LIST

n°11

LE JOURNAL DES AMATEURS DE PROGRAMMATION

ISSN 0761-9936

JUILLET-AOÛT 1985

PROGRAMMATION
Tester son niveau

LES CASSE-TÊTE DE L'ÉTÉ
Jeux informatiques sans ordinateur

LA BOÎTE A MALICES
Des ficelles et des programmes

AMSTRAD-FICHES
Disques, cassettes, programmes... comment gérer ses collections

CANON X-07
Où mettre un programme en langage-machine

MSX ET TRACK BALL
Le dessin facile

LE SOMMAIRE DES NUMEROS 6 A 10 DE LIST

M2712-11-20F

Belgique : 154 FB - Canada : 2.95$C - Suisse : 6 FS

20 F

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 11 · JUILLET-AOÛT 1985

SOMMAIRE

RÉDACTION-RÉALISATION
Directeur de la rédaction : Bernard Savonet
Rédacteur en chef : Jean Baptiste Comiti
Responsable de rubrique : Anne-Sophie Dreyfus
Conception graphique et secrétariat de rédaction : Eliane Gueylard
Assistante de rédaction : Maryse Gros
Administration : Marie-Hélène Muniz

Ont collaboré à ce numéro : *Olivier Arbey, Michel Arditti, Pierre Barnouin, Viviane Bazin, Robin Bois, Frédéric Boucher, Eric Bouchet, Pierre Brandeis, Laura Campagnet, Jean-Paul Carré, Thierry Chamoret, Mathieu Chante, Gilles Cizeron, Raymonde Coudert, Jacques Deconchat, Augustin Garcia, Florence Gautier-Louette, Pierre Ladislas Gedo, Laurent Gras, Patrick Gérard, Max Hagenburger, Antoine Joux, Marie-Christine Jugeau, Renée Koch, Jean-Christophe Krust, Xavier de La Tullaye, Jean-Pierre Lalevée, Hervé Le Marchand, Franck-Olivier Lelaidier, Sylvain Lemaire, Bénédicte Lizon, Thierry Lévy-Abégnoli, Alain Mariatte, Jean-Claude Martin, Pierrick Moigneau, Claude Nowakowski, Jean Ortega, Antoine Terlinden, Benoît Thonnart, André Warusfel, Frank Wettstein, Françoise Zerbib.*

Illustrations : *Philippe Burel, Antoine Chereau, Chimulus, Dik, Frapar, Bernard Helme, Sylvain Lemaire, Philippe Mairesse, Alain Mangin, Jean-Marc Rubio, Nicolas Spinga, Eric Théocharidès.*

ÉDITION-PUBLICITÉ-PROMOTION
Éditeur : Jean-Pierre Nizard
Éditeur-adjoint : Jean-Daniel Belfond
Administration : Maryse Marti, assistée d'Anne Stolkowski
Publicité : Béatrice Ginoux Defermon
assistée de Nadine Schops

VENTES
Diffusion NMPP : Béatrice Ginoux Defermon
Abonnements : Muriel Watremez assistée de Denise Martinon, Cécilia Mollicone et Sylvie Trumel.

Directeur de la publication : Jean-Luc Verhoye

LIST est une publication du groupe tests

5 place du Colonel Fabien - 75491 Paris Cédex 10
Téléphone : (1) 240 22 01 - Télex : LORDI 215 105 F

A VOS CLAVIERS

DES IDÉES... A SUIVRE

LE programme DIR/BAS de Roger Brousmiche, publié dans *LIST* 6 page 67, est d'un intérêt évident.

Pour stimuler encore plus l'imagination féconde de Roger Brousmiche, je lui soumets les idées suivantes :

- il serait intéressant de pouvoir sauver le fichier sur disque de façon à permettre une mise à jour ultérieure, pour les seules disquettes nouvelles ou qui ont été modifiées (adjonctions ou suppressions de programmes) ;
- le nom de la disquette devrait pouvoir être lu automatiquement et une rédaction en langage-machine accélèrerait notablement les opérations ;
- Enfin, un catalogue alphabétique par disquette serait le bienvenu.

Henri VOLLEAU
Nouméa (Nouvelle Calédonie)

■ *Nous transmettons votre lettre à Roger Brousmiche et nous vous remercions pour vos suggestions.*

HECTOR HRX ET LA SAUVEGARDE SUR CASSETTE

JE ne sais pas comment sauvegarder une partie de la mémoire d'Hector sur cassette. Pouvez-vous m'indiquer une routine telle qu'en donnant l'adresse de départ et le nombre de cellules 16 bits (ou d'octets) à sauvegarder, je puisse écrire ou lire une cassette ? Ceci afin de créer des fichiers...

J'attends avec impatience votre réponse.

Mathieu CHANTE
92 Saint-Cloud

■ *Sous Basic 3X, la sauvegarde est différente selon qu'il s'agit de variables ou de programmes :*

- *INPUT# variable X,X$ et PRINT# variable X,X$ pour sauver et lire la variable X ou X$;*
- *LOAD"PR" et SAVE "PR" pour sauver et lire le programme PR ;*
- *LOADC* adr 1, adr 2 *et SAVEC* adr 1, adr 2 *pour sauver et lire une partie de la mémoire comprise entre* adr 1 *et* adr 2 *;*
- *LOADD et SAVED, idem sur disquette ;*
- *USR &BFD6 sauve la totalité des variables d'un programme sur cassette ; elles seront relues par un simple LOAD.*

UN BIEN MAUVAIS SOUVENIR

LA lecture de l'article « Une vraie machine de base » pour TI-57 (*LIST* 7, page 73) m'a rappelé un bien mauvais souvenir. En effet, il y a quelques années, j'ai commis un programme semblable, aux numéros de mémoires et labels près. Naturellement, je n'en ai parlé à personne parce que ça ne marchait pas toujours, étant donné que y^x suivi de INT est mortel à (presque) tous les coups.

Je me permets donc de vous indiquer la parade. Taper le programme tel qu'il est proposé, revenir au pas 6, faire :
= DSZ GTO3
Aller au pas 14 et faire :
INS RCL7 INS STO0

Si après cela, et pour purifier l'atmosphère, vous éliminez INT du pas 9 — désormais inutile — et remplacez les deux PAUSE par R/S,vous obtiendrez un programme qui tient toujours en 37 pas, mais qui donnera, cette fois-ci, pour EF78A en hexadécimal, 3573612 en octal. C'est d'ailleurs logique, puisqu'un nombre hexa qui se termine par A donne un nombre octal qui se termine par 2.

J'espère que ceci ne vous empêchera pas de continuer à donner de temps en temps des programmes pour TI-57 (ou 58/59) qui, pour moi, sont des machines plus amusantes que les ordinateurs de poche programmables en Basic, et pas toujours moins efficaces.

Mis à part ce contretemps, *LIST* se porte bien !

François SUSANI
78 Sartrouville

■ *Merci de nous faire profiter de votre expérience. Elle nous permet de ne pas attendre pour passer l'obstacle du Y^X suivi de INT.*

Pour pouvoir continuer à donner — de temps en temps — des programmes pour TI - 57, 58 ou 59 (ou toute autre machine), nous comptons en partie sur vous : les astuces inédites ou les programmes originaux que vous nous envoyez sont étudiés et, s'ils nous semblent de nature à rendre service à nos lecteurs, nous en envisageons la publication.

LA MÉMOIRE-ÉCRAN DU X-07

CHAQUE mois, mon besoin d'informatique est assouvi grâce à *LIST* et à mon Canon X-07. C'est justement pour lui que je vous écris. En effet, il me semble avoir trouvé la mémoire-écran (de 532 à 611), mais quand je « poke » dedans, il faut que je positionne le curseur sur la ligne où j'ai poké et que j'appuie sur DEL pour voir le caractère s'afficher. Devant ma perplexité, je compte sur votre savoir informatique pour trouver l'explication qui m'amènera à la « vraie » mémoire-écran. Pourriez-vous aussi m'indiquer le moyen de protéger le BREAK sur le X-07 de manière à ce qu'on puisse stopper l'exécution d'un programme. J'attends votre réponse avec impatience.

François FINE
05 Briançon

■ *Effectivement, la mémoire-écran pour les caractères se trouve bien de 532 à 611. Mais le processus d'affichage est assez complexe (voir* LIST *7, page 71). Chaque caractère à afficher est envoyé au deuxième processeur qui gère réellement l'écran (point par point), puis il est mis en mémoire pour l'instruction SCREEN, et surtout pour certaines opérations mettant en jeu le contenu de l'afficheur (recherche du mot suivant ou du précédent, insertion, effacement).*

Lors de l'utilisation de INS ou DEL, le système doit relire et réafficher les caractères mis précédemment en mémoire, certaines variables système déterminant alors les limites de la ligne

A VOS CLAVIERS

concernée. Pour cette raison, le traitement est assez long si la ligne est importante.

Un exemple de ce qu'il est possible de faire sans routine langage-machine particulière :

```
10 FOR I = 533 TO 611 :
   POKEI,65 + C : C = C +
   1 : NEXT
20 POKE192,0 : POKE193,0 :
   POKE194,0 : EXEC 60673
```

(60673 est l'adresse de la routine de décalage vers la gauche pour effacement).

Quant au BREAK, a priori seule une routine langage-machine peut le rendre inopérant, en modifiant la routine d'interruption dont l'adresse en mémoire vive, donc accessible, est 60-61 (→51097).

ATTENTION AUX GRANDS NOMBRES

LES « coefficients du binôme » dont Bernard Kokanosky donne dans *LIST* 9 (page 61) un calcul exact portant au besoin sur de très grands nombres, sont de bons exemples de résultats qui ne peuvent être obtenus directement, même dans des limites beaucoup plus étroites, qu'au prix d'extrêmes précautions.

Tout d'abord l'application pure et simple de la formule classique : C(n,p) = n!/p!(n – p)! provoquerait ici un dépassement de capacité à partir de n = 28 (« overflow » à 1.7 D + 38 sur Epson PX8.

Avec une boucle équivalente mieux adaptée telle que :

```
10 DEFDBL K:K = 1:INPUT
   ''n,p'';N,P
20 IF P > N/2 THEN P = N – P
30 FOR I = 1 TO P
40 K = (N – I + 1)*K/I
50 NEXT:PRINT K
```

nous pourrons pousser jusqu'à C(124,61) avant de tomber prématurément en « overflow ». Pour aller plus loin, il est indispensable que le résultat soit atteint par valeurs intermédiaires toujours inférieures à lui. Nous modifierons donc la ligne 40 comme suit :

```
40 K = (N – I + 1)/I*K
```

Catastrophe : nous obtenons cette fois une avalanche de résultats fractionnaires, ce qui rend les autres fortement suspects.

Notons seulement C(56,28) = 7648690858488428 et C (130,63) = 8.943497787187532 D + 37.

Enfin ce n'est qu'avec la ligne :

```
40 K = K/I*(N – I + 1)
```

que nous atteindrons à la fois le plafond de capacité C (130,63) = 8.94349442434004 D + 37, et 16 chiffres exacts, C(56,28) = 7648690600760440.

La cascade d'erreurs d'arrondi engendrées par la formule précédente suffisait donc à fausser une bonne moitié des seize chiffres de la double précision du PX8.

Considérons maintenant la boucle suivante :

```
10 A$ = ''WHILE...WEND''
20 FOR I = 1 TO 0 : A$ =
   ''REPEAT...UNTIL'':NEXT
30 PRINT ''FOR.NEXT = '';A$
```

Avec certains Basic, nous vérifierons que les boucles ''FOR... NEXT'' se comportent bien comme des ''REPEAT...UNTIL'', mais d'autres testent les bornes supérieures avant de parcourir la boucle, et se comportent à cet égard comme avec ''WHILE...WEND''.

Les bogues qui peuvent en résulter sont particulièrement vicieuses parce qu'intermittentes le plus souvent. L'exemple ''FOR I = 1 TO 0'' peut paraître absurde, mais vous le rencontrerez assez souvent sous la forme ''FOR I = A TO B'' avec A et B variables où B devient parfois plus petit que A.

Charles DOARÉ
06 Grasse

ARTISTE OU SCIENTIFIQUE ?

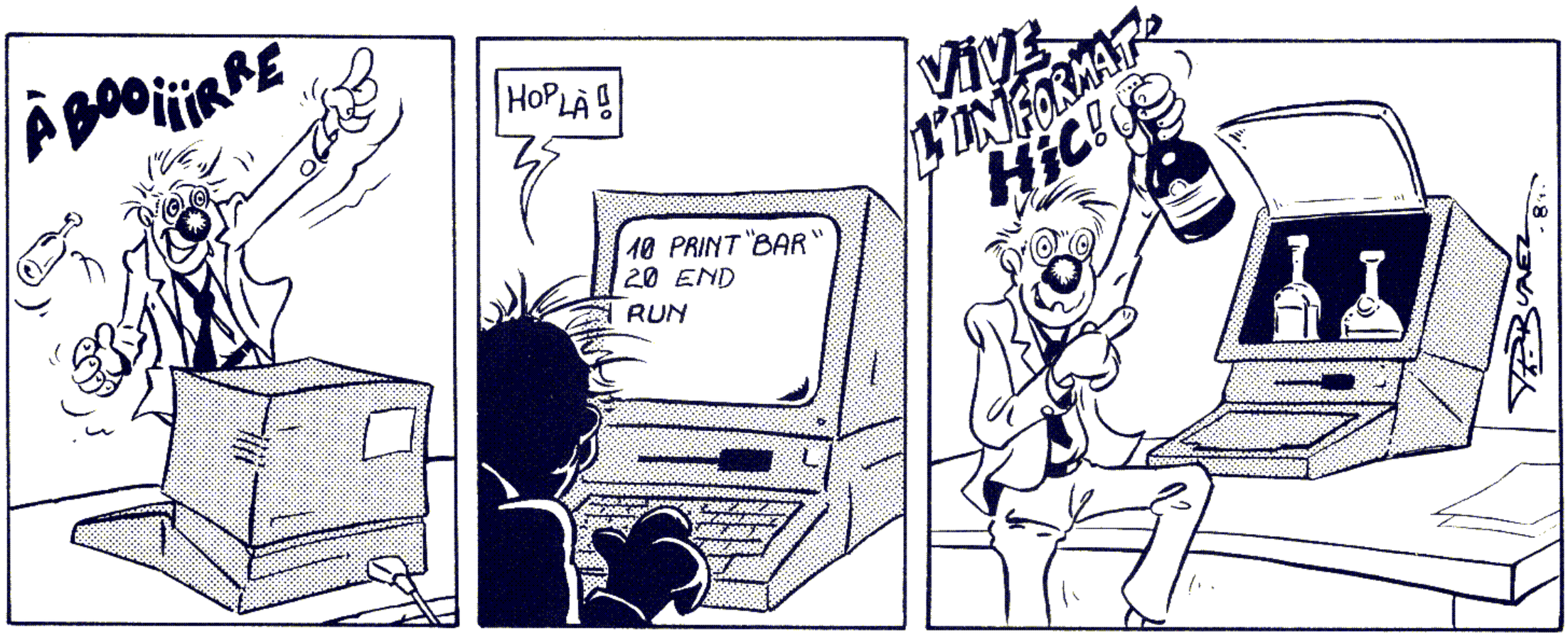
DANS son numéro daté de mai 85, la revue américaine *Computer language* consacre plusieurs pages très intéressantes à la philosophie de la programmation.

On y trouve notamment deux longs entretiens : l'un avec Niklaus Wirth (1), l'autre avec Donald Knuth (2). Ce dernier y aborde la question de savoir si la programmation est un art ou une science, et si le programmeur doit être original et créatif ou s'il doit surtout veiller à la clarté et à la « maintenabilité » de son œuvre.

Sur ces questions et toutes celles qui en découlent (« To goto or not to goto », par exemple), certains d'entre vous ont certainement une opinion personnelle, un point de vue particulier. Si vous voulez bien nous en faire part par écrit, sachez que nous vous lirons avec un grand intérêt, et d'avance nous vous en remercions.

LIST

(1) Niklaus Wirth est le père du langage Pascal, d'Algol W et de Modula-2.
(2) Donald Knuth est réputé pour ses études sur les algorithmes et spécialement pour les trois tomes de The Art of Computer Programming *qui est une sorte de bible en la matière.*



LA GAZETTE DE LIST

Microtom : une revue consacrée aux Thomson

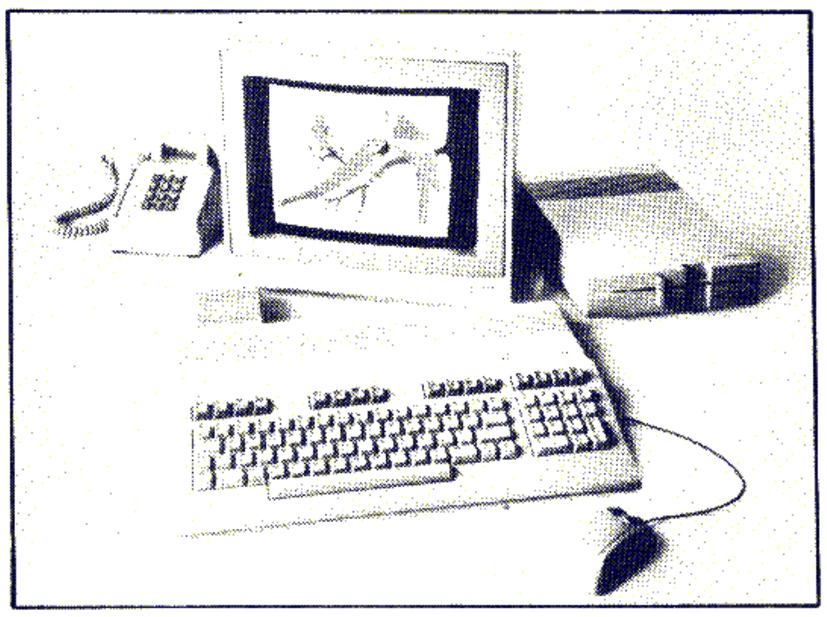
SOIXANTE-HUIT pages qui ne parlent que des TO 7, MO 5 et autres TO 7-70 : leurs logiciels, les livres qui les concernent, les nouveautés qui, de près ou de loin, ont un rapport avec eux et, bien sûr, des astuces et des programmes. Microtom est en vente dans les kiosques au prix de 28 FF. ■

Vu et entendu au Computer Electronic Show

RIEN de vraiment sensationnel n'a marqué le Computer Electronic Show (CES) 1985 qui s'est tenu du 2 au 5 juin dernier à Chicago (Illinois).

Les ordinateurs au standard MSX ont brillé par leur absence et **Philips**, qui avait retenu un stand, s'est même décommandé. D'autre part, il semble bien que la commercialisation des **Commodore** 16 et Plus 4 soit arrêtée aux États-Unis. Quelques produits intéressants ont cependant retenu notre attention.

Video Technology présentait deux modèles : le Laser 50, à moins de 100 $ et le Laser 3000. Ce dernier (64 Ko de mémoire vive et 24 Ko de mémoire morte - Basic Microsoft) est déjà disponible en France à 4 980 FF, ce prix comprenant un lecteur de disquettes. Il peut être configuré pour faire tourner directement les logiciels de l'Apple II. Une carte Z 80, fournie en option, lui donne accès à CP/M 80. Gonflé au maximum, il peut aller jusqu'à 1922 Ko de mémoire vive. En mode graphique, sa définition est deux fois supérieure à celle de l'Apple II : 560 × 192 points.



Le nouveau C. 128 de Commodore

Le Laser 50, ordinateur de cartable (15 × 27,2 × 5 cm), n'est pas encore introduit sur notre marché. S'il dispose d'un véritable clavier mécanique, son affichage à cristaux liquides est limité à une ligne de 16 caractères. La mémoire vive (1,5 Ko) peut être étendue jusqu'à 16 Ko par l'ajout de modules de 4 Ko (19,50 $). Autonome, il fonctionne sur batteries.

Commodore semble abandonner certains modèles au profit du C. 128 et de l'Amiga (ce dernier devrait concurrencer le 520 ST d'**Atari,** mais il joue très bien l'Arlésienne). Le 128, doté de deux microprocesseurs, dont un Z 80, a de multiples facettes : c'est d'abord un Commodore 64 (64 Ko de mémoire morte) compatible avec tous les périphériques et logiciels de celui-ci. C'est aussi un ordinateur au Basic très étendu et très puissant qui fonctionne avec 128 Ko de mémoire vive dans sa version de base. Enfin, son Z 80 lui permet, en accédant à CP/M, de disposer d'une impressionnante bibliothèque de programmes. Son prix : moins de 300 $.

Le CPC 664 ne sera pas resté longtemps le dernier né d'**Amstrad,** arrivé au CES avec le CPC 6128, proposé en deux configurations. Toutes deux comprennent outre 128 Ko de mémoire vive, deux systèmes d'exploitation de disquettes (CP/M et Amsdos), deux langages (Basic et Logo), une unité de disquettes incorporée et un moniteur.

Les différences résident dans les moniteurs et les logiciels fournis : écran couleur (640 × 200), traitement de texte Amstrad et programme de jeu dans un cas ; écran monochrome vert et Wordstar dans le deuxième cas. La première version était annoncée à 799 $ et la seconde à 699 $. ■

CHEZ LE LIBRAIRE

L'Amstrad avec plaisir : comment faire de bons programmes
R.A. et J.W. Penfold
Traduit par Pierre Maurice
Editions Edimicro
1985, 116 pages
Format poche
Prix : 59 FF

UN petit livre au format de poche qui est la traduction d'un ouvrage anglais *(An introduction to programming the Amstrad CPC 464).* Il résume les principales instructions de la machine avec explications et exemples.

La conduite des Alice 32 et Alice 90
François Bernard
Editions Eyrolles
1985, 174 pages
Prix : 99 FF

APRÈS un bref aperçu sur les commandes et les instructions du Basic, l'auteur passe à des choses plus sérieuses : architecture du système et carte de la mémoire. Une étude du microprocesseur 6803, de ses registres internes et de son jeu d'instructions, permet de s'initier à la programmation en Assembleur sur Alice 32 et 90.

Le Logo en pratique
Jean-Pierre Regourd
et François-Xavier Testard
Vaillant
Editions du PSI
1985, 136 pages
Prix : 95 FF

DES programmes d'application commentés permettent de mettre en pratique les connaissances du programmeur en Logo : jeux (pendu, master-brain) et graphisme (courbes fractales), mais aussi maths, tests de connaissance, inversion vidéo, etc.

Apple chouchoute les étudiants

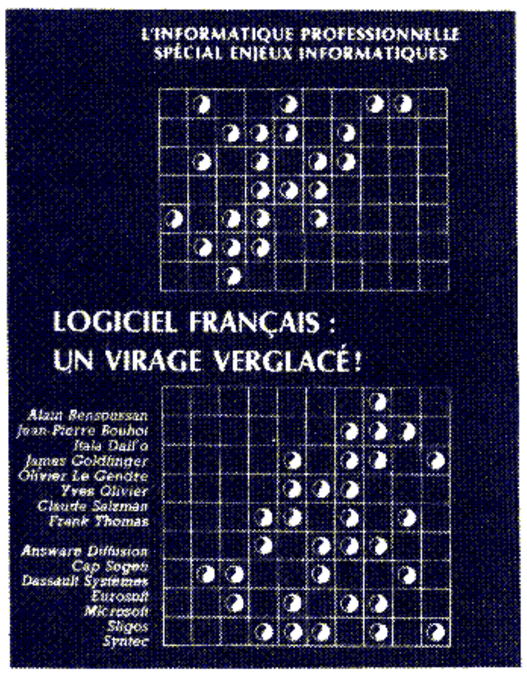
SI vous voulez avoir une chance d'acquérir pour 9990 Francs ttc un Apple IIc (et son sac de transport) accompagné du logiciel Apple Works, il faut vous dépêcher. Cette proposition spéciale, qui s'adresse aux étudiants, s'achèvera avec les commandes reçues au 12 juillet 1985.

N'oubliez pas de joindre à votre demande la photocopie de votre carte d'étudiant en cours de validité ou un certificat de scolarité original et d'indiquer la référence « A2c-été » de cette offre limitée.

Apple Séedrin
Z.A. de Courtabœuf
Avenue de l'Océanie - BP 131
91944 Les Ulis Cedex
(6) 928 01 39 ■

LA GAZETTE DE LIST

UN LIVRE



Logiciel français :
un virage verglacé !
Ouvrage collectif
Numéro spécial de *L'informatique professionnelle*
N° 31 - mars 85, 130 pages
Prix : 75 FF

CETTE étude fait suite à celle qui fut, l'année dernière, consacrée aux constructeurs de matériel (Micro-informatique française : dur, dur...). Elle est plus optimiste que la précédente puisqu'elle traite de l'industrie du logiciel en France et que nous avons dans ce domaine, une place enviée. La France exporte en effet une grande partie de son savoir-faire. Mais, malgré sa haute technicité, le logiciel français se trouve à un tournant qui pourrait se révéler difficile à prendre. Le problème vient principalement du fait que, si nous réalisons d'excellents produits sur les grosses machines, nous sommes souvent à la traîne, dans le domaine de la micro-informatique, par rapport à ce qui se fait aux États-Unis.

Si vous vous intéressez à l'industrie du logiciel, soit parce que vous allez y entrer, soit parce que vous en faites déjà partie, ce numéro spécial de *L'informatique professionnelle* vous concerne. Après tout, avoir l'avis de Jean-Pierre Bouhot, de Bernard Laur (Answare Diffusion), de Philippe Dreyfus (Cap Gemini Sogeti), de Michel Vergnes (Microsoft) et de bien d'autres est loin d'être inintéressant. En prime, vous apprendrez (avec humour) comment gagner votre premier milliard grâce à l'informatique (enfin, c'est ce que nous vous souhaitons...). TC ■

Jack Tramiel présente le 520 ST à Paris

L'ATARI 520 ST devrait être disponible en France dès le mois de juillet 85, annonce Atari en présence de Jack Tramiel lui-même. Sa configuration de base comprendra un clavier Azerty, 512 Ko utilisateur, un système d'exploitation Gem francisé, une souris, un écran monochrome haute résolution (640 sur 400 points), un Basic, un Logo, un traitement de texte (Gemwrite), un logiciel graphique (Gempaint), un lecteur de disquettes (3 pouces 1/2). Cet ensemble pour moins de 10 000 F ttc. Les autres périphériques arriveront plus tard. ■

Festival du logiciel de Villeneuve-lez-Avignon

DU 15 au 27 juillet, la grande Chartreuse de Villeneuve accueillera, pour la troisième année consécutive, les participants au Festival du logiciel (créé par RTL, l'Ordinateur Individuel et le CIRCA).

Les visiteurs, qui pourront y essayer près de 300 logiciels, seront invités à donner leur appréciation ; celle-ci sera prise en compte par le jury pour le classement final. Des journées professionnelles sont prévues réunissant auteurs, éditeurs et constructeurs.

Ces derniers se retrouveront le 24 juillet à 21 h 30 pour la soirée des auteurs de logiciels.

Festival du logiciel
La Chartreuse
Espace de la Boulangerie
30400 Villeneuve-lez-Avignon
(90) 25 05 46

Entrée : 10 FF

UNE CASSETTE

Bêta Basic
Cassette pour
ZX Spectrum 48 ko
Edité par Infogrames
Prix : 190 FF

LES Basic les plus récents sont souvent très étendus. Celui du Spectrum, qui commence à dater, peut néanmoins s'enrichir grâce à des ajouts qui lui donnent une nouvelle jeunesse.

Bêta Basic, que propose Infogrames, est simple d'emploi et met à la disposition de l'utilisateur 30 commandes et environ 25 fonctions supplémentaires. La cassette ne peut être utilisée que sur un Spectrum 48 Ko, mais la notice (36 pages) explique de façon détaillée comment réaliser une sauvegarde sur microcartouche : c'est un bon point.

Les possibilités offertes par le programme sont attrayantes. On retrouve, bien sûr, les grands classiques : AUTO, RENUM, DELETE, IF... THEN... ELSE, FILL (pour colorier une figure), GET (pour attendre la frappe d'un caractère), PRINT USING, TRACE, ON ERROR, etc.

Ces nouvelles instructions sont obtenues en mode KEYWORD 1, mode normal de fonctionnement : on les introduit directement en mode graphique (après avoir fait CAPS SHIFT et GRAPHICS). Les mots clés apparaissent à l'écran. On peut toutefois visualiser les caractères graphiques et les figures définies par l'utilisateur en

passant en mode KEYWORD 0.

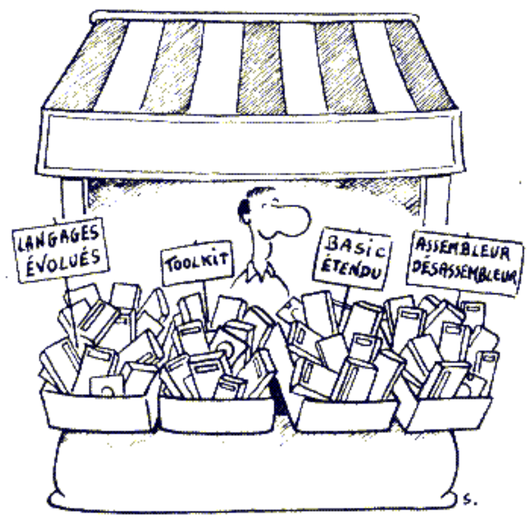
Voyons maintenant les aspects les plus originaux de cette extension qui en font une sorte de super Basic. De nouvelles structures de boucle sont possibles avec DO, DO WHILE, DO UNTIL et LOOP, ou encore LOOP WHILE ou LOOP UNTIL... avec EXIT IF pour sortir convenablement d'une boucle et POP pour faire sauter un niveau de retour de sous-programme. JOIN et SPLIT permettent de réunir ou de séparer deux lignes de programme et DEF KEY attribue à n'importe quelle touche une chaîne de caractères. Avec DEF PROC, on définit des procédures terminées par END PROC ; SORT, effectue des tris ultra-rapides. Enfin, l'association de POKE et de la fonction MEMORY$ permet de remplir ou de déplacer instantanément de vastes portions de mémoire vive.

Certaines fonctions "classiques" ont été améliorées : ainsi PLOT peut désormais placer une chaîne de caractères à n'importe quel endroit de l'écran haute résolution, et il devient possible de simuler des « lutins » avec des déplacements pixel par pixel. SCROLL et ROLL, quant à eux, autorisent des manipulations de l'écran dans quatre directions avec ou sans déplacement.

Cette visite un peu superficielle vous a-t-elle mis en appétit ? Alors, quand vous saurez qu'une ligne 0 contient plus de 20 fonctions nouvelles accessibles par FN..., qu'il existe une horloge consultable à tout moment, que l'on peut reconnaître sur l'écran les caractères définis par l'utilisateur, qu'un programme peut être listé de la ligne A à la ligne B, etc., vous vous demanderez certainement pourquoi ces instructions ne figuraient pas dans le Basic d'origine.

Revers de la médaille : ces extensions ne peuvent pas être utilisées sans le logiciel *Bêta Basic*. Il n'est en effet pas possible, contrairement à d'autres Basic étendus, de sauvegarder à la fois le programme réalisé avec les nouvelles instructions et le Basic. Cela limite les possibilités de diffusion d'un programme et donc l'intérêt du logiciel. Mais les incontestables améliorations apportées par cette cassette (meilleure vitesse d'exécution des programmes, plus grande facilité d'écriture et de lecture, moins d'utilisation de sous-programmes en langage-machine) peuvent justifier totalement cet investissement pour les amateurs de programmation. JD ■

LOGICIELS



Pascal Base
Pascal compilé
Cassette pour TO 7
+ extension 16 Ko
TO 7/70 et MO 5
Edité par Free Game Blot
Prix : 195 FF

LA vocation première de ce logiciel, accompagné d'une petite notice d'une cinquantaine de pages, est l'initiation au Pascal. Les principales caractéristiques du langage sont disponibles : structuration, récursivité, modularité des sous-programmes et fonctions. Seule la structuration des données fait défaut. Outre un compilateur et un éditeur, Pascal Base met à la disposition de l'utilisateur un « chargeur » destiné à créer, à partir d'un programme intermédiaire issu d'une compilation, un programme exécutable par la machine.

X-07 Forth
Langage Forth
Cassette pour X-07 16 Ko
Auteurs : Christophe Aemmer
et Patrick Leclercq
Edité par Pocketsoft
Distribué par X-Log
Prix : environ 300 FF

LA notice (une trentaine de pages) qui accompagne le coffret n'étant pas destinée à l'apprentissage du langage, le débutant devra faire l'acquisition d'un ouvrage d'initiation. Compatible avec l'interface Péritel, X-07 Forth dispose, en plus des possibilités classiques du Forth, de fonctions spécifiques au Canon et propose sur la face B deux bibliothèques d'extensions : traitement de chaînes alphanumériques et gestion de l'imprimante graphique.

Supercopy
Utilitaire de copie d'écran
Cassette pour Amstrad
Edité par Cobra Soft
Prix : 120 FF

EN mode graphique, Supercopy permet de reproduire sur DMP-1 les figures créées à l'écran. En mode texte, le logiciel est compatible avec toutes les imprimantes possédant une interface Centronics (attention, seuls les caractères ASCII sont reproduits sur le papier). Enfin, un tampon appelé « spooler » en anglais (sorte de salle d'attente en mémoire où sont stockés les caractères à imprimer) permet d'utiliser l'ordinateur dans le même temps que s'effectue la copie d'écran.

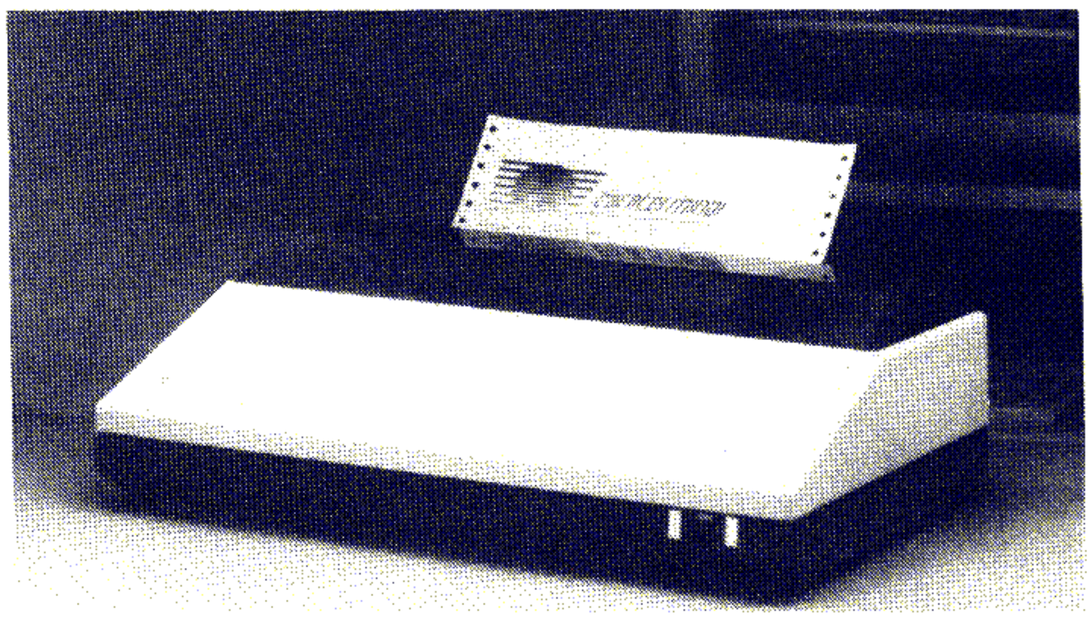
Quickcopy
Utilitaire de copie rapide
Disquette pour C.64
Edité par Micro Application
Prix : 295 FF

QUICKCOPY s'utilise avec le périphérique 1541 et fonctionne avec une ou deux unités ; il effectue rapidement les copies de disquettes avec ou sans vérification. L'option formatage est également disponible.

Calcul
Programme éducatif
Cartouche pour MSX
Edité par Hal (Japon)
Distribué par Maubert
Electronique
Prix : 295 FF

POUR redresser les rails sur lesquels le train de sa fiancée va passer, Mogura doit vite donner les réponses aux opérations proposées. Les enfants qui manient fort bien les poignées de jeu seront assez vifs pour utiliser ce programme qui requiert aussi de bonnes qualités de calcul mental. Les autres auront peut-être plus de mal. Le plus dur n'est pas de trouver la solution des opérations qui sont de plus en plus complexes, mais de hisser le résultat jusqu'à la case qui rétablit la travée où le train s'engage. Un jeu qui demande suffisamment d'attention pour nous faire oublier les efforts de calcul mental qu'il impose.

Une imprimante avec interface vidéotex



CONÇUE par **Euroterminal**, fabriquée par la **CGCT**, l'EXL 80-PC est compatible IBM PC, mais peut également fonctionner avec des familiaux, comme l'EXL 100 et le TO 7, par le biais d'interfaces série (RS-232 C) ou parallèle (type Centronics).

Il s'agit d'une imprimante matricielle à impact (80 ou 132 colonnes) qui dispose de 256 caractères alphanumériques et semi-graphiques. Munie d'origine d'une interface parallèle, elle peut en option devenir compatible vidéotex.

Pour équiper un EXL 100, il en coûte environ 4 300 FF ttc, version interface parallèle ou 4 900 FF, version interface série. Vous ajouterez 595 FF si vous choisissez l'option vidéotex.

Les modèles compatibles IBM PC et TO 7 sont au même prix : près de 4 570 FF, version série, environ 5 160 FF, version parallèle et toujours 495 FF pour l'interface vidéotex.

La fabrication du même modèle en couleurs est prévue. Euroterminal annonce également la venue d'une mini-imprimante thermique (elle est actuellement en cours d'industrialisation), baptisée EXL 40, dotée de toutes les fonctions vidéotex et d'une taille mémoire importante (16 Ko de mémoire tampon) qui devrait être commercialisée aux alentours de 1 900 FF : un rapport performance/coût qui, s'il se confirme, semble tout à fait intéressant. ■

CHEZ LE LIBRAIRE

Assembleur et Périphériques des MSX
Pierre Brandeis
et Frédéric Blanc
Editions du PSI
1985, 204 pages
Prix : 110 F

APRÈS une première partie d'initiation à l'Assembleur du Z80 suivie de la description du microprocesseur et de son jeu d'instructions (plus de 700) classées par groupe, on trouvera une description des différents périphériques MSX et leur programmation en Assembleur (processeur vidéo, générateur sonore, circuits d'entrée/sortie, etc.). En annexe, on trouvera, entre autres, un tableau d'assemblage et une table de conversion binaire-décimal-hexadécimal.

Le nouvel Atari ST
Edité par Micro Application
1985, 154 pages
Prix : 129 FF

Il s'agit vraiment d'une avant-première puisque le livre est en vente avant la machine !... On y passe en revue les possibilités du 520 ST et de son microprocesseur (68000 de Motorola) : l'architecture générale, les interfaces, le système d'exploitation, la souris et les langages.

LA GAZETTE DE LIST

CHEZ LE LIBRAIRE

Assembleur et périphériques des MO 5 et TO 7/70
Frédéric Blanc et François Normand
Editions du PSI
1985, 124 pages
Prix : 85 FF

L'OUVRAGE, consacré surtout au MO 5, débute par une table des mnémoniques renvoyant aux pages où elles sont traitées. Les particularités du TO 7/70 sont placées en annexe. Après une présentation du langage-machine, du système MO 5 et des instructions du 6809, les auteurs donnent les routines et adresses utiles pour l'utilisation des périphériques (écran, crayon optique, joystick, clavier, magnétophone, lecteur de disquettes, etc.).

Créez vos jeux d'aventures sur micro-ordinateur
Méthodes et idées
Jean-Marc Pezeret
Editions Eyrolles
1985, 140 pages
Prix : 98 FF

UNE analyse des ingrédients nécessaires à la réalisation d'un programme d'aventure : le scénario, la structure, les données, jusqu'à la présentation de la liste. Un exemple, développé au fil des chapitres, illustre les principes décrits.

La programmation des jeux d'aventure
Gérald Anfossi
Editions du PSI
1985, 122 pages
Prix : 90 FF

LA réalisation d'un jeu d'aventure envisagée comme une recette. A partir des éléments dont il dispose (ses idées, les règles à suivre pour réussir et les possibilités de la machine), le joueur concocte pour lui-même et pour les futurs amateurs, un parcours semé d'embûches. Un programme proposé par l'auteur (Prisonnier à Lock Largo) sert de base à l'étude.

FX-4000 P, une excellente calculatrice

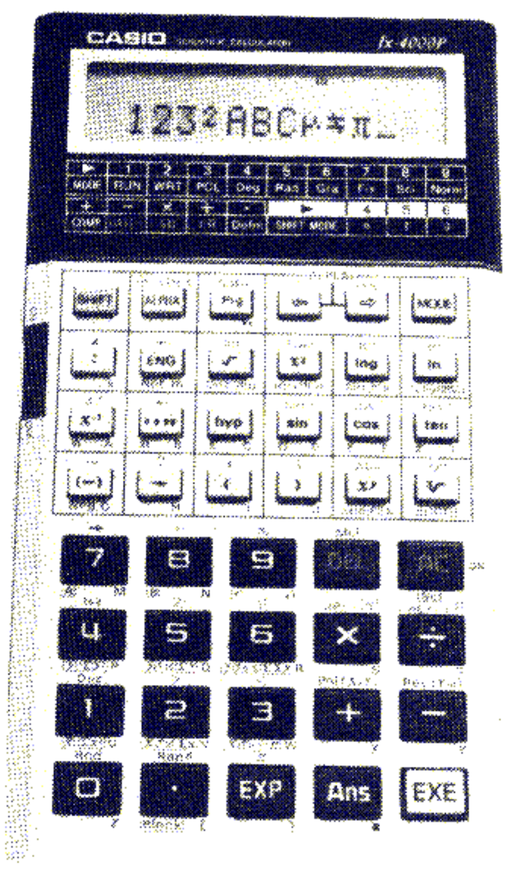
LES ordinateurs de poche, même ceux qui parlent Basic, ne font plus l'objet de l'engouement d'il y a trois ans. Que Casio présente une nouvelle calculatrice programmable surprendra peut-être avant que l'on ne découvre de quelle calculatrice il s'agit. La programmation ne vient qu'en complément de modes de calculs multiples et musclés.

Il semble bien que les calculatrices programmables en Langage Machine Spécialisé soient condamnées. Pour le moment, le Basic a supplanté le LMS qui n'est pas toujours évident à exploiter ; cet état de fait est regrettable pour les utilisateurs préoccupés en priorité par la simplicité d'exécution des calculs.

Si la Casio FX-4000 P possède un petit côté rétro avec sa programmation en LMS, le mode de calcul a été poussé très loin et présente des facilités et une philosophie générale très plaisantes. L'aspect extérieur de la machine rappelle beaucoup celui des 502 et 602 P : présentation verticale, prédominance du clavier de type calculatrice et afficheur monoligne alphanumérique de douze caractères. La partie supérieure de ce dernier est occupée par 19 indicateurs : un vrai tableau de bord d'avion qui augure bien de la capacité de la machine. Et, de fait, on dispose en plus du mode de calcul classique, de trois autres modes distincts : base n, statistique et régression linéaire.

Le premier des quatres modes réalise les opérations habituelles d'une calculatrice scientifique musclée. Il opère sur douze chiffres significatifs dont dix sont affichés. Jusque-là, rien de nouveau. Mais on sort un peu des sentiers battus avec le tampon d'entrée des calculs qui accepte 79 pas et qu'il est possible d'éditer même après la demande du résultat. On peut aller y corriger une erreur, insérer ou supprimer des pas. Plusieurs séquences d'opérations sont susceptibles d'y cohabiter avec affichage des résultats intermédiaires. C'est une réussite.

Autre attrait, une pile algébrique qui accepte huit niveaux de valeurs numériques et vingt niveaux d'opérations (ou de commandes). Avec elle, on tape les formules de calcul comme on les écrirait sur une feuille de papier, d'autant plus simplement que les priorités de la hiérarchie algébrique ont été étendues. Pour faciliter encore les choses, les mémoires ne sont plus désignées par des identificateurs numériques mais, comme en Basic, par des lettres (avec



éventuellement des indices), et le symbole d'affectation n'est pas le signe égal (souvent générateur de confusion).

Le remplissage des mémoires est réalisé par la flèche à droite, sous la forme 123 → A. La variable est indiquée après le nombre à conserver, ce qui aide a posteriori le stockage d'un résultat.

A la mise en route, 26 tiroirs à données sont disponibles (lettres de A à Z). Ce nombre peut monter jusqu'à 94 par désallocation de tout ou partie de la mémoire programme. Les variables supplémentaires sont notées Z(1), Z(2), etc. Une touche Ans rappelle un registre temporaire contenant le résultat du dernier calcul effectué et autorise son inclusion dans une formule.

Le deuxième mode de calcul, base n, s'occupe des cas du binaire, de l'octal, du décimal ou de l'hexadécimal. Il traite également les conversions entre ces différentes bases et les opérations logiques. Une redéfinition d'une partie du clavier facilite l'entrée des données hexadécimales et des opérateurs logiques. Ce mode, à lui seul, suffirait à justifier l'achat de la FX-4000 P.

Les deux suivants s'occupent de statistiques sur un seul ensemble ou sur deux ensembles corrélés (régressions linéaires, exponentielles, logarithmiques ou de puissances). Ici encore, le clavier change de définition pour s'adapter au type de travail et rendre plus simple l'introduction des données.

Reste enfin la programmation qui permet de mémoriser les séquences de calculs. Le langage reste rudimentaire, comme tous les LMS, mais les possibilités sont suffisantes : 550 pas de programme que l'on peut répartir en dix zones indépendantes mais capables de communiquer. Une définition d'étiquettes (Lbl), des branchements (Goto), des tests et des boucles compteur (Isz, Dsz) sont disponibles. Cela ne représente pas une débauche d'instructions, mais ce mode a été conçu comme complément d'une calculatrice performante et, à ce titre, il apparaît suffisant pour la majorité des applications. Une seule remarque : il n'y a pas de connecteur pour imprimante, et l'absence de cet élément ne permet pas à la FX-4000 P de mettre parfaitement en évidence ses excellentes qualités de calcul. Dernier point, le prix qui reste modéré : vous devriez trouver la nouvelle calculatrice de Casio aux alentours de 600 FF.

XdLT ■

Machine à écrire et imprimante

AU Spécial Sicob, Canon présentait la S-50 R, une machine à écrire portable (alimentée par piles). Elle possède un véritable clavier mécanique qui n'est malheureusement pas francisé. Reliée à un ordinateur muni de l'interface RS-232 C, elle devient imprimante. L'impression se fait par transfert thermique et produit des documents de qualité courrier. La S-50 R coûte 2 400 FF. ■

DEUX LIVRES

Logic Basic
Patrick Senicourt
et Michel Massiou
Editions du PSI
Tome 1
1983, 208 pages
Prix : 130 FF
Tome 2
1984, 276 pages
Prix : 165 FF

IL est rare de trouver une étude de la programmation structurée dans le domaine de la micro-informatique. Logic Basic s'adresse à ceux qui pensent que la programmation ne doit plus être un art mais doit satisfaire, pour son efficacité, à des normes de type industriel.

Les auteurs ont choisi d'appliquer une méthodologie au Basic, langage privilégié des micro-ordinateurs : la méthode Warnier.

Le premier tome expose avec une précision remarquable les principes de la méthode ; on y apprend à bien programmer. Le lecteur trouvera, entre autres, des règles et des conseils pleins de bon sens. Les grandes structures répétitives, alternatives et même la structure élémentaire y sont présentées sous des formes décomposées.

Dans le second tome sont abordées toutes les manipulations de fichiers : accès direct, séquentiel et même l'organisation séquentielle indexée.

L'idée d'appliquer la méthode Warnier au Basic était bonne et pédagogique. Le résultat nous semble cependant très lourd. On s'intéresse à la forme des structures et non au fond, à leur fonction, au sens du programme.

Si l'on nous recommande la décomposition du problème en sous-programmes jusqu'à n'avoir plus qu'un ensemble de structures, on n'indique pas suivant quel principe procéder. Les programmes qui en résultent sont souvent confus et encore trop techniques. Et que dire des noms de variables sans signification ?

On peut regretter que les auteurs aient choisi une méthode un peu vieillotte. Elle a quinze ans, et beaucoup de choses ont évolué depuis ses débuts.

Logic Basic intéressera ceux qui pensent qu'il faut absolument se mettre à la programmation structurée (les livres sur le sujet sont extrêmement rares), mais la méthode est plutôt maladroite et encore incomplète.

MH ■

DU CÔTÉ DES CLUBS

Dans les Hauts-de-Seine

A Genevilliers, le tout jeune club des Yenophiles, association sans but lucratif déjà forte de 55 adhérents, invite tous les utilisateurs de Yeno SC 3000 et de DPC 64 (MSX) à venir grossir les rangs de ses adhérents. La cotisation est trimestrielle et se monte à 100 F, moyennant quoi il sera possible de participer aux activités du club et de recevoir son bulletin mensuel.

Une permanence téléphonique est assurée tous les mercredis de 14 h à 18 h 30. Sachez enfin que les Yenophiles utilisent un modem. Si c'est aussi votre cas, vous pourrez donc échanger des informations par téléphone. Pour plus de renseignements, vous contacterez Philippe Tolokonnikoff et Cyril Cellier au 792 07 98.

Club des Yenophiles
86/108 avenue Louis-Roche
92230 Gennevilliers

Dans le Var

DEUX fois par semaine, les amis de l'informatique de Hyères se retrouvent au sein de la RAM (Réunion des Amis de la Micro-informatique), un club créé autour du matériel Amstrad. La cotisation d'adhésion s'élève à 200 F pour six mois. L'association hyéroise, qui se réunit au Parc Hôtel tous les lundis et vendredis à partir de 20 h, se divise en trois départements : initiation, étude de thèmes (par exemple, le graphisme), et formation des adhérents à l'utilisation des logiciels qui fonctionnent sur Amstrad.

Pour en savoir davantage, vous pouvez écrire ou téléphoner au président de la R.A.M., Eric Destribois.

LEP Golf Hôtel
83400 Hyères
(94) 57 39 66 poste 23

Des vacances studieuses

Dans l'Ardèche

SI vous n'avez pas encore retenu vos dates de vacances, vous serez peut-être intéressé par l'un des stages que propose Microtel Ardèche Sud. Pendant treize jours, vous pourrez allier travail et détente (initiation à l'informatique, piscine, tennis, randonnées pédestres, descente des gorges de l'Ardèche, etc.) pour 2 270 F tout compris. Les stagiaires sont hébergés dans une maison familiale.

Vous avez le choix entre trois périodes : du 15 au 27 juillet, du 29 juillet au 10 août ou du 12 au 24 août 1985. Pour obtenir un dossier d'information (n'hésitez pas à écrire même s'il est un peu tard, il peut rester quelques places), envoyez votre demande à :

Microtel Ardèche Sud
La Croix de Malet
BP 36
07110 Largentière

Dans l'Aude

L'association CERA (Centre d'Etude et de Recherche Audiovisuel) organise, au Château des Cheminières de Castelnaudary, une initiation au Basic (du 5 au 9 août) et un stage de perfectionnement à l'informatique (du 12 au 16 août). Les deux stages, bien que totalement indépendants, peuvent être suivis par les mêmes personnes, chacune d'elle ayant un ordinateur à sa disposition, TRS-80, TO 7, Amstrad ou Sanyo.

Le prix de la première session est de 1 300 F, celui de la seconde de 1 700 F auxquels il faudra ajouter 600 F si vous souhaitez une prise en charge pour les repas et l'hébergement en chambre individuelle. Il est possible d'obtenir, par l'intermédiaire du CERA, une réduction SNCF de 20 % pour se rendre sur place.

Pour tous renseignements, écrivez au :

CERA
« La Dominique »
11170 Villespy
(68) 60 21 89

Dans l'Aveyron

SI vous préparez le Brevet d'Aptitude aux Fonctions d'Animateur (BAFA), vous choisirez peut-être de suivre le stage de l'association des Eclaireurs et Eclaireuses de France qui se déroulera cet été à Bécours (Aveyron).

Cinq secteurs sont prévus : informatique, astronomie, électronique, photographie, énergie solaire. Vous pourrez choisir

LA GAZETTE DE LIST

entre participer à une seule activité ou préparer une étude faisant appel à plusieurs secteurs — un programme d'astronomie par exemple.

Avec l'option informatique, on peut s'initier au Basic, au Logo, apprendre à utiliser des logiciels ou créer les siens sur la gamme Thomson. Ces derniers, tous conçus comme outils, seront tournés vers l'animation des jeunes.

Le prix de ce stage, agréé BAFA, est de 1 600 FF nourriture et hébergement compris.

L'association reçoit les inscriptions jusqu'au 30 juillet.

Les dates : du 24 août au 3 septembre 1985.

Vous pouvez vous renseigner auprès de
Daniel Champier
EEDF
Secteur activités scientifiques et techniques
66 rue de la Chaussée d'Antin
75009 Paris
(1) 874 51 40

En Ile-de-France

LE Centre X2000/Les Corolles de Courbevoie propose, durant le mois d'août prochain, des sessions de Basic et d'initiation à l'informatique. Chacune d'elles accueillera dix personnes pendant trois jours (de 9 h 30 à 12 h 30 et de 14 h à 17 h), pour une participation de 800 F. Des stages de Logo figurent également sur le calendrier (disponible sur simple appel téléphonique) du Centre X2000.

Deux sessions de Basic auront lieu les 7, 8, 9 août et 21, 22, 23 août. Les 12, 13 et 14 août seront réservés à l'initiation à l'informatique.

Centre X2000/Les Corolles
13 place des Corolles
La Défense 2
92400 Courbevoie
(1) 773 64 07

Initiation au langage Logo

COMME chaque année, l'association Grepacific (Groupe de Recherche et d'Etudes pour une Pratique Active et Coordonnée de l'Informatique en Formation Initiale et Continue) organise des stages d'initiation au Logo. Ceux-ci se déroulent sur trente heures réparties en cinq sessions de six heures chacune pendant lesquelles cinq thèmes seront abordés : nombres, texte, tortue, musique et lutins.

La participation s'élève à 1 200 FF pour ceux qui sont pris en charge par un organisme de formation, à 750 FF pour les autres (les adhérents de Grepacific bénéficient d'une réduction de 150 FF). Ces prix ne comprennent pas les frais d'hébergement.

Du 26 au 30 août, on a le choix entre trois lieux de « villégiature » : dans le Gard à Alès, dans le Finistère à Quimper ou en Lozère à Florac.

Pour s'inscrire ou obtenir des précisions, on peut contacter l'association.

Grepacific
51 boulevard des Batignolles
75008 Paris ■

Apple qui rit Apple qui pleure

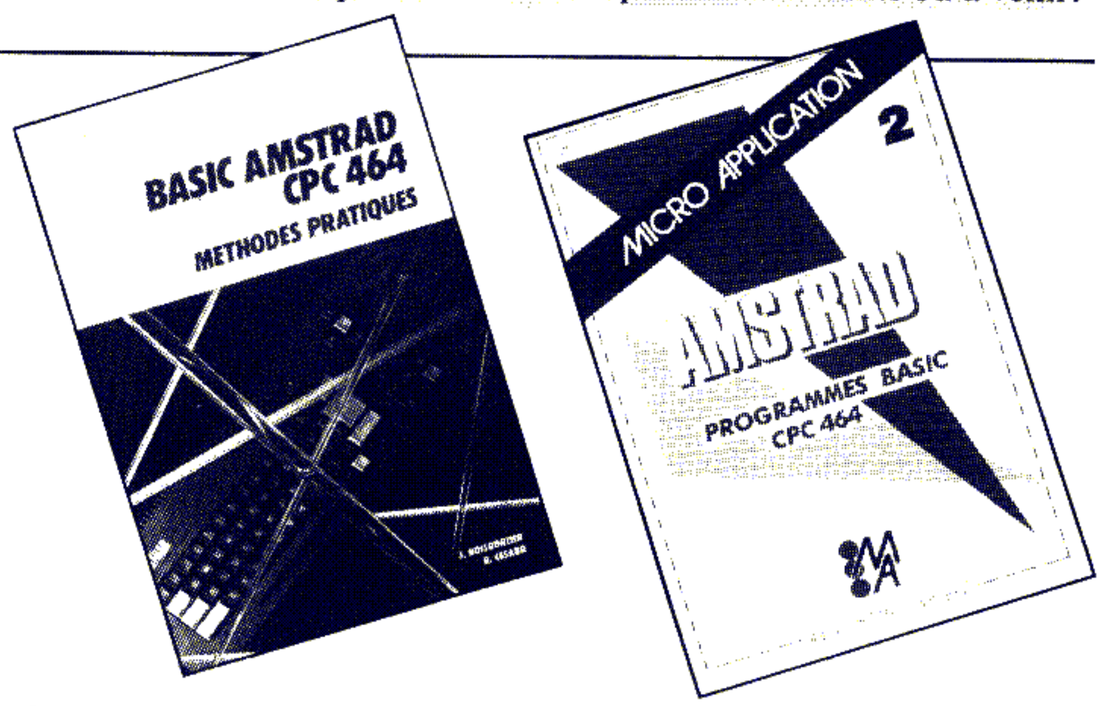
LES développeurs qui travaillent sur Macintosh n'auront plus de jus de fruits à l'œil, décision du Boss. Ce n'est pas tout, John Sculley a également décidé de supprimer les massages gratuits, une prestation dont ils bénéficiaient jusqu'alors. Les prévisions de vente sont certainement à l'origine de ces restrictions. Beaucoup plus grave, on parle très sérieusement à Cupertino de licencier près de 1 000 employés liés par des contrats à durée déterminée. Ces nouvelles n'ont pas empêché le patron d'Apple-France d'affirmer que « tout le monde est né avec le désir de posséder un jour un Apple ». Quelques milliards de clients potentiels... ■

Oric change de mains et de prix

LES stocks d'Oric Products International ont été rachetés par Eureka Informatique qui commercialise désormais l'Atmos à 990 FF. Avec un moniteur couleur et un magnétocassette, on pourra le trouver à 3 490 FF.

Outre les stocks, Eureka a acquis tous les droits et licences sous le nom d'Oric, et sur tous les produits existants ou à venir.

DEUX LIVRES



Basic Amstrad CPC 464
Méthodes pratiques
Jacques Boisgontier
et Bruno Césard
Editions du PSI
1985, 166 pages
Prix : 100 FF

AUX dires des auteurs, ces méthodes pratiques ont été écrites à l'intention de lecteurs qui connaissent déjà la micro-informatique. Si l'on se contente de feuilleter distraitement le livre, on peut avoir l'impression qu'ont été rassemblées, en treize chapitres, toutes les instructions de l'Amstrad. En fait, plutôt que de les présenter « froidement » l'une après l'autre, les auteurs les ont regroupées par thème d'utilisation : par exemple, les instructions concernant les « entrées au clavier », ou les fonctions qui se rapportent aux chaînes de caractères, etc., chacune d'elles étant accompagnée d'un ou plusieurs exemples représentatifs.

Le choix de ce classement aurait pu nuire à la recherche d'un mot particulier dans le vaste éventail offert par le « Basic Locomotive », mais les auteurs ont prévu le cas et l'on peut trouver, en annexe, une liste complète du vocabulaire de l'Amstrad avec, pour chaque expression, le numéro de page où elle apparaît.

Cet ouvrage complétera donc utilement la documentation fournie avec la machine, servant peut-être moins bien le débutant authentique que le programmeur plus averti qui désire s'initier sérieusement au Basic de l'Amstrad CPC 464.

RB ■

Amstrad CPC 464 (2)
Programmes Basic
Rainer Lueers
Edité par Micro Application
1985, 182 pages
Prix : 129 FF

CE deuxième tome d'un ouvrage venu d'outre-Rhin, et destiné aux amateurs d'Amstrad, est bien différent du tome 1. Ce dernier apportait, sous le titre « Trucs et Astuces », une véritable mine de trésors réservée aux programmeurs confirmés.

Le tome 2 est, à l'évidence, destiné en grande partie aux débutants. Il rassemble une collection de 23 programmes, tous écrits en Basic, dont le manque d'homogénéité laisse rêveur. On commence par une série de cinq listes traitant du stockage des données et programmes en mémoire (POKE, PEEK et Hexadécimal). Mais ce départ sur les chapeaux d'octets est suivi de propositions beaucoup moins pointues (éditeur de dessins, de musique ou de textes), accompagnées d'études légèrement simplistes (calendrier...) et de jeux (Mastermind, réflexes, etc.).

Enfin, un désassembleur, complété en annexe par un tableau des « tokens » du Basic, hisse à nouveau la barre des difficultés à un niveau bien supérieur à celui du débutant.

Peu de commentaires sur les applications des programmes, pratiquement pas d'explications sur leur fonctionnement si ce ne sont les lignes de REM qui jalonnent les listes.

Tel est donc le livre, très inégal, dont on ne sait s'il est destiné aux novices ou aux initiés.

JPL ■

CHEZ LE LIBRAIRE

Basic Plus
80 routines sur Commodore 64
Michel Martin
Éditions du PSI
1985, 130 pages
Prix : 85 FF

VOUS savez sur le bout des doigts le Basic de votre C.64 et vous pensez qu'il n'a plus de secrets pour vous. Avant de délaisser votre machine, attendez donc d'avoir lu ce livre. Il décrit certaines des fonctions les plus évoluées du C.64 et propose un ensemble de 80 sous-programmes en Basic qui en améliorent la puissance : "PRINT USING", "REPEAT... UNTIL", etc. (il ne s'agit pas d'extensions du langage mais seulement de routines en Basic). L'accent est mis sur le générateur sonore et le graphisme haute résolution. ■

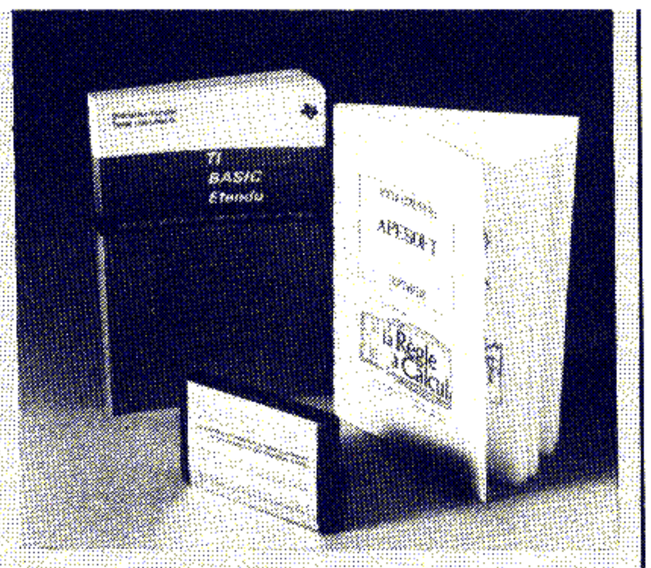
Dictionnaire d'informatique bureautique-télématique
Anglais-Français
Michel Ginguay
8e édition
Editions Masson
1985, 320 pages
Prix : 113 FF

IL donne la traduction ou l'équivalent en français des termes anglais liés à l'informatique : non seulement le vocabulaire technique, mais aussi les abréviations, néologismes et autres expressions (par exemple certaines appellations commerciales très connues) relevant du « jargon informatique ».

La Bible du programme de l'Amstrad CPC
Bruckmann, Lothar, English et Gerits
Edité par Micro Application
1985, environ 700 pages
Prix : 249 FF

UN pavé qui se présente comme référence pour les possesseurs d'Amstrad. Il se divise en trois parties : la première s'adresse à ceux qui connaissent l'électronique, la seconde est consacrée au système d'exploitation et la troisième au Basic. On trouve en annexe les routines de la mémoire et les « tokens » du Basic.

Haute résolution graphique pour le TI-99/4A



XBasic
Basic étendu
Cartouche pour TI-99/4A
Edité par Mechatronic
Distribué par la Règle à calcul
Prix : 1 200 FF

NOUVELLE version du Basic étendu 1.1 de Texas Instruments, la cartouche XBasic graphique d'Apesoft est construite par Mechatronic et distribuée par la Règle à Calcul. La cartouche contient 8 Ko de mémoire morte supplémentaires dans lesquels on trouve des fonctions complémentaires du Basic aux applications pratiques très utiles.

FIND localise une chaîne de caractères donnée dans une tableau de variables dimensionné par DIM. MOVE transfère un bloc de mémoire vers une adresse quelconque (sauvegarde instantanée de pages d'écran par exemple) avec une rapidité équivalente au langage-machine. BHCOPY réalise des copies d'écran graphique vers une imprimante, un lecteur de cassette ou de disquette. MLOAD et MSAVE chargent ou sauvent des documents au format « PROGRAM » : l'encombrement est moindre pour un transfert plus rapide.

Cette nouvelle version permet également de disposer d'un NEW et d'un BYE programmables, d'un RESTORE suivi d'un nom de variable et d'un WAIT (boucle d'attente).

Plus prosaïquement, certaines de ces extensions viennent combler les lacunes de la version précédente : QUIT ON/OFF permet d'éviter l'interruption d'un programme par le clavier en désactivant le « QUIT » (remise à zéro). SPR ON/OFF commande le départ ou l'arrêt instantané de tous les sprites au même moment.

La grande originalité du module réside cependant dans les 8 Ko de routines graphiques en assembleur que l'on peut charger dans l'extension mémoire par un CALL APESOFT. De nouvelles fonctions sont alors disponibles par CALL LINK (NOM, paramètres) ; on peut, par exemple, créer une ou plusieurs fenêtres graphiques (WINDOW) de 128 × 120 points avec une résolution de 256 × 192 points et disposer de huit couleurs pour le fond et huit couleurs pour le premier plan (il est impossible d'adresser directement les points de l'écran avec seize couleurs et une résolution maximale, car la mémoire vidéo — ne pas confondre avec la mémoire d'affichage — du TI-99/4A est limitée en octets). Dans chaque fenêtre, des commandes sont actives : droite, cercle, arc, ellipse, rectangle, copie fenêtre, déplace fenêtre. Certaines instructions proches du Logo sont également présentes : déplacement relatif, fonction TURN pour pivoter dans la direction demandée suivant l'angle choisi, CENTRE pour déplacer les coordonnées du repère des coordonnées graphiques.

Chaque ordre possède son double : effacer cercle, rectangle, etc., trouver coordonnées point, centre, angle, etc. Ajoutons des macro-commandes pour construire diagrammes et axes de coordonnées, et le tableau sera complet.

Le « Basic du diable » est livré avec une notice en mauvais anglais, traduite de l'allemand, et il vaut 1 200 FF. A ce prix-là, on aurait aimé un manuel en français, en particulier pour les routines graphiques.

MA ■

La Règle à Calcul
65/67 boulevard Saint-Germain
75005 Paris
(1) 325 68 88

UN LIVRE

Programmer en assembleur sur Commodore 64
Bruce Smith
Editions Cedic/Nathan
1985, 204 pages
Prix : 125 FF

TROUVER le juste équilibre est difficile en toute chose. Les ouvrages sur l'assembleur n'échappent pas à cette règle. Trop complexes, ils sont inacessibles aux débutants, trop dilués et c'est l'ennui garanti pour les connaisseurs. Le livre de Bruce Smith constitue-t-il un bon compromis ?

Les quelque 200 pages se découpent en un nombre impressionnant de chapitres (24), certains d'entre eux ne dépassant guère les deux pages. Si les premiers sont plutôt axés sur la théorie (représentations numériques, arithmétique binaire, opérations logiques), les choses prennent, avec les suivants, un tour résolument plus concret (descriptif des registres du 6510 et nombreux programmes).

Ecrits en Basic, les programmes sont construits sur un modèle assez ingénieux : chacun d'eux est proposé sous la forme d'un chargeur Basic et les lignes de DATA représentant les codes d'opération sont complétées par des valeurs hexadécimales et par les mnémoniques correspondants. Il y a par ailleurs, pour les paresseux, un court programme de chargeur hexadécimal Basic. Les possesseurs de logiciels d'as-

LA GAZETTE DE LIST

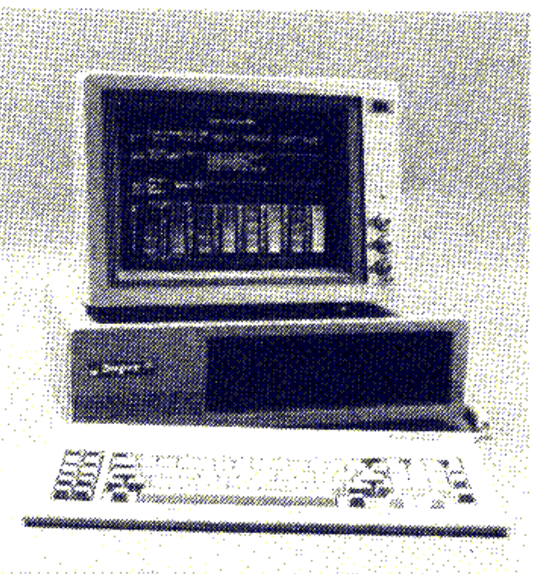
semblage devraient également y trouver leur compte.

Les études d'ordre général qui sont proposées sont accompagnées d'explications limpides. Quelques chapitres plus spécifiques au Commodore 64 parlent des lutins et du Kernal (dont toutes les routines sont décrites et analysées). On trouve enfin plusieurs annexes, la plus intéressante étant une liste commentée du jeu d'instructions du 6510.

La mise en page aérée permet une lecture agréable : du confort dans l'apprentissage en quelque sorte. Aussi, s'il ne satisfait pas pleinement aux exigences d'équilibre souhaitées plus haut, ce livre assurera au moins une tâche ingrate : celle de servir au mieux les débutants qui y trouveront matière à étancher leur soif de connaissance. JPL ■

Un compatible IBM PC à moins de 12 000 FF ttc

LA société **Vidéo Technologie** introduit sur le marché un ordinateur compatible IBM PC. Disponible dès maintenant, le Laser Super PC/XT est commercialisé à moins de 12 000 FF ttc. Ce prix comprend l'unité centrale (128 Ko de mémoire vive, clavier Azerty), un lecteur de disquettes 320 Ko avec sa carte contrôleur, une carte graphique couleur avec une sortie parallèle et deux sorties série, enfin, une alimentation de 135 W permettant de lui adjoindre plusieurs lecteurs et un disque dur. ■



Une nouvelle tête à la tête d'Apple-France

JEAN CALMON remplace Jean-Louis Gassée à la direction d'Apple-France, ce dernier ayant été nommé Vice-Président responsable des produits Apple à Cupertino (États-Unis). ■

CHEZ LE LIBRAIRE

Le guide Magnard du tout logiciel 1985/86
Editions Magnard
1985, 420 pages
Format 21 × 29,7
Prix : 149 FF

LES quelque 3 000 logiciels référencés dans ce guide sont répartis en quatre catégories (éducation, jeux, graphisme et musique, utilitaires et gestion domestique) avec, pour chacun des titres, un résumé du programme ainsi que les noms des éditeurs et distributeurs. Une série de symboles (type guide Michelin) légendés en début d'ouvrage fait office de descriptif technique. Outre un petit lexique des termes informatiques et les adresses de boutiques classées par département, on trouve quatre index croisés particulièrement utiles pour retrouver rapidement un logiciel dont on ne connaît pas toutes les références.

Introduction à l'intelligence artificielle sur micro-ordinateur
Mike James
Editions Eyrolles
1985, 160 pages
Prix : 95 FF

CETTE « introduction » fait le point sur les différentes facettes de l'intelligence artificielle : heuristique et algorithmique, structure de la mémoire, synthèse vocale, reconnaissance des formes, langage, etc. Chacun de ces aspects est étayé d'un programme Basic.

LOGICIELS

Assembleur
Microcartouche pour
Sinclair QL
Edité par Metacomco
Distribué par Sinclair/Direco
Prix : 690 FF

CE macro-assembleur conditionnel est doté d'un éditeur pleine page complet. Il possède les commandes habituelles : LIST, NOLIST, OBJECT, TITLE, PAGE, ... La place importante qu'il occupe en mémoire le destine plutôt aux versions 512 Ko du QL (avec une unité de disquettes). Son utilisation n'est pas aisée et il rebutera sans doute le débutant, mais il offre des possibilités quasi professionnelles.

Assembleur-Editeur
Debogueur
Cartouche pour Atari 800 XL
et 130 XE
Edité par Atari
Prix : 450 FF

L'éditeur est pleine page avec numérotation des lignes. Il offre les commandes classiques : recherche (FIND), remplacement d'une chaîne par une autre (REP), renumérotation (REN), etc. L'assembleur conditionnel (ce n'est pas un macro-assembleur) est accessible depuis l'éditeur en tapant ASM, de même le débogueur avec BUG.

Ce dernier contient un mini-assembleur (une seule instruction à la fois). Il comporte aussi le pas à pas et l'affichage du contenu des mémoires, la visualisation des registres...

Un outil sérieux pour la programmation en langage-machine et en Assembleur.

Profimat
Moniteur assembleur
Disquette pour C.64
Edité par Micro Application
Prix : 350 FF

LE logiciel comprend deux modules indépendants : un moniteur, Profi-mon 64 et un assembleur, Profi-ass 64. Il est compatible avec Pascal 64 (même éditeur). Comme c'est souvent le cas pour ce type de produit, la notice — quarante pages — ne s'adresse pas au débutant qui devra s'initier à l'Assembleur avant de pouvoir utiliser Profimat.

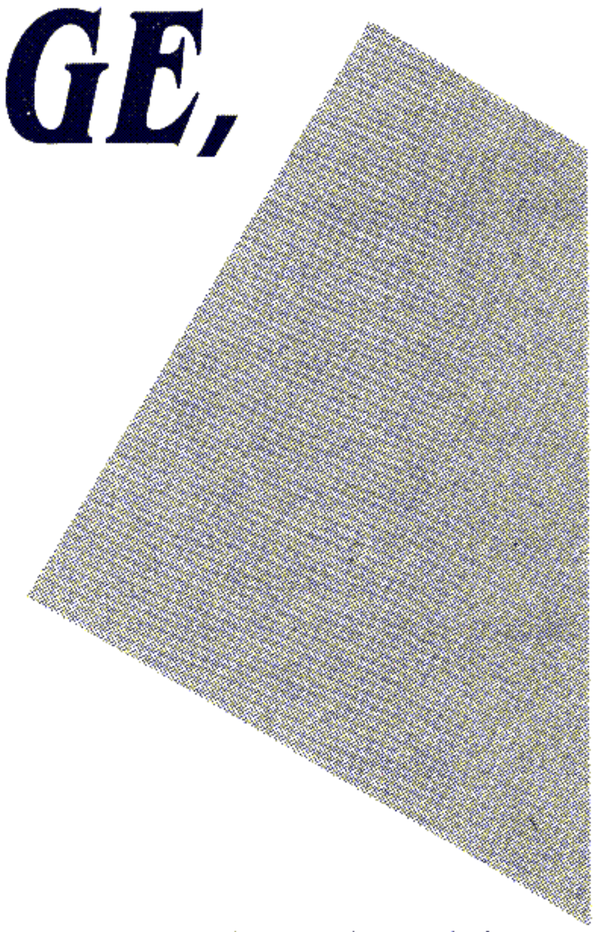
QL Toolkit
Ensemble de programmes
constituant une extension
au SuperBasic
Microcartouche pour
Sinclair QL
Edité par Sinclair
Distribué par Sinclair/Direco
Prix : 280 FF

EXTENSION au SuperBasic et gestion du système multitâche, Toolkit apporte plus de cinquante nouvelles procédures : accès direct aux fichiers, éditeur pleine page, spooling, utilitaires de copie et de sauvegarde, etc.

L'utilisateur a le contrôle des tâches qui tournent sur l'ordinateur ; il peut à tout moment décider de la priorité de l'une d'elles, de son existence même (seul le Basic est indestructible).

LES DRÔLES DE MACHINES DE CHARLES BABBAGE, PHILOSOPHE

« Non est facta pro his qui olera aut pisculos vendunt, sed pro observatoriis aut cameris computorum, aut aliis, qui sumptus facile ferunt et multo calculo egent »(*)
Leibniz



A un siècle de distance, deux mathématiciens-philosophes, Leibniz et Babbage, ont défié la technologie de leur temps. Ils ont conçu des appareils prodigieux d'intelligence : des machines à calculer. Celles de Babbage inaugurent l'ère de la programmation.

Schickard, Pascal et Leibniz furent les pionniers de la machine à calculer. Ils la définirent et la réalisèrent au moins partiellement.

Mais c'est Babbage qui marque le début d'une nouvelle étape vers l'ordinateur. Pour la première fois, ce sont ses *Difference Engines* (machines à différences) qui effectuent automatiquement une chaîne de calculs, et non plus un seul calcul. Plus importante encore, l'*Analytical Engine* inaugurera une ère et un art : ceux de la programmation...

La vie mouvementée et acide de Charles Babbage (26 décembre 1791 — 18 octobre 1871), partiellement décrite dans ses *Passages from the Life of a Philosopher*, est d'une richesse étonnante. Il fut mathématicien, homme public et économiste, de l'école de Ricardo. En tant que mathématicien, il traduisit le traité de Lacroix dès 1816 et fut ainsi le premier à introduire en Angleterre les notations de Leibniz — bien plus efficaces que celles de Newton — en calcul différentiel et intégral (dx/dy, par exemple).

Babbage influença même Karl Marx

Son grand ouvrage (*On the Economy of Machinery and Manufactures*, 1832) fut assez original pour influencer fortement Karl Marx. D'ailleurs, Herbert Spencer, Charles Dickens et lui-même surent en faire triompher l'une des thèses : l'abrogation du prix fixe pour les livres. Babbage fut encore expert en chemins de fer, mais aussi directement responsable de la promulgation d'un acte (25 juillet 1864) règlementant les nuisances des musiciens ambulants dont orgues et tambours hindous interrompaient si souvent son travail. Il fut l'un des meilleurs spécialistes du chiffre de

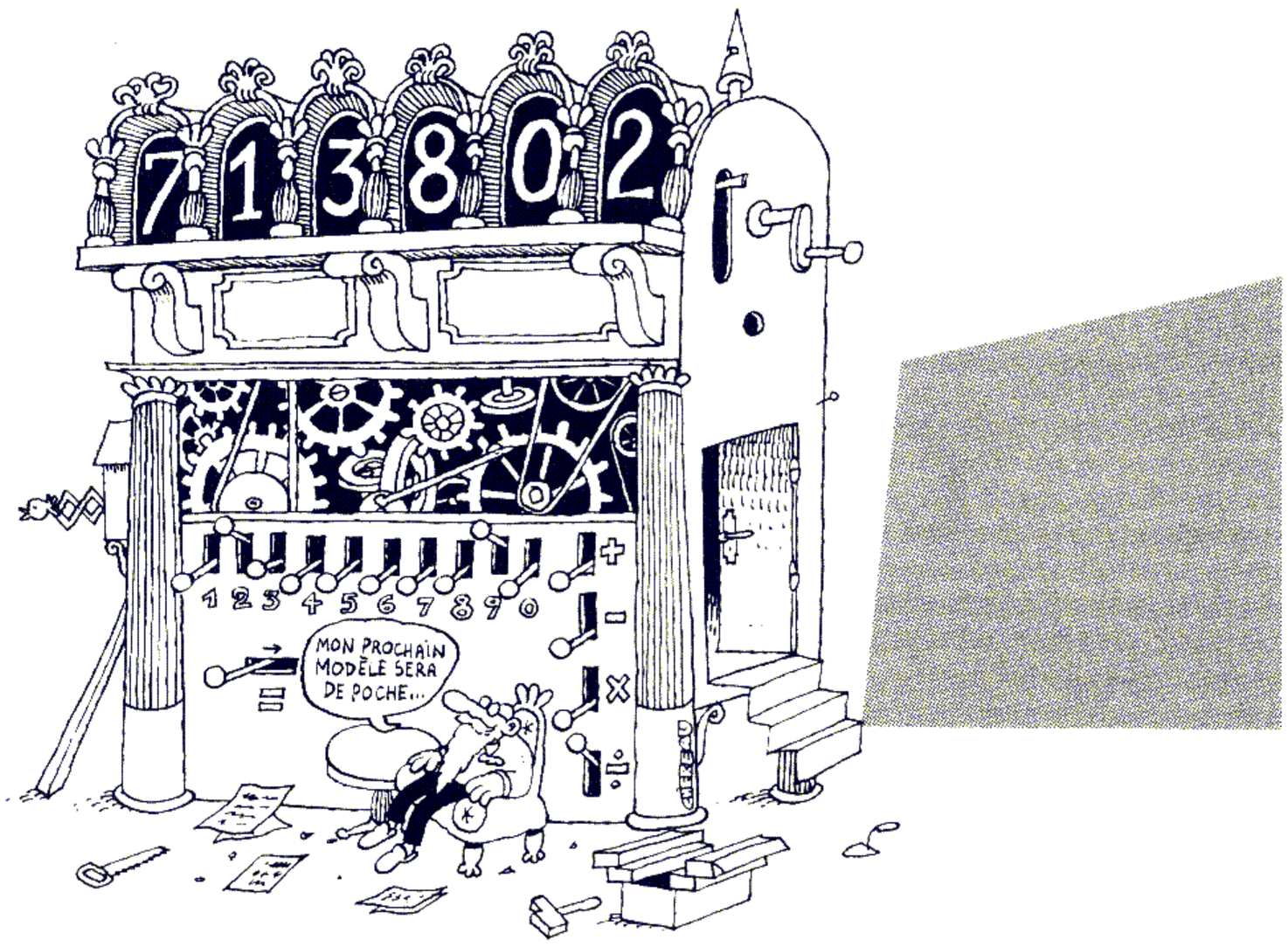
(*) « Elle n'est pas faite pour les épiciers ni les poissonniers, mais pour les observatoires, les bureaux privés, et pour qui peut en supporter les frais pour de gros calculs ». Ce jugement orgueilleux de Leibniz a été appliqué aux ruineuses machines de Charles Babbage par un de ses contemporains.

LES DRÔLES DE MACHINES DE CHARLES BABBAGE

son époque et donc doublement précurseur direct de l'anglais Alan Turing — constructeur d'ordinateurs et briseur des codes de guerre.

L'activité de Babbage ne s'arrête pas là. On le trouve pêle-mêle réformateur de la Royal Society, géologue, physicien expérimentateur de talent, assureur, théologien, inlassable quémandeur de subventions, solitaire misanthrope, libéral avant 1848 (conservateur ensuite), fils de banquier dînant avec des ducs, ami d'Arago et de Lucien Bonaparte, polémiste atrabilaire, singulier en tout, bref : un caractère.

Mais c'est évidemment le concepteur des *Engines* (et tout particulièrement de l'*Analytical*) qui mérite notre attention, sinon notre reconnaissance. Avant de lire ensemble quelques-uns des tout premiers programmes jamais écrits, il faut dire quelques mots de ces extraordinaires machines. Bien que l'on puisse distinguer plusieurs projets de *Difference Engine* — au moins trois —, il suffira ici de parler assez brièvement du premier qui l'occupera de 1812 à 1834.



Tout faire mécaniquement

Lui-même auteur d'un recueil de logarithmes des entiers jusqu'à 108 000 (en 1826), Babbage fut toujours intéressé par le travail numérique considérable des astronomes (il avait fondé l'Astronomical Society en 1820) et surtout des éditeurs de tables mathématiques. Il raconte dans ses mémoires qu'un de ses amis l'avait entendu, à Cambridge, remarquer que tout aurait dû se faire mécaniquement. Un tel but se retrouvera plusieurs fois dans l'histoire de l'ordinateur : le surnom du premier IBM, le Mark I dit « Bessie », rappelle par exemple qu'il servit à calculer des fonctions de Bessel ; d'autres prototypes contemporains auront pour but de fournir des éléments de tir plus précis à l'artillerie alors en guerre...

Les *Difference Engines* eurent donc, dès le départ, un objectif limité que Babbage, fort de son expérience, saura ensuite dépasser. Elles étaient basées sur deux idées assez simples. La première est un théorème de mathématiques qui ne sera démontré (par Weierstrass) qu'aux environs de 1886 : les fonctions continues (dont les logarithmes) peuvent être « bien » approchées par des suites de polynômes. Certes non encore démontrée explicitement à cette époque, cette propriété apparaît comme si intuitive qu'il aurait paru incongru alors de chercher à formaliser une telle « évidence ».

La seconde concernait justement les polynômes. Elle était, quant à elle, bien connue depuis longtemps. Considérons une fonction polynôme f(x) de degré n; les fonctions g, h... définies par les différences successives : $g(x) = f(x+1) - f(x)$, $h(x) = g(x+1) - g(x)$, etc. sont encore des polynômes, mais de degré de plus en plus petit. Ainsi, au bout de n opérations, on tombera nécessairement sur une fonction constante. Newton avait fait grand usage de ces « différences finies », aujourd'hui encore essentielles en analyse numérique.

Pour la vérifier, prenons justement l'exemple d'une table de logarithmes et considérons quelques valeurs consécutives comme 4771213, 4785665, 4800069, 4814426, 4828736, 4842998. Les différences séparant ces six nombres sont respectivement égales à 14452, 14404, 14357, 14310 et 14262. Les différences de ces différences sont, à leur tour, égales à −48, −47, −47 et −48. Nous voyons bien apparaître — avec une incertitude sur le dernier chiffre — une constante, dès le deuxième pas ! On comprend dès lors que, certains de cette invariance et connaissant seulement par exemple les deux nombres extrêmes de la liste, on puisse cependant retrouver pratiquement les quatre manquants avec très peu d'erreurs (surtout si, en fait, on n'a en vue qu'une table moins précise à cinq décimales).

Tel était le procédé par lequel Babbage, qui n'avait d'ailleurs pas été le premier à remarquer cette possibilité, désirait construire une machine automatique qui permettrait des calculs de tables. Le nom *Difference Engine* s'explique alors par la méthode suivie elle-même. Il y consacra énormément d'argent — sa famille était riche —, mais dut évidemment demander de l'aide car la réalisation technique exigeait un travail soigné de très grande précision, à la limite des possibilités de l'époque. Il avait déjà construit entre 1820 et 1822 un premier modèle réduit qui manipulait des nombres de six chiffres et allait jusqu'à deux différences (comme dans notre exemple). Il l'avait appliqué à la tabulation d'une fonction du second

ALGORITHMES DES DIFFÉRENCES

Et si on reprenait le chemin en sens inverse : aujourd'hui, il est très facile de transformer son ordinateur en *Difference Engine* (machine à différences), les calculs mathématiques se ramenant toujours aux quatre opérations arithmétiques de base, voire même à l'addition et à la soustraction.

Le calcul différentiel est à la base de l'analyse numérique, ce qui explique que les algorithmes faisant appel aux différences sont nombreux.

Prenons un exemple : les différences appliquées aux cubes (table 1). On observe que le cube d'un nombre est égal à la somme des différences qui le précèdent. Dans le tableau, on a bien :
$9^3 = 512 + 169 + 42 + 6$, soit 729.

D'autre part, à la troisième différence, D_3, on remarque, en utilisant les mêmes notations que la table, que le résultat est constant, égal à 6. Et donc D_4 est à zéro. Ce résultat est général car :

- $D_1 = n^3 - (n-1)^3$ (comme différence entre le cube d'un nombre n et celui du nombre qui précède, à savoir $n-1$) ; donc, en développant, on trouve : $D_1 = 3n(n-1) + 1$.
- D_2 est égal à la différence entre D_1 appliqué à n et D_1 appliqué à $n-1$. A savoir, $D_2 = 3n(n-1)+1-(3(n-1)(n-2)+1)$. Donc, $D_2 = 6(n-1)$.
- De même, $D_3 = 6(n-1)-6(n-2)$. On retrouve bien : $D_3 = 6$.

Table 1
Différences appliquées aux cubes

n	n^3	D_1	D_2	D_3	D_4
0	0				
		1			
1	1		6		
		7		6	
2	8		12		0
		19		6	
3	27		18		0
		37		6	
4	64		24		0
		61		6	
5	125		30		0
		91		6	
6	216		36		0
		127		6	
7	343		42		
		169			
8	512				
...					

Table 2
Différences appliquées aux puissances de 2

n	2^n	D_1	D_2	D_3	D_4	D_5	D_6	D_7	...
0	1								
		1							
1	2		1						
		2		1					
2	4		2		1				
		4		2		1			
3	8		4		2		1		
		8		4		2		1	
4	16		8		4		2		...
		16		8		4		2	
5	32		16		8		4		...
		32		16		8		4	
6	64		32		16		8		...
		64		32		16		8	
7	128		64		32		16		
		128		64		32			
8	256		128		64				
		256		128					
9	512		256						
		512							
10	1024								
...									

Examinons alors la table 2, celle des puissances successives de 2. Elle est bien monotone ! On peut toutefois en déduire un algorithme original de calcul de 2^x. Pour limiter le nombre de boucles du programme déduit (programme 1), on décale X de 8, car ce nombre augmente rapidement quand X devient petit. Le domaine de recherche est vaste, d'autres algorithmes sont à découvrir.

Claude NOWAKOWSKI

degré, $x^2 + x + 41$ (cette fonction admet des propriétés arithmétiques curieuses, en particulier face aux nombres premiers) : les différences secondes y sont toutes rigoureusement égales à 2.

Fort de sa réussite, il présenta une requête auprès du gouvernement, sans doute l'une des premières d'un genre bien développé depuis, pour obtenir de quoi fabriquer une machine traitant de nombres à 20 chiffres et 6 niveaux de différences, capable d'imprimer automatiquement les résultats obtenus. (Babbage savait bien qu'il y avait là une très classique cause d'erreurs ; dans un texte, il examine avec soin les « bogues » relevées dans les tables existantes, et classe avec rigueur différentes sortes de fautes de calcul, citant tout particulièrement celles dues à la recopie.)

Son entrevue de juin 1823 avec Robinson, Chancelier de l'Echiquier, eut deux conséquences capitales : la pre-

Programme 1
Calcul des valeurs successives de 2^x

```
20 INPUT X:E=X
30 IF X<8 THEN E=E+8
40 N=0:S=1:DP=1
50 N=N+1
60 DP=DP*(E-N+1)/N
70 S=S+DP
80 IF ABS(DP)<.000002 THEN 100
90 GOTO 50
100 IF X<8 THEN S=S/256
110 LPRINT "2 puissance ";X;" =
   ";S;"  ( ";N;" iterations )"
```

Programme 2
Calcul de e^x

```
10 DEFDBL A-Z
20 INPUT X:E=X/.69314718055994#
30 IF X<8 THEN E=E+8
40 N=0:S=1:DP=1
50 N=N+1
60 DP=DP*(E-N+1)/N
70 S=S+DP
80 IF ABS(DP)<2E-14 THEN 100
90 GOTO 50
100 IF X<8 THEN S=S/256
110 LPRINT "exp( ";X;" )= ";S;"
    ( ";N;" iterations )"
```

L'algorithme utilisé ici calcule directement e^x, en passant par : $e^x = 2^{n+d+z} = 2^n 2^d 2^z$, c'est-à-dire $x \log_2 e = n+d+z$, où n est la partie entière de $x \log_2 e$ et $\log_2 e = 1/\ln 2$.

LES DRÔLES DE MACHINES DE CHARLES BABBAGE

mière fut qu'une série de versements (culminant à 17 000 livres, soit plusieurs millions d'aujourd'hui) commença à l'aider pour la construction des rouages, axes, tringleries, etc. — Babbage lui-même ne demandait aucun paiement pour ses innombrables dessins et épures. La seconde fut moins agréable : aucun compte rendu écrit n'ayant été dressé de l'entrevue, chacun des participants quitta l'autre sur un malentendu. Babbage crut s'entendre promettre que le gouvernement, désireux de s'approprier la machine, par exemple pour les besoins de l'Amirauté, paierait jusqu'à réalisation complète. Inutile de dire que les hommes politiques successifs auxquels cette prétention fut soumise par la suite ne crurent pas devoir interpréter ainsi ce gentlemen's agreement...

Une partie de la *Difference Engine n° 1*, la seule à vrai dire qui vit le jour, sera construite en 1833 de façon assez accomplie pour qu'un long article puisse la décrire en juillet 1834 dans l'*Edinburgh Review* (où l'on peut lire la citation de Leibniz en exergue). Son influence sera très surprenante : deux imprimeurs suédois, George et Edward Scheutz, surent en tirer assez d'enseignements pour réaliser à leur tour en vingt années, non sans subventions de leur gouvernement d'ailleurs, des modèles simplifiés de l'œuvre incomplète du trop exigeant Babbage. Ce dernier, en dépit de son mauvais caractère, les félicita et les aida à obtenir une Médaille d'Or à l'Exposition de Paris de 1855. Des copies en furent réalisées, dont une devait même être utilisée par le gouvernement anglais ! On peut retrouver l'influence de ces machines, par exemple, dans un tabulateur d'IBM de 1930.

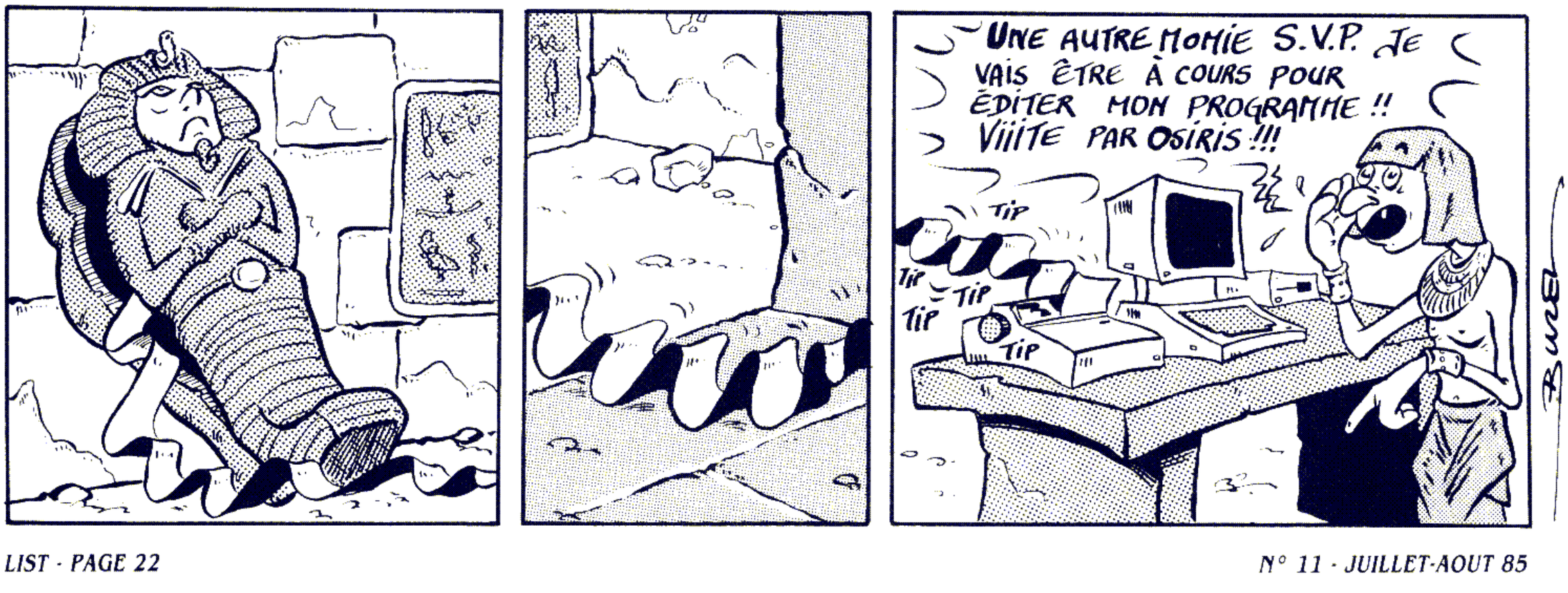
La partie construite jusqu'en juillet 1834 par Babbage lui-même, déposée en janvier 1843 au musée de King's College de Somerset House, fut exposée en 1862 à l'Exposition Universelle de Londres puis transportée définitivement au troisième étage du Science Museum de South Kensington où encore aujourd'hui on peut la voir, paraît-il toujours en état de marche. Ce musée contient également un fragment (posthume) d'une autre machine de Babbage, celle qui, bien davantage, peut être considérée comme l'ancêtre authentique de l'ordinateur : *l'Analytical Engine*. Sans doute peu de visiteurs, devant ces engrenages, ces roues et ces tiges, peuvent concevoir clairement l'immense fossé qui sépare les deux projets.

Des projets toujours plus ambitieux

Leur auteur, pour sa part, en était bien conscient. Tout en continuant, sa vie durant, à tracer des plans plus perfectionnés pour des modèles à différences finies, il avait commencé à réfléchir à des projets beaucoup plus ambitieux. Par exemple, pendant d'extravagants conflits où ses techniciens impayés lui volèrent toutes les épures déjà prêtes (!). Il déclarera d'ailleurs avec franchise, aux fonctionnaires qu'il ne cessait de harceler, qu'il serait finalement bien plus rentable pour tous d'abandonner les travaux en l'état pour commencer la nouvelle machine, au lieu de terminer l'ancienne. On devine l'effet de telles déclarations sur des responsables des deniers publics... décidés alors à ne plus lui accorder aucun crédit. Il était pourtant en train de concevoir, cette fois-ci, une œuvre véritablement géniale.

La nature des *Difference Engines*, comme on peut le comprendre d'après ces fragments de leur histoire, était relativement simpliste : leur programmabilité était réelle, dans la mesure où la mise au point d'une table mathématique était ramenée à une suite de calculs en chaîne automatiques. Mais la souplesse de la machine était très limitée par son projet même : elle n'était pas plus un ordinateur que ne l'est aujourd'hui une calculatrice donnant une racine carrée, ou des fonctions trigonométriques, par des procédés mis une fois pour toutes à l'intérieur. L'astucieuse application des différences finies des polynômes était capable de répondre efficacement à un besoin très réel, mais ne pouvait pas sortir de ce champ trop étroit. Combinant mentalement les subtils assemblages de ses roues dentées, Babbage avait fini par comprendre, vers 1834, qu'elles avaient en elles une force bien plus importante, qui gouverne encore aujourd'hui nos machines. En la décrivant une prochaine fois, nous rencontrerons la première « reine du soft » : Ada Lovelace soi-même, qui nous entraînera, cette fois-ci, à la découverte d'une véritable programmation.

André WARUSFEL



BASIC QL

ON PEUT TOUJOURS BRISER UNE LIGNE

LES instructions propres au Basic structuré facilitent notablement la programmation. En voici un exemple qui touche à la récursivité et à l'art de couper les segments de droite en quatre.

Pour l'amateur de graphisme, la courbe de Von Koch est un vrai régal ou une torture de chaque instant. Si vous n'êtes pas convaincu, acceptez d'y consacrer quelques minutes (*).

Selon l'expression communément employée, cette courbe très particulière est un « exemple-type de problème récursif ». Même si les maths ne sont pas votre point fort, ne vous laissez pas effaroucher par les mots. Il est relativement simple de comprendre ce dont il s'agit. A cette fin, je me propose, non pas de vous décrire le programme final, mais de vous montrer comment je l'ai construit, étape par étape.

Mi-Logo mi-Pascal

Pour commencer, voyons la définition de cette célèbre courbe, et considérons d'abord la transformation qui, à tout segment de droite I, associe quatre sous-segments I_1, I_2, I_3, I_4, chacun de ces sous-segments mesurant le tiers du segment initial (voir figure 1).

Le principe de la courbe de Von Koch est de réitérer cette transformation un nombre infini de fois. A la figure 2, où la transformation est notée "T", on voit le résultat obtenu après deux transformations. Bien entendu, il est hors de mon propos de dépasser cinq ou six itérations : au-delà, le temps d'exécution deviendrait prohibitif et, la résolution graphique étant ce qu'elle est, le tracé ne changerait pratiquement plus.

Le plus simple, pour effectuer une première approche, est sans doute de partir de la forme de base en la traduisant dans un langage approprié. J'ai choisi une manière de Logo français (figure 3).

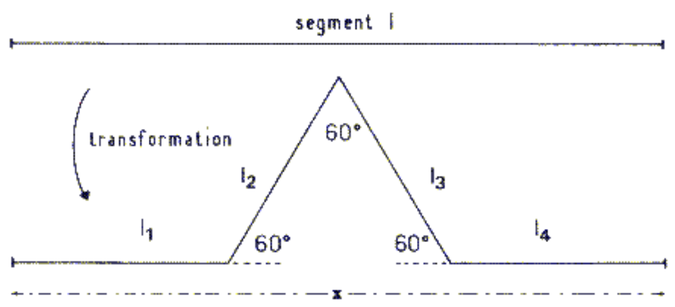
Si nous faisons subir à la courbe obtenue une deuxième transformation, nous obtenons une nouvelle courbe et une nouvelle procédure nommée *Von Koch (2)* et reproduite à la figure 4.

Nous voyons apparaître une constante et, avec la figure 5 (page suivante), notre intuition nous guide vers une fonction récursive. Ça y est, le mot est lâché. Découvrons donc la récursivité en exécutant cette procédure *à la main*, avec N = 1 et N = 2.

Pour y parvenir, il suffit de se conformer aux instructions des figures 3 et 4. Si l'on se demande ce que représente

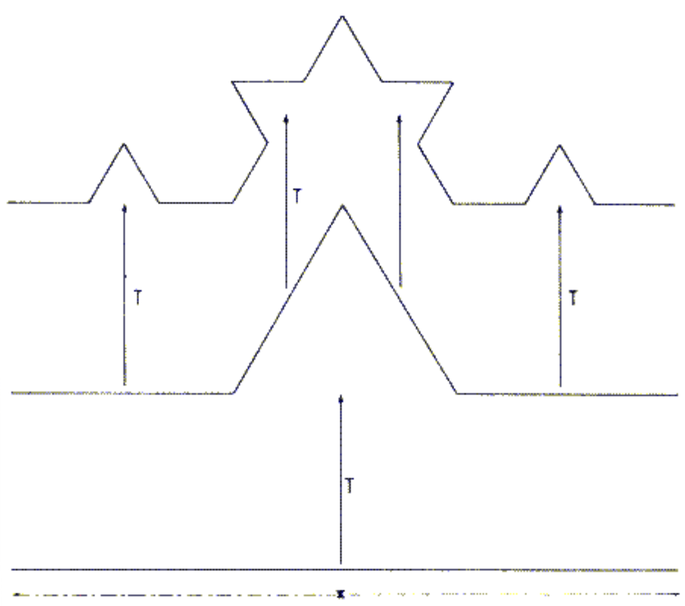
(*) Pour une autre approche du même problème, on pourra lire le Guide pratique du QL *(p. 129 à 138) d'Éric Tenin et Jean-Manuel Van Thong, édité par Edimicro.*

Figure 1



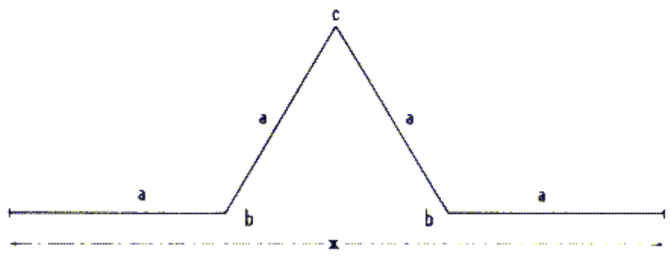
Sous le segment de départ, I, le motif obtenu après une première transformation.

Figure 2



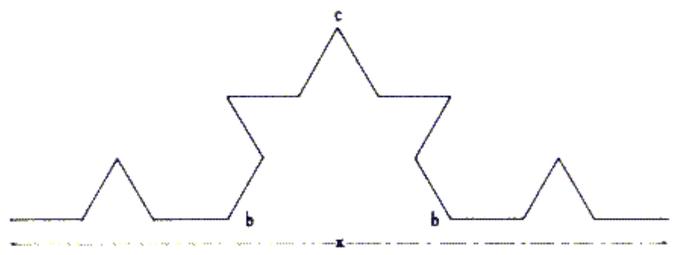
Après deux transformations.

Figure 3



Procédure Von Koch (1)
(a) avance x/3
(b) gauche 60
(a) avance x/3
(c) droite 120
(a) avance x/3
(b) gauche 60
(a) avance x/3
Fin de procédure Von Koch (1)
Dans l'énoncé de la procédure, x représente la longueur du segment initial.

Figure 4



Procédure Von Koch (2)
(a) Von Koch (1)
(b) gauche 60
(a) Von Koch (1)
(c) droite 120
(a) Von Koch (1)
(b) gauche 60
(a) Von Koch (1)
Fin de procédure Von Koch (2)

Figure 5

Procure Von Koch (N)
Von Koch (N-1)
gauche 60
Von Koch (N-1)
droite 120
Von Koch (N-1)
gauche 60
Von Koch (N-1)
Fin de procédure Von Koch (N)
  On pose Von Koch (0) = avance ($x/3^N$)

la procédure *Von Koch (1)* citée dans *Von Koch (2)*, on est tenté de répondre que c'est tout simplement la procédure de la figure 3. Attention toutefois, si c'est très ressemblant, il y a un détail qui diffère, et ce détail est important : $x/3^1$ est remplacé par $x/3^2$.

La dernière étape va consister à traduire notre « Logo français » en « Logo QL », c'est-à-dire à remplacer : *avance K* par *move K, droite L* par *turn L* et *gauche M* par *turn M.*

Et voilà, le tour est joué ; notre programme est pratiquement écrit. Je vous en propose d'ailleurs deux versions différentes : la première est strictement conforme à ce qui a été expliqué précédemment, et la deuxième comporte une itération sur un triangle équilatéral. Comme vous le voyez sur ces listes, si le Basic structuré du QL dessine un peu à la manière du Logo, il vous a aussi certains petits airs de Pascal...

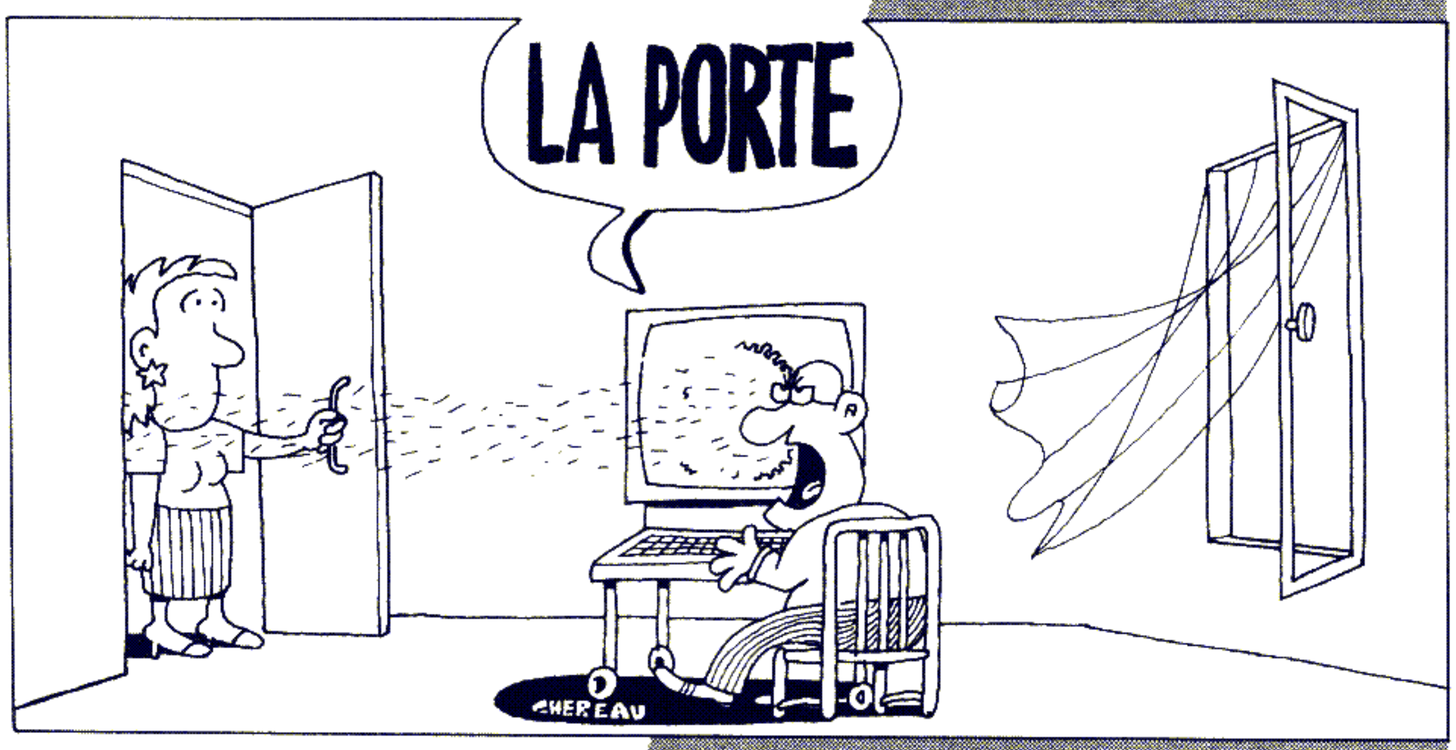
Courbe de Von Koch
Programme pour QL
Auteur Jean Ortega

Liste n° 1 : le départ est un segment de droite

```
10 INK 7 : PAPER 0
20 MODE 512
30 SCALE 65,0,0
40 CLS#1
50 REPEAT BOUCLE
60   PENDOWN
70   REPEAT ALORS
80     CLS#0
90     PRINT#0,"ENTRER LA PROFONDEUR 1 <> 9 ?";
100    A$ = INKEY$(-1) : PRINT#0,A$
110    IF CODE(A$) < 58 AND CODE (A$) > 48 THEN X = A$ : EXIT ALORS
120  END REPEAT ALORS
130  PRINT#0, "VOULEZ-VOUS EFFACER L'ECRAN O/N F)IN ?";
140  A$ = "": A$ = INKEY$(-1) : PRINT#0,A$
150  IF A$ = "F" OR A$ = "f" THEN CLS#0 : CLS#1 : STOP
160  IF A$ = "O" OR A$ = "o" THEN CLS#1
170  VON_KOCH(X)
180  REMARK ON REPOSITIONNE LA TORTUE
190  PENUP : MOVE -100
200 END REPEAT BOUCLE
```

Liste n° 2 : on part d'un triangle équilatéral

```
10 INK 7 : PAPER 0
20 MODE 512
30 SCALE 120,0,0
40 PENUP : TURN 90 : MOVE 50 : TURN -90 : MOVE 25 : TURN 45
50 CLS#1
60 REPEAT BOUCLE
70   PENDOWN
80   REPEAT ALORS
90     CLS#0
100    PRINT#0,"ENTRER LA PROFONDEUR 1 <> 9 ?";
110    A$ = INKEY$(-1) : PRINT#0,A$
120    IF CODE(A$) < 58 AND CODE (A$) > 48 THEN X = A$ : EXIT ALORS
130  END REPEAT ALORS
140  PRINT#0, "VOULEZ-VOUS EFFACER L'ECRAN O/N F)IN ?";
150  A$ = "": A$ = INKEY$(-1) : PRINT#0,A$
160  IF A$ = "F" OR A$ = "f" THEN CLS#0 : CLS#1 : PENUP : TURN -45 :
     MOVE -25 : TURN 90 : MOVE -50 : TURN -90 : STOP
170  IF A$ = "O" OR A$ = "o" THEN CLS#1
180  VON_KOCH(X)
190  TURN -120 : VON_KOCH(X)
200  TURN -120 : VON_KOCH(X)
210  REMARK ON POSITIONNE LA TORTUE
220  TURN -120
230 END REPEAT BOUCLE
```



Partie commune aux deux versions

```
260 DEFINE PROCEDURE VON_KOCH(N)
270   TRY_AGAIN
280   TURN 60
290   TRY_AGAIN
300   TURN -120
310   TRY_AGAIN
320   TURN 60
330   TRY_AGAIN
340 END DEFINE VON_KOCH

360 DEFINE PROCEDURE TRY_AGAIN
370 IF N-1 = 0 THEN
380   MOVE 100/(3 ^ X)
390 ELSE
400   VON_KOCH(N-1)
410 END IF
420 END DEFINE TRY_AGAIN
```

Jean ORTEGA

LE PIED A L'ÉTRIER

SUR TO7 et TO7/70, la programmation ne se limite pas au seul Basic. Elle est aussi possible en Assembleur. Encore faut-il connaître ce langage peu évolué, auquel on a accès grâce à certaines instructions du Basic justement. Une première approche est donc nécessaire, mettre le pied à l'étrier en quelque sorte.

Les instructions de l'Assembleur du 6809, processeur du TO7, sont au nombre de 70. Le *programme-source* (*) devra être écrit à partir de ces instructions. Pour accéder alors au *code-objet*, seul compréhensible par la machine, on peut utiliser soit un tableau de correspondance *mnémoniques*/codes binaires ou hexadécimaux, soit un programme abusivement appelé assembleur, qui fera cette conversion (ou assemblage).

Octet par octet c'est le jeu

Un programme écrit en *Assembleur* puis assemblé en mémoire centrale peut être exécuté à partir du Basic, ou à partir du *moniteur* de l'assembleur, par la commande GO.

Sur le TO7 muni du seul Basic, il faut commencer par implanter le code-objet en mémoire. Chaque adresse de la mémoire centrale ne peut contenir qu'une seule valeur de 0 à 255 (0 à FF, en hexadécimal). Le code-objet doit alors se présenter sous la forme d'une suite de nombres. Le « jeu » consiste à les implanter à partir d'une adresse donnée, octet par octet. La routine Basic d'implantation du code-objet se présentera donc toujours comme celle qui est présentée ci-dessous.

Attention, sans aucune précaution, le Basic pourra déborder sur le code-objet. Il faut donc auparavant réserver une zone mémoire à ce code-objet. Cette réservation se fait à l'aide de l'instruction CLEAR,ADR – 1 (la virgule est nécessaire car il s'agit du deuxième argument de CLEAR). Ceci signifie que toutes les adresses supérieures à ADR – 1 sont réservées.

L'exécution de ce programme Basic (par RUN) ne donne aucun résultat apparent, mais elle installe le code-objet en mémoire. Il sera alors exécuté à son tour par EXEC ADR. Si le code-objet se termine par &H39 (RTS pour Return To System, en source), à la fin de l'exécution, le microprocesseur rendra la main au Basic et vous pourrez alors en reprendre le contrôle.

Parmi les neuf *modes d'adressage* dis-

Routine Basic d'implantation du code-objet

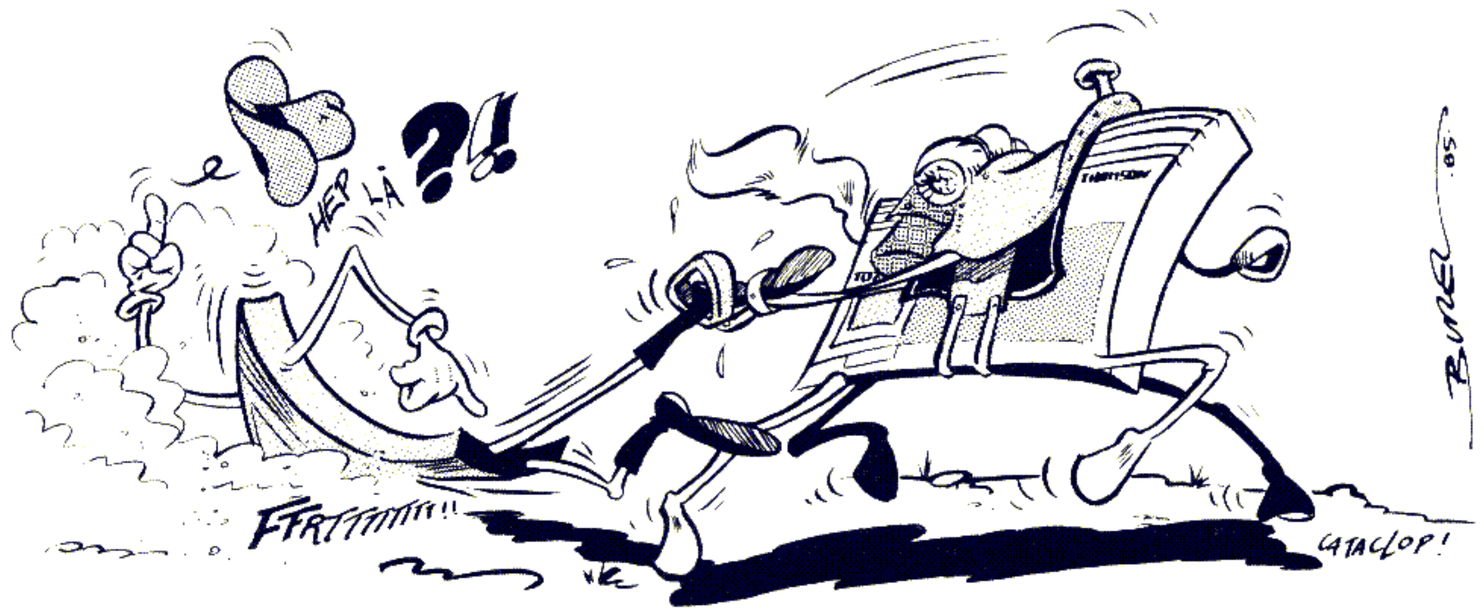
```
5 CLEAR,ADR-1
10 FOR I=0 TO N-1          'N nombre de codes à implanter
20 READ C$                 'Lecture du code dans les tables de DATA
30 C$="&H"+C$              'Le préfixe &H est ajouté pour préciser qu'il s'agit de
                            valeurs hexadécimales
40 POKE&HADR+I,VAL(C$)     'Implantation du code à l'adresse adéquate
50 NEXT I                  'Passage à la valeur suivante
100 DATA ,,,,,,,,,         'N valeurs hexadécimales entre les virgules correspondent
                            au code-source
```

(*) Les termes en italique sont définis dans l'encadré « Petit lexique de l'Assembleur », page suivante.

ponibles, le plus simple est l'adressage immédiat. Il charge une valeur donnée dans un *accumulateur*. Deux instructions de l'Assembleur du 6809 vont nous permettre d'illustrer ceci à travers un exemple : LDB et JSR.

L'instruction LDB est l'abréviation de LoaD B, charger dans l'accumulateur B. Ainsi, LDB #$40 signifie en Assembleur : charger en mode immédiat (#), dans B, la valeur hexadécimale $40. En Assembleur, on distingue les valeurs hexadécimales par le préfixe $, à ne pas confondre avec les $ des variables chaînes en Basic. Quant à JSR, c'est l'abréviation de Jump to SubRoutine, saut à un sous-programme. Ainsi, JSR $ADR2 ira au sous-programme implanté à l'adresse ADR2 (en hexa).

Nous voulons maintenant écrire le mot LIST sur l'écran. Certains esprits chagrins diront qu'il suffit de faire PRINT "LIST". C'est vrai, mais l'Assembleur ne peut s'apprendre qu'avec des exemples simples, voire simplistes. Un des sous-programmes du moniteur est chargé uniquement d'afficher les caractères du clavier sur l'écran. Car ce que vous voyez sur votre écran est en fait le résultat d'une « cuisine » faite par ce moniteur. Ce sous-programme — on dit encore routine — qui affiche un caractère à l'écran s'appelle PUTC$. Il est implanté à l'adresse $E803. Avant d'appeler cette routine, il faut charger le code *ASCII* du caractère à écrire, dans l'accumulateur B.



Code-source qui permettra d'écrire "LIST" à l'écran

```
LDB #$4C    'on charge B avec le code de L
JSR $E803   'on saute à la routine d'affichage
LDB #$49    'on charge B avec le code de I
JSR $E803   'on saute à la routine d'affichage
LDB #$53    'on charge B avec le code de S
JSR $E803   'on saute à la routine d'affichage
LDB #$54    'on charge B avec le code de T
JSR $E803   'on saute à la routine d'affichage
RTS         'retour au Basic
```

Tellement plus intelligent

Dans la table ASCII, les codes sont : 76 (4C en hexa) pour le L, 73 (49 en hexa) pour le I, 83 (53 en hexa) pour le S, 84 (54 en hexa) pour le T. Le code-source qui nous permettra d'arriver au résultat est représenté ci-contre. Evidemment, c'est plus compliqué qu'en Basic, mais tellement plus intelligent !

Maintenant, deux cas peuvent se présenter : ou bien vous disposez d'un assembleur qui vous transformera ce code-source en code-objet, et qui l'implantera à l'adresse que vous choisirez ; ou bien vous allez assembler à la main, et pour cela, il vous faut une table des codes des instructions Assembleur.

Dans cette table, on voit que le code de LDB en adressage immédiat est C6 en hexa, et que celui de JSR est BD. Le code-objet s'écrit donc : C6 4C BD E8 03 C6 49 BD E8 03 C6 53 BD E8 03 C6 54 BD E8 03 39.

Il ne reste plus qu'à introduire ces codes dans la ligne de DATA de la routine Basic (ligne 100), en séparant bien chaque code par une virgule. On choisira, par exemple, 42000 pour ADR, ce qui donne CLEAR, 41999 à la ligne 5 et POKE&HA410+I,VAL (C$) à la ligne 40. Le programme-objet étant composé de 21 codes, à la ligne 10, I va de 0 à 20. Enfin, après avoir fait tourner cette routine (par RUN), on lance l'exécution par EXEC 42000. Et le tour est joué.

Cette petite promenade dans l'Assembleur n'a pas la prétention de faire de vous des champions, mais peut-être pourra-t-elle donner envie à certains de soulever le voile d'un langage plus obscur que le Basic. Et si, comme moi, vous avez le vertige, laissez-vous tomber, l'Assembleur est absolument inoffensif.

Jean-Paul CARRÉ

Bibliographie sommaire

- La programmation du 6809, Rodnay Zaks - Editions Sybex.
- Manuel de l'Assembleur 6809 du TO7/TO7-70, Michel Weissgerber - Editions Cedic-Nathan.
- Faites vos jeux en Assembleur sur TO7/TO7-70, Michel Oury - Editions Cedic-Nathan.
- Programmation en Assembleur 6809, Bui Minh Duc - Editions Eyrolles.
- Le microprocesseur 6809, C. Dardanne et J. Boulesteix - Editions Eyrolles.
- Assembleur et périphériques des MO5 et TO7/70, Frédéric Blanc et François Normand - Editions PSI.

Petit lexique de l'Assembleur

Accumulateur : registre de travail du microprocesseur. Le 6809 en possède deux, les accumulateurs A et B, de 8 bits chacun, qui ne peuvent donc contenir que des valeurs inférieures à 255 (FF en hexadécimal). Ces deux accumulateurs peuvent être « concaténés », ou encore juxtaposés, pour former un accumulateur à 16 bits.

ASCII : American Standard Code for Information Interchange, c'est-à-dire code standard américain pour l'échange d'informations, l'Esperanto de l'informatique en d'autres termes.

Assembleur : langage informatique composé de *mnémoniques* traduisant des instructions en langage-machine.

Code-objet ou programme-objet : programme écrit en langage-machine.

Code-source ou programme-source : programme écrit en Assembleur.

Mnémoniques : termes symboliques assimilables et compréhensibles par l'homme.

Modes d'adressage : ils sont au nombre de 9, chacun correspond à la façon dont on charge telle ou telle valeur à une adresse ou dans un registre.

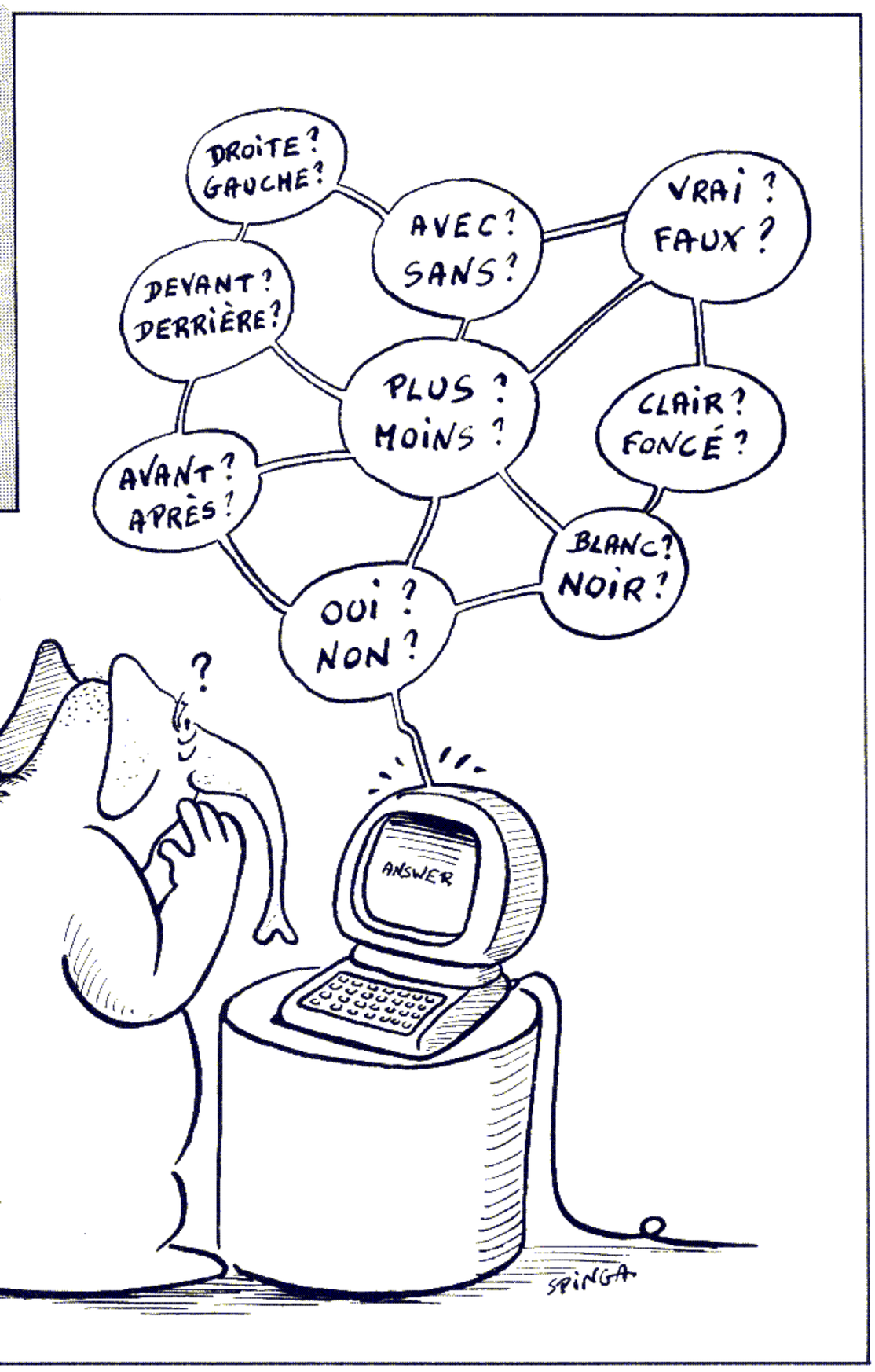
Moniteur : programme-objet implanté en mémoire morte dans l'ordinateur. Il contient des routines indispensables à l'utilisateur de l'ordinateur, comme la routine d'affichage des caractères à l'écran, par exemple.

Programme-objet : voir code-objet.

Programme-source : voir code-source.

L'ORDRE INPUT ET LES RÉPONSES DE NORMAND

SOUVENT, les programmes demandent une réponse ou une confirmation qui se traduit par un oui ou par un non. Mais en réalité, le test n'est effectué que sur l'une des deux réponses. Est-ce bien correct vis-à-vis de l'utilisateur ? Et est-ce seulement habile ?



La logique de l'informatique est-elle binaire ? Au vu de certains programmes, on pourrait parfois se poser la question. Voici un exemple choisi (au hasard) dans un article de *LIST* :

```
630 INPUT "ON RECUPERE";R$
640 IF R$ = "N" THEN 990
650 IF R$ < > "O" THEN 630
```

De toute évidence, le programmeur suspecte l'utilisateur de son logiciel de commettre parfois des erreurs, ou du moins de ne pas toujours répondre par O ou par N qui sont les réponses attendues à la question posée. Sans cette suspicion, le test de la ligne 650 n'a pas de raison d'être. Ecartons l'hypothèse de l'utilisateur facétieux qui s'ingénie à ne jamais donner l'une des réponses admises, et retenons la banale erreur, celle qui est commise de bonne foi. Notre uti-

L'ORDRE INPUT ET LES RÉPONSES DE NORMAND

lisateur se trompe donc de temps à autre. Bien. Mais alors demandons-nous carrément : quand se trompe-t-il, et quand ne se trompe-t-il pas ?

Le programmeur a décidé d'attendre un O ou un N et de considérer toute autre réponse comme fausse, comme une erreur de l'utilisateur. Si cette erreur survient, on lui pose de nouveau la question de la ligne 630 (qui, soit dit en passant, ne précise pas que c'est O ou N qui sont attendus : à l'utilisateur de le deviner par tâtonnements...). En revanche, une réponse O ou N est acceptée sans la moindre hésitation : pour O ou N, l'utilisateur ne peut pas s'être trompé. Bizarrement, dans ce cas-là, il est réputé infaillible.

Être logique avec soi-même

Reprenons le raisonnement. En se contentant de vérifier que la réponse est bien un O ou un N, on suppose que l'utilisateur peut taper par erreur toutes les touches qui ne sont ni celle du O ni celle du N, mais il y a une erreur dont on a décidé qu'elle ne pouvait pas se produire, c'est celle qui consiste à taper un O à la place d'un N ou vice-versa. Or il faut être logique avec soi-même : si l'on doit suspecter l'utilisateur d'un programme de se tromper de touche dans ses réponses, autant le suspecter totalement.

L'un de mes amis souffre d'un fâcheux travers : il ne fait confiance à personne. Constant dans sa manie, il prévoit dans ses programmes que, même si l'on répond O, c'est peut-être aussi par erreur. Je vous laisse imaginer le plaisir que c'est d'utiliser ses logiciels.

Et pourtant, ne serait-il pas juste, dans la mesure du possible, de faire confiance totalement à celui que l'on interroge ? Nous allons envisager ce point de vue en abordant le problème d'une tout autre façon. Dans l'exemple que nous avons retenu, posons-nous la question du but recherché.

Que veut-on savoir à vrai dire ? On veut savoir si le programme doit exécuter une certaine tâche (ligne 660 et suivantes) ou ne pas l'exécuter (ligne 990). Nous avons donc affaire à un choix binaire. Et malgré cela, le programmeur a admis, de facto, trois réponses possibles : *Oui, Non* et *ni-Oui-ni-Non*. La troisième réponse, quand elle est rencontrée, renvoie à la question. On pourrait en déduire qu'en réalité le programme ne sait pas ce qu'il doit faire en cas de réponse différente de O et de N. Il y a d'ailleurs une bonne raison à cela : c'est que le programme ne fait quelque chose que pour O. Un véritable choix binaire serait donc « ou c'est *Oui* ou c'est tout ce qui n'est pas *Oui* », autrement dit « ou c'est O ou c'est tout ce qui n'est pas O ».

Il paraît raisonnable de poser qu'il n'y a lieu de s'inquiéter que de ce qu'il y a à faire. En fin de compte, ce qu'on ne doit pas faire n'a aucune importance. Ce n'est pas le rôle des programmes que de passer au crible toutes les réponses possibles au risque de lasser ceux qui les utilisent. Pour faire un parallèle avec la vie quotidienne, je dirais que, lorsque vous croisez quelqu'un sur votre chemin, il ne vous est en rien utile de lui demander de confirmer que vous ne lui avez pas demandé l'heure.

Le programmeur croit peut-être qu'il est efficace en contrôlant tout, mais sa logique n'est pas celle de l'utilisateur. En se méprenant de la sorte, il risque en fait de prendre ce dernier pour un imbécile.

Il faut alléger les programmes de toutes ces vérifications tatillonnes, inutiles la plupart du temps, et dévalorisantes pour l'utilisateur. Certains programmes demandent :

```
630 INPUT "On récupère O/N";R$
640 IF R$ < > "O" THEN 990
650 REM On récupère...
```

Ce mensonge (il faut bien appeler les choses par leur nom) se rencontre souvent : on affirme attendre aussi N, alors que toute lettre différente de O aura le même résultat que N. Mentir à l'utilisateur d'un programme, c'est d'une part prendre le risque de le perturber, mais aussi et surtout celui de le lasser. En ce qui me concerne, je suis partisan de la formule :

```
630 INPUT "On récupère 'O'ui";REC$
640 IF REC$ = "O" THEN GOSUB
    "module-récupère"
650...
```

ou encore

```
630 INPUT "On récupère 'O'ui";
    REC$ : RECUPE =
    (REC$ = "O")
```

et plus loin : `IF RECUPE THEN GOSUB "module-récupère"`

La seule bonne raison d'imposer une contrainte à l'utilisateur, c'est d'éviter soit un danger pour le déroulement du programme, soit une perte importante de travail. Mais ceci est un problème de programmation : l'informatique est un outil au service de celui qui l'utilise. C'est-à-dire que celui-ci doit être libre de faire ce qu'il veut. Si le programme doit être intelligent, c'est ici qu'il doit le montrer. Il faut qu'il soit conçu pour prendre des précautions (par exemple empêcher les divisions dont le quotient serait nul, ou empêcher la perte involontaire d'un fichier).

Apparemment, beaucoup de programmeurs n'imaginent pas que l'homme puisse, à l'inverse de l'ordinateur, se tromper ou (chose très différente) changer d'avis. De ce point de vue, combien de programmes fonctionnent comme un piège : une fois qu'une fonction a été demandée, alors elle se referme et il faut la réaliser jusqu'au bout !

Abomination : je connais un programme de comptabilité dans lequel on demande la sauvegarde d'une écriture lorsqu'elle est entièrement saisie ; en cas de réponse négative, l'écriture est tout simplement perdue ! Est-ce obligatoire ? Le programme ne peut-il garder provisoirement l'écriture ?

La micro-informatique est essentiellement interactive. C'est-à-dire que chaque commande est réalisée au fur et à mesure de la volonté de celui qui la demande. On travaille par actions successives, une fonction peut être réalisée au moment où l'on veut et toute erreur peut être rattrapée après coup ; le programme doit le permettre.

A mon sens, n'importe quelle fonction peut être examinée selon deux critères : toute question doit entraîner la possibilité de se dédire, et toute réponse erronée doit annuler une fonction plutôt que de conduire à la répétition de la question.

Prenons le cas très classique de la création d'un fichier :

```
600 PRINT "création d'un fichier :"
605 INPUT "nom du fichier";FICH$
```

```
610 IF LEN(FICH$) < 1 THEN 605
615 IF LEN(FICH$) > 10 THEN
    PRINT "trop long...": GOTO 605
620 INPUT "Nombre de rubri
    ques";R$
625 IF VAL(R$) < 0 OR VAL(R$)
    > 10 THEN 620
630 FOR R = 1 TO VAL(R$)
635 PRINT "nom de la rubri
    que";R;"  ";
640 INPUT RUB$(R)
645 IF LEN(RUB$(R)) < 1 THEN 635
650 NEXT R
655 REM suite...
```

L'utilisateur est piégé non seulement sur le nom du fichier, mais aussi sur le nombre des rubriques et sur chaque nom de rubrique. A chaque fois, il faut absolument qu'il donne une bonne réponse. S'il ne le fait pas, on lui reposera la question inlassablement... Et ceci va se répéter d'une façon infernale jusqu'à la dernière rubrique. Heureusement que l'ordinateur n'est pas encore pourvu de règle pour frapper le bout des doigts du vilain utilisateur.

Il est pourtant si facile d'exprimer un refus : si l'on ne veut rien, alors on ne répond rien, et REPONSE$ = "". S'agit-il maintenant de traiter une erreur ? On indiquera clairement dans le programme ce qui fera l'objet du contrôle : excès de caractères, par exemple. Et si l'erreur survient malgré tout, on fera en sorte que le programme abandonne la fonction par RETURN. Programmons par modules.

A la question : IF CREE THEN GOSUB 600, le programme exécute :

```
600 PRINT "C)réation d'un fichier :"
605 INPUT "nom du fichier (max. 10
    cars)";FICH$
610 IF FICH$ = "" OR LEN(FICH$)
    > 10 THEN RETURN
615 INPUT "nombre de rubri
    ques";NRUB$
620 NRUB = VAL(NRUB$)
625 IF NRUB < 1 OR NRUB > 10
    THEN FICH$ = "" : RETURN
630 FOR RUB = 1 TO NRUB
635 PRINT "nom de la rubrique";
    RUB;"  ";
640 INPUT RUB$(RUB)
645 IF RUB$(RUB) = "" THEN
    FICH$ = "" : RETURN
650 NEXT RUB
655 REM suite...
695 RETURN
```

L'utilisateur de notre programme peut ainsi changer d'avis. S'il s'est trompé, il redemandera à nouveau la fonction, et puis c'est tout. Notons que si FICH$ = "", c'est que le fichier n'est pas encore créé.

Le programme devient intelligent

Remarquons aussi l'utilisation de la variable alphanumérique NRUB$: la variable numérique doit être définitivement bannie dans l'instruction INPUT (la fonction VAL n'est pas si difficile à comprendre pour un débutant). Cela évite de détruire les beaux écrans préparés d'avance à cause d'un REDO FROM START ou d'un autre message très désagréable. De plus VAL (REPONSE$) < 1 détecte une réponse non numérique, puisqu'alors la valeur numérique est nulle.

On peut aussi transformer son programme en véritable « bête intelligente », en proposant les réponses entrées précédemment dans la fonction abandonnée. Il faut alors s'adapter aux possibilités de son ordinateur.

Par exemple, en Basic Microsoft (Thomson), on peut « remonter sur une ligne », on affiche l'ancienne valeur et on remonte saisir la nouvelle par-dessus :

```
90 RMT$ = CHR$(11) : REM carac-
   tere special:remonte
100 PRINT "<C>re ... <Q>ui
    choix:";
110 CHOI$ = INKEY$ : IF CHOI$ =
    "" THEN 110
120 IF CHOI$ = "C" THEN GOSUB
    600
130 IF CHOIX < > "Q" THEN 100
199 END
(...)
600 PRINT "Creation de fichier :"
605 INPUT "son nom (max. 10
    cars)";FICH$
610 IF FICH$ = "" OR LEN(FICH$)
    > 10 THEN FICH$ = "" :
    RETURN
615 PRINT TAB(29) ; NRUB$ ; RMT$
    : REM ancien
620 INPUT "nombre de rubriques
    (max. 10)"; RUB$
625 IF RUB$ > "" THEN NRUB$
    = RUB$ : REM nouveau
630 NRUB = VAL(NRUB$)
635 IF NRUB < 1 OR NRUB > 10
    THEN FICH$ = "":RETURN
640 FOR RUB = 1 TO NRUB
645 PRINT TAB(11) ; RUB$(RUB) ;
    RMT$ : REM ancien
650 INPUT "Rubrique";RUB$
655 IF RUB$ > "" THEN RUB$
    (RUB) = RUB$ : REM nouveau
660 NEXT RUB
680 REM suite...
695 RETURN
(...)
```

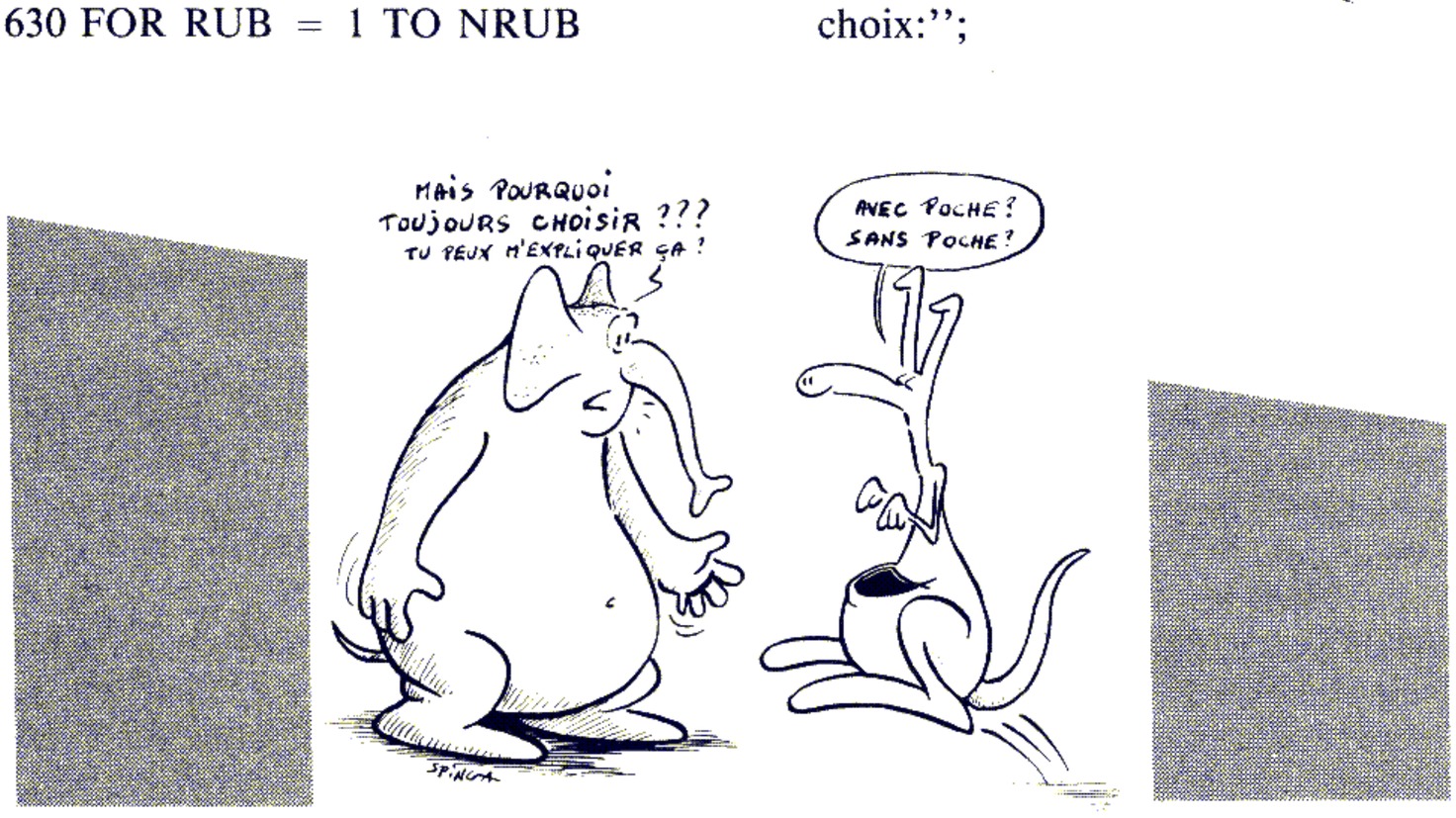
Pour reprendre en partie les caractères qui se trouvent sur la ligne, on peut utiliser la flèche gauche. RETURN permet de se déplacer rapidement et de passer directement à la rubrique qui doit être saisie à nouveau.

Ainsi l'information déjà entrée n'est pas perdue. Et si l'utilisateur n'est pas satisfait, il n'est pas remboursé, mais il peut reprendre la fonction "création de fichier" et ne changer que ce qui doit être changé. On peut d'ailleurs prévoir aussi une fonction d'effacement des valeurs devenues inutiles.

Le programme le plus diffusé en micro-informatique est l'interpréteur de langage Basic : il est vendu avec presque chaque appareil. C'est celui qui probablement a fait l'objet de la plus grande quantité de soins (voir article de *LIST* n° 9, page 24). Il serait bon que les concepteurs de programmes prennent modèle sur ce logiciel interactif : le programmeur peut commettre des erreurs et changer d'avis comme il le veut.

L'avenir appartient aux programmes souples et intelligents, à ceux qui se comportent en outils et qui déchargent l'homme de travaux répétitifs et laborieux. Ils ne se substituent donc pas à lui en lui dictant ce qu'il doit faire ou en le rappelant à l'ordre.

Max HAGENBURGER



LE BASIC DE L'EINSTEIN

COMME son nom ne l'indique pas, l'Einstein est particulièrement faible en maths. Son Basic offre même peu de possibilités dans ce domaine. Mais il se rattrape sur le graphisme, et il est très ouvert sur l'extérieur. Le lecteur de disquettes intégré explique en partie le prix relativement élevé de ce matériel.

Énorme est le terme qui vient immédiatement à l'esprit pour qualifier cet ordinateur baptisé Einstein. Énorme, mais pas encombrant puisque le plateau peut accueillir un téléviseur, même large. Résultat : seul le clavier occupe de la place. L'Einstein est ouvert sur le monde extérieur puisqu'il dispose d'une sortie série, d'une sortie parallèle, d'un convertisseur analogique-numérique, de l'indispensable sortie Péritel et d'un lecteur de disquettes intégré.

Son éditeur est de type plein écran, mais il n'est accessible que par deux touches de déplacement de curseur, utilisées directement ou avec la touche SHIFT. De même, insertion et destruction de caractères se font à partir de la même touche, avec ou sans SHIFT. Cette dernière est donc vouée à être fréquemment utilisée. Heureusement, le toucher du clavier est agréable.

Quant au Basic, il présente une forte prédominance pour le graphisme avec une définition de 256 sur 192 points en 16 couleurs, et surtout, la présence de 32 plans de lutins. L'affichage d'un plan à une position donnée et dans une couleur désirée se fait par SPRITE. L'affichage d'un lutin correspond en fait à l'affichage d'un caractère (généralement graphique). L'instruction qui permet d'en définir un nouveau s'appelle SHAPE. Un lutin est, en principe, défini dans une matrice de 8 points sur 8. Mais il est possible de le « grossir » en 16 points sur 16, par l'instruction MAG.

La haute résolution dispose d'un éventail d'instructions très complet : PLOT ou UNPLOT allument ou éteignent un point, POINT teste son état, ELLIPSE trace un cercle ou une ellipse, DRAW une ligne ou une série de lignes. Ainsi, DRAW 10,100 TO 100,100 TO 100,10 TO 10,100 tracera un rectangle. Quant à FILL, il remplit une zone graphique avec la couleur désirée.

Trois canaux au service du son

Même les polygones sont disponibles dans le Basic de l'Einstein : l'instruction POLY permet de les tracer en pointillés ou en trait continu, avec le nombre de côtés voulus. La définition des couleurs passe par BCOL pour le fond seul, par GCOL suivie de deux arguments pour le fond et les graphiques, par TCOL pour le fond et les caractères.

Ce Basic, bien équipé en possibilités graphiques, dispose aussi de fonctions sonores. Outre le simple BEEP, seulement paramétrable en durée, il possède des instructions qui agissent sur un générateur sonore capable d'émettre sur trois canaux. La fréquence, l'amplitude, l'enveloppe des sons peuvent être choisies. L'instruction MUSIC, suivie de trois chaînes de caractères, envoie des séries de notes sur chacun des trois canaux. Il est possible d'agir sur la durée de chaque note, ainsi que sur le tempo. VOICE assigne une voie sur les huit disponibles, pour l'instruction MUSIC. Ces voies étant indépendantes, il est possible de jouer plusieurs airs simultanément.

Dans le domaine mathématique, ce Basic est nettement moins complet. Les variables numériques ont été négligées : le type réel est le seul disponible avec six chiffres significatifs à l'affichage et sept pour les calculs. C'est vraiment très peu. En trigonométrie, on trouve l'inverse de la tangente, ATN, et la possibilité de faire les calculs en degrés ou en radians puisque la conversion est possible (par DEG et RAD).

L'instruction DEFFN est là, pour créer ses propres fonctions. Quant à EVAL, elle évalue la valeur numérique d'une chaîne de caractères. Elle est plus puissante que la classique fonction VAL, jugez plutôt :

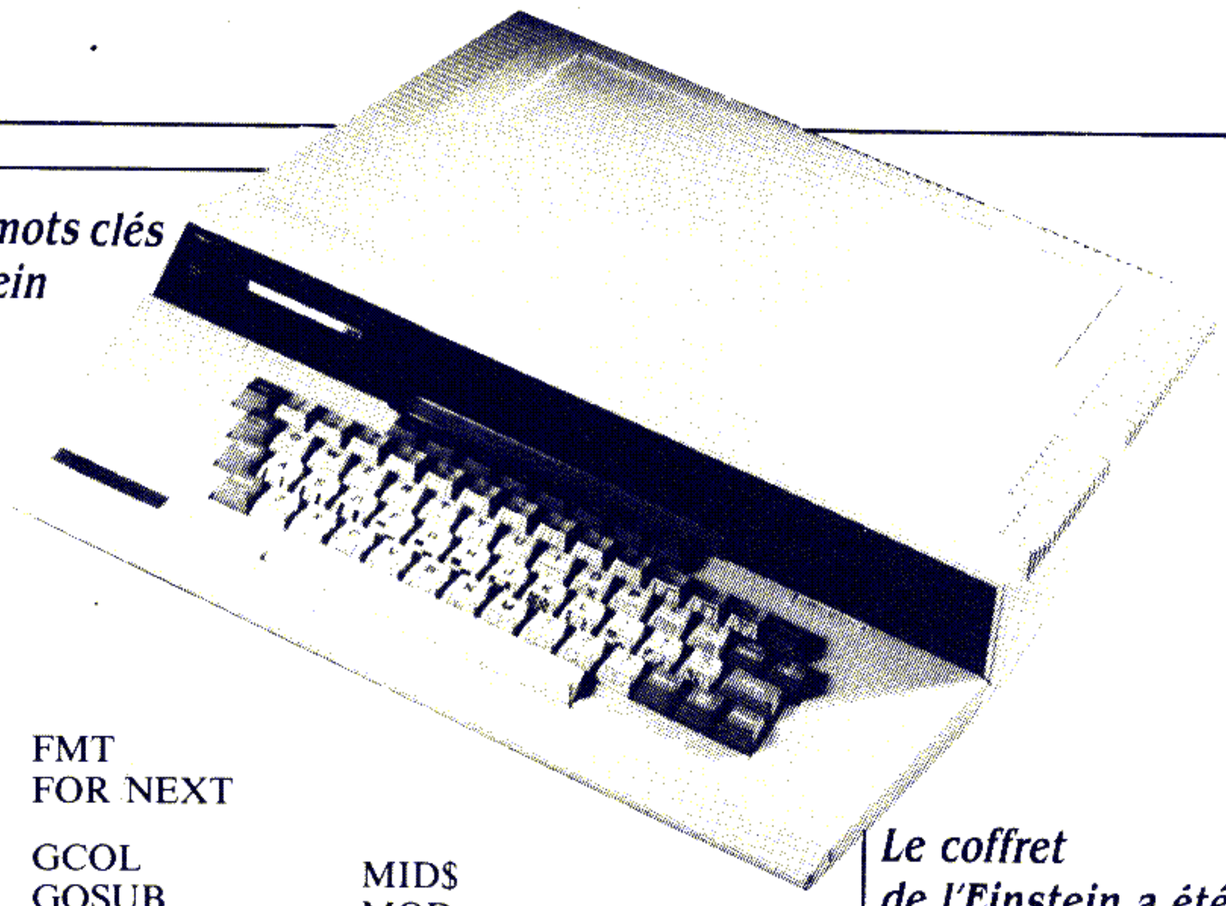
```
10X = 10:Y = 20
20A$ = "X + 2*Y"
30PRINT EVAL (A$)
```

Ce petit programme affichera le nombre 50. Entre autres applications, EVAL simplifie l'entrée des équations de fonctions dans des programmes. L'expression de la fonction étant directement introduite dans une variable alphanumérique (par un INPUT, par exemple), il est inutile de sortir du programme pour aller placer cette expression dans une ligne Basic.

Au chapitre des fonctions logiques, on trouve AND, OR, NOT, XOR, c'est-à-dire "et", "ou", "négation", "ou exclusif". Les changements de base sont assurés par BIN$ (binaire) et HEX$ (hexadécimal).

Le traitement des chaînes de caractères est assez classique, mis à part MUL$ (A$,n) qui duplique n fois la chaîne A$ (plus puissant que STRING$ (A$,n) du Basic Microsoft).

L'échange du contenu de deux variables est ici possible grâce à l'instruction SWAP (particulièrement utile dans les programmes de tri). La longueur des



Le coffret de l'Einstein a été conçu suffisamment profond pour servir de socle à un téléviseur

Liste des mots clés de l'Einstein

ABS
ADC
AND
APPEND
ASC
ATN
AUTO

BCOL
BEEP
BIN$
BTN

CALL
CHAIN
CHR$
CLEAR
CLOSE
CLS
CONT
COS
CREATE

DATA
DEEK
DEFFN
DEG
DEL
DIM
DIR
DOKE
DOS
DRAW
DRIVE

ELLIPSE
END
EOF
ERA
ERL
ERR$
EVAL
EXP

FILL
FMT
FOR NEXT

GCOL
GOSUB
GOTO

HEX$
HOLD

IF THEN ELSE
IF AND THEN
INCH
INCH$
INP
INPUT
INPUT #
INT
IOM

KBD
KBD$
KEY

LEFT$
LEN
LET
LIST
LISTP
LN
LOAD
LOCK
LOG

MAG
MGE
MID$
MOD
MOS
MUL$
MUSIC

NEW
NOT
NULL

OFF
ON ERR
ON GOSUB
ON GOTO
OPE
OR
ORIGIN
OUT

PEEK
PI
PLOT
POINT
POKE
POLY
POP
POS
PRINT
PRINT @
PRINT #
PSG
PSW
PTR

RAD
READ
REM
REN
RENUM
RESTORE
RETURN
RIGHT$
RND
RUN

SAVE
SCRN$
SEP
SGN
SHAPE
SIN
SIZE
SPC
SPEED
SPRITE
SQR
STEP
STOP
STR$
SWAP

TAB
TAN
TCOL
TEMPO

UNLOCK
UNPLOT

VAL
VDEEK
VDOKE
VERIFY
VOICE
VPEEK
VPOKE

WAIT
WIDTH

XOR

ZONE

Fiche technique de l'Einstein

Constructeur : *Tatung*
Distributeur : *Goal Computer*
Prix public : *7990 FF*
Processeur : *Z 80 A*
Mémoire vive : *64 Ko + 16 Ko de mémoire vidéo ; 43324 octets de mémoire utilisateur annoncés à la mise sous tension*
Mémoire morte : *8 Ko*
Langage : *Basic sur disquette, langage-machine*
Affichage : *256 x 192 points en 16 couleurs, 24 lignes de 32 ou 40 caractères*
Variables : *6 chiffres significatifs*
Variables alphanumériques : *longueur maximum 255 caractères*
Nombre de mots clés du Basic : *142*

noms de variables n'est limitée que par la longueur maximum d'une ligne, mais seuls les cinq premiers caractères seront significatifs.

L'affichage de texte s'effectue en 24 lignes de 40 ou 32 caractères, c'est faible. Mais il était difficile de faire mieux sur un ordinateur principalement destiné à être connecté à un téléviseur. Cet affichage dispose de certaines facilités originales : FMT prépare le format d'affichage avec un arrondi sur la dernière décimale.

Ainsi, la ligne :
10 FMT 2,3:PRINT 11.6666
donnera 11.667. SPC (X) affiche X espaces ; WIDTH permet de déterminer la largeur de l'affichage et, plus généralement, le nombre de caractères nécessaires pour provoquer l'envoi d'un retour chariot, et cela, quel que soit le périphérique ; SCRN$ (X), enfin, retourne le contenu de la Xième ligne de l'écran.

Si les premiers Basic ne pouvaient consulter le clavier que par le fameux INPUT, celui de l'Einstein dispose d'une panoplie d'instructions nettement plus évoluée : INCH attend la frappe d'une touche et retourne le code ASCII du caractère correspondant, INCH$ est semblable au précédent, mais retourne le caractère frappé ; KBD et KBD$ font de même, sans suspendre l'exécution. En fait, KBD$ est l'équivalent de INKEY$. INCH$(1) saisit une chaîne de caractères de longueur 1. KEY permet d'assigner une séquence de frappe (y compris une pression sur la touche ENTER) à une touche de fonction.

La présence d'instructions de numérotation et de renumérotation facilite la mise au point de programmes. Plus originale, HOLD extrait une partie du programme en mémoire, sans pour autant faire disparaître le reste. Alors, la commande de renumérotation (RENUM) peut renuméroter le bloc extrait, et uniquement lui. Cela permet, par la commande CHAIN, de charger un programme utilisant les mêmes numéros de ligne que le programme déjà en mémoire, puis de procéder à un RENUM de manière à faire coexister ces deux programmes. MGE annule l'effet de HOLD. Enfin, il est possible d'intercepter les erreurs, de connaître les messages qui leur sont associés (ERR$), le code (ERR) et même les lignes dans lesquelles elles se sont produites (ERL).

Accès au langage-machine

Le langage-machine est accessible grâce à PEEK et POKE, mais aussi DEEK et DOKE (écriture et lecture sur deux octets). Il existe les équivalents pour la mémoire vidéo (VPEEK, VPOKE, VDEEK et VDOKE). L'instruction POKE peut écrire une succession d'octets : une syntaxe telle que POKE 18000,10,20,50 est autorisée et placera les valeurs 10,20 et 50 dans les octets 18000, 18001 et 18002.

Pas très rapide, le Basic de l'Einstein est puissant dans le domaine du graphisme, mais pauvre dans celui des calculs. Il est important de noter enfin que ce Basic est sur disquette et que, par conséquent, il peut évoluer à l'avenir.

Thierry LÉVY-ABÉGNOLI

AFFICHER LE TITRE

Un programme bien présenté n'est pas forcément un bon programme, mais un bon programme est toujours bien présenté. Une procédure Pascal devrait permettre de donner une bonne allure à vos programmes, sans en augmenter la complexité.

Il arrive souvent que les programmes pèchent par la qualité de présentation de leurs écrans. De nos jours, il existe trois principales techniques d'affichage : l'écran continu, la gestion par page, la gestion d'écrans entièrement graphiques.

Seules les deux premières techniques peuvent à l'heure actuelle être utilisées sur tous les ordinateurs. La dernière, que l'on rencontre sur Macintosh, sort un peu de notre propos. En effet, les systèmes d'exploitation permettant une telle gestion des écrans sont encore peu répandus en micro-informatique, et pratiquement inexistants en mini ou en grande informatique.

La première technique est dite de l'*écran continu*. Dans ce cas, l'écran n'est jamais effacé, et le dialogue entre l'ordinateur et son utilisateur s'effectue ligne à ligne. Chaque nouvelle ligne est affichée en bas de l'écran, faisant remonter toutes les autres. Un tel affichage est aussi appelé « déroulant » (*scrolling*, en anglais).

Les bons vieux télétypes

Cette technique est très ancienne. Elle provient de l'époque où l'on travaillait avec des télétypes. Il ne s'agissait pas de consoles de visualisation électroniques, comme maintenant, mais de terminaux électro-mécaniques. Un télétype (TTY, en abrégé) est constitué d'un clavier à touches cylindriques, associé à une imprimante très lente (10 caractères par seconde), utilisant un cylindre sur lequel sont gravés les caractères.

Cette technique de gestion des écrans est toujours utilisée, car elle est extrêmement simple du point de vue de la programmation : les lignes sont affichées les unes après les autres, le système de gestion d'écran se chargeant de faire remonter l'ensemble de l'écran d'une ligne à chaque fois.

La seconde technique d'affichage des écrans est la gestion dite *par page*. Le dialogue entre l'ordinateur et l'utilisateur est structuré en fonctions. Ce système est analogue à un livre où l'on changerait de page à chaque début de chapitre. Ainsi, à chaque entrée dans

Affichage d'un titre
Procédure Pascal
Auteur Thierry Chamoret
Copyright *LIST* et l'auteur

```
procedure titre (le__titre : string) ;
const     largec    = 80 ;
var       decalage : integer ;
          i, espace : integer ;
          largeur   : integer ;
          l1, l2    : string [80] ;
          procedure chercher__l1__et__l2 (le__titre : string ; var l1, l2 : string) ;
          var       i, j      : integer ;
                    largeur   : integer ;
                    coupe     : integer ;
begin
  largeur : = length (le__titre) div 2 ;
  i : = largeur ;
  while (le__titre [i] < > ' ') and (i < length (le__titre)) do
     i : = i + 1 ;
  j : = largeur ;
  while (le__titre [j] < > ' ') and (j > 1) do
     j : = j - 1 ;
  if i - largeur < largeur - j
    then
      coupe : = i
    else
      coupe : = j ;
  l1 : = copy (le__titre,1,coupe - 1) ;
  l2 : = copy (le__titre,coupe + 1, length (le__titre) - coupe)
end ;
procedure espacer (var l : string) ;
var       i : integer ;
begin
  i : = 1 ;
  while i < length (l) do
     begin
       i : = i + 2 ;
       insert (' ', l, i - 1)
     end
  end ;
  function max (a, b : integer) : integer ;
  begin
     if a > b then max : = a else max : = b
  end ;
begin
  page (output) ;
  if pos (' ', le__titre) > 0
    then
      chercher__l1__et__l2 (le__titre, l1, l2)
    else
      begin
        l1 : = le__titre ;
        l2 : = ' '
      end ;
  espacer (l1) ;
  espacer (l2) ;
  largeur : = 4 + max (length (l1), length (l2)) ;
  decalage : = (largec - largeur) div 2 ;
  gotoxy (0, 9) ;
  write (' ' : decalage) ;
  for i : = 1 to largeur + 2 do write ('-') ;
  writeln ;
  espace : = (largeur - length (l1)) div 2 ;
  writeln ('' : decalage, '!', '' : espace, l1, '' : espace, '!') ;
  if length (l2) > 0
    then
      begin
        espace : = (largeur - length (l2)) div 2 ;
        writeln ('' : decalage, '!', '' : espace, l2, '' : espace, '!')
      end ;
    write ('' : decalage) ;
    for i : = 1 to largeur + 2 do write ('-')
end ;
```

une nouvelle fonction ou à chaque nouveau menu dans le logiciel, l'écran est effacé pour laisser la place à un autre.

Cette technique nécessite une meilleure étude de l'application et une programmation un peu plus importante. Mais un utilisateur ayant travaillé avec des logiciels où les écrans sont affichés par page n'aimera pas retourner sur des systèmes avec des écrans déroulants.

La procédure Pascal proposée ici utilise une gestion par page de l'écran pour afficher le titre d'un traitement. Dès le lancement d'un programme, elle en indiquera l'objet, en quelques mots. De plus, à l'intérieur d'un logiciel ayant un menu associé à des sous-menus, elle signalera à l'utilisateur dans quelle fonction il se situe.

Cette procédure est déclarée par :
procedure titre (le__titre : string);

Le paramètre LE__TITRE contient le texte spécifiant, en quelques mots, le traitement réalisé par le programme ou par la fonction qui a été appelée.

Le fonctionnement de cette procédure est simple à comprendre. Si le titre qui a été passé en paramètre est constitué d'un seul mot, il sera affiché sur une seule ligne. Par contre, si le titre comporte plusieurs mots, deux lignes seront utilisées. Ainsi, un appel à cette procédure tel que :
titre ('COPIE DE DISQUETTES')
provoquera l'affichage du cadre ci-dessous, centré à l'écran.

Exemple d'affichage d'un titre

```
-------------------------
!   C O P I E  D E    !
!  D I S Q U E T T E S  !
-------------------------
```

Une telle procédure devrait être utilisée dans tous les programmes, dès leur initialisation, afin de leur donner un aspect plus fini, sans en augmenter la complexité.

AFFICHER LE TITRE

D'autre part, le titre indiqué a un aspect documentaire, c'est-à-dire qu'il permet immédiatement de voir si l'on a bien exécuté le programme voulu, ainsi que l'objet du traitement qu'il réalise. Enfin, face à des choix multiples, où chaque choix oriente sur un autre, un tel affichage permet à l'utilisateur de savoir exactement dans quelle partie du logiciel il se trouve.

Sur une ligne ou sur deux ?

Bien que le résultat de cette procédure soit simple, sa programmation est assez complexe, notamment en raison de son caractère général. La constante LARGEC détermine la largeur de l'écran. Elle vaut 80, ici, car nous utilisons un écran de 80 colonnes de large. Bien entendu, avec des écrans à 40, 60 ou 132 colonnes, il faudra modifier la valeur de ce paramètre.

La procédure commence par déterminer si le titre doit être écrit sur une ou deux lignes. La fonction POS, qui indique la position d'une sous-chaîne dans une chaîne décèle la présence d'un espace. Si le titre en comporte, cela signifie qu'il est composé de plusieurs mots. Il pourra donc être affiché sur deux lignes. En l'absence d'espace, le titre est affiché sur une seule ligne.

Il faut noter, à ce sujet, que cette procédure n'effectue aucun contrôle sur la longueur des titres qui lui sont fournis. Un titre trop long provoquera des défauts d'affichage, mais aucune erreur ne sera signalée.

Si le titre peut être affiché sur plusieurs lignes, la procédure CHERCHER_L1_ET_L2 est activée. Un des intérêts de Pascal est ici mis en évidence : cette dernière procédure est locale, c'est-à-dire qu'elle est cachée à l'ensemble du programme qui déclare la routine TITRE.

Ainsi, on voit immédiatement qui appartient à qui, et les logiciels gagnent en fiabilité et en clarté.

Cette sous-procédure détermine à quel niveau le titre peut être coupé. Elle part du milieu de la chaîne LE_TITRE, et l'explore, dans un premier temps vers la droite (indice I), puis vers la gauche (indice J), pour trouver le premier espace qui fournira l'emplacement de la coupure. Comme il est possible que l'on ait trouvé deux de ces emplacements, cette procédure détermine ensuite quel est le meilleur. Il s'agit, bien entendu, de celui qui se rapproche le plus du milieu de la chaîne de caractères.

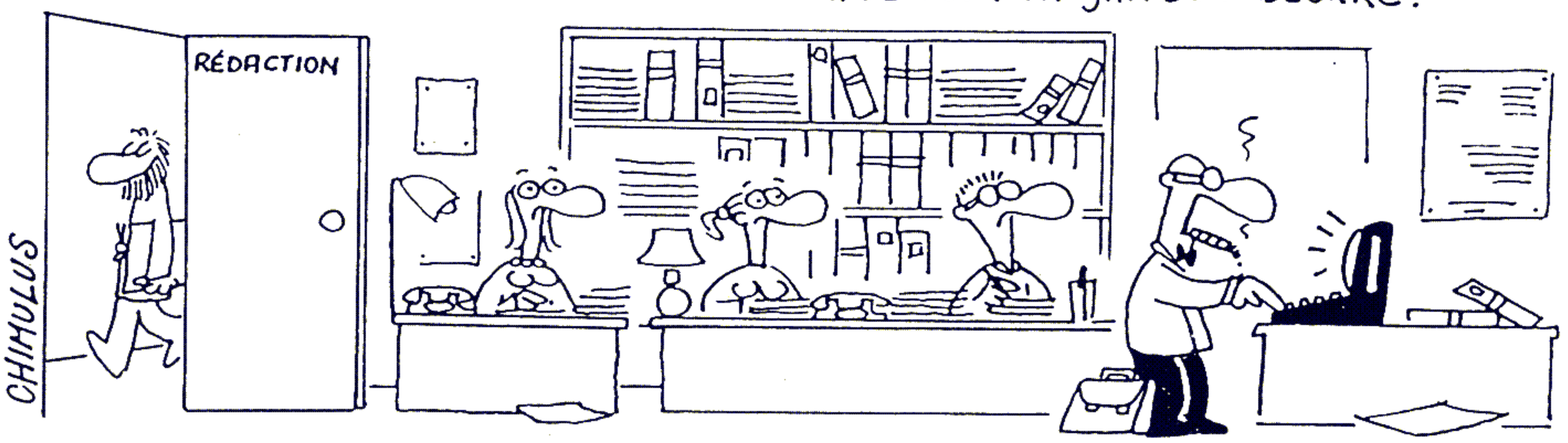
Il est mémorisé dans la variable COUPE. Ensuite, les COUPE – 1 premiers caractères sont placés dans la chaîne L1, et les suivants, à l'exception du blanc séparateur, dans la chaîne L2. La fonction COPY, qui est spécifique au Pascal que nous utilisons, est généralement disponible sous une autre forme. Il est nécessaire de lui fournir trois paramètres. Le premier est la chaîne de caractères, le second l'indice du premier caractère, le troisième le nombre de caractères. Cette fonction extrait de la chaîne originale celle qui est spécifiée par ces paramètres.

Aérer le titre par exemple

Une fois que le titre a été séparé en deux chaînes d'une façon optimale, sa présentation doit être améliorée, il sera par exemple aéré. La sous-procédure ESPACER permet d'effectuer ce travail. Elle insère un espace entre chaque caractère d'une chaîne, ce qui améliore sa lisibilité. Comme Pascal est un langage qui ne dispose pas, dans sa définition, d'instructions de traitement de chaînes, nous avons dû, ici encore, utiliser une procédure non standard. Ainsi, INSERT(C1,C2,I) insère la chaîne C1 dans la chaîne C2 à la position du caractère d'indice I.

Enfin, mise à part la fonction MAX, la procédure TITRE ne possède plus que des instructions d'affichage. La seule particularité à connaître est le format d'affichage en Pascal. Ainsi, "c:f" permet d'afficher la chaîne c sur f caractères de large. Si cette chaîne est vide (notée ' '), ce format permet d'afficher f espaces. Les autres instructions consistent à déterminer la dimension du cadre et à centrer la ou les deux lignes qui y sont présentes. Mais ceci ne peut se faire que grâce aux formules de calcul regroupées en fin de procédure.

Thierry CHAMORET



LE LANGAGE-MACHINE DU CANON X-07 S'IMPLANTE

MÊME bien écrit, un programme en langage-machine n'est rien s'il n'a pas été implanté en mémoire. Pour réaliser cette opération relativement simple, une routine Basic suffit. Il est toutefois nécessaire d'être prudent sur le choix des adresses d'écriture. Pour guider ce choix, étudions l'ensemble des zones qui occupent la mémoire vive.

Après avoir écrit un programme en langage-machine, sur le Canon, il faut l'implanter en mémoire. Une mauvaise implantation risque d'endommager un programme en mode texte, des fichiers en mémoire vive, voire même la zone système, créant inévitablement un « plantage » et le recours obligatoire à un « reset » destructeur. Il s'agit donc de viser juste, le meilleur moyen étant encore d'étudier l'ensemble des zones, de savoir où elles se situent et de quoi elles sont formées.

Tout d'abord, la *zone système* qui va des adresses 0 à 1361. Elle contient des informations essentielles au fonctionnement de la machine, comme par exemple : les adresses d'exécution des instructions graphiques, les coordonnées des curseurs (texte, graphique), les variables temporaires de travail, l'adresse des sous-programmes accessibles par RST en langage-machine.

Cette partie est donc zone interdite, bien que certains utilisent les trous existants pour le stockage de leurs propres informations.

Les zones autorisées

Après la zone système, sont stockés le *programme Basic* en mode texte (modifiable et listable), et les *variables utilisateurs* (simples ou tableaux). Ces variables sont générées par ordres directs, par programme en texte, par programme en fichier. Les adresses respectives de ces trois parties sont fournies par les pointeurs systèmes et évoluent constamment en fonction des besoins.

- Zone programme : de 1362 à VARTAB − 1, où VARTAB − 1 = PEEK (802) + PEEK (803)*256 − 1
- Zone variables simples : de VARTAB à ARYTAB − 1, où ARYTAB − 1 = PEEK (804) + PEEK (805)*256 − 1
- Zone variables tableaux : d'ARYTAB

Exemple d'implantation dans la partie programme

```
                1  xxxxxx...xxxx      '30 'x' si la routine contient 30 codes,
                                       l'adresse du premier étant 1367
implantation    2  AD=1367
                3  READ A:IF A<0 THEN END ELSE
                   POKEAD, A:AD = AD+1:GOTO 3
                4  DATA....., -1      'codes LM

programme       10
réel            ...
                100 EXEC 1367         'appel à la routine LM
                ...
```

Lors de la première utilisation, taper RUN 2 pour mettre en place la routine. Par la suite, RUN 10 suffira.

Il faut noter que l'exécution d'un tel programme ne peut se faire que dans la zone texte (pas en fichier).

à STREND − 1, où STREND − 1 = PEEK (806) + PEEK (807)*256 − 1

Ces trois zones sont à déconseiller, le risque étant grand de détériorer un programme ou des variables. En outre, les modifications peuvent entraîner une action sur les codes du langage-machine. En fait, il est possible d'utiliser la partie programme lorsqu'une routine est propre à un programme Basic. Dans ce cas, il suffit de définir une ligne suffisamment longue pour contenir la routine, puis de *poker* les codes langage-machine (codes LM) à la place des instructions de la ligne (voir l'« Exemple d'implantation dans la partie programme », page précédente).

La zone suivante va de STREND à STKTOP (soit, PEEK (477) + PEEK (478)*256). C'est l'emplacement réservé à la *pile du système*. Elle est utilisée pour stocker de façon temporaire des informations relatives à certaines instructions Basic (GOSUB, FOR, etc.), et surtout aux instructions langage-machine qui interprètent le Basic et gèrent la machine.

La pile n'a pas de taille définie, les données sont « empilées » les unes à la suite des autres. Elle est de type LIFO (Last In - First Out), les données les plus récentes sont récupérées les premières. L'ensemble est suivi par un pointeur, interne au processeur, indiquant constamment le haut de la pile. Pour retarder le plus possible les problèmes d'intersection avec les variables, la pile se déplace des adresses les plus hautes aux adresses les plus basses (voir figure 1). Ainsi, chaque nouvelle ligne Basic ou chaque nouvelle variable réduit la place allouée à la pile, place contrôlée par un pointeur spécial (787-788, ne pas toucher).

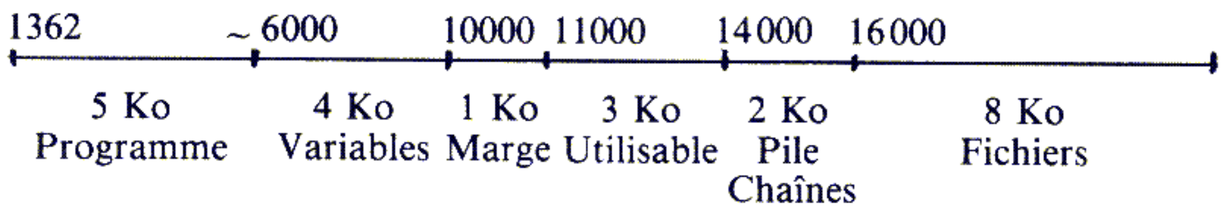
L'espace libre entre la fin des variables et le début — ou la fin — de la pile est utilisable à condition de laisser de chaque côté une marge de sécurité. Du côté de la pile 500 octets sont suffisants, mais il faut noter que :

- FOR consomme 25 octets ; GOSUB 5 octets ;
- la zone réservée aux chaînes, qui est extensible par CLEAR, peut décaler le début de la pile vers le bas de la mémoire.

Pour les variables, les pointeurs systèmes VARTAB et ARYTAB donnent une bonne idée de la zone maximale atteinte à un instant précis. Pour tester, il suffit de mettre en zone texte un programme Basic de grande taille utilisant beaucoup de variables, dont des tableaux. En fait, le problème se pose pour les faibles capacités. Avec le maximum, soit 24 Ko, un exemple de configuration possible est proposé par la figure 2. L'avantage d'une telle méthode est d'avoir en permanence, les routines avec des adresses d'appel fixes qui ne sont pas obligatoirement relogeables.

Figure 1
Rencontre entre variables et pile

Figure 2
Exemple de configuration de la mémoire
(la représentation n'est pas à l'échelle)



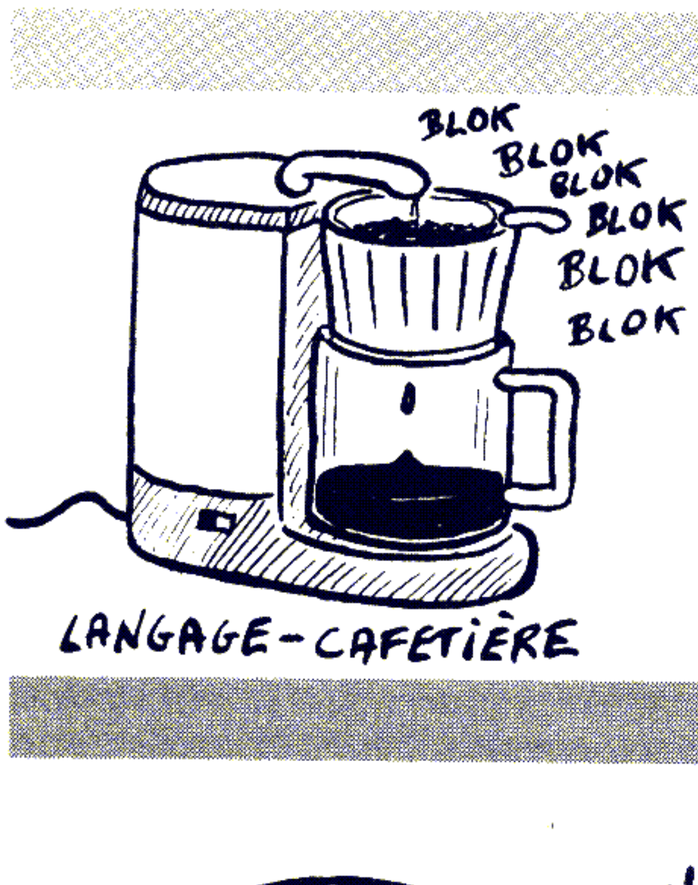
Les 3 Ko de la zone « Utilisable » sont utilisables pour stocker les routines langage-machine avec très peu de risque.

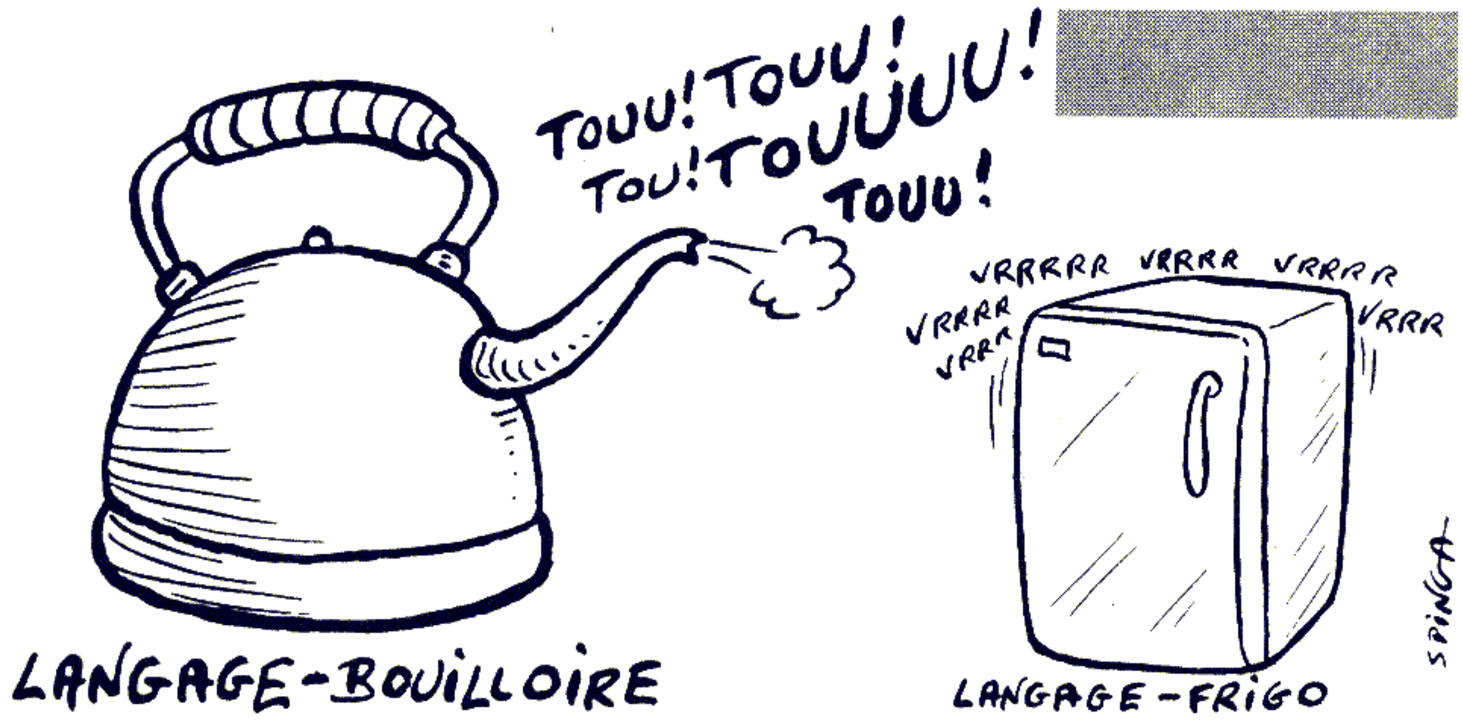
Chaînes et langage-machine

Après la pile, les *chaînes*. En effet, presque toutes les variables alphanumériques ont leur contenu dans cette zone. Sa taille est en principe de 50 octets, mais elle est ajustable par CLEAR. Les pointeurs système STKTOP et MEMSIZ (PEEK(479) + PEEK(480)*256) en donnent la dimension exacte. Comme elle est sujette à modifications, elle est aussi interdite pour le stockage de routines.

Les chaînes étant peu propices au *langage-machine*, passons à la zone suivante qui lui est entièrement dédiée. A la mise en route, sa taille est égale à zéro : MEMSIZ = RAMSTRT (PEEK(528) + PEEK(529)*256). La modification s'effectue à l'aide de CLEAR *x, y* où y est l'adresse maximale que pourra utiliser le Basic. Cette opération repousse d'autant la pile et les chaînes. C'est théoriquement la meilleure zone d'implantation des codes. Malheureusement, à chaque mise en route, elle est remise à zéro (taille nulle). Elle est tout de même intéressante pour le stockage de routines temporaires, mais aussi lorsqu'on ne dispose que de faibles capacités.

Enfin, la zone des *fichiers* en mémoire





Un exemple d'exécution de CHARLM

Soit la routine appelée TEST 1, allant de 8000 à 8100.

Taper RUN''CHARLM''

A l'écran : < S > ai/ < C > ha

En tapant S :

Nom fichier ? Taper TEST 1
Db, Fn ? Taper 8000,8100

La routine est maintenant sauvegardée dans le fichier TEST1.L. Pour la recharger, par exemple à partir de 7050, taper C après le message < S > ai/ < C > ha :

Nom fichier ? Taper TEST 1
Adr début ? Taper 7050

La routine est maintenant chargée à partir de cette adresse.

Utilisation du programme BASLM

Réparation du programme exécutable avec BASLM

RUN''BASLM''

NOM PGM LM ? Taper TEST2 (par exemple) pour le programme à exécuter.

TAILLE ? Taper le nombre d'octets composant la routine.

OCTET 1 ? Taper au fur et à mesure chaque code.

A l'écran : PGM PRET

RUN''TEST2'' exécutera la routine correspondante qui naturellement doit être relogeable.

Programme CHARLM

```
10 CLEAR300:CLS:INPUT"Nom fichier ";F$:DEFINTA,D,F,I,T,V
20 INPUT"<S>ai/<C>ha ";C$:IFC$="C"THEN200
100 INPUT"Db,Fn ";D,F:A$="":T=F-D:FORI=DTOF:A=PEEK(I)
110 A$=A$+CHR$(A):NEXT
120 INIT#1,F$,T+10,"L":PRINT#1,T:PRINT#1,A$:END
200 INPUT"Adr Debut ";D
210 INIT#1,F$,1,"L"::INPUT#1,T:V=PEEK(713)+PEEK(714)*256
220 FORI=1TOT+1:A=PEEK(V+I)
230 POKED,A:D=D+1:NEXT:END
```

va de RAMSTRT à RAMEND (PEEK(530) + PEEK(531)*256). Cette zone permet le stockage de codes langage-machine dans un fichier de données classique, avec une minimisation de l'occupation (1 octet = 1 code). Cette solution intéressante, du fait de la protection de la routine, nécessite des programmes relogeables. Mais l'adresse d'exécution n'est pas fixe, elle évolue avec les créations et les supressions de fichiers. A chaque utilisation, il faut donc déterminer la nouvelle adresse. Avec un programme adéquat, chargeant la routine dans une autre zone de la mémoire, le fichier peut jouer le rôle de sauvegarde. C'est le but du programme CHARLM, facile à utiliser : la routine étant déjà en place à un endroit quelconque de la mémoire (mise en place par un programme classique, avec lignes de DATAs), il faut la sauvegarder dans un fichier. C'est le rôle de la première partie de CHARLM.

Le problème de stockage dans la zone de fichiers est celui de l'évolution de l'adresse d'utilisation de la routine. Pour le résoudre, il suffit de combiner Basic et langage-machine. Ce n'est pas de l'alchimie, mais la simple constatation qu'un fichier programme n'est différent d'un autre que par le *qualifieur* 'P' et la présence de 10 zéros à la fin. En reproduisant, dans un fichier de type 'P' un programme Basic qui détermine l'adresse des codes mémorisés juste après le programme et en exécutant à cette adresse la ligne ci-dessous, on résoud le problème (voir le programme BASLM et son utilisation).

```
1389  10  0  EXEC  PEEK(785)+PEEK(786)*256+29  0 0 0  codes  LM
```

- 1389 : pointeur
- 10 : n° ligne
- 29 : fin programme
- 0 0 0 : adresse obtenue

La dernière possibilité de cette zone est d'utiliser l'espace libre de la fin de la zone fichier s'il y a de la place (voir figure 3). Il est préférable d'implanter en commençant par les adresses les plus proches de RAMEND pour laisser la place à un ajout possible de fichiers. Cette solution a l'avantage d'accepter toutes les routines, relogeables ou non. Elle ne requiert de l'attention que lors d'un SAVE.

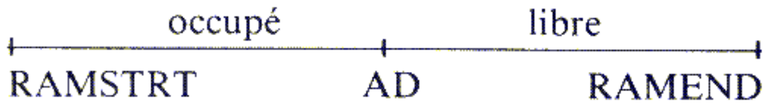
La figure 4 résume les diverses zones de la mémoire. Il ne reste plus qu'à choisir la bonne pour implanter des routines en fonction de leur utilisation et de la place disponible.

Laurent GRAS

```
10 CLS:INPUT"NOM PGM LM ";N$:INPUT"TAILLE ";T
20 INIT#1,N$,T+28,"P":FORI=1TO28:READA:OUT#1,A:NEXT
30 DATA109,5,10,0,168,236,40,55,56,53,41,209,236,40,55
40 DATA56,54,41,211,50,53,54,209,50,57,0,0,0
50 FORJ=1TOT:PRINT"OCTET";J;:INPUTV:OUT#1,V:NEXT
60 CLS:PRINT"PGM PRET":END
```

Programme BASLM

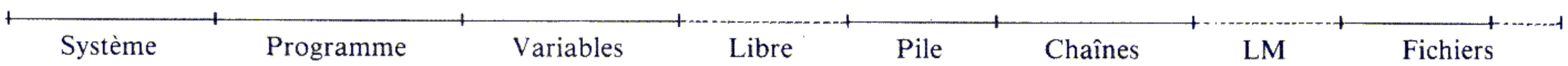
Figure 3
L'espace libre de la zone fichier



AD est obtenu en soustrayant à RAMEND la taille de la zone libre donnée par DIR.

Figure 4
Résumé des diverses zones de la mémoire

(la représentation n'est pas à l'échelle)



VOTRE ORDINATEUR A-T-IL LA BOSSE DES MATHS ?

CERTAINS ordinateurs sont de bons outils pour le dessin, d'autres gèrent mieux les tableaux, d'autres les chaînes de caractères, ou les entrées/sorties. Qu'en est-il des maths ? Là encore, les ordinateurs ne sont pas tous interchangeables. Ils se différencient par le nombre de fonctions mathématiques de leur Basic, ou par la précision des variables dans les calculs. La comparaison est riche d'enseignements.

Les qualités mathématiques d'un ordinateur sont essentiellement évaluées par les fonctions spécifiques et par la précision des variables numériques. Dans le tableau de comparaison ci-contre, nous avons passé en revue sept machines. On remarque déjà que, concernant seulement le nombre des instructions ou fonctions mathématiques, tous les Basic ne se valent pas.

Parmi les machines retenues ce mois-ci, le Lansay arrive largement en tête avec 56 instructions, contre 32 pour le Spectrum. Cependant, il serait absurde de juger la qualité d'un Basic en faisant simplement le total de ses instructions. Il y a plusieurs raisons à cela.

Qui peut le plus peut le moins, bien sûr, mais l'essentiel est de disposer des instructions dont on a personnellement besoin. Ainsi, le Lansay, malgré l'impressionnante collection de fonctions qu'il propose, n'effectue pas les calculs directement en hexadécimal, en binaire ou en octal. Il n'est pas pourvu, non plus, des opérateurs logiques XOR, NOT, EQV ou IMP, contrairement aux machines MSX, par exemple.

Cela dit, le plus souvent, l'absence de fonctions n'est pas rédhibitoire, et l'on obtient tout de même le résultat escompté avec quelques lignes de programme, à partir des autres instructions. L'absence d'un opérateur de division entière n'a rien de tragique si l'on dispose par ailleurs de la fonction INT (partie entière).

Concernant plus particulièrement les maths, la précision des calculs, le nombre de chiffres significatifs, les différents types de variables numériques sont des éléments qui peuvent être déterminants. Pour prendre un exemple extrême, un Basic qui ne connaîtrait que les nombres entiers (cela s'est vu naguère) présenterait un très grave handicap.

Sans aller aussi loin, le Dai ne peut traiter les nombres que dans l'intervalle de – 1E18 à 1E18, alors que le Lansay et le MSX vont de – 9.99999999E62 à 9.99999999E62. Autour du zéro, la précision aussi est importante : pour l'Amstrad, l'Atmos ou le MO 5, tout ce qui se situe entre –2.9E–39 et 2.9E–39 est pris pour zéro, alors que sur le Lansay ou les MSX, l'intervalle pris pour zéro va de –1E–64 à 1E–64.

Il reste que l'ordinateur doit répondre aux besoins de chacun. En ce qui concerne les mathématiques, les points essentiels sont :

- la fonction qui permet de définir des fonctions justement, DEFFN (absente du Dai, du Lansay et du MO 5) ;
- les intervalles de calculs (particulièrement importants sur le Lansay et le MSX) ;
- la présence de la double précision (sur le MSX).

Il y a sans doute d'autres critères qui peuvent être pris en compte pour des applications particulières, mais nous avons retenu les plus importants. Dans le tableau, seuls sept ordinateurs sont présents. D'autres suivront.

LIST

Tableaux récapitulatifs des fonctions mathématiques de sept Basic

Ordinateurs	Amstrad	Atmos	Dai	Lansay 64	MO 5	MSX	Spectrum
Domaine de définition des variables							
entières	– 32 768 à 32 767	– 32 768 à 32 767	-2^{31} à $2^{31}-1$		– 32 768 à 32 767	– 32 768 à 32 767	-2^{32} à $2^{32}-1$
réelles	± 1.7E38 à ± 2.9E – 39	± 1.7E38 à ± 2.9E – 39	± 1E18 à ± 1E – 18	± 9.999E62 ± 1E – 64	± 1.7E38 à ± 2.9E – 39	± 9.999E62 à ± 1E – 64	± 1E38 à ± 2.9E – 39
Précision des calculs (en double précision)	9 chiffres	9 chiffres	6 chiffres	9 chiffres	6 chiffres	7 chiffres (16)	8 chiffres
Conversion de variables							
en entier signé	UNT						
en entier	CINT				CINT	CINT	
en simple précision	CREAL					CSNG	
en double précision						CDBL	
Définition de variables numériques				NUMERIC			
entières	DEFINT		IMP INT		DEFINT	DEFINT	
simple précision	DEFREAL		IMP FPT		DEFSNG	DEFSNG	
double précision						DEFDBL	
Définition de fonction	DEFFN	DEFFN				DEFFN	DEFFN
Arithmétique simple							
4 opérations	+ – * /	+ – * /	+ – * /	+ – * /	+ – * /	+ – * /	+ – * /
puissance	↑	↑	↑	^	^	^	↑
division entière	\				@	\	
reste division	MOD		MOD	REM	MOD	MOD	
reste entier et positif				MOD			
racine carrée	SQR	SQR	SQR	SQR	SQR	SQR	SQR
valeur absolue	ABS	ABS	ABS	ABS	ABS	ABS	ABS
signe	SGN	SGN	SGN	SGN	SGN	SGN	SGN
partie entière	INT	INT	INT	INT	INT	INT	INT
partie décimale			FRAC	FP			
nombre sans sa partie décimale				IP	FIX	FIX	
entier supérieur				CEIL			
arrondi	ROUND			ROUND			
troncature	FIX			TRUNCATE			
maximum	MAX			MAX			
minimum	MIN			MIN			
Opérateurs et fonctions logiques							
6 opérateurs de relation	oui	oui	oui	oui	oui	oui	oui
et	AND	AND	AND	AND	AND	AND	AND
ou	OR	OR	OR	OR	OR	OR	OR
ou exclusif	XOR				XOR	XOR	
négation	NOT	NOT			NOT	NOT	NOT
équivalence					EQV	EQV	
implication					IMP	IMP	
et binaire		AND	IAND	BAND			
ou binaire		ORA	IOR	BOR			
ou exclusif binaire		EOR	IXOR				
non binaire			INOT				
décalage binaire			SHL, SHR				

Ordinateurs	Amstrad	Atmos	Dai	Lansay 64	MO 5	MSX	Spectrum
Fonctions trigonométriques et angulaires							
sinus	SIN	SIN	SIN	SIN	SIN	SIN	SIN
cosinus	COS	COS	COS	COS	COS	COS	COS
tangente	TAN	TAN	TAN	TAN	TAN	TAN	TAN
cotangente				COT			
arcsinus			ASIN	ASIN			ASN
arccosinus			ACOS	ACOS			ACS
arctangente	ATN	ATN	ATN	ATN		ATN	ATN
sinus hyperbolique				SINH			
cosinus hyperbolique				COSH			
tangente hyperbolique				TANH			
sécante				SEC			
cosécante				CSC			
pi	PI	PI	PI	PI			PI
option mesures d'angles				OPTION			
conversion radians-degrés	DEG			DEG			
conversion degrés-radians	RAD			RAD			
calculs en	degrés ou radians	radians	radians	degrés ou radians	radians	radians	radians
Fonctions logarithmes							
exponentielle	EXP	EXP	EXP	EXP	EXP	EXP	EXP
E, puissance de 10	E	E	E	E	E	E ou D	E
log népérien	LOG	LN	LOG	LOG	LOG	LOG	LN
log décimal	LOG 10	LOG	LOGT	LOG 10			
log base 2				LOG 2			
inverse du log népérien			ALOG				
Les bases							
calculs hexa	&H ou &		#			&H	
calculs binaires	&X					&B	BIN
calculs en octal						&O	
Conversions d'une base à l'autre							
binaire - décimal	STR$(&X			BIN			
hexadécimal - décimal	STR$(&H		PRINT #				
décimal - binaire	BIN$					BIN$	
décimal - hexadécimal	HEX$	HEX$	HEX$			HEX$	
décimal - octal						OCT$	
Autres fonctions scientifiques							
générateur aléatoire	RANDOMIZE		interne	RANDOMIZE			RANDOMIZE
générateur d'entiers	RND			RND (X)			
générateur de réels entre 0 et 1		RND	RND (1)	RND	RND	RND	RND
l'infini				INF			
epsilon				EPS			
Nombre de fonctions	50	33	44	56	34	46	32

LE BASIC DE L'ATARI 130 XE

SOUS une forme nouvelle et agréable à l'œil, l'Atari 130 XE cache un Basic semblable à celui de son prédécesseur, le 800 XL. Il se distingue de ce dernier par la taille de sa mémoire vive : elle est deux fois plus importante, c'est-à-dire riche de 128 Ko. C'est un avantage pour cet ordinateur relativement bon marché qui va pouvoir profiter de tous les logiciels – et ils sont nombreux – conçus pour le 800 XL.

L'Atari 130 XE s'intègre parfaitement dans la ligne des ordinateurs de jeux propre à Atari : la couleur de l'affichage change après quelques minutes d'immobilité afin de ne pas abîmer le téléviseur, et la compatibilité avec les précédents modèles est totale.

Son Basic est classique chez Atari, il se trouvait déjà dans les 600 et 800 XL. Toutefois, l'utilisateur plus exigeant peut le gonfler avec une cartouche « Basic Microsoft » proposée en option (voir la description du contenu de cette cartouche dans l'encadré page suivante).

Sa principale qualité est une grande simplicité. Les instructions sont très classiques et la syntaxe est souple. En revanche, le traitement des tableaux est déficient : ce Basic n'admet que des tableaux alphanumériques unidimensionnels. Et pourtant, il traite des variables alphanumériques et numériques, avec pour ces dernières, une précision tout à fait appréciable : il fait les calculs sur 9 chiffres, les valeurs pouvant aller jusqu'à 9.999999E + 97.

Graphisme et son, de quoi faire

L'affichage de 24 lignes de 40 caractères bénéficie d'un très bon éditeur pleine page. Le graphisme est soigné : 16 modes graphiques qui forment un bon compromis entre la résolution, la palette et l'encombrement en mémoire. Par exemple, en deux couleurs, la résolution est de 320 points sur 192. Par rapport à de telles possibilités graphiques, les instructions du Basic qui s'y rapportent sont peu nombreuses. Par exemple, l'instruction CIRCLE qui trace un cercle sur certains Basic, est absente ici.

Le 130 XE fait aussi de la musique grâce au mot clé SOUND accompagné de quatre paramètres : la voie (0 à 3),

la fréquence (0 à 255), la distorsion (0 à 14), le volume (0 à 15). Mais, sous peine de risquer de détériorer le haut-parleur du téléviseur, le volume ne doit pas dépasser la valeur 12. Les quatre voies sont utilisables simultanément.

Les 128 Ko disponibles sont gérés par le Basic de façon transparente à l'utilisateur. Cette mémoire supplémentaire n'est pas directement utilisable par le 6502 (processeur du 130 XE) et par la carte Antic (écran et ports d'entrées/sorties). C'est la raison pour laquelle cette gestion est assurée par pages de 16 Ko.

Fiche technique de l'Atari 130 XE

Constructeur : *Atari*
Prix public : *2 300 FF en Secam, 2 000 FF en Pal*
Mémoire vive : *128 Ko*
Mémoire morte : *24 Ko*
Processeur : *6502 C*
Langages : *Basic, langage-machine*
Précision des calculs : *9 chiffres*
Variables : *alphanumériques ; numériques, sans distinction entre entières et décimales, jusqu'à 9.999999E+97*
Nombre de mots clés du Basic : *74*

Pour illustrer simplement les capacités de ce Basic, on peut se reporter à la liste des mots clés en encadré, et pour plus de précision, voir l'essai du Basic de l'Atari 800 XL publié dans *LIST* 4, page 34.

Pour 2300 FF ttc, l'Atari 130 XE est une belle machine qui dispose d'un Basic simple, de la possibilité d'être programmé en langage-machine, et surtout des nombreux logiciels déjà existants.

Franck-Olivier LELAIDIER

Liste des mots clés de l'Atari 130 XE

ABS
ADR
AND
ASC
ATN

BYE

CHR$
CLOAD
CLOG
CLOSE
CLR
COLOR
COM
CONT
COS
CSAVE

DATA
DEG
DIM
DOS
DRAWTO

END
ENTER
EXP

FOR NEXT
FRE

GET
GOSUB
GOTO
GRAPHICS

IF THEN
INPUT
INT

LEN
LET
LIST
LOAD
LOCATE
LOG
LPRINT

NEW
NOT
NOTE

ON
OPEN
OR

PADDLE
PEEK
PLOT
POINT
POKE
POP
PRINT
POSITION
PTRIG
PUT

RAD
READ
REM
RESTORE
RETURN
RND

SGN
SETCOLOR
SIN
SOUND
SQR
STEP
STOP
STR$

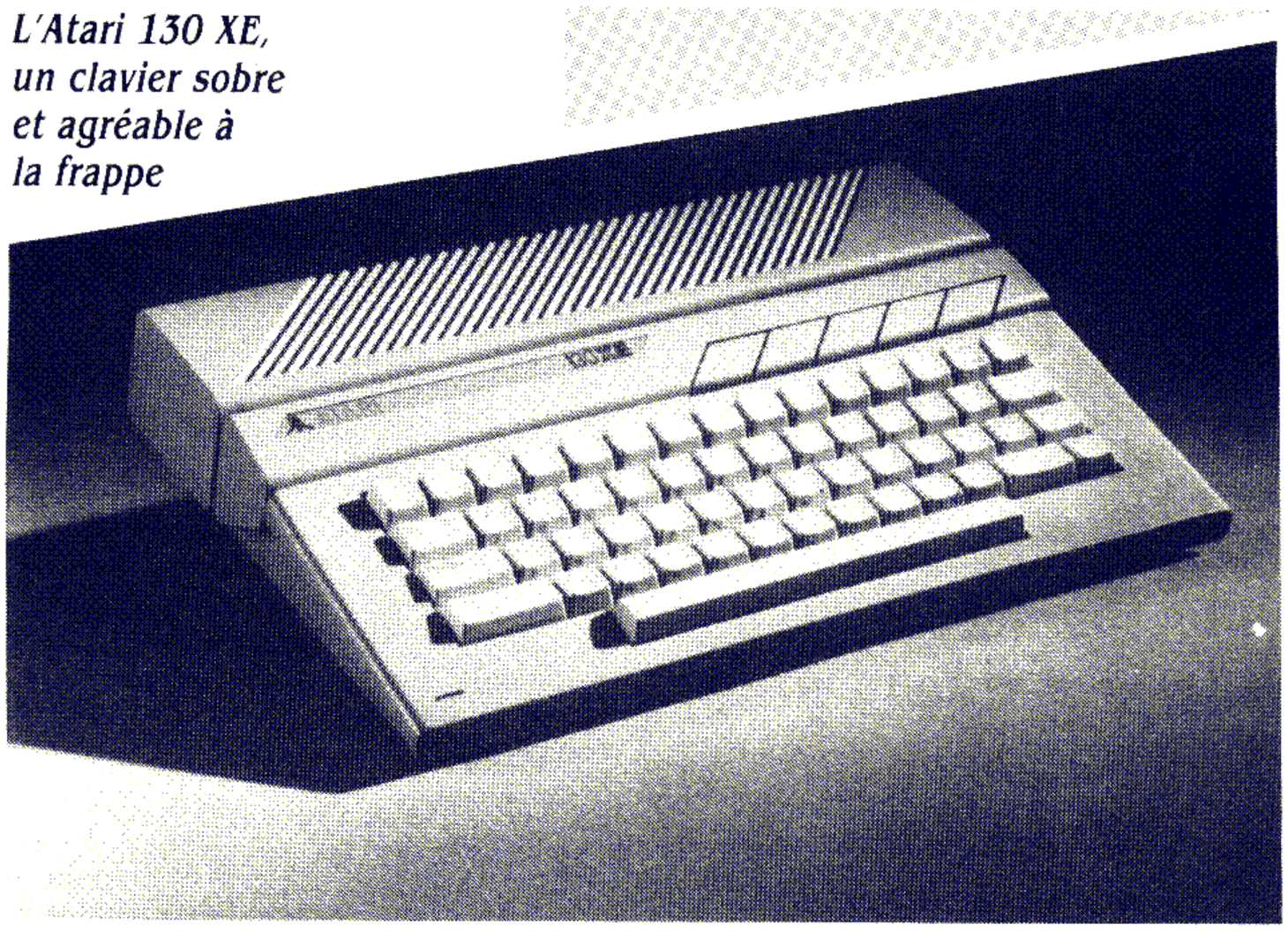
TRAP

USR

VAL

XIO

L'Atari 130 XE, un clavier sobre et agréable à la frappe



La cartouche Microsoft étend le Basic du 130 XE

Sous la forme d'une cartouche et d'une disquette, le Basic Microsoft Atari vient compléter le Basic initial du 130 XE. La cartouche peut être utilisée sans la disquette, mais cette dernière apporte une extension appréciable.

L'amélioration la plus tangible de cette extension réside dans le traitement des tableaux et des variables. Les tableaux alphanumériques admettent maintenant 255 dimensions, et la distinction est faite entre les variables numériques entières, en virgule flottante, en simple précision ou en double précision.

L'instruction IF THEN est complétée par ELSE, les chaînes de caractères sont traitées par INSTR et STRING$. La mise au point des programmes, un peu oubliée dans la version de base, s'équipe : RENUM pour la renumérotation, AUTO pour la numérotation automatique, TRON et TROFF pour suivre son programme pas à pas.

La chasse aux erreurs est désormais possible avec toute la panoplie du traitement des erreurs : ERR, ON ERROR GOTO, RESUME, RESUME NEXT. Le formatage des sorties est géré par l'instruction PRINT USING auréolée d'options. Par exemple :
• PRINT USING "## ^^^^ ";X prépare l'affichage d'un nombre sous forme exponentielle ;
• PRINT USING "- - - -" imprimera le signe "-" derrière le nombre négatif. Ainsi, -998 RUN donne 998-.

Le point d'exclamation extrait et affiche un caractère d'une chaîne ; le signe "%" extrait une portion de chaînes, etc.

L'horloge interne du 130 XE est accessible par les instructions TIME et TIME$ qui renvoient respectivement le temps écoulé en un multiple de 1/60e de seconde et sous forme d'une chaîne de caractères (HH:MM:SS).

Cette extension du Basic Atari, cartouche et disquette, coûte environ 600 FF ttc.

Augustin GARCIA

METTRE DES DONNÉES EN BOÎTE

APRÈS Amstrad-graphe, série de programmes graphiques (1), voici Amsfiche qui gère des fichiers. Ce programme — didactique — quitte le domaine des graphismes pour aborder les rivages, jonchés d'écueils, de la conservation des données. Écrit sur Amstrad, il peut facilement être adapté à d'autres Basic.

Pour mettre des données en boîte, rien de tel qu'un programme de gestion de fichiers, même simplifié. Il nécessitera pour son fonctionnement un magnétophone à cassettes, celui qui est intégré à l'Amstrad CPC 464 convenant fort bien. Avec une unité de disquettes ou un CPC 664, des adaptations s'imposent. Dans tous les cas, une imprimante sera bienvenue, mais pas indispensable. Nous avons tout de même choisi de l'utiliser (elle est plus utile, ici, qu'un lecteur de disquettes) pour étendre le domaine d'étude.

Puisque notre choix est limité au stockage de données sur bande magnétique, nous restreignons du même coup le champ des possibilités d'accès à ces données. En effet, la bande ne se déroulant que dans un seul sens (du début à la fin des données), nous ne pourrons atteindre la huitième donnée qu'après avoir accédé à la première, puis à la seconde, etc. Notre fichier sera donc *séquentiel* par la force des choses, c'est-à-dire qu'il sera parcouru de la première à la dernière donnée, et dans cet ordre nécessairement.

Avec un lecteur de disquettes, nous aurions pu bénéficier d'un autre type d'accès, l'*accès direct*. En effet, la structure du lecteur est telle qu'elle permet d'accéder à la huitième donnée (par exemple !) sans passer par celles qui précèdent. Le lecteur de disquettes permet encore d'autres modes d'accès, mais pour l'instant nous ne possédons qu'un magnétophone : à nous d'en tirer le meilleur parti !

Supposons que vous déteniez une collection de disques (ou de bandes vidéo, ou de programmes, etc.) et que vous souhaitiez en tenir un répertoire. La solution pré-informatique — j'allais dire préhistorique — consistait à tenir à jour des fiches cartonnées, toutes stockées dans une jolie boîte ou un magnifique tiroir, contenant les informations indispensables sur les éléments de la collection. Par exemple, le nom de l'interprète, celui du morceau, le numéro du disque, etc.

Une bonne volonté dévorante

Oui mais voilà, un Amstrad flambant neuf brûle de vous servir... Aussi, faisant feu de sa bonne volonté dévorante, vous allez pouvoir fabriquer ce même fichier sans boîte, sans bristol, sans gomme ni crayon : il s'agit donc d'une économie substantielle compensée mal-

(1) Voir LIST *6 p. 61*, LIST *8 p. 65*, LIST *9 p. 40.*

Amsfiche
Programme pour Amstrad CPC 464
Auteur Jean-Pierre Lalevée


```
100 KEY 139,"MODE 1:LIST"+CHR$(13)
110 REM-----------------------------------
120 REM  AMSFICHE - FICHIER SEQUENTIEL
130 REM         AMSTRAD CPC 464
140 REM-----------------------------------
150 :
160 ON ERROR GOTO 1460
170 MX=99:REM NBRE MAXI DE FICHES PREVUES
180 RU=1:REM NOMBRE DE RUBRIQUES PREVUES
190 DIM RB$(RU),F$(MX,RU)
200 RB$(0)="INTERPRETE":RB$(1)="NO DU DISQUE"
210 SL$=" "+STRING$(38,CHR$(154))
220 MODE 1:LOCATE 11,2:PEN 2:PRINT"FICHIER SUR CASSETTES":PRINT SL$
230 :
240 REM ++++++++++++ MENU ++++++++++++++
250 LOCATE 18,5:PEN 3:PRINT"MENU":PRINT:PEN 1:ZONE 8
260 PRINT:PRINT,"1- CHARGEMENT EN MEMOIRE"
270 PRINT:PRINT,"2- MODIFICATION DE FICHE"
280 PRINT:PRINT,"3- SAUVEGARDE DU FICHIER"
290 PRINT:PRINT,"4- CREATION  DU  FICHIER"
300 PRINT:PRINT,"5- IMPRESSION SUR PAPIER"
310 PRINT:PRINT,"0- FIN DU TRAVAIL"
320 R$="":WHILE R$<"0" OR R$>"5":LOCATE 2,22:INPUT"VOTRE CHOIX )",R$:WEND
330 CH=VAL(R$)
340 IF CH=0 THEN CLS:PRINT"AU REVOIR !":GOTO 1530
350 ON CH GOSUB 380,500,890,1000,1180:GOTO 220
360 :
370 REM ******* SOUS/PROGRAMMES *******
380 REM ===== CHARGEMENT FICHIER =====
390 CLS:PEN 2:PRINT,"MISE EN MEMOIRE DU FICHIER":PRINT SL$:PEN 3
400 PRINT:PRINT" POSITIONNEZ LA CASSETTE-FICHIER SVP..."
410 PEN 1:LOCATE 2,6:INPUT"TITRE DU FICHIER ";R$
420 :
430 OPENIN R$:INPUT#9,NF
440 CLS:PEN 2:LOCATE 5,10:PRINT"LE FICHIER CONTIENT";NF;"FICHES."
450 :
460 FOR I=1 TO NF:FOR J=0 TO RU:INPUT#9,F$(I,J):NEXT J,I
470 CLOSEIN:PEN 3:PRINT:PRINT:PRINT"CHARGEMENT TERMINE..."
480 GOSUB 1490:RETURN
490 :
500 REM ==== MODIFICATION FICHIER ====
510 CLS:PEN 2:PRINT,"MODIFICATION DE FICHES":PRINT SL$
520 IF NF=0 THEN PEN 3:LOCATE 2,4:PRINT"LE FICHIER N'EST PAS EN MEMOIRE !":GOSUB
 1490:RETURN
530 PEN 3:LOCATE 11,5:PRINT"MENU MODIFICATIONS":PRINT:PEN 1
540 PRINT,"1- SUPPRIMER UNE FICHE"
550 PRINT,"2- AJOUTER UNE FICHE"
560 PRINT,"3- MODIFIER UNE FICHE"
570 PRINT,"0- RETOUR AU MENU PPAL"
580 PEN 2:R$="":WHILE R$<"0"OR R$>"3":LOCATE 1,15:INPUT"VOTRE CHOIX ";R$:WEND
590 R=VAL(R$):IF R=0 THEN RETURN
600 ON R GOSUB 630,710,790
610 GOSUB 1490:CLS:GOTO 530
620 :
630 REM #### SUPPRIMER UNE FICHE
640 CLS:PEN 2:PRINT"SUPPRESSION DE FICHE:"
650 R$="":WHILE VAL(R$)=0 OR VAL(R$)>NF:LOCATE 1,6:INPUT"NO DE LA FICHE A SUPPRI
MER ";R$:WEND
660 R=VAL(R$):IF R=NF THEN 690
670 :
680 FOR I=R+1 TO NF:FOR J=0 TO RU:F$(I-1,J)=F$(I,J):NEXT J,I
690 NF=NF-1:GOTO 860
700 :
710 REM ### AJOUTER UNE FICHE
720 CLS:PEN 2:PRINT"AJOUT D'UNE FICHE:"
730 NF=NF+1:PEN 3:LOCATE 1,4:PRINT"FICHE NO";NF
740 IF NF>MX THEN PEN 3:LOCATE 1,20:PRINT"PLUS DE PLACE EN MEMOIRE !":RETURN
750 PEN 1:FOR I=0 TO RU
760 R$="":WHILE R$="":LOCATE 1,6+I*2:PRINT RB$(I);:INPUT R$:WEND
770 F$(NF,I)=UPPER$(R$):NEXT I:GOTO 860
780 :
790 REM #### MODIFIER UNE FICHE
800 CLS:PEN 2:PRINT"MODIFICATION DE FICHE:"
810 PEN 1:R$="":WHILE VAL(R$)=0 OR VAL(R$)>NF:LOCATE 2,4:INPUT"NO DE FICHE A MOD
IFIER ";R$:WEND
820 R=VAL(R$):FOR I=0 TO RU
830 R$="":WHILE R$="":LOCATE 1,6+I*2:PRINT RB$(I);:INPUT R$:WEND
```

```
▶ 840 F$(R,I)=UPPER$(R$):NEXT I
  850 :
  860 PEN 3:LOCATE 1,20:PRINT"MODIFICATION EFFECTUEE...":GOSUB 1490
  870 RETURN
  880 :
  890 REM ==== SAUVEGARDE FICHIER ======
  890 REM ==== SAUVEGARDE FICHIER ======
  900 CLS:PEN 2:PRINT,"SAUVEGARDE DU FICHIER":PRINT SL$
  910 IF NF=0 THEN PEN 3:LOCATE 2,4:PRINT"LE FICHIER N'EST PAS EN MEMOIRE !":GOSUB
   1490:RETURN
  920 PEN 3:LOCATE 2,4:PRINT"    POSITIONNEZ LA CASSETTE SVP...    ":PEN 1
  930 TF$="":WHILE TF$="":LOCATE 1,8:INPUT"TITRE DU FICHIER ";TF$:WEND
  940 OPENOUT TF$:PRINT#9,NF
  950 FOR I=1 TO NF:FOR J=0 TO 1:PRINT#9,UPPER$(F$(I,J)):NEXT J,I
  950 CLOSEOUT
  970 PEN 3:LOCATE 1,20:PRINT"SAUVEGARDE TERMINEE...":GOSUB 1490
  980 RETURN
  990 :
 1000 REM ======= CREATION FICHIER =====
 1010 CLS:WINDOW#1,1,40,1,6:WINDOW#2,1,40,9,25
 1020 PEN#1,2:PRINT#1,TAB(11)"CREATION DU FICHIER":PRINT#1,SL$
 1030 PEN#1,3:PRINT#1:PRINT#1,RU+1;"RUBRIQUES &";MX;"FICHES PREVUES."
 1040 PRINT#1,"    ))) POUR FINIR, TAPER '!' ((("
 1050 :
 1060 NF=1:REM NBRE DE FICHES
 1070 PEN#2,1:CLS#2:PRINT#2,"FICHE N.";NF:PEN#2,2
 1080 J=0:R$="":WHILE J<=RU AND R$<>"!"
 1090 LOCATE#2,1,4+J*2:PRINT#2,RB$(J);:INPUT#2,R$:F$(NF,J)=UPPER$(R$)::J=J+1
 1100 WEND
 1110 PEN#2,3:LOCATE#2,1,16:PRINT#2,"OK ?":S$="":WHILE S$<"N"OR S$>"O":S$=UPPER$(
INKEY$):WEND
 1120 IF S$="N" THEN 1070:REM ERREUR
 1130 IF R$="!" THEN NF=NF-1:GOTO 1160:REM TERMINE
 1140 NF=NF+1:IF NF<=MX THEN 1070:REM SUITE
 1150 PEN#2,3:LOCATE#2,1,13:PRINT#2,"PLUS DE PLACE EN MEMOIRE !":PRINT#2,"SAUVEGA
RDEZ LE FICHIER SVP...":GOSUB 1490
 1160 RETURN
 1170 :
 1180 REM ===== IMPRESSION FICHIER =====
 1190 MODE 1:PEN 2:PRINT,"IMPRESSION SUR PAPIER":PRINT SL$:PEN 3:LOCATE 2,4
 1200 IF NF=0 THEN PRINT"LE FICHIER N'EST PAS EN MEMOIRE !":GOSUB 1490:RETURN
 1210 PRINT"METTEZ EN FONCTION L'IMPRIMANTE SVP..."
 1220 LOCATE 1,10:PRINT STRING$(39," ")
 1230 PEN 1:LOCATE 18,7:PRINT"MENU":PRINT:PEN 2
 1240 PRINT,"1- TOUT LE FICHIER"
 1250 PRINT,"2- SELON RUBRIQUE CHOISIE"
 1260 PEN 1:R$="":WHILE R$<"1"OR R$>"2":LOCATE 1,15:INPUT"VOTRE CHOIX ";R$:WEND
 1270 WIDTH 40:PRINT#8,"   NO       ";RB$(0);"       ";RB$(1)
 1280 IF R$="2"THEN 1370
 1290 :
 1300 REM ##### TOUT LE FICHIER
 1310 CLS:LOCATE 8,8:PRINT"IMPRESSION DE TOUT LE FICHIER"
 1320 FOR I=1 TO NF
 1330 PRINT#8,TAB(4);I;TAB(13);F$(I,0);TAB(35);F$(I,1)
 1340 NEXT:GOTO 1440
 1350 :
 1360 REM #### UNE RUBRIQUE AU CHOIX
 1370 CLS:PEN 2:PRINT"RUBRIQUES DEFINIES:":PEN 1:PRINT:FOR I=0 TO RU:PRINT,I;"-";
RB$(I):NEXT
 1380 PEN 2:R$="":WHILE R$<"0" OR VAL(R$)>RU:LOCATE 1,6:INPUT"NO DE LA RUBRIQUE C
HOISIE ";R$:WEND
 1390 NR=VAL(R$):PEN 3:PRINT:PRINT"RUBRIQUE:";RB$(NR)
 1400 R$="":WHILE R$="":LOCATE 1,9+RU:INPUT"CLE DE TRI ";R$:WEND
 1410 R$=UPPER$(R$):FOR I=1 TO NF
 1420 IF F$(I,NR)=R$ THEN PRINT#8,TAB(4);I;TAB(13);F$(I,0);TAB(35);F$(I,1)
 1430 NEXT
 1440 PRINT#8,"OK.":PRINT#8:RETURN
 1450 :
 1460 REM ********** ERREUR ! ***********
 1470 PRINT:PRINT"ERREUR";ERR;"EN LIGNE";ERL;":END
 1480 :
 1490 REM ******* TEMPORISATION ********
 1500 FOR I=0 TO 1500:NEXT
 1510 RETURN
 1520 :
 1530 END
```

Le programme ligne à ligne

Ligne 100 : cette ligne permet de mettre au point le programme avec plus de facilité.

Ligne 160 : associée aux lignes 1460 et 1470, elle pourra faciliter le travail de recherche des erreurs lors de la mise au point du programme, ou servira d'aide à l'utilisateur du programme au cas où surviendrait une bogue intempestive. Le contenu de la ligne 1470 pourra être modifié à volonté. Ce n'est ici qu'un exemple.

Ligne 170 : on prévoit la possibilité d'enregistrer 99 fiches dans le fichier. Si la collection de disques est importante, cette valeur est modifiable, la limite étant ici celle de la capacité de la mémoire de l'Amstrad.

Lignes 180 et 200 : pour des raisons de simplification, deux rubriques seulement sont prévues par fiche, l'interprète et le numéro du disque. Après avoir essayé le programme, vous pourrez ajouter les rubriques qui vous intéressent : leur nombre moins un en ligne 180, leur intitulé en ligne 200 (on peut créer de nouvelles lignes si besoin est).

Ligne 190 : toujours pour simplifier, les données du fichier seront contenues dans un tableau de caractères à deux dimensions. Pour plus de précisions, voir le tableau F$ (X, Y), avec un exemple de contenu (ci-dessous).

Lignes 220 à 340 : c'est le menu principal qui énumère les fonctions disponibles que l'utilisateur peut appeler.

Ligne 350 : en fonction du choix fait par l'utilisateur, le programme se dirige vers les sous-programmes correspondants. Le programme est donc modulaire, ce qui facilite à la fois son écriture et sa mise au point.

Lignes 410 et 430 : pour mettre le fichier en mémoire (il est censé se trouver sur la cassette-fichier), le programme sollicite l'entrée du titre attribué au fichier (si vous l'avez oublié, pressez simplement sur ENTER). Ensuite, la commande OPENIN R$ a pour effet de provoquer l'ouverture du fichier en mode lecture. Le magnétophone doit donc se mettre à marcher après les injonctions habituelles. Mais n'oubliez pas de positionner la cassette à l'emplacement où commence le fichier !

Lignes 430 et 440 : la première donnée lue représente le nombre de fiches existantes. Pour la lire, l'instruction INPUT est suivie d'un paramètre qui représente le numéro de périphérique du magnétophone. Rappelons que les valeurs 0 à 7 sont les numéros des fenêtres d'écran, que le 8 représente l'imprimante, le 9 est donc réservé au magnétophone.

Ligne 460 : de la même façon, on extrait tous les éléments du fichier, qui sont replacés dans leur tableau.

Ligne 470 : lorsque tous les éléments ont été lus, on referme le fichier avec CLOSEIN.

Lignes 500 à 610 : ici est prévu un menu secondaire qui appelle les sous-programmes divers destinés aux modifications de fiches. Toutes les modifications s'effectuent dans le tableau : aucun accès au magnétophone n'est nécessaire et la cassette-fichier peut être sans inconvénient sortie du magnétophone.

Ligne 680 : lorsqu'une fiche est supprimée, on décale vers le haut tous les éléments du tableau pour combler l'espace provoqué par la suppression.

Ligne 930 : pour la sauvegarde du fichier (nouvellement créé ou modifié), le programme demande le titre sous lesquel il pourra être rappelé.

Ligne 940 : après ouverture en écriture (OPENOUT), le programme inscrit aussitôt le nombre de fiches (voir ligne 430)...

Ligne 950 : ...puis tous les éléments du tableau sont transférés sur la bande.

Ligne 960 : le fichier est refermé.

Lignes 1000 à 1170 : la création d'un nouveau fichier consiste à stocker en mémoire (dans le tableau créé en début de programme) tous les éléments qui constitueront le futur fichier-cassette.

Ligne 1120 : permet de corriger immédiatement un enregistrement erroné.

Ligne 1130 : l'appui sur la touche "!" en lieu et place d'une réponse cohérente permet de sortir de ce sous-programme lorsque tous les éléments du fichier ont été entrés.

Ligne 1140 : le programme surveille constamment le nombre de fiches entrées, pas plus de 99 dans notre exemple (voir ligne 170).

Lignes 1180 à 1440 : l'impression du fichier s'effectue, comme nous l'avons annoncé, sur une imprimante. L'utilisateur peut choisir l'impression de la totalité du fichier, ou d'une partie seulement selon le critère de son choix.

Si vous ne possédez pas d'imprimante, il vous suffira de supprimer tous les '#8' ou '#8,' qui dirigent les caractères vers l'imprimante. Vous devrez aussi prévoir une possibilité d'arrêt du défilement de l'écran lors d'un affichage long.

heureusement par l'achat d'une cassette neuve pour le magnétophone.

Pour que notre exposé soit complet, sachez que la collection fictive de fiches que votre machine est susceptible de créer portera également le nom de *fichier* ; que les fiches qui le composent s'appellent des *enregistrements* ; que chaque enregistrement est découpé en *rubriques* ou *champs*.

Amsfiche, c'est son nom, est un programme de gestion de fichiers simplifié. Il permet donc les manipulations indispensables telles que la création et la modification du fichier, l'impression éventuellement triée de son contenu, et bien sûr sa sauvegarde et sa récupération sur cassette, sans lesquelles le programme n'aurait aucun intérêt. Par contre, il ne permet pas le rangement alphabétique des fiches, ou le tri selon plusieurs critères ; toutes choses qu'il est possible d'ajouter, les explications sur le fonctionnement du programme devant le permettre (voir l'encadré ci-contre).

Quant à savoir si l'utilisation d'un fichier sur cassette présente des avantages par rapport à son homologue en tiroir, vous n'aurez la réponse qu'après l'avoir essayé. Car rien ne vaut l'expérience !

Grâce aux explications sur le programme, on remarque que le magnétophone ne sert finalement que de moyen de stockage. En effet, l'accès séquentiel oblige à stocker toutes les données en mémoire centrale avant de pouvoir les modifier. Ces modifications étant faites, une réécriture de l'intégralité du fichier est indispensable.

Aussi, n'oubliez surtout pas, après avoir travaillé sur le fichier, d'en effectuer une nouvelle sauvegarde. Si par hasard vous quittez le programme en oubliant cette manœuvre essentielle, ne le relancez pas par RUN : les données présentes en mémoire seraient immédiatement détruites. En revanche, un GOTO 220 effectué avant que l'irréparable ne soit accompli pourra éviter un tel accident.

Jean-Pierre LALEVÉE

Le tableau F$ (X, Y), un exemple de contenu

Numéro de fiche	**. . . 6**	**7**	**8**	**9**	**10**	**11 . . .**
Interprète F$ (X, 0)	. . .	"Mireille Mathieu" F$ (7, 0)	"Johnny Halliday" F$ (8, 0)	"Chantal Goya" F$ (9, 0)	"Johnny Halliday" F$ (10, 0)	. . .
Numéro du disque F$ (X, 1)	. . .	"3" F$ (7, 1)	"4" F$ (8, 1)	"2" F$ (9, 1)	"7" F$ (10, 1)	. . .

CREATIVE GRAPHICS ET TRACK BALL

LA CRÉATION GRAPHIQUE SUR MSX

LE Macintosh, ses icônes, sa souris, son célèbre logiciel graphique Macpaint font des émules. La preuve ? Un logiciel de création graphique pour MSX, Creative Graphics, dirigé par un « track ball », entendez par là une souris renversée. Édité par Sony, ce logiciel se présente sous la forme d'une cartouche de mémoire morte. Il est complété par une petite documentation très bien conçue.

Un logiciel graphique, *Creative Graphics*, est disponible sur les ordinateurs au standard MSX. Ses performances sont convaincantes, à condition qu'il soit utilisé avec un track ball (une souris à l'envers). Ce dernier est donc l'outil principal du logiciel. Il présente l'avantage sur la souris de ne pas exiger une grande surface de travail, sa boule directrice étant actionnée par les doigts ou la paume de la main. Toutefois, la précision et la vitesse de déplacement qu'il autorise sont moindres. Ce n'est pas vraiment gênant, dans la mesure où le pavé de déplacement du curseur (les touches fléchées du clavier de l'ordinateur) peut se substituer à tout moment à ce track ball : très pratique entre autres pour les déplacements au pixel près.

Trois boutons viennent l'orner :

- un rouge, le plus important, dont le rôle est de valider les étapes successives de la création d'une figure ;
- un orange permet de revenir à tout moment au menu courant ;
- un blanc efface une figure en cours de création.

A l'exécution, *Creative Graphics* présente quatre menus, initialement situés à droite de l'écran, composés d'icônes correspondant aux options disponibles. Les choix peuvent être effectués indifféremment depuis le track ball ou les touches de curseur du clavier. A ce niveau, les deux méthodes sont pratiques, mais la seconde a ma préférence.

L'enfance de l'art

Pour créer ou modifier les dessins, on peut faire appel à un « crayon » (pour dessiner), un « pinceau » (pour peindre) ou à la « main » (pour désigner une figure).

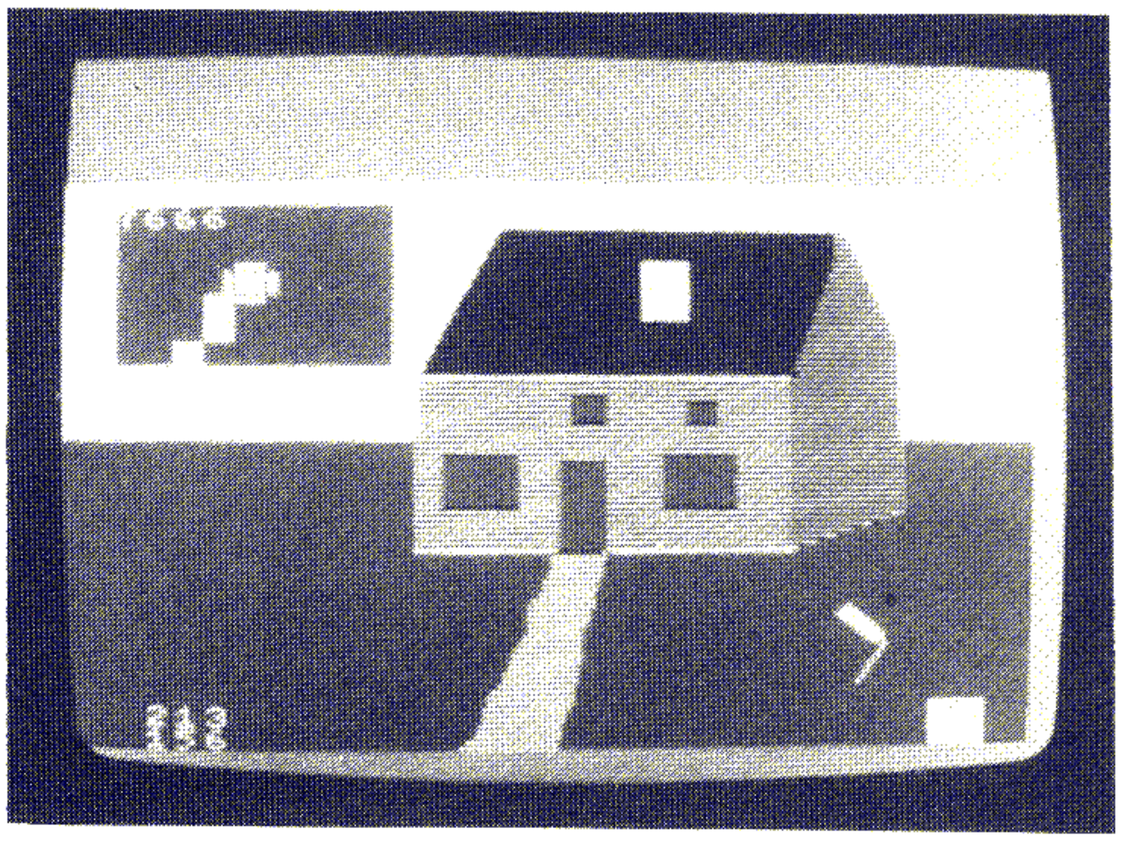
Le *menu dessin* est le premier menu. Il est composé de 22 icônes. Onze d'entre eux concernent le tracé de figures. Par exemple, pour tracer une ligne, on déplace le « curseur » vers l'icône représentant un crayon sur une ligne horizontale. Le crayon apparaît sur le dessin. Il faut alors l'amener sur la première extrémité de la ligne à tracer, appuyer sur le bouton rouge, déplacer le crayon vers la deuxième extrémité, et valider la ligne par une seconde pression sur le bouton rouge. Durant le trajet, la ligne provisoire est matérialisée dans une couleur claire. La deuxième pression sur le

bouton rouge la fixe dans sa couleur définitive, que l'utilisateur aura choisie auparavant.

En utilisant des principes similaires, que l'on devine souvent sans consulter la documentation, il est possible de tracer rectangles (vides ou pleins), cercles, ellipses, arcs de cercles, de peindre (avec le pinceau) l'intérieur d'une figure, ou tout simplement d'utiliser le crayon comme un vrai crayon en dessinant à main levée une figure de son choix. Les autres options du *menu dessin* permettent de redéfinir la couleur des bords ou du fond, de gommer la dernière figure, de sortir le dessin sur imprimante matricielle. Si l'imprimante est monochrome, seules les huit couleurs les plus sombres seront imprimées. Plus original, on peut déplacer le menu où bon nous semble. Enfin, il existe des icônes pour accéder aux trois autres menus ou pour revenir au Basic.

Le second menu, le *menu taille* permet de redéfinir la largeur des traits : quatre épaisseurs sont disponibles, chacune en continu ou en pointillé. On peut aussi choisir la palette de couleurs (16 ou 120 couleurs obtenues par mélanges de lignes) pour le remplissage des figures.

Les quatre icônes suivants offrent des variantes de tracé : tracé d'un segment tous les deux, trois, cinq, ou huit pixels. Dans ce menu, il est aussi prévu de pouvoir « moucheter » le fond en modulant la densité de points. Une loupe, que la documentation nomme poétiquement « œil magique », reste en permanence affichée à l'écran et montre, de manière agrandie, ce qui entoure le crayon. Cet accessoire « optique » facilite les travaux de précision : deux icônes du *menu taille* permettent de le faire disparaître ou réapparaître à volonté. Enfin le dernier icône permet de revenir au *menu dessin*, duquel il est possible d'aller explorer *le menu correction*.



En bas et à gauche, les coordonnées du crayon

Ce dernier est composé, lui aussi, de 22 icônes. Il permet de remonter le temps en faisant clignoter la dernière figure, puis la précédente et ainsi de suite, avant de revenir à la dernière figure composée.

Toutes les retouches sont possibles

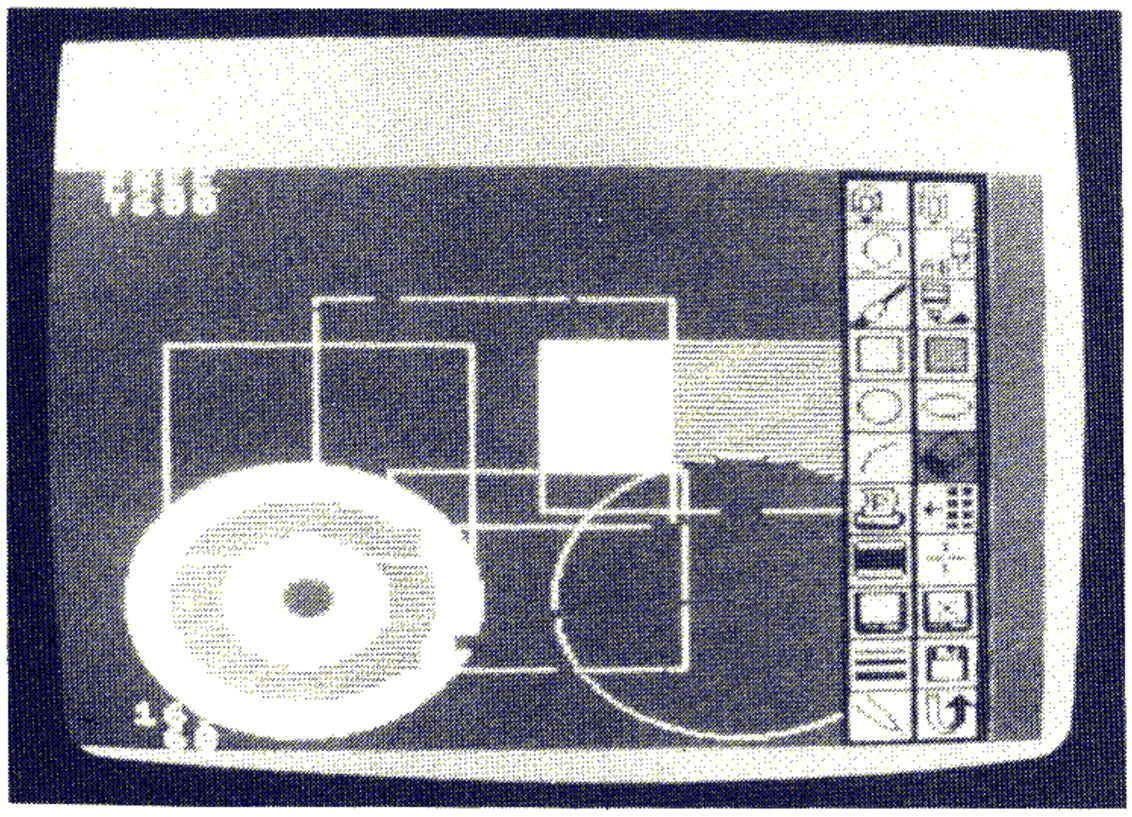
Il est facile alors d'effectuer des modifications sur la figure choisie, de la déplacer, d'en changer la couleur, de la supprimer seule ou avec toutes celles qui la suivent. En choisissant l'icône représentant une main, on obtient un autre moyen de désigner une figure, tout simplement en la montrant du doigt.

Le dernier menu concerne les mémoires de masse. Il se divise en deux parties : l'une pour les disquettes, et l'autre pour les cassettes. C'est grâce à ce menu que le dessin sera lu, sauvegardé avec ou sans la possibilité de retouches ultérieures, par étapes ou globalement, ou même sauvegardé sous forme de programme Basic. Un tel programme est alors réutilisable à l'intérieur d'un autre.

Le logiciel en quelques lignes

Nom : *Creative graphics*
Ordinateurs : *MSX*
Forme : *cartouche de mémoire morte, s'emploie avec un track ball*
Édité et distribué par : *Sony*
Prix public : *990 FF avec le track ball*
Principales orientations : *conception graphique, sauvegarde des dessins sous forme de programmes Basic.*

Sur la droite de l'écran, l'un des menus par idéogrammes



Le système est plus souple avec le lecteur de disquettes, la partie du menu lui correspondant étant plus riche. Elle propose en plus, le choix du lecteur de travail, l'affichage du catalogue, et la destruction d'un fichier.

Creative Graphics est à la fois puissant et simple d'emploi. Malgré les menus et leurs icônes, la documentation est indispensable pour maîtriser totalement ce logiciel en quelques heures à peine. Les profanes, les enfants, les développeurs de logiciels, ou même les artistes devraient pouvoir profiter d'un tel outil de création.

Thierry LÉVY-ABÉGNOLI

AUTEUR

TRAITEMENT DE TEXTE POUR ORIC-1 ET ATMOS

UN peu rustique, certes, mais simple d'emploi et offrant les fonctions indispensables à ce type de logiciel, Auteur est un traitement de texte dont le rapport qualité/prix est excellent.

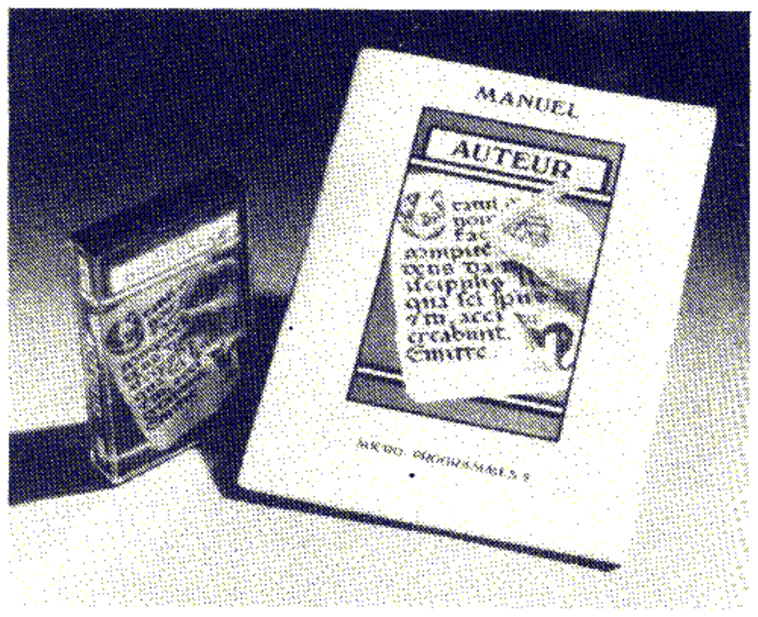
Voici l'un des rares logiciels de traitement de texte digne de ce nom destiné à un ordinateur bon marché. A première vue, on est tenté de se demander à quoi peut bien servir un traitement de texte, c'est-à-dire un logiciel à orientation nettement professionnelle, sur une machine grand public.

En fait, cela sert à une foule de choses. Si le traitement de texte est simple d'emploi, s'il offre le minimum de services que l'on est en droit d'en attendre (et c'est le cas d'*Auteur*), beaucoup de monde en a l'utilisation : mise au propre de notes ou de mémoires pour les étudiants, rédaction de lettres, de C.V., de rapports, de mémos, et plus généralement de tous documents écrits dont la rédaction demande à être soignée. La seule contrainte est qu'il faut disposer d'une imprimante. Toute personne qui passe une bonne partie de son temps à écrire (à la machine ou à la main) peut se simplifier beaucoup la vie avec un traitement de texte.

A l'ouverture du coffret, on découvre un petit manuel (35 pages en version française) accompagnant la cassette. C'est une documentation complète, mais quelques fautes de composition en rendent la lecture parfois un peu difficile : certaines commandes du logiciel sont en effet des signes de ponctuation, et on a parfois du mal à les distinguer de la ponctuation du texte. Cela dit, ce sont des défauts mineurs, et on s'y retrouve tout de même assez bien avec un peu d'attention.

Le chargement ne pose pas de problème, et *Auteur* démarre automatiquement. Après un court instant d'initialisation, le programme affiche son menu principal où l'on découvre 9 options. On fait son choix en tapant seulement la première lettre du nom de l'option, ce qui fait gagner du temps une fois que l'habitude en est prise. Autre bon point : passé ce menu, la plupart des fonctions disponibles fonctionnent de la même façon, ce qui est bien agréable. Le programme, qui est écrit en Assembleur, répond immédiatement à toutes les commandes, même les plus compliquées.

On peut aussi — et c'est l'un des attraits des traitements de texte — se livrer à ce que l'on appelle « la frappe au kilomètre » : on frappe le texte de manière ininterrompue sans avoir à taper RETURN à la fin de chaque ligne.



Le logiciel Auteur et sa documentation, pour écrire toutes sortes de textes sur Oric-1 et Atmos

A l'intérieur d'un même alinéa, le programme se charge d'aller à la ligne sans couper les mots. Avec un tel système (classique pour un traitement de texte sur ordinateur de table), la touche RETURN sert pour forcer le passage à la ligne (point final ou fin de paragraphe), ou pour sauter une ligne. On voit alors à l'écran un caractère spécial en forme de flèche qui indique la position des sauts de ligne, mais qui, bien sûr, ne sera pas imprimé.

Dix options au menu principal

Au départ, le clavier donne les minuscules (et les chiffres). Les caractères majuscules sont obtenus en pressant la touche SHIFT. Enfin, la fonction dite CAPS LOCK (clavier bloqué en majuscules) est donnée par Ctrl-A (qui fonctionne en bascule).

Les options proposées au menu principal sont les suivantes :

Write permet d'écrire un nouveau texte. Cette commande efface l'ancien texte en mémoire, et le logiciel demande une confirmation, sage précaution contre les risques d'effacement accidentel.

Continue reprend l'écriture à la suite d'un texte déjà présent en mémoire, par exemple après un passage en mode éditeur.

Tape speed sélectionne la vitesse de transfert sur bande (mode *Fast* ou *Slow*).

Retrieve charge un texte, soit d'un nom spécifié, soit le premier rencontré sur la cassette. Cette commande agit comme *Write* et demande les mêmes confirmations. Un gadget qui n'a l'air de rien mais qui fait patienter : à mesure que le texte se charge, les caractères lus défilent à toute allure en haut et à droite de l'écran. On peut toujours y poser son regard en attendant la fin du chargement (1000 mots environ en 35 secondes, mode *Fast*).

Append ajoute un texte lu sur cassette à celui qui se trouve en mémoire, sans effacer ce dernier.

Store sauvegarde le texte en mémoire sur la cassette après que l'utilisateur lui ait donné un nom.

Print imprime le texte en mémoire suivant la mise en page définie à l'intérieur du texte par les « commandes-points » et par les paramètres d'impression définis avec la commande suivante.

Install printer initialise les paramètres d'impression (imprimante Oric ou autre, nombre de lignes à sauter en fin ou en début de page, de paragraphe, de texte). On peut aussi y définir cinq macro-instructions d'impression. L'ensemble de cette mise en page est sauvée sur cassette avec le texte.

Z compte le nombre de mots en mémoire, et indique la place restante (en octets).

Edit permet de revenir, pour le modifier, sur le texte en mémoire, et c'est ici bien entendu que l'on aborde ce qui fait l'essentiel d'un traitement de texte. On peut de manière simple :

- déplacer le curseur dans le texte, caractère par caractère, vers la gauche ou vers la droite, passer d'une ligne à l'autre, sauter au début d'une page, au début du texte ;
- faire défiler l'écran ligne par ligne, page par page ;
- insérer du texte à l'endroit du curseur ou recouvrir le texte déjà écrit ;
- supprimer le texte depuis le curseur caractère par caractère, par mots, par phrases, par paragraphes entiers, ou jusqu'au point final ;
- annuler la dernière suppression (ce qui autorise les repentirs) ;
- sauvegarder sur cassette la partie du texte située après le curseur ;
- rechercher ou remplacer toutes les occurrences d'une chaîne de caractères donnée ;
- déplacer ou dupliquer un bloc de texte d'un point à un autre.

Mise en page : le strict nécessaire

Voilà pour l'essentiel. L'ensemble de ce travail se trouve facilité grâce à une ligne où sont résumées les commandes et qui se trouve en permanence sur l'écran.

On peut aussi insérer différentes « choses » dans le courant du texte : elles ne seront pas imprimées, mais elles correspondent à des ordres pour l'impression, ou à d'autres marqueurs spécifiques au logiciel. Ces marqueurs sont de plusieurs ordres. Ils peuvent, entre autres, servir par l'intermédiaire de commandes spéciales à retrouver rapidement tel ou tel passage du texte pour y inscrire, par exemple, un nom et une adresse dans le cas d'une lettre-type.

Les autres signaux que l'on peut insérer dans le texte en cours de traitement sont les « commandes-points ». On les obtient en tapant Ctrl-D, ce qui affiche un point en vidéo inverse que l'on fait suivre d'une lettre-code. On peut de cette façon fixer les marges droite et gauche, la longueur de la page, le nombre de lignes à imprimer, le nombre d'exemplaires désirés, spécifier si l'on utilisera du papier continu ou feuille à feuille, imprimer un en-tête sur chaque page, définir la pagination, centrer une ligne sur la feuille, et encore bien d'autres choses.

Le logiciel en quelques lignes

Nom : *Auteur*
Ordinateur : *Oric-1 et Atmos*
Forme : *cassette avec manuel de 35 pages*
Distributeur : *Micro-Programmes 5*
Prix public : *180 FF*
Principale orientation : *traitement de texte*

On aurait préféré, évidemment, que le texte imprimé, le produit fini, soit « justifié » sur sa marge de droite comme sur celle de gauche. Cette qualité de la présentation n'est pas indispensable (les machines à écrire les plus courantes ne justifient pas à droite), mais elle apporte un petit air d'imprimé très flatteur pour l'œil. L'autre inconvénient, un peu plus ennuyeux, vient non du programme, mais de l'Oric qui ne peut pas fonctionner en mode 80 colonnes. En deux mots, cela signifie que, le plus souvent, la mise en page que l'on voit à l'écran ne correspond pas à celle du document qui sera imprimé. Ou bien alors, il faut fixer les marges de telle sorte qu'il ne reste que 40 caractères par ligne, ce qui, dans certains cas, est un peu court.

Autre critique : malgré les différents réglages possibles entre les commandes-points et *Install printer*, il n'est pas très facile de conserver la synchronisation saut de page/saut de feuille sur plus de quelques pages. Si le document est long, on aura souvent intérêt à se mettre en feuille à feuille. On regrettera aussi que lorsqu'on introduit du texte au clavier, on ne puisse déplacer le curseur que sur la dernière ligne tapée. Toute correction située ailleurs que sur cette dernière ligne contraint à passer en mode d'édition (ce qui replace le curseur au début du texte) et à rechercher alors le passage à corriger.

Somme toute, ces défauts restent véniels comparés aux facilités offertes par ce traitement de texte dont le rapport qualité/prix est excellent. *Auteur* est importé d'Angleterre par *Micro-Programmes 5* et fonctionne aussi bien sur Oric-1 que sur Oric-Atmos.

Pierre BRANDEIS

COMPILATEUR INTÉGRAL POUR SPECTRUM

Une cassette qui peut faire gagner un temps fou

LE Basic du Spectrum n'est pas particulièrement rapide. Pour l'accélérer, il existe des compilateurs.** Compilateur intégral **en est un qui présente l'avantage de traiter les nombres en virgule flottante, et d'avoir la même syntaxe que le Basic du Spectrum.

Compilateur intégral est un logiciel sur cassette, accompagné d'un manuel d'emploi dont trois pages seulement sont utiles, les suivantes comportant la garantie (bien nécessaire vu la qualité d'enregistrement de ce type de logiciels) et un peu de publicité.

Par une brillante astuce, le compilateur laisse toute la mémoire utilisable. Il est chargé pendant la compilation dans la mémoire écran, ceci en un peu plus d'une minute. Malheureusement, il n'est pas possible d'implanter le compilateur sur microdrive, et il faut le charger à partir de la bande à chaque compilation. Cette dernière est très rapide. Elle se termine par l'affichage d'une seule ligne de programme, nantie d'un numéro étrange et signée par Compere (Compilateur Ere Informatique).

On peut néanmoins lancer le programme à partir de n'importe quel numéro de ligne par un GOTO, et non un RUN. Il faut garder une liste du programme-source, justement pour savoir à quelle ligne il doit démarrer.

Le programme une fois compilé peut être sauvegardé avec un numéro de ligne d'autolancement, sur cassette et même sur microdrive.

Avec *Compilateur intégral,* un programme Basic peut pratiquement être compilé sans modification. A deux exceptions près : le dimensionnement des tableaux qui doit être constant et non dépendant d'une autre variable ; les nouvelles commandes et instructions apportées par l'interface ZX1 ne sont pas reconnues par le compilateur, et provoquent une erreur à l'exécution.

Pour tester les performances du compilateur, on a choisi quatre tests (voir en encadré). Les comparaisons avec les résultats obtenus en Basic interprété montrent que le compilateur est très rapide quand il s'agit d'incrémenter des compteurs, mais beaucoup moins performant lorsqu'il traite des fonctions scientifiques (test 3). Dans ce cas, vraisemblablement, il doit faire appel à des routines en mémoire morte, aucun algorithme plus puissant n'ayant été conçu.

Le test 4 montre l'efficacité de ce compilateur face au graphisme. Il est aussi beaucoup plus rapide dans la manipulation des adresses de variables, ou dans les calculs arithmétiques (ceux qui n'utilisent que les quatre opérateurs de base : +, −, *, /).

Ce *Compilateur intégral* est d'un maniement aisé, on peut l'utiliser sans connaissance approfondie de la machine. Sa principale contrainte est l'absence de reconnaissance des fonctions microdrive. Ses performances varient en fonction du type de programme, mais le plus souvent le gain de temps à l'exécution est appréciable.

Benoît THONNART

Tester la rapidité de Compilateur intégral

Test 1
Boucle vide
```
10 FOR I = 1 TO 10000
20 NEXT I
30 END
```
Résultats (en secondes) :
- en Basic, 42 s
- avec le compilateur, moins d'une seconde

Test 2
Sous-programmes
```
10 FOR I = 1 TO 10000
15 GOSUB 100
20 NEXT I
30 END
100 GOTO 110
110 RETURN
```
Résultats :
- en Basic, 106 s
- avec le compilateur, moins de 2 s

Test 3
Calcul scientifique
```
10 FOR I = 1 TO 10000
15 J = SIN (LOG(I))
20 NEXT I
30 END
```
Résultats :
- en Basic, 1 180 s
- avec le compilateur, 1 100 s

Test 4
Graphisme
```
10 FOR Y = 175 TO 0 STEP −1
20 FOR X = 0 TO 255
30 PLOT X, Y
40 NEXT X
50 NEXT Y
```
Résultats :
- en Basic, 360 s
- avec le compilateur, 65 s

Le logiciel en quelques lignes

***Nom :** Compilateur intégral*
***Ordinateur :** ZX Spectrum*
***Forme :** cassette*
***Édité et distribué par :** Ère Informatique*
***Prix public :** 250 FF*
***Principale orientation :** compiler le Basic*

ÊTES-VOUS UN PROGRAMMEUR SI COMPLIQUÉ ?

ÊTES-vous un bon programmeur, utilisez-vous des algorithmes simples, et écrivez-vous des programmes structurés ? L'article qui suit vous fournit les éléments nécessaires à l'écriture de deux programmes qui mesureront, sans discussion, d'une part la complexité, et d'autre part le niveau de structuration de vos programmes.

La solution d'un problème doit être bien analysée et quantifiée avant d'être transcrite sous la forme d'un logiciel. En revanche, les programmes eux-mêmes échappent en général à cette règle. Comment, en effet, quantifier, c'est-à-dire mesurer la complexité d'un programme? Nous allons passer en revue trois méthodes permettant de chiffrer objectivement cette complexité.

Mesurer la complexité

Pour un problème donné, il existe pratiquement une infinité de manières de coder sa solution sous forme d'un programme. Cela est vrai même si les logiciels qui en résultent ont un comportement strictement identique. Ainsi, deux programmeurs fourniront toujours deux programmes différents pour la résolution d'un même problème.

Si l'on tente d'analyser la nature de ces différences, on constate qu'elles sont de deux ordres. La première est liée à la complexité des algorithmes mis en œuvre, et la seconde est liée à la qualité de la programmation.

La première approche que nous allons évoquer est appelée mesure de Halstead. Elle constitue un instrument permettant d'apprécier la complexité des méthodes utilisées dans un programme. Elle repose donc sur la définition d'un algorithme qui est une succession d'opérateurs et d'opérandes. Pour évaluer la complexité d'un programme avec la mesure de Halstead, il est nécessaire de rechercher quatre valeurs liées au programme :

- n_1 est le nombre d'opérateurs distincts qui apparaissent dans le programme ;
- n_2 est le nombre d'opérandes distincts qui apparaissent dans le programme ;
- N_1 est le nombre total d'opérateurs qui apparaissent dans le programme ;
- N_2, enfin, est le nombre total d'opérandes qui apparaissent dans le programme.

ÊTES-VOUS UN PROGRAMMEUR SI COMPLIQUÉ ?

Partant de ces quatre valeurs, on détermine facilement ce que l'on appelle le *vocabulaire* du programme. Il vaut n, où $n = n_1 + n_2$. Autrement dit, il est la somme du nombre d'opérateurs et du nombre d'opérandes distincts.

De la même façon, il est possible de déterminer la longueur N du programme : $N = N_1 + N_2$, soit le nombre total d'opérateurs et d'opérandes présents dans le programme.

Par ailleurs, Halstead a déterminé qu'il existait une relation entre N, n_1 et n_2. Plus précisément vocabulaire et longueur d'un programme sont liés par la relation $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$ où $\log_2$ est le logarithme binaire (base 2). On peut ainsi calculer une longueur approchée du programme grâce à la méthode de Halstead sans avoir à rechercher les valeurs de N_1 et de N_2.

Enfin, il est possible de déterminer, toujours selon Halstead, le volume du programme qui est le nombre de bits nécessaires pour décrire le programme. Le volume V d'un programme est égal à $V = N \log_2 n$. Ainsi, la valeur V donne une mesure de la complexité d'un programme.

Toutefois, la prudence s'impose si cette mesure doit servir à effectuer des comparaisons. Appliquée brutalement, sans pondération, elle n'est valable que si on l'applique à un même type de langage de programmation. En effet, si l'on utilise un langage de programmation évolué tel le Pascal, le nombre d'opérateurs et d'opérandes est beaucoup plus faible que si le programme est écrit en langage Assembleur. Il faut donc pondérer le résultat par ce que Halstead appelle le *niveau d'abstraction*. Ce niveau d'abstraction peut être évalué par la relation $L = (2/n_1) \times (n_2/N_2)$. Alors, la complexité E d'un programme est égale à $E = V / L$.

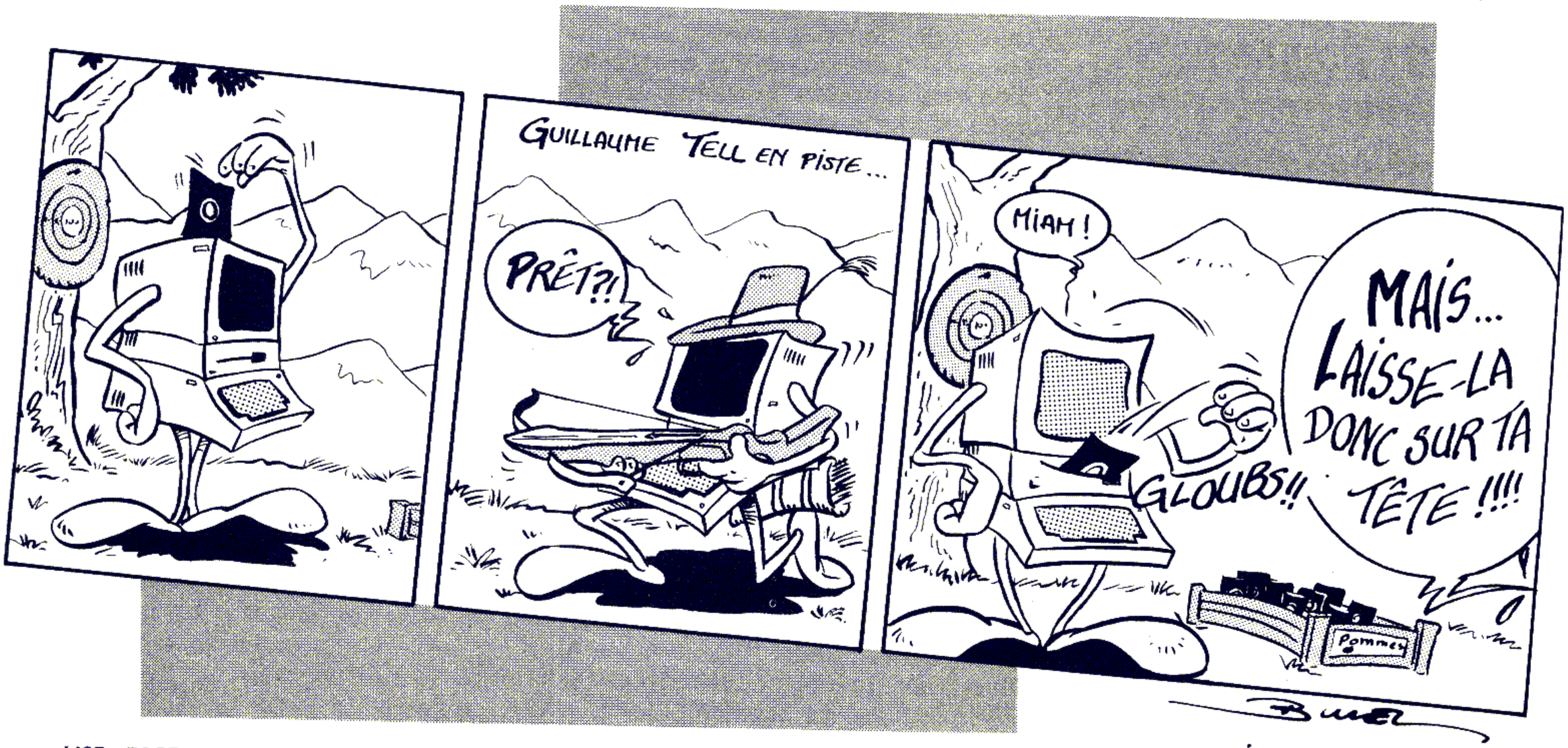
Après la complexité, la qualité

Par conséquent, pondérée par le niveau d'abstraction du langage de programmation utilisé, la méthode de Halstead fournit un moyen efficace de mesurer la complexité des algorithmes utilisés dans un programme.

Deuxième méthode que nous aborderons ici, la mesure du McCabe n'évalue pas la complexité des algorithmes utilisés dans un programme (comme le fait la méthode de Halstead), mais elle évalue plutôt la qualité de la programmation. Au lieu d'être basée sur la notion d'algorithmes, elle repose sur la notion de graphe. Un programme peut toujours être schématisé sous la forme d'un graphe où chaque instruction est représentée par un nœud. Ces différents nœuds sont reliés entre eux par des flèches qui indiquent le cheminement suivi lors de l'exécution du programme.

La mesure de McCabe consiste à évaluer le nombre de nœuds qui sont des prédicats, autrement dit le nombre de situations où une décision doit être prise dans le programme. En fait, cela correspond approximativement au nombre de branchements (GOTO) qui y sont présents. Toutefois, il faut noter que si une instruction telle que IF C1 THEN compte pour un point de décision, une instruction comme IF C1 OR C2 THEN compte pour deux points. La mesure de la complexité d'un programme au sens de McCabe est donc égale à V(G) = P + 1 où P est le nombre de points de décision présents dans le programme.

Ainsi, comme nous le constatons, la méthode de Halstead évalue la complexité d'un programme d'après les algorithmes qui sont mis en jeu : pour deux programmes donnant les mêmes résultats mais écrits par des personnes distinctes, elle conduira à deux valeurs différentes si l'une des deux personnes a mis en œuvre des algorithmes plus compliqués que l'autre. Par contre, la



méthode de McCabe n'est pas liée à la complexité des algorithmes utilisés, mais elle évalue plutôt la qualité de la programmation. Alors, la complexité de deux programmes réalisant le même traitement apparaîtra très différente si l'un est structuré et si l'autre, ne l'étant pas du tout, possède un grand nombre de branchements.

Et la longueur des programmes

On peut encore mesurer et comparer un troisième type de complexité des logiciels. Cette dernière approche consiste tout simplement à mesurer la longueur des programmes, soit en nombre d'instructions si l'on utilise un langage interprété, soit en évaluant la longueur du code si l'on utilise un compilateur.

Cette méthode présente deux avantages. Tout d'abord, elle est simple à mettre en œuvre puisqu'il est très aisé de déterminer le nombre de lignes ou la longueur du fichier-code d'un programme. Ensuite, elle évalue la complexité globale d'un programme. En effet, elle ne s'attache pas, comme la méthode de Halstead, à la complexité des algorithmes ou, comme la méthode de McCabe, au nombre de décisions présentes dans le programme. Cela dit, si cette méthode est plus générale, elle est également beaucoup moins fiable.

On se rendra compte de ce peu de fiabilité à l'aide de trois remarques. La première concerne le niveau du langage. Un programme tout d'abord écrit en langage évolué, puis compilé, paraîtra, avec cette méthode, plus complexe que le même programme écrit en Assembleur. Et cela tout simplement parce que sa longueur sera plus importante. Chacun sait pourtant qu'il est beaucoup plus facile de lire un programme écrit dans un langage de haut niveau qu'un programme écrit en langage-machine.

La deuxième remarque s'applique également aux programmes écrits dans un langage compilé. Un logiciel utilisant beaucoup de texte paraîtra ici encore nettement plus compliqué qu'un logiciel mettant en œuvre des recherches très sophistiquées dans un fichier. La raison en est tout simplement que les chaînes de caractères se retrouvent dans le code généré par le compilateur et augmentent donc sa longueur. Le programme n'en devient pas plus complexe pour autant.

La troisième et dernière remarque concerne les programmes écrits dans un langage interprété. Ici, l'on mesure en évaluant le nombre de lignes ou le nombre de caractères dont est composé le programme. Mais dans ce cas aussi, le caractère significatif de la mesure est affecté par une chose aussi simple que les commentaires, le texte des remarques... Plus les commentaires seront abondants, plus ils contribueront à exagérer l'apparente complexité du programme. Or la présence de commentaires est au contraire un moyen de simplifier au moins la lecture d'un logiciel.

En résumé, on constate que la recherche de la longueur du code ou du nombre d'instructions qui composent un programme permet d'obtenir une bonne estimation de sa complexité globale. En revanche, si l'on s'attache plutôt à la complexité interne d'un programme, c'est-à-dire au niveau de compréhension des algorithmes utilisés, la mesure de Halstead est indiscutable.

Enfin, si l'on recherche la complexité externe d'un logiciel, son niveau de structuration, la mesure de McCabe permet de noter avec précision la qualité d'un programme.

Sans penser à un concours du programme le plus simple et le mieux structuré, vous pouvez vous entraîner à un excellent exercice afin de progresser dans l'écriture de bons programmes : en partant d'un programme que vous avez déjà écrit, essayez de le réécrire de façon à ce qu'il obtienne de meilleures notes aux tests de Halstead et de McCabe. Pour cela, il vous faudra trouver de meilleurs algorithmes, plus simples, mais tout autant performants, et améliorer la structuration du programme en réduisant le nombre de branchements (GOTO) et de tests (IF). Maintenant que vous avez le juge-arbitre, c'est à vous de jouer.

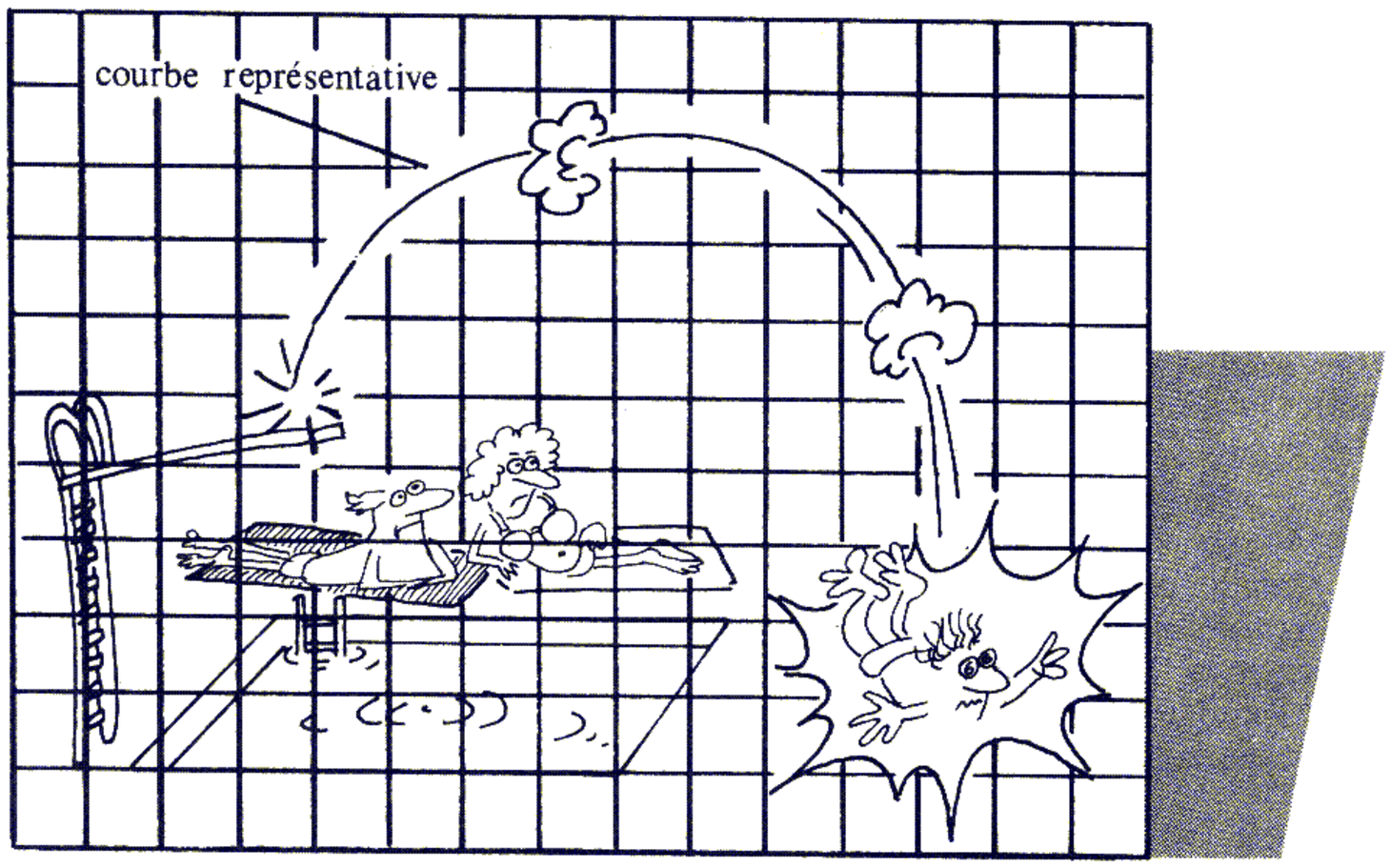
Thierry CHAMORET

LA COURBE ET SA DÉVELOPPÉE

L'ÉTUDE des fonctions mathématiques passe le plus souvent par celle des courbes représentatives. Mais sous le terme « courbe », se cachent des formes différentes, certaines étant plates, d'autres plus bombées. Tout dépend du rayon de courbure, du centre de courbure, et de la développée. C'est pourquoi il est intéressant de mieux connaître cette dernière et de la tracer.

Savez-vous ce qu'est la développée d'une courbe ? Peut-être pas. Eh bien, ceci est lié très simplement à la notion de *courbure* d'une courbe en l'un de ses points. Savoir qu'elle passe par M, de coordonnées *x* et *y*, est évidemment très imprécis ; connaître la tangente en ce point est déjà beaucoup mieux. Mais il est aussi très important de savoir comment la courbe « touche » cette tangente, de quel côté elle se situe, si elle arrive très vite (et part de même), bref quel est son comportement au voisinage du point considéré.

Le fin du fin consiste à pouvoir approcher la courbe étudiée, non seulement par une droite — c'est-à-dire pouvoir tracer une tangente — mais par un arc de cercle. Plus le rayon de ce cercle est petit, plus la « courbure » est grande : l'approximation par un petit segment de droite est bien médiocre. Au contraire, si le rayon (appelé, comme vous vous en doutez bien, *rayon de courbure*) est très grand, c'est que la courbe se prélasse, traîne en quelque sorte, sans grande envie de virages vertigineux.



Un cercle bien précieux

Connaître ce cercle (dit, lui aussi, de courbure) est donc bien précieux. Il suffit d'avoir son centre puisqu'on en a déjà un point. L'ensemble des centres de courbure est ce que l'on appelle la *développée* de la courbe.

Le PC-1500, avec sa table traçante, permet d'obtenir une très grande variété de graphiques. Pratiquement, une paire de fonctions numériques *f(t)* et *g(t)* définit une courbe que notre imprimante nous livre bien volontiers, à condition d'être aidée par un programme de tracé.

Ce que nous avons voulu faire ici, c'est, à partir de *f* et de *g*, ainsi que de limitations sur les valeurs de l'abscisse *x* et de l'ordonnée *y* (le papier ne va pas très loin avec ses 5 cm de large), définir un logiciel qui imprime (en deux couleurs distinctes, sur le PC-1500), d'abord la courbe elle-même, ensuite sa développée.

Évidemment, ce programme n'est pas parfait. Je vous le propose simplement comme point de départ. A vous de faire mieux ; ce n'est sans doute pas très difficile.

Expliquons d'abord les notations. L'abscisse *x* est comprise entre A et B ; l'ordonnée *y* varie, elle, entre C et D ; le programme demande ensuite les limites F et G du paramètre *t* qui sert à définir la courbe (peu importe ici que la let-

LIST - PAGE 54

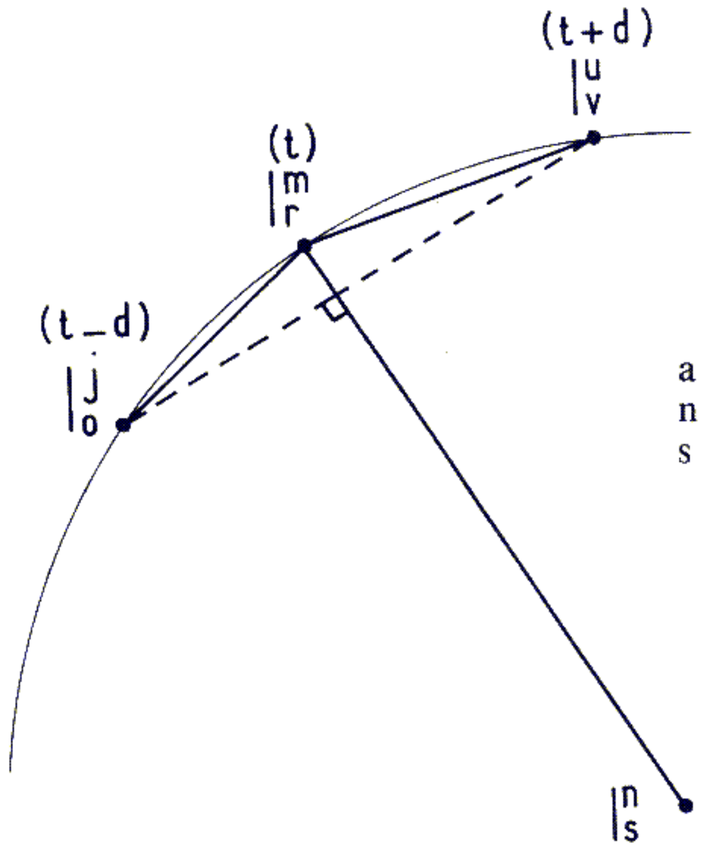
Développées
Programme pour PC-1500
Auteur André Warusfel


```
5:REM COURBE ET
  DEVELOPPEE
10:"A"INPUT "A=";
  A,"B=";B,"C=";
  C,"D=";D,"E=";
  E,"F=";F:W=1:G
  =(F-E)/200
20:K=200/(B-A):P=
  K*A:Q=K*D
30:GRAPH :
  GLCURSOR (-P,-
  Q):SORGN :
  COLOR 3
40:FOR I=0TO 200:
  T=E+I*G:GOSUB
  "B
50:IF Z=1LET W=1:
  GOTO 80
60:IF W=1LET W=0:
  GLCURSOR (N,S)
  :GOTO 80
70:LINE -(N,S)
80:NEXT I:COLOR 1
  :W=1:L=0
90:FOR I=0TO 200:
  T=E+I*G:J=M:M=
  N:O=R:R=S:
  GOSUB "B
110:IF Z=1LET W=1:
  GOTO 190
120:IF L<2GOTO 190
130:F=J*(R-S)+M*(S
  -O)+N*(O-R):IF
  ABS F<1E-4LET
  W=1:GOTO 190
140:J=N-J:O=S-O:L=
  J*J+O*O:F=L/(8
  *F)
150:X=M-F*O:Y=R+F*
  J
160:IF (X<(K*A))+(
  X>(K*B))+(Y<(K
  *C))+(Y>(K*D))
  >0LET W=1:GOTO
  190
170:IF W=1LET W=0:
  GLCURSOR (X,Y)
  :GOTO 190
180:LINE -(X,Y)
190:L=L+1:NEXT I:
  END
200:"B"REM CALCUL
  DE U=F(T) ET D
  E V=G(T)
210:REM SI NON CAL
  CULABLES, LET
  Z=1:L=-1:GOTO
  300
220:Z=0:REM SINON.

280:IF (U<A)+(U>B)
  +(V<C)+(V>D)>0
  LET Z=1:L=-1
290:N=K*U:S=K*V
300:RETURN
```

Quelques précisions mathématiques sur le programme

Pour ceux que n'effraie pas un peu de mathématiques, levons un coin de voile sur la détermination de n et de s, valeurs approchées des coordonnées du centre de courbure associé au point (u(t), v(t)) de la courbe (voir schéma ci-dessous). Le principe consiste à prendre trois valeurs successives du paramètre t, engendrant les points successifs notés ici (j,o), (m,r) et (u,v), et à trouver une valeur approchée des coordonnées (n,s) du centre du cercle circonscrit au petit triangle qu'ils forment (on constate facilement que ce dernier est presque isocèle, et possède un angle au sommet pratiquement plat). En fait, cela revient à tracer, à partir de ce sommet, la perpendiculaire au côté opposé (autrement dit la hauteur) et à y porter, dans la concavité de la courbe, une longueur approximativement égale au rayon de courbure ϱ en ce point. La formule exacte donnant ϱ est bien connue. C'est : $\varrho = \alpha^3/\beta$, avec $\alpha^2 = (x')^2 + (y')^2$ et $\beta = x'y'' - y'x''$.



$a = (u-j)^2 + (v-o)^2 = 4\,d^2\,\alpha^2$
$n = m - (a/16S)\,(v-o)$
$s = r + (a/16S)\,(u-j)$

Pour calculer les dérivées x', y', x'' et y'', on utilise des développements limités classiques ; comme j, m et u sont par exemple des valeurs successives de l'abscisse x correspondant à trois valeurs du paramètre t deux à deux distantes d'une même quantité d, on a, approximativement, $2\,dx' = u - j$. Ensuite le calcul est plus délicat ; on emploie ici une « astuce » pour obtenir en fait d'un seul coup le nombre ß considéré comme la mesure du produit vectoriel des deux vecteurs (x', y', 0) et (x'', y'', 0) donc (à un coefficient près) égal à l'aire S du petit triangle considéré. Un développement un peu technique montre que l'on a (toujours de manière approchée) : $8\,\beta d^3 = 8\,d^3(x'y'' - x''y') = 16\,S$.

Et l'aire S est elle-même obtenue très classiquement par le développement d'un déterminant d'ordre 3 à partir des six nombres (j,m,u,o,r,v) respectivement égaux aux coordonnées des trois sommets du triangle.

tre f ait deux significations distinctes ; aucune confusion n'est possible). Après avoir placé entre les lignes 200 et 279 sur le PC-1500, 300 et 389 sur les autres Basic (voir le programme page suivante), les procédures de calcul de $x = f(t)$ et $y = g(t)$, lancez le programme. Il commence par vous demander de rentrer les six constantes fondamentales A, B, C, D, E et F, puis il amorce le tracé de la courbe proprement dite. Si c'est la seule chose qui vous intéresse, alors remplacez la ligne 80 par 80:NEXT I:END, et supprimez les instructions des lignes 90 à 190, sur le PC-1500.

Si vous désirez admirer la développée elle-même, il vous faudra un tout petit peu plus de patience. Par quelques diableries mathématiques que vous découvrirez bien tout seul si cela vous inté-

LA COURBE ET SA DÉVELOPPÉE

Développées
Programme en Basic graphique
Auteur André Warusfel


```
10 INPUT "A=";A
20 INPUT "B=";B
30 INPUT "C=";C
40 INPUT "D=";D
50 INPUT "E=";E
60 INPUT "F=";F
70 G=(F-E)/639:K=639/(B-A):H=229/(B-A):W=1
80 P=K*A:Q=H*D:IF(D-C)>1.04*(B-A)PRINT"STOP":END
90 CLR:LINE(-P,0)-(-P,239),,,&HCCCC
100 LINE(0,Q)-(639,Q),,,&H0F0F
110 SCREEN0:LPRINT CHR$(18)
120 FOR I=0 TO 639:T=E+I*G:GOSUB 300
130 IF Z=1 LET W=1:GOTO 160
140 IF W=1 LET W=0:X=K*U-P:Y=Q-H*V:PSET(X,Y):GOTO 160
150 X=K*U-P:Y=Q-H*V:LINE-(X,Y)
160 NEXT:W=1:L=0
170 FOR I=0 TO 639:T=E+I*G:J=M:M=U:O=R:R=V:GOSUB 300
180 IF Z=1 LET W=1:GOTO 260
190 IF L<2 GOTO 260
200 F=J*(R-V)+M*(V-O)+U*(O-R):IF ABS(F)<1E-7 LET W=1:GOTO 260
210 J=U-J:O=V-O:F=(J*J+O*O)/(8*F)
220 N=M-F*O:S=R+F*J
230 IF(N<A)+(N>B)+(S<C)+(S>D)<0 LET W=1:GOTO 260
240 IF W=1 LET W=0:X=K*N-P:Y=Q-H*S:PSET(X,Y):GOTO 260
250 X=K*N-P:Y=Q-H*S:LINE-(X,Y)
260 L=L+1:NEXT:END
300 Z=0:REM CALCULER U=F(T) ET V=G(T)
390 IF (U<A)+(U>B)+(V<C)+(V>D)<0 LET Z=1:L=-1
400 RETURN
```

A propos du programme en Basic graphique

Les lignes 90 et 100 tracent à l'écran les parties des axes de coordonnées figurant à l'intérieur du cadre défini par les inégalités $A \leqslant x \leqslant B$ et $C \leqslant y \leqslant D$.

Les lignes 120 à 160 tracent à l'écran la courbe définie par les équations paramétriques $u = f(t)$, $v = g(t)$ pour $E \leqslant t \leqslant F$.

Les lignes 170 à 260 tracent à l'écran la développée de la courbe précédente. Pour cela, on détermine des valeurs approchées n(t) et s(t) des coordonnées du centre de courbure associé au point (u(t), v(t)) de la courbe.

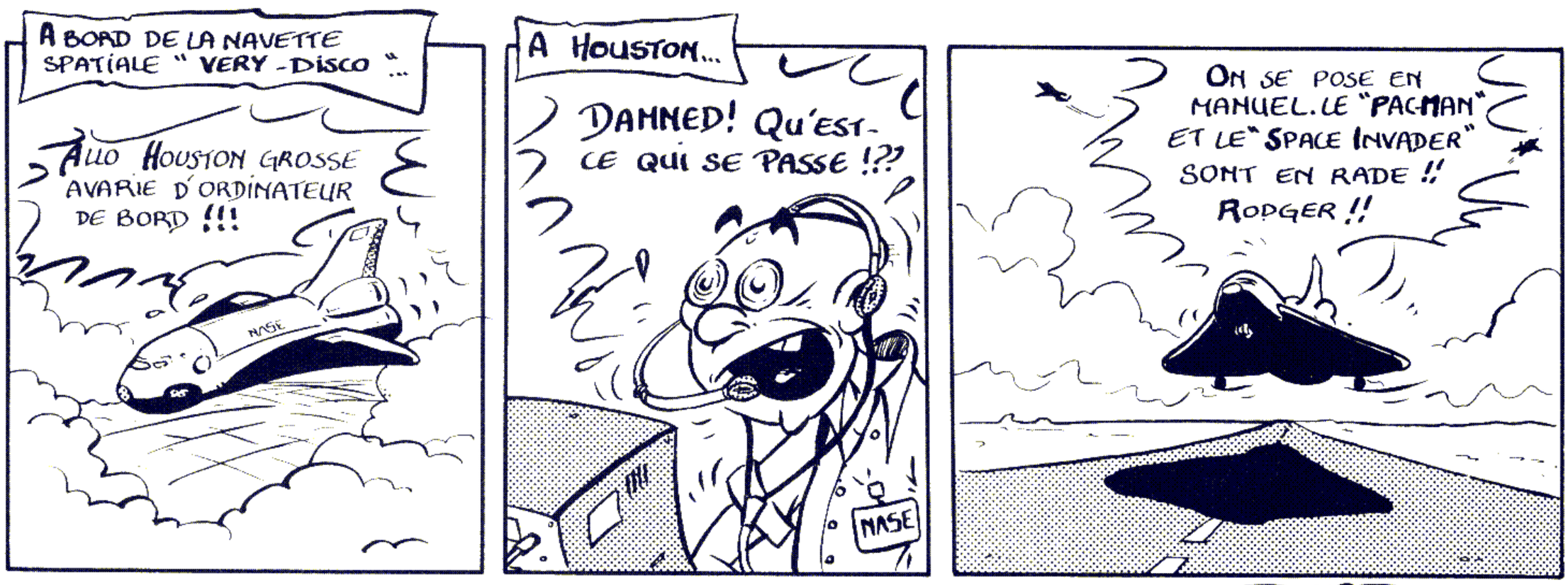
Le sous-programme des lignes 300 à 400 calcule les coordonnées (u(t), v(t)) du point courant de la courbe étudiée.

resse, votre ordinateur montera sur ses grands chevaux, et fournira un graphique plutôt correct dans les cas usuels.

Quelques précisions mathématiques sur le programme sont tout de même données dans l'encadré de la page précédente.

Il faut noter, pour terminer, que le test de la ligne 130 (ligne 200 pour le programme en Basic graphique) compare une certaine variable F — troisième occurrence, sans rapport avec les précédentes ! — au réel 1/10 000 (pour le PC-1500, 1E-7 pour les autres Basic). Si le tracé vous paraît incomplet, essayez une valeur de comparaison plus faible encore ; mais cela peut nuire à l'exactitude des calculs.

André WARUSFEL



MISEZ P'TIT : OPTIMISEZ

L'INFLEXION, AU PLUS JUSTE...

Si jongler avec la pile opérationnelle de votre HP-41 C, traquer la milliseconde perdue et rogner le moindre octet est votre pain quotidien... Ou si, à l'inverse, vous échappe parfois un peu de la subtile recherche des programmes en Notation Polonaise Inverse...
Alors voici qui doit vous intéresser. En matière de programmation, est-on jamais certain d'avoir fait aussi bien que possible ? Dans cette rubrique, les défis — vos défis — se succèdent : des programmes toujours plus courts, plus rapides... Et les records vivent !

LIST n° 9 lançait le défi. *LIST* n° 10 en apportait toutes les solutions algorithmiques : trouver l'extremum d'une équation du second degré et préciser s'il s'agit d'un maximum ou d'un minimum.

Ce sont Antoine Joux et Franck Wettstein qui l'emportent avec chacun 9 octets sur 6 pas de programme (routines JOUX et WETT) : les coefficients de l'équation du second degré sont introduits dans l'ordre c, b, et a mais il suffit d'ajouter X< >Z en première instruction pour retrouver l'ordre "naturel" a, b, c.

Quant à l'équation du troisième degré, c'est Antoine Terlinden qui est parvenu à optimiser au mieux la routine de détermination du point d'inflexion.

Algorithmiquement parlant, c'est au point d'inflexion que la dérivée seconde de l'équation s'annule (voir *LIST* n° 10). L'équation s'exprimant $aX^3 + bX^2 + cX + d$, ses dérivées premières et secondes sont :
$f'(X) = 3aX^2 + 2bX + c$
$f''(X) = 6aX + 2b$

Cette dernière s'annule quand $6aX + 2b = 0$. Soit, lorsque $X = -2b/6a$ ou encore $X = -b/3a$.

La valeur de Y correspondante est donc (en reportant dans l'équation de départ, et après simplifications) :
$Y = 2b^3/27a^2 - bc/3a + d$.

C'est cette expression qui est calculée par le programme TERL, en seulement 23 octets.

Les coefficients a, b, c et d sont introduits dans l'ordre et, au retour, les coordonnées X et Y du point d'inflexion sont dans la pile opérationnelle.

Jean-Christophe KRUST

```
01♦LBL "JOU
X"
02 SF 00
03 X>0?
04 CF 00
05 ST+ X
06 /
07 CHS
08 END
```

```
01♦LBL "WET
T"
02 CF 00
03 X<=0?
04 SF 00
05 ST+ X
06 /
07 CHS
08 END
```

```
01♦LBL "TER
L"
02 X<> Z
03 R↑
04 /
05 3
06 ST/ Y
07 CLX
08 LASTX
09 ST+ X
10 X<>Y
11 CHS
12 ST* Y
13 *
14 LASTX
15 RDN
16 -
17 R↑
18 *
19 +
20 X<>Y
21 END
```

FONCTIONS HYPERBOLIQUES (suite)

Le programme Fonctions hyperboliques publié dans *LIST* n° 9 emploie une instruction du module X-FONCTIONS. Mais si l'on remplace ANUM par RCLN (fonction synthétique), tous les XEQ03 par STON et si l'on supprime les pas 87 à 90 (inclus), le programme fonctionne sur toute HP-41. Gain de la modification : 4 pas.

Raoul HATTERER

LES JEUX ET CASSE-TÊTE INFORMATIQUES

de Thierry CHAMORET

LES jeux et casse-tête qui vous sont proposés dans cette rubrique ont plusieurs aspects. Tout d'abord, ils peuvent être pris sous l'angle ludique, c'est-à-dire qu'il s'agit de jeux, de petits problèmes plus ou moins faciles à résoudre. Ils ont également un aspect pratique. Ils permettent en effet à chacun d'exercer son agilité logique. Et il n'est pas nécessaire, pour trouver la solution, d'avoir un ordinateur sous la main...

34

Rotations

Un tableau unidimensionnel de longueur L, c'est-à-dire disposant de L cases, peut être représenté de la manière suivante :

1	2	3	.....	L – 1	L

Une rotation de pas 1 consiste à décaler de *une* case tous les éléments du tableau pour donner :

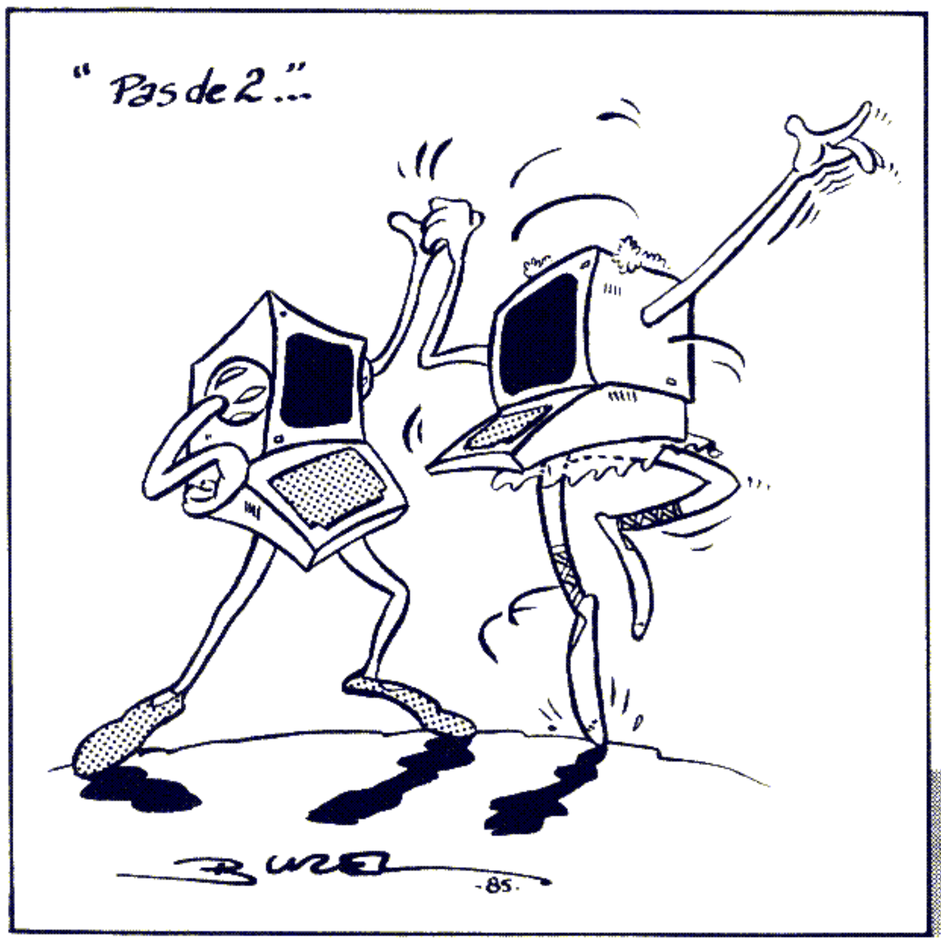
L	1	2	.....	L – 2	L – 1

De même, une rotation de pas 3 décale tous les éléments de 3 cases :

L – 2	L – 1	L	.....	L – 4	L – 3

Ecrire un sous-programme chargé d'effectuer une rotation de pas P dans un tableau T de longueur L, le premier indice du tableau étant égal à 1.

Il faut veiller à ce que ce sous-programme tourne pour toutes les valeurs entières de P, et soit le plus rapide possible.



35

Le langage du programmeur

Un programmeur un peu badin a, dans un programme, une routine baptisée GIRL qui est appelée fréquemment. Dans quel langage ce programmeur écrit-il ?

**Les solutions des jeux proposés ici paraîtront dans le prochain numéro de* LIST.

36

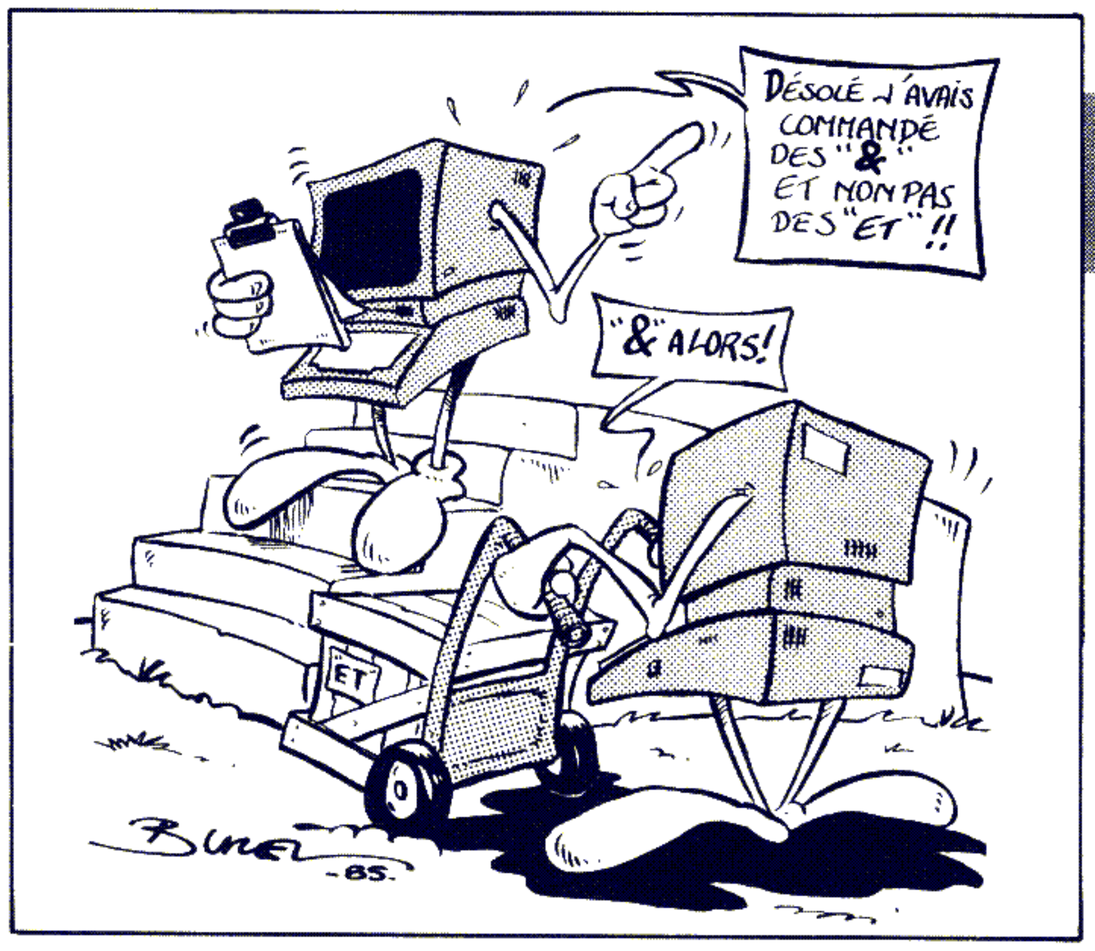
Opérations prioritaires

Soient trois variables entières A, B, C, et un ordinateur effectuant les opérations entières sur 16 bits. Trouver des valeurs pour A, B et C telles que l'égalité : A*(B/C) = (A*B)/C soit fausse.

37

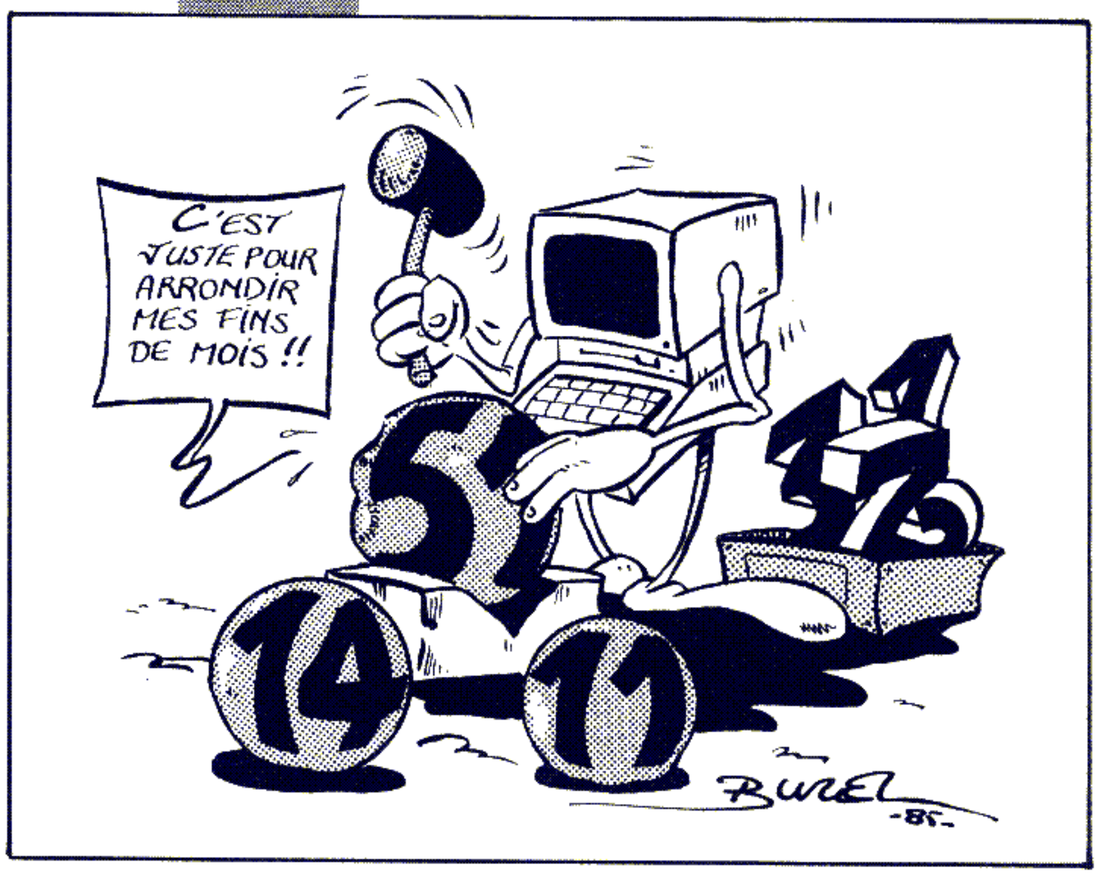
Dans un fichier d'adresses

En analysant de près le contenu d'un fichier d'adresses, on constate que la chaîne de caractères « M. et Mme » apparaît avec une fréquence assez élevée. Afin de réduire la taille d'un tel fichier, un programmeur décide de remplacer toutes les occurrences de « et » par le symbole « & ». Cette méthode permet d'économiser de la place en mémoire et elle ne demande pas de calcul supplémentaire lors de l'édition des adresses, la chaîne « M. & Mme » pouvant parfaitement apparaître dans une adresse. Qu'y a-t-il de faux dans ce raisonnement ?



38

Arrondir des entiers



La fonction ARRONDI de certains Basic arrondit le nombre réel auquel elle est appliquée. Si elle n'existe pas, on peut la remplacer facilement : il suffit d'ajouter 0.5 au nombre et d'en prendre la partie entière. Ainsi, on peut écrire :

ARRONDI(X) = INT(X + 0.5)

Cette méthode fournit des résultats « acceptables ». Il faut savoir qu'il existe d'autres méthodes. En particulier, pour les calculs bancaires : l'arrondi s'attache non pas à avoir de bons résultats sur des nombres pris individuellement, mais sur de grandes séries de valeurs.

Les nombres réels ne sont pas les seuls concernés par l'arrondi. Il est parfois nécessaire d'arrondir des nombres entiers. Par exemple, la valeur 18392 arrondie sur deux chiffres fournit la valeur 18400. Une fonction permettant une telle transformation n'est généralement pas disponible. Il faut donc la programmer. Elle ne devra s'appliquer qu'à des entiers, sans convertir le nombre en réel, la précision des entiers pouvant être supérieure à celle des réels.

Le tableau ci-contre donne les résultats d'une telle fonction appliquée à la valeur 18392, selon le nombre de chiffres sur lequel l'arrondi est effectué.

Résultats de l'arrondi de 18392

Nombre de chiffres	Valeur retournée
0	18 392
1	18 390
2	18 400
3	18 000
4	20 000
5	0

SOLUTIONS DU NUMÉRO PRÉCÉDENT

Les solutions présentées ici répondent aux problèmes posés dans le précédent numéro de LIST. Ce ne sont pas forcément les meilleures !

31

Les fractions en mémoire

1. Des fractions qui ont un développement fini en décimal et en binaire sont, par exemple : 1/2, 1/4, 1/8, 1/16, 1/32 ... En effet, la fraction 1/2 en décimal est égale à 0,5. Et en binaire, cette fraction s'écrit 1/10 et vaut 0,1.

2. Les fractions 1/5, 1/10, 1/20, 1/25, 1/40, etc. ont un développement fini en décimal et infini en binaire. Par exemple, la fraction décimale 1/5 vaut 0,2 alors qu'en binaire, elle s'écrit 1/101 et vaut 0,0011001100110011001100...

3. Les fractions qui admettent un développement infini à la fois en décimal et en binaire sont les plus nombreuses. Par exemple, 1/3, 1/6, 1/7, 1/9, 1/11 répondent à la question. Ainsi, 1/3 en décimal vaut 0,33333333... Ce qui donne en binaire : 1/11 qui vaut 0,010101010...

4. Il n'existe pas de fraction telle que son développement décimal soit infini et son développement binaire soit fini : l'équivalent décimal d'une fraction binaire qui tombe juste tombe également juste et se termine par un 5. Peut-être pourriez-vous faire une démonstration rigoureuse de ce résultat ?

Question subsidiaire : quelle est la proportion relative des fractions entrant dans la première, la deuxième et la troisième catégorie ?

32

Néologismes

Cinq mots couramment employés aujourd'hui sont en fait des néologismes.

• Ordinateur : créé en 1956 par Jacques Perret à la demande d'IBM.

• Informatique : construit à partir des mots information et automatique, ce terme a été introduit dans la langue française par Ph. Dreyfus en 1962.

• Progiciel : créé par J.E. Forges en 1962 à partir des mots produit et logiciel. Il s'agit d'un ensemble complet et documenté de programmes conçu pour être fourni à plusieurs utilisateurs, en vue d'une même application ou d'une même fonction.

• Télématique : inventé vers 1978 par les Français Simon Nora et Alain Minc. Ce terme regroupe l'ensemble des services de nature ou d'origine informatique pouvant être fournis à travers un réseau de télécommunications.

• Bureautique : inventé en 1979 par Louis Naugès. Avant que ce mot ne soit approuvé par le Journal Officiel, la société de Louis Naugès poursuivait quiconque l'utilisait. Il désigne l'ensemble des techniques et des moyens tendant à automatiser les activités de bureau et principalement le traitement de la communication de la parole, de l'écrit et de l'image.

33

5 et 5 ne font pas toujours 10

Les deux instructions A = 5*X + 5*X et A = 10*X ne sont pas équivalentes lorsque X représente une fonction qui ne retourne pas toujours la même valeur. C'est le cas en particulier de la fonction RND qui renvoie un nombre aléatoire. Il est très facile de vérifier que A = 5*RND + 5*RND n'est pas équivalent à A = 10*RND.

De la même façon, si X est une fonction qui lit un fichier séquentiellement, la première instruction multipliera par cinq une valeur lue, et par cinq la valeur suivante. Quant à la seconde instruction, elle multipliera par dix le premier nombre lu. Et les résultats seront différents, à moins que toutes les valeurs du fichier ne soient égales.

DE MEILLEURES SOLUTIONS AUX JEUX ET CASSE-TÊTE

Le n° 9 de *LIST* (page 14) présentait vos nombreuses solutions au jeu 17. L'objet de ce jeu consistait à écrire un programme d'initialisation d'une matrice unité en un temps minimum. Comme rien n'est impossible en informatique, le jeu 29 publié dans ce même numéro (page 84), proposait lui aussi d'écrire un programme d'initialisation de la matrice unité. Mais il devait être le plus court possible, et ne pas tenir compte du temps d'exécution. La solution proposée en cinq lignes (*LIST* 10, page 63), si elle était astucieuse, n'était pas la plus simple. En effet, Philippe Bérenger de Wizernes, Daniel Glazman de Paris et Philippe Matet de Liancourt ont tout simplement repris la définition de la matrice unité : elle comporte des 1 aux emplacements où les numéros de lignes sont égaux aux numéros de colonnes, et des 0 partout ailleurs. Ainsi, son initialisation peut s'écrire facilement de la façon suivante :

```
10 FOR L = 1 TO MAX
20     FOR C = 1 TO MAX
30         A(L,C) = ABS(L=C)
40     NEXT C
50 NEXT L
```

La fonction ABS de la ligne 30 retourne la valeur absolue de l'expression à laquelle elle s'applique. Ici, elle rend le programme assez général, puis-

que l'expression logique qui la suit, L = C, ne donne pas exactement les mêmes résultats selon les Basic. Certains attribuent la valeur 1 à cette expression quand elle est vraie, d'autres la valeur – 1. La fonction ABS met tout à 1. Or l'expression est vraie lorsque L est égal à C, c'est-à-dire lorsqu'on se trouve sur la diagonale principale justement.

Les autres solutions qui nous sont parvenues tiennent en trois ou quatre lignes. Elles sont moins claires que ce premier programme, mais il faut dire que l'objectif du jeu n'était pas de faire un programme facile à comprendre. Toutes ces solutions ramènent la matrice qui a deux dimensions, à un vecteur à une dimension. Cela permet de supprimer une boucle et donc de gagner deux lignes en faisant varier l'indice de 1 à MAX × MAX. Mais il est nécessaire de calculer, à partir de la valeur de I, d'une part le numéro de ligne et d'autre part le numéro de colonne. Si l'on dispose de l'opérateur DIV — qui effectue la division entière —, le numéro de ligne est égal à (I – 1) DIV MAX + 1. Sinon, on prend la formule : INT ((I – 1) / MAX) + 1.

Le numéro de colonne est calculé d'une façon analogue, en utilisant l'opérateur modulo qui retourne le reste de la division entière. On dispose donc de l'expression (I – 1) MOD MAX + 1. Et, sans l'opérateur MOD : I – MAX × INT((I – 1) / MAX).

A partir de là, il est possible d'indexer convenablement les éléments de la matrice. Reste donc à trouver une formule dont le résultat serait 1 sur la diagonale principale de la matrice et 0 partout ailleurs. La solution proposée par Pierre Barnouin de Cabris et par Philippe Bérenger s'inspire de celles que l'on utilise couramment en APL.

On peut considérer une matrice unité comme une succession de séquences de MAX + 1 valeurs, la première étant égale à 1 et les MAX suivantes à 0.

Ainsi, lorsque I est divisible par MAX + 1, la valeur 1 doit être mise dans l'élément de la matrice. C'est ainsi que nous obtenons le programme suivant :

```
10 FOR I = 1 TO MAX * MAX
20 A (1 + INT ((I – 1) / MAX), I –
   MAX * INT ((I – 1) / MAX)) = ABS
   (((I – 1) / (MAX + 1)) = INT
   ((I – 1) / (MAX + 1)))
30 NEXT I
```

Deux autres lecteurs, Jean Besse de Genève (Suisse) et M. Pradines de Versailles, proposent une solution plus simple en testant si le numéro de ligne est égal au numéro de colonne. C'est ainsi qu'ils obtiennent le programme suivant :

```
10 FOR I = 1 TO MAX * MAX
30 A((I – 1) DIV MAX + 1, (I – 1)
   MOD MAX + 1) = ABS (I DIV
   MAX = I MOD MAX)
40 NEXT I
```

Philippe Matet a écrit ce même programme, mais sans utiliser les opérateurs DIV et MOD ; il obtient donc :

```
10 FOR I = 1 TO MAX * MAX
30 A(1 + INT ((I – 1) / MAX),
   I – MAX * INT ((I – 1) / MAX))
   = ABS ((1 + INT ((I – 1) / MAX))
   = (I – MAX * INT ((I – 1) / MAX)))
40 NEXT I
```

Comme nous le constatons, toutes ces solutions reposent sur l'évaluation d'une expression logique dont le résultat, égal à 0 ou à 1, est stocké dans l'élément correspondant de la matrice.

Marie-Claude Babron et Douglas Rutledge, de Paris, ont recherché une expression arithmétique qui vaut 1 quand N est égal à C et 0 dans le cas contraire. Le programme qu'ils obtiennent est le suivant :

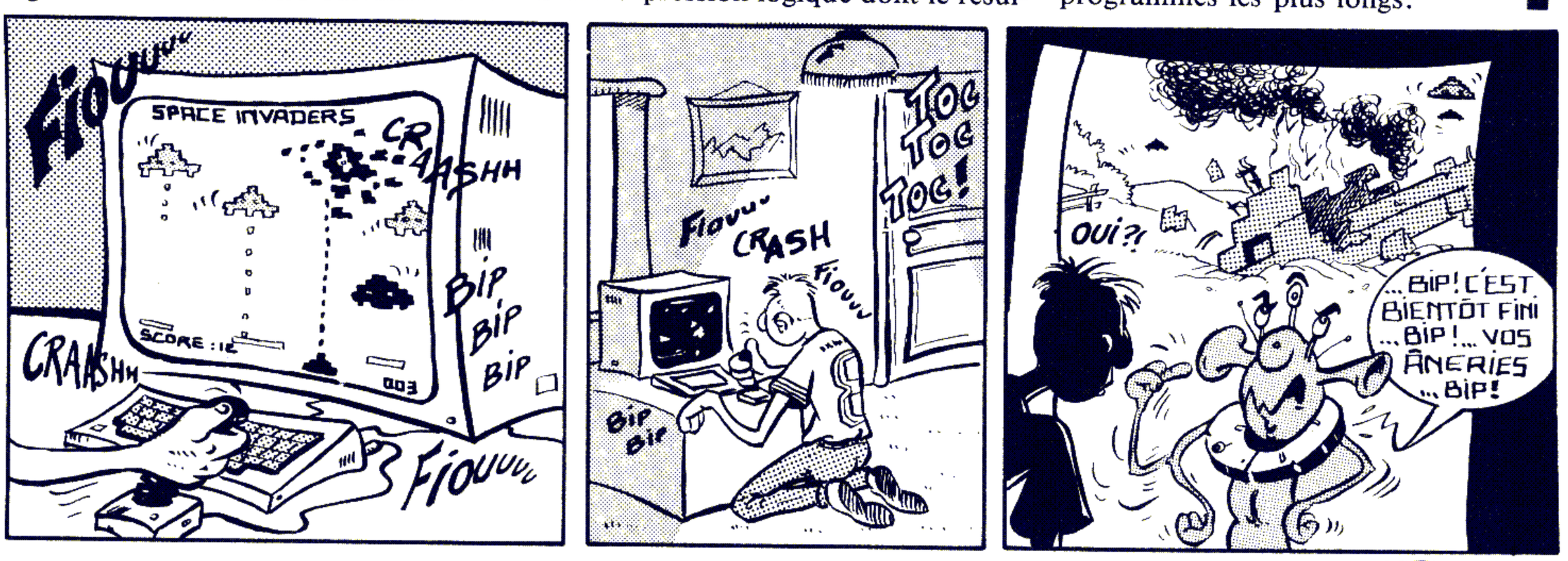
```
10 FOR I = 1 TO MAX * MAX
30 A(1 + INT ((I – 1) / MAX), I –
   MAX * INT ((I – 1) / MAX)) =
   INT (1 – ABS ((INT ((I – 1) / MAX)
   * (MAX + 1) – I + 1) / MAX))
40 NEXT I
```

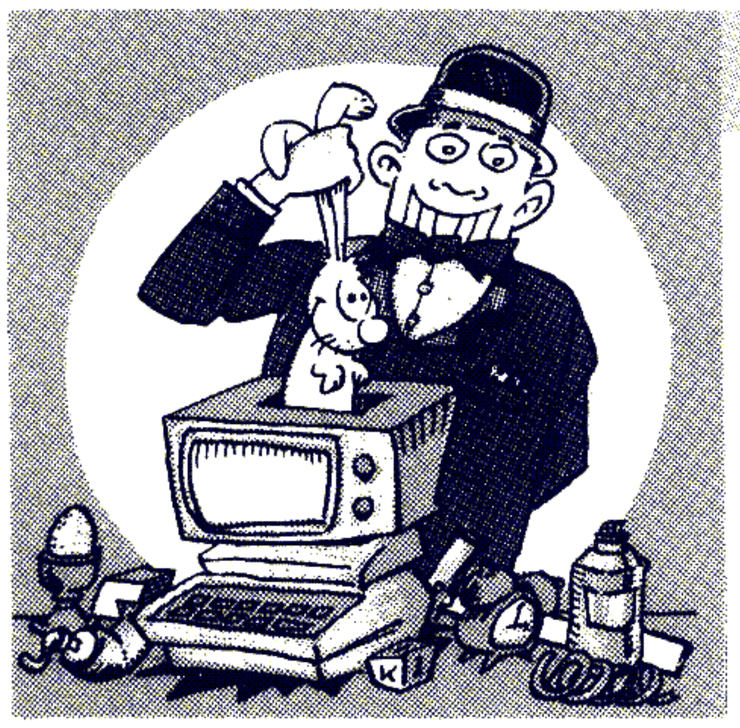
La formule utilisée crée une matrice temporaire comportant sur la diagonale principale des valeurs égales aux numéros de ligne et partout ailleurs des valeurs comprises entre 1 et MAX^2.

En soustrayant de chacun des éléments de la matrice la valeur de I qui leur correspond, on obtient une nouvelle matrice ayant des 0 sur la diagonale principale et des valeurs comprises entre – MAX + 1 et MAX – 1. En divisant chaque élément de la matrice par MAX et en soustrayant cette valeur de 1, on obtient encore une nouvelle matrice. Elle est formée de 1 sur la diagonale principale et de valeurs comprises entre 0 et 1 partout ailleurs. Enfin, en prenant la valeur entière de chacun de ces éléments, on obtient une matrice unité.

Sans qu'il soit besoin de faire un test de performances, on constate que les solutions obtenues, si elles sont courtes, sont très longues en temps d'exécution. Il est intéressant de les rapprocher des solutions publiées pour le jeu 17 (programme d'initialisation de matrice unité le plus rapide possible).

Nous voyons clairement ce que l'on appelle en informatique le compromis espace/temps, c'est-à-dire que les solutions les plus courtes sont les plus longues en temps d'exécution, alors que les solutions les plus rapides conduisent aux programmes les plus longs. ■





LA BOÎTE A MALICES...

PRENEZ un programme et ôtez-en très soigneusement toutes les astuces, des plus élémentaires aux plus subtiles. Vous êtes certain de n'en avoir laissé passer aucune ? Bien. Que reste-t-il ? Rien, ou peut-être une bogue ou deux (tout le monde peut se tromper). En fait, tout programme n'est qu'une suite d'astuces. Dans les pages qui suivent, vous en trouverez un grand nombre. Certaines sont de portée très générale. D'autres ne valent que pour un matériel particulier. Mais dans tous les cas, vous aurez intérêt à être curieux, à fouiner dans la boîte à malices. Même s'il ne s'agit pas de votre machine, vous trouverez souvent des idées à reprendre. Par ailleurs, vous avez sans doute vos propres recettes, vos façons de faire... Si ce ne sont pas des secrets que vous cherchez à conserver jalousement, faites-en part au journal. Celles qui nous paraîtront les plus intéressantes enrichiront à leur tour la boîte à malices. Tous les lecteurs pourront ainsi en profiter.

LIST

INVERSION VIDÉO

L'extension Basic proposée (LEX) a pour but, une fois assemblée, d'offrir une inversion vidéo instantanée de tout ce qui est présent à l'affichage.

La principale difficulté rencontrée dans l'écriture de cet ordre, dont le nom est INVERSE, a été la discontinuité de la mémoire d'écran du 71 : les 132 octets représentant les 132 colonnes de l'afficheur sont répartis en 3 zones distinctes ! Ces zones vont de 2E104 à 2E160 pour les 46 colonnes de gauche, de 2E200 à 2E260 pour les 48 du milieu et enfin de 2E300 à 2E34C pour les 38 de droite.

De cette disposition découle la structure du LEX comprenant (à partir de l'étiquette INV) trois appels successifs au sous-programme à l'étiquette SUB. L'adresse de début de zone est mise dans le pointeur D1 puis celle de fin dans le registre B. On charge ensuite FF dans les quartets 0 et 1 de C et on

Inversion vidéo
LEX pour HP-71B
Auteur Olivier Arbey


```
        LEX     'INVERSE'
        ID      #5D
        MSG     0
        POLL    0
DD1ST   EQU     #2E300
DD1END  EQU     #2E34C
DD2ST   EQU     #2E200
DD2END  EQU     #2E260
DD3ST   EQU     #2E104
DD3END  EQU     #2E160
OUTELA  EQU     #05303
NXTSTM  EQU     #08A48
        ENTRY   INV
        CHAR    #D
        KEY     'INVERSE'
        TOKEN   5
        ENDTXT
        NIBHEX  00
        REL(5)  INVERd
        REL(5)  INVERp
INV
        SETHEX
        D1=(5)  DD3ST
```

lit l'octet de la colonne à inverser dans le champ X de A, préalablement mis à zéro.

On effectue le complément à FF pour obtenir l'effet d'inversion puis on remet tout en place par l'intermédiaire du pointeur D1 que l'on incrémente de 2 (on opère octet par octet) avant de le comparer au champ B de A pour savoir si le travail est achevé. Si ce n'est pas le cas, on repart à l'étiquette LP. Sinon, on se branche au prochain ordre à exécuter par la routine NXTSTM.

Les deux dernières étiquettes forment les indispensables routines de codage et de décodage de chaque mot Basic en son identificateur pour interprétation. Dernière chose : surveillez vos ID et vos « tokens » pour éviter les conflits...

Voilà, les ombres chinoises sont à votre portée désormais...

Olivier ARBEY

```
        LC(5)   DD3END
        GOSUB   SUB
        D1=(5)  DD2ST
        LC(5)   DD2END
        GOSUB   SUB
        D1=(5)  DD1ST
        LC(5)   DD1END
        GOSUB   SUB
        GOVLNG  NXTSTM
SUB     BCEX    A
LP      C=0     X
        P=      0
        LCHEX   FF
        A=0     X
        A=DAT1  2
        C=C-A   X
        DAT1=C  2
        D1=D1+  2
        CD1EX
        A=C     A
        CD1EX
        ?B#A    A
        GOYES   LP
        RTN
INVERd  GOVLNG  OUTELA
INVERp  RTNCC
        END
```

TO7 et TO7/70

SAUVEGARDER L'ÉCRAN

Il vous est sûrement arrivé de vouloir sauvegarder quelque part en mémoire tout ce qui est affiché à l'écran, aussi bien textes que graphiques, pour les réutiliser en les faisant réapparaître automatiquement.

La routine Basic ci-dessous, qui fait appel au langage-machine, vous permet

Sauvegarde d'écran
Routine Basic pour TO7 et TO7/70
Auteur Frédéric Boucher


```
10 CLEAR,48899
20 A$="B6E7C38A01B7E7C38E4140108E7FBCA680A7A08C5F4026F7
 B6E7C384FEB7E7C38E4140108E9DD0A680A7A08C5F4026F739B6E7
 C38A01B7E7C38E4140108E7FBCA6A0A780108C9DBC26F6B6E7C384
 FEB7E7C38E4140108E9DD0A6A0A780108CBBD026F639"
30 ADR=48900 : FOR I=0 TO 99
40 POKE ADR+I,VAL("&H"+MID$(A$,2*I+1,2))
50 NEXT
```

Sauvegarde d'écran, la liste assemblée

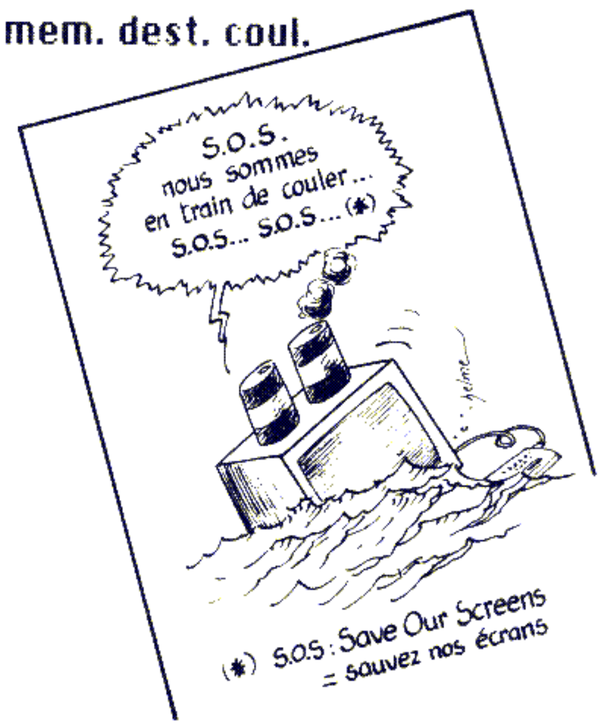
```
SAUVEGARDE

START   LDA   >59331
        ORA   #1
        STA   >59331        Mode graphique
        LDX   #16704        Début mem. écran
        LDY   #32700        Début mem. dest. graph
LOOP    LDA   ,X+
        STA   ,Y+
        CMPX  #24384        Fin mem. écran
        BNE   LOOP

        LDA   >59331
        ANDA  #254
        STA   >59331        Mode couleur
        LDX   #16704
        LDY   #40400        Début mem. dest. coul.
LOOP1   LDA   ,X+
        STA   ,Y+
        CMPX  #24384
        BNE   LOOP1
        RTS

RECUPERATION

START   LDA   >59331
        ORA   #1
        STA   >59331
        LDX   #16704
        LDY   #32700
```



```
LOOP2   LDA   ,Y+
        STA   ,X+
        CMPY  #40380        Fin mem. dest. graph.
        BNE   LOOP2

        LDA   >59331
        ANDA  #254
        STA   >59331
        LDX   #16704
        LDY   #40400
LOOP3   LDA   ,Y+
        STA   ,X+
        CMPY  #48080        Fin mem. dest. coul.
        BNE   LOOP3
        RTS
```

Pour que tout l'écran soit recopié, remplacer les 16704 par des 16384.

de sauvegarder la totalité de l'écran en un clin d'œil, sans avoir à subir la lenteur du Basic et du lecteur de cassettes.

Après avoir entré et fait exécuter le programme, sauvegardez sur cassette les codes machines (100) placés en mémoire entre &HBF04 (48900) et &HBF67 (48999) en tapant :

SAVEM '' SAVECRAN '' , &HBF04, &HBF67, &HBF04

La fausse mémoire écran « graphique » se situe entre les adresses &H7FBC (32700) et &H9DBC (40380) et la fausse mémoire « couleur » entre &H9DD0 (40400) et &HBBD0 (48080).

Remarquez au passage que chaque fausse mémoire écran ne tient pas tout à fait en 8 Koctets. Ceci a été fait volontairement pour que la ligne 0 ne soit pas recopiée : c'est là que vous pourrez entrer vos instructions.

Ainsi, pour recopier une image écran en mémoire, placez-vous en ligne 0 et tapez : EXEC 48900. Pour la faire répparaître, tapez : EXEC 48949.

Ce procédé coûtera malheureusement presque 16 Ko de mémoire vive. Pourtant il est toujours possible, si vous ne voulez plus de l'image, « d'écraser » cette zone par un programme Basic devenu trop long...

Frédéric BOUCHER

FX-702P

GAGNER DES PAS

Dans un programme, il est souvent nécessaire de savoir combien de fois une opération a été effectuée. On ajoute — ou on retranche — alors 1 à une variable déterminée, comme dans les lignes suivantes :

```
1VAC:FOR I=0 TO 99:IF KEY≠''
'';A=A+1
2 NEXT I:PRT A
```

Ici, la variable A est augmentée de 1 si une touche a été frappée. Avec le FX-702P, l'utilisation des fonctions statistiques permet de gagner en nombre de pas. Ainsi, les lignes suivantes font la même chose que les précédentes, mais avec moins de pas (la première version fait 32 pas, la seconde en a 29) :

```
1SAC:FOR I=0 TO 99:IF KEY≠''
'';STAT 0
2 NEXT I:PRT CNT
```

Maintenant, à chaque pression sur une touche, un nombre sera entré dans STAT et le nombre de données CNT sera augmenté de 1. Pour enlever 1 à CNT, il suffit de remplacer STAT 0 par DEL 0.

Patrick GÉRARD

FORTH

Encore plus fort :
le programme qui reproduit
le programmeur !!!

CLONE INFORMATIQUE

Un programme clone, c'est-à-dire un programme dont l'exécution entraîne la reproduction exacte de sa liste, a déjà été publié en Basic (il faisait l'objet du jeu 22, la solution se trouvant dans *LIST 8* page 84). Voici un nouveau clone, écrit en Forth celui-là. Son seul but : se reproduire. Un programme pour le sport ! Qui relèvera le défi d'écrire un clone en Pascal ?

Liste et/ou exécution du programme clone en Forth

```
30 MSG" MSG"
31 MSG" : CLONE PAGE"
32 MSG" 12 MESSAGE 4 SPACES 1 . CR"
33 MSG" CR 38 30 DO"
34 MSG"   I DUP . 30 MESSAGE 34 EMIT"
35 MSG"   SPACE MESSAGE 34 EMIT CR"
36 MSG" LOOP CR"
37 MSG" 38 31 DO I MESSAGE CR LOOP ;"

: CLONE PAGE
12 MESSAGE 4 SPACES 1 . CR
CR 38 30 DO
  I DUP . 30 MESSAGE 34 EMIT
  SPACE MESSAGE 34 EMIT CR
LOOP CR
38 31 DO I MESSAGE CR LOOP ;
```

Mathieu CHANTE

NOMBRES DÉCIMAUX

ARRONDIR LA OÙ IL FAUT

Lorsqu'on fait beaucoup de calculs, on rencontre souvent des problèmes d'arrondi. En effet, il est toujours pénible de « traîner » avec soi une liste de chiffres dont on n'a que faire (et qui ne sont pas forcément exacts !), alors que, le plus souvent, trois ou quatre chiffres suffisent à la mantisse. De plus, il est parfois nécessaire d'arrondir un résultat pour pouvoir poursuivre les calculs.

Une première solution, très connue, consiste à arrondir à une puissance de 10 près, que l'on choisit. Ainsi, arrondir X à 10^{-N} près, peut se programmer :

```
100 X=INT(X*10^N+0.5)*10^-N
```

Ou encore :

```
100 DEFFNR(X)=INT(X*10^N+
    0.5)*10^ -N
```

Et même, avec certains Basic :

```
100 DEFFN ROUND (X,N)=INT
    (X * 10^N + 0.5) * 10^-N
```

Mais on se trouve vite limité lorsqu'un programme doit traiter des valeurs très variées. Il serait donc intéressant d'arrondir à une proportion près de la valeur de départ, autrement dit ne conserver qu'un nombre donné de chiffres significatifs. Par exemple, arrondir au millième de la valeur considérée (à 0,1 %), soit conserver quatre chiffres significatifs. Ainsi, pour arrondir X à $X*10^{-N}$ près, ou conserver N+1 chiffres significatifs :

```
100 Y=N-INT(LOG(ABS(X))/
    LOG (10)):X=INT(X*10^Y+.5)*
    10^ -Y
```

Ici, LOG représente le logarithme naturel. Si on dispose du logarithme décimal (aussi appelé LOG), on remplace LOG(ABS(X))/LOG(10) par LOG (ABS(X)). Par exemple, avec un Basic puissant comme celui du BBC, on obtient :

```
100 DEFFN ROUND(X,N)
110 LOCAL  Y:Y=N-INT(LOG
    (ABS(X)))
120 X=INT(X*10^Y+.5)*10^-Y
```

Gilles CIZERON

DAI

TOUT SAVOIR SUR LES FICHIERS

Les heureux possesseurs d'un Dai agrémenté d'un système de disquettes Ken-Dos doivent absolument avoir cet utilitaire dans leur boîte à outils. En effet, le programme Ken-Zapper liste en clair les attributs des fichiers disque, leur localisation en pistes et secteurs, et même le code de verrouillage, s'il y en a un. Pratique, pour les étourdis, mais à ne pas laisser traîner entre toutes les mains !

Pour fonctionner correctement, ce programme ne doit être lancé qu'après s'être assuré que le tampon de DIRectory se trouve bien à sa place habituelle en mémoire. Pour plus de sûreté, faire "BUF" avant toute chose. Ensuite, lancer (RUN) Ken-Zapper. Il demande d'appeler le DIRectory désiré, puis de le relancer par RUN 100. C'est tout. Le programme fait le reste.

Comme j'étais pressé, quand je l'ai écrit, je ne me suis pas trop embarrassé de finasseries. C'est ainsi que le numéro du fichier testé (ligne 180) n'est juste que pour le premier DIRectory, et que le STATUS n'est décodé en clair que pour certains états (lignes 260 à 280). Vous pouvez ajouter ce qui manque.

Outre le code de protection (pour les étourdis qui bloquent un fichier, puis

Ken-Zapper
Programme pour Dai et Ken-Dos
Auteur Alain Mariatte


```
10    MODE 0:COLORT 8 0 0 0:PRINT CHR$(12);CHR$(13)
20    PRINT "            KEN       Z    A    P    P    E    R"
30    PRINT "            ######################################"
40    PRINT
50    PRINT :PRINT
60    PRINT "CHARGER LE  ZAPPER.  APPELER  LE  BLOCK DE  DIRECTORY"
70    PRINT "DESIRE (ex: DIR2) PUIS <RUN 100>.  RECOMMENCER AUTANT"
80    CURSOR 0,1:STOP
100   GOSUB 750
110   GOSUB 130
120   END
130   REM DECODE 1 SECTOR DIR
140   REM
150   FOR I=#AF50 TO #B350 STEP 32
160   IF PEEK(I+26)=0 THEN RETURN:REM SECTOR=0 :FICHIER NON CREE
170   FN=PEEK(I)
180   PRINT :PRINT "FILE NUMBER                    :";FN-#20
190   P=I
200   P=P+1:L=PEEK(P):P=P+1
210   N$="":FOR K=P TO P+L-1:N$=N$+CHR$(PEEK(K)):NEXT
220   PRINT "FILE NAME                             : ";N$
230   P=I+16:ST=PEEK(P)
240   PRINT "STATUS                                : ";
250   IF ST<>0 AND ST<>1 AND ST<>#10 THEN PRINT "#";HEX$(ST):GOTO 290:REM #28 DN
A ASS.
260   IF ST=0 THEN PRINT "OPENED":GOTO 290
270   IF ST=1 THEN PRINT "LOCKED":GOTO 290
280   IF ST=#10 THEN PRINT "DELETED":GOTO 290
290   P=I+17:T=PEEK(P)
300   PRINT "FILE TYPE                             : ";
310   IF T=0 THEN PRINT "BASIC":GOTO 400
320   IF T=1 THEN PRINT "BINAIRE":GOTO 400
330   IF T=2 THEN PRINT "FPT ARRAY":GOTO 400
340   IF T=3 THEN PRINT "SRC DNA <or new SPL source>":GOTO 400
350   IF T=#12 THEN PRINT "INTEGER ARRAY":GOTO 400
360   IF T=#22 THEN PRINT "ALPHANUM.ARRAY":GOTO 400
370   IF T=#F6 THEN PRINT "DBS <or SPL source >":GOTO 400
380   IF T=#F4 THEN PRINT "RND <or PASCAL source>":GOTO 400
```

```
390   PRINT
400   IF T IAND 2=2 OR T IAND #12=#12 OR T IAND #22=#22 THEN 430
410   P=I+18:GOSUB 740
420   PRINT "START ADDRESS                : #";HEX$(AD)
430   P=I+20:GOSUB 740
440   PRINT "FILE LENGTH                  : #";HEX$(AD)
450   P=I+22:GOSUB 740
460   IF T=0 THEN PRINT "LENGTH OF TEXTBUFFER        : #";HEX$(AD)
470   IF T=1 AND AD=0 THEN PRINT "NO EXECUTION ADDRESS"
480   IF T=1 AND AD<>0 THEN PRINT "EXECUTION ADDRESS          : #";HEX$(AD)
490   P=I+24:LO=PEEK(P):P=I+25:HI=PEEK(P)
500   IF LO+HI=0 THEN 530
510   CO=256*(HI-#D)+LO-#D
520   PRINT "LOCK CODE OF FILE            :";CO
530   P=I+28:BL=PEEK(P):P=I+29:BH=PEEK(P):GOSUB 820
540   PRINT "CREATED                      : ";JJ$+MM$+AA$
550   P=I+30:BL=PEEK(P):P=I+31:BH=PEEK(P):GOSUB 820
560   PRINT "UPDATED                      : ";JJ$+MM$+AA$
570   P=I+26:SE=PEEK(P)
580   PRINT "NUMBER OF SECTORS            :";SE
590   STCOUNT=#AD90+1:EFAM=#AF4F:C=0:TTR=0:AL=0:TR=0
600   PRINT "LOCATION ON DISK :";
610   FOR J=STCOUNT TO EFAM:C=C+1:NF=PEEK(J)
620   IF NF<>FN THEN 660
630   TR=C/5:SEC=(C MOD 5)+1:IF TR<10 THEN AL=1
640   IF TTR<>TR THEN PRINT :PRINT "track :";TR;"   sector(s)  ";SPC(AL);"    :"
;:TTR=TR
650   PRINT SEC;" - ";
660   AL=0:NEXT:IF TR=0 THEN PRINT ">>>ACTUALLY KILLED<<<"
670   PRINT
680   G=GETC
690   IF G=ASC("S") THEN RETURN
700   IF G=0 THEN 730
710   G=GETC:IF G=0 THEN 710
720   IF G=#20 THEN 730:GOTO 690
730   PRINT :NEXT I
740   L=PEEK(P):H=PEEK(P+1):AD=L+256*H:RETURN
750   REM FENETRE DE SCROLLING
760   REM
770   POKE #8A,#69:POKE #8B,#BF
780   POKE #8C,#E5:POKE #8D,#B3
790   POKE #84,#D5:POKE #85,#B3
800   POKE #8E,#D5:POKE #8F,#B3
810   RETURN
820   REM DECODAGE DE LA DATE
830   REM
840   BJ=BL IAND 31:REM 5 BITS
850   BM=BL IAND 15:REM 4 BITS LOW NIBBLE
860   AA=BH IAND 240:REM BH HIGH NIBBLE
870   AL=AA SHR 4:REM LOW PART OF YEAR
880   AT=BL IAND 192:REM BITS 7&8
890   IF AT=0 THEN BA=80+AL:REM 80-95
900   IF AT=64 THEN BA=96+AL:REM 96-99
910   IF AT=128 THEN BA=AL:REM 00-15
920   IF AT=192 THEN BA=16+AL:REM 16-31
930   BBJ$=STR$(BJ):BBM$=STR$(BM):BBA$=STR$(BA)
940   BJ$=MID$(BBJ$,1,1):IF BJ>9 THEN BJ$=MID$(BBJ$,1,2)
950   BM$=MID$(BBM$,1,1):IF BM>9 THEN BM$=MID$(BBM$,1,2)
960   BA$=MID$(BBA$,1,1):IF BA>9 THEN BA$=MID$(BBA$,1,2)
970   JJ$="0"+BJ$:IF BJ>=10 THEN JJ$=BJ$
980   MM$="0"+BM$:IF BM>=10 THEN MM$=BM$
990   AA$="0"+BA$:IF BA>=10 THEN AA$=BA$
1000  RETURN
*
```

Exemple d'utilisation de Ken-Zapper

```
FILE NUMBER                    : 2
FILE NAME                      : DNA
STATUS                         : #2A
FILE TYPE                      : BINAIRE
START ADDRESS                  : #1100
FILE LENGTH                    : #1F00
EXECUTION ADDRESS              : #1100
CREATED                        : 190984
UPDATED                        : 190984
NUMBER OF SECTORS              : 8
LOCATION ON DISK :
track : 3   sector(s)          : 2 -  3 -  4 -  5 -
track : 4   sector(s)          : 1 -  2 -  3 -  4 -
```

```
FILE NUMBER                    : 1
FILE NAME                      : AUTOEXEC.HELLO
STATUS                         : #2
FILE TYPE                      : BASIC
START ADDRESS                  : #3EC
FILE LENGTH                    : #1C
LENGTH OF TEXTBUFFER           : #1B
CREATED                        : 100884
UPDATED                        : 100884
NUMBER OF SECTORS              : 1
LOCATION ON DISK :
track : 3   sector(s)          : 1 -
```

oublient le mot de passe ensuite !), le programme donne les renseignements les plus importants concernant les fichiers : l'adresse de début d'implantation, la longueur, l'adresse d'exécution (si c'est un programme binaire), ou la longueur du tampon texte (si c'est du Basic). Ceci permettra de récupérer un fichier accidenté et de le replacer où il faut.

Mais, pour récupérer quoi que ce soit, encore faut-il savoir où chercher ! Rassurez-vous, Ken-Zapper veille et vous aide : les derniers renseignements fournis concernent le nombre de secteurs utilisés par le fichier, et leur localisation, exprimée en pistes et secteurs. Au passage, cet utilitaire montre immédiatement les fichiers dont les secteurs sont disjoints, par suite de nombreuses réécritures sur le disque. Vous saurez donc quand il est nécessaire de COMPACter un disque, pour remettre tout cela en ordre.

Pour ce qui est de la technique de programmation, le programme se contente d'exploiter ce que dit le mode d'emploi du Ken-Dos. Il décode les 32 octets de DIRectory alloués à chaque fichier, à partir de l'adresse AF50H, et la File Allocation Map (à partir de AD50H). Il exploite aussi ce que le mode d'emploi ne dit pas ! Pour la curiosité, regardez donc dans la liste comment le LOCK CODE est constitué (lignes 490 à 520), et comment la DATE est composée (lignes 820 à 990). Pas évident, n'est-ce pas ?

Au fait, savez-vous que vers l'an 2035, les dates inscrites sur les fichiers seront fausses ? Dépêchez-vous d'utiliser votre Ken-Dos !

Alain MARIATTE

INTERROMPRE LE BALAYAGE

Le programme présenté ici montre la manière de gérer les interruptions provoquées par le passage du spot à une certaine ligne de l'écran. Pour comprendre ce qui se passe, il faut savoir que Monsieur Processeur est la personne qui, à l'intérieur du C.64, s'occupe personnellement de l'exécution des programmes, que ceux-ci soient écrits en langage-machine ou en Basic.

Dans le cas d'un Basic interprété, il est beaucoup moins performant car il passe la majeure partie de son temps à chercher la signification des mots dans son dictionnaire. Son principal handicap est son manque total de mémoire. Ainsi, s'il doit exécuter 100 fois une boucle comportant le mot PRINT, il cherchera 100 fois ce mot dans son dictionnaire.

De temps à autre, il est interrompu dans son travail par le téléphone (le signal d'interruption IRQ). Comme il n'a aucune mémoire, il se rachète par un soin extrême et avant d'aller répondre, il note sur son bloc-notes (la pile du 6502) l'endroit où il abandonne son travail et l'organisation de ses instruments de bureau (sauvegarde des registres).

Ensuite, il sait par habitude qu'il s'agit d'un appel de Monsieur Clavier qui est le seul à connaître son numéro de téléphone. Il doit alors aller vérifier chaque touche du clavier pour savoir si elle a été pressée ou non, surveiller le moteur du magnétophone, remettre le coucou à l'heure (horloge TI) et faire clignoter le curseur.

Pour ne pas être dérangé de nouveau pendant ce travail de routine, il prend soin de débrancher la sonnerie du téléphone avant toute chose (instruction SEI). Lorsqu'il a terminé, il rebranche la sonnerie (instruction CLI), saisit la feuille supérieure du bloc-notes pour savoir où il en était, réorganise son bureau et reprend son travail jusqu'à ce que la sonnerie retentisse à nouveau un soixantième de seconde plus tard.

Rassurez-vous, il a eu le temps de faire plein de choses entre temps. Le processus recommence alors de manière identique.

Maintenant, nous voudrions que Monsieur Processeur affiche la partie supérieure de la bordure d'écran en vert et la partie inférieure en rouge. Pour cela, nous communiquons son numéro à Monsieur Peintre qui est chargé de repeindre cette bordure ligne par ligne, 50 fois par seconde, avec la couleur du pot située en 53280. Monsieur Peintre devra appeler Monsieur Processeur à chaque fois qu'il tracera la ligne dont le numéro se trouve aux adresses 53266 et 53265 (bit 7) pour que celui-ci lui change le pot de peinture. Il faudra également signaler à Monsieur Processeur que Monsieur Clavier ne sera plus le seul à l'appeler et qu'il devra décrocher le téléphone pour demander l'identité de son correspondant. Si le bit 0 de l'adresse 53273 vaut 1, ce sera le peintre, sinon il s'agira bien sûr de Monsieur Clavier.

Il y a encore un autre problème : si Monsieur Peintre appelle juste après Monsieur Clavier, Monsieur Processeur ne peut pas l'entendre car il vient

Interruptions balayage
Routine Basic pour Commodore 64
Auteur Hervé Le Marchand


```
150 REM LECTURE DES DATA
160 :
170 FOR I=49152 TO 49240
180  READ X:POKE I,X
190  S=S+X
200 NEXT I
210 :
220 IF S <>9734 THEN PRINT"ERREUR DANS LES DATA":STOP
230 SYS 49152:END
240 :
250 REM DATA LANGAGE MACHINE
260 :
1000 DATA 120,169,31,141,20,3,169,192
1010 DATA 141,21,3,169,27,141,17,208
1020 DATA 173,26,208,9,1,141,26,208
1030 DATA 169,1,141,13,220,88,96,173
1040 DATA 25,208,41,1,240,30,173,18
1050 DATA 208,201,140,208,13,169,2,141
1060 DATA 32,208,169,0,141,18,208,76
1070 DATA 68,192,169,5,141,32,208,169
1080 DATA 140,141,18,208,169,1,141,25
1090 DATA 208,173,13,220,41,1,240,3
1100 DATA 76,49,234,104,168,104,170
1110 DATA 104,64

READY.
```

de débrancher la sonnerie. Il ne pourra prendre en compte sa requête qu'une fois sa routine terminée et il changera le pot de peinture avec du retard, lorsque Monsieur Peintre sera arrivé quelques lignes plus loin. Le résultat ne sera pas bien joli à voir. Il y a plusieurs manières de s'en sortir, heureusement.

On peut tout d'abord demander à Monsieur Processeur de considérer en priorité les besoins du peintre et, une fois le pot de peinture remplacé, d'aller voir sur le répondeur automatique si Monsieur Clavier a appelé pendant ce temps (bit 0 de l'adresse 56333, à 1). Si oui, on repart aussitôt lire le clavier. Il est peu probable que le peintre rappelle pendant ce temps car il vient tout juste d'être consulté. Si, malgré tout, il faut changer de pot très fréquemment (toutes les 10 lignes, par exemple), on ne peut plus s'en sortir de cette manière. Il est alors possible de demander à Monsieur Clavier d'appeler moins fréquemment.

Interruptions balayage, la liste assemblée

```
190 033C          POT         =  53280
190 033C          VERT        =  5
200 033C          ROUGE       =  2
210 033C          DEBUTVERT   =  0                ! LIGNE OU DEBUTE LE VERT
220 033C          DEBUTROUGE  =  140              ! ET LE ROUGE
230 033C          REPERE      =  53266            ! LIGNE OU LE PEINTRE DOIT APPELER
240 033C          CORRESPOND#  =  53273           ! IDENTIFICATION DE L'INTERLOCUTEUR
250 033C          NUMERO      =  53274            ! AUTORISATION D'APPELER
260 033C          COMPTEUR    =  56333            ! SIGNAL DE FIN DE COMPTAGE DE MR CLAVIER
270 033C          VECIRQ      =  788
280 033C          LITCLAVIER  =  $EA31
290 033C          !
300 C000          * =$C000
310 C000          !
320 C000          ! NOUVELLES INSTRUCTIONS A MR PROCESSEUR
330 C000          !
340 C000 78                   SEI
350 C001 A91F                 LDA #<INTER           ! ADRESSE DES NOUVELLES CONSIGNES
360 C003 8D1403               STA VECIRQ
370 C006 A9C0                 LDA #>INTER
380 C008 8D1503               STA VECIRQ+1
390 C00B A91B                 LDA #27               ! PAS DE LIGNE > 255
400 C00D 8D11D0               STA REPERE-1
410 C010 AD1AD0               LDA NUMERO            ! LE PEINTRE EST AUTORISE A APPELER
420 C013 0901                 ORA #1
430 C015 8D1AD0               STA NUMERO
440 C018 A901                 LDA #%00000001
450 C01A 8D0DDC               STA COMPTEUR          ! MR CLAVIER COMPTE SANS APPELER
460 C01D 58                   CLI
470 C01E 60                   RTS
480 C01F          !
490 C01F          ! ROUTINES A FAIRE LORS DE L'APPEL
500 C01F          !
510 C01F          INTER       =  *
520 C01F          !
530 C01F AD19D0               LDA CORRESPONDANT
540 C022 2901                 AND #1                ! EST-CE LE PEINTRE ?
550 C024 F01E                 BEQ CLAVIER
560 C026 AD12D0               LDA REPERE            ! OUI,OU EST IL ?
570 C029 C98C                 CMP #DEBUTROUGE       ! AU MILIEU ?
580 C02B D00D                 BNE ENVERT            ! NON, EN HAUT
590 C02D A902     ENROUGE     LDA #ROUGE            ! OUI, ON MET UN POT ROUGE
600 C02F 8D20D0               STA POT
610 C032 A900                 LDA #DEBUTVERT        ! ET ON INDIQUE QUAND LE PEINTRE DEVRA RAPPELER
620 C034 8D12D0               STA REPERE
630 C037 4C44C0               JMP CLAVIER
640 C03A A905     ENVERT      LDA #VERT             ! IDEM POUR LE VERT
650 C03C 8D20D0               STA POT
660 C03F A98C                 LDA #DEBUTROUGE
670 C041 8D12D0               STA REPERE
680 C044 A901     CLAVIER     LDA #1
690 C046 8D19D0               STA CORRESPONDANT     ! REPOND OK AU PEINTRE
700 C049 ADODDC               LDA COMPTEUR          ! MONSIEUR CLAVIER A T-IL FINI DE COMPTER ?
710 C04C 2901                 AND #%00000001
720 C04E F003                 BEQ SORTIE            ! NON
730 C050 4C31EA               JMP LITCLAVIER        ! OUI, LECTURE CLAVIER,ETC...
740 C053 68       SORTIE      PLA                   ! ON REMET LES INSTRUMENTS EN PLACE
750 C054 A8                   TAY
760 C055 68                   PLA
770 C056 AA                   TAX
780 C057 68                   PLA
790 C058 40                   RTI                   ! LECTURE BLOC NOTE ET REPRISE DU TRAVAIL
```

Pendu au téléphone

En effet, c'est quelqu'un de très bête qui compte jusqu'à 64*256 = 16384, téléphone et recommence à compter. On peut lui imposer de compter jusqu'à 255*256 = 65280 entre chaque appel en faisant POKE 56325,255. Dans ce cas, le curseur clignote plus lentement. Seulement ce n'est pas toujours suffisant et comme il ne sait pas compter plus loin, il faut trouver autre chose. Eh bien, on peut tout simplement lui demander de ne plus appeler lorsqu'il a fini de compter en forçant à zéro le bit 7 de l'adresse 56333.

Ce sera alors à nous de lui demander s'il a fini son comptage en consultant le bit 0 de cette même adresse. Si nous avons le temps, nous prendrons l'initiative d'aller lire le clavier, sinon il faudra incorporer une routine de lecture clavier dans le travail habituel de Monsieur Processeur et ne le déranger que pour répondre au peintre. En pratique, il suffit d'appeler la routine qui s'en charge en $FF9F.

De façon analogue, il est possible de communiquer le numéro de Monsieur Processeur aux infirmiers spécialisés qui le préviendront de toute collision entre sprites (adresse 53273, bit 1) ou lorsque l'un d'eux se sera payé le décor (bit 2) en positionnant à 1 le bit adéquat du registre 53274.

Pour signaler à ces correspondants que leur demande a été satisfaite, il faut écrire un 1 dans le bit associé du registre 53273. Cela aura paradoxalement pour effet de remettre ce bit à 0 jusqu'à la prochaine interruption.

Hervé LE MARCHAND

BASIC

TENDRE DES FILS


```
10 'TRACES DE TABLEAUX  "FILS TENDUS"
20 '
30 ' AUTEUR :  Jean-Claude MARTIN
40 '
50 LX=640 : LY=400
60 RANDOMIZE(TIME/4)
70 CLS : PRINT "TRACE MANUEL      (1)" :
   PRINT "      AUTOMATIQUE (2)"
80 PRINT : INPUT "VOTRE CHOIX ";C
90 IF C=1 THEN 150
100 XA1=INT( LX*RND) : YA1=INT(LY*RND)
110 XB1=INT(LX*RND) : YB1=INT(LY*RND)
120 XA2=INT(LX*RND) : YA2=INT(LY*RND)
130 XB2=INT(LX*RND) : YB2=INT(LY*RND)
140 GOTO 190
150 INPUT "XA1,YA1";XA1,YA1
160 INPUT "XB1,YB1";XB1,YB1
170 INPUT "XA2,YA2";XA2,YA2
180 INPUT "XB2,YB2";XB2,YB2
190 INPUT "N= (0 POUR TERMINER  , -1
    POUR UN AUTRE TRACE) ";N
200 IF N=-1 THEN 70
210 IF N=0 THEN END
220 XPAS1=(XB1-XA1)/(N-1)
230 YPAS1=(YB1-YA1)/(N-1)
240 XPAS2=(XB2-XA2)/(N-1)
250 YPAS2=(YB2-YA2)/(N-1)
260 CLS
270 FOR I=0 TO N-1
280 X1=XA1+I*XPAS1
290 X2=XA2+I*XPAS2
300 Y1=YA1+I*YPAS1
310 Y2=YA2+I*YPAS2
320 LINE (X1,Y1)-(X2,Y2)
330 NEXT I
340 IF INKEY$="" THEN 340
350 GOTO 190
```

Fils tendus
Programme en Basic
Auteur Jean-Claude Martin

Si votre Basic est doté de l'instruction LINE (ou d'une instruction équivalente) qui trace une ligne droite entre deux points, vous allez pouvoir créer de beaux dessins rien qu'en tendant des fils. Ce résultat n'est pas difficile à obtenir : il suffit de tracer un segment de droite (A1,B1), un autre segment (A2,B2), de partager chacun d'eux en un même nombre de segments, et de joindre les points ainsi obtenus.

Le programme ci-contre réalise de telles œuvres en envisageant quelques facilités. Au départ, deux possibilités de placement des points A1, B1, A2 et B2 sont proposées : manuel ou automatique. Dans le cas d'un choix manuel, le programme demande à l'utilisateur les coordonnées des quatre points, tandis que le choix d'un placement automatique génère au hasard les coordonnées des quatre points, dans les limites de l'écran.

Le programme demande alors de préciser le nombre de segments de droites à dessiner. Puis le tracé s'effectue et reste à l'écran tant qu'on n'appuie sur aucune touche du clavier. Le fait d'appuyer sur une touche offre la possibilité de faire varier le nombre de segments sans changer le choix des points A1, B1, A2 et B2.

Le programme propose aussi de choisir d'autres points pour créer de nouvelles œuvres.

Jean-Claude MARTIN

Précisions sur le programme

Lignes 270 à 330 : tracé proprement dit.

Ligne 340 : attente de la frappe d'une touche pour passer à la suite de l'exécution.

Lignes 150 à 180 : entrée des coordonnées des points dans le cas d'un choix manuel.

Ligne 190 : saisie d'un nombre de segments à tracer.

Ligne 50 : initialisation de LX et LY qui contiennent respectivement le nombre de points en largeur et en hauteur, de l'écran.

Ligne 60 : pour éviter d'avoir toujours le même dessin en début d'exécution.

PC-1211/1212

MULTIPLIER AVEC PRÉCISION

Sur un ordinateur de poche comme le PC-1211 (ou le PC-1212), la multiplication offre un beau terrain de chasse à celui qui veut en démonter le mécanisme. Pour commencer, une observation rassurante : le résultat de la multiplication de *deux* nombres est indépendant de l'ordre des facteurs, ce qui ne signifie pas que ce résultat soit tout à fait satisfaisant. Le dernier chiffre retenu est obtenu par troncature brutale des chiffres suivants, donc toujours par défaut, au lieu d'être arrondi à l'unité la plus proche de la valeur réelle.

Par exemple, on introduit, en mode calcul, A = 2.0000942 et B = 3.000019, puis A*B – 6. Le 1211 fait les calculs sur 12 chiffres significatifs, il affiche alors : 3.2060178E – 04, ce qui signifie que la valeur calculée pour A*B est 6.00032060178 alors que la valeur réelle de ce produit est 6.0003206017898. Que 17898 soit tronqué à 178 au lieu d'être arrondi à 179 est non seulement agaçant, mais peut avoir des conséquences fâcheuses dans le cas de calculs qui exploitent les dernières décimales du produit.

Pour pallier cette imperfection, un petit utilitaire (page suivante) donne le produit de deux nombres A et B, positifs et comportant au plus 10 chiffres, avec 12 chiffres significatifs exacts, le dernier étant arrondi à l'unité la plus

proche de la valeur réelle (si le treizième chiffre est 5, le nombre est arrondi par excès ; par exemple, 13.5 est arrondi à 14).

Après la mise en route du programme par RUN, on introduit successivement les deux nombres dont on veut le produit. Celui-ci se trouve alors dans P + Q. Par exemple, si on avait introduit les valeurs 2.0000942 et 3.000019, l'écran afficherait, pour P + Q − 6 : 3.2060179E − 04. Ceci prouve que la valeur retenue pour A*B est maintenant 6.00032060179. Ce qui correspond mieux à la réalité.

Les opérations en chaîne, comme A*B*C, sont effectuées de gauche à droite, avec troncature des résultats partiels à partir du treizième chiffre. Si le treizième chiffre du produit A*B est inférieur à 5, alors le résultat partiel stocké dans la pile de données sera exact. Mais sinon, le résultat partiel sera inexact, et multiplié par C, il pourra éventuellement le rester. Mais si l'on modifie l'ordre d'introduction des facteurs, le produit A*C peut très bien donner un résultat partiel exact.

Prenons un exemple : A = 3, B = 3.000058 et C = 5.000086. Le résultat du produit A*B*C apparaît à l'écran : 45.00164401. Et pour récupérer les chiffres de garde, on fait A*B*C − 45 ; ce qui donne 0.0016440149. Mais avec C*B*A − 45, on obtient le résultat : 0.0016440147. L'ordinateur évalue donc le produit des trois nombres, tantôt à 45.0016440149, tantôt à 45.0016440147, selon l'ordre des facteurs (la valeur réelle de ce produit est 45.001644014964).

Ici, l'ordre C*B*A est moins favorable que l'ordre A*B*C car la valeur réelle de C*B, 15.000548004988, a été tronquée à 15.0005480049, et l'erreur induite (de 0.88 sur le dernier chiffre) a été multipliée par A, soit 3. Ce qui fait une erreur finale de 2.64 sur le dernier chiffre.

Malheureusement, il semble difficile de donner une règle simple d'introduction des termes, dans le cas de la multiplication, puisque ce sont les derniers chiffres significatifs des nombres introduits qui déterminent la précision du résultat.

Les propriétés d'associativité sont décevantes. On sait que les opérations entre parenthèses sont calculées en priorité et en bloc dans la pile opérationnelle. Il en va de même pour les multiplications sans le signe (AB au lieu de A*B). En reprenant l'exemple précédent, on trouve :

A*B*C − 45 = 0.0016440149
A*(B*C) − 45 = 0.0016440147
A*BC − 45 = 0.0016440147

De même pour les propriétés de distributivité. Par exemple, avec A = 2.000094, B = 3.000019 et C = 5, les résultats à l'écran sont les suivants :

A*(B + C) = 16.00079
A*(B + C) − 16 = 0.0007900017
A*B + A*C − 16 = 0.000790001
A*C + A*B − 16 = 0.0007900017

Le résultat réel de l'opération étant 16.000790001786, on voit que la disposition la plus défavorable (A*B + A*C) peut provoquer une erreur de 7 unités sur le douzième chiffre.

La morale de cette historiette ? C'est que les résultats de simples opérations de multiplication doivent être accueillis avec prudence... quand on demande une aussi grande précision.

Pierre Ladislas GEDO

Le programme

Cet utilitaire calcule les chiffres significatifs, en double précision, du produit de deux nombres A et B, sous la variable Z. Le produit recherché est P + Q.

Le programme est établi en virgule flottante et permet de calculer par exemple A*B avec : A = 2.417687129 *E − 30 et B = 4931.774388.

Le résultat indiqué est : P = 1.194766433*E − 26, Q = 2.3*E − 36. On en déduit que P + Q = 1.9476643323 *E − 26.

Sans passer par l'utilitaire, le calcul sans arrondi aurait conduit au résultat A*B = 1.19476643322*E − 26, alors que la valeur réelle de A*B avec 15 chiffres est égale à 1.19476643322895*E − 26.

Multiplication arrondie
Programme pour PC-1211
Auteur Pierre Ladislas Gedo
Copyright *LIST* et l'auteur

```
500: INPUT A,B
510: R=A/10^(INT
     LOG A-4)
520: S=B/10^(INT
     LOG B-4)
530: T=INT R
540: U=INT S
550: V=T*(S-U)+S*
     (R-T)
560: V=V-INT V
570: W=R*S-INT (R
     *S)
580: X=10^(INT
     LOG (A*B)-11
     )
590: Y=10^(INT
     LOG (R*S)-11
     )
600: Z=V-W-INT (V
     -W)
610: K=X*INT (Z/Y
     +.5)
620: P=A*B
630: Q=A*B-P+K
640: END
```

FX-802P

NE PAS EFFACER

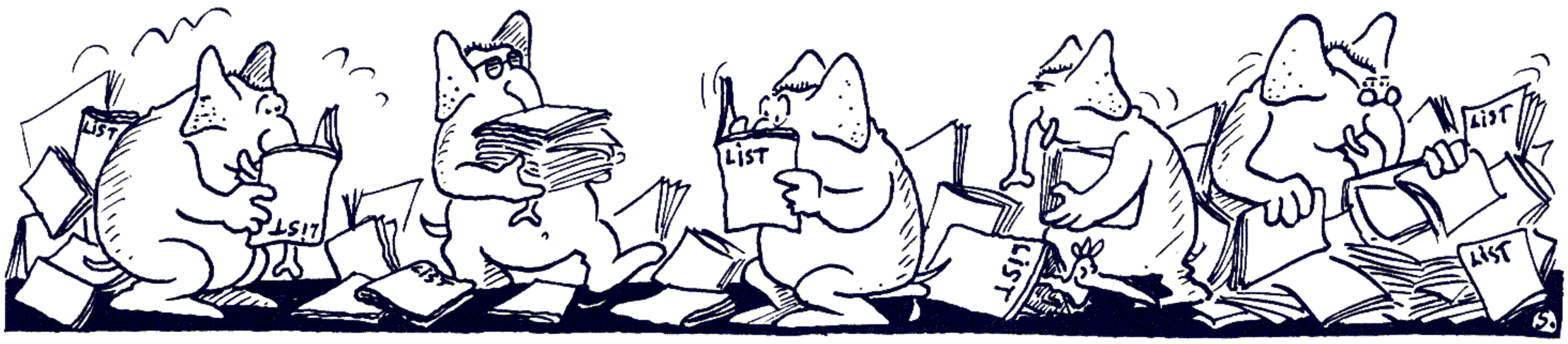
Sur le FX-802P, comme sur d'autres ordinateurs de poche, quand on utilise la fonction KEY (caractère tapé au clavier et lu pendant l'exécution d'un programme), l'écran s'efface jusqu'à ce qu'une touche soit enfoncée. Pour éviter cet effacement, surtout lorsqu'un message est affiché, il suffit de placer la séquence PRINT CSR 0 ; après l'ordre d'impression du message.

Par exemple :

```
95   REM Fin de partie
99   PRINT "Une autre
     partie ?";:PRINT CSR0;
100  IF KEY = "" "THEN 100
110  IF KEY = "N";END
```

Grâce à cette ligne 99, le message "Une autre partie ?" ne sera pas effacé.

Eric BOUCHET



DANS QUEL NUMÉRO ÉTAIT-CE ?

INDEX récapitulatif des articles publiés dans les numéros 6 à 10 de LIST

LIST a maintenant plus d'un an. Sans compter ceux du présent numéro, nous avons déjà publié plus de trois cents articles. Une bonne partie de ces articles nous ont d'ailleurs été proposés par vous, lecteurs. C'est particulièrement vrai concernant ce sympathique bazar aux idées baptisé la boîte à malices.

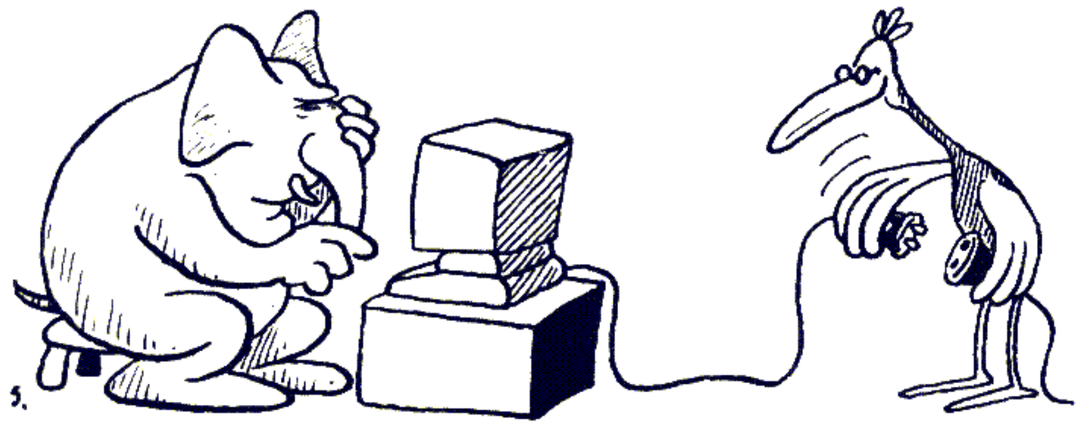
Pour vous aider à retrouver en un clin d'œil les textes et les programmes qui vous intéressent, nous vous proposons aujourd'hui cet index récapitulatif. Pour les cinq premiers numéros, consultez l'index publié dans LIST 6.

Comment fallait-il répertorier ces articles ? Nous avons distingué six grands axes : programmation, système, langages, coups d'œil sur des logiciels et essais de différents Basic. A l'intérieur de ces grandes catégories, nous avons de nouveau classé, le plus souvent par machines ou par langages.

Vous trouverez donc dans les pages qui suivent une sorte de table des matières de LIST.

Maryse GROS
Eliane GUEYLARD

PROGRAMMATION



PROGRAMMATION

BASIC

LANGAGE-MACHINE

UTILITAIRES

PROGRAMMATION

UTILITAIRE

LANGAGE-MACHINE SPÉCIALISÉ

SYSTÈME

ALGORITHME

LANGAGES

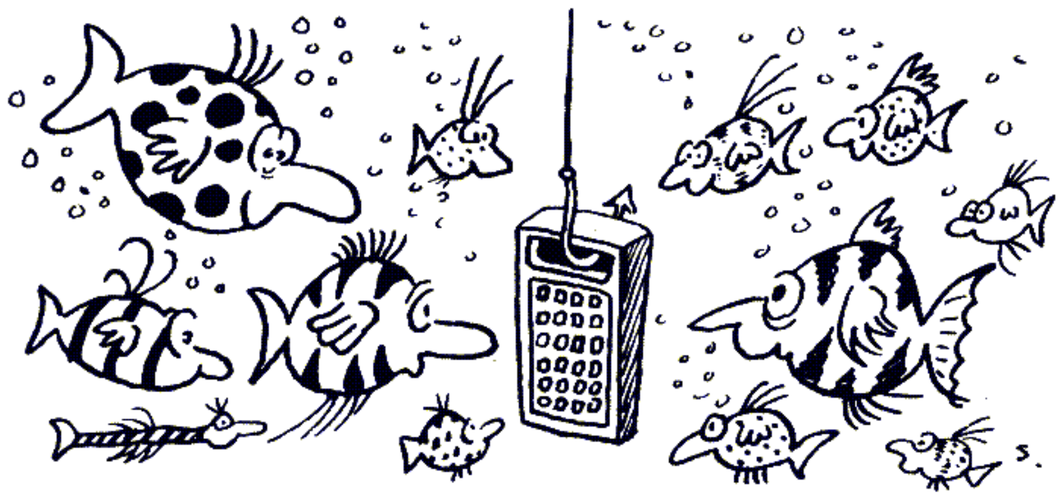
L'HISTOIRE DES LANGAGES

LANGAGES

LES COUPS D'ŒIL DE LIST

LIST A TESTÉ LES BASIC

LES JEUX ET CASSE-TÊTE INFORMATIQUES



LES JEUX ET CASSE-TÊTE INFORMATIQUES

BOÎTE A MALICES (Classement par machines)

BOÎTE A MALICES (Classement par machines)

Imprimerie SIMA, rue des Epinettes, ZI de Torcy. Printed in France. Directeur de la publication : Jean-Luc Verhoye.
Dépôt légal JUIN 1985. Diffusion NMPP. Numéro de commission paritaire : 63617