 ' ** PREPARACION DEL CAMPO BF$(1) **
810 N=FA(1)-FD(1)+1   ' NUMERO DE CARACTERES EN BF$(1)
820 BF$(1)=SPACE$(N)  ' INICIALIZA BF$(1) CON ESPACIOS EN BLANCO
830 A$=STR$(ME)       ' CONVIERTE EN ASCII EL NUMERO DE RECORD
840 RSET BF$(1)=A$    ' ALINEA A LA DERECHA EN EL BF$(1)
850 MH=3:GOSUB 1200   ' ESCRITURA
860 GOTO 640          ' PROXIMA INTRODUCCION
870 ' ** ANULACION DATOS **
880 '
885 PRINT BI$
890 GOTO 640
900 ' ** FIN **
910 PRINT BI$
920 PRINT "*** RUN END ***"
930 END
```

```
9000 '  ** SUBRUTINAS DE HABILITACION TECLAS FUNCIONALES **
9010 '
9020 '     FORMA DEL CURSOR
9030 PRINT CHR$(27)+"c4AA"
9040 '
9050 '
9060 '
9070 A$=CHR$(27)+CHR$(46)     ' PRIMERA PARTE DE LA INSTRUCCION QUE HABILITA
9080 '                          LAS TECLAS FUNCIONALES
9090 B=98                     ' CODIGO CORRESP. A LA PRIMERA TECLA FUNCIONAL
9100 I1=49                    ' 49 y 48 ASCII = 10 HEX porque (49=1; 48=0)
9110 I2=48                    ' 10 HEX vale 16 en DEC
9120 FOR I=1 TO 8             ' CONTADOR PARA HABILITAR 8 TECLAS
9122 ' LA PRIMERA TECLA FUNCIONAL (núm. 98) TIENE ASOCIADO EL VALOR DECIMAL 16
9130 C$=CHR$(I1)+CHR$(I2)     ' CODIGO A ASOCIAR A LA TECLA I (1..8)
9140 D$=CHR$(B)               ' SELECCION TECLA
9150 E$=A$+D$+C$              ' CADENA QUE CONTIENE LOS CODIGOS DE HABILITACION
9160 PRINT E$                 ' HABILITACION
9170 B=B+1
9180 I2=I2+1
9190 NEXT I
9200 RETURN
```


La compilación

El empleo del Basic interpretado es muy útil en las fases de escritura y de prueba de programas, pero la ejecución resulta muy lenta para el uso normal. Los motivos por los que el interpretado emplea más tiempo en la ejecución de los programas son básicamente tres:

- En todas las llamadas del tipo GOTO o GOSUB, para encontrar el número de línea a que se hace referencia, el Intérprete debe leer todo el programa.
- Cada vez que el programa utiliza un nombre de variable, para determinar en qué posición de memoria se encuentra el valor correspondiente al nombre simbólico, el Intérprete debe leer toda la tabla de variables
- En el desarrollo de los bucles, el Intérprete debe descodificar de nuevo en cada iteración todas las instrucciones contenidas en el bucle. Si bien el tiempo necesario para descodificar una sola instrucción es muy corto, multiplicándolo por el número de veces que se ejecuta el bucle se obtiene un tiempo total que puede ser importante

Todos estos defectos, consecuencia del modo específico de trabajar del Intérprete, pueden eliminarse utilizando los programas compilados. Al final de la fase de compilación, el programa de aplicación es traducido a binario y las direcciones consideradas son las absolutas; por tanto, la velocidad de ejecución puede aumentar de 4 a 10 veces. El Compilador realiza el diagnóstico global y puede detectar los errores que puedan haber escapado al Intérprete.
Un caso típico se tiene en las instrucciones de salto condicionado. Si la dirección de llegada (número de línea) a la que se hace referencia en la instrucción de salto no existe, el Intérprete no puede detectar el error hasta que se verifican las condiciones que determinan el salto.
En algunos casos es probable que durante las pruebas del programa nunca se haya verificado esta condición; en consecuencia, el error permanece oculto y puede manifestarse en seguida, durante la aplicación, con la consiguiente pérdida de los datos durante el proceso.
En cambio, durante la compilación, todas las direcciones se comprueban, y un eventual error se detecta inmediatamente.
En el gráfico de la pág. 718 se esquematiza el procedimiento completo de introducción, compilación y ejecución de un programa. A continuación describiremos con detalle cada fase.

La fase de preparación

El Compilador puede trabajar sólo sobre files en formato ASCII. La primera operación a realizar consiste, por tanto, en asegurarse de que el fuente tenga formato "A". Si no es así, puede variarse el formato simplemente cargando el programa en memoria y después transfiriéndolo al disco con el mismo nombre, pero con la opción "A". Por ejemplo, las instrucciones que permiten variar el formato del programa PRUEBA residente en el disco A son las siguientes:

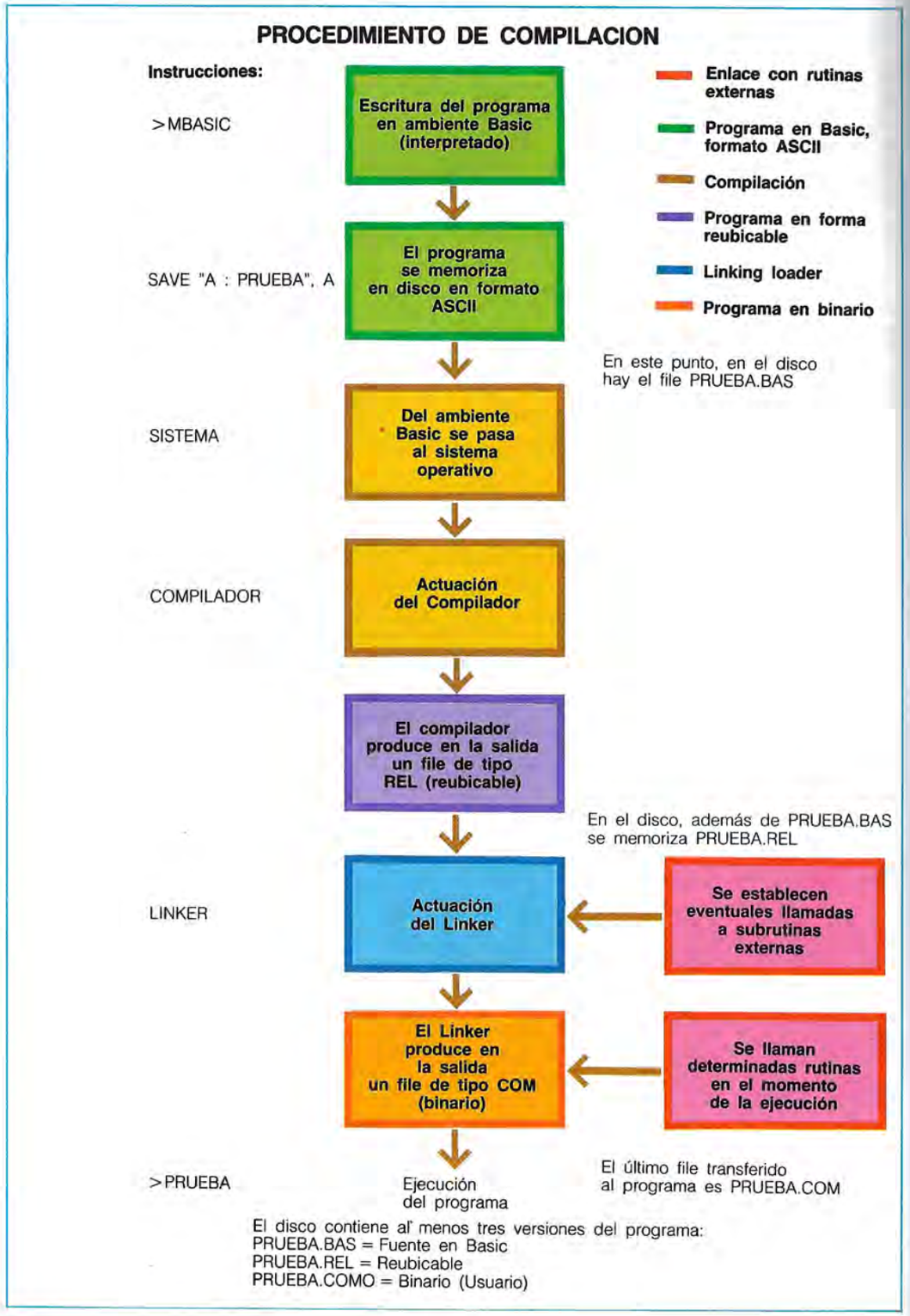
```
LOAD "A:PRUEBA"
SAVE "A:PRUEBA",A
```

En la escritura del fuente (en Basic), a fines de la velocidad de ejecución, también es útil asegurarse de que todos los índices de los bucles se han declarado enteros, implícitamente con la instrucción DEFINT o explícitamente con el símbolo % (DEFINT I tiene el mismo efecto que I%).
El último paso de preparación a realizar es asegurarse de que en el disco "A" existen todos los programas (de sistema) o los files de servicio que se utilizarán durante la compilación y la siguiente operación de link*.
Se ha hecho referencia al disco montado en la unidad A porque el Compilador trabaja utilizando esta unidad. Si bien es posible direccionar algunos pasos a la otra unidad, conviene tenerlo todo en el mismo disco. Luego el programa compilado podrá transferirse a otro lugar.
A continuación se relacionan los programas y los files necesarios para la fase de compilación.

BASCOM.COM. Es el programa de sistema que realiza la traducción de Basic a Assembler; su salida es la versión reubicable del programa fuente.

BRUN.COM. Contiene la mayor parte de las rutinas llamadas por el programa de aplicación. Es un módulo usado al trabajar el programa de aplicación, pero debe estar presente desde el principio para permitir los oportunos links.

* El link entre varios programas o una parte de programa se describe más adelante. Por ahora debe considerarse sencillamente como una serie de operaciones que permiten «reunir» varias partes compiladas por separado.



BASLIB.REL. Contiene otras rutinas que implantan las de BRUN.COM y que pueden utilizarse en los programas de aplicación. El programador accede a las rutinas de sistema sin especificar dónde residen. Todo el trabajo de traducción y de link, que consiste en analizar cada línea del programa fuente, en reconocer la llamada a una rutina de sistema, en su búsqueda en varios módulos del sistema operativo y en establecer los oportunos direccionamientos que permitirán su utilización durante la ejecución del programa, está a cargo del sistema a través del Compilador y del Linker (más apropiadamente Linking Loader o Montador).

OBSLIB.REL. Contiene rutinas que desarrollan funciones similares a las de BRUN y BASLIB. La diferencia consiste en una forma diferente de gestión, debida a un empleo diferente del programa de aplicación.
En algunas aplicaciones, el programa de aplicación, una vez definido, no debe sufrir modificaciones ni debe preverse el uso de otros programas en la misma máquina. En estos casos conviene que el programa de aplicación esté almacenado en memoria ROM (de sólo lectura), o sea que se transforme en lo que se conoce como **firmware** (software permanente).
De este modo puede tenerse la activación automática del programa en el momento de la puesta en marcha de la máquina y, sobre todo, se evitan posibles errores en la fase de carga. Esta técnica se adopta en los casos particulares que tienen en común dos puntos esenciales:

- la máquina y el programa forman un conjunto único, dedicado a una sola finalidad precisa y no modificable (si no es sustituyendo las memorias)
- el sistema debe poderse fabricar en un cierto número de ejemplares, puesto que no conviene preparar una ROM para una sola aplicación.

El hecho de que un programa deba escribirse en una ROM implica una estructuración diferente de aquél, puesto que no es posible escribir valores en el área de memoria reservada para ellos. Las variables deberán colocarse en una memoria RAM, separada de la ROM que contiene el programa, a través de procedimientos más complicados que los usuales. A fines prácticos, esta estructuración obliga a utilizar dos áreas separadas de memoria: el área de programa (que estará en ROM) y el área de datos (en RAM). Estas indicaciones particulares se proporcionan durante las operaciones de link colocando oportunos indicadores (switches), cuyo uso obliga a cargar la librería OBSLIB en lugar de las librerías BRUN y BASLIB.

BCLOAD. Contiene los links, o sea las referencias a las posiciones de memoria en que se encuentran las diversas partes del programa y las rutinas de sistema utilizadas por aquéllas.

L80.COM. Es el programa que, enviado a ejecución después del BASCOM, procede a crear los links necesarios y a enlazar las diversas partes para generar el programa de salida a nivel ejecutable. Este módulo toma el nombre de Montador, apropiado a las funciones que realiza.
Además de estos módulos de utilización general existen otros dedicados a funciones particulares, algunos de los cuales se describirán más adelante.

La fase de compilación

El Compilador (BASCOM) necesita en la entrada un programa fuente y en la salida produce la versión reubicable, o sea traducida al Assembler, con todas las referencias **correspondientes**. Al escribir un programa, para asignar el valor 2.5 a una variable de nombre V utilizaremos la instrucción V = 2.5 en la misma forma clara y sintética que emplearíamos desarrollando el cálculo manualmente. Si no se descodifica, esta instrucción no tiene sentido para la máquina; es necesario especificar en qué dirección de memoria se encuentra la variable V y traducir la instrucción especificando dicho número.
Análogamente, la instrucción R = SQR(V), que toma el contenido de V, calcula su raíz cuadrada y deposita el resultado en R, necesita a nivel de lenguaje máquina la especificación de las direcciones. Durante la compilación, algunas partes del programa se traducen separadamente de las otras, y a cada una se le da una numeración en términos de posición de memoria.
De esta forma, las diversas partes del programa son **reubicables**, puesto que todas las direcciones utilizadas se refieren al primer valor: variando este valor varían los otros en consecuencia. El cálculo, que del valor relativo permite hallar el valor absoluto de colocación en memoria, lo realiza el Montador.

La salida de la fase de compilación (forma reubicable) se memoriza en un file de tipo REL. El formato del comando para la compilación es

BASCOM Objeto,Lista = Fuente

El significado de los parámetros que aparecen en la instrucción se explican a continuación.

Objeto. Es el nombre que se quiere dar al file de salida (tipo REL) en el que se memorizará la versión reubicable del programa. Este tipo de file se indica con el nombre genérico «Objeto» (Object).

Lista. Es el nombre del file «lista» que contendrá el listado del programa y su traducción.
En las págs. 721 y 722 se ha representado la impresión del contenido de este file en un ejemplo de compilación. El programa fuente (arriba de dicho tabulado) se ha listado en ambiente Basic. La parte inferior del tabulado contiene la explosión de cada instrucción Basic en las correspondientes instrucciones en Assembler, y constituye el contenido del file Lista.

Como puede verse, una instrucción Basic da lugar a una serie de instrucciones Assembler que también contienen llamadas a rutinas de sistema, como por ejemplo CALL $CINA (la llamada a una rutina Assembler utiliza el código CALL en lugar del GOSUB). En el listado, cada instrucción del fuente va precedida de dos números hexadecimales. El primero representa la dirección de la línea correspondiente a la dirección del principio del programa (en el supuesto que sea cero) y, el segundo, el área de datos utilizada; por ejemplo, la instrucción de la línea Basic 40 ocupa la dirección correspondiente 1F (decimal 31). Al cargar el programa en memoria, su dirección inicial ya no será cero, sino igual a un cierto valor que se sumará a todos los demás; así, la instrucción 40, que tenía una dirección relativa 1F, se posicionará a una dirección absoluta igual a 1F más «baja» de la que se ha colocado el principio del programa.

Fuente. Es el nombre del file en que reside el programa a compilar.
Durante la compilación, el file objeto se crea en cualquier caso, incluso si no se declara expre-

Equipo del centro de adiestramiento aeronaval de Tarento. Las consolas reproducen la situación de las centrales operativas de combate.

B.A. Licitta/I. Dagherrotipo

EJEMPLO DE COMPILACION

```
BASCOM 5.30 - Copyright 1979,80,81 (C) by MICROSOFT - 25298 Bytes Free
 0014 0007       10 ' ** PRUEBA DE COMPILACION
 0014 0007       20 OPTION BASE 1
 0014 0007       30 DIM A(5)
 0014 0007       40 FOR I=1 TO 5
        ** 0014'I00000: CALL    $530
        ** 0017'L00010: L00020: L00030:L00040:
        ** 0017'        CALI    $LFMA
        ** 001A'        DW      <const>
        ** 001C'        JMP     I00001
        ** 001F'I00002:
 001F 001B       50 B=I*A(I)
        ** 001F'L00050: LXI     H,I!
        ** 0022'        CALL    $CINA
        ** 0025'        DAD     H
        ** 0026'        DAD     H
        ** 0027'        LXI     D,A!+FFFC
        ** 002A'        DAD     D
        ** 002B'        CALL    $LFHA
        ** 002E'        CALL    $FMUC
        ** 0031'        DW      I!
        ** 0033'        CALL    $FASO
        ** 0036'        DW      B!
 0038 0023       60 NEXT I
        ** 0038'L00060: CALL     $FADA
        ** 003B'        DW       I'
        ** 003D'        DW       <const>
        ** 003F'I00001:
        ** 003F'        CALL     $FASO
        ** 0042'        DW       I!
        ** 0044'        CALL     $LEJA
        ** 0047'        DW       I!
        ** 0049'        DW       <const>
        ** 004B'        DW       I00002
 004D 0023       70 FOR I=1 TO 5
        ** 004D'L00070: CALL     $LFMA
        ** 0050'        DW       <const>
        ** 0052'        JMP      I00003
        ** 0055'I00004:
 0055 0023       80 PRINT "I = ";I,"A(I) = ";A(I)
        ** 0055'L00080: CALL     $PROA
        ** 0058'        LXI      H,<const>
        ** 005B'        CALL     $PV1D
        ** 005E'        LXI      H,I!
        ** 0061'        CALL     $PVOA
        ** 0064'        LXI      H.<const>
        ** 0067'        CALL     $PV1D
        ** 006A'        LXI      H,I!
        ** 006D'        CALL     $CINA
        ** 0070'        DAD      H
        ** 0071'        DAD      H
        ** 0072'        LXI      D,A!+FFFC
        ** 0075'        DAD      D
        ** 0076'        CALL     $LFHA
        ** 0079'        LXI      H,SAC%
        ** 007C'        CALL     $PVZA
 007F 0023       90 NEXT I
        ** 007F'L00090: CALL     $FADA
        ** 0082'        DW       I!
        ** 0084'        DW       <const>
        ** 0086'I00003:
        ** 0086'        CALL      $FASO
        ** 0089'        DW        I!
```

```
        ** 008B'          CALL      $LEJA
        ** 008E'          DW        I'
        ** 0090'          DW        <const>
        ** 0092'          DW        I00004
0094 0023       100 END
        ** 0094'L00100    CALL      $END
0097 0023
        ** 0097'          CALL      $END
00B6 002D

00000 Fatal Error(s)
24882 Bytes Free
```

samente en la llamada del Compilador. En este caso asume el mismo nombre que el fuente. Así, el comando

BASCOM = PRUEBA

lanza el Compilador y, al final, genera el file PRUEBA.REL, nombre con el que deberá llamarse en la siguiente fase de encadenamiento (linking). Sin embargo, el file que contiene el listado no se crea si no se pide expresamente; no obstante, puede obtenerse directamente en pantalla o en impresora sin recurrir al paso intermedio por el file. En este caso, el comando debe tener la siguiente forma:

BASCOM PRUEB,LST: = PRUEBA

La máquina crea entonces el file PRUEB como reubicable y produce la impresión de la lista. El código LST: se interpreta como la petición de impresión (en CP/M, la unidad LST es la impresora); para direccionar la lista a la pantalla, el código es TTY:. Los files de llegada (fuente) y los de destinación (reubicable, lista) pueden residir en otra unidad de disco, pero debe especificarse en el comando. Así, el comando

BASCOM B:PRUEB,LST: = PRUEBA

realiza las mismas funciones efectuadas en la forma anterior, aunque transfiriendo la salida (reubicable de nombre PRUEB) hacia la unidad de disco B. El Compilador también puede trabajar sólo a efectos de diagnóstico, sin producir ningún file de salida. La instrucción a utilizar es

BASCOM, = Fuente

En la salida se obtendrán, en pantalla, la lista de los eventuales errores y la ocupación de memoria del programa. Finalmente, el Compilador puede emplearse en la modalidad «paso a paso», o sea aplicando una instrucción cada vez. El comando

BASCOM,TTY: = TTY:

no ocupa ningún file (ni siquiera el fuente) y acepta las instrucciones del vídeo.
Cada instrucción se traduce en el momento de su introducción y se presenta en pantalla junto con los eventuales diagnósticos. Esta opción se emplea normalmente para la verificación formal (de sintaxis) de las instrucciones y puede ser útil en la fase de aprendizaje del Basic, puesto que puede servir de guía al usuario señalando los eventuales errores instrucción por instrucción.

Funciones particulares del Compilador

El comando que activa la compilación puede completarse con algunos indicadores (switches) que activan determinadas funciones. Un mismo Compilador puede trabajar en diversos tipos de Basic y en diversos modos; los indicadores especifican el tipo de Basic y el modo de tratar algunas instrucciones particulares.
Los indicadores se dividen en tres categorías:

- Especificación de las convenciones
- Funciones de trampa de los errores
- Códigos especiales

A continuación, los tres tipos se describirán sólo desde el punto de vista cualitativo, puesto que el modo de funcionamiento exacto y los códigos a adoptar dependen del Compilador particular que se emplee.

Especificación de las convenciones. En algunas versiones del Basic pueden tenerse diferen-

EJEMPLO DE COMPILACION (1)

En esta página y en la siguiente se ha ilustrado un ejemplo de compilación de un sencillo programa que contiene tres errores.


```
LIST
10 ' **PRUEBA COMPILACION**
20 DIM A(5),AS(6)
30 PRINT "ENTRADA VALORES MATRIZ (A)"
40 FOR I=1 TO 5
50 PRINT "I= ";
60 INPUT "VALOR ";A(I)
70 NEXT I
80 FOR I=1 TO 5:PRINT A(I):NEXT I
90 '
100 PRINT "ENTRADA VALORES MATRIZ (AS)"
110 FOR I=1 TO 6
120 PRINT "I= ";
130 INPUT "VALOR.";AS(I)
140 NEXT I
150 FOR I=1 TO 6:PRINT AS(I):NEXT I
160 '
170 IF A(3)=1743 GOTO 2000
180 GOTO 30
190 END
Ok
RUN
ENTRADA VALORES MATRIZ (A)
I= VALOR?
```

En las líneas 20, 130 y 150 se ha utilizado el nombre AS para una matriz, que es una palabra reservada al sistema. Efectivamente, la palabra se usa para la asignación de los campos de un buffer I/O para la gestión del disco (p.e., FIELD 1.20 AS BX$...) y, por tanto, no puede utilizarse para otras finalidades. En algunas versiones del Basic interpretado, el sistema puede determinar si una palabra reservada se emplea indebidamente.

El segundo error se debe a la falta de la línea OPTION BASE 1. Por tanto, la numeración de los elementos de las matrices A y AS parte de 0, y como las propias matrices están dimensionadas respectivamente con 5 y 6, no será posible asignar ningún valor a los elementos A(5) y AS(6).
En algunos casos, el Basic corrige automáticamente este tipo de errores sin evidenciarlos.
El último error está contenido en la línea 170, que hace referencia a una línea 2000 inexistente. El intérprete no puede revelar el error hasta el momento de la ejecución de la línea 170. Tal como está estructurado el programa, sólo se tendría error si el operador introdujese A(3) = 1743.

Como puede verse en las últimas tres líneas de la pantalla, el programa, a pesar de contener los errores indicados, se ha lanzado y a empezado a correr de manera aparentemente normal.

EJEMPLO DE COMPILACION (2)

En esta página se ha representado el resultado de la compilación del programa considerado en la página anterior.


```
SYSTEM

A>BASCOM =FOT027
 0014 0007        20 DIM A(5),AS(6)
                        ↑ SN
 00B7 0023        130 INPUT "VALOR ";AS(I)
                            ↑ SN
 00D4 0023        150 FOR I=1 TO 6:PRINT AS(I):NEXT I
                                  ↑ SN

00003 Fatal Error(s)
24815 Bytes Eree

A>
```

La primera línea contiene el comando, introducido por el operador, que pide el paso bajo sistema operativo; la segunda línea contiene la llamada del Compilador.
La verificación de la sintaxis de las instrucciones empieza desde el principio (línea 10) y procede hacia abajo.

El Compilador ha detectado un uso incorrecto de la palabra reservada, AS en la línea 20, y envía a pantalla esta última.

En la siguiente línea, el Compilador indica con una flecha el punto en que se ha detectado el error y visualiza unas siglas que indican el tipo de error detectado: SN es la contracción de «syntax error» (error de sintaxis).

El mismo tipo de error se ha detectado después en las líneas 130 y 150. El error de la línea 170 no se ha detectado porque estaba «enmascarado» por los anteriores: aparecerá en la primera compilación que seguirá a la fase de corrección de los errores detectados.

Al final de la compilación, en la pantalla aparece el número total de errores hallados.

cias formales en las instrucciones; por ejemplo, existen versiones que para el nombre de las variables sólo utilizan dos caracteres. Los indicadores de especificación de las convenciones señalan la particular versión del Basic y, por tanto, la notación empleada. Se pueden tener a propósito sustanciales diferencias de función en la convención sobre las palabras reservadas.
Los nombres de las instrucciones y de las funciones Basic están reservados al sistema y no pueden emplearse como nombres de variables.
En algunas formas de Basic, las palabras reservadas deben estar separadas de las adyacentes con un espacio en blanco, mientras que en otras versiones esto no es necesario. Si no se especifica al Compilador qué notación se utiliza, pueden tenerse errores durante el desarrollo del programa. Por ejemplo, el inicio del bucle FOR I = A TO C STEP 2, eliminando los espacios, se convierte en FORI = ATOCSTEP2; si el Compilador utiliza una opción equivocada, la instrucción puede asumir cualquier otro significado que el de inicio de bucle y, por ejemplo, puede asignar a la variable de nombre FORI el valor contenido en la variable ATOCSTEP2.
Con respecto a este tipo de errores, existe una curiosidad muy interesante.
En el lenguaje Fortran (similar al Basic), los bucles se indican con instrucciones del tipo

```
DO 100 I = 3,10
```

El código DO sustituye al código FOR, e I = 3,10 equivale a I = 3 TO 10. El número 100 se refiere al número de línea en que se cierra el bucle. Comparando las dos formas de bucle (Fortran y Basic) se tiene:

Fortran	Basic
10 DO 100 I = 3,10	10 FOR I = 3 TO 10
100 CONTINUE	100 NEXT 1

El Compilador Fortran elimina todos los espacios contenidos en la instrucción; por tanto, la línea 10 se convierte en DO100I = 3,10 y su interpretación exacta sólo está ligada a la presencia de la coma entre el número 3 y el número 10.
En un programa que debía controlar el lanzamiento de una sonda espacial, la coma fue sustituida por error por un punto, y la instrucción se convirtió en DO100I = 3.10.

En la ejecución, en lugar de realizar el bucle, se asignó el valor 3.10 a la variable DO100I, con efectos desastrosos sobre el éxito del lanzamiento*. En Basic, un error de este tipo es menos probable, ya sea por la forma del bucle, ya sea, sobre todo, por la presencia de la instrucción NEXT que cierra el propio bucle. Si el principio (FOR...) se interpreta de forma incorrecta , la compilación dará un error en la línea NEXT, puesto que faltará el correspondiente FOR. Para el Fortran, en cambio, no existe cierre del bucle; por tanto, si la interpretación de la línea de inicio no es exacta, el bucle desaparece y no puede tenerse ningún diagnóstico.
Otras opciones particulares del Compilador Basic corresponden a los bucles, a los redondeos y a los números de línea.
En el desarrollo de un bucle parametrizado puede producirse una situación particular en la que coinciden los dos límites; por ejemplo, el siguiente bucle empieza y termina con el valor 3:

```
10 A = 3:B = 3
20 FOR I = A TO B
```

La incertidumbre de que este bucle se realice o no, puede superarse con una opción que fuerza la ejecución de cada bucle por lo menos una vez. La misma opción también sirve normalmente para convertir en enteros los números en doble precisión realizando un truncado y no un redondeo.
Mucho más útil es la opción que permite ignorar el número de línea. Utilizando esta particularidad, las líneas pueden numerarse en cualquier orden, o bien no numerarse, a excepción de aquellas a que se hace referencia explícita en las instrucciones GOTO o GOSUB. Así, un programa Basic se hace muy similar al correspondiente en Fortran (en el que los números de líneas no son obligatorios), es más rápido y más seguro.

Funciones de trampa de los errores. El uso de la instrucción ON ERROR GOTO implica además un complejo sistema de links del programa y, por tanto, una forma de compilación diferente. La correspondiente opción debe indicarse con adecuados indicadores del Compilador, que producen la memorización de una tabla de di-

* De «Software Reliability», Wyley & Sons, Nueva York, 1976.

reccionamiento que contiene el número de línea de fuente Basic. De esta manera, cuando se verifica un error durante la ejecución de un programa, el sistema puede presentar el número de línea en el diagnóstico.
En los programas compilados, el número de línea desaparece y cada instrucción se identifica por su colocación en la memoria (dirección). Al producirse un error, si el sistema no ha memorizado la tabla de conversión de direcciones-números de línea, en pantalla se presenta la dirección en la que se ha producido el error. La dirección de memoria, en notación hexadecimal, no puede ser útil para encontrar la línea del fuente que ha generado el error si no es a través de procedimientos muy complicados.
En la fase de depuración de los programas **(debugging)** conviene emplear estos indicadores, aunque el programa así compilado necesite un mayor espacio; terminada la fase de depuración, el programa puede compilarse de nuevo omitiendo los indicadores de los errores.
La tabla que contiene las direcciones de cada número de línea se dice que tiene un punto de entrada (entry point) para cada número de línea. En general, el punto de entrada es la puerta a través de la cual se accede a una parte del programa o a una tabla. Normalmente, las tablas tienen un solo punto de entrada y los otros puntos se encuentran en base a la posición relativa con respecto al de entrada.
Para las tablas de conversión de los números de línea esto no es posible, y debe preverse un punto de entrada para cada instrucción del programa. Una gestión tan amplia de los puntos de entrada hace notablemente pesado el programa y aumenta la amplitud del file objeto (binario); de ahí se deriva la necesidad de no utilizarla más que en las fases de depuración.

Códigos especiales. Las principales funciones de estos indicadores contemplan la elección del tipo de Assembler, la habilitación de las instrucciones TRON y TROFF y la elección de la forma particular de compilación para la transferencia del programa en ROM. Las instrucciones TRON y TROFF, que si están en el fuente Basic interpretado siempre vienen a continuación, en el caso de un programa compilado se ignoran, a menos que se hayan declarado expresamente activas. Como ya se ha dicho, el Compilador produce una lista en la que cada instrucción Basic está indicada junto a su traducción en Assembler (ver pág. 721). Durante la compilación, un determinado indicador permite especificar en qué forma de Assembler se desea el listado. El Assembler, aparte de las características principales, es un lenguaje muy ligado al microprocesador y formalmente diferente entre tipo y tipo (una misma instrucción traducida puede asumir formas diferentes según el microprocesador).

La fase de encadenamiento

La última fase necesaria para obtener un programa compilado ejecutable es la operación de unión entre las diversas partes, como por ejemplo entre el file reubicable generado por el Compilador y las librerías de sistema. Esta función la realiza un programa llamado Montador. Al enviarlo a ejecución debe proporcionarse el nombre del file (reubicable) generado como salida del Compilador; al terminar la ejecución, el Montador genera el programa ejecutable y presenta en pantalla una serie de informaciones sobre la ocupación de memoria. Las principales informaciones presentadas son las siguientes (la numeración se refiere al gráfico de la pág. 732):

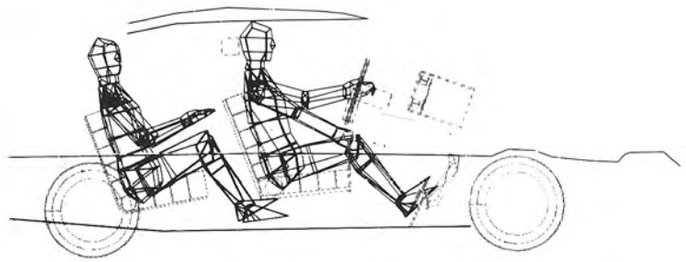
— Inicio del programa (program-start); dirección inicial expresada en notación hexadecimal (**1**)
— Fin del programa (program-end); dirección final hexadecimal (**2**)
— Extensión; número de bytes ocupados por el programa (**3**)
— Memoria disponible (cantidad de memoria no utilizada)
— Dirección de partida (start-address), expresada en hexadecimal. Téngase en cuenta que la dirección de partida puede no coincidir con la dirección inicial del mismo (**5**)
— Número de páginas; número decimal que expresa la cantidad de páginas de memoria de 256 bytes utilizadas por el programa (**4**)

En el gráfico mencionado se ha incluido el mapa de distribución de la memoria. Como puede verse, el inicio del programa comprende el área COMMON y el área reservada a los valores asignados con la instrucción DATA, mientras que la dirección de partida coincide con la primera instrucción ejecutable. El área COMMON es una zona de memoria accesible a más programas, utilizada para el paso de uno a otro de datos y parámetros.
Pueden llamarse varios programas entre sí, pe-

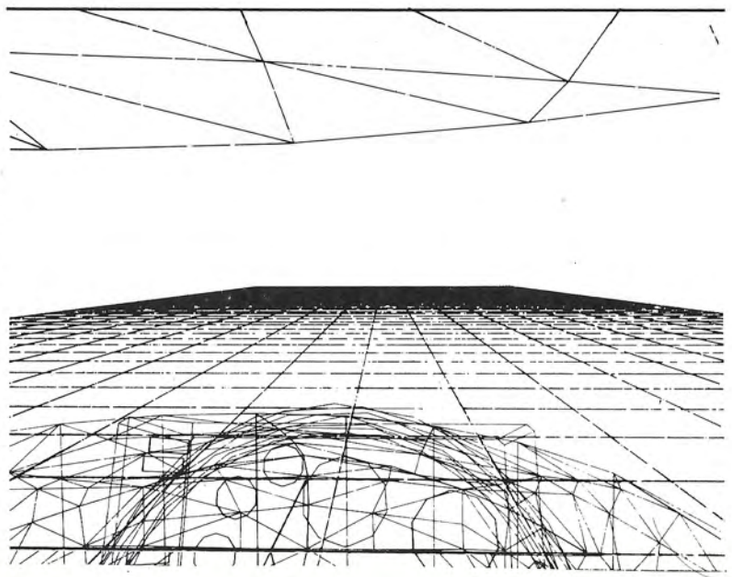
Sammie y el automóvil

En el diseño de un automóvil, el problema más complicado de resolver es probablemente el del justo compromiso que debe encontrarse entre las numerosas características técnico/económicas que no siempre son compatibles. Un automóvil debe tener prestaciones muy brillantes, respetar normas impuestas cada vez más por el mundo exterior y responder en su imagen a un mercado en constante evolución.
Desde el día en que la industria del automóvil decidió ir a por el «modelo del año» para sostener su ritmo de desarrollo, no pasa ningún otoño sin que en las carreteras aparezcan nuevas carrocerías y nuevos estilos de diseño, o que se anuncien mejoras funcionales o de prestaciones. En otras palabras, en el sector del automóvil, la complejidad del proyecto no es tanto un hecho tecnológico cuanto un problema del número de componentes a utilizar y de la multiplicidad de las funciones que deben realizarse. A diferencia de lo que sucede por ejemplo en un avión militar, que normalmente tiene un papel y una función bien definidos, un cuatro ruedas de uso familiar debe satisfacer objetivos mucho más sofisticados y subjetivos.
Efectivamente, como mínimo, debe tener una buena capacidad de carga, ser espacioso para los pasajeros, de fácil conducción, agradable en el plano estético, dócil con los conductores menos expertos, fácil y económico de producir, mantener y reparar.
Pero como producto comercial, también debe tener un precio suficiente para financiar el lanzamiento de nuevos modelos. En resumen, desde un cierto punto de vista, un automóvil puede compararse a un paquete de software, lo cual es bastante lógico si se piensa que ambos pertenecen a la categoría de los bienes amortizables. Además, entre el diseño de un nuevo modelo y su comercialización, normalmente pasan varios años y, en muchos casos, las estrategias de marketing y los proyectistas han tenido que intentar adivinar las características técnicas de un producto que no habría tenido la aceptación del mercado antes de unos diez años. Desde este momento se pone en marcha toda una serie de actividades complejas e interconexas que se resumen en la realización de bocetos, modelos, prototipos, dibujos de ejecución, prensas, utensilios e instalaciones de producción. Y sólo al final de este largo proceso puede empezar la producción en serie, en la que también deberá transcurrir un cierto tiempo antes de alcanzar los ritmos previos como objetivos. Todas estas fases de preparación, por tanto, contribuyen a aumentar el intervalo de tiempo que transcurre entre el diseño y la comercialización de un nuevo modelo.
Sin embargo, la introducción de los sistemas CAE (Computer Aided Engineering) puede abreviar considerablemente los tiempos de proyecto y facilitar el empleo de técnicas de análisis estructural, dinámico y vibracional que, normalmente, permiten mejorar el índice de agrado expresado por el usuario final. En particular, con los sistemas CAE, las soluciones erróneas pueden descartarse desde el principio sin necesidad de esperar a que queden listos los modelos o prototipos.
Sin embargo, en el pasado, los problemas que se planteaban con las técnicas manuales en las fases más avanzadas del proyecto corrían el riesgo de no poder eliminarse por falta de tiempo o de dinero, a causa de las inversiones necesarias en las estructuras experimentales.
Los mejores paquetes de software disponibles actualmente en el mercado permiten simular la disponibilidad de modelos tridimensionales. En particular, el SAMMIE ofrece la posibilidad de definir rápidamente formas geométricas utilizando un lenguaje formado por comandos de alto nivel. Los elementos de esta geometría pueden combinarse después entre sí para representar los objetos más complicados o ambientes completos de trabajo, como por ejemplo una cocina, una parte de una cadena de producción, el fuselaje de un avión, un automóvil, o cualquier otro entorno destinado a contener un hombre. Sin embargo, después de haber definido la estructura geométrica, hay que asegurarse de que pueda ser adecuada para todos los seres humanos destinados a integrarse en ella. Este proceso puede describirse mejor recurriendo a un ejemplo concreto, representado por el proyecto de un nuevo modelo de automóvil.
En este caso, la parte de concepto del proyecto partía del estilista, de un boceto basado en las medidas más significativas (longitud total, anchura y altura) y en las formas propuestas para cada superficie. Estos pocos datos, obtenibles del estilista, bastaron para la traducción del concepto básico en un modelo tridimensional. Inicialmente, éste se limitó a representar la carrocería con una trama de hilo metálico. Esta

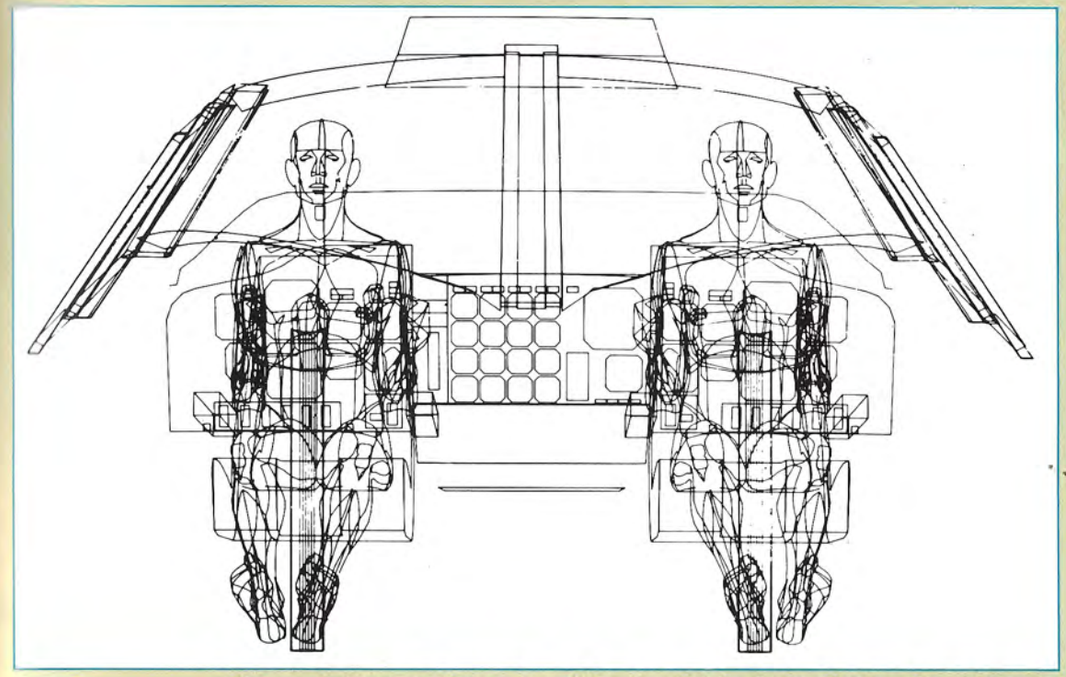
operación resultó tan rápida y completa que no requirió más del 30% del tiempo total del proyecto. Entonces, el proyectista ya pudo hacerse una idea bastante precisa de lo que había de ser el nuevo automóvil.
El modelo así obtenido se hizo girar después sobre sí mismo para poder examinarlo en las distintas perspectivas: un lujo que con las técnicas tradicionales habría necesitado muchas horas de fatigosas proyecciones con el tecnígrafo.
Una vez definido el exterior de la carrocería, se empezó con el interior. En este punto, el proyectista sagaz debe intentar aprovechar, hasta donde sea posible, componentes ya utilizados en modelos anteriores. Por ejemplo, los asientos delanteros podrían tomarse de otro proyecto en fase de producción y adaptarse a las dimensiones de la nueva carrocería con una operación de edición que el SAMMIE consigue hacer muy sencilla. Este enfoque permite obtener una notable reducción de los tiempos y de los costos de proyecto pero, evidentemente, no puede extenderse a todo el habitáculo. La distribución de la instrumentación de a bordo plantea problemas de tipo particular, porque debe mantenerse bajo el control del conductor con una rápida ojeada y sin modificar la distancia focal. En otras palabras, debe colocarse de manera que se encuentre a la mayor altura posible del suelo y a la mayor distancia posible del conductor.
Evidentemente, estas exigencias están en contraposición con los problemas planteados por la visibilidad y la geometría exterior del vehículo.
En realidad, todas estas operaciones también podrían realizarse con instrumentos de tipo tradicional pero, de ahora en adelante, SAMMIE tiene bastante que decir.
Efectivamente, el panel con los instrumentos se



Actualmente, los programas CAE permiten proyectar racionalmente, y en poco tiempo, incluso una estructura tan complicada como la de un automóvil.



Estas imágenes, producidas por el programa SAMMIE empleando un trazador gráfico, ilustran dos momentos del proyecto del interior de un turismo. Arriba, la verificación de la correcta posición de los asientos, del volante y del salpicadero; a la derecha, el proyectista controla la visual del conductor. En primer plano puede verse el salpicadero.

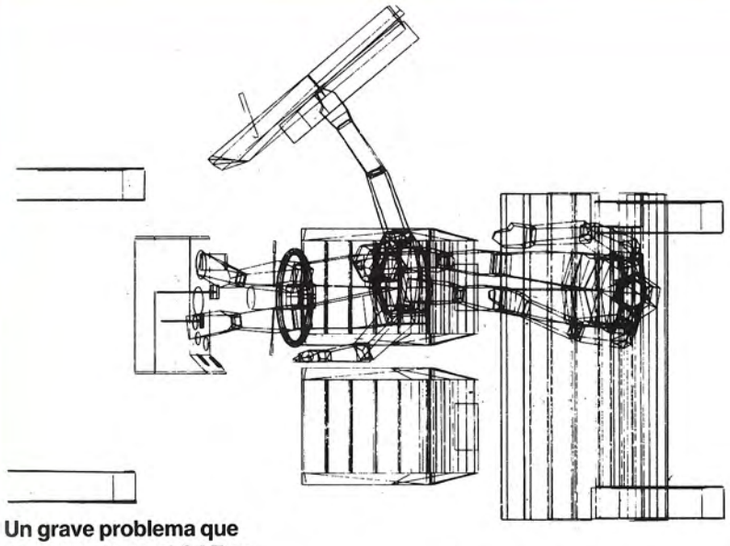


Dos maniquís sentados en la cabina de pilotaje de un Boeing. En este caso, su aspecto es decididamente humano.

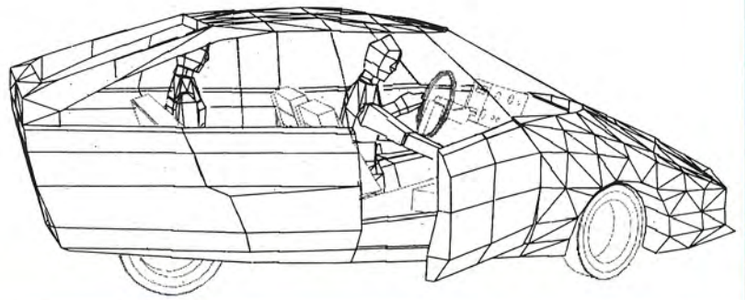
sitúa rápidamente en un espacio tridimensional, respetando las limitaciones impuestas por la envoltura exterior. En la práctica, el proyectista debe hacer casi una docena de tentativas, que con un sistema manual le habría ocupado bastantes semanas. En conjunto, la distribución del habitáculo necesita cerca del 25% del tiempo total del proyecto.
Una de las prestaciones más interesantes del SAMMIE permite representar una figura humana, llamada operador o modelo-hombre, y colocarlo en el interior del ámbito de trabajo. Esta especie de maniquí reproduce las características cinemáticas del cuerpo humano mediante una serie de uniones articuladas que enlazan los miembros rígidos del modelo, de forma similar a una musculatura real. A primera vista podría pensarse que una figura tan rígida podría tratarse también con ecuaciones matemáticas, pero de esta manera ya no se tendría la prontitud de comunicación para las comparaciones entre los demás proyectistas y de la dirección. El operador da rápidamente la idea de la orientación y de la posición asumidas por el cuerpo humano, y el proyectista puede así verificar fácilmente la relación que se establece entre el hombre y su ambiente de trabajo. Por otra parte, el usuario puede modificar fácilmente la tabla de datos que sirven al SAMMIE para crear el modelo-hombre para analizar las diversidades de comportamiento debidas a la variedad de cuerpos existente en el mundo real. Finalmente, el maniquí está completamente articulado en todas sus uniones, pero sus movimientos son controlados por una tabla que impide que asuma posiciones anómalas. Por tanto, el proyectista puede intervenir también sobre esta segunda tabla para simular, por ejemplo, el efecto de un vestido demasiado ajustado o las restricciones impuestas por el ámbito de trabajo.
La flexibilidad del modelo corresponde a la flexibilidad con que el usuario hace mover las uniones. A estas instrucciones, el sistema responde con varios niveles de posiciones estándar y de rotaciones implícitas y explícitas de las uniones, cuyo nivel de precisión depende de las dimensiones del tipo particular de operador que se ha elegido. Si ahora volvemos a nuestro modelo de automóvil constituido por superficies exteriores y objetos interiores, descubrimos que el proyectista puede simular que se coloca en el puesto de conducción. Para ello, le bastará con hacer coincidir la visual del operador, con la perspectiva del modelo.

De esta manera podrá echar una ojeada también sobre el cuadro de instrumentos y, con una sencilla petición de estado, podrá hacerse una idea exacta del desplazamiento que deben realizar sus ojos para efectuar estas operaciones y qué parte de la carretera que tiene delante queda fuera de su campo de visión. Para hacer todavía más real esta verificación, pueden construirse cilindros y distribuirlos por el cuadro en posiciones adecuadas para simular la presencia de instrumentos. Utilizando el módulo SIGHT (visión) y un modelo de volante pueden estudiarse después diversas posibilidades alternativas para minimizar la ocultación del cuadro y la reducción de la visual. Finalmente, ulteriores afinaciones pueden comprender un estudio paramétrico de estos efectos en función de las diferentes envergaduras posibles del conductor.
Después de resolver el problema de la visibilidad, había que comprobar la posibilidad de maniobra.
El módulo REACH (alcanzar) permite que el proyectista pida al modelo-hombre que toque, o intente tocar, un punto cualquiera del entorno de trabajo. Esta operación puede realizarse con los dedos o con la palma de una mano, con el tacón o con la suela de un zapato. Este módulo puede hacerse funcionar con un posicionado de tipo grosero, obtenido con el cursor, o con precisión geométrica basada en coordenadas tridimensionales. En la práctica puede servir para descubrir que el conductor no puede alcanzar el cuadro de instrumentos y que las palancas de mando deben transferirse a una consola. Gracias a la naturaleza jerárquica del sistema, las funciones cinemáticas pueden asociarse también a los componentes mecánicos. Por ejemplo, los pedales y la palanca de cambio de marchas pueden hacerse girar alrededor de un sistema de coordenadas locales para simular las posiciones límite y volver a verificar las posibilidades de REACH. Por tanto, pueden ser necesarias iteraciones para evitar, por ejemplo, que, en cuarta, la palanca del cambio produzca calambres al 95% de conductores o que el 5% deba realizar un esfuerzo excesivo para entrar la primera o la marcha atrás. Esta posibilidad de un rápido cambio de la perspectiva, junto a la gran biblioteca de las macro y a la interactividad con el ambiente de trabajo y el modelo-hombre, permite una notable reducción de los tiempos de proyecto con respecto a las metodologías tradicionales de tipo manual.

Después de haber completado el interior, todavía hace falta volver más extensamente sobre la cuestión de la visibilidad, que en nuestro caso aportó modificaciones al capó y a la parte posterior del portaequipajes. Estas superficies de tipo curvilíneo se habían construido con interpolaciones trianguladas, aunque la técnica permite actualmente determinar complejas descripciones de superficie de modelos muy precisos. Una vez definidos estos poliedros, que no deben ser necesariamente de tipo cerrado, pueden entrar a formar parte de la estructura jerárquica del sistema e integrarse al resto del modelo. De esta manera, el proyectista puede desplazar subestructuras en el interior del vehículo o en todo el vehículo en el ámbito del ambiente de trabajo.
Una vez mejorada la superficie exterior de la carrocería pueden hacerse posteriores evaluaciones, sobre todo en lo que respecta a la visualidad de la instrumentación y de la carretera. Por ejemplo, puede añadirse una retícula para representar el terreno y emplazar en ella modelos de objetos, como por ejemplo bordillos o semáforos, para verificar la reducción de visibilidad debida a la presencia de un pasajero en el asiento delantero o de espejos retrovisores. De esta forma puede diseñarse un mapa completo de visibilidad y obtener muchas informaciones útiles incluso antes de tener los modelos en madera o los prototipos.
El SAMMIE también dispone de un módulo llamado MIRROR (espejo) que transforma algunas superficies interiores del modelo en espejos dotados de unas características ópticas que hacen aparecer en perspectiva las imágenes reflejadas.
Además, el usuario puede servirse del comando adjust for object *(apuntar a un objeto) que, basándose en la última perspectiva reclamada, modifica la posición de un espejito de manera que la bisectriz de los dos vectores, que van del ojo del conductor al espejo y de éste al objeto a encuadrar, resulte perpendicular a la superficie del espejo. Esta operación hace que la imagen del objeto en cuestión aparezca en perspectiva en el espejo, acompañada de eventuales zonas de sombra o de interferencia. La eliminación de las líneas ocultas es una de las cuestiones más controvertidas en las aplicaciones gráficas por la cantidad de tiempo de máquina que requiere.*
El SAMMIE ofrece la posibilidad de eliminar las líneas ocultas, tanto en los dibujos como en el



La inserción de los oportunos comandos permite verificar las posibilidades de movimiento de las personas sentadas en el habitáculo. En esta imagen, el SAMMIE está comprobando que la manija de la puerta quede al alcance de la mano.



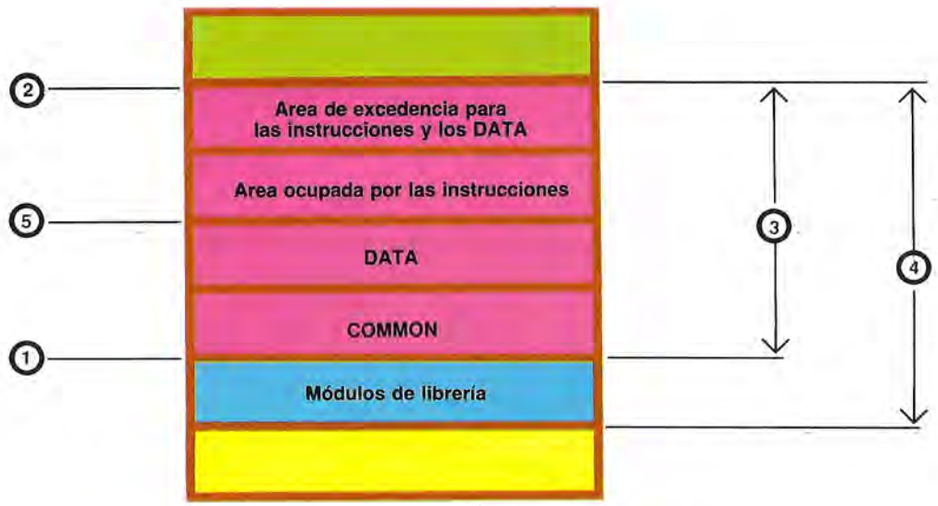
Un grave problema que se presenta en el CAE es la escasa legibilidad de las imágenes de trazador gráfico debida a la permanencia de líneas internas que en realidad deberían quedar ocultas por las superficies exteriores. Con el SAMMIE pueden eliminarse mediante un comando.

vídeo, pero las modalidades normales de uso se basan en la construcción de un modelo (representado por una trama de líneas), en las rotaciones y en la memorización de las diversas perspectivas. Al final de la jornada, el proyectista también puede obtener una copia de las perspectivas que ha utilizado, empleando estas funciones de hidden lines *(líneas ocultas) y conservarlas como documentación.*
Por tanto, el proceso del proyecto sigue la misma pauta de las técnicas de tipo tradicional, pero puede compactarse en tiempos más cortos y proporciona todas las informaciones necesarias para construir el mock-up *(simulación) exacto desde el primer momento.*
En el actual estado de la técnica, los sistemas CAE permiten reducir los costos de proyecto con relaciones variables entre 5 a 1 y 100 a 1 y más. Normalmente, estas evaluaciones no tienen en cuenta los costos debidos a la realización del modelo, pero incluso añadiendo este factor, el resultado final continúa siendo ampliamente positivo. En nuestro ejemplo, cerca del 30% del tiempo total del proyecto fue absorbido por la carrocería exterior, comprendido el tiempo dedicado al capó y al portaequipajes y disponiendo de una biblioteca de superficies muy limitada. El interior, que a su vez no podía contar con una gran biblioteca de componentes, necesitó un 25% más, y el 45% restante fue absorbido por la evaluación e interpretación de los resultados. En total, todo el proyecto no precisó más de cinco jornadas y media.
En conclusión, puede decirse que con el SAMMIE, la fantasía del proyectista puede ir más allá de los límites impuestos por una hoja de papel bidimensional o por costosos y laboriosos modelos tridimensionales.

(Extraido de «La ergonomía en la práctica», DATA MANAGER n.º 27, octubre 1983. © Systems International, Londres, Reino Unido)

ro la carga en memoria de uno cubre el área ocupada por el anterior; todas las variables se ponen a cero y no se produce ninguna transferencia de los valores. Para obtener la transferencia de los datos de un programa a otro, las variables a transferir deben declararse «comunes» (COMMON). Éstas se colocan en un área de memoria separada y no se ponen a cero durante la siguiente carga de los programas.
El área reservada al sistema contiene todos los punteros y las otras informaciones necesarias para el diálogo con el sistema operativo.
El área denominada «módulos de librería» contiene las funciones particulares del Basic reclamadas por el programa de aplicación. En realidad hay más funciones de las necesarias para conseguir un ahorro de tiempo.
Si para cada programa de aplicación tuviesen que seleccionarse o tomarse únicamente las rutinas estrictamente necesarias, se produciría una larga operación de selección y de encadenamiento, todavía más complicada por el hecho de que algunas rutinas de sistema llaman a otras. Por tanto, para tener una carga selectiva, sería necesario trabajar con una compleja cadena de punteros. El método elegido, en cambio, ha sido el de dividir las rutinas en dos categorías: las más frecuentes y las menos utilizadas. Durante la fase de encadenamiento se cargan las más frecuentes (contenidas en BRUN. COM) y, sólo si es necesario, las otras (contenidas en BASLIB. REL).
Por esta razón, un programa, incluso mínimo, ocupa por lo menos 16 kbytes de memoria, que es la extensión mínima que puede alojar el módulo que contiene las rutinas de sistema.

MAPA DE DISTRIBUCION DE LA MEMORIA



Area utilizada por las instrucciones, los DATA y los COMMON del fuente

Area reservada a los módulos de librería (p. e. las funciones) reclamados por el fuente

Memoria no ocupada

Area reservada al sistema

TEST 20

1 / La siguiente subrutina de gestión de disco se utiliza en diversos puntos del programa principal, que le transmite el dato a memorizar (BF$) y el número de record (REC).

```
110 OPEN "R",1,"A:PRUEBA",128
120 FIELD 1,128 AS A$
130 LSET A$ = BF$
140 PUT 1,REC
```

¿Qué errores contiene?

2 / ¿Cuáles de estas afirmaciones son verdaderas y cuáles falsas?
a) Un programa compilado ocupa más espacio en la memoria que el fuente en Basic.
b) El Compilador puede detectar errores que pueden escapar al Intérprete.
c) Un programa compilado no puede copiarse.

3 / ¿Cuáles son las operaciones a realizar para obtener un programa compilado?

4 / ¿Está ligado un Compilador al tipo de microordenador, o puede utilizarse en cualquier máquina?

Las soluciones, en la pág. 737.

Instrucciones particulares y compendio del Basic 80

En este apartado presentamos algunas instrucciones y algunos comandos que no se han descrito anteriormente porque su comprensión requiere un buen conocimiento del lenguaje Basic y de la estructura de la máquina.
Empezaremos con cuatro instrucciones de empleo general y a continuación pasaremos a examinar, a grandes rasgos, los problemas relacionados con la gestión de las funciones gráficas y de algunos periféricos particulares. Este último tema se profundizará en un capítulo aparte.

Instrucciones de uso general

En la exposición de las instrucciones del Basic hemos introducido ya algunos conceptos correspondientes a la segmentación y encadenamiento de programas y a la llamada de subrutinas. Para completar el cuadro es necesario ilustrar otras tres funciones:

CALL
COMMON
USR

Después se hablará de la instrucción STOP.

CALL. Permite llamar una subrutina escrita en lenguaje Assembler.
La sintaxis de la instrucción es la siguiente:

CALL Variable (Argumentos)

donde Variable es un nombre simbólico que contiene la dirección de partida de las subrutinas a llamar, y Argumentos es una lista de variables a pasar a la subrutina externa para que pueda utilizarse. Por ejemplo, las instrucciones

```
10 SUBA = &HA000
20 CALL SUBA(I,X)
```

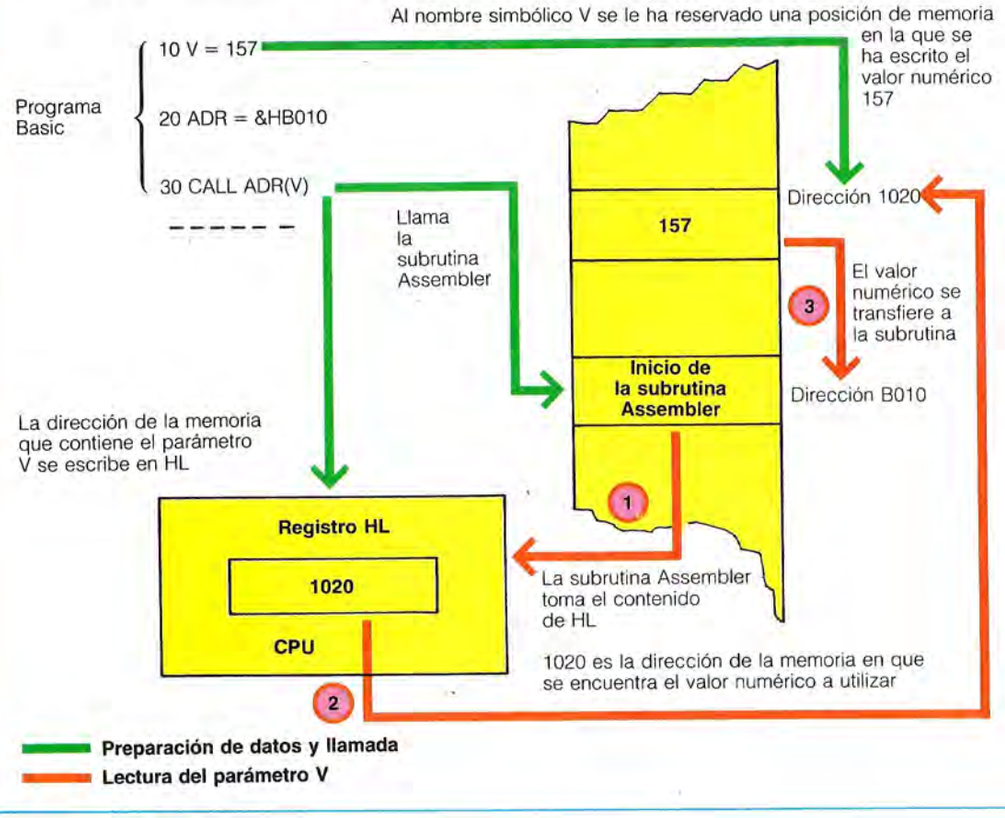
llaman una subrutina escrita en Assembler que empieza por la localización de memoria A000 (los símbolos &H indican que el valor numérico que sigue, A000, es hexadecimal) pasándole los valores contenidos en las variables I y X. Los datos a transmitir deben haberse asignado antes a otras tantas variables, puesto que generalmente no es posible transferir valores constantes. Cuando debe transferirse un valor constan-

te, antes debe asignarse a una variable cualquiera y después pasar esta variable. Por ejemplo, las instrucciones que permiten transferir el valor 157 a una subrutina que empieza en la posición B010 son las siguientes:

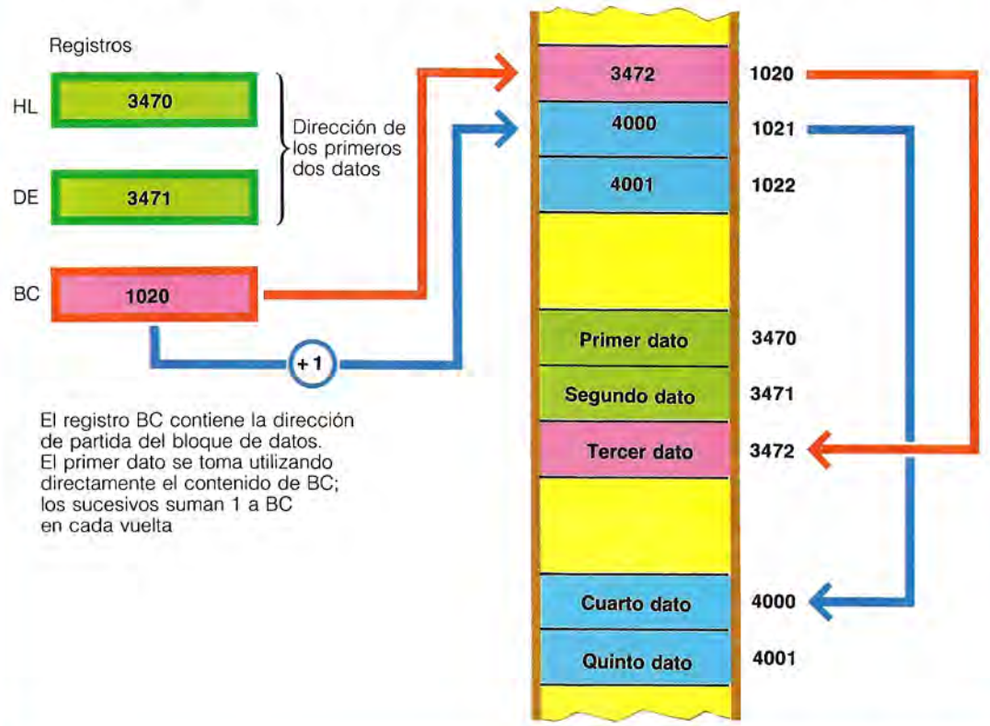
```
10 V = 157
20 ADR = &HB010
30 CALL = ADR(V)
```

En otros lenguajes, como por ejemplo el Fortran, el código CALL se utiliza para pasar el control a una subrutina interna, escrita en el mismo lenguaje que el programa que llama. En Fortran, la instrucción CALL ADR (157) pasa el control a una subrutina llamada ADR, transfiriéndose el valor numérico 157. Por tanto, se trata de una instrucción más similar a la GOSUB del Basic que no a la CALL.
En el Basic compilado no es estrictamente necesario especificar la dirección de partida de la subrutina llamada: el Montador la busca y la asigna durante la carga.
La instrucción CALL puede utilizarse para acceder a posiciones de memoria no gestionadas por el Basic. Algunos Intérpretes y Compiladores sólo permiten direccionar una determinada área de memoria; en estos casos puede accederse a la parte restante escribiendo una o más subrutinas en Assembler y llamarlas al ambiente Basic con la instrucción CALL. La transferencia de los parámetros, que vista desde el Basic es muy sencilla, en la subrutina en Assembler puede presentar algunas dificultades debido al complejo mecanismo de transferencia.
Los parámetros transmitidos (Argumentos) son interpretados como direcciones de las memorias en que se encuentran los valores propiamente dichos. Cada memoria está compuesta de dos bytes; por tanto, los parámetros ocupan siempre dos bytes, independientemente del tipo (entero, real, etc.).

TRANSFERENCIA DE UN PARAMETRO CON LA INSTRUCCION CALL



ESQUEMA LOGICO DE LA TRANSFERENCIA DE 5 PARAMETROS



La transferencia de los parámetros del Basic al Assembler se realiza de acuerdo con diferentes modalidades, en función de su número.
Si el número de parámetros es inferior o igual a tres, se utilizan los registros*, mientras que si su número es mayor, se utiliza un bloque de memorias contiguas.
Por ejemplo, si se transfieren dos parámetros, el primero se coloca en HL** y el segundo en DE (HL y DE son los nombres de los dos registros). El programa Assembler debe tomar el contenido de estos dos registros y utilizarlo como dirección para encontrar los valores de la variable en la correspondiente posición de memoria.
La llamada CALL ADR(V) escribe pues en el registro HL la dirección en que se encuentra el valor asociado a la variable V; la subrutina Assembler toma el contenido de HL, lo considera como una dirección y, por tanto, toma el valor numérico contenido en la posición de memoria indicada por HL.
La transferencia de un número de parámetros superior a tres es más complicada y requiere el uso de subrutinas de sistema para el alojamiento de las áreas de memoria a transferir. Normalmente, los dos primeros parámetros se pasan utilizando registros como en el caso anterior, mientras que los restantes se alojan en una zona de memoria que constituye el bloque de datos. Para tomar los valores del bloque de datos, inicialmente se transfiere a un nuevo registro (por ejemplo de nombre BC) la dirección más baja entre las reservadas a las posiciones de memoria del bloque. La lectura de los valores se hace tomando el contenido del registro (BC) y utilizándolo como dirección. En la posición de memoria I «apuntada» así, está contenido el primer valor; el siguiente se encontrará en la posición de memoria I + 1, y así sucesivamente. Arriba aparece el esquema de la lógica utilizada en este segundo tipo de transferencia.

* Recordemos que los registros son memorias particulares contenidas en la CPU y contraseñadas con nombres bien definidos.
** HL es un par de registros (H + L) utilizados conjuntamente.

COMMON. Esta instrucción permite transmitir los necesarios parámetros de un programa que llama a otro llamado con CHAIN. La sintaxis es

COMMON Lista de variables

Las variables indicadas en la lista se transfieren de uno a otro programa. Para transferir una matriz deben añadirse los símbolos ( ) junto al nombre. Por ejemplo, las instrucciones

```
10 DIM B(30),E(9)
20 COMMON V,A,B( ),E( ),K$
.....
276 CHAIN "PRUEBA",10
```

transfieren al programa PRUEBA los valores contenidos en las variables numéricas V y A, en las matrices B(30),E(9) y en la cadena K$. Un área COMMON, constituida por más variables como en el ejemplo precedente, también puede indicarse con un nombre único; en este caso se trata de una característica peculiar del Fortran extendida después al Basic. Utilizando un COMMON con un nombre (labelled common) se simplifica el paso de parámetros a las subrutinas «externas» (en Fortran o en Assembler) porque ya no es necesario incluir en la CALL toda la lista de variables a transferir; basta con citar el nombre «colectivo» utilizado en el COMMON. La sintaxis de la instrucción es:

COMMON / NOMBRE / V1,B( ),V3

Las variables V1, V3 y la matriz B están agrupadas bajo el nombre colectivo NOMBRE.
Todos los programas encadenados que utilizan un COMMON deben declararlo expresamente. En el ejemplo anterior, el programa PRUEBA encadenado a la línea 276 debe contener una declaración de COMMON de formato idéntico al que aparece en la línea 20 del programa que llama. Para evitar escribir más veces las mismas líneas (una por cada programa llamado) se usa una instrucción particular (%INCLUDE) que permite a cada uno de los programas llamados tomar las declaraciones COMMON contenidas en un file separado. De este modo, el COMMON se escribe una sola vez y se carga más veces según las necesidades.
Por ejemplo, supongamos que los programas a encadenar son MAIN, PROG y PROG2, memorizado cada uno en un file (de tipo BAS) y que el COMMON debe proporcionar las siguientes variables (utilizadas por todos los programas): V1, V2, B(30) y A$(6).
Las direcciones COMMON y de dimensionado de las matrices están memorizadas en un file separado, por ejemplo de nombre COMUN. Para cargar el contenido en los programas encadenados debe utilizarse la instrucción

%INCLUDE COMUN

Los tres programas indicados en el ejemplo tendrán entonces la siguiente estructura:

```
MAIN.BAS (Nombre del primer file; llama a los otros)
10 %INCLUDE COMUN
.....
165 CHAIN "PROG"
.....
460 CHAIN "PROG2"

PROG.BAS (Primer encadenado)
10 %INCLUDE COMUN
.....

PROG2.BAS (Segundo encadenado)
10 %INCLUDE COMUN
.....

COMUN.BAS (Declaraciones)
20 DIM B(30),A$(6)
30 COMMON V1,V2,B( ),A$( )
```

La instrucción de la línea 10 produce la inclusión, en cada programa, del contenido del file COMUN; será como si las instrucciones contenidas en éste (líneas 20 y 30) estuviesen presentes en cada programa.

USR. Define la dirección de partida de una subrutina en Assembler. La sintaxis es:

DEF USRN = VALOR

donde N es un número (de 0 a 9) que identifica una de las rutinas definidas en el mismo programa (cada una está asociada al número empleado) y VALOR es la dirección de partida. El objeto de la USRN es definir, en el interior del programa Basic, las subrutinas Assembler como si fuesen funciones y, por tanto, es análoga a la DEF FNX. Sin embargo, mientras esta última

SOLUCIONES DEL TEST 20

1 / Los errores son dos. El primero es determinante y no permite la ejecución del programa; el segundo sólo podría producirse en condiciones particulares.
Primer error. La instrucción 110 (OPEN) se realiza cada vez que se llama la rutina. La primera vuelta es correcta, y tiene como efecto la apertura del file (o su creación); en cambio, la siguiente vuelta produce un error de sistema, con la consiguiente interrupción del programa, puesto que intenta abrir un file ya abierto.
Segundo error. En la fase de escritura o de lectura de disco es conveniente controlar que el número del record (REC, instrucción 140) esté comprendido entre 1 y el máximo previsto para el file. Efectivamente, en la llamada a la subrutina podría verificarse por error la asignación REC = 0. En este caso, la ejecución se detendría, porque el record 0 normalmente no existe (sólo en algunas máquinas se ha previsto que el primer record de un file sea el número 0). Por otra parte, si a la variable REC se le asignase un valor superior al máximo número de records contenidos en el file, podría tenerse la escritura en una zona del disco que no perteneciese al file, y esto produciría la destrucción de otros datos.

2 / a) falsa; b) verdadera. La afirmación c) debe interpretarse: el programa compilado no puede ser copiado como fuente, pero puede ser duplicado en otro disco. Con determinados trucos, la copia puede inhibirse o limitarse a un número máximo preestablecido. Sin embargo, se trata de métodos muy complicados, utilizados solo por determinados programadores.

3 / La sucesión de las operaciones a realizar es la siguiente:
1 - Escritura del fuente en Basic y su memorización en disco (en formato ASCII)
2 - Lanzamiento del Compilador
3 - Lanzamiento del Montador
Al final se obtiene un programa ejecutable memorizado en un file en disco. Para ordenar su ejecución (bajo sistema operativo) sólo debe introducirse el nombre.

4 / El Compilador está estrechamente ligado a la máquina o, mejor, a la CPU utilizada.

define funciones en Basic, la USRN utiliza una subrutina Assembler como función Basic.
Por ejemplo, la instrucción DEF USR3 = 1740 define la subrutina Assembler que empieza desde la posición de memoria 1740 y a ésta le asocia el número 3. Para utilizarla es suficiente con llamarla como si se tratase de una función de usuario. La instrucción X = USR3(K) transfiere a X el resultado de cálculo realizado en la subrutina Assembler 3 utilizando K como parámetro.
El argumento (K) puede ser de cualquier tipo (numérico o de cadena). Para comunicar al Assembler de qué tipo se trata se utiliza un determinado registro, el registro A, que contiene un código ligado al tipo del argumento (por ejemplo, el valor 2 indica un entero, el valor 3 una cadena, y así sucesivamente).
El valor del argumento (valor de K) se transfiere de dos maneras: para los numéricos se utiliza un par de registros que indica la dirección de la memoria que contiene el valor y, para las cadenas, otros dos registros que contienen la dirección de la descripción de la cadena.
Normalmente, esta descripción es una zona de memoria de 3 bytes de longitud, el primero de los cuales contiene la longitud de la cadena en caracteres (de 0 a 255), mientras que los otros dos contienen la dirección de inicio del área de memoria en que se encuentran los valores (caracteres de la cadena).
El valor del argumento se toma con un solo paso si se trata de un dato numérico (los registros

proporcionan su dirección y, por tanto, es suficiente con tomar el contenido de la memoria «apuntada» por aquéllos), mientras que para las cadenas debe hacerse una lectura de más. En este caso, los registros contienen la dirección del bloque de descripción (3 bytes) donde está memorizada la dirección del dato. Normalmente, el bloque de descripción de las cadenas y las propias cadenas están alojados en el área de memoria de usuario, mientras que los datos numéricos están memorizados en áreas particulares llamadas Acumulador de Coma Flotante (FAC), y los registros de direccionamiento apuntan al acumulador particular interesado en el intercambio. En algunas versiones del Basic, las subrutinas Assembler no pueden definirse utilizando la DEF USRN.
En estos casos, existen determinadas posiciones de memoria reservadas a las que debe transferirse la dirección de inicio antes de utilizar cada subrutina. El nombre simbólico de estas posiciones de memoria depende de la versión particular del Basic. Para llamar una subrutina, primero debe escribirse la dirección de partida con la instrucción POKE en la posición de memoria reservada a este fin.
Estas versiones del Basic no permiten elegir la subrutina especificando su número: en cualquier caso se llamará la subrutina cuya dirección de inicio se haya escrito con la POKE; para llamar otra rutina, antes deberá transferirse la nueva dirección de inicio.
Como ejemplo, en la página siguiente se ha representado el programa que utiliza las instrucciones POKE y PEEK para escribir dos símbolos-color a elegir entre 5 (línea 90) y releer, para cada uno, el color seleccionado. El programa corre sobre el COMMODORE VIC-20 y, por tanto, las direcciones empleadas sólo valen para esta máquina.

STOP. La función que realiza es detener la ejecución del programa y volver al estado de comandos de forma similar a la instrucción END. La diferencia consiste en un mensaje emitido por el sistema, que indica el número de línea en la que se ha verificado el STOP.
El uso más difundido de la instrucción STOP es el de terminador en caso de errores no recuperables. La terminación normal del programa siempre se define con la instrucción END (que es conveniente que sea única, porque todas las posibles salidas del programa deben direccionarse a la misma línea END), mientras que, al producirse errores, pueden preverse más salidas con la instrucción STOP.
En los puntos en que hay un STOP se suspende la ejecución del programa, pero los files eventualmente utilizados no se cierran, al contrario de lo que sucede con la END; la ejecución del programa puede volver a arrancar desde la siguiente línea del STOP introduciendo el comando CONT (continuar).
La suspensión momentánea del programa permite analizar de forma inmediata la situación que se ha presentado en el Basic y reemprender su ejecución después de las eventuales correcciones. Por ejemplo, supongamos que un programa debe realizar algunas divisiones utilizando como divisores las variables D1, D2 y D3 calculadas en el propio programa. Un control a prever consiste en verificar que las variables (divisores) no tengan valor cero. Puede darse que esto suceda por un error o en cálculos anteriores, lo que conduciría a ulteriores errores en el desarrollo de las sucesivas instrucciones (divisiones por cero). Los necesarios controles pueden efectuarse con las sucesiones de IF y STOP, por ejemplo, de la siguiente manera:

```
10 IF D1 = 0 THEN STOP
20 IF D2 = 0 THEN STOP
30 IF D3 = 0 THEN STOP
```

Al producirse una de las condiciones de error, el programa se detiene, en pantalla aparece el número de línea de la instrucción que ha generado el STOP y el sistema pasa al estado de comandos. En estas condiciones, utilizando el Basic de modo inmediato, pueden examinarse los contenidos de todas las variables que han participado en el cálculo del divisor erróneo y, a continuación, reemprender la ejecución del programa para examinar el efecto de este error sobre los cálculos que siguen.
También puede asignarse un valor ficticio al divisor erróneo y proseguir normalmente el programa (en este caso no pueden aportarse correcciones al programa).

Instrucciones orientadas a los gráficos

Muchas formas de Basic poseen instrucciones particulares para la presentación de gráficos. El tema se tratará más a fondo más adelante; en este capítulo sólo se considerarán algunas formas principales de instrucciones gráficas, sin

EJEMPLO DE USO DE POKE Y PEEK

```
0 REM:MAIN DE EJEMPLO SOBRE EL USO DE POKE Y PEEK*
7PRINT " ■ "; POKE 36879,29
10 DIM VCOL(5),NICOL$(5)
20 FOR I=1 TO 5
30 READ VCOL(I)
40 NEXT I
50 DATA 0,2,5,7,6
60 FOR I=1 TO 5
70 READ NCOL$(I)
80 NEXT I
90 DATA NEGRO,ROJO,VERDE,AMARILLO,AZUL
100 INPUT "PRIMER COLOR";COL$
110 K1=0
120 FOR I=1 TO 5
130 IF COL$=NCOL$(1) THEN K1=I
140 NEXT I
150 IF K1=0 THEN PRINT "Error": GOTO 100
160 N1=VCOL(K1)
170 INPUT "SEGUNDO COLOR";COL1$
180 K2=0
190 FOR I=1 TO 5
200 IF COL1S=NCOLS(I) THEN K2=I
210 NEXT I
220 IF K2=0 THEN PRINT "Error": GOTO 170
230 N2=VCOL$(K2)
240 POKE 8040,83
250 POKE 38760,N1
260 POKE 8044,83
270 POKE 38764,N2
275 PRINT "CONTROL COLOR"
280 INPUT "COLOR (1/2)";NL
290 IF NL=1 THEN A=PEEK (39860)
300 IF NL=2 THEN B=PEEK (38764)
310 IF A=N1 THEN PRINT "COLOR="NCOL$(K1):GOTO 360
320 IF B=N2 THEN PRINT "COLOR="NCOL$(K2):GOTO 360
360 END
```

entrar en el detalle de su empleo. En el uso gráfico, la pantalla está dividida en **pixels** (picture elements, elementos de imagen).
La subdivisión más difundida es la de 512 pixels en horizontal por 256 pixels en vertical; esto significa que en lugar de las 80 columnas de caracteres es posible utilizar 512 puntos de pantalla y, en lugar de las 24 líneas de texto, pueden direccionarse 256 puntos de pantalla. La presentación de un segmento en la pantalla se obtiene desplazando el cursor desde el principio al final del segmento, proporcionando las coordenadas de los dos puntos de pixels.
Por ejemplo, las instrucciones

```
LINE (10,5) – (70,100)
HPLOT 10,5 TO 70,100
```

generan la presentación de un segmento que empieza en las coordenadas 10,5 y termina en el punto 70,100.
Con una adecuada sucesión de instrucciones del tipo anterior pueden generarse figuras muy diversas; por ejemplo, un rectángulo puede trazarse con cuatro llamadas sucesivas, una para cada lado. En algunas máquinas, una sucesión de llamadas puede memorizarse en disco en forma binaria para volver a utilizarlas en su totalidad, con una sola instrucción. Por ejemplo, la instrucción DRAW "n" determina la presentación en pantalla de la «tabla gráfica» número n memorizada antes y que puede contener figuras incluso muy complicadas, obtenidas mediante un número de desplazamientos muy elevado.
En casi todas las máquinas que utilizan este método (o uno equivalente) hay previstas instrucciones para el cambio de escala, para la rotación del dibujo o para la coloración de partes de la pantalla (las formas más utilizadas para activar las dos primeras funciones son SCALE = ... y ROT = ...). El uso de las instrucciones vistas y de las otras existentes en gran cantidad en casi todas las máquinas se expondrá en el capítulo dedicado a la gráfica de ordenador.

Instrucciones particulares

Dada su facilidad de aprendizaje, el lenguaje Basic ha sido adoptado en todas las máquinas de la categoría micro, personal y doméstica. Su notable difusión ha conducido a los fabricantes a desarrollar interfaces particulares, con software asociado, para la gestión de periféricos desconocidos en sistemas más grandes.
Estos periféricos, nacidos con los videojuegos, se han utilizado en aplicaciones profesionales, y se han convertido en un excelente medio de comunicación entre el usuario y la máquina.

Generación de sonidos. La generación de sonidos es un tema muy vasto y complejo, que cubre desde la simple emisión de un «bip» hasta la síntesis vocal, la cual permite al calculador emitir frases en diversos idiomas. Las aplicaciones más complicadas, que implican la presencia de un hardware especializado, se ilustrarán más adelante. En este capítulo sólo se considerarán algunas instrucciones Basic que permiten generar sonidos o música.
La forma de las instrucciones varía mucho de máquina a máquina; sin embargo, pueden identificarse dos categorías principales: las máquinas que tienen subrutinas de sistema dedicadas, para las cuales bastan unas cuantas instrucciones en simbólico, y las máquinas que requieren el uso de la instrucción PEEK o del lenguaje Assembler.
La existencia de esta diferencia está originada sobre todo en la baja velocidad de ejecución de las instrucciones en Basic interpretado.
La generación de un sonido se obtiene enviando las oportunas señales a un altavoz. Cuanto más agudo es el sonido, tanto más elevada debe ser la frecuencia* de la señal. Si se supera un cierto valor, la frecuencia pedida no puede generarse, ya que el Intérprete no dispone de una velocidad adecuada. En estos casos deben utilizarse rutinas Assembler (la generación de sonidos en Assembler se tratará más adelante).
Si la máquina no posee las necesarias instrucciones Basic, puede utilizarse el PEEK. Muchos ordenadores personales y domésticos disponen de una particular posición de memoria utilizada como «interruptor» para el altavoz. Cada vez que se accede a esta memoria se emite un sonido. Para crear un sonido que tenga una determinada altura (p.e. un «do»), debe establecerse un bucle que active oscilatoriamente, y durante un cierto tiempo, la posición del altavoz. Regulando adecuadamente los tiempos puede generarse el sonido deseado, siempre limitado a frecuencias bajas.
Para obtener las frecuencias más elevadas, el método es el mismo, pero el programa debe estar escrito en Assembler.
En algunos sistemas existen rutinas ya preparadas que pueden llamarse desde el Basic en forma de instrucciones. Por ejemplo, la

```
SOUND f,d
```

es una instrucción que genera un sonido de frecuencia f (normalmente comprendida entre 35 y 3200 Hz) y de duración d (en segundos). Una instrucción más compleja es la PLAY, con la que pueden especificarse diversos parámetros (estrechamente ligados a los conocimientos musicales del programador) que permiten la ejecución de verdaderos fragmentos musicales.

Joystick y paddle. Estas funciones particulares hacen pensar en seguida en los videojuegos; se trata en realidad de comandos para la gestión de interfaces con transductores de posición que pueden utilizarse en numerosas aplicaciones.
Un transductor (eléctrico) es un dispositivo que puede transformar una magnitud física en una señal eléctrica proporcional a aquélla.
Los transductores pueden ser de varias naturalezas; por ejemplo, el termómetro es un transductor que convierte las temperaturas en longitudes (de la columna de mercurio). Para el ordenador son necesarios los transductores eléctricos: para que los circuitos de la máquina puedan interpretar una magnitud física cualquiera a medir, previamente debe convertirse en señales eléctricas. El tipo de dispositivo más sencillo es el **paddle**, constituido por un transmisor que genera una salida en función de un desplazamiento. La función Basic asociada a este elemento tiene la forma PDL(n), que restituye un número comprendido entre 0 y 255, e indica la entidad del desplazamiento efectuado. El valor n es la puerta de acceso a la que está conectado el transductor. Por ejemplo, la instrucción

```
V = PDL(2)
```

adquiere la posición del paddle conectado a la puerta 2. Una vez leído, este valor puede utilizarse para posicionar un símbolo en la pantalla del vídeo.
Desarrollando repetidamente (bucle) las operaciones de lectura y de posicionado del cursor se reproduce en la pantalla, de forma continua, el desplazamiento de la palanca del paddle.
Este transcurso permite desplazamientos a lo largo de un solo eje, por lo que se reproducen como desplazamientos del cursor a lo largo de una sola línea de la pantalla.
Para tener el desplazamiento según dos direcciones (o sea en todo el plano de la pantalla) debe utilizarse el **joystick**, que está constituido por un transductor bidireccional, o sea que se comporta como dos paddles, uno dispuesto según el eje horizontal y el otro según un eje vertical. Los principales comandos de control de joystick son los siguientes:

STICK (n) restituye las coordenadas del joystick; n indica el eje. Por ejemplo, STICK(0) restituye el desplazamiento a lo largo del eje horizontal; STICK(1), el del eje vertical. Las coordenadas del punto de llegada se obtienen con las instrucciones

```
X = STICK(0)
Y = STICK(1)
```

STRING restituye el estado del botón normalmente cerrado del joystick (el que «dispara» en los videojuegos). Normalmente, el estado se indica con – 1 si el botón se aprieta, y con 0 si no. En algunas máquinas se ha previsto un parámetro [la instrucción se convierte en STRING(n)] para utilizar más joysticks o para determinar si existen peticiones «colgadas», o sea formuladas antes de la activación de la instrucción, o también para desactivar y reactivar el joystick (STRING OFF, STRING ON)

ON STRING... es una decisión de la instrucción anterior que permite activar una subrutina según el estado del botón del joystick. Así ON STRING(2) GOSUB 1000 activa la subrutina 1000 con la presión del botón 2.

K. Reese/Marka

Empleo del lápiz óptico en el diseño asistido. La operadora está elaborando la imagen de un F15.

El lápiz óptico. Es un accesorio que puede utilizarse en unión de los terminales de vídeo gráficos para la adquisición de las coordenadas de los puntos de la pantalla. En algunos dialectos Basic existen instrucciones dedicadas para la activación del lápiz y la adquisición de las coordenadas. Las principales son:

PEN ON habilita la lectura del lápiz

PEN OFF deshabilita la función anterior

Z = PEN(N) adquiere las coordenadas. El valor restituido (en la variable Z) depende del valor impuesto en el parámetro N. Por ejemplo, con N = 1, Z restituye la coordenada X; con N = 2, la coordenada Y. Los posibles valores de N y las correspondientes funciones dependen estrechamente del tipo de máquina

* La frecuencia de una señal periódica es el número de impulsos que se suceden en un segundo y se mide en Hz. Por ejemplo, a 35 Hz, en un segundo se generan 35 impulsos; cada uno tiene la duración de 1/35 de segundo (esta magnitud se llama período).

Compendio del Basic 80

INSTRUCCIONES Y COMANDOS

Instrucción (Comando)	Descripción	Ejemplo
AUTO	Numera automáticamente las líneas de programa	AUTO 10,5
CALL	Llama una rutina en Assembler	CALL SUBA
CHAIN	Llama un programa y le pasa las variables	CHAIN "PRUEBA"
CLEAR	Pone a cero todas las variables	CLEAR
CLOAD	Carga un programa residente en cassette	CLOAD "PROG"
CLOSE	Cierra un file	CLOSE 2
COMMON	Transfiere las variables entre programas (bajo CHAIN)	COMMON A(10)
CONT	Reactiva la ejecución de un programa (después de STOP)	CONT
C SAVE	Memoriza un programa en cassette	C SAVE "PROG"
DATA	Memoriza los valores de los datos constantes	DATA 3,7,9
DEF FN	Define funciones preparadas por el usuario	DEF FNX = A + B
DEFINT	Define variables enteras	DEFINT K
DEFSNG	Define variables en simple precisión	DEFSNG R
DEFDBL	Define variables en doble precisión	DEFDBL Z
DEFSTR	Define cadenas	DEFSTR A
DEFF USR	Especifica la dirección de inicio de una rutina en Assembler	DEF USR2 = 1275
DELETE	Cancela líneas de programa	DELETE 5-25
DIM	Reserva un área de memoria de variables dimensionales	DIM A(3), B$(10)
EDIT	Entra en el estado editor para correcciones en el programa	EDIT 20
END	Termina un programa	END
ERASE	Elimina las matrices, que así pueden redefinirse	ERASE A, B$
ERR	Variable utilizada por el sistema para los códigos de error	IF ERR = 3 THEN...
ERL	Variable de sistema para señalar la línea que genera error	IF ERL = 127 THEN...
ERROR	Simula la verificación de un error	ERROR 3
FIELD	Define las asignaciones de un buffer I/O disco	FIELD 2,5 AS C$
FOR...	Bucle	FOR I = 1 TO 100
NEXT	Lectura de un record	NEXT I
GET	Recupera un record de un file de entrada	GET 1,56
GOSUB	Transfiere el control a una subrutina interna	GOSUB 5740
GOTO	Salta a la línea especificada	GOTO 250
IF... THEN	Realiza una elección en función de una o más condiciones	IF K = 1 THEN...
INPUT	Entrada de datos por consola	INPUT "Datos"; A,B

Instrucción (Comando)	Descripción	Ejemplo
INPUT #	Entrada de datos de file secuencial	INPUT # 1,A
KILL	Cancela un file en disco	KILL "Datos"
LET	Asigna un valor a una variable (puede omitirse)	LET A = 3 + B
LINE INPUT	Entrada de una línea completa de caracteres	LINE INPUT A$
LINE INPUT #	Entrada de una línea de caracteres de un file secuencial	LINE INPUT # 1,A$
LIST	Lista en pantalla el programa residente en memoria	LIST 10-50
LLIST	Lista un programa a impresora	LLIST 21-90
LOAD	Carga un programa del disco	LOAD "A:PRUEBA"
LPRINT	Imprime las variables especificadas	LPRINT A,B,V
LPRINT USING	Imprime con formato	LPRINT USING "# . #"
LSET	Transfiere los datos agrupándolos a la izquierda	LSET A$ = C$
MERGE	Une al programa en memoria uno del disco	MERGE "A:PROG"
MID$	Sustituye una parte de una cadena	MID$(A$,3) = "X Y"
NAME	Cambia el nombre de un file en disco	NAME "PRUEBA" AS "NUEVO".
NEW	Cancela el programa contenido en memoria	NEW
NULL	Predispone un determinado número de espacios a escribir al final de cada línea	NULL 5
ON ERROR...	Define una trampa para los errores de sistema	ON ERROR GOTO 20
ON... GOTO	Hace un salto en función del valor de un parámetro	ON K GOTO 10, 20...
ON... GOSUB	Transfiere el control a una subrutina en función de un parámetro	ON L GOSUB 30, 80...
OPEN	Abre un file en el disco	OPEN "I", 2, "DATOS" OPEN "R", 1, "X", 50
OPTION BASE	Define el primer valor de los índices (1 o 0)	OPTION BASE 1
OUT	Transfiere un byte en la puerta de salida	OUT 2, 170
POKE	Escribe un byte en la posición de memoria especificada	POKE 2320, 170
PRINT	Escribe en pantalla	PRINT A,B,C$
PRINT USING	Escribe en modo formateado (en pantalla)	PRINT USING "# . #"; A
PRINT #	Escribe en un file secuencial	PRINT #3, V, C$
PRINT # USING	Escribe en un file secuencial utilizando un formato	PRINT #3 USING "# . #"; A
PUT	Escribe un record en un file directo	PUT 3,26
RANDOMIZE	Inicializa el generador de números aleatorios	RANDOMIZE
READ	Adquiere los valores definidos en un DATA	READ K,L,N$
REM	Permite insertar comentarios. En algunas máquinas puede ser el símbolo '. y en otras !	REM ** Comentario **
RENUM	Renumera un programa residente en memoria	RENUM 10,20,5

Instrucción (Comando)	Descripción	Ejemplo
RESTORE	Reposiciona el puntero en los DATA	RESTORE 1475
RETURN	Restablece el control al programa que llama	RETURN
RESUME	Reactiva la ejecución de un programa después de un error	RESUME 5
RSET	Transfiere los datos agrupándolos a la derecha	RSET A$ = B$
RUN	Pone en ejecución un programa	RUN
SAVE	Memoriza un programa en disco	SAVE "A:PRUEBA"
STOP	Termina la ejecución de un programa y vuelve al estado de comandos	STOP
SWAP	Intercambia el contenido de dos variables	SWAP A$, B$
TRON	Activa la visualización de las instrucciones ejecutadas	TRON
TROFF	Desactiva TRON	TROFF
WAIT	Produce una espera hasta la llegada de un dato valor desde la puerta especificada	WAIT 2,240
WHILE... WEND	Realiza un bucle hasta que una condición es verdadera	WHILE K WEND
WIDTH	Asigna a la impresora el máximo número de caracteres	WIDTH 60
WRITE	Escribe en un terminal	WRITE A,B
WRITE #	Escribe en un file secuencial	WRITE # 2,A,B

FUNCIONES

Función	Descripción	Ejemplo
ABS(V)	Restituye el valor absoluto	ABS (–3 * 5)
ASC (A$)	Restituye el código ASCII correspondiente al primer carácter de la cadena	ASC (A$)
ATN (A)	Arcotangente de A	ATN (2)
CDBL (R)	Convierte en doble precisión	CDBL (37.4)
CHR$ (N)	Restituye el símbolo ASCII correspondiente al valor numérico decimal N	CHR$ (68)
CINT (K)	Convierte K en entero con redondeo	CINT (4.75)
COS (A)	Calcula el coseno del ángulo A (en radianes)	COS (2.1)
CSNG (D)	Convierte el número D en simple precisión	CSNG (7.569114)
CVD (A$)	Convierte la cadena en un número en doble precisión	CVD (A$)
CVI (A$)	Convierte la cadena en un número entero	CVI (A$)
CVR (A$)	Convierte la cadena en un número real	CVR (A$)
EOF (N)	Restituye –1 si se encuentra al final de un file secuencial	IF EOF (1)...
EXP (N)	Calcula el exponencial e^N	EXP (3.5)
FIX (X)	Trunca el valor de X a su parte entera	FIX (57.921)
FRE (N)	Restituye el número de bytes todavía disponibles	FRE (0)
HEX$ (M)	Restituye una cadena que es la representación hexadecimal de M	HEX$ (75)

Función	Descripción	Ejemplo
INKEY$	Adquiere un carácter «pendiente» del terminal	A$ = INKEY$
INP(N)	Adquiere un byte de la puerta N	A = INP (4)
INPUT$(N)	Adquiere N caracteres del terminal (sin eco)	B$ = INPUT$(2)
INSTR(A$,B$)	Verifica si la cadena A$ está contenida en B$ y restituye su posición	K = INSTR(A$,B$)
INT (L)	Restituye el entero <= L	I = INT (87,3)
LEFT$ (A$,N)	Extrae de A$ N caracteres a partir de la izquierda	B$ = LEFT$ (A$,4)
LEN (A$)	Restituye la longitud de la cadena A$	K = LEN (A$)
LOC (N)	Restituye el número del último record leído o escrito en un file directo. Para los secuenciales, restituye el número de sectores	N = LOC (1)
LOG (K)	Calcula el logaritmo natural de K	R = LOG (7.21)
LPOS (X)	Restituye la posición de la cabeza de impresión	N = LPOS (A)
MID$ (A$,I,N)	Extrae de A$ N caracteres a partir de la posición I	B$ = MID$ (A$,5,3)
MKI$ (V)	Convierte V en una cadena de 2 bytes	A$ = MKI$ (2)
MKS$ (V)	Convierte V en una cadena de 4 bytes	A$ = MKS$ (71.5)
MKD$ (V)	Convierte V en una cadena de 8 bytes	A$ = MKD$ (1521.76)
OCT$ (N)	Restituye una cadena que representa la traducción en octal del número N	A$ = OCT$ (12)
PEEK (A)	Restituye el byte leído en la posición de memoria	PEEK (&H4701)
POS (N)	Restituye la posición del cursor (vídeo)	L = POS (1)
RIGHT$(A$,N)	Toma N caracteres de la cadena A$ a partir de la derecha	B$ = RIGHT$(A$,7)
RND (K)	Restituye un número aleatorio comprendido entre 0 y 1	R = RND (3)
SGN (A)	Restituye –1,0,1 según el signo de A	S = SGN (–3)
SIN (A)	Calcula el seno del ángulo A (expresado en radianes)	S = SIN (3.14)
SPACE$ (N)	Crea una cadena de N espacios en blanco	A$ = SPACE$ (3)
SPC (I)	Envía I espacios en blanco al terminal	PRINT "A",SPC(7)
SQR (V)	Calcula la raíz cuadrada de V	A = SQR (7)
STR$ (N)	Convierte en cadena el número N	A$ = STR$ (3)
STRING$(N,A$)	Crea una cadena de N caracteres iguales al primer carácter de la cadena A$	B$ = STRING$ (5,A$)
TAB (N)	Posiciona el cursor en la columna N	PRINT "X"; TAB (6)
TAN (A)	Calcula la tangente del ángulo A (expresado en radianes)	Y = TAN (7)
USR (K)	Llama una subrutina en Assembler transfiriendo el argumento K	A = USR1 (X)
VAL (A$)	Restituye el valor numérico de la cadena A$	N = VAL (A$)
VARPTR (A)	Restituye la dirección en la que está memorizada la variable especificada	VARPTR (X)

Los dialectos del Basic

Del lenguaje Basic existen numerosas variantes, algunas muy similares entre sí, otras notablemente diferentes.
En general, las instrucciones y las funciones del Basic 80 se conservan en casi todas las demás versiones, aunque en algunos casos tienen una sintaxis o una forma lexical diferente. Las diferencias más notables corresponden a las instrucciones que van delante de las siguientes funciones:

— Gestión de las cadenas
— Gestión de los archivos en disco
— Gestión de las funciones I/O
— Gestión de los periféricos
— Transmisión y recepción de datos

Gestión de las cadenas

Las cadenas pueden contener un número de caracteres comprendidos entre 0 (cadena nula) y el máximo previsto por la particular versión del Basic. El espacio de memoria reservado para éste varía dinámicamente en función de los caracteres contenidos; para conocer la longitud efectiva de una cadena, el usuario debe utilizar la función LEN. En algunas formas del Basic existe la posibilidad de asignar a cada cadena una longitud máxima (en número de caracteres) que nunca será superada. La asignación puede utilizar la habitual instrucción DIM o la ALLOCATE. Esta última, disponible solamente en máquinas más grandes, permite reservar un área de memoria sólo momentáneamente. Por ejemplo, supongamos que una subrutina necesita la memorización de 1000 cadenas. La instrucción DIM A$(1000) asigna permanentemente un área de memoria a las 1000 cadenas A$; si estas últimas no se utilizan en otras zonas del programa, se produce un notable desperdicio de memoria. Utilizando la instrucción

ALLOCATE A$(1000)

el área de memoria necesaria por las cadenas A$ se reserva sólo temporalmente. Al final de la fase de utilización de las cadenas, la memoria reservada puede liberarse y, por tanto, reutilizarse para otras finalidades. La instrucción para la liberación de una zona de memoria anteriormente reservada gracias a la instrucción ALLOCATE es la siguiente:

DEALLOCATE A$(1000)

Las tres instrucciones que reservan áreas de memoria (DIM, COM, ALLOCATE) también pueden definir la longitud máxima de cada cadena, especificando su número de caracteres entre corchetes. Por ejemplo:

DIM A$[5]	Asigna a la cadena A$ la longitud de 5 caracteres
COM A$[5]	Similar a la precedente, trabaja en COMMON, es decir, en un área de memoria común a todos los subprogramas (DIM sólo tiene un uso local)
ALLOCATE A$[5]	Tiene el mismo efecto que la anterior, pero el área de memoria puede liberarse sucesivamente y asignarse a otras variables
DIM B$(7)[20]	Reserva un área de memoria para 7 cadenas de 20 caracteres cada una

Normalmente, los sistemas que utilizan esta simbología tienen otras dos características muy importantes:

— la longitud máxima de una cadena puede llegar hasta 32767 caracteres
— existen las subcadenas

En el Basic 80, para extraer parte de una determinada cadena, se emplean las funciones LEFT$, RIGHT$ y MID$, mientras que en estas variantes del Basic, la extracción es posible simplemente indicando entre corchetes los caracteres que deben tomarse.
Por ejemplo, sea A$ = "Esta es una prueba". Para extraer la palabra "prueba", la instrucción que debe utilizarse en el Basic 80 es

B$ = RIGHT$(A$,5)

mientras que en la otra forma puede utilizarse:

B$ = A$[15]

La instrucción toma de la cadena A$ todos los caracteres a partir del que está en la posición 15 (atención con los espacios) hasta el final de la cadena. La cadena B$, tomada de la A$, se dice que es una **subcadena** de A$.
De estas instrucciones existe también la forma

Basic 80	Otras formas	Función realizada
B$ = LEFT$(A$,3)	B$ = A$[1,3]	Extrae los primeros 3 caracteres de la izquierda
B$ = RIGHT$(A$,2)		Extrae los últimos 2 caracteres de la derecha
B$ = MID$(A$,4,1)	B$ = A$[4;1]	Extrae el carácter que está en la posición 4
DIM B$(3)	DIM B$(3)	Reserva un área de memoria para 3 cadenas
no prevista	DIM B$(3)[20]	Reserva un área de memoria para 3 cadenas, cada una de una longitud máxima igual a 20 caracteres
no prevista	ALLOCATE B$(3)	Reserva un área de memoria de manera provisional
no prevista en esta forma; una instrucción análoga es ERASE	DEALLOCATE B$(3)	Libera el área de memoria reservada por la ALLOCATE

que permite elegir el carácter en que empieza la transferencia y el carácter en el que termina. Por ejemplo, la línea

B$ = A$[3,8]

transfiere a B$ el contenido de A$ a partir del carácter número 3 hasta el número 8. Una variante de la misma instrucción es la siguiente:

B$ = A$[3;6]

que transfiere a B$ el contenido de A$ a partir del carácter 3 y para 6 caracteres. Con los números del ejemplo, las 2 instrucciones tienen un efecto idéntico: extraer los caracteres cuyas posiciones van desde la 3 hasta la 8 equivale a extraer 6 caracteres a partir del carácter que ocupa la tercera posición. En la tabla de arriba se han recopilado las principales diferencias que se tienen en la gestión de las cadenas entre el Basic 80 y las formas que prevén las subcadenas.
Las instrucciones ALLOCATE y DEALLOCATE tienen un empleo general y pueden utilizarse también para una matriz numérica (en el Basic 80 la instrucción análoga a la DEALLOCATE es ERASE). Además de estas diferencias sustanciales pueden existir algunas variantes formales; por ejemplo, la función ASC(A$) de algunas versiones del Basic se convierte en VAL(A$), la STRING$(n) se convierte en VAL$(n). Normalmente se trata sólo de diferencias formales: las funciones realizadas son las mismas.

Gestión de los archivos en disco

Las instrucciones de entrada y de salida que difieren mayormente de una máquina a la otra son las inherentes a los discos.
Si bien, para las diferentes máquinas, la lógica permanece aproximadamente sin variaciones, la forma de las instrucciones puede ser notablemente diferente.
A título de ejemplo, indicamos a continuación los principales comandos para la gestión del disco soportados por una versión del DOS, un sistema operativo ampliamente difundido.

CATALOG	Presenta en pantalla la relación de los files contenidos en el disco; es la instrucción análoga a la DIR bajo CP/M
LOAD, SAVE	Tienen las mismas funciones que se han descrito para el CP/M
INIT	Inicializa el disco. Es la instrucción homóloga a la FORMAT, pero presenta algunas caracte-

rísticas particulares. Durante la inicialización, es necesario memorizar en disco un file que contiene un programa que se envía a ejecución con cada carga de diskette. El programa debe ser residente en memoria en el momento de la inicialización, y se transfiere al disco automáticamente. La sintaxis es

INIT NOMBRE, Sn, Dm

donde NOMBRE es el nombre del file de inicialización (en el cual se memorizará el programa);
n es el número de vía de acceso a la que está conectada la unidad de disco que se usa*;
m es el número de la unidad de disco (normalmente cada vía de acceso puede aceptar dos).
También existe otro parámetro opcional (número del volumen) que puede utilizarse como identificador asociado al diskette. Por ejemplo, la instrucción

INIT PRUEBA,S6,D2,Vp

inicializa un diskette colocado en la unidad 2 (D2) de la vía de acceso 6(S6) y le asigna el número de volumen p

DELETE — Anula un programa; es el equivalente de ERA bajo CP/M

LOCK — Permite proteger un file impidiendo su cancelación. El file puede liberarse con el comando UNLOCK

RENAME — Cambia el nombre a un file (como bajo CP/M)

VERIFY — Permite verificar la integridad de un file. En condiciones particulares accidentales, el contenido de un file puede resultar parcialmente destruido; este comando verifica su integridad. El método es similar al empleado en las transmisiones con bit de paridad. Al principio de cada record se registra un valor (llamado check-sum) función del contenido del record. Si el contenido se modifica por cualquier causa accidental, la check-sum no se actualiza y se crea así una desavenencia que la pone de relieve el sistema operativo

* Ver la pág. 759.

EJEMPLO DE GESTION DE LOS FILES SECUENCIALES (DOS)

```
90  REM : PROGRAMA PROG1
100  REM : FUNCIONES I/O DISCO
120  REM : CADENA DE COMANDO
140 A$ = CHR$ (4)
160  REM :
180  REM : APERTURA
200  PRINT A$; "OPEN PRUEBA"
220  REM :
260  INPUT " CADENA: ";B$
280  REM : ESCRITURA
300  PRINT A$;"WRITE PRUEBA"
320  PRINT B$
325  PRINT A$;"CLOSE PRUEBA"
340  REM  :
360  REM  : LECTURA
365  PRINT A$;"OPEN PRUEBA"
380  PRINT A$;"READ PRUEBA"
400  INPUT D$
420  PRINT A$
440  PRINT "DATOS LEIDOS :"
460  PRINT D$
480  END
```


K. Reese/Marka

Empleo del trazador gráfico (plotter) para la representación gráfica del rendimiento de los pozos petrolíferos.

Los tipos de files utilizados en el DOS son los mismos previstos en el CP/M (secuenciales y directos), mientras que la técnica utilizada para las funciones de lectura y escritura es notablemente diferente.
En el DOS, las operaciones de I/O se realizan con las instrucciones READ y WRITE puestas en forma de cadena y precedidas por el código CTRL + D [teclas CONTROL y D pulsadas al mismo tiempo = CHR$(4)].
Por ejemplo, una instrucción de lectura M puede tener la siguiente forma:

```
PRINT CHR$(4); "READ XYZ"
INPUT V
```

La primera instrucción activa el mecanismo de lectura (XYZ es el nombre del file), mientras que la segunda adquiere un valor y lo transfiere a la variable V. También para esta variante del Basic, los files pueden ser secuenciales o directos.
Aquí al lado se ha representado el listado de un programa de ejemplo que ilustra la técnica I/O sobre file secuencial. El programa está montado sobre el ordenador personal SIPREL mod. 2040S, pero puede utilizarse también en el Apple o en otras máquinas compatibles. La línea 140 asigna a la variable A$ el valor CHR$(4) (CTRL + D) que deberá enviarse antes de cada cadena de comando enviada al disco. La apertura del file (línea 200) se obtiene enviando la cadena de control A$ seguida del comando OPEN PRUEBA (PRUEBA es el nombre del file). Para la escritura de los datos debe enviarse el comando WRITE (línea 300) seguido de la instrucción PRINT (línea 320) con las variables que se desee transferir. Como puede observarse, las funciones I/O en el disco se activan con las mismas instrucciones que las I/O en el vídeo

OPERACIONES DE I/O SOBRE FILES SECUENCIALES CON VALORES NUMERICOS

```
10  REM : PROGRAMA PROG2
20  REM : FILES SECUENCIALES
30  REM
100  REM : DATOS NUMERICOS
110  DIM V(30)
120  INPUT "NUMERO DE DATOS ";N
140  PRINT "INTRODUCIR LOS VALORES"
160  FOR I = 1 TO N
180  PRINT " DATO NUMERO : ";I
200  INPUT V(I)
210  NEXT I
220  REM  :
240 A$ = CHR$ (4)
260  PRINT A$;"OPEN PRUEBA2"
280  PRINT A$;"WRITE PRUEBA2"
300  FOR I = 1 TO N
320  PRINT V(I)
340  NEXT I
360  REM :
380  PRINT A$;"CLOSE PRUEBA2"
400  PRINT A$;"OPEN PRUEBA2"
420  PRINT A$;"READ PRUEBA2"
440  FOR I = 1 TO N
460  INPUT V(I)
480  NEXT I
500  PRINT A$;"CLOSE PRUEBA2"
520  PRINT A$
540  PRINT " ** DATOS LEIDOS **"
560  FOR I = 1 TO N
580  PRINT " DATO N.:";I,V(I)
600  NEXT I
620  END
```

(INPUT, PRINT) precedidas por los comandos DOS que especifican el file. En las líneas 360 a 460 se ha presentado un ejemplo de lectura. La sucesión de las instrucciones a enviar es esta:

— Apertura del file (línea 365)
— Comando de lectura (línea 380)
— Lectura de datos (línea 400)
— Liberación del disco (línea 420)
— Escritura de datos en pantalla (líneas 440 y 460)

La línea 420 envía la única cadena A$ = CHR$(4) y anula todas las predisposiciones a los comandos DOS, permitiendo así la impresión de los datos en la unidad del sistema (vídeo). Esta instrucción es necesaria para informar al sistema que se desea cambiar la unidad de I/O. Compárense por ejemplo las líneas 320 y 420: son idénticas, pero una se refiere al disco (y esto se ha especificado en la línea 300 que la precede), y la otra al vídeo.
Arriba se ha representado el listado de un programa para la escritura y la sucesiva lectura de los datos numéricos en un file secuencial. Las instrucciones de escritura se encuentran en las líneas 260 a 340. Después de la apertura del file (línea 260) y el envío del comando de escritura (línea 280), el bucle (líneas 300 a 340) transfiere los valores al disco (N se ha introducido por teclado en la línea 120).
Este modo de trabajar, característico de la familia de máquinas examinada, se debe al hecho de que el sistema operativo utiliza como separador de los campos sólo el carácter CR [retorno de carro = CHR$(13)]; por tanto, para separar un valor numérico de otro, debe insertarse un carácter CR entre los dos valores.
En el ejemplo, la inserción se obtiene de forma automática aprovechando la instrucción PRINT. Como alternativa, el carácter CHR$(13) puede ser enviado por programa. Así, para transferir las variables X, Y, Z, el programa

```
A$ = CHR$(4)          Código de control
B$ = CHR$(13)         Retorno de carro
```



```
PRINT A$;"OPEN PRUEBA"
PRINT A$;"WRITE PRUEBA"
PRINT X;B$;Y;B$;Z
PRINT A$
```

es equivalente al siguiente

```
A$  CHR$(4)
PRINT A$;"OPEN PRUEBA"
PRINT A$;"WRITE PRUEBA"
PRINT X
PRINT Y
PRINT Z
PRINT A$
```

En este caso, el código CR lo inserta el sistema al final de cada instrucción PRINT.
En el sistema operativo DOS se han previsto

AMPLIACION Y RELECTURA DE UN FILE SECUENCIAL

```
10  REM : PROGRAMA PROG4
20  REM : FILES SECUENCIALES
30  REM : COMANDO APPEND
35  HOME
100 A$ = CHR$ (4)
105  ONERR GOTO 1000
120  PRINT A$;"APPEND PRUEBA"
160  PRINT "INTRODUCIR LOS DATOS"
180  PRINT "FIN PARA TERMINAR"
190 K = 0
200  INPUT B$
220  IF B$ = "FIN" GOTO 320
240  PRINT A$;"WRITE PRUEBA"
260  PRINT B$
270  PRINT A$
280  PRINT "DATO TRANSFERIDO"
290 K = K + 1
300  GOTO 200
320  REM : FIN INTRODUCCION
340  PRINT A$;"CLOSE PRUEBA"
360  PRINT A$
365  IF K = 0 GOTO 960
380  PRINT "SE HAN AÑADIDO:
     "
400  PRINT K;"  RECORDS"
410  PRINT
420  PRINT "RELECTURA DATOS:"
430  PRINT A$;"OPEN PRUEBA"
680  INPUT "Cuántos son los records
      a leer ?";NF
700  INPUT "A partir de qué po-
     sicion ? ";P%
760  PRINT A$;"OPEN PRUEBA"
770  PRINT A$;"POSITION PRUEBA,R";
     P%
780  FOR I = 0 TO NF - 1
800  PRINT A$;"READ PRUEBA"
820  INPUT D$
840  PRINT A$
845 IN% = I + P%
860  PRINT "DATO N.: ";IN%
880  PRINT D$
900  NEXT ;
920  PRINT A$;"CLOSE PRUEBA"
940  PRINT A$
960  END
1000  REM  :  ** ERROR **
1020  HTAB 10: VTAB 24: INVERSE
1030  PRINT " ** FIN FILE **"
1055  NORMAL
1060  PRINT " END "
1080  GOTO 960
```

dos comandos que no encuentran correspondencia en el CP/M

APPEND
POSITION

El comando APPEND permite añadir records a un file secuencial; bajo CP/M debería reescribirse todo el file con, al final, los records añadidos.
El comando POSITION permite desplazar el puntero del file un cierto número de campos.
El desplazamiento sólo se produce hacia delante: el puntero no puede posicionarse en campos anteriores al examinado.
El reconocimiento de cada campo se efectúa sobre el carácter de retorno de carro.
En la pág. 751 se ha representado un programa de ejemplo para la ampliación de un file secuencial (comando APPEND) y para su relectura total o a partir de una determinada posición (comando POSITION).
En la lectura de files secuenciales de los que no se conoce previamente la longitud, puede suceder que se intente el acceso a un record inexistente. En este caso, el sistema emite un diagnóstico (END OF DATA) y detiene la ejecución.
La situación de bloqueo del programa puede evitarse utilizando la instrucción ONERR GOTO nn (similar a la instrucción homóloga bajo CP/M), que al producirse un error transfiere el control a la línea especificada (nn).
El programador puede determinar la causa del error (controlando en memoria el código de error generado) y tomar las oportunas medidas.
Abajo aparece un programa de ejemplo del uso de la instrucción ONERR GOTO...

EJEMPLO DE USO DE LA INSTRUCCION ONERR (DOS)

```
10  REM : PROGRAMA PROG5
20  REM :
30  REM : USO DE ONERR GOTO
40  REM :
50  ONERR  GOTO 1000
60  REM :
80 A$ = CHR$ (4)
100  PRINT A$;"OPEN PRUEBA"
120  FOR I = 1 TO 100
140  PRINT A$;"READ PRUEBA"
160  INPUT D$
180  PRINT A$
200  PRINT"DATO N. :";I
220  PRINT D$
240  NEXT I
260  PRINT A$
280  PRINT "** RUN END **"
300  END
1000  REM : RECUPERACION ERRORES
1020  REM
1040  REM : LOCALIZACION 222
1060  REM :
1080  REM : CODIGOS RECONOCIDOS
1100  REM : 5 = FIN DATOS
1120  REM : 9 = DISCO COMPLETO
1140  REM:
1160 T = PEEK (222)
1180  IF T < > 5 GOTO 1300
1200  REM : FIN DATOS
1220  PRINT A$
1225  HOME
1240  PRINT "DETECTADO EL FINAL FILE
      E"
1260 K = 1
1280  PRINT A$;"CLOSE PRUEBA"
1285  INVERSE
1300  PRINT "ERROR : ";T
1320  PRINT "NO RECUPERABLE"
1325  NORMAL
1340  STOP
```


USO DE LA INSTRUCCION GET

```
10  REM : PROGRAMA PROG6
20  REM :
30 A$ = CHR$ (4)
100  PRINT A$;"OPEN PRUEBA"
120  PRINT A$;"READ PRUEBA"
140  GET D$
160  PRINT A$
182  PRINT "Carácter leído:";
184  INVERSE : PRINT " ";D$;" ":NORMAL
200  PRINT "CONTINUA ? (S/N)"
210  PRINT A$
220  GET R$
225  PRINT A$
240  IF R$ ="S" GOTO 120
260  PRINT "** RUN END **
280  END
```

El código correspondiente al error generado se lee con la instrucción PEEK (222).
Esta posición de memoria sólo es válida para la máquina utilizada (SIPREL 2040S); para otras máquinas podría ser diferente. Controlando el valor numérico contenido en esta posición de memoria (línea 1160 y 1180) puede reconocerse el tipo de error.
En los sistemas (compatibles Apple) que utilizan el DOS existe una notable diferencia con respecto al CP/M que corresponde al uso de la instrucción GET, utilizada para leer un solo byte cada vez (en CP/M adquiere todo el record).
Arriba se ha representado el listado de un programa que lee un carácter, lo presenta en pantalla y espera la introducción de la autorización para proseguir (carácter S).
La instrucción GET se utiliza en los dos modos posibles: en la línea 140 adquiere un carácter del file del disco, mientras que en la línea 220, el carácter se lee por teclado.
En el sistema operativo DOS, el tratamiento de los files directos es análogo al de los secuenciales; la única diferencia consiste en la posibilidad de acceder directamente a un determinado número de record, cuyo valor debe especificarse en los comandos.
Por ejemplo, las líneas

```
A$ = CHR$(4)
PRINT A$; "OPEN NOMBRE,L"
.....
PRINT A$; "WRITE NOMBRE,R"
.....
PRINT A$; "READ NOMBRE,R"
```

realizan las siguientes funciones que a continuación se detallan:

— Apertura del file NOMBRE con records de longitudes L (en bytes)
— Escritura en el file NOMBRE del record R
— Lectura del file NOMBRE del record R

El listado de un programa para la gestión de files directos se indica en la pág. 754.
Otra notable característica del DOS es la posibilidad de direccionar el único byte al interior de un campo simplemente especificando su número en el comando READ.
Por ejemplo, la instrucción

PRINT A$; "READ NOMBRE,7,2"

adquiere el byte número 2 del record 7 del file NOMBRE.
En la pág. 755 se han representado el esquema lógico y los niveles de detalle que pueden obtenerse con los diversos comandos para los files.

Gestión de las funciones I/O

Las diversas versiones del Basic presentan las mayores diferencias en las instrucciones que gobiernan las funciones I/O. La gestión de las funciones de comunicación entre la máquina y el mundo exterior puede asumir aspectos muy complejos que dependen del tipo de periférico interesado, puesto que está extraordinariamente condicionada por el hardware.
Las comunicaciones entre el calculador y el vídeo normalmente se realizan en la modalidad RS 232, y su gestión tiene aspectos que son

PROGRAMA DE EJEMPLO DE ACCESO A FILES DIRECTOS

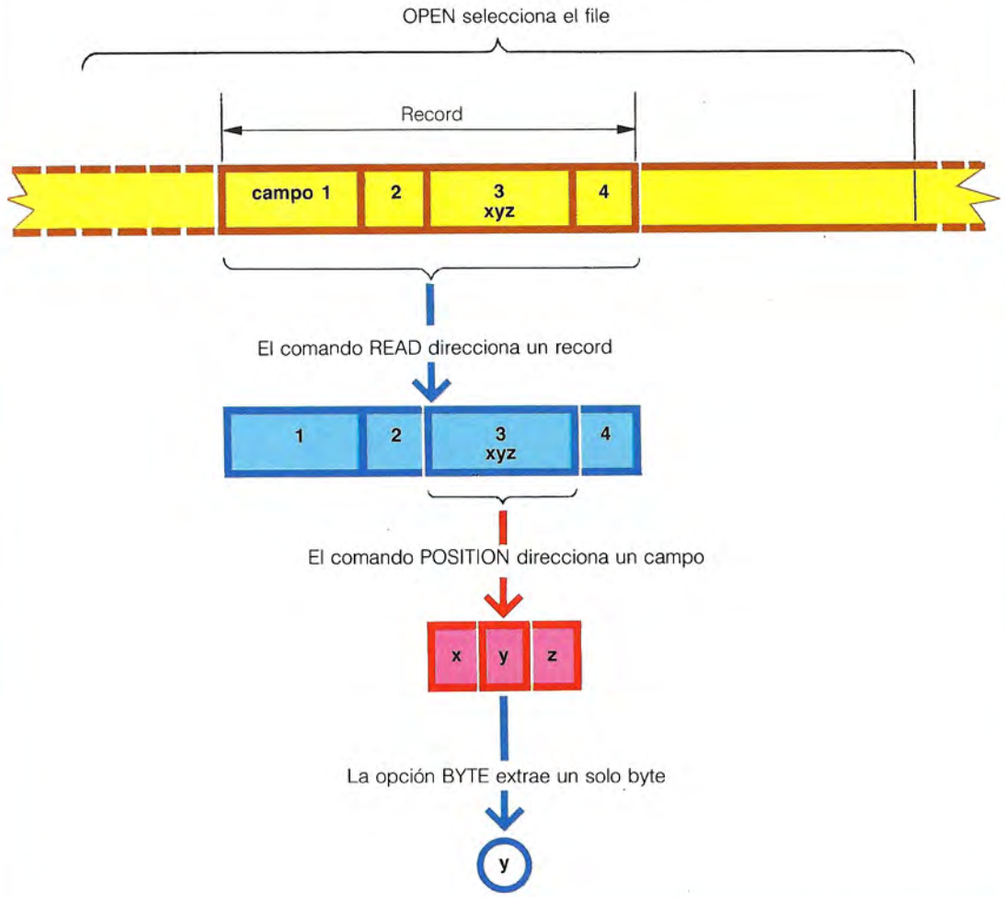
```
10  REM : PROGRAMA PROG3
20  REM : FILES DIRECTOS
30  REM :
40  REM : PARAMETROS
50  REM F$=longitud en bytes
60  REM NR=número del record
100 A$ = CHR$ (4)
120  REM : longitud record = 40

125  REM :Más 1 terminador
130 F$ = "41"
140  PRINT A$;"OPEN PRUEBA3." + "L
     " + F$
160  REM : introducción de datos
180  REM :
200  INPUT "Número de datos? ";N
210  IF N = 0 GOTO 710
220  FOR I = 1 TO N
225 B$ = "":C$ =""
240  PRINT " Dato N. :";I
260  INPUT " Valor (CADENA) ";B$

270  REM : CONTROl  LONGITUD
280 K = LEN (B$)
290  IF K < 40 GOTO 400
295  IF K = 40 THEN C$ = B$
300  IF K = 40 GOTO 500
320  REM : Cadena de longitud
340  REM superior al record
360 C$ = LEFT$ (B$,40)
380  GOTO 500
400  REM : Cadena de longitud
420  REM   inferior al record
440 M = 40 - K
442 C$ = B$
445  FOR J = 1 TO M
460 C$ = C$ +","
470  NEXT J
500  REM : Cadena de longitud
520  REM   igual al record
540  REM : ninguna variación
600  REM :
620  INPUT " Número del Record? ";
     NR
640  PRINT A$;"WRITE PRUEBA3.R";NR

660 PRINT C$
685 PRINT A$
700 NEXT I
710 PRINT A$;"CLOSE PRUEBA3"
740 INPUT "Se quiere leer?";F
    S$
760 IF RS$ < > "SI" GOTO 960
780 INPUT "Qué record?";NR
800 PRINT A$;"OPEN PRUEBA3." + "L
    " + F$
820 PRINT A$; "READ PRUEBA3,R";NR
840 INPUT D$
860 PRINT A$; "CLOSE PRUEBA3"
870 PRINT A$
880 PRINT "DATOS LEIDOS :"
900 PRINT D$
920 INPUT "CONTINUA? (SI/NO) ";R
    S$
940 IF RS$ = "SI" GOTO 780
960 END
```


NIVELES DE EXPLORACION DE UN FILE



prácticamente idénticos para los diversos tipos de máquina.
Para las unidades de disco, o para otros dispositivos, en cambio, la gestión está ligada al hardware y, por tanto, se realiza a través de instrucciones y comandos que son específicos de cada máquina.
En la pág. 756 se ha representado un esquema que resume las principales formas de adaptación con los periféricos. Todos los eventuales interfaces son susceptibles de subdividirse en dos grupos, en función del tipo de software y de lo que necesitan:

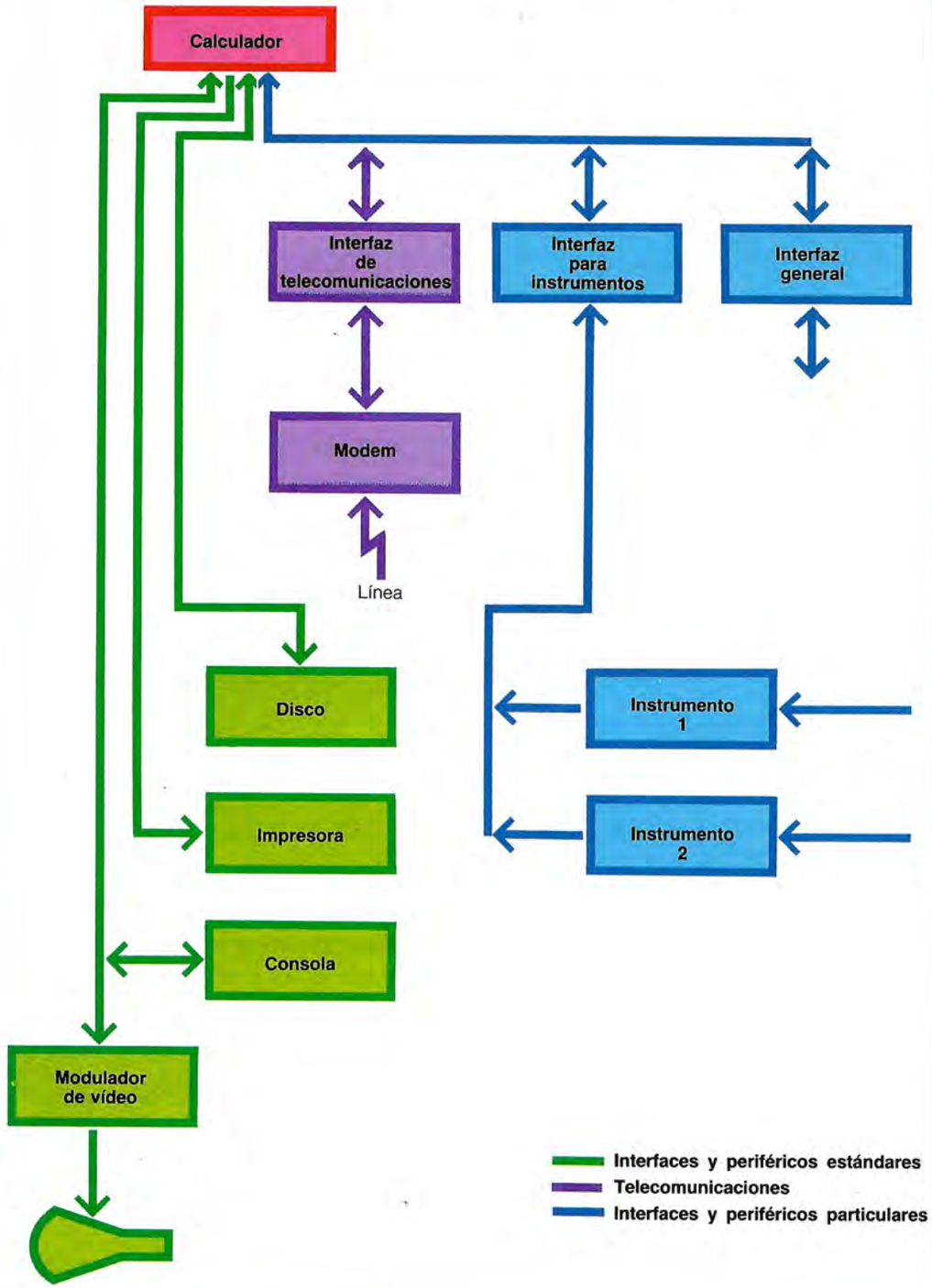
— Interfaces gestionados por el sistema operativo
— Interfaces programables

Al primer tipo pertenecen los interfaces hacia los aparatos de uso normal (vídeo, impresora, discos, etc.) para los cuales no es necesario desarrollar subrutinas particulares, puesto que pueden controlarse utilizando las instrucciones I/O previstas en el lenguaje de programación. Al segundo tipo pertenecen los interfaces que debe gestionar el usuario, para los cuales es necesario preparar adecuadas rutinas de gestión en simbólico (Basic) para las máquinas más evolucionadas o en Assembler para las otras.

Gestión de los periféricos: la adaptación

Los circuitos electrónicos que componen un calculador no son adecuados para controlar aparatos exteriores a causa de la reducidísima

ESQUEMA DE ADAPTACION CON LOS PERIFERICOS



potencia de las señales que elaboran. Por tanto, entre el ordenador y sus periféricos es necesario interponer determinados circuitos que puedan adaptar las señales haciéndolas compatibles con la electrónica del ordenador por un lado y con la estructura de los periféricos por el otro. Estos circuitos se denominan normalmente **interfaces**.

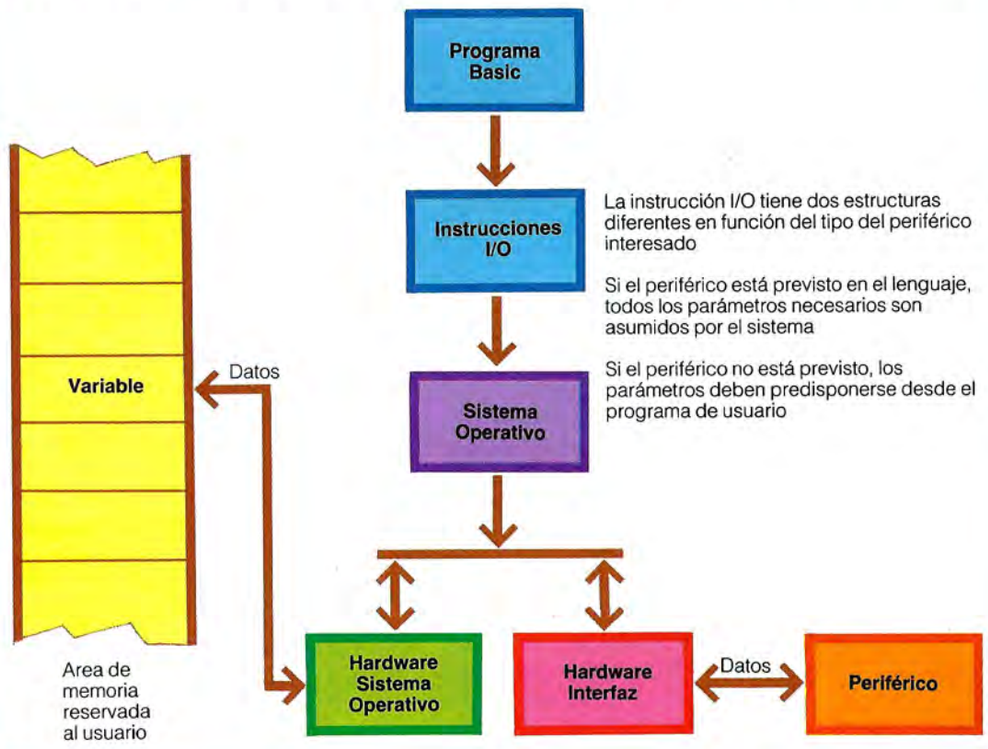
Las principales funciones realizadas por un interfaz pueden relacionarse de la siguiente manera:

— Adaptación de los niveles de la señal
— Adecuación del formato de datos
— Temporización de las señales

En muchas aplicaciones, los niveles de la señal necesarios para los periféricos no son los mismos que los emitidos por el ordenador (los niveles eléctricos a los que corresponden los estados lógicos 1 y 0 no se corresponden). En estos casos, el interfaz debe asumir la tarea de adaptar las señales haciendo de manera que cada unidad sea controlada con los niveles adecuados. Además, normalmente el ordenador envía las señales en transmisión paralela (los bus están en paralelo), mientras que algunos periféricos necesitan entradas en serie. Las transformaciones paralelo-serie también corren a cargo del interfaz . Finalmente puede producirse que los periféricos trabajen con tiempos muy diferentes a los típicos del ordenador; también en este caso debe ser el interfaz el que regule adecuadamente el flujo de los datos. En algunos casos, los problemas de tiempo son superados por la técnica «handshake», que permite la transmisión de un dato cada vez y esperar, para el envío del siguiente, que la unidad receptora haya adquirido el primero y comunicado la autorización para un nuevo envío.

Abajo se ha representado el esquema lógico de un intercambio de datos. El programa de aplicación podrá tener dos formas diferentes en función del tipo de periférico que debe controlarse. En el caso en que los periféricos vuelvan a entrar entre los previstos en el lenguaje simbólico,

ESQUEMA FUNCIONAL DEL INTERCAMBIO DE DATOS CON UN PERIFERICO



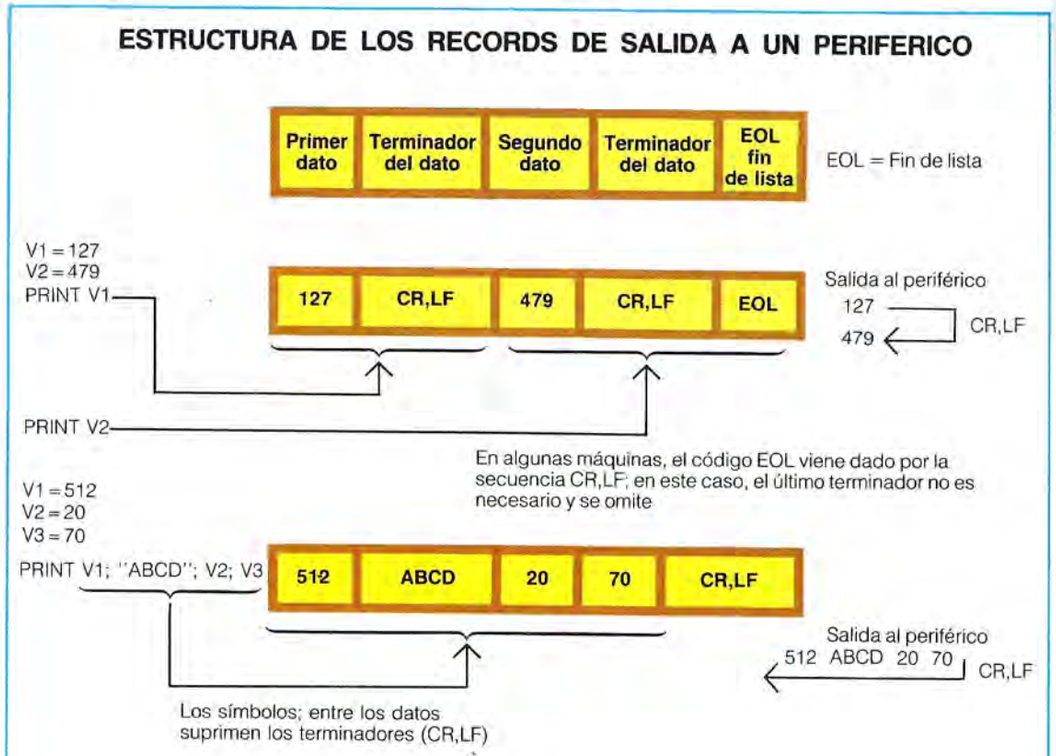
todos los parámetros necesarios son predispuestos por el sistema operativo; el usuario sólo debe llamar la rutina adecuada utilizando una o más instrucciones de alto nivel (por ejemplo INPUT, PRINT, PRINT USING, etc.). Cada una de estas instrucciones se decodifica y activa una serie de rutinas del sistema que permiten la gestión de los periféricos.
En el caso en que los periféricos no estén entre los previstos, el usuario deberá escribir las rutinas necesarias para su gestión.
La estructura de esas rutinas depende estrechamente del hardware: en algunas máquinas existen interfaces que pueden controlarse con determinadas instrucciones Basic y, en otras, es necesario utilizar el lenguaje Assembler.
Abajo se han representado dos ejemplos de salida hacia periféricos (por ejemplo vídeo e impresora). El proceso es activado con la única instrucción PRINT, y es el sistema el que debe completar con los códigos necesarios (por ejemplo, CR = retorno de carro y LF = nueva línea) o el que inserte los oportunos espacios en blanco en caso de que se imponga un formato. En pág. 759 se han representado dos ejemplos de instrucciones PRINT USING que utilizan la simbología del Basic 80 (#). En otras versiones del Basic existen simbologías más complejas; las principales opciones de formato son las siguientes:

A Indica un formato ASCII:USING "4A" significa 4 caracteres ASCII.
D Digit (= cifra); cada letra D indica una cifra decimal (de 0 a 9). Las cifras pueden ir precedidas del signo; así, la opción

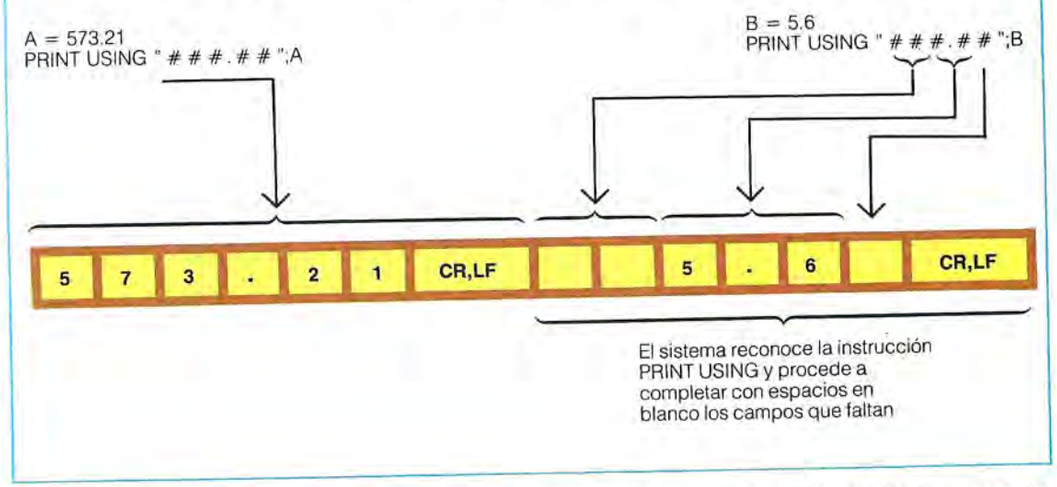
USING "SDD.D"

especifica un valor numérico, compuesto por dos cifras enteras y una decimal, precedido por el signo (S)
E Exponente; el número se imprime en notación exponencial

ESTRUCTURA DE LOS RECORDS DE SALIDA A UN PERIFERICO



ESTRUCTURA DE LOS RECORDS EN UNA SALIDA CON FORMATO



K Especifica el formato compacto; se eliminan todos los espacios en blanco
X Especifica que deben dejarse uno o más espacios en blanco. La opción

USING "SD.DD,5X,SDD.D"

escribe el primer número (SD.DD), salta cinco posiciones (5X) y escribe el segundo valor numérico (SDD.D).

En los ejemplos se ha omitido expresamente la instrucción que expresa la acción que debe realizarse (escritura), que para algunas máquinas puede ser el código habitual PRINT, mientras que para otras puede indicarse con un código diferente, como por ejemplo OUTPUT. En este último caso, una instrucción completa puede asumir la siguiente forma:

```
R  – 51.6
OUTPUT 3 USING "SDD.D";R
```

que genera un record sobre el periférico 3 en el formato de signo (S), dos enteros (DD), punto (.), un decimal (D).
Finalmente, existe una categoría de máquinas para las cuales el periférico estándar no es diferente en base a las funciones, sino a la **vía de acceso** (slot) a la que está conectada.
Vía de acceso es el término con que se indican los conectores que permiten acoplar al ordenador los dispositivos auxiliares, que incluyen no solamente los periféricos, sino también las eventuales expansiones de memoria u otros dispositivos.
En general, se trata de un alojamiento conectado eléctricamente al bus del ordenador.
En estos casos, el sistema utiliza normalmente el vídeo y el teclado como periféricos de I/O. Para direccionar una salida, por ejemplo sobre impresora, basta con activar, con las adecuadas instrucciones, la correspondiente vía de acceso; la instrucción de salida continúa siendo PRINT, pero está precedida por un código que especifica la activación del interfaz de la impresora. Este modo de gestionar los periféricos es muy utilizado en los ordenadores personales y presenta algunas ventajas, entre las cuales hay la posibilidad de configurar el sistema según las propias necesidades. Para incluir un nuevo periférico basta con insertar en una vía de acceso la tarjeta de interfaz; en el programa de aplicación, después podrán direccionarse las funciones hacia aquel número de vía de acceso.
Otra notable ventaja viene dada por la posibilidad de incluir en el sistema un segundo microprocesador.
El sistema operativo está estrechamente vinculado al microprocesador montado.
Por ejemplo, el CP/M, en sus diversas versiones, puede utilizarse con la familia de microprocesadores Z80 (8080, 8086, etc.); otros sistemas, como el DOS, gestionan otros tipos de microprocesador.
Utilizando la arquitectura descrita, en una má-

quina puede incluirse una segunda CPU que permite trabajar con un segundo sistema operativo como alternativa al preexistente.
En estos casos, la CPU original se dedica solamente a tareas secundarias, y el ordenador puede utilizarse como si perteneciese a la otra familia. Esta dualidad permite elegir el sistema operativo más adecuado a las aplicaciones particulares, o en función del tipo de software de aplicación de que se dispone. A continuación se indican dos peticiones de salida en impresora y en vídeo bajo CP/M y bajo DOS.

```
CP/M
10 A$ = "PRUEBA"
20 LPRINT A$
30 PRINT A$

DOS
10 A$ = "PRUEBA"
20 LPRINT CHR$(4); "PR # 1"
30 PRINT A$
```

La línea 20 sirve para direccionar la salida en impresora; CHR$(4) es el código que activa el comando, mientras que la cadena PR # 1 es el comando.
Los códigos citados son comunes a muchas máquinas que emplean el microprocesador 6502 (APPLE, SIPREL, etc.), los cuales constituyen actualmente la familia más difundida como alternativa a las máquinas gestionadas por el CP/M.
En la gestión del vídeo se encuentran otros aspectos particulares.
En el Basic bajo CP/M normalmente no hay previstas instrucciones específicas de direccionamiento del cursor. Para obtener su posicionamiento en una zona cualquiera de la pantalla deben enviarse algunos códigos, que dependen en secuencia y en valor del tipo de la unidad de vídeo utilizada.
Para la otra familia (microprocesador 6502) hay dos instrucciones Basic de direccionamiento

```
HTAB n para el posicionamiento horizontal
VTAB m para el posicionamiento vertical
```

Un programa que permite posicionar la escritura "Prueba de posicionamiento" puede tener la siguiente forma:

```
10 INPUT "Columna"; C
20 INPUT "Línea"; R
30 A$="Prueba de posicionamiento"
100 HTAB C:VTAB R
110 PRINT A$
120 END
```

Aparte de la línea 100, las restantes instrucciones son idénticas a las utilizadas en el Basic 80.

Transmisión y recepción de datos

La creciente necesidad de disponer de informaciones en tiempo real ha creado un notable interés por los sistemas de elaboración distribuidos, o sea por las redes constituidas por más microprocesadores conectados entre sí o con un ordenador principal. La ventaja principal asociada al uso de un microprocesador en lugar de un terminal normal consiste en la capacidad de elaboración del primero, que permite realizar la doble función de terminal y de unidad de proceso independiente.
El protocolo de transmisión de datos más sencillo es el EIA RS-232-C; por este motivo, muchos fabricantes han dotado su ordenador con interfaces que puedan gestionar este protocolo a través de instrucciones de alto nivel, y que, por tanto, entran a formar parte del lenguaje Basic.
La instrucción principal es la que proporciona los parámetros del protocolo, es decir:

— **Velocidad de transmisión**. Expresa el número de bits transmitidos cada segundo (bps bits por segundo); los valores utilizados son 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600 bps.

— **Paridad**. Especifica el tipo de control de paridad, es decir, la condición en la que recae el bit de paridad. Los casos posibles son:

- no transmitido
- siempre 0
- siempre 1
- transmitido como paridad impar
- transmitido como paridad par

— **Bits de datos**. El dato puede representarse con un número de bits comprendido entre 4 y 8 y, por tanto, debe "fijar" el interfaz con el formato justo.

— **Bits de stop.** Al final del dato puede presentarse o no el bit de stop. Los casos son:

K. Reese/Marka

Interior de una oficina de reservas hoteleras que se sirve de aparatos computerizados.

- ausencia
- 1 bit de stop
- 2 bits de stop

— **Número del file**. Para gestionar un canal de telecomunicación, a éste debe asociarse un número de file, sobre el cual después pueden realizarse las funciones GET y PUT de la misma manera que para un file en disco.
Por ejemplo, la instrucción

```
OPEN "COM 1:2400,S,8,1" AS # 3
```

abre un file de comunicación con el número 3 para el interfaz 1 (COM 1), velocidad de transmisión 2400 bps, sin paridad (el símbolo S), con formato de 8 bits más un bit de stop.
La forma generalizada de la instrucción prevé también los códigos de control de las señales de línea. El protocolo RS-232 es útil y ventajoso (dada su sencillez de uso) cuando deben realizarse conexiones entre dos unidades solas. Para poder dialogar desde un mismo ordenador con diversos periféricos, debe disponerse de más puertas de acceso o de un sistema de selección exterior. Las complicaciones de hardwa-

re que se derivan pueden ser superiores a las ventajas y limitan las posibilidades de aplicación del protocolo RS-232.
El intercambio de datos entre un calculador y más periféricos es una situación que se produce habitualmente en la gestión automática de instrumentaciones, y para este objeto se han realizado determinados protocolos; el más conocido es el HP-IB (Hewlett-Packard Interface Bus), que ha sido propuesto como estándar internacional a la I.E.C. (International Electrotechnical Commission).
La conexión HP-IB prevé 16 cables, de los que 8 se dedican a la transmisión de los datos y los 8 restantes, a las señales de control. Los datos se transmiten en código ASCII de 7 bits (el octavo es el bit de paridad). Las instrumentaciones, o más en general las unidades, que pueden llegar a conectarse se dividen en tres categorías:

LISTENER — Unidad capaz de recibir datos de otras, por ejemplo la impresora, la pantalla, etc.

TALKER — Unidad que puede transmitir datos, como por ejemplo un registrador magnético o cualquier instrumento que, detectando el valor de una magnitud física, lo convierte en una señal eléctrica (digital) y la envía al bus

CONTROLLER — Unidad de control, responsable de la gestión de las comunicaciones entre la otra unidad. Normalmente es el ordenador

Sala de impresoras de un centro de cálculo. Las siglas permiten identificar al usuario.

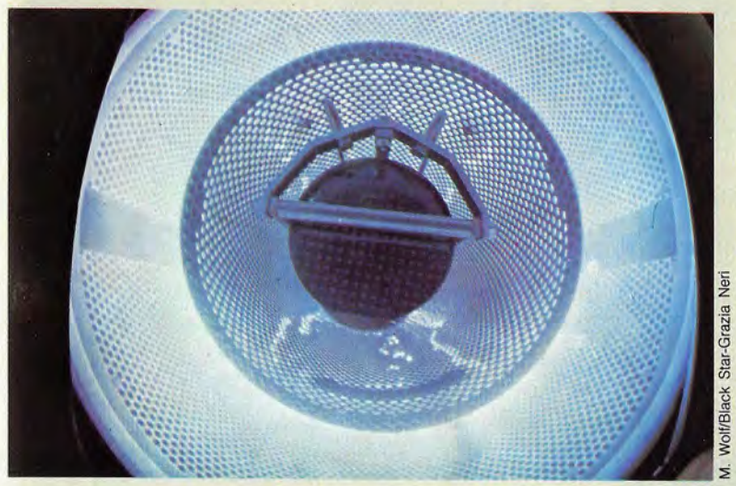
J. Pickerell/Marka

Una unidad cualquiera puede asumir de vez en cuando uno de los tres papeles previstos: un instrumento de medida puede ser LISTENER mientras recibe las instrucciones del ordenador y TALKER cuando transmite los resultados de una medición.
El sistema HP-IB también prevé que en cada instante exista un solo TALKER y un solo CONTROLLER activos; los demás aparatos eventualmente presentes deben ser LISTENER. En algunas versiones del Basic se han previsto instrucciones de trampa para los datos teletransmitidos que, si se activan, interrumpen el desarrollo del programa y pasan el control a una rutina de servicio para la adquisición de los propios datos.
Las instrucciones de este tipo hacen que en el intervalo de tiempo que transcurre entre la llegada de un carácter y la llegada del siguiente carácter, la máquina se dedique a realizar una tarea cualquiera, en espera de que transcurra el tiempo necesario.
En la pág. 767 se ha indicado el diagrama de flujo de principio de la lógica utilizada en la teletransmisión de datos (recepción).
La principal función encargada para la adquisición de un dato en recepción se denomina **interrupt**. El objeto del interrupt es el de activar determinadas rutinas al producirse una intervención exterior. Un ejemplo inmediato es la función realizada bajo el comando RESET (en CP/M se activa por el código CTRL + C), que interrumpe la ejecución de cualquier programa. La máquina, ocupada en desarrollar el programa de aplicación, no interroga al teclado y, de esta manera, no adquiere eventuales comandos; el único modo de interrumpir el proceso es el de utilizar un interrupt. Al producirse esta acción, el trabajo se suspende y se activa la función ligada al interrupt.

Del silicio al ordenador

M. Wolf/Black Star-Grazia Neri

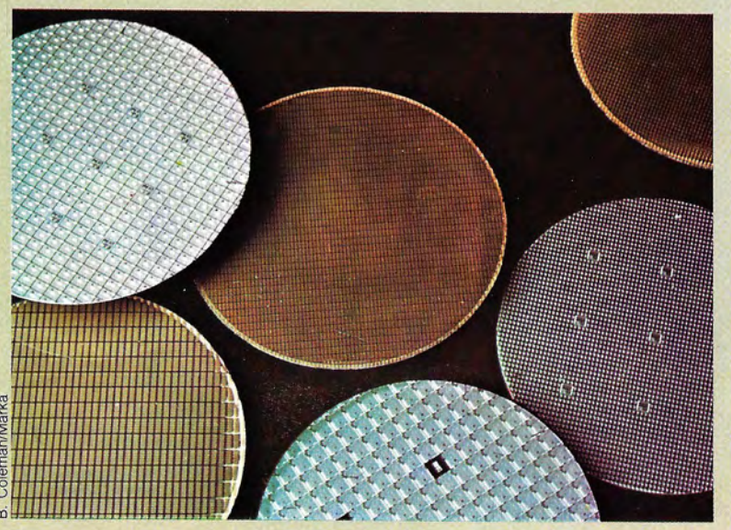
Stock Photos/Grazia Neri



M. Wolf/Black Star-Grazia Neri

El proyecto y el diseño del microcircuito se realizan con el hardware y con el software típicos del CAD (Computer Assisted Design). La fotografía de al lado ilustra la fase del trazado del diseño del circuito con un trazador gráfico de alta resolución. El ordenador memoriza la estructura, que a petición puede presentarse sobre el monitor de un terminal. Así, los proyectistas podrán modificarla hasta hacerla definitiva y de ella obtener, con procedimientos fotográficos, las máscaras necesarias para el proceso fotolitográfico.
De una fusión de silicio purísimo, al cual se añaden impurezas específicas para obtener las características eléctricas deseadas, se extrae lentamente la barra (monocristal) de semiconductor de la que se obtendrán a continuación las obleas.
El crecimiento del monocristal se realiza a partir de una minúscula «semilla» alrededor de la cual se agregan los átomos de silicio, que van a formar una estructura cristalina prácticamente libre de defectos.
La fotografía del centro muestra la extracción de una barra de silicio. A continuación, el monocristal se mecaniza y se lleva a un diámetro uniforme. Cortándolo a rebanadas delgadas como laminitas se obtienen las obleas, que se pulen y bruñen cuidadosamente.
Las diversas capas del microcircuito se imprimen sobre la oblea con un procedimiento fotolitográfico (ver pág. 360). La fotografía visible aquí al lado ilustra la fase de bombardeo de las zonas conductoras con plasma de oxígeno.

M. Wolf/Black Star-Grazia Neri



B. Coleman/Marka

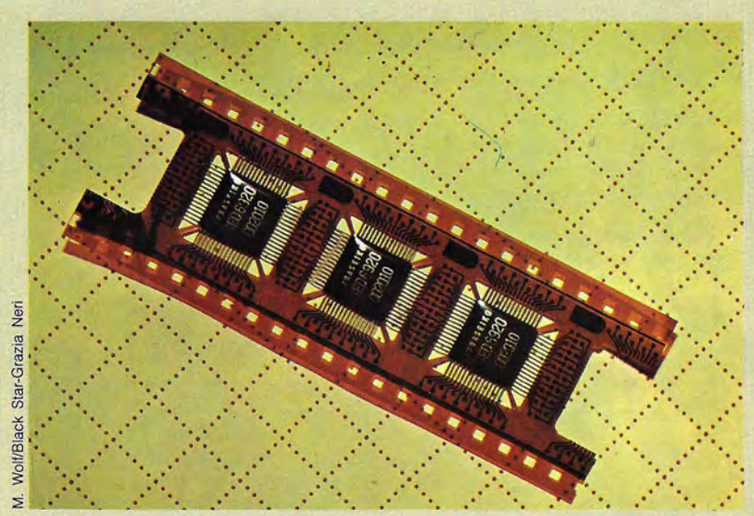
Las obleas, junto con los adecuados materiales generadores de impurezas específicas, se introducen en un tubo de cuarzo (fotografía de al lado) y se colocan en un horno a alta temperatura. Cuando, por evaporación, las impurezas se transforman en gas, sus átomos se difunden en el silicio a través de las «ventanas» creadas por el procedimiento fotolitográfico. Las zonas interesadas para la difusión se convierten en los transistores, las resistencias y los diodos que constituyen el microcircuito.
Al final del proceso de incisión, las obleas se presentan como se ve en la fotografía de abajo. Cada una de ellas tiene grabado el mismo microcircuito repetido hasta llenar completamente la superficie disponible.
Por tanto, las dimensiones del microcircuito determinan el número de ejemplares que puede obtenerse de una oblea, y al mismo tiempo el costo de cada chip sencillo.
La fotografía muestra cómo los microcircuitos grabados cerca de la periferia de la oblea son incompletos a causa de la forma circular de esta última. Por esto, se descartarán en la fase de corte de la oblea.
Antes del corte, cada oblea se controla cuidadosamente con microscopio para identificar los circuitos defectuosos. Estos últimos se marcan con esmalte de color para reconocerlos. Los defectos pueden originarse por las causas más diversas. Dadas las pequeñísimas dimensiones de la zona interesada selectivamente por el proceso fotolitográfico, incluso un simple grano de polvo puede determinar la interrupción de una o más conexiones.
Por este motivo, todas las fases de elaboración se realizan en ambientes controlados, cuya

M. Wolf/Black Star-Grazia Neri

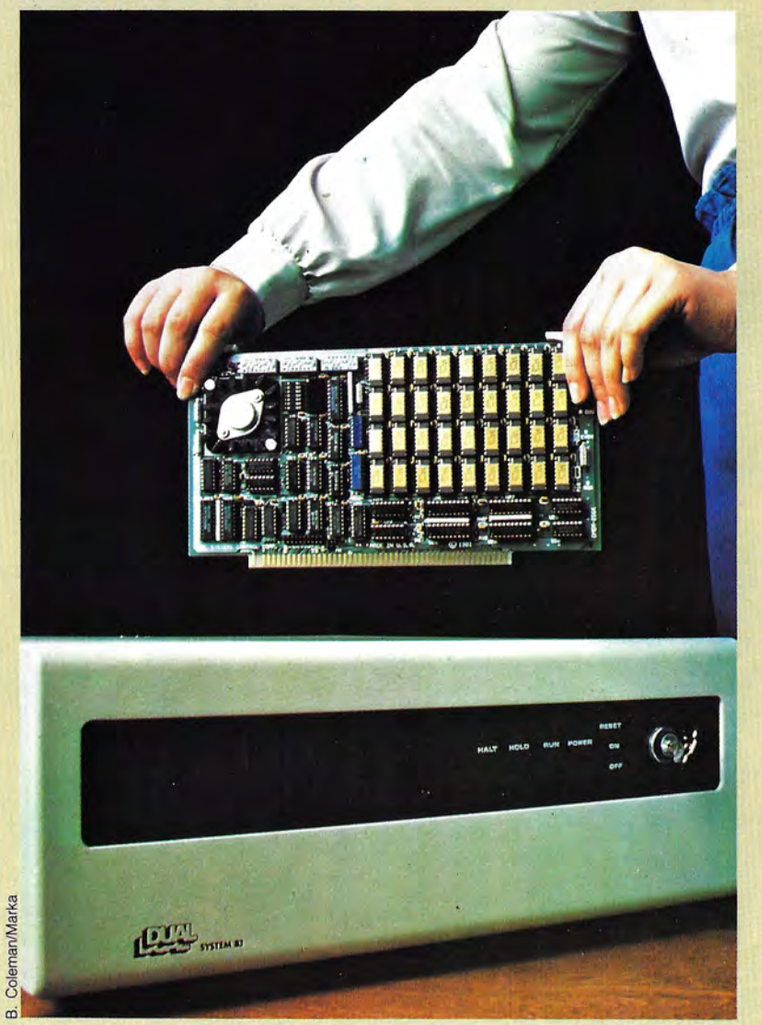
M. Wolf/Black Star-Grazia Neri

K. Reese/Marka

atmósfera se filtra continuamente. La indumentaria de la operaria (fotografía de al lado y del centro) da una idea de la rigurosidad del control. En el laboratorio de la planta IBM de East Fishkill (Nueva York) recientemente se ha desarrollado una técnica experimental de montaje que puede mejorar sensiblemente el rendimiento de la producción de los chips, la cual comprende más de 300 fases diferentes. Para fabricar un transistor se disponen las impurezas, llamadas dopantes, sobre una laminita de silicio. Los dopantes forman los elementos característicos de cada transistor o componente. Los átomos de dopante deben disponerse con la máxima precisión en lugares preestablecidos. Si esto no se consigue, el producto resulta defectuoso. Para disponer el material dopante en posiciones predefinidas son necesarios controles dimensionales extremadamente precisos. El nuevo método experimental asegura el alineamiento de las máscaras que establecen la disposición exacta de los circuitos sobre un microchip. El alineado de las máscaras es uno de los momentos más delicados: centenares de miles de componentes deben disponerse con la máxima precisión sobre chips de sólo 40 milímetros cuadrados, mientras que cada componente puede tener unas dimensiones de una micra. Después del examen óptico se pasa al control de la funcionalidad de los microcircuitos, aún unidos a la oblea (fotografía del centro). Cada microcircuito se somete a un programa de verificación controlado por un procesador electrónico. La máquina posiciona automáticamente las puntas conductoras en las



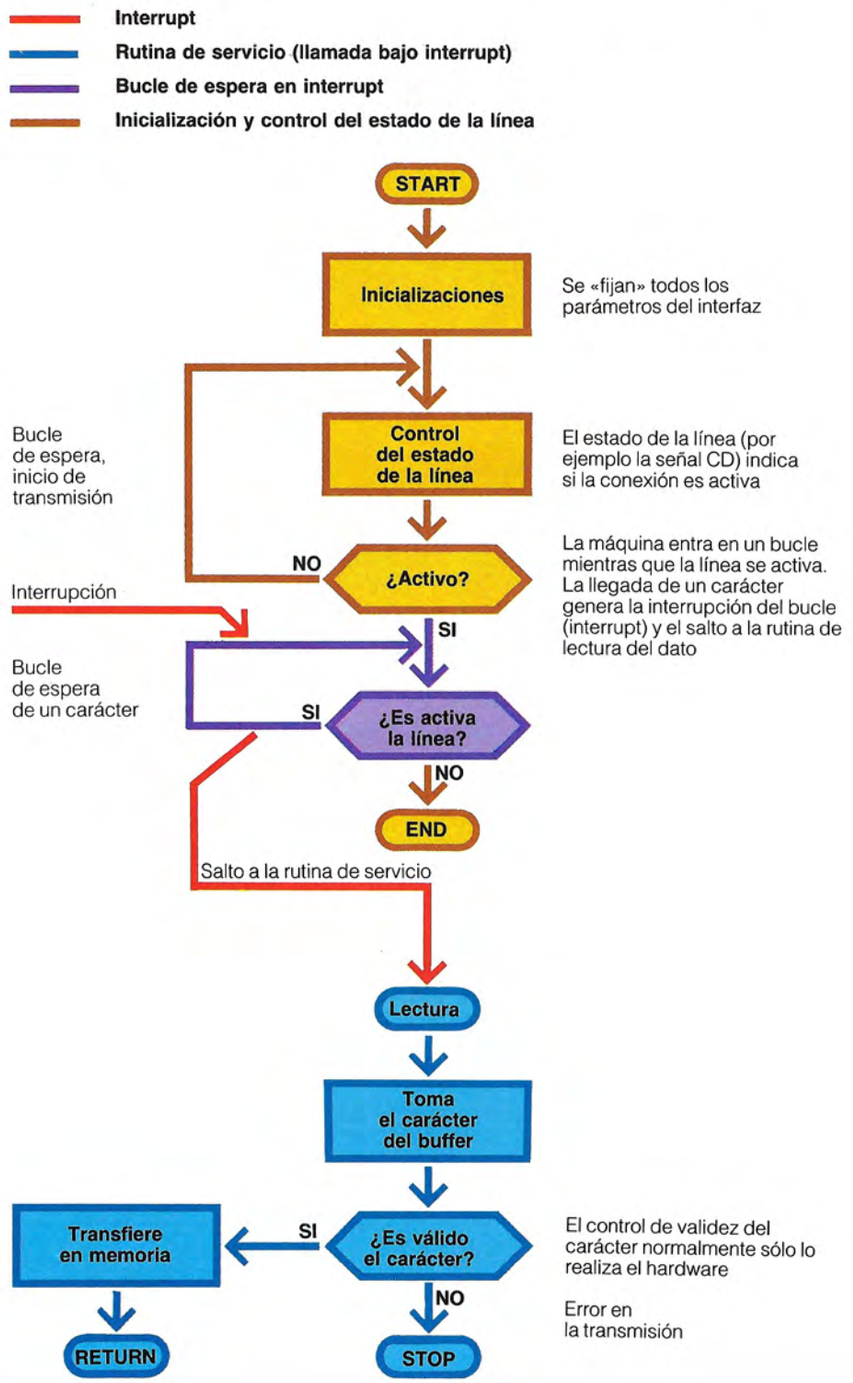
M. Wolf/Black Star-Grazia Neri



B. Coleman/Marka

zonas oportunas del microcircuito y envía débiles señales eléctricas.
La respuesta del circuito se compara con la respuesta estándar del proyecto; las eventuales diferencias conducen a la necesidad de descartar el componente, que se marca para reconocerlo.
Finalizados los controles, se procede a la separación de los microcircuitos sencillos, que se realiza utilizando una sierra de diamante controlada por una máquina micrométrica.
Los chips sencillos pasan después a la máquina de cableado, que suelda en las adecuadas posiciones los conductores que constituyen las conexiones entre el microcircuito y el mundo exterior (fotografía de abajo de la página anterior).
Después de la soldadura de los conductores de conexión, el chip se engloba en una matriz cerámica protectora, que lo aísla definitivamente y sólo deja salir los terminales.
Los módulos sencillos se montan después en una estructura continua muy similar a una película fotográfica, que tiene la función de proteger los delicados hilos que salen de las matrices cerámicas (fotografía de arriba). Los orificios en los bordes se utilizan para desplazar los módulos sin peligro de que se dañen en la máquina que los inserta y los suelda en las tarjetas.
En la fotografía de al lado, los microcircuitos ya están definitivamente en su puesto en la tarjeta, y esta última se inserta en la vía de acceso específica prevista en el aparato que la utiliza. Las conexiones eléctricas se realizan automáticamente en el momento de la inserción, aunque a veces es necesario intervenir directamente para soldarlas de forma definitiva.

EJEMPLO DE PROGRAMA PARA TELECOMUNICACION



En algunas máquinas se han previsto diferentes teclas para generar el interrupt; las principales son la **clave** (key) y el **knob**.
Las claves son teclas funcionales que, en lugar de restituir un valor numérico, generan un interrupt. La selección de la rutina a llamar deberá realizarse en base al interrupt generado. En las máquinas que prevén este tipo de teclas funcionales también hay presentes algunas instrucciones Basic para su gestión. Por ejemplo, la instrucción

```
ON KEY 3 GOTO 100
```

cede el control a la línea 100 al producirse un interrupt generado por la clave 3.
El knob es un dispositivo (generador de impulsos) que puede servir para desplazar el cursor como alternativa a las teclas. Sustancialmente, se trata de un generador, activado por el operador, que emite un determinado número de impulsos en cada giro. El calculador, al contar el número de impulsos generados, puede determinar el ángulo descrito en la rotación y desplazar en consecuencia el cursor. Igualmente para este dispositivo se han previsto determinadas instrucciones Basic (ON KNOB) que pueden pasar el control a las adecuadas zonas del programa.
La gestión de los interrupt en realidad no es tan sencilla como parece. El primer problema es la diferente prioridad que pueden tener los diversos periféricos.
Los periféricos que pueden lanzar un interrupt son numerosos y puede suceder que una nueva petición llegue antes de que se haya terminado la rutina activada por la anterior petición, o que dos peticiones lleguen casi al mismo tiempo. Entonces, el sistema debe decidir si recoge las instrucciones y cuándo. Con este objeto existen prioridades asociadas a cada periférico: cuanto más alto es el valor de la prioridad, más rápidamente se produce el interrupt; las otras peticiones eventuales, si son de prioridad inferior, se ignoran.
El nivel de prioridad de un suceso puede establecerse por software o por hardware. Los niveles de prioridad software, ligados por ejemplo al producirse un error o alguna otra situación anómala, pueden ser asignados con determinadas instrucciones.
La lógica es la siguiente.
Si no existe interrupt en curso, y el programa está trabajando normalmente, se asigna el nivel de prioridad 0 (este nivel lo asigna automáticamente el sistema, y se llama system priority). De este modo, un interrupt cualquiera, también de nivel bajo (que no obstante puede ser inferior a 0), puede atenderse inmediatamente (como al final de la ejecución de la instrucción en curso). Si durante la ejecución de la rutina activada así interviene un nuevo interrupt, pueden producirse únicamente dos casos:

1 / El nuevo interrupt tiene la prioridad menor; entonces se coloca a la espera, y se activará al final del anterior
2 / El nuevo interrupt tiene la prioridad mayor (o igual) al que hay en aquel momento. La rutina en curso (interrupt precedente) se suspende, y la ejecución pasa a la nueva función. Al final se reactiva la rutina suspendida anteriormente.

La prioridad software determina el orden en que se atienden varios interrupts; la prioridad hardware determina el orden con que se acogen los interrupts y es independiente de la prioridad software. La prioridad hardware sirve para programar las solicitudes previas de servicio por parte de los periféricos. A la llegada de un interrupt, el ordenador abandona la tarea que estaba realizando e interroga el interfaze, determina cuál de éstos ha generado la solicitud y lo coloca «en lista de espera», aislando provisionalmente otras instrucciones del mismo interfaze. Una vez realizadas todas las funciones que tienen prioridades software mayores, vuelve a examinar las solicitudes que aún esperan, y eventualmente las atiende.
En las transmisiones, el uso del interrupt es esencial no únicamente para la adquisición, sino también para el control de los tiempos. Durante el intercambio de datos entre el ordenador y el periférico, puede producirse una avería hardware (caída de la línea, o más simplemente, el periférico puede apagarse). Si no se hubiesen previsto controles sobre el tiempo, el ordenador seguiría en espera de la transmisión entrando en un bucle sin salida. Con el objeto de prevenir este inconveniente puede activarse un interrupt sobre los tiempos: si la transmisión no se realiza dentro de un cierto tiempo, se genera un interrupt y se emite un diagnóstico, con lo que el programa o bien se interrumpe o bien pasa a otra actividad.

El software

El software de aplicación orientado a la solución de un determinado problema puede considerarse válido cuando satisface dos condiciones principales:

- Permite obtener las elaboraciones y las salidas deseadas
- Presenta un buen grado de fiabilidad

La primera condición no puede discutirse. Si el programa no permite obtener el resultado deseado en las especificaciones, sólo existen dos alternativas: o se modifican las necesidades o se modifica el software.
El control de la correspondencia entre prestaciones y especificaciones es inmediato: introduciendo las entradas previstas deben obtenerse todas las salidas que se desean en correspondencia.
La verificación de la segunda condición presenta mayores dificultades y no lleva a un resultado tan claro. La ausencia de errores puede comprobarse hasta un cierto punto, e incluso recurriendo a técnicas de control muy sofisticadas, permanece la posibilidad de que exista un error escondido, cuya influencia puede manifestarse únicamente después de presentarse determinadas situaciones.
Para tener una idea del alcance de los problemas generados por los errores escondidos en el software, sólo hay que pensar que para el proyecto espacial Apolo se gastaron para el desarrollo de los programas 660 millones de dólares, y esto a pesar de que la mayor parte de los inconvenientes encontrados en todo el proyecto se debían a errores de software. Un error de este tipo causó también el fallo de la primera misión de la sonda Venus.
Estos precedentes no deben generar desconfianza sobre la eficiencia de los ordenadores. En las aplicaciones normales, los programas no presentan excesiva complejidad y, por tanto, están muy poco sujetos a error.
El procedimiento que permite obtener un programa libre de errores atraviesa dos fases complementarias: la **prevención** y la **detección**.
La fase de prevención deberá empezar junto con la fase de proyecto del software y condicio-

Una sugestiva imagen de los aparatos de una sala de control de Italcable.


Italcable

nará el desarrollo en todas las siguientes fases. La detección de los errores interviene al final, durante la operación de aprobación, pero está condicionada a la imposición inicial del trabajo. La prevención de los errores puede perseguirse desarrollando el proyecto del software de modo que se evidencien las áreas que tienen mayores probabilidades de contener errores. Para cada una de ellas existen técnicas de proyecto y de documentación muy detalladas y complejas. Por tanto, a continuación se tratarán los aspectos principales, teniendo sobre todo en cuenta la clase de máquinas a que hasta ahora se ha hecho referencia. Las principales áreas de intervención que interesan en la producción del software de aplicación funcionalmente correcto son las siguientes:

— Definiciones de los objetivos
— Arquitectura del sistema, en particular de los interfaces con el mundo exterior
— Estructura del software
— Métodos de programación y elección de lenguajes

El primer resultado que debe obtenerse es una visión completa de los objetivos a conseguir. Existen dos tipos de objetivos: los indicados por el usuario final, que constituyen el objeto de las funciones del programa, y los objetivos de carácter general, que siempre deben tender a un software correctamente escrito.
Los objetivos generales que deberán conseguirse siempre en todos los programas de una cierta complejidad son los siguientes:

Generalización. La inversión económica para el desarrollo del software siempre es muy importante, y si el resultado está dirigido hacia un objetivo excesivamente particularizado (y por tanto sólo útil a pocos usuarios), es posible que no se presente la necesaria conveniencia económica.
La habilidad del proyectista consiste en evaluar, caso por caso, cuál es para el usuario final la utilidad real de una generalización. Muy a menudo una excesiva generalización conduce a una cantidad de comandos y de opciones que terminan con la confusión de las ideas y con hacer el programa de uso muy farragoso.

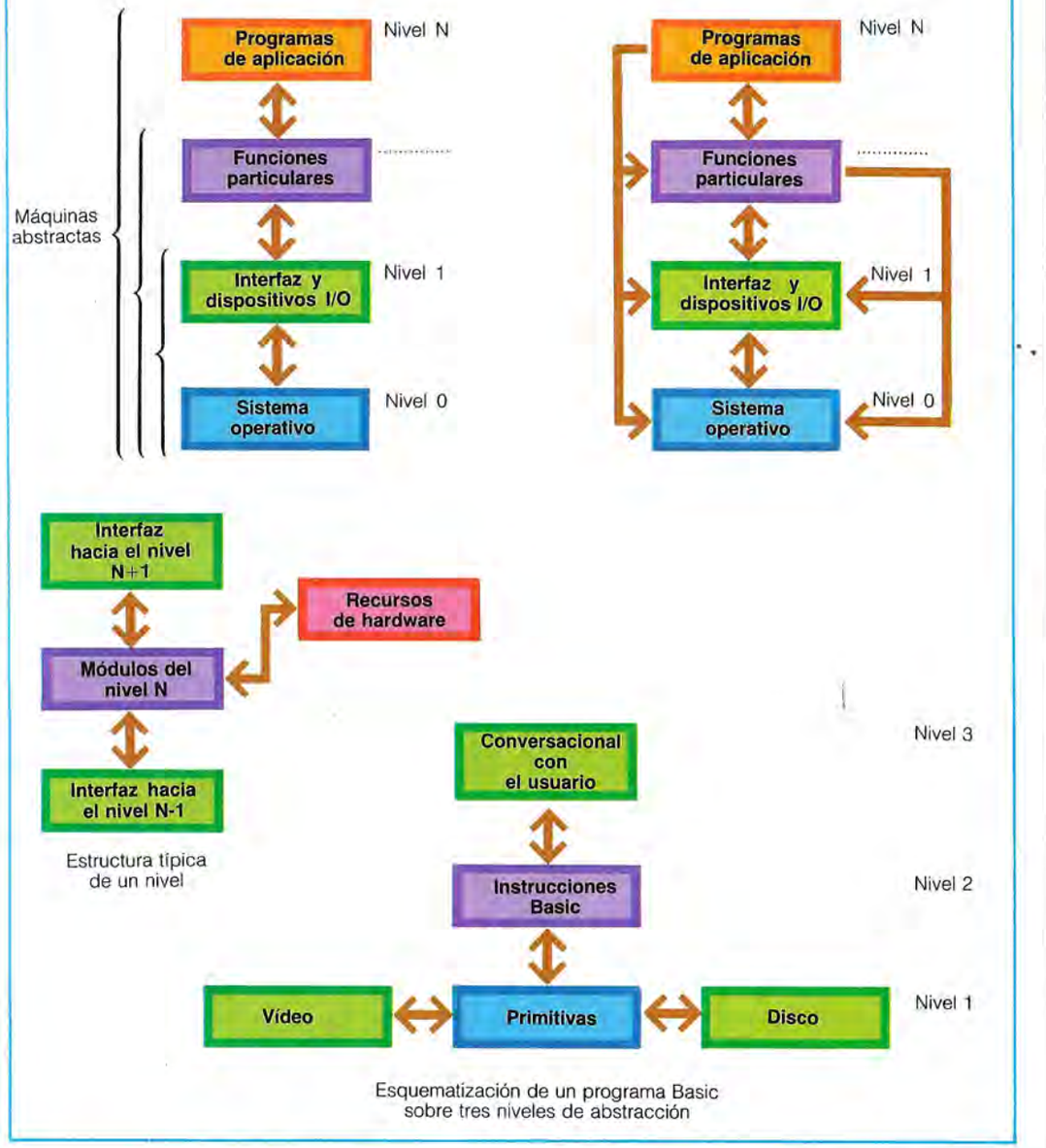
Adaptabilidad. Una escasa generalización del programa puede sustituirse ventajosamente por una buena adaptabilidad, es decir, la posibilidad de implantar nuevas funciones. De este modo puede generalizarse o adaptarse el programa a empleos diferentes.
La adaptabilidad de un paquete de software se consigue principalmente separando las diversas funciones en varias subrutinas o, en los casos más complejos, en diferentes programas, y añadiendo o sustituyendo los módulos según la necesidad. Esta característica favorece además la fiabilidad, puesto que permite la integración en tiempos sucesivos.

Mantenimiento. El software debe permitir correcciones o actualizaciones sin excesivas complicaciones. La medida de esta característica es proporcional a la fiabilidad: un programa, por tanto, es más fiable cuanto más fácilmente puede corregirse.

La arquitectura del software

El software de aplicación destinado a la solución de un determinado problema representa un conjunto de varias funciones distintas pero correlacionadas, cada una de las cuales puede realizarse con un programa o con una o más subrutinas. Las modalidades de correlación de este conjunto, que constituirá un procedimiento, son el objeto de la **arquitectura del software** del sistema, que se ampara en la arquitectura del hardware, la cual define los componentes físicos, sus conexiones y las modalidades de adaptación.
El enfoque prevé al principio la subdivisión del programa general en un cierto número de problemas secundarios, cada uno con su grado (nivel) de abstracción.
La modelización sobre diversos niveles puede obtenerse de dos modos distintos.
En el primer modo, que puede definirse en serie, cada nivel se comunica con el anterior y con el siguiente, mientras que la segunda forma prevé la posibilidad de comunicación, además de con los niveles contiguos, también con un determinado nivel y todos los interiores. En la página siguiente se han esquematizado las dos estructuras, con la descripción de algunos posibles contenidos para cada nivel.
Un análisis de este tipo general es útil principalmente en el proyecto del software de notable complejidad, destinado a grandes sistemas; no obstante, los conceptos y la lógica globales pueden adoptarse en máquinas pequeñas.

NIVELES DE ABSTRACCION EN LA ARQUITECTURA DEL SOFTWARE



Esquematización de un programa Basic sobre tres niveles de abstracción

Cada nivel de abstracción permite diseñar el conjunto software + hardware como una **máquina abstracta** dotada de capacidades que dependen del nivel de abstracción: al nivel 0, la máquina abstracta está constituida (además del software) sólo por el sistema operativo; al nivel 1 puede haber también los interfaces entre el mundo exterior y las subrutinas del sistema, y así sucesivamente. Al elevar el nivel aumenta paralelamente el grado de abstracción hasta llegar al último que permite el diálogo con el usuario final realizando el máximo grado de abstracción. Por ejemplo, en un programa destinado a la lectura de un archivo, el último nivel de abs-

tracción se conseguirá por la subrutina que pedirá al usuario qué record quiere leer; el nivel inmediatamente inferior se conseguirá en cambio con la instrucción GET (lectura del record), que a su vez comunica con un nivel aún inferior (traducción en Assembler) y a través de éste con el nivel 0 (primitivas de acceso al disco).
El ejemplo expuesto evidencia algunas propiedades de este tipo de arquitectura. Las principales pueden enunciarse así:

— Cada nivel no conoce las características del superior
— Las comunicaciones entre niveles diferentes se realizan a través de interfaces definidos previamente
— Cada nivel tiene recursos propios, y debe ser responsable de su funcionamiento proporcionando la adaptación con los de los demás niveles
— El intercambio de datos en tres niveles diferentes queda limitado solamente a algunos valores (como por ejemplo los argumentos transmitidos en las funciones Basic), mientras que el uso de los datos en su totalidad sólo se permite a aquellos módulos que pertenecen al mismo nivel al que pertenecen los datos.

La arquitectura basada en la distribución de las funciones a desarrollar a lo largo de una escala jerárquica (los diversos niveles de abstracción) puede aplicarse ventajosamente en el proyecto del software que no comporte la necesidad de comunicarse entre diferentes sistemas. En el caso de que se hayan previsto comunicaciones, debe adoptarse una estructura diferente, basada en tres categorías fundamentales:

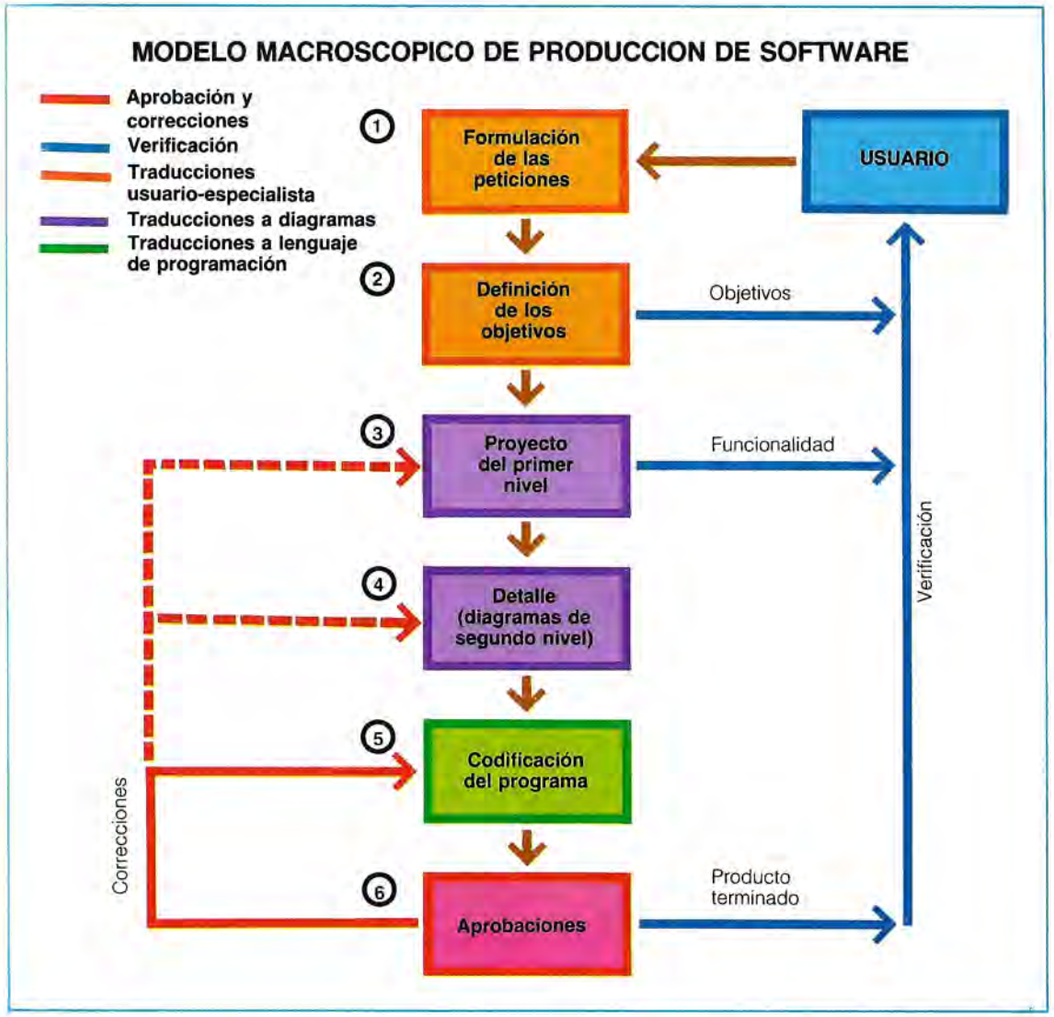
- Módulos que transfieren el control pero que no transfieren datos (por ejemplo el GOTO)
- Módulos que transfieren bien el control bien los datos (subrutinas)
- Módulos para transferir únicamente los datos (puertas de acceso)

La transferencia de datos entre dos módulos (cada uno dotado de más puertas de acceso) se ha visto como proceso asíncrono: un módulo que deba transmitir algunos datos a otro módulo enviará los mismos datos a la puerta de acceso del segundo módulo que, a su vez, cuando esté preparado, los leerá. El proceso (SEND/RECEIVE) se llama asíncrono porque cada módulo trabaja independientemente del otro.
La estructura del interfaz sencillo debe ser proyectada en función del caso específico, observando no obstante algunas reglas fundamentales:

1 / Los mensajes deben tener una claridad y una plenitud que esté en relación con el uso que se hace de ellos. Para un uso esporádico pueden ser suficientes las formas abreviadas, pero si el mensaje aparece con frecuencia y está dedicado a personas ocupadas en otras actividades, debe ser extremadamente claro, de manera que puedan repetirse inmediatamente sin ninguna forma de elaboración por parte del usuario
2 / La introducción de datos, parámetros y comandos por parte del usuario debe reducirse al mínimo utilizando, donde sea posible, valores por omisión. De este modo, si un parámetro necesario en la entrada no es esencial, el usuario puede ignorarlo y proseguir.
3 / Todos los mensajes (de entrada y de salida) deben tener la misma forma estética, de manera que se realice la máxima continuidad en el diálogo con el usuario
4 / En los procedimientos más complejos es interesante prever un comando que active la presentación en pantalla de las explicaciones principales sobre el uso del programa. Durante el período de puesta en marcha de las instalaciones, con frecuencia sucede que el usuario no recuerda cómo debe proceder. En casos similares, algunas explicaciones, incluso muy abreviadas, suministradas en tiempo real, pueden evitar errores debidos a la introducción de comandos no previstos. El módulo predispuesto para esta función se llama normalmente **ayuda** (help).

La modelización

El proceso de producción del software de aplicación puede pensarse como constituido por una serie de traducciones: el usuario final traduce las propias necesidades al sistema, que en la fase de análisis traduce el problema global a otros problemas más sencillos (diagramas de flujo); al final, los diagramas de flujo se traducen a un lenguaje de programación.
Todo el proceso puede dividirse en dos momentos diferentes, cada uno de los cuales comporta a su vez diferentes fases de traducción:



— el primer momento (modelo macroscópico), en el que se identifican los objetivos del software y, a grandes rasgos, también los modelos para conseguirlos.
— el segundo momento (modelo microscópico), en el que cada actividad cuya necesidad se ha evidenciado en el paso anterior, se subdivide, a su vez, en actividades más elementales.

Así se tienen los dos niveles de modelización, macroscópico y microscópico, que servirán como guía ya sea en la fase de implantación ya sea, y sobre todo, en la fase de búsqueda de los errores. En la casi totalidad de los casos, los errores se generan en una (o más de una) de las traducciones contenidas en los modelos; reexaminando el modelo puede determinarse la causa (el origen) del error.

Modelo macroscópico. Arriba se ha representado un modelo macroscópico simplificado del proceso de producción del software.
El proceso comienza con la formulación de los requisitos por parte del usuario ①. La detección tardía de un error cometido en esta fase puede ser extremadamente costosa, puesto que puede requerir una modificación sustancial de toda la estructura del software. Por tanto, esta fase debe ser afrontada con mucha circunspección, teniendo en cuenta el hecho de que raramente el usuario final consigue exponer con precisión todas las necesidades propias. El analista (o el programador) deberá ser quien inicie un proce-

so de interlocución destinado a dar luz a todas las posibles necesidades del usuario.
El segundo paso consiste en la definición de los objetivos a obtener ② y puede considerarse como la traducción del anterior nivel a un nivel más orientado hacia la máquina. En otras palabras, se trata de identificar, para cada uno de los objetivos, el costo en términos de tiempo de programación y de configuración hardware necesaria, y establecer si existen algunos objetivos renunciables que tengan una relación costo/resultado demasiado elevada. Al final de esta fase, que requiere una verificación continua con el usuario, se obtendrán las especificaciones iniciales redimensionadas para ser realizables con una determinada configuración hardware y con determinados límites de complejidad del software.
La fase de definición de los objetivos es tanto más importante como cuanto más nuevo es el usuario ante los problemas de la mecanización. La tendencia más difundida en los usuarios es la de considerar sencillo cualquier problema de software, atribuyendo a la máquina posibilidades que en realidad no tiene. Por tanto, el usuario desea mucho más de lo que realmente le da (porque el trabajo lo realiza la máquina), sin darse cuenta del hecho de que el costo de la instalación depende en su mayor parte del costo del software, y que si debe añadirse un cierto número de requisitos de escasa utilidad puede llevar a un aumento del costo mucho mayor que el que se deduce de las sencillas leyes de proporcionalidad. Por otro lado, también el programador debe ser muy cauto en la orientación del usuario hacia un software que sólo prevé lo esencial. Muy a menudo, las que parecen funciones accesorias son en realidad elaboraciones necesarias; si el programa no las prevé, existe la posibilidad de que deban ejecutarse manualmente, con la pérdida de tiempo que se obtiene y que, a largo plazo, puede conducir al abandono del procedimiento automatizado. El siguiente paso ③ consiste en la traducción de toda la estructura software en un esquema lógico (diagrama de flujo) de primer nivel. El enfoque deberá calibrarse sobre el software de aplicación en su conjunto. El modo de proceder más inmediato comporta la utilización de la técnica top-down.
En esta fase deberán identificarse los flujos de las informaciones y los procesos a realizar. Las eventuales subrutinas deberán identificarse y documentarse, por ejemplo con técnica HIPO, principalmente a fin de controlar la congruencia en el intercambio de datos entre los diversos puntos del programa.
Recuérdese que normalmente, las salidas de una subrutina constituyen las entradas para las siguientes; disponiendo de los esquemas HIPO puede controlarse muy rápidamente la interfactibilidad efectiva. Dada la elevada cantidad de informaciones a tratar, este paso es una de las principales fuentes de error, y como se encuentra todavía en un nivel de primera definición, los eventuales errores cometidos en esta fase tienen un peso determinante. La fase ④ consiste en la escritura de los diagramas de flujo de cada componente sencillo del software (procedimiento, subrutina, etc.). A este nivel, los errores cometidos en la fase anterior pueden detectarse y corregirse con un empleo todavía limitado de recursos.
En el paso siguiente ⑤ se produce la traducción de la solución lógica del problema a un lenguaje de programación (ésta, generalmente, no es la última traducción, dado que existe la que se realiza en el Compilador).
La mayor parte de los errores se produce en esta fase, pero son fácilmente identificables y eliminables.
El proceso podría considerarse terminado con la fase anterior; pero en realidad, la realización de los borradores y la aprobación ⑥ se interfieren íntimamente.
El modelo descrito, que sintetiza todo lo expuesto anteriormente, podrá servir como referencia en el proyecto del software, como guía para la búsqueda de los errores o como término de referencia para expresar un juicio sobre el programa ya preparado o en fase de escritura.

Modelo microscópico. Cualquiera de las fases que llevan a la formulación del modelo macroscópico está compuesta a su vez por numerosas traducciones microscópicas.
Cada traducción microscópica se obtiene en base al siguiente proceso:

1 / Adquisición de las informaciones, por ejemplo una petición por parte de un usuario final

2 / Elaboración mental del procedimiento a seguir para obtener el resultado deseado

3 / Emisión de notas o apuntes que contengan el resultado de la elaboración mental

J. Pickerell/Marka

Sistema de cálculo IBM 370. En el centro puede verse al cuadro de monitorización del sistema de teletransmisión de datos.

Esta esquematización conduce en un proceso mental a las mismas operaciones realizadas por un ordenador (Input, Process, Output) pero con una diferencia fundamental: la naturaleza humana del hardware que realiza el proceso.
La mente humana, bajo este punto de vista, tiene el «defecto» de poseer enormes capacidades asociativas. En nuestra mente, una información nunca se trata separadamente de las otras, y siempre se inserta en un proceso mental que tiene en cuenta las experiencias pasadas y que permite encontrar la justa interpretación de las informaciones no correctas o incompletas. Esta forma de extrapolación puede conducir a errores debidos a interpretación distorsionada de las informaciones recibidas. También, si estas últimas son completas y exactas, pueden ser deformadas por los procesos asociativos inconscientes. Un segundo tipo de error generado en ese proceso se debe a las limitadas capacidades de la memoria humana.
Muy a menudo sucede que se olvidan algunos puntos esenciales y, en tal caso, se desarrolla un proceso de traducción incompleto.
Estos errores pueden minimizarse utilizando determinados trucos, como por ejemplo una documentación muy detallada del procedimiento de análisis bien definido y la elección del lenguaje de programación más adecuado al objetivo.
Cuando en el desarrollo de un proyecto de software concurre más de una persona, se produce una situación particular. Entonces pueden generarse notables problemas de adaptación entre las diversas partes escritas por diferentes personas.
El método para eliminar este tipo de errores consiste en aplicar la ley del «+ 1 – 1»: la verificación de una cierta fase n de un proyecto debe ser realizada por los proyectistas de las fases n – 1 y n + 1.
De este modo, el proyectista de la fase que le precede (n – 1) podrá verificar la compatibilidad entre las salidas de su módulo y las entradas necesarias del que está en examen (el módulo n), mientras que el proyectista de la fase siguiente (n + 1) verificará la correspondencia entre las salidas del módulo que se examina y las entradas requeridas por el propio módulo.

Estructuración de los programas

Terminada la fase de proyecto general (arquitectura del software), el siguiente paso consiste en la definición de los módulos previstos en todos los software.
Cada módulo podrá estar constituido por subrutinas o ser un programa en sí mismo, aunque deberá tener tres atributos principales:

1 / La función realizada
2 / La lógica de realización
3 / El contexto en que opera

La función realizada está definida en la fase anterior (arquitectura) y está constituida por la descripción de las operaciones que el módulo realiza cuando es llamado.
La lógica de realización viene dada por el «cómo» se realizan estas funciones (por ejemplo por los algoritmos), y depende mucho del lenguaje utilizado, así como de las preferencias del programador.
Cada módulo debe ser lo más independiente posible de los otros y no debe contener un número excesivo de instrucciones. Si las funciones realizadas necesitan un elevado número de instrucciones, conviene dividirlo en más módulos de dimensiones reducidas.
En la fase de definición de los módulos es muy importante la documentación. En la preparación de un módulo, la forma correcta de documentación constituye un verdadero plan de trabajo, mientras que en la evaluación de un módulo ya escrito es un procedimiento muy válido para juzgar la calidad, así como la factibilidad de las implantaciones.
En ambos casos, la documentación es un medio indispensable en la fase de aceptación.
La documentación debe contener por lo menos las siguientes informaciones:

- **Nombre del módulo** En algunas formas de Basic y en otros lenguajes (por ejemplo el Fortran), las subrutinas pueden designarse mediante un nombre simbólico (alfabético). Por ejemplo, en Basic estándar, una subrutina de lectura de datos del disco que empieza en la línea 1000 puede ser llamada con la instrucción GOSUB 1000; en su lugar de entrada (entry point) y la línea 1000. En otras versiones del Basic (o en otros lenguajes) les puede ser asignado un nombre como DISCO. En estos casos, el punto de entrada es el nombre DISCO, y la llamada se convierte en GOSUB DISCO.
En ambos casos, la documentación deberá contener el nombre del módulo (1000 para el Basic, DISCO en el otro caso). En general, existe la posibilidad de que un módulo tenga más puntos de entrada, cada uno de los cuales deberá ponerse en evidencia oportunamente
- **Funciones realizadas** Descripción de las funciones realizadas por el módulo
- **Parámetros** Lista de las variables y de las constantes utilizadas
- **Entradas** Descripción de los datos a proporcionar a la entrada
- **Salidas** Descripción de los datos restituidos
- **Errores** Descripción de las eventuales condiciones de error, de los correspondientes valores de los indicadores (flags) utilizados y de las eventuales acciones de recuperación posibles

La técnica de desarrollo top-down

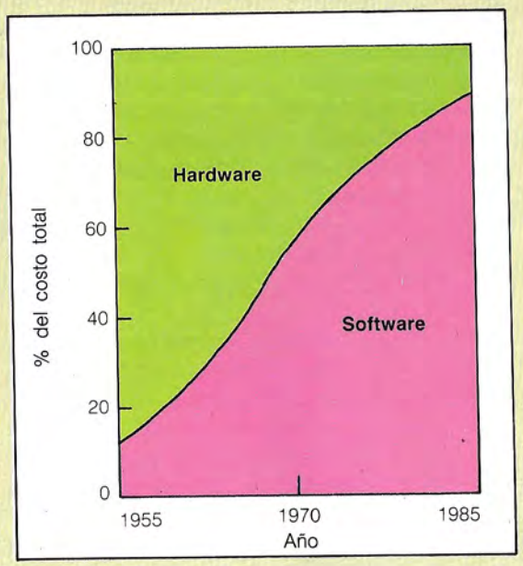
El método top-down es, en sustancia, la trasposición sintética en el plano práctico de los conceptos expuestos en los capítulos dedicados a los diagramas de flujo y a las técnicas de implantación de los programas. Se ha preferido introducirlo en este punto de la obra porque el conocimento del Basic hace menos abstracto el tema y facilita su comprensión.
En el proyecto del software, el problema principal es el de obtener una estructura que pueda interpretarse fácilmente, corregirse o enriquecerse con nuevas funciones. Estos objetivos

¿Qué software?

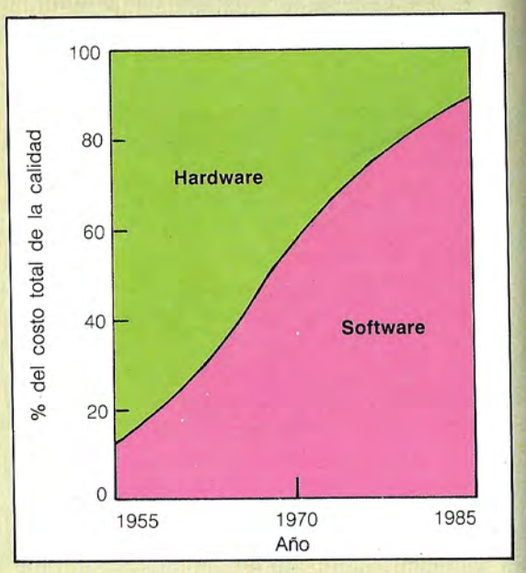
Desde la mitad de los años 50 el mundo del Proceso de Datos está atravesando la «crisis del software» oportunamente prevista por B. W. Boehm y después representada eficazmente con un famoso diagrama aparecido por primera vez en un número de Datamation en 1973 (ver la figura de la parte inferior izquierda de la pág. 778). En realidad han sido necesarios varios años para hacer comprender que los costes del software estaban destinados a superar ampliamente los del hardware y sólo desde hace poco tiempo nos damos plena cuenta del alcance de esta sorpresa y de la dimensión que han asumido los problemas de control de calidad.
Efectivamente, la crisis del software tiene en su interior una «crisis de calidad del software» que, con una oportuna modificación de las escalas de referencia, todavía puede representarse con una curva análoga a la de la figura anterior (ver la figura en la parte inferior derecha de la pág. 778). Con la introducción de la programación estructurada, el software ha dado un enorme paso hacia adelante. Para comprender rápidamente y de forma superficial su significado, conviene recorrer las etapas fundamentales de esta importante innovación conceptual. La necesidad de una revisión de las metodologías de programación se había hecho muy evidente con el aumento de la complejidad de los programas escritos con lenguajes tradicionales del tipo Fortran, Basic, Assembler, etc. El primer paso significativo en la nueva dirección fue realizado por el profesor Edsger W. Dijkstra que, en un famoso artículo allá en el año 1965 observó que «...la calidad del software, o mejor, de un programa de software es inversamente proporcional a la densidad de las fases de GOTO presentes en el propio programa».
El siguiente paso fue realizado por C. Boehm y Jacopini que, en mayo de 1966, anunciaron su teorema de estructura por el que «...realizando una serie de manipulaciones elementales o transformaciones en la estructura de control lógico de cualquier programa no estructurado se puede obtener un programa estructurado».
En 1971, Niklaus Wirth formuló las reglas de base de la programación estructurada, sugiriendo que se llegase a través de una secuencia de pasos sucesivos de refinado. Poco más tarde, siempre en 1971, el doctor Halan Mills y F. Terry Baker demostraron por primera vez qué ventajas podrían conseguirse con esta nueva metodología en las aplicaciones industriales.
Un posterior paso adelante, de gran importancia en la historia de la programación estructurada, lo realizó Parnas en 1972 con el descubrimiento de la descomposición modular. Con esta técnica, la versión final de un programa fuente queda subdividida en unidades caracterizadas por una forma cualquiera de «abstracción».
En 1972 se escribieron los primeros libros de textos universitarios sobre el tema por Weinberg, Dahl, Dijkstra y Moore. En diciembre de 1973 Datamation, con una serie de famosos artículos, presentó la programación estructurada como «una revolución de la programación».
El concepto clave de la programación estructurada es la abstracción, o sea el énfasis selectivo sobre el detalle, que conduce a retener únicamente las propiedades esenciales de un objeto y, como corolario, a despreciar los detalles no esenciales. Por ejemplo, normalmente los usuarios de los lenguajes programación adoptarán «máquinas abstractas» en las que se han eliminado algunos detalles no esenciales como la asignación de las posiciones de memoria. Ignorando el detalle, la abstracción sirve por tanto para reducir la aparente complejidad de la tarea. En otras palabras, una abstracción *es una descripción simplificada o* específica *de un sistema que acentúa ciertos detalles o ciertas propiedades del sistema y desprecia otras. Una buena abstracción es aquella en que la información significativa se acentúa, mientras que los detalles inmateriales o divergentes por lo menos se suprimen temporalmente.*
La abstracción *en los sistemas de programación corresponde muy de cerca a lo que en muchos otros campos se llama «modelo analítico». Efectivamente, estos dos procesos tienen muchos problemas en común que necesitan decidir qué características del sistema son importantes, qué variabilidad/parámetros se incluye, qué formalismos descriptivos deben emplearse, qué modelos van a convalidarse y así sucesivamente.*
Como en muchos otros campos, a menudo se definen jerarquías de modelos *en las que los modelos de nivel más bajo carecen de más detalles que los que aparecen en modelos de más alto nivel. En estos modelos, la descripción es suficientemente diferente de los sistemas reales como para necesitar una convalidación específica. La descripción abstracta de un modelo*

constituye su «especificación» y el modelo de nivel inmediatamente inferior constituye su «actuación». La convalidación que la especifica es coherente con la actuación y recibe el nombre de «verificación».
Muchas técnicas para la organización de los programas y de los lenguajes se basan en el principio de la abstracción. Estas técnicas han evolucionado, por grados, no sólo por la comprensión de la programación propiamente dicha, sino también por nuestra habilidad de percibir las abstracciones como especificaciones formales de los sistemas.
En 1960, por ejemplo, los desarrollos más importantes en la metodología y en el lenguaje de programación se centraron sobre las funciones y sobre los procedimientos, que reasumen una parte del programa en términos de un nombre y de una lista de parámetros. En 1970, los desarrollos se concentraron en el proyecto de las estructuras de datos y sobre las técnicas de especificación, construidas con métodos de lógica matemática y de semántica del lenguaje muy sofisticados para permitir una verificación formal de la coherencia entre programas y especificaciones.
Las nuevas metodologías y los nuevos lenguajes de programación han aparecido y se han desarrollado para responder a los nuevos requisitos necesarios para cubrir la complejidad de los viejos lenguajes (Basic, Fortran, Algol, etc.) y sistemas de programación. A medida que el lenguaje evoluciona para satisfacer dichas ideas nuevas, se vuelven a formular las percepciones de los problemas y de las soluciones de respuesta a las nuevas experiencias.
Así se ha llegado al empleo común del metalenguaje, o sea de un lenguaje formalizado que habla otros lenguajes.
Actualmente, la búsqueda se concentra sobre nuevos lenguajes y sobre la sistematización teórica de los de alto nivel, en particular de los lenguajes algebraicos, o mejor, de los lenguajes normales. Matemáticos y lógicos de fama mundial están dedicando sus mejores esfuerzos a la búsqueda de un lenguaje formal universal y riguroso. Para citar solamente algunas fuentes, baste recordar la escuela vienesa (lenguaje VSM) que asegura que el único tipo de programación es el basado en la matemática más avanzada y rigurosa, en el álgebra abstracta avanzada, o bien la escuela de IBM, a cargo de John Backus y Naur con su estilo de programación por funcionales (FP style), o todavía la escuela SRI International, siempre para el empleo del álgebra abstracta avanzada o de la teoría de los lenguajes formales.
Mientras tanto, el Departamento de Defensa de Estados Unidos ha hecho proyectar, ad hoc, el lenguaja ADA que, empleando algoritmos del

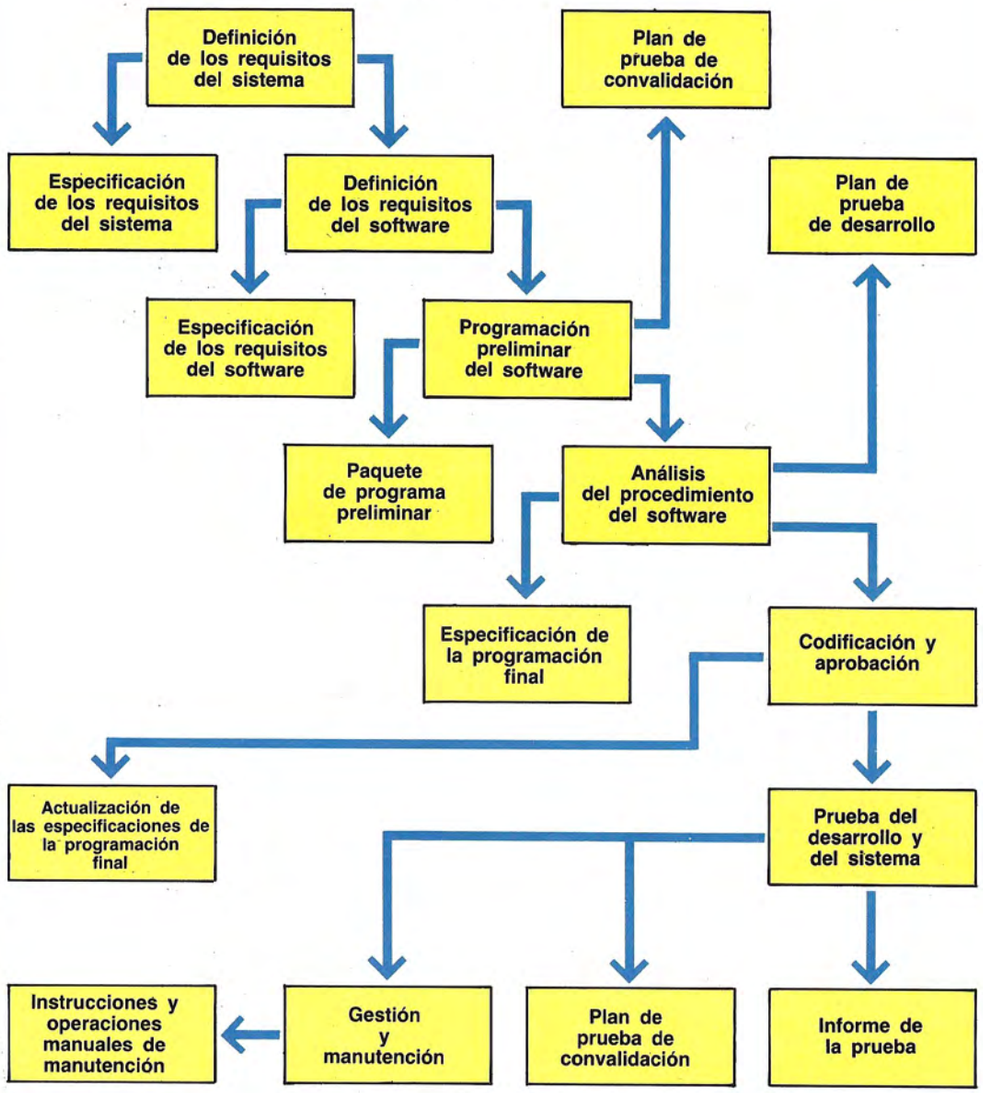
Composición porcentual del costo de un sistema de proceso



Composición porcentual del costo de la calidad de un sistema de proceso



FASES DE DESARROLLO DEL SOFTWARE



lenguaje moderno con las usuales estructuras de control, es capaz de definir los tipos y los subprogramas, de programar en tiempo real, realizando tareas de programación paralela y de gestión .
En consecuencia, aun sin saber qué nos reserva el futuro, razonablemente puede preverse el desarrollo de lenguajes de tipo ADA que, entre otras cuestiones, ha nacido de las tres exigencias de fondo:

1) *reconocer la importancia de las características de calidad, comprendidos la fiabilidad y el mantenimiento del programa;*
2) *considerar la programación como una actividad humana (human engineering = warm);*
3) *conseguir una elevada eficacia tanto total como particular.*

(extraído de «La calidad del software» de Giuseppe Quartieri, DATA MANAGER n.° 27, octubre 1983)

pueden conseguirse utilizando una de las numerosas técnicas de desarrollo conocidas, cada una de las cuales presenta un cierto grado de dificultad.
La técnica de **top-down** es un método no esencialmente complejo y al mismo tiempo plenamente válido.
El enfoque consiste en identificar el problema en sus líneas principales, escindiéndolo sucesivamente en un cierto número de problemas más sencillos. A su vez, estos problemas que podrían definirse de segundo nivel, podrían resolverse directamente, o bien escindirse a su vez en funciones elementales.
La estructura que se deriva de ello es un árbol ramificado que, partiendo de los problemas principales, describe las funciones a desarrollar con un grado de detalle creciente a medida que se procede hacia el particular. La ejecución de cada función, o grupo de funciones, está delegada a una subrutina diferente. La salida de cada subrutina constituirá la entrada para la que le sigue; por tanto, es necesario asegurarse de que los nombres y los tipos de las variables utilizadas se corresponden entre ellas.
En esto puede ser de notable ayuda la documentación con técnicas HIPO.

El método top-down permite construir un programa por grados. Inicialmente se preparan los módulos principales mientras los otros podrían sólo ser simulados; a continuación se sustituyen los módulos simulados con los reales, desarrollando las oportunas pruebas a medida que se integra cada módulo.
La estructura que resulta es muy flexible y fácilmente modificable (para variar una función cual-

La calidad del software en 45 palabras

Software. Es el conjunto de todas las instrucciones y de los datos que deben cargarse en un calculador para realizar un determinado procedimiento. Por tanto, comprende los sistemas operativos, los supervisores, los compiladores, las rutinas de prueba, los programas de aplicación y el llamado «paperware».

Programa. Un programa α es un conjunto de pares de estados (s, t) en el que s es el estado inicial del propio cálculo de α y t es el estado final.

Ingeniería de software. Es el conjunto y el uso de potentes y eficaces métodos y principios de ingeniería orientados a la producción de programas fiables y funcionando correctamente.

Comprensibilidad. Un producto software es comprensible siempre que su objetivo (final) resulte claro al aprobador.

Plenitud. Un producto software está completo si todas sus partes están disponibles y si cada parte se ha desarrollado completamente.

Concisión. Un producto software es conciso si no tiene redundancias de informaciones.

Portabilidad. Un producto software es portátil si puede transferirse fácilmente y hacerlo funcionar correctamente en sistemas diferentes al que se había desarrollado originalmente.

Coherencia. Un programa es coherente si tiene uniformidad de notaciones, simbología y terminología. Además es *coherente exteriormente* si su contenido puede referirse a las especificaciones del proyecto.

Mantenibilidad. Un producto software puede mantenerse a condición de que pueda adaptarse fácilmente a nuevos requisitos.

Demostrabilidad. Un producto software es demostrable si pueden definirse fácilmente los criterios de aprobación y evaluación de sus prestaciones técnicas. La demostrabilidad comprende por tanto dos aspectos:
— la definición de los criterios de demostración;
— la verificación del cumplimiento de los criterios definidos anteriormente.

Utilidad. Un producto software es útil si resulta adecuado y práctico con respecto al uso a que va destinado. También la utilidad presenta dos aspectos diferentes:
— la posibilidad de reutilización parcial o total en otros proyectos (no confundir con la portabilidad);
— la calidad del interfaz hombre-máquina y de las salidas y la correcta definición de los archivos y de los datos de entrada.

Fiabilidad. Definida en sentido llano es la probabilidad que un producto software pueda asumir, en los tiempos y en las condiciones ambientales previstas, las funciones para las que ha sido proyectado.

Estructuración. Un producto software está estructurado si la interdependencia entre sus partes están organizadas en una estructura del tipo camino-tipología.

Eficiencia. Un producto software es eficiente si responde al objeto para el que ha sido proyectado sin desaprovechamiento de los recursos.

Recursos de software. En sentido llano son las magnitudes de la memoria, el número total de las instrucciones, los archivos en línea, la capacidad del canal de I/O, etc.

Independencia. Un producto software es independiente de los dispositivos funcionales si puede hacerse funcionar sobre configuraciones de hardware diferentes a la que se ha proyectado inicialmente.

Precisión. Un producto software es preciso si los resultados que proporciona son suficientemente precisos en relación al empleo a que van destinados.

Accesibilidad. Un producto software es accesible si los datos de entrada están claramente definidos y si los de salida son fácilmente comprensibles y útiles tanto en la forma como en el contenido.

Legibilidad. Un producto software es legible si funciona bien y si su lógica puede deducirse fácilmente de la lectura del código.

Incrementabilidad. Un producto software es incrementable si puede acoger fácilmente una expansión de los datos o de las funciones de ensamblaje.

Ingeniería humana. Un producto software se dice que tiene una cantidad de ingeniería en sus relaciones con el usuario que satisface el fin previsto sin hacerle perder tiempo o energía y sin influir negativamente en su moral.

Modificabilidad. Un producto software es modificable si es que puede aceptar fácilmente la inclusión de una variante.

Comunicatividad. Un producto software es comunicativo si facilita la definición de las entradas y si proporciona resultados fácilmente comprensibles y útiles tanto en la forma como en el contenido.

Autodescriptibilidad. Un producto software es autodescriptible si contiene informaciones suficientes para que un lector pueda determinar fácilmente los objetivos, las asunciones, los vínculos, las entradas y las salidas, los componentes y los estados de los programas sencillos.

Cobertura funcional. Es la inserción de flags, etiquetas y otros símbolos de identificación (eventualmente acoplados con frases de aserción) después de cada función de un programa para señalar, durante la fase de ejecución, si todas las funciones se han realizado en la secuencia adecuada.

Verificación del software. Sirve para determinar si un producto software funciona de manera que satisfaga los requisitos contractuales.

Convalidación del software. Sirve para establecer si un producto software funciona de manera satisfactoria en el ámbito de un sistema global (y, por tanto, también comprende una evaluación de los requisitos de software).

Certificación del software. Ratifica la calidad (fiabilidad, etc.) del software. Puede realizarse solamente por una entidad que tenga la autoridad.

Prueba del software. Consiste en una ejecución del software con el fin de determinar si los resultados producidos son correctos.

Depuración del software. Es el proceso de identificación, localización y corrección de los errores.

Inspección del software. Es un examen del software y de los correspondientes materiales de soporte.

Aprobación del software. Es un proceso de control final y de mejora del software para hacerlo utilizable al usuario.

Prueba de las prestaciones. Es un proceso de demostración de la respuesta de un producto software ante ciertos requisitos de ejecución, que por ejemplo pueden corresponder a los tiempos de máquina o a la dimensión de la memoria ocupada.

Prueba de aceptación. Es el conjunto de todos los procesos formales que permiten establecer si un producto software es aceptable por el usuario.

Ciclo de vida del software. Comprende la definición de los requisitos y de las especificaciones, la fase del proyecto, la implantación de un código legible y asequible por el calculador, el ensamblaje, el ejercicio a gestionar, la manutención.

Modelo de software. Es una entidad que representa útilmente un proceso o un producto.

Algoritmo. Es un texto, es decir un conjunto finito de símbolos, que describe un conjunto, generalmente infinito, de secuencias de ejecución o de procesos secuenciales.

Intervalo de fiabilidad. Es la probabilidad de que en un determinado espacio de tiempo T, el sistema sea operativo y continúe funcionando durante un intervalo de duración X.

Disponibilidad. Se divide en dos funciones:
— la disponibilidad puntual es la probabilidad de que en un determinado instante de tiempo T, el producto software pueda funcionar entre los límites previstos de tolerancia técnica y ambiental.
— el intervalo de disponibilidad es el periodo de tiempo durante el cual se puede esperar probabilísticamente que el producto software funcione dentro de las tolerancias previstas.

Error (causativo). Es una discrepancia conceptual, sintáctica u operativa (de menor nivel) que puede conducir a uno o más fallos del software.

Fault (sistemático). Es una manifestación específica de error debido a una discrepancia del software que impide la posibilidad de realizar la función pedida. Un error puede ser la causa de diferentes fault.

Fallo. Se produce cuando un dato de entrada produce un fault del programa que, en consecuencia, no consigue realizar correctamente la función pedida. Un fallo puede ser:
— detectado, si se ha detectado efectivamente
— no detectado, en caso contrario
— crítico, si crea una situación crítica
— hard, si provoca la caída del sistema
— soft, si es crítico pero no hard.

Safe software. Es un software a prueba de errores, es decir, tan seguro que ningún error de software puede causar un fallo crítico.

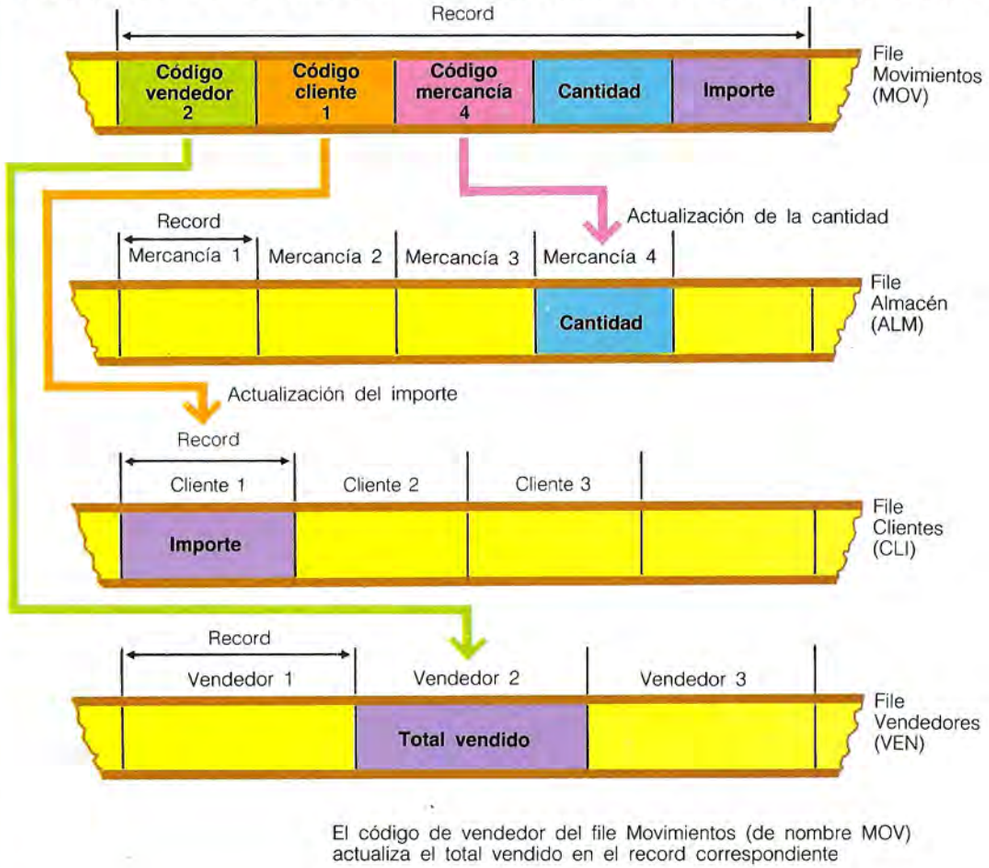
Código estructurado. Es el código de las soluciones de software en que el acento se coloca sobre las estructuras lógicas directas.

Metodología del árbol de los fallos. Es una metodología de análisis de la propagación de los errores, primitivos y causativos, a lo largo del programa y de la consiguiente aparición de fallos de nivel superior.

(extraido de la «La calidad del software» de Giuseppe Quartieri, DATA MANAGER n.° 27, octubre 1983).

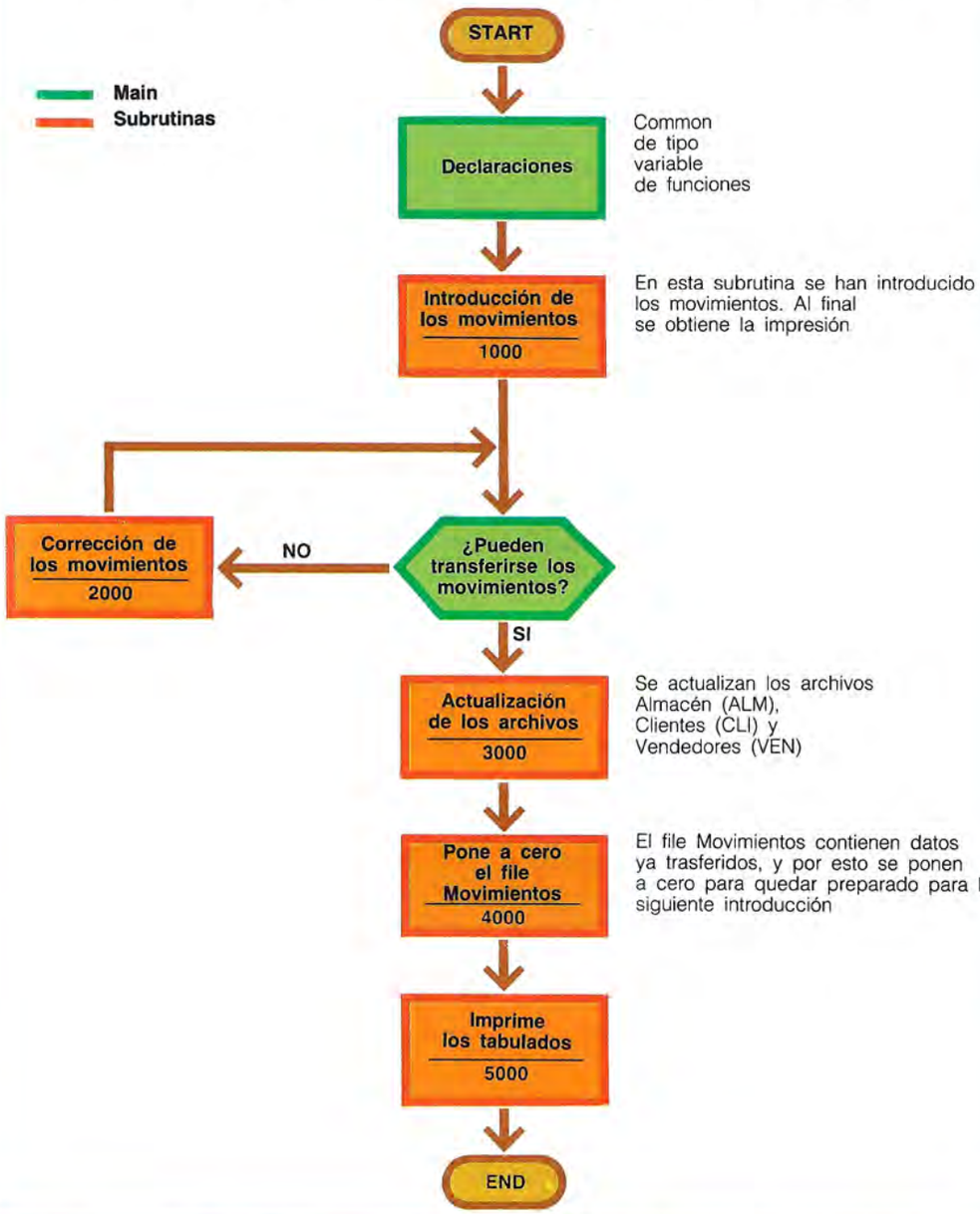
quiera es suficiente sustituir el módulo o la subrutina que la realiza) y permite una aprobación más sencilla y segura de todo el software.
Como aplicación de esta técnica considérese el siguiente ejemplo.
Una sociedad tiene varios vendedores; se quiere memorizar para cada uno las mercancías vendidas (código, cantidad e importe), así como el cliente.
Al mismo tiempo, los datos de las ventas deben actualizar tanto las cantidades que quedan en el almacén, cuanto los acumulados del archivo de clientes.
Las salidas necesarias son las siguientes:
— Total vendido por cada vendedor
— Situación del almacén
— Situación de los clientes
El primer paso consiste en comparar los datos de entrada con los necesarios en la salida para identificar la estructura de los files y los procesos a realizar.
En este caso concreto, los datos de entrada están constituidos por los movimientos, mientras que los procesos consisten en la actualización de los archivos de almacén, proveedores y vendedores.
Por tanto, se identifican 4 files:
— Movimientos: contienen los datos relativos a las ventas
— Almacén: contiene los datos del almacén
— Clientes: acumulación por cada cliente
— Vendedores: acumulación de ventas para cada vendedor
La introducción de los datos se realizará en el file Movimiento; al final (por ejemplo, con frecuencia diaria) se pedirá la impresión y, a continuación, después de la corrección de los eventuales errores, los datos de los movimientos irán a actualizar los archivos. Al lado se ha esquematizado la estructura de los files y los formatos de los records.
En esta página se ha representado el diagrama de flujo de primer nivel del programa.

ESTRUCTURA DE LOS FILES EN UN PROGRAMA DE GESTION DE VENTAS



El código de vendedor del file Movimientos (de nombre MOV) actualiza el total vendido en el record correspondiente

DIAGRAMA DE FLUJO DEL PROGRAMA DE GESTION DE VENTAS



Inicialmente, todas las subrutinas son simuladas y producen simplemente la aparición de un mensaje en pantalla; de este modo puede verificarse si la lógica en su conjunto es exacta. A continuación se pasa a la integración de una rutina cada vez y a su ensamblaje. En el main se han previsto las declaraciones.

Los errores del software

Prevenir y buscar los errores de programación significa conocer sus causas. Los enfoques al problema pueden ser dos, según como estén clasificados los errores. Excluidas las inexactitudes lexicales o de sintaxis, que son diagnosticadas inmediatamente por el sistema, los errores pueden dividirse según la función realizada por la parte del programa que lo contiene y en función del tipo de instrucción.
En la clasificación del tipo de función, cerca del 15% de los errores interesa la zona de acoplamiento del programa, o sea aquellas partes que sirven de comunicación con el mundo exterior; el 25% de los errores en cambio se debe a una colocación secuencial no correcta de las instrucciones. En la clasificación en base al tipo de instrucción, cerca del 30% de los errores están en los DATA o en otras asignaciones, y el 20% en las instrucciones condicionadas.
Por tanto, los datos estadísticos indicados sugieren la siguiente escala de prioridad en la búsqueda de los errores:

- Instrucciones de asignación (30%)
- Secuencia de las instrucciones (25%)
- Instrucciones condicionadas (20%)
- Interfaz (15%)
- Otros (10%)

En cualquier caso, las causas se buscan en el detalle del programa; por tanto, estos errores que generan estadísticamente cerca de la mitad de los inconvenientes son fácilmente identificables. La otra mitad de los problemas tiene un origen más oculto y, a veces, de interpretación muy compleja. Se trata de los errores que son debidos a la inexactitud en la transferencia de las informaciones.
La producción del software implica la descripción, en un lenguaje comprensible por la máquina, de todas las acciones necesarias para obtener un determinado resultado. La identificación de cuál es el resultado a obtener y cuáles son los algoritmos más adecuados para obtenerlo es la misión de la fase de análisis. En esta fase, el usuario final debe transferir al analista (o directamente al programador) una notable cantidad de información inherente al problema específico. A su vez, el analista debe adaptar el procedimiento, descrito según un desarrollo manual, a las necesidades y a los vínculos impuestos por la máquina, y debe conseguir orientar al usuario hacia métodos alternativos más adecuados al procedimiento automatizado.
Así se crea una situación en la que los dos interlocutores, ambos especialistas en un determinado sector, intentan transferirse recíprocamente las propias necesidades objetivas.
De esta manera se determina un caso clásico de incomunicabilidad al menos parcial. Cada uno de los dos interlocutores supondrá, sin darse cuenta, que se dan por descontados algunos puntos, y no los expondrá con el debido detalle. El resultado será un análisis incompleto, que indefectiblemente llevará a un software defectuoso. En la fase de aprobación saldrán los defectos, todos o en parte, y esto obligará a adoptar soluciones de repuesto no evaluadas adecuadamente antes de esto.
Las correcciones conducirán a la introducción de nuevas partes sobre un programa ya definido y no estructurado para estas nuevas funciones. El software perderá entonces la arquitectura original, corriendo el peligro de transformarse en un conjunto de módulos superpuestos al desarrollo de funciones más o menos conectadas, y no integradas armoniosamente. El único modo de evitar este resultado catastrófico consiste en proceder desde la fase inicial según una lógica sistemática basada en tres momentos:

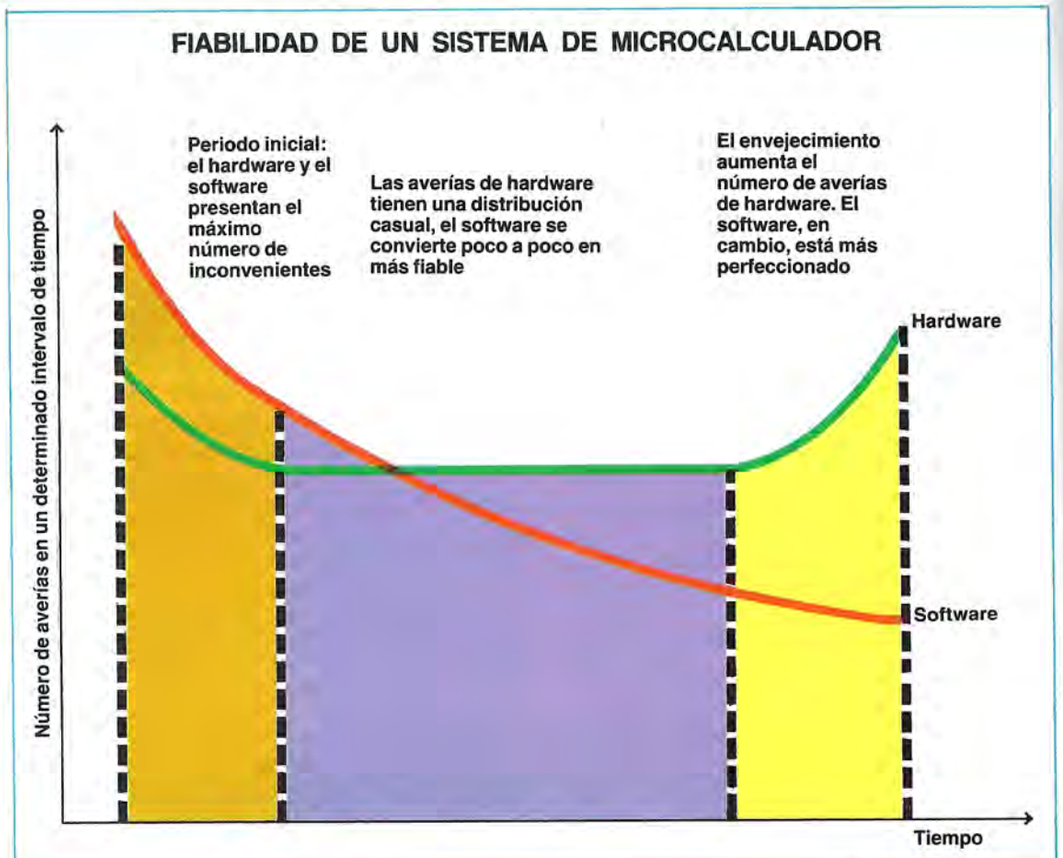
- Exposición de las necesidades
- Elaboración
- Verificación

Aprobación del software

Aprobar un programa significa verificar su eficacia en cada posible condición operativa.
Los resultados de la aprobación permiten responder a algunas preguntas fundamentales que se plantea ya sea el usuario, ya sea el programador:

- ¿Funciona el programa que voy a adquirir o que he escrito? ¿Y hasta qué punto?
- En caso de error ¿corren peligro de destruirse los datos ya introducidos?
- ¿Será posible en el futuro una eventual modificación? ¿Cuál será su costo?
- ¿Con qué frecuencia tendré el sistema parado a causa de errores? ¿Y por cuánto tiempo el sistema deberá permanecer inactivo en espera de las correcciones?

En la práctica, difícilmente se obtiene un grado similar de plenitud. La práctica normal realizada prevé la verificación de las únicas situaciones principales, confiando que éstas puedan cubrir la casi totalidad de los casos. Esta suposición no puede hacerse tan gratuitamente; para que pueda tener algo de veracidad deben adoptarse determinadas técnicas de aprobación y, sobre todo, métodos particulares de estructuración y de detalle de los programas. Por tanto, deben tenerse en cuenta los problemas de aprobación desde las primeras fases del análisis, de manera que se proyecte una estructura que puede convalidarse fácilmente.
Antes de iniciar la búsqueda de los errores debe establecerse qué es un error de software.
Desde el punto de vista del usuario, cualquier funcionamiento no conforme con las propias expectativas es un error de software; desde el punto de vista del programador, en cambio puede tratarse de un funcionamiento normal, ya deseado.
Por ejemplo, las fases de tránsito de los aviones sobre los aeropuertos están guiadas por sistemas de radar controlados por calculador. Al acercarse un objeto volante (no sólo un avión) el radar lo intercepta y transmite las oportunas informaciones al calculador. Este último, o mejor dicho el software de gestión, asocia al objeto un símbolo gráfico y lo presenta en el monitor.
La presentación del símbolo gráfico se produce en cada caso, tanto si el eco ha sido originado por un avión, como si se trata de una bandada de pájaros o de una reflexión esporádica de las señales de radar. El resultado es la presentación de un avión inexistente. ¿Es esto un error de software?
El usuario afirma que sí: no quiere ver cosas que no le interesan y no existen. El programador tiene puntos de vista netamente opuestos: para él, el programa funciona perfectamente, puesto que presenta y gestiona todo aquello que intercepta el radar.
La solución más sencilla es considerar erróneas todas las respuestas en que el programa no está conforme con las especificaciones (con el término especificaciones se entiende la descripción de las funciones que el software deberá realizar y de sus límites en términos de cantidad de los datos entrados, velocidad de ejecución, etcétera).
Esto equivale a establecer la suposición de que las especificaciones son correctas, mientras que, en la mayor parte de los casos, los errores son imputables propiamente a su inexactitud o falta de acabado.
Por tanto, puede concluirse que los errores de software, aparte de los banales que pueden eliminarse fácilmente, no son intrínsecos de la programación, sino que derivan de una compleja concomitancia de causas, las principales de las cuales deben buscarse en cómo se han preparado las especificaciones. Por tanto, para obtener un producto «profesional» deben determinarse exactamente las posibilidades y los límites que quieren obtenerse.
Sin embargo, los errores pueden producirse a continuación de una acción inesperada por parte del usuario, y por otra parte no es posible prever todas las eventuales acciones no correctas e insertar un control para cada una de ellas; el software podrá prever las más peligrosas, pero no todas.
El segundo parámetro a considerar, para una instalación correcta de un programa, es la fiabilidad. Con este término se entiende la probabilidad de que el software pueda trabajar durante un determinado periodo de tiempo sin incurrir en errores.
La fiabilidad del software depende del tiempo: inicialmente es baja (a la puesta en marcha del sistema se tiene la máxima concentración de errores), y aumenta con el tiempo a medida que se aportan las correcciones. La fiabilidad del hardware tiene un proceso diferente: excluida una fase inicial muy breve, el número de las averías se mantiene limitado durante un largo período, transcurrido el cual, a causa del envejecimiento de los componentes, aumenta con mucha rapidez.
En la pág. 786 se ha representado un gráfico que muestra cualitativamente los procesos de ambos fenómenos. La fiabilidad del sistema depende de las dos fiabilidades, y proporciona una medida del intervalo de tiempo que puede transcurrir entre dos paros sucesivos debidos bien a una avería de hardware, bien a un error de software.
En los procesos indicados se ha supuesto que las correcciones aportadas al software no generan otros errores. En realidad esto se produce muy raramente y sólo a condición de que el programa esté bien estructurado. Normalmente sucede que algunas correcciones eliminan un tipo de error pero generan otro. El problema puede minimizarse solamente estructurando los pro-



gramas de modo muy lineal y proveyéndolos de numerosos comentarios que expliquen, paso por paso, su funcionamiento. Esta forma de estructuración (que se ilustrará con más detalle más adelante) constituye la previsión esencial para una buena manutención del software.
Difícilmente un programa podrá ser un producto estático. En la casi totalidad de los casos deberá sufrir modificaciones o actualizaciones. En los programas de carácter económico o administrativo, por ejemplo, deberán insertarse las variaciones consiguientes a las nuevas normativas, mientras que en los científicos podrán variar los algoritmos de cálculo o los parámetros utilizados. Estas actualizaciones son posibles, desde el punto de vista de la conveniencia económica, solamente si el programa es de fácil comprensibilidad.
En la página siguiente se ha representadó un gráfico que muestra el peso de las diversas operaciones sobre el costo total del programa, suponiendo que éste es igual a 100: el costo del mantenimiento es aproximadamente el 50% del total (en los grandes sistemas puede llegar al 75%) y, por tanto, representa el punto más importante en la economía del sistema. No son raros los casos en los que la incorrecta estructuración del programa obliga a reescribir todo el software en lugar de continuar aportándole modificaciones.

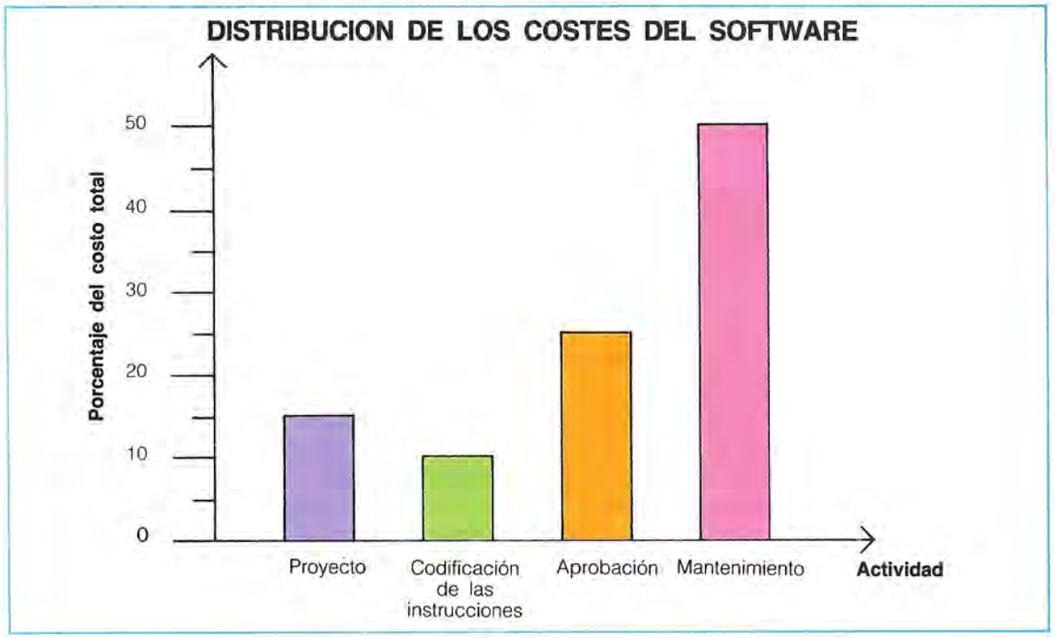
Métodos de aceptación

La aceptación es la fase más delicada y compleja de todas las actividades de producción del software, principalmente por dos motivos. En primer lugar, el argumento es poco conocido y el enfoque suele ser erróneo. En segundo lugar, la finalidad de una aceptación a menudo se entiende como la demostración de la ausencia de errores, mientras que debería ser exactamente lo opuesto: el objetivo de una aceptación es el de evidenciar los errores del software, haciendo lo posible para provocarlos. Los dos puntos de vista no son equivalentes, puesto que

conducen a resultados opuestos: siguiendo la primera metodología se evitan todas aquellas acciones que podrían producir errores; en el segundo caso se utiliza deliberadamente el software de modo incorrecto para controlar también la capacidad de recuperación de los errores cometidos por el operador. La aceptación del software puede estructurarse y realizarse por pasos sucesivos, con métodos análogos a los utilizados para escribir un programa.
En otras palabras, se tratra de programar con un diagrama de flujo las diversas actividades a realizar.
Un programa, también de modestas dimensiones, contiene numerosas ramificaciones, y procede a lo largo de una o de otra en función de la forma en que se utiliza; es imposible prever una aceptación que pueda seguir todas las líneas de flujo del programa. La aparición de un error puede depender de los valores asumidos por las variables numéricas; los tiempos necesarios para comprobar todas las posibles combinaciones pueden evaluarse en términos de años. Estas dificultades sólo pueden superarse aplicando metodologías muy complejas.
En las aplicaciones en los ordenadores personales y microordenadores, la aceptación puede realizarse de forma más sencilla verificando solamente los casos principales, en particular los módulos de interfaz con el usuario.
Las principales técnicas utilizadas en la aceptación se denominan

- Bottom-up
- Top-down
- Big-bang
- Sandwich
- Gráficos causa-efecto

Bottom-up. Un programa normalmente está estructurado como un conjunto de módulos que se envían a ejecución en base a las llamadas (GOSUB) efectuadas por un módulo principal (main), al igual que una determinada subrutina puede llamar otras. En cada caso existe un módulo terminal. El método consiste en aceptar separadamente el módulo terminal para integrarlo después gradualmente con los de nivel superior, hasta llegar al primero. En este punto todo el programa está integrado y aceptado.
El método bottom-up necesita el uso de módulos de servicio, utilizados solamente para la aprobación, que sirven para controlar las partes del programa que se examinan. Así, si el módulo terminal es una subrutina de impresión, para su aprobación deberá disponerse de un módulo que prepare las variables. En el segundo paso, al integrar la subrutina real, utilizará un módulo de servicio que prepare los nuevos datos necesarios para ella, y así sucesivamente.



Las funciones de «control» que desarrollan estos módulos de servicio se llaman **drivers**.

Top-down. Es el método inverso al anterior. El primer módulo controlado es el inicial, por ejemplo el main. A continuación se integran y prueban los otros módulos, procediendo en el sentido del flujo del programa. Aunque en este caso son necesarios módulos de servicio que simulen las funciones realizadas por los módulos aún no integrados. La complejidad de estos módulos (llamados **stubs**) depende de las funciones que simulan. En muchos casos, estos módulos contienen sólo la instrucción RETURN y sirven para verificar que el módulo se ha llamado de forma correcta. El método top-down presenta dos desventajas. La primera viene dada por el riesgo de que, para controlar correctamente las partes que faltan, los stubs pueden hacerse demasiado complejos. La segunda desventaja consiste en que la parte inicial del programa normalmente es la última en quedar completada, puesto que se resiente de todas las variaciones aportadas a los demás módulos; en cambio, con esta técnica, la parte inicial debe ser la primera en superar la aprobación. Este impedimento puede superarse aprobando separadamente los diversos módulos antes de la integración, pero esto requiere la escritura bien de los oportunos módulos de servicio, bien de los stubs.

Bib-bang. Consiste en la aceptación separada de los diversos módulos y, por tanto, en proceder a una aceptación general después de la integración completa. Este método puede aplicarse sólo a programas de modestas dimensiones; en los otros casos puede presentar dificultades prácticamente insuperables.

Sandwich. Consiste en utilizar simultáneamente el top-down y el bottom-up. De esta manera puede evitarse la escritura de algunos drivers o stubs. Puede presentar alguna dificultad y, por tanto, su uso sólo se justifica en los programas más complejos.

Gráficos causa-efecto. Una de las mayores dificultades que se encuentra en la aprobación de los programas es la verificación de la corrección de todos los algoritmos utilizados en las principales situaciones posibles. Afrontar este tema sin una precisa metodología puede significar encontrarse con dificultades insuperables tanto en términos de tiempo como en términos de elección de los parámetros más adecuados para emplear.
Los gráficos causa-efecto pueden ser de gran ayuda en la definición de la estrategia de aprobación. La técnica que utiliza consiste en el análisis del contenido «semántico» de las especificaciones y en su transformación en un gráfico. Posteriormente, el gráfico se transformará en una tabla (**tabla de decisiones**), que representa los casos que deben verificarse durante la aprobación.
Los pasos a realizar son los siguientes:

— Subdivisión de las especificaciones en funciones simples
— Análisis de cada función para determinar cada posible causa (entrada) y cada posible efecto (salida)
— dibujo del gráfico
— conversión del gráfico en una tabla de decisiones

La construcción del gráfico de realizaciones causa-efecto requiere una particular simbología que describe todas las posibles relaciones elementales. En la página siguiente, arriba, se han indicado algunos símbolos principales.
Para ilustrar el método, consideremos el caso de ejemplo en que se quiere producir una subrutina que pueda convertir el primer carácter de una cadena en mayúscula.
Las posibles causas son las siguientes:

a el primer carácter es un número
b el primer carácter es una letra mayúscula
c el primer carácter es una letra minúscula
d el primer carácter no es ni un número ni una letra

mientras que los efectos posibles son:

x la cadena queda sin variación (causa a)
x la cadena queda sin variación (causa b)
y el primer carácter se ha convertido en una letra mayúscula (causa c)
z se emite un mensaje de error (causa d)

además:

1 / si existe error, el programa se detiene
2 / si no hay error, el programa prosigue

SIMBOLOS PRINCIPALES UTILIZADOS EN LOS GRAFICOS CAUSA-EFECTO

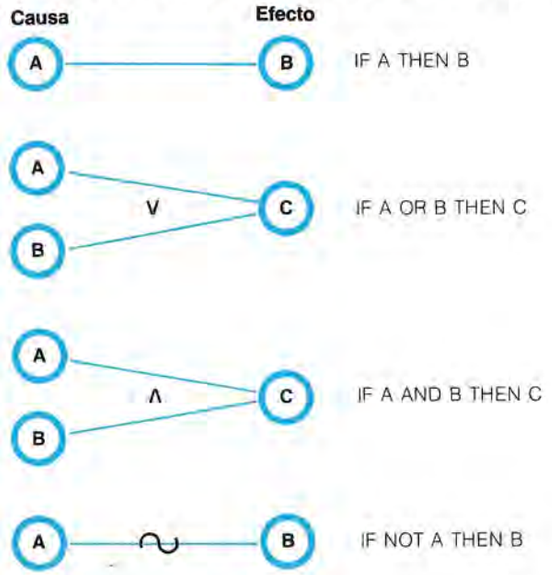
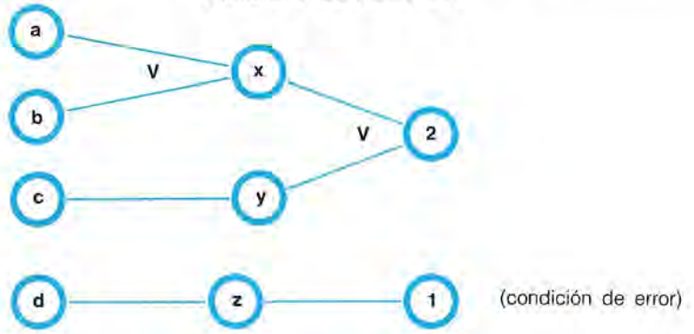


DIAGRAMA CAUSA-EFECTO DEL PROGRAMA DE CONVERSION PRIMER CARACTER



El diagrama de la parte inferior de esta página se construye con la simbología vista poniendo a la izquierda la causa y a la derecha el efecto.
La tabla de decisiones se construye extrayendo cada efecto e indicando todas las posibles causas.
Resumiendo en una tabla todo lo que se ha expresado gráficamente se obtiene el plan de desarrollo de los ensayos (ver pág. 790).

Eliminación de los errores

La aprobación de un programa permite poner en evidencia los errores; la siguiente fase de depuración (**debugging**) permite su eliminación.
En realidad, las dos actividades no pueden distinguirse tan netamente.
A medida que se procede a la aprobación, cualquiera que sea la técnica elegida, al producirse un error debe suspenderse la aprobación y pasarse a la depuración: encontrado y eliminado el error se prosigue con la aprobación. También la operación de depuración debe realizarse con método y siguiendo una lógica predeterminada.
Una de las formas de análisis más utilizada es «el método» (the method), que permite llenar una tabla con tres parámetros principales:

TABLA CAUSA-EFECTO

Causa	Efecto				
	x	y	z	1	2
a	SI	—	—	—	SI
b	SI	—	—	—	SI
c	—	SI	—	—	SI
d	—	—	SI	SI	—

SI = Prueba a efectuar
— = Caso a no considerar

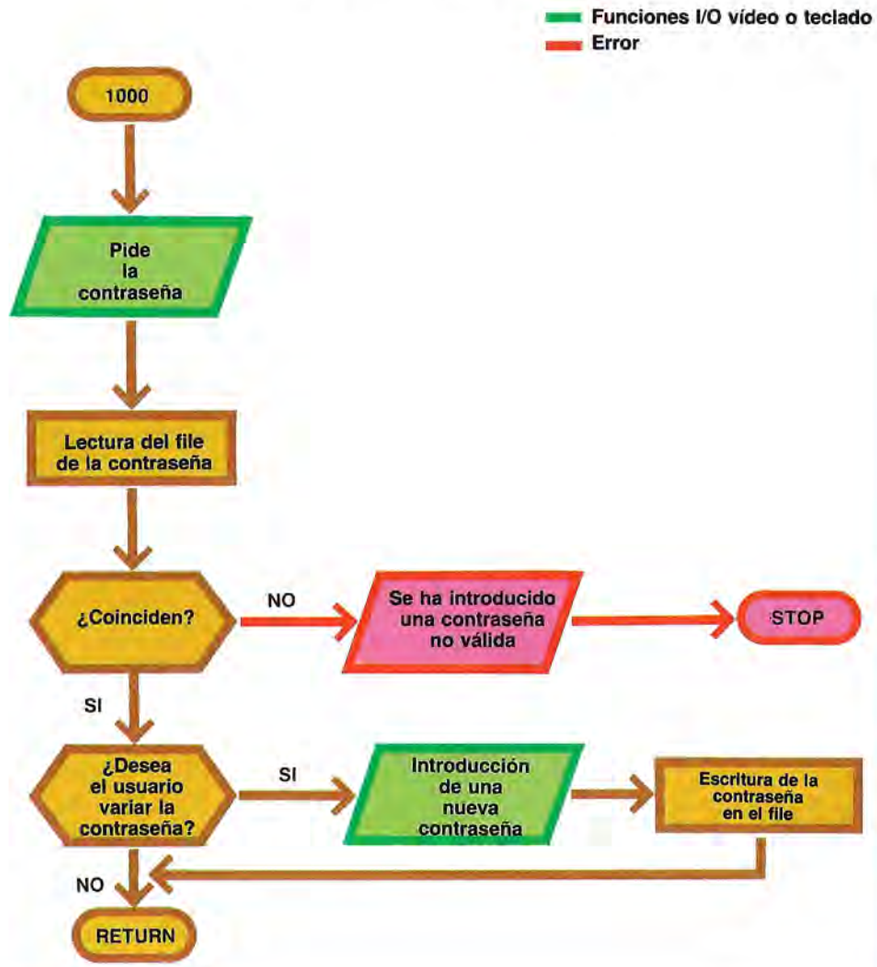
— What (qué): describe el síntoma base del error
— Where (dónde): describe las zonas del programa en que se verifica el error
— When (cuándo): describe las condiciones bajo las cuales se tiene el error

El método prevé un cuarto parámetro (objeto del síntoma) que se utiliza sólo en los sistemas más complejos y sólo raramente. El análisis de las respuestas proporciona la guía para buscar la causa del error.
En Basic, la operación de depuración queda muy simplificada por la posibilidad de utilizar el lenguaje de forma inmediata. Al producirse un error puede detenerse la ejecución del programa y pedir a la impresora todas las variables interesadas, con el fin de detectar eventuales errores anómalos.
En los lenguajes que tienen sólo Compilador, la misma operación requiere la corrección del programa fuente (para añadir las instrucciones de impresión), una nueva compilación del mismo y por tanto de nuevo lanzamiento. Solamente durante este segundo lanzamiento se tendrán a disposición los valores numéricos de las variables utilizadas en la zona del programa que contiene el error. En los casos más complejos puede ser necesario repetir la investigación varias veces antes de identificar el error, pero mientras que en el Basic (interpretado) esto no requiere un empleo importante de tiempo, en los otros lenguajes, la carga puede ser extremadamente importante. Por este motivo se han producido programas que sirven como auxilio en la búsqueda de los errores.
Este tipo de software (**debugging tools**) permite examinar el contenido de las memorias utilizadas por el programa que se examina, es decir presentar en pantalla los valores asumidos por las variables y sin necesidad de insertar instrucciones de impresión. Utilizando esta posibilidad pueden realizarse nuevas tentativas hasta identificar la causa del error. El defecto de esta técnica reside en la necesidad de conocer el lenguaje Assembler.
Los programas compilados se cargan en memoria en forma binaria; cada instrucción está representada por el respectivo código, de manera que las variables se identifican a través de su dirección de memoria y ya no con los nombres simbólicos utilizados en el programa fuente. El software de depuración (salvo casos particulares aplicados sólo en grandes máquinas) prevé sólo esta forma y, por tanto, también si el programa está escrito en un lenguaje de alto nivel, es posible que se esté obligado a realizar la depuración utilizando, por lo menos parcialmente, el Assembler.
En los sistemas más complejos existen determinados programas que pueden dialogar con el usuario y ayudarlo en la búsqueda de los errores. Por ejemplo, el sistema EXDAMS (Extendable Debugging And Monitoring System) mantiene el registro en la cinta de todo lo que sucede durante el desarrollo de un programa; el usuario puede reexaminar la ejecución atendiendo a las informaciones de la cinta, y en cualquier momento puede pedir la presentación en pantalla de las variables.
El método permite hacer correr el programa en el sentido inverso con respecto al flujo (se puede girar al revés como si se utilizase una moviola) pero no es posible aportarle ninguna modificación, puesto que el examen se realiza una vez que el programa ha terminado de correr.
El otro método consiste en utilizar algunas funciones intrínsecas del lenguaje que se adopta. Por ejemplo, en el lenguaje PL/1 existe la cláusula CHECK que, al verificarse determinadas condiciones, genera un interrupt, suspende momentáneamente la ejecución del programa y emite un mensaje.

La seguridad del software

El problema es asegurar la integridad de los datos. En los microordenadores y ordenadores personales, la solución debe preverse en los programas de aplicación.
Las causas principales de pérdida de los datos son imputables a averías de hardware y a los errores del operador.
La prevención contra averías de hardware puede consistir simplemente en hacer periódicamente una copia de los discos que contienen los datos. En el software de aplicación podrá preverse un control sobre la fecha, de modo que informe al operador que ha transcurrido un determinado tiempo desde la última copia y, en consecuencia, que debe realizar una nueva más actualizada.
La prevención contra los errores del operador es más compleja y no siempre tan eficiente. El primer nivel de garantías se consigue permitiendo el uso de determinados puntos del programa solamente a personas autorizadas que poseen una determinada contraseña. Los sucesivos niveles consisten en pedir confirmación de cada operación de anulación o de modificación de los datos antes de efectuarla. Al pedir la confirmación se reclama la atención del operador

DIAGRAMA DE FLUJO PARA EL CONTROL Y MODIFICACION DE LA CONTRASEÑA



que, en caso de error, puede detener el procedimiento.
En la pág. 791 se ha representado el diagrama de flujo simplificado de una subrutina para el control de la contraseña y aquí debajo, el listado de un programa que la utiliza. Obsérvese la línea 1200 en que la introducción se obtiene con la función INPUT$, que no tiene eco en la pantalla; de este modo, eventuales observadores no pueden leer la contraseña introducida. La subrutina tiene una buena validez si está compilada, de otro modo basta leer (en el listado del programa) el nombre del file que contiene la contraseña para poderla modificar o leer su contenido, evitando así los controles. Compilando la rutina puede obtenerse un buen grado de seguridad e impedirse el acceso incluso a personas expertas. La contraseña, por ejemplo, puede enmascararse antes de su transferencia al file, por lo que una eventual lectura no proporcionará elementos útiles, a menos que se conozca el valor de la máscara utilizada. Este valor está contenido en el programa compilado y, por tanto, no es legible.

PROGRAMA PARA EL CONTROL DE UNA CONTRASEÑA

```
10 '   **** MAIN DE PRUEBA ****
12 INPUT" SE DESEA INICIALIZAR EL FILE " ;RESP$
14 IF RESP$<>"SI" GOTO 95
20 OPEN "R"  ,1, "A:WW",5
30 FIELD 1,5 AS W$
40 A$=SPACE$(5)
50 LSET W$=A$
55 ' *** PONE A CERO LOS PRIMEROS 9 RECORDS PARA INICIALIZAR EL FILE
60 FOR RZ=1 TO 9
70 PUT 1,RZ
80 NEXT RZ
90 CLOSE 1
95 GOSUB 1000
96 END
1000 ' ** SUBRUTINA DE CONTROL CONTRASEÑA **
1010 '                FILE = WORD
1020 K=0
1030 PRINT "INTRODUCIR EL PROPIO CODIGO"
1040 '   Cada usuario tiene un código numérico que indica
1050 '   el record en que está memorizada la propia
1060 '   contraseña
1070 A$=INPUT$(2)
1080 IF VAL(A$)=0 OR VAL(A$)>10 GOTO 1290
1090 '  LOS CODIGOS RECONOCIDOS VAN de 01 a 10
1100 OPEN "R",1,"A:WW".5          'WW es el file de las contraseñas
1110                              cada una de 5 caracteres de longitud
1120 FIELD 1,5 AS W$
1130 RZ=VAL(A$)          '  Número del record correspondiente al usuario
1140 B$=SPACE$(5)
1150 GET 1,RZ         ' Lectura contraseña presente en disco
1160 LSET B$=W$
1170 IF B$=SPACE$(5) GOTO 1240    '  La primera vez el file está vacio y
1180                                 debe escribirse la Contraseña
1190 PRINT "Introducir la contraseña"
1200 A$=INPUT$(5)       ' Introducción sin eco
1210 IF A$<>B$ GOTO 1290
1220 INPUT "Se desea variar la contraseña ": RESP$
1230 IF RESP$ <> "SI" GOTO 1280
1240 PRINT "Introducir la nueva contraseña"
1250 A$=INPUT$(5)            Introducción sin eco
1260 LSET W$=A$
1270 PUT 1 RZ                Escritura en disco
1280 CLOSE 1: RETURN
1290 PRINT " ** CONTRASEÑA NO AUTORIZADA **"
1300 IF K=1 THEN STOP
1310 PRINT "TIENE OTRA TENTATIVA DESPUES DE CERRARSE EL PROGRAMA"
1320 K=1:GOTO 1150
1330 '************
```


Los programas de aplicación generalizados

El empleo de calculadores en la actividad laboral no presenta casos tan variados como puede pensarse. Analizando atentamente las tareas a que únicamente se dedica la máquina, pueden identificarse 2 actividades principales:

— Gestión de archivos
— Tratamiento de datos

En las diversas aplicaciones es muy frecuente que se produzcan analogías. Incluso si cambian los datos y las operaciones de cálculo a realizar, la lógica, y por tanto la estructura de base del programa, permanecen invariadas.
Por ejemplo, las funciones a realizar en un programa para la gestión de un listín telefónico son la introducción y la corrección de los datos, la memorización en disco y la búsqueda. Estas funciones pueden realizarse utilizando oportunamente los módulos ilustrados anteriormente (data entry, gestión de disco, etc.). Para adaptar el programa a la gestión de una biblioteca deberá variarse sólo la estructura de los datos: todas las funciones a realizar permanecen invariables y es posible utilizar los mismos módulos.
Estas consideraciones han inducido a los productores de software a preparar algunos programas, utilizables también por parte de quien tiene conocimientos escasos de programación, orientados al desarrollo de las dos principales funciones: tratamiento y archivo.
La evolución de este software ha conducido a programas siempre más completos que reúnen en un solo paquete todas las funciones de base, y que prevén implantaciones particulares como la preparación de gráficos.

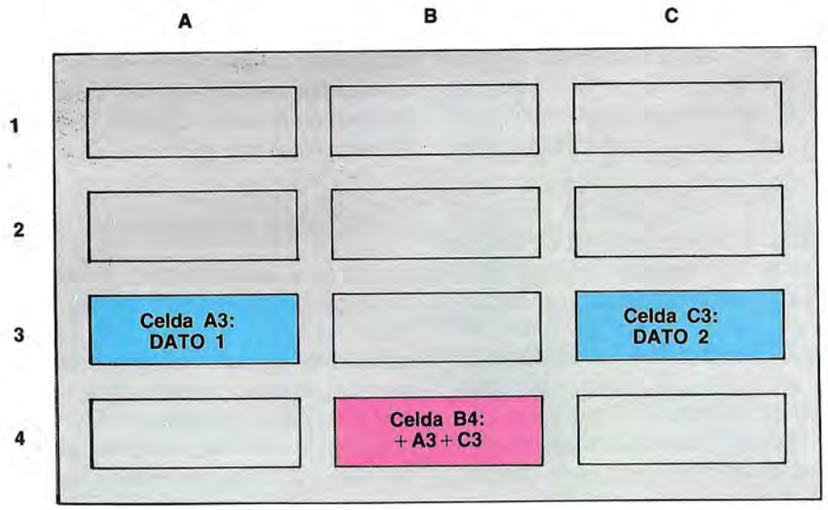
El tablero electrónico

Los programas que realizan principalmente funciones de tratamiento de datos (por ejemplo Visicalc, Multiplan, Lotus) tienen todos la misma estructura, llamada **tablero electrónico**, que permite utilizar la memoria de la máquina como una gran pizarra dividida en filas y columnas.
El cruce de cada fila con cada columna identifica un elemento llamado **celda**, que puede contener un dato (numérico o de cadena) o una fórmula. El contenido de cada celda puede variarse en cualquier momento; para las celdas que

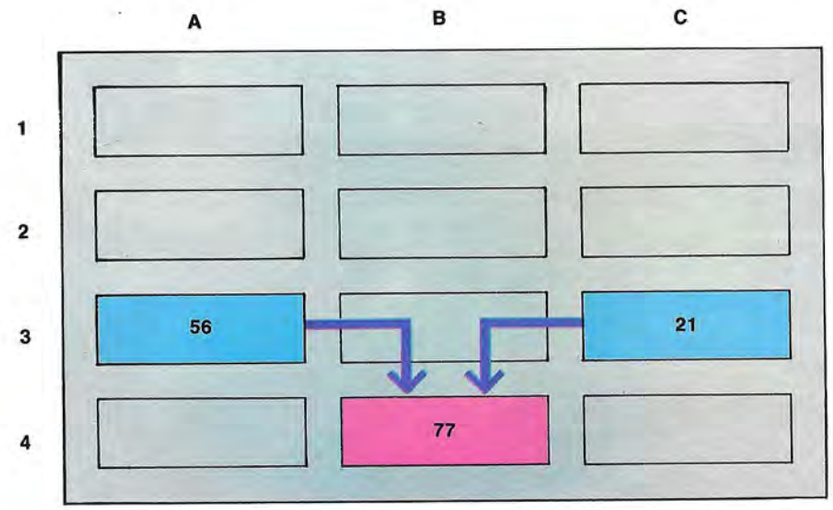
Empleo del ordenador en las Olimpiadas de Moscú de 1980.

M. Vuorela/Marka

IMPLANTACION DE UNA SUMA EN EL TABLERO ELECTRONICO

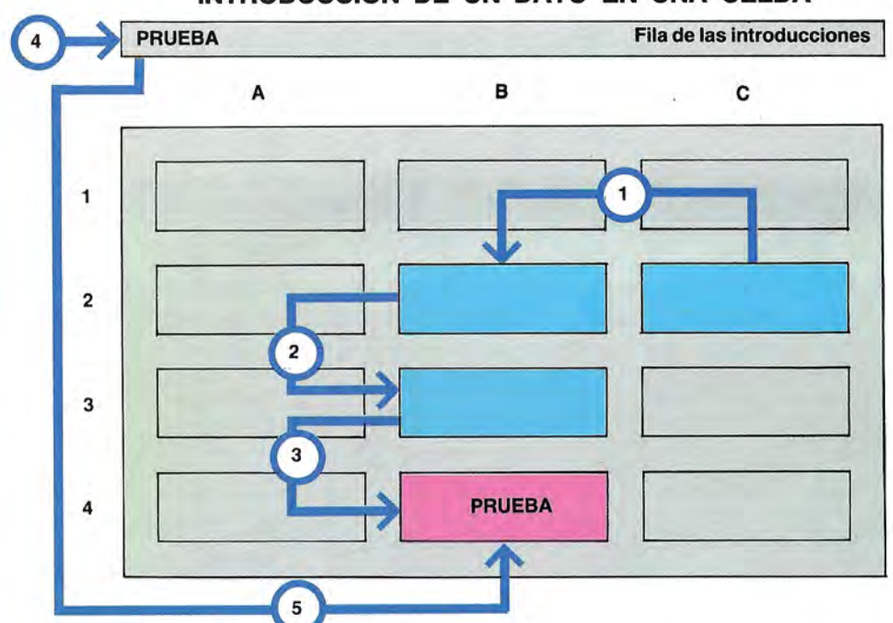


La celda B4 contiene la indicación del cálculo a realizar:
+A3+C3 significa «presenta la suma de los datos contenidos en las celdas A3 y C3».
Introduciendo dos valores numéricos en las celdas A3 y C3 se tiene el cálculo automático y la presentación de la suma deseada en la celda B4.



contienen fórmulas se vuelve a tener inmediatamente el cálculo de resultados.
Utilizando estas funciones, y las numerosas otras que se presentarán a continuación, puede implantarse un cálculo y realizarlo inmediatamente introduciendo sólo los nuevos datos. El usuario no necesita ninguna preparación específica. Únicamente debe implantar las diversas operaciones, como si quisiese realizarlas manualmente; por tanto, el programa se adapta y realiza todos los cálculos necesarios. Enfrente se ha representado el esquema de un tablero electrónico, en el que cada columna se identifica por una letra del alfabeto y cada fila por un número. Por tanto, la sigla B4 identifica la celda situada en el cruce entre la columna B y la fila 4.
La capacidad de la celda normalmente es de 8 a 9 caracteres, pero puede variarse con oportunas instrucciones.
En el esquema se han ilustrado los dos usos principales de una celda: memorización de datos y memorización de fórmulas. Las celdas A3 y C3, que deberían contener los datos a procesar, inicialmente están vacías, mientras que en la celda B4 se ha escrito la fórmula de elaboración (por ejemplo + A3 + C3, es decir, suma de los dos contenidos de A3 y C3). El programa detecta la presencia de una fórmula en B4, y en la fase de presentación no visualiza el contenido de la celda (que aparecería como + A3 + C3), sino el resultado de la fórmula de elaboración aplicada a los datos.
Inicialmente, la celda B4 contendrá el valor 0, puesto que no se han introducido los valores de los sumandos, y su contenido se adecuará automáticamente cuando se varíen los valores de A3 y C3.
El número de las siglas y de las columnas presentadas depende de la amplitud de cada celda y de la de la pantalla vídeo. Normalmente, el vídeo tiene 80 columnas y 24 files, pero existen modelos con 40 columnas, para los cuales la parte del tablero presentada es la mitad que la visualizada en el caso anterior. Para tener en presentación las celdas, por ejemplo, a la derecha de las que pueden verse, es suficiente colo-

INTRODUCCION DE UN DATO EN UNA CELDA

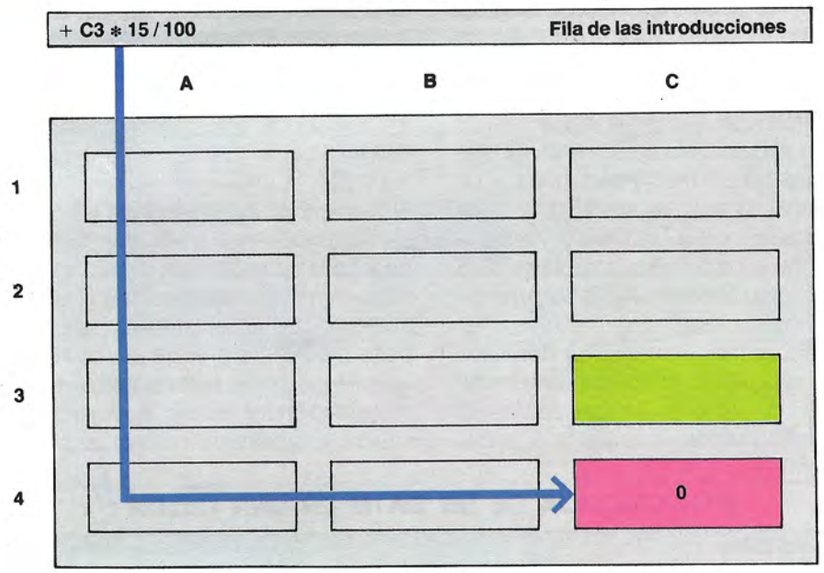


1 Posición inicial del cursor (C2)
Primer desplazamiento: posicionamiento horizontal (tecla ←)
2-3 Después del posicionamiento horizontal, dos desplazamientos verticales consecutivos (espacio + ←) llevan el cursor a la posición de introducción (B4)
4 Digitación del dato
5 Después de haber posicionado el cursor, cualquier dato introducido se transfiere automáticamente a la celda "apuntada".

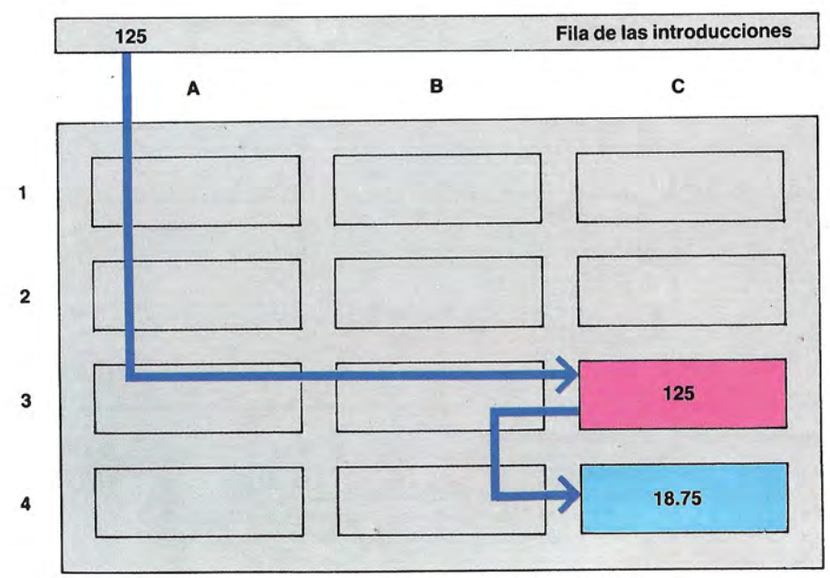
INTRODUCCION DE UNA FORMULA

Celda del dato
Celda ocupada por el cursor
Celda de la fórmula

CORRECTO



Apenas efectuada la introducción de la fórmula, en la celda C4 aparece el valor 0, porque inicialmente el contenido de la celda C3 es 0 (por omisión)

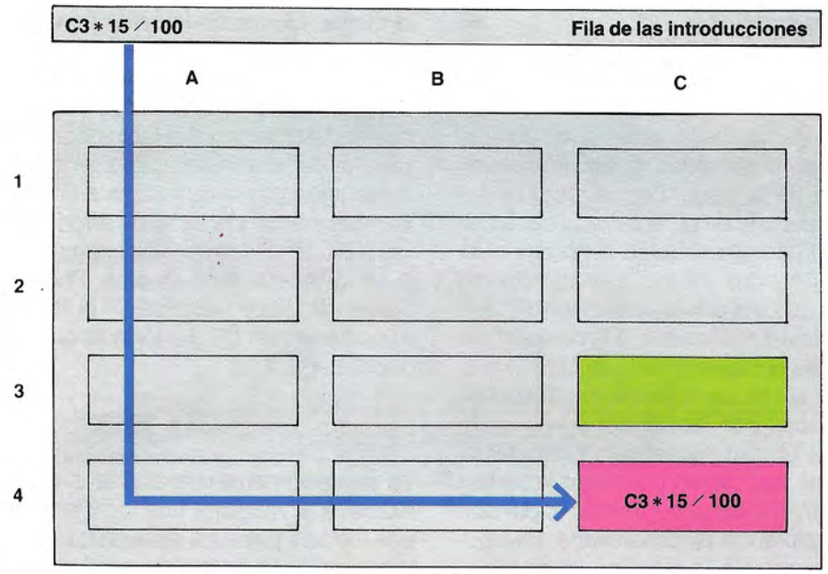


Introduciendo un valor en la celda C3, en C4 se presenta el resultado del cálculo, que se realiza automáticamente

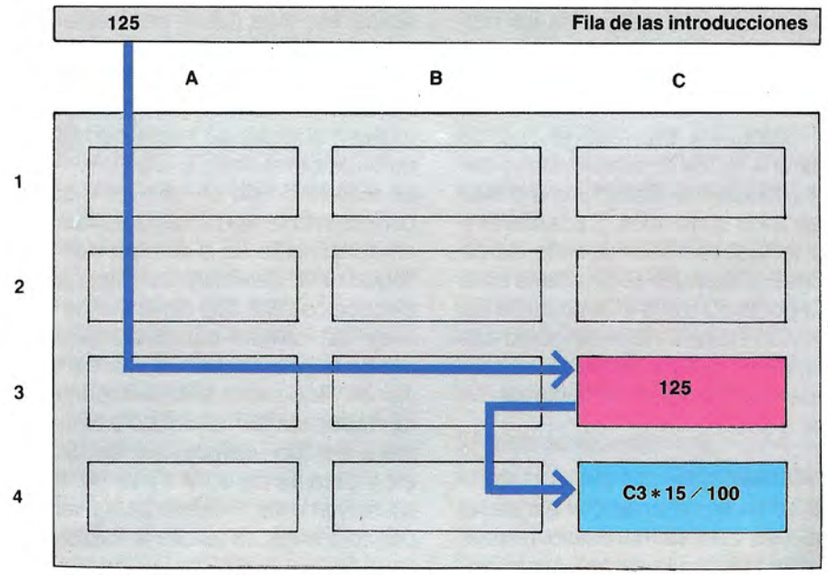
INTRODUCCION DE UNA FORMULA

Celda del dato
Celda ocupada por el cursor
Celda de la fórmula

ERRONEO



La omisión del símbolo + genera un error de interpretación.
La fila introducida se considera un título y simplemente se reescribe en la celda ocupada por el cursor



Cualquier introducción realizada en la celda C3 no genera ninguna variación del contenido de la celda C4, puesto que esta última no contiene fórmulas

car el cursor como si se desease hacerlo salir del margen derecho del vídeo. Se tendrá la aparición de la primera columna a la izquierda (A, B, etc.) y la aparición de nuevas columnas a la derecha. De forma análoga puede colocarse todo el tablero en el sentido vertical. Esta adecuación de las posiciones del tablero se indica con el término **scroll**.

En los programas que utilizan el tablero electrónico, el cursor no tiene la forma habitual (pequeño trazo o pequeño rectángulo) sino que está representado por una zona rectangular presentada normalmente con vídeo invertido de amplitud igual a la de la celda. Desplazar el cursor significa posicionarlo en el rectángulo de la celda deseada. El desplazamiento del cursor puede obtenerse de dos modos: direccionándolo directamente con una adecuada instrucción a la celda de llegada o moviéndolo una posición cada vez a través del tablero. Si el teclado lo prevé, se utilizan las teclas normales de desplazamiento (indicadas con los símbolos ←, →, ↑, ↓), o bien se utilizan secuencias particulares. Los programas que gestionan los tableros han nacido para un empleo generalizado y, por tanto, no tienen referencia en el hardware. En algunos casos, ocurre que la máquina no posee el teclado en la forma más completa, pero entonces el programa está adaptado a estas situaciones particulares.

El desplazamiento del cursor obtenido con el direccionamiento directo sólo es uno de los muchos comandos previstos en el tablero electrónico, cuya complejidad está en relación con el tipo de software.

Normalmente, cualquiera que sea el tipo de software, existe una fila de la pantalla vídeo dedicada a la introducción de datos, y una o más filas reservadas a los comandos. Cada dato introducido (por teclado) se escribe en la fila de las introducciones, y después se posiciona en la celda ocupada por el cursor, a excepción de las introducciones que hemos indicado como comando. En este último caso, el comando se realiza sin que aparezca nada sobre la fila de las introducciones.

El estado introducción

En la pág. 795 se ha representado el esquema de las operaciones que es necesario realizar para posicionar un dato en la celda B4, partiendo con el cursor en la posición C2.

El primer paso consiste en posicionar el cursor en la celda B4. A continuación puede introducirse el dato (en el ejemplo la palabra PRUEBA), que aparece en la fila de las introducciones mientras se digita. Una vez ultimada la introducción, el programa ya ha ocupado la celda de destino, pero el dato todavía no aparece, puesto que la misma celda todavía está cubierta por el cursor. Liberando la celda con un desplazamiento cualquiera del cursor, aparecerá el dato introducido, es decir, la palabra PRUEBA. En este ejemplo, el dato introducido está constituido por caracteres, y por tanto no podrá utilizarse para los cálculos. Si en cualquier celda se escribiese una fórmula que utilizara el contenido de B4, aparecería una señal de error; por el contrario, si B4 contuviese un valor numérico, el cálculo se realizaría normalmente. Por ejemplo, se desea calcular en la celda C4 el 15% del número contenido en C3. La fórmula que debe introducirse en C4 es

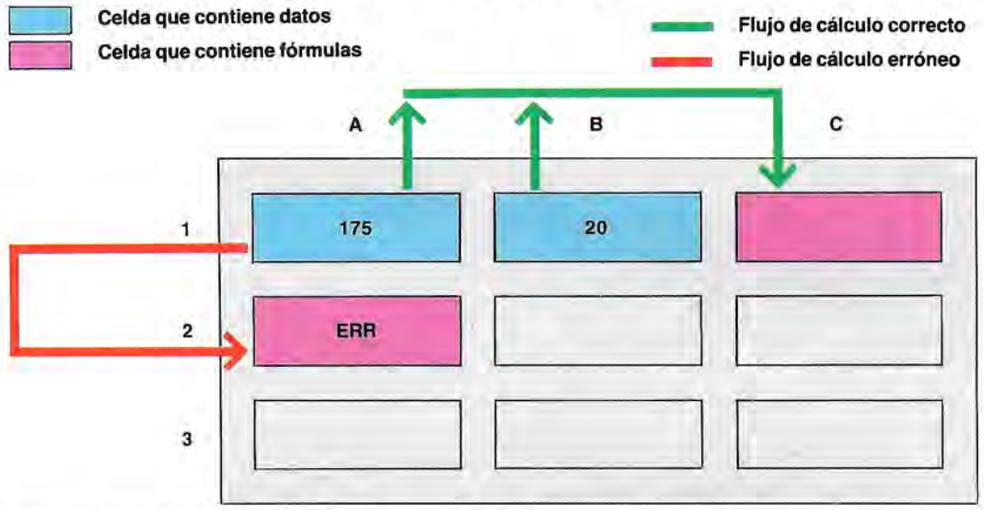
+C3 * 15/100

La expresión empieza con el símbolo + para indicar a la máquina que todo lo que sigue es una fórmula y no una cadena de caracteres. Si la expresión se escribiese omitiendo el símbolo + sería interpretada como un dato alfanumérico que se escribiría en la celda de llegada, pero sin tener las características de un cálculo a realizar. En otras palabras, se trataría de una simple escritura (label) en el tablero electrónico, y no produciría ningún resultado. En las págs. 796-797 se han esquematizado las operaciones que deben realizarse para la inserción de la fórmula, y además se ha evidenciado lo que se obtendría omitiendo el signo +.

La realización de un cálculo, si se ha indicado correctamente, se produce de forma automática en el momento de la introducción de los datos, según una determinada prioridad. El tablero electrónico tiene dos dimensiones (filas y columnas). Un cálculo cualquiera puede realizarse dando la precedencia a una de las dos direcciones. Realizando el cálculo por filas se realizarán todas las fórmulas que aparecen en la primera fila (por ejemplo en las celdas, B1, D1, etc.), para las de la fila 2 (A2, B2, C2, etc.), y así sucesivamente. Y viceversa, haciendo el cálculo por columnas, se tendrá la exploración

A1, A2, A3, etc.
B1, B2, B3, etc.

EJEMPLO DE ERROR GENERADO EN LA ELECCION DE UNA DIRECCION DE CALCULO ERRONEA



La celda C1 contiene la fórmula +A1−B1, cuyo resultado se utiliza en la celda A2.
La celda A2 contiene la fórmula +A1/C1.
Realizando los cálculos por columnas, el paso de A1 a A2 activa el cálculo en A2.
Como el cálculo en C1 aún no se ha efectuado, el contenido de C1 es 0,
y en A2 se genera un error de división por 0.
El flujo de cálculo correcto es por filas.

Al implantar las fórmulas debe tenerse en cuenta el sentido en el que procede el cálculo (el sentido puede seleccionarse introduciendo un comando adecuado), de otra forma, pueden generarse errores debidos al uso de valores todavía no calculados.

Arriba se ha representado un ejemplo de cálculo que utiliza un resultado intermedio. En la celda A2 se ha utilizado el contenido de C1, que a su vez es el resultado de un cálculo anterior. La dirección del cálculo debe ser por filas, de manera que se llegue a la celda A2 con un contenido válido en C1. En el ejemplo, el error que se comete procediendo en el otro sentido (por columnas) se debe al hecho de que en la celda A2 se tiene un divisor igual a 0 (puesto que el contenido de A2 se calcula antes de C1, cuando C1 todavía no se ha calculado y contiene un 0). El sistema detecta la condición anómala (cualquier número no puede ser dividido por 0) y emite un diagnóstico. Si en la celda A2 la expresión fuese exacta, por ejemplo, una suma, el sistema no habría detectado error, y se habría calculado un resultado inexacto sin ninguna señalización. La posibilidad de identificar el error sería escasa.

Para variar el modo de proceder en los cálculos debe impartirse un comando. Cada comando se activa eligiendo la denominación correspondiente en un menú que aparece en una zona reservada de la pantalla. Para algunos programas, la zona reservada al menú es la segunda fila del vídeo, y para otros la última, o las últimas si los comandos son particularmente numerosos. En cada caso existen dos modos de funcionamiento: el **estado ejecución** y el **estado comandos**.

En el estado ejecución pueden introducirse datos, realizar cálculos o funciones particulares (impresión, memorización en disco, gráficos). En el estado comandos pueden implantarse las condiciones de trabajo, que serán válidas hasta una eventual modificación.

El estado comandos

Al principio del trabajo, el tablero electrónico se encuentra normalmente en el estado ejecución, con el cursor en la celda de arriba a la izquierda (A1). Para pasarlo al estado comandos debe activarse una tecla funcional determinada (por ejemplo, en la máquina SIPREL es el símbolo

+). La activación de este código provoca la aparición del menú comandos. En la página siguiente se ha representado un ejemplo de tablero en el estado comandos.
Normalmente, los parámetros presentados son los siguientes:

— Posición del cursor (en el ejemplo B1)
— Modo de realización de los cálculos (por ejemplo, F significa por filas, C por columnas)
— Menú de los comandos

Los primeros dos parámetros tienen carácter informativo (obsérvese que, dada su importancia, la modalidad de cálculo, C o F, siempre se visualiza), mientras que el menú es un recordatorio que sirve para guiar la elección. En dicho tablero se ha representado un menú típico común a muchas versiones de tablero electrónico. Entre una versión y otra se tienen diferencias en las siglas de identificación de cada comando, pero las funciones realizadas son similares, al menos para los comandos principales. En la versión representada, las siglas tienen estos significados:

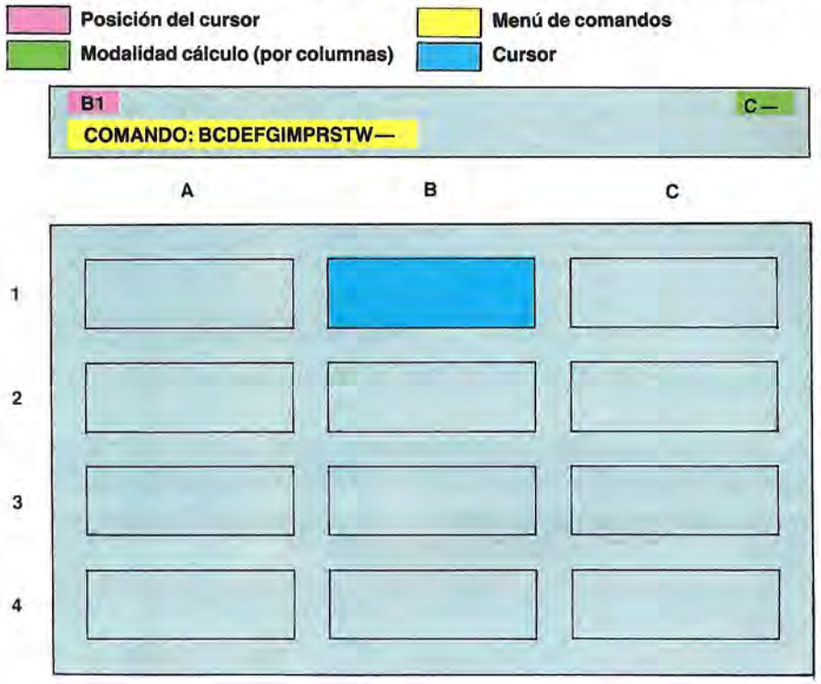
B BLANK	Pone a 0 el contenido del campo donde está posicionado el cursor
C CLEAR	Pone a 0 todo el tablero
D DELETE	Anula una fila entera o una columna entera
E EDIT	Permite corregir el contenido de una celda
F FORMAT	Establece el formato de presentación de los datos en la celda que interesa
G GLOBAL	Varía el formato de presentación de los datos en todo el tablero
I INSERT	Inserta en el tablero una fila o una columna
M MOVE	Desplaza una fila o una columna de una zona a otra del tablero
P PRINT	Establece los parámetros para la impresión
R REPLICATE	Copia una zona del tablero (source range) a otra (target range)
S STORAGE	Activa la memorización del tablero (con los eventuales datos) en un file en disco, o la recuperación de tableros precedentes
T TITLE	Permite la introducción de títulos, es decir, de filas o columnas las cuales permanecen fijas durante el scroll del tablero
W WINDOW	Crea una ventana vídeo
— REPEAT	Repite un símbolo para toda la longitud del campo

Para algunos de estos comandos, la misma descripción de las funciones realizadas sugiere el modo de empleo. Otros comandos, más complejos, prevén ulteriores opciones.
A continuación se relacionan algunos de los principales comandos complejos, siempre con la advertencia de que en otras versiones del tablero electrónico pueden llegar a tenerse siglas diferentes.

FORMAT. Las opciones normalmente previstas en este comando son:

D Vuelve al formato original
G Realiza los cálculos con la máxima precisión permitida
I Presenta sólo la parte entera de los valores
L Alinea a la izquierda el contenido de las celdas
R Alinea a la derecha el contenido de las celdas
$ Escribe los valores numéricos con dos decimales
* Sustituye los valores numéricos por un número de asteriscos proporcional a su valor. Con este comando pueden obtenerse representaciones gráficas (si bien muy rudimentarias). En otros tipos de programas se han previsto opciones gráficas más complejas, que permiten obtener gráficos x-y, de barras o de zonas.

EJEMPLO DE PRESENTACION DE UN MENU DE COMANDOS



En las págs. 802 a 805 se han representado algunos ejemplos que ilustran el uso del comando FORMAT con algunas de las principales opciones.

GLOBAL. Las principales opciones disponibles para este comando son:

C Varía la anchura de la columna. El número de caracteres contenidos en cada celda puede variarse entre amplios límites, y esto permite adecuar la capacidad del tablero a la necesidad particular de la aplicación

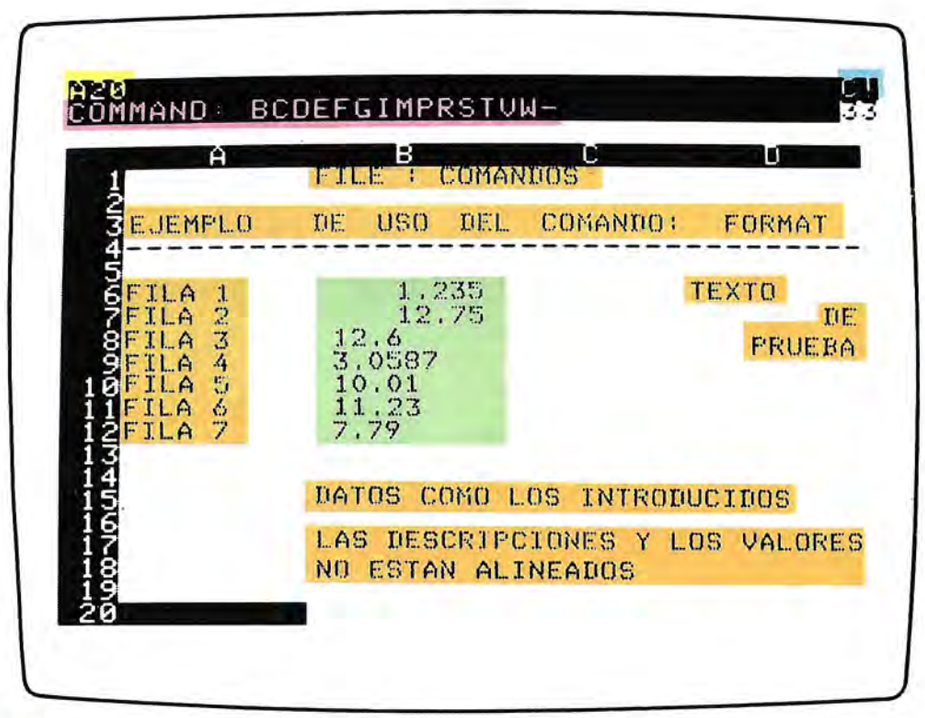
R Con esta opción puede establecerse el recálculo automático de todo el tablero en cada introducción de datos o la modalidad manual. En el segundo caso, el sistema procede al recálculo sólo después de la introducción de un acuerdo (por ejemplo, el signo !). Esta opción resulta útil principalmente cuando deben introducirse numerosos datos. Utilizando la modalidad automática, se tendría un recálculo para cada dato introducido, y por tanto una notable pérdida de tiempo. Utilizando la opción manual pueden introducirse los datos uno después de otro sin tiempos de espera. Al final, con el oportuno comando, se procede al recálculo total.

O Orden del cálculo (fila, columna). Con esta opción se establece el orden en que procederá el cálculo

TITLES. Permite definir como títulos algunas filas o columnas, que permanecen fijas incluso durante el scroll del tablero. El uso de este comando permite tener constantemente en presentación las descripciones de algunas filas (o columnas) incluso cuando se trabaja en zonas distantes del tablero (a causa del necesario scroll, la presentación en vídeo de una cierta zona comporta la desaparición de otras zonas). Con esta opción, las zonas definidas como títulos siempre se visualizan, pero esto implica una reducción del área de trabajo. En la secuencia fotográfica de las págs. 802 a 805 se ilustra un ejemplo del uso del comando TITLE.

EJEMPLO DE USO DEL COMANDO FORMAT (1)

En esta página y en las tres páginas siguientes pueden verse cuatro imágenes vídeo que ilustran el uso del comando FORMAT en el tablero electrónico.
En los ejemplos se hace referencia a una versión cargada sobre el ordenador personal SIPREL 2030 S. Para las otras versiones, los códigos de los comandos pueden ser diferentes, aunque realizan las mismas funciones.



La primera fila de la pantalla contiene la indicación A20, que especifica la posición actual del cursor (columna A, fila 20).

Los símbolos C! indican que se ha activado el cálculo por columnas (C) en modalidad manual (!). De este modo, introduciendo los oportunos valores en las células de datos, los cálculos no se activan automáticamente, sino que es preciso insertar un comando explícito. La elección entre modalidad manual (cálculo sólo sobre petición explícita) y automática (cálculo automático después de la introducción se realiza utilizando una de las voces previstas por el comando GLOBAL (G).

En la segunda fila se han presentado los símbolos de los comandos disponibles. El comando FORMAT se activa introduciendo la letra F.
La fila de los comandos sólo se presenta a petición del usuario. En la máquina que se ha tomado en consideración, la presentación se activa con la tecla ÷.

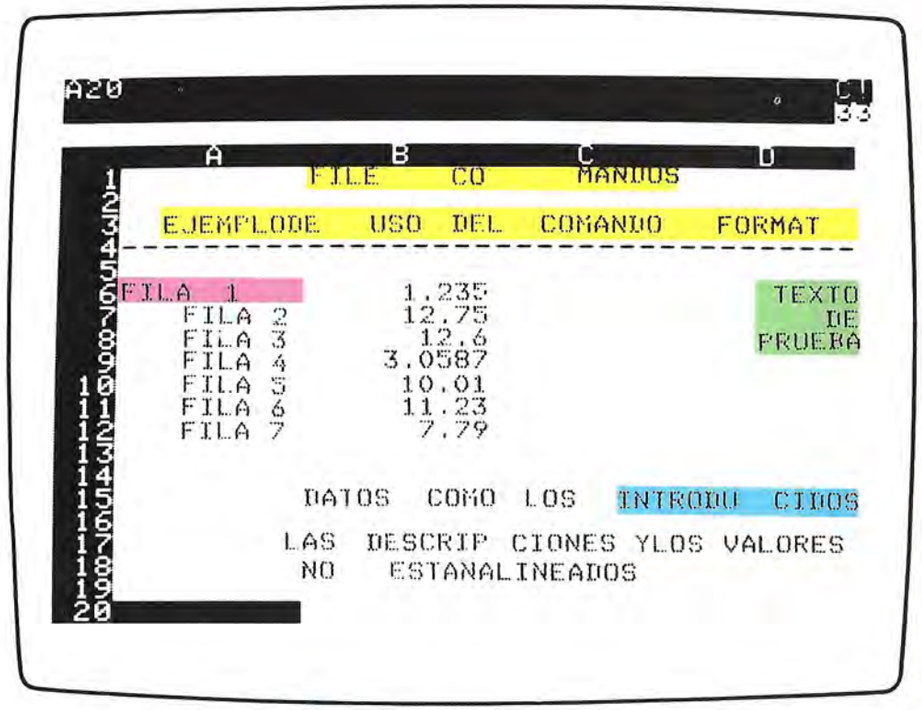
En las diversas celdas del tablero se han introducido los datos sin poner una particular atención a su exacta colocación en los campos previstos.

En particular, las escrituras de los comentarios (titles) se han posicionado al azar en el tablero.

EJEMPLO DE USO DEL COMANDO FORMAT (2)

En esta pantalla pueden verse los mismos datos introducidos en la pantalla anterior después de un alineamiento a la derecha obtenido con el comando G (GLOBAL), opción F (FORMAT), voz R (RIGHT).

Todos los contenidos de las diferentes celdas se han acercado a la derecha, a excepción del contenido de la celda A6, alineado anteriormente a la izquierda utilizando el comando para el alineado de esta sola celda.



En general, en los tableros electrónicos existen dos modos diferentes para implantar un formato. El primero, activado con el comando F, interesa sólo a la celda sobre la que está posicionado el cursor en el momento de la introducción del comando. El segundo (GLOBAL) interesa a todas las celdas, menos las que están vinculadas con el comando F. A causa del realineamiento, algunas escrituras quedan descompuestas.

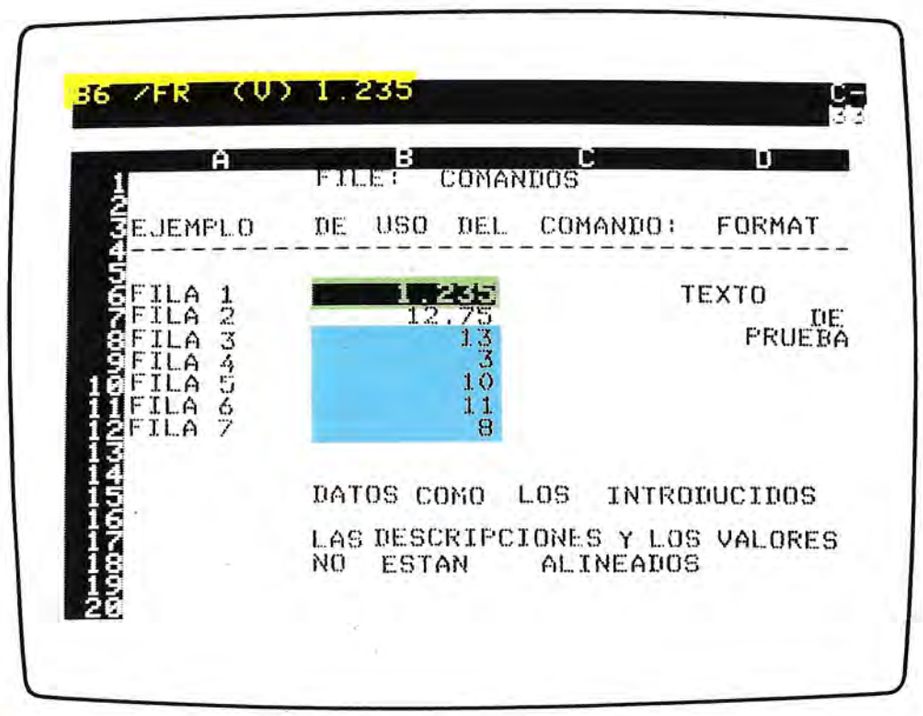
La palabra INTRODUCIDOS, por ejemplo, que ocupaba dos celdas contiguas (C15 y D15), queda dividida en dos.

El mismo problema se plantea para las escrituras FILE COMANDOS y EJEMPLO DE.

La escritura TEXTO DE PRUEBA, en cambio, compuesta por el contenido de tres celdas diferentes, pero que pertenecen a la misma columna, no presenta este inconveniente.
Al insertar en el tablero algunos escritos, se han proseguido naturalmente sobre más celdas contiguas, y esta solución puede aceptarse si se tiene la precaución de no utilizar el comando GLOBAL-FORMAT. Como alternativa es posible ampliar la capacidad de las celdas hasta que quepa toda la escritura más larga, pero de este modo se reduce el número de celdas presentes al mismo tiempo en la pantalla.

EJEMPLO DE USO DEL COMANDO FORMAT (3)

En esta imagen de vídeo se ha ilustrado el funcionamiento de la opción que permite evidenciar el contenido y las especificaciones de formato atribuidos a una determinada celda.



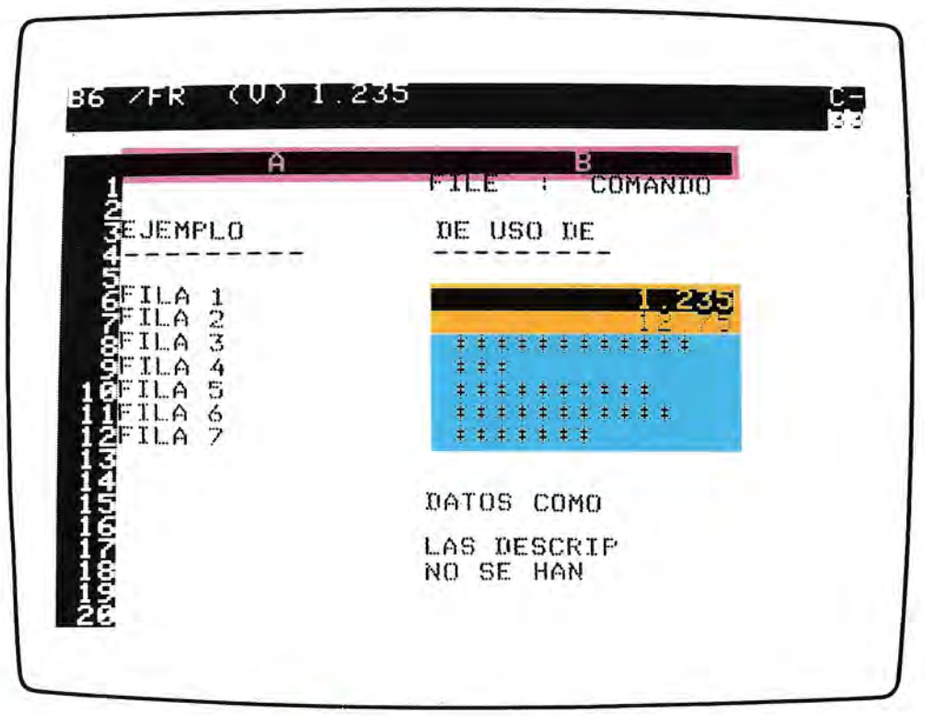
Las informaciones acerca del contenido y las especificaciones de formato asignados a una determinada celda se obtienen simplemente posicionando el cursor en la celda a examen.

En la primera fila de la pantalla aparece la dirección de la celda seleccionada (B6), su contenido (1.235) y la especificación impuesta [/FR y (V)].
En este caso, la celda B6 se ha formado con alineamiento a la derecha (/indica comando, F formato, R right = derecha) y contiene un dato [(V) = value = valor] que es 1.235. El mismo dato podía haberse introducido como cadena de caracteres; en este caso, faltaba la escritura (V), y el dato 1.235 no podría haberse utilizado para realizar cálculos.

Los datos son los mismos proporcionados en la fase de introducción (ver foto anterior), pero la foto se ha tomado después de la introducción del dato G/I (GLOBAL/INTEGER), con el cual se impone a todas las celdas el formato entero.
Los valores se han redondeado automáticamente (p.e., 7.79 se ha convertido en 8) y se han presentado sin decimales. La excepción son los datos contenidos en las celdas B6 y B7, a las que se ha asignado anteriormente el formato «local» /FR.
El redondeado sólo se ha realizado en la presentación.

EJEMPLO DE USO DEL COMANDO FORMAT (4)

La foto de esta página muestra el mismo tablero considerado en las páginas anteriores, después de la introducción de la petición de ampliación de las celdas.



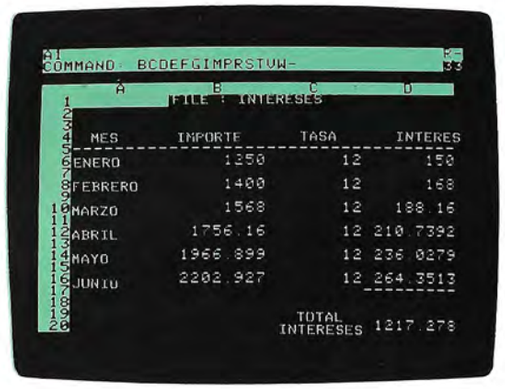
El número de caracteres por celda se ha duplicado, y el número de columnas presentes al mismo tiempo en la pantalla se ha disminuido automáticamente.
Con la introducción de la opción G,* activa una forma rudimentaria de presentación gráfica de los datos introducidos.

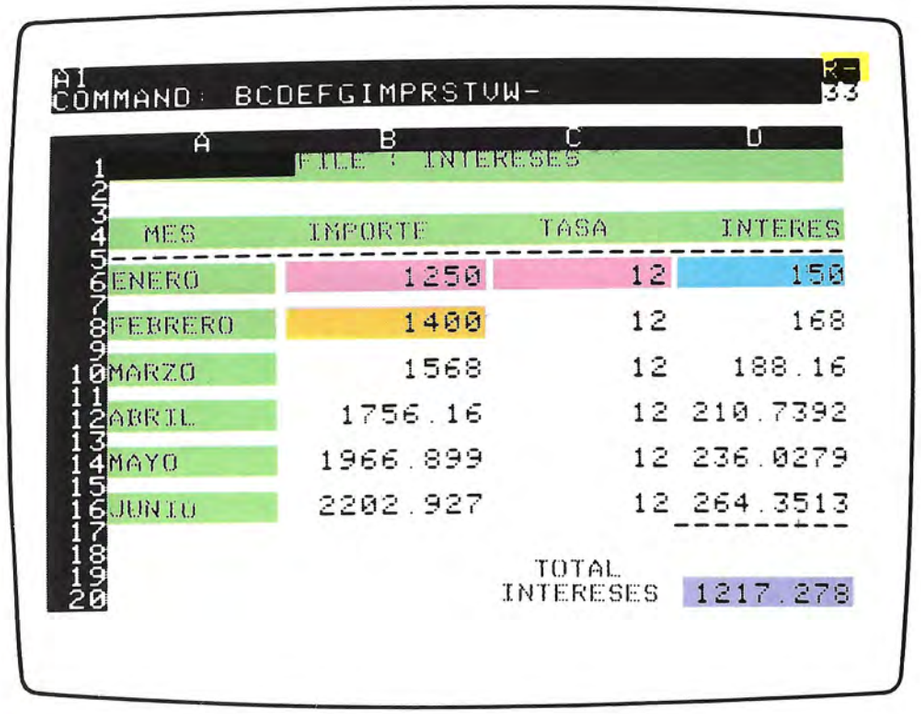
En cada una de las celdas aparece una fila de asteriscos proporcional al valor numérico almacenado.

Normalmente, las celdas B6 y B7 están excluidas a causa de la anterior introducción de especificaciones de formato «locales».
Este tipo de presentación puede ser útil para un examen total del proceso de los valores numéricos introducidos. En los tableros más sofisticados existen opciones que permiten activar la generación de gráficos propiamente dichos. En estos casos, todas las escrituras accesorias desaparecen, y en la pantalla son visibles sólo los gráficos pedidos.

CALCULO DEL INTERES COMPUESTO

La foto muestra un tablero electrónico predispuesto para el cálculo de un plan de pagos de intereses compuestos para un determinado capital inicial.
Para cada mes, el capital a considerar es el del mes anterior más los intereses.





La primera fila muestra que se ha elegido la modalidad de desarrollo de los cálculos por filas.

Las celdas de las filas 1 y 4 y de la columna A contienen las descripciones.

En B6 se ha introducido el capital inicial, mientras que la celda C6 contiene la tasa de interés (12%).

En la celda D6 se ha introducido la fórmula para el cálculo del interés que utiliza los datos introducidos en B6 y C6 en la manera +B6*C6/100. Debido a que se ha implantado el cálculo por filas, el valor 150 (D6) aparece automáticamente apenas se han introducido los datos 1.250 (B6) y 12 (C6).

Para el cálculo del interés correspondiente al segundo mes debe hacerse referencia al capital del primer mes más los intereses devengados. En la celda B8 por tanto, se ha introducido la fórmula +B6+D6, que calcula automáticamente el valor del capital.
El cálculo continúa para todas las celdas con las mismas modalidades. Obsérvese que los contenidos de las celdas de las columnas B y D se calculan prácticamente todos en el mismo instante que sigue a la introducción del capital inicial en B6.

El total de los intereses se calcula en D20 utilizando la función SUM (D6..D16).

La automatización del trabajo de oficina (1)

La burótica, la mecanización de oficinas y la automatización de la oficina y su racionalización, son modos diferentes de representar una única realidad que se va perfilando en el horizonte de las sociedades industriales.
La perspectiva es de que se va a realizar un salto sin precedentes en la productividad económica pero, en estos casos, si no se utiliza bien, puede crear situaciones de despilfarro tecnológico y agravar las tensiones existentes en el mundo laboral.
Los componentes sociales, metodológicos y organizativos del fenómeno juegan un papel tan determinante que sería ingenuo pensar que se pueden afrontar con medios y competencias exclusivamente tecnológicas: la propia realidad de la oficina es, por definición, tan compleja y rica de interconexiones (formales e informales) que hace absolutamente imprescindible un enfoque multidisciplinario.
En otras palabras, en general es verdad que cualquier proceso de automatización nunca es el primero, sino más bien el último paso de una compleja actividad de revisión y de proyecto organizativo, que es misión específica de los cuadros directivos dirigirlo y controlarlo. Cuando el proceso se aplica a la oficina, este hecho todavía es más verdadero, por el número de las personas involucradas y por el profundo impacto sobre su modalidad de trabajo y, por tanto, sobre la propia estructura de los papeles ejecutivos, profesionales y directivos.
La automatización de la oficina se convierte así en una variable de la estrategia de gestión de la empresa, en unión de los recursos humanos y financieros a los que están estrechamente unidos, y el modo en que se afronta la planificación y la actuación puede condicionar la propia estrategia de supervivencia competitiva.
También emplearemos la denominación automatización de la oficina, *pero hablando de* informática en el trabajo de oficina *queremos subrayar dos aspectos que creemos determinantes para una correcta comprensión del tema.*
El primer aspecto se refiere al hecho de que, como la informática ya tiene una historia de 20 años de aplicaciones al trabajo de oficina (o mejor, al trabajo creador de empleo), desde el punto de vista tecnológico se trata de extender aquellas tareas que son sus áreas tradicionales de intervención. En segundo lugar, la denominación «automatización de oficina» resulta inadecuada para describir la complejidad del fenómeno que deseamos tratar, puesto que es evocador de los modelos de parcelación del trabajo de fábrica que, si se transfieren en un contexto absolutamente diferente, sea por objetivos como por estructura de los procesos de trabajo, pierden cualquier significado concreto. Esto no impide que continuemos adoptando esta denominación, por otra parte aceptada universalmente, para significar la actual situación de desarrollo del proceso de informatización del trabajo de oficina, proceso por otra parte en curso desde hace bastantes decenios en las sociedades occidentales. Las motivaciones que sustentan el crecimiento exponencial de la petición de automatización en el tratamiento de las informaciones pueden resumirse todas al valor que asume la propia información en el mundo industrial avanzado y, por tanto, a la importancia que su actividad de gestión reviste.
En realidad, en las economías occidentales (y en la japonesa) se está asistiendo desde hace tiempo a un progresivo desplazamiento de la fuerza laboral de la actividad de producción de bienes a la actividad de tratamiento de informaciones.
El gráfico de la pág. 809 representa este fenómeno tal como se manifiesta en las economías de los países adheridos a la Organización para la Cooperación y el Desarrollo Económico (OCDE) y muestra cómo en tales economías ya se ha superado la cuota del 50% de la ocupación por lo que respecta al trabajo de tratamiento de las informaciones.
Pero, si bien menos conocido no por eso menos interesante, este desplazamiento se está produciendo de manera fuertemente acelerada desde hace unos 20 años, en particular en el caso de Suecia, que ciertamente representa para Europa un modelo socioeconómico de referencia. Aunque no se conocen muy bien las estadísticas análogas referentes a otros países, es claro que en ellos se está frente a un proceso del todo comparable: la explosión de las actividades terciarias y la creciente atención prestada al movimiento de los cuadros son todos elementos que podemos comprobar.
Algunos economistas prevén en ese aspecto el nacimiento de un sector, el terciario avanzado, *en el que la mercancía de intercambio es la información, la competencia y el saber.*

Si ésta es la prospectiva hacia la que van las naciones industriales, tiene un preciso sentido preguntarse cuáles son, a nivel conjunto en primer lugar, y a nivel administrativo después, los componentes del costo de la actividad del tratamiento de las informaciones, su incidencia relativa sobre el costo total y su proceso correspondiente al tiempo.

Siempre con referencia a los países de la OCDE, la siguiente tabla muestra cómo las actuales estimaciones atribuyen un neto incremento anual del valor real (al neto de la inflación) al costo de personal y (recordémoslo, estamos razonando en paridad de prestaciones) un decremento aún más neto al de los aparatos de soporte al hombre para el tratamiento de las informaciones en las áreas del archivo, proceso y comunicación:

Personal	*+ 8%*	
Comunicaciones	*– 10%*	*por año*
Cálculo	*– 25%*	*en los próximos*
Memoria	*– 35%*	*diez años*

Este segundo hecho no debe asombrar. Efectivamente, debe considerarse que los tres tipos de aparatos pertenecen a la tecnología electrónica que ya sabemos, después de 16 años, que junto al proceso de miniaturización (es decir de contracción de las dimensiones a igualdad de funciones lógicas) tiene un proceso de reducción de los costos que se produce de modo proporcional.

Aquí puede preguntarse por qué esta drástica disminución de los costos tecnológicos encuentra una respuesta muy limitada en la moderada reducción del precio de las comunicaciones indicadas en la tabla (válida de forma media en los países OCDE) o precisamente por qué ésta se resuelve tal vez en incrementos como en el caso específico italiano.

Las razones de esta aparente anomalía son múltiples, pero en realidad todas pueden resumirse a la consideración de que en los países europeos está vigente el régimen de monopolio en las comunicaciones, y las tarifas se establecen no sólo en base al costo, sino en base a consideraciones políticas y de oportunidad social. También debe añadirse que, siempre en el caso de las comunicaciones, la tecnología electrónica todavía no ha eliminado del todo en el mercado la electromecánica y esta última presenta indudablemente costes de gestión bastante superiores. Particularmente en Italia, puede estimarse que actualmente menos del 5% de las centrales de conmutación son del tipo electrónico.

Estas simples consideraciones sobre el proceso del costo de los componentes correspondientes al tratamiento de las informaciones, conducen naturalmente a una primera importante conclusión.

Teniendo en cuenta lo dicho, puede calcularse que, en ausencia de cualquier forma de automatización (es decir, de adopción de las tecnologías de archivo, cálculo y comunicaciones electrónicas), en el transcurso de aproximadamente 8 años, el costo del sistema informativo quedaría multiplicado por 2.14, a paridad del volumen de información tratado.

En otras palabras, después de 8 años se haría exactamente lo mismo (al menos en cuanto a volumen) pero con costes más que duplicados. Si en cambio se introduce un índice de automatización, precisamente como el indicado entre el costo de la tecnología utilizada y el costo total del sistema informativo, se conseguiría limitar el crecimiento de los costos para llegar a anularlo en el caso en que el valor de dicho índice sea igual a 0.5, manteniendo así los costos sin variación en el tiempo.

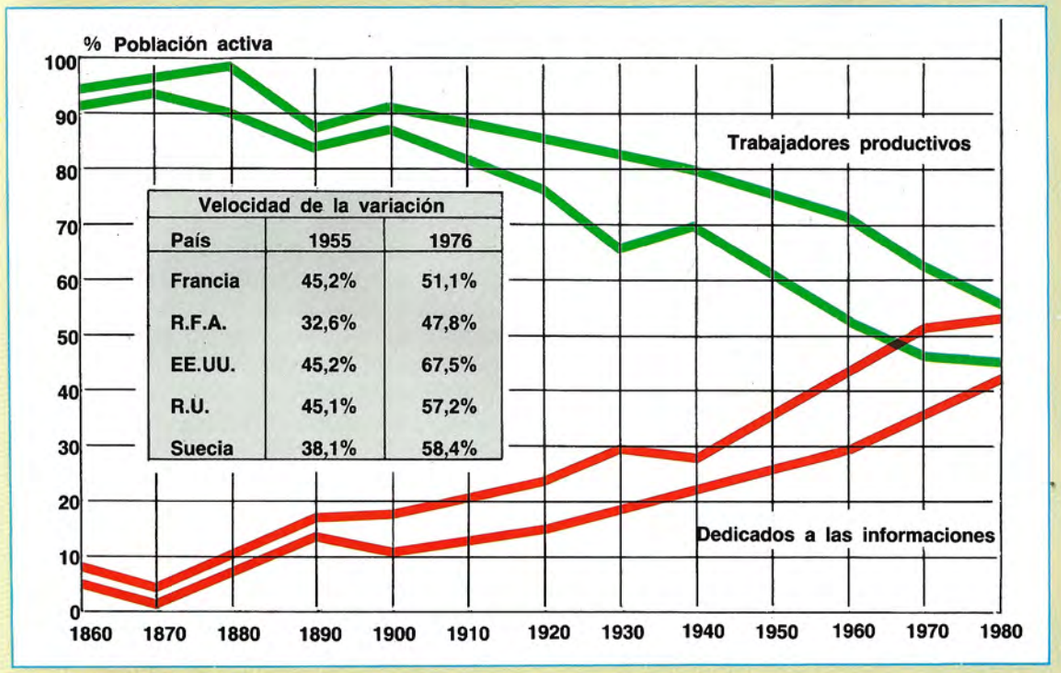
Para tasas de automatización superiores al 50%, se conseguiría precisamente reducir el coste del sistema informativo hasta un nivel mínimo igual al 0.16 del original si se llegase al límite del 100% de automatización.

Una tasa de automatización del 100% debe considerarse obviamente asintótica, incluso si valores del 80%-90% no son irrealizables en ciertos sectores industriales (piénsese en las fábricas de producción continua) o en ciertas áreas empresariales (piénsese en el ciclo integrado-pedidos-materiales-producción-proveedor en tantas empresas de producción discreta, o bien en la gestión en línea de las operaciones por ventanilla realizadas en un gran número de Institutos de Crédito).

Ya se ha comentado la dominante importancia que la información va asumiendo en el mundo industrial avanzado.

Este fenómeno se ha manifestado particularmente en los últimos 10-15 años, en los que se ha verificado un crecimiento exponencial de las exigencias informativas necesarias para gestionar una organización cualquiera, tanto desde el punto de vista cualitativo, como desde el cuantitativo.

Múltiples son las razones que pueden justifi-



Evolución del empleo en los sectores productivos y de la información. Las dos curvas definen más o menos restrictivamente la denominación «Dedicados a las informaciones».

car este proceso, destinado a acentuarse posteriormente en los próximos años, y que pueden hacerse realidad bien por la internacionalización de los mercados, bien por la difusa inestabilidad económica y política, por la variabilidad de la tasa de cambios como por la crisis de las fuentes energéticas tradicionales.

En realidad, si todas estas causas se sometiesen a un análisis suficientemente preciso, quedarían resumidas a una matriz conceptual única que podría expresarse en estos términos: la información resulta ser el recurso más conveniente para gestionar, en situaciones de incertidumbre, una organización cualquiera que tenga la finalidad de eficiencia económica y/o social.

Efectivamente, la alternativa a la información representada por el acopio de reservas (de materiales, de personal, de capital) movilizados al producirse el suceso imprevisto (e imprevisible, dado el contexto general de incertidumbre) para poderlo afrontar y superar. Pero esta estrategia, dados los actuales costos de inmovilización, en la práctica es perdedora, desde el punto de vista económico, con respecto a la de aportar un sistema informativo suficientemente puntual y flexible que permita a la organización adaptarse dinámicamente al contexto en que se encuentra.

A la luz de esta consideración, aparte de las presentadas anteriormente, surge un valor diferente. Efectivamente, no se trata ya de minimizar simplemente los costos de tratamiento de la información sino precisamente de desarrollar una capacidad informativa (que no puede valerse de la automatización como soporte esencial) para poder mantener o aumentar la competitividad de la organización afinando el proceso de decisión.

La productividad, en su acepción común, es una medida de la eficiencia definida como la cantidad de trabajo necesario para producir un determinado resultado, tanto si es cualitativo como cuantitativo (en el segundo caso, la definición necesita obviamente ser aclarada). Si el resultado mejora sin que sea necesario añadir trabajo, la productividad aumenta, así como aumenta si el mismo resultado puede obtenerse con menos trabajo.

Las estadísticas sobre la productividad de la oficina, o no existen, o son notablemente controvertidas, puesto que son obtenidas por

complemento, respecto a la productividad económica general, de las correspondientes a la actividad de producción de bienes. Efectivamente, si bien puede tener un sentido medir, según la definición clásica indicada anteriormente, la productividad del empleado a nivel ejecutivo (aunque sea con las oportunas y no inmediatas modificaciones), para la actividad de tipo directivo y profesional, tal definición da lugar a evidentes absurdos. ¿Vale más un responsable de sección si produce 10 documentos al día en lugar de 5, o bien pasa más tiempo en reuniones que otro? Evidentemente no, y de ahí la necesidad de referirse a medidas de tipo sustancialmente diferente; estas medidas sólo pueden referirse a las llamadas ***prestaciones****, es decir al grado de consecución de los objetivos.*

La planificación marketing de una empresa, sólo para poner un ejemplo, puede medirse por el costo, por la calidad y por la oportunidad de sus planes, y no por su capacidad de actuar en cooperación con los responsables comerciales.

De hecho, también para el trabajo ejecutivo, las prestaciones son las medidas por excelencia del rendimiento de una persona, de un grupo o de un sector empresarial.

En cierto sentido, las prestaciones son un concepto bastante más amplio que el de la productividad; mejorar las prestaciones significa obtener resultados mucho más eficaces, a los fines de los objetivos empresariales fijados, puesto que no sólo se limitan a aumentar la simple productividad.

En el trabajo ejecutivo quizá puede simplificarse el problema, limitándolo a la medida de la productividad cuantitativa, pero también en este caso no debe olvidarse que esta productividad no es un valor de por sí, sino que debe tener una finalidad adaptada a los objetivos de la empresa.

Actualmente, es el sector terciario que debe hacerse más eficiente y esto se logrará gracias al desarrollo de este sector cuya mercancía de intercambio, como ya se ha indicado anteriormente, está constituida por la información y la competencia, es decir por el sector que hoy empieza a denominarse como terciario avanzado *y del que la informática forma parte por definición.*

Se ha hablado de inversiones en automatización, pero a la luz de las consideraciones sobre la productividad (o sobre las prestaciones) hechas anteriormente, en el caso del tratamiento de las informaciones es necesario identificar por lo menos en términos generales las funciones que se quieren resolver prioritariamente.

El gráfico de la página siguiente muestra una distribución indicativa del costo del trabajo de oficina en una empresa del secundario evolucionado (por lo menos, en este caso, de la componente de los empleados) o del terciario.

Como puede comprobarse, poco más de un tercio del costo está concentrado en los papeles ejecutivos; mientras que casi dos tercios se refieren a profesionales y directivos.

Esta constatación sugiere inmediatamente algunas consideraciones. La primera, que la cuota relativamente minoritaria del costo del trabajo ejecutivo pueda interpretarse como el resultado de 20 años de inversión de tipo informático actuales en el sector: efectivamente, como es sabido, las aplicaciones corrientes de EDP se han concentrado históricamente en la automatización del trabajo de empleados de carácter repetitivo y, por tanto, debería conseguirse reducir su incidencia sobre los costos totales.

La segunda, que en el ámbito del trabajo ejecutivo, más particularmente al secretariado, la actividad del tratamiento de textos presenta una incidencia en el total de poco más del 1% de los costos. Esto significa que, si el objetivo de la automatización de la oficina es el de una significativa recuperación de productividad, no es cierto que actuando sobre la actividad de la mecanografía pueda pensarse en conseguirlo.

En cambio es necesario apuntar a la actividad profesional y directiva, que en conjunto constituyen casi los dos tercios de los costos totales, y sobre la cual, salvo casos particulares, la informática sólo ha incidido en una medida marginal.

Por otra parte, el tratamiento de textos tiene una indiscutible importancia como medio de introducción de las informaciones en el sistema de automatización de la oficina. Puede sostenerse que la mecanografía es a la automatización de la oficina como el data entry *es al tratamiento de datos tradicional.*

Pero esto no debe justificar, precisamente por las razones antes apuntadas, la confusión entre inversiones en automatización de la oficina e inversiones en sistemas de mecanografía, que o bien son contextuales, o pueden dar lugar a incompatibilidades insuperables y a fragmentaciones del trabajo absolutamente contraproducentes en términos de productividad y eficacia.

Identificados los diferentes papeles al que hay que dirigir prioritariamente la automatización de la oficina, se plantea el problema de cómo, gracias a ésta, pueden mejorarse sus prestaciones en el sentido indicado anteriormente.

A este fin es necesario profundizar en las características del trabajo de tratamiento de las informaciones, así como de sus modalidades de automatización.

El trabajo de oficina consiste en:

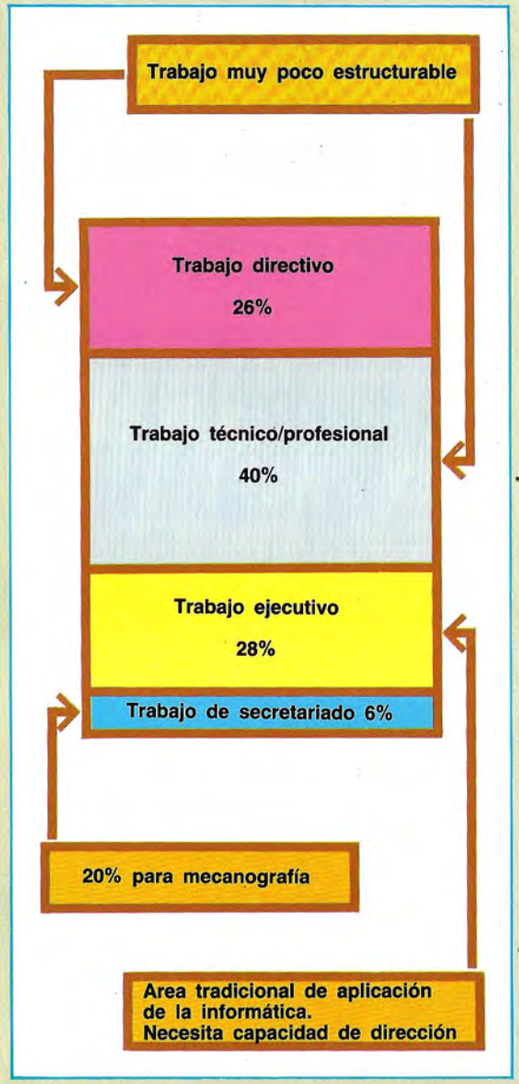
- *Funciones (contabilidad de clientes, gestión de pedidos, control de producción, etc.)*
- *Actividad (de tratamiento/intercambio de informaciones). Las funciones hacen referencia a la estructura empresarial.*
  Las actividades están en la base de las funciones y corresponden a la actividad individual.
  La Office Automation (OA) se orienta a la automatización de las actividades.
  El Data Processing (DP) es el de las funciones.

El desarrollo de las aplicaciones tradicionales de la informática ha actuado en el sentido de «procedurizar» las diversas funciones empresariales que se prestaban a este proceso (las funciones, es decir, las llamadas estructurables) y que, por esto, resultaban adecuadas a las funciones ejecutivas. A título de ejemplo, baste recordar, en el mundo industrial, las varias contabilidades, la gestión de pedidos, la de producción y la de compras, etc., mientras que en el sector bancario los diversos servicios operativos, desde las cuentas corrientes a las libretas de ahorro, a los títulos al portador, efectos, etc.

Al hacer esto, la informática ha considerado implícitamente como parte integrante de estas funciones las actividades elementales de archivo, búsqueda e intercambio de informaciones, incorporándolas así a los respectivos procedimientos y, por tanto, a los correspondientes programas en el procesador.

La automatización de tales actividades constituye precisamente uno de los objetivos prioritarios de la automatización de la oficina y uno de los ejemplos de diferenciación de la informática clásica. Otra consideración corresponde al destino del servicio automatizado.

En el caso de las funciones empresariales de tipo estructurado (es decir, el objeto de la informática) los destinatarios forman parte, en el ámbito de la estructura organizativa de la empresa, de grupos (y/o oficinas) homogéneos en cuanto a objetivos y a modalidades de trabajo (precisamente, el procedimiento de ejecución de las funciones empresariales específicas). En cambio, en el caso de las actividades elementales de tratamiento de las informaciones, precisamente por su carácter de soporte, el usuario puede serlo a título individual, si bien como perteneciente a un determinado sector operativo es. Por tanto, el empleo del instrumento de automatización puede convertirse en una elección profesional suya de oportunidades.



Costo del trabajo de oficina.

En otras palabras, en las aplicaciones informáticas hay implícita una componente de obligación al uso, bajo pena de falta de éxito en las propias aplicaciones. En la de automatización de la oficina, en cambio, no puede ser que el usuario se decida sobre la oportunidad de su empleo en función de la tarea que debe desarrollar y de su profesionalidad. Esto significa que la automatización de la oficina se considera en gran parte como un servicio de soporte para el desarrollo del trabajo, hoy no estructurable, del especialista y del dirigente, y es en esta acepción que puede contribuir significativamente, como profundizaremos a continuación, en la calidad de sus prestaciones.

Teniendo que esquematizar para claridad de exposición, puede afirmarse que los enfoques tradicionales a la automatización de la oficina pueden subdividirse en dos grandes categorías:

a) Enfoques que adaptan sistemas existentes del tipo multifuncional a las aplicaciones de oficina, dejando inalterada la interfaz EDP
b) Enfoques que insertan aparatos específicos para actividades específicas

A la primera categoría pertenecen las soluciones realizadas a través del desarrollo de un oportuno software sobre procesadores de gestión de tipo multifuncional de pequeña, media o gran dimensión. Este software puede utilizarse mediante terminales y, generalmente, proporciona la funcionalidad de tratamiento automático de las informaciones textuales, permite su archivado en memoria masiva y su puesta a disposición, a través del servicio llamado de «correo electrónico» (electronic mail), a todos los interesados que, siempre a través del terminal, se hayan dirigido para acceder a ellos para consultarlos o eventualmente modificarlos.

Como sobre este mismo procesador se ha realizado la automatización de los procedimientos informáticos de la empresa, este tipo de solución tiene el gran mérito de realizar la unificación, no sólo física sino también lógica, del mundo tradicional del procesador de datos con el de las informaciones de oficina; es decir, traducir en actos la complementariedad entre los dos ámbitos de aplicación que, a nuestro juicio (y no sólo el nuestro), es esencial para cualquier estrategia de desarrollo del sistema informativo empresarial.

Los límites de esta implantación están en su escasa posibilidad de difusión entre los usuarios que no aceptan la interfaz de comando característica del sistema de elaboración de los datos. Por otra parte, transformar esta interfaz de manera que sea de utilización más sencilla significaría añadir un peso excesivo a la carga conjunta de gestión del sistema, con la consiguiente reducción drástica de sus prestaciones.

Si se refleja el hecho de que las capacidades esenciales de la automatización de la oficina debe ser precisamente las de dirigirse a personal absolutamente privado de competencia informática, es posible darse cuenta del porqué esta solución sólo ha encontrado hasta ahora una modesta aceptación.

En cambio, a la segunda categoría pertenece todo el conjunto de aparatos de oficina que tiene por objeto hacer más eficiente ciertas actividades específicas de tratamiento de la información.

Así, por ejemplo, la máquina de escribir electrónica permite reducir el costo secretarial de elaboración de los textos, la fotocopiadora programable racionaliza el proceso de duplicación, el teléfono de teclado con memoria acelera la formación de los números y las llamadas que se utilizan con más frecuencia y así sucesivamente. Las observaciones que deben hacerse sobre esta implantación, cuando son en sí mismas y no encuadradas en un contexto más general, son de otro tipo.

Antes de nada, la recuperación de eficiencia que permiten estos dispositivos corresponden esencialmente al ámbito secretarial que, tal como se ha indicado anteriormente, incide en una medida relativamente modesta sobre los costos del trabajo de oficina (ver el gráfico de la pág. 811).

Pero, elemento mucho más importante, esta hipotética recuperación se obtendría posteriormente aceptando la fragmentación de las tareas que representa cada causa principal de la ineficiencia (y de la frustración) del trabajo de los empleados. En consecuencia, también si esta solución puede permitir una mayor eficiencia a nivel de la actividad sencilla, al nivel de la empresa está en profundo contraste con el objetivo prioritario de recomposición e integración de las tareas. Por tanto, se considera como conceptualmente errónea e incapaz de dar contribuciones que no sean marginales.

(extraido de «L'Informática nel avoro d'ufficio», de G. Occhini, CUADERNOS DE INFORMATICA, año X, núm. 1, 1983, Honeywell Information Systems Italia).

Las funciones del tablero electrónico

Los programas de gestión del tablero electrónico poseen numerosas funciones ya predispuestas para la realización de cálculos o de operaciones lógicas.

Las funciones numéricas constituyen una valiosa ayuda en la preparación de tableros que no tengan particular complejidad, mientras que las funciones lógicas permiten desarrollar aplicaciones también complejas. En algunas formas de tablero electrónico (por ejemplo en el Lotus) hay además previstas las **macroinstrucciones** (o macro), funciones definidas y escritas por el usuario que pueden ser activadas con la presión de una tecla predeterminada. Su estructura y su uso son muy similares a las de las subrutinas del Basic. Con el uso de las macro, el sistema es completamente programable, pero necesita notables conocimientos de base. Sea cual sea la forma de la función a utilizar, puede colocarse en el interior de una celda. Durante la fase de cálculo, en esta celda se presentará el valor numérico resultante de la realización de la función definida. En general, los argumentos de una función pueden estar constituidos por números o direcciones de otras celdas. En el primer caso, el cálculo de la función se realiza de forma inmediata sobre los valores de los argumentos, en el otro, el valor numérico se toma de la dirección (fila, columna) especificada. La sintaxis general para introducir una función es

@ xyz (n)

donde

@ es un símbolo de reconocimiento de función (dependiente del tipo de programa)
xyz es el nombre de la función
n es el argumento (o una serie de argumentos).

Por ejemplo, el cálculo del valor absoluto del número se realiza, en casi todas las versiones de tablero electrónico, con la función ABS. En este caso, la sintaxis se convierte en

@ABS (– 7 + 10)	de modo inmediato, sobre valor numérico
@ABS (+ D3)	indirecto, sobre el valor contenido en la celda D3

La variedad de las funciones disponibles depende del tipo de tablero electrónico, pero sin embargo existe un cierto conjunto de funciones fundamentales comunes a todas las versiones.

Funciones matemáticas

Las funciones matemáticas son muy similares a las utilizadas en el Basic; la única diferencia se debe al símbolo de reconocimiento que siempre debe preceder al nombre de la función. Con la convención de indicar con n un valor o una dirección, las principales funciones matemáticas son las siguientes:

ABS(n)	Extrae el valor absoluto
EXP(n)	Exponencial
INT(n)	Parte entera
LN(n)	Logaritmo natural (en base e)
LOG(n)	Logaritmo decimal
SQRT(n)	Raíz cuadrada
COS(n)	Coseno
ACOS(n)	Arco-coseno (función inversa del coseno)
SIN(n)	Seno
ASIN(n)	Arco-seno (función inversa del seno)
TAN(n)	Tangente
ATAN(n)	Arco-tangente (función inversa de la tangente)

El uso de estas funciones (que pueden ser tratadas como si se trabajase en ambiente Basic) no necesita ningún conocimiento particular del tablero electrónico. Por ejemplo, se desea escribir un programa que, dados los dos catetos de un triángulo rectángulo, calcule el valor de la hipotenusa utilizando el teorema de Pitágoras. Las funciones a realizar son:

— Introducción de los catetos
— Cálculo de la hipotenusa
— Impresión del valor de la hipotenusa

En Basic, el programa puede estructurarse de esta manera:

```
10 INPUT "Cateto A"; A
20 INPUT "Cateto B"; B
30 V = SQR (A^2 + B^2)
40 PRINT "Hipotenusa ="; V
50 END
```

Con el tablero electrónico puede escribirse el mismo programa, tal como se indica en la página 814, sin conocer las instrucciones Basic.

En las celdas A1, A2 y A3 se han introducido las descripciones correspondientes a las indicadas en las instrucciones 10, 20 y 40.
Las celdas B1 y B2 contendrán los datos (que corresponden a las variables A y B) y la celda B3 contiene la fórmula de cálculo.
En esta última celda se habrá presentado el resultado del cálculo; ésta constituye por tanto las dos líneas 30 y 40.

Funciones numéricas específicas

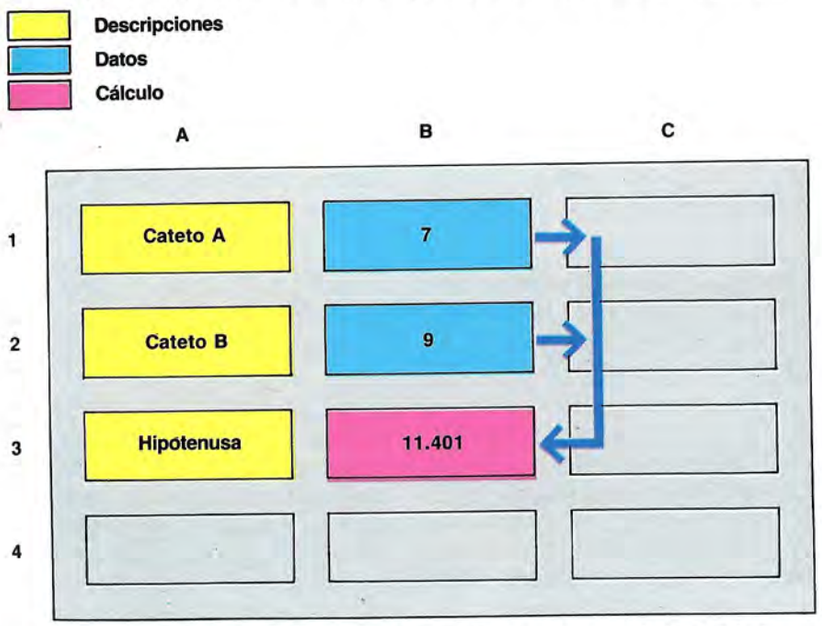
La principal aplicación de los tableros electrónicos corresponde a los cálculos de naturaleza económica o financiera, como por ejemplo previsiones de gastos, intereses, planes de inversión, modelización. Para resolver estos problemas específicos se han introducido una serie de funciones que no están previstas en el Basic (a menos que no se hayan definido como funciones).
Las principales se relacionan a continuación.

AVERAGE (n1, n2,...). Calcula el valor medio de los contenidos de las celdas especificadas como parámetros. Por ejemplo:

AVERAGE (+ B1, + B3, + C6)

restituye la media de los contenidos de B1, B3 y C6 (suma/3).
En el caso en que se deba calcular el valor medio de muchas celdas es muy incómodo tenerlas que indicar todas.
Existe una simbología más sencilla que permite definir un conjunto de valores (**rango**). No obstante, la utilización de un rango está subordinada a una adecuada colocación de los datos.
En el ejemplo anterior (B1, B3, C6), la dispersión de las localizaciones (se saltan tanto columnas como filas) impide la definición de un rango.
Para esto es necesario disponer de datos que estén todos ya sea en la misma fila, ya sea en la misma columna, incluso si en algún caso es posible distribuirlos sobre más filas o columnas contiguas.
Volviendo al ejemplo, si se sitúa el contenido de C6 en B2, los datos se encuentran en las posiciones B1, B2 y B3, es decir, alineados sobre la misma columna (B). En este caso, es posible definir un rango que cubra las direcciones de B1 a B3.

EJEMPLO DE APLICACION DEL TABLERO ELECTRONICO



Las celdas B1 y B2 están destinadas a albergar los datos (valores de los catetos).
En la celda B3 se ha introducido la fórmula de cálculo de la hipotenusa @SQRT (+B1∧2+B2∧2)

Int'l Stock Photo/Marka

El empleo de los sistemas de gestión contribuye de un modo determinante a racionalizar el trabajo de oficina.

La sintaxis más utilizada para definir un rango es

B1...B3

donde el primer valor es el principio del rango, los símbolos... significan «hasta» y el último valor es el final del rango.
Con esta simbología, la función de cálculo del valor medio se convierte en

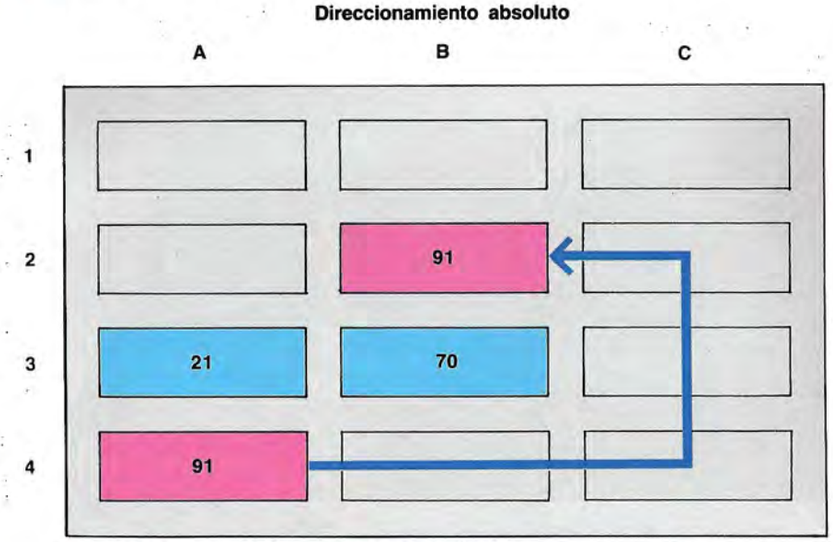
AVERAGE (B1...B3)

El uso del rango representa otra notable ventaja. En la definición de una función ya no es necesario escribir directamente las direcciones de la celda (de partida y de llegada): es suficiente situar el cursor en la celda que interesa y el sistema procede a escribir la dirección correcta. El uso de las direcciones necesita solamente una precaución: la de controlar si el sistema utiliza el modo absoluto o el modo relativo.
Indicando una celda cualquiera, por ejemplo B8, normalmente se entiende que se hace referencia a todo el contenido del tablero en el cruce de las columnas B y la fila 8, y éste es el método directo (o absoluto). Un segundo método de direccionamiento es el relativo.
Supongamos que el cursor ocupa en el tablero la posición B9. Con el direccionamiento relativo, indicar la localización B8 significa referirse a la celda que se encuentra en la misma columna (B) una fila más arriba (número de fila de 9 a 8). Análogamente, siempre con el cursor en B9, indicar C10 significa hacer referencia a la celda que se encuentra una columna más a la derecha (de B a C) y una fila más abajo (de 9 a 10).
Supongamos que la celda B9 contiene la fórmula para el cálculo de la suma de B8 y C8 (+ B8 + C8). Si el direccionamiento es absoluto, situando el contenido de la celda B9 en otra localización (por ejemplo E3), el cálculo permanece sin variación, puesto que el sistema toma los datos de las localizaciones ya especificadas. Y viceversa, si es relativo, se tienen resultados erróneos, puesto que el sistema modifica la fórmula de manera que ya no se refiere a la localización anterior (B8 y C8), sino a nuevas celdas que tienen con respecto a E3 la misma posición relativa que tenían las anteriores con respecto a B9.
La diferencia se ilustra en la pág. 816.

DIRECCIONAMIENTO ABSOLUTO Y RELATIVO

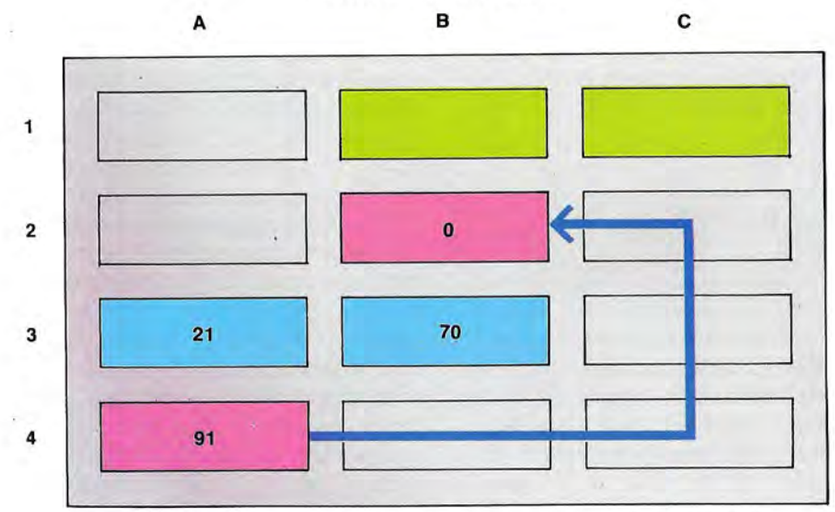
Celda que contiene la fórmula
Celda que contiene datos no válidos
Celda que contiene datos válidos

Direccionamiento absoluto



La celda A4 contiene la fórmula +A3+B3 y proporciona el resultado de la suma de los contenidos de las celdas A3 y B3
Si se coloca la fórmula en una cualquiera de las otras celdas, el resultado permanece invariable, puesto que las coordenadas de las celdas a que se refiere el cálculo continúan siendo las mismas.

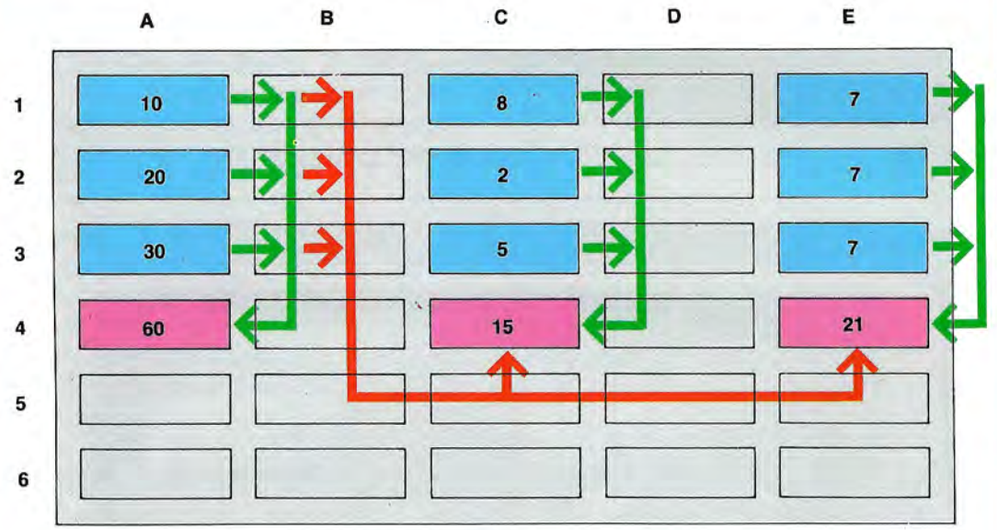
Direccionamiento relativo



Si el sistema adopta el direccionamiento relativo, el desplazamiento del cálculo en la celda B2 presenta un resultado erróneo.
A continuación del desplazamiento, el sistema ha modificado la fórmula al modo +B1+C1, puesto que las celdas B1 y C1 ocupan con respecto a B2 las mismas posiciones relativas que ocupaban las celdas A3 y B3 con respecto a A4.

APLICACION DEL DIRECCIONAMIENTO CORRESPONDIENTE A UNA SUMA POR COLUMNAS

Celdas que contienen los datos
Celdas en las que hay la fórmula
Flujo del cálculo erróneo (direccionamiento absoluto)
Flujo del cálculo correcto (direccionamiento relativo)



La celda A4 contiene la fórmula + A1 + A2 + A3, que produce la visualización de la suma de los datos de la columna A. Si el sistema implanta el direccionamiento relativo, desplazando el contenido de A4 primero a C4 y después a E4, la fórmula se adecúa automáticamente, y el resultado todavía es la suma de los datos encolumnados en C y en E. Si en cambio el tablero está gobernado por el sistema de direccionamiento absoluto, en las celdas C4 y E4 se visualiza todavía el valor 60, puesto que las direcciones siempre se refieren a los datos de la columna A.

Por tanto, el direccionamiento relativo tiene una utilidad no despreciable. Por ejemplo, considérese el tablero de arriba, en el que se piden algunas sumas por columna. Utilizando el direccionamiento relativo, si se coloca la celda A4 en C4 y en E4 el contenido (fórmula) se adecúa a la nueva situación, y el cálculo permanece válido. Con el direccionamiento absoluto, en las celdas C4 y E4 se tendría la misma fórmula de la celda A4, con resultados erróneos.

COUNT (NS..NE). Restituye el número de las celdas no vacías comprendidas en el rango definido por las direcciones NS y NE. En la pág. 818 se ha representado un ejemplo de uso de la función COUNT para la determinación del número de artículos presentes en un almacén. La fila 4, predispuesta para aceptar un nuevo artículo, no está incluida en el cálculo hasta que no se introduce la descripción del artículo.

SUM (NS..NE). En el ejemplo de la página siguiente se ha utilizado también la función SUM para obtener la suma de valores que aparecen en las diversas filas. El uso de esta última función es inmediato: SUM(C2..C5) equivale a + C2 + C3 + C4 + C5. Obsérvese que el uso conjunto de la función COUNT(...) y SUM(...) proporciona la AVERAGE(...):

$$\text{AVERAGE (A1...A9)} = \frac{\text{SUM(A1..A9)}}{\text{COUNT(A1..A9)}}$$

MAX (NS..NE), MIN (NS..NE). Extraen respectivamente el valor máximo y el mínimo obtenidos en el margen especificado.

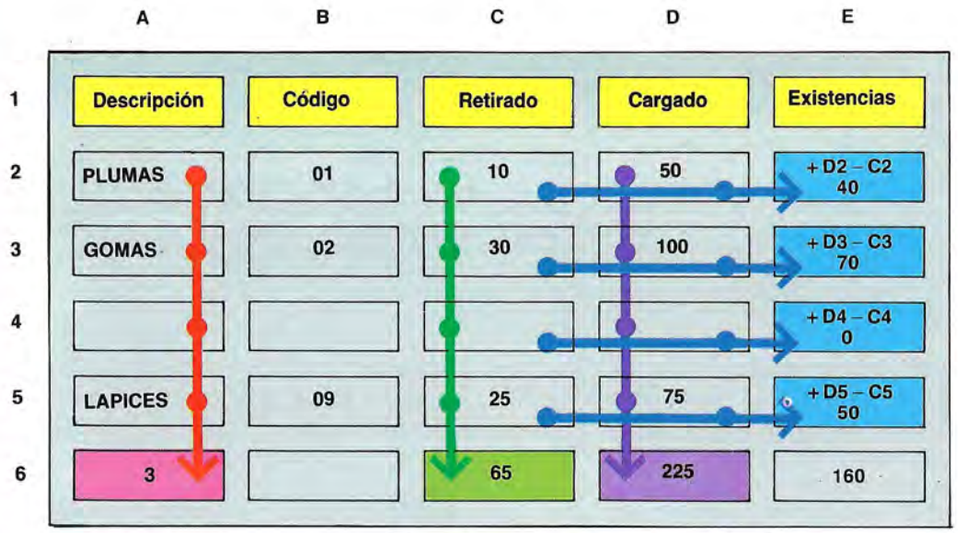
NPV (S,NS..NE). La sigla de esta función significa Net Present Value, y calcula el monto del capital inicial necesario para producir una cierta cifra final con una determinada tasa de interés. El parámetro S es la tasa de interés, y los valores NS..NE son las cifras que se desean obtener al final de cada período.
Por ejemplo, con una tasa del 20% (S = 0.2) y para un sólo período, queriendo obtener una cifra final (inicial más intereses) igual a 1500 debe invertirse un capital de 1250. Efectivamente, el rendimiento de esta cifra a la tasa del 20% es 250, que sumada al capital inicial da 1500.

Funciones de búsqueda

A esta categoría pertenecen todas aquellas funciones que permiten una selección de los datos. La variedad de las funciones de que se dispone depende mucho del tipo particular de tablero electrónico. En este capítulo ilustramos solamente la función principal LOOKUP, común a todos los programas.

LOOKUP (M,NS..NE). Busca el valor M en el rango NS..NE. El parámetro restituido depende del tipo de tablero utilizado. En algunos casos (Multiplan) es el valor de la última celda correspondiente a la columna o a la fila donde se ha

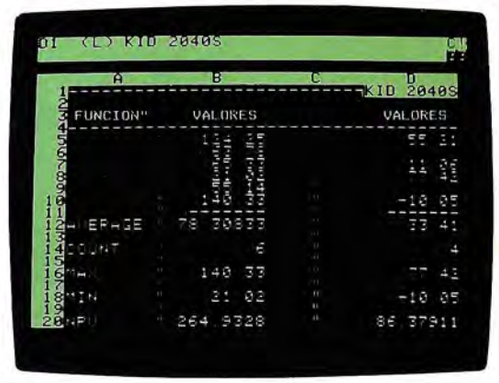
USO DE LAS FUNCIONES COUNT Y SUM EN LA GESTION DE UN ALMACEN

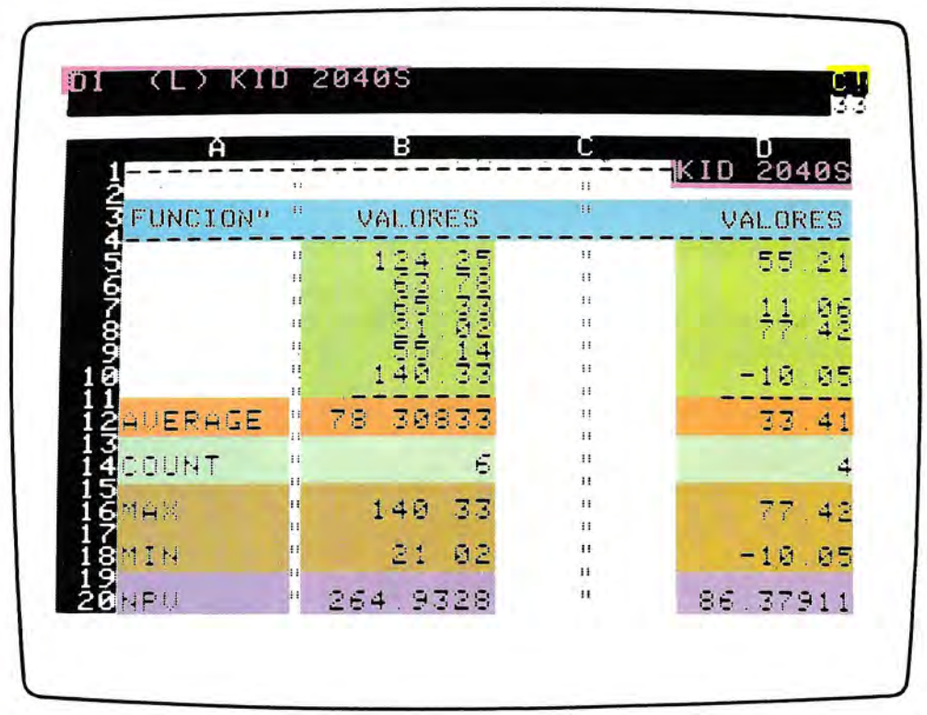


Insertando en la celda...	la función...	se visualiza...
A6	@COUNT (A2..A5)	el número de artículos diferentes que hay en el almacén
C6	@SUM (C2..C5)	el total retirado
D6	@SUM (D2..D5)	el total cargado
E6	@SUM (E2..E5)	el total de existencias

FUNCIONES DEL TABLERO ELECTRONICO

El vídeo muestra el cálculo de algunas funciones comunes a todos los tableros electrónicos.





Se pide el cálculo por columna.

La celda D1 contiene el nombre de la máquina utilizada (SIPREL KID 2040S). Para otro sistema, la función presentada, aunque conservando el mismo significado, puede tener denominaciones diferentes.

La fila 3 contiene la descripción de las columnas.

Las celdas de las columnas B y D contienen los datos utilizados para el cálculo de las funciones. Los valores introducidos no tienen un significado particular.

La función AVERAGE proporciona la media de los valores que se han aplicado. Así, en B14 aparece la media de los valores de la columna B y en D14 la media de los valores de la columna D.

La función COUNT proporciona el número de las celdas ocupadas por los valores numéricos por ambas columnas. Obsérvese que para la columna D, el valor es 4, puesto que las celdas vacías no se consideran.

Las funciones MAX y MIN calculan los valores máximos y mínimos que aparecen en cada una de las dos columnas.

La función NPV calcula el capital inicial necesario para producir un determinado capital final con una cierta tasa de interés.

encontrado M. En otros (Visicalc) es el valor de la columna o fila inmediatamente a la izquierda o encima de dicho punto.

Funciones lógicas

En los tableros electrónicos se han previsto algunas funciones lógicas que pueden utilizarse para las elecciones o para condicionar la realización de algunos cálculos. Su uso es muy similar al de las correspondientes instrucciones Basic. La variedad de las funciones lógicas disponibles depende del tablero que se utiliza. A continuación se presentarán las comunes.

AND (lista de expresiones). Restituye el valor TRUE (verdadero) si las condiciones indicadas en la lista se verifican todas, de otro modo, el resultado es FALSE (falso).
Las escrituras TRUE y FALSE aparecen en la celda donde está posicionada la función AND y tienen el significado de flags.
Por ejemplo, poniendo en la celda A7 la función

AND(A1>10,C3 = 5)

se tendrá en A7 la escritura TRUE si las relaciones entre ellos son verdaderas (A1>10 y C3 = 5), de otro modo se tendrá la escritura FALSE. La escritura que aparece en A7 se considera como cadena y puede utilizarse en otras funciones. Por ejemplo, insertando en la celda A11 la siguiente función

AND(A7 = TRUE,A9 = FALSE)

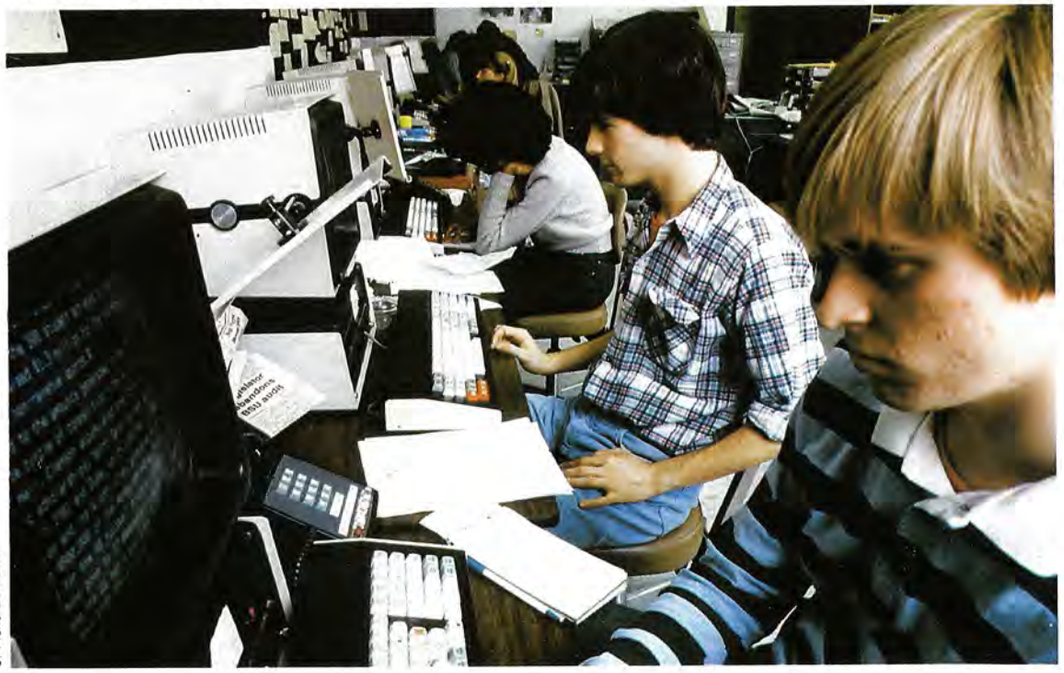
se obtiene el control simultáneo de las dos condiciones A7 = TRUE y A9 = FALSE que pueden depender de otras condiciones implantadas en las respectivas celdas (A7 y A9).
En la página siguiente se ha presentado un ejemplo de aplicación de la función AND.

IF (condición, verdadero, falso). Tiene el significado del IF Basic. Controlada la «condición» se ha desarrollado todo lo indicado en el campo «verdadero» o en el campo «falso» según que la «condición» se haya verificado o no.
Por ejemplo, poniendo en la celda B2 la función

IF(A7 = FALSE,A1 + C3,0)

a ésta se transfiere el resultado del cálculo A1 + C3 si A7 es falso (condición verdadera); de otro modo se coloca B2 = 0.

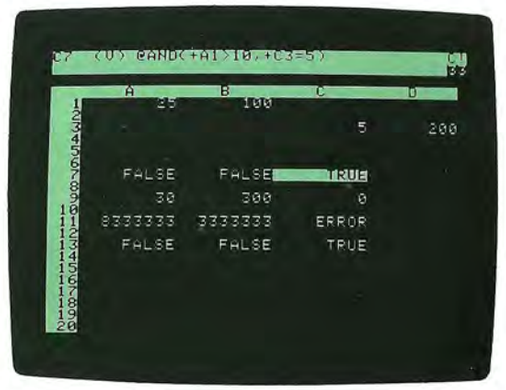
Un centro de proceso de la Universidad de Maryland (EE.UU.).

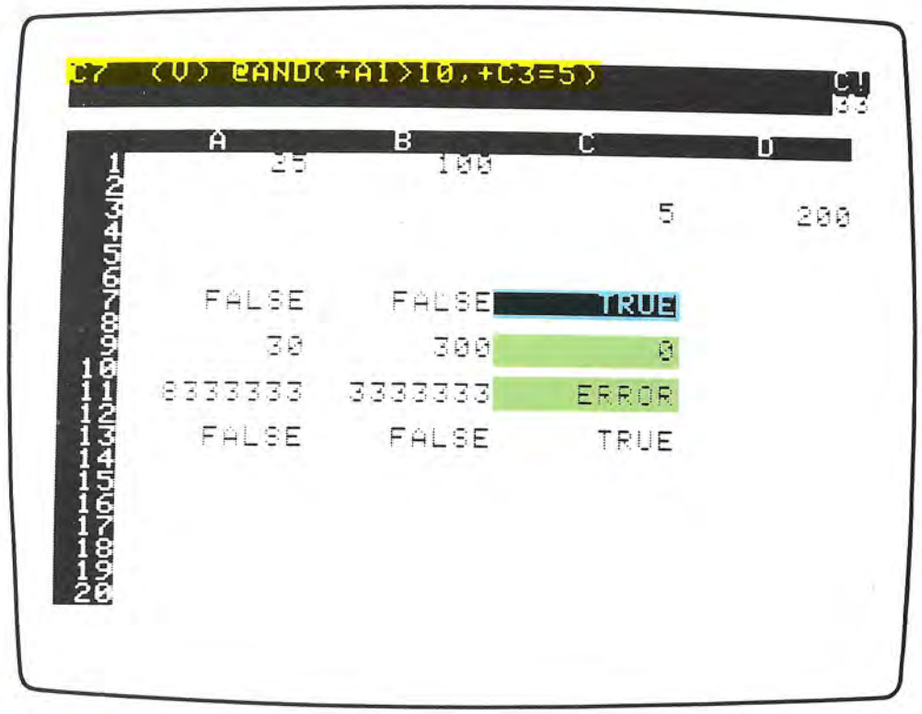


J. Pickerell/Marka

APLICACION DE LA FUNCION AND

La pantalla vídeo muestra un tablero en el que se han insertado algunas expresiones lógicas que utilizan la función AND.





En la celda C7 se ha introducido la fórmula @AND(+A1>10, +C3=5).

La respuesta es TRUE (verdadero), porque A1 = 25 y C3 = 5. La escritura TRUE no constituye el contenido original de la celda C7 (función lógica). Se trata de un flag de una sola presentación, que se actualiza al variar los datos en la celda interesada por la expresión lógica.

En la celda C11 se ilustra el uso del flag ERROR. En C11 se ha introducido una fórmula que utiliza como divisor el contenido de la celda C9. El sistema detecta una división por 0 y señala el error. Introduciendo en C9 un valor diferente de 0, la condición de error desaparece y el sistema presenta el resultado del cálculo.

Las funciones lógicas son muy útiles para realizar controles sobre datos introducidos o sobre resultados de las elaboraciones. A título de ejemplo, supongamos que se quieren procesar algunos datos estadísticos.
Los términos del problema son los siguientes.
Tres operadores han realizado encuestas del sexo sobre muestras de población. Cada uno ha comunicado el total de las personas examinadas y el número de varones que ha encontrado. Se quiere conocer el tanto por ciento de mujeres en cada una de las tres muestras examinadas y su parte porcentual sobre el total.
La preparación del tablero no presenta ninguna dificultad; sólo debe tenerse la previsión de insertar un control en las instrucciones de cálculo.
El número de personas pertenecientes al sexo femenino se halla como la diferencia entre el total y el número de varones:

$$\text{Mujeres} = \text{Total} - \text{Varones}$$

El porcentaje de mujeres sobre el total es:

$$P = \frac{\text{Mujeres}}{\text{Total}} \times 100 =$$

$$= \frac{\text{Total-Varones}}{\text{Total}} \times 100$$

Si en un muestreo han sido todos varones, el número de mujeres es 0 y también su porcentaje. En este caso puede evitarse el cálculo utilizando la función IF.
La implantación que debe darse al tablero electrónico para realizar los cálculos se indica en la página siguiente. Los datos se introducen en las celdas B2, B4, B6 (totales) y C2, C4, C6 (número de varones); en las correspondientes celdas de la columna D se han desarrollado los cálculos de los porcentajes de mujeres y en la fila 8 los totales generales.
El control sobre los datos se obtiene condicionando la celda de cálculo de los porcentajes de mujeres (D) a los valores contenidos en las correspondientes celdas de introducción (columnas B y C). En el ejemplo se ha indicado la IF en la forma utilizada para el cálculo del contenido de la celda D8. Para las otras celdas, la fórmula permanece formalmente sin variación, pero cambiando las direcciones. Por ejemplo, para la celda D4 (Operador 2), la fórmula se convierte en la siguiente:

IF(C4< >B4, 100 * ((B4 – C4)/B4),0)

NOT (condición). Restituye FALSE si la condición se produce y TRUE si no. Por ejemplo,

NOT(A3 = C6)

restituye en la celda que contiene la función el valor TRUE si A3 ≠ C6 y FALSE si A3 = C6. También en este caso, el contenido de la celda (TRUE, FALSE) puede utilizarse como flag para condicionar otros cálculos.

ISNA (celda). Asume el valor TRUE o FALSE según que en la celda especificada esté o no la función NA (Not Available = no disponible).
La función NA (sin argumentos) es un flag que puede utilizarse para bloquear los cálculos que se refieren a las celdas todavía no utilizadas.
Por ejemplo, sea A1 una celda de introducción de datos. Para condicionar un cálculo en presencia o no de valores en A1, en la celda se coloca la función NA escribiendo en A1 @NA (recuérdese que el símbolo @ tiene valor en el sistema utilizado, para otros sistemas puede ser diferente) y en otra celda (por ejemplo B1) la función ISNA (A1). Todos los cálculos que deben estar condicionados a la presencia de datos en A1 deberán escribirse controlando la condición TRUE/FALSE en B1; por ejemplo, en

IF(B1 = FALSE,.....)

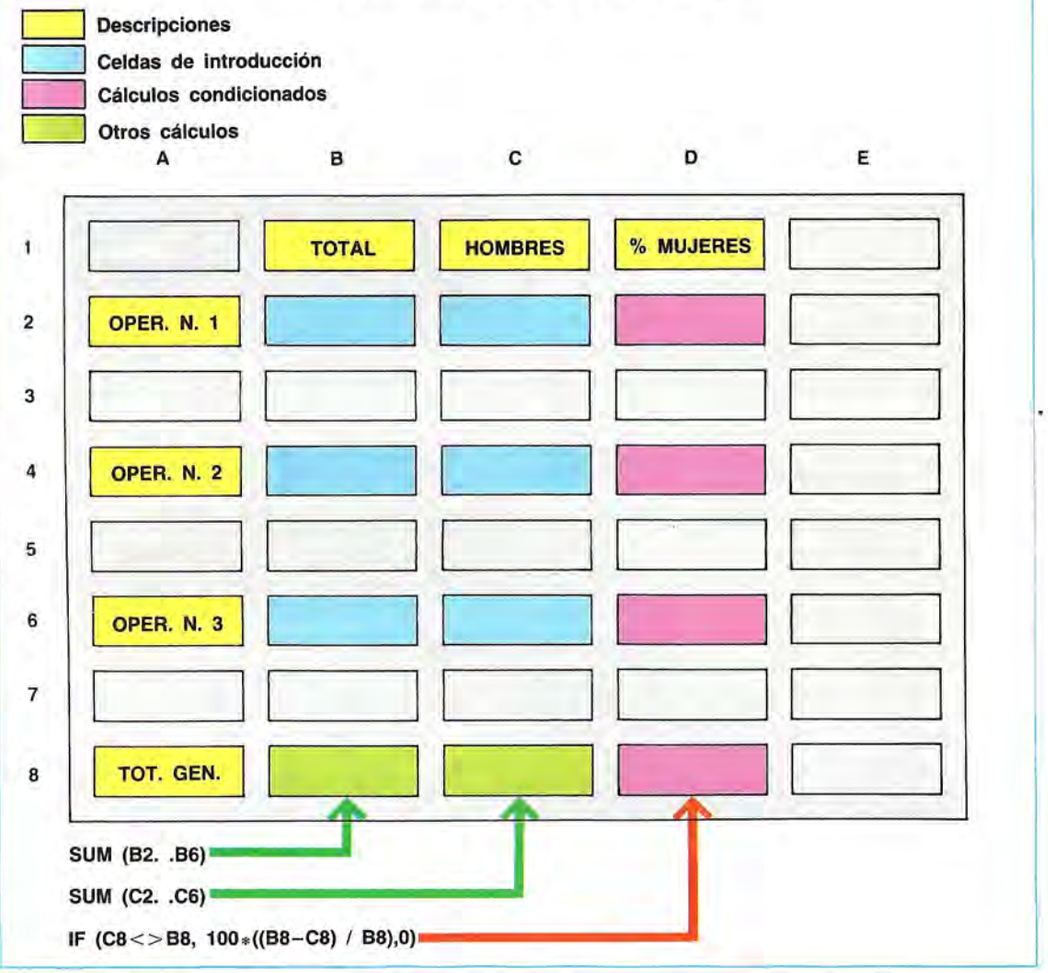
el contenido de B1 es TRUE hasta que en la celda A1 aparece NA. Introduciendo en A1 un dato cualquiera en la celda B1 se transfiere FALSE, y en todas las celdas que utilizan este flag se realizan los cálculos.

ISERROR (celda). Controla la producción o no de un error en la celda especificada. Por ejemplo, la función

ISERROR(B6)

restituye TRUE si en la celda B6 se ha producido un error, de otro modo da FALSE.
La señalización de error se tiene también con la escritura ERROR en la celda en que se produce el error y, por tanto, la situación anómala aparece evidente para el que está trabajando. Sin embargo, pueden tenerse aplicaciones con un número elevado de celdas activas. En estos casos se trabaja teniendo en la presentación sólo una parte del tablero, y un eventual error que interese a alguna de las celdas no presentadas no puede evidenciarse. El empleo de la función ISERROR (.....) permite tener bajo control lo que sucede en las zonas del tablero no visibles.

APLICACION DE LA FUNCION IF



OR (condiciones). Tiene el mismo significado que OR en Basic. Por ejemplo, escribiendo en la celda A3 la función

OR (A1 = 15, C3>D2, F4< >0)

se tiene A3 = TRUE al producirse una cualquiera de las condiciones relacionadas.

Funciones privadas de argumentos

Algunas funciones utilizadas en los tableros electrónicos restituyen constantes numéricas o lógicas. Por tanto, están privadas de argumentos. Las principales son:

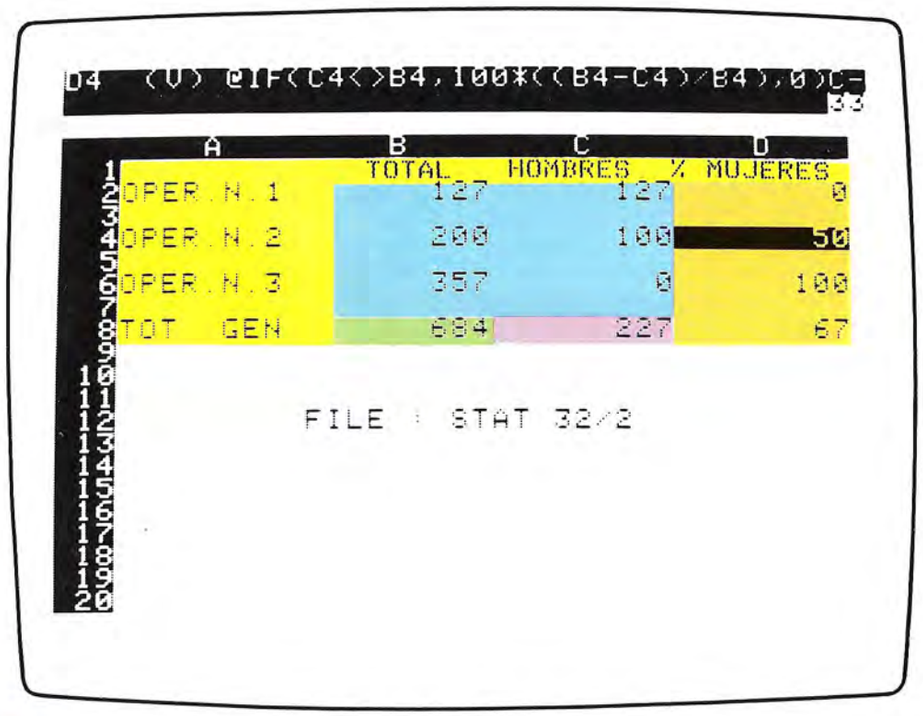
ERROR. Transfiere a todas las celdas que se refieren a esta función el flag ERROR, que aparece como escritura en la celda.

FALSE/TRUE. Implantan los correspondientes flags.

NA. Bloquea los cálculos en todas las celdas que se refieren a la celda que contiene la función NA. En estas celdas también aparece la escritura NA hasta la introducción de un valor en la celda que contenía inicialmente NA.

APLICACION DE LA FUNCION IF

El monitor presenta el tablero al que se ha hecho referencia en el ejemplo de aplicación de la función IF a cálculos estadísticos.



Las celdas de la columna A y de la fila 1 están reservadas a las descripciones.

Las celdas B2, B4, B6 y C2, C4, C6 contienen el total de las personas examinadas por cada operador y el total de los hombres encontrado.

La celda B8 contiene la función SUM (B2...B6), que activa el cálculo de la suma de los datos contenidos en las celdas de la columna B de B2 a B6.

La celda C8 contiene la función análoga para el cálculo de la suma de los datos de la columna C.

Las celdas de la columna D contienen el cálculo de los tantos por ciento de mujeres en base a los datos correspondientes a cada operador. El cálculo está condicionado y, para todas las celdas, la función implantada es análoga a la que se ve en la fila de las introducciones a la celda D4.

La función IF activa el cálculo de 100*((B4-C4)/B4) solamente si la condición C4<>B4 es verdadera. En caso contrario, transfiere a la celda el valor 0.

Ejemplos de aplicación del tablero electrónico

El tablero electrónico se emplea mucho en todos los paquetes de aplicación de gestión (Visicalc, Multiplan, Lotus, etc.) porque permite reunir de la forma más inmediata y accesible la solución de los problemas correspondientes a la contabilidad. Su estructuración permite al usuario tener constantemente ante sus ojos un cuadro completo del procedimiento de cálculo utilizado, y por muchos motivos recuerda a la mente las páginas de los libros contables tradicionales. Prácticamente, el usuario no debe hacer ningún esfuerzo para adaptarse a módulos ni a esquemas predeterminados: cada procedimiento puede proyectarse y estructurarse de forma natural y personalizada, sin que sea necesario adentrarse en los problemas típicos de la programación.

En las páginas siguientes se examinarán tres aplicaciones típicas del tablero electrónico. La primera corresponderá a la distribución milesimal de los gastos en un condominio; la segunda servirá para presentar la estructuración del tablero para la gestión de un almacén. En la última aplicación se hablará de un método de análisis de previsiones muy difundido en la práctica comercial: el WHAT IF (qué sucede si...).

Distribución de los gastos comunitarios

El ordenamiento jurídico de la mayor parte de los países europeos prevé que los gastos de gestión de una propiedad comunitaria deban repartirse entre los diversos condominios proporcionalmente al coeficiente milesimal de cada propiedad sencilla.

Los coeficientes milesimales son cifras que expresan (en ‰ = por mil) la parte alícuota de todo el condominio que representa cada propiedad individual. El valor del condominio se sitúa convencionalmente igual a mil milésimas. Una cuota milesimal igual a 100 milésimas atribuida, por ejemplo, a un apartamento incluido en el condominio significa que el valor del apartamento es igual a una décima parte (100/1000 = 1/10 = 10%) del valor completo del condominio: el propietario del apartamento deberá pagar, en consecuencia, el 10% de todos los gastos de mejora o mantenimiento de las partes comunes del condominio.

Montaje de las tarjetas en una línea de producción de hardware.

C. Pozzoni/Marka

DISTRIBUCION DE LOS GASTOS CON TABLAS MILESIMALES

Fórmulas — Funciones

Introducciones no recurrentes — Introducciones recurrentes

	A	B	C	D	E	F	G	H
1		DISTRIBUCIONES MILESIMALES			IMPORTES			TOTAL
2								
3	PISO	TAB.1	TAB.2	TAB.3	TAB. 1	TAB. 2	TAB. 3	
4	1	250	65	120	+B15*B4/1000	+B16*C4/1000	+B17*D4/1000	@SUM(E4..G4)
5	2	100	35	80	+B15*B5/1000	+B16*C5/1000	+B17*D5/1000	@SUM(E5..G5)
6	3	50	290	250	+B15*B6/1000	+B16*C6/1000	+B17*D6/1000	@SUM(E6..G6)
7	4	300	410	150	+B15*B7/1000	+B16*C7/1000	+B17*D7/1000	@SUM(E7..G7)
8	6	100	100	300	+B15*B8/1000	+B16*C8/1000	+B17*D8/1000	@SUM(E8..G8)
9	10	200	100	100	+B15*B9/1000	+B16*C9/1000	+B17*D9/1000	@SUM(E9..G9)
10								
11								
12								
13								
14								
15	Gasto 1	500						
16	Gasto 2	360						
17	Gasto 3	2500						

Arriba se ha representado en forma de tabla la estructura de un tablero para el cálculo de la distribución de los gastos comunitarios.
En la columna A (que será la columna A del tablero) se han indicado los números de los apartamentos de un mismo condominio.
En las columnas B, C, D se han indicado los coeficientes milesimales correspondientes a cada apartamento para tres tipos de gasto diferentes, en la previsión de que gastos diferentes pueden tener coeficientes de distribución milesimal diferentes. La tabla de distribución milesimal prevé, por tanto, tres distribuciones diferentes, pero puede plantearse como se desee.

La segunda parte del tablero (columna E, F, G) indica el cálculo de la cuota de participación de cada uno en un cierto gasto. La última columna es el total debido por cada uno, como suma de los tres parciales correspondientes.
La introducción de cada gasto se produce en las líneas 15, 16 y 17; la primera se referirá a la tabla 1 (columna B), la segunda a la 2 (columna C), etc.
La parte de gasto atribuida a cada condominio es el producto entre la cifra total y las milésimas de la tabla correspondientes a dicho gasto dividido por mil. La preparación de la tabla puede realizarse en dos etapas:

La automatización del trabajo de oficina (2)

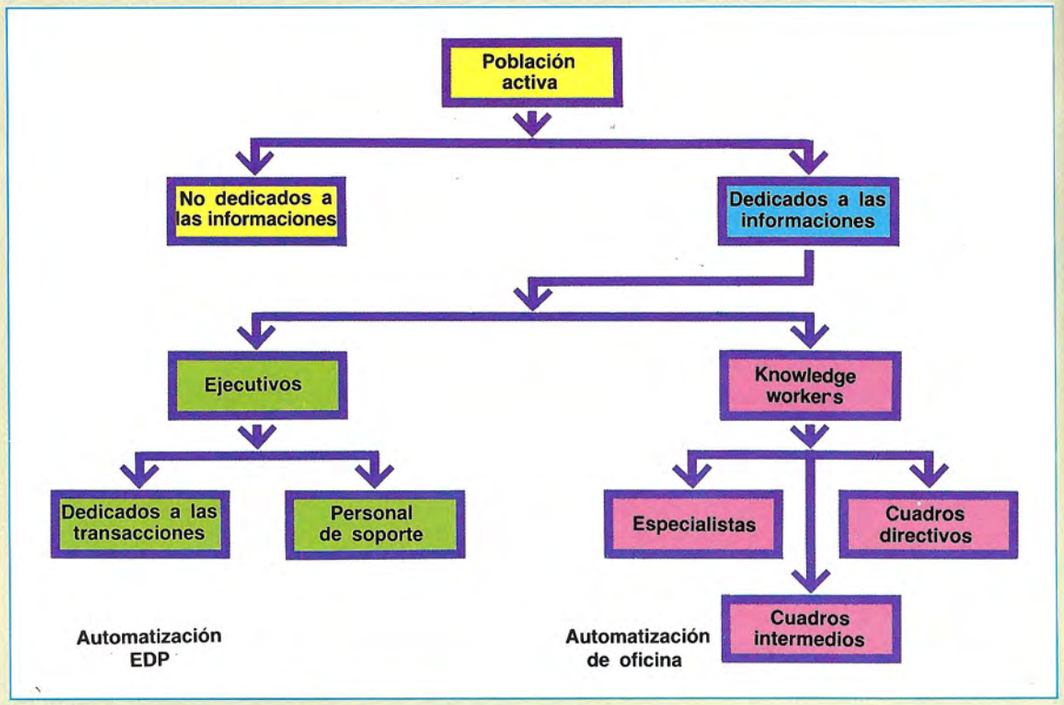
A la luz del análisis que se realiza aquí resulta claro como para la automatización de la oficina debe entenderse lo que se especifica en la siguiente definición:

- Modelo técnico organizativo que se refiere al empleo coordinado de instrumentos automáticos de proceso y comunicación de datos para proporcionar informaciones y servicios de soporte individual directamente al director, al profesional, así como al empleado ejecutivo.

- Comprende:
  El tratamiento de textos
  El correo electrónico
  La consulta de archivos
  El cálculo
  El soporte de decisiones
  El soporte para el desarrollo de las tareas personales (agenda, registro de vencimientos, etc.)
  Otras aplicaciones particulares eventuales

- Los servicios deben poder estar disponibles a través de un interfaz integrado que no necesite pasos intermedios.

Esta definición todavía sugiere algunas precisiones ulteriores.
El modelo ciertamente presenta una componente tecnológica que hoy en día lo hace, a diferencia de lo que sucedía hace pocos años, económicamente perseguible, pero sobre todo es de tipo organizativo, puesto que tiende a la racionalización del trabajo de oficina y, por tanto, a la profesionalidad de sus componentes. Esto significa que los aspectos de coordinación y asimilación deben asumir una importancia prioritaria en su aplicación.
Las funcionalidades relacionadas incluyen no sólo la actividad del tratamiento de informaciones que hemos llamado elementales, sino también actividades de carácter más evolucionado cuya automatización puede realizarse exclusivamente en términos individuales, según las características de la profesionalidad y del estilo del usuario.
Como máximo se trata de simples procedimientos de análisis y cálculo sobre datos contenidos en el archivo, orientados al soporte del proceso de decisión, a la planificación del trabajo y a la realización de las tareas personales.
Como consecuencia de las consideraciones realizadas, se subraya la necesidad de acceder a los diversos servicios a través de un interfaz unificado que no levante absurdas barreras entre el mundo del tratamiento de datos y el mundo de la oficina.
Finalmente, el destinatario de estos servicios debe poderlos utilizar de manera directa, sin intermediarios de ninguna clase, y éste es otro de los elementos que lo diferencia de la informática clásica.
Aquí es posible darse cuenta de esta exigencia observando el hecho de que la búsqueda de informaciones para el soporte de una actividad no estructurable, por definición no puede delegarse, puesto que la elección de dichas informaciones depende estrechamente de las modalidades individuales de realización de la tarea. En otras palabras, solamente quien sabe qué informaciones necesita (y por qué) puede seleccionar entre las informaciones disponibles las útiles, así como utilizar otras ante la eventual carencia de las informaciones buscadas.
La falta de procesos intermediarios característica de la informática clásica comporta una radical revisión de las modalidades de coloquio hombre-máquina, revisión que efectivamente está en curso, pero sobre la cual queda mucho por hacer también en términos de investigación teórica.
En particular, además de la lingüística, la rama de la informática conocida con el nombre de inteligencia artificial está destinada a encontrar en este área, junto con la psicología cognoscitiva, el terreno ideal de aplicación.
A continuación se proporcionarán algunos elementos de planificación de un proyecto de automatización de la oficina.
Como ya se ha indicado, se tendrán en cuenta tanto los aspectos de aplicación como los económico-financieros, en la convicción de que estos últimos deberán ser objeto de una particular atención por parte de los cuadros directivos, dadas las dimensiones de la inversión necesaria a medio y a largo plazo (a corto plazo la inversión también puede ser muy modesta si se limita a un experimento de alcance oportunamente reducido).
El esquema de la pág. 828, volviendo a las consideraciones presentadas, puntualiza los pape-



Distribución de las intervenciones de las actividades en una oficina.

les hacia los que se dirige el proceso de automatización en el sentido llano.
Entre los dedicados a las informaciones se distinguen los empleados ejecutivos de los llamados knowledge workers, *es decir, aquellas personas que en la empresa realizan una tarea que necesita un alto grado de profesionalidad, tanto de tipo especialista como de dirección.*
En el ámbito del trabajo ejecutivo pueden distinguirse posteriormente los dedicados a las transacciones, es decir, los que realizan determinados procedimientos de gestión del tipo que ya se ha indicado anteriormente, y el personal de soporte, constituido esencialmente por secretarias, mecanógrafas, archiveras, etc., que tienen la misión de hacer más productivo el trabajo de los knowledge workers.
Como ya se ha dicho más veces, el trabajo de los dedicados a las transacciones es sobre el que se ha insistido mayormente con la automatización a través de las aplicaciones de informática clásica, mientras que es sobre los otros papeles que se debería actuar prioritariamente con la automatización de la oficina.
Sobre la base de esta clasificación, la tabla que sigue proporciona la distribución del tiempo de trabajo diario (y por tanto del costo) de los especialistas y de los cuadros directivos a nivel intermedio y a nivel superior.

Actividad	Especialistas %	Cuadros intermedios %	Cuadros directivos %
Comunicaciones	38	54	64
Creación de documentos	15	11	8
Análisis	9	7	4
Lectura	4	8	6
Otras no directamente productivas	34	20	18

Las cinco actividades sobre las que se ha realizado la distribución terminan lógicamente, en su conjunto, con la totalidad de las funciones presentadas por los tres papeles indicados.
Efectivamente, en la actividad comunicaciones *se comprenden tanto las reuniones como las conversaciones telefónicas. En la* creación de los documentos *(o mensajes) se incluyen las anotaciones y las redistribuciones de documentos (o mensajes) recibidos visualmente.*

El análisis de datos *o* de situaciones *se entiende como finalizado a la toma de decisiones, mientras que la* lectura *tiene finalidades más generales de enriquecimiento de los conocimientos profesionales.*
Finalmente, las actividades no directamente productivas *forman parte de la espera de un acontecimiento, de la organización del trabajo propio o de los colaboradores, de la coordinación temporal de las actividades, de la búsqueda de informaciones o de personas (directamente o por teléfono), del archivo, de las transcripciones, etc.*
Teniendo en cuenta la población media en los tres papeles, es interesante observar, en la tabla de abajo, la forma en que la evaluación objetiva de la distribución del tiempo en las cinco actividades se aparta de la estimación que subjetivamente se ha hecho por los representantes de los propios papeles.

Actividad	% del tiempo total		% desviación
	Estimado	Efectivo medio	
Comunicaciones	29,1	45,6	−36
Creación de documentos	12,7	12,9	− 2
Análisis	14,6	8,2	+78
Lectura	9,1	7,9	+20
Otras no directamente productivas	34,5	25,4	+34

En particular, puede constatarse cómo la actividad comunicaciones *queda notablemente subestimada por los interesados, mientras que se sobredimensionan las de* análisis *y la denominación relativa a las* actividades no directamente productivas.
La desviación en el tiempo de comunicación se justifica con la inclinación natural a considerar el proceso de intercambios de informaciones más eficiente de lo que en realidad es.
En lo que respecta al análisis, es evidente la tentativa de dar mayor peso a una actividad que se considera cualificante en la empresa, mientras que para las tareas no productivas *puede comprobarse inmediatamente que la sobrevaloración deriva del juicio negativo que se tiene sobre ellas, incluso inconscientemente, porque se consideran de carácter operativo y no forman parte de la imagen del papel.*
En cada caso, volviendo a la distribución efectiva, se observa invariablemente cómo más del 70% del tiempo de trabajo de los papeles que estamos examinando se emplea en las actividades de comunicación y en las no directamente productivas.
Esta comprobación es suficiente para formular una primera y sencilla estrategia de intervención que, respecto a las otras, por lo menos tiene la ventaja de la claridad de los objetivos.
Esta primera intervención, si se realiza correctamente (es decir, con la oportuna gradación y las oportunas inversiones en formación, además de las de automatización), potencialmente puede aportar dos ventajas significativas a la organización del trabajo: la primera consistente en el enriquecimiento del personal de soporte y la segunda en el aligeramiento de la carga de trabajo. En lo que respecta a las actividades de comunicación, en el recuadro de abajo se establece una comparación entre la comunicación telefónica y la obtenible mediante el llamado correo electrónico, *que permite la transmisión de un mensaje (o de un documento) a uno o más destinatarios sin que se necesite la presencia inmediata de los interlocutores en los terminales. El sistema de correo electrónico puede memorizar las informaciones y ponerlas a disposición del destinatario cuando este último las pida.*

MENSAJE TELEFONICO:

1. Sólo el 26% se evade de la primera llamada
2. Del restante 74%:
   El receptor no puede ser molestado 38%
   La llamada no tiene respuesta 38%
   El número comunica 14%
   Otras causas 10%
3. No hay posibilidad de volver a pedir el mensaje (a menos de dictar-editar-archivar-buscar)
4. Dificultad de enviar el mismo mensaje a más usuarios
5. La «voz» no es económica
6. Las frases de cortesía no pueden eliminarse

MENSAJE ELECTRONICO:

1. Siempre tiene buen fin: no necesita la presencia del receptor
2. No interrumpe la actividad del receptor
3. Puede controlarse si se ha recibido
4. Puede reclamarse con operaciones sencillas
5. El envío del mensaje circular es sencillo
6. Reduce los tiempos de transmisión

Tiempo medio mensaje telefón. 4.8 min
Tiempo medio mensaje electr. 1.3 min
△3.5 min/mensaje

Como puede comprobarse, el correo electrónico presenta un conjunto de ventajas (esencialmente resumibles en la asincronía del intercambio debida a la capacidad de archivo) que, para toda una vasta gama de comunicaciones de oficina (de carácter operativo y técnico), la hacen, con mucho, preferible al teléfono.

Un profundo análisis realizado en el Stanford Research Institute lleva precisamente a prever que sólo el empleo sistemático del correo electrónico en la actividad directiva podía dar lugar a un ahorro de tiempo de orden del 20%.

En cada caso, a través del correo electrónico, se racionaliza el flujo de las comunicaciones que así puede disponer de tres canales de transporte diferente según el contenido de dichas comunicaciones. Los argumentos de carácter «político» continuarían siendo tratados en reuniones interpersonales: las que necesitan decisiones urgentes a través del teléfono, mientras que todas las otras podrían realizarse por el nuevo canal con una gran ventaja sobre la planificación del trabajo, donde las interrupciones y las pérdidas de tiempo en las comunicaciones adicionales llegan a ser de una hora y media sobre las ocho laborales.

La siguiente tabla proporciona una estimación cautelosa del factor de mejora de la productividad que la adopción de la estrategia indicada puede llevar a los papeles de especialistas y directivos.

Instrumento	Ahorro de tiempo potencial (en %)	% del ahorro total de tiempo
Teleconferencia	**0,5**	**2,7**
Transferencia de informaciones	**4,4**	**29,9**
Memorización y búsqueda de informaciones	**4,2**	**28,6**
Gestión personal	**4,0**	**27,2**
Gestión de actividades	**1,7**	**11,6**
	14,8	**100**

También se ponen en evidencia los instrumentos de aplicación que soportan esta estrategia, teniendo en cuenta la actual tecnología, instrumentos que en parte ya se han comentado en las consideraciones anteriores.

Precisamente, la transferencia de informaciones *corresponde específicamente al uso del correo electrónico; la* memorización y búsqueda de informaciones *corresponde a la posibilidad que tiene dicho sistema de almacenar y poner a disposición de quien tiene autorización las informaciones; la* gestión de actividades *representa todo lo que se ha comprendido en las actividades no directamente productivas y que se ha cubierto con los servicios denominados de agenda electrónica, de registro de previsiones, etc.*

Por elaboraciones personales *se entiende la posibilidad de dar al usuario (especialista, cuadro intermedio o directivo) una capacidad autónoma de crear simples secuencias de cálculos sobre los datos de su interés específico, siempre que, a su juicio, la repetibilidad de dichas secuencias lo justifique. Finalmente, con el término* teleconferencia *se quiere indicar un medio ulterior de comunicación que puede utilizarse con provecho, a medio y largo plazo, para el encuentro entre personas, con el teléfono y el correo electrónico.*

Se trata de la posibilidad de tener reuniones entre dos o más grupos de personas geográficamente distantes utilizando canales de comunicación audio, o también vídeo (en este caso, con frecuencia de dos órdenes de magnitud superiores a las de la voz y, por tanto, de costos todavía hoy prohibitivos). Esta posibilidad ha sido experimentada hace años y con éxito por la NASA, pero todavía no se ha difundido en la realidad empresarial europea y, además del costo, necesitará ciertamente un notable tiempo de adecuación (de ahí la exigua contribución que puede aportar hoy al ahorro de tiempo potencial que podría obtenerse).

Una observación conclusiva sobre el instrumento gestión de actividades *constituido por un soporte automático de gestión de las tareas, de las previsiones, de la planificación en el empleo de recursos comunes, etc., está en el hecho de que la contribución que este instrumento aporta a la mejora del trabajo especializado/directivo es relativamente modesto porque se evalúa sobre la hipótesis de que gran parte de las actividades no directamente productivas se sitúan según la estrategia sugerida al personal de soporte, a su vez dotado de todo lo necesario para mejorar la productividad y, por tanto, para poder absorberlo adecuadamente.*

Se ha demostrado la necesidad sin dilación de la opción informática en el campo de las actividades de tratamiento de informaciones de las sociedades industriales avanzadas. En este ámbito general se comentan diferencias y aspectos de continuidad de la automatización de la oficina con las aplicaciones clásicas del tratamiento de datos. Finalmente, se han propuesto estrategias de automatización de las que nos hemos preocupado de proporcionar una razonable justificación.

Voluntariamente no se ha afrontado el tema de la modelización de las actividades de oficina, tanto por el carácter introductivo de la presente panorámica, cuanto porque hoy día se considera prematuro afrontar el problema en estos términos al faltar todavía una visión orgánica del tema. Efectivamente, la estrategia de intervención sugerida prescinde de este aspecto, presentando como simple punto de arranque un análisis de los papeles destinatarios más aptos para la automatización.

Sobre estas bases, el proceso de planificación de una intervención en la oficina puede esquematizarse en las siguientes fases.

El modelo de partida está constituido por el universo de los usuarios potenciales del sistema de automatización de la oficina; de estos usuarios pueden definirse los perfiles de actividad con la ayuda de cuestionarios.

Después se identifican los instrumentos que potencialmente pueden mejorar las prestaciones en el sentido discutido muchas veces. Estos instrumentos están encuadrados (punto de fundamental importancia) en una adecuada visión sistemática que en la prospectiva debe conciliarse con la de la informática.

Dada la importancia de la intervención y del impacto sobre la organización del trabajo, el proyecto sometido a estudio debe ser parte integrante de los objetivos y de las estrategias de la empresa y, bastante más que en el caso de la informática clásica, debe ser dirigido y seguido en su desarrollo por la dirección. Esto en particular significa que oportunidad y recursos deben ir conciliados en el ámbito de la planificación empresarial a medio y largo plazo (3-5 años de horizonte temporal).

El plan financiero y el de adquisición de los productos se deducen del plan estratégico de automatización que la empresa ha elaborado.

Pero ¿a quién debe asignarse la responsabilidad de actuación en este plano? Una respuesta a esta pregunta se obtuvo de una encuesta realizada a finales de 1981 por la revista quincenal «Computer World» sobre 400 empresas.

El 50% de las empresas asigna la responsabilidad al sector de la informática empresarial, pensando que la competencia específica es un requisito previo esencial también para la automatización de la oficina; a favor de esta tesis hay la necesidad de la integración entre los dos mundos ya indicados.

El 20% constituye un adecuado comité de dirección en el que son representados, además del sector informático, todos los sectores interesados en el proyecto considerado, con el fin de facilitar la plena colaboración de la empresa en dicho proyecto.

En este ámbito es lícito pensar, aunque no se ha especificado en la encuesta, que la tarea de propuesta y la específicamente operativa han sido de entrada del sector informático.

Un 30%, finalmente, juzga el problema de la oficina un problema de costos y, por tanto, que pertenece a la administración. Sin embargo se recuerda que, incluso en Estados Unidos, a menudo el sector informático está encuadrado en la Dirección Administrativa.

Siempre con referencia a la misma encuesta, también es interesante observar la difusión de las aplicaciones de automatización de la oficina en la muestra de empresas elegida.

En este aspecto, los resultados muestran que sólo una exigua minoría (3%) todavía no había tomado en consideración el problema a finales de 1981: casi la totalidad de las empresas o ya tenía un proyecto, o lo estaba haciendo, o cuando menos estaba evaluando la oportunidad.

Aquí vuelve a ser útil repetir la observación ya hecha sobre la mayor cultura informática que puede encontrarse en Estados Unidos. Además, debido a las razones que ya se han indicado al inicio de esta exposición, es sobre el terreno del tratamiento de las informaciones que se medirá en el próximo futuro la competitividad del sistema económico y, por tanto, es sobre este terreno que Europa deberá competir con Estados Unidos (y, no lo olvidemos, con Japón) bajo pena de retroceder a posiciones irremediablemente subalternas.

Por tanto, es absolutamente indispensable que las decisiones se tomen con la máxima rapidez para recuperar el retraso y ponerse cuanto antes en condiciones de obtener la máxima ventaja de las experiencias ya maduras.

(extraído de «L'Informatica nel lavoro d'ufficio», de G. Occhini, QUADERNI DI INFORMATICA, año X, n. 1, 1983, Honeywell Information Systems Italia).

— Introducción y memorización en disco de las distribuciones milesimales y de las fórmulas. En este momento, los datos forman parte del archivo y pueden reclamarse para actualizaciones como modificaciones o impresión.
— Introducción de los gastos. El tablero anteriormente preparado se reclama a la memoria y cada nuevo gasto se introduce en el respectivo campo.

El programa procede automáticamente a realizar todos los cálculos y a presentar el resultado. Pidiendo la impresión se tiene la situación de cada uno y el encuadre.
El tablero, tal como está estructurado, no conserva memoria para la acumulación de los gastos. Por ejemplo, introduciendo un gasto en la línea 15 se tendrá el cálculo de todo lo que se debe por parte de cada uno, sólo válido para esta cifra; si en un segundo momento se produce un nuevo gasto, siempre inherente a la distribución 1, el tablero indica la deuda inherente a este nuevo gasto, y no el total (la deuda actual más la deuda anterior). Por tanto, debe insertarse una nueva columna que acumule las sucesivas introducciones y presente el total general para cada condominio.
Una solución puede ser la de memorizar por separado la columna de los totales para después reclamarla en un tablero sucesivo que las sume con los resultados anteriores.
La memorización de los datos se obtiene transfiriéndolos a un file con comandos similares a los utilizados por todo el tablero (SAVE, LOAD, etc.), con los caracteres más adecuados que indiquen la naturaleza particular de dicho file (es un file de datos y no un tablero; no puede contener fórmulas).
Por ejemplo, el símbolo # en algunos tableros indica que se quiere memorizar datos. Los parámetros a proporcionar entonces son el nombre del file (datos) y la fila o columna a transferir. Otros tableros sucesivos pueden utilizar los valores memorizados así cargándolos en memoria con la misma opción (por ejemplo, el comando L = LOAD, seguido del símbolo #).
Sin embargo, debe prestarse mucha atención a los valores que se transfieren. En este ejemplo, la columna H, también si se presenta como una serie de números (suma de los importes parciales), no contiene en realidad valores numéricos (es decir, datos), sino las fórmulas necesarias para su cálculo.

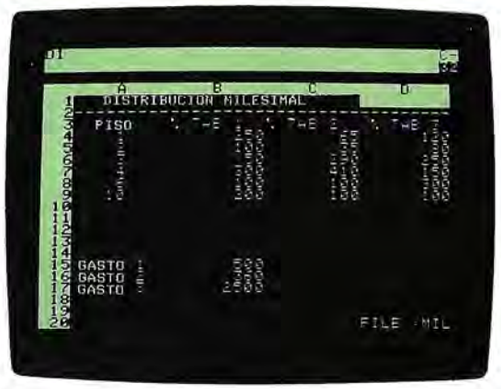
Memorizando esta columna como si estuviese construida por datos es posible llegar a tener algunos errores.
Los casos posibles son dos: o el programa no transfiere nada, puesto que reconoce la presencia de fórmulas, o bien transfiere las fórmulas, y cargando esta columna en el nuevo tablero se presentarían valores erróneos, ya que las fórmulas se aplicarían de forma incontrolada.
Para superar este obstáculo es suficiente con crear una nueva columna que contenga la anterior. La transferencia se produce a nivel numérico y se obtiene una columna que contiene los resultados del cálculo y no las fórmulas.
Por ejemplo, en HI hay contenido el cálculo @ SUM (E4...G4), es decir, la suma de los valores comprendidos entre E4 y G4.
El resultado es un valor numérico, y como tal se presenta en la celda HI, pero el contenido real de la celda es una fórmula. Transfiriendo esta celda a un nuevo tablero (siempre que el programa acepte una transferencia de fórmulas) el resultado ya no será el deseado.
Para copiar el contenido numérico de la celda HI (donde se realiza el cálculo) en primer lugar debe transferirse el valor resultante a otra celda del mismo tablero y, por tanto, memorizar esta nueva celda en formato de datos. El valor numérico contenido en ella se hace así accesible a otros tableros. La transferencia del valor numérico en el ámbito del mismo tablero no puede producirse con el comando de copia, puesto que este comando transferiría el comando de la celda, es decir, la fórmula y no el resultado.
Para transferir el resultado es suficiente con escribir en la celda de llegada la dirección (las coordenadas) de la celda de partida. Por ejemplo, escribiendo la dirección +HI en la celda LI se obtiene la transferencia del valor contenido en HI a la celda LI.
El símbolo + que precede a HI es necesario para informar al programa que se trata de una dirección y no de una serie de caracteres; omitiéndolo, en la celda LI se transferiría la escritura HI sin ningún significado de comando.

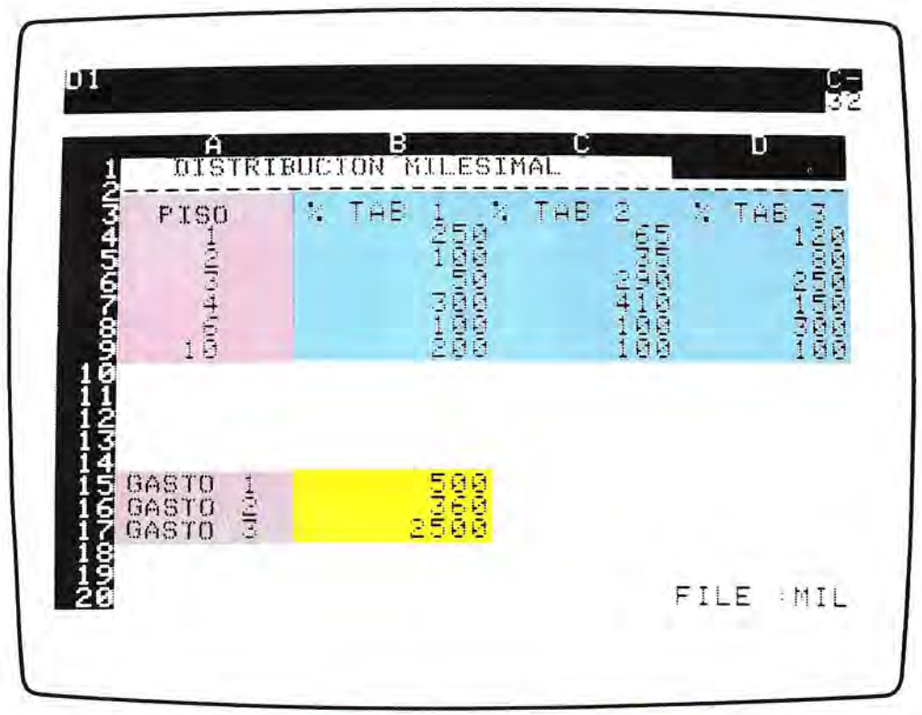
Gestión de un almacén

Aprovechando la notable capacidad de memoria de los modernos ordenadores personales, es posible realizar aplicaciones complejas incluso con máquinas de costo muy limitado.
En consecuencia, también para el empleo de los tableros electrónicos es necesaria una fase

DISTRIBUCION MILESIMAL DE GASTOS COMUNITARIOS (1)

En esta página y en las siguientes se muestra el empleo del tablero electrónico para el cálculo de la distribución de gastos comunitarios sobre la base de las tablas milesimales.





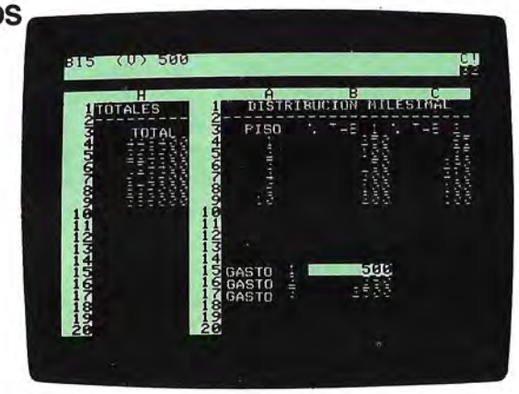
La columna A contiene una serie de descripciones (número de piso de cada condominio e indicaciones de gastos) que no van a variar. Por tanto, resulta más cómodo definirla como título (TITLE) vertical.
Si han de aportarse modificaciones al contenido de la columna A deberá eliminarse la opción TITLE, modificado el contenido de una o más celdas y recuperar la opción.

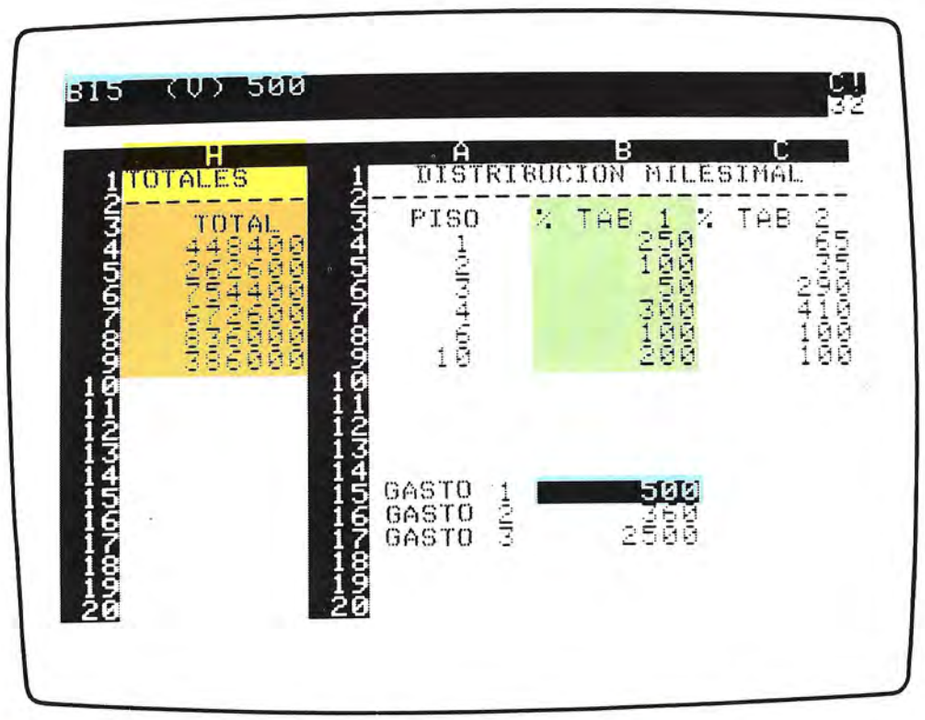
Las columnas B, C, D contienen los coeficientes milesimales, expresados en tanto por mil. En la descripción se ha adoptado el símbolo % (porcentaje) porque el símbolo exacto (‰) no está incluido en el teclado USA-ASCII.
Cada una de las tres columnas contiene una distribución diferente, inherente a un particular índice de gasto. Las tablas milesimales no van a variar y, por tanto, deben introducirse una vez por todas.

Las celdas B15, B16, B17 se reservan para la introducción de las sumas a repartir. Después de la introducción de un valor en una de ellas se activa el cálculo de la distribución en base a la tabla milesimal correspondiente.
La representación de los importes calculados necesita el scroll (deslizamiento) lateral del tablero, que puede activarse colocando a la derecha el cursor más allá del límite de la zona presentada.

DISTRIBUCION MILESIMAL DE LOS GASTOS COMUNITARIOS (2)

El vídeo ilustra el empleo de una opción particular que permite crear en el tablero electrónico una ventana en la presentación.





La columna H, que aparece como primera columna a la izquierda, en realidad está posicionada en el extremo derecho del tablero. Utilizando el comando W (WINDOW = ventana) pueden obtenerse en la presentación simultánea dos zonas del tablero físicamente distantes entre sí, de manera que puedan compararse directamente.

El cursor, posicionado en B15, produce la introducción del importe correspondiente al GASTO 1 presentado en la fila de las introducciones como valor numérico (V).

Para cada piso se calcula automáticamente la cuota de gastos en base a la tabla 1 (columna B).

Las cuotas calculadas se acumulan en los totales para cada piso.

Globe Photos/Marka

Operadora en la unidad de cinta de un centro EDP. A la izquierda pueden verse una serie de disk-packs.

de análisis del problema que conduzca a una implantación adecuada. El objetivo de la fase de análisis ya no es el de preparar los diagramas de flujo, sino determinar la mejor estructuración para un empleo racional de las posibilidades del programa.
Por ejemplo, la gestión de un almacén puede consistir en una simple relación de los artículos y de las cantidades en existencia, o puede estructurarse de modo que permita el control de las reposiciones, las simulaciones, etc.
El conjunto de las funciones a realizar puede dividirse en los siguientes grupos:

— **Parte descriptiva.** Deberá contener los datos característicos de cada artículo, como código, tiempo de aprovisionamiento, costo unitario.
— **Movimientos.** Proporciona la situación contable del almacén. Los datos tratados son los siguientes: acumulación de salidas, acumulación de entradas, existencias.
— **Valoración.** Contiene los datos anteriores valorados según los costos unitarios contenidos en la parte descriptiva.
— **Costos de producción.** Si los componentes del almacén se utilizan para la producción de productos acabados, para cada uno de estos productos puede memorizarse la cantidad necesaria de cada componente sencillo, para tener así el cálculo automático del costo de cada producto como suma de los costes de los componentes sencillos.

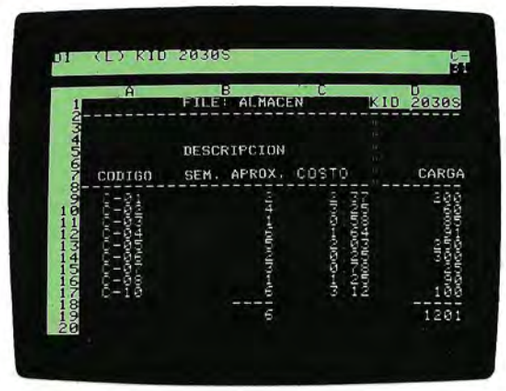
Un ejemplo de aplicación se ilustra en las págs. 836-840. Para conservar la rapidez del contacto con el vídeo se ha preferido ilustrar el ejemplo con una secuencia fotográfica de lo que aparece en el monitor. Sin embargo, siempre existe la posibilidad de producir copia sobre papel de todo lo que presenta el vídeo. El procedimiento que permite dirigir a la impresora el contenido del vídeo se ha previsto en general en el mismo programa de gestión del tablero electrónico.
En la pág. 840 se presenta un ejemplo de salida sobre impresora.

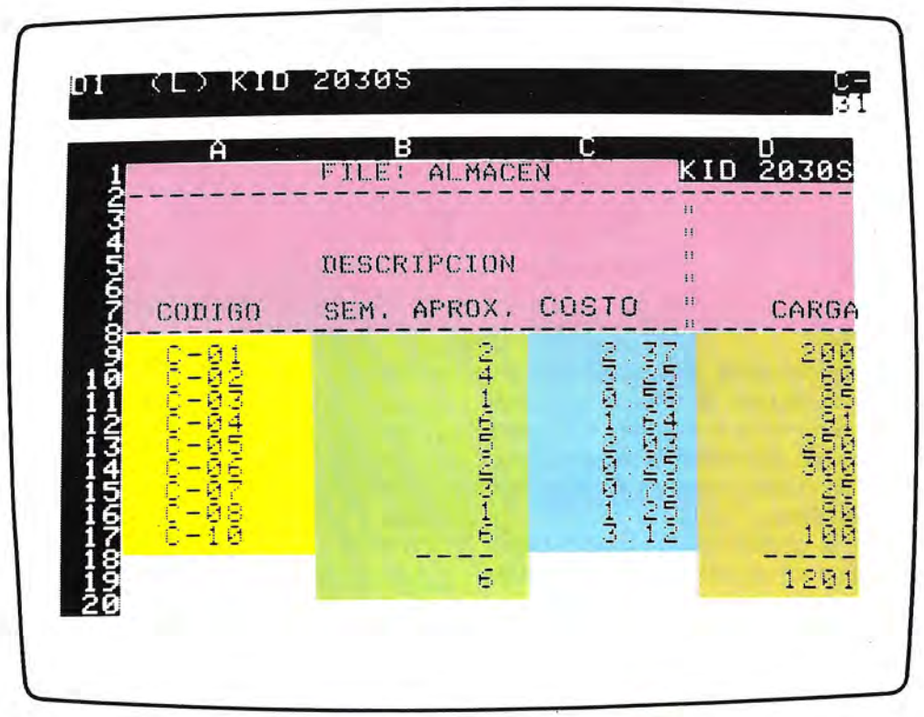
Análisis de las variaciones (WHAT IF)

Con este término se indica un tipo particular de análisis, realizado sobre fenómenos de carácter

GESTION DE UN ALMACEN: DESCRIPCION

En el monitor se ve la parte descriptiva de un tablero de gestión de un almacén.





Las filas 1 a 8 contienen los títulos (TITLE).

Cada artículo está identificado en base a un número de código; los códigos se indican en las celdas de la columna A.

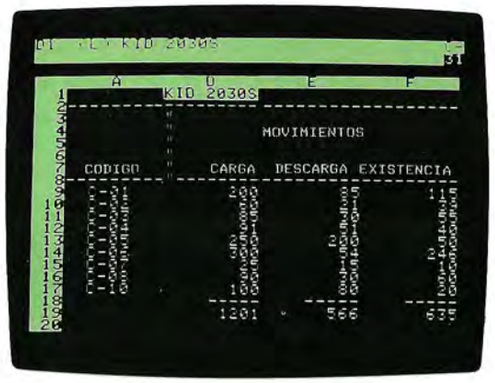
En la columna B se presentan los períodos de aprovisionamiento de cada artículo (en semanas). La celda B19 contiene la función @MAX (B9,..., B17), que calcula el máximo entre los valores contenidos en las celdas de la columna B. El período de aprovisionamiento de 6 semanas es común a los dos artículos C-04 y C-10.

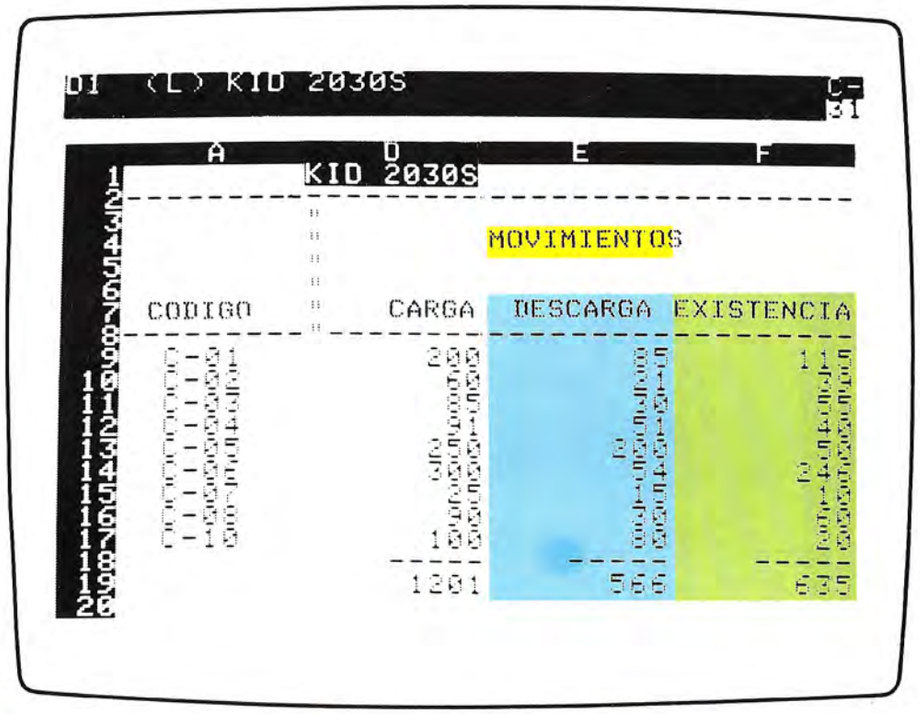
En la columna C se han introducido los costos de los diversos artículos.

La columna D forma parte de la sección siguiente del tablero (MOVIMIENTOS).

GESTION DE UN ALMACEN: MOVIMIENTOS

Activando el scroll lateral del tablero puede presentarse la zona de los movimientos.





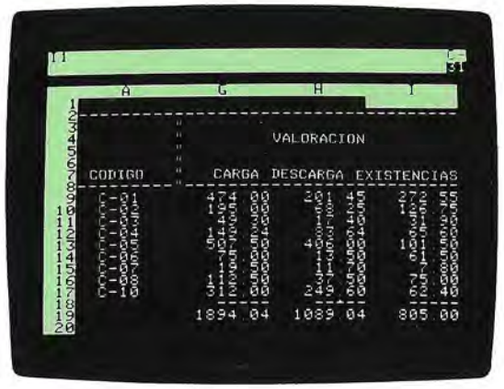
La parte de movimientos ocupa una zona del tablero distante de la columna A, que contiene los códigos de los artículos. Para tener al mismo tiempo en presentación la descripción y los movimientos es necesario definir la columna A como título (TITLE) vertical.

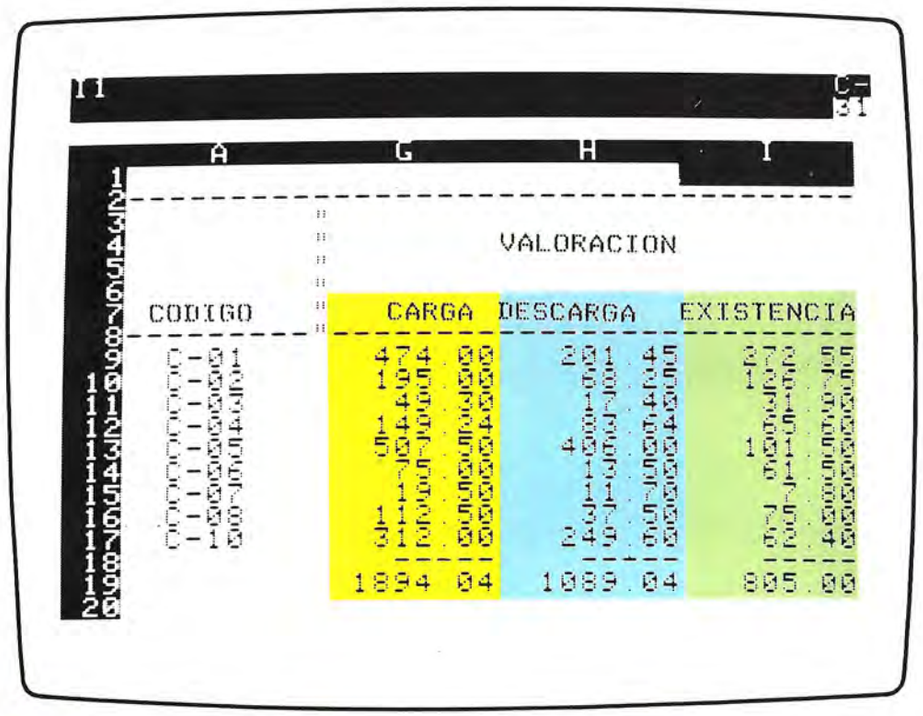
Introduciendo un dato correspondiente a un movimiento de descarga (columna E) se tiene el recálculo en E19 de la cantidad total descargada del almacén [la celda E19 contiene la solución SUM (E9...E17)]. Recordamos que se ha implantado el cálculo por columnas.

Inmediatamente después se calcula la existencia del artículo interesado a partir de la operación de descarga introducida (columna F). Por ejemplo, si se ha descargado un determinado número de artículos C-01, la celda F9 se recalcula como diferencia entre la carga D9 y la descarga E9.

GESTION DE UN ALMACEN: VALORACIONES

En la zona de las valoraciones para cada artículo se han presentado los valores acumulados de carga, descarga y existencia. En la fila 19 se han presentado los totales.





Las valoraciones de carga (columna G) se obtienen como producto entre la cantidad de carga de un determinado artículo (zona de movimientos, columna D) y el costo unitario (zona descriptiva, columna C). Por ejemplo, el dato 474.00 que aparece en G9 se obtiene con el cálculo +C9*D9.

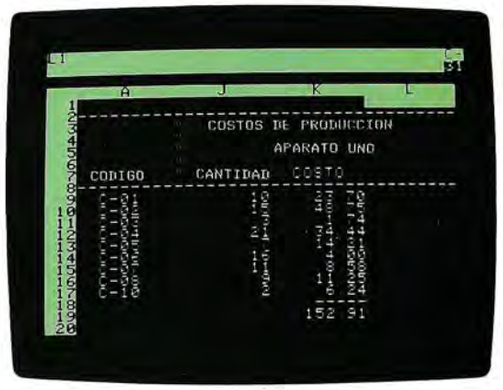
Los valores de acumulación de descarga (columna H) se obtienen como producto entre la cantidad de descarga de un determinado artículo (movimientos, columna E) y el costo unitario (descripción, columna D). El dato 201.45 en H9 viene dado por la fórmula +C9*E9.

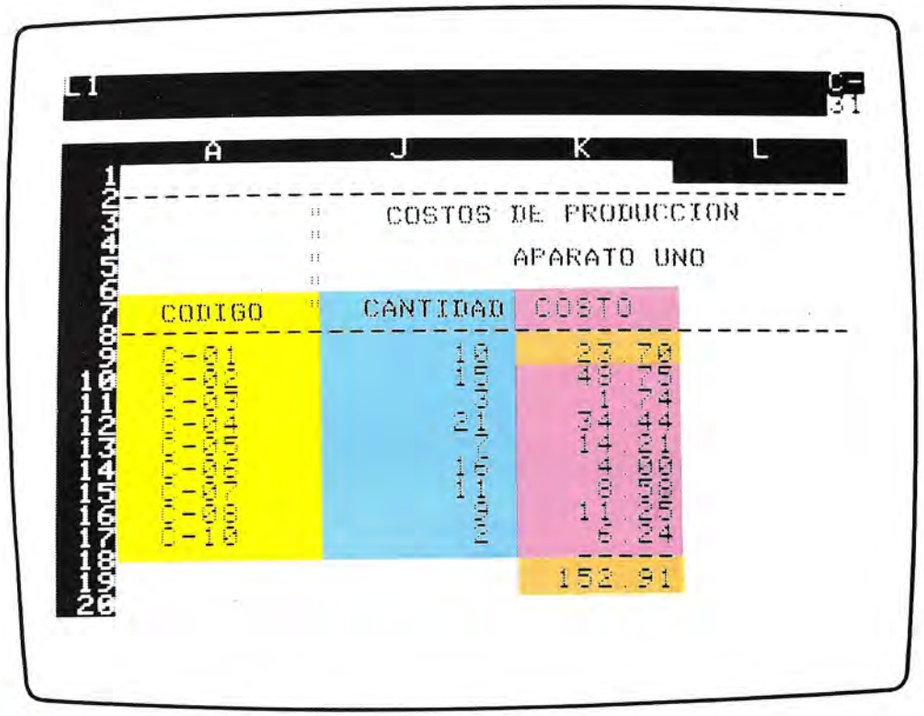
La valoración de la existencia se obtiene de forma análoga, con la fórmula +C9*F9.

Variando una de las cantidades que aparecen en la zona de movimientos se tiene el recálculo automático de toda la situación de las valoraciones.

GESTION DE UN ALMACEN: COSTOS DE PRODUCCION

El vídeo muestra ahora la zona de los costos de producción de los aparatos que contienen como componentes los artículos indicados en la descripción.





La columna A contiene los códigos de los artículos disponibles, y sirve exclusivamente como memoria previa.

La columna J se utiliza para introducir la cantidad de componentes sencillos que constituirán el producto. En el ejemplo, la construcción del aparato UNO (indicado en el título) requiere 10 unidades del componente C-01, 15 del componente C-02, etc.

La columna K contiene las fórmulas que permiten el cálculo del costo de producción del aparato, excluida la mano de obra. Una extensión del tablero podría prever también esta última denominación en términos de costos unitarios y de tiempo empleado para el ensamblaje de los componentes.

La cantidad que aparece en K9 viene dada por la fórmula +C9*J9. La estructura del tablero se presta también para realizar el análisis de los costos. Por ejemplo, si cambiando el proveedor del componente se tuvieran disminuciones en el costo de algunos artículos y aumentos del costo de otros, el programa podría calcular inmediatamente en K19 el nuevo costo del aparato, en base al cual se podría decidir si conviene o no adquirir dichos componentes al nuevo proveedor.

EJEMPLO DE IMPRESION DEL TABLERO ELECTRONICO

FILE: ALMACEN KID 20305

CODIGO	SEM. APROX.	COSTO	MOVIMIENTOS CARGA	DESCARGA	EXISTEN.
C-01	2	2.37	200	85	115
C-02	4	3.25	60	21	39
C-03	1	0.58	85	30	55
C-04	6	1.64	91	51	40
C-05	5	2.03	250	200	50
C-06	2	0.25	300	54	246
C-07	3	0.78	25	15	10
C-08	1	1.25	90	30	60
C-10	6	3.12	100	80	20
	6		1201	566	635

VALORACION CARGA	DESCARGA	EXISTEN.	COSTOS DE PRODUCCION APARATO UNO CANTIDAD	COSTO
474.00	201.45	272.55	10	23.70
195.00	68.25	126.75	15	48.75
49.30	17.40	31.90	3	1.74
149.24	83.64	65.60	21	34.44
507.50	406.00	101.50	7	14.21
75.00	13.50	61.50	16	4.00
19.50	11.70	7.80	11	8.58
112.50	37.50	75.00	9	11.25
312.00	249.60	62.40	2	6.24
1894.04	1089.04	805.00		152.91

económico o comercial, con el objetivo de determinar cómo evoluciona una situación al variar uno o más parámetros.
La técnica WHAT IF («qué sucede si...») consiste en la introducción de una variable en los datos que regulan la situación para observar sus efectos sobre el resultado final. En los casos más complejos se realiza utilizando un modelo matemático del fenómeno con algoritmos muy sofisticados. En los casos más normales (previsiones de ventas, determinación de inversiones, etc.) puede realizarse utilizando la posibilidad del tablero electrónico.
El método más sencillo consiste en implantar un tablero que describa, paso a paso, la evolución del fenómeno a analizar y en la introducción

diferentes datos de entrada. Para cada uno de ellos se tiene el recálculo de toda la estructura, con la presentación del nuevo resultado.
Abajo se ha representado el esquema de un tablero para el cálculo de previsiones de beneficios en función de la cantidad de piezas producidas. En el ejemplo sólo se han considerado algunas de las denominaciones que entran en el balance, pero la estructura puede ampliarse simplemente introduciendo nuevas filas que tengan en cuenta otros factores, o nuevas columnas si se desea analizar un intervalo de tiempo superior a 4 meses.
En general, un balance está constituido por la

ESQUEMA DEL TABLERO PARA EL CALCULO DE LOS BENEFICIOS

Introducción de datos
Cálculos relativos a los costos
Cálculos relativos a los cobros
Cálculos relativos a los beneficios

	A	B	C	D	E	F	G
1							
2							
3							
4			ENERO	FEBRERO	MARZO	ABRIL	TOTAL
5							
6	CANTIDAD PRODUCIDA						@ SUM (C6. .F6)
7							
8	COSTOS						
9							
10	PRODUCTO		+ C6 * 0.2 + 50				
11	DISTRIBUCION		+ C6 * 0.1				
12	INTERESES PASIVOS		+ 0.18 * (C10 + C11)				
13							
14	COSTOS TOTALES		@ SUM (C10. .C12)				
15							
16							
17	COBROS		+ C6 * 1.02				
18							
19	BENEF.		+ C17 − C14				
20	BENEF. %		+ 100 * (C19/C14)				

diferencia entre el total de los ingresos y el total de los costos. En el ejemplo se han previsto tres tipos de costos:

— **Costo de producción** (PRODUCTO), igual a una cuota fija (50) más una cifra proporcional al número de las piezas producidas (50 + 0.2 × Número de piezas)
— **Costo de distribución** (DISTRIBUCION), proporcional a la cantidad de producto (0.1 × Cantidad producida)
— **Intereses pasivos** (INTERESES PASIVOS). Esta denominación tiene en cuenta los intereses pasivos debidos a la inmovilización del capital empleado para la adquisición de materias primas u otras y, por tanto, es proporcional a los importes anteriores [0.18 × (PRODUCTO + DISTRIBUCION)].

El total de los costos se obtiene sumando las tres denominaciones anteriores. Introduciendo para cada mes la cantidad de producto prevista (fila 6), el programa realiza los cálculos para todas las fórmulas y presenta el resultado. El significado de las cifras depende de la unidad de medida adoptada por el usuario. Por ejemplo, la cantidad de producto puede expresarse como número de piezas, peso o volumen; lo mismo sucede para las cifras que expresan dinero. Como puede verse en la tabla de abajo, se ha utilizado la representación de los valores en formato entero; los decimales, por tanto, no se presentan, pero también se consideran en los cálculos. La estructura así formada puede memorizarse en disco (en el ejemplo se ha utilizado un file de nombre PREVISIONES) para volverla a utilizar más veces. El método WHAT IF se aplica variando uno o más datos de entrada, en este caso, la cantidad de productos, y analizando los efectos de dichas variaciones sobre el total. Por ejemplo, si causas accidentales determinan un paro de la producción en el mes de marzo, llevando la cantidad del producto a 75, en este mes se tendrá una pérdida igual al 11% de la cifra invertida, y el rendimiento total del trimestre (columna TOT) pasará a ser del 12%.
Supongamos que el objetivo económico del trimestre sea la obtención de un beneficio del 20%; ¿cuál debe ser la producción de abril para compensar el paro de marzo? El método más sencillo para obtener la respuesta consiste en la introducción de la cantidad de producto mayor del mes de abril y observar cómo varía la respuesta de la celda G20 (beneficio total %).
El proceso se repite hasta obtener un valor final cercano al deseado; la última estimación introducida es el dato buscado. Por tanto, el cálculo a realizar es del tipo iterativo; debe introducirse un valor estimado, calcular un resultado y decidir en base a éste si llevar a cabo o no un nuevo cálculo con una estimación diferente.

VARIACION DE LOS BENEFICIOS EN FUNCION DE LAS VARIACIONES DE PRODUCCION

/2/84 FILE :PREVISIONES KID

	ENE	FEB	MAR	ABR	TOT
CANT. PROD.	104	96	75	150	425
COSTOS					
PRODUCTO	71	69	65	80	135
DISTRIB.	10	10	8	15	43
INT. PAS.	15	14	13	17	32
TOT. COST.	96	93	86	112	386
COBROS	106	98	77	153	434
BENEF.	10	5	-9	41	47
BENEF. %	11	5	-11	36	12

En el tablero electrónico, las iteraciones son mucho más fáciles que con los programas en Basic (o en otros lenguajes). Efectivamente, la característica principal del tablero es la de presentar toda la estructura, permitiendo la introducción de los datos en un punto cualquiera. Por ejemplo, si se quieren analizar los efectos de las variaciones de la producción en el mes de enero, basta con cambiar el contenido de la celda C6 para obtener el recálculo completo.
Otra notable ventaja adicional es la posibilidad de variar las fórmulas de cálculo. Para analizar cómo varía el beneficio al variar el gasto de distribución, basta con modificar el contenido de la celda C11 (y de los homólogos de los otros meses). En Basic debería realizarse una modificación del programa, a menos que quiera escribirse este último en forma completamente parametrizada, obteniendo la misma forma estructural del tablero.
Sin embargo, todo lo que se ha dicho no significa que el tablero electrónico pueda sustituir el lenguaje de programación. En los problemas más complejos, o en las aplicaciones que necesitan lógicas diversificadas, la solución del tablero electrónico resulta inadecuada o inaplicable. El ejemplo presentado ilustra un caso de interactividad gestionada por el usuario. Efectivamente, la repetición del cálculo y el control de la validez del resultado están totalmente a su cargo. En las aplicaciones más complejas pueden presentarse dos dificultades: la elección inadecuada del incremento a dar a las variables, entre una iteración y otra, y el elevado número de interacciones necesarias para obtener un resultado aceptable.
Para resolver estas dificultades, en algunos tipos de tableros electrónicos se incluyen también funciones particulares que facilitan la realización de las iteraciones.

Las funciones para el cálculo recurrente

Sin embargo, en las formas más sencillas del tablero, el cálculo debe realizarse con comandos proporcionados por el operador. Ahora bien, en las formas más evolucionadas existen funciones particulares dedicadas a los cálculos iterativos. Las principales funciones son:

J. Piekerell/Marka

Ensamblaje de componentes de hardware.

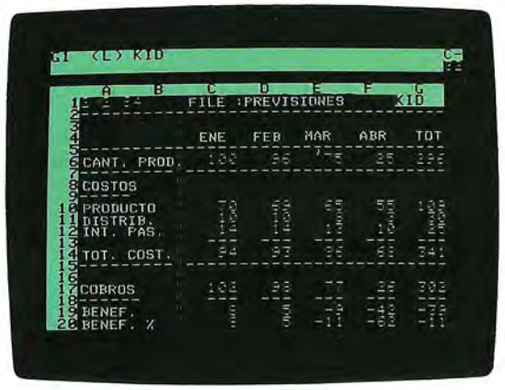
ITERCNT ( ). Restituye el valor de la iteración en curso, empezando por 1 para la primera. No tiene parámetros, y la simbología ( ) tiene el significado de argumento fantasma.
El principal uso es como prueba de término del proceso iterativo. Por ejemplo, la condición ITERCNT ( ) = 10 puede utilizarse para determinar el paro del cálculo en la décima iteración.

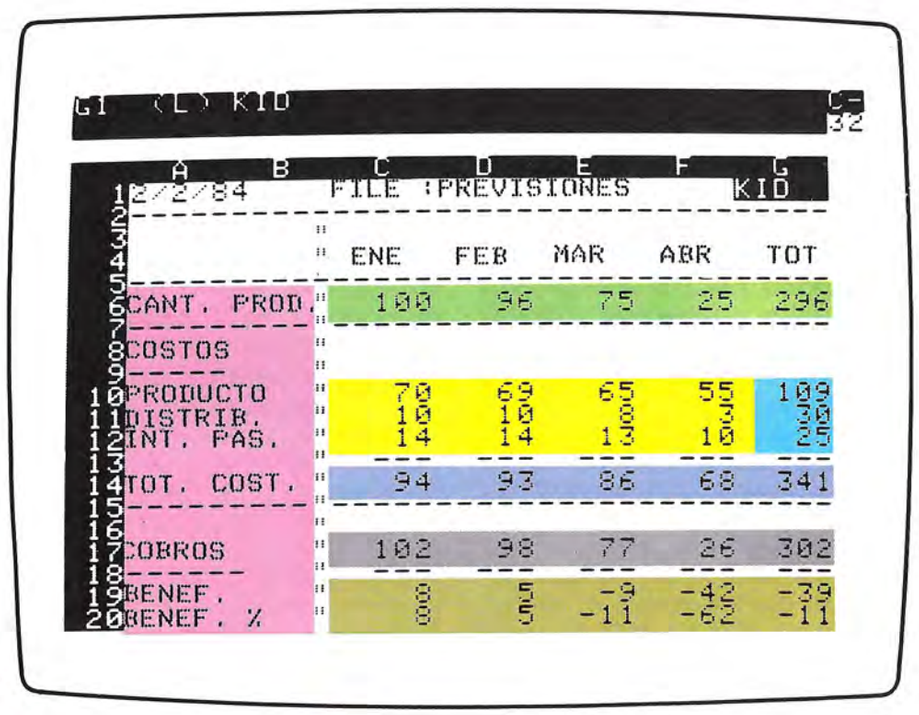
DELTA ( ). Restituye el valor absoluto máximo que se registra en las variaciones de los valores entre dos iteraciones sucesivas. Puede utilizarse como criterio de convergencia. Por ejemplo, DELTA ( ) < 0.1 detiene el cálculo cuando el valor máximo de las variaciones es inferior a 0.1.

Estas dos funciones son indispensables para detener el cálculo en los tableros que poseen el automatismo de iteración. En estos casos existe un comando (normalmente llamado ITERATION) que activa el proceso de cálculo recurrente de manera automática. En las otras formas del tablero, o se adoptan determinados artificios para simular el comando ITERATION, o bien debe ser el usuario el que introduzca, entre interaciones, el nuevo valor de la variable independiente y reactivar el cálculo. En estos casos no se produce el automatismo de fin de prueba.

ANALISIS DE LAS VARIACIONES METODO WHAT IF

En esta página se ha ilustrado la aplicación del método WHAT IF para el análisis de las variaciones en el tablero electrónico.
El vídeo repropone la estructura del tablero ya considerada. Para obtener una visión completa de todas las columnas utilizadas se ha implantado (en presentación, no en los cálculos) el formato completo.





Las descripciones contenidas en la columna A ocupan más espacio que el permitido por la amplitud de las celdas, por tanto, se han escrito utilizando las celdas contiguas de las columnas A y B. Esto implica la imposibilidad de ordenar alineados los contenidos de la celda, puesto que se necesitaría dividir todas las descripciones.

Las celdas de la fila 6 se utilizan para la introducción.

Las celdas que pertenecen al rectángulo de vértices C10-F12 contienen los cálculos relativos a los costos.

Las celdas 10, 11, 12 de la columna G contienen el cálculo de los totales de las filas PRODUCTO, DISTRIB., INT. PAS., que utiliza la función SUM.

Las celdas de la fila 14 contienen el cálculo de los totales de las tres filas anteriores.

La fila 17 contiene los cálculos de los cobros.

Las celdas de las últimas dos filas contienen las fórmulas para el cálculo de los beneficios totales y porcentuales.

El método WHAT IF puede aplicarse simplemente introduciendo un nuevo dato en una de las celdas de la fila 6. El cálculo del tablero será instantáneo.

Evolución del tablero electrónico

Los comandos y las funciones vistas se refieren a las formas más sencillas del tablero electrónico. Como en el lenguaje Basic, también en estos programas se ha producido en breve tiempo una notable evolución, que ha llevado a las versiones actuales, más completas y enriquecidas con nuevos comandos y nuevas funciones.
En la tabla de abajo se comparan, como ejemplo, las funciones previstas en el Visicalc con las del Multiplan de la versión para el Olivetti M20. Las principales funciones no varían, aunque a veces se emplean simbolismos diferentes, mientras que aparecen varias funciones nuevas (ver la tabla de la izquierda de la pág. 846). En cambio, en la tabla de la derecha se comparan los comandos. También en este caso, los fundamentales son comunes, pero en el Multiplan existen algunas nuevas implantaciones que son exclusivas del Olivetti M20.
Entre éstas, las principales se han representado por los comandos que siguen:

FUNCIONES MULTIPLAN Y VISICALC CORRESPONDIENTES

Multiplan	Visicalc
ABS(N)	@ ABS ()
PR()/2-ATAN(N/SQRT(1-N*N))	@ ACOS(N)
AND(lista)	@ AND(lista)
ATAN(N/SQRT(1-N*N))	@ ASIN(N)
ATAN(N)	@ ATAN(N)
AVERAGE(lista)	@ AVERAGE(lista)
INDEX(área, índices)	@ CHOOSE
COS(N)	@ COS(N)
COUNT(lista)	@ COUNT(lista)
	@ ERROR
EXP(N)	@ EXP(N)
FALSE()	@ FALSE
IF(1,v1,v2)	@ IF(1,v1,v2)
INT(N)	@ INT(N)
ISERROR(N)	@ ISERROR(N)
ISNA(N)	@ ISNA(N)
LN(N)	@ LN(N)
LOG10(N)	@ LOG10(N)
LOOKUP(N, área)	@ LOOKUP(N, rango)
MAX(lista)	@ MAX(lista)
MIN(lista)	@ MIN(lista)
NA()	@ NA
NOT(1)	@ NOT(1)
NPV(dr, lista)	@ NPV(dr, rango)
OR(lista)	@ OR(lista)
PI()	@ PI
SIN(N)	@ SIN(N)
SQRT(N)	@ SQRT(N)
SUM(lista)	@ SUM(lista)
TAN(N)	@ TAN(N)
TRUE()	@ TRUE

FUNCIONES ESPECIFICAS DEL MULTIPLAN

Función	Descripción
COLUMN( )	Número de la columna actual
DOLLAR(N)	El contenido de N se presenta como cifra expresada en dólares. Si N es negativa, la cifra se pone entre paréntesis
FIXED(N,d)	N formateada con d cifras decimales
LENT(T)	Longitud del texto T en caracteres
NID(T,s,c,)	Crea una cadena del texto T empezando por s para un total de c
MOD(N1,N2)	Resto de N1/N2
REPT(T,N)	Texto compuesto por N repeticiones del texto T
ROUND(N,d)	Valor de N redondeado a d cifras decimales
ROW( )	Número de fila actual
SIGN(N)	–1, 0, o + 1 según sea negativo, nulo o positivo
STDEV(lista)	Desviación estándar de los valores de la lista
VALUE(T)	Valor numérico del texo T; T puede presentar un número en cualquier formato, comprendido un importe en dólares

COMANDOS MULTIPLAN Y VISICALC CORRESPONDIENTES

Multiplan	Visicalc
Blank	/B
Transfer Clear	/C
Delete Column, Delete Rows	/D
Edit, Alpha	/E
Format Cells	/F
Format Width	/GC
Format Default	/GF
not needed; see text	/GO
Option	/GR
Insert Columns, Insert Rows	/I
Move Columns, Move Rows	/M
Print	/P
Copy	/R
Transfer Load	/SL
Quit	/SQ
Transfer Save	/SS
Window Split Titles	/T
Option	/V
Window Open, Window Split, ecc.	/W
Window Link	/WS, /WU
Goto Row-Col	>
NEXT WINDOW key	;
RECALC key	!
use references	#
REPT	/
Format Options	
Help	
Lock	
Name	
Sort	
Window	
External	

Para la selección de los comandos Multiplan sólo se introduce la letra inicial del nombre del comando

Format Options. Selecciona algunas opciones del formato de presentación del contenido de las celdas.

Help. Presenta en el vídeo una serie de instrucciones sobre el uso del programa.

Window. Permite la gestión de ventanas en el vídeo. Con este comando, la pantalla puede dividirse en partes, llamadas ventanas, cada una de las cuales puede utilizarse como un tablero en sí mismo.

External. Permite utilizar datos memorizados en una tabla «externa» a la de trabajo, por ejemplo, en una ventana definida anteriormente.

Lock. Impide la modificación de una o más celdas. Este comando es muy útil cuando en el tablero aparecen intercaladas celdas de introducción de datos y celdas de cálculos.
Desplazando el cursor para introducir los datos, es fácil confundir las posiciones y escribir el valor donde había una fórmula; en este caso, éste se pierde.
En cambio, protegiendo la celda, se impide la introducción de datos y se evitan errores.

Name. Permite definir más celdas con un mismo nombre simbólico. Es un comando que activa una lógica similar a la activada por las funciones DIM del Basic. En algunas fórmulas del tablero electrónico, el conjunto de celdas identificadas con un mismo nombre simbólico se llama rango. Todas las funciones que hacen referencia a estas celdas pueden realizarse indicando como parámetro el nombre simbólico del rango en lugar de las direcciones.

Sort. Permite ordenar el contenido de una columna. El ordenado puede comprender números o textos y ser creciente o decreciente. Otras implantaciones en otros tipos de tableros (por ejemplo en el Lotus 1, 2, 3) prevén la posibilidad de construir gráficos y de gestionar una base de datos del contenido de las celdas.
Para dar una idea de las diferencias existentes entre los diversos programas de gestión del tablero electrónico, en la pág. 848 se describe una aplicación de ejemplo del Multiplan en la versión para el Olivetti M20.

Los programas de gestión de los archivos

Una segunda clase de programas de empleo generalizado se dedica a la gestión de los archivos. También en este caso, los programas más sencillos prevén el uso de un solo file de datos y de un número limitado de funciones, mientras que los más evolucionados permiten gestiones complejas de una verdadera base de datos. Estos últimos programas ofrecen además notables posibilidades de programación, con implantaciones que en algunos casos son muy similares a las típicas del lenguaje Basic.
Las principales funciones realizadas por estos programas son las siguientes:

— Creación de los files y definición de los campos correspondientes
— Introducción y actualización de los datos
— Búsqueda
— Ordenación
— Elaboración de los datos
— Impresión de los tabulados

Creación de los files y definición de los campos

Es la primera función a activar para la creación de una base de datos. Los parámetros a proporcionar son:

— Nombre del file (o de los files)
— Unidad de disco (A/B o 0/1, según el Sistema Operativo)
— Nombres y atributos de los diferentes campos

Generalmente, el comando tiene la sintaxis

CREATE X:NOMBRE

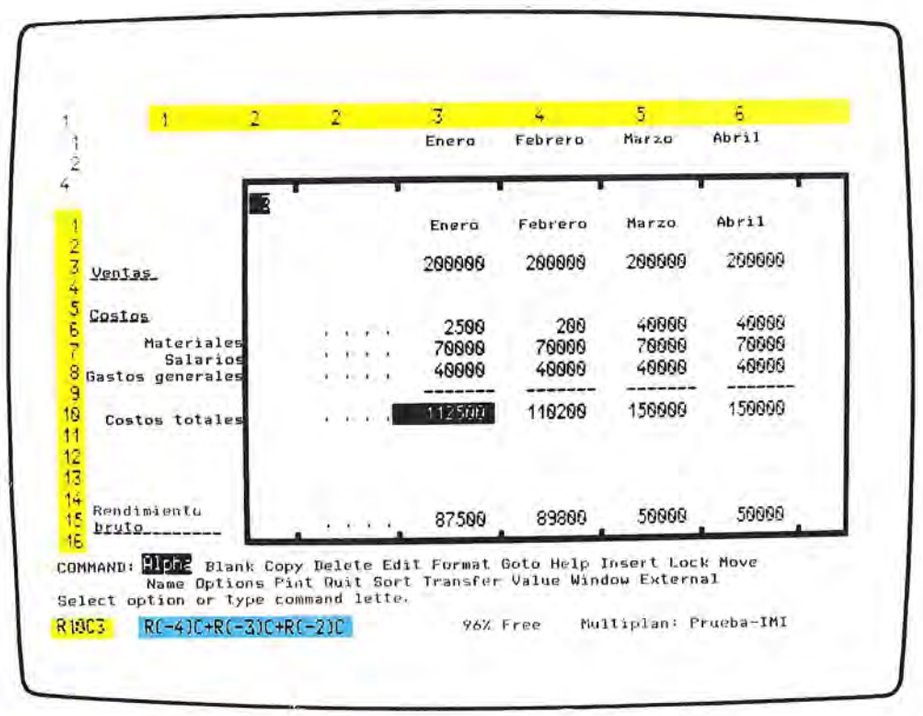
donde X es la unidad de disco seleccionada (puede omitirse, en cuyo caso es asumida por omisión por el sistema) y NOMBRE es el nombre del file a crear. Después de haber verificado la disponibilidad de espacio en el disco, el programa memoriza el nombre de un directorio y pasa a la fase de introducción de los atributos de los diversos campos que formarán el record. En estos programas, los records que pertenecen a un mismo file tienen todos la misma longitud. Esta longitud, expresada en número de bytes, se calcula como la suma de las longitudes de los campos que constituyen el record a medida que se introduce la descripción. Una vez iniciada la introducción de los datos, generalmente no es posible variar la longitud o el tipo de los campos sin perder los datos introducidos anteriormente.
Los parámetros que definen los campos son:

— Nombre: es el nombre simbólico con el que el usuario y el programa identifican el contenido de cada campo.
— Tipo: define el tipo de campo (alfanumérico o numérico).
— Longitud: es el número máximo de caracteres previstos en el campo.

Para los campos numéricos también se necesita el número de cifras decimales. Téngase presente que la longitud total del campo también debe prever el espacio reservado al punto deci-

TABLERO ELECTRONICO DEL MULTIPLAN

El tablero electrónico del Multiplan difiere en algunos aspectos del ya considerado. La presentación de los comandos, por ejemplo, se posiciona al pie, así como el contenido de la celda ocupada por el cursor.



La diferencia más marcada corresponde al direccionamiento de las celdas. Las direcciones están constituidas por dos valores numéricos precedidos por la letra R (fila) y por la letra C (columna). R10C3 es la dirección de la celda posicionada en el cruce entre la fila 10 y la columna 3, que en la fotografía está ocupada por el cursor.

En las fórmulas, los direccionamientos (relativos) se expresan en términos de desplazamiento de la posición de la celda que contiene la fórmula. Con el cursor en R10C3, la simbología que permite hacer referencia a la celda R6C3 es R[−4]C, que indica la celda que pertenece a la misma columna (C) pero posicionada 4 filas más arriba encima de (−4).

Los desplazamientos hacia arriba o hacia la izquierda se expresan con números negativos; hacia abajo y hacia la derecha, con números positivos.
La fórmula insertada en la celda R10C3 calcula la suma de los contenidos de las celdas R6C3, R7C3 y R8C3.

mal. Por ejemplo, un campo numérico con tres cifras enteras y dos decimales ocupará seis caracteres (XXX.XX). Si se especifica un campo de tipo numérico, en la fase de introducción se realiza automáticamente un control sobre los datos (control de numericidad) y se aceptan sólo datos numéricos.

Introducción y actualización de los datos

Definida la estructura del file puede pasarse a la introducción de los datos.
Normalmente, el programa presenta una máscara vídeo que indica los nombres de los campos proporcionados en la fase de generación y el espacio disponible para cada campo. La principal diferencia entre los métodos de presentación adoptados por los diversos programas se debe a las variaciones del número máximo de columnas que pueden gestionarse con la máscara vídeo. Algunos programas presentan todos los campos sobre una sola columna (uno por cada fila) en el orden en que se han definido. En otros, es posible posicionar diversos campos sobre la misma fila (entonces se tendrá una máscara de más columnas), eligiendo un orden cualquiera de presentación.
La fase de introducción de datos puede tener dos aspectos diferentes según que se realice sobre un file recién creado o sobre uno que ya existía antes.
En el primer caso es el propio programa que, al final de la fase de creación, va a la fase de introducción. En el segundo caso, los nuevos datos se consideran como añadidos y el comando que activa la transferencia es normalmente APPEND. Los dos métodos, aparte del formalismo diferente, realizan las mismas funciones, que son las de introducir datos en el formato definido durante la estructuración del file, asociando a cada nuevo dato un número de record acumulativo. Un tercer modo de introducir los datos es la inserción. La palabra clave que activa esta función suele ser INSERT y, con este comando, pueden insertarse datos en la posición deseada por el usuario.
En la fase de escritura de datos de introducción, tanto en creación como en APPEND, los datos se memorizan uno detrás de otro, según el orden de introducción, mientras que algunas aplicaciones –para hacer por ejemplo más rápida la fase de búsqueda– necesitan la posibilidad de insertar datos en cualquier posición intermedia (inserción direccionada).

El comando de inserción direccionada sigue una sintaxis que depende del software particular empleado. La mayor parte de los programas utiliza dos métodos: el **apuntamiento** y el **direccionamiento explícito**.
En el primer caso, cada dato introducido tiene una dirección propia de colocación, constituida por el número de record. Naturalmente, todos los datos que ocupan posiciones sucesivas a la especificada sufrirán un desplazamiento y se transferirán en el siguiente record al que ocupaban antes de la inserción del nuevo dato. En muchas formas de bases de datos, el posicionamiento sobre un determinado record se obtiene con un comando del tipo GOTO n, con n = número de record. Por ejemplo, la secuencia de comandos

GOTO 3
INSERT BEFORE

posiciona el puntero sobre el record 3 e inserta «antes» de éste (BEFORE) el nuevo dato. De esta manera, este último se posicionará entre los antiguos datos 2 y 3 y ocupará el record 3. El dato que ocupaba el record 3 anteriormente se desplaza al 4, y así sucesivamente hasta el final del file. Con este método pueden aportarse correcciones a los datos existentes. Un comando muy frecuente en esos programas es EDIT n, que permite la realización de correcciones en el contenido del record n especificado. Este método de apuntamiento de los datos, basado en el número de record, puede ser útil sólo en casos particulares; normalmente es incómodo de usar, puesto que se limita al empleo de un tabulado de referencia en el cual, para cada dato, se presenta el correspondiente número de record. Por tanto, existe la posibilidad de cometer errores por adoptar una referencia anterior en una operación de inserción o de adición.
El mejor modo de proceder es aprovechar las posibilidades de búsqueda que ofrecen esos programas. La operación de búsqueda, por ejemplo en base al contenido de un determinado campo, toma el nombre de **Find** (busca). Normalmente, este término también se utiliza como siglas del comando.

Búsqueda (Find)

La función de búsqueda permite seleccionar los datos en base al valor contenido en uno o más campos y es una de las funciones fundamenta-

les para la gestión de los archivos. El archivo de los datos tiene un sentido propio porque está destinado a las operaciones de búsqueda y tiene la misión de presentar un dato particular o de realizar eventuales agregaciones.
Todos los programas que pertenecen a esta categoría están orientados a la automatización del trabajo de oficina. El objeto principal de una oficina es el de recoger informaciones de las otras oficinas y de las unidades periféricas para extraer informes sintéticos sobre los que basar las decisiones y las estrategias empresariales. Este proceso se realiza con sucesivas agregaciones de los datos que, partiendo de los detalles, conducen a situaciones de síntesis. Las lógicas de selección dependen de la aplicación particular, pero también pueden ser reconducidas a una selección basada en los valores de algunos campos. En otras palabras, la principal función a realizar en un archivo es la de búsqueda. Las funciones Find pueden realizarse de tres modos principales, según el tipo de organización que se haya impuesto a los datos.

- Si el file no tiene ninguna clave ni está ordenado, la búsqueda debe realizarse leyendo todos los records y comparando el contenido del campo de selección con el valor (o el límite) deseado.
- Si el file está ordenado, puede adoptarse la técnica de ruptura de código. En este caso, la búsqueda se realiza de modo similar a la anterior, pero termina antes de la lectura de todo el file, a continuación de la ruptura del código de control.
- Si el file tiene uno o más índices, el método es más rápido. En la fase de creación del file puede formarse un índice que permita la selección en un determinado campo (campo clave). En este caso, el límite viene dado por el número de las claves de acceso a los records que contiene el file.

Mientras que en los grandes sistemas el espacio en el disco no es un problema y el tiempo de actualización de los índices es despreciable, en los microordenadores y ordenadores personales se impone una economía más restrictiva.
Un segundo inconveniente viene dado por la escasa elasticidad de una estructura como esta. Creando a priori una serie de índices puedèn preverse las principales situaciones, pero no todas las posibles. A medida que evoluciona la aplicación pueden surgir nuevas necesidades que requieren el uso de un tipo de agregación no previsto, cuyo índice no se ha creado.
Estas consideraciones han conducido a una diferente lógica de utilización de los índices.
Inicialmente no se genera ningún índice, y los datos se consideran privados de claves.
Sólo en el momento de la agregación, o de una operación cualquiera de selección, se crea un índice del campo o de los campos correspondientes. Esta metodología es común a casi todos los programas de gestión de archivos.
El flujo de las funciones a realizar está definido de esta manera:

— Creación del file
— Introducción de los datos
— Definición de las claves de selección (campo clave)
— Creación del índice en base al contenido del campo clave

Al final se dispone de un file que direcciona los datos según un determinado campo: utilizando el índice pueden realizarse las operaciones de búsqueda o de agregación.
La sintaxis utilizada para la creación del file tiene la forma:

INDEX ON "CAMPO" TO "FILE"

donde CAMPO es el nombre simbólico del campo a direccionar, y debe coincidir con uno de los campos proporcionados en la fase de generación del file de datos
FILE es el nombre del file índice
Para realizar una selección o una búsqueda debe proporcionarse al sistema el nombre del índice a utilizar (pueden existir varios al mismo tiempo), y la sintaxis tiene la forma

USE "NOMBRE" INDEX "CAMPO"

que significa: utiliza (USE) el file (índice) NOMBRE direccionado en el contenido del CAMPO.
Con este comando se informa al sistema que debe utilizar el file índice NOMBRE y realizar búsquedas (definidas con un comando sucesivo) sobre el dato que tiene el nombre simbólico CAMPO.
La siguiente instrucción deberá proporcionar el valor de comparación. La sintaxis tiene la forma

FIND XXXX

donde XXXX es el dato (clave) a buscar. Por ejemplo, la secuencia de comandos

1 - INDEX ON NOMBRE TO ANAGR
2 - USE ANAGR INDEX NOMBRE
3 - FIND Juan Pérez

realiza las siguientes funciones:

1 - Crea un file índice de nombre ANAGR en el contenido del campo NOMBRE definido previamente en la fase de creación del file de datos
2 - Utiliza (USE) el file ANAGR como índice (INDEX) del campo NOMBRE
3 - Busca todos los records que contienen en el campo NOMBRE el dato Juan Pérez

Los comandos se indican en la forma utilizada por un determinado programa.

Ordenación (Sort)

Todos los programas de gestión de archivos disponen de la función Sort (ordenación). La metodología normalmente utilizada consiste en la creación de un file de apoyo en el que se reescriben los datos, o las direcciones, en el orden deseado.
El usuario no tiene necesidad de conocer cuál es el contenido real de este file. Si el programa gestiona el Sort por medio de las direcciones de los records será siempre el programa el que, en la fase de tabulación, tomará el dato, obteniéndolo del file en base a la dirección.
El usuario sólo debe activar la función especificando el campo en el que quiere efectuar la ordenación y el nombre del file que contendrá los datos ordenados (o sus direcciones).
Una forma típica de activación del sort es

SORT ON nnn TO mm

La función se activa con el comando SORT; la expresión ON nnn indica sobre cuál de los campos se desea efectuar la ordenación (nnn debe ser el nombre simbólico de uno de los campos definidos en la base de datos utilizada) y la expresión TO mm indica el nombre del file de salida. Por ejemplo, la frase

SORT ON Dirección TO File A

activa un Sort en el campo de Dirección y memoriza la salida en un file de nombre File A. Utilizando este file para una impresión se obtienen los datos en orden creciente del contenido del campo Dirección. El file de datos no se ha modificado, y conserva el orden de introducción. El uso de un file diferente para la salida del Sort es necesario para no variar la ordenación original del file de datos.

Elaboración de los datos

El archivo de los datos casi nunca tiene una finalidad en sí mismo; el contenido de los diversos campos debe elaborarse en alguna forma. Para llenar esta necesidad, en muchos programas de gestión se han previsto funciones de cálculo. Normalmente, estas últimas siguen las mismas reglas y las mismas sintaxis de las correspondientes funciones del Basic, pero se activan con comandos específicos del programa utilizado.
Un comando muy corriente es REPLACE, que permite sustituir el contenido de un campo cualquiera por el de otro campo, o por el resultado de una elaboración.
Por ejemplo, la frase

REPLACE ALL Total WITH Cantidad*Costo

sustituye en todo el file (REPLACE ALL) el contenido del campo Total por (WITH) el resultado del producto de los otros dos campos (Cantidad * Costo). Los nombres Cantidad y Costo pueden referirse a campos del mismo record o bien indicar variables cualesquiera definidas como exteriores a la base de datos.
En algunas formas de bases de datos, el comando REPLACE (y otros) puede haberse puesto bajo condición. Por ejemplo, la operación anterior puede escribirse en la forma

REPLACE ALL Total WITH Cantidad * Costo
FOR Total = 0

La parte FOR Total = 0 condiciona la sustitución del contenido del campo Total: si este campo todavía no se ha calculado (su contenido es 0) la operación se realiza; de otra forma, el record no se procesa y el programa pasa al record siguiente.
En los programas más complejos se han previsto instrucciones muy similares a las del Basic, las cuales permiten escribir verdaderas rutinas de cálculo o de elaboración del contenido de

La automatización del trabajo de oficina (3)

Para definir un sistema para la automatización de las actividades de oficina, es útil tomar como línea de partida el examen de la «estación de trabajo» tradicional, la mesa de oficina, para identificar las posibles y diferentes configuraciones, los instrumentos habitualmente disponibles, las funciones realizadas, las interrelaciones entre las diversas actividades y las exigencias de automatización.

Una estación de trabajo automatizada deberá permitir realizar electrónicamente (es decir, con el mínimo movimiento de papeles y con las mínimas intervenciones del personal de oficina) las mismas operaciones o sus equivalentes, antes de efectuarlas a mano, con la voz o bien con máquinas sin memoria electrónica (creación y/o actualización y/o transformación, comunicación, archivo, búsqueda, reproducción de informaciones escritas u orales).

Algunas actividades se realizan localmente en la estación de trabajo (los datos y los instrumentos de proceso están disponibles cerca de la propia estación de trabajo), otras requieren interrelaciones que involucran de forma total toda la estructura de la oficina o el ambiente exterior.

Un posible modo de esquematizar esta compleja situación se ha representado en la figura de la página siguiente, donde se muestran los diversos subsistemas en que puede pensarse que está inmersa cada estación de trabajo. Los componentes que se han indicado en la figura como subsistema de oficina, *pueden considerarse los siguientes:*

— comunicaciones internas
gestión de la correspondencia y de las comunicaciones internas; comunicaciones informales; etc.
— archivos de oficina
archivos de la documentación de oficina, accesibles a más personas; comprende catálogos, direcciones, guías telefónicas; archivo personal
— procedimientos de oficina
el conjunto de las normas, formalizadas o semiformalizadas, que regulan las diversas actividades y los flujos de informaciones

En la figura se ha presentado, como entidad en sí misma, el subsistema de proceso de datos, *para indicar el conjunto de actividades y de informaciones que, en el ámbito de la oficina tradicional, actualmente se realizan con sistemas EDP y que en el modelo examinado están disponibles en la oficina de acuerdo con oportunos criterios de competencia.*

Con ambiente exterior *se indica, genéricamente, el sistema en que está inmersa la oficina: la empresa de la que forma parte, los procedimientos organizativos a nivel empresarial, las normas legales, el correo con entidades exteriores, los archivos públicos de información, etc.*

Partiendo de las consideraciones anteriores, y pensando en la utilización de un procesador central (de dimensiones pequeñas y medianas) para la gestión de un conjunto de estaciones de trabajo interrelacionadas en alguna forma, puede pensarse en una arquitectura general del tipo mostrado en la figura de la pág. 855. Se subraya que se piensa en un procesador en general ya utilizado para procedimientos EDP.

Podemos imaginar que todos los subsistemas principales (DP, oficina, núcleo de estación de trabajo, comunicaciones y servicios de impresión central) están comunicados entre sí.

El interfaz entre el usuario y el terminal debe satisfacer estrictos requisitos de sencillez y flexibilidad operativa, entre otros:

— diálogo guiado (menús y esperas autoexplicativos, presentación constante del contexto operativo, señalizaciones de error eficaces, funciones de ayuda...)
— máxima flexibilidad en el diálogo y en el acceso a la información (el operador debe poder cambiar el contexto operativo en cada instante).
— lenguaje con comandos reducidos al mínimo (amplio uso de teclas funcionales, uso del cursor para realizar selecciones...)
— posibilidad de utilizar el sistema en su conjunto o sólo en algunas funcionalidades básicas (subdivisión en grupos de funciones «autónomas», entrenamiento básico reducido al mínimo, uso de conceptos y nomenclatura habituales en el ambiente de oficina...)
— posibilidad de personalización del interfaz con el usuario
— «robustez» del interfaz

Todos los objetos manejados en el trabajo de oficina están constituidos, en varias formas, por informaciones escritas. Se diferencian entre sí por su estructura, por su modalidad de construcción y por su modalidad de uso.

Lo que pone en común los objetos de la oficina es su naturaleza textual que permite, en gran medida, el uso de instrumentos idénticos para su construcción y el empleo de criterios informativos homogéneos para su archivo y búsqueda. La estructura de dichos objetos puede asumir varias formas porque las reglas de composición no son rígidas: una hoja de apuntes puede convertirse en una carta, más hojas pueden «unirse» para componer un documento, informes y listas pueden convertirse en parte de un documento, etc. Un análisis de la estructura y de las modalidades de uso de los objetos de la oficina ha llevado a identificar cinco clases diferentes de objetos.

■ Documentos

Son textos considerados no estructurados, correlacionados con un cierto conjunto de «atributos» adecuados para facilitar su identificación, la clasificación/archivo, la búsqueda y el acceso a las informaciones contenidas. Los atributos principales son: el nombre del documento, su clasificación, el nivel de reserva, la versión, el autor, la firma, el título, los comentarios/anotaciones adjuntas, las palabras clave, el índice, las informaciones sobre las dimensiones y sobre el formato de impresión (número de páginas, líneas por página, etc.).

No todos estos atributos son obligatorios; cuando es posible, los compila automáticamente el sistema (por ejemplo, el identificador que personaliza unívocamente cada objeto en el sistema).

■ Cartas

Son textos que poseen atributos específicos: el remitente (en el lugar del autor del documento), el objeto (en el lugar del título), la relación de los destinatarios (que puede ser proporcionada también asignando el nombre de una lista de distribución), algunos de los cuales pueden ser «para su conocimiento», la «referencia Ns/Vs», el nivel de reserva, la relación de los eventuales documentos adjuntos, la fecha y la hora de expedición, etc.

Obligatoriamente, el operador sólo debe redactar los campos del remitente y del destinatario/s. El sistema redacta automáticamente la fecha y la hora de expedición.

Obsérvese que estas cartas son objetos «formales» que se unen a la correspondencia for-

La compleja situación en que puede pensarse inmerso cada puesto de trabajo.
Se identifican tres subsistemas:
el subsistema de oficina,
el subsistema EDP y el ambiente externo.

mal de una oficina tradicional, diferenciándose, por esto, de los mensajes.

Mensajes

Son objetos generalmente «de consumo» y corresponden a breves comunicaciones informales; normalmente se destruyen después de la lectura por parte del destinatario, que si lo desea, puede memorizarlos y reutilizarlos.
Sus atributos se reducen al mínimo: de..., a..., fecha.

■ Listas

Son objetos estructurados, constituidos por secuencias de registros de estructura idéntica; pueden utilizarse para intercambio de datos entre archivos DP y otros, para la formación de listas de distribución, para la personalización de las cartas o documentos que contienen campos variables (los llamados «form documents»), etc.

■ Módulos

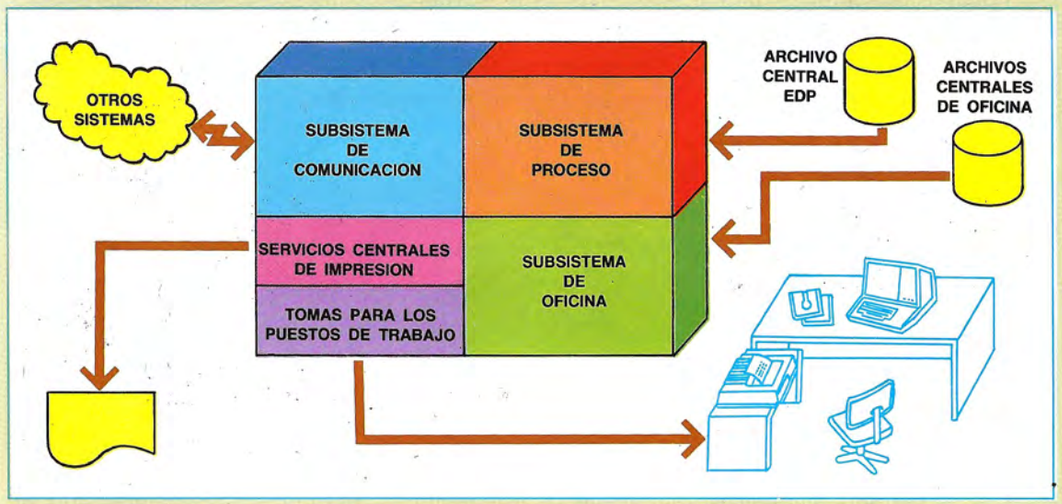
Son objetos estructurados, más complejos que las listas. Corresponden a los módulos habituales de oficina y se utilizan, por ejemplo, en el ámbito de procedimientos de oficina formalizados.
Están constituidos por campos dotados de nombre y tipificados (un cierto campo puede contener un tipo específico de datos). En su variedad de formatos, están dotados de un número mínimo de atributos fijos. En general, un objeto se crea, se archiva, se actualiza día a día, se compone con otros objetos, se expide, se copia, se anota, etc.
Para realizar estas operaciones sobre un objeto es necesario hacerlo «disponible» y visible solo o en conjunto con otros objetos.
Subrayamos aquí la necesidad de «componer» objetos de naturaleza diferente o de transformar (cambiando sus atributos) un objeto de un tipo en un objeto de tipo diferente.
Todos los objetos se memorizan en «archivos». Un archivo puede ser personal o colectivo, local o centralizado, y contener varios objetos de diferente tipo.
A cada usuario se le asocian dos buzones (uno de entrada y otro de salida). Son unos archivos adecuados para contener el «correo» de llegada y de salida. Definimos como «correo» todos los objetos expedidos entre varios usuarios según las reglas expuestas más adelante.
Para conseguir la máxima flexibilidad organizativa al usuario, a cada archivo (o buzón) hay asociada una «lista de acceso», que especifica qué usuarios pueden acceder a dicho archivo.
Son posibles diversos grados de acceso: en sólo lectura, en lectura y escritura, en lectura con posibilidad de hacer copias pero no de modificar los originales. Por tanto, desde el punto de vista lógico, el sistema puede representarse como una red de «nudos», a cada uno de los cuales hay asociado un par de buzones y un conjunto de ramas con las listas de acceso. Las funciones que el sistema ofrece a los usuarios están agrupadas en clases.

1. CREACION/REVISION DE OBJETOS

Estas funciones operan sobre un «objeto corriente» que se encuentra fuera del archivo; por tanto, la creación y la revisión de un objeto no modifican el estado de los archivos. Para revisar un objeto archivado es necesario «agarrarlo». Entonces, el sistema procede automáticamente, y de modo invisible por el usuario, a crear una copia de trabajo. Al terminar la sesión de revisión, el usuario podrá pedir la sustitución de la versión en el archivo.
Hay disponibles todas las funciones de creación/revisión proporcionadas por un potente tratamiento de textos en el que se incluyen:
— funciones de «cut and paste»: ensamblaje de objetos a partir de otros objetos, memorización de «retales» para utilizar en fases sucesivas, memorización de «Form Documents»
— inserción de notas fuera de texto
— formación automática de índices, con posibilidades de acceso directo a los «párrafos» del documento. En el caso de objetos formateados (listas y módulos):
— funciones de definición del formato (mediante diálogo guiado, en que el usuario define la imagen de la lista o del módulo)
— funciones de compilación, con entrada guiada del formato (tabulación vertical/horizontal, validación de los tipos de datos, compilación automática de los datos calculados, y, por ejemplo, totales, porcentajes, etc.)
— selección sobre listas: por ejemplo, creación de una sublista de todos los valores que gozan de una cierta propiedad en otra lista, etc.
— funciones de inserción (construcción de documentos/cartas, con inserción de datos presentes en la lista)
— adquisición/transferencia de datos de/a archivos de Proceso de Datos.

2. IMPRESION

Además de las funciones normales de formateado para la impresión, la impresión de inserción *permite fundir, directamente durante la impresión y sin conservar copias en archivos de los resultados de la fusión, documentos que contienen partes variables y listas (con posibilidad de selección de los valores de la lista que satisfacen condiciones específicas).*



Estructura del sistema para la gestión de las actividades de oficina.

3. BUSQUEDA EN LOS ARCHIVOS

Este grupo de funciones permite realizar la búsqueda de un objeto en el interior de uno o más archivos.
El usuario puede seleccionar (buscar) grupos de objetos utilizando tres niveles de selección:

1) *selección en el ámbito de* un *archivo especificado, o de* todos *los archivos a los que el usuario tiene visibilidad*
2) *selección del tipo de cada objeto: cartas, documentos, etc.*
3) *selección basada en:*
   — *palabras clave especificadas en la fase de introducción del objeto*
   — *valores especificados de los atributos (título, autor, etc.).*

Los tres niveles de selección pueden utilizarse al mismo tiempo.
La función de selección visualiza la lista (sumario) de todos los objetos seleccionados.
Frente a esta lista, el usuario puede seleccionar un solo objeto y examinar secuencialmente los diversos objetos hacia adelante o hacia atrás (partiendo de un elemento cualquiera). El objeto seleccionado se visualiza, junto con todos sus atributos, en un formato análogo al formato de documentos y de cartas tradicionales.
Son posibles todas las operaciones del Editor, que no modifican el texto preexistente: scroll horizontal y vertical, búsqueda de trozos de texto, búsqueda de un párrafo especificado, extracción de un recorte para un uso posterior, etc...
Los objetos listados pueden manipularse colectivamente con varias operaciones: cancelación-/transferencia o copia a otro archivo, impresión de elementos listados, impresión de la lista.
Obsérvese que, para permitir trabajar sobre objetos deseados con frecuencia, es posible efectuar posteriores subselecciones en la lista.

4. GESTION DEL BUZON DE SALIDA

Cada carta, para poder ser expedida a través del servicio de «correo electrónico», debe depositarse en el buzón de salida. *En éste son posibles todas las operaciones lícitas para un archivo general, y además, las siguientes:*

Firma — *El propietario del buzón (y cualquier otro usuario incluido en la lista de acceso) puede aportar, con el adecuado comando, su propia «firma» a la carta visualizada en aquel momento.*
Ésta se convierte en un atributo de la carta, que se visualizará con la propia carta («carta firmada»).

Expedición — *Una carta redactada correctamente puede expedirse mediante un simple comando.*

Los objetos de clase diferente a las cartas pueden expedirse adjuntados a una carta.

5. GESTION DEL BUZON DE LLEGADA
Las cartas de llegada se depositan en el sistema dentro del buzón de entrada. *A petición del usuario se visualiza el resumen del contenido del buzón: cartas y eventuales objetos adjuntos. Obsérvese que un buzón puede ser visionado por cualquiera que pertenezca a la lista de acceso y no necesariamente sólo por su propietario. En cualquier caso, el objeto y el texto de una carta declarada reservada por el remitente sólo puede visualizarlos el usuario habilitado.*
Las operaciones aceptadas por el buzón son las posibles en un archivo general.
Las cartas recibidas no pueden modificarse. Un adecuado comando produce la formación automática y el envío inmediato, al remitente de la carta visualizada, de una carta de «recepción» sin que el usuario deba especificar texto o atributos. Después de la lectura de una carta, el usuario puede archivarla especificando el archivo deseado.
En otro caso, la carta no se archiva y se presenta a la visión del usuario en las sucesivas reaperturas del buzón.

6. AGENDA PERSONAL
Este grupo de funciones permite gestionar una agenda de notas y memorando privados. El usuario tiene a su disposición una agenda fácilmente consultable especificando el día (hoy, mañana, una determinada fecha) o seleccionando una semana entera.
Es posible instruir al sistema para que presente al usuario, en las fechas oportunas, los mensajes contenidos en la agenda.

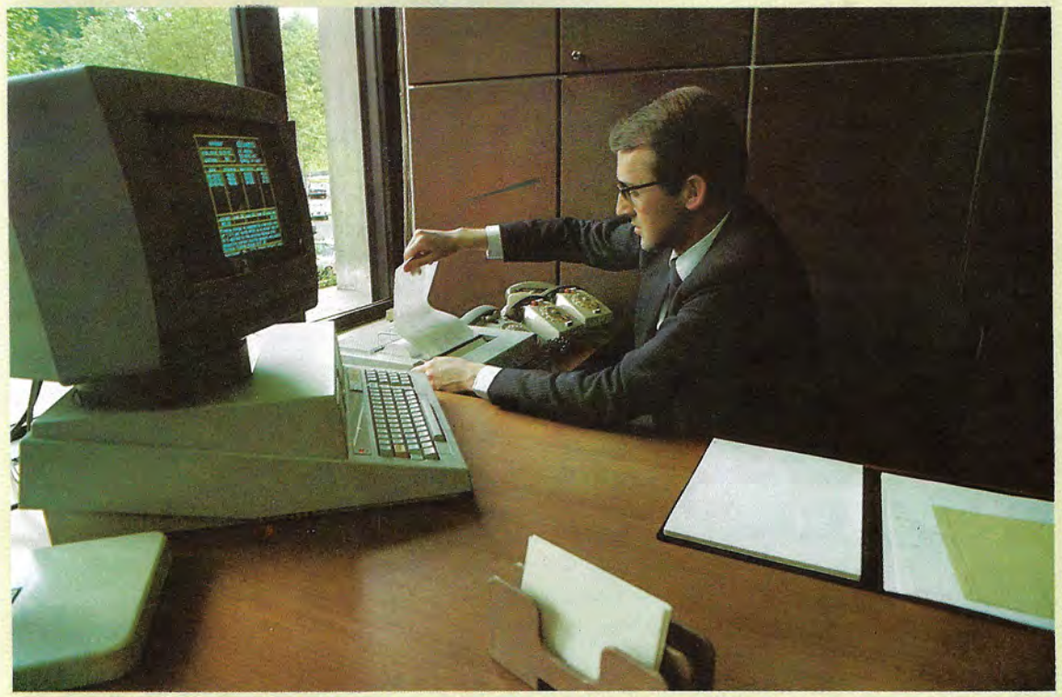
7. DEFINICION DE LOS PROCEDIMIENTOS
El sistema descrito hasta aquí se ha concebido según el criterio de la máxima flexibilidad operativa: esto, sustancialmente, no impone al usuario ningún procedimiento de trabajo preestablecido. Efectivamente, el sistema debe poderse insertar con la máxima flexibilidad en situaciones de organización también muy diferentes, sin perturbarlas.
Un sistema avanzado de automatización de las actividades de oficina, por tanto, deberá poderse adecuar a dichas normas y procedimientos. En otras palabras, el usuario específico deberá tener la posibilidad de personalizar el sistema a los diversos procedimientos en uso. Por ejemplo, deberá ser posible comunicar al sistema que, en un determinado contexto organizativo:
- *«la expedición de una carta puede producirse sólo si la carta está firmada por el remitente»*
- *«cada carta llegada con ciertos atributos deberá archivarse en un archivo específico y enviarse, con una anotación específica, a un destinatario específico»*
- *«Cada variación de la lista de acceso de ciertos archivos debe comunicarse mediante una carta estándar a todos los usuarios interesados»*
- *etc.*

Esto puede realizarse permitiendo al usuario definir «macrocomandos» (p.e.: EXPEDIR, CLASIFICAR...), en términos de:
- *comandos elementales (todos los comandos tratados por el sistema y resumidos en las páginas anteriores)*
- *condiciones que hacen posible la ejecución de dichos comandos (p.e.: «Firma presente», «atributo x = yyyy»).*

8. DEFINICION DE LOS ARCHIVOS
Este grupo de funciones soporta una gestión normal de los archivos (creación/cancelación-/duplicación) además de la gestión de las correspondientes listas de acceso.

9. ADMINISTRACION DEL SISTEMA
Este núcleo de funciones es visible exclusivamente por un usuario específico, que tiene la misión de administrar el sistema:
- *creación y gestión de todos los usuarios y de los correspondientes perfiles*
- *habilitación de los diversos usuarios en el uso de las distintas clases de funciones ofrecidas por el sistema*
- *creación y gestión de los archivos centrales y del «back-up»*
- *definición de los procedimientos a nivel global.*

La subdivisión en las clases indicadas se ha realizado sobre la base de estrechos criterios de homogeneidad operativa: cada clase ha asociado un submenú de funciones elementales, seleccionables a partir de un menú principal del subsistema de gestión de las actividades de la oficina que, a su vez, puede personalizarse sobre la base de las necesidades del usuario.



C. Pozzoni/Marka

Empleo del tablero electrónico en la Olivetti M20

Obsérvese que, si bien todas las funciones del sistema se presentan al usuario con un interfaz uniforme (el modelo operativo del sistema es sencillo; los comandos aceptables en contextos similares son siempre los mismos), el operador puede aprender el uso del sistema gradualmente, por ejemplo a través de la siguiente secuencia de entrenamiento/uso:
- Núcleo base de gestión de textos: *creación-/revisión/composición de textos, impresión, búsqueda en los archivos (no todas las funcionalidades). Trabaja sobre archivos ya definidos y productos de forma adecuada. No comprende las funciones (más complejas) de creación-/reestructuración de archivos completos, ni funciones de comunicación con otras estaciones de trabajo*
- Núcleo de comunicación: *gestión del buzón de llegada, gestión del buzón de salida. Estas informaciones se apoyan completamente en el concepto de archivo (con las respectivas operaciones de base) utilizado en el punto anterior (los buzones de entrada/salida son archivos), con la ayuda de simples operaciones específicas*
- Núcleo avanzado de gestión de textos: *definición de los archivos; búsqueda en los archivos (todas las funcionalidades). Permite operaciones de carácter global sobre los recursos accesibles al usuario particular: definición o reestructuración de archivos, búsqueda de informaciones en el conjunto de los archivos, etc.*
- Núcleo de funciones personales: *agenda personal. Constituye un núcleo independiente de los otros que permite la gestión de tareas (con solicitación) y apuntes personales*
- Núcleo de procedimientos locales: *definición de los procedimientos. Permite la definición de macrocomandos que indican conjuntos de operaciones y procedimientos de uso corriente. Requiere sensibilidad e implantaciones de tipo sistemático (a nivel local)*
- Núcleo de administración del sistema: *administración del sistema. Soporta la creación de nuevos usuarios y la definición de las normas operativas a las cuales deben someterse los diversos usuarios locales. Necesita competencias sistemáticas y procedimientos de nivel global.*

(Extraido de «Un modelo de automatización de la oficina», de A. Cicu, M. Gualzetti, R. Polillo, QUADERNI DI INFORMATICA, año IX, n.º 2, 1982, Honeywell Information Systems Italia)

los records. La instrucción considerada en el último ejemplo es una forma de programación, aunque sea simple. En Basic habría sido

IF Total = 0 THEN Total = Cantidad * Costo

Es evidente que, aparte de la forma, se trata de la misma instrucción. En realidad existe una diferencia sustancial, y está constituida por el automatismo de exploración del file inserto en el comando de la base de datos. En Basic, un programa que sustituye el contenido de cada record de todo un file necesita un bucle de lectura y escritura desde el primero al último record del file. Las operaciones a realizar son

— Lectura de la longitud del file (por ejemplo, memorizado en el record 1)
— Bucle entre el primer dato y el último con Lectura de un record - Elaboración - Escritura

Las instrucciones de bucle, lectura, escritura del programa están condensadas en el comando REPLACE ALL. El sistema lee del directorio la longitud del file y activa el bucle y las funciones de I/O.
La variedad y la complejidad del conjunto de instrucciones previstas en la base de datos dependen del tipo del programa. Por este motivo, en algunos programas se ha previsto la posibilidad de memorizar en disco, en forma de instrucciones, las funciones a realizar para después lanzarlas a ejecución con un solo comando cada vez que son necesarias. El programador así puede utilizar las instrucciones y los comandos como si fuesen una forma particular de Basic, dejando al usuario la sola misión de ejecutar el procedimiento predefinido.

Impresión de los tabulados

La última función prevista en todos los programas de gestión de las bases de datos es la preparación de la emisión de los tabulados.
El usuario puede estructurar la impresión especificando qué campos desea imprimir y en qué posición. En los procedimientos más completos también es posible efectuar cálculos sencillos, por ejemplo totalizaciones, y pedir la presentación al final del tabulado. En algunos casos es posible la impresión de resultados intermedios utilizando el contenido de un campo como código de ruptura (la impresión se activa al variar el contenido del campo considerado). Esta forma de condicionamiento de los procesos, aunque está presente en algunas bases de datos, es peculiar del lenguaje RPG, orientado específicamente a la generación de informes.

Tratamiento de textos

La automatización del trabajo de oficina debe incluir la posibilidad de gestionar textos generales con el calculador, por ejemplo cartas, documentos, etc. A este fin se han producido diversos programas (**Editores de Textos**), cuya función es la de permitir la preparación de un texto como si se tratase de un conjunto de datos cualquiera. Esto permite simplificar mucho las correcciones y tener la impresión de un texto en un número cualquiera de copias.
Las ventajas que presenta un Editor de Textos son evidentes:

— La memorización del texto permite corregir errores o aportar modificaciones sin volver a escribirlo todo
— El archivo en disco flexible es mucho más compacto que sobre un soporte de papel
— La posibilidad de obtener más impresiones elimina la necesidad de fotocopias y evita los problemas de reproducción que se tienen cuando el original no es perfecto.

También en este campo, la variedad de las funciones disponibles depende del programa empleado. En los Editores de Textos más modernos se incluyen funciones muy particulares y útiles, como las siguientes:

— Controles de ortografía, en varias lenguas, según un vocabulario definido por el usuario
— Posibilidad de inserción de una o más palabras en el interior de un texto ya completo, con repaginación automática
— Control del corte de las palabras en los márgenes (silabación)
— Preparación de tablas
— Algunas sencillas funciones de búsqueda en el contenido de los documentos

En los sistemas más grandes existen aparatos especiales que permiten la introducción directa de documentos. Complejos sistemas hardware leen el texto escrito en papel y lo traducen en datos elementales que el ordenador puede procesar y memorizar.

GLOSARIO

GLOSARIO DEL MULTIPLAN (VERSION OLIVETTI M20)

Activo. Cualquier elemento actualmente en uso e inmediatamente accesible, como ventana activa, celda activa o campo activo de un comando.

Alineado. Posicionado de los valores de una celda en el interior de la propia celda. Los valores pueden alinearse a la izquierda, a la derecha, o centrarse.

Alta luminosidad. Zona de la pantalla evidenciada con luminosidad acentuada. La alta luminosidad se emplea para el cursor de edición, la celda activa, el número de ventana activa y el comando seleccionado en el menú principal de los comandos.

Barras de movimiento del cursor. Teclas que desplazan el puntero de celda. Las teclas ↑, ↓, ←, →, desplazan el puntero de una celda a la vez. La tecla HOME lo desplaza a la celda de la parte superior izquierda de la ventana activa.

Bloqueo. Protección de las celdas que contienen fórmulas o textos contra alteraciones accidentales.

Campo. Parte del comando en que se introduce una respuesta para proporcionar al Multiplan instrucciones acerca de la modalidad de ejecución del comando Cuando el Multiplan visualiza un campo, propone automáticamente una respuesta. La respuesta propuesta debe aceptarse o sustituir a la respuesta efectiva que se pretende definir para aquel campo.

Carácter. Símbolo visualizable sobre la pantalla; los caracteres son letras, cifras, signos de interrupción y caracteres especiales como $, +, %.

Carga. Permite hacer nuevamente activa una tabla salvaguardada. La tabla a cargar debe haber sido salvaguardada con el comando TRANSFER SAVE. El comando TRANSFER LOAD se emplea para cargar la tabla del disco en la memoria del sistema.

Celda. Elemento de la tabla donde deben memorizarse valores numéricos, fórmulas o textos. Una celda se individualiza por sus coordenadas y a ella puede referirse con un nombre. El contenido de una celda determina su valor; el formato de la celda determina el modo de presentar el valor.

Celda activa. Celda indicada por el puntero de celda. El contenido de la celda activa puede verse en la fila de estado y editado con el comando EDIT.

Columna. Fila vertical de celdas sobre la tabla. Hay 63 columnas, indicadas con los números 1 al 63.

Comando. Impone al Multiplan la realización de cualquier cosa. Un comando puede tener uno o más campos en los que especificar las modalidades de ejecución del propio comando.

Contenido (de una celda). Es lo que se ha introducido en una celda. Si no se ha introducido nada, la celda está vacía y contiene espacios. De otra forma, la celda contiene textos, valores numéricos o una fórmula. Si una celda contiene una fórmula, el valor de la celda, que es el resultado de la fórmula, normalmente se visualiza en pantalla.

Correlación. En ambiente Multiplan, el término correlación indica la conexión entre las tablas para las que los datos de una tabla no activa se utilizan en los cálculos de la tabla activa. La tabla inactiva se llama tabla de soporte. En los datos a copiar debe haberse atribuido un nombre con el comando NAME o deben haberse identificado con una referencia absoluta, después de que los datos de una tabla de soporte puedan ser utilizados en fórmulas en la tabla activa. La correlación también se utiliza para contener dos ventanas, una dentro de otra, de manera que los scroll entre ambas se realice simultáneamente.

Cursor. Ver **Cursor de edición**.

Cursor de edición. Parte en alta luminosidad de un comando sobre la fila de comandos que puede tener dimensiones comprendidas entre un carácter y un campo entero. El cursor de edición se desplaza con las teclas de edición. Éstas indican los puntos en que el comando puede modificarse.

Directorio. Tabla de los nombres de los files memorizados en disco. En la práctica, el directorio corresponde a la lista de los files.

Edición. Modificación de una respuesta en un campo de comando. Las teclas de edición se utilizan para desplazar el cursor en la respuesta, mientras que las teclas de carácter se utilizan para sustituir o insertar caracteres.

Fila. Línea horizontal en la tabla. Hay 255 filas posibles, identificadas con los números entre 1 y 255.

Fila de comandos. Las filas de la pantalla que está inmediatamente debajo de la parte visualizada de la tabla empiezan con la palabra COMMAND y representan el menú principal de los comandos. Es el lugar en que se definen los comandos.

Fila de estado. Ultima fila de abajo de la pantalla, donde el Multiplan presenta informaciones sobre el estado del sistema, como la posición de la celda activa y su contenido.

Fila de mensajes. La fila inmediatamente después de la última fila de la tabla presentada. Zona de la pantalla donde se presentan los mensajes Multiplan.

File. Flujo homogéneo de datos memorizados en disco. Cuando una tabla se preserva, está escrita en un file. No todos los files son tablas preservadas, pero los que lo son pueden cargarse en memoria o correlacionarse con otras tablas.

Formato. Modo de presentación del valor de una celda. El formato define la posición de la coma en el número y el alineado del valor presentado. Un formato puede es-

pecificarse para una o más celdas con el comando FORMAT CELL; las celdas sin un formato específico se presentan con el formato de omisión predispuesto con el comando FORMAT DEFAULT.

Fórmula. Indicación del modo de cálculo de un valor. Todas las veces que se intercambia el contenido de una celda, el Multiplan recalcula todas las fórmulas de la tabla (a menos que el recálculo automático se inhiba).

Función. Operación matemática o estadística preconstituida que el Multiplan puede realizar sobre uno o más valores; por ejemplo, SUM o AVERAGE.

Grupo de celdas. Conjunto de una o más celdas al que puede asignarse un nombre; por ejemplo: Ventas.

Iteración. La iteración es la repetición de un cálculo utilizando los resultados de los cálculos anteriores en lugar de una cantidad desconocida.

Memorización. Operación que permite salvaguardar en un file del disco de forma permanente la tabla activa.

Mensaje. Aviso presentado por el Multiplan en la fila de mensajes para explicar las causas de un error operativo o de cálculo, o para sugerir el tipo de introducción que el sistema está esperando.

Menú. Lista de alternativas. Una elección en un menú se realiza en uno de los siguientes modos: desplazándose sobre la lista con SPACE o S2 (la alta luminosidad indica la selección actual sobre el menú) y seleccionando el elemento en alta luminosidad con la tecla CR, o bien introduciendo la inicial del elemento deseado.

Nombre (de una celda o de un grupo de celdas). Referencia asociada a una o más celdas con el comando NAME. El nombre puede utilizarse en las fórmulas como referencia a la celda o al grupo de celdas.

Nombre del file. Nombre usado como referencia a una tabla cuando ésta está salvaguardada, cargada o correlacionada con otra.

Pantalla/Vídeo. Elemento del sistema que permite la visualización de números y/o textos. La pantalla está dividida horizontalmente en 25 filas. El Multiplan usa las últimas cuatro filas para informaciones y control. La última fila de abajo es la fila de estado, la anterior es la fila de mensajes y las otras dos constituyen las filas de comandos. Todas las demás filas están a disposición de la tabla y se utilizan para visualizar parte de la tabla a través de una o más ventanas.

Puntero de celda. Zona de alta luminosidad que identifica la celda activa en el vídeo. El puntero de celda se desplaza de una celda a otra con las barras de movimiento del cursor o con el comando GOTO.

Rango. El rectángulo más pequeño de celda que contiene dos referencias. Un rango está definido con el símbolo «dos puntos» (:). El rango R3: R8 define todas las celdas comprendidas en las filas de 3 a 8 (3, 4, 5, 6, 7, 8). Ver también **Referencia**.

Referencia. Modalidad de direccionamiento que identifica una o más celdas en el interior de la tabla.
La referencia puede ser simplemente para una sola celda, por ejemplo: R9C2.
Una referencia puede ser:
en relación a la posición de la propia celda que contiene la referencia, como R(– 1)C;
con una sola celda, como anteriormente, o para un grupo de celdas: R6 es una referencia a todas las celdas de la fila 6 constituida por intersecciones de referencia, rango o uniones de referencia;
o un nombre definido para definir una o más celdas. Ver también **Rango** y **Nombre**.

Referencia absoluta. Referencia a una celda utilizando números de fila y de columna explícitos; por ejemplo, R17C12. Es opuesto a la referencia relativa, como R(+ 1)C(– 2).

Referencia relativa. Referencia a una celda en relación a la posición de la propia celda; p.e. R(– 1)C significa «la celda en la fila inmediatamente encima de la propia columna», es opuesta a la referencia absoluta, p.e. R17C12.

Scroll. Desplazamiento de la imagen en la pantalla fila por fila o columna por columna. El scroll se realiza con las barras de movimiento del cursor. Por ejemplo, pulsando la barra → hasta que el puntero de celda alcanza el borde derecho de la pantalla y, si se continúa pulsando, el Multiplan desplaza la imagen hacia la izquierda columna por columna. Este tipo de scroll se llama «scroll horizontal». Cuando la imagen se desplaza por la pantalla fila por fila, el scroll se llama «vertical».

Tabla activa. Es el conjunto de celdas que contienen los valores en proceso. Corresponde a una matriz de dimensiones máximas iguales a 63 celdas (filas) por 255 (columnas). Cada celda de la tabla puede contener valores numéricos, textos o fórmulas.

Tabla dependiente. Tabla que utiliza valores procedentes de otra tabla. La tabla dependiente está subordinada a los valores calculados sobre otra tabla memorizada a la que ésta está correlacionada por medio del comando EXTERNAL COPY. Ver también **Correlación**.

Tabla de soporte. Tabla que contiene valores que se utilizarán por otra tabla (tabla dependiente). Los valores interesados en la copia sobre la tabla dependiente deben estar identificados con un nombre especificado con el mando EXTERNAL NAME. Ver también **Correlación**.

Tecla de cancelación. Tecla funcional que produce la cancelación del comando en curso y la visualización del menú principal de los comandos.

Tecla Next Unlocked Cell (siguiente celda no bloqueada). Tecla de función que desplaza el puntero de celda sobre la siguiente tecla que no esté vacía o bloqueada. Utilizable para localizar inmediatamente celdas que contienen números (el lugar de fórmulas o texto) que puedan modificarse para verificar el efecto de la modificación en toda la tabla (p.e. el método WHAT IF).

Teclas de edición. Teclas que desplazan el cursor de edición por la fila de comandos. Comprenden por ejemplo: WORD RIGHT, WORD LEFT, CHARACTER RIGHT, CHARACTER LEFT.

Teclas de función. Teclas que hacen realizar rápidamente una acción al Multiplan. Las teclas de función comprenden CANCEL, NEXT WINDOW y CR. Ver también **Barras de movimiento del cursor** y **Teclas de edición**.

Ventana. Zona de la pantalla en la que el Multiplan presenta parte de la tabla activa. Pueden estar abiertas al mismo tiempo hasta 8 ventanas: éstas se abren o se cierran con el comando WINDOW. Cada ventana tiene su propio número (de 1 a 8) en el ángulo superior izquierdo. El número de la ventana activa es en alta luminosidad; esta ventana contiene el puntero de celda y la celda activa.

Ventana activa. La ventana que contiene la celda activa, indicada en la pantalla con un número de ventana en alta luminosidad.

ESTRUCTURA DE LOS COMANDOS MULTIPLAN

Los comandos presentados en el menú principal de los comandos pueden seleccionarse pulsando la letra inicial (p.e. **A** por ALPHA) o posicionando el cursor en el comando deseado con las barras **S1** y **S2** y pulsando **CR** para ejecutar el comando.
En la celda activa pueden introducirse valores numéricos sin que sea necesario seleccionar el comando VALUE.
A la mayor parte de los comandos hay asociados subcomandos cuyo menú se presenta al seleccionar comando principal.

CTRL + C	Cancela un comando, vuelve al menú de los comandos
S1	Desplaza el cursor luminoso sobre el siguiente comando/subcomando del menú. Desplaza el cursor luminoso sobre el próximo campo en el subcomando
S2	Mueve el cursor sobre el comando/subcomando anterior en el menú
SHIFT + ↑ ↓ ← →	Desplaza el cursor según las flechas, cierra la introducción manteniendo el estado ALPHA/VALUE
CR	Ejecuta el comando. Introduce el texto o el valor
@	Convierte las referencias a las celdas de relativas a absolutas

A ALPHA — Permite introducir un texto o un valor numérico en la celda activa
CR o SHIFT + ↑ ↓ ← → — El uso de las barras de movimiento de cursor confirma la introducción y desplaza el puntero en la dirección indicada

B BLANK — Permite especificar las referencias de una o más celdas
CR — El contenido de las celdas especificadas se anula

C COPY
- **R Right** — Permite definir el número de celda a la derecha de la celda actual en que se quiere copiar el contenido de la celda activa
  - **CR**
- **D Down** — Permite definir el número de la celda que está debajo de la celda actual sobre la que se quiere copiar el contenido de la celda activa
  - **CR**
- **F From** — Permite definir las referencias de una o más celdas a copiar en las referencias especificadas como celdas receptoras
  - **CR** — El comando Copy From incluye prácticamente tanto el comando Copy Right como Copy Down y además permite la copia de celdas no contiguas

D DELETE
- **R Row** — Permite definir las referencias a todas o parte de las celdas de una o más filas
  - **CR** — Las áreas especificadas se eliminan de la tabla. La ejecución del comando provoca la compresión de la tabla hacia arriba
- **C Column** — Permite definir las referencias a todas o parte de las celdas de una o más columnas
  - **CR** — El área especificada se elimina de la tabla. La ejecución del comando produce la compresión de la tabla hacia la izquierda

E EDIT
CR o SHIFT + ↑ ↓ ← → — Permite editar, a través de las teclas de edición, textos y fórmulas

F FORMAT
- **C Cells** — Permite definir las referencias de una o más celdas de las que se quiere modificar el formato de presentación
  - **CR**
- **D Default**
  - **C Cells** — Permite definir el formato de presentación y el tipo de alineado por omisión de las celdas
    - **CR**
  - **W Width** — Permite definir la longitud de omisión (en número de caracteres) para todas las columnas de la tabla
    - **CR**

	O Options		Permite definir y verificar las opciones de formato correspondientes a las celdas
	CR		Las respuestas presentadas por el comando corresponden a las opciones actuales
	W Width		Permite definir, para las columnas especificadas, una longitud diferente de la por omisión (ver comando Format Default Width)
	CR		
G GOTO	**N Name**		Posiciona el puntero de celda sobre la celda correspondiente al nombre especificado
	CR		En este caso, el nombre identifica un área de celda y el puntero de celda se posiciona sobre la primera celda del área especificada (celda en la parte de arriba izquierda)
	R Row-Col		Introduce el número de RC
	CR		Posiciona el puntero de celda en la celda especificada en la referencia
	W Window		Posiciona el puntero de celda en la celda especificada en la referencia del interior de la ventana indicada
	CR		El comando provoca el scroll de la ventana hasta posicionar la celda especificada en la primera posición de la parte superior izquierda de la ventana
H HELP	**R Resume**		Recupera la tabla activa a partir del punto en que se había reclamado el comando HELP
	S Start		Presenta la primera página del file de HELP
	N Next		Presenta la página siguiente del file de HELP
	P Previous		Presenta la página anterior del file de HELP
	A Applications		Presenta una lista de problemas comunes que pueden aparecer en la utilización del Multiplan y los comandos a utilizar para su resolución
	C Commands		Presenta la descripción de cada comando Multiplan para empezar desde el primero: ALPHA
	E Editing		Presenta la descripción del ambiente Editing del Multiplan
	F Formulas		Presenta informaciones relativas a las reglas a seguir y una lista de las funciones que pueden utilizarse en las definiciones de fórmulas
	K Keyboard		Presenta una lista de las teclas y su significado para utilizar en ambiente Multiplan
I INSERT	**R Row**		Inserta en las posiciones anteriores de la fila activa todas o parte de una o más filas con contenido nulo
	CR		
	C Column		Inserta en las posiciones anteriores de la columna especificada todas o parte de una o más columnas con contenido nulo
	CR		
L LOCK	**C Cells**		Permite bloquear/desbloquear una o más celdas para evitar su modificación accidental
	CR		
	F Formulas		Permite bloquear/desbloquear todas las celdas que contienen textos o fórmulas para evitar que se modifiquen accidentalmente
	Y		Confirma la petición
M MOVE	**R Row**		Permite definir las referencias correspondientes a filas que deben colocarse en el interior de las tablas y las referencias de la posición en que deben colocarse
	CR		
	C Column		Permite definir las referencias correspondientes a columnas a colocar en el interior de la tabla y las referencias de la posición en que deben colocarse
	CR		

N NAME			Permite asignar un nombre a una o más celdas según lo que se ha especificado en los campos del comando
CR			
SHIFT + ↑ ↓ ← →			
O OPTIONS			Permite seleccionar algunas opciones que se reflejan en el cálculo de la tabla, como recálculo automático, o la definición de una celda que contiene el «test de complemento» para las operaciones de iteración
CR			
P PRINT	**P Printer**		Imprime la tabla activa según el formato definido con los comandos Print Margins y Print Options
	CR		
	F File		Memoriza la tabla activa en un file en disco con comando imprimible
	CR		
	M Margins		Permite definir el formato de las páginas de impresión en términos de margen izquierdo, longitud, línea de impresión, números de línea por página
	CR		
	O Options		Permite definir las opciones de impresión, como: 1. Imprimir sólo una parte de la tabla 2. Imprimir fórmulas, así como valores 3. Supresión en impresión de los números de fila o de columna 4. Definición de algunos parámetros hardware
	CR		
Q QUIT			Permite salir de la sección del Multiplan en curso.Se pide la confirmación para la ejecución del comando. La tabla activa no se salvaguarda en disco
Y			
S SORT			Permite definir parte o toda la columna en que se quieren ordenar los valores
CR			La ordenación puede hacerse de forma creciente o decreciente y tanto para valores numéricos como para textos y valores lógicos
T TRANSFER	**L Load**		Permite definir un nombre de un file que se cargará en memoria para la elaboración
	SHIFT + ↑ ↓ ← →		Utilizando las teclas de movimiento del cursor y no especificando el nombre (campo vacío) se presenta la lista de los files residentes en disco y la selección del file a reclamar en memoria puede hacerse sobre la lista seleccionando el file deseado
	CR		
	S Save		Memoriza la tabla activa en disco atribuyéndole el nombre especificado
	CR		En el caso de una tabla preexistente en disco con el mismo nombre, se pide la confirmación para la ejecución del comando
	Y		
	C Clear		Cancela de la memoria la tabla activa. Se pide la confirmación para la ejecución del comando
	Y		

D Delete
SHIFT + ↑ ↓ ← →
CR

Cancela del disco la tabla especificada. Pueden utilizarse las teclas de movimiento del cursor para visualizar la lista de todos los files existentes en disco

O Options
CR

Permite definir la unidad de disco por omisión y el formato de memorización/petición de las tablas

R Rename
CR

Permite salvaguardar la tabla activa en disco atribuyéndole un nombre diferente. El comando vuelve a definir las eventuales correlaciones externas

V VALUE
CR

Introduce una fórmula

La fórmula está introducida

W WINDOW

S Split

H Horizontal
CR

Permite definir una ventana en el ámbito de la tabla. La tabla activa se divide horizontalmente

V Vertical
CR

Permite definir una ventana en el ámbito de la tabla. La tabla activa se divide verticalmente

T Titles
CR

Permite definir una ventana que contiene los títulos de las filas y columnas presentadas
La ventana activa se subdivide en dos o cuatro ventanas según que los títulos correspondan sólo a las filas, a las columnas o a ambas

B Border
CR

Permite encuadrar o eliminar el recuadro de la ventana especificada

C Close
CR

Elimina de la pantalla la ventana especificada

L Link
CR

Permite definir/eliminar conexiones entre las ventanas especificadas. La conexión implica el scroll simultáneo de las ventanas conectadas

X EXTERNAL

C Copy
CR

Introduce el número de la tabla, el RC de las celdas receptoras y la conexión

Los valores se copian de la tabla externa

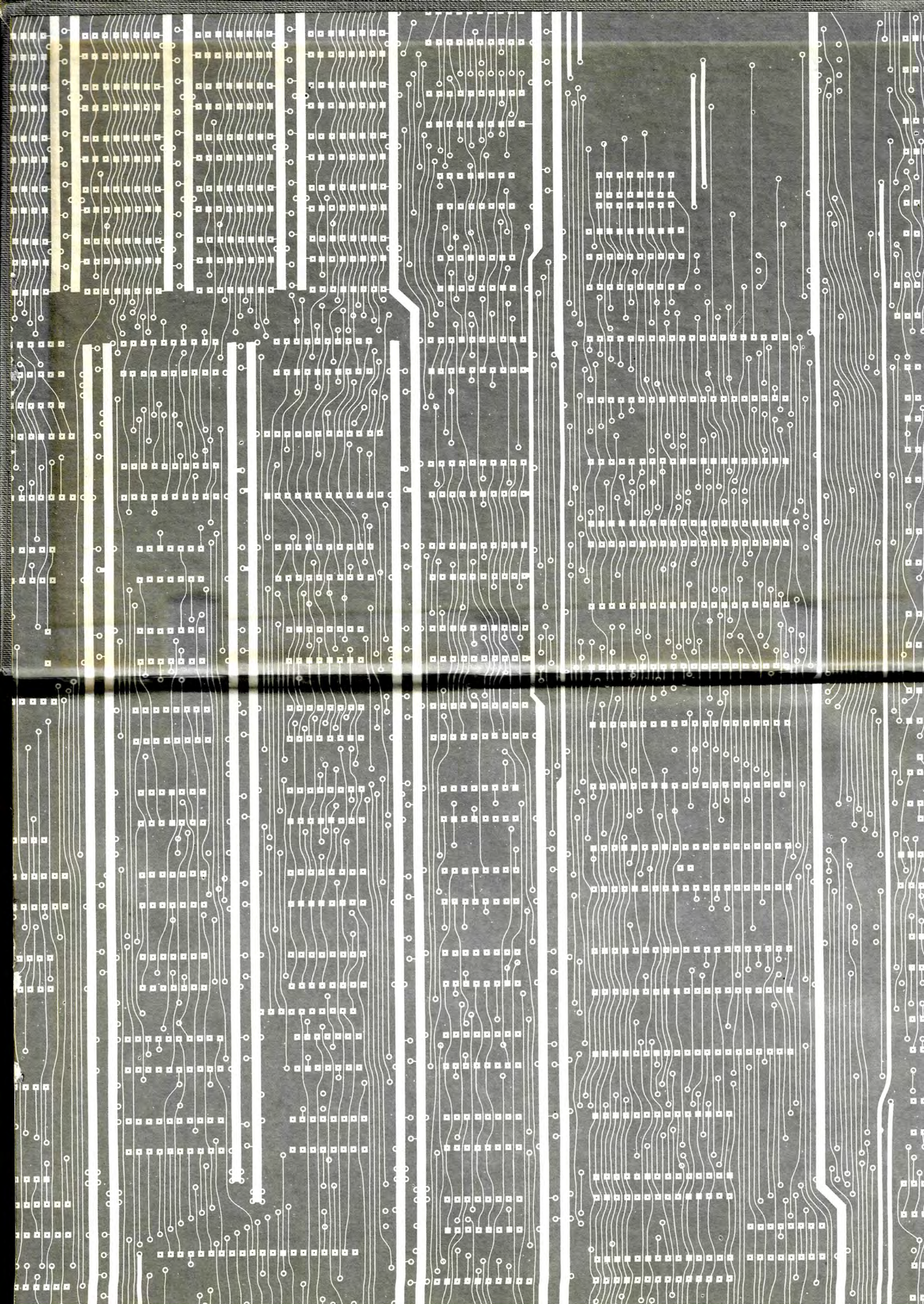
L List

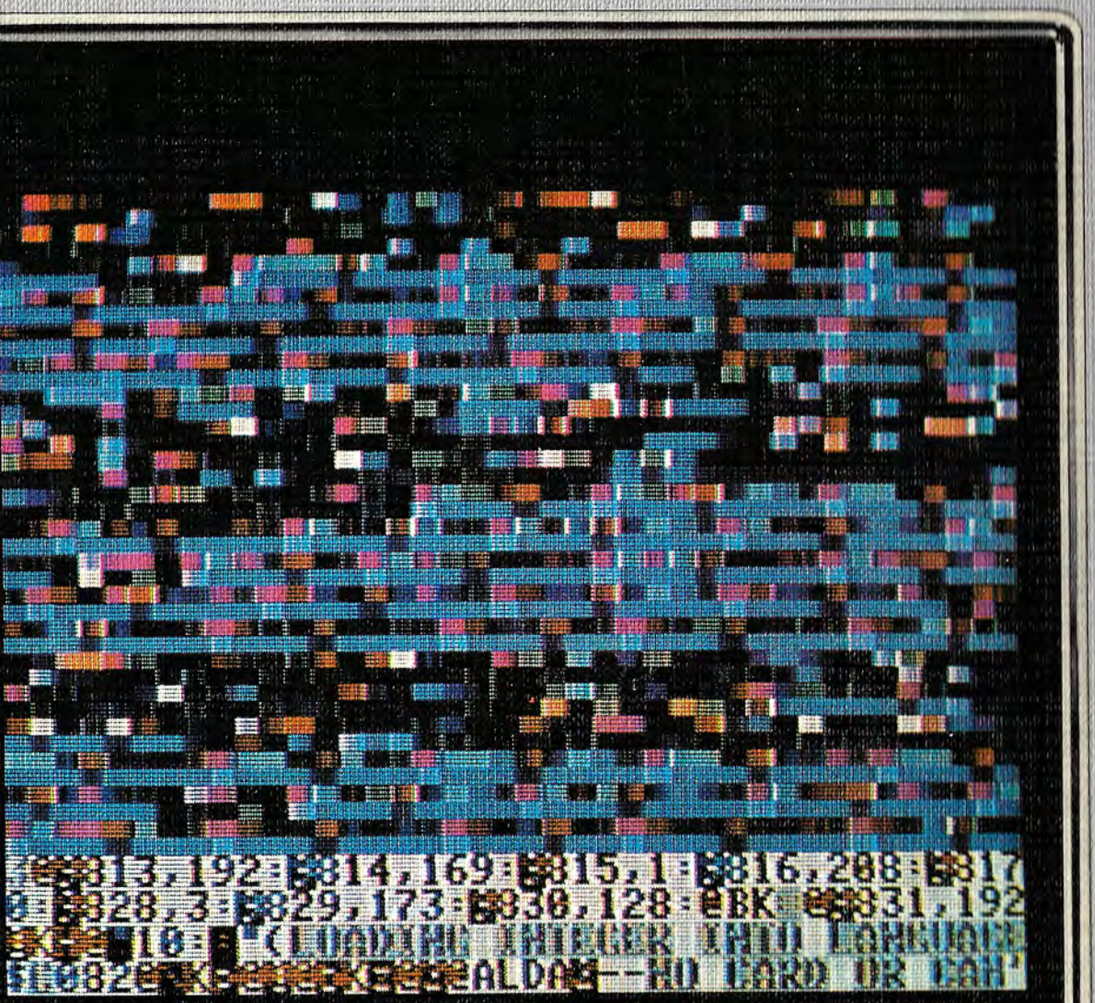
Se reproduce el nombre de la tabla que soporta la tabla activa

U Use
CR

Introduce el nombre del file

Sustituye el nombre a la tabla que se produce





EDICIONES FORUM

3

BASIC

ENCICLOPEDIA DE LA INFORMATICA
MINIORDENADORES Y ORDENADORES PERSONALES

BASIC

ENCICLOPEDIA DE LA INFORMATICA
MINIORDENADORES Y ORDENADORES PERSONALES

EDICIONES FORUM