LIST

LE JOURNAL DES AMATEURS DE PROGRAMMATION

nº 10

ISSN 0761-9936

JUIN 1985

AU PROGRAMME

- Le premier Pascal interprété
- Voyage au centre d'un fichier : l'accès direct
- Faites marcher votre Logo pas à pas
- Les casse-tête informatiques du mois

GROS PLAN SUR TROIS LOGICIELS

- Pour dessiner sur Oric-1 et Atmos
- Un Forth pour la famille MSX
- Base de données sur Dai

LIST A TESTÉ

- Le Basic du Lansay 64 : énorme et structuré
- Et, pour la poche, le PC-1350 de Sharp

M 2712-10-20 F

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

20 F

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 10 - JUIN 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 : *Bernard Allaud, Olivier Arbey, Pierre Barnouin, François J. Bayard, Frédéric Blondiau, Bruno de la Boisserie, Michel Brochand, Laura Campagnet, Jean-Paul Carré, Thierry Chamoret, Stéphane Chiche, Francis Chigot, Thierry Coquard, Walter Costa, Raymonde Coudert, Robert Daguesse, Jacques Deconchat, Denis Descause, Patrick Emin, Augustin Garcia, Jean-Michel Gaudin, Florence Gautier-Louette, Pierre Ladislas Gedo, Olivier Gérard, Laurent Gras, Christian de Guillebon, Max Hagenburger, Renée Koch, Jean-Christophe Krust, Xavier de La Tullaye, Jean-Pierre Lalevée, Bénédicte Lizon, Thierry Lévy Abégnoli, Alain Mariatte, Pierrick Moigneau, Arnaud Peruta, Robert Pulluard, Yvon Pérès, Jean Thiberge, Franck Wettstein.*

Illustrations : *Philippe Burel, Antoine Chereau, Chimulus, Frapar, Bernard Helme, Gilbert Macé, Alain Mangin, Alain Prigent, Nicolas Spinga, Hadi Temglit, 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

QUAND L'AMSTRAD SOUHAITE LA BIENVENUE

DEPUIS peu, je programme sur un Amstrad CPC 464 monochrome vert et je rencontre quelques problème que je ne parviens pas à résoudre. J'espère que vous pourrez m'aider. Les voici :

• si un carré est dessiné au milieu de l'écran, comment faire pour le colorier en vert clair, par exemple ?

• comment empêcher l'ordinateur de continuer à exécuter un programme pendant une commande SOUND ?

• comment charger une cassette en faisant apparaître en même temps un message de bienvenue, autre que les traditionnels « Press play then any key » ou « Loading * Block 1 » ?

Je vous remercie d'avance.

Stéphane PETIT
20 Ajaccio

L'instruction PAINT étant absente du Basic Amstrad, le remplissage d'une figure géométrique ne peut guère se faire qu'avec une série de DRAW qui auront l'effet souhaité. Par exemple, un carré au milieu de l'écran :

```
10 X=200:Y=100
20 FOR I=1 TO 50
30 MOVE X,Y+I
40 DRAW X+50,Y+I
50 NEXT I
```

Une autre solution consisterait à définir une fenêtre ayant la taille et la couleur souhaitées, le dessin étant alors plus rapide, mais l'utilisation en serait moins souple.

Sachant qu'à la rencontre d'une commande SOUND, le Z80 transmet les paramètres au processeur spécialisé, chacun poursuivant son travail séparément, on peut supposer qu'il n'existe pas de moyen simple d'interrompre ce SOUND.

Enfin, pour charger une suite de programmes sans les messages habituels, il suffit de terminer le programme de bienvenue par RUN"!Titre". Le point d'exclamation a pour effet de supprimer l'affichage des messages. « Titre » est le titre réel du programme à charger à la suite : il aura été sauvegardé sous ce même nom (le "!" n'a donc rien à voir avec le titre du programme). La même méthode peut s'appliquer à LOAD, SAVE, OPENIN et OPENOUT.

UNE LIGNE MYSTÉRIEUSE SUR X-07

DANS le programme pour X-07 publié dans *LIST* 6 (page 20), la ligne 9 m'intrigue :

```
9 D=D\3+(A=-2) -(A=2) :IFDTHEN8
```

Premièrement, le signe "\" n'existant pas sur le Canon X-07, de quelle fonction s'agit-il donc ?

Deuxièmement, si A = −2 et A = 2, pourquoi les écrire ainsi ?

Troisièmement, quelle condition impose-t-on à D ?

Veuillez excuser mon ignorance et bien vouloir éclairer ma lanterne. Merci.

C. de BAILLON
68 Illzach

Le signe \ représente la division entière sur le Canon X-07. On l'obtient au clavier par la touche ¥ (Yen).

Pour essayer de vous faire comprendre les mystères de cette ligne 9, nous vous suggérons de faire tourner le programme suivant :

```
10 INPUT A
20 D=(A=-2)-(A=2)
30 PRINT D
```

Selon la valeur donnée à A lors de l'INPUT, celle de D qui s'affichera sera égale à −1, à 1 ou à 0. Ce sera −1 si la valeur introduite dans A est −2, 1 si A = 2 et 0 dans tous les autres cas. En effet, si −2 a été introduit dans A, l'expression (A = −2) de la ligne 20 prend la valeur vraie, soit −1 sur le X-07, et l'expression (A = 2) prend la valeur faux, soit 0. Donc, D = −1 + 0, soit −1. Mais si 2 a été introduit dans A, alors l'expression (A = −2) vaut 0 et (A = 2) vaut −1. Donc, D = 0 − (−1) ce qui fait bien 1.

Enfin, si vous ajoutez à ce programme, les lignes :

```
40 IF D THEN PRINT "D est différent de 0":END
50 PRINT "D=0"
```

vous comprendrez que D tout seul est vrai s'il est différent de zéro (le programme exécute alors ce qui se passe après le THEN), et que D est faux s'il est légal à zéro, auquel cas le programme passe à la ligne suivante (ligne 50). Tout ceci tient en une ligne fort astucieuse dans le programme que vous citez. Ce que c'est que d'optimiser...

AUSSI FULGURANT QU'INFAILLIBLE

EST-ON sûr d'avoir épuisé toutes les ressources de la programmation pour faire d'une machine un adversaire excellent au Tic-tac-toe ? Pour aider à y parvenir, voici quelques réflexions.

A la notion d'alignement, qui n'est guère évidente que pour l'œil humain, on commence par substituer celle de « somme égale à 15 », beaucoup plus accessible à l'ordinateur, en remplaçant le numérotage habituel par celui d'un carré magique d'ordre 3. Par exemple :

```
8 1 6                 1 2 3
3 5 7    au lieu de   4 5 6
4 9 2                 7 8 9
```

Bien entendu, le numérotage « magique » va rester masqué par le programme. On aura ainsi toute latitude pour ajuster la correspondance entre numérotages en fonction du premier coup du joueur, de façon à réduire la « bibliothèque d'ouvertures » à une toute petite pincée de coups « non obligés », commençant tous par 1 − 5, 5 − 8, ou 8 − 5 si le joueur a le trait, par 5 − 1 ou 5 − 8 si c'est l'ordinateur (numérotage magique). Dès le troisième coup de l'ordinateur une routine très simple se charge de tous les coups « obligés » (assurant le gain immédiat ou nécessaire pour éviter de perdre au coup suivant).

Fort utiles au demeurant sur un plan théorique plus général, le programme d'auto-apprentissage de Tic-tac-toe (*LIST* 2) et celui d'exploration d'arbres (*LIST* 3) restent en fait aussi éloignés l'un que l'autre du programme idéal, aussi fulgurant qu'infaillible que j'espère trouver prochainement dans *LIST*.

Charles DOARÉ
06 Grasse

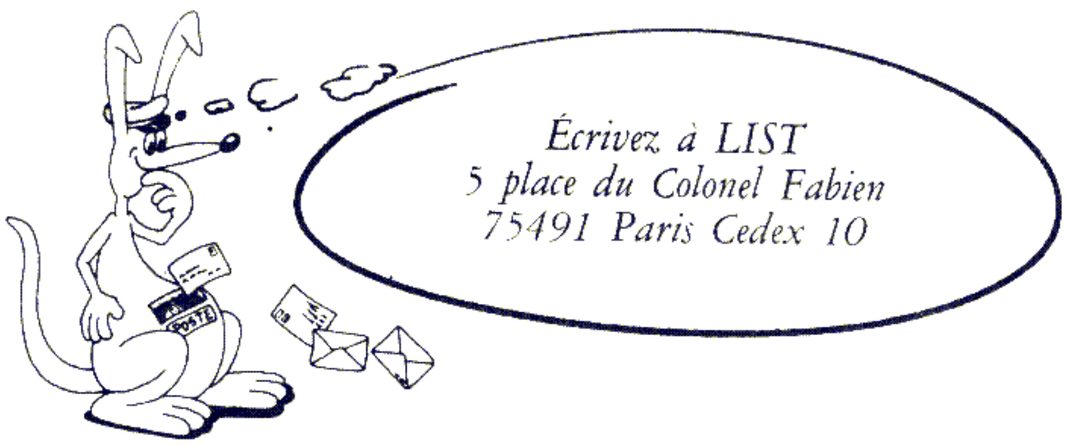
Pour trouver ce programme « idéal, aussi fulgurant qu'infaillible » dans LIST, *il suffit de nous l'envoyer. Nous l'attendons avec impatience.*

DE MEILLEURES SOLUTIONS POUR LES JEUX

VOUS connaissez certainement le principe qui régit la programmation : « Tout programme convenablement testé et débogué contient encore au moins une erreur ». Les extraits de programme proposés dans la rubrique des jeux et casse-tête informatiques, même s'ils ne dépassent pas quelques lignes, n'échappent pas, bien sûr, à cette règle.

Pour s'en convaincre, il suffit de considérer l'énoncé du jeu 27, *Recherche du DU* (*LIST* 8, page 83), qui a provoqué de nombreuses réactions. En effet, si le jeu consistait à enlever la petite « paille » qui empêchait le programme de donner le bon résultat dans un cas précis, il n'était fait aucune mention de la « poutre », comme nous l'écrit Pierre Barnouin, qui empêchait le programme de tourner dans tous les cas. Bien entendu, c'est à la suite d'une correction de dernière minute que des OR ont pris la place des AND et ont rendu le jeu plus difficile que prévu.

Du côté des fonctions logiques, Hervé Gouessant (Vandœuvre)



est, lui, plus à l'aise que nous. C'est ainsi qu'il s'est attaché à généraliser le jeu 19 (*LIST* 7, page 80) qui consistait à trouver des formules pour remplacer les fonctions logiques. Il est possible de trouver, pour chaque opérateur, deux expressions de remplacement, l'une utilisant les opérateurs relationnels (=, >, <=,...), l'autre mettant en œuvre les opérateurs arithmétiques (+, −, *). Les résultats sont récapitulés dans le tableau ci-dessous.

Tableau de remplacement des fonctions logiques

Opération logique	Expression avec des opérateurs relationnels	Expression avec des opérateurs arithmétiques
non A	A = 0	1 − A
A et B	A + B = 2	A * B
A ou B	A + B > 0	A + B − A * B
A ox B	A <> B	A+B − 2*A*B
A equ B	A = B	1−(A+B − 2*A*B)
A imp B	A <= B	1 − A * (1−B)
A rpc B	A >= B	1 − B * (1−A)

Pour le jeu 21 (*LIST* 7, page 81) qui consiste à rechercher le plus grand élément d'un tableau, Pierre Barnouin remarque qu'il est possible de gagner une instruction si le tableau est indicé de 0 à 9. Le programme Basic devient alors :

```
10 for I=1 to 9
20 if T(I)>T(J) then J=I
30 next I
40 print T(J)
```

Le jeu 24 (*LIST* 8, page 82) consistait à écrire un programme permettant de calculer le bit de parité d'un octet. Pierre Barnouin, toujours, nous propose les deux lignes suivantes :

```
10 def fn K(Y) = sgn(Y mod 3)*sgn(Y mod 5)
20 def fn P(X) = fn K(X div 16) <> fn K(X mod 16)
```

Ainsi P est la fonction qui retourne la valeur du bit de parité de l'octet qui est passé en paramètre. Ici, SGN est la fonction signe qui vaut −1 si son paramètre est négatif, 0 s'il est nul, +1 s'il est positif ; DIV donne le résultat de la division entière et MOD le reste de cette division.

Cette solution qui est à la fois courte et élégante, est particulièrement difficile à comprendre. Voici donc quelques explications sur la manière dont elle fonctionne.

Prenons toutes les valeurs qui peuvent être mémorisées sur un demi-octet, c'est-à-dire tous les entiers de 0 à 15. Séparons ces valeurs en fonction de la parité du nombre de bits à 1. On obtient le tableau :

Nombre pair de bits à 1		Nombre impair de bits à 1	
0	(0000)	1	(0001)
3	(0011)	2	(0010)
5	(0101)	4	(0100)
6	(0110)	7	(0111)
9	(1001)	8	(1000)
10	(1010)	11	(1011)
12	(1100)	13	(1101)
15	(1111)	14	(1110)

On peut ainsi remarquer que les valeurs de la colonne de gauche sont toutes divisibles soit par trois soit par cinq, et que celles de la colonne de droite ne le sont jamais. Ainsi, le reste de la division par 3 ou par 5 des nombres de la colonne de droite donnera toujours un résultat différent de 0. Et la fonction K qui retourne la valeur de l'expression « sgn(Y mod 3)*sgn(Y mod 5) » permettra de déterminer la parité d'un demi-octet.

Pour avoir la parité d'un octet, on le divise par 16 : on obtient ainsi les 4 premiers bits, ceux dits de poids fort. Le reste de cette division par 16 donne les 4 derniers bits, ceux dits de poids faible. Ainsi, l'octet est séparé en deux demi-octets. Et si les deux demi-octets ont la même parité, alors le nombre de bits de l'octet sera pair. Dans le cas contraire, ce nombre sera impair. Ce résultat est donné par la fonction P qui évalue l'expression : fn K(X div 16) <> fn K(X mod 16).

Avouons que si cette solution est particulièrement complexe, elle est d'une élégance rare. ■

INDEX DES ANNONCEURS

LA GAZETTE DE LIST

Sony lance sur le marché un MSX francisé

SONY fait donc le pari du standard MSX en France. Après avoir commercialisé avec succès son ordinateur au Japon, il l'introduit en Europe et s'attaque au marché français de la micro-informatique familiale. Le *Hit Bit 75 F*, nouveau modèle MSX, présente bien. Il vaut 3500 F.

La mémoire vive de l'unité centrale est de 64 Ko auxquels s'ajoutent 16 Ko pour la gestion de l'écran. Inutile de décrire la plupart des autres caractéristiques qui découlent en fait du standard (voir l'essai du Basic MSX, *LIST* 5).

Deux améliorations très nettes cependant pour les francophones : le clavier, tout d'abord, est azerty avec caractères accentués. Du côté de la mémoire morte, d'autre part, le Sony se singularise avec un logiciel intégré entièrement francisé. A la mise sous tension, un menu apparaît, donnant le choix entre l'agenda, le carnet d'adresses, le bloc-memo ou le Basic. Les trois premières options possèdent des fonctions empruntées aux logiciels de gestion de fichiers (création, effacement, modification, sauvegarde, tri) que l'on sélectionne par l'intermédiaire du pavé de gestion du curseur.

La compatibilité matérielle étant l'un des points forts du standard, tous les utilisateurs de MSX seront intéressés par la commercialisation des périphériques du Hit Bit. Le plus original d'entre eux est sans doute le *track ball* qui coûte environ 900 FF et qui ne fonctionne pour l'instant qu'avec un seul logiciel, *Creative Graphics*. Il s'agit d'une souris « ventre en l'air » (la boule qui déplace le curseur est tournée vers le haut et actionnée par la main), livrée avec un programme de création graphique en cartouche. Ce track ball permet d'accéder aux icônes du logiciel et bien sûr de dessiner.

Des manettes de jeu sont disponibles en deux versions : avec cordon (environ 250 FF), à infrarouge (environ 600 FF). La table traçante, 2600 FF, accepte des feuilles de papier au format 21 × 29,7. Elle écrit en quatre couleurs et sa vitesse d'impression moyenne est de 6 caractères à la seconde. Une cartouche de mémoire vive de 4 Ko, à alimentation intégrée, peut être utilisée comme petite mémoire de masse (notamment pour le logiciel intégré). On devrait la trouver aux alentours de 500 FF.

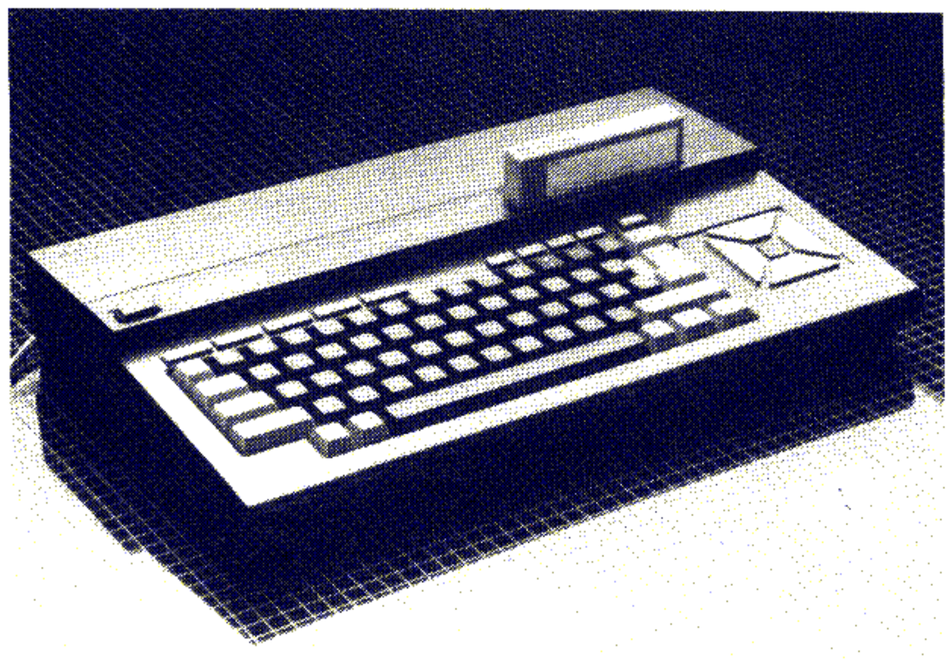
Enfin, un lecteur de disquettes trois pouces et demi et un magnétophone à cassettes, qui coûtent respectivement 3500 FF et 600 FF.

TLA ■

Lecteurs de disquettes ◀

Une grosse souris : le track ball ▶

Le clavier du Hit Bit ▼



Tandy baisse en France le prix du Coco 2

APRÈS le Spectrum Plus et l'Alice 32 Ko (*LIST* 9, page 23), c'est au tour du Coco 2 de réviser ses prix, à la baisse heureusement.

Au début de sa carrière, il valait 1895 FF dans sa version de base (16 Ko de MEV et Basic standard) et 3295 FF avec 64 Ko de MEV et un Basic étendu. Puis les deux versions du même TRS Couleurs 2 ont valu respectivement 1695 et 2895 FF.

Une nouvelle baisse vient de survenir ; la version 16 Ko vaut maintenant 1395 FF et le 64 Ko vaut 2695 FF. ■

UNE CASSETTE

Graphix 81
Cassette pour ZX 81
(extension 16 Ko minimum)
Auteurs : Joël Berthelin et Philippe Fassier
Édité par Ère Informatique
Prix : environ 150 FF

SI votre ZX 81 ne quitte presque plus votre placard, si vous en êtes revenu, si vous pensez en avoir fait le tour, l'utilitaire de Joël Berthelin et de Philippe Fassier pourrait bien vous réserver une petite surprise.

Tout le programme (4 Ko de langage-machine) tient en une REM, en ligne 0. Il n'en faut pas plus pour que le ZX fonctionne en haute résolution graphique : 192 × 256, soit 49152 points, et donc 16 fois plus que les 3072 points d'origine. D'ailleurs, les gros pâtés sont toujours là, mais en mode basse résolution.

En contrepartie, c'est vrai, l'utilisation de Graphix n'est pas vraiment simple. Et si l'on veut modifier un vieux logiciel, cela ne se fait pas en un clin d'œil, il faut prendre son courage à deux mains : RAND USR machin, RAND USR truc, etc...

Tout est expliqué, sur 52 pages, dans une notice soignée mais pas toujours évidente pour le profane. C'est assez ardu, mais complet : chargement, introduction, tableau de la mémoire, choix des configurations, et tout le reste, c'est-à-dire les nouvelles instructions graphiques. En fait, on utilise astucieusement les REM et certains ordres déjà existants (THEN, INKEY$, AT, NOT, TO, PI...).

Et voilà le ZX qui comprend plusieurs instructions par ligne, le voilà qui se met à tirer des traits, à tracer des cercles, à dérouler son affichage. Voilà qu'il admet des chaînes de vecteurs comme les machines MSX. On peut encore définir une fenêtre, remplir une zone fermée,

redéfinir ses caractères, faire une copie d'écran haute résolution...

Mais attention : tout cela ne se fait pas comme par enchantement. Si ce logiciel est remarquable, il est assez difficile à utiliser. On arrive certes à des résultats impressionnants pour un ZX, mais on doit s'attendre à de patients essais et à une lecture très attentive du mode d'emploi avant d'être à son aise.

Deuxième restriction : le logiciel peut fonctionner sous plusieurs configurations différentes, depuis la banale extension 16 Ko (minimum indispensable) jusqu'à celle de 64 Ko. Et même le possesseur de cette dernière extension risque de ne pas pouvoir disposer de toutes les possibilités du logiciel. Le manuel indique un test qui permet de savoir quelles sont les nouvelles fonctions apportées par Graphix.

On devra tenir compte aussi des limitations techniques liées à l'appareil : les caractères utilisés pour la haute résolution sont évidemment codés en mémoire, or il n'est pas possible de définir le jeu complet des 256 caractères nécessaires. Résultat, sur l'écran, la haute résolution n'est pas parfaite (elle peut l'être en revanche sur l'imprimante).

Malgré ces limitations et la difficulté de mise en œuvre, Graphix est un programme très attrayant, il transforme complètement le ZX 81. Les fanatiques de cette machine doivent au moins essayer ce remarquable logiciel.

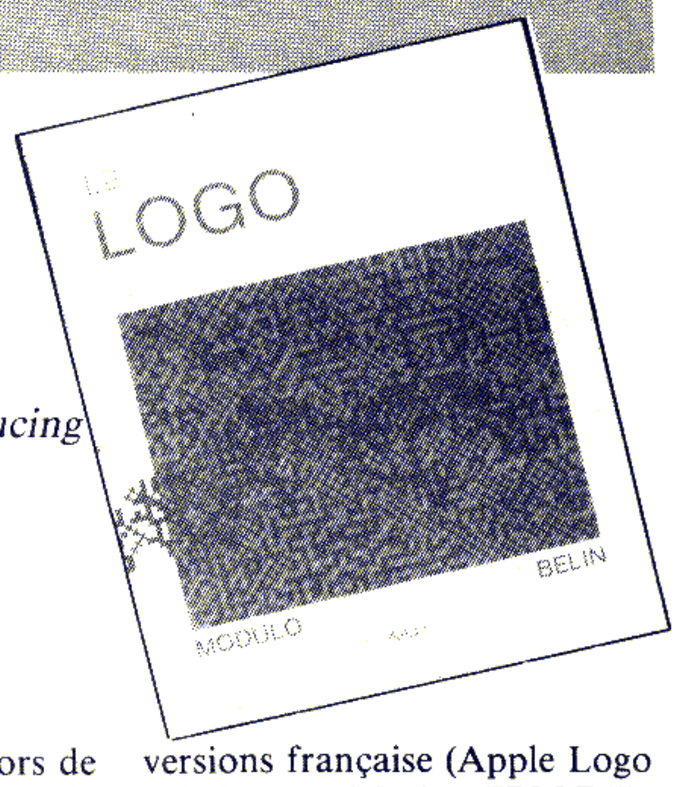
JD ■

32 % de moins sur l'Atari 800 XL

ATARI a récemment modifié le prix de vente de son modèle 800 XL (64 Ko de mémoire vive). On le trouve désormais à 1700 FF au lieu de 2500 FF.

Quant au 600 XL (16 Ko de MEV), dont la fabrication a cessé, il est définitivement retiré du catalogue Atari. ■

UN LIVRE



Le Logo
Boris Allan
Titre original : *Introducing Logo*
Éditions Belin
Collection Modulo
Paris, 1984
Broché, 110 pages
Prix : 95 FF

VOICI un livre qui, lors de sa traduction, a changé de visage. Le titre français lui donne en effet une prétention que l'auteur n'a pas. Ce n'est pas "Le" Logo, mais bel et bien une introduction à Logo. Le ton du livre est donné par la préface : « Tout adepte de Logo s'oppose naturellement à l'utilisation injustifiée des jargons et à toute tendance visant à faire de l'informatique un domaine à part. »

Fruit d'une collaboration franco-québécoise, les exemples de procédures sont donnés en versions française (Apple Logo LCSI) et américaine (IBM PC). Feuilleté pour la première fois, le livre paraît dense. En fait, il demande à être lu dès la première ligne, et c'est de cette façon qu'il devient alors vivant, puis de plus en plus technique.

Le chapitre 9, « Programmation élaborée », pèche un peu par son titre, me semble-t-il : Eliza est amusant et peut donner des idées, c'est tout. Mais ici aussi, il convient de se rappeler du titre original de l'ouvrage : « Introducing Logo ».

Il est dommage que Boris

LA GAZETTE DE LIST

Allan paraisse ignorer les travaux Logo menés ailleurs qu'en Angleterre et aux USA, mais il reste que cette introduction au Logo, langage de programmation, devrait concurrencer beaucoup de livres écrits directement en français pour Logo. Si l'on ne tient pas compte du côté *voyez comme Logo est simple* et de la légèreté des touches pédagogiques, on y trouvera à la fois des idées de progressions dans l'enseignement du Logo et des exemples de procédures qui, je l'espère, en feront germer d'autres.

RD ■

Un clavier Azerty pour le Sinclair QL

LE QL à clavier français, présenté au Sicob, est déjà en vente au prix de 6950 FF. Un prototype disponible fin avril 1985 nous avait donné une idée de ce que serait cette version Azerty. Je parle de prototype car ce matériel n'était pas équipé des mémoires mortes masquées de la version française. Le « driver » de clavier avait pris place dans des Eproms installées

sur une carte d'extension raccordée au connecteur de bus. Le clavier, cependant, présentait son aspect définitif.

La différence avec le modèle anglais réside dans la disposition de ses touches, Azerty bien sûr, et dans l'ajout du c cédille et des caractères accentués. Pour ne pas devoir redéfinir entièrement le clavier, Sinclair a choisi de mettre ces nouvelles touches à la place de caractères peu usités (qui restent tout de même accessibles avec Contrôle). Ce nouveau clavier n'est donc pas la réplique exacte de celui d'une machine à écrire. Les accents ne se trouvent pas sur la rangée numérique, ils sont à droite, du côté de la touche Enter. Une telle disposition ne doit vraiment être gênante que lors de l'utilisation du traitement de texte et dans ce domaine, le clavier du QL se montre déjà assez discutable. Cela mis à part, l'emplacement des accents change peu de choses et on s'y habitue plus facilement qu'au toucher un peu spécial du clavier.

Avec ce prototype Azerty nous ont été confiées les versions françaises des quatre logiciels fournis avec la machine. Le numéro de version 2.2 inclut toutes les améliorations apportées à la 2.0 (essentiellement perceptibles par une réduction des accès disques) avec bien sûr une traduction des messages affichés à l'écran (commandes et aide). Le travail apparaît bien fait et ces logiciels n'en deviennent que plus intéressants. Il me reste un motif d'inquiétude : ces versions françaises refusent de tourner sur le modèle Qwerty du QL. Cela signifie très probablement qu'il faudra transformer en Azerty les ordinateurs qui auraient été achetés en version anglaise. Sinon, pas de français à l'écran.

XdLT ■

Un "Kit-programmeur" pour Oric-1 et Atmos

COBRA Soft propose aux possesseurs d'Oric, un "Kit-programmeur" composé de cinq logiciels utilitaires et de l'ouvrage "Au cœur de l'Atmos" de Gilles Bertin, le tout pour 290 FF, dans la limite des stocks disponibles (prix initial : 635 FF).

Trois de ces logiciels, Kit écran, Data Save et Supercopy, ont été présentés dans *LIST* n° 9, page 15. Ils sont accompagnés d'un utilitaire de création d'écrans graphiques Haute Résolution (D.A.O.) et d'un générateur de caractères (Caractor).

Cobra Soft
5 avenue Monnot
71100 Châlon sur Saône
Tél. : (85) 93 34 82

CASSETTES ET DISQUETTES

Bêta Basic
Basic étendu
Cassette pour
ZX Spectrum 48 Ko
Édité par Infogrames
Prix : 190 FF

CE Basic étendu gonfle le Spectrum 48 Ko de près de soixante commandes et fonctions supplémentaires. La plupart d'entre elles sont nouvelles, d'autres sont des fonctions usuelles améliorées. Les explications concernant leur utilisation sont consignées dans un fascicule d'une quarantaine de pages joint à la cassette. ■

Tablo 5
Tableur
Tracés d'histogrammes
Cassette pour MO 5
Édité par Ère Informatique
Prix : 250 FF

UN logiciel qui permet de disposer d'une feuille de calcul électronique de 26 colonnes sur 71 lignes. Les données introduites pourront ensuite être affichées et imprimées sous forme de graphiques. Trois représentations sont possibles : la courbe, le découpage en fromage et l'histogramme. ■

D.A.O.
Logiciel graphique
Cassette pour Amstrad
Édité par Cobra Soft
Prix : 120 FF

CE logiciel de D.A.O. — Dessin Assisté par Ordinateur — fourni avec une notice très succincte (une feuille 15 × 21 cm glissée dans la cassette), permet de créer et de sauvegarder des dessins en couleurs qui pourront être réutilisés à l'intérieur d'autres programmes. ■

Édit-plus
Éditeur pleine page
Basic francisé en option
Utilitaires
Cassette pour Oric-1 48 Ko
et Atmos
Édité par Isosoft
Prix : 195 FF

LE programme chargé, le menu s'affiche et propose huit options : l'éditeur, la numérotation automatique des lignes, un utilitaire de renumérotation, une fonction de recherche qui permet de remplacer un mot par un autre, la possibilité de compacter les lignes, de supprimer les REM, de lister le programme et... de programmer en Basic français. Dans les dernières

pages de la notice se trouve une table de francisation des mots Basic que l'utilisateur pourra modifier et augmenter afin de créer sa propre traduction. ■

LM Plus
Compilateur Basic
Cassette pour Oric-1 et Atmos
Édité par Isosoft
Prix : 250 FF

POUR obtenir des exécutions beaucoup plus rapides sans connaître le langage-machine, l'une des solutions consiste à utiliser l'aide d'un compilateur. *LM Plus* rend les programmes jusqu'à sept fois plus rapides et sa présence en mémoire n'est pas nécessaire à l'exécution des programmes compilés. Notons qu'il ne travaille que sur les nombres entiers. ■

Deux assembleurs pour Oric et MSX

As des As
Éditeur, assembleur,
désassembleur
Cassette pour Oric-1 et Atmos
Édité par Isosoft
Prix : 350 FF

LA notice, comportant une dizaine de pages, indique que le logiciel s'adresse en priorité aux programmeurs qui connaissent déjà l'assembleur du 6502. Les débutants en la matière devront auparavant consulter un bon ouvrage d'initiation à la programmation de ce microprocesseur. L'*As des as* occupe environ 6 Ko en mémoire. ■

Odin
Macro-assembleur, éditeur,
désassembleur
Cassette pour MSX
Édité par Loriciels
Prix : 295 FF

UNE bonne connaissance de l'assembleur du Z 80 sera ici aussi nécessaire, mais il existe de nombreux livres sur le sujet. Une notice de vingt pages est fournie avec le logiciel qui se compose d'un éditeur-assembleur (place occupée en mémoire vive : environ 6 Ko) et d'un moniteur-désassembleur (8 Ko de MEM). ■

Un nouveau poquette chez Casio

IL a la forme du PB-100, il en a l'aspect, mais il s'agit d'un produit nouveau. Chez Casio, le dernier ordinateur de poche s'appelle PB-410. Il contient une carte interchangeable de mémoire vive et continue, un Basic résident, et surtout une « data bank », c'est-à-dire une petite banque de données intégrée, ce qui permet d'envisager des applications peu courantes jusqu'à présent sur des appareils de taille aussi réduite.

S'il est vraisemblable que les ordinateurs de poche ne sont plus aussi compétitifs qu'avant comme appareils d'initiation, ils n'en conservent pas moins des atouts sérieux (réelle autonomie, mémoire permanente, format calepin) qui les rendent parfois irremplaçables pour des utilisations sur le terrain. De ce point de vue, le PB-410 et sa « data bank » incorporée méritent l'attention.

Le principe de fonctionnement est très simple : il s'agit d'un stockage de données alphanumériques sous forme d'enregistrements en mémoire. Chaque enregistrement, numéroté, contient jusqu'à 62 caractères. Il est divisé en différents articles séparés par une virgule. Si l'on veut, par exemple, se constituer un répertoire, on consignera dans chaque enregistrement un nom, un prénom, un numéro de téléphone et une adresse.

On peut, sans la moindre notion de Basic, exploiter les données enregistrées au moyen de la touche « mémo ». L'appareil se comporte alors comme un répertoire électronique permettant, entre autres opérations, la recherche d'un article ou d'un enregistrement (recherche normale, rapide, sélective ou conditionnelle). Ainsi, dans le cas où les données sont des noms et des numéros de téléphone, en tapant un nom, on obtient en réponse le numéro correspondant. On peut également rechercher tous les noms commençant par la même initiale, ou tous ceux dont le numéro associé commence par 875, ou encore les enregistrements qui répondent à ces deux conditions. Par ailleurs, il est possible de corriger les enregistrements (créer, détruire, modifier). Toutes ces fonctions sont appelées directement du clavier.

Si l'on utilise maintenant cette banque de données par l'intermédiaire d'un programme Basic, qu'obtient-on ? Grâce à un jeu d'instructions spéciales (READ#, LIST#, WRITE#, RESTORE#, etc.), les deux systèmes communiquent entre eux. Le répertoire

électronique se transforme alors en un fichier de données exploitables par le programme. De nouvelles opérations deviennent possibles : sorties sur imprimante, sauvegarde sur cassette, tri, protection par un mot de passe... On peut également se faciliter les choses avec un programme qui pose les questions utiles à l'entrée ou à la sortie des données. Les applications ne manquent pas : agenda, horaire, tarif, etc., le tout sous la main, avec un programme qui permet d'accéder facilement à la donnée recherchée.

Comme sur la plupart des ordinateurs de poche, la capacité de la mémoire est réduite. Pour pallier en partie cette limitation, Casio a opté pour la carte de mémoire vive interchangeable. Il en existe deux modèles : 2 ou 4 Ko, en fait 1568 ou 3616 pas de programme. Une petite pile au lithium confère à cette mémoire un caractère permanent, et l'on peut donc avoir ainsi à sa disposition plusieurs unités de fichiers et de programmes. Cette forme de mémoire de masse est très pratique, mais elle reste relativement onéreuse : 415 FF ttc pour la carte de 2 Ko et 600 FF pour celle de 4 Ko. Le PB-410 dans sa version de base est livré avec une carte de 2 Ko et vaut environ 900 FF.

Dernière question : si l'on oublie la petite banque de données et ses commandes spécifiques, que reste-t-il ? Un PB-100 standard ? Pas vraiment. S'il y a bien compatibilité entre les programmes du PB-100 qui tournent sans modification sur le PB-410, le Basic de ce dernier comporte quelques améliorations non négligeables. C'est ainsi que nous trouvons BEEP (le PB-100 était muet), PASS, READ/DATA/RESTORE, ON/GOTO, ON/GOSUB et REM. Une bonne chose aussi pour les calculs horaires : les fonctions de conversion sexagésimal-décimal DEG et DMS$ sont apparues. Enfin, quelques mots clés ont été modifiés. L'ordre VAC par exemple (effacement général du contenu des variables) est devenu CLEAR. Dans l'ensemble, cette évolution vers un Basic plus standard facilitera l'adaptation des programmes conçus pour d'autres matériels.

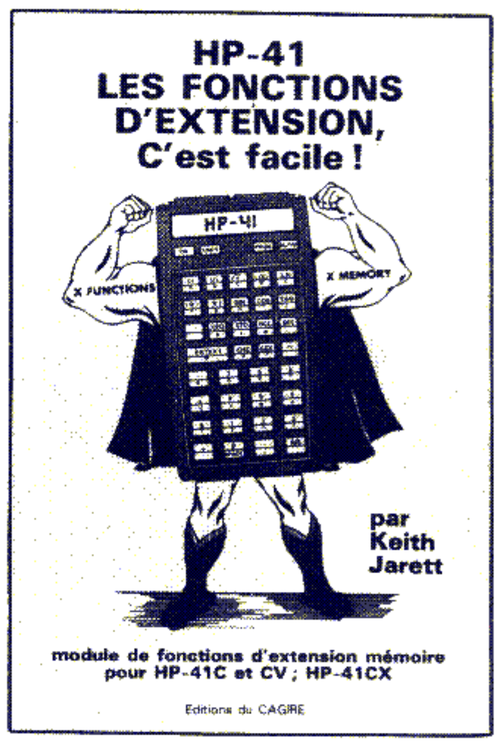
PM ■

DEUX LIVRES

La programmation synthétique, c'est facile
Keith Jarett
Traduit de l'américain par Gilles Barret
Éditions du Cagire
Toulouse, 1984
Broché, 128 pages
Prix : 150 FF

Les fonctions d'extension, c'est facile
Keith Jarett
Traduit de l'américain par M.-D. Dodin
Éditions du Cagire
Toulouse, 1984
Broché, 188 pages
Prix : 150 FF

LE premier de ces deux ouvrages (qui sont exclusivement consacrés à la HP-41) est un panorama des dernières découvertes en programmation synthétique. On y trouve, en particulier, un exposé des instructions synthétiques les plus courantes, comment les générer — d'abord au coup par coup, puis toutes à la fois — et comment les assigner. L'utilisateur de la 41 pourra lire également une étude de la fonction F0 qui permet de comprendre le mécanisme de l'édition des programmes. Une carte détaillée de la mémoire et quelques applications pratiques (mesure de la durée d'exécution des programmes, par exemple) complètent l'ensemble. Signalons pour finir qu'une large place est faite au PPC ROM produit par et pour le PPC US.



Le second ouvrage est entièrement consacré aux fonctions d'extension. Si la programmation synthétique y est abordée dans un chapitre, c'est appliquée aux fonctions étendues. L'auteur traite longuement des bogues du module X-Fonctions. Elles ont été corrigées sur les modèles les plus récents, mais des parades sont données pour les versions antérieures.

Les fonctions d'extension sont traitées par grandes classes : fichiers de programmes, de données, de textes et fonctions diverses. Parmi les applications pratiques développées figure un traitement de texte destiné aux utilisateurs qui, n'ayant pas la version CX de la 41, ne disposent pas de la fonction ED.

En annexe, dans les deux livres, on trouvera tous les codes-barres correspondant aux programmes commentés. Les traductions, assez littérales, donnent un style un peu monotone à l'ensemble, mais cela n'empêchera pas les fanatiques de la HP-41 de dévorer ces livres.

OA ■

Atari et le Sicob

En attendant la bombe d'Atari, le 520 ST, voici le 130 XE déjà disponible.

UN absent de taille au Sicob : Atari. Et pourtant, deux produits nouveaux de ce constructeur étaient là. Le premier, l'Atari 130 XE (présent sur les stands de Galaxie Distribution et de Codewriter International) avec 24 Ko de mémoire morte, le même Basic que l'Atari 800 XL mais une mémoire vive deux fois plus importante (128 Ko) serait disponible dès le mois de juin. Il devrait coûter 2300 FF ttc dans sa version Pal.

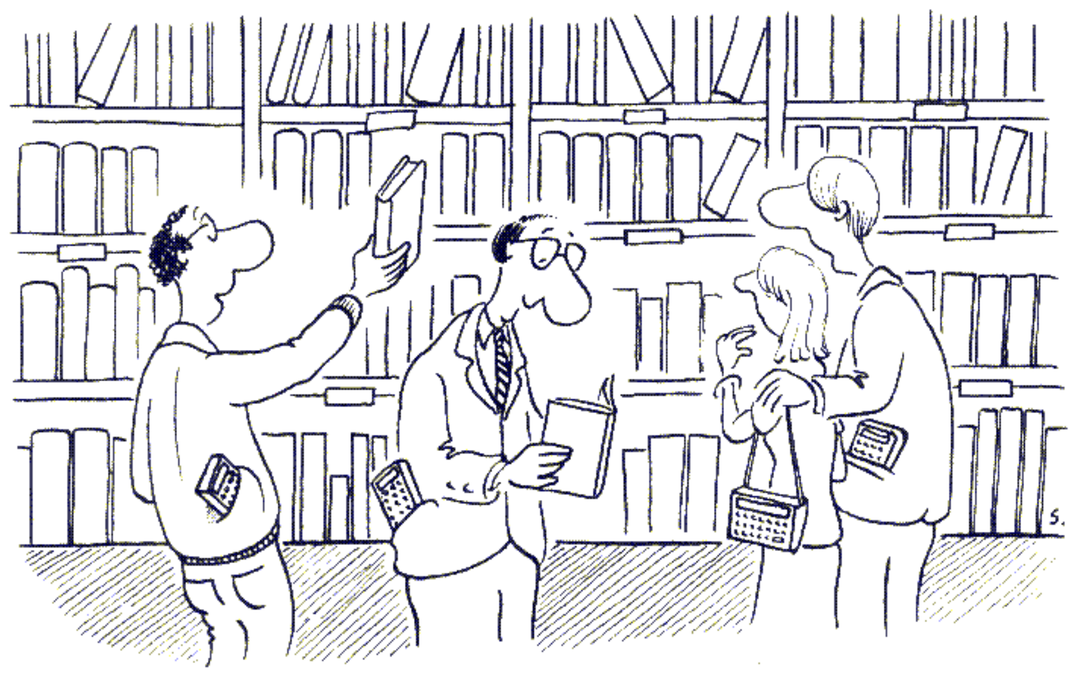
Un deuxième Atari se trouvait sur les stands de Galaxie Distribution et de Micro Application : le 520 ST, surnommé aussi « Jackintosh ». Si l'on sait qu'il doit concurrencer le Macintosh d'Apple, avec 512 Ko, de la couleur, le logiciel GEM lui donnant des possibilités analogues à celles de Macintosh et un prix nettement inférieur, on ne sait toujours pas quand il sera disponible en France...

■

L'Apple IIe change de microprocesseur

LE microprocesseur de l'Apple IIe, le 6502, est remplacé par le NCR 65C02, microprocesseur à technologie C-MOS qui se trouve déjà sur l'Apple IIc. Ainsi, l'Apple IIe deviendrait, selon Apple, « 100 % compatible avec les logiciels de l'Apple IIc » et ne serait plus compatible qu'à « 90 % avec les logiciels développés auparavant sur l'Apple IIe ancienne version ». ■

UN PETIT TOUR CHEZ LE LIBRAIRE



Clefs pour MO5
Gilles Blanchard
Éditions du PSI
1985, 146 pages
Reliure spirale
Prix : 120 FF

COMME les autres titres de la collection « Clefs pour... », cet ouvrage n'est pas un manuel d'initiation : l'information y est plus importante que le style. Ce mémento d'utilisation du Basic et du langage-machine du MO 5 est destiné aux programmeurs. Un livre qui reste à côté de l'ordinateur et qui permet de trouver rapidement les réponses aux problèmes que peut poser l'écriture d'un programme. ■

Listes et tableaux numériques en Basic
Henri Hunic
Éditions Techniques et Scientifiques Françaises
Paris, 1985
Broché, 128 pages
Prix : 85 FF

DANS les programmes Basic, on est souvent amené à traiter des listes et des tableaux numériques. Dans quels cas et comment ? Ce livre tente de répondre à ces questions uniquement avec des exemples d'application. On n'y trouve pas forcément la solution à son problème particulier, mais au moins un chemin qui mène à cette solution. ■

L'ordinateur de poche
Luc Smeesters
Éditions Marabout
Paris, 1985
Broché, 192 pages
Prix : 25 FF

CONSACRÉ uniquement aux ordinateurs de poche programmables en Basic, ce livre est censé guider le futur utilisateur dans son choix : il expose ce que l'on peut faire avec un ordinateur de poche, présente la programmation Basic, et enfin donne une description détaillée de chaque matériel et de son Basic (HP-71B, Canon X-07, PC-1500A/1260/1245/1401, PB-700/110, FX-750P/720P). Reste donc à choisir. Les prix sont absents, mais pas les adresses des importateurs dans les pays francophones. ■

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

CET ouvrage devrait permettre au nouvel acquéreur d'un Amstrad d'approfondir ses connaissances de façon théorique et pratique. Les auteurs y décrivent l'utilisation des instructions et s'attachent surtout à apporter au programmeur des idées et des exemples d'application. ■

Microprocesseurs 16 bits
Michel Aumiaux
Éditions Masson
1985, 220 pages
Prix : 150 FF

CET ouvrage décrit plus particulièrement deux microprocesseurs, le 8086 et le 68000. L'étude de ces 16 bits, qui ont des possibilités comparables à celles des processeurs des mini-ordinateurs, est d'un abord relativement difficile. C'est pourquoi le lecteur potentiel devra déjà bien connaître la structure et le fonctionnement des microprocesseurs 8 bits pour comprendre ce livre. De fait, il s'agit bien plus d'une suite de remarques que d'un ouvrage vraiment didactique qui ferait découvrir les 16 bits comme une « histoire ». ■

Guide pratique du Sinclair QL
Eric Tenin et Jean-Manuel Van Thong
Éditions Édimicro
1985, 192 pages
Prix : 135 FF

Programmes sur Sinclair QL
Cassette contenant les exemples du livre
Prix : 110 FF

CONFORME à l'idée qu'en donne son titre, ce livre guidera les premiers pas de l'utilisateur peu attiré par le manuel du QL. Il débute sur une présentation générale de l'ordinateur, entrées/sorties et extensions pour aborder ensuite les éléments indispensables à la mise au point et à la sauvegarde des programmes. Le Super Basic est présenté de façon assez précise (ses instructions, ses fonctions, ses procédures et leurs applications, le concept de récursion, etc.) ainsi que les possibilités graphiques du QL, exemples à l'appui. ■

Amstrad lance le CPC 664

JUSQUE-LA, on parlait de l'*Amstrad* quand on voulait nommer le CPC 464. Dorénavant, il faudra faire la différence entre le CPC 464 et le nouveau CPC 664. Pour les distinguer physiquement, c'est simple : à la place du lecteur de cassettes du

CPC 464, se trouve un lecteur de disquettes (3 pouces) sur le CPC 664. Cet ordinateur est livré avec un moniteur monochrome ou couleur, 64 Ko de mémoire vive (dont 41 disponibles à l'utilisateur), un processeur Z80A, et un Basic plus riche que celui du 464. Le CPC 664 est vendu 4490 FF avec un moniteur monochrome vert, et 5990 FF avec un moniteur couleur. Un second lecteur de disquette est disponible au prix de 1990 FF. ■

Nouveau TO

UN nouvel ordinateur Thomson devrait apparaître à l'automne 1985 : le TO9. Cette machine serait, d'après Thomson, « familiale et semi-professionnelle », ou encore « familiale haut de gamme ». A part cela, Thomson reste étrangement avare d'informations. ■

Nos copains de L'Ordinateur Individuel

UN numéro hors série de notre confrère L'Ordinateur Individuel est entièrement consacré au Commodore 64 : périphériques, langage-machine, Pascal sur Commodore, graphisme, programmes de jeux, etc. Mais surtout, on y trouve un véritable roman policier qui ▶

LA GAZETTE DE LIST

▶ nous fait saisir, en vingt pages (avec quel suspens !), les secrets de la mémoire vive du C.64. Ce numéro *Spécial Commodore* est en vente dans les kiosques, au prix de 35 FF. ■

Les 24 heures de l'informatique : Champions toutes catégories, deux amateurs

LES épreuves de ces premières 24 heures de l'informatique se sont déroulées les samedi 4 et dimanche 5 mai 1985 dans les locaux et sous l'égide du Centre de Bureautique et Informatique (CBI).

Cent candidats sélectionnés à partir de 1500 dossiers se regroupaient en équipes de deux personnes. Répartis en deux catégories (professionnels ou amateurs), les concurrents devaient concevoir un logiciel à vocation pédagogique destiné au HP-150, un ordinateur à écran tactile. Le sujet : à partir d'un des chapitres d'une grammaire destinée aux élèves de CE 2 (le choix du chapitre était laissé aux candidats), écrire un programme qui rende attrayant l'apprentissage de cette discipline.

Les vainqueurs toutes catégories confondues se sont vu remettre un HP-150. Engagés sous l'étiquette « amateurs », Jean-Marc Leroy (20 ans) et Thierry Orliac (21 ans) sont étudiants en informatique. Ils ont en outre été sélectionnés d'office pour le concours Diane 85 organisé par l'Agence de l'Informatique.

Le premier prix de la catégorie « professionnels » a été remporté par Gérard Arnaud (45 ans), informaticien, qui faisait équipe avec son fils Rémi (20 ans), étudiant en mathématiques supérieures.

Enfin, les gagnants de la catégorie « amateurs » sont deux étudiants en informatique du CBI, Jean-Luc Marquilly et Jean Trémolières (respectivement 19 et 20 ans).

Les organisateurs du concours prévoient bien évidemment de renouveler l'expérience l'année prochaine. Préparez-vous ! ■

Concours pour tous

UN concours est ouvert, jusqu'au 31 août 1985, aux créateurs de logiciels sur Apple IIe, IIc, Commodore 64, TO7/70, MO5 ou Alice 90. Le sujet de ce concours, organisé par Édiciel et Télérama : écrire un logiciel original figurant un jeu de rôle ou d'aventure. Le vainqueur verra son logiciel édité et commercialisé par Édiciel Hachette.

Le règlement complet peut être demandé à :

Télérama Promotion
50 rue de Miromesnil
75008 Paris. ■

Troisième festival du logiciel : il n'est pas trop tard

VOUS pouvez encore présenter votre candidature au 3e festival du logiciel qui se déroulera du 15 au 27 juillet prochain à Villeneuve-lez-Avignon. Il suffit d'envoyer votre demande avant le 15 juin 1985 à l'adresse suivante : 5 place du Colonel Fabien, 75491 Paris Cedex 10.

Comme les années précédentes, tous les sujets sont acceptés (jeux, didacticiels, etc.). Mais les 22, 23 et 24 juillet seront plus particulièrement consacrés à des réalisations professionnelles (utilitaires, logiciels de développement, etc.). Le 24 juillet, « journée des auteurs », réunira autour des programmeurs de nombreux constructeurs et éditeurs. Des débats et tables rondes sont prévus. Ordre du jour : comment devenir auteur, comment se faire éditer, comment protéger ses droits...

Les délibérations du jury et l'annonce du palmarès auront lieu le 28 juillet et la remise des prix se fera en septembre, au Sicob.

UN LIVRE

PEEKs et POKEs du Commodore 64
Liesert
Édité par Micro Application
1984, 200 pages
Prix : 99 FF

CETTE traduction d'un ouvrage allemand, même si elle arrive un peu tard, fera plaisir à ceux — et ils sont nombreux — qui veulent aller plus loin avec leur C.64 au Basic un peu étriqué.

L'auteur commence avec prudence, et à juste titre, par une présentation théorique de la structure générale de la machine et des instructions auxquelles le titre fait référence (PEEK, POKE, USR et autre SYS), tout en jetant un œil sur l'arithmétique binaire. C'est un peu rapide mais très utile.

Suit une quantité phénoménale de trucs qui, à coup de PEEKs et de POKEs, permettent de faire à peu près n'importe quoi, entre autres dans les domaines du graphisme, de la musique, des périphériques et des accessoires. Les exemples d'application sont généralement étayés d'un court programme de démonstration.

Les dernières pages sont consacrées à une approche simplifiée du langage-machine, accompagnée d'un intéressant programme de simulation qui aidera à comprendre le fonctionnement de base de la machine. Un tableau complet des adresses de la page zéro est fourni.

Un livre dense, touffu même (mais un index permet de s'y retrouver) qui n'est visiblement pas destiné aux débutants. Beaucoup de renseignements précieux pour une machine qui en a besoin.

JPL ■

La gamme Hector : Micronique aligne ses prix

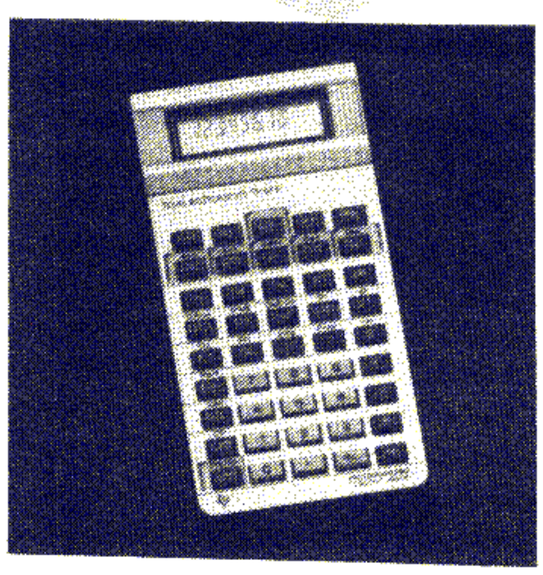
L'HECTOR HRX (Forth résident, 64 Ko de MEV) passe de 3990 à 2990 F s'il est accompagné du Basic 3 X en cassette et de 4350 à 3450 FF avec le Basic 3 X en cartouche.

L'Hector 2 HR (Basic 3 résident, 48 Ko de MEV), proposé dans son coffret Loisirs-Plus, coûte 1500 F de moins : 2490 FF vous suffiront donc pour l'acquérir.

Enfin, le dernier né, l'Hector HR MX (Basic 3 X, Forth, Assemblex et Monitrix résidents, 64 Ko de MEV) qui était à 5490 FF dans sa version 40 colonnes, vaut désormais 4190 FF. La version 80 colonnes de ce modèle n'était pas sortie. Vous pouvez maintenant la trouver ; elle coûte 4690 FF.

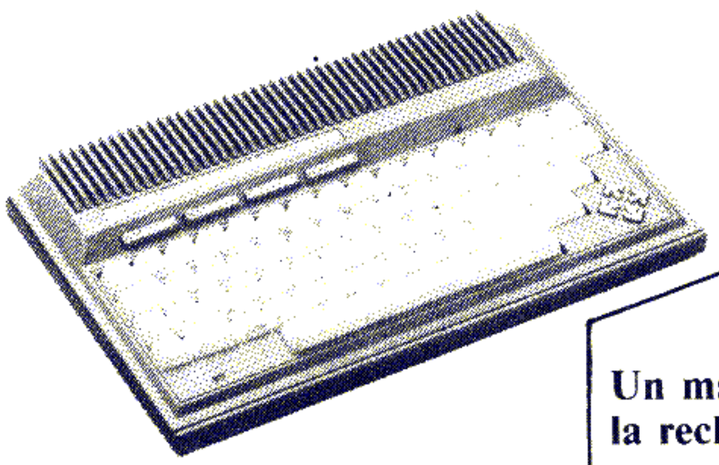
Tandy, Atari, Hector, espérons qu'à l'avenir nous aurons l'occasion de multiplier l'annonce de pareilles nouvelles, tant en ce qui concerne les matériels que les logiciels. ■

TI 57 nouvelle version



LES calculatrices programmables préoccupent encore Texas Instruments. Une des plus célèbres était la bonne vieille TI 57 à diodes rouges. Après la TI 57 LCD, elle est aujourd'hui (re)lancée dans une nouvelle version, la TI 57 II. Cette dernière est dotée d'une mémoire permanente, négociable entre 48 pas de programme et 8 registres de données. En tout, elle possède 80 fonctions (tests, branchements, sous-programmes, etc.) et un clavier plus proche de la TI 57 LCD que de la TI 57. Elle coûte environ 295 FF, et elle est garantie deux ans. ■

Un Commodore à logiciels intégrés



ON l'avait déjà rencontré à l'étranger. Depuis la fin mai, on peut l'acheter en France : le Commodore Plus/4. Outre un Basic, version 3.5, comprenant plus de 75 commandes, un clavier Qwerty, 64 Koctets de mémoire vive — dont 60 sont disponibles à l'utilisateur, cet ordinateur est doté de quatre logiciels intégrés : un traitement de texte, un tableur, un logiciel de gestion de fichiers et un logiciel graphique. Dans sa version Pal, il devrait être vendu 1990 FF. ■

Un magazine sur la recherche informatique

LES étudiants des écoles d'ingénieurs ou des universités de Paris ou de Province trouveront, par l'intermédiaire de leur département informatique, un nouveau magazine consacré à la recherche informatique : *5e génération*. Diffusée gratuitement, mais pour un public restreint, cette revue trimestrielle devrait « constituer, aussi bien un complément de la formation acquise dans les écoles, qu'une ouverture sur les innovations technologiques ».

UN PETIT TOUR CHEZ LE LIBRAIRE

Techniques de programmation sur Apple II
René Belle
Éditions du PSI
1985, 164 pages
Prix : 95 FF

COMMENT programmer ? L'auteur propose de réaliser un programme simple de gestion de données. C'est une façon intéressante de s'initier. Si l'application est classique, elle permet néanmoins des mises au point riches d'enseignement. ■

Dessins géométriques et artistiques avec votre micro-ordinateur
Jean-Paul Delahaye
Éditions Eyrolles
1985, 256 pages
Prix : 120 FF

SI votre ordinateur dispose de quelques possibilités graphiques, vous pourrez certainement, moyennant quelques adaptations, utiliser ces programmes qui sont écrits dans un Basic standard. Vous obtiendrez figures, courbes, étoiles, fractales et quadrillages de toutes sortes qui vous donneront certainement des idées pour créer vos propres dessins. ■

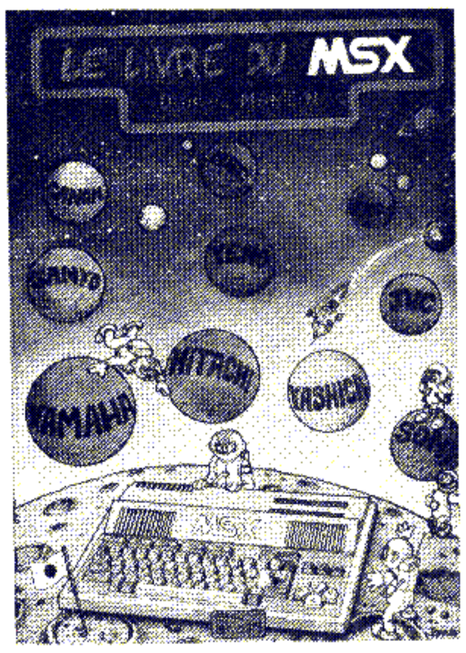
Harrap's informatique dictionnaire
Anglais-Français
Français-Anglais
Claude Camille
et Michel Dehaine
Éditions Harrap
1985, 200 pages
Prix : 225 FF

DEPUIS la dernière édition du Harrap's informatique, en 1976, le langage de la profession a considérablement évolué. Voici une remise à jour qui tient compte des termes les plus récents. On y trouvera tous les mots qui se rapportent aux matériels et aux logiciels et la terminologie employée dans les domaines proches de l'informatique, l'électricité, l'électronique et les télécommunications. Une large part du dictionnaire a été consacrée à la micro-informatique. ■

DEUX LIVRES

Le livre du MSX
Daniel Martin
Édité par BCM (Belgique)
Distribué par PSI
1984, 204 pages
Prix : 110 FF

LE livre du MSX, voilà un titre que l'on pourrait trouver bien définitif ! Cette étiquette de manuel de référence cadre cependant assez bien avec la réalité car on recueille au fil des pages une importante somme d'informations. Si vous voulez connaître l'organisation interne de votre machine, vous y trouverez présentés avec leur mode d'emploi les « banks », les « slots », les diverses tables et registres des différents circuits et les adresses mémoire utiles. Attention, si la lecture est intéressante, elle n'est cependant pas toujours aisée (elle nécessite une bonne vue, la taille de certains caractères étant fort réduite). Des connaissances de base en Assembleur ainsi qu'une bonne habitude des notions hexadécimales seront nécessaires.



Une série de petits programmes, en Basic ou en Assembleur, sont proposés pour illustrer les divers chapitres. Il est possible de se les procurer sur disquette, dans l'un des trois formats disponibles en MSX, en écrivant aux Éditions BCM. Cette formule vaut surtout pour les programmes longs : le désassembleur en Basic, le générateur de caractères, le moniteur, etc.

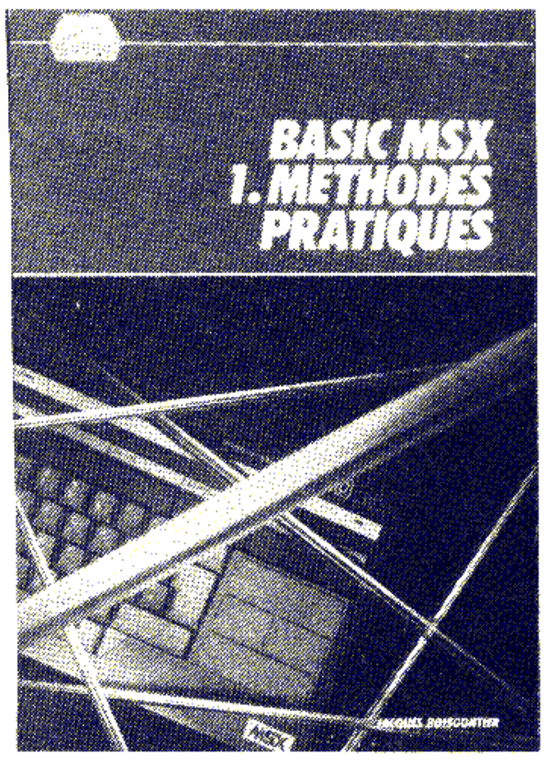
On pourra regretter l'aspect un peu succinct de certaines parties ; celle qui traite du générateur sonore, par exemple, n'est pas vraiment exploitée à fond. Mais l'auteur promet lui-même un complément à cet ouvrage, alors...

Éditions BCM
24 route de la Sapinière
4960 Banneux
Belgique

Basic MSX
Méthodes pratiques
Jacques Boisgontier
Éditions du PSI
1985, 216 pages
Prix : 120 FF

LES manuels livrés d'origine avec les ordinateurs sont souvent des listes d'instructions assez rébarbatives présentant sous un aspect plutôt technique les diverses possibilités de la machine. Certes, les initiés y trouvent leur compte, mais le débutant reste souvent « bloqué », ne comprenant ni le sens, ni l'intérêt, ni même parfois le fonctionnement de certains ordres. Quant aux livres d'initiation, ils sont souvent trop généraux.

Les « méthodes pratiques » de Jacques Boisgontier abordent le problème sous un autre angle. L'ouvrage va plus loin que la plupart des manuels, expliquant ce qu'il est possible de faire avec les instructions du Basic MSX. Une première partie fait le tour des possibilités de la machine



avec de nombreux exemples courts et bien commentés. La deuxième partie permettra au lecteur d'approfondir ses connaissances par l'étude détaillée d'une vingtaine de programmes.

S'il s'adresse en priorité au débutant, l'ouvrage séduira aussi l'amateur averti par son côté pratique.

JD ■

LA GAZETTE DE LIST

DU CÔTÉ DES CLUBS

Dans la Seine-Maritime

MOYENNANT une cotisation annuelle de 50 F à laquelle s'ajoute un droit d'inscription de 100 F, vous pourrez adhérer au Club 2000 d'Etran, près de Neuville les Dieppe. Association à but non lucratif, le club met à la disposition de ses membres un Apple IIe, un Amstrad CPC 464 (couleur) et un Oric Atmos. Deux rendez-vous hebdomadaires : le mercredi de 20 h à 22 h 30 et le samedi de 13 h 30 à 16 h. Pour en savoir un peu plus, vous pouvez écrire à l'adresse suivante :

Thierry Courant
15 rue de l'Ancien Port
Etran
76370 Neuville les Dieppe

Dans les Pyrénées-Atlantiques

DANS le cadre des fêtes patronales de Maslacq (situé entre Orthez et Pau, non loin du complexe de Lacq), le Microclub du village organise le samedi 29 juin prochain une foire à la micro-informatique d'occasion qui accueillera ceux qui veulent vendre, acheter ou échanger ordinateurs et périphériques. Aux personnes qui désirent vendre du matériel, une participation de 30 F est demandée pour la location de l'emplacement (les intéressés doivent écrire dès maintenant au club).

A partir de 14 heures, la salle du Trinquet accueillera les participants à cette première foire du micro-ordinateur et du périphérique d'occasion. L'entrée est gratuite.

Microclub de Maslacq
Maison Menat
64300 Orthez

Dans l'Hérault

A Montpellier, un nouveau club vient de naître. Il a pour but de rassembler le plus grand nombre possible d'utilisateurs de Sanyo MBC 550/555. Il n'est pas nécessaire d'habiter la ville pour participer à ses activités. Des quatre coins de France, et même de l'étranger, les programmeurs de tous horizons sont les bienvenus.

Association régie par la loi de 1901, le Sanyo-Club demande à ses adhérents une cotisation de 100 F. Ses projets sont multiples et variés : échanges de connaissances et de documentation, élaboration commune de programmes, études d'extensions matérielles et diffusion parmi les membres du club d'un bulletin périodique.

Pour tous renseignements, écrire à :

Philippe Chardon
1 rue de Clémentville
34000 Montpellier

En Haute-Savoie

LA Maison des Jeunes et de la Culture d'Annemasse annonce la création de trois clubs, le premier consacré à l'Amstrad, le second à l'Apple II et le troisième au Commodore 64. Ils trouveront leur place au sein de l'activité informatique qui fonctionne déjà à la MJC. Chaque semaine se tiennent six cours qui s'adressent aux débutants autant qu'aux « branchés ».

Pour obtenir des informations plus précises, vous pouvez téléphoner au 16 (50) 92 10 20, à partir de 16 heures.

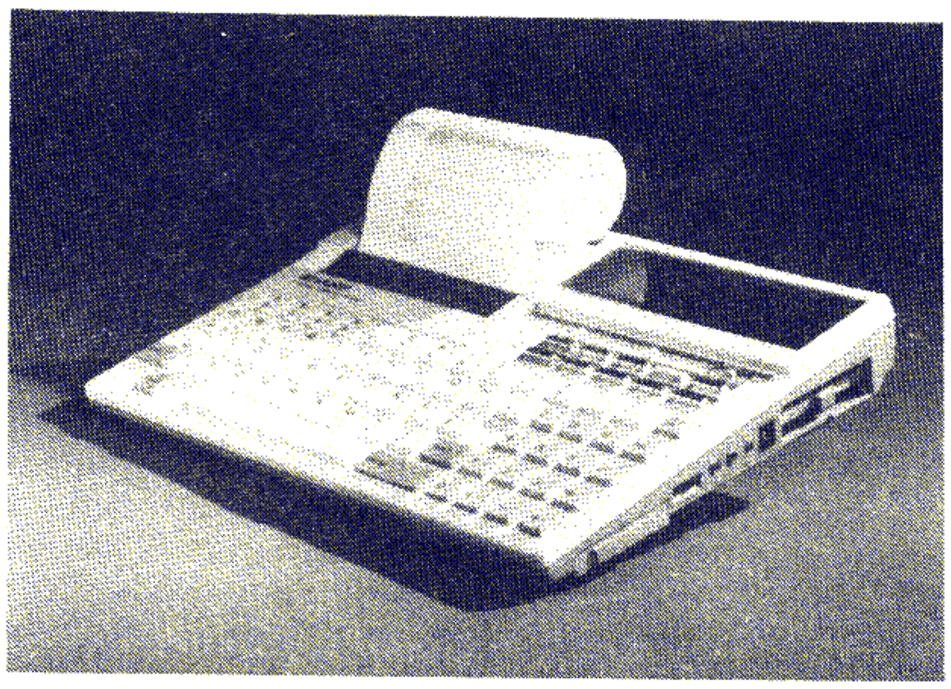
MJC Maison pour tous
3 rue du 8 Mai
74100 Annemasse ■

Des accessoires informatiques vendus par correspondance

POUR recevoir gracieusement le deuxième catalogue de fournitures informatiques diffusé par la Société Moore Paragon (cartouches de rubans encreurs, papier pour imprimante, disquettes, mais aussi imprimantes, modems et mobilier de bureau), il suffit d'en faire la demande en utilisant leur numéro de téléphone vert (l'appel est gratuit) :
16 (05) 27 78 11

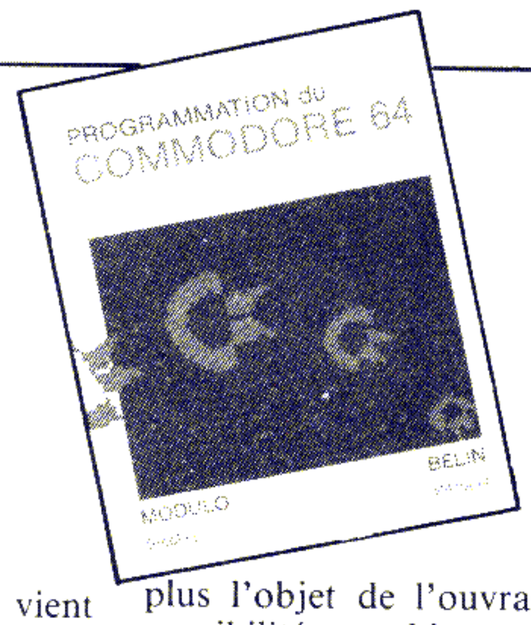
Moore Paragon
22 rue de Sèvres
92100 Boulogne Billancourt ■

Nouveau chez Sharp, le PC-2500



PRÉSENTÉ au Spécial Sicob du 6 au 11 mai, le PC-2500 (21 × 29,7 × 5 cm) devrait être en vente dès maintenant. Il dispose d'un écran à cristaux liquides de quatre lignes de 24 caractères et d'un clavier « mécanique » Qwerty. L'imprimante, intégrée, est une petite table traçante quatre couleurs munie de stylos à bille. Le Basic de ce nouveau modèle, qui contient en mémoire morte un tableur et un répertoire téléphonique, est celui du Sharp PC-1350 (voir page 58 de ce numéro). D'origine, le PC-2500 offre 3 Ko de mémoire vive qui pourront être augmentés par l'adjonction de cartes d'extension (8 Ko, environ 800 FF ; 16 Ko, environ 1600 FF). Il coûte déjà, dans sa version de base, près de 5000 FF... ■

UN LIVRE



Programmation du Commodore 64
Ian Sinclair
Éditions Belin
Collection Modulo
1984, 130 pages
Prix : 95 FF

VOICI un livre qui vient grossir les rangs des ouvrages d'initiation au Basic sur le C.64. Allant progressivement de notions simples (l'instruction PRINT) aux informations plus compliquées (on aborde le langage-machine), les chapitres ont au moins le mérite de la clarté faute d'avoir celui de l'originalité.

L'auteur a donc adopté une démarche rassurante pour le lecteur. Ce dernier pourra, tenant le livre d'une main et pianotant sur le clavier de son C.64 de l'autre main, expérimenter à loisir en suivant pas à pas les exemples d'application proposés.

Quelques pages seulement sont consacrées à la conception des programmes (analyse et découpage en modules). C'est trop peu, mais ce n'était pas non plus l'objet de l'ouvrage. Les possibilités graphiques et sonores de l'ordinateur y sont traitées moins succinctement et l'on trouve les éléments pour de premières recherches ; cela ne donne néanmoins qu'une idée superficielle des vraies capacités de la machine.

Le chapitre « Touches de fonctions programmables » est un peu irritant : dix lignes de texte et un petit tableau font un peu trop brièvement le tour du sujet.

Cet ouvrage, qui se termine sur une description des extensions possibles (imprimante, joystick, cartouches, disquettes, etc.) sera utile aux vrais débutants en leur fournissant la matière pour démarrer leur apprentissage.

JPL ■

VOYAGE AU CENTRE D'UN FICHIER

L'ACCÈS à une information contenue dans un fichier est particulièrement rapide s'il est direct. Mais il met en œuvre toute une série d'opérations. Pour mieux comprendre ce qui se passe, une visite dans le système d'exploitation est nécessaire.

Parmi les techniques de recherche d'informations à l'intérieur d'un fichier, il en est deux particulièrement simples, et auxquelles les autres font appel : l'accès séquentiel et l'accès direct. Ces techniques ont pour objet de restituer le contenu d'un élément d'un fichier. Cet élément du fichier est appelé « enregistrement ».

L'accès séquentiel effectue la lecture complète d'un fichier, enregistrement par enregistrement. Il part de l'enregistrement numéro zéro, et continue avec les enregistrements numéro un, numéro deux, etc., jusqu'à la fin du fichier. Ainsi, l'accès à l'enregistrement numéro x peut devenir extrêmement long si, la taille du fichier étant importante, l'information demandée se situe très loin du début.

L'accès direct, à partir d'un numéro, va directement à un enregistrement donné. Le temps d'accès à l'information ne dépend plus alors que dans une faible part du numéro d'enregistrement. Ainsi, avec un disque dont le temps d'accès moyen est de 20 millisecondes (performance honorable pour un disque dur), la lecture en accès direct de l'enregistrement numéro 1000 d'un fichier ne mettra guère plus de 20 millisecondes. Si l'on exploite le fichier en séquentiel, il faudra, avant de lire l'enregistrement 1000, accéder inutilement aux 1000 précédents qui portent les numéros 0 à 999. Une telle lecture met donc, dans le pire des cas, 1001 fois plus de temps.

Où se trouve au juste quoi ?

Si l'exploitation d'un fichier par accès direct est particulièrement rapide, sa mise en place n'est pas simple. Bien entendu, la plupart des langages de programmation disposent d'une routine réalisant cette fonction. Mais voyons exactement ce qui se passe à l'intérieur de l'ordinateur. Par exemple, dans le cas le moins compliqué, celui de la lecture, on insère dans un programme Basic l'instruction : READ # 1,21;I

Cette instruction va rechercher dans le fichier ouvert sous le numéro 1, le vingt et unième enregistrement, et mettre son contenu dans la variable I. Le système d'exploitation est appelé à ce moment-là. Il effectue la lecture en activant la routine adéquate qui aura reçu (généralement) trois paramètres. Le premier est le numéro logique du fichier, le second est le numéro d'enregistrement, et le troisième est l'adresse en mémoire et la longueur de l'information à restituer (ici, la variable I).

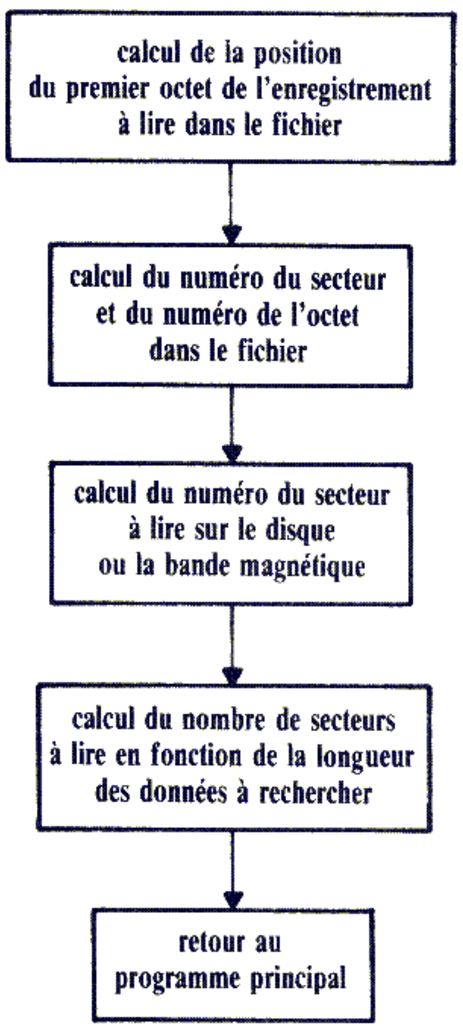
Dans un premier temps, il est nécessaire de localiser précisément à quel endroit du fichier se situe l'information. Prenons le cas le plus simple et le plus fréquent, celui où tous les enregistrements ont la même longueur. Cette longueur est généralement indiquée dans le répertoire du disque (ou de la bande magnétique) sur lequel le fichier est mémorisé. Elle se trouve également en mémoire centrale, car le système d'exploitation, dans le but d'économiser le nombre d'accès au disque, n'a pas manqué de charger cette information lors de l'ouverture du fichier.

A partir du début de ce fichier, il est donc possible de calculer simplement le numéro de l'octet où commencent les informations recherchées : il est égal au produit du numéro d'enregistrement demandé par la longueur d'un enregistrement. Soit : (n° de l'octet) = (n° d'enregistrement) × (longueur d'un enregistrement).

Bien entendu, nous considérons que le premier octet d'un fichier ou que son

VOYAGE AU CENTRE D'UN FICHIER

Calcul de la position des données (sous-programme)



premier numéro d'enregistrement est le numéro zéro. Toutefois, il est rarement possible d'adresser, à partir du numéro de son premier octet, une information située sur une mémoire de masse (un disque ou une bande magnétique). En effet, les données sont presque toujours lues par secteurs de 256 octets. C'est pourquoi il faudra convertir le numéro de l'octet en un couple [numéro de secteur, numéro d'octet dans ce secteur]. Grâce aux opérateurs DIV et MOD (le premier donne le résultat entier d'une division, le second, le reste de la division entière), on a : (n° de secteur dans le fichier) = (n° de l'octet) DIV 256 et (n° de l'octet dans le secteur) = (n° de l'octet) MOD 256.

Si la position de l'information est maintenant localisée dans le fichier, il faut encore rechercher où elle se situe sur le disque ou la bande magnétique. Et pour cela, on doit connaître la position du début du fichier sur le support. De même que la longueur des enregistrements, cette information est mémorisée dans le répertoire du disque, et elle a été recopiée en mémoire centrale par le système d'exploitation. Elle indique souvent un numéro de secteur qui permet de savoir où se trouve la première information du fichier sur lequel on travaille. Le numéro de secteur du disque sur lequel est mémorisé le début de l'enregistrement est donné par : (n° du secteur sur le disque) = (n° du secteur dans le fichier) + (n° du premier secteur du fichier).

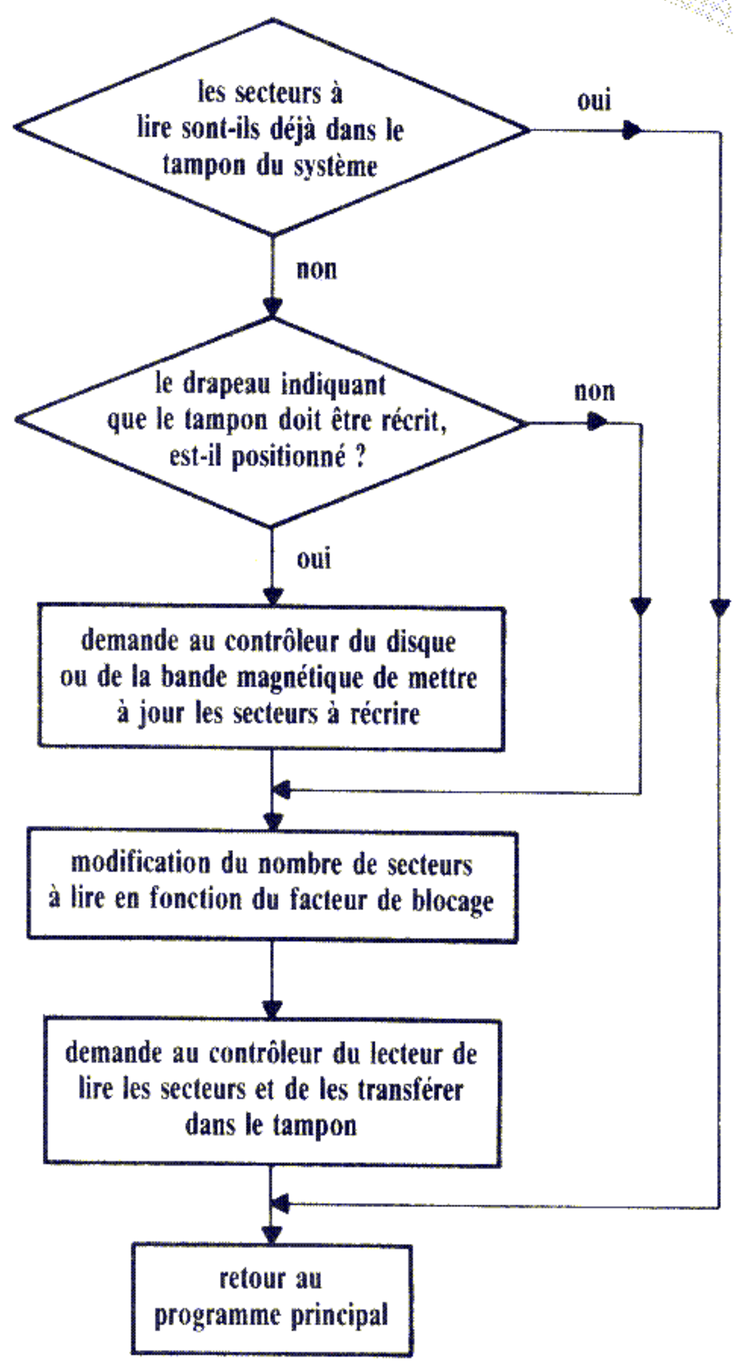
Ainsi, le début de l'enregistrement qui nous intéresse est parfaitement localisé sur le disque ou la bande magnétique. Il reste alors à savoir combien de secteurs doivent être lus. En effet, si l'enregistrement à lire commence au 255e octet et s'il a une longueur supérieure à un octet, l'information à restituer se trouve « à cheval » sur deux secteurs. Il sera donc nécessaire de lire ces deux secteurs. Il en est de même pour un enregistrement de longueur supérieure ou égale à 258 octets : il nécessite la lecture d'au moins deux secteurs (parfois trois !). Le nombre de secteurs à lire est donné par le calcul suivant : (nombre de secteurs à lire) = ((numéro de l'octet dans le secteur) + (longueur d'un enregistrement) + 255) DIV 256.

Une salle d'attente pour les données

La préparation de la lecture est ainsi terminée. Toutefois, il faut savoir que les accès aux disques ou aux bandes magnétiques sont extrêmement longs comparés aux accès à la mémoire centrale. Il est donc important de réduire le plus possible le nombre d'accès. Pour cela, les systèmes d'exploitation utilisent ce que l'on appelle une mémoire tampon (*buffer*, en américain). Ce tampon représente une fenêtre du fichier, il conserve en mémoire centrale une partie de celui-ci, d'une longueur de quelques secteurs. Sur certains systèmes, ce mécanisme est optionnel et doit être explicitement demandé lors de l'ouverture du fichier, par le mot BUFFERED.

Après un accès à une certaine partie du fichier, il est donc possible que le tampon contienne déjà les données recherchées. Dans ce cas, la lecture à partir du disque est inutile. Les informations sont directement transférées

Chargement du tampon (sous-programme)

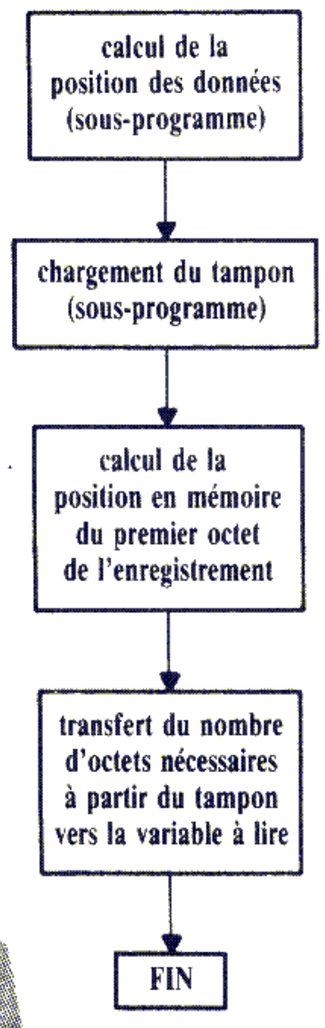


depuis le tampon. S'il a une capacité de N secteurs et si l'on doit en lire NL, il faut alors vérifier que le numéro du premier secteur à lire se trouve dans les (N – NL + 1) premiers secteurs mémorisés dans le tampon. Dans le cas contraire, le tampon est mal placé et les données doivent être recherchées sur le support magnétique.

Mais il faut savoir que, toujours pour limiter les accès au disque, beaucoup de systèmes d'exploitation n'écrivent pas immédiatement les données lorsqu'un ordre d'écriture est envoyé. Ils placent ces informations dans le tampon, et remettent l'écriture réelle à plus tard. Dans ce cas, un drapeau (*flag*, en américain) indique si le tampon doit faire l'objet d'une écriture. Il est alors nécessaire de recopier les informations de ce tampon, avant de le détruire pour y placer les données à lire.

Cette opération terminée, il reste à

Lecture d'un enregistrement



déterminer le nombre de secteurs à lire. Ce nombre a déjà été calculé par le système à partir de la longueur et de la position des données sur le disque. Mais peu de systèmes adaptent la longueur du tampon de lecture/écriture aux données auxquelles on veut accéder.

Les systèmes d'exploitation moins performants (ceux que l'on trouve sur beaucoup d'ordinateurs familiaux) gèrent des tampons de longueur fixe. Sur les plus gros ordinateurs, il est possible de fixer, pour chaque fichier, un *facteur de blocage* qui caractérise la dimension du tampon. Il dépend de deux paramètres. Le premier concerne le nombre d'enregistrements que le tampon est susceptible de mémoriser, le second est la longueur d'un enregistrement. Le facteur de blocage s'exprime donc en Koctets. Ainsi, un mini-ordinateur tel que le HP-3000 permet d'avoir des facteurs de blocage jusqu'à 28 Koctets, avec une limite de 255 enregistrements.

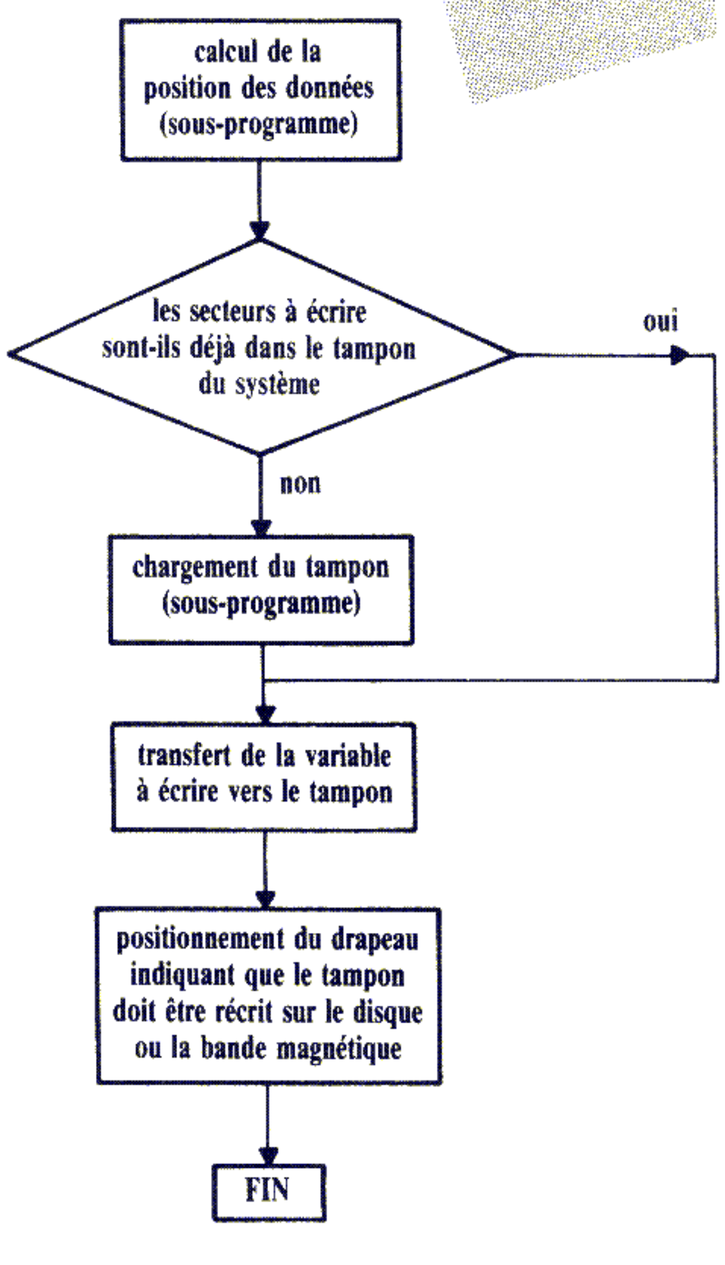
Une lecture à tête reposée

Lorsque le nombre de secteurs à lire — éventuellement modifié par le facteur de blocage — a été déterminé, les données peuvent être recherchées sur le disque ou la bande magnétique. Cette opération est du ressort du contrôleur de disque qui se charge aussi de lancer la rotation du disque, de demander le positionnement de la tête de lecture et de transférer les informations dans le tampon.

Il est alors nécessaire de calculer à partir de quelle adresse les données du tampon doivent être transférées dans la variable à lire : (adresse de transfert) = (adresse du début du tampon) + (n° de l'octet dans le secteur). Un nombre d'octets égal à la longueur de la variable à lire est donc déplacé de cette adresse de transfert à l'adresse de la variable. Et la variable est enfin lue.

L'écriture d'une information nécessite quelques opérations supplémentaires. Elles apparaissent dans l'algorithme d'écriture (voir ci-dessous). Il faut toutefois remarquer qu'une écriture est presque toujours précédée d'une lecture. En effet, un enregistrement couvre rarement une longueur exacte de secteurs. Les lecteurs de disques ou de bandes magnétiques ne permettant que des écritures complètes de secteurs, si l'on ne veut pas détruire les enregistrements voisins, il est indispensable de commencer par lire intégralement les secteurs qui seront touchés par la modification. Dès que ces secteurs auront été remis à jour, ils pourront être réécrits sans danger pour le reste du fichier.

Écriture d'un enregistrement



Concevoir un accès direct est donc plus complexe qu'on le croit souvent. En fait, cela ne représente qu'une toute petite partie de ce qu'est capable d'effectuer un système d'exploitation. On peut donc imaginer la difficulté qu'il y a à écrire un système d'exploitation, surtout lorsque les programmes du système atteignent la taille respectable de plusieurs méga-octets. Inutile d'ajouter que l'on y découvre beaucoup de bogues...

Thierry CHAMORET

CHERCHER LA CHAÎNE

L'interpréteur Basic contenu dans la mémoire morte du Canon X-07 est une mine d'or pour le programmeur en langage-machine qui connaît les bonnes adresses. Il va nous permettre ici de créer une routine capable de rechercher une suite de caractères dans un programme et d'afficher la ligne où elle se trouve.

Lorsque l'on écrit un programme en langage-machine, il est plus astucieux d'utiliser les routines existantes que de les réinventer. Cela permet à la fois de réduire le nombre d'instructions (un gain de place souvent très appréciable), et de faciliter les développements en évitant l'écriture et le test de routines complexes.

Pour illustrer ces principes, voici un petit utilitaire qui apporte une fonction rencontrée couramment sur les éditeurs de gros ordinateurs : la localisation de caractères. Son but consiste à rechercher une suite de caractères dans un programme et à afficher la ligne où elle se trouve.

Pour l'implanter, il suffit de rentrer le programme Basic *(Recherche)* qui met en place la routine en langage-machine. La seule difficulté est de déterminer l'adresse de début d'implantation : elle est fonction de la capacité mémoire disponible et de la zone réservée aux fichiers. Il est donc préférable de protéger le programme par un CLEAR 50,AD – 1. Par exemple, pour une capacité de 8 Ko avec environ 4 Ko de fichiers, il faudra faire CLEAR 50,3799 (voir *Le plan de la mémoire,* ci-dessous).

Pour ceux qui disposent de beaucoup de place, l'implantation après les variables et avant la fin de pile (attention, la pile va en décroissant) est possible aussi, et ne nécessite pas de CLEAR. Le programme est totalement relogeable.

Une fois introduit et les adresses modifiées, taper RUN, l'utilitaire *Recherche* est prêt à l'emploi.

Mode de recherche

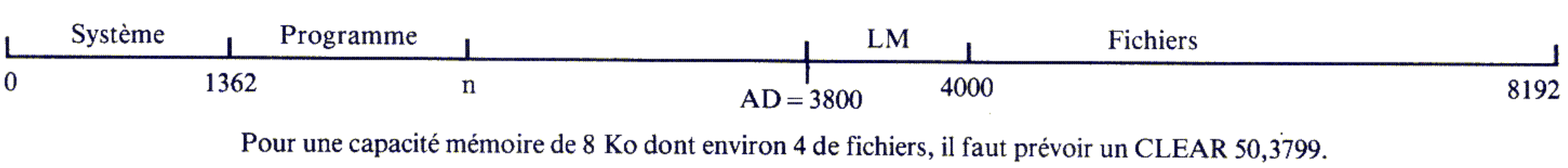
Sa syntaxe d'utilisation est assez simple. Il suffit de taper :
EXEC AD : [NL] @ ['"] SUITE @
puis RETURN.

Il faut alors le texte d'un programme Basic. Les crochets indiquent les opérandes facultatifs.

EXEC AD : exécution du programme en langage-machine situé à l'adresse d'implantation. Les deux-points sont obligatoires.

[NL] : numéro de ligne facultatif. S'il est absent, la recherche commence au début du programme, sinon à partir du

Le plan de la mémoire



Pour une capacité mémoire de 8 Ko dont environ 4 de fichiers, il faut prévoir un CLEAR 50,3799.

La routine Recherche en Basic

```
10  CLS:AD = XXXX
20  CLEAR50,AD – 1
30  READA:IFA < 0THENPRINT"PRET":ENDELSEPOKEAD,A:AD = AD + 1:GOTO30
40  DATA225,225,225,225,225,225,225,205,150,245,35,126,254,34,32,1,35,229,221,225,
50  DATA205,13,243,197,225,43,35,229,253,225,126,35,182,40,47,35,78,35,70,221,229
60  DATA209,35,26,190,32,3,19,24,248,254,64,40,17,126,183,40,224,27,26,254,34,40
70  DATA231,254,64,40,227,43,24,224,225,197,253,229,253,33,0,64,195,139,254,221
80  DATA229,225,205,102,246,201, – 1
```

A la ligne 10, remplacer XXXX par l'adresse à définir.

La routine Recherche désassemblée

Adresse	Instruction	Code (lignes Data)	Commentaire
12000	POP hl	225	
12001	POP hl	225	
12002	POP hl	225	
12003	POP hl	225	
12004	POP hl	225	
12005	POP hl	225	
12006	POP hl	225	
12007	CALL 62870	205 150 245	Recherche numéro de ligne → de
12010	INChl	35	
12011	LDa(hl)	126	
12012	CP 34	254 34	Cas du @ ''
12014	JR NZ 1 12017	32 1	
12016	INChl	35	
12017	PUSHhl	229	Début de chaîne
12018	POP IX	221 225	
12020	CALL 62221	205 13 243	Recherche adresse ligne → bc
12023	PUSHbc	197	
12024	POP hl	225	
12025	DEChl	43	
12026	INChl	35	
12027	PUSHhl	229	
12028	POP IY	253 225	Adresse ligne → IY
12030	LDa(hl)	126	
12031	INChl	35	
12032	OR (hl)	182	
12033	JR Z 47 12082	40 47	Test fin de programme
12035	INChl	35	
12036	LDc(hl)	78	Numéro de ligne → bc
12037	INChl	35	
12038	LDb(hl)	70	
12039	PUSHIX	221 229	
12041	POP de	209	
12042	INChl	35	Test égalité chaîne/caractère programme
12043	LDa(de)	26	
12044	CP (hl)	190	
12045	JR NZ 3 12050	32 3	
12047	INCde	19	
12048	JR 248 12042	24 248	Si égalité
12050	CP 64	254 64	Si différence, test fin de chaîne
12052	JR Z 17 12071	40 17	
12054	LDa(hl)	126	
12055	OR a	183	
12056	JR Z 224 12026	40 224	Fin ligne
12058	DECde	27	
12059	LDa(de)	26	Traitement si début de chaîne
12060	CP 34	254 34	
12062	JR Z 231 12039	40 231	
12064	CP 64	254 64	
12066	JR Z 227 12039	40 227	
12068	DEChl	43	
12069	JR 224 12039	24 224	
12071	POP hl	225	Chaîne trouvée
12072	PUSHbc	197	
12073	PUSHIY	253 229	
12075	LDIY 16384	253 33 0 64	
12079	JP 65163	195 139 254	Saut routine LIST
12082	PUSHIX	221 229	Chaîne non trouvée
12084	POP hl	225	
12085	CALL 63078	205 102 246	Détermination fin de ligne
12088	RET	201	

Les registres

hl : pointeur sur la ligne des programmes
de : pointeur sur la suite
IX : pointeur sur le début de la suite
IY : adresse de la ligne en cours
bc : numéro de la ligne en cours

numéro de ligne supérieur ou égal à celui indiqué.

@ : indique le début de la suite obligatoire.

[''] : guillemets facultatifs qui permettent de différencier les caractères des fonctions.

SUITE : suite de caractères et/ou de fonctions, instructions, etc.

@ : indique la fin de la suite obligatoire.

Si la suite existe, la ligne déterminée s'affiche et reste en position comme avec LIST @ . Les modifications sont possibles après BREAK (une fois), s'il n'y a pas d'occurrence retour du curseur. Pour mieux comprendre, voici quelques exemples :

EXEC 8000 : @ A – B @ recherche la suite ''A(code 210)B'' depuis le début du programme.

EXEC 8000 : 1002@''A – B @ recherche la chaîne ''A(code 45)B'' à partir de la ligne 1002.

EXEC 8000 : @ HELENE @ recherche la suite ''HE(code LEN)E''.

EXEC 8000 : @ ''HELENE @ recherche la chaîne ''HELENE''.

La liste désassemblée commence ici à l'adresse 12000. Les sept premières lignes contiennent POP *hl* pour retrouver l'adresse des deux-points qui suivent EXEC et pour mettre à jour la pile. EXEC protège les registres par empilage.

En 62870 se trouve une routine qui transforme une suite de chiffres ASCII en un nombre binaire sur deux octets (utilisation habituelle dans GOTO, GOSUB,...) ; *hl* pointe un caractère avant les chiffres et le résultat est stocké dans *de* (voir à l'adresse 12007 de la routine désassemblée).

En 62221 se situe une routine qui recherche l'adresse réelle mise dans *bc* de la ligne dont le numéro est contenu dans *de* (voir à l'adresse 12020). En 65163, c'est la partie de la routine LIST. Il faut alors mettre en pile le numéro de la ligne finale puis l'adresse de la première ligne à lister. De plus, si IY contient 16384, on obtient un effet identique à LIST @ (voir aux adresses 12075 et 12079).

Enfin, en 63078 (adresse 12085), on trouve la routine recherchant la fin d'instruction (:) ou la fin de ligne (o) à partir de *hl*.

Laurent GRAS

LE BASIC DU LANSAY 64

ALORS que d'un côté, on cherche à uniformiser le Basic, de l'autre, certains constructeurs laissent libre cours à leur imagination et proposent des Basic presque fous tant ils sont riches. C'est le cas de celui de l'Enterprise commercialisé en France sous le nom de Lansay 64, qui se trouve être, en outre, un Basic structuré.

La version de base de l'Enterprise 64 (1) tient dans un boîtier à peine plus gros que le clavier. Quelques touches de couleur et une poignée de jeu l'égaient. Le côté droit reçoit l'ensemble d'édition : l'insertion, l'effacement de caractères à gauche (ERASE) ou à droite (DEL) du curseur, son déplacement dans toutes les directions (grâce à la poignée) sur deux pages d'écran.

Une ouverture à gauche accueille les cartouches préprogrammées. Le Basic réside justement dans une de ces cartouches. A la mise en route, l'écran annonce que sur les 64 Ko de mémoire vive, il en reste un peu plus de 49 à la disposition de l'utilisateur.

Une partition de la mémoire en différentes zones de programme est possible, comme sur certains ordinateurs de poche. Les numéros de ligne des programmes ne peuvent pas dépasser 9999. La numérotation automatique est disponible, avec une syntaxe originale : AUTO AT x STEP y numérote automatiquement à partir de la ligne x avec un pas de y. La renumérotation opère de la même façon, avec RENUMBER, et sur des procédures (par exemple, sur la procédure TEST, RENUMBER TEST AT 1000 STEP 5).

Le Basic de l'Enterprise fait preuve de rigueur. Ainsi, lors de l'écriture d'un programme, on peut omettre l'instruction LET, mais de toutes les façons l'interpréteur l'ajoute automatiquement après la validation de la ligne.

Les identificateurs de variables jouissent de toute la liberté octroyée par 31 caractères significatifs. Mais il n'existe pas de spécificateur de type numérique. Les calculs utilisent des nombres qui vont jusqu'à 1E62 avec une précision de 9 chiffres. Entre autres originalités, l'infini est présent dans ce Basic, sous le nom de INF. Il correspond à la valeur $9.999999999 \times 10^{62}$. On trouve aussi EPS(X), la plus petite quantité qui fait changer la valeur de X, *Epsilon* en termes mathématiques.

Les chaînes de caractères acceptent jusqu'à 254 caractères. Et les fonctions du Basic qui les traitent sont originales :

(1) Au moment où nous avons testé ce matériel ses importateurs n'avaient pas encore décidé de le rebaptiser Lansay 64.

Liste des mots clés du Basic du Lansay 64

ABS	EDIT	LOG	RUN
ACOS	END	LOG2	SAVE
ALLOCATE	ENVELOPE	LOG10	SCROLL
AND	EPS	LOOK	SEC
ANGLE	EXIT	LPRINT	SELECT
ASK	EXLINE	LTRIM$	SELECT CASE
ASIN	EXP	MAGENTA	SET
ATN	EXSTRING$	MAX	SGN
ATTRIBUTES	EXT	MAXLEN	SIN
AUTO	EXTYPE	MERGE	SINH
		MIN	SIZE
BAND	FLUSH	MOD	SOUND
BEAM	FOR NEXT		SPEAKER
BIAS	FP	NEW	SPEEK
BIN	FREE	NUMERIC	SPOKE
BLACK			SQR
BLUE	GET	ON GOSUB	START
BOR	GOSUB	ON GOTO	STEP
BORDER	GOTO	OPEN	STOP
	GRAPHICS	OPTION	STR$
CALL	GREEN	OR	STRING
CAPTURE	HANDLER	ORD	
CAUSE EXCEPTION	HEX$	PALETTE	TAB
CEIL		PAPER	TAN
CHAIN	IF MISSING	PEEK	TANH
CHARACTER	IF THEN ELSE	PI	TEXT
CHR$	IMAGE	PING	TIME/TIME$
CLEAR	IN	PLOT	TIMER
CLOSE	INF	POKE	TOGGLE
CODE	INFO	POS	TRACE ON/OFF
COLOUR	INK	PRINT	TRUNCATE
CONTINUE	INKEY$		TYPE
COPY	INPUT	RAD	
COS	INPUT PROMPT	RANDOMIZE	UBOUND
COSH	INT	READ	UCASE$
COT	INTERRUPT	RED	UNTIL
CSC	IP	REDIRECT	USR
CYAN		RELEASE	VAL
	LBOUND	REM	VERIFY
DATA	LCASE$	REMarque	VIDEO
DATE/DATE$	LEN	RENUMBER	
DEF	LET	RESTORE	WAIT DELAY
DEG	LINE INPUT	RETRY	WHEN
DELETE	LINE MODE	RGB	WHILE
DIM	LINE STYLE	RND	WHITE
DISPLAY	LIST	ROUND	WORD$
DO LOOP	LLIST	RTRIM$	YELLOW
	LOAD		

passage de minuscules en majuscules et inverse, suppression des espaces à gauche ou à droite de la chaîne, position d'une chaîne dans une autre. Ici, les traitements classiques sont effectués directement, sans instructions. Par exemple, pour prendre les trois premiers caractères de B$ et les mettre dans A$, avec ce Basic, on écrira A$ = B$ (1:3) ; avec un Basic plus classique on aurait écrit A$ = LEFT$ (B$,3), ou encore A$ = MID$ (B$,1,3).

En maths, mention très bien

Le jeu des fonctions mathématiques satisfera les plus exigeants. La trigonométrie dispose de toutes les fonctions habituelles (sinus, cosinus, tangente et les inverses), mais aussi de fonctions originales comme la sécante, la cosécante, sinus et cosinus hyperboliques, etc. Les calculs sont effectués au choix en degrés ou en radians, et la conversion est toujours possible. Il faut encore noter une fonction originale, ANGLE, qui donne les valeurs d'angles dans un système en coordonnées cartésiennes.

Les logarithmes existent sous trois formes : logarithme népérien, logarithme décimal et logarithme en base 2. Selon la partie que l'on veut extraire d'un nombre, on dispose de plusieurs fonctions : INT(X) retient la partie entière de X, IP(X) retient le nombre X sans sa partie fractionnaire, CEIL(X) retient le plus petit entier immédiatement supérieur à X, FP(X) retient la partie fractionnaire de X. La liste est encore longue : ce Basic compte plus de 40 fonctions mathématiques.

Quant aux autres instructions, elles sont aussi très originales. Ce Basic pénètre ouvertement dans le domaine des structures répétitives conditionnelles : il est structuré. FOR...NEXT existe, assurément, mais avec une petite originalité que l'on retrouvera avec les autres boucles. Dès l'introduction d'une structure répétitive, la liste présente automatiquement une indentation de la zone comprise entre les deux bornes, c'est-à-dire que l'intérieur de la boucle est décalé pour favoriser son repérage dans la liste. Inutile de recourir à une boucle vide pour les temporisations, il existe un WAIT DELAY.

Ce Basic étant structuré, d'autres types de boucles sont présents. Le choix est tellement large que l'on s'y perd au début. DO WHILE relancera la boucle fermée par LOOP tant que la ou les conditions ne seront pas remplies, DO UNTIL agira de même jusqu'à la réalisation de la condition. Nuance subtile ? Simple pourtant. Si j'écris la boucle :

```
DO WHILE A<3
  A = A + 1
LOOP
```

la sortie se fera lorsque A aura atteint la valeur 3.

Tandis que le même résultat avec UNTIL sera obtenu par :
DO UNTIL A>2 ou DO UNTIL A = 3.

Mais ce n'est pas tout. La condition peut aussi s'écrire sur la deuxième borne de la boucle, après le LOOP. Ainsi :

```
DO
  A = A + 1
LOOP UNTIL A>2
```

ou

```
LOOP WHILE A<3
```

Pour sortir des boucles en cours de route, EXIT DO ou EXIT FOR sont prévus. Inutile donc de se préoccuper de mettre la variable à la valeur limite.

Les instructions de tests sont de plusieurs formes, elles aussi. On rencontre IF et ELSE, mais ils s'accompagnent de END IF qui clôt une série de tests relatifs entre eux et disposés dans des lignes successives. Il faudra manœuvrer avec prudence. Les tests à plus de deux sorties possibles obligent, en Basic traditionnel, à recourir à une batterie de

LE BASIC DU LANSAY 64

questions successives. Avec SELECT CASE, le choix est donné entre les possibilités multiples.

Le Basic de l'Enterprise semble déjà ressembler au Pascal, au moins dans sa philosophie. Et ce n'est pas terminé : il reste encore à examiner deux structures pascaliennes. La première apparaît dans la manière d'employer les variables et surtout les tableaux. Leur prédéclaration stipule le type du contenu, numérique ou chaîne. Avec les chaînes, on précise également leur longueur maximum autorisée (132 caractères par défaut). Autre point de similitude avec le Pascal : les procédures. Leurs bornes sont précisées par DEF et END DEF, et l'appel est réalisé par CALL. Les variables locales n'ont pas besoin d'être mentionnées au début de la procédure. Si elles n'ont pas été utilisées dans le programme principal, elles sont automatiquement reconnues comme locales. Il existe même des variables factices (*dummy*) qui rendent les procédures parfaitement transparentes pour le passage des arguments.

Le traitement des erreurs s'opère également sous une forme de procédure définie entre WHEN EXCEPTION et END WHEN. Seule différence, l'appel ne se fait pas par un CALL, mais directement par la cause de l'erreur.

Si ce Basic présente des similitudes pascaliennes, la gestion du graphisme s'inspire plutôt de Logo. On y rencontre une instruction à tout faire : PLOT. Elle permet de tracer des droites, des lignes brisées en tournant d'un angle précisé par ANGLE. Elle dessine également des cercles ou ellipses (PLOT ELLIPSE) et sait colorier des figures (PLOT PAINT).

Le son réagit à un classique SOUND opérant sur trois canaux avec sortie stéréo et modulation d'enveloppe.

Ouvert aux communications

Dernier point intéressant de l'Enterprise, et non des moindres, sa faculté d'opérer en réseau. Les échanges d'informations sont régis directement depuis le Basic, grâce à SET NET NUMBER suivi d'un numéro de 0 à 32, pour ouvrir les canaux de communication. Ensuite, les instructions SAVE et LOAD servent à envoyer et à recevoir des programmes tandis que le dialogue interactif se fait avec les instructions PRINT et LINE INPUT après ouverture du canal par OPEN.

Le Basic de l'Enterprise sort agréablement des sentiers battus. Mais ses originalités ne présentent pas que des avantages. D'abord, l'interpréteur est très lent. Il faut préciser que le mode de codage décimal codé binaire (DCB) des variables défavorise l'Enterprise dans des courses contre la montre. En effet, une boucle vide FOR...NEXT de 1 à 10000 tourne pendant 83 secondes (ce qui est très long, par rapport aux autres Basic), alors que la même boucle vide de 0 à 9999 ne met plus que 21 secondes. Et ces 21 secondes représentent encore un score médiocre par rapport à d'autres Basic.

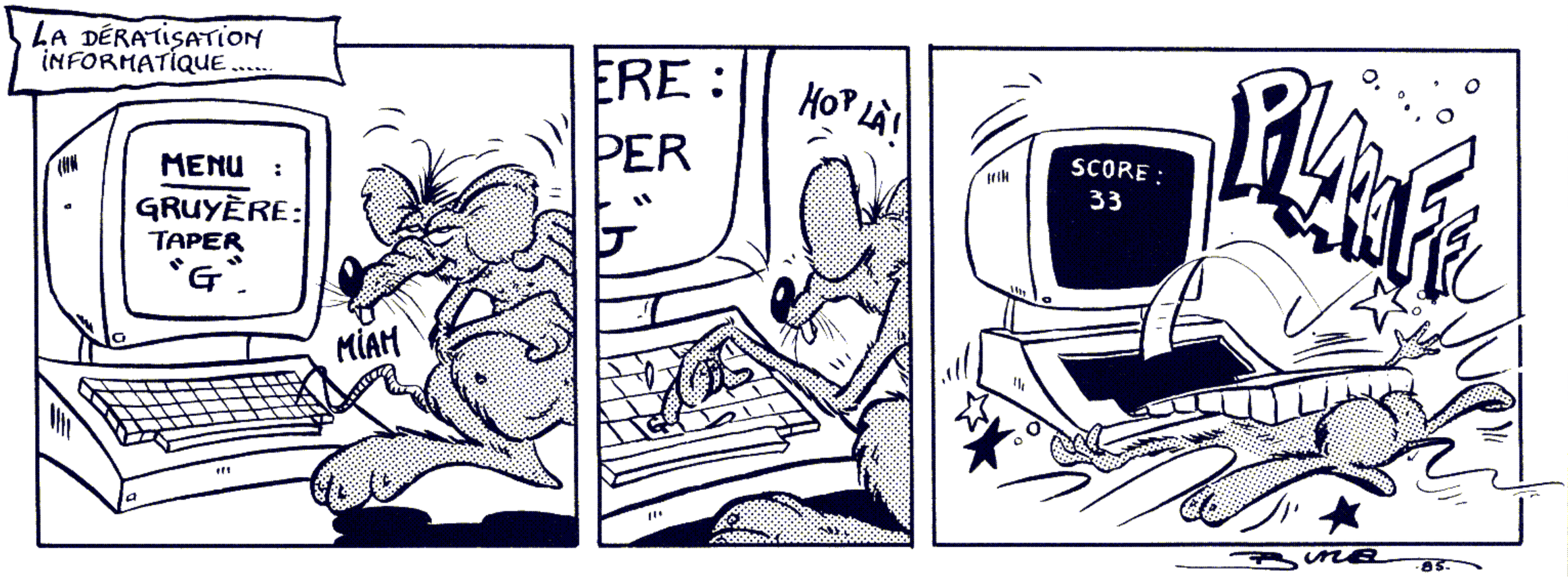
Un autre inconvénient de l'IS Basic (c'est le nom de ce Basic) est son abord rébarbatif, il a trop oublié les facilités du modèle standard. Résultat : les programmeurs en IS Basic écriront des œuvres très belles, certes, mais inexploitables sur d'autres machines et très difficiles à traduire.

On retrouve là encore un problème qui n'est pas près d'être résolu : uniformité ou originalité ?

Xavier de LA TULLAYE

Fiche technique du Lansay 64

Constructeur : *Enterprise (Grande-Bretagne)*
Importateur exclusif : *Lansay*
Prix public : *2 990 FF*
Processeur : *Z 80 A*
Mémoire vive : *64 Ko dont 49,4 Ko disponibles à l'utilisateur*
Langage : *IS Basic sur cartouche de mémoire morte*
Variables numériques : *jusqu'à 1E62 et sans spécification entre variables entières ou décimales*
Précision : *9 chiffres*
Nombre de mots clés du Basic : *188*



PETIT i

AVEC OU SANS COMPLEXE

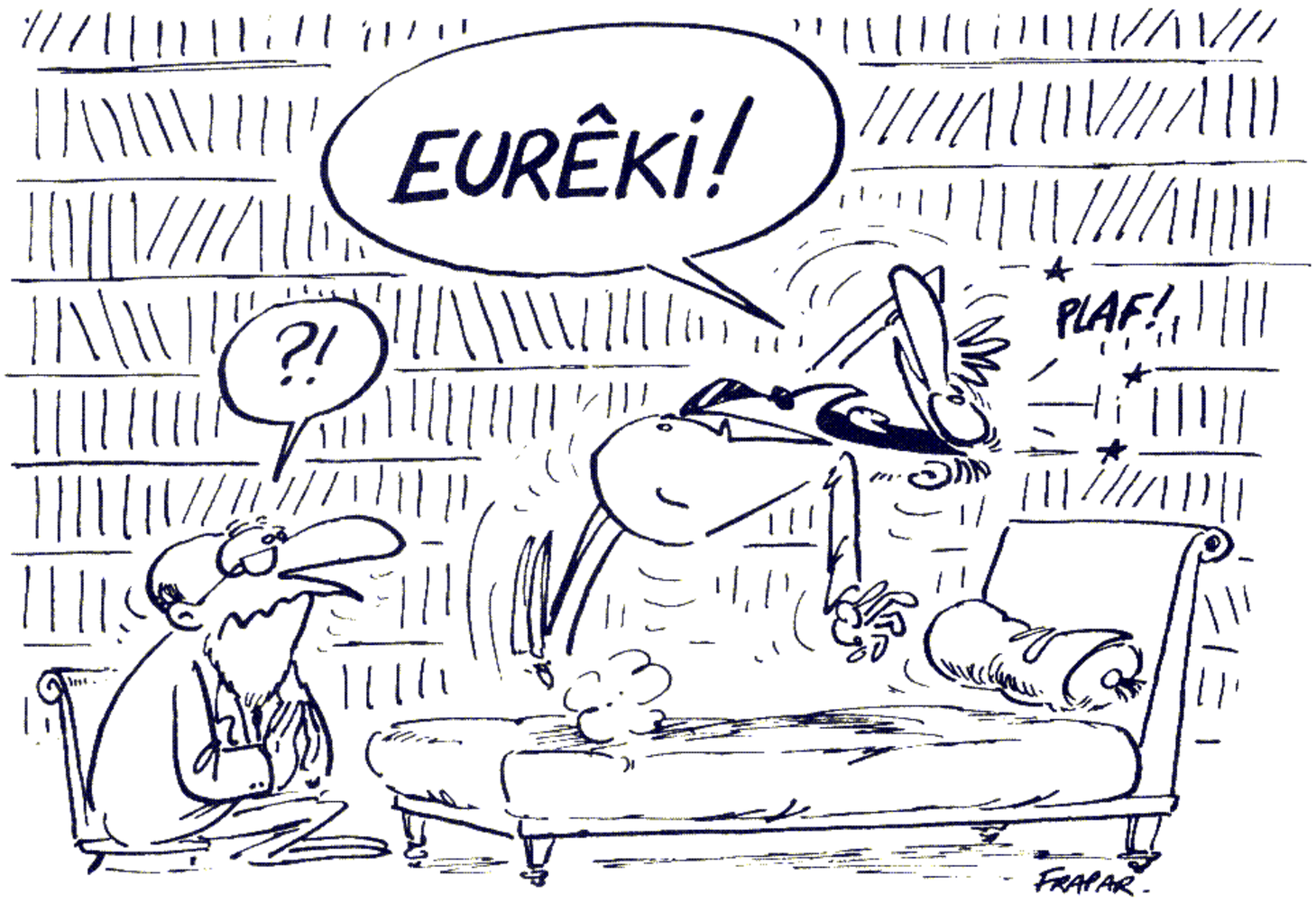
L'évaluation d'expressions complexes assistée par ordinateur fait ses premiers pas sur un PC-1500. Si vous avez un problème de complexes, le programme « Œdipe » l'interprétera et nul doute qu'il saura bien vous proposer une solution...

Savez-vous que $i^2 = -1$ ou encore $\sqrt{-1} = i$? Voici certainement de quoi donner des complexes au plus réel des nombres ! Un beau soir, un mathématicien navré de ne pas trouver de solution à l'équation $X^2 + 1 = 0$ décida d'inventer le nombre i, solution de ce mesquin petit problème. C'était dit, $i = X$ tel que $\sqrt{-1} = X$... Inutile de discuter.

Sans doute était-il loin de soupçonner la puissance de l'outil qu'il venait de concevoir, ainsi que les nombreux problèmes qu'il poserait à des générations d'étudiants... Cette partie dite imaginaire "i" qui introduit les nombres complexes servira aussi à distinguer le programme Œdipe.

Petite devinette : que vaut $(i-1)*(i+1)$? Réponse : -2 ; il suffit de développer normalement le produit et de remplacer en fin de calcul l'imaginaire i^2 par son équivalent -1.

Une grosse devinette maintenant, que vaut : $H(S) = ((S^2 - 2*S + i)*(3*S - 2))/(e^S*(3*S^2 + 12*S - 5))$? Réponse : demandez-le donc à Œdipe !



L'interprétation des expressions

Œdipe est conçu comme un sous-programme qui interprète une expression mathématique complexe introduite au clavier. Elle est traitée comme une chaîne de caractères et explorée lettre à lettre. En ce qui concerne les calculs, un organigramme est souvent plus parlant qu'un discours, voyez donc celui d'Œdipe, page suivante.

Une pile, en informatique, sert à ranger provisoirement des données dont on ne sait pas encore quoi faire. Par exemple, si Œdipe doit évaluer $1-2$, il devra mettre en attente le "1", puis prendre connaissance de l'opération en lisant le

"–" et calculer seulement après avoir lu le "2". En attendant, "1" et "–" ont été mis de côté en pile, la valeur un dans la pile des nombres et le signe "–", dans celle des opérations.

Il faut savoir aussi que certaines opérations sont prioritaires sur les autres. Par exemple, 1 + 2*3 vaut 7 car la multiplication l'emporte toujours sur l'addition. L'interpréteur devra donc empiler successivement les lettres 1, +, 2, * et 3 avant de calculer 2*3 et d'y ajouter 1.

En fait, ŒEdipe reconnaît les opérations suivantes :

+(A + i*B) prend l'identité,
–(A + i*B) prend l'opposé,
*(A + i*B) prend le conjugué,
e(A + i*B) prend l'exponentielle complexe,

(A + i*B) + (C + i*D) calcule la somme,
(A + i*B) – (C + i*D) calcule la différence,
(A + i*B)*(C + i*D) calcule le produit,
(A + i*B)/(C + i*D) calcule la division.

ŒEdipe interprète la chaîne alphabétique contenue dans B$(0) et place les résultats (partie réelle et partie imaginaire) dans A3 et A4. ŒEdipe n'est qu'un sous-programme, ce qui permet de le réutiliser dans d'autres programmes à la seule condition que ces derniers respectent la structure générale de celui qui est donné ci-contre en exemple.

L'imaginaire sur papier

Pour lancer l'application, faire DEF A. Un point d'interrogation apparaît, c'est ainsi qu'ŒEdipe réclame une expression à interpréter. On la tape alors au clavier, puis on presse sur ENTER. En fait, le mode opératoire d'ŒEdipe est identique à celui de l'ordinateur lorsqu'on effectue au clavier des calculs sur des nombres réels (j'allais écrire "normaux"). Pour préciser la partie imaginaire i, en minuscule dans l'équation, il convient de presser SHIFT I.

Si l'imprimante est connectée, les états du calcul et les résultats sont sauvegardés sur papier, sinon l'écran affiche à sa gauche la partie réelle et à droite l'imaginaire. Enfin, le PC-1500 admet pour ses calculs les abréviations telles que 3E8 à la place de 3*10^8. De même, ŒEdipe admet les abréviations portant sur i ; ainsi, on pourra écrire "i45" en lieu et place de i*45 à condition que "i" précède toujours le nombre (iE4 est incorrect, taper : i1E4 ou i*1E4).

En cas d'erreur d'écriture d'une équation, ŒEdipe affiche un point d'interrogation sur la lettre fautive dans l'expression mais il faudra la retaper toute entière corrigée.

Evidemment, certains calculs plus complexes demandent une certaine préparation. Ainsi 1/(√2*i) nécessite l'emploi d'une variable intermédiaire du Basic dans laquelle on rangera √2, par exemple C = √2. Ensuite, il suffira de demander à ŒEdipe d'interpréter 1/(C*i) ou 1/iC pour abréger.

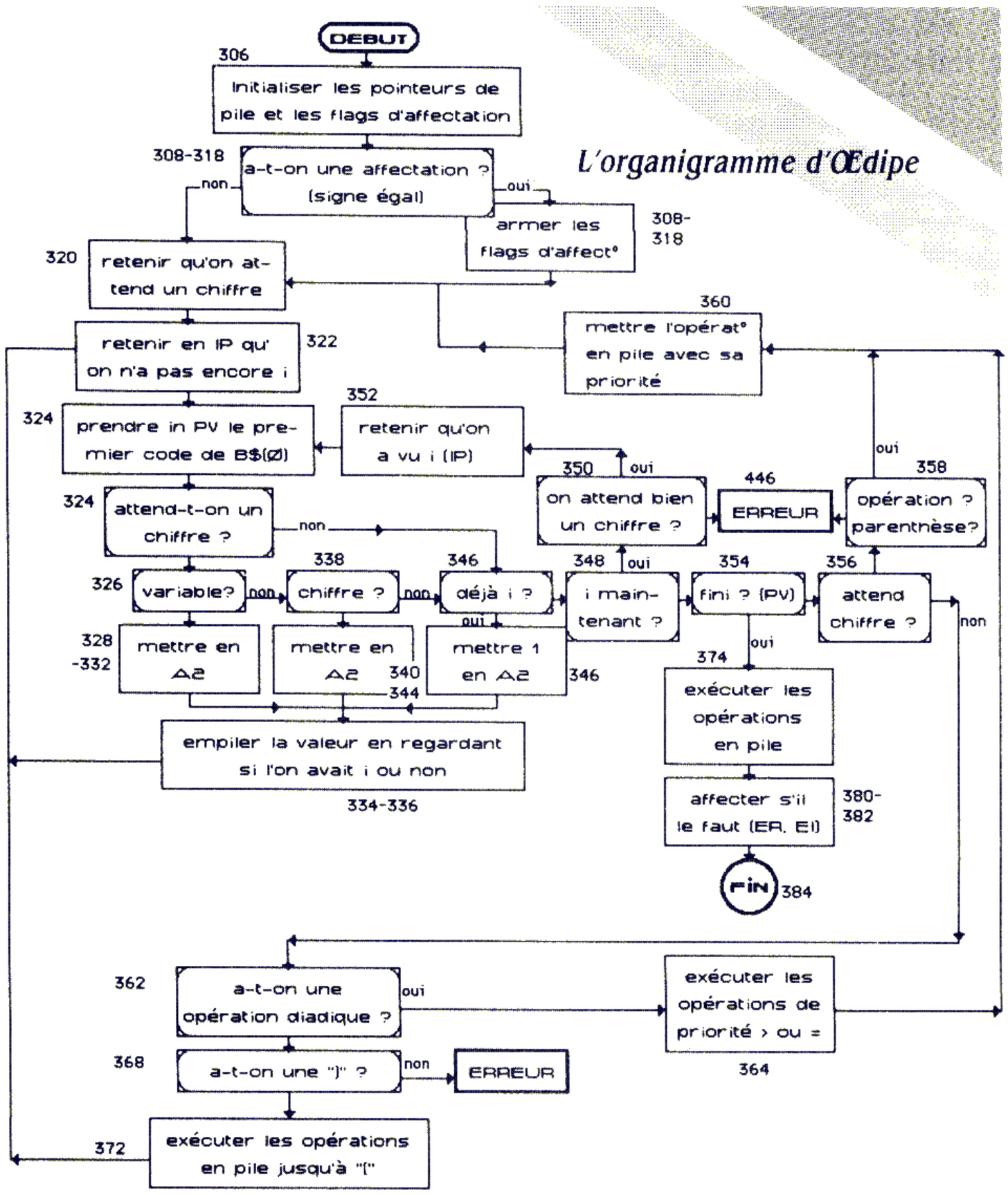
On pourra aussi utiliser toutes les variables A à Z du Basic pour stocker des résultats. Ainsi, on conservera la partie réelle en tapant : **A** = *nombre* ou

Programme d'utilisation d'ŒEdipe
Programme pour PC-1500
Auteur Frédéric Blondiau
Copyright *LIST* et l'auteur

```
10:"A"CALL 53393:
   DIM B$(0)*80
12:ON ERROR GOTO
   0:INPUT B$(0):
   ON ERROR GOTO
   16:LPRINT B$(0
   ):WAIT 0:GOTO
   0
14:WAIT :PRINT ">
   ":GOTO 12
16:GOSUB "OEDIPE"
18:"C"PRINT A3,A4
   :ON ERROR GOTO
   12:LPRINT "Re"
   ,"Im":LF -2:
   LPRINT A3,A4:
   LPRINT :GOTO 0
20:"V"A1=√(A3*A3+
   A4*A4):A2=0:IF
   A1<>0LET A2=
   ACS (A3/A1)*
   SGN ((A4>=0)-.
   5)
22:PRINT A1,A2:ON
   ERROR GOTO 12:
   LPRINT "Md","A
   r":LF -2:
   LPRINT A1,A2:
   LPRINT :GOTO 0
```

DEF A : active ŒEdipe
ENTER : désactive/réactive
DEF V : module et argument
DEF C : retour au mode rectangulaire

L'organigramme d'ŒEdipe



Exemples

```
A=12.22
Re                12.22
Im                    0

A=45.21-i*55.25
Re                45.21
Im               -55.25

A=i*18.54
Re                    0
Im                18.54

;B=87.21-i*25.25
Re                87.21
Im               -25.25
```

```
A
Re                    0
Im                    0

B
Re               -25.25
Im                    0

A;B=(5-i)/(5+5*i)
Re                  0.4
Im                 -0.6

A
                    0.4

B
                   -0.6

(A+i*B)/(12-i*B*5)
Re 1.960784314E-02
Im-5.490196079E-02

X;Y=
Re 1.960784314E-02
Im-5.490196079E-02

X
    1.960784314E-02

Y
   -5.490196079E-02

RADIAN
S=13
                   13

C=-√69
       -8.306623863
(S;C*S;C-2*S;C+i)*
(3*S;C-2)/(eS;C*(3
*S;C*S;C+12*S;C-5)
)
Re 7.826423168E-06
Im 2.316372735E-05
DEF V
Md 2.445017718E-05
Ar      1.244964678
```

ŒEdipe
Programme pour PC-1500
Auteur Frédéric Blondiau


```
300:"OEDIPE"ON
    ERROR GOTO 302
    :DIM P(1,16)
302:ON ERROR GOTO
    416
306:PP=0:PO=1:ER=0
    :EI=0:A1=1:Z$=
    ""
308:IF MID$ (B$(0)
    ,2,1)="="GOSUB
    392:GOTO 316
310:A1=2:IF MID$ (
    B$(0),3,1)="="
    GOSUB 390:EI=P
    V:GOTO 318
312:A1=3:IF MID$ (
    B$(0),4,1)<>"=
    "THEN 320
314:GOSUB 390:EI=P
    V,A1=1:GOSUB 3
    92:A1=3
316:ER=PV
318:A1=A1+2:GOSUB
    396
320:OA=0
322:IP=0
324:PV=ASC B$(0):
    IF OATHEN 346
326:IF PV<65OR PV>
    90THEN 338
328:A1=1,A2=@(PV-6
    4):GOSUB 398
330:IF MID$ (B$(0)
    ,A1+1,1)=";"
    LET A1=3:GOSUB
    392:P(IP=0,PP)
    =@(PV)*SGN (.5
    -IP)
332:A1=A1+1
334:GOSUB 396
336:PP=PP+1,OA=1:
    GOTO 322
338:IF PV<48OR PV>
    57IF PV<>46
    THEN 346
340:FOR A1=1TO LEN
    B$(0):PV=ASC
    MID$ (B$(0),A1
    ,1)
342:IF PV>47AND PV
    <58OR PV=46OR
    PV=69LET A1=A1
    +(PV=69):NEXT
    A1
344:A2=VAL B$(0):
    GOSUB 398:GOTO
    334
346:IF IPLET A2=1:
    GOSUB 398:GOTO
    336
348:IF PV<>105THEN
    354
350:IF OATHEN 0
352:IP=1:GOSUB 394
    :GOTO 324
354:IF PV=0THEN 37
    4
356:IF OATHEN 362
358:PS=0:IF PV<>43
    IF PV<>45IF PV
    <>42IF PV<>101
    LET PS=5:IF PV
    <>40THEN 0
360:@$(PO)=STR$ PS
    +CHR$ PV,PO=PO
    +1:GOSUB 394:
    GOTO 320
362:PS=2:IF PV<>43
    IF PV<>45LET P
    S=1:IF PV<>42
    IF PV<>47THEN
    368
364:IF PO>1IF VAL
    @$(PO-1)<=PS
    GOSUB 378:GOTO
    364
366:GOTO 360
368:IF PV<>41THEN
    0
370:IF @$(PO-1)<>"
    5("GOSUB 378:
    GOTO 370
372:PO=PO-1:GOSUB
    394:GOTO 322
374:IF PO>1GOSUB 3
    78:GOTO 374
376:PP=PP+(PP=0)
378:A3=P(PP-1),A4=
    P(1,PP-1):IF P
    O>1LET PO=PO-1
    :GOSUB @$(PO):
    P(PP-1)=A3,P(1
    ,PP-1)=A4:
    RETURN
380:IF ERLET @(ER)
    =A3
382:IF EILET @(EI)
    =A4
384:RETURN
390:IF MID$ (B$(0)
    ,A1-1,1)<>";"
    THEN 0
392:PV=ASC (MID$ (
    B$(0),A1,1))-6
    4:RETURN
394:A1=2
396:Z$=RIGHT$ (Z$+
    LEFT$ (B$(0),A
    1-1),16),B$(0)
    =MID$ (B$(0),A
    1,80):RETURN
398:P(IP,PP)=A2,P(
    IP=0,PP)=0:
    RETURN
400:"0-"A3=-A3
402:"0*"A4=-A4
404:"0+"RETURN
406:"0e"A2=EXP A3,
    A3=A2*COS A4,A
    4=A2*SIN A4:
    RETURN
408:"2-"GOSUB 400
410:"2+"PP=PP-1,A3
    =P(PP-1)+A3,A4
    =P(1,PP-1)+A4:
    RETURN
412:"1/"A2=A3*A3+A
    4*A4,A3=A3/A2,
    A4=-A4/A2
414:"1*"PP=PP-1,A2
    =P(PP-1)*A3-P(
    1,PP-1)*A4,A4=
    P(PP-1)*A4+P(1
    ,PP-1)*A3,A3=A
    2:RETURN
416:WAIT :BEEP 1:
    PRINT Z$+"?"+B
    $(0):END
```

Attention : il existe entre les différentes versions de PC-1500 des modifications du Basic, notamment en ce qui concerne la boucle FOR...NEXT. La liste du programme ŒEdipe concerne les PC-1500 A et les dernières versions. Si vous possédez un « vieux » PC-1500, transformez la ligne 340 en :
340 : FOR A1 = 1 TO LEN B$(0) + 1 : etc.

calcul complexe (par exemple A = i*18.54 – 45.21), et la partie imaginaire avec **;B = ...** (ne pas omettre le point-virgule : ";B = 87.21 – i*25.25").

Ainsi, il devient possible d'exprimer une expression sophistiquée comme (A + i*B)*(C + i*D)/(E + i*F) avec seulement A;B*C;D/E;F sans parenthèses et signes inutiles si l'on a évidemment affecté correctement les variables A à F.

La quasi-totalité des calculs complexes est donc réalisable par ŒEdipe. Il accepte jusqu'à 80 caractères pour l'expression, ou 25 opérations et 16 nombres en piles d'attente.

Frédéric BLONDIAU

LOGO

PAS A PAS

NOUS avons défini dans le numéro 9 de LIST un mode hybride permettant l'exécution en mode direct d'instructions qui sont pourtant conservées au sein d'une procédure nommée a posteriori. Nous abordons aujourd'hui un autre mode : le pas à pas grâce auquel l'exécution d'une procédure est visualisée instruction par instruction.

Contrairement à ce que nous avions fait dans les articles précédents, nous allons d'abord écrire les procédures qui nous intéressent, puis les analyser et donner ensuite une version moins concise, mais peut-être plus simple pour les débutants. La principale caractéristique des premières procédures est l'absence de variables globales, sauf en ce qui concerne la sauvegarde de la procédure originale que l'on doit restituer à la fin du pas à pas.

```
POUR PASAPAS :P
COPIEDEF MOT ''.  :P  :P
DEFINIS :P (LISTE PREMIER TEXTE :P
  (PH [EC PH [ENTREE DANS]] MOT
  '''' :P PARAMETRES PREMIER
  TEXTE :P [] CORPS SP TEXTE :P [])
FIN

POUR PARAMETRES :L :RES
SI :L = [] [RETOURNE :RES]
RETOURNE PARAMETRES SP :L (PH
  :RES [(EC [LA VALEUR DE ]] MOT ''''
  PREMIER :L ''EST MOT '': PREMIER
  :L''))
FIN

POUR CORPS :L :RES
SI :L = [][RETOURNE :RES]
RETOURNE CORPS SP :L (PH :RES
  ''TAPE (LISTE PREMIER :L)[DONNE
  ''A. LISLISTE] PREMIER :L)
FIN
```

Voilà les trois procédures nécessaires pour transformer une procédure en procédure pas à pas.

Quel en est le principe ? Il s'agit, après avoir sauvé le texte original, de réécrire la procédure de manière à ce qu'elle s'exécute instruction par instruction avec attente du caractère *retour* entre deux étapes de l'exécution. Prenons un exemple :

```
POUR SPI :N
AV :N DR 90
SPI :N + 10
FIN
```

va devenir :

```
POUR SPI :N
EC [ENTREE DANS SPI]
(EC [LA VALEUR DE] ''N [EST] :N)
EC [AV :N DR 90]
DONNE ''A. LISLISTE
AV :N DR 90
EC [SPI :N + 10]
DONNE ''A. LISLISTE
SPI :N + 10
FIN
```

Et notre procédure nouvelle manière s'exécutera de la façon suivante :

```
SPI 10
ENTREE DANS SPI
LA VALEUR DE N EST 10
AV :N DR 90
```

(Appui sur *retour* et exécution de l'instruction.)

```
SPI :N + 10
```

(Appui sur *retour* et exécution de l'instruction.)

```
ENTREE DANS SPI
LA VALEUR DE N EST 20
AV :N DR 90
```

(Appui sur *retour* et exécution de l'instruction.) Etc.

Attention aux faux pas

Le mode pas à pas est donc très « mémorivore ». Non seulement il conserve la procédure originale mais en crée une nouvelle qui compte trois fois plus d'instructions. Ainsi, chaque instruction devient :

```
EC [instruction]
DONNE ''A. LISLISTE qui attend un
   caractère retour
instruction
```

Pour mettre fin au pas à pas, nous écrirons :

```
POUR FINPASAPAS :P
COPIEDEF :P MOT ''.  :P
EN MOT ''.  :O
FIN
```

Attention aux étourderies : si une procédure a déjà été convertie et qu'elle est, par inadvertance, transformée une

seconde fois en mode pas à pas, elle risque fort de consommer beaucoup de mémoire. Mais il y a pire : vous allez exécuter un pas.à pas du pas à pas, et surtout vous aurez perdu l'original qui est maintenant le pas à pas du véritable original. Vous aurez beaucoup de mal à retrouver votre première version.

Deuxième piège : lorsqu'une procédure est en mode pas à pas, elle se présente sous la forme d'une liste. Si vous l'éditez, il faut à la fois modifier le véritable corps de la procédure et les commentaires. Or cela devient pratiquement impossible si vous ajoutez une instruction. Il faut donc penser à revenir à l'original avant de l'éditer, puis éventuellement remettre ensuite la procédure en pas à pas.

```
PASAPAS ''SPI
SPI 20
.....
FIN PASAPAS ''SPI
ED ''SPI
.....
.....
PASAPAS ''SPI
```

Bien sûr, ce pas à pas n'est pas très souple, mais il est créé pour les versions de Logo qui en sont dépourvues. Malheureusement, il reste certaines versions qui ne possèdent pas le pas à pas et qui n'ont pas non plus la primitive DEFINIS. C'est entre autres le cas des versions pour matériels Thomson. Donc aucun secours lorsqu'il s'agit de déboguer une procédure.

Examinons maintenant les procédures de pas à pas. En Logo, la primitive DEFINIS permet de définir une procédure sous forme de liste de listes. La première liste est obligatoirement celle des paramètres. Chaque instruction du corps de la procédure est une liste. Ainsi SPI pourrait se définir par :

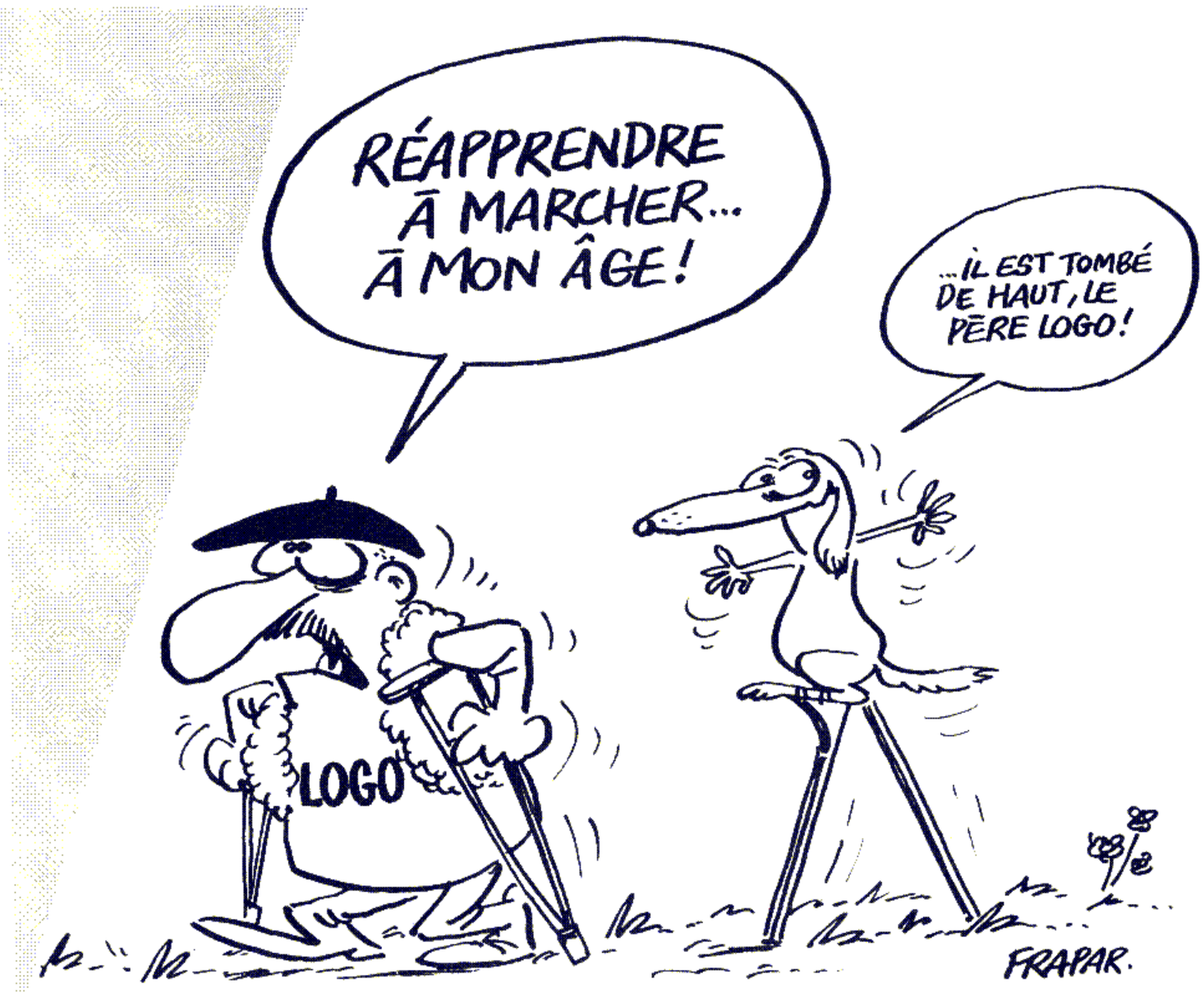
```
DONNE ''SPI [[N][AV :N DR 90][SPI :N
  + 10]]
```

Dans la liste des paramètres, le caractère deux-points (:) n'apparaît plus. Les mots POUR et FIN non plus. La primitive TEXTE permet de transformer une procédure écrite avec POUR-FIN en une liste.

```
DONNE ''SPI TEXTE :SPI
```

Examinons la procédure PASAPAS. Tout d'abord, PREMIER TEXTE :P représente la liste des paramètres qu'il faut conserver comme première liste de la nouvelle procédure.

Le but de la procédure PARAMETRES est de créer pour chaque paramètre une instruction qui affichera : LA VALEUR DE nom du paramètre EST



valeur du paramètre, puis de retourner l'ensemble des instructions ainsi créées à la procédure d'appel (PASAPAS).

Il en va de même pour le corps de la procédure, qui est SAUFPREMIER TEXTE :P. La procédure CORPS retourne la liste des nouvelles instructions.

Pas touche à l'original

Essayons de réécrire ces procédures en passant par deux variables globales. La première DEFPROC sera la procédure écrite sous forme de liste. La seconde NOUVPROC sera la liste dans laquelle nous accumulerons toutes les instructions anciennes et nouvelles. Les parenthésages servent à spécifier que les primitives PHRASE, LISTE ou ECRIS admettent un nombre quelconque d'arguments.

```
POUR PASAPAS :P
COPIEDEF MOT ''.P :P
DONNE ''DEFPROC TEXTE :P
DONNE ''NOUVPROC []
DONNE ''NOUVPROC (PH :NOUV
  PROC [(EC [ENTREE DANS ]] MOT ''''
  :P '') )
PARAMETRES PREMIER :DEFPROC
CORPS SP :DEFPROC
DEFINIS :P LISTE PREMIER :DEFPROC
  :NOUVPROC
FIN
```

Ce qui se traduit par :

1. Créer une procédure dont le nom est celui de la procédure à mettre en pas à pas, ce nom étant précédé d'un point de manière à ne pas retomber sur une procédure existante. Ainsi, SPI devient .SPI. Cette copie permettra de restituer l'original en fin de pas à pas.

2. Créer une variable DEFPROC dans laquelle sera mémorisée la procédure initiale, mais sous forme de liste de listes.

3. Initialiser une variable globale NOUVPROC à [].

4. Ajouter à NOUVPROC la liste correspondant à l'instruction (EC [ENTREE DANS] ''nom de la procédure). Ce nom est généré par MOT dont le premier caractère est le guillemet — d'où les doubles guillemets — et le contenu du mot paramètre P. Il faut ensuite générer le caractère *parenthèse fermante* — d'où la suite guillemets-parenthèse fermante — avant de refermer la parenthèse ouverte devant PH. Nous retrouverons le même mécanisme dans les procédures PARAMETRES et CORPS.

5. Compléter la variable NOUVPROC en appelant deux procédures récursives PARAMETRES et CORPS.

6. Définir la nouvelle procédure en y faisant figurer, au début, la liste des paramètres.

Écrivons maintenant PARAMETRES et CORPS en distinguant bien les actions.

```
POUR PARAMETRES :L
SI :L = [][STOP]
DONNE ''NOUVPROC (PH :NOUV
  PROC [(EC [LA VALEUR DE ]] MOT
```

LOGO PAS A PAS

```
"" PREMIER :A "EST MOT ": PRE
  MIER :L") )
PARAMETRES SP :L
FIN

POUR CORPS : L
SI :L = [][STOP]
DONNE "NOUVPROC (PH :NOUV
  PROC "TAPE (LISTE PREMIER :L))
DONNE "NOUVPROC PH :NOUVPROC
  [DONNE "A. LISLISTE]
DONNE "NOUVPROC PH :NOUVPROC
  PREMIER :L
CORPS SP :L
FIN
```

Quelques remarques :

1. Dans CORPS, TAPE est utilisé à la place de EC qui fait revenir à la ligne. Il n'y aura donc pas de saut de ligne entre deux instructions exécutées.

2. DONNE "A. LISLISTE sert à attendre un caractère retour pour continuer. La valeur prise par A. n'a pas besoin d'être exploitée. On pourrait remplacer cette ligne par : SI LISCAR = "S [STOP]. Ainsi, le passage d'une instruction à l'autre se fera en tapant sur un caractère quelconque, sauf sur S qui arrête l'exécution du pas à pas.

Rappel de la syntaxe Logo

Logo est un langage procédural. Les procédures disponibles à l'initialisation sont appelées **primitives.** Celles que vous créez sont nommées **procédures.** Une procédure commence par le mot POUR et se termine par FIN.

En Logo, un nombre est écrit tel quel, éventuellement précédé d'un signe. Un mot est toujours précédé de guillemets (on ne referme pas les guillemets à la fin du mot). Une liste est encadrée de crochets carrés [].

Les noms de variables qui ne sont pas liés à leur contenu sont des mots. Enfin, le contenu de la variable "A est :A.

Il y a Logo et Logo...

La principale différence entre la première et la seconde série de procédures réside dans le fait que la première n'utilise pas de variables globales intermédiaires. Les procédures PARAMETRES et CORPS retournent des listes dont le nom RES (pour résultat) figure dans leur titre. Ces listes, à l'appel, sont initialisées à []. Faire figurer le nom du résultat dans le titre est une astuce puissante qui permet à la fois de limiter le nombre de variables globales et d'initialiser des variables à l'appel.

La première version, plus courte, est caractéristique de ce que vous obtiendrez lorsque vous manipulerez bien Logo. La seconde est plus proche des langages informatiques classiques et sera écrite par un utilisateur ayant pratiqué ces langages.

Si vous voulez devenir performant, essayez de créer le moins possible de variables et évitez au maximum le DONNE.

Signalons pour finir les variantes à connaître pour adapter ces procédures aux différents Logo :

- DONNE s'écrit aussi CREE, FIXE, RELIE.
- Le parenthésage peut ne pas exister. Il faut alors ajouter des primitives PH lorsque le nombre d'arguments est supérieur à 2. (PH :A :B :C) s'écrit PH PH :A :B :C.

S'il n'y a qu'un argument, il faut ajouter la liste vide : (PH :A) devient PH :A [], mais (LISTE :A) devient SD LISTE :A [].

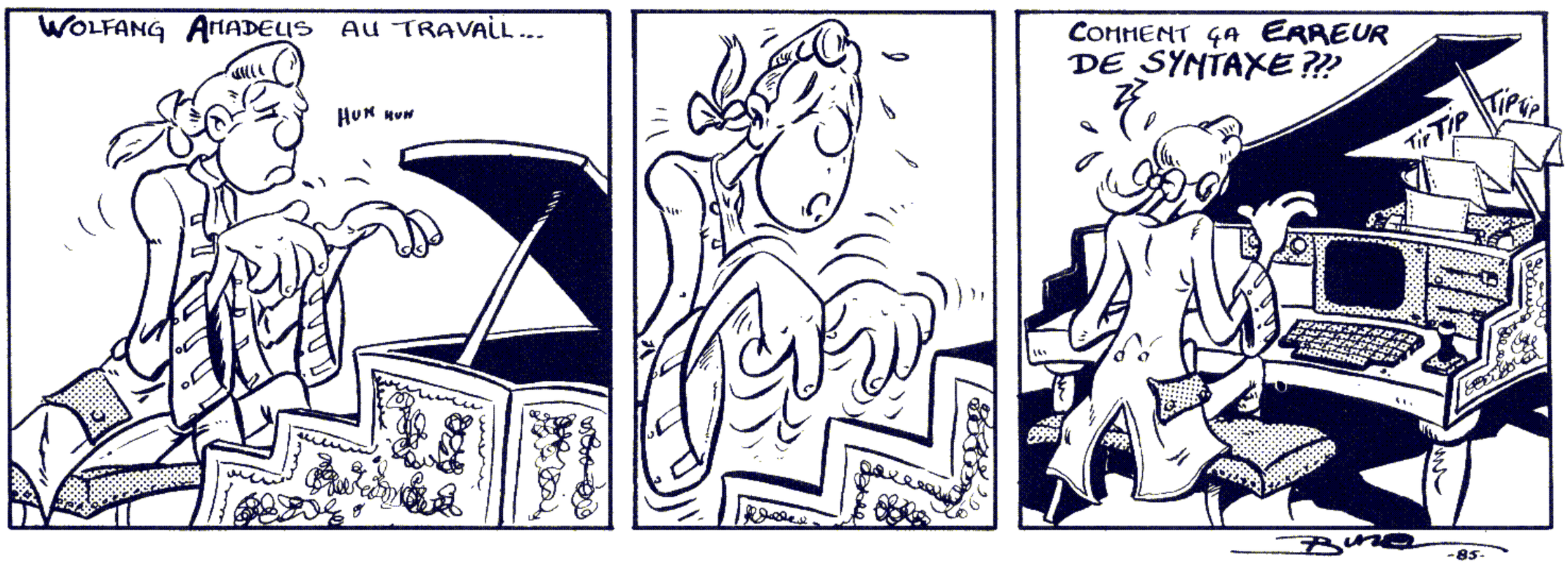
Il convient aussi de se méfier de (EC...). On devra décomposer en plusieurs TAPE suivis d'un ECRIS, mais on obtiendra des informations qui ne seront pas séparées par des espaces. Il faudra donc les réintroduire éventuellement. Ainsi, (EC :A :B :C) devient TAPE :A TAPE CAR 32 TAPE :B TAPE CAR 32 EC :C.

- SI :L = [] [STOP] s'écrit aussi SI :L = [] ALORS STOP ou SI VIDE? :L [STOP]
- LISLISTE peut devenir LISLIGNE ou LL.

Quant aux primitives COPIEDEF, TEXTE, DEFINIS, nous espérons qu'elles existent dans votre version, en remarquant que TEXTE pour la version Commodore 64 signifie MODE TEXTE et qu'il se traduit par DEFINITION.

Un prochain article sera consacré à des modifications plus ou moins illicites de procédures par d'autres.

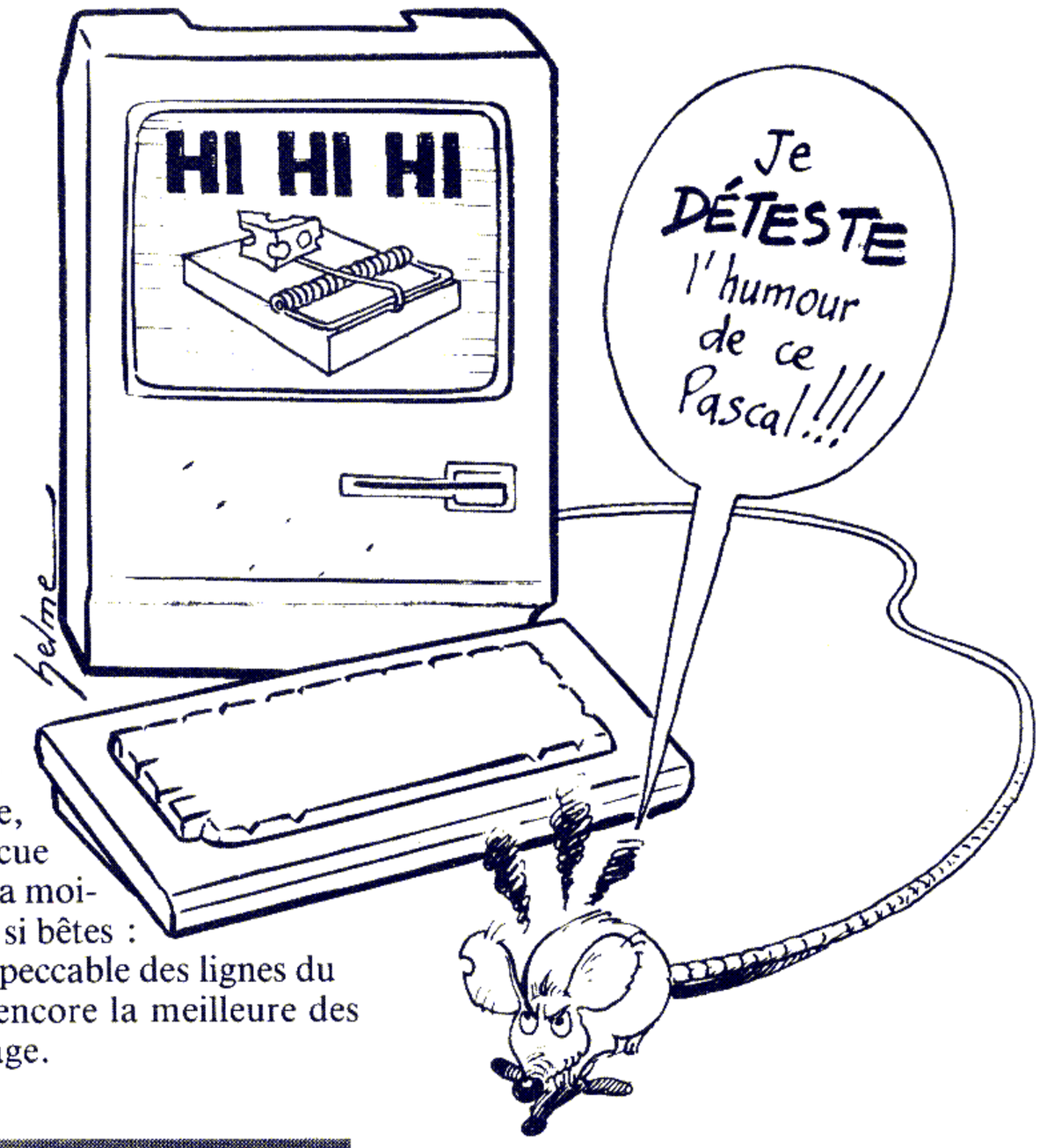
Robert DAGUESSE



NOUVEAUTÉ

UN PASCAL PAS COMME LES AUTRES

BIEN que de nombreux langages de programmation soient aujourd'hui disponibles sur Macintosh, le langage Pascal reste privilégié sur cette machine dont le système d'exploitation a été pensé et développé en Pascal. Toute la documentation technique fournie par Apple aux programmeurs, toutes les routines internes de la mémoire sont décrites et utilisables en Pascal.



Le Macintosh est en passe de devenir une machine à programmer, notamment en Pascal, un outil de développement d'applications. Non seulement il est équipé d'un excellent microprocesseur (le très rapide 32 bits 68000 de Motorola), et de 512 Ko de mémoire vive dans sa configuration optimale, mais il se dote aussi de langages performants tant pour l'apprentissage que pour la réalisation de programmes d'envergure professionnelle.

Le Pascal du Macintosh est d'abord un Pascal *interprété* doté d'une grande souplesse d'emploi. L'intérêt de l'interprétation se révèle surtout dans la phase d'apprentissage du langage : à mesure que l'on écrit le programme, il est contrôlé par l'ordinateur qui signale les fautes de syntaxe et organise de lui-même la mise en page et l'indentation des lignes. Comme le Pascal est un langage très strict sur le plan de l'écriture, ce contrôle évacue instantanément la moitié de ces erreurs si bêtes : la disposition impeccable des lignes du programme est encore la meilleure des aides au débogage.

Adieu les phases pénibles

Qui connaît le Pascal traditionnel sait que la réalisation d'un programme passe au moins par deux pénibles phases : l'écriture et la compilation. Sans entrer dans le détail, disons que, dans un premier temps, on écrivait le texte du programme comme s'il s'était agi d'une lettre à un ami, sans provoquer la plus petite réaction de l'ordinateur (pas de contrôle). Dans une deuxième phase, on lançait la compilation de ce texte, c'est-à-dire, en bloc, l'interprétation, le contrôle et, enfin, la production d'un code spécial exécutable par l'ordinateur. Seulement, à la moindre petite erreur, idiote souvent, comme l'oubli d'un simple point virgule, tout s'arrêtait et il fallait recommencer. Pour écrire le pro-

UN PASCAL PAS COMME LES AUTRES

gramme, il fallait un programme d'édition et pour le compiler, un programme compilateur. Et l'on passait souvent, trop souvent, de l'un à l'autre.

Avec ce nouveau Pascal, adieu les va-et-vient incessants entre l'édition et la compilation ! Comme dans un Basic classique, interprété, il n'y a pas de phase de compilation séparée. C'est à l'exécution que chaque instruction est compilée séparément et exécutée. En cas d'erreur, corriger est pratiquement un jeu d'enfant puisqu'on a toujours le programme à l'écran. D'ailleurs, ce langage fait encore mieux : il signale l'instruction qui a causé l'erreur et affiche un message d'explication (en anglais). Il est parfois si précis qu'on se demande pourquoi il n'effectue pas de lui-même la correction...

L'environnement Macintosh impose aux applications un « look » spécifique : fenêtres, menus et dialogues en sont les clefs. Une fenêtre est une zone de l'écran, bien marquée, qui reçoit un type déterminé d'informations. Ainsi, le Pascal occupe plusieurs fenêtres pour le programme d'écriture (nommé *Bullsey* dans l'exemple page suivante) et les affichages qu'il produit, texte *(Text)* ou dessin *(Drawing)*. A cela s'ajoutent deux autres fenêtres *(Observe* et *Instant)* qui sont en fait des instruments d'aide à la programmation. La barre de titre de la fenêtre active apparaît en grisé, ornée de ses instruments : la case de fermeture de fenêtre (haut et gauche), les barres de défilement (bas et le cas échéant à droite). Pour activer une fenêtre (elle revient au premier plan et on peut y écrire) il suffit de la désigner avec la souris (représentée par la petite flèche à l'écran) et de presser son bouton (cliquer).

C'est à l'aide de cinq menus qu'on contrôle le Pascal, l'édition, le débogage et l'exécution : avec *File, Edit, Search, Run* et *Windows*. Le premier de ces menus contient les choix relatifs aux disquettes, ouvertures et sauvegardes, et l'impression de programmes. Le second, classique sur Macintosh, regroupe les fonctions de *couper-copier-coller* qui assouplissent les possibilités d'édition en permettant le déplacement *(couper)* ou la recopie *(copier)* d'une partie d'un programme à un autre endroit *(coller)*. Le menu *Search* termine ces fonctions de traitement de texte par la recherche dans le programme d'un texte quelconque, voire son remplacement automatique par un autre.

Le menu *Run,* qui contient les options *Check, Reset, Go, Go-Go, Step, Step-Step* et *Stop In,* est le cœur de ce Pascal. *Check* commande la vérification du programme sans pour autant l'exécuter. Si *Go* est l'instruction classique de démarrage d'un programme, *Step* permet son exécution pas à pas, une instruction seulement, tandis que *Step-Step,* en ralentissant l'exécution, permet de suivre à l'écran le cheminement du programme : l'instruction en cours d'exécution est désignée à l'écran. Idéal pour rechercher une erreur ou plus simplement bien comprendre le déroulement d'un programme.

Avec *Stop In,* il est même possible d'installer des points d'arrêt sur certaines instructions : à l'exécution du programme, une pause se produira durant laquelle on pourra vérifier les informations de son choix, et on redémarrera avec *Go.* Enfin, *Go-Go,* transforme tous ces arrêts en brèves pauses pendant lesquelles est actualisée la fenêtre *Observe.*

Programmer avec les instruments

Activable, comme toute fenêtre, depuis le menu *Windows, Observe* peut recevoir n'importe quelles expressions (variables, calculs,...) qui seront calculées et affichées à la moindre pause, à tout arrêt, et même en permanence en cas d'exécution automatique pas à pas *(Step-Step).* Cette possibilité donne au programmeur le moyen de contrôler totalement l'état de ses variables durant tous les stades de l'exécution du programme. Avec la fenêtre *Instant,* on dispose d'un moyen d'exécuter à côté du programme n'importe quelle fonction du langage et d'en obtenir instantanément le résultat. Outil de test des fonctions et procédures, *Instant* permet aussi de forcer les variables employées par ailleurs dans le programme à prendre une certaine valeur à un moment donné : stopper l'exécution, modifier des valeurs grâce à *Instant* et reprendre l'exécution avec *Go.*

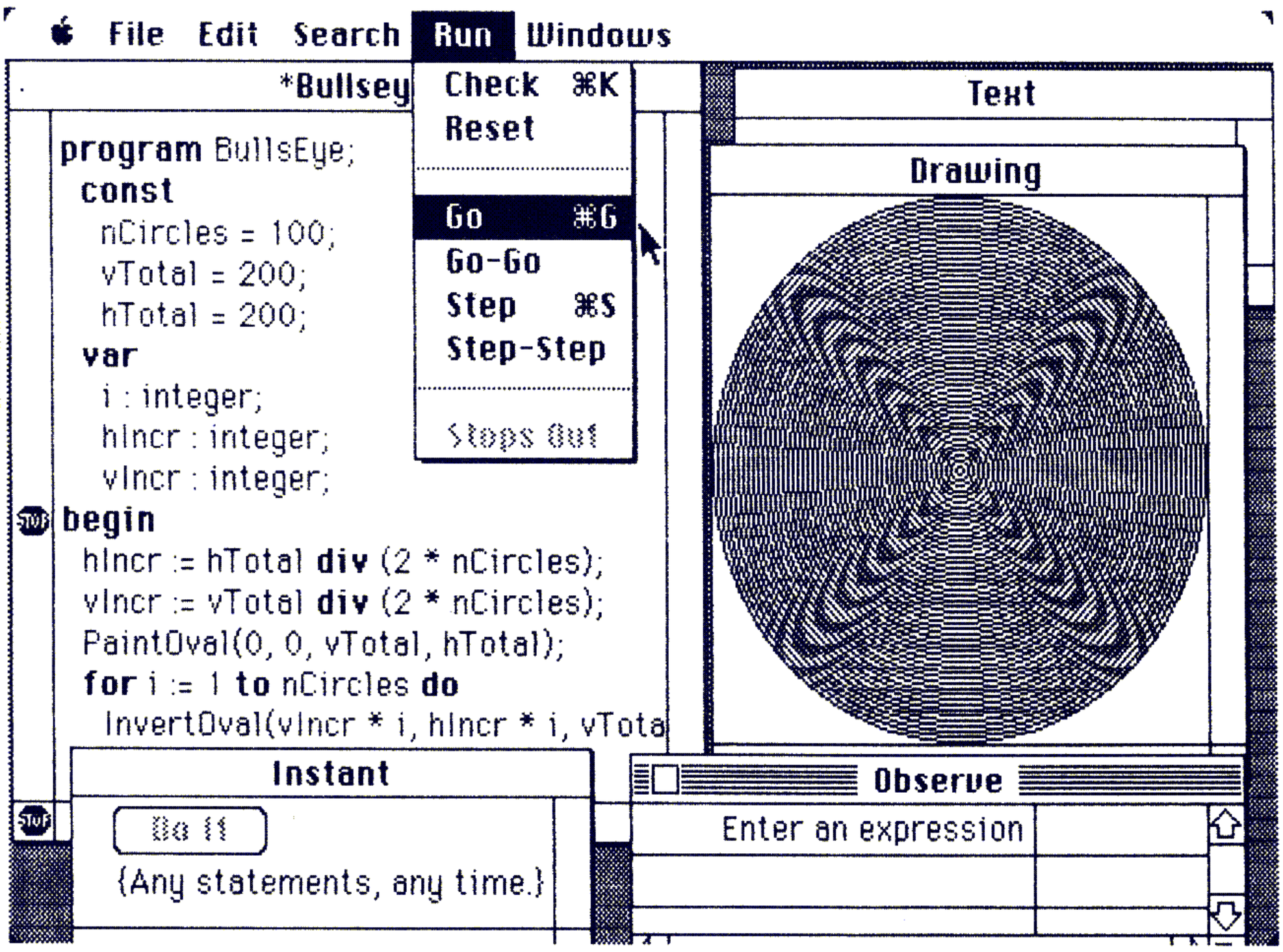
Mots réservés de Macintosh Pascal

and	nil
array	not
begin	of
case	or
const	otherwise
div	packed
do	procedure
downto	program
else	record
end	repeat
file	set
for	string
function	then
goto	to
if	type
in	until
label	uses
mode	var
	while
	with

Cette grande souplesse dans l'édition, le contrôle et l'exécution est l'atout principal de ce logiciel, mais les fonctionnalités du langage ne sont pas ignorées, loin de là. Ainsi, le jeu des caractères admis comprend les lettres (indifféremment majuscules ou minuscules car l'éditeur mettra automatiquement tout mot clef en minuscules et caractères gras), les chiffres et symboles spéciaux comme@, {, }, [,], etc. Malheureusement, s'agissant d'un produit américain, l'éditeur rejette systématiquement tous les caractères accentués français ; dommage, mais on pourra tout de même les afficher en recourant à la pénible gymnastique de la fonction CHR ().

Outre les types de données simples et classiques : *Integer* (entier de $2^{15}-1$ maximum), *Longint* (entier long jusqu'à $2^{31}-1$), *Boolean* (binaire) et *Char* (caractère), on trouve les types énumérés (combinaisons d'éléments), intervalles, et *Real* (nombres réels en précision simple [1.5×10^{-45} à 3.4×10^{38}], double [5.0×10^{-324} à 1.7×10^{308}] ou étendue [1.9×10^{-4932} à 1.1×10^{4932}]). Enfin, le type *Computational* est spécialisé dans les applications de gestion où le calcul exact est exigé. On le voit, ce Pascal est

Un programme en cours d'écriture et les différentes fenêtres correspondantes



bien armé pour ce qui concerne la manipulation de chiffres.

Et celle des chaînes de caractères n'est pas absente avec le type *String* (jusqu'à 255 caractères). En *Record*, on pourra se constituer des fiches de types composés d'un mélange des autres...

Goto toujours présent

Les tableaux ont la souplesse habituelle qui permet de se concocter des tableaux de tableaux de types standard ou particuliers : de belles salades en perspective... On en terminera avec les types standard de données en mentionnant les *Set types* (ensembles), *Files types* (fichiers), *Pointer-types* (pointeurs, très employés sur Macintosh).

Pascal est un langage structuré, on retrouve donc les *Procedure* et *Function* standard, les *Begin...End*, *If... Then...Else...*, *Case...Of...Otherwise...End*, *Repeat...Until*, *While... Do...*, *For...To* (ou Downto)...*Do...*, *With...Do...*, etc. Mais le *Goto* n'est pas absent pour autant.

L'utilisateur dispose en outre d'une importante batterie de procédures et fonctions prédéfinies. Cela va de la lecture et l'affichage de données, à l'appel de routines internes du Macintosh (comme *OldFileName* qui fait apparaître la fenêtre de dialogue pour choisir un fichier sur la disquette, avec la possibilité d'en changer, ou *NewFileName* qui réalise l'opération inverse en vérifiant que le nom tapé ne correspond pas déjà à un fichier existant) via des fonctions sophistiquées de dessins, affichages de textes, de calculs scientifiques et financiers, etc.

A deux points de vue, ce langage est une réussite : sur un plan standard, et comme outil spécifique au Macintosh.

Il reste en effet, relativement compatible avec la norme UCSD, classique désormais, et le *LisaPascal*. D'un côté donc, on pourra apprendre avec des ouvrages en français (comme le bon manuel *Apple Pascal* aux éditions McGraw Hill) car la documentation en langue anglaise est assez succincte, et reprendre sans grande modification des programmes Pascal classiques.

D'autre part, le programmeur peut employer la majorité des fonctions et procédures standard implantées dans les mémoires mortes du Macintosh, soit directement, soit via la fonction *InLine* (à condition de connaître les numéros d'appel de ces fonctions et leur syntaxe).

Il est techniquement possible de programmer une application similaire à Macwrite ou Macpaint — bien sûr aux temps d'exécution près —, de gérer soi-même fenêtres, menus, dialogues et tout ce qui fait la puissance de l'interface utilisateur du Macintosh. En fait, ce Pascal sera souvent préféré pour ses puissantes fonctionnalités d'édition et la mise au point de programmes, mais les versions utiles d'une certaine envergure seront avantageusement modifiées puis reprises définitivement par *Mac Advantage* (Pascal compilé) ou même *LisaPascal*.

Jean-Christophe KRUST

MISEZ P'TIT OPTIMISEZ

EXTREMUM OPTIMUM EST

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 !

Ah, le beau défi ! Par le nombre de participations reçues au journal, un nouveau record vient d'être battu : depuis deux ans (à un mois près) que vit la rubrique, jamais autant de courrier ne nous était parvenu.

Avec *Extremum optimum est*, il s'agissait de déterminer par programme l'extremum d'une classique équation du second degré : $Y = aX^2 + bX + c$. La solution, tant mathématique qu'informatique, est évidemment simplissime : c'est le point d'abscisse $-b/2a$. Ce calcul donne bien la valeur de X pour laquelle l'équation du second degré $aX^2 + bX + c$ est optimale, c'est-à-dire soit un maximum, soit un minimum.

Dans un premier temps donc on décrira techniquement le(s) meilleur(s) algorithmes dégagés par les lecteurs. Ensuite, on saisira l'occasion d'introduire le concept de dérivées première et seconde d'une fonction mathématique. Mais si... le lien avec l'extremum est facile !

Nous avons dit $-b/2a$, et Jean Thiberge, Robert Pulluard, Franck Wettstein, Thierry Coquard, Arnaud Peruta, Bernard Allaud et Denis Descause l'ont bien compris qui ont réalisé tous simultanément la meilleure routine de calcul.

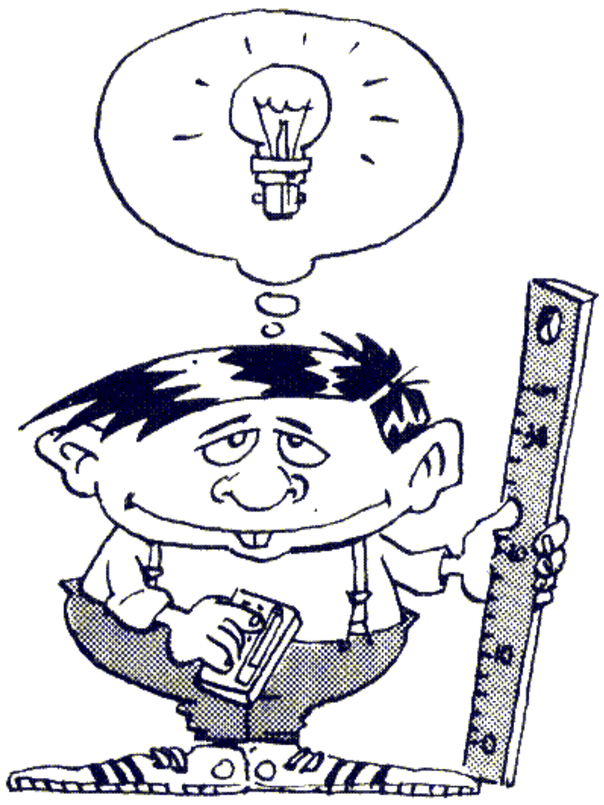
Cette équation ne tombe évidemment pas du ciel, elle est l'expression de la dérivée de la fonction $Y = aX^2 + bX + c$ soit $Y' = 2aX + b$ que l'on trouve en appliquant à chaque monome du type aX^n la transformation naX^{n-1} (alors on obtient $2aX^1$ et $1bX^0$ soit b). Si vous l'ignoriez, notez bien cette transformation car elle permet de retrouver simplement la plupart des formules des dérivées (sachant que $\sqrt{x} = x^{1/2}$, $1/x = x^{-1}$, etc.).

Maintenant, il convient de trouver la valeur de X pour laquelle Y est maximum ou minimum : il s'agit justement du point où la dérivée s'annule ! On posera donc $2aX + b = 0$ ce qui donne, enfin, $X = -b/2a$.

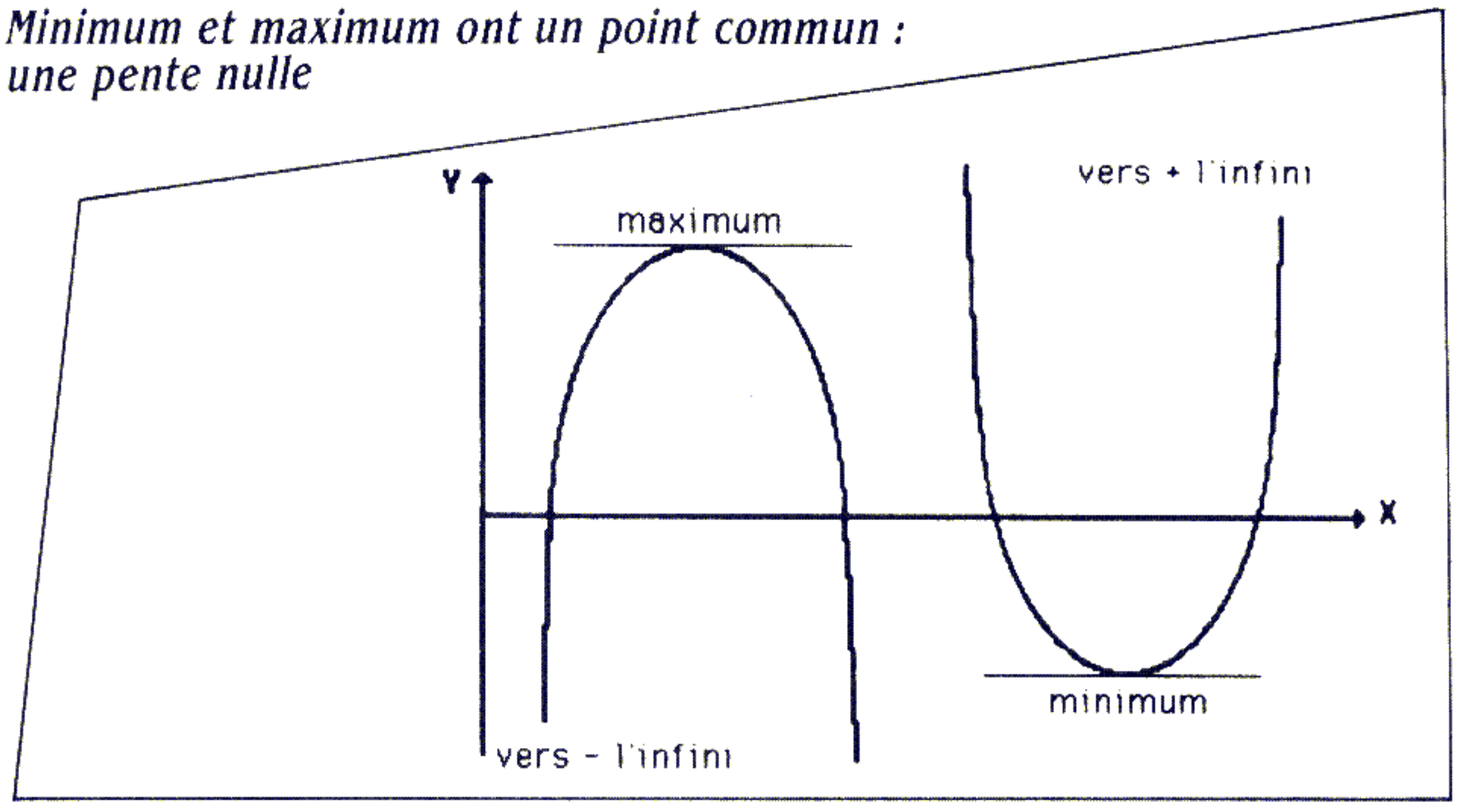
Des rappels informatiques

Pour ce qui est de l'expression informatique de ce calcul, rappelons qu'on devait introduire les coefficients a, b et c dans la pile opérationnelle de la manière suivante : a ENTER b ENTER c, et d'un XEQ allègre lancer le programme de calcul.

Laissons à Jean Thiberge — le premier, chronologiquement, des vainqueurs — le soin d'expliquer la démar-



Minimum et maximum ont un point commun : une pente nulle



che de la routine gagnante reproduite ci-dessous :

```
01  LBL ''EXTR''
02  X < > Y
03  CHS
04  2
05  /
06  ENTER↑
07  R↑
08  /
09  *
10  ST – Y
11  X < > L
12  .END.
```

« Parmi une douzaine de solutions légèrement différentes, tournant toutes en 12 octets de programme seulement (sans compter ni le LBL de tête, ni le END final), voici la plus rapide : 0,322 seconde sur une HP-41 CX.

« On aura pu déplacer à loisir le CHS dans le programme (éventuellement en changeant le ST – Y en ST + Y), ou remplacer CHS 2 par – 2 ce qui fait gagner 1 pas, aucun octet, et perdre 23,3 microsecondes...

*« Enfin, on pourra sacrifier 0,4 ms et 1 ligne en remplaçant *ST – YX < > L par STO T * – X < > Y ceci ayant pour but de conserver dans Z et T l'abscisse de l'extremum X au lieu de la désormais inutile constante c. »*

Calculer la dérivée d'une fonction mathématique en un point donné, c'est calculer sa pente, tout simplement. Ainsi, avec $f(x) = 3x^2 - 4x + 5$ et la transformation naX^{n-1} on obtient la formule de la dérivée : $f'(x) = 6x - 4$.

Comme on recherche un extremum (maximum ou minimum), une déduction s'impose du graphe ci-dessus : minimum et maximum ont un point commun, une pente nulle ! En effet, là, la tangente est symbolisée par un trait horizontal et chacun sait qu'alors si cela ne descend ni ne monte, c'est que la pente vaut zéro...

Et c'est pourquoi la solution du problème de la recherche de l'extremum d'une fonction réside toujours dans la simple annulation de sa fonction dérivée... Comme ici, s'agissant d'une équation du second degré, $f'(X) = 2aX + b$ et que cela est nul, on obtient bien $X = -b/2a$.

En haut ou en bas ?

Donc, on a la valeur $(-b/2a)$ de l'extremum X qui optimise le résultat Y. Mais s'agit-il d'un maximum ou d'un minimum ? C'était le défi de *LIST* n° 9 dont nous donnerons la solution informatique le mois prochain. Mais, déjà, compte tenu de ce qui précède, on peut orienter plus précisément les recherches qui n'auraient pas encore abouti...

Qu'est-ce qu'une dérivée seconde ? La fonction dérivée de la dérivée, tout simplement. Intuitivement, cette dérivée seconde serait la pente de la pente... On la calcule simplement, à l'aide de la même transformation naX^{n-1} que précédemment sur une équation du second degré :

$f'(X) = 2aX - b$

$f''(X) = 1\ (2a)\ X^0 = 2a$

Si cette dérivée seconde est positive, on se trouve toujours dans le cas où de part et d'autre du point optimal, la fonction remonte vers $+\infty$, alors il s'agit d'un minimum. Si la dérivée seconde est négative, la fonction va vers $-\infty$ et l'optimum était un maximum. Oserons-nous dire que tester le signe de 2a, c'est tester celui de a ? Si maintenant un optimiseur « sèche » encore sur la solution du défi du prochain *LIST*...

Jean-Christophe KRUST

QUI DIT MIEUX ?

COMME la détermination des points particuliers des courbes passionne, dérivons au troisième degré ! Trouver les coordonnées du point d'inflexion d'une courbe du troisième degré dont l'équation a la forme : $Y = aX^3 + bX^2 + cX + d$. Par exemple, pour $Y = X^3 - 9X^2 + 18X + 1$ ce point est en $X = 3$ et $Y = 1$.

Là encore, les coefficients a, b, c et d doivent être introduits dans l'ordre en pile opérationnelle (a ENTER b...) ne laissant que LASTX pour jongler un peu...

Ma, ou plutôt mes solutions ne nécessitent que 24 octets chacune et n'emploient que la pile opérationnelle. La première ''INFLEX'' comporte 19 lignes (toujours sans compter ni le LBL de tête, ni le END final) et s'exécute en 436 microsecondes environ, tandis que la seconde ''INFLEX'', est réduite à 18 lignes mais s'éternise... sur 503 microsecondes.

Bien sûr, les coordonnées X et Y du point d'inflexion sont chacune dans le registre du même nom à l'arrivée.

Robert PULLUARD

THOMSON

SAFARI-MÉMOIRE

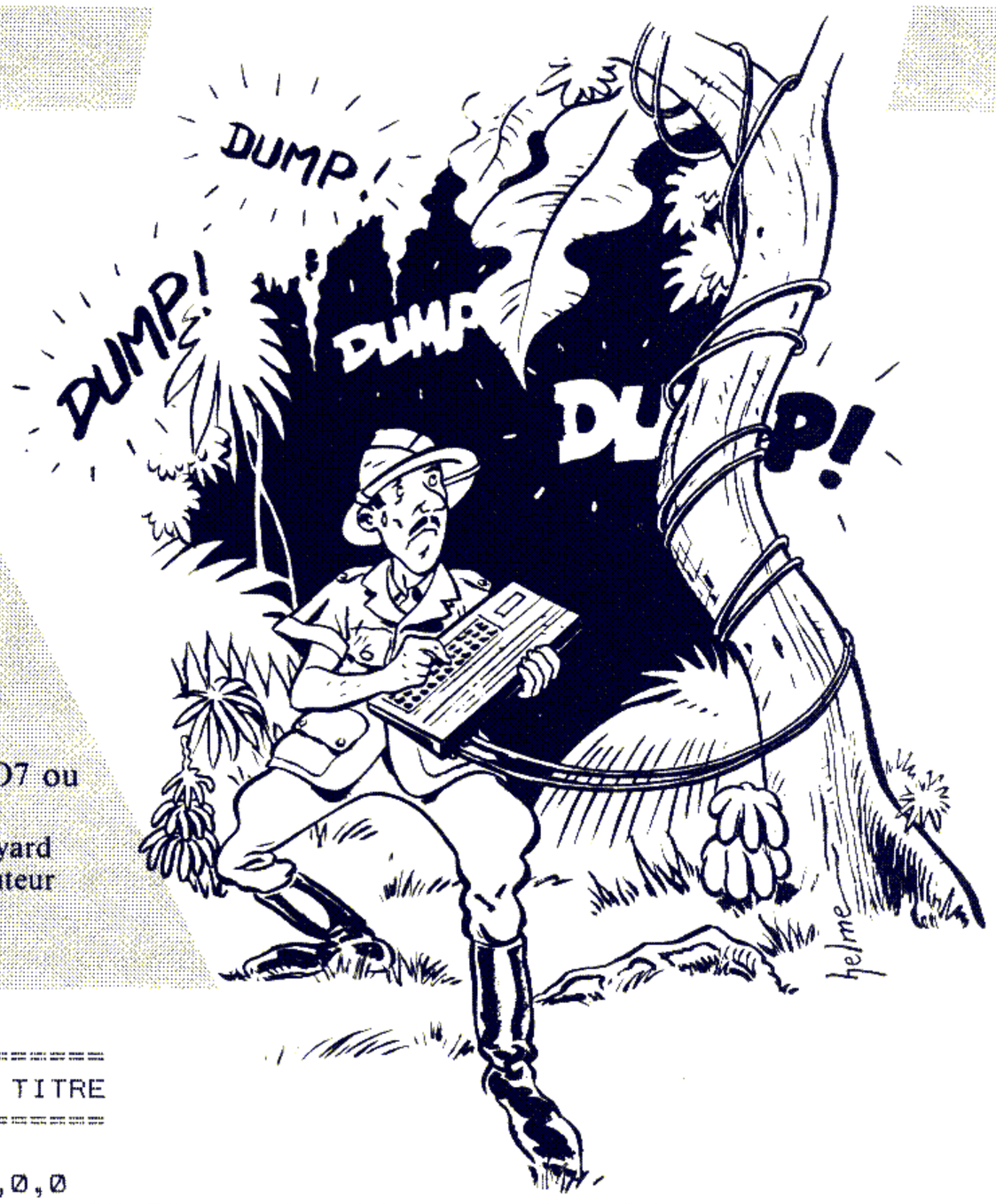
POUR explorer commodément les méandres de votre Thomson – à moins que ce ne soit celui de votre ministère de tutelle – rien de tel qu'un bon vieux DUMP HEXA, avec les codes ASCII en colonne de droite. Un petit programme Basic fera cela très bien. Naturellement, les REMarques ne sont là que par souci de clarté et vous pourrez vous dispenser de les frapper si vous avez hâte de voir le résultat.

Profitant des possibilités graphiques des ordinateurs Thomson, le programme de DUMP présenté ici va commencer par nous offrir une belle page de titre. Bordure et fond seront noirs (ligne 200), le message DUMP HEXA sera en quadruple taille et en bleu (ligne 210), les commandes seront en blanc et l'entrée de l'adresse de départ sera en rouge (lignes 220 à 270). On retrouvera ce bel affichage tricolore (matériel français oblige !) dans la boucle principale, puisqu'on sortira les adresses en bleu, les octets en blanc et les caractères ASCII en rouge.

Or donc, on a la possibilité, selon ses convictions personnelles de donner au programme l'adresse de départ, soit en décimal, soit en hexadécimal, auquel cas on est invité à la faire précéder du signe "$" pour que l'ordinateur s'y retrouve. La présence, dans le Basic Microsoft du TO7, de la fonction HEX$ et de la notation "&H" évite des acrobaties de conversion qui s'imposent sur le MO5. Rappelons que HEX$(x) renvoie une chaîne de caractères qui est la valeur hexadécimale du nombre décimal x, et que VAL("&H"+X$) renvoie la valeur décimale du nombre hexadécimal X$. La ligne 300 affiche donc l'adresse de départ A, en hexadécimal, et la formate avec ce qu'il faut éventuellement de zéros en tête. La boucle en 400-410 effectue simultanément deux tâches : elle prépare la traduction ASCII, contenue dans la chaîne A$, et affiche un par un les octets, eux aussi présentés en hexadécimal, avec zéro forcé en tête.

Des caractères impitoyablement éliminés

Mais pour l'affichage des caractères, on ne peut afficher les codes inférieurs à 31, qui sont des caractères de contrôle et viendraient perturber la belle ordonnance de notre écran. De même, les caractères de code supérieur à 127, n'appartenant pas au code ASCII proprement dit, seront impitoyablement éliminés et remplacés par des points (ligne 410). A la ligne 430, un compteur de lignes assure une présentation par page,



Safari-mémoire
Programme pour TO7 ou TO7/70
Auteur François J. Bayard


```
195 REM
196 REM ==============
197 REM PAGE DE TITRE
198 REM ==============
199 REM
200 CLS:SCREEN ,0,0
210 PRINT:ATTRB 1,1:COLOR 4:PRINT TAB(5);"DUMP HEXA"
220 ATTRB 0,0:PRINT:COLOR 7:PRINT"COMMANDES:":PRINT
230 PRINT"FLECHE EN HAUT:  ADRESSES PRECEDENTES"
240 PRINT"FLECHE EN BAS:   ADRESSES SUIVANTES"
250 PRINT"RETURN:          NOUVELLE ADRESSE"
260 PRINT"RAZ:             FIN":PRINT
270 COLOR 1:PRINT"ADRESSE DEPART (PREFIXE '$' SI HEXA)":INPUT A$
280 CLS:L=0:IF LEFT$(A$,1)="$" THEN A=VAL("&H"+MID$(A$,2)):GOTO 300
290 A=VAL(A$)
291 REM
292 REM ==================
293 REM BOUCLE DE LECTURE
294 REM ==================
295 REM
296 REM -------
297 REM ADRESSE
298 REM -------
299 REM
300 COLOR 4:PRINT RIGHT$("000"+HEX$(A),4);" ";:COLOR 7
395 REM
396 REM ------------------------------
397 REM OCTETS HEXA & CHAINE ASCII A$
398 REM ------------------------------
399 REM
400 A$="":FOR I=0 TO 7:X=PEEK(A+I):PRINT RIGHT$("0"+HEX$(X),2);" ";
410 IF X>31 AND X<128 THEN A$=A$+CHR$(X) ELSE A$=A$+"."
420 NEXT I:COLOR 1:PRINT A$:COLOR 2
430 L=L+1:A=A+8:IF L<24 THEN 300
440 L=0
495 REM
496 REM ---------------
497 REM ATTENTE CLAVIER
498 REM ---------------
499 REM
500 K$=INKEY$
510 IF K$=CHR$(13) THEN 270
520 IF K$=CHR$(10) THEN CLS:GOTO 300
530 IF K$=CHR$(11) THEN A=A-384:CLS:GOTO 300
540 IF K$=CHR$(12) THEN END
550 GOTO 500
```

Pour adapter ce programme au MO5, il suffit d'écrire une routine convertissant un nombre décimal en une chaîne hexadécimale.
L'instruction HEX$ effectue une telle conversion. Elle existe sur le TO7 (et le TO7/70), mais pas sur le MO5.

et l'affichage s'arrête dès que 24 lignes d'écran sont remplies.

On passe alors à la boucle d'attente des lignes 500 et suivantes. La touche de curseur *flèche en bas* provoque l'effacement de l'écran et l'affichage de la page suivante. La touche de curseur *flèche en haut* ôte de l'adresse en cours le nombre d'octets affichés dans deux pages-écran (2 fois 192) et affiche donc la page précédente. La touche RETURN renvoie à l'entrée d'une nouvelle adresse de départ. La touche RAZ, enfin, permet d'effectuer une gracieuse sortie.

Voir la mémoire depuis le début

Si donc vous répondez zéro à la question "Adresse départ ?", vous verrez le début de la mémoire Basic. Elle est signée (octets 0 à $19) et vous constaterez qu'à partir de l'octet $92, on a la liste des mots clés, tassés les uns contre les autres. Mais ils se distinguent les uns des autres par cette particularité que la dernière lettre de chaque mot a le bit de poids fort à 1, ou, si vous préférez, un code arbitrairement majoré de 128 (le premier chiffre hexa est C ou D au lieu de 4 ou 5). Vous demandez des preuves ? Bon d'accord, vous l'aurez voulu ! Ajoutez donc :

```
1000 FOR I=&H92 TO &H269
1010 A=PEEK(I)
1020 IF A<128 THEN PRINT CHR$
     (A); ELSE PRINT CHR$(A-
     128)
1030 NEXT I
```

Quant à désosser vos propres programmes Basic, une bonne adresse de départ : $65F4. Vous y retrouverez les composantes traditionnelles du stockage : un zéro pour commencer, un *link* (ou adresse du début de la prochaine ligne) sur deux octets, deux octets encore pour le numéro de la ligne, puis le texte de la ligne, dans lequel les mots clés de basic sont représentés par des codes commençant à 128 ($80) pour END, et se suivant dans l'ordre que vous aurez pu voir avec le petit programme précédent.

Vous avez donc votre permis de chasse en mémoire, profitez de ce "safari-mémoire".

François J. BAYARD

MASTER PAINT
POUR DESSINER SUR ORIC

LOGICIEL de création graphique pour Oric-1 et Atmos, Master Paint s'utilise directement au clavier ou avec une manette de jeu. Pour faciliter le dessin depuis le clavier, les commandes ne comportent qu'une seule lettre. Et le curseur va, à gauche, à droite, en biais, en cercle, etc. On peut même tout effacer, ou enregistrer son œuvre sur cassette afin de la réutiliser à l'intérieur d'un programme.

Présenté sur cassette à chargement rapide, *Master Paint* dévoile immédiatement une partie de ses possibilités par une petite démonstration. Un appui sur la barre d'espace et l'écran devient tout noir, un seul petit point — le curseur — apparaissant en haut à gauche. Ce noir n'est pas rassurant du tout, et il ne reste plus qu'à se plonger dans le manuel d'emploi. On y apprend une chose essentielle : l'appui sur H (comme Help) donne une "Table Help", c'est-à-dire en fait des explications.

On a alors devant soi un très beau bandeau graphique qui indique les différentes commandes et leur mnémonique : C R T B L P E Q W Y Z O. Si les sept premiers mnémoniques sont faciles à retenir (Cercle, Rectangle, Traits, Bloc à transporter, Loupe, Plein, Enregistrement sur cassette), les suivants nécessitent plus de mémoire : Q pour la lecture d'un dessin, W pour l'écriture (W comme Write), Y pour tracer un triangle, Z pour redéfinir des caractères, et O pour choisir la couleur.

Une autre commande, X, doit être utilisée avec une grande prudence : elle efface l'écran sans demander, malheureusement, de confirmation (un appui sur cette touche par inadvertance, et l'œuvre en cours est perdue !).

Avec toutes ces possibilités, on arrive très vite à remplir l'écran de formes géométriques et colorées. On évolue facilement et on dessine n'importe quoi. Le résultat est parfois spectaculaire. Tout semble facile. Mais à la première erreur, au premier choix mauvais, à la première touche confondue, on risque d'être pris au piège, sans pouvoir en sortir. Alors, on se replonge attentivement dans le manuel d'emploi, et on essaye de trouver le moyen de sortir de la routine dans laquelle on est entré par erreur. Or, à ce niveau, aucune annulation de fonction n'est prévue, l'utilisateur est condamné à ne jamais se tromper. Le plus souvent, il ne lui reste qu'à tout effacer (par la commande X) et à recommencer depuis le début.

Le dessin — sans erreur — est très simple. On se déplace dans huit directions, on tire des traits et on règle la vitesse de déplacement.

L'utilisation des commandes T, C, R et Y est précédée du choix de l'origine de la figure (déplacement et validation), ainsi que de celui du ou des autres points. Elle est suivie par la possibilité de tracer la forme, de remplir ou même de gommer. Pour sortir de la routine, on appuie sur F, comme Fin.

Pour connaître la position du curseur, on tape A : les coordonnées du point de l'écran s'affichent. Seule la redéfinition des motifs de remplissage (par Z) autorise l'utilisateur à changer d'avis. Et les motifs sont de toutes formes. On peut même redéfinir la police de caractères.

Pour fignoler, prenez la loupe

Les blocs d'image sont transportables de plusieurs manières : par reproduction (N), superposition (S) ou inversion (I). Les effets sont garantis. Les dessins peuvent aussi être globalement translatés dans les quatres directions, grâce aux quatre flèches.

Une fonction des plus importantes est celle qui permet d'entrer dans les détails d'un dessin et d'y apporter des rectifications fines : la loupe (L). Elle agrandit 864 points sur les 48000 de l'écran.

Il est possible enfin de mémoriser (M) le dessin, ou de l'enregistrer (E) et de le

Le logiciel en quelques lignes

Nom : *Master Paint*
Ordinateurs : *Oric-1 et Atmos*
Forme : *cassette*
Édité et distribué par : *Ère Informatique*
Prix public : *250 FF*
Principale orientation : *conception graphique*

relire sur cassette (Q). Cette dernière opération concerne la totalité de la page haute résolution et demande environ trois minutes. Il est regrettable de ne pas pouvoir traiter seulement une partie de l'écran, par exemple pour l'animer plus tard. Mais ce logiciel n'est pas orienté vers le dessin animé. Il permet déjà de préparer un dessin sur une page entière et on peut imaginer de l'utiliser pour confectionner le décor d'un jeu. Il ne faut pas oublier alors de noter les coordonnées des principaux points.

Ce logiciel complet dans ses propositions graphiques est parfois complexe : selon la routine dans laquelle on se trouve, une même lettre signifie différentes choses (par exemple, dans la routine commandée par B, N signifie reproduction ; dans la routine de création de cercle, commandée par C, ce même N déclenche un retour au mode d'évolution normal). C'est pourquoi, le logiciel ne se suffit pas à lui-même, la documentation est indispensable.

Mais s'il est facile de tracer n'importe quoi, il faut savoir que *Master Paint* ne fera de vous un artiste... que si vous l'êtes déjà !

Max HAGENBURGER

LES COUPS D'ŒIL DE LIST

KUMA FORTH POUR LES ORDINATEURS MSX

LES MSX sont en grande partie définis par leur Basic super-étendu, mais on commence à voir apparaître d'autres langages pour cette famille d'ordinateurs. Nous avons essayé le Forth de Kuma sur un Yashica 64. La cassette devrait fonctionner sans problème sur toutes les machines MSX.

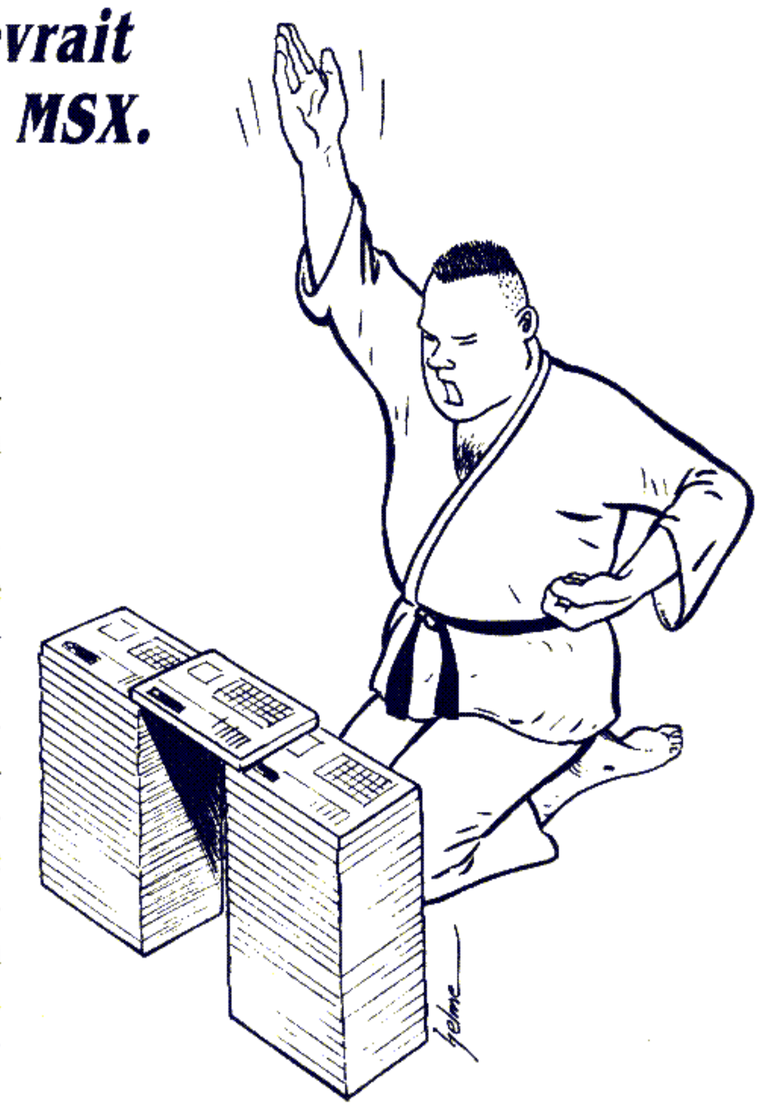
Avant de charger la cassette, on doit lui réserver de la place car le Forth va cohabiter avec le Basic, et non prendre sa place en mémoire centrale. Cette opération se fait en un clin d'œil : CLEAR 200, &H87FF, et nous pouvons frapper BLOAD ''KFORTH'', R. Au total, il faut compter deux minutes avant de voir un O.K. (message traditionnel du Forth) nous saluer à l'écran.

Quelques manipulations très simples nous font alors découvrir un noyau de base très classique et conforme au standard ''FIG FORTH''.

Tous les mots de manipulation de la pile en simple et double longueur sont présents, mais on regrettera que le mot (n)PICK permettant la recopie du nième élément de la pile sur son sommet, absent du vocabulaire Forth, ne soit disponible qu'après l'appel à l'extension virgule flottante.

Pas de modes FAST ni SLOW, contrairement à de nombreux Forth (le mode FAST, en simplifiant les vérifications effectuées par l'interpréteur, accroît la vitesse d'exécution des programmes une fois leur mise au point terminée). Il suffit d'une mauvaise utilisation de la pile des retours (ou d'une autre fausse manœuvre) pour que l'ordinateur se plante. L'absence de bouton RESET sur notre système oblige à recharger le langage : tout est à reprendre à zéro.

Si les boucles et les structures répéti-



Démonstration de Forth par Maître Kuma

tives sont bien présentes (DO...LOOP ; DO... +LOOP ; BEGIN AGAIN ; BEGIN... WHILE... REPEAT), on ne trouve pas contrôle de cas du type CASE... OF... ENDOF... ENDCASE, ce qui devient pourtant très courant et évite l'emploi de trop nombreux IF... THEN imbriqués.

La manipulation des caractères est, elle aussi, très classique et conforme au stantard (EXPECT, KEY, EMIT, etc.), avec toutefois quelques améliorations (MEXPECT, MEXIT) qui contrôlent l'appui sur CTRL-STOP. Nous retrouvons également tous les mots de définitions (CREATE, VARIABLE, < BUILDS... DOES >, etc.), ainsi que ceux dont l'action s'exerce sur la mémoire et le dictionnaire : C, ALLOT, HERE, ERASE, FILL, CMOVE. En revanche, pas de déplacement mémoire commençant par la fin.

Une notice hélas en anglais

Beaucoup plus originale est l'extension « virgule flottante » que l'on trouve dans le programme de base et qui permet de manipuler des nombres entre 1,469380 E-39 et 1,7014118 E 38. L'accès à cette extension est réalisé par le nom du vocabulaire FLOATING. La plupart des primitives de manipulation des nombres entiers se retrouvent en virgule flottante précédées de la lettre F. C'est le cas de F., F*, F–, FDUP, FROT, F<, etc., et de F(n)PICK dont nous avons déjà parlé.

Jusqu'à présent, un programmeur ayant déjà tâté du Forth peut très bien s'en sortir sans la notice, mais dès que l'on aborde la gestion de la mémoire de masse, cette notice devient absolument nécessaire, et c'est là que les difficultés commencent pour quiconque ne connaît pas bien la langue de Shakespeare. La notice fournie (70 pages) est en effet rédigée en anglais.

La première opération à effectuer consiste à formater une cassette vierge grâce au mot FORMAT. Le magnétophone étant en position enregistrement, la bande se trouve segmentée en blocs d'un Ko comportant chacun une en-tête numérotée. Ainsi, quand le magnétophone sera en position *lecture*, l'ordinateur pourra retrouver le bloc sur lequel il doit travailler. Il lui faut vingt secondes pour lire un bloc.

Au début, il faut s'habituer à ces manœuvres, mais cette façon d'organiser la cassette s'apparente de près à ce qui se passe avec une disquette. Toutefois, le temps de recherche est long et on doit faire très attention avec les boutons du magnétophone afin de ne pas effacer accidentellement un bloc.

La modification ou l'écriture d'un bloc s'obtient en tapant *n* EDIT, ce qui donne accès à un éditeur pleine page vraiment très pratique. Chaque bloc est organisé en 16 lignes de 64 caractères ($16 \times 64 = 1024$, soit 1 Ko). Le bloc apparaît à l'écran avec ses 16 lignes numérotées. Malheureusement, une ligne d'enregistrement ne correspond pas à une ligne d'écran, mais environ à une ligne et demie : lors de l'écriture, rien ne signale à l'utilisateur qu'il est en train de dépasser les 64 caractères ; c'est lors du listage — et donc trop tard — qu'il s'apercevra de la disparition pure et simple des caractères en trop...

Six blocs peuvent résider simultanément en mémoire centrale. Pour graver sur la cassette ceux que l'on vient de créer ou de modifier, on tape FLUSH et l'on met le magnétophone en position *lecture*. L'ordinateur indique les blocs rencontrés et s'arrête au bon numéro. On place alors le magnétophone en position *enregistrement* : une pression sur RETURN et l'écriture s'effectue.

Après le noyau Forth proprement dit, on trouve aussi sur la cassette, huit blocs qui constituent une manière de notice. On peut les consulter par *n* LIST ou les compiler si nécessaire par *n* LOAD. Le premier affiche des renseignements sur l'auteur et la conformité du logiciel au standard FIG. Le deuxième bloc contient la définition d'un mot très utile, *point*-S (.S) qui permet d'afficher le contenu de la pile sans la modifier. Pour essayer cette extension et la faire entrer dans le dictionnaire, on tape 2 LOAD. Le bloc n° 3 est entièrement vierge. Les deux suivants contiennent la liste des messages d'erreur. Les trois derniers sont consacrés à la définition de mots permettant de travailler sur des chaînes de caractères. Ces mots, qui ont un petit air de Basic, sont $VARIABLE, $=LEFT$, RIGHT$, MID$. Pour les rendre opérationnels, il suffit de taper 6 LOAD.

Il aurait été très utile que le concepteur de ce programme définisse un certain nombre de mots non standard pour la gestion du générateur sonore et surtout de la haute résolution graphique. Malheureusement, rien n'a été prévu, et c'est dommage, car avec le Forth et sa rapidité, nous aurions pu réaliser des jeux très rapides sans avoir recours au langage-machine.

En conclusion, les possesseurs d'un MSX disposent avec ce logiciel d'un bon outil d'initiation au langage Forth, conforme au stantard FIG, avec quelques lacunes certes, mais aussi une bonne dose d'originalité et des extensions du langage très intéressantes.

Une version en cartouche, une notice en français, quelques mots supplémentaires pour la gestion du son et du graphisme, et l'outil serait parfait. Jamais content ! Allez, bye... c'est le mot (de la fin) qui permet de quitter le Forth pour retrouver le Basic.

Michel BROCHAND

Petit rappel sur le langage Forth

Le Forth est un langage à la fois compilé et interprété. Il est très rapide et occupe peu de place en mémoire.

Les données sont manipulées grâce à deux piles avec lesquelles on travaille en notation pólonaise post-fixée. Les variables sont possibles mais à éviter...

La notion même de programme n'existe pas en Forth : au moyen de quelques mots de base (appelés *primitives* et qui sont en quelque sorte l'équivalent des instructions), l'utilisateur crée d'autres mots qui peuvent à leur tour servir à en définir d'autres, et ainsi de suite. Un programme complet tient en fin de compte en un seul mot !

Le langage est extensible indéfiniment, chacun l'adaptant à ses besoins et au type de problèmes qu'il doit résoudre en programmant.

La gestion de la mémoire de masse utilise le principe de la mémoire virtuelle. Elle est segmentée en blocs d'un Ko pouvant être appelés séparément. Avec les systèmes les plus courants, entre 1 et 10 blocs peuvent résider simultanément en mémoire.

Enfin, un grand nombre d'extensions du langage de base sont maintenant disponibles, mais pour rester fidèle au principe du minimum de mémoire utilisé, on ne les introduit que lorsqu'elles sont nécessaires (double précision, virgule flottante, manipulation de chaînes, éditeur pleine page, assembleur, décompilateur, etc.).

Le logiciel en quelques lignes

Nom : *Forth*
Ordinateur : *MSX*
Forme : *cassette*
Edité par : *Kuma*
Distribué par : *Innelec*
Prix public : *485 FF*
Principale orientation : *langage Forth*
La notice d'accompagnement (70 pages) est rédigée en anglais

SUPERBASE

GÈRE DES FICHIERS SUR LE DAI

TÔT ou tard, les ordinateurs familiaux sont amenés à servir à des applications « sérieuses ». Pour ce faire, il faut des logiciels adéquats. Prenez le Dai. Il est surtout connu pour ses capacités graphiques. Mais, si vous lui adjoignez le logiciel Superbase, il devient un gestionnaire capable de manipuler un fichier d'environ 36000 caractères, avec des options de tri, de sortie formatée et de « mailmerge », c'est-à-dire d'insertion de données dans un texte préparé à l'avance.

Le logiciel *Superbase* est distribué par le DAInamic Club, sous forme d'un « package », sur cassette audio ou micro-cassette digitale. Les programmes contenus dans ce package ne sont pas protégés, si bien qu'il est facile de les transférer sur disquette. Ce qui est quasiment nécessaire, pour une application sérieuse, en raison de la relative lenteur de chargement des autres media.

Saluons au passage cette attitude courageuse, qui va dans le sens de l'intérêt de l'utilisateur. Le package comprend le logiciel *Superbase* proprement dit. Il se charge sous UTILITY, par le classique UT Z3 G400 (ou CALLM #400, depuis Basic). Les connaisseurs auront remarqué que le début en #400 signifie que le programme est « indestructible », même par un RESET général. Encore un très bon point.

Les utilitaires livrés avec *Superbase* permettent de convertir un ancien fichier d'adresses, et de le mettre au format *Superbase*. Ainsi, il n'y aura pas à réécrire de longues listes de noms et d'adresses. *Append*, un autre utilitaire, permet la fusion de plusieurs fichiers *Superbase*. Utile aussi, quand il s'agit d'aller « à la pêche », dans d'anciens logiciels, pour créer un fichier rénové. Enfin, *SB.Extentions* est un utilitaire qui ajoute quelques commandes au Basic du Dai. Ces commandes permettent de manipuler un fichier *Superbase* directement depuis un programme Basic, par exemple, pour y insérer des données provenant de ce fichier. Quiconque a déjà utilisé un gestionnaire de fichiers voit que les options habituelles sont toutes là, dans *Superbase*.

Au menu, les indications nécessaires

Après le lancement, le programme affiche son menu principal (photo 1, page suivante). Les commandes sont repérées par l'initiale de l'action (L pour Load, S pour Save, etc.). Facile donc de s'y retrouver, d'autant que le menu donne toutes les indications nécessaires. La première chose à faire, dans un gestionnaire de données, est la création du masque de saisie (commande F, pour File definition). Chaque fiche peut être constituée d'au maximum 20 lignes de 40 caractères. C'est plus que suffisant, en usage courant.

Classiquement, la création du masque demande le nombre de lignes désiré, le nom de chaque rubrique et la longueur des entrées. Du choix judicieux de ce formatage dépend le nombre de fiches possible, sachant que le tampon global du fichier contient la place pour 36000 caractères. Inutile donc d'attribuer 40 caractères pour la rubrique CODE POSTAL ! En cas d'erreur de jugement, il sera toujours possible de modifier le masque ultérieurement (commande T, pour Transform the file), voire d'ajouter des rubriques.

La commande E (Enter data) permet la saisie des données, à partir d'un quel-

conque numéro de fiche (à condition qu'il soit existant !). Pour chaque rubrique, un bandeau vert pâle indique clairement l'espace autorisé pour la saisie. En cas de tentative de dépassement, le superflu est ignoré, simplement. Par contre, l'écriture d'une ligne s'effectue sous éditeur (déplacement du curseur à gauche et à droite, insertion, effacement). Très pratique. Les touches flèche en haut et flèche en bas permettent de se promener dans les différentes rubriques, pour écrire, modifier, ajouter à volonté. De même, le passage à la fiche suivante, ou précédente, s'obtient en pressant SHIFT flèche à droite ou SHIFT flèche à gauche. Ainsi, la commande E fait-elle office de saisie et de modification, tout à la fois. Cependant, il est encore possible d'éditer une partie du fichier (commande A). Quand la saisie est achevée, il est de bon ton de trier alphabétiquement le fichier (commande Q), puis de passer dans le menu secondaire, qui regroupe les commandes d'impression.

Treize manières d'imprimer

Treize commandes d'impression sont disponibles, et il serait fastidieux de les énumérer toutes (photo 2). Pour résumer, les sorties habituelles sont là : tout le fichier, entre deux bornes numériques ou deux bornes alphabétiques, tout ou partie des rubriques pour chaque fiche. J'ai trouvé un « truc », au sujet duquel la notice reste muette : habitué à d'autres bases de données, j'ai naturellement utilisé l'astérisque(*) pour abréger les clés de recherche. Par exemple, la consigne R*, puis Z, ordonne de lister toutes les fiches, de R à Z. Ça marche !

Là où *Superbase* devient franchement intéressant, c'est la possibilité de l'employer en « mailmerge », c'est-à-dire fabriquer les fameuses lettres personnalisées qu'on retrouve parfois dans nos boîtes aux lettres. Pour cela, *Superbase* possède un mini-traitement de texte (éditeur pleine page), pour confectionner la missive. Aux endroits appropriés, l'utilisateur place les numéros de rubrique, correspondant aux données qu'il désire insérer (par exemple, 01 pour le nom, 02 pour le prénom, etc.).

Raffinement supplémentaire, le suffixe dollar ($) provoque l'insertion au mieux, c'est-à-dire en calculant l'espace nécessaire, sans tenir compte de la totalité du champ défini, lors de la création du masque de saisie. Cela fait plus naturel, dans le texte. Il est possible également de fixer la longueur du champ

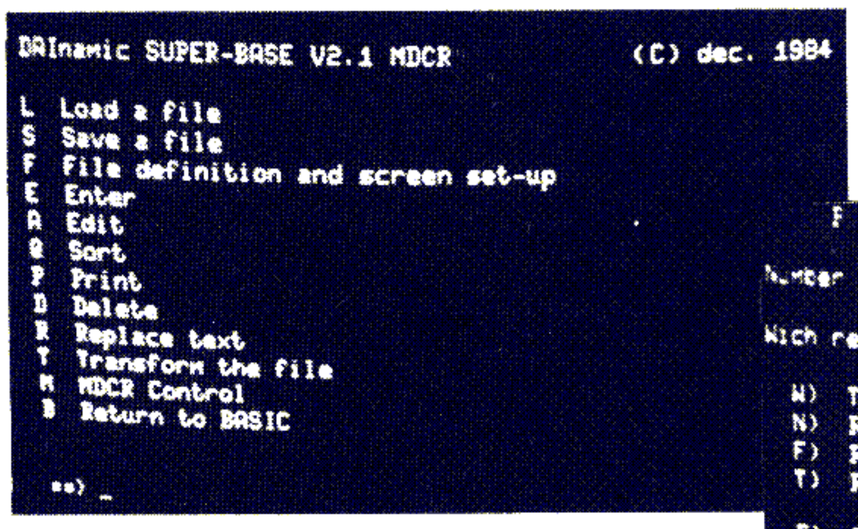


Photo 1 ▲
Menu principal

d'insertion (équivalent de PRINT USING), par la syntaxe 01.....01. Cela signifie : insertion du champ 01 (le nom, en général), avec un cadrage de 10 caractères (le nombre de points définit la consigne de cadrage). Enfin, une série de commandes additionnelles permet d'insérer dans le texte les codes de contrôle habituels des imprimantes Epson (taille des caractères, type d'impression, espacement de ligne, souligné, etc.).

Quand ce travail est prêt, il suffit de revenir au menu d'impression, et de choisir un intervalle, dans le fichier de données, pour provoquer l'impression des textes personnalisés. Comme cette action effectue *de facto* une recherche dans le fichier, le concepteur du logiciel a prévu une astuce, bien utile à l'usage : la commande X permet la sortie du fichier sélectionné, non pas sur l'imprimante, mais directement dans un tampon d'édition. A la fin de l'affichage sur l'écran, ce tampon est automatiquement sauvé sur mémoire de masse. Très pratique, pour extraire des fichiers triés, du fichier principal.

A l'usage, *Superbase* se révèle performant et agréable à utiliser. Le programme est correctement protégé contre les erreurs de manipulation (la création de masque de saisie détruisant le fichier courant, une confirmation est demandée). En cas de situation désespérée : un RESET. Puis CALLM#400 revient au niveau du premier menu, sans perte des données.

Le logiciel en quelques lignes

Nom : *Superbase*
Ordinateurs : *Dai PD et Dai T*
Auteur : *Uwe Wienkop*
Édité par : *DAInamic Club Belgique (Mottaart 20, B-3170 Herselt, Belgique) et DAInamic Club France (9 rue Lavoisier, 59140 Dunkerque)*
Prix public : *240 FF*
Principale orientation : *base de données*
Autres orientations : *mailmerge, sorties formatées, courrier personnalisé*

Photo 2 ▼
Les commandes d'impression du menu secondaire

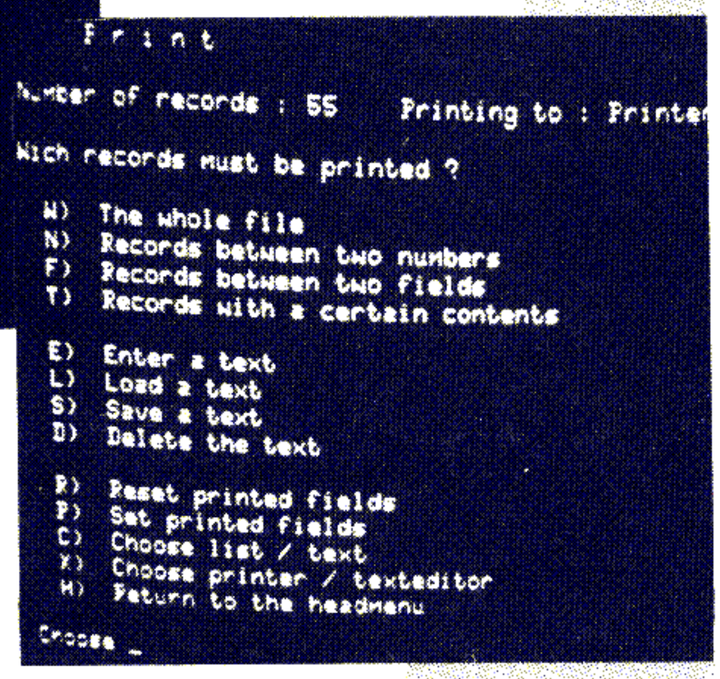


Photo 3 ▼
Une fiche dans le masque de saisie

Le seul point contestable concerne la documentation, fournie avec le logiciel. La notice, en anglais, est sommaire et contient des erreurs. J'ai vérifié : le texte en flamand (!) ne vaut guère mieux. Par exemple, les adresses permettant de modifier les couleurs de l'affichage à l'écran sont erronées. Un petit coup de désassembleur m'a permis de m'y retrouver assez facilement, mais, pour faire cela, il faut bien connaître la machine. Voilà donc les bonnes adresses : la couleur de fond de texte se trouve en #2843, celle des lettres en #2844. Le fond et les lettres des entrées clavier sont colorés respectivement par les codes situés en #2845 et #2846.

Dernier détail agaçant, le programme convertisseur des fichiers d'anciennes bases de données (tableau Basic alphanumérique) est à configurer. Là non plus, la notice ne donne pas de renseignements. Heureusement, *Converter* est écrit en Basic. L'utilisateur devra lister la partie située en 50000, pour modifier les pointeurs à sa guise (nombre de rubriques de l'ancien fichier, comptage à partir de 0, ou à partir de 1, etc.). A signaler, aussi : *Superbase* configure automatiquement le clavier en mode minuscules, au passage dans le menu principal. Cela part d'un bon sentiment, et on peut aimer. Moi pas.

Alain MARIATTE

DÉMONTER LE HASARD

L'ÉQUILIBRE précis des six faces d'un dé, livré à l'imprévisible complexité des mouvements de la main qui le lance, illustre parfaitement la rencontre un peu merveilleuse du « même » et de l'« autre », source de toute la fascination des jeux de hasard. Mais si telle est l'image qui vient à l'esprit quand on parle de générateur aléatoire, on risque d'être déçu : rien d'aussi poétique n'entre dans ce domaine où ce qui semble imprévisible est en fait parfaitement déterminé.

Pour générer des nombres aléatoires sur ordinateur, on parle souvent de « générateur aléatoire ». Ce terme cache en fait des « fonctions génératrices de nombres pseudo-aléatoires ». Les nombres générés sont parfaitement déterminés en fonction d'un ensemble très restreint de données initiales. Ils ne restent imprévisibles que dans la mesure où l'on ignore tout ou partie de ces données.

La qualité essentielle recherchée par les concepteurs des algorithmes correspondants était la régularité de la distribution statistique des résultats. Nous donnerons la préférence à la difficulté de remonter aux données initiales, difficulté qui conditionne l'emploi des résultats comme clefs de chiffrement. Sans être formellement liées, ces deux caractéristiques vont fort heureusement de pair.

Il n'en va pas forcément de même, a priori, de la simplicité des formules et de la rapidité des calculs, deux contraintes tout aussi impératives aux yeux des informaticiens. On leur doit en tous cas d'avoir refermé l'éventail des solutions acceptables à tel point que la grande majorité des générateurs en service n'utiliserait finalement qu'un seul et même algorithme, dit de « congruence linéaire » : $X_n = (a\ X_{n-1} + c)$ MOD m.

Le hasard et l'heure qu'il est

Dans cette formule :

- m est généralement lié à la longueur k du « mot » de l'ordinateur ($m = 2^K$, ou $m = 2^K \pm 1$, par exemple) ;
- a, habituellement compris entre 0.1m et 0.9m, est soumis à certaines restrictions ; a MOD 8 = 3 ou 5, par exemple, si $m = 2^K$;
- c doit seulement être premier avec m ;
- la graine X_0 peut être soit introduite par l'utilisateur si ce dernier veut initialiser une séquence reproductible, soit initialisée par l'ordinateur qui donnera à X_0 une valeur fixe à chaque RUN, soit au contraire, calculée à partir de l'heure de l'horloge interne. Dans ce cas, la valeur est dite aléatoire.

Pour améliorer la protection du secret des clefs principales, on ne recourt à un générateur de ce type que pour créer une

DÉMONTER LE HASARD

ou plusieurs séquences de nombres qui vont constituer les véritables générateurs de clefs particulières par l'intermédiaire d'un autre type d'algorithme, additif cette fois : $X_n = (X_{n-a} + X_{n-b}) \text{ MOD } m$, avec $a < b$. Seuls certains couples (a,b), tels que (24,55) ou (27,98) par exemple, donnent des séquences de période maximale égale à 2^b.

Si l'on opère sur des « entiers », entre −32768 et +32767, on adoptera de préférence l'algorithme voisin : $X_n = X_{n-a}$ XOR X_{n-b} qui fonctionne avec les mêmes couples (a,b).

XOR : une drôle d'exclusivité

Ouvrons une parenthèse au sujet de l'opérateur XOR, qui tire son nom du « ou exclusif » entre deux variables booléennes. Sur des nombres exprimés en binaire, XOR effectue une addition chiffre à chiffre sans retenues.

Pour trouver une documentation sur ces générateurs aléatoires il a fallu recourir à la « bible » de Knuth (The Art of Computer Programming, vol 2).

Le programme de chiffrement présenté ici (écrit sur un PX-8) reprend les principes des programmes déjà publiés dans *LIST* (1), avec transposition encadrée par deux substitutions. Ce programme diffère de celui publié dans le numéro 9 par le calcul beaucoup plus sophistiqué des clefs particulières.

L'algorithme de création des générateurs de clefs se présente sous la forme : $X_n = (a\ X_{n-1} + Z) \text{ MOD } m$ avec $m = 10^{16}$, $a = (10^8 + 1)\ W$, $X = 10^8\ U + V$. Ici, U, V, W et Z sont des entiers de 8 chiffres, avec W MOD 20 = 1, et Z premier avec 10.

La forme particulière du multiplicateur facilite et accélère les calculs en multiprécision.

Pour éviter tout risque de répétition accidentelle, les nombres U et V sont calculés à partir d'une clef principale X, de 16 chiffres, de la date et de l'heure à la seconde près, et de la longueur du message. Seul ce calcul préliminaire est confié au RND de l'ordinateur.

L'algorithme défini là sert à calculer deux séquences, l'une de 71 et l'autre de 73 entiers, stockées dans les tableaux G(i) et H(i). Ces séquences sont constamment renouvelées grâce aux algorithmes : G(M) = G(M−36) XOR G(M−71) et H(N) = H(N−42) XOR H(N−73), dont on trouvera des équivalents cycliques, lignes 270 et 280. Les clefs particulières sont définies par G(M) XOR H(N).

La préparation de ces séquences entraîne deux temps morts d'une vingtaine de secondes, l'un avant et l'autre aussitôt après la saisie des textes (clair ou crypto). Celle du clair s'effectue au moyen d'un LINE INPUT, ce qui permet de taper à loisir, et de corriger le cas échéant, des blocs de 255 caractères au plus. Celle du crypto, par INPUT$(4), n'accepte que des groupes de quatre chiffres hexadécimaux.

Pour des raisons de facilité de démonstration, la version proposée suppose une transmission du crypto sous forme écrite. Ce système sera beaucoup plus commode à utiliser si le crypto reste sous forme binaire, qu'il soit stocké sur cassette ou disquette, ou transmis directement par fil. Le programme doit alors être modifié en fonction de l'environnement. L'économie de 40 % environ en nombre d'octets par rapport au texte clair prend alors tout son intérêt.

L'ensemble du système peut paraître excessivement compliqué. Tenter d'écrire en Basic un équivalent du Data Encryption Standard (DES) d'IBM permettrait sans doute de réviser ce jugement : en gros, le chiffrement de chaque bloc de 64 bits se traduit, avec la DES, par un millier d'opérations élémentaires variées, comportant de subtils « effets d'avalanche », chaque bit dépendant de tous les autres et pouvant à son tour les modifier.

Si vous pensez que le système proposé reste trop peu hermétique, et si vous entrevoyez une méthode d'attaque, autre qu'un essai exhaustif des clefs, n'hésitez pas à nous en faire part.

Pierre BARNOUIN

Au fil du programme

Lignes 100 à 140 : initialisation et choix (chiffrer/déchiffrer)

Lignes 150 à 210 : chiffrement
- calcul du groupe date.heure Y (150)
- création du générateur de clefs n° 1 (160)
- entrée du clair (170)
- codage numérique et substitution n° 1 (180)
- fin de saisie, création du générateur n° 2 (190)
- transposition et substitution n° 2 (200)
- impression du crypto (210)

Lignes 220 à 280 : sous-programmes communs
- création de générateurs de clefs (220 à 240)
- substitution n° 1 (250)
- transposition (260)
- génération des clefs (270 et 280)

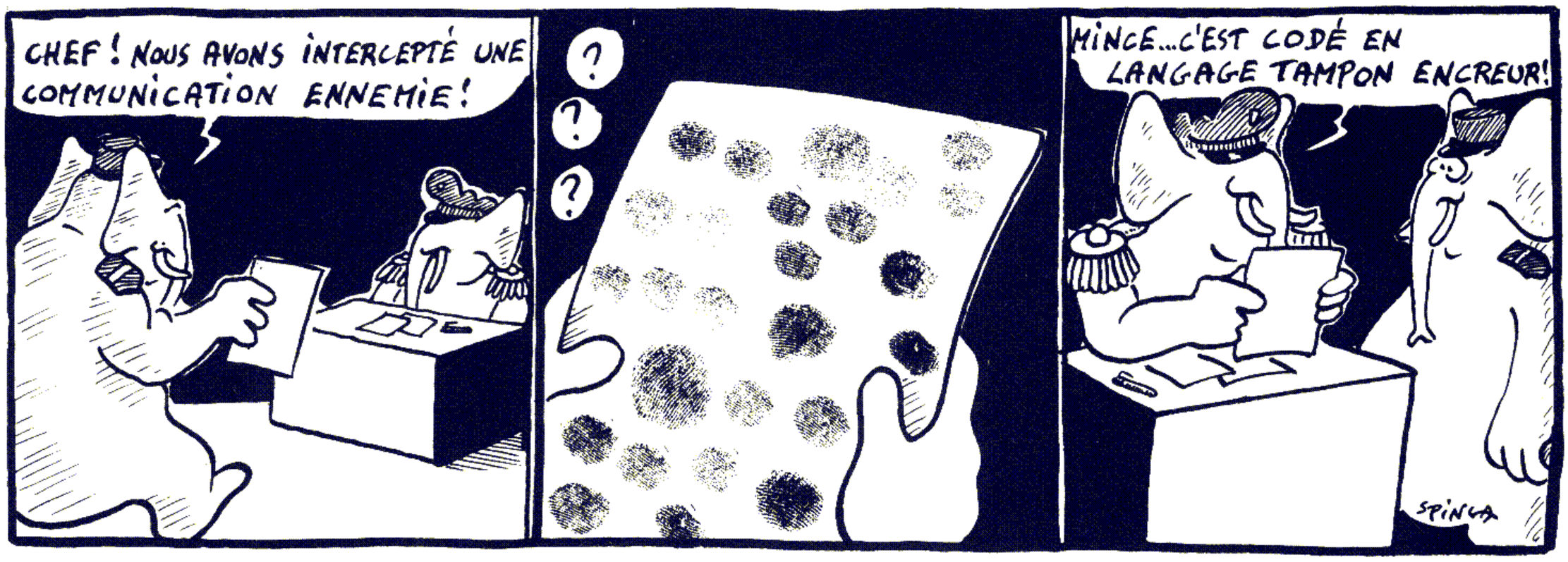
Lignes 290 à 350 : déchiffrement
- initialisation, création du générateur n° 2 (290 et 300)
- saisie contrôlée du crypto (310)
- transposition, substitution n° 2, création du générateur n° 1 (320)
- substitution n° 1 (330)
- décodage alphabétique (340)
- impression du clair (350)

Ligne 360 : ligne DATA

Exemple d'exécution

```
Clefs : W =  45678901, X = 1234567891234567, Z = 87654321
 26082015        32
88BA AFC8 8B96 83F2 D263 42D6 9A2A A296 60A3 A7C7 94B8 81C2 64AD 4C62
AEBA 49D9 261B F70B C6AC C969 F75E 4BE2 5468 4E45 E690 3A80 9C6D 8E3F
9920 5A0D 58AA B930
L'ESSENTIEL EST DE PROTEGER LE SECRET DES CLEFS, SI UNE METHODE DOIT RESTER SECRETE, C'EST QU'ELLE EST MAUVAISE.    26082015        32
```

(1) Dans LIST *5, page 28, avec des programmes pour X-07 ; dans* LIST *9, page 44, avec un programme pour PX-8.*



Cryptographie sophistiquée
Programme pour PX-8
Auteur Pierre Barnouin


```
100 DEFSTR A-D:DEFINT E-M:DEFDBL S-Z:DEF FN A=RIGHT$("000"+HEX$(F),4)
110 S=100000000#:DEF FN T(X)=X-S*INT(X/S):DEF FN K(X)=INT(2*P*X/S)-P
120 CLS:DIM G(72),H(72):PRINT"Préparer l'imprimante":P=32768!
130 D="AAACDEILMNORSTU BFGHJKPQVWXYZ(,.:'/?-)0123456789":INPUT"Clefs";W,X,Z
140 PRINT"0=Chiffrement,  1=Déchiffrement":IF VAL(INPUT$(1)) THEN 290
150 C=TIME$:Y=VAL(MID$(DATE$,4,2)+LEFT$(C,2)+MID$(C,4,2)+MID$(C,7))
160 U=X/Y:GOSUB 220:DIM F(2500):CLS:FOR I=1 TO 4:READ A:PRINT A:NEXT
170 LINE INPUT;"*";C:FOR J=1 TO LEN(C):E=INSTR(D,MID$(C,J,1))
180 IF E=0 THEN BEEP ELSE B=B+HEX$(E-1):IF LEN(B)>3 THEN GOSUB 250
190 NEXT:IF C>"" THEN 170 ELSE B=B+"FFFF":GOSUB 250:U=X*Y/L:GOSUB 220
200 LPRINT Y,L:FOR I=0 TO L-1:GOSUB 260:F=G(M)XOR H(N)XOR F(I)
210 LPRINT FN A+" ";:NEXT:RUN
220 U=INT(S*RND(-U)):V=INT(S*RND):FOR I=0 TO 72
230 U=FN T(W*(U+V)+INT(V*W/S)):V=FN T(W*V+Z)
240 G(I)=FN K(U):H(I)=FN K(V):NEXT:M=0:N=0:GOTO 270
250 F(L)=G(M)XOR H(N)XOR VAL("&H"+LEFT$(B,4)):B=MID$(B,5):L=L+1:GOTO 270
260 GOSUB 270:K=I+ABS(G(M)XOR H(N))MOD(L-I):SWAP F(I),F(K)
270 M=(M+1)MOD 71:G(M)=G(M)XOR G((M+35)MOD 71)
280 N=(N+1)MOD 73:H(N)=H(N)XOR H((N+31)MOD 73):RETURN
290 INPUT"Date.heure, Longueur";Y,L:DIM F(L),J(L):U=X*Y/L:GOSUB 220
300 FOR I=0 TO L-1:F(I)=I:NEXT:PRINT"Taper le crypto":FOR I=0 TO L-1
310 B=INPUT$(4):F=VAL("&H"+B):IF B<>FN A THEN BEEP:GOTO 310
320 PRINT B+" ";:GOSUB 260:J(F(I))=G(M)XOR H(N)XOR F:NEXT:U=X/Y:GOSUB 220
330 FOR I=0 TO L-1:F=G(M)XOR H(N)XOR J(I):GOSUB 270:C=C+FN A:WHILE LEN(C)>1
340 E=2+(C<"1")+(C>"3"):K=VAL("&H"+LEFT$(C,E)):C=MID$(C,E+1)
350 LPRINT MID$(D,K+1,1);:WEND:NEXT:LPRINT Y,L:RUN
360 DATA "Taper le texte par blocs de 255 caractères max.","MAJUSCULES, ESPACE,
CHIFFRES ou SIGNES (.,:'/?-)","Autres caractères pas pris en compte et signalés
par BEEP","Après le dernier bloc, retaper un <RETURN>"
```

LES DISQUETTES COMMODORE

DU CÔTÉ DE LA BAM

SOUS le vocable de BAM se dissimule un élément fondamental pour la gestion des disquettes : la "Block Availability Map", ou carte des blocs disponibles. Vous avez la carte, voici donc une boussole : ne perdez pas le Nord.

Sur les disquettes de Commodore, la carte des blocs disponibles (BAM) est placée sur la piste 18 où elle occupe le secteur numéro 0. Lorsqu'une disquette est mise en place dans le lecteur, c'est à cet endroit que se place immédiatement la tête de lecture pour y trouver une quantité de renseignements dont le système a besoin pour gérer physiquement le contenu de la disquette.

Le formatage consomme des secteurs

Vous savez, bien sûr, que, lors du formatage, les disquettes sont découpées en 35 pistes concentriques, elles-mêmes partagées en un nombre variable de secteurs (de 21 à 17 selon la piste). Au total, il existe 682 secteurs sur une disquette, dont 664 sont libres pour stocker vos propres données (d'où le "664 blocks free" du catalogue d'une disquette fraîchement formatée). La différence entre les deux nombres correspond au nombre de secteurs de la piste 18 que le système réserve pour son propre usage. Ces blocs réservés contiennent d'une part le catalogue de la disquette et d'autre part la table des blocs libres. C'est elle qui nous préoccupe aujourd'hui.

La nécessité impérieuse de la présence d'une telle table est évidente si l'on réfléchit au processus d'une écriture sur disquette. Si la tête d'écriture se positionnait où bon lui semble pour inscrire les données à destination de la disquette, il y a fort à parier que le résultat serait désastreux. La relecture correcte d'un fichier enregistré dans de telles conditions relèverait de la chance la plus inouïe ! Dieu merci, la BAM est là pour diriger l'organisation des opérations, et le système y fait référence à chaque accès en écriture.

A l'écriture d'un programme sur la disquette, c'est la BAM qui indique quels blocs sont restés disponibles. Le système dirige la tête d'écriture vers ces blocs où les données sont alors transférées. A l'issue de ce travail, de nouveaux blocs se retrouvent donc occupés, qui ne l'étaient pas auparavant. Aussi, la tête d'écriture se dirige-t-elle à nouveau vers le secteur BAM pour en modifier le contenu qui doit être remis à jour.

Inversement, lorsque vous décidez de détruire un fichier présent sur la disquette, les blocs occupés se retrouvent libérés : la BAM en est aussitôt informée, en même temps que le catalogue se trouve remis à jour.

Quand l'utilisateur décide

Enfin, la remise à jour de la BAM peut être forcée par une décision émanant de l'utilisateur : la commande VALIDATE (ou COLLECT du Basic 4.0) est destinée à recréer une BAM d'après l'analyse des secteurs successifs de la disquette. En fait, la tête de lecture ne passe pas son temps à lire et écrire sur la BAM : son contenu est mémorisé dans une zone mémoire du lecteur, et c'est cet espace qui est remis à jour, et transféré quand c'est nécessaire sur la disquette. Ouf, merci l'usure ! C'est d'ailleurs ce qui explique la nécessité du CLOSE, car cet ordre provoque le transfert sur le bloc BAM. Un CLOSE oublié, la BAM n'est plus à jour, et les données sont quasi-perdues !

La commande "I:" (Initialise) a enfin pour effet de provoquer une lecture de la BAM, avec enregistrement de son contenu dans la mémoire du lecteur de disquettes.

La figure 1 représente le contenu hexadécimal de la BAM de deux disquettes. En haut, une disquette nouvellement formatée, et en bas une disquette

plutôt bien remplie. La comparaison des deux va nous fournir quelques éléments de base.

Les 4 premiers octets de chaque BAM sont semblables. Les 2 premiers (ils ont été encadrés : 12 et 01) représentent en hexadécimal les numéros de piste et de secteur qui constituent la suite du bloc. Il s'agit donc du catalogue (piste 18, secteur 1).

L'octet suivant (cerclé) a pour valeur 41. Il s'agit d'un signe de reconnaissance grâce auquel le système identifie le format de la disquette. La valeur 41 correspond au format type 1541. Le quatrième octet (cerclé) contient 0 ; son usage est indéterminé. Ensuite, toute une série d'octets précède les codes ASCII du titre de la disquette (16 octets) et de son numéro d'identification (2 octets). C'est ce groupe de 140 octets qui constitue la carte des blocs...

Comme l'indique le schéma, les octets sont groupés par 4, chaque groupe définissant l'occupation des blocs d'une piste. Il y a donc 35 groupes de 4 octets.

Le premier représente la piste 0, le dernier représente la piste 35. Au sein d'un groupe, le premier octet représente le nombre de blocs libres. La valeur 15 (hexa) signifie que la piste en question contient 21 blocs libres. Le second octet est l'image des secteurs 0 à 7. Le troisième représente les secteurs 8 à 15. Le

```
PISTE 18 SECTEUR 0
  0 > 12 01 41 00 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ..A.............
 16 > 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ................
 32 > 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ................
 48 > 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ................
 64 > 15 FF FF 1F 15 FF FF 1F 11 FC FF 07 13 FF FF 07 : ................
 80 > 13 FF FF 07 13 FF FF 07 13 FF FF 07 13 FF FF 07 : ................
 96 > 13 FF FF 07 12 FF FF 03 12 FF FF 03 12 FF FF 03 : ................
112 > 12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01 : ................
128 > 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01 : ................
144 > 54 45 53 54 53 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 : TESTS...........
160 > A0 A0 38 38 A0 32 41 A0 A0 A0 A0 00 00 00 00 00 : ..88.2A.........
176 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
192 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
208 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
224 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
240 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
```

Figure 1

Contenu hexadécimal de la BAM de deux disquettes

```
PISTE 18 SECTEUR 0
  0 > 12 01 41 00 15 FF FF 1F 05 80 12 0A 00 00 00 00 : ..A.............
 16 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
 32 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
 48 > 0B 1F 7C 10 15 FF FF 1F 10 57 7F 1D 00 00 00 00 : ..<......W?.....
 64 > 00 00 00 00 00 00 00 00 0F 6C FF 07 00 00 00 00 : .........,......
 80 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
 96 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
112 > 00 00 00 00 00 00 00 00 00 00 00 00 0A F0 E3 01 : ................
128 > 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01 : ................
144 > 55 54 49 4C 49 54 41 49 52 45 53 20 30 32 A0 A0 : UTILITAIRES 02..
160 > A0 A0 55 32 A0 32 41 A0 A0 A0 A0 00 00 00 00 00 : ..U2.2A.........
176 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
192 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
208 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
224 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
240 > 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : ................
```

Figure 2

Octets 72 a 75: piste 18, secteur 0

00000111 $07	01001000 $48	10010010 $92	00000110 $06	← valeurs binaires / valeurs hexa
7 blocs libres	01234567	89111111 012345	11112222 67890123	← nos des secteurs

0=occupe - 1=libre

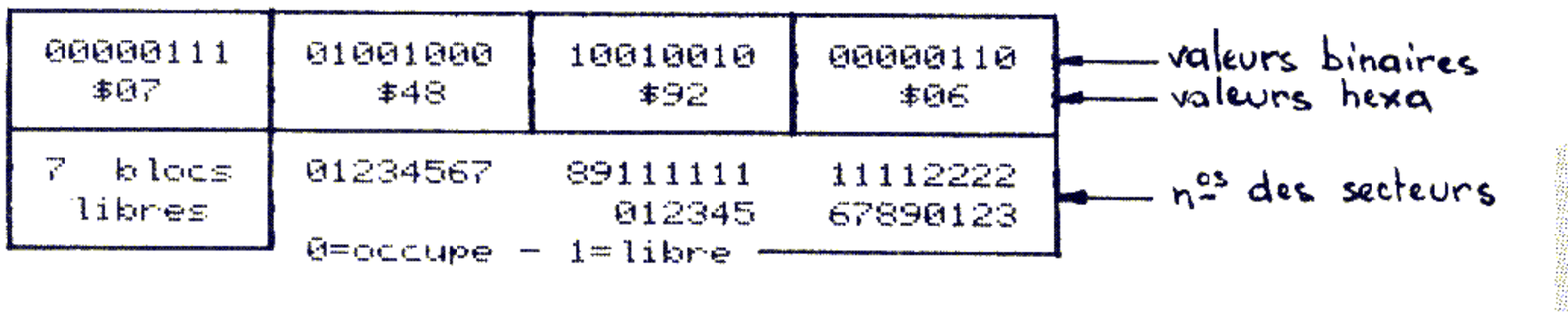
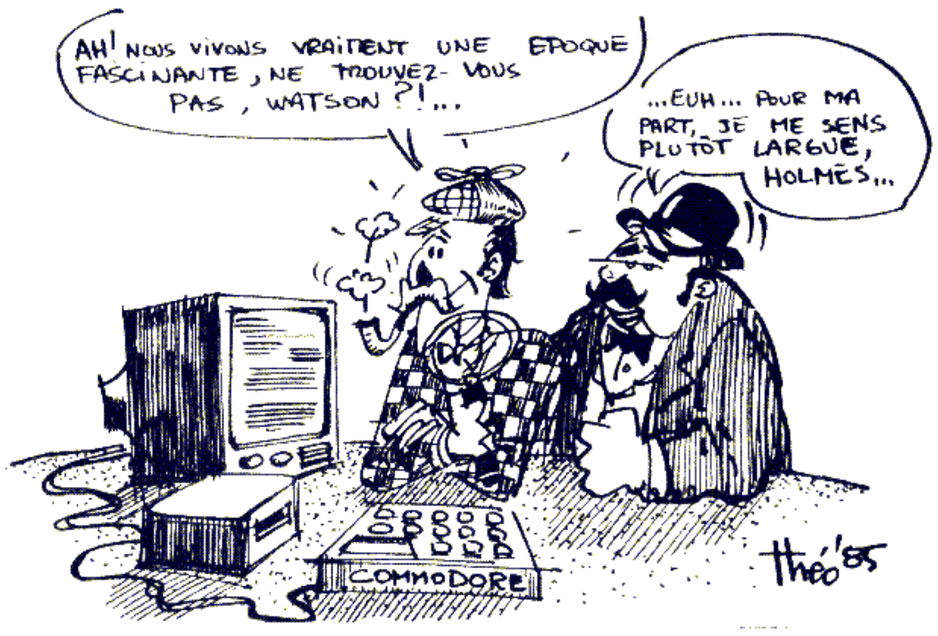
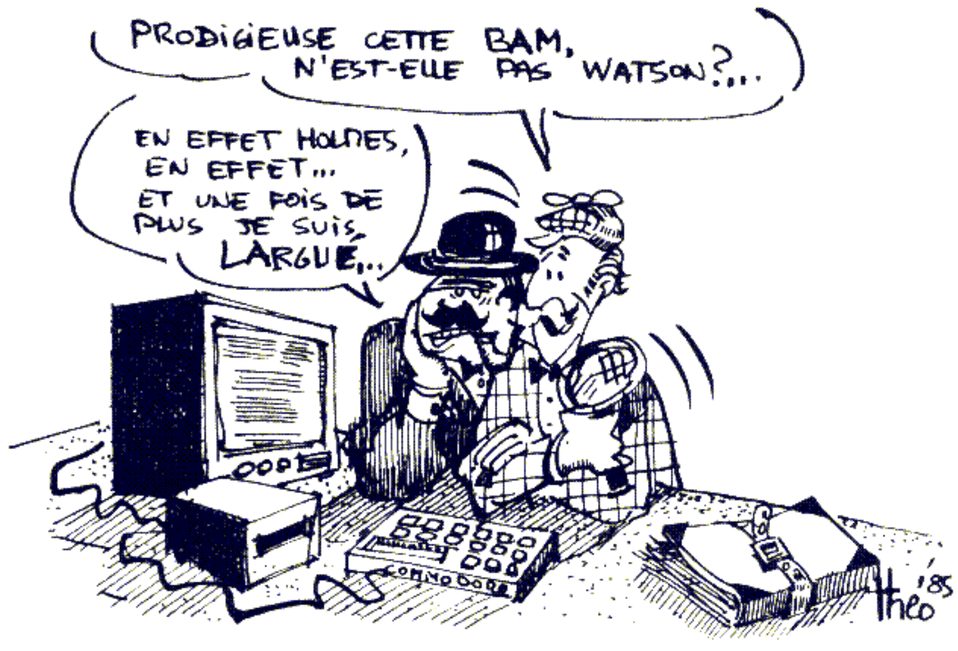


Figure 3





Lecture de la BAM

Utilitaire pour disquettes Commodore

Auteur Jean-Pierre Lalevée


```
100 REM ******************************
110 REM * UTILITAIRE DISQUE NO.6 *
120 REM *     LECTEUR DE BAM     *
130 REM ******************************
140 :
150 PRINT"■","LECTEUR DE BAM"
160 PRINT"■PERMET DE VISUALISER LA BAM SUR L'ECRAN"
170 :
180 C0$=CHR$(.)
190 DIM S(35):REM NB DE SECTEURS PAR PISTE
200 FOR I=1 TO 17:S(I)=20:NEXT
210 FOR I=18 TO 24:S(I)=18:NEXT
220 FOR I=25 TO 30:S(I)=17:NEXT
230 FOR I=31 TO 35:S(I)=16:NEXT
240 DEF FNC(S)=NS(INT(S/8)) AND (21(S-8*INT(S/8)))
250 :
260 REM ******************* INITIALISATIONS DISQUE *************************
270 OPEN15,8,15,"I0":GOSUB 610:REM INITIALISE LA DISQUETTE
280 OPEN 2,8,2,"#":GOSUB 610:REM OUVRE UNE MEMOIRE TAMPON
290 :
300 PRINT#15,"U1:";2;0;18;0:GOSUB 610:REMCLOSELIT LA BAM
310 PRINT#15,"B-P";2;4:GOSUB 610:REM POSITIONNE LE POINTEUR
320 :
330 REM ********************** PRESENTATION ECRAN ***************************
340 PRINT"■■■"
350 FOR I=0 TO 20:IF I<10 THEN PRINT" ";
360 PRINT MID$(STR$(I),2):NEXT:PRINT"  ";
370 FOR I=1 TO 35:PRINT RIGHT$(STR$(I),1);:NEXT
380 :
390 REM ************************ PROG PRINCIPAL ****************************
400 FOR P=1 TO 35:PRINT"■■■":REM TOUTES LES PISTES
410 REM ++++++++++++++++++++ LECTURE DES 4 OCTETS ++++++++++++++++++++++++
420 GET#2,A$,B$,C$,D$
430 BL=BL+ASC(A$+C0$)
440 NS(0)=ASC(B$+C0$)
450 NS(1)=ASC(C$+C0$)
460 NS(2)=ASC(D$+C0$)
470 REM ++++++++++++++++++++++++++ AFFICHAGE ++++++++++++++++++++++++++++++
480 FOR S=0 TO S(P)
490 C$="▒":IF FNC(S) THEN C$=":"
500 PRINT TAB(X+2);C$
510 NEXT S:X=X+1
520 NEXT P
530 :
540 REM ++++++++++++++++++++++++++ TERMINE ++++++++++++++++++++++++++++++
550 PRINT#15,"B-P";2;144
560 N$="":FOR I=1TO20:GET#2,A$:IF A$<>CHR$(160) THEN N$=N$+A$
570 NEXT:PRINT"■"LEFT$(N$,LEN(N$)-2);"-";MID$(N$,LEN(N$)-1);
580 PRINTBL-17;"BLOCS LIBRES."
590 GOTO 660
600 :
610 REM ********************** ERREURS DISQUE ***************************
620 INPUT#15,E1,E$,E3,E4
630 IF E1>20 THEN PRINT"ERREUR DISQUE:":PRINT E1","E$","E3","E4:STOP
640 RETURN
650 :
660 CLOSE 2:CLOSE 15
670 END
```

quatrième concerne les secteurs 16 à 23 (!) de cette même piste.

Pour éclaircir cette situation, **la figure 2** vous indiquera comment procéder. En tout état de cause, l'important est de remarquer qu'un secteur occupé se traduit par un bit à 0, et qu'un secteur libre a pour reflet un bit à 1.

Le programme que nous vous soumettons aujourd'hui vous permettra d'obtenir facilement sur l'écran une image complète de la BAM d'une disquette. **La figure 3** constitue un exemple de ce que vous pourrez obtenir. Si vous envisagez de réaliser l'affichage sur une imprimante, vous devrez le remodeler assez sérieusement, en particulier aux lignes 480-520 car l'affichage se fait ici en colonnes successives.

Peu d'explications sont nécessaires pour ce programme qui est suffisamment commenté par des REMarques. La seule astuce réside dans la ligne 240 qui permet d'effectuer rapidement la recherche de la valeur d'un bit dans l'un des 3 octets lus. A l'affichage, les blocs libres sont visualisés par deux points et les blocs occupés par un carré grisé.

Le programme traite les octets du bloc BAM, puis le titre et l'ID de la disquette. Les octets 165 à 170 (qui sont encore un signe de reconnaissance du DOS) sont ignorés. Tous les octets suivants restent toujours à 0 : ils ne sont pas utilisés. Seules les unités de disquettes « professionnelles » Commodore les utilisent.

Jean-Pierre LALEVÉE

GÉNÉRATEUR DE LOGARITHMES A 50 DÉCIMALES

LES logarithmes avec cinquante décimales des six premiers nombres premiers peuvent être calculés par un « gros » ordinateur. Les propriétés des logarithmes permettent alors de trouver ceux des nombres composés à partir de ces six nombres premiers. On conçoit ainsi une table incomplète, mais très précise.

En partant de calculs effectués par un « gros » ordinateur (Hitachi), et en appliquant les propriétés des logarithmes, on va pouvoir créer une table de logarithmes à 50 décimales. Cette table sera incomplète : elle contiendra les logarithmes de tous les nombres composés à partir des six premiers nombres premiers (2, 3, 5, 7, 11 et 13).

Les données de base du programme seront les valeurs des logarithmes à cinquante décimales de ces six premiers nombres premiers. Les propriétés des logarithmes vont permettre de déduire ceux des nombres composés à partir de ces six nombres premiers. En effet, a et b étant deux nombres positifs, on a : $\log (a \times b) = \log a \times \log b$, et $\log (a^b) = b \times \log a$. Par exemple, on pourra trouver les 50 décimales de log 504 car $504 = 2^3 \times 3^2 \times 7$ et donc : $\log 504 = 3 \log 2 + 2 \log 3 + \log 7$.

Le principe du programme est le suivant : on calcule une fois pour toutes les logarithmes des six plus petits nombres premiers (donc 2, 3, 5, 7, 11, 13) avec 50 décimales. La combinaison par addition de certains de ces logarithmes permet d'obtenir le logarithme des nombres composés à partir de ces six nombres premiers. Si, de plus, on accepte de faire l'effort d'effectuer la multiplication d'un logarithme à 50 décimales par un nombre d'un seul chiffre, on obtient un nombre impressionnant de combinaisons. Pour effectuer de tels calculs, il faut disposer — en plus des données de base — d'une table de multiplication et d'une procédure d'addition, toutes deux en multiprécision. C'est ce que réalise également le programme.

Quelques mètres de papier...

En fait, les choses se passent un peu différemment. Les entrées, c'est-à-dire les nombres dont les logarithmes sont calculés, sont imprimées avec un pas théorique de 0,01 sur une étendue de 1 à 10, bornes exclues.

Or les seuls nombres justiciables du procédé de calcul adopté sont ceux divi-

GÉNÉRATEUR DE LOGARITHMES A 50 DÉCIMALES

Log à 50 décimales
Programme en Basic
Auteur Pierre Ladislas Gedo


```
10:A=2
11:B=3
12:C=5
13:D=7
14:E=11
15:F=13
20:A(27)=.3010299956
21:A(28)=.6398119521*E-
   10
22:A(29)=.3738894724*E-
   20
23:A(30)=.4930267681*E-
   30
24:A(31)=.898814621*E-4
   0
25:A(32)=.85413*E-50
30:A(33)=.4771212547
31:A(34)=.1966243729*E-
   10
32:A(35)=.5027903255*E-
   20
33:A(36)=.1153092001*E-
   30
34:A(37)=.2886419069*E-
   40
35:A(38)=.58648*E-50
40:A(39)=.84509804
41:A(40)=.1425683071*E-
   10
42:A(41)=.2216258592*E-
   20
43:A(42)=.6361934835*E-
   30
44:A(43)=.7239632396*E-
   40
45:A(44)=.54065*E-50
50:A(45)=.0413926851
51:A(46)=.5822504075*E-
   10
52:A(47)=.0199971243*E-
   20
53:A(48)=.0242417067*E-
   30
54:A(49)=.0219046645*E-
   40
55:A(50)=.30946*E-50
60:A(51)=.1139433523
61:A(52)=.068367692*E-1
   0
62:A(53)=.6505157942*E-
   20
63:A(54)=.3284308297*E-
   30
64:A(55)=.2918838706*E-
   40
 65:A(56)=.82718*E-50
100:FOR Z=104 TO 990
110:N=Z
120:FOR Y=1 TO 6
130:A(Y+6)=0
140:M=N/A(Y)
150:IF M= INT M LET N=M:
    A(Y+6)=A(Y+6)+1:
    GOTO 140
160:NEXT Y
170:IF N>1 NEXT Z: END
180:G=G-I
190:O=E50
200:P=1/O* INT ((G*A(32)
    +H*A(38)+J*A(44)+K*A
    (50)+L*A(56))*O+.5)
210:FOR X=31 TO 27 STEP
    -1
220:O=O/E10
230:Q=1/O* INT ((G*A(X)+
    H*A(X+6)+J*A(X+12)+K
    *A(X+18)+L*A(X+24)+P
    )*O)
240:A(X-9)=O*E10*(G*A(X)
    +H*A(X+6)+J*A(X+12)+
    K*A(X+18)+L*A(X+24)+
    P-Q)
250:P=Q
260:NEXT X
300:I=Z/100
310:PRINT USING "##.##";
    " LOG";I;" ="
320:USING
330:FOR W=18 TO 22
340:PRINT A(W)
350:NEXT W
360:PRINT " "
370:NEXT Z
380:END
```

Analyse du programme

• Première partie (lignes 10 à 65) : initialisation.

Dans cette partie du programme sont stockés une fois pour toutes les nombres premiers 2, 3, 5, 7, 11 et 13 (lignes 10 à 15) et les logarithmes de 2, 3, 7, 11 et 13 (pour les deux derniers, plus exactement 1,1 et 1,3). Chacun de ces logarithmes est stocké sur six mémoires. Le logarithme de 2 est stocké (aux lignes 20 à 25) dans les mémoires A(27) à A(32). Ainsi, log 2 = A(27) + A(28) + ... + A(32).

Il en va de même pour les autres nombres de base dont les logarithmes sont stockés aux lignes 30 à 35 pour log 3, 40 à 45 pour log 7, 50 à 55 pour log 1,1 et 60 à 65 pour log 1,3.

Notez que la valeur de log 5 a été omise, car on peut toujours la remplacer par 1—log 2 et économiser ainsi de nombreux octets. On observera également que les logarithmes ont été calculés avec 55 décimales, alors que l'on ne doit en afficher que 50, les cinq dernières constituant une sécurité quant à l'exactitude de l'arrondi du cinquantième et dernier chiffre imprimé.

• Deuxième partie (lignes 100 à 190) : détermination des nombres retenus et de leur structure.

Le pointeur passe en revue tous les nombres Z de 100 à 1000 en ne retenant que ceux qui répondent au critère de divisibilité fixé. Le plus petit d'entre eux est 104 et le plus grand 990 (d'où la boucle de la ligne 100). Ils sont au nombre de 179. La table comportera donc 179 entrées, de 1,04 à 9,90, bornes incluses, variant de 0,01 en 0,01, mais avec de nombreux trous correspondant aux nombres non retenus.

Les exposants de chaque facteur premier de Z sont calculés à la ligne 150 et sont stockés en mémoires A (7) à A (12) (boucle 120-160).

Notez l'instruction G = G—I (correspondant à A(7) = A(7)—A(9) sur les PC de Sharp) de la ligne 180. Elle réalise l'opération que nous avons évoquée plus haut à propos de log 5 : chaque exposant de 5 (en mémoire I), affecté du signe — (moins), est imputé à l'exposant de 2 (en mémoire G). Il s'ensuit que l'exposant de 2 peut être négatif, comme par exemple dans le cas de Z = 125, où il est égal à —3, ce qui se justifie en observant que le logarithme de 5^3 ou celui de 2^{-3} ont même valeur, si l'on s'en tient à leur mantisse.

La variable I est donc abandonnée à la ligne 180 et la mémoire correspondante devient disponible pour la suite du programme (elle sera réaffectée en ligne 300).

• Troisième partie (lignes 200 à 260) : calcul des logarithmes.

Les additions et multiplications en quintuple précision sont exécutées en une seule ligne, la ligne 240. Le résultat est stocké en cinq mémoires (boucle 210-260), de R à V. Les additions s'effectuent comme à la main, de droite à gauche (donc de V à R), pour préserver le mécanisme correct des retenues. La retenue à ajouter à une tranche est en mémoire P, celle qui doit en être soustraite se trouve en mémoire Q (lignes 230 et 250). L'arrondi de la dernière décimale est calculé en ligne 200.

• Quatrième partie (lignes 300 à 380) : affichage.

Les entrées sont imprimées avec deux décimales (ligne 310), les résultats en cinq tranches successives de 10 décimales (boucle 330-350), le dernier chiffre étant arrondi à la valeur la plus proche. Selon l'usage, si une tranche comporte moins de 10 chiffres, elle doit être complétée par des zéros, à gauche du premier chiffre, à concurrence de 10 chiffres.

Liste des variables

A(1) à A(6) : (A à F sur les PC de Sharp) chacun des nombres premiers de 2 à 13

A(7) à A(12) : (G à L sur les PC de Sharp) exposants des facteurs premiers dans la décomposition du nombre à traiter, $Z = 2^{A(7)} \times 3^{A(8)} \times 5^{A(9)} \times 7^{A(10)} \times 11^{A(11)} \times 13^{A(12)}$

A(27) à A(32) : valeur de log 2 stockée sur six mémoires (5 tranches de 10 chiffres et une tranche de 5 chiffres)

A(33) à A(38) : idem pour log 3

A(39) à A(44) : idem pour log 7

A(45) à A(50) : idem pour log 1,1

A(51) à A(56) : idem pour log 1,3

Z : suite pour les entiers consécutifs de 104 à 990

I = Z/100

M à O : variables de calcul

Q : retenue de la tranche en cours de traitement

P : retenue de la tranche précédente

A(18) à A(22) : (R à V sur les PC de Sharp) valeur de log I stockée sur 5 mémoires (5 tranches de 10 chiffres)

W à Y : compteurs de boucles

Début de la table

```
LOG 1.04 =
          170333392.
         9878035484.
         7721842115.
         8075111342.
         9883277339.

LOG 1.05 =
          211892990.
         6993807279.
         3505267123.
         2584759155.
         1137905255.

LOG 1.08 =
          334237554.
         8694970231.
         2561499214.
         3319811367.
         6635549630.

LOG 1.10 =
          413926851.
         5822504075.
          199971243.
          242417067.
          219046645.

LOG 1.12 =
          492180226.
         7018161156.
         7171837490.
         6083005563.
         3192217240.

LOG 1.17 =
          681858617.
         4616164379.
         6560964452.
         5590492299.
         8691676846.

LOG 1.20 =
          791812460.
         4762482772.
         2505692704.
         1013627365.
          862711491.
```

sibles exclusivement par 2, 3, 5, 7, 11 ou 13. Ils sont naturellement entiers. Si on les désigne par Z, c'est I = Z/100 qui sera imprimé en entrée (dans notre exemple, Z = 504 et I = 5,04). On fera donc varier Z de 100 à 1 000 (bornes exclues) et on ne retiendra que les Z répondant au critère de divisibilité fixé, en imprimant en entrée les valeurs Z/100 correspondantes.

Nous reproduisons ci-contre le début de la table.

La durée d'impression de la table complète est extrêmement variable d'une machine à l'autre. Dans le cas le plus défavorable, c'est-à-dire avec un PC-1211, elle est d'un peu moins de trois heures. Mais une ou deux dizaines de minutes devraient suffire avec une machine équipée d'un processeur 8 bits.

En tout état de cause, n'oubliez pas d'alimenter votre imprimante en quantité suffisante de papier. Avec un PC-1211, il en faut environ 4 mètres 50, soit la longueur d'un rouleau complet, mais si vous disposez d'une imprimante de 80 colonnes, vous pouvez modifier l'affichage pour imprimer chaque résultat sur une seule ligne, et consommer moins d'un mètre de papier.

Pierre Ladislas GEDO

D'un Basic à l'autre

Le programme tourne sur les Basic dont les variables numériques vont au moins de 1E-55 à 1E55, avec une précision de 10 chiffres significatifs. La liste imprimée ici est celle d'un PC-1251 de Sharp. Une particularité de cet ordinateur de poche (et de tous ceux de cette gamme) est de réserver la même place en mémoire pour les variables A(1) à A(26) que pour les variables A à Z. Aussi pour transposer ce programme à d'autres Basic, il faudra remplacer A par A(1), B par A(2), C par A(3), etc.

En outre, sur les PC de Sharp, il est inutile de dimensionner un tableau, ici A(i). C'est indispensable avec les autres Basic.

UNE MOULINETTE POUR LES ENTRÉES NUMÉRIQUES

QUAND l'utilisateur fait des fautes de frappe, il faut bien que le programme se charge de les corriger. C'est le rôle d'une courte routine, présentée ici en Pascal, que chacun pourra traduire dans son langage favori.

Rares sont les personnes qui écrivent différemment des caractères tels que la lettre O majuscule et le chiffre 0. Dans une moindre mesure, c'est également vrai pour la lettre I et le chiffre 1 qui font, mais quelquefois seulement, l'objet d'une distinction.

Avec une machine à écrire, la différence est un peu plus sensible. Cependant, il existe toujours certaines machines dépourvues, par exemple, de chiffres 0 et 1, ces derniers devant être remplacés par les lettres O et I (pour des raisons d'esthétique, on préfère parfois le l minuscule au I majuscule).

Avec un ordinateur, la différence est très nette, puisque ces caractères sont différents, à la fois par leur représentation et par le code interne. Hélas, cela n'empêche pas les nouveaux utilisateurs de systèmes informatiques de confondre certains caractères. Cette confusion est gênante, car les programmes ne pardonnent pas de telles erreurs, en particulier dans la saisie des valeurs numériques. Pour que les logiciels soient d'un usage plus aisé, il est utile d'effectuer une conversion de tels caractères après toute entrée d'un nombre sous forme de chaîne. Ainsi, par exemple, la chaîne « I2O.3O » doit être transformée en « 120.30 », ce qui permet d'accepter une donnée qui n'était pas valable au départ.

La procédure qui nous permettra de repêcher ainsi certaines fautes de frappe est déclarée de la façon suivante :

procedure correction (*var* valeur : string) ;

A l'appel de la procédure, la chaîne de caractères VALEUR doit contenir le ou les nombres à rendre conformes. Ce même paramètre retourne, après activation de la procédure, la chaîne corrigée.

Le premier travail effectué par la procédure consiste à transformer les caractères qui font l'objet d'une confusion. Il s'agit des lettres O, I, Z, S qui sont respectivement transformées en chiffre 0, 1, 2 et 5. Le caractère du soulignement (_), quant à lui, est transformé en caractère *moins* (−).

D'autre part, pour se conformer à la pratique française, le point décimal est systématiquement transformé en virgule décimale.

Enfin, la procédure transforme aussi les caractères résultant d'une véritable faute de frappe. En effet, sur les claviers des ordinateurs, les chiffres sont souvent accessibles par l'intermédiaire de touches à double fonction, c'est-à-dire correspondant à deux caractères différents selon que la touche majuscule (SHIFT) est ou non enfoncée. Si cette touche n'est pas correctement enfoncée, l'autre caractère peut être pris en compte. La procédure rétablit donc le

Azerty ou qwerty

Caractères pris en compte	Clavier azerty		Clavier qwerty	
	caractères	codes ASCII	caractères	codes ASCII
0	à	64	)	41
	O	79	O	79
	o	111	o	111
1	&	38	!	33
	I	73	I	73
	i	105	i	105
2	é	123	@	64
	Z	90	Z	90
	z	122	z	122
3	''	34	#	35
4	'	39	$	36
5	(	40	%	37
	S	83	S	83
	s	115	s	115
6	§	93	^	94
7	è	125	&	38
8	!	33	*	42
9	ç	92	(	40
—	−	45	−	45
,	.	46	.	46

Correspondance entre les caractères non numériques et les caractères pris en compte par les deux procédures.

chiffre qui correspond au caractère tapé par erreur.

Il faut toutefois remarquer qu'il existe plusieurs catégories de claviers. La plus répandue dans le monde est dite internationale, ou QWERTY car la première rangée de lettres y est composée des touches Q, W, E, R, T et Y. Ce type de clavier ne comporte le plus souvent que les caractères définis par la norme ASCII (American Standard Code for Information Interchange).

Une seconde catégorie de claviers est couramment rencontrée sur les machines à écrire de l'hexagone, elle est dite nationale ou AZERTY. Ces claviers possèdent les caractères spécifiques à la langue française tels que les lettres accentuées (à, é, è, ù), le tréma et le circonflexe, le ç dit cédille et d'autres symboles encore comme celui du paragraphe (§) et de la livre (£) qui remplacent certains caractères internationaux.

Comme la disposition des touches n'est pas la même selon la catégorie du clavier, on aura recours à deux traitements différents. C'est pour cela que nous avons proposé deux procédures distinctes, l'une associée aux claviers QWERTY, et l'autre aux claviers AZERTY. Bien entendu, il existe plusieurs variantes de claviers AZERTY ou QWERTY. Au besoin, on modifiera le sous-programme en conséquence.

Ces procédures réalisent une boucle sur tous les caractères de la chaîne qui est passée en paramètre. Afin de réduire la longueur du code, le caractère en cours de traitement est mémorisé dans la variable CAR, ce qui permet de ne pas faire référence à un élément de tableau chaque fois qu'une affectation ou un test est réalisé.

Tester l'entrée dans la conversion

Un premier test permet de n'entrer dans la partie de conversion du caractère que si celui-ci est autre qu'un chiffre ou un signe valide.

Il faut noter que certains compilateurs acceptent mal le CASE tel qu'il est écrit ici, car tous les cas possibles n'ont pas été prévus. Pour de tels compilateurs, il est indispensable de placer, juste avant chaque instruction *case* car *of*, le test :

if car *in* autorises *then*

où AUTORISES est un ensemble mémorisant tous les états qui ont été prévus dans le cas. On résout ainsi le problème occasionné par les compilateurs strictement conformes au Pascal défini par Wirth (ces compilateurs, soit dit en passant, se font de plus en plus rares).

Il faut prendre garde, lorsqu'un programme utilise une telle procédure, qu'elle soit bien appelée à chaque entrée d'une valeur numérique, faute de quoi le logiciel manquerait d'homogénéité et pourrait être la cause d'erreurs de saisie non détectées. En contrepartie, si cette procédure est bien utilisée, il devient possible, en quelques instructions, de rendre les logiciels plus souples et plus agréables d'emploi.

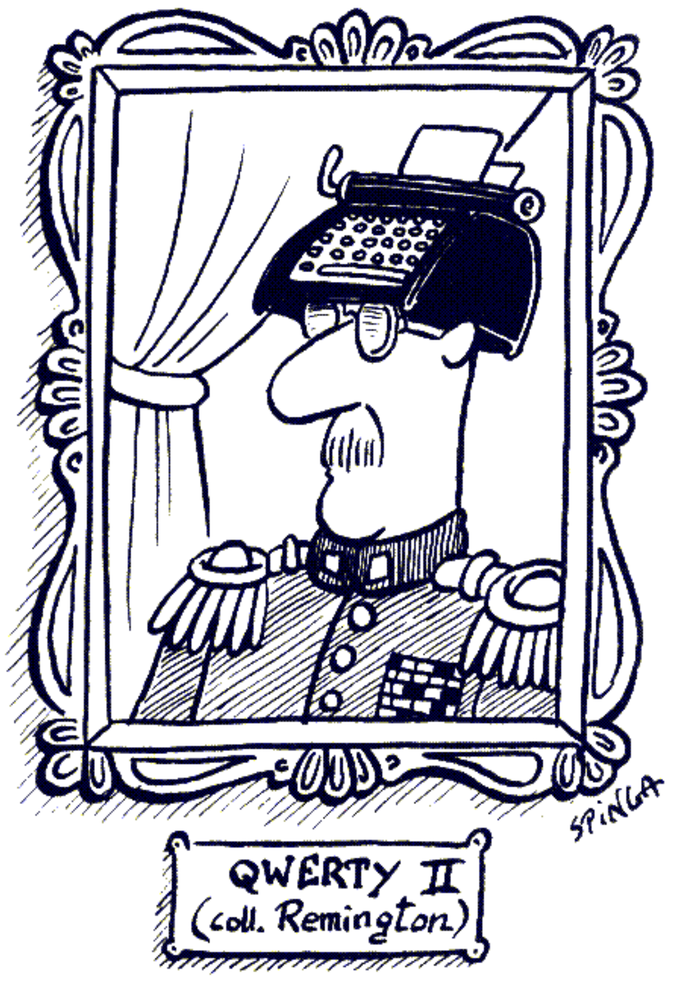
Correction automatique des entrées numériques
Procédure Pascal
Auteur Thierry Chamoret

Version azerty

```
procedure correction (var valeur : string) ;
var i   : integer ;
    car: char ;
begin
  for i : = 1 to length (valeur) do
    begin
      car : = valeur [i] ;
      if not (car in ['0'..'9','+','-'])
        then
          begin
            case car of
              'à', 'O', 'o' : car : = '0' ;
              '&', 'I', 'i'  : car : = '1' ;
              'é', 'Z', 'z'  : car : = '2' ;
              '"'            : car : = '3' ;
              ''''           : car : = '4' ;
              '(', 'S', 's'  : car : = '5' ;
              '§'            : car : = '6' ;
              'è'            : car : = '7' ;
              '!'            : car : = '8' ;
              'ç'            : car : = '9' ;
              '_'            : car : = '-' ;
              '.'            : car : = ',' ;
            end ;
            valeur [i] : = car
          end
    end
end ;
```

Version qwerty

```
procedure correction (var valeur : string) ;
var i   : integer ;
    car: char ;
begin
  for i : = 1 to length (valeur) do
    begin
      car : = valeur [i] ;
      if not (car in ['0'..'9','+','-'])
        then
          begin
            case car of
              ')', 'O', 'o' : car : = '0' ;
              '!', 'I', 'i'  : car : = '1' ;
              '@', 'Z', 'z' : car : = '2' ;
              '#'            : car : = '3' ;
              '$'            : car : = '4' ;
              '%', 'S', 's': car : = '5' ;
              '^'            : car : = '6' ;
              '&'            : car : = '7' ;
              '*'            : car : = '8' ;
              '('            : car : = '9' ;
              '_'            : car : = '-' ;
              '.'            : car : = ',' ;
            end ;
            valeur [i] : = car
          end
    end
end ;
```



Thierry CHAMORET

LE BASIC DU SHARP PC-1350

CINQ ans déjà que le constructeur japonais Sharp a présenté le tout premier ordinateur de poche programmable en Basic, le PC-1211. Quelques mois après, ce fut le Sharp PC-1500 d'une puissance alors inégalée pour un ordinateur de poche. Depuis, régulièrement, Sharp sort un nouveau poquette : PC-1212, 1251, 1261, 1401... et enfin PC-1350 offrant ainsi une large gamme de Basic de poche.

Le 1350 est à peine plus volumineux que son ancêtre 1211 mais il incorpore un « grand » écran à cristaux liquides de 4 lignes de 24 caractères pour une définition graphique de 150 × 32 points. Le clavier est bien un peu étriqué, mais qu'attendre d'un clavier de poche ? Assez complet, il possède un pavé de touches numériques et regroupe ensemble, enfin, toutes les touches d'édition.

L'éditeur, c'est l'interface utilisateur ; la manière d'écrire, de vérifier et, éventuellement, de corriger les lignes d'un programme comme d'un calcul direct. De sa qualité dépendent le confort d'utilisation et la facilité d'apprentissage du fonctionnement de l'ordinateur.

Traditionnel chez Sharp, l'éditeur du PC-1350 est très souple d'emploi bien que souffrant encore de quelques carences. Quand un programme Basic est en mémoire, on déplace l'écran sur ses lignes avec les touches de défilement vertical : ↑ et ↓. Pour modifier le contenu d'une ligne, on l'amène en haut de l'écran et son édition débute sur pression de l'une des touches de déplacement horizontal : → et ←.

S'agissant d'un programme, le curseur clignotant est amené à l'endroit de la correction où l'on pourra insérer (INS), détruire (DEL) ou écrire de nouveaux caractères. Notons enfin que sur un ordinateur de poche, INS et DEL sont d'un accès direct, mais, cependant, pas encore à répétition. Seules les touches de déplacements horizontal et vertical du curseur sont répétées en cas de pression prolongée. Deux fonctions d'édition permettent la mise en place instantanée du curseur en début ou en fin de la ligne en cours d'édition.

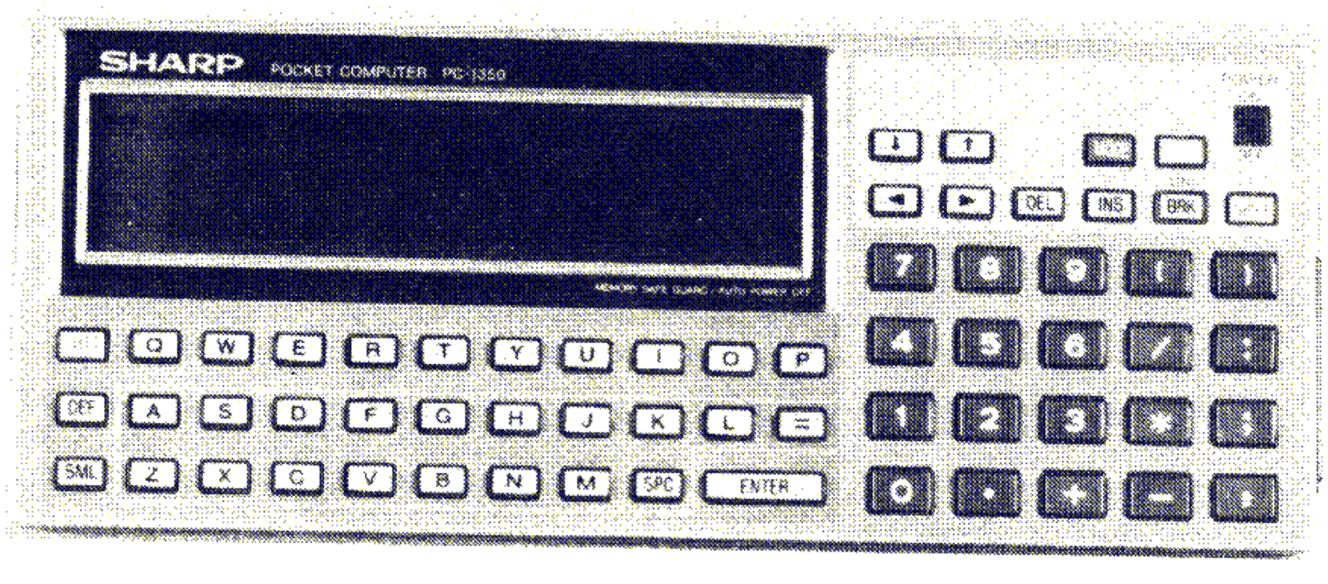
L'espace n'est pas significatif pour l'éditeur du 1350. Si un espace est automatiquement affiché après chaque mot clef du Basic reconnu par l'éditeur, ce n'est que pour en faciliter la relecture. Les lignes Basic sont optimisées en mémoire dès leur introduction : suppression de tout caractère d'espace superflu, compactage sur un seul octet des caractères composant un mot Basic. Ainsi peut-on abréger les mots clefs du Basic : P. pour PRINT, PO. pour POKE, etc., car l'éditeur les reconnaîtra immédiatement et les affichera en toutes lettres (après ENTER).

Ouvert au langage-machine

Chaque ligne peut contenir jusqu'à 82 octets (où chaque fonction, quel que soit le nombre de lettres de son nom, occupe 1 octet ; chaque signe, chiffre ou lettre, compte pour un octet ; le numéro de ligne pour 2 octets ; enfin chaque ligne comporte 2 octets de contrôle : sa longueur et le code d'ENTER). Ceux que la *bidouille* intéresse pourront débuter l'étude de la mémoire interne du PC-1350 avec PEEK à l'adresse 24625 (&6031), celle du premier octet des programmes Basic. On notera avec satisfaction la présence des instructions PEEK, POKE et CALL utiles à la manipulation directe de codes dans la mémoire de l'ordinateur et à l'exécution de programmes en langage-machine. On utilisera pour cela le manuel du langage-machine du PC-1251 disponible auprès du *Club des sharpentiers*.

On ne dispose malheureusement ni d'une fonction AUTO (numérotation automatique des lignes), ni d'un RENUM (renumérotation des lignes et des GOTO/GOSUB...), ni, enfin, d'une fonction DELETE qui permettrait la destruction d'un bloc entier de lignes du programme.

L'éditeur intervient aussi dans les calculs directs effectués avec le PC-1350,



Basic, système, interfaces intégrées, 40 Ko de mémoire morte

comme s'il s'agissait d'un simple calculateur. On écrit le calcul à réaliser, en utilisant n'importe lesquelles des fonctions scientifiques du Basic, et sur pression de la touche ENTER s'affiche le résultat. On pourra rééditer par sa droite ou sa gauche l'expression calculée en pressant une touche de déplacement horizontal du curseur (→ ou ←) ou même récupérer le résultat du dernier calcul effectué sur simple pression de ↑ ou ↓.

Si l'éditeur ne limite pas la longueur des noms des variables, seuls les deux premiers caractères sont significatifs. On pourra, par exemple, écrire : TAUX = 1.186, mais pas simultanément : TARIF = 1460.20, car ces deux variables ne seraient en fait qu'une seule et unique : TA. La précision des calculs demeure bonne bien qu'il n'existe qu'un seul type de variable numérique : 10 chiffres sont conservés mais tous les calculs sont réalisés sur 12 chiffres afin de tenir compte des arrondis. Cela peut sembler insuffisant, cependant, le système est très fiable et rivalise largement avec celui d'autres Basic (comme le Basic Microsoft, par exemple).

On stockera jusqu'à 7 caractères dans les variables alphabétiques A$ à Z$ contre 16 dans les variables dont le nom comprend au moins deux lettres telle AA$. On pourra aussi déclarer, avec DIM, des tableaux de chaînes alphabétiques dont les variables auront une capacité déterminée (jusqu'à 80 caractères) : DIM A$(5)*80 crée six variables A$(0) à A$(5) d'une capacité de 80 caractères.

Le traitement en Basic des chaînes de caractères se réalise grâce au jeu standard d'instructions : CHR$ (conversion d'un code ASCII en caractère), ASC (conversion inverse), STR$ (conversion d'un nombre en chaîne de caractères, VAL (conversion inverse), LEFT$ (extraction de la partie gauche d'une chaîne, MID$ (...de sa partie centrale), RIGHT$ (...de sa partie droite) et LEN (calcul de la longueur d'une chaîne).

Les tableaux de variables auront une ou deux dimensions : par exemple DIM A(10) ou DIM A(2,5). Les 26 variables A à Z ne peuvent être à la fois numériques et alphabétiques (!) : si l'on emploie la variable A, son pendant alphabétique A$ n'est plus disponible et sa sollicitation telle que PRINT A$, provoquerait une erreur. En revanche, il suffit d'affecter à A$ un nouveau contenu pour qu'aussitôt il remplace l'ancienne valeur de A (qui devient à son tour inaccessible, etc.). Enfin, notons qu'un ordre RUN de lancement d'un programme n'efface pas les contenus de ces 26 variables alphanumériques.

Un Basic à étiquettes

Avec les fonctions de référence à une ligne de programme (GOTO, GOSUB, RESTORE, etc.) on atteint presque le grand rêve de modularité des programmes Basic. En effet, non seulement ces instructions peuvent adresser directement des lignes de programme par l'intermédiaire de leurs numéros (GOTO 100), comme indirectement par le calcul (GOTO A*100 + 10), mais aussi de manière totalement paramétrée à l'aide d'étiquettes alphanumériques. Une telle étiquette est un nom, long de 76 caractères au plus, écrit entre guillemets en début d'une ligne. On pourra toujours accéder à cette ligne grâce à son nom : GOTO « PROGRAMME 1 » (d'ailleurs, le GOTO réduit la longueur utile des étiquettes à 75 caractères...).

Les étiquettes longues d'un seul caractère (si ce dernier appartient aux touches des deux rangées inférieures du clavier) pourront aussi servir de point d'entrée direct dans un programme : DEF Z lance directement le programme à sa ligne étiquetée « Z », si elle existe. Dans le mode RESERVE de fonctionnement, on pourra aussi programmer ces mêmes touches, leur assigner des fonctions spéciales jusqu'à concurrence de 144 octets. Ainsi, la pression ultérieure de SHIFT 'touche' correspondra à l'exécution des fonctions qui y sont assignées. On mémorisera donc les séquences de calcul ou ordres Basic fréquemment employés afin d'y accéder plus rapidement.

Dans le domaine des boucles FOR...NEXT, le PC-1350 est plus souple que ses frères ordinateurs car il admet des conditions de variation moins strictes : FOR I = 1 TO 1000000 STEP 0.1 est parfaitement correct. La souplesse du trio READ, DATA et RESTORE, ainsi que de l'INPUT, est remarquable car admettant non seulement des données mais aussi des expressions à calculer. En règle générale, partout où l'on peut préciser une simple valeur, on pourra aussi écrire une formule sophistiquée de calcul de cette valeur. Le Basic du 1350 offre donc d'intéressantes possibilités de paramétrage.

Quatre lignes de 24 caractères, pour un ordinateur de poche, c'est bien. Les fonctions CURSOR et PRINT USING aideront à afficher des messages à l'endroit et dans le format souhaité. L'écran est aussi graphique car chaque point peut être noirci (PSET) ou effacé (PRESET) tandis que POINT en teste l'état. LINE trace une ligne droite — aussi bien que possible — entre deux points. L'instruction d'affichage PRINT peut être temporisée d'un WAIT n, de zéro (WAIT 0) à l'infini (WAIT).

Classique pour un ordinateur de poche dont la cousine germaine est bien la calculatrice, une belle panoplie d'instructions mathématiques : logarithmes (décimaux et népériens) et inverses, fonctions trigonométriques en trois modes (DEG, RAD et GRAD, bien qu'aucun indicateur ne vienne rappeler à l'écran le mode en fonction) et inver-

Mots clés du Basic du Sharp PC-1350

ABS
ACS
AND
AREAD
ASC
ASN
ATN
BASIC
BEEP
CALL
CHAIN
CHR$
CLEAR
CLOAD
CLOSE
CLS
CONSOLE
CONT
COS
CSAVE
CURSOR
DATA
DEG
DEGREE
DIM
DMS
END
EXP
FOR
GCURSOR
GOSUB
GOTO
GPRINT
GRAD
IF
INKEY$
INPUT
INT
LEFT$
LEN
LET
LINE
LIST
LLIST
LN
LOAD
LOG
LPRINT
MEM
MERGE
MID$
NEXT
NEW
NOT
ON
OPEN
OPEN$
OR
PASS
PAUSE
PEEK
PI
POINT
POKE
PRESET
PRINT
PSET
RADIAN
RANDOM
READ
REM
RESTORE
RETURN
RIGHT$
RND
RUN
SAVE
SGN
SIN
SQR
STEP
STOP
STR$
TAN
TEXT
THEN
TO
TROFF
TRON
USING
VAL
WAIT

ses. Pas de statistique, en revanche, ni de fonctions hyperboliques ; il faudra les programmer en Basic. L'algèbre de Boole est représentée par les fonctions AND, OR et NOT qu'on utilisera aussi bien pour programmer des imbrications de conditions dans des tests et des calculs en binaire.

La mémoire de masse

Via l'interface CE-126P, qui est aussi une imprimante thermique, on connectera un magnétophone à cassettes afin d'enregistrer les programmes, les instructions du mode RESERVE et des données. On retrouve les classiques CSAVE (sauvegarde), CLOAD (chargement), PRINT # (enregistrement de données) et INPUT # (rechargement de données). Avec CHAIN (programmable) et MERGE (en commande uniquement), un programme Basic pourra de lui-même se compléter en chargeant de nouvelles lignes depuis une cassette.

Au dos de l'ordinateur se situe une petite trappe destinée à recevoir une carte d'extension de la mémoire vive de 8 Ko (800 F) ou 16 Ko (1600 F) portant, respectivement, la capacité totale à 11262 et 19454 octets. Dotées chacune d'une pile, ces cartes — si l'on oublie leur prix — peuvent servir de mémoire de masse : on change alors de programmes comme de cartes.

Originalité : l'interface série est installée d'origine dans le PC-1350. Présentée à la norme RS-232C, on devrait donc pouvoir connecter n'importe quel périphérique à cette norme : modem, imprimante, ordinateur, etc. Il demeure que la tension aux bornes n'est que de 5 volts (au lieu de 12 V), et pratiquement, dans la plupart des cas, cela rend l'interface inutilisable sans convertisseur. Ce dernier n'est pas encore disponible. On se renseignera donc avant l'achat si l'on compte utiliser la RS-232 C. D'ores et déjà, le Basic contient quelques fonctions intéressantes destinées à l'exploitation de cette interface. Ainsi, TEXT convertit un programme Basic (codé d'une façon particulière et personnelle au PC-1350) en codes ASCII (format standard de codage des lettres et caractères). On pourra, via la RS-232C, « vider » un programme du 1350 vers le traitement de textes d'un autre ordinateur, etc.

Fiche technique du PC-1350

Constructeur : *Sharp (Japon)*
Distributeur : *SBM, 151-153, av. J.-Jaurès, 93307 Aubervilliers*
Prix public : *2300 FF*
Mémoire vive : *3070 octets, extensions en option 8 Ko ou 16 Ko*
Mémoire morte : *40 Ko*
Langages : *Basic, langage-machine*
Précision des calculs : *sur 12 chiffres*
Nombre de mots clés Basic : *92*
Variables : *numériques de 10 chiffres significatifs + exposant ; alphabétiques de 7 à 80 caractères*

Sharp a donc repris pour son PC-1350 un Basic traditionnel, performant, très souple et idéal pour débuter en informatique car doté d'un excellent éditeur. On pourra regretter l'absence de fonctions scientifiques vraiment sophistiquées (statistique, finances, etc.) car c'est un des domaines où les ordinateurs de poche ont à l'avenir une excellente carte à jouer.

Thierry LEVY-ABEGNOLI
Jean-Christophe KRUST

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...

31

Les fractions en mémoire

La représentation des nombres décimaux est très variable selon les ordinateurs utilisés. Mais d'une façon générale, il n'est pas possible de stocker exactement la valeur des nombres réels en mémoire. Prenons par exemple la valeur :

1/3 = 0,333333...

Quel que soit le nombre de chiffres significatifs que nous prenons, il n'est pas possible de représenter exactement cette fraction. En effet, comme elle admet un développement infini, il y aura toujours une erreur de troncature. Bien entendu, la plupart des ordinateurs effectuent leurs calculs en base 2 au lieu de la base 10. Mais le même type de problème existe, car le système binaire est un système de numération positionnel, comme le système décimal.

Toutefois, une fraction admettant un développement infini en binaire peut avoir un développement fini en décimal.

Vous pourrez peut-être trouver quatre fractions telles que :

1. la première ait un développement fini en décimal et fini en binaire ;

2. la seconde ait un développement fini en décimal et infini en binaire ;

3. la troisième ait un développement infini en décimal et infini en binaire ;

4. la quatrième ait un développement infini en décimal et fini en binaire.

32

Néologismes

Les mots que l'on utilise maintenant couramment en informatique sont souvent des néologismes. Ainsi, parlait-on de « traitement de l'information » pour désigner ce que l'on appelle maintenant informatique. Voici trois listes. Il s'agit de faire correspondre à chaque mot, sa date de création

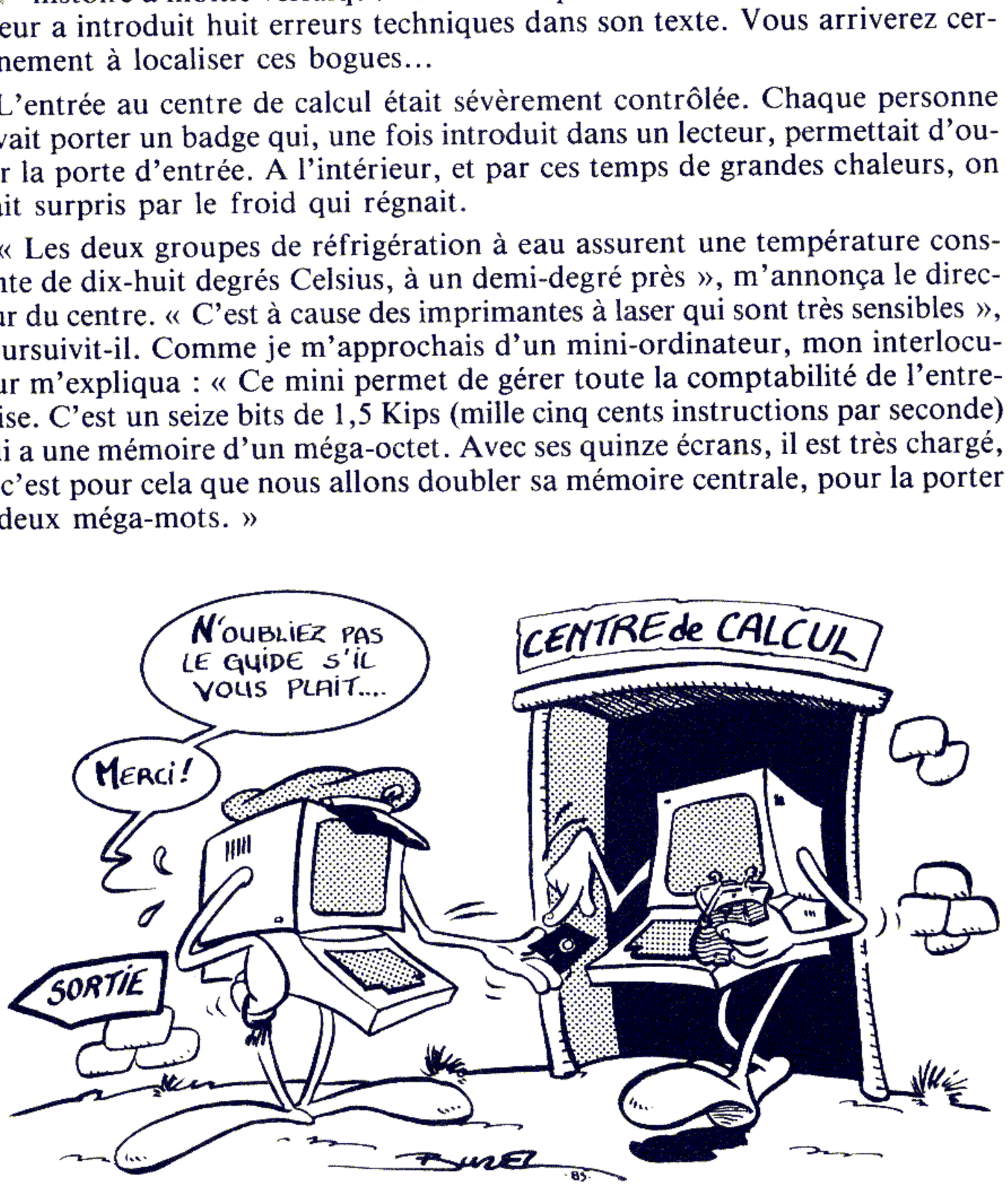
** Les solutions des jeux proposés ici paraîtront dans le prochain numéro de* LIST.

Visite au centre de calcul

Le texte suivant raconte une visite dans un centre de calcul. Il s'agit d'une histoire à moitié véridique : si la description des lieux est exacte, le narrateur a introduit huit erreurs techniques dans son texte. Vous arriverez certainement à localiser ces bogues...

L'entrée au centre de calcul était sévèrement contrôlée. Chaque personne devait porter un badge qui, une fois introduit dans un lecteur, permettait d'ouvrir la porte d'entrée. A l'intérieur, et par ces temps de grandes chaleurs, on était surpris par le froid qui régnait.

« Les deux groupes de réfrigération à eau assurent une température constante de dix-huit degrés Celsius, à un demi-degré près », m'annonça le directeur du centre. « C'est à cause des imprimantes à laser qui sont très sensibles », poursuivit-il. Comme je m'approchais d'un mini-ordinateur, mon interlocuteur m'expliqua : « Ce mini permet de gérer toute la comptabilité de l'entreprise. C'est un seize bits de 1,5 Kips (mille cinq cents instructions par seconde) qui a une mémoire d'un méga-octet. Avec ses quinze écrans, il est très chargé, et c'est pour cela que nous allons doubler sa mémoire centrale, pour la porter à deux méga-mots. »



Alors que les imprimantes à laser débitaient silencieusement depuis plus d'un quart d'heure leur flot continu de papier, les imprimantes à plasma se mirent à crépiter. « C'est ainsi que l'on voit le progrès technique », me dit mon guide, « ces imprimantes ont cinq ans de différence. Celles-ci éditent six cents lignes à la minute, alors que les nouvelles à laser débitent huit pages dans le même temps. Mais, bien entendu, il faut dire que nous avons des imprimantes à laser qui ont d'excellentes performances. »

Malgré ce vacarme, des analystes-programmeurs travaillaient tranquillement. « Ils font la chasse aux cafards », me dit-on. « La chasse aux cafards ? », m'exclamai-je, m'attendant à voir des insectes dans une salle aussi bien ordonnée. « Oui. Du debugging, ou débogage, comme vous voulez. La chasse aux erreurs, si vous préférez. »

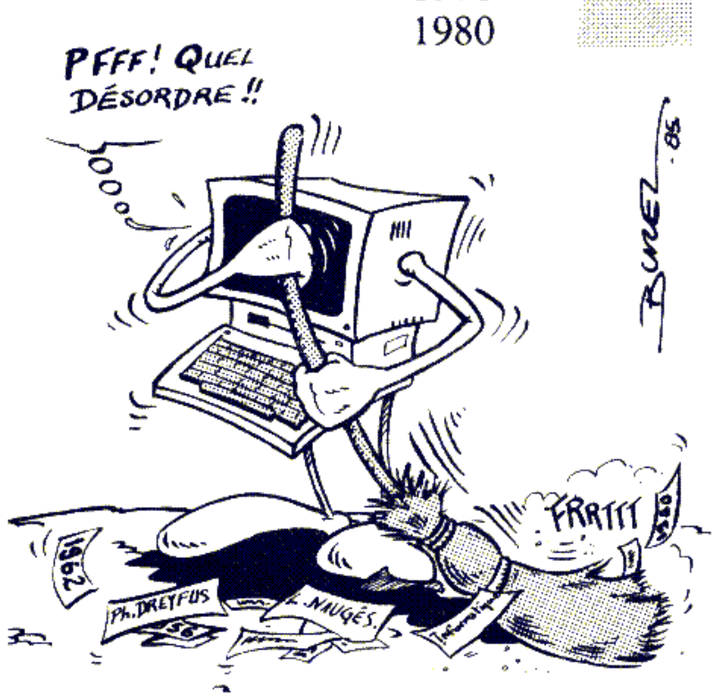
« Nous utilisons un compilateur Pascal interactif. Chaque instruction est lue par l'ordinateur, analysée, puis exécutée. Ainsi les erreurs sont très rapidement détectées, à condition qu'elles ne proviennent pas du compilateur lui-même, comme cela nous est arrivé plusieurs fois ! Et en ce qui concerne les éventuels cafards, poursuivit-il, espérons qu'il n'y en ait pas. Avec des têtes de lecture qui volent à un dixième de millimètre des disques durs, le moindre insecte pourrait causer de sérieux dégâts, non seulement à notre matériel, mais aussi aux données qui y sont enregistrées ! »

Pendant ce temps, insouciants, les dérouleurs de bandes continuaient à rechercher tranquillement leurs informations, tantôt dans un sens, tantôt dans l'autre, et les petites lampes rouges des disques clignotaient régulièrement, comme si elles étaient installées sur un arbre de Noël...

(Solution page suivante)

et le nom de la ou des personnes qui l'on inventé.

Bureautique	
Informatique	
Ordinateur	
Progiciel	
Télématique	1956
	1962
	1962
	1978
	1980

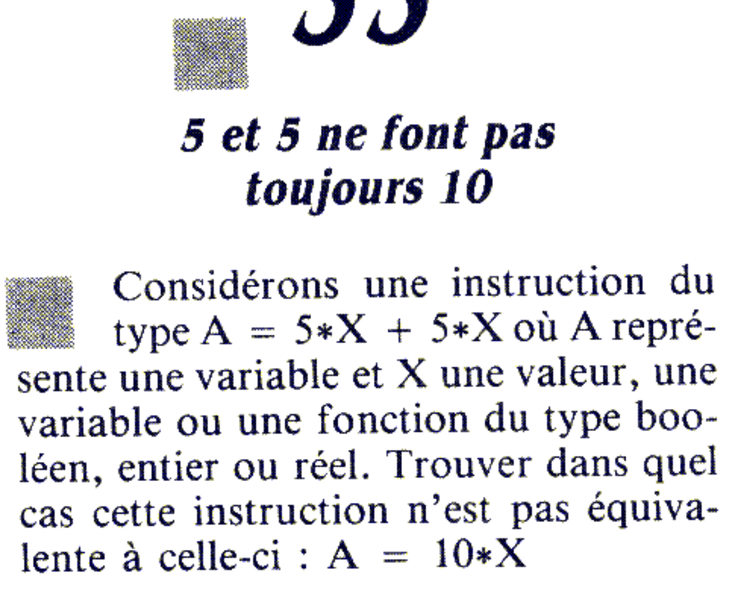


Ph. Dreyfus
J. E. Forges
L. Naugès
S. Nora et A. Minc
J. Perret

33

5 et 5 ne font pas toujours 10

Considérons une instruction du type A = 5*X + 5*X où A représente une variable et X une valeur, une variable ou une fonction du type booléen, entier ou réel. Trouver dans quel cas cette instruction n'est pas équivalente à celle-ci : A = 10*X



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 !

28

La ronde des variables

Les séquences qui échangent les contenus de trois variables A, B et C, sans passer par une variable auxiliaire sont différentes selon que la rotation se fait à gauche ou à droite.

1. Pour une rotation à gauche (A, B, C) devient (B, C, A) par :

```
C = C + B
B = C - B
C = C - B
B = B + A
A = B - A
B = B - A
```

2. Pour une rotation à droite (A, B, C) devient (C, A, B) par :

```
A = A + B
B = A - B
A = A - B
B = B + C
C = B - C
B = B - C
```

29

En peu d'instructions

En utilisant les instructions présentes sur toutes les implantations du langage Basic, l'initialisation d'une matrice unité peut s'effectuer en cinq lignes de la façon suivante :

```
10 for L = 1 to MAX
20    for C = 1 to MAX
30       A (L, C) = int (L/C)*int (C/L)
40    next C
50 next L
```

Un rapide calcul montre que la ligne 30 est exécutée L*C fois. Comme elle comporte deux divisions et une multiplication, elle nécessite donc 3*L*C opérations, ajoutées à L*C affectations.

30

Modulo

L'opérateur modulo (MOD) fournit le reste de la division entière et l'opérateur DIV, le résultat de cette division.

MOD n'existe pas sur tous les Basic. Il va falloir le remplacer par une fonction. Or, quelles que soient les valeurs de deux variables A et B, nous avons toujours : A = B* (A DIV B) + (A MOD B). En d'autres termes, cela signifie que le quotient d'une division entière multiplié par le diviseur, auquel on ajoute le reste de la division, donne toujours la valeur du dividende.

Cette égalité est équivalente à celle-ci : A MOD B = A − B * (A DIV B).

Notre fonction peut donc s'écrire : FNMOD(A,B) = A − B*(A DIV B) et, si nous ne disposons pas de l'opérateur DIV : FNMOD(A,B) = A − B*INT (A/B) où INT est la fonction retournant la partie entière d'un nombre réel.

Visite au centre de calcul

1. La température dans le centre de calcul n'est pas régulée pour les imprimantes à laser, peu sensibles aux écarts de température, mais pour les disques et les unités centrales.

2. Le mini-ordinateur n'a certainement pas une puissance de 1,5 Kips (mille cinq cents instructions par seconde), mais plutôt de 1,5 mips (1,5 million d'instructions par seconde).

3. Doubler la mémoire centrale qui est d'un méga-octet va la porter à deux méga-octets. Comme le mini-ordinateur a un processeur de seize bits, un mot est équivalent à deux octets. La mémoire sera donc portée à un méga-mot (deux méga-octets), et non deux méga-mots.

4. La technologie du plasma n'est pas utilisée pour les imprimantes, mais plutôt pour certains écrans de visualisation plats.

5. Une imprimante qui imprime six cents lignes par minute a approximativement les mêmes performances (voire supérieures) qu'une imprimante éditant huit pages dans le même temps (8 pages de 66 lignes faisant 528 lignes).

6. Une imprimante à laser qui imprime huit pages à la minute n'est certainement pas un modèle haute performance, mais au contraire un des plus petits modèles. Les grosses imprimantes à laser impriment une centaine de pages à la minute.

7. Un « compilateur Pascal interactif » où chaque instruction est lue et analysée avant d'être exécutée n'est pas un compilateur, mais un simple interpréteur.

8. La tête de lecture d'un disque dur ne vole pas à une altitude d'un dixième de millimètre de son plateau, mais de 0,4 millième de millimètre (avec la technologie Winchester) à 1,2 millième de millimètre (pour les disques durs amovibles).

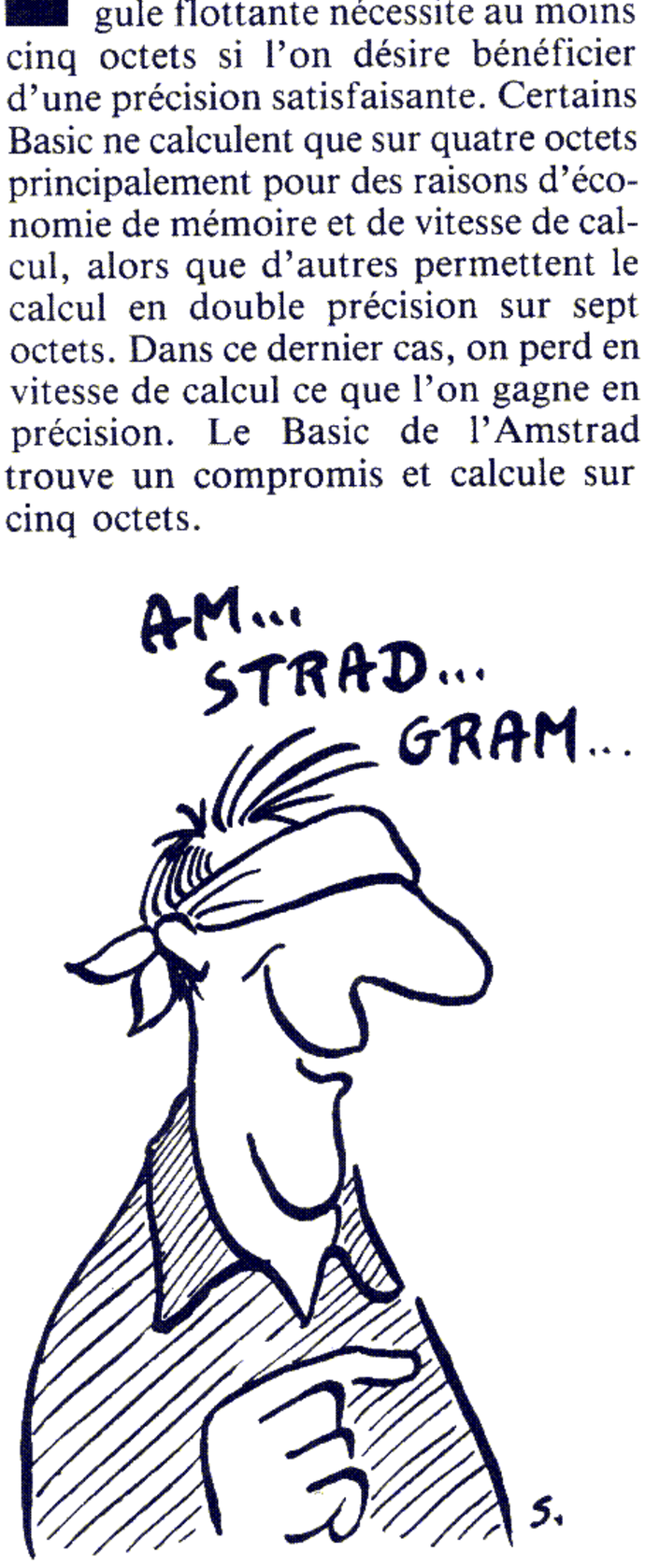
LA BOÎTE A MALICES...

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

CODER LA VIRGULE FLOTTANTE

Représenter un nombre en virgule flottante nécessite au moins cinq octets si l'on désire bénéficier d'une précision satisfaisante. Certains Basic ne calculent que sur quatre octets principalement pour des raisons d'économie de mémoire et de vitesse de calcul, alors que d'autres permettent le calcul en double précision sur sept octets. Dans ce dernier cas, on perd en vitesse de calcul ce que l'on gagne en précision. Le Basic de l'Amstrad trouve un compromis et calcule sur cinq octets.



L'un de ces cinq octets est utilisé pour l'exposant e, les quatre autres représentent la valeur de la mantisse m avec 9 chiffres significatifs. La formule est la suivante :

nombre = $0.m \times 2\hat{}(e-128)$

Le bit le plus significatif de m (le plus à gauche) est toujours à 1. Pour le Basic, ce bit représente le signe : 0 pour positif, 1 pour négatif. C'est-à-dire que si ce bit est à 0, il sera remplacé par 1, et un " + " sera mis devant le nombre ; si il est à 1, il garde cette

valeur et un " – " sera placé devant le nombre.

Par exemple, le nombre flottant 5467.75 est codé par la machine avec cinq octets : 0 0 222 42 141, en décimal. Soit en hexadécimal : 0 0 DE 2A 8D. L'exposant e est alors égal à 8D en hexa, ou 141 en décimal. Donc : e – 128 = 13 (en décimal).

Et la mantisse m est représentée sur quatre octets, en hexa, par : 2A DE 00 00. En binaire, cette mantisse est donc : m = 00101010 11011110 00000000 00000000.

Le bit le plus à gauche est à 0, il est donc mis à 1 et le signe " + " est placé devant le nombre. La formule donnant ce nombre est appliquée. Soit : nombre = + 0.10101010 11011110 00000000 00000000 × 2^13

Pour multiplier un nombre par 2^n, on déplace la virgule de n rangs vers la droite si n est positif, et de n rangs vers la gauche si n est négatif. Le nombre devient alors : nombre = 10101010011011.11

Soit : nombre = (2^12 + 2^10 + 2^8 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0) + (2^-1 + 2^-2) = 5467 + 0.75

On retrouve bien notre valeur initiale ! Il existe toutefois une exception à la règle pour le nombre zéro qui est codé par cinq octets nuls.

Les systèmes de codage sont différents d'un Basic à l'autre, surtout pour le codage du signe et l'ordre des octets. Aussi, cette règle ne saurait être toujours applicable sur des ordinateurs autres que l'Amstrad.

Stéphane CHICHE

SINCLAIR QL

LES PARAMÈTRES DU QL

Sur le QL, à chaque fois que l'on DEFinit une fonction ou une procédure qui utilise, sous d'autres noms, les données du programme appelant, on emploie des paramètres formels. Ils ne sont pas connus par le programme appelant, on dit qu'ils sont locaux.

Il ne faut pas confondre ces paramètres formels avec les paramètres actuels, utilisés lors de l'appel d'une fonction ou d'une procédure.

Sur le QL, les variables réagissent de manière inattendue. Ainsi, avec l'exemple 1, si nous tapons :
affiche "bonjour"
affiche 2 + 2
les résultats s'affichent, le premier ordre renvoie "bonjour" et le second, 2 + 2. Ceci laisse supposer que la donnée numérique 2 + 2 a été convertie automatiquement en chaîne, aucun message d'erreur n'apparaissant.

Mais, si avec l'exemple 2, nous tapons :
montre 14
montre "bonjour"
alors, "14" puis "bonjour" s'affichent sans aucun message d'erreur. Or si la variable avait été forcée automatiquement dans le type numérique, on aurait dû avoir "14" suivi d'un message d'erreur. On peut donc raisonnablement penser que les paramètres formels n'ont pas de type en eux, mais prennent automatiquement le type du paramètre actuel qui leur est associé lors de chaque appel. Ainsi, en tapant :
montre "bonjour"
la variable chiffre contient vraiment le mot "bonjour" et elle est considérée comme une variable alphanumérique. De même, ce n'est pas parce que le nom d'une fonction se termine par le signe "$" qu'elle renvoie un résultat alphanumérique.

Une instruction pose le même type de problèmes. Il s'agit de SELect : elle ne fonctionne qu'avec des variables réelles. Il faut donc connaître le type de résultat que renvoie chaque fonction (voir le tableau ci-dessous). Ainsi, on peut forcer la conversion dans le type voulu, au début de la procédure, en déclarant des variables locales.

Exemple 1

```
10 DEFine PROCedure affiche(message$)
20 PRINT message$
30 RETurn
40 END DEFine
```

Exemple 2

```
10 DEFine PROCedure montre(chiffre)
20 PRINT chiffre
30 RETurn
40 END DEFine
```

Tableau des types de résultats renvoyés par chaque fonction

Résultats entiers :				
CODE	DIMN	DIV	INSTR	KEYROW
LEN	MOD	PEEK	PEEK__W	RND( )
Résultats réels :				
ABS	ACOS	ACOT	ASIN	ATAN
COS	COT	DEG	INT	LN
LOG10	PEEK__L	RAD	RND	SIN
SQRT	TAN	EXP	+	–
*	/	OR	AND	XOR
NOT	< >	<	>	< =
> =	= =	=	BEEPING	

Frédéric BLONDIAU

PROBLÈME

Comment faire éditer, ligne par ligne, le fameux Triangle de Pascal par un très court programme Basic ? Quelques indications sont nécessaires :

1. Chaque terme du triangle (voir schéma ci-contre) est la somme des deux termes situés au-dessus de lui, sur la ligne précédente. Le stockage d'un tableau carré de $(N+1)\times(N+1)$ nombres en double précision serait vite prohibitif mais avec un peu de réflexion, on se contentera de stocker une seule ligne de coefficients. Neuf instructions devraient suffire.

Le début du Triangle de Pascal

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
    .   .   .   .   .   .   .
```

2. Le K+1ème terme de la N+1ème ligne vaut N!/(K!*(N−K)!). L'utilisation des factorielles en Basic est particulièrement pénible, et entraîne des dépassements de capacité prématurés. Elle est donc à déconseiller.

3. Vous pouvez tirer du point **2.** le moyen de calculer chaque terme à partir du précédent sur la même ligne. Votre programme devrait se réduire à huit instructions.

4. Avec un Basic acceptant les expressions booléennes, et avec des circonvolutions cérébrales suffisamment tordues, vous économiserez encore une instruction... en en compilant deux autres. Ce genre d'acrobatie n'est guère recommandé, mais toujours bon à connaître.

5. N'hésitez pas à faire encore plus court si une heureuse inspiration vous favorise.

Quelques précisions :

- si vous ne disposez pas du NEXT multiple, et devez écrire NEXT J:NEXT I, au lieu de NEXT J,I, vous aurez droit à une instruction de plus ;
- n'oubliez ni l'entrée du nombre de lignes désirées (en fait, on affiche N+1 lignes), ni le retour à gauche après affichage de chaque ligne de coefficients ;
- en double précision, on dépasse les 16 chiffres pour N=57, et on dépasse la capacité de l'Epson PX-8 pour N=124.

(Voir les solutions proposées, page 75).

Pierre BARNOUIN

ALICE

LES MALICES D'ALICE

Des possibilités bien « cachées » de l'Alice, 32 ou 90, sont directement accessibles à partir du Basic. Certaines sont étonnantes. Par exemple, POKE 251,*n* définit à la fois la couleur des caractères et celle du fond de l'écran avec pour seul paramètre, le nombre *n* : ce nombre doit être de la forme 16*L+F où L est la couleur des lettres et F celle du fond. Ainsi, L=6 et F=1 donnent des lettres bleu pâle sur un fond rouge. Une valeur de F supérieure à 8 provoque un clignotement.

Plus subtile, la suite d'instructions :

POKE 48935,X où X est le numéro de la colonne

POKE 48934,Y où Y est le numéro de la ligne

POKE 48929,AA où AA est le code du caractère à afficher

POKE 48931,A où A définit la couleur du caractère et celle du fond

POKE 48930,B où B définit la taille du caractère, son type, un clignotement, etc.

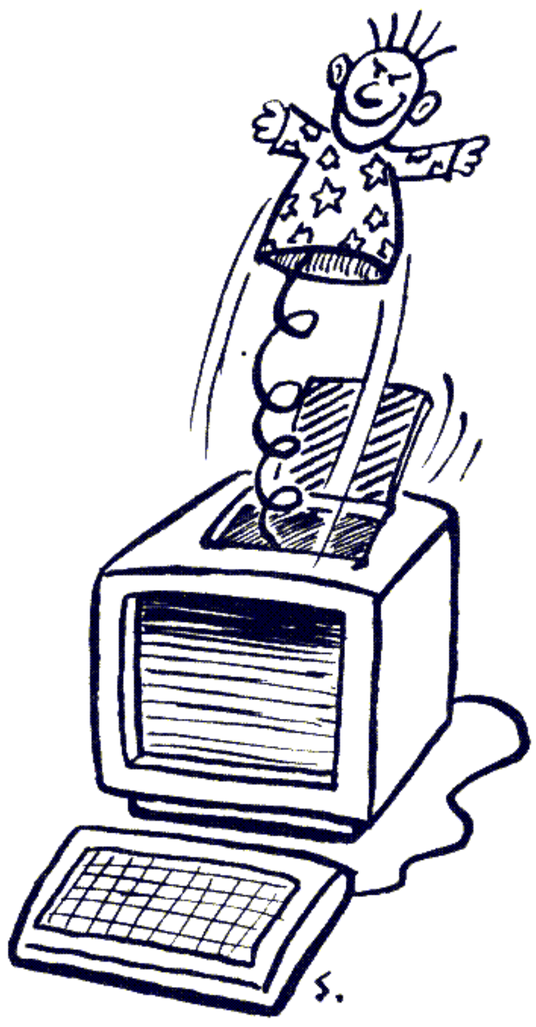
POKE 48936,C où C correspond au mode (0 ou 1)

Les paramètres X, Y et AA sont très classiques. Le paramètre A doit être de la forme 16*L+F (comme *n*, plus haut). Les couleurs obtenues sont différentes selon la valeur de C (0 ou 1). Pour B, selon la valeur des restes dans les divisions successives par 2, on obtiendra divers résultats : double largeur, double hauteur, etc.

Enfin, le petit programme ci-dessous permettra une exploration plus systématique des diverses possibilités. La première ligne (EXEC 54879) offre un passage direct en grand écran dans le programme, ce que ne fait normalement pas le Basic. Ici, l'affichage sera en blanc sur fond bleu. La ligne 15010 étudie le paramètre B : si on est en double hauteur ou en double largeur, il faut doubler le

En couleur
Programme pour Alice 32 ou 90
Auteur Jean-Michel Gaudin


```
0 EXEC 54879
5 INPUT"CLS";CL:INPUT"A";A:INPUT
"B";B:INPUT"C";C
10 CLS(CL):A$="LIST":LO=10:PO=5:
GOSUB 15000
100 GOTO 5
15000 A1=48935:A2=48934:A3=48929
:A4=48931:A5=48930:A6=48936
15010 HH=-(B-4*INT(B/4)>1):LL=-(
B-16*INT(B/16)>7)
15020 LN=LEN(A$):FOR NN=1 TO LN:
AA=ASC(MID$(A$,NN,1)):GOSUB 1503
0:NEXT NN:RETURN
15030 FORNL=0TOLL:FORNH=0TOHH:PO
KE A1,PO+(1+LL)*NN-NL:POKE A2,LO
+6+NH:POKE A3,AA:POKE A4,A:POKE
A5,B:POKE A6,C:NEXTNH:NEXTNL
15040 RETURN
```



caractère à afficher. Il est bien sûr possible de mettre n'importe quel texte dans A$ en ligne 10, et de modifier les positions de début d'affichage LO et PO.

Jean-Michel GAUDIN

Conversions
Programme pour TO7 et TO7/70
Auteur Jean-Paul Carré
Copyright *LIST* et l'auteur

```
140 CLS:SCREEN7,0,0,,0
150 INPUT"Entrez votre nombre suivi de D,H,O ou B suivant que c'est un
    Nombre decimal,    Hexadecimal,Octal ou Binaire";N$
160 TYPE$=RIGHT$(N$,1)
170 IF TYPE$="D" THEN PRINT"il s'agit d'un nombre decimal":GOSUB 1000
180 IF TYPE$="H" THEN PRINT"il s'agit d'un nombre hexadecimal":GOSUB 2000
190 IF TYPE$="O" THEN PRINT"il s'agit d'un nombre octal":GOSUB 3000
200 IF TYPE$="B" THEN PRINT"il s'agit d'un nombre binaire":GOSUB4000
210 PRINT:PRINT:GOTO150
1000 '-----------------------------------------traitement des decimaux
1010 N=VAL(LEFT$(N$,LEN(N$)-1))
1020 PRINT"dont la valeur Hexadecimale est:";HEX$(N)
1030 PRINT"dont la valeur Octale est      :";OCT$(N)
1040 PRINT"dont la valeur Binaire est     :";
1050 '   ***calcul de la valeur binaire***
1060 FOR I=7 TO 0 STEP-1 :B(I)=N@(2^I):N=N-B(I)*(2^I):NEXT I
1070 FOR I=7 TO 0 STEP-1:PRINT RIGHT$(STR$(B(I)),1);
1080 NEXT I
1090 RETURN
2000 '-----------------------------------------traitement des Hexadecimaux
2010 N=VAL("&H"+(LEFT$(N$,LEN(N$)-1)))
2020 PRINT"dont la valeur Decimale est    :";N
2030 PRINT"dont la valeur Octale est      :";OCT$(N)
2040 PRINT"dont la valeur Binaire est     :";:GOSUB 1050
2050 RETURN
3000 '-----------------------------------------traitement des Octaux
3010 N=VAL("&O"+(LEFT$(N$,LEN(N$)-1)))
3020 PRINT"dont la valeur Decimale est    :";N
3030 PRINT"dont la valeur Hexadecimale est:";HEX$(N)
3040 PRINT"dont la valeur Binaire est     :";:GOSUB 1050
3050 RETURN
4000 '-----------------------------------------traitement des binaires
4010 N$=LEFT$(N$,LEN(N$)-1):N$="&B"+N$
4020 N=VAL(N$)
4030 PRINT"dont la valeur decimale est    :";N
4040 PRINT"dont la valeur Hexadicimale est:";HEX$(N)
4050 PRINT"dont la valeur Octale est      :";OCT$(N)
4060 RETURN
```

TO7-TO7/70

D'UNE BASE A L'AUTRE

Des nombres donnés en base décimale, hexadécimale, octale ou binaire peuvent être convertis dans une autre de ces bases, grâce à un programme écrit ici pour TO7 ou TO7/70. Les nombres entrés doivent être entiers et positifs. De plus, ils ne peuvent dépasser la valeur maximale possible d'un octet, à savoir 255 en décimal, FF en hexadécimal, 377 en octal et 11111111 en binaire.

De la ligne 160 à la ligne 200, le dernier caractère du nombre envoyé est décodé (D, H, O ou B). Suivant le code, un branchement a lieu vers le sous-programme de traitement de la base considérée. Ainsi, le sous-programme de traitement des décimaux va de la ligne 1010 à la ligne 1090. Après avoir débarrassé le nombre de son code et transformé la chaîne N$ en variable numérique M, on trouve :

- la valeur hexadécimale avec l'instruction HEX$;
- la valeur octale avec OCT$;
- la valeur binaire avec un calcul préalable. Il faut décomposer le nombre en puissances décroissantes de 2 et générer un octet B(I) (I allant de 7 à 0) correspondant respectivement aux bits de poids décroissant. En ligne 1070, chaque variable numérique B(I) est retransformée en chaîne et le premier caractère, en l'occurrence un espace, est retiré.

Le sous-programme de traitement des nombres hexadécimaux va de la ligne 2010 à la ligne 2050. Pour obtenir la valeur décimale, il suffit de demander l'affichage du nombre hexa N$ débarrassé de son code et précédé de &H. Pour la valeur octale, il ne reste plus qu'à utiliser OCT$. Enfin, pour la valeur binaire, on fait appel au même sous-programme que pour le traitement des nombres décimaux.

Les nombres octaux sont traités comme les hexadécimaux (lignes 3010 à 3050). Quant aux nombres binaires (lignes 4010 à 4050), comme pour les nombres hexadécimaux, on débarrasse la chaîne N$ de son suffixe et on lui ajoute le préfixe &B de façon à obtenir la valeur décimale lors de la demande de l'affichage. Il ne reste plus qu'à appliquer HEX$ ou OCT$ à ce nombre pour avoir son équivalent hexadécimal ou octal.

Jean-Paul CARRÉ

DES MENUS PLUS PRÉSENTABLES

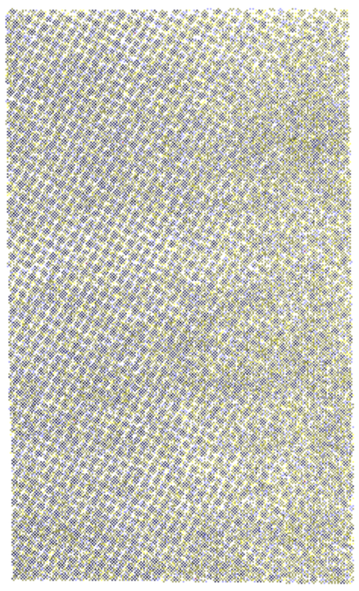
Sur le ZX 81, la présentation classique des menus est assez peu attrayante, en raison en particulier de l'attente d'une réponse en bas d'écran (instruction INPUT). Le petit programme ci-dessous, facilement adaptable sur d'autres machines, permet de proposer des menus à la manière des ordinateurs les plus perfectionnés.

Le principe est élémentaire : les différents choix possibles sont affichés à l'aide d'instructions PRINT et la touche NEWLINE permet de déplacer un curseur en vidéo inverse devant les options. La validation est obtenue avec une pression sur la touche A.

Le programme dans ses grandes lignes

10-30 : affichages pratiques
40-60 : affichage des choix proposés
70 : attente de la pression d'une touche
80-130 : boucle de déplacement du curseur
90-120 : affichage et effacement du curseur
140 : retour en début de boucle
110 : test de validation et branchement vers la suite du programme
150-510 : affichage du résultat

```
  5 CLS
 10 PRINT AT 3,10;"MENU"
 20 PRINT AT 6,0;"SELECTION: TO
UCHE ""NEWLINE"""
 30 PRINT AT 8,0;"VALIDATION:TO
UCHE ""A"""
 40 PRINT AT 12,4;"CHOIX NO 1"
 50 PRINT AT 14,4;"CHOIX NO 2"
 60 PRINT AT 16,4;"CHOIX NO 3"
 70 IF INKEY$="" THEN GOTO 70
 80 FOR R=1 TO 3
 90 PRINT AT 10+R*2,0;"    "
100 IF INKEY$="" THEN GOTO 100
110 IF INKEY$="A" THEN GOTO 150
120 PRINT AT 10+R*2,0;"    "
130 NEXT R
140 GOTO 80
150 CLS
160 GOTO 200+R*100
300 PRINT AT 2,9;"CHOIX NO 1"
310 STOP
400 PRINT AT 2,9;"CHOIX NO 2"
410 STOP
500 PRINT AT 2,9;"CHOIX NO 3"
510 STOP
```



Menu
Programme pour ZX 81
Auteur Yvon Pérès
Copyright *LIST* et l'auteur

Dans certains cas, il sera intéressant de modifier ce programme pour en faire un sous-programme paramétré que l'on appellera dès que l'on aura besoin d'un menu.

Yvon PÉRÈS

IMPRESSION DE FICHIERS BASIC

A l'aide d'une HP-41C, d'une boucle HP-IL, d'une imprimante HP-82143A, d'un module Forth et de deux petits programmes, il est possible de lister n'importe quel fichier Basic du HP-71B sur l'imprimante HP-82143A. Après avoir entré les deux programmes dans leur ordinateur respectif (101 octets dans la HP-41C, 305 dans le HP-71B au nom de PRPBASIC), on suit la procédure :

- mettre l'interrupteur de la boucle HP-IL de la HP-41C sur DISABL ;
- faire CONTROL OFF sur le HP-71 (la 41 est obligatoirement contrôleur) ;
- faire XEQ^{T}PRPBSIC sur la 41 ; entrer le nom du fichier Basic à lister sur le 71, puis ENDLINE ;
- une fois le listage achevé, la 41 demande si vous désirez un nouvel exemplaire (répondre O pour oui, N pour non, puis R/S).

Le principe utilisé ici consiste à transformer le fichier Basic en un fichier TEXT. Le fichier est alors envoyé enregistrement par enregistrement vers la HP-41 qui les imprimera au fur et à mesure. La première liste demandera un peu de patience, les suivantes seront instantanées : la transformation initiale ne s'effectue qu'une fois.

Le programme PRPBASIC utilise

PRPBSIC
Programme pour la HP-41C

```
 01♦LBL "PRPBSIC"
AUTOIO  REMOTE
"RUNPRPBASIC"  OUTA  IND

 07♦LBL 02
RCL X

 09♦LBL 00
INA  PRA  DSE X  GTO 00
ADV  ADV  ADV  ADV  ADV
ADV  "ENCORE ? O/N"  AON
PROMPT  AOFF  ATOX  79
X≠Y?  GTO 01  R↑
"RUN,AG"  OUTA  GTO 02

 32♦LBL 01
"RUN,CLR"  OUTA  BEEP
CLST  CLA  END

END          101 BYTES
```

PRPBASIC
Programme pour le HP-71B

```
10 ! PRPBASIC
20 OPTION BASE 1
30 DESTROY ALL @ DIM C$[
96]
40 INPUT "Nom ? ";A$
50 COPY A$ TO WF
60 TRANSFORM WF INTO TEX
T
70 D=FILESZR("WF")
80 ASSIGN #1 TO WF
90 FOR A=0 TO D-1
100 READ #1,A;C$
110 B=1+INT(LGT(VAL(C$[1
,4])))
120 C$=C$[5-B,4]&C$[5] @
 E=E+CEIL(LEN(C$)/24)
130 REPLACE #1,A;C$
140 NEXT A
150 OUTPUT :LOOP ;E
160 'AG': ASSIGN #1 TO W
F
170 FOR A=0 TO D-1
180 READ #1;C$ @ OUTPUT
:LOOP ;C$
190 NEXT A
200 END
210 'CLR': PURGE WF @ EN
D
```

deux mots Basic apportés par le module Forth/Assembleur : le premier FILESZR retourne le nombre d'enregistrement d'un fichier TEXT, le second REPLACE remplace un enregistrement par un autre.

Les lignes Basic qui font exactement 24 caractères doivent être évitées : elles produisent des sauts de ligne inesthétiques et problématiques. En cas d'arrêt avant la fin de l'impression, pour reprendre le contrôle, il faut faire SEND RDY64 sur le HP-71, imprimante éteinte. Enfin, en interrompant le HP-71 pendant les transmissions, on prend le risque de voir un indésirable Memory Lost...

Olivier ARBEY

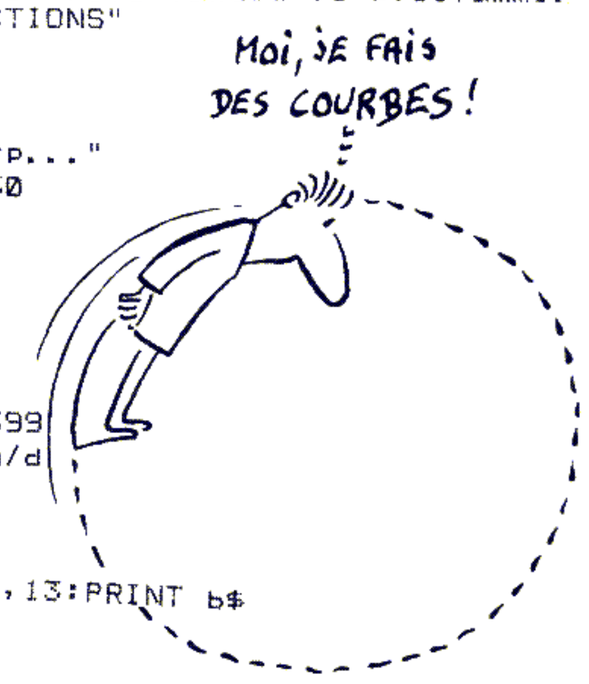
AMSTRAD

TOUJOURS DES COURBES

L'Amstrad peut, lui aussi, tracer les courbes représentatives de fonctions avec un petit programme Basic. C'est simple, on introduit la fonction choisie, en éditant la ligne 30, et en remplaçant la suite de points par la définition de la fonction (par exemple, 2*x – 3, x↑2 – 1, etc.). Aux lignes 40 et 50, on introduit les bornes de l'intervalle sur lequel on souhaite voir la courbe.

Traceur de courbes
Programme pour Amstrad CPC 464
Auteur Christian de Guillebon


```
10 REM introduisez la fonction en ligne 30, puis lancez le programme.
20 MODE 1:LOCATE 11,1:PRINT"TRACE DE FONCTIONS"
30 DEF FN f(x)=.........
40 LOCATE 1,5:INPUT"minimum (x)=";a
50 INPUT"maximum (x)=";b
60 IF a)b THEN 40:ELSE PRINT"patientez svp..."
70 DIM y(639):c=(b-a)/639:m=1E+30:p=-1E+30
80 FOR i=0 TO 639
90 y(i)=FN f(a+c*i)
100 p=MAX(p,y(i))
110 m=MIN(m,y(i))
120 NEXT
130 d=(p-m)/399:MODE 2
140 IF a*b<0 THEN PLOT -a/c,0:DRAW -a/c,399
150 IF m*p<0 THEN PLOT 0,-m/d:DRAW 639,-m/d
160 PRINT STR$(ROUND(p,1))
170 LOCATE 1,25:PRINT STR$(ROUND(m,1))
180 LOCATE 1,13:PRINT STR$(ROUND(a,1))
190 b$=STR$(ROUND(b,1)):LOCATE 80-LEN(b$),13:PRINT b$
200 FOR i=0 TO 639
210 PLOT i,(y(i)-m)/d
220 NEXT
230 IF INKEY$="" THEN 230
240 END
```



La ligne 70 initialise quelques variables : c définit la variation de l'abscisse en passant d'un pixel horizontal au suivant (640 pixels, soit 639 intervalles) ; y(639) va recevoir les valeurs de f(x) correspondant aux 640 pixels horizontaux. Les valeurs minimale et maximale de f(x) dans l'intervalle se trouvent respectivement en m et p.

La boucle FOR...NEXT des lignes 80 à 120, calcule les 640 valeurs de f(x), les mémorise dans y(i) et actualise m et p. La variation d de l'ordonnée qui va d'un pixel vertical au suivant (400 pixels, soit 399 intervalles) est calculée à la ligne 130.

Les lignes 140 et 150 tracent les axes (s'ils existent). Les lignes 160 à 190 affichent à l'écran les valeurs minimales et maximales des abscisses et des ordonnées. La boucle FOR...NEXT des lignes 200 à 220 trace la courbe. Enfin, la ligne 230 fixe l'écran en attendant l'appui sur une touche, pour relancer le programme.

Christian de GUILLEBON

TÉLÉPHONE A ACCÈS DIRECT

Un répertoire téléphonique dans un ordinateur de poche, c'est pratique. Avec un PB-700, par exemple, il suffit de le programmer en Basic. Le principe de base est le suivant : pour stocker un nom et un numéro de téléphone, on crée une ligne de DATA dont le numéro sera calculé par le programme en fonction des codes ASCII des deux premières lettres du nom (lignes 90 et 230). Pour chercher ce nom, le programme va lire cette ligne de DATA , et uniquement celle-là, en utilisant RESTORE suivi du numéro de ligne.

Mais que se passe-t-il si l'on doit stocker deux noms commençant par les mêmes deux premières lettres ? Dans ce cas, on les stocke sur la même ligne de DATA. Ils seront différenciés par les autres lettres : le nom complet étant tapé au clavier, s'il ne correspond pas au premier stocké sur la ligne, le pointeur ira chercher le suivant et vérifiera que c'est le bon. Seule restriction : pas plus de trois noms commençant par les mêmes deux premières lettres.

Lors de la création du répertoire, le numéro de la ligne de DATA à construire est affiché : c'est ce numéro de ligne qui permettra l'accès direct lors de l'utilisation de ce programme (une fois le répertoire créé).

Ainsi, pour rechercher un numéro de téléphone, il suffit de taper RUN, puis le nom du correspondant : son numéro apparaît aussitôt, suivi du numéro de la ligne de DATA sur laquelle il se trouve. Ceci pour le cas où une modification s'avérerait nécessaire.

Comme la création de l'accès direct aux numéros de téléphone de ce répertoire tient essentiellement dans les lignes 90 et 230, le programme est facilement transposable à d'autres Basic, pourvus des instructions ASC, READ, DATA, RESTORE, et dont l'instruction RESTORE peut être suivie d'un numéro de ligne.

Patrick EMIN

Répertoire téléphonique
Programme pour PB-700
Auteur Patrick Emin


```
10 REM REPERTOIRE TELEPHONIQUE
20 REM------------------------------
30 CLEAR
40 CLS
50 REM----------Recherche----------
60 LOCATE 0,2
70 INPUT "QUEL NOM";NM$
80 REM---------Calcul du no de ligne
90 D=ASC(NM$)*100+ASC(MID$(NM$,2,1))
100 REM--Lecture du nom dans les data
110 RESTORE D
120 READ AA$,BB$
130 REM Controle 2 premieres lettres-
140 IF LEFT$(AA$,2)<>LEFT$(NM$,2) THEN
     60
150 REM -------Controle du nom entier
160 IF AA$<>NM$ THEN 120
170 REM--------------Affichage du nom
    et du numero de ligne
180 CLS :PRINT AA$,BB$:LOCATE 14,3:PRI
    NT D;
190 IF INKEY$="" THEN 190 ELSE 30
200 REM---------Calcul du no de ligne
    pour une entree----------------
210 CLS :LOCATE 0,2
220 INPUT "QUEL NOM";NM$
230 D=ASC(NM$)*100+ASC(MID$(NM$,2,1))
240 PRINT D
250 REM-----------------------------
6560 REM Espace reserve pour les lignes
      de data-----------------------
6582 DATA ARTHUR,56 15 12
6885 DATA DUPONT,123 45 67,DURAND,765 4
    3 21,DUBOIS,478 25 36
7269 DATA HENRI,16 (47) 38 52 14
7765 DATA MARTIN,465 71 23, MARLIN,579
     23 54
9100 REM-----------------------------
```

Exemple d'exécution

Pour créer le répertoire, taper RUN 200. Un message s'affiche : QUEL NOM ?

On introduit, par exemple : MARTIN. Un numéro apparaît : 7765

On crée alors la ligne suivante : 7765 DATA MARTIN, 9990000

Si un autre nom commençant par MA devait être enregistré, il suffirait de le placer à la suite du premier.

Et pour rechercher le numéro de téléphone de MARTIN, on tape RUN, puis le nom. Alors 9990000, suivi de 7765, apparaissent.

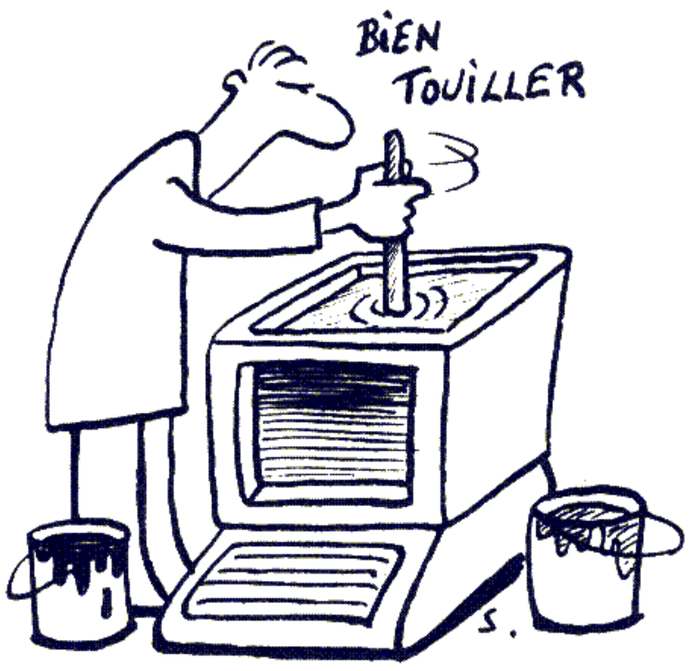
MÉLANGER LES COULEURS

La routine FILL du Dai est d'usage très pratique, car elle remplit un rectangle avec une couleur donnée. Pour cela, il suffit d'indiquer les coordonnées des deux coins opposés. En outre, cette routine opère rapidement, ce qui n'est pas le cas de bien de ses homologues, sur d'autres machines. Que demander de plus, alors ?

Justement, il y a mieux à faire ! Le programme proposé ci-contre permet de remplir un rectangle, *en mélangeant deux couleurs,* au moyen de hachures verticales alternées. Cela aboutit à toute une série d'effets graphiques nouveaux, pour le plaisir des yeux, et la satisfaction du programmeur.

Si l'on analyse la routine FILL du Dai, on remarque que celle-ci commence par appeler l'adresse E83A, en ROM2. Il existe, à cet endroit, un CALL E9C3, qui branche sur une routine initialisant les adresses C2 et C3 de la page 0, avec des valeurs correspondant précisément à la couleur du FILL. En mode 4 couleurs, ces deux adresses contiennent 0 ou FF. Notre programme se propose de supprimer l'appel en E9C3. Ainsi, la routine FILL, abusée par tant de rouerie, utilisera les valeurs en C2 et C3, sans se rendre compte, la pauvre, que c'est le programmeur qui les a définies, pour avoir un remplissage bicolore.

Voici comment faire. D'abord,



taper et assembler le source en langage-machine (eh, oui !). Ensuite, définir un mode graphique 4 couleurs, et choisir ces couleurs (COLORG...) le plus artistement possible. Puis, ajuster C2 et C3. Par exemple, POKE # C2, # AA: POKE # C3, # 55 intimera au FILL l'ordre de faire du remplissage avec des lignes verticales alternant les couleurs des registres 2 et 3, que vous venez de choisir. Enfin, il faut passer à la routine les coordonnées du rectangle à peindre :

POKE # 34A, X1 MOD 256:
POKE # 34B, X1/256:
POKE # 34C, Y1
POKE # 34D, X2 MOD 256:
POKE # 34E, X2/256:
POKE # 34F, Y2

Vous l'avez deviné : X1,Y1 et X2, Y2 sont les coordonnées cartésiennes habituelles d'un rectangle, quelque part dans l'écran graphique. Bien entendu, avant de bidouiller tout cela, vous aurez pris la précaution de placer les pointeurs Basic au-dessus de # 350 (POKE # 29C, # 04, pour être tranquille), puis NEW. Ne pas oublier aussi, pendant que j'y pense, d'annuler l'animate flag, en faisant : POKE # C1,0. Tapez CALLM # 300, et admirez le travail. C'est beau, n'est-ce pas ?

Walter COSTA

```
0000                    ;
0000                    TITL        'ROUTINE PAINT'
0000            ;
0000            ;
0000                    ORG         300H
0300            ;
0300  E5                PUSH H
0301  C5                PUSH B
0302  D5                PUSH D
0303  F5                PUSH PSW
0304  CD2B03            CALL        ROM2            ;ROM 2 SELECTIONNEE
0307  2A4A03            LHLD        X1
030A  EB                XCHG                        ;X1 DANS DE
030B  3A4C03            LDA         Y1
030E  47                MOV B,A                     ;Y1 DANS B
030F  2A4D03            LHLD        X2              ;X2 DANS HL
0312  3A4F03            LDA         Y2
0315  4F                MOV C,A                     ;Y2 DANS C
0316  CD2103            CALL        FILL
0319  CD3E03            CALL        OLDROM          ;RESTITUE LA ROM
031C  F1                POP PSW
031D  D1                POP D
031E  C1                POP B
031F  E1                POP H
0320  C9                RET
0321            ;
0321  F5        FILL    PUSH PSW
0322  C5                PUSH B
0323  D5                PUSH D
0324  E5                PUSH H
0325  CD40E8            CALL        0E840H          ;CONTROLE LES ARGUMENTS DE FILL
0328  C320E8            JMP         0E820H          ;SAUT DANS FILL
032B  F3        ROM2    DI
032C  3A4000            LDA         40H
032F  E1                POP H
0330  F5                PUSH PSW
0331  E5                PUSH H
0332  E63F              ANI         3FH
0334  F680              ORI         80H
0336  324000            STA         40H
0339  3206FD            STA         0FD06H
033C  FB                EI
033D  C9                RET
033E            ;
033E  E1        OLDROM  POP H
033F  F1                POP PSW
0340  E5                PUSH H
0341  F3                DI
0342  324000            STA         40H
0345  3206FD            STA         0FD06H
0348  FB                EI
0349  C9                RET
034A            ;
034A            ;COORDONNEES DU RECTANGLE
034A  0000      X1      DW          0H
034C  00        Y1      DB          0H
034D  0000      X2      DW          0H
034F  00        Y2      DB          0H
0350            ;
0350            END     END
```

Peinture
Routine pour Dai
Auteur Walter Costa
Copyright *LIST* et l'auteur

PETITES ROUTINES EN MUSIQUE

Les programmes professionnels ont souvent des « bips » ou des « jingles » qui leur sont propres pour indiquer une fausse manœuvre, un problème ou l'achèvement d'un travail. En Basic, on peut y arriver aussi. On utilise alors une ou plusieurs routines en langage-machine puisque ni l'Applesoft, ni l'Integer n'ont d'instructions musicales intégrées.

Tout d'abord, comment fait-on du bruit sur l'Apple ? L'ordinateur dispose d'un haut-parleur qui se fait entendre par des petits clics. Ces clics sont produits par l'appel d'une adresse particulière : $C030 (hexa), 49200 ou −16336 (décimal). Essayez le petit programme suivant :

```
10 FOR A=0 TO 100
20 B=PEEK (49200)
30 NEXT
```

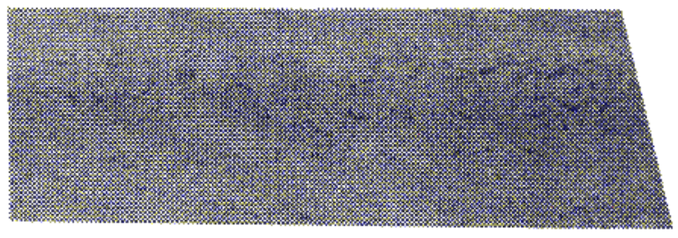
Des sons et... des effets spéciaux

Approchez-vous alors de votre Apple en faisant RUN : vous entendrez un son continu. En ajoutant la ligne 25FOR C=0 TO 3:NEXT, le son sera plus grave. Ceci s'explique par le fait que la boucle vide de cette ligne 25 augmente l'intervalle entre chaque clic du haut-parleur.

Pour provoquer un son, il suffit donc de lire plus ou moins rapidement à cette adresse. Et plus la vitesse de lecture sera grande, plus le son sera aigu.

En langage-machine, on peut obtenir un son plus exact et même faire des effets spéciaux. On utilise généralement la routine WAIT du moniteur (instruction JSR $FCA8) pour provoquer la temporisation entre chaque clic. L'illustration de ces phénomènes est donnée par le programme *Paquet-musique* qui regroupe de petites routines.

Quant à *Apple's Klavier* (page suivante), c'est un petit programme spécial qui scrute le clavier et interprète la touche pressée comme une hauteur



Paquet-musique
Ensemble de routines pour Apple II
Auteur Olivier Gérard
Copyright *LIST* et l'auteur

```
SOURCE FILE: PAQUET.S
0000:                1 ********************************
0000:                2 * PAQUET-MUSIQUE              *
0000:                3 * (C) 1985 O. GERARD ET LIST *
0000:                4 ********************************
0000:                5 ;
----- NEXT OBJECT FILE NAME IS PAQUET
0300:                6          ORG  $300
0300:                7 ;
0300:                8 TWEEP    EQU  *          ;CALL 768
0300:A0 25           9          LDY  £$25       ;DUREE.
0302:A9 02          10 TWP1     LDA  £$02       ;FREQ1=2.
0304:20 A8 FC       11          JSR  $FCA8
0307:AD 30 C0       12          LDA  $C030
030A:A9 20          13          LDA  £$20       ;FREQ2=20.
030C:20 A8 FC       14          JSR  $FCA8
030F:AD 30 C0       15          LDA  $C030
0312:88             16          DEY
0313:D0 ED          17          BNE  TWP1
0315:60             18          RTS
0316:               19 ;
0316:               20 WRONG    EQU  *          ;CALL 790.
0316:A2 02          21          LDX  £$02       ;DEUX NOTES EGALES
0318:A0 05          22 WR1      LDY  £$05
031A:A9 70          23          LDA  £$70       ;DUREE/5
031C:20 A8 FC       24          JSR  $FCA8
031F:A9 20          25 WR2      LDA  £$20
0321:20 A8 FC       26          JSR  $FCA8
0324:AD 30 C0       27          LDA  $C030
0327:A9 38          28          LDA  £$38
0329:20 A8 FC       29          JSR  $FCA8
032C:AD 30 C0       30          LDA  $C030
032F:88             31          DEY
0330:D0 ED          32          BNE  WR2
0332:CA             33          DEX
0333:D0 E3          34          BNE  WR1
0335:60             35          RTS
0336:               36 ;
0336:               37 TRALALA  EQU  *          ;CALL 822.
0336:A2 03          38          LDX  £$03       ;COMPTEUR POUR 3 SOLS.
0338:A0 80          39 SOL0     LDY  £$80       ;DUREE DE LA NOTE.
033A:A9 0D          40 SOL1     LDA  £$0D       ;FREQUENCE.
033C:20 A8 FC       41          JSR  $FCA8
033F:A9 01          42          LDA  £$01       ;CORRECTION.
0341:20 A8 FC       43          JSR  $FCA8
0344:AD 30 C0       44          LDA  $C030
0347:88             45          DEY
0348:D0 F0          46          BNE  SOL1
034A:A9 30          47          LDA  £$30       ;ATTEND UN PEU POUR
034C:20 A8 FC       48          JSR  $FCA8      ;DETACHER LA NOTE.
034F:CA             49          DEX
0350:D0 E6          50          BNE  SOL0
0352:A9 06          51          LDA  £$06       ;FREQ=MI.
0354:20 5C 03       52          JSR  NOTE
0357:A9 07          53          LDA  £$07       ;FREQ=DO.
0359:4C 5C 03       54          JMP  NOTE
035C:               55 ;
035C:48             56 NOTE     PHA             ;SAUVE LA FREQ.
035D:A2 03          57          LDX  £$03
035F:A0 00          58 NOT0     LDY  £$00       ;DUREE/X.
0361:68             59 NOT1     PLA             ;REPREND LA FREQ.
0362:48             60          PHA             ;LA SAUVE POUR + TARD.
0363:20 A8 FC       61          JSR  $FCA8
0366:AD 30 C0       62          LDA  $C030
```

```
0369:88          63          DEY
036A:D0 F5       64          BNE  NOT1
036C:CA          65          DEX
036D:D0 F0       66          BNE  NOT0
036F:68          67          PLA
0370:A9 20       68          LDA  £$20       ;POUR DETACHER
0372:20 A8 FC    69          JSR  $FCA8      ;LA NOTE.
0375:60          70          RTS             ;RETOUR.
0376:            71 ;
0376:            72 ;POUR SAUVER SUR DISQUETTE :
0376:            73 ;FAIRE BSAVE PAQUET,A$300,L$75
0376:            74 ********************************

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Apple's Klavier
Routine pour Apple II
Auteur Olivier Gérard
Copyright *LIST* et l'auteur

```
SOURCE FILE: KLAVIER.S
0000:             1 ********************************
0000:             2 * APPLE'S KLAVIER              *
0000:             3 * (C) 1985 O. GERARD ET LIST *
0000:             4 ********************************
0000:             5 ;
----- NEXT OBJECT FILE NAME IS KLAVIER
0300:             6          ORG  $300
0300:             7 ;
0300:AD 00 C0     8          LDA  $C000      ;PREND LA TOUCHE.
0303:8D 10 C0     9          STA  $C010
0306:20 A8 FC    10          JSR  $FCA8      ;VA ATTENDRE.
0309:2C 30 C0    11          BIT  $C030      ;EQUIVAUT A LIRE.
030C:F0 F2       12          BEQ  $0300      ;A=0 APRES WAIT.
030E:            13 ;
030E:            14 ;POUR SAUVER SUR DISQUETTE :
030E:            15 ;FAIRE BSAVE KLAVIER,A$300,L$0E
030E:            16 ********************************

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

de note. C'est très amusant, surtout en utilisant les codes CTRL – ... Il faut faire CTRL – RESET pour arrêter. On appelle toutes ces routines par une instruction CALL suivie de l'adresse de celle-ci, indiquée sur la liste.

Ces routines peuvent être utilisées sous forme d'un fichier binaire à charger avant de les utiliser (les coordonnées pour une sauvegarde de cette manière sont en bas de la liste). Dans un programme, on a ainsi :

```
104 PRINT CHR$ (4) "BLOAD PAQUET"
2065 IF REP$="2" THEN CALL 790:PRINT "C'EST FAUX"
```

Si vous ne disposez pas d'un assembleur pour générer toutes ces routines, faites :

CALL – 151

* 300:AD 00 C0 8D 10 ... (entrer les codes tels qu'ils apparaissent dans les listes des programmes — ici ceux de *Apple's Klavier* — puis taper RETURN)
* 3D0G

§BSAVE PAQUET,A768,L$76 ou §BSAVE KLAVIER,A768,L$0C (selon la routine).

Et après avoir découvert — avec plaisir ! — les Tweep, Wrong, Tralalaitou et Apple's Klavier, expérimentez donc vos propres sons en faisant appel à vos idées originales !

Olivier GÉRARD

SOLUTIONS PROPOSÉES AU PROBLÈME DE LA PAGE 66

L'édition du Triangle de Pascal en neuf instructions :

```
1 INPUT N : DIM T(N+1) : T(1)=1
2 FOR I=1 TO N+1 : PRINT : FOR J=I TO 1 STEP-1
3 T(J)=T(J)+T(J-1) : PRINT T(J) ; : NEXT J,I
```

L'édition du Triangle en huit instructions :

```
1 INPUT N : FOR I=0 TO N : PRINT
2 A#=1 : FOR J=0 TO I : PRINT A#;
3 A#=A#* (I-J)/(J+1) : NEXT J,I
```

L'édition du Triangle en sept instructions :

```
1 INPUT N : FOR I=0 TO N : PRINT
2 FOR J=0 TO I : PRINT A#-(A#=0) ;
3 A#=(A#-(A#=0))* (I-J)/(J+1) : NEXT J,I
```

Pierre BARNOUIN

PC-1500 + IMPRIMANTE

P COMME PRINT

L'ordinateur Sharp PC-1500 équipé de son imprimante dispose de deux ordres différents d'affichage de données : PRINT et LPRINT, respectivement, sur l'écran et sur l'imprimante. Cette dernière est munie en outre d'un commutateur manuel "P" qui, lorsqu'il est armé, dévie sur l'imprimante tous les messages tapés au clavier.

Un bon programme doit tenir

compte de l'environnement matériel : l'imprimante est-elle connectée ou non ? Mieux, il doit prendre aussi en considération les desiderata de l'opérateur !

Comment, en Basic, déterminer la présence de l'imprimante et la position du commutateur ? Avec PEEK#&B 00E bien sûr ! Si la valeur de cette expression est 33, alors non seulement l'imprimante est branchée mais son commutateur P est en position *print*.

Immédiatement, une application de recopie d'écran :

```
2 : IF PEEK#&B00E < > 33 RE
    TURN
3 : POKE &79AC, &FF, &7B, &60,
    26 : LPRINT V
4 : PAUSE '' '' : RETURN
```

Le second truc employé ici — "Dump ASCII" — est une astuce déjà bien connue des amateurs.

Francis CHIGOT

X-07

INVERSION VIDÉO

Cinq lignes suffisent au Canon X-07 pour effectuer une inversion vidéo (programme 1). Aucune difficulté de programmation n'apparaît, mais une succession de petites astuces. Par exemple, la décomposition d'un caractère et de ses codes à l'aide de la fonction MID$ (ligne 20), ou l'assignation de chaque caractère défini à son homologue normal en tapant à la ligne 30 : A$ = "(caractères graphiques)0123... XYZ". Chaque caractère sera localisé par MID$.

Basé sur le même principe, le programme 2 inverse n'importe quel caractère et le positionne au code ASCII de son choix.

Globalement, ces programmes se distinguent par leur petite taille, leur simplicité et leur relative rapidité. Ils peuvent même être utilisés comme sous-programmes.

Bruno de LA BOISSERIE

Programme 1

```
10 CLS:PRINT"Inversion 0-Z","Patience...
":P=48:Q=90:W=1:FORZ=PTOQ
20 FORA=2TO23STEP3:R(A¥3)=255-VAL("&H"+M
ID$(FONT$(Z),A,2)):NEXTA
30 A$="âΣìîíÜαôθó♡☯éεè♤ñäßÇδë£Ü¢φç êλμàõπ
áρòúüùωÉΩó":T=ASC(MID$(A$,W,1)):W=W+1
40 FONT$(T)="R(0),R(1),R(2),R(3),R(4),R(
5),R(6),R(7)":NEXTZ
50 FONT$(255)="255,255,255,255,255,255,2
55,255":KEY$(6)="noi"+CHR$(255):END
```

Programme 2

```
0 CLS:INPUT"Caractere a inverser";CA$:IN
PUT"Code graphique";X:CC=ASC(CA$)
10 FORA=2TO23STEP3:CL(A¥3)=255-VAL("&H"+
MID$(FONT$(CC),A,2)):NEXT
20 FONT$(X)="CL(0),CL(1),CL(2),CL(3),CL(
4),CL(5),CL(6),CL(7)"
30 PRINTX"--> "CHR$(X)
```