LISTI

LE JOURNAL DES AMATEURS DE PROGRAMMATION

n° 9

ISSN 0761-9936

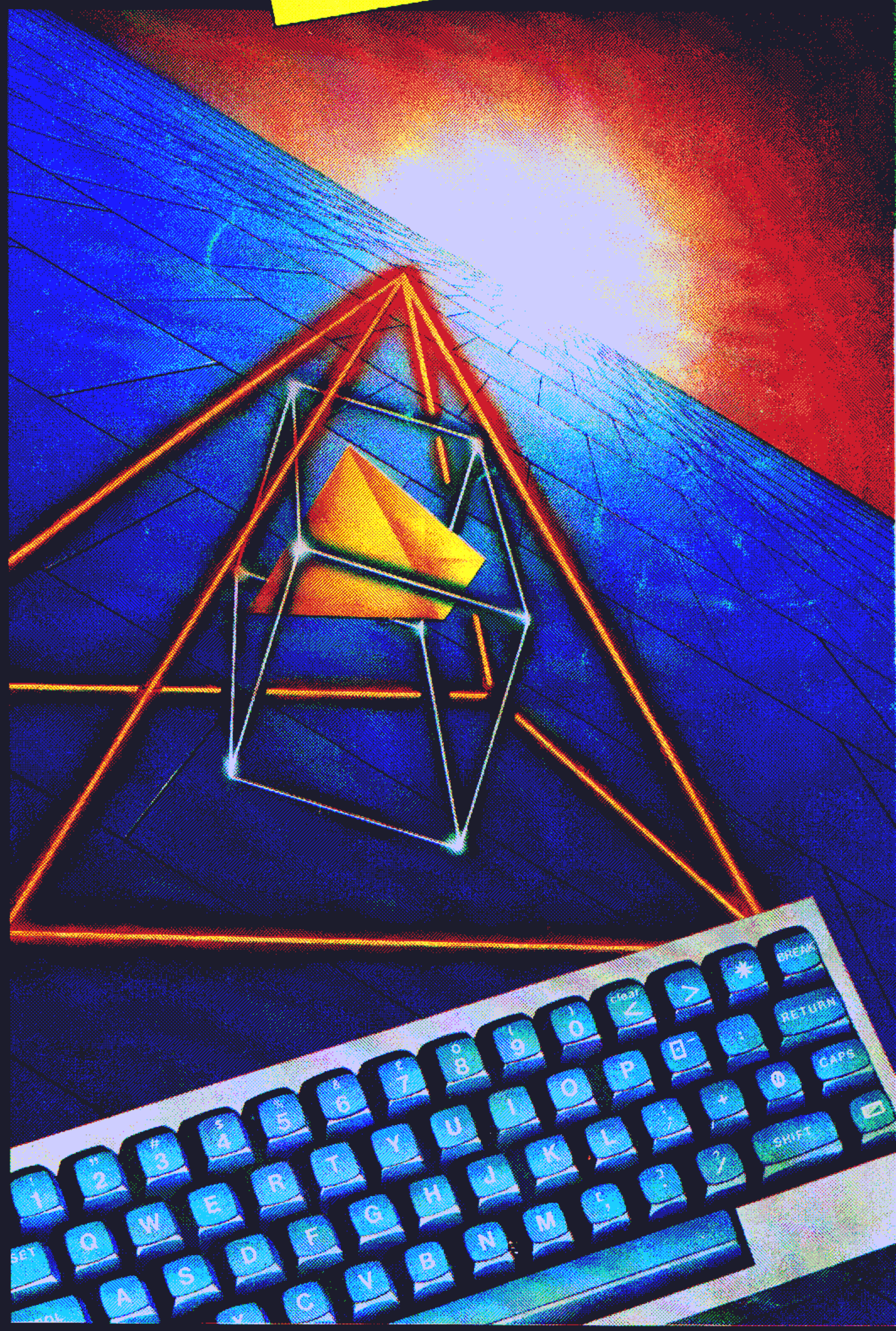
MAI 1985

A L'AFFICHE

GROS PLAN SUR TROIS LOGICIELS

A L'ESSAI

M 2712-9-20 F



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

20 F

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 9 - MAI 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 : *Gérald Anfossi, Michel Aubry, Pierre Barnouin, François J. Bayard, Robin Bois, Christian Boyer, Roger Brousmiche, Laura Campagnet, Jean-Paul Carré, Thierry Chamoret, Frédéric Charles, Raymonde Coudert, Robert Daguesse, Marc Dangla, Jacques Deconchat, Olivier Dufailly, Bruno Fiter, Philippe François, Augustin Garcia, Florence Gautier-Louette, Pierre Ladislas Gedo, Alain Goubault de Brugière, Max Hagenburger, Bernard Kokanosky, Jean-Christophe Krust, Jacques Labidurie, Jean-Pierre Lalevée, Stéphane Lastrajoli, Franck-Olivier Lelaidier, Alain Mariatte, Christophe Masurel, Pierrick Moigneau, Yvon Pérès, Edouard Rencker, Lucien Strebler, André Warusfel.*

Illustrations : *Philippe Burel, Antoine Chereau, Chimulus, Frapar, Bernard Helme, Renée Koch, Fabien Lacaf, Dominique Le Nouaille, Sylvain Lemaire, Alain Mangin, Alain Prigent, Nicolas Spinga.*

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

Limité par Basic ?
Impatient, lorsqu'un compilateur classique prend 5 mn pour compiler 100 lignes ?
TURBO Pascal a été conçu pour vous.

TURBO PASCAL 625 F HT

avec manuel en français

TURBO PASCAL EST UN SYSTÈME COMPLET

Il comprend un éditeur et un un compilateur dans le même programme. (28 K CPM. 36 K sous MS DOS).

Vous passez de l'un à à l'autre par une touche : plus besoin de jongler avec disquettes ou fichiers.

LA COMPILATION SE FAIT EN MÉMOIRE

Un compilateur classique utilise des fichiers intermédiaires sur disque; jusqu'à 90 % du temps peut être consacré aux opérations de lecture/écriture sur disque.

Avec TURBO Pascal, la compilation se fait en mémoire en une seule passe : le temps de compilation est réduit au strict minimum.

Par exemple, Microcalc, programme de démonstration de 1.200 lignes fourni avec TURBO Pascal est compilé en 30 secondes (à 4 Mhz).

Si une erreur survient lors de la compilation, l'emplacement de l'erreur est retrouvé dans le code source et le mode éditeur activé : corriger un point-virgule oublié ne prend que quelques secondes.

L'ÉDITEUR INTÉGRÉ EST CONFIGURABLE.

Vous pouvez redéfinir toutes les commandes. Il lit les programmes écrits avec d'autres traitements de texte.

DE NOMBREUSES EXTENSIONS

vous permettent d'utiliser à fond votre ordinateur.

- *appels aux fonctions du DOS*
- *opérations sur la mémoire, les ports d'entrée/sortie*
- *fonctions AND OR, XOR, sur les entiers*
- *type String avec fonctions de traitement de chaînes*
- *procédures de gestion de l'écran*
- *8 fichiers prédéfinis*
- *modules de recouvrement (overlays) permettant d'écrire de très grands programmes*
- *fichiers à accès aléatoire avec "seek"*
- *constantes structurées permettant d'initialiser rapidement ensembles et tableaux*
- *identificateurs de 127 caractères*
- *programmes chaînés avec partage des données*
- *variables absolues placées à 1 adresse précise en mémoire.*

REVUES...

"Des performances à faire pâlir"
« LIST », Nov. 1984

"TURBO Pascal offre tout ce qu'un utilisateur de Pascal peut attendre en dépassant même largement ses espérances".
« ORDI », Nov. 1984

"The best cost less"
« Creative Computing », juillet 1984

"TURBO Pascal appears to violate the laws of thermodynamics".
« SOFTALK », Mars 1984

"This dynamic new language compiler is a "VolksPascal", with most of Pascal plus a few extras. It introduces a new programming environment and runs like magic".
« PC MAGAZINE », Nov. 1984

***TURBO 87** permet d'utiliser le coprocesseur 8087 sur les machines 16 bits.*

Remboursement possible par retour de la disquette non ouverte sous 15 jours.

ENVOYEZ-MOI DE SUITE :

- ☐ **TURBO PASCAL 625 F + 116,25 F TVA**
- ☐ **TURBO 87 1.150 F + 213,90 F TVA**
- ☐ **JE PRÉFÈRE LE MANUEL ANGLAIS**
- ☐ **ORDINATEUR :** _____

DISQUES ☐ **3 1/2"** ☐ **51/4** ☐ **8"**

SYSTÈME D'EXPLOITATION :
☐ **MS-DOS** ☐ **CMP-80**
☐ **PC-DOS** **CPM 86**

RÈGLEMENT :
☐ *CHÈQUE JOINT*
☐ *CONTRE-REMBT. (+ 50 F)*

- Remboursement garanti si renvoi de la pochette non ouverte sous 15 jours.
- Remplissez soigneusement le bon de commande.

NOM _____

ADRESSE _____

_____

_____

TÉL. _____

SIGNATURE :

FRACIEL (47) 64.08.52
42, rue des Prébendes
37000 TOURS

A VOS CLAVIERS

ET UNE RUSTINE POUR L'AMSTRAD-GRAPHE, UNE !

UNE erreur s'est glissée dans le programme pour Amstrad (Picologo) que nous avons publié dans *LIST* 8, page 67. Rien de très grave, mais essayez de nouveau le programme en modifiant la ligne 500 qui doit être :

500 FOR I = NC + 1 TO 99 : C$(I) = "" : NEXT I

C'est mieux ainsi, non ? Avec toutes nos excuses. Bogue avouée est à demi pardonnée.

DANS LE COUP ?

ENFIN *LIST* découvre l'existence de Basic structurés sur les micros pour amateurs ! J'en arrivais à me demander si cette sympathique revue était bien, comme on dit, « dans le coup » !

La lecture d'un certain article du numéro 4 (page 56), dont le sommaire disait : « Récursivité, le Basic y vient... » (et mieux, sur du matériel français), m'avait en particulier laissé perplexe.

C'est donc avec grand plaisir que je viens de lire l'exposé de Bernard Kokanosky que je félicite pour sa clarté.

Puis-je me permettre de dire la satisfaction de l'amateur que je suis, rebuté par Pascal, à l'usage d'un Basic réellement structuré ?

Après avoir débuté en 1978 avec un PET, et continué avec plusieurs autres matériels j'ai depuis près d'un an un Electron.

Ce n'est certes pas la machine sans défauts, son principal étant d'être très mal commercialisée, mais ceci est une autre histoire.

Son Basic est celui du BBC, dernière version, et je peux vous assurer — maintenant je le possède bien — qu'il est vraiment de qualité, et très rapide.

L'Electron est par ailleurs une machine faite pour l'amateur de programmation pure ; la liste des possibilités offertes est considérable. A titre d'exemple, connaissez-vous beaucoup de micros de la même classe qui réservent une zone de la page zéro à l'utilisateur ?

Et des variables entières qui le sont réellement et non reconverties en réelles ? Quand en plus elles sont sur 32 bits, et pour une part résidentes, alors là...

C'est pourquoi d'ailleurs les tests de vitesse que vous publiez m'agacent un peu — pardon —, mais ils ne tiennent pas compte des particularités propres à chaque machine. Ainsi, j'écris le test n° 1 :

```
10 FOR I % = 1 TO 10000
20 NEXT I %
```

Résultat : 3,3 s ! Au lieu de 9 avec un indice réel.

M'étant défoulé, j'arrête là, en vous disant, maintenant que vous êtes sur la bonne voie, continuez... si vous ne craignez de faire trop de peine à certains.

Salutions cordiales d'un vieux et fidèle lecteur.

Jean RENAUD
78 Marly le Roi

Le Basic de l'Electron est en effet le même que celui de son frère le BBC, deux ordinateurs conçus par le constructeur britannique Acorn. Dans le numéro 3 (page 30) de LIST, *nous avons d'ailleurs testé le Basic du BBC qui nous a paru « étonnant : à la fois riche et rapide ».*

Les dix tests de LIST ont pour but de « situer » la vitesse d'un ordinateur par rapport à celle des autres. Il ne s'agit pas ici de comparer les apports spécifiques de chaque machine, de chaque Basic. Nous avons choisi de sérier les questions, pour éviter la confusion.

Nous avons commencé par les tests de vitesse, les caractéristiques du Basic viendront bientôt. Encore un peu de patience, merci.

Quant au Basic structuré de Bernard Kokanosky, vous le retrouverez dans ce numéro, à la page 60 (1).

(1) Bernard Kokanosky est l'auteur d'un Basic structuré pour Sharp MZ 700, que l'on peut se procurer au Club des Sharpentiers, 151-153 avenue Jean Jaurès, 93307 Aubervilliers Cedex.

POUR ÊTRE AUTEUR DE LIST

PERMETTEZ-moi de vous féliciter pour votre magazine que je trouve si intéressant. Je souhaiterais vous poser une petite question : achetez-vous les programmes d'amateurs, qui ne sont pas recopiés et qui sont tout à fait personnels ? Et si oui, comment est-il possible d'écrire l'article ?

Merci d'avance pour votre réponse.

Frédéric HARLE
62 Lapuenoy

JE tiens à vous remercier, par la présente, de l'intérêt que vous avez porté à mon article « Top secret » pour PB-700 et que vous avez fait paraître dans *LIST* 6, page 75. Je dois avouer que j'étais extrêmement content de voir mon article porté par un journal aussi remarquable que le vôtre.

Aussi, imaginez ma joie lorsque, aujourd'hui, je reçois en plus un chèque... Autant vous dire que je ne m'y attendais pas.

Longue vie à *LIST* !

Pierrick GRESLIN
95 Argenteuil

DANS le but de pouvoir répondre aux adhérents d'un club dont je m'occupe, je souhaiterais avoir les précisions suivantes :

- sur quels critères se fait la publication d'astuces dans la *Boîte à malices* ?
- donne-t-elle lieu à une rémunération ?
- quelle est la différence entre les programmes publiés dans la *Boîte à malices* et les autres ? (Certains trucs de la Boîte à malices sont supérieurs aux programmes hors boîte).
- est-il possible à « Monsieur tout-le-monde » d'écrire un vrai article ? En d'autres termes, faut-il des « références » particulières pour être publié dans *LIST* (si l'article en vaut la peine, bien sûr).

Bruno de LA BOISSERIE
27 Le Vaudreuil

Bien sûr, « Monsieur tout-le-monde » (« Madame tout-le-monde », aussi), amateur ou professionnel, sans référence aucune (et même s'il en a !), jeune ou moins jeune, peut nous proposer programmes, astuces, ou toute autre idée qui touche à la programmation.

Nous étudions tout ce qui nous est envoyé. Si nous jugeons le sujet intéressant, nous en envisageons la publication. Si le texte de l'article mérite une remise en forme, nous le « révisons ».

Ce qui est spécifique à une machine et ce qui est court, entrera dans la Boîte à malices. Ce qui est complet et général fera le plus souvent l'objet d'un article.

Il faut tout de même rappeler que nous « croulons » sous les biorythmes, les pendus ou les Master Mind. Non, ils ne nous intéressent pas. Par contre, toute idée originale de programme ou d'aide à la programmation, toute astuce inédite spécifique ou non à un matériel, est susceptible de nous intéresser.

Enfin, bien entendu, tout article publié est rétribué. S'il ne s'agit pas d'un pont d'or, c'est tout de même loin d'être symbolique.

INDEX DES ANNONCEURS

A VOS CLAVIERS

DE MEILLEURES SOLUTIONS POUR LE JEU 17

RÉDUIRE au minimum le temps d'initiation d'une matrice carrée, c'était le but du jeu 17 (*LIST* 6, page 80). Il a suscité beaucoup de courrier et l'étude des différents programmes proposés montre qu'il était encore possible de réduire — et même considérablement — le temps d'exécution de notre solution (*LIST* 7, page 82).

Une matrice unité est un tableau carré à deux dimensions composé de un dans la diagonale principale et de zéros ailleurs :

$$\begin{bmatrix} 10 \ldots 00 \\ 01 \ldots 00 \\ .. \ldots .. \\ 00 \ldots 10 \\ 00 \ldots 01 \end{bmatrix}$$

Un tel tableau peut être initialisé par le programme 1, où MAX mémorise la dimension de la matrice, c'est-à-dire le nombre de ses lignes ou de ses colonnes. Ici, pour les estimations de temps d'exécution, nous prendrons MAX égal à 100.

Programme 1 :

```
10 for L = 1 to MAX
20    for C = 1 to MAX
30       if L = C then A (L, C)=1 else A(L, C)=0
40    next C
50 next L
```

Le temps d'exécution de ce programme est fonction des temps élémentaires de chaque instruction. Ceux-ci étaient donnés dans l'énoncé :

- 850 microsecondes par itération d'une boucle for...next ;
- 1130 microsecondes pour l'instruction d'affectation d'une valeur à un élément du tableau (comme A (L,C) = 0 ou A(L,C)=1) ;
- 810 microsecondes pour le test d'égalité.

Ainsi, avec l'algorithme élémentaire du programme 1, on obtient un temps de 27,985 secondes. Cet algorithme peut être modifié et donner le programme 2, programme proposé comme solution du jeu (*LIST* 7).

Programme 2 :

```
10 for L=1 to MAX
20    for C=1 to MAX
30       A(L,C)=0
40    next C
50    A (L,L)= 1
60 next L
```

Le programme 2 s'exécute en 19,998 secondes, ce qui représente déjà un résultat appréciable. A ce niveau, notre programme est assez simple, puisqu'il n'utilise que deux types d'instructions : la boucle et l'affectation. Pour réduire encore son temps de calcul, il est possible de s'attacher à la boucle la plus consommatrice de temps, c'est-à-dire celle qui est la plus interne. Nous pouvons donc initialiser deux éléments en même temps. Le programme 3, suggéré par Philippe Matet (Liancourt) et Jean-Jacques Poujade (Paris), le fait.

Programme 3 :

```
10 for L=1 to MAX
20    for C=1 to MAX step 2
30       A(L,C)=0
40       A(L,C+1)=0
50    next C
60    A(L,MAX)=0
70    A(L,L)=1
80 next L
```

Le temps d'exécution d'un tel programme s'établit à 15,748 secondes avec nos données. Mais nous pouvons remarquer que, bien que cet algorithme soit plus rapide que les précédents, il effectue des affectations en double. Par exemple, lorsque L est égal à C, il réalise trois affectations à A(L,L) : deux à zéro, et une à un. Il est possible d'éliminer ces cas en modifiant les bornes des boucles. Nous obtenons donc le programme 4, dû à Eric Levenez (Mantes la Jolie).

Programme 4 :

```
10 A(MAX, MAX)=1
20 for L=1 to MAX-1
30    for C=L+1 to MAX
40       A(L,C)=0
50       A(C,L)=0
60    next C
70    A(L,L)=1
80 next L
```

Ainsi, par élimination des affectations inutiles, le temps est encore réduit. Il est maintenant égal à un peu moins de 15,592 secondes.

Toutefois, nous pouvons encore tirer parti de la remarque de symétrie qui nous a permis d'obtenir cet algorithme. En effet, nous n'avons considéré que la symétrie par rapport à la diagonale principale. Il est possible d'envisager des symétries supplémentaires. En considérant par exemple les symétries par rapport aux deux diagonales de la matrice, nous obtenons un nouveau programme, envoyé par Philippe Bérenger (Wizernes).

Programmes 5 :

```
10 for L=1 to int((MAX-1-(int(MAX/2)=MAX/2))/2)
20    for C=L+1 to int((MAX+
      (int(MAX/2) < > MAX/2))/2)
30       A(L,C)=0
31       A(L,MAX-C+1)=0
32       A(C,MAX-L+1)=0
33       A(MAX-C+1,MAX-L+1)=0
34       A(MAX-L+1,MAX-C+1)=0
35       A(MAX-L+1,C)=0
36       A(MAX-C+1,L)=0
37       A(C,L)=0
40    next C
50 next L
60 for L=1 to MAX
70    A(L,MAX-L+1)=0
71    A(L,L)=1
80 next L
```

Si nous ne comptons pas les opérations nécessaires au calcul des indices du tableau, il apparaît que cet algorithme est encore plus rapide que le précédent. En effet, il s'exécute en 12,468 secondes.

Bien entendu, il est possible de concevoir des algorithmes plus rapides. Mais dans ce cas, la taille occupée par le programme d'initialisation devient importante. Par exemple, en dépliant complètement les boucles, jusqu'à les faire disparaître comme le propose encore Eric Levenez :

```
    1 A(1,1)=1
    2 A(2,1)=0
    3 A(3,1)=0
  ... .......
 9999 A(99,100)=0
10000 A(100,100)=1
```

Nous avons bien entendu gagné un temps appréciable, puisque ce programme s'exécute en 11,3 secondes. Mais avouons qu'il est assez long à écrire !

Peut-on faire plus rapide encore ? Certainement, en tirant parti des facilités du langage de programmation utilisé. Ainsi, en Basic, Pierre Barnouin (Cabris) propose la séquence d'instructions suivante :

```
10 erase A
20 for L=1 to MAX
30    A(L,L)=1
40 next L
```

Comme ERASE s'exécute très rapidement, un tel programme ne mettra guère plus de 0,198 seconde à initialiser la matrice unité. L'instruction est peut-être spécifique, mais le résultat atteint est tellement bon.

LA GAZETTE DE LIST

Du nouveau pour le Dragon 32

LE Dragon 64 avec lecteur de disquette est doté d'un système d'exploitation OS9 qui permet l'accès à divers outils logiciels : traitement de texte, Pascal, langage C, etc.

Une transformation interne et une disquette OS9 rendent ce système d'exploitation disponible sur Dragon 32 avec lecteur de disquette. **Goal Computer** commercialise tout ce qui est nécessaire pour cette modification au prix de 1 150 FF.

Goal Computer
15 rue de Saint Quentin
75010 Paris
Tél. 200 57 71 ■

Le départ d'un président

LE président et fondateur du *Centre Mondial Informatique et Ressource Humaine*, Jean-Jacques Servan-Schreiber, a confirmé officiellement, dans une lettre adressée au Président de la République, son intention de ne pas « conserver la responsabilité du Centre ». Il ne demande donc pas le renouvellement de son mandat et doit être remplacé par Jean-Louis Funck-Brentano, haut conseiller à l'*Agence de l'informatique* et expert en informatique médicale. ■

IBM ne veut plus faire joujou

IBM a décidé d'arrêter la production du PC Junior, tout en continuant à fabriquer des cartouches de programmes et des pièces détachées. Le PC Junior aurait été vendu à 270 000 exemplaires en 1984, selon Future Computing. La France n'y aura jamais goûté. ■

Basic de l'EXL 100

LA première notice du Basic de l'EXL 100 a été corrigée et complétée. Elle est envoyée gratuitement sur simple demande à :

Exelvision
Place Joseph Bermond
06560 Valbonne ■

TROIS CASSETTES

Trois utilitaires pour Oric-1 et Atmos
Kit écran
Data save (Oric-1 seulement)
Supercopy écran
Édités par ARG Informatique
Prix de chaque cassette : 120 FF

TROIS programmes pour l'Oric, trois programmes du genre « boîte-à-outils ». Chacun d'entre eux peut être chargé à tout moment qu'il y ait ou non un programme Basic en mémoire. On dispose alors de fonctions supplémentaires.

Ainsi, **Kit écran** permet d'animer un titre ou la présentation d'un programme en faisant défiler dans les quatre directions un texte géant ou encadré. Il permet de plus d'afficher en permanence une horloge sur l'écran, mais il oblige à utiliser une série de POKEs. A titre d'exemple, un texte en mouvement demande cinq POKEs et un CALL, une page complète avec horloge trente POKEs et six CALLs...

Data save apporte de précieuses fonctions de gestion des cassettes qui manquent sur l'Oric-1. Il ajoute sept nouvelles commandes de sauvegarde, de lecture et de vérification d'un programme, d'un fichier ou d'un écran sur magnétophone. La syntaxe est simple : !C comme Catalog recherche le nom d'un programme enregistré ; !V "PROG" vérifie qu'une sauvegarde s'est effectuée correctement ; !S"PROG" ou !L "PROG" sauve ou lit un programme et l'ensemble de ses données.

Le dernier utilitaire intéresse les possesseurs d'une imprimante Seikosha GP100A (ou similaire). **Supercopy écran** reproduit un écran sur papier dans un temps honorable qui ne dépend que de l'imprimante. Il permet huit modes de copie : TEXT ou HIRES, normal ou inverse, simple ou double taille.

Si **Data save** est destiné au seul Oric-1, les deux autres cassettes conviennent également à l'Atmos.

MH ■

Tout en Forth

A Sens, **Corbou Informatique**, organisme de formation, service et conseil s'est fait une spécialité dans la distribution de produits Forth, essentiellement des systèmes à implanter sur C.64, Apple, QL (prochainement Amstrad), ordinateurs sous CP/M ou MS-Dos, etc., mais aussi livres de tous niveaux.

Les mordus de Forth pourront se procurer le catalogue en écrivant ou en téléphonant à :

Corbou Informatique
12 rue des Vieilles Étuves
89100 Sens
Tél. (86) 65 78 99 ■

Machine à écrire ou terminal d'ordinateur : la Brother EP-44

UNE campagne de pub drôle, certes, mais un peu envahissante, a voulu faire du produit un événement. Rien de tel que de tester pour s'en faire une idée.

La Brother EP-44 utilise le procédé d'impression thermique ou par transfert thermique, c'est-à-dire que la matrice chauffe soit un papier spécial qui noircit à la chaleur (ne pas laisser tomber sa cigarette ni poser sa tasse de thé sur la réserve...), soit un ruban qui dépose l'encre sur un papier normal ou presque (il ne doit présenter aucun relief).

Sur un clavier AZERTY à touches non standard, mais pas désagréables, une profusion d'inscriptions faciles à distinguer grâce aux codes de couleur, et trois inverseurs : l'un pour l'interligne, un autre pour les divers modes d'impression (simultanée/différée) ou de calcul, le troisième enfin, marqué NORMAL/TERMINAL.

En mode normal, une machine à écrire étonnamment silencieuse, et des caractères matriciels étonnamment soignés. Il est vrai que la matrice est de 24 × 18, ce qui est loin des normes courantes sur les imprimantes à aiguilles classiques. Un afficheur à cristaux liquides, chose

Sicob de printemps

LE Spécial Sicob, ou Sicob de printemps, se tient cette année au Palais du CNIT à La Défense (Paris), du 6 au 11 mai 1985. Il est ouvert à tous les publics les 8 et 11 mai (prix d'entrée : 20 FF), et est spécialement tourné vers la micro-informatique, matériels comme logiciels. Les 6, 7, 9 et 10 mai sont les journées professionnelles (prix d'entrée : 60 FF).

LA GAZETTE DE LIST

désormais courante sur les machines à écrire électroniques, permet de s'arrêter en cours de frappe et d'utiliser les fonctions de la calculatrice intégrée. Il est censé en outre offrir bien d'autres facilités, en particulier la frappe différée, après corrections éventuelles, dans les limites du tampon (15 caractères), mais aussi la mise en mémoire de pages de texte (jusqu'à 3726 caractères) où l'on pourra insérer ou détruire des lignes.

Les fonctions de centrage, de soulignement automatique, d'indice inférieur et supérieur, de justification à droite (pas à la fois à droite et à gauche) sont également présentes. L'ensemble est donc séduisant. De là à vouloir en faire une machine à traitement de texte, avec les 15 caractères de l'écran à cristaux liquides, il ne faut rien exagérer.

En mode terminal, on a non seulement le terminal, mais aussi le programme de communication. Car c'est là une des propriétés les plus curieuses de la machine : vous pouvez la connecter directement à un modem pour dialoguer sans ordinateur avec un serveur. Il suffit de choisir la vitesse de transmission, la longueur des mots (7 ou 8 bits), la parité, le retour chariot avec ou sans saut de ligne, le code de caractères (7 ou 8 bits), etc. Transpac n'est pas loin, les réseaux sont au bout de la ligne, et si l'on n'était pas obligé d'interrompre régulièrement la transmission pour changer de feuille, ce serait parfait. Heureusement, c'est prévu : un message PAPER EMPTY apparaît sur l'afficheur et la communication est suspendue, puis reprend sans caractères perdus.

Enfin, le mode terminal permet d'utiliser la Brother EP-44 avec un ordinateur muni de l'interface RS-232C. L'un des deux manuels (en français) fournit les réglages pour un bon nombre d'ordinateurs courants, c'est un bon point. La possibilité de choisir le code de caractères sur 7 ou 8 bits ne limite pas cette possibilité aux seuls ordinateurs qui respectent scrupuleusement le code ASCII, mais un essai chez le vendeur éventuel s'impose avant de casser définitivement sa tirelire. Évidemment, il n'y a pas de possibilité graphique, ce qui est normal, mais on peut cependant sortir des listes avec un programme de décodage des caractères spéciaux.

L'impression thermique permet donc d'obtenir une qualité d'écriture plus qu'acceptable avec une mise de fonds restreinte : 2395 FF ttc. Restreinte quant au prix d'achat de la machine, car pour ce qui est des crédits de fonctionnement, sachant que cinq rubans sont vendus 130,60 FF et qu'un paquet de 100 feuilles de papier spécial coûte 59,50 FF, vous aurez à surveiller vos comptes de près.

FJB ■

UN LIVRE

DU LOGO POUR APPLE

Du Logo pour Apple
Nicole Bréaud-Pouliquen
Éditions du PSI
Lagny, 1984
Broché, 118 pages
Prix : 85 FF

DU Logo pour Apple est, comme son nom l'indique, un livre sur Logo spécialisé Apple. Il ne se perd pas dans des explications pédago-informatico... plus ou moins bien énoncées. Cette spécialisation entraîne des exemples concrets, directement exécutables sur la machine. J'aurais cependant quelques remarques à faire. L'auteur semble bien connaître Basic et LSE. Elle vise très haut dès le départ en supposant chez le lecteur une connaissance des processus informatiques. Pratiquer Logo n'est pas évident lorsque l'on aborde les listes et la récursion.

Une remarque « matérielle » : les crochets [ ] n'existent pas sur toutes les imprimantes et c'est sans doute la raison pour laquelle ils n'apparaissent pas dans les listes de programme et sont systématiquement remplacés par □ pour le crochet ouvrant, et par § pour le crochet fermant. Ceci est dit au début du livre (page 11) mais il ne faut pas l'oublier sous peine d'erreurs fréquentes.

D'une manière générale, la présentation du livre est claire et le jeu de caractères utilisé permet de mettre en évidence les éléments importants. Des exercices retiennent l'attention.

Les programmes publiés sont de véritables listes (clichées) ce qui évite les erreurs de frappe, surtout sur les espaces ou les caractères mal interprétés. En outre, ces programmes sont véritablement opérationnels et non approximatifs.

Ce livre peut donc devenir le « livre de chevet » de ceux qui font du Logo sur Apple. On peut à tout moment y trouver des idées ou des exemples de programmes qui tournent. Pour être un guide vraiment pratique, il ne lui manque plus qu'à être accompagné d'une disquette des programmes proposés. Ce livre n'appartient-il pas en effet à la collection Guide Pratique ? Une suggestion à l'éditeur.

RD ■

L'EXL s'entoure de périphériques

UN clavier professionnel, un modem, un décodeur Antiope, un double lecteur de disquettes et beaucoup de mémoire vive, **Exelvision** ne viendra pas aux *Sicob* (Spécial Sicob et Sicob de septembre) les mains vides. La société a décidé en effet de « gonfler » l'EXL 100 en présentant de nouveaux périphériques. De quoi, précise Exelvision, offrir aux consommateurs un système modulable compris entre 3 000 et 10 000 F.

Première innovation, de taille, une unité de mémoire de masse comprenant un double lecteur de microdisquettes, double face, double densité. Le système se présente sous la forme d'une unité de disquette de base (3500 F) et d'une unité additionnelle (environ 1500 F). L'originalité de l'unité de base est qu'elle apporte 40 Ko de mémoire vive, un traitement de texte et un tableur intégrés, un connecteur pour cartouche de 64 Ko de mémoire morte. Une fois formatée, chaque disquette a une capacité de 640 Ko (elle en a 1000 au départ).

En supplément, Exelvision annonce des modules de mémoire, Exelmémoire, de 16 Ko autonomes, alimentés par une pile au lithium, qui fonctionnent comme des disques virtuels. Ils coûteront 590 F chacun.

Autre innovation, un clavier professionnel mécanique, plus

spacieux que son prédécesseur, également à infrarouge, il devrait s'avéret plus fiable. Pour convaincre les éventuels sceptiques, Exelvision n'a d'ailleurs pas hésité à garantir une capacité de « 15 millions de manœuvres ». Son prix : 295 F.

Côté communication, Exelvision propose un modem (en cours d'homologation) et un décodeur Antiope. Pourvu d'une fonction « numérotation automatique » et d'un détecteur de sonnerie, le modem devrait permettre en outre de transformer l'EXL en véritable télex avec possibilité pour un correspondant de laisser un message sur l'appareil. Son prix, 1 090 F, comprend le modem muni d'une incrustation vidéo, d'une interface parallèle et d'une interface série.

Le décodeur Antiope (disponible fin septembre pour environ 1 000 F) permettra quant à lui de visualiser des magazines radiodiffusés ainsi que les sous-titrages de films télévisés, à l'intention des mal-entendants. On pourra dès lors bénéficier de certains bulletins d'information transmis par le réseau Antiope et utiliser l'incrustation d'images vidéo au sein d'un programme quelconque. Quant à savoir ce que diffuse Antiope ? Une bonne occasion de le découvrir.

Enfin, pour finir avec panache, Exelvision a également présenté deux versions de moniteur : monochrome et couleur. L'ordinateur livré avec l'écran monochrome coûtera 3 290 F. Pour avoir l'écran couleur, il faudra ajouter environ 1 500 F.

Une remarquable boîte à rythmes programmable, baptisée Exeldrums, a été développée en collaboration avec Hohner Electronique (un gage de « savoir-faire ») et offre haute fidélité, enregistrement numérique, affichage, réglage du temps et seize rythmes préprogrammés. Dix-sept instruments de percussion différents sont à la disposition du mélomane. Le tout, bien entendu, peut être sauvegardé sur cassette ou disquette. Du rythme pour 1 100 F.

Dès la fin septembre, il sera donc possible de disposer d'une configuration maximale du système. Pour l'instant, on peut la voir au Spécial Sicob (du 6 au 11 mai 1985, au CNIT-La Défense). ■

Kit de nettoyage pour disquettes et lecteurs

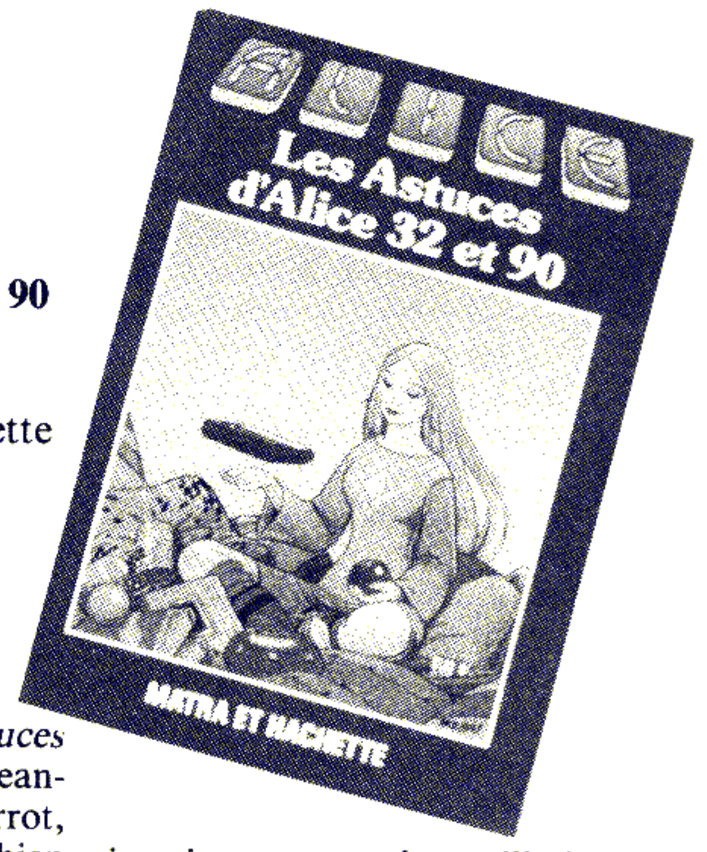
UN petit clic vaut mieux qu'un grand scratch. C'est à peu près l'argument de *Compuclean* qui propose un « kit de nettoyage » complet pour microdisquettes, 3,5 pouces et lecteurs correspondants.

Un clic pour ouvrir le sachet de solvant, un coup de pouce pour dégager la glissière métallique de la disquette en question, ne pas oublier d'humidifier la surface nettoyante, glisser le tout dans le lecteur et le tour est joué.

Au bout de 30 secondes, on obtient un nettoyage parfait (non abrasif). Une précision pour les clients pointilleux : le solvant est un habile dosage de fluocarbone et d'alcool isapropylique ! ■

UN LIVRE

Les astuces d'Alice 32 et 90
Jean-François Gallet
et Alain Pierrot
Edité par Matra et Hachette
Paris, 1985
Broché, 182 pages
Prix : 98 FF



LES auteurs des *Astuces d'Alice 32 et 90*, Jean-François Gallet et Alain Pierrot, étaient particulièrement bien placés pour écrire cet ouvrage, puisqu'ils ont fait partie de l'équipe qui a mis au point l'éditeur-assembleur d'Alice, chez Matra.

Et, de fait, le livre est remarquable. Contrairement à ce qui se produit trop souvent dans les livres traitant d'Assembleur, les auteurs ne se contentent pas de passer en revue la liste des instructions du microprocesseur 6803 (ils renvoient pour cela à une documentation fournie en annexe), mais proposent des outils, largement expliqués et commentés, qui permettront de réaliser des programmes d'un bon niveau.

A la fin du volume, on trouve un moniteur, qui facilitera la mise au point des programmes en Assembleur. Pour travailler plus efficacement, il sera préférable de le frapper dès le début (même si le lecteur débutant ne le comprend pas encore).

Tout est ici examiné en détail : réaffectation des touches, gestion du clavier, sortie écran ou imprimante, et des utilitaires (Renum, Merge, horloge temps réel, liaison Alice-imprimante, liaison Alice-Alice) viennent compléter le Basic d'origine, somme toute un peu décevant. Le processeur de visualisation (EF 9345) se voit consacré une étude assez poussée et certaines possibilités mal ou non gérées par le Basic sont passées en revue : jeu complet de caractères, caractères agrandis, couleurs, etc.

Dans les dernières pages, sont fournis les schémas complets des entrées-sorties série et cassette, les adresses mémoire utiles, ainsi qu'un programme qui permettra (si l'on retourne le câble Péritel) l'affichage de lignes avec incrustation.

Voici donc un ouvrage intéressant qui sera particulièrement utile aux possesseurs d'Alice (32 et 90). Il est dommage qu'il ait fallu attendre si longtemps pour que les possibilités de cet ordinateur (l'incrustation, entre autres), soient révélées.

JD ■

LA GAZETTE DE LIST

L'Apple IIe et le temps qui passe

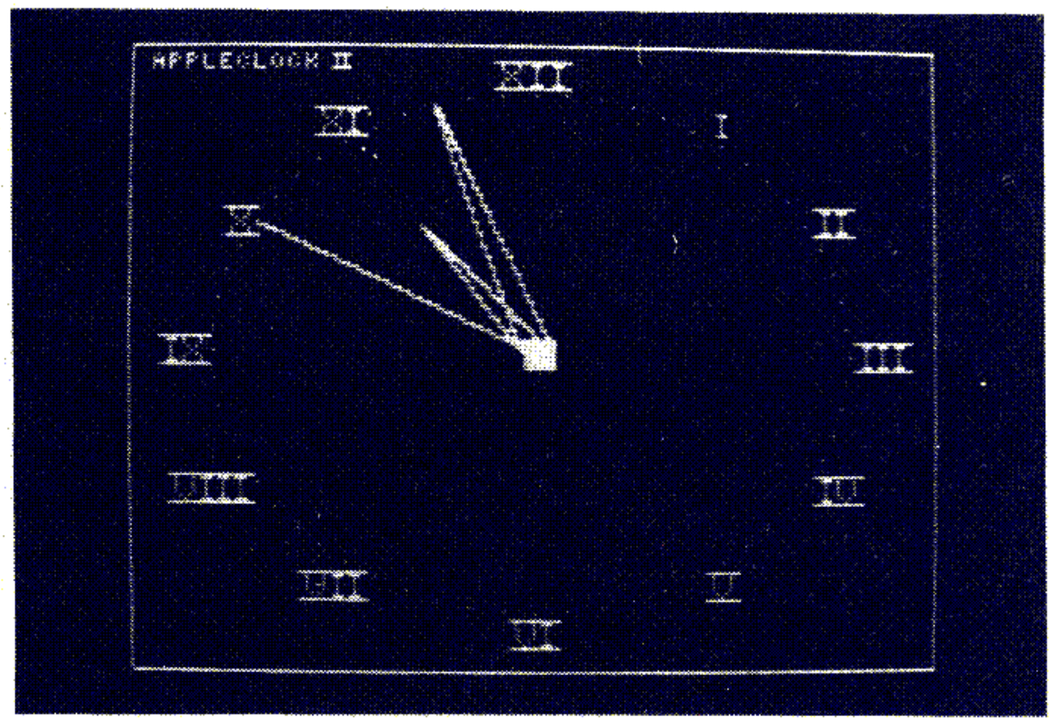
SEULS, dans le passé, les systèmes d'exploitation des ordinateurs autres que les micros, IBM, PDP-11, VAX, etc. gardaient trace dans leurs « directory » des dates et heures de création et de mise à jour des fichiers qui leur étaient confiés. Depuis quelque temps déjà les micros, et notamment les IBM-PC et Apple, ont aussi, grâce à l'adjonction de cartes horloge spécifiques, cette possibilité.

Proclock, de Practical Peripherals (distribué par Alpha Systèmes), est la dernière-née de ces cartes horloge pour l'Apple IIe. Elle permet de garder trace des dates de création/mise à jour des fichiers sous ProDOS, le nouveau système d'exploitation des Apple II, mais aussi sous DOS 3.3 (Proclock propose un très joli « patch ») et sous CP/M.

ProDOS, outre sa rapidité accrue (8 Ko transmis par seconde) et la possibilité qu'il offre de gérer des volumes importants comme des disques durs (type Profile), tient compte de la date et de l'heure. Il reconnaît la présence d'une carte horloge et lit les informations que lui délivre celle-ci. De plus, les derniers logiciels d'Apple comme *AppleWorks* ou *ProDOS Assembler Tools* récupèrent les informations fournies par cette carte horloge et permettent des listages très documentés gardant trace des dates de création ou/et de mise à jour.

Outre un manuel très bien fait (mais en anglais !), Proclock est accompagnée d'une disquette double face. Une des faces contient de nombreux programmes en Applesoft, en langage-machine et en Basic Microsoft sous CP/M (les programmes se présentant alors sous forme de fichiers « textes » au format DOS 3.3 qu'il sera nécessaire de « passer » au système CP/M en utilisant pour cela l'utilitaire « APDOS »). La deuxième face contient des programmes Pascal qui permettront aux amateurs de ce langage d'utiliser Proclock au mieux de ses possibilités.

Côté Applesoft, les programmes offerts sont très variés et comprennent des simulations (une horloge en graphisme haute résolution), des exemples de lecture de temps en différents formats, de décomptage de temps, permettant ainsi la réalisation de jeux de rôle ou d'arcade en temps réel. Enfin un programme gérant les interruptions (avec un bel exemple : l'affichage de la date et de l'heure, secondes comprises, sur la première ligne de l'écran pendant qu'un autre programme se déroule).



Proclock offre également une version modifiée du DOS 3.3 permettant de garder trace de la dernière sauvegarde d'un fichier. Les fichiers (ou programmes) peuvent être néanmoins utilisés avec un DOS normal.

Concernant CP/M, deux programmes illustrent la façon d'utiliser Proclock (mais seulement en émulation « Superclock ») en Basic Microsoft.

L'autre face de la disquette est réservée à des programmes Pascal ainsi qu'à une nouvelle « library » comportant des fonctions pré-compilées permettant la lecture et l'affichage du temps. Proclock fournit aussi un nouveau « system startup », l'équivalent Pascalien du « Hello » Basic, qui met à jour la date et l'inscrit sur la disquette (évitant ainsi la frappe des multiples commandes réalisant normalement cette opération).

Malheureusement ce programme comporte deux petites erreurs :

• Pascal ne garde pas trace de l'heure mais seulement du jour, aussi restreindra-t-on l'affichage à la date seulement ;

• le *startup* de Proclock met à jour la date (mais seulement sur la disquette et non pas en mémoire). On peut évidemment corriger le programme.

L'électronique est extrêmement soignée (*Practical Peripherals* offre d'ailleurs une garantie de 5 ans, ce qui est assez exceptionnel, la garantie sur ce genre de matériel se chiffrant plutôt en mois).

Proclock dispose d'une série de commutateurs et peut simuler plusieurs autres cartes horloge (Apple Clock, SuperClock II et Thunderclock Plus).

De plus Proclock fonctionne avec une grosse pile au lithium qui lui donne une autonomie de 10 ans (à cette époque l'Apple figurera avec la De Dion-Bouton au Musée des Sciences et Techniques !)

Un seul point noir (mais de taille !), l'horloge Proclock coûte une fortune : 2384 FF ttc, pratiquement le prix d'un micro familial.

Proclock est distribuée par Alpha-Systèmes, 29 boulevard Gambetta, 38000 Grenoble.

PF ■

UN PETIT TOUR CHEZ LE LIBRAIRE

Meca Basic
20 programmes en Basic
de mécanique appliquée
pour 1ères et terminales
S et TS et IUT
Thierry Tacquet
et Francis Legroux
Éditions PSI
Lagny, 1984
Broché, 146 pages
Prix : 95 FF

Le livre du lecteur de disquette 1541
Lothar English
et Norbert Szczepanowski
Traduit par Pascal Hausmann
Micro Application
Paris, 1984
Broché, environ 300 pages
Prix : 179 FF

Amstrad - le Basic au bout des doigts
Rampow
Traduit par Pascal Hausmann
Micro Application
Paris, 1985
Broché, 190 pages
Prix : 149 FF

La pratique du Fortran 77
Patrice Lignelet
Éditions Masson
Paris, 1985
Broché, 232 pages
Prix : 130 FF

Graphisme scientifique de la 2e à la 3e dimension
50 applications résolues
en Basic
Robert Dony
Éditions Masson
Paris, 1985
Broché, 254 pages
Prix : 110 FF

Le langage LM
E. Mazer
et J.-F. Miribel
Éditions Cepadues
Toulouse, 1985
Broché, 112 pages
Prix : 110 FF

Dictionnaire lexique micro-informatique
H. Chuquet
et J.-C. Fantou
Éditions Radio
Paris, 1984
Broché, 262 pages
Prix : 145 FF

Amstrad programmes Basic
Tome 2
Luers
Traduit par Pascal Hausmann
Micro Application
Paris, 1985
Broché, 182 pages
Prix : 129 FF

L'utilisation de l'Amstrad CPC 464
Ian Sinclair
Traduit par Alain Pierrot
Hachette Informatique
Paris, 1985
Broché, 254 pages
Prix : 125 FF

Basic sans peine
Auto-initiation au Basic MO5 TO7/70
Livre accompagné de deux cassettes
André Deledicq
Cedic-Nathan TO-TEK
Paris, 1985
Broché, 192 pages
Prix : 175 FF

Assembleur et périphériques des MO5 et TO7/70
Frédéric Blanc
et François Normand
Éditions PSI
Lagny, 1985
Broché, 124 pages
Prix : 85 FF

Introduction à la robotique
Pierre Lopez
et Jean-Numa Foulc
Éditions Éditests

Tome 1 : Notions de base, architecture, systèmes actionneur et sensoriel, modes de fonctionnement
Paris, 1984
Broché, 320 pages
Prix : 190 FF

Tome 2 : Communication homme-machine, programmation et commande, transformations homogènes
Paris, 1985
Broché, 286 pages
Prix : 190 FF

Manuel technique du MO5
Michel Oury
Cedic-Nathan TO-TEK
Paris, 1985
Broché, 120 pages
Prix : 125 FF

Introduction à MSX
Alain Perbost
et Didier Berthet
Éditions Édimicro
Paris, 1985
Broché, 142 pages
Prix : 108 FF

Premiers pas en Basic
Susan Curran
et Ray Curnow
Traduit par Gérard Schmitt
Hachette Informatique
Paris, 1984
Broché, 192 pages
Prix : 99 FF

L'ordinateur pour apprendre
Susan Curran
et Ray Curnow
Traduit par Gérard Schmitt
Hachette Informatique
Paris, 1985
Broché, 168 pages
Prix : 99 FF

Introduction à la micro-informatique
Peter Laffety
Traduit par Pierre Grammat
Hachette Informatique
Paris, 1984
Broché, 184 pages
Prix : 99 FF

Destination aventure
Programmes de jeux de rôle et d'aventure sur Commodore 64
Delton T. Horn
Adapté par Gigi
Éditions PSI
Lagny, 1984
Broché, 244 pages
Prix : 140 FF

Une souris pour le Dai

DEPUIS l'arrivée des « phénomènes » Apple qu'ont été dans leur temps Lisa puis Macintosh, le concept de « souris » qu'on pointe et « clique » est devenu un mythe dont rêvent les possesseurs d'ordinateur. Sans discuter à perte de vue sur les avantages et les inconvénients de ce système (et le clavier, alors ?), un peu de bon sens montre que ce gadget n'est pas forcément réservé à des machines privilégiées. Témoin, cette prothèse sur Dai, qui fonctionne parfaitement.

C'est la nouvelle fédération des clubs Dai belges (IDC, International Dai Club) qui a eu l'idée de repenser le problème à la base. Une souris n'est ni plus ni moins qu'une manette de jeu un peu spéciale : au lieu de l'habituel « manche à balai », c'est généralement une bille qui transmet l'information de position aux potentiomètres, par l'intermédiaire de galets à frottement doux. Quant au bouton qui « clique », c'est un simple contacteur à fermeture. Plutôt que de « bricoler » tout ça, les membres de l'IDC ont préféré faire le tour des souris du marché, afin de voir s'il n'existait pas de modèle modifiable facilement, pour l'adapter au Dai. Leur choix s'est porté sur la *TRS 80 Color Mouse*. Les potentiomètres ont la valeur requise, et il suffit simplement d'en modifier le câblage, et de changer la prise Din, à l'autre extrémité de la « queue » de la souris. Aussitôt dit, aussitôt fait. Reste à inventer le logiciel qui utilise ce nouveau périphérique. IDC propose déjà un *Éditeur de dessin* qui reprend le concept des icônes : on les sélectionne en « cliquant » dessus la flèche que la souris promène à l'écran. Inutile de vous faire un dessin...

Ce genre d'adjonction de périphérique est envisageable sur tout ordinateur qui possède *des entrées analogiques pour ses manettes de jeu* (entrées poten-

tiométriques). Selon le cas, il sera peut-être nécessaire de rectifier la résistance de la piste carbonée, avec un petit ajustable (trimmer) en parallèle sur le curseur et l'extrémité « froide » du potentiomètre. Si, comme pour le Dai, la bonne valeur se situe aux alentours de 100 Kohms, ce travail est inutile.

Pour plus de renseignements techniques, ou bien pour obtenir l'éditeur de dessin fonctionnant avec cette souris, contacter :

International Dai Club Belgique
Fabrice Duluins
4 allée Tour Renard
B-1400 Nivelles
ou
International Dai Club France
Bruno Delannay
Résidence Les Acacias
Bat. B3, avenue de Saige
33600 Pessac

AM ■

UN LIVRE

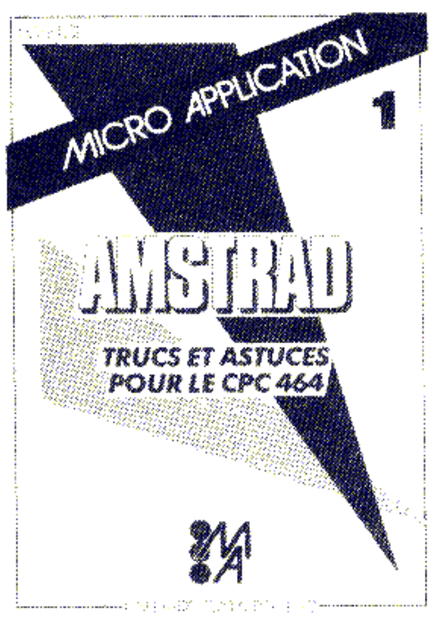
Amstrad trucs et astuces pour le CPC 464
English, Germer,
Scheuse, Thrlin
Édité en français par
Micro Application
Rueil-Malmaison, 1985
Broché, 280 pages
Prix : 149 FF

A l'heure où je rédige ces lignes, la littérature concernant l'Amstrad est encore rare... Aussi, l'édition d'un ouvrage consacré à cette nouvelle machine ne pouvait qu'aiguiser une saine curiosité. *Trucs et astuces pour le CPC 464* est la traduction française d'un livre édité en Allemagne par Data Becker, éditeur informatique bien connu outre-Rhin.

Ses quatre auteurs ont été visiblement passionnés par l'Amstrad et certainement pressés par le temps. Il en résulte un livre dense, un peu « touche à tout », de qualité inégale selon ses chapitres.

Les premiers thèmes abordés concernent le graphisme haute résolution et la programmation des caractères. Quelques courts programmes sont fournis, qui complètent la notice de l'appareil. Certains éléments ne sont qu'une répétition de cette notice. Une étude de plus haut niveau concerne la gestion de l'écran. Cette étude ne sera guère accessible aux purs débutants : elle est un peu « pointue ».

Viennent ensuite les possibilités sonores, et un petit programme d'édition musicale. Ici, les explications sont soit trop simples pour celui qui « connaît



la musique » ; soit un peu trop ardues pour le complet béotien.

Quant à l'introduction au langage-machine, elle ne s'adresse ni aux vrais débutants ni aux champions du Z80. L'étude rapide du système d'exploitation et du Basic ravira les passionnés bien armés pour en tirer profit : plusieurs pages sont consacrées à diverses routines système utilisables depuis Basic. C'est passionnant. Tout n'étant pas dit, il y a là de quoi encourager des recherches.

D'autres indications arrivent ensuite, destinées plutôt aux débutants : précision et vitesse de calcul, tri de données. Sur le thème « transmissions de données du CPC au Commodore 64 », un programme est proposé, destiné au Commodore !

Le livre s'achève sur une étude approfondie de deux programmes plus complexes (fichiers et traitement de texte), et sur deux courts programmes de jeu... pour débutants.

Voici donc un livre un peu décousu, certes, mais à coup sûr un trésor pour qui veut en savoir plus sur son Amstrad. Débutants et connaisseurs y trouveront matière à réflexion.

JPL ■

UN CLUB... UN SALON... UNE FOIRE...

Apple Expo 85 au Parc des Expositions les 14, 15 et 16 juin prochains

POUR la seconde fois (la première Apple Expo s'était tenue les 22, 23 et 24 juin 1984), Apple donne rendez-vous au Parc des Expositions de Paris, à ceux qui, de près ou de loin, s'intéressent à ses productions.

Pendant trois jours et sur plus d'un hectare, les « applemaniaques » retrouveront concessionnaires, développeurs, éditeurs et médias spécialisés dans les produits de la firme américaine.

A la disposition du public, plus de 120 ordinateurs en démonstration dont le Macintosh géant et, sur un stand spécial, accès libre aux serveurs du monde entier.

Apple Expo
Parc des Expositons
Porte de Versailles (Paris)
Bâtiment 1.1
vendredi 22 juin, de 11 à 22 h
samedi 23, de 9 h 30 à 22 h
dimanche 24, de 9 h 30 à 19 h
Entrée : 50 FF

A Cogolin dans le Var, 1er Salon informatique

ORGANISÉ par le club informatique, sous l'égide de la municipalité et de la région, le premier Salon informatique de Cogolin se tiendra les 2, 3 et 4 juin prochains au Centre culturel de la ville.

Journée réservée aux associations, le dimanche 2 juin sera consacré à l'accueil des clubs varois. Les lundi 3 et mardi 4 suivants, ce sont les professionnels qui présenteront leurs matériels.

L'entrée de ce Salon est gratuite.

Centre culturel de Cogolin
Avenue Georges Clémenceau
83310 Cogolin

Pour plus de renseignements, vous pouvez appeler aux numéros suivants :
(94) 56 36 52, (94) 56 32 40

On achève bien les programmeurs

AVIS aux forçats du microprocesseur, stakhanovistes de la programmation, les 4 et 5 mai prochains auront lieu à Paris les premières *24 heures de l'Informatique*. Cette compétition non-stop rassemblera, 24 heures durant, cent concurrents originaires de différents pays francophones qui devront concevoir un logiciel à vocation pédagogique, destiné au Hewlett-Packard 150.

Les équipes de deux personnes qui disputeront ce premier marathon de l'informatique seront réparties en deux catégories : amateurs et professionnels salariés des métiers de l'informatique.

Cette épreuve, patronnée par le *Centre d'Études Supérieures en Bureautique et Informatique*, sera ouverte aux amis et supporters des concurrents. Un véritable 24 h du Mans, version circuits intégrés.

Les vainqueurs du Grand Prix recevront un Hewlett-Packard HP-150 à écran tactile, d'une valeur de 41500 F.

D'ores et déjà, les organisateurs de ces 24 heures prévoient de recevoir quelque 5000 dossiers d'inscription.

Conséquence : les épreuves de qualification seront particulièrement sévères. Chaque candidat devra faire montre de ses aptitudes à la programmation et à l'analyse informatique avant de pouvoir accéder aux demi-finales, qui éprouveront la résistance physique et la capacité à travailler en équipe des candidats. Les cent meilleurs participants de ces demi-finales pourront enfin accéder à la grande finale.

Là encore des surprises avec des « micro-sprints », où les programmeurs devront résoudre quelques petits problèmes informatiques au cœur de la nuit.

Coup d'envoi de ces 24 heures samedi 4 mai à 16 h (40, rue de Liège, Paris 8e). Les bulletins

d'inscription sont à retirer à l'adresse suivante :
Les 24 heures de l'informatique
40 rue de Liège
75008 Paris
Tél. : 293 12 50

Si vous souhaitez adhérer au club ou en savoir plus sur ses activités, vous pouvez écrire à :
Microfer
1 bis rue d'Athènes
75009 Paris

Microfer le club informatique de la SNCF

MICROFER, qui compte déjà plus de 1 000 adhérents, propose un ensemble d'activités variées ouvertes aux débutants en informatique comme aux confirmés : initiation à la programmation en Basic, réductions sur le prix de nombreux matériels, publication d'un bulletin intitulé « Interface », etc.

En collaboration avec l'Agence de l'Informatique et la Direction commerciale voyageurs de la SNCF, Microfer participe également aux séances d'initiation à l'informatique organisées dans les trains Loisirail.

Les conditions d'adhésion 1984/1985 sont les suivantes : pour les cheminots et leur famille, la cotisation s'élève à 200 F et son renouvellement à 100 F, pour les autres personnes, l'adhésion se monte à 400 F et son renouvellement à 200 F.

Dans les Hauts-de-Seine, stages d'informatique au Centre X2000/ Les Corolles

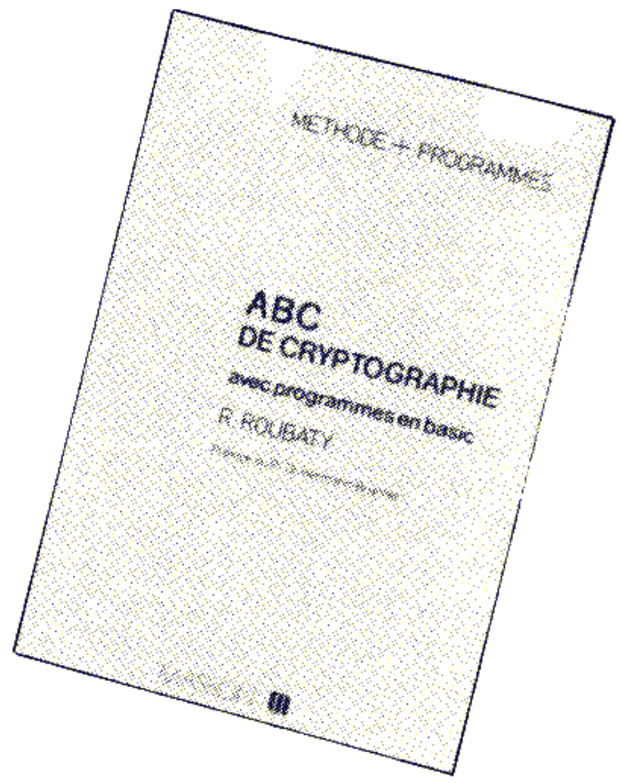
LE Centre X2000/Les Corolles de Courbevoie propose des stages d'initiation à l'informatique, au Basic et au Logo et des stages de perfectionnement par l'approche de logiciels professionnels (gestion de fichier, traitement de texte, etc.).

Il est possible de choisir entre deux formules : trois jours complets ou deux soirées par semaine pendant cinq semaines (le lundi et le jeudi ou le mercredi et le vendredi de 18 à 20 h).

Le prix des stages d'initiation est de 400 F, celui des stages de perfectionnement de 800 F. Ils se déroulent au
Centre X2000/Les Corolles
13 place des Corolles
92400 Courbevoie

Pour plus de renseignements, vous pouvez contacter Mlle Edith Perrinet au *(1) 773 64 07.*

UN LIVRE



ABC de cryptographie
Romain Roubaty
Editions Masson
Paris, 1984
Broché, 210 pages
Prix : 136 FF

CE livre de la collection *Méthode + Programmes* est, comme son nom l'indique, un ouvrage d'initiation. Mais la cryptographie y est directement abordée par la face Nord, je veux dire sous son angle le plus abrupt, celui du décryptement. En contrepartie le choix d'escalades qui nous est proposé est sagement limité à des procédés de substitution ou de transposition simples et classiques. En prime un exposé du *One-time,* des indications sur les procédés à clefs publiques et une version abrégée de l'un d'entre eux, le *Deffie-Hellman,* sans lesquels le titre d'ABC du décryptement eut été plus justifié. Mais on reste inévitablement assez en deçà du niveau actuel des problèmes majeurs liés au développement explosif des capacités de calcul et des besoins de sécurité, et à celui plus progressif des théories de la complexité.

L'auteur a le très grand mérite de montrer avec clarté et rigueur comment l'informatique permet la démolition rapide et quasi automatique des procédés de la cryptographie classique. Chaque application fait l'objet de programmes annotés de REM permettant d'en suivre le fonctionnement.

Si les décrypteurs se sont souvent illustrés par des intuitions géniales, c'est à leur patience méthodique qu'ils doivent l'essentiel de leurs succès. L'informatique prend aujourd'hui en compte tout le côté fastidieux des décomptes statistiques, de l'application des hypothèses successives, des retours en arrière, etc. Les outils développés par Romain Roubaty semblent parfaitement adaptés à tous ces travaux et, soit tels quels, soit comme prototypes à remanier en fonction des besoins, ils seront très utiles à ceux que tente un apprentissage du décryptement. Un matériel assez complet avec disquettes, imprimante, et gestion de fichiers leur sera nécessaire. Le *Deffie-Hellman* exige de son côté des fonctions de cal-

cul en multi-précision sur 28 chiffres. Encore s'agit-il d'un exercice d'école, les applications en vraie grandeur, mettant en jeu des nombres de 100 à 200 chiffres, n'ont finalement pas résisté aux assauts des décrypteurs qui ont « cassé » tout dernièrement ce procédé.

L'auteur pousse parfois le scrupule jusqu'à suivre fidèlement les méandres ayant abouti à certaines créations (alphabet cyclique de Delastelle) plutôt que les raccourcis conduisant à un résultat équivalent (ici, alphabet désordonné vertical calé de façon à redonner, en clef Z, l'alphabet normal). Le caractère didactique de cet ouvrage nous prive en revanche des anecdotes dramatiques ou cocasses qui jalonnent l'histoire de la cryptographie.

Enfin les textes poétiques — de qualité — choisis comme exemples récompenseront agréablement les lecteurs qui auront suivi pas à pas leur reconstitution à partir de rébarbatives versions chiffrées.

PB ■

Lecteurs de disquettes portables

A quoi bon un portable si son utilisation requiert impérativement une prise de courant ? Une simple question qui pour l'instant n'avait pas encore de réponse concluante. Et pour s'en convaincre un rapide coup d'œil à l'impressionnante armada de périphériques et de fils s'y rattachant suffisait.

Le tout est à mettre au passé puisque les Japonais (en l'occurrence **Citizen Watch Company**) viennent d'annoncer la sortie de deux modèles de lecteurs de disquettes 3,5 pouces portables, entièrement autonomes, alimentés par des piles de 5 volts. Le lecteur ne pèse que 450 grammes et accepte, selon les versions, des disquettes simple ou double densité.

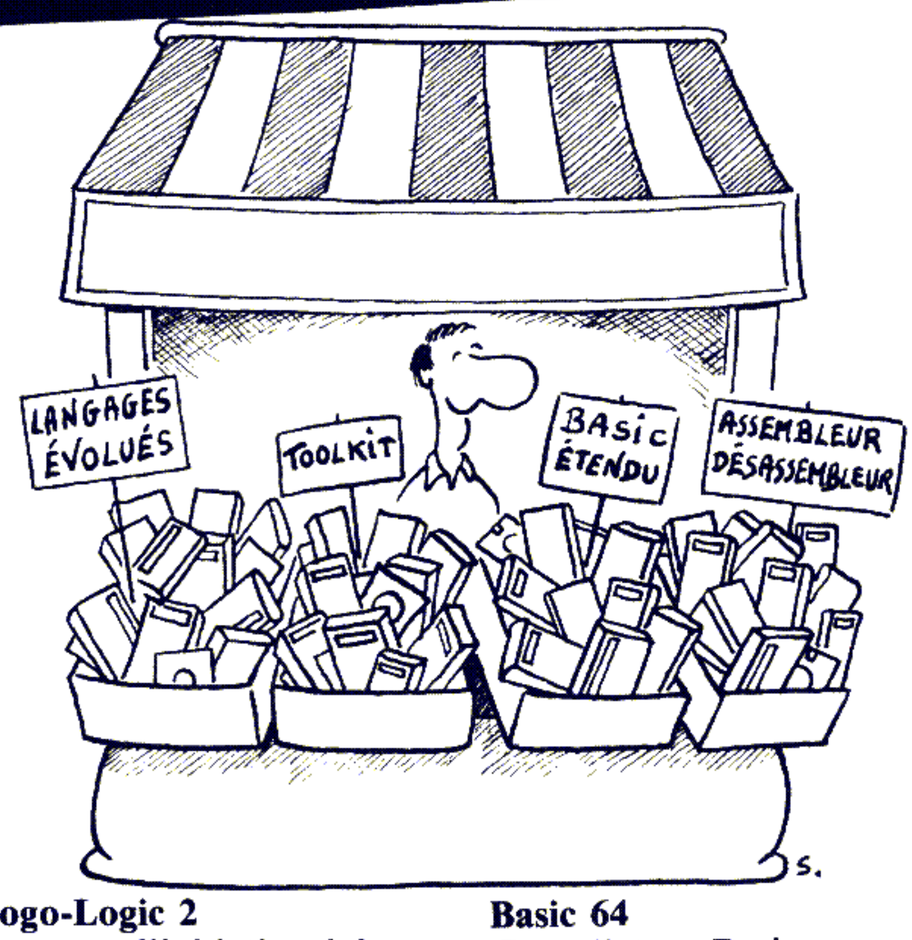
Il en coûte environ 2000 F pour un lecteur 500 Ko et 2500 F pour la version 1 méga-octet. Une désillusion toutefois, ces lecteurs « miracles » ne sont disponibles, pour l'instant, qu'au Japon. ■

Un MSX coréen en France

A la suite d'un accord entre la firme coréenne **Lucky Goldstar International Corporation** et la société française **ASN Diffusion Electronique,** cette dernière est habilitée à distribuer le FC-200 (conçu par la firme coréenne) sous le nom de Goldstar/Asn.

Accompagné d'un manuel d'utilisation en français, d'une cassette de démonstration et de cordons de raccordement Péritel et magnétophone, cet ordinateur familial aux normes MSX devrait être disponible début mai au prix public de 2 590 FF. ■

CASSETTES ET DISQUETTES



Logo-Logic 2
Langage d'initiation à la programmation
Cassettes et disquettes pour C.64
Edité par No Man's Land
Distribué par Innelec
Prix : 495 FF

Es Forth
Cassette pour Atari 400/800 et XL 32 Ko
Mode d'emploi en anglais
Edité par English Software
Distribué par Innelec
Prix : 180 FF

Odin
Editeur-désassembleur
Assembleur symbolique
Cassette pour MO 5
Edité par Loriciels
Prix : 295 FF

Compilateur
Compilateur Basic
Cassette pour ZX Spectrum
Edité par Ere Informatique
Prix : 250 FF

Extra Tool 64
Basic étendu
S'utilise avec la cartouche Tool 64
87 instructions nouvelles (dont 42 spécifiques à Extra Tool)
Disquette ou cassette pour C.64
Edité par Micro Application
Prix : 245 FF

Basic 64
Compilateur Basic
Disquette pour C.64
Edité par Micro Application
Prix : 350 FF

Logo
Initiation à la géométrie d'exploration par l'intermédiaire de la tortue
Cassette pour MSX
Mode d'emploi en anglais
Edité par Kuma
Distribué par Innelec
Prix : 235 FF

AssDesass
Assembleur-désassembleur
Cassette pour TO 7-TO 7/70 et MO 5
Edité par Infogrames
Prix : 350 FF

Max
Assembleur, moniteur
Cassette ou disquette pour C.64
Edité par Micro Application
Prix de la cassette : 195 FF
Prix de la disquette : 350 FF

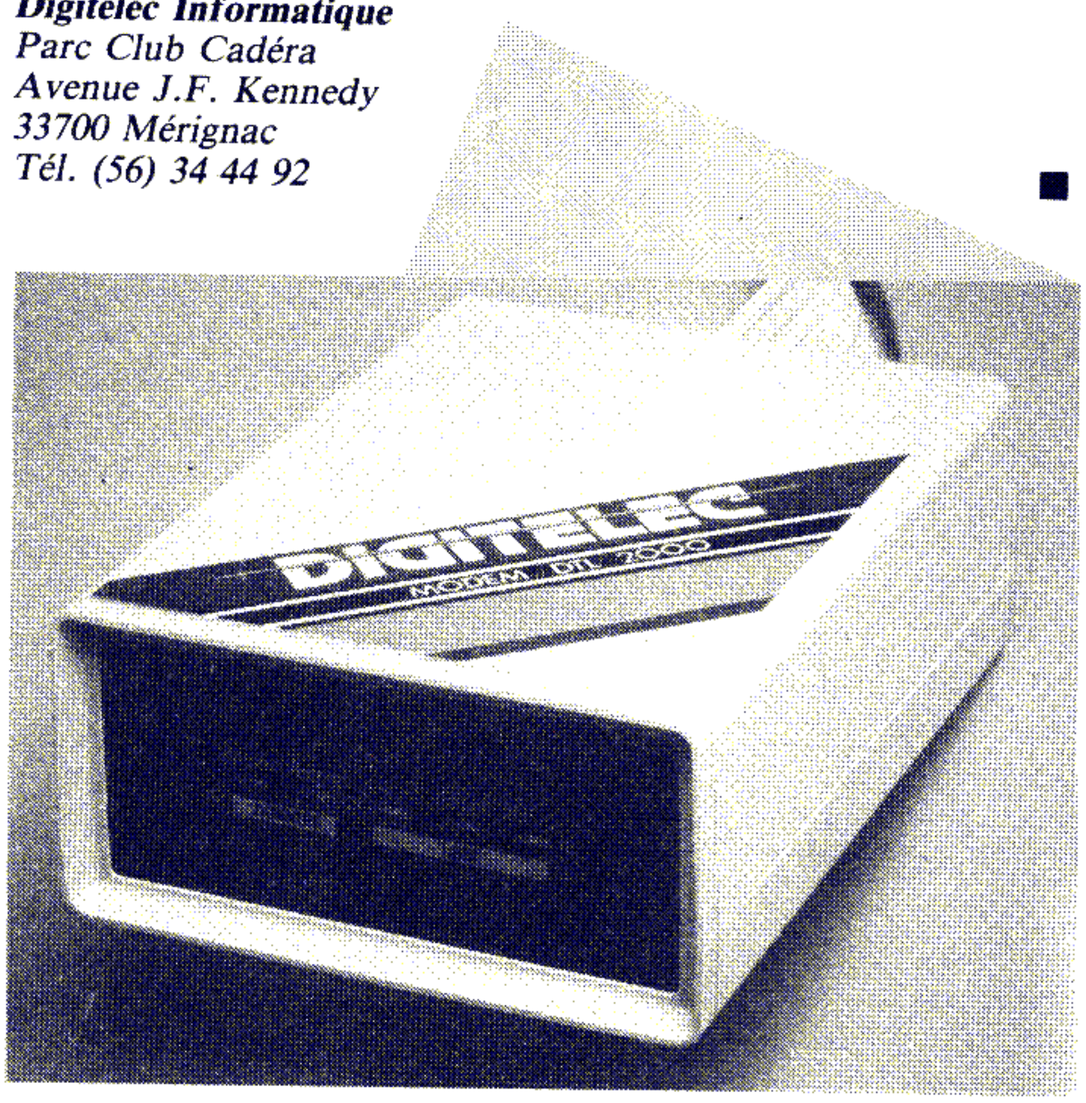
Logor
Logiciel-langage destiné aux apprentis de la programmation.
Permet de s'initier à la géométrie informatique de Logo
Cassette pour Oric 1/Atmos
Edité par Infogrames
Prix : 160 FF

DTL 2000 Plus, un téléphone pour les ordinateurs

LES ordinateurs ont aussi besoin de communiquer entre eux. Leur téléphone s'appelle « modem ». Le DTL 2000 Plus, présenté à *Micro-Expo* (du 16 au 19 février 1985, à Paris), est compatible avec les micro-ordinateurs : Apple IIe, Apple II+, Commodore 64, Oric, Spectrum, et avec une sortie RS-232C pour les autres ordinateurs. Le prix de ce modem est de 1 990 FF ttc.

Digitelec Informatique
Parc Club Cadéra
Avenue J.F. Kennedy
33700 Mérignac
Tél. (56) 34 44 92

■



Baisse de prix pour l'Alice 32 Ko

APRÈS le Spectrum Plus de **Sinclair**, qui a vu son prix passer de 2 320 F à 1 660 F (voir *LIST* 8 page 16), c'est l'Alice 32 Ko de **Matra** qui enregistre une baisse de près de 17 %.

Depuis le 15 mars dernier en effet, 995 F suffisent pour acquérir un Alice 32 Ko qui coûtait jusqu'alors 1 195 F.

Souhaitons que d'autres constructeurs s'engagent sur la même voie et que nous puissions, dans les mois prochains, annoncer de nouvelles baisses intéressantes.

■

Série limitée

DEPUIS le 15 avril 1985, **Vidéo Technologie** diffuse le Laser VZ 200, ordinateur disposant de 4 Ko de mémoire vive, 16 Ko de mémoire morte, 9 couleurs programmables, 16 caractères graphiques, 3 modes d'affichage, un écran de 12 lignes sur 32 colonnes, une haute résolution de 128 sur 64 points pour 8 couleurs, un clavier de 45 touches, un microprocesseur Z80 A.

Ce VZ 200 est livré en Pal avec tous les accessoires permettant de le modifier en Secam Péritel. Il dispose des logiciels pour Laser 200 et 310.

Son principal intérêt : il ne coûte que 690 FF ttc (+40 FF de port pour un règlement à la commande, +70 FF de port pour une expédition en contre-remboursement). De quoi découvrir le Basic pour une somme vraiment modique.

Mais attention, la série est limitée à 3 000 exemplaires.

Vidéo Technologie France
19 rue Luisant
91310 Montlhéry
Tél. (6) 901 93 40

■

Atari à aiguille

LES ordinateurs Atari peuvent choisir leur imprimante : après l'Atari 1020 à 4 couleurs et l'Atari 127 qualité courrier, voici maintenant l'Atari 1029. C'est une imprimante matricielle à aiguille, dont la vitesse d'écriture est de 50 caractères par seconde. Avec l'Atari 800XL, elle permet aussi de créer des graphismes propres à la machine, sur simple feuille ou sur bande de papier. Son prix public (moyen) est de 2 100 FF.

■

Pour la défense de logiciel

SOULAGEMENT pour les uns, angoisse pour les autres, un groupe de juristes, d'experts en informatique et de conseillers juridiques et fiscaux vient de créer à Paris *l'Association Française de Droit de l'Informatique* (AFDI).

Son but : « réunir toutes les personnes concernées directement ou indirectement par les problèmes juridiques en relation avec l'informatique ». A cet effet, l'AFDI organisera des réunions, des manifestations nationales et internationales « susceptibles de favoriser la connaissance et l'évolution du droit de l'informatique ».

L'AFDI devrait en outre permettre aux praticiens et aux chercheurs de « confronter leurs expériences et leurs réflexions afin de dégager des lignes directives en matière de droit informatique ».

Des travaux qui devraient aider, à plus ou moins long terme, à élaborer une législation cohérente pour la protection des logiciels. Pour tout renseignement :

Tribunal de Commerce
1 quai de Corse
75001 Paris

■

LA SECONDE NAISSANCE DU BASIC

(AUTHENTIQUE CONTE DE NOËL)

Si le Basic initial, créé par John Kemeny, était une merveilleuse astuce pédagogique d'universitaire, il devient, grâce à William H. Gates, dit Bill, un puissant outil de développement aux répercussions culturelles et économiques considérables pour notre décennie. Ainsi, après John Kemeny, le nom le plus célèbre du Basic est celui de Bill Gates.

Quelle voie normale un juriste puissant et renommé de Seattle (siège de Boeing, dans l'état de Washington) peut-il envisager pour son fils né en 1956 ? Le droit, naturellement, surtout dans un pays où les carrières juridiques jouissent d'un tel prestige et peuvent conduire à tous les pouvoirs. C'est, en tout cas, le choix que fit le père de William H. Gates. Aujourd'hui, Bill (pour tous les mordus de micro-informatique) devrait être, à près de trente ans, l'une des étoiles montantes du barreau ou de la magistrature américaine. Mais il aimait tellement les mathématiques !

Étudiant à Harvard, il hésite encore entre ses désirs et la pression paternelle quand, une nuit de Noël... mais n'anticipons pas. Il faut dire aussi que, déjà vers 1969, il s'était découvert un hobby plutôt illégal : le « piratage » d'un système de temps partagé professionnel, aussi sérieux que celui de Digital Equipement ou celui de Control Data Corporation (en France, il aurait peut-être intéressé l'hebdomadaire « Le Canard Enchaîné »). Il lui arriva même de se faire pincer et d'être engagé, du coup, par la Control Data Corporation pour l'aider à mieux protéger ses filières d'accès ! C'est devenu plus banal en 1985, mais à l'époque...

Avec son camarade de classe Paul Allen, il acheta pour 360 dollars l'un des premiers microprocesseurs Intel 8008 (sorti en 1971), puis créa en 1972 la société Traf-O-Data pour tenter de régulariser le trafic dans la banlieue de Seattle. (On trouvera des détails passionnants sur cette première, mais aussi sur l'histoire peu connue de l'ordinateur individuel, dans le best-seller de Paul Freiberger et Michael Swaine, *Silicon Valley*, chez McGraw Hill).

Le mythique Altair

Une intéressante livraison d'avril 1984 de la revue *Time* révèle, par ailleurs, une autre « affaire » de Bill Gates alors âgé d'une dizaine d'années à peine : il aurait mis au point un emploi du temps scolaire lui permettant d'optimiser, en choisissant ses « unités », le nombre de ses petites camarades de classe, prises bien entendu parmi les plus mignonnes...

En 1974, il entrait en première année de droit à Harvard, sans avoir vraiment abandonné toute idée de carrière mathématique ; mais il n'aura pas le temps, nous le verrons, de régler tout à fait ce conflit personnel. Le numéro de janvier 1975 de *Popular Electronics* portait, en couverture, la photographie du premier

véritable micro-ordinateur commercialisé sur une certaine échelle : le mythique Altair. Aux environs de Noël, le vieux copain Paul Allen, alors programmeur chez Honeywell à Boston, acheta un exemplaire de ce journal au « Nini's Corner » de Harvard Square et bondit chez Bill Gates, tout excité. Effrayés à l'idée que leur chance allait peut-être leur passer sous le nez, les deux gamins (respectivement dans leur vingtième et vingt-deuxième année) téléphonèrent à Albuquerque, dans le Nouveau-Mexique, au siège de la compagnie MITS (Micro Instrumentation and Telemetry Systems) de Roberts et ils proposèrent, sous six semaines, un Basic (le MBasic) pour l'Altair 8800.

Ce n'était pas — même pour cette machine alors toute récente qui tournait à peu près correctement depuis l'été précédent — le premier Basic qu'on avait écrit à des fins utilitaires ou commerciales. On peut par exemple citer un *Tiny* Basic d'Allison et d'autres encore. Le problème technique, cependant, était redoutable : l'ordinateur construit sur le 8080 d'Intel (une version améliorée du 8008), n'avait en standard que 256 octets (!) de mémoire vive ; une version de travail chez MITS comportait sept cartes, chacune d'un Koctet, et l'interpréteur Basic devait se loger en 4 Koctets.

Bill Gates et Paul Allen, qui avaient déjà vainement essayé d'implanter un langage de ce type sur leur vieux microprocesseur personnel, travaillèrent comme des fous, calmant tant bien que mal leurs familles. Le plus extravagant est qu'ils écrivirent ces 4096 octets sans avoir vu l'ordinateur, à des milliers de kilomètres de là, qu'ils avaient dû simuler assez imprécisément sur une grosse machine (sans doute plus ou moins clandestinement, à l'Université).

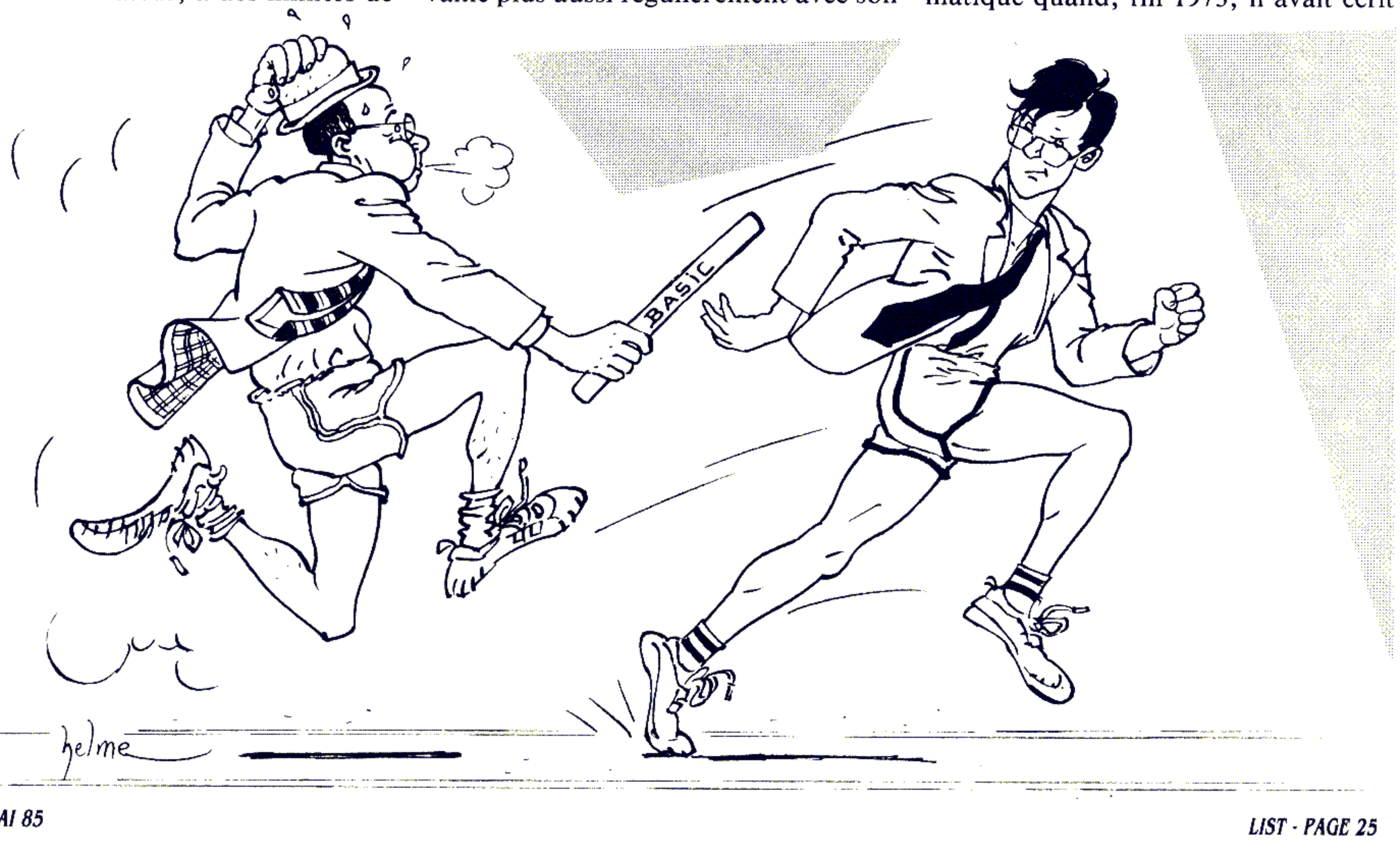
On peut lire dans *Silicon Valley* comment Allen, seul dans l'avion, dut écrire à la dernière seconde un programme d'assembleur 8080 pour pouvoir charger, à l'aide de la bande perforée, leur langage à l'intérieur du véritable Altair qu'il ne connaissait pas... Enfin le célèbre MEMORY SIZE? s'imprima — pas encore d'écran à l'époque, bien entendu — et, dit-on, Allen osa demander à son programme d'exécuter l'incroyable instruction PRINT 2 + 2, qui voulut bien réagir conformément à toutes les axiomatiques connues et à venir de l'arithmétique classique et/ou moderne.

Le Basic le plus célèbre

Le reste est toute l'histoire de Traf-O-Data devenu Microsoft, heureusement très vite indépendante de l'Altair et de son échec commercial ultérieur. La firme devait inonder le marché (Tandy, Apple — même si l'Applesoft, écrit partiellement à partir du propre Basic de Wozniak pour l'Apple I est, assez bizarrement, orné de quelques particularités discutables —, puis enfin IBM avec le Basic-A de son PC) de versions multiples du plus célèbre Basic de tous les temps, devenu rapidement un standard.

Aujourd'hui Allen, malade, ne travaille plus aussi régulièrement avec son ami. Microsoft a su — chose rare en micro-informatique — ne pas être la société d'un seul succès. Elle a pu répandre sous sa marque au moins trois best-sellers dont MS-DOS et Multiplan. Sa réussite semble moins facile aujourd'hui (il est clair par exemple que la mise au point de MS-Win est assez problématique). Mais le génie de Gates, par ailleurs manager hors pair (comme il sut le prouver en réussissant à dégager sa compagnie au moment de la vente de la MITS qui considérait le Basic comme sa propriété et non celle de ses auteurs) nous promet sans doute encore de belles réussites. Rien qu'avec le Basic, il a déjà marqué de façon indélébile toute la branche logicielle, et peut-être sur certains points la branche matérielle de la micro-informatique.

Une partie de l'histoire des applications de l'ordinateur individuel consiste en la rivalité de Microsoft et de Digital Research. Curieusement, cette dualité se retrouve aussi dès la mise au point des premiers Basic, même si, par exemple, le CBasic plus ou moins issu de Gary Kildall est méconnu par rapport à celui de Gates et Allen. Kildall était un bon prof d'informatique, spécialisé dans la théorie des compilateurs, à l'US Naval Postgraduate School de Monterey (Californie). Tenté par le défi, il avait déjà implanté sur un Intel 8008 une version PL/M d'un petit bout du célèbre PL/I, tâche inouïe compte tenu de la lourdeur de ce dernier langage. Mais très vite il comprit la nécessité d'un logiciel spécialisé dans le contrôle des unités de disquettes ; celles-ci étaient encore pratiquement réservées à la « grande » informatique quand, fin 1973, il avait écrit



PETITE HISTOIRE DES LANGAGES

LA SECONDE NAISSANCE DU BASIC

le CP/M (Control Program Monitor), dans son PL/M, puis en Assembleur. On sait quel fut le succès de ce logiciel, et la gloire de la société Intergalactic (sic) Digital Research qu'il créa pour le diffuser.

Mais il avait également travaillé sur un interpréteur Basic et c'est ce sujet de recherche qu'il proposa en 1976 à un certain Gordon Eubanks (dont on peut admirer la barbe bien taillée et les lunettes d'intellectuel dans *Silicon Valley*). Eubanks accepta mais mit au point un CBasic, C signifiant ici « compilé » et non plus interprété, compatible avec CP/M, commercialisé dès avril 1977 par Digital Research, vite adopté par les constructeurs de l'IMSAI (autre ancêtre du micro), puis également disponible pour l'Altair lui-même.

Les Basic innombrables qui virent le jour furent généralement interprétés. Ce fut sans aucun doute la source de leur développement prodigieux chez les amateurs, entraînant par contrecoup celui de l'ordinateur personnel. Toutefois, certains Basic existent en versions semi-compilées comme Pascal presque dès le début de la préhistoire. (Microsoft se dépêcha d'ailleurs de sortir son propre compilateur Basic.)

Les Basic débordèrent même très vite du cadre strict de l'ordinateur de table. C'est ainsi que, parmi une foule de nouveautés, un ordinateur de poche fit sensation à la West Coast Computer Fair en mars 1980 au Civic Center de San Francisco : le fameux Sharp 1210 (devenu 1211, puis 1212 aujourd'hui), avec un Tiny Basic et 400 octets pour l'utilisateur. Il sera tout de suite célèbre, surtout aux États-Unis quand Tandy vendra le modèle 1211 (1424 octets de mémoire vive) sous le nom de TRS-80 Pocket. Beaucoup d'étudiants, de lycéens ou de professeurs lui doivent leurs premiers PRINT 2 + 2...

Encore quelques petites années...

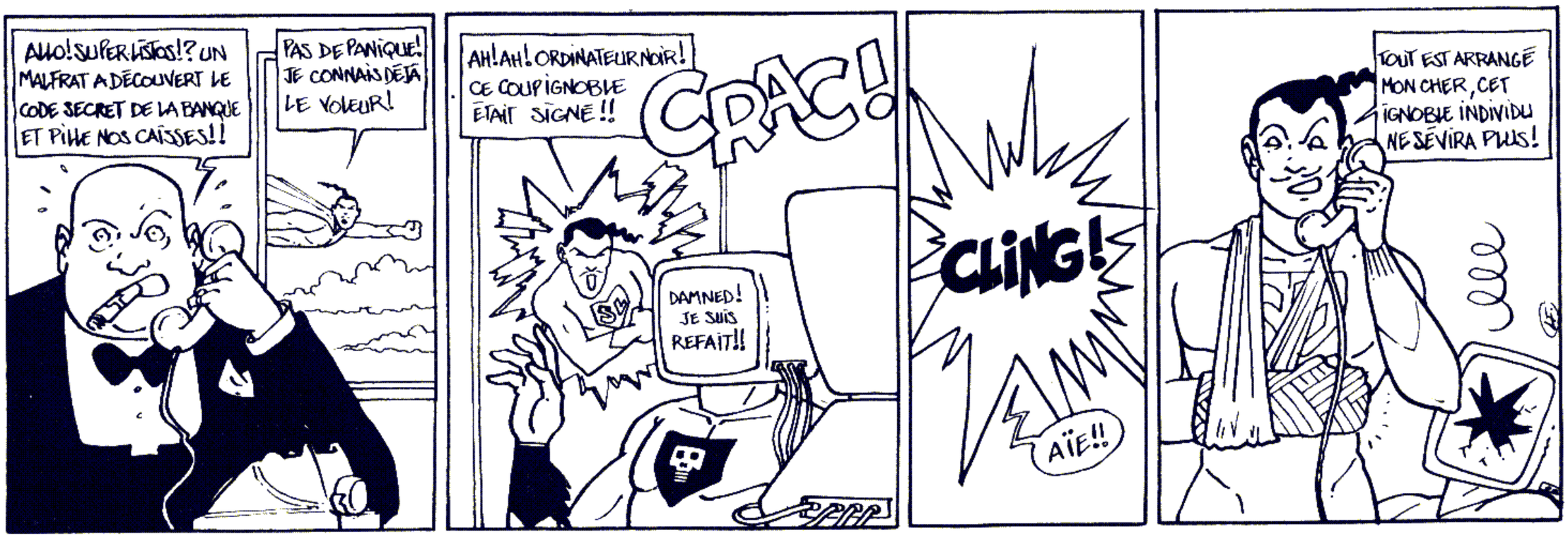
Essayer même de citer les principales versions de notre langage depuis 1975 serait totalement utopique. Peut-être est-il tout juste possible, parmi les nombreuses variantes, de parler un peu de l'une des plus récentes, qui présente une particularité peu courante : elle a été écrite par John Kemeny et Thomas Kurtz, mais en1984 ! En juillet de l'année précédente, ils avaient créé la société « True Basic » afin de promouvoir un Basic portable de haut niveau, tout particulièrement graphique, compatible avec des standards exigeants mis au point par l'ANSI. Il est compilé « interactivement » et non plus interprété, comme l'ancêtre, et se veut évidemment « moderne ».

Citons pêle-mêle une grande structuration (IF-ELSE-ENDIF, DO LOOP, SELECT CASE...), un éditeur plein-écran, l'adaptabilité aux périphériques de type souris, des fenêtres, des définitions de fonctions pouvant dépasser la longueur fatidique d'une ligne, des sous-programmes avec variables locales, bref une foule d'héritages discrets de Pascal, le jeune concurrent. Le dernier livre de Kemeny et Kurtz, *Back to Basic* (publié à l'automne 1984 par Addison et Wesley) raconte l'histoire de ce langage, de l'une à l'autre des deux versions écrites à vingt ans d'intervalle par l'équipe de Dartmouth College.

Bien sûr, True Basic est loin d'être le seul dans son genre : le Basic 2.0 par exemple, pour Macintosh, lui ressemble comme un frère, et il existe des versions semblables, parfois même écrites par des amateurs enthousiastes et doués pour tel club d'utilisateurs. Pourtant, l'avenir de Basic est aujourd'hui peu clair. Sans aucun doute ses descendants musclés lui donneront un bon bol d'oxygène ; mais il est permis de croire que son règne sur la micro-informatique approche de son terme.

Le « Beginner's All Purpose » de demain sera plus vraisemblablement une nouvelle construction largement inspirée de Pascal (rendu moins raide), mâtinée d'un peu de Forth ou d'un autre avatar de Lisp, surtout si l'intelligence artificielle sort de son ghetto encore très professionnel. Mais le grand nombre de machines bon marché existantes, le coup de fouet donné (par Gates interposé) avec le Basic A adopté par IBM sur sa machine aujourd'hui omniprésente font évidemment que la mort de Basic n'est peut-être pas pour demain ! Encore quelques petites années, monsieur le bourreau... Au moins pour bons services rendus, Basic mérite bien cela !

André WARUSFEL



LES EMPRUNTS SE PRÊTENT... AUX CALCULS

AVANT de s'engager, parfois pour plusieurs années, à rembourser un prêt, on a tout intérêt à envisager différentes solutions. A la main, cela peut prendre des heures, de quoi renoncer souvent. Quelques minutes suffisent avec un ordinateur.

Les renseignements dont on a besoin pour décider d'un prêt personnel accordé par une banque (« prêt personnel ordinaire », prêt au titre du Plan Épargne-Logement ou autres) sont en fin de compte peu nombreux. Si l'on confie à un petit ordinateur la tâche d'effectuer les calculs, on peut facilement, et sans perdre de temps, choisir la formule qui convient le mieux.

Un remboursement anticipé

Le programme proposé (ici dans une version pour Spectrum et Spectrum + avec imprimante, mais l'adaptation à d'autres matériels est très simple) permet notamment de connaître le coût et les conditions d'un remboursement anticipé.

D'une façon générale, un prêt se définit par :

- le montant du prêt ;
- le taux annuel consenti ;
- l'une des deux conditions qui sont soit la durée du prêt, soit le montant de la mensualité régulière choisie.

L'avant-dernière condition (durée du prêt) est le plus souvent imposée par l'organisme prêteur qui exige évidemment un nombre entier de mois et quelquefois même un nombre entier d'années.

Quant à la dernière condition (montant de la mensualité régulière choisie), elle s'impose d'elle-même à l'emprunteur puisqu'elle découle de ses possibilités de financement. Dans ce cas, celui-ci se fixe une mensualité régulière maximum. Celle-ci, décidée sans rapport avec les conditions du prêt, détermine immanquablement un nombre fractionnaire de mois pour assurer le remboursement, ce qui n'est pas admissible. Le programme calcule alors quelle est la mensualité la plus proche pouvant être compatible avec un nombre entier de mois.

Il faut enfin envisager le cas du remboursement anticipé dont les conditions peuvent être avantageuses ou pas. Il est accordé par l'organisme prêteur contre le paiement d'une pénalité de rachat qui dépend du nombre de mois à courir. Cette pénalité sera calculée à partir d'un pourcentage sur la somme restant due (3 % en général) ou de l'intérêt perçu par le prêteur (le total des intérêts des six mois suivants). Le problème se pose presque toujours de cette façon bien que

LES EMPRUNTS SE PRÊTENT... AUX CALCULS

Emprunt à remboursement mensuel
Programme pour Spectrum
(ou Spectrum Plus) et
imprimante ZX Printer ou Alphacom 32
Auteur Lucien Strebler


```
  5 PRINT AT 1,0;"EMPRUNT A REM
BOURSEMENT MENSUEL": PLOT 0,157:
 DRAW 248,0: PRINT AT 4,2;"Somme
 empruntee : ": INPUT s: PRINT A
T 4,20;s
  10 PRINT AT 6,2;"Taux de l'emp
runt en %: ": INPUT ta: PRINT AT
 6,26;ta: LET tm=ta/1200
  15 PRINT AT 9,3;"Choisir :";AT
 11,12;"1/ Mensualite";AT 13,15;
"ou";AT 15,12;"2/ Duree du pret"
: INPUT "Numero choisi ? ";no
  20 IF no=1 THEN GO SUB 220: GO
 TO 40
  25 IF no=2 THEN INPUT "Duree e
n mois ? ";ni: GO SUB 220: PRINT
 AT 12,2;"Duree du pret : ";ni:
GO TO 35
  30 GO TO 15
  35 LET mc=s/ni*(1+(ni+1)*ta/24
00): PRINT AT 10,2;"Mensualite r
equise : ": GO SUB 215: LET n=ni
: PRINT INVERSE 1;AT 14,11;"PATI
ENCE": GO TO 100
  40 PRINT AT 10,2;"Mensualite c
hoisie : "
  45 INPUT mc: PRINT AT 10,24;mc
;AT 14,11; INVERSE 1;"PATIENCE"
  98 GO SUB 215
 100 GO SUB 200
 105 IF Rd<0 THEN LET m=n: PRINT
 AT 12,2;"Nombre de mensualites
: ";m
 106 IF Rd<0 THEN : IF ta*m<1500
 THEN LET u=1: GO TO 120
 107 IF Rd<0 THEN : IF ta*m<2820
 THEN LET u=2: GO TO 120
 108 IF Rd<0 THEN LET u=3: GO TO
 120
 110 LET n=n+1: GO TO 100
 120 LET mc=mc+Rd/m/u: PRINT AT
10,24;"      ";AT 10,24;INT (mc+.
5)
 125 GO SUB 215
 130 FOR K=1 TO m: GO SUB 200: L
ET n=n+1: NEXT K
 135 IF ABS Rd<1 THEN PRINT AT 1
4,2;"Cout de l'operation : ";INT
 (I+.5): BEEP 3,0: COPY : GO TO
145
 140 GO TO 120
 145 GO SUB 215: LPRINT TAB 0;"M
";TAB 4;"Rb.TOT";TAB 13;"R.DU";T
AB 20;"INT";TAB 26;"I.TOT": LPRI
NT
 150 FOR j=1 TO m: GO SUB 200: G
O SUB 205: LET n=n+1: NEXT j: PR
INT AT 18,0;"(Details par l'impr
imante A 32)": STOP
 200 LET I=I+Rd*tm: LET Rp=mc*n-
I: LET Rd=s-Rp: RETURN
 205 LPRINT TAB 0;n;TAB 4;INT (R
p+.5);TAB 12;INT (Rd+.5);TAB 20;
INT (Rd*tm+.5);TAB 26;INT (I+.5)
: RETURN
 215 LET Rd=s: LET n=1: LET I=0:
 RETURN
 220 PRINT AT 9,3;"          ";AT
 11,12;"          ";AT 13,15;"
   ";AT 15,12;"
: RETURN
```

On pourra bien entendu utiliser le programme sans imprimante au prix de quelques modifications faciles à apporter.

Ligne 120 : ... AT 10,24 ; "5 intervalles"
Ligne 220 : ... AT 9,3 ; "9 intervalles"... AT 11,12 ; "13 intervalles" ... AT 13,15 ; "2 intervalles" ... AT 15,12 ; "16 intervalles"

ces paramètres puissent varier. Dans tous les cas, le montant le plus faible est retenu.

Le programme ci-contre va permettre, d'une part, de connaître le montant de la mensualité ou la durée du prêt et, d'autre part, d'avoir le détail mensuel de l'opération. Il sera intéressant, dans ce dernier cas, de conserver une trace écrite des calculs ; l'imprimante devra alors avoir été préalablement branchée.

On introduit d'abord le montant du prêt et le taux de l'emprunt en pourcentage, puis, soit le montant d'une mensualité, soit la durée en mois.

Un message d'attente fait patienter l'utilisateur car le calcul est parfois long, surtout s'il s'agit de prêts portant sur quinze ou vingt ans ; l'affichage du résultat est précédé d'une sonnerie indiquant que le calcul est terminé. La réponse (le versement mensuel ou le nombre de mois) apparaît à l'écran, ainsi que le coût total de l'opération.

Tous les renseignements utiles sont donnés, mis à part les éléments du remboursement anticipé. Pour les connaître, l'imprimante devra avoir été branchée. On obtient alors une copie de l'écran tel qu'il était précédemment, ainsi que le tableau donnant, mois par mois, les remboursements effectués, le solde dû, les intérêts de chaque versement et le total des intérêts payés. Il est ainsi facile de calculer, à n'importe quel moment, le montant des six mois d'intérêt à venir et les 3 % de la somme qui reste due, afin de savoir si, oui ou non, il est intéressant de faire un remboursement anticipé.

```
EMPRUNT A REMBOURSEMENT MENSUEL

  Somme empruntee : 100000
  Taux de l'emprunt en %: 5.31

  Mensualite choisie :  4401
  Nombre de mensualites : 24
  Cout de l'operation : 5625

M    Rb.TOT    R.DU     INT     I.TOT
1    3959      96041    425     443
2    7935      92065    407     867
3    11928     88072    390     1275
4    15939     84061    372     1665
5    19969     80031    354     2037
6    24015     75985    336     2391
7    28080     71920    318     2727
8    32163     67837    300     3045
9    36264     63736    282     3345
10   40383     59617    264     3627
11   44520     55480    245     3891
12   48676     51324    227     4137
13   52849     47151    209     4364
14   57042     42958    190     4572
15   61253     38747    171     4763
16   65482     34518    153     4934
17   69731     30269    134     5087
18   73998     26002    115     5221
19   78284     21716    96      5336
20   82589     17411    77      5432
21   86913     13087    58      5509
22   91256     8744     39      5567
23   95618     4382     19      5605
24   100000    0        0       5625
```

Affectation des variables

S	Somme empruntée
Ta	Taux annuel en %
tm	Taux mensuel décimal
no	Numéro de l'option
ni	Nombre de mois choisis
n	Nombre de mois utilisé par les boucles
m	Nombre de mois calculés
Rd	Somme restant due
mc	Mensualité
k	Boucle FOR... NEXT
j	Boucle FOR... NEXT
Rp	Remboursement en principal
I	Intérêt total
u	Paramètre pondérant l'influence de la somme restant due sur la mensualité

Les détails dans un tableau

Prenons un exemple : pour un emprunt de 100 000 francs, remboursable sur 24 mois avec un taux de 5,31 %, le programme indiquera un versement mensuel de 4 401 francs (coût total : 5 625 francs). En partant d'une mensualité de 4 500 francs, on retrouvera 24 mois et 5 625 francs ; dans ce cas, la mensualité sera automatiquement ajustée à 4 401 francs. Avec une somme très différente, il serait possible de trouver un résultat exact mais avec un nombre de mois différent. L'impression du tableau donnant le détail mensuel permettrait de juger de l'opportunité d'un remboursement anticipé. Dans cet exemple et en choisissant le quatorzième mois comme date de rachat, il coûterait 859 francs (la somme des intérêts des six mois suivants, de 14 à 19) ou 1 288,74 francs (3 % du capital restant dû). Le montant le plus faible étant retenu dans

tous les cas, la pénalisation se monterait donc à 859 francs.

Tel qu'il est conçu, ce programme ne tourne que si le produit du nombre de mois par le taux d'intérêt ne dépasse par 3 500. Ceci permet de faire les calculs dans le minimum de temps possible tout en couvrant la quasi-totalité des cas, le taux d'intérêt maximum pratiqué actuellement étant de 18,25 % (3 500 divisé par 18,25 % donne 192 mois, c'est-à-dire 16 ans).

D'un Basic à l'autre

Pour l'essentiel, ce sont les instructions d'affichage PRINT AT et PRINT TAB qui risquent de devoir être modifiées sur d'autres machines que le Spectrum.

A vérifier aussi la syntaxe de la fonction ABS (ligne 135) : l'argument doit-il ou non être placé entre parenthèses ?

Pour le reste, l'adaptation va de soi. A noter que les noms de variables utilisent deux caractères significatifs.

Pour effectuer des calculs avec un taux d'intérêt plus élevé en choisissant une période de remboursement plus longue, il suffira de modifier la ligne 108 en remplaçant LET u = 3 par LET u = 7. Le programme tournera tout aussi bien pour les valeurs plus courantes mais cela se fera au détriment des temps d'exécution.

Lucien STREBLER

PROGRAMME SUR APPLE II

UN TRACEUR DE COURBES

Un traceur de courbes paraît être un programme peu original, connaissant les possibilités graphiques de l'Apple II. Mais il va nous aider à mieux comprendre l'organisation d'un programme Basic et son implantation en mémoire. Et surtout, il s'automodifie, ce qui le rend particulièrement intéressant.

Quand j'ai conçu la première version — simplifiée — d'un traceur de courbes, le programme opérait de la manière suivante : après l'affichage du titre, il était arrêté par une instruction STOP à la ligne 100. Il demandait alors à l'utilisateur d'entrer, en mode direct, la ligne 100 suivie de la définition de la fonction, puis de reprendre l'exécution à partir de cette ligne 100. Il fallait donc faire, par exemple :

```
100 DEF FNA(X)=EXP(SIN(X))
RUN 100
```

Deux ans plus tard...

Le programme s'exécutait alors de la même manière que la version proposée ici. Après l'avoir laissé dormir au fond d'une disquette pendant deux longues années, j'ai eu envie de le reprendre. J'ai voulu alors supprimer les manipulations décrites ci-dessus et saisir la fonction dans le corps même du programme. Ceci impliquait donc que le programme s'automodifie. Restait à découvrir comment y arriver.

Au départ, il me manquait deux informations que je pouvais trouver dans la documentation :

- l'organisation du texte d'un programme par l'Apple ;
- le codage des mots clés par l'interpréteur Applesoft.

En ce qui concerne le premier point, si LOMEM (adresse basse de la mémoire) est fixée à sa valeur par défaut, l'implantation d'un programme Applesoft (attention, ceci n'est pas vrai pour l'INTEGER) commence à l'adresse 2049 ($801). Chaque ligne est alors codée comme suit :

- deux octets formant l'adresse du début de la ligne suivante, codés dans le format 6502 (bas-haut) ;
- deux octets formant le numéro de la ligne courante toujours dans le même format ;
- la ligne d'instructions Basic est elle-même codée, les mots clés l'étant sur un

UN TRACEUR DE COURBES

octet (par exemple, END est codé par $80), les autres caractères l'étant de façon normale, c'est-à-dire avec le code ASCII en hexadécimal ;

• enfin, un octet 00 signale la fin de ligne (de la même façon, un double octet nul, 00 00, signale la fin de la zone programme).

Avant que le programme ne modifie la ligne 5, voyons comment elle se présente (voir sa représentation, ci-dessous).

3A est le code des deux-points (:). Ils occupent donc toutes les cases mémoire de $820 à $852. Pour que le programme s'automodifie, il faut qu'il décrive à l'emplacement correct (aux adresses où le texte de la ligne est codé) ce qu'il veut y mettre. Encore faut-il savoir quoi y mettre. Soit pour le texte « normal » (les noms de variables et les valeurs numériques), les codes ASCII de tous les caractères, mais aussi les codes des mots clés de l'Applesoft.

Notons tout d'abord que tous les mots clés n'interviennent pas dans la définition d'une fonction. (Pour l'analyse du programme, voir l'encadré page suivante).

Comment se servir du traceur

Il reste à voir comment se servir de ce traceur de courbes. Après l'affichage du titre, le programme demande la fonction à tracer. Entrez-la en respectant la syntaxe Basic. Donnez ensuite les bornes inférieures et supérieures suivant X et Y. Enfin, entrez le pas de trace, c'est-à-dire le pas suivant X pour le calcul des points.

La définition d'une fenêtre vous permet d'obtenir un agrandissement d'une portion de la courbe, fenêtre que vous définirez avec une manette de jeux en choisissant deux coins opposés d'un rectangle.

Il est possible d'améliorer ce programme en permettant par exemple la recopie d'écran graphique.

Gérald ANFOSSI

Traceur de courbes
Programme pour Apple II
Auteur Gérald Anfossi


```
1  GOSUB 2000
2  GOSUB 1000
5  DEF  FN A(X) = ::::::::::::::
     ::::::::::::::::::::::::::::
     :::::::
100  REM
                50 ":" APRES LE "="

101  REM
     NE RIEN MODIFIER AVANT 100


112  HTAB 4: VTAB 12: PRINT "ENTR
     EZ VOTRE FONCTION": PRINT
120  VTAB 16: HTAB 4: CALL  - 868

130  INPUT "MIN X>>";BX
135  VTAB 16: HTAB 20
140  INPUT "MAX X>>";HX
141  IF BX >  = HX THEN 120
145  VTAB 13: HTAB 4: CALL  - 868

150  INPUT "MIN Y>>";BY
155  HTAB 20: VTAB 13
160  INPUT "MAX Y>>";HY
161  IF BY >  = HY THEN 145
165  VTAB 20: HTAB 4: CALL  - 868

170  INPUT "DONNEZ LE PAS SUIVANT
      X >";PAS
175  IF PAS <  = 0 THEN 165
180 IX = 279 / (HX - BX):IY = 159
     / (HY - BY)

190  HGR : HCOLOR= 3
191  HOME : INVERSE : VTAB 20:A$ =
     "<PRESSEZ UN BOUTON POUR ARR
     ETER.>": GOSUB 10100:A$ = "<
      PRESSEZ <ESC> POUR TERMINER
     . >": VTAB 21: GOSUB 10100:
     NORMAL : POKE 34,22
195  REM
                PLACEMENT DES AXES

200  IF HX * BX <  = 0 THEN  HPLOT
      - BX * IX,0 TO  - BX * IX,1
     59
210  IF HY * BY <  = 0 THEN  HPLOT
     0,HY * IY TO 279,HY * IY
219  REM
                TRACE LA FONCTION

220  FOR Z = BX TO HX STEP PAS
221  IF  PEEK ( - 16287) > 127 OR
      PEEK ( - 16286) > 127 THEN
     Z = HX: FOR T = 1 TO 500: NEXT
     : GOTO 300

222 Q =  PEEK ( - 16384): POKE  -
     16368,0: IF Q = 27 THEN 2000
     0
230  ONERR  GOTO 280
240 A = (Z - BX) * IX:B = 159 + (
     BY -  FN A(Z)) * IY
250  IF A < 0 OR A > 279 OR B < 0
      OR B > 159 THEN 275
260  HPLOT A,B
275  HOME : VTAB 23: PRINT "X="Z;
     : HTAB 20: PRINT "F(X)=" FN
     A(Z)
280  POKE 216,0: NEXT : GOTO 300
300  POKE 216,0: POKE 34,0: HOME
     : VTAB 21: INVERSE : FOR I =
     1 TO 40: PRINT ":";: NEXT :
     NORMAL : VTAB 22: HTAB 17:
     PRINT "FENETR
     E": INVERSE : FOR I = 1 TO 4
     0: PRINT ":";: NEXT : POKE 3
     4,22
310  GOSUB 5000
320 RX = P1:RY = P2
325  DRAW 1 AT RX,RY
326  FOR I = 1 TO 500: NEXT
330  GOSUB 5000
350 CX = P1:CY = P2
355  DRAW 1 AT CX,CY
359  HCOLOR= 3
380  IF CX > RX THEN X2 = RX:X1 =
     CX: GOTO 400
390 X2 = CX:X1 = RX
400  IF CY > RY THEN Y1 = CY:Y2 =
     RY: GOTO 420
410 Y1 = RY:Y2 = CY
415  REM
          DESSIN DE LA FENETRE

420  HPLOT X1,Y1 TO X1,Y2: HPLOT
     X1,Y2 TO X2,Y2: HPLOT X2,Y2 TO
     X2,Y1: HPLOT X2,Y1 TO X1,Y1
421 BY = (BY + (159 - Y1) / IY):H
     Y = (BY + (159 - Y2) / IY)
430 HX = BX + X1 / IX:BX = BX + X
     2 / IX
440  HOME : VTAB 21: HTAB 4: GOTO
     170
1000  REM
               CREE LA FONCTION

1120  HTAB 4: VTAB 12: PRINT "ENT
     REZ VOTRE FONCTION": PRINT
1130  HTAB 4: INPUT "DEF FNA(X)="
     ;A$
1140 HEX$ = "820:"
1150  FOR L = 1 TO  LEN (A$)
```

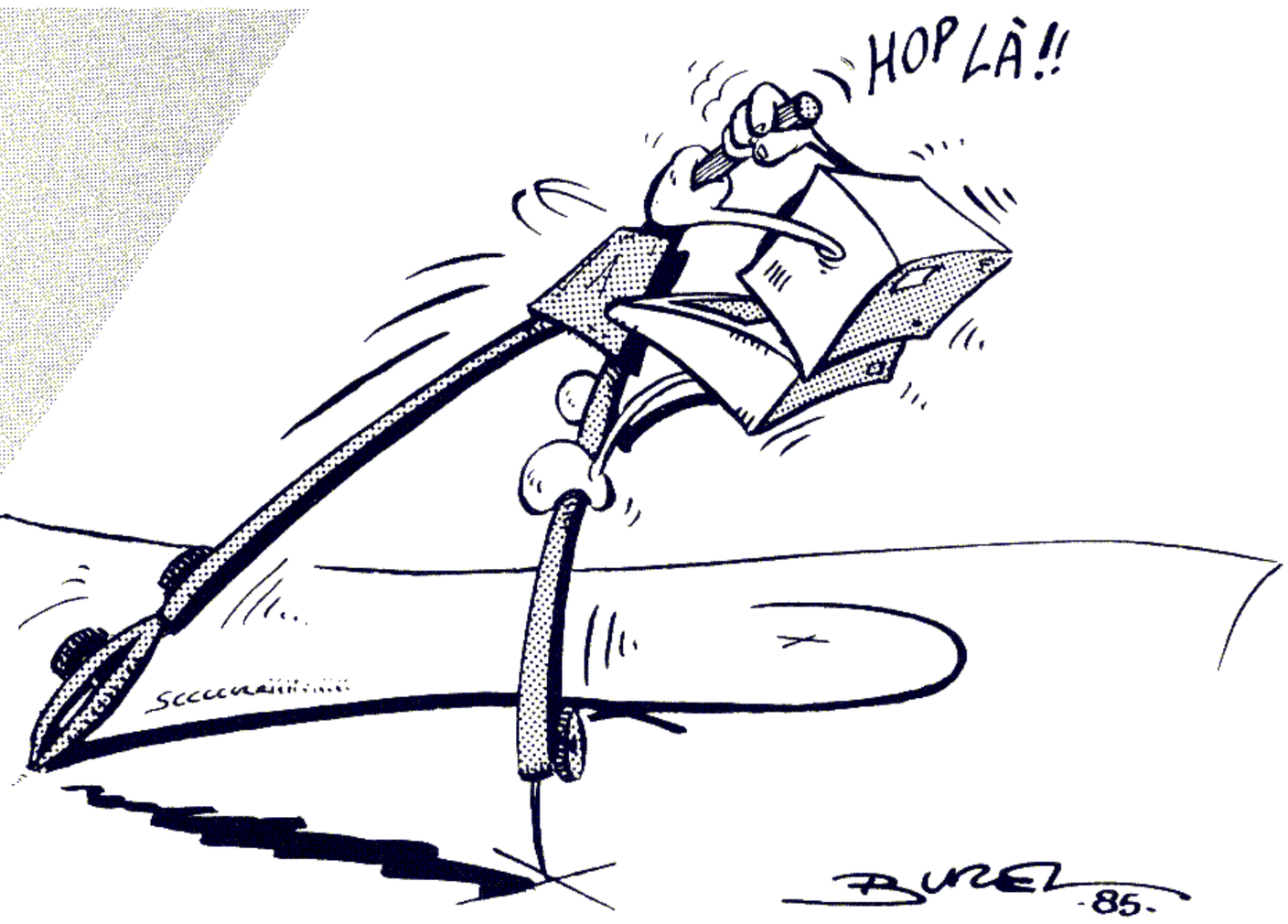
Représentation de la ligne 5 avant modification

```
0814:00 [53 08] 05
0818:00 B8  C2 41  28  58  29 D0
0820:3A 3A 3A 3A 3A 3A 3A 3A
0828:3A 3A 3A 3A 3A 3A 3A 3A
0830:3A 3A 3A 3A 3A 3A 3A 3A
0838:3A 3A 3A 3A 3A 3A 3A 3A
0840:3A 3A 3A 3A 3A 3A 3A 3A
0848:3A 3A 3A 3A 3A 3A 3A 3A
0850:3A 3A 00
```

$853 adresse ligne 100, ligne 5
B8 = DEF ; C2 = FN ; 41 = A ; 28 = (; 29 =)
58 = X ; D0 = " = "

3A = ":"

00 = pointeur de fin de ligne




```
1160  FOR H = 1 TO 61
1170  IF  MID$ (A$,L, LEN (EQU$(H
      ,1))) <  > EQU$(H,1) THEN 12
      00
1180 HEX$ = HEX$ + EQU$(H,2) +  CHR$
      (32)
1190 L = L +  LEN (EQU$(H,1)) - 1

1195 H = 61
1200  NEXT H
1210  NEXT L
1220 HEX$ = HEX$ + " N D9C6G"
1300  GOSUB 20100
1310  REM
       VA MODIFIER LA LIGNE 5

1999  RETURN
2000  REM
        MOTS-CLES & LEURS CODES


2001  GOSUB 10000
2005  DIM EQU$(61,2)
2010  DATA +,C8,-,C9,*,CA,/,CB,^,
      CC,AND,CD,OR,CE,>,CF,=,D0,<,
      D1,SGN,D2,INT,D3,ABS,D4,SQR,
      D5,RND,D6,LOG,D7,EXP,D8,COS,
      D9,SIN,DA,TAN,DB,ATN,E1,(,28
      ,),29,,,2E," ",20
2020  FOR I = 1 TO 25: READ EQU$(
      I,1),EQU$(I,2): NEXT I
2030  REM
        CHIFFRES & LEURS CODES


2040  FOR I = 0 TO 9:EQU$(I + 26,
      1) =  STR$ (I):EQU$(I + 26,2
      ) =  STR$ (30 + I): NEXT I
2045  DATA 41,42,43,44,45,46,47,4
      8,49,4A,4B,4C,4D,4E,4F,50,51
      ,52,53,54,55,56,57,58,59,5A

2050  FOR I = 1 TO 26:EQU$(35 + I
      ,1) =  CHR$ (64 + I): READ E
      QU$(35 + I,2): NEXT I
3000  POKE 232,192: POKE 233,3: FOR
      I = 960 TO 970: READ Z: POKE
      I,Z: NEXT I: DATA 1,0,4,0,54
      ,196,111,32,86,53,0
3010  SCALE= 1: ROT= 0
3020  RETURN
4999  REM
        LECTURE DE LA FENETRE
5000 P1 =  PDL (0) * (279 / 255):
      P2 =  PDL (1) * (159 / 255)
5020  VTAB 23: PRINT "X="DX + P1 /
      IX;: HTAB 20: PRINT "Y=";(BY
       + (159 - P2) / IY)
5030  XDRAW 1 AT P1,P2: IF  PEEK
      ( - 16287) < 128 AND  PEEK (
       - 16286) < 128 THEN  XDRAW
      1 AT P1,P2: GOTO 5000
5040  RETURN
10000  REM
             TITRE
10001  TEXT : HOME : INVERSE
```

```
10002 VT = 10:DB = 1:FIN = 39: GOSUB
      10015
10003 VT = 1:DB = 8:FIN = 23: GOSUB
      10015
10004 VT = 23:DB = 1:FIN = 39: GOSUB
      10015
10010  GOTO 10025
10015  VTAB VT: HTAB DB
10020  FOR I = 0 TO FIN: PRINT "
      ";: NEXT I
10021  RETURN
10025 DB = 1:FIN = 10:T1 = 8:T2 =
      31: GOSUB 10030
10027  NORMAL : GOTO 10040
10030  FOR I = DB TO FIN: VTAB I:
       HTAB T1: PRINT " ";: HTAB T
      2: PRINT " ": NEXT
10035  RETURN
10040  VTAB 2:A$ = "TRACEUR DE CO
      URBE": GOSUB 10100
10050 A$ = "PAR G. ANFOSSI": GOSUB
      10100
10060 A$ = "(C) LIST ET L'AUTEUR"
      : GOSUB 10100
10070  RETURN
10100  PRINT : HTAB 20 -  LEN (A$
      ) / 2: PRINT A$: RETURN
20000  REM
             SORTIE DU TRACEUR ?
20010  TEXT : HOME
20020  PRINT "VOULEZ-VOUS :"
20030  VTAB 10: PRINT "1)UNE AUTR
      E FONCTION": VTAB 15: PRINT
      "2)TERMINER"

20040  HTAB 1: VTAB 22: PRINT "EN
      TREZ VOTRE CHOIX :";: CALL  -
      868: GET S$
20050  IF  VAL (S$) < 1 OR  VAL (
      S$) > 2 THEN 20040
20060  IF  VAL (S$) = 2 THEN  END

20070 HEX$ = "820:": FOR I = 1 TO
      50:HEX$ = HEX$ + "3A ": NEXT
      :HEX$ = HEX$ + "N D9C6G"
20080  GOSUB 20100: RUN
20090  REM
         MODIFIE LA LIGNE 5


20100  FOR I = 1 TO  LEN (HEX$)
20110  POKE 511 + I, ASC ( MID$ (
      HEX$,I,1)) + 128
20120  NEXT
20130  POKE 72,0: CALL  - 144
20140  RETURN
```

L'analyse du programme

Les lignes 120 à 440 forment le corps du programme : le traceur de courbes. La ligne 5 est celle qui sera modifiée. Au départ, elle contient l'amorce d'une définition de fonction, le signe égal (=) étant suivi de cinquante signes deux-points (:) qui réservent la place de la fonction.

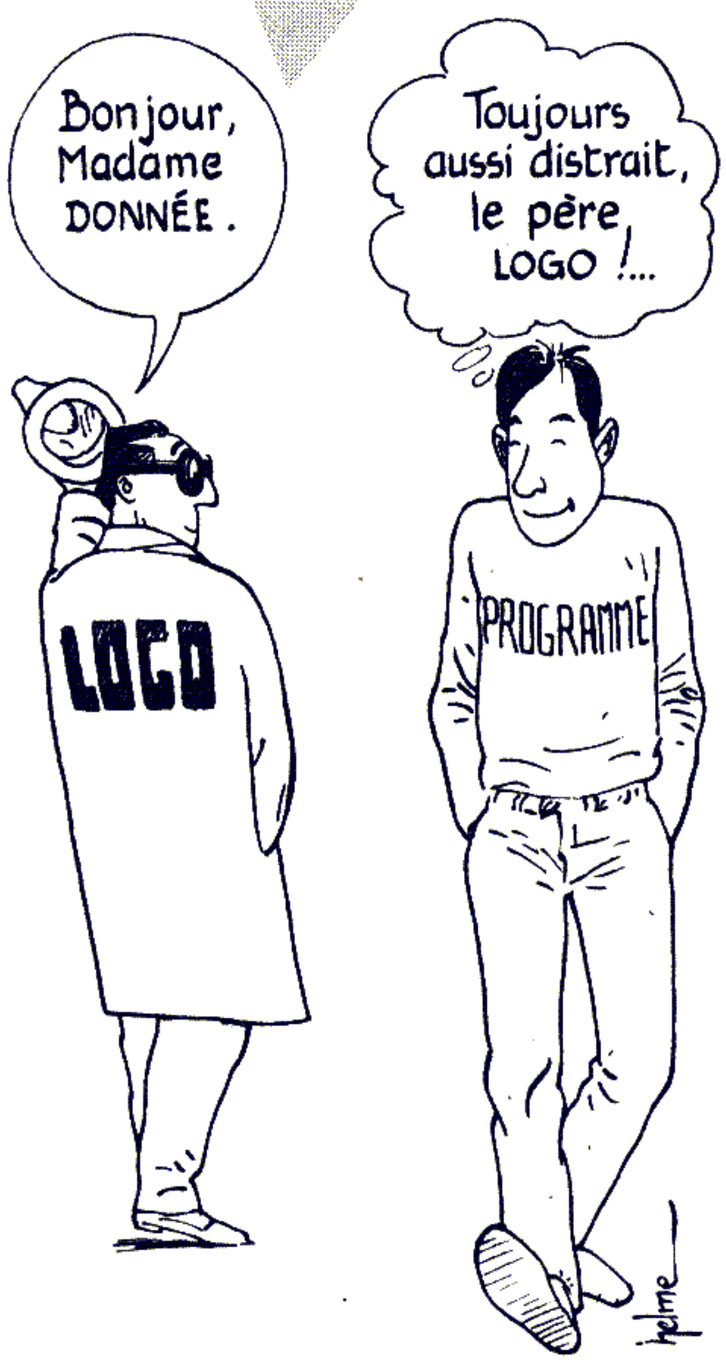
De la ligne 2005 à la ligne 2050, on range dans un tableau EQU$ les mots clés et leurs codes. Ils serviront plus tard à l'analyse de la fonction et au codage de la nouvelle ligne. Dans ce tableau, on range aussi tous les chiffres et les lettres (ainsi que leurs codes) qui peuvent intervenir dans la définition de la fonction. Cette dernière est saisie par l'INPUT de la ligne 1130.

On construit alors une chaîne HEX$ équivalente à la suite de caractères que l'on entrerait à partir du moniteur, en mode immédiat, pour écrire la définition de la fonction en se substituant à l'interpréteur Applesoft (lignes 1140 à 1220). La chaîne HEX$ commence par « 820: » car $820 est l'adresse qui suit le signe égal dans la ligne 5. Elle se termine par l'instruction D9C6G qui force un retour au Basic (ligne 1220).

Les lignes 20100 à 20120 forment une boucle qui écrit notre chaîne hexadécimale dans le buffer clavier (adresse $200, soit 512 en décimal). Le CALL −144 entraîne la prise en compte de ce qui se trouve dans le buffer, et notre ligne 5 vient d'être modifiée.

LE MODE D'APPRENTISSAGE

CONTRAIREMENT à beaucoup d'autres langages, Logo ne fait aucune distinction entre programmes et données. C'est pourquoi une procédure peut devenir un objet, peut être modifiée ou même créée par une autre procédure. Nous décrirons aujourd'hui un ensemble de procédures destinées à en créer d'autres.



Logo, comme tous les « langages de liste » dérivés de Lisp (List Processing), permet des applications de type intelligence artificielle. Contrairement aux langages de programmation classiques, comme Basic ou Pascal par exemple, Logo ne fait pas de distinction entre programmes et données. L'astuce consiste simplement à présenter les procédures sous forme de listes.

La primitive POUR permet de définir des procédures classiques dont la structure peut être imitée grâce au GOSUB du Basic, avec quelques difficultés pour les variables locales. En revanche, la primitive DEFINIS permet de définir une procédure comme une liste de listes. Ainsi, pour calculer un nombre N à la puissance P, on peut s'y prendre d'au moins deux manières selon que l'on utilise les primitives POUR ou DEFINIS.

```
POUR PUISSANCE :N :P :R
SI :P = 0 [RETOURNE :R]
RETOURNE PUISSANCE :N :P – 1 :R*:N
FIN
```

Ou bien, seconde façon de faire :

```
DEFINIS "PUISSANCE [[N P R][SI :P = 0
  [RETOURNE :R][RETOURNE
  PUISSANCE :N :P – 1 :R* :N]]
```

On remarque donc, avec DEFINIS, que :

- la première liste est celle des paramètres N P R dans laquelle le caractère deux-points (:) a disparu ;
- chaque instruction de la procédure est une liste ;
- le mot FIN n'apparaît plus.

La liste des paramètres est obligatoire. S'il n'y a aucun paramètre, elle doit apparaître tout de même, mais vide. En voici un exemple : DEFINIS "TRIANGLE [[][REPETE 3 [AV 50 DR 120]]].

La primitive TEXTE permet de transformer une procédure écrite avec POUR en une liste. Il n'est donc pas indispensable de choisir DEFINIS pour créer des procédures modifiables par procédure. De plus, TEXTE est particulièrement utile pour simuler un mode « pas à pas » que nous verrons une autre fois.

Faire évoluer une procédure suppose que l'on en connaisse le nom. Les nouvelles versions de Logo pour les microprocesseurs 16 bits contournent cette

difficulté grâce à la primitive LISTEPROC qui retourne la liste de toutes les procédures présentes dans l'espace de travail ; LISTEPROC permet donc de travailler sur ces dernières sans connaître leur nom.

A elles seules, les trois primitives DEFINIS, TEXTE et LISTEPROC ouvrent Logo à la plupart des applications de l'intelligence artificielle.

Un troisième mode pour Logo

Logo travaille habituellement en deux modes : soit le mode direct (appelé parfois « pilotage en graphique »), soit le mode procédural ou de programmation.

En mode direct, les instructions sont immédiatement exécutées au coup par coup. C'est le mode employé pour explorer les divers micromondes. Ce mode permet bien des découvertes, mais il présente l'inconvénient sérieux de ne laisser aucune trace des différents essais.

En mode programmation au contraire, les instructions sont conservées, mais l'exécution n'est pas immédiate, elle est différée. Le résultat n'est visible que lorsque la procédure est terminée.

Pour réunir les avantages de ces deux modes, nous allons en définir un troisième dans lequel les instructions seront non seulement exécutées en mode direct, mais aussi mémorisées pour être ensuite incluses dans une procédure à laquelle on donnera un nom.

Pour parvenir à ce résultat, il nous faut :

1 - Entrer les instructions, les vérifier et les exécuter.

2 - Conserver les instructions.

3 - Pouvoir rectifier un essai en annulant l'instruction précédente.

4 - Nommer la procédure nouvellement créée en vérifiant qu'elle n'existe pas déjà.

Pour travailler dans ce mode, que nous appellerons mode d'apprentissage, nous créerons trois mots. Le mot APPRENDS, d'abord, initialise le mode. Le mot FIN termine le mode. Le mot GOMME fait oublier la dernière instruction exécutée.

Première tâche : entrer les instructions et les exécuter.

```
POUR ENTRER
DONNE "INST LISLISTE
SI :INST = [FIN][STOP]
SI :INST = [GOMME][GOMME]
  [EXECUTE :L DONNE
  "PROC PH :PROC LISTE :L]
```

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.

```
ENTRER
FIN
```

La variable "PROC, globale, a été initialisée à [[]], et elle accumule les instructions retenues après exécution. Pour vérifier les instructions, nous définirons une procédure VERIFIER chargée « d'attraper » les erreurs :

```
POUR VERIFIER
ATTRAPE "ERREUR [SUITE]
EC [INSTRUCTION NON VALIDE]
VE
EXECL :PROC
VERIFIER
FIN
```

Si une erreur intervient, la procédure VERIFIER s'exécute. Le marqueur d'erreur est alors à VRAI et la procédure SUITE est ignorée. On affiche le message INSTRUCTION NON VALIDE ; puis VE, EXECL :PROC sont exécutés. Le marqueur d'erreur est automatiquement remis à FAUX. VERIFIER étant récursif, la réexécution de la séquence ATTRAPE "ERREUR [SUITE] entraînera l'appel de la procédure SUITE, qui est en fait la procédure principale. Quant à la procédure EXECL, elle a été créée pour exécuter une liste de listes car dans certaines versions de Logo, EXECUTE n'exécute qu'une liste simple. Écrivons les procédures SUITE et EXECL :

```
POUR SUITE
ENTRER
FINIR
FIN

POUR EXECL :L
SI :L = [] [STOP]
EXECUTE PREMIER :L
EXECL SP :L
FIN
```

Il ne nous reste plus qu'à écrire la procédure FINIR et la procédure GOMME, puis à les enchaîner au moyen de la procédure APPRENDS.

```
POUR FINIR
EC [VOULEZ-VOUS GARDER VOTRE
  TRAVAIL ?]
EC [SI OUI, TAPEZ SON NOM]
DONNE "N LISLISTE
SI :N = [][RENVOIE "NIVEAUSUP]
SI DEFINIP PREMIER :N [EC[CE NOM
  EXISTE DEJA] FINIR]
DEFINIS PREMIER :N :PROC
(EC PREMIER :N [EST DEFINIE])
RENVOIE "NIVEAUSUP
FIN
```

Une remarque sur cette dernière procédure. Nous aurions pu écrire, pour simplifier l'expression, DONNE "N PREMIER LISLISTE. Cependant, certains Logo signalent une erreur lorsque la liste spécifiée est vide. Or nous voulons qu'elle soit effectivement vide lorsque l'utilisateur ne veut pas conserver son travail. RENVOIE "NIVEAUSUP permet de revenir en mode direct sans passer par tous les niveaux de récursion.

```
POUR GOMME
EC PH [J'ANNULE L'INSTRUCTION :]
  DERNIER :PROC
DONNE "PROC SD :PROC
VE
EXEC :PROC
FIN
```

Et enfin

```
POUR APPRENDS
VE
DONNE "PROC [[]]
VERIFIER
FIN
```

Utile pour les débutants

L'écriture de ces procédures nous permet de travailler maintenant selon le troisième mode, c'est-à-dire d'exécuter des instructions en mode direct et, au vu des résultats, de conserver ou non ces instructions pour les mémoriser à la fin dans une procédure.

Si vous possédez un Logo qui tourne sur un matériel Thomson, ce n'est pas de chance. Ne vous faites aucune illusion : les concepteurs de ce langage ont simplement oublié (peut-être faute de place ?) la primitive DEFINIS. Quel dommage ! Ces Logo, dans leur état actuel, ne permettent pas de définir de procédures du style « intelligence artificielle ». Pour les autres, ATTRAPE se traduit par ENERREUR, DONNE par CREE, RELIE ou FIXE, RENVOIE "NIVEAUSUP par LOGO et DEFINIP par DEFINI?.

En conclusion, le mode APPRENDS est un mode très utile, même et surtout pour les débutants. Un autre mode utile est le PAS A PAS qui visualise l'exécution d'une procédure, instruction par instruction. Malheureusement, il n'est pas implanté sur tous les Logo. Nous verrons comment remédier à ce défaut dans un prochain article.

Robert DAGUESSE

LE BASIC DU PX-8

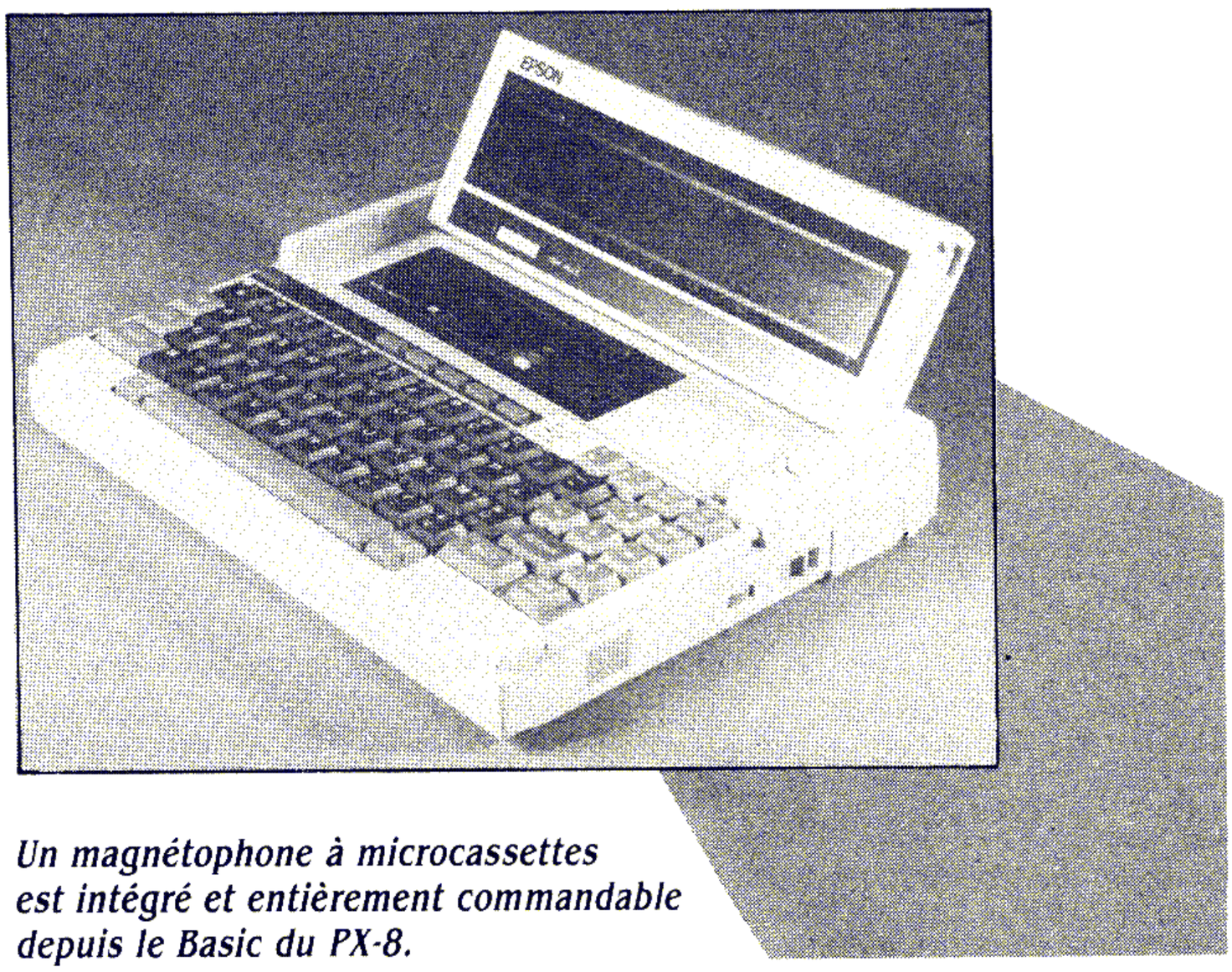
L'EPSON PX-8 est un ordinateur séduisant : vraiment portatif, doté de logiciels puissants (dont le fameux traitement de texte Wordstar, un tableur et un agenda), d'une mémoire permanente et d'un afficheur à cristaux liquides de 8 lignes de 80 caractères, il possède tout ce qu'il faut pour travailler. Mais, qu'on ne s'y trompe pas, le PX-8 est aussi doté d'un langage Basic assez performant avec lequel on pourra facilement programmer.

Le Basic du PX-8 se présente sous la forme d'une mémoire morte (ROM) enfichable au-dessous de l'appareil. Cette particularité permet de disposer de tous les avantages d'un Basic résident (disponibilité, rapidité) mais amovible à volonté. Le PX-8 possède deux connecteurs pour ROM : on devra donc choisir entre les programmes d'application, ceux qui resteront en permanence à demeure dans l'appareil. Quant au support classique de ces ROM, il interdit leur destruction accidentelle par pliage des pattes du circuit. On les connectera donc en confiance.

Pour utiliser le langage Basic, il faut en commander le chargement en mémoire grâce au menu général de présentation du PX-8. Son origine s'affiche immédiatement : Epson Basic version 1.0. Il s'agit en fait d'un bon vieux Basic de Microsoft auquel Epson a apporté quelques adaptations. On dispose d'un espace mémoire programmable de taille variable en fonction de la configuration utilisée. En version de base, sans extension mémoire, on aura 14749 octets libres (14,4 Ko), ce qui est honnête. On pourra programmer dans cinq zones différentes, indépendantes les unes des autres.

Equipé pour les sauvegardes

Enfin, après une première utilisation du Basic, le langage demeure en mémoire ainsi que tous ses programmes. Et ceci tant qu'aucune autre application, par exemple Wordstar, n'aura été



Un magnétophone à microcassettes est intégré et entièrement commandable depuis le Basic du PX-8.

lancée. On pourra ainsi, en fonction de l'emploi de l'ordinateur, conserver en mémoire les programmes, ou bien les sauvegarder. Le PX-8 dispose de différentes possibilités de sauvegarde : sur disquette (à condition de disposer du lecteur), sur microcassette (le magnétophone est intégré à l'ordinateur) ou, enfin, sur disque de mémoire virtuelle.

Car le système d'exploitation de l'Epson n'est rien moins que le célèbre (et antique) CP/M, et celui-ci peut gérer la mémoire centrale comme s'il s'agissait d'une petite disquette. On y conservera donc en permanence, par exemple, tous les programmes Basic d'usage fréquent.

Le moins qu'on puisse dire de ce Basic, c'est qu'il possède abondance de fonctions. La liste ci-contre n'en recense pas moins de 157. Aucun domaine n'est oublié : mathématiques, traitement de caractères, programmation, etc.

Un très bon éditeur

Au moins aussi important que le langage lui-même est l'éditeur de programmes. Il s'agit pratiquement de l'art et la manière d'écrire, vérifier et corriger les lignes du programme Basic. Si l'écran du PX-8 est seulement haut de huit lignes, on dispose cependant d'un écran virtuel de 24 lignes de 80 caractères affichés sur lequel on se déplacera à l'aide de touches fléchées. Sous le contrôle du Basic, ce sont même deux écrans virtuels que l'on pourra utiliser indépendamment l'un de l'autre.

La frappe des programmes Basic s'effectue facilement car on dispose d'une fonction AUTO de numérotation automatique des lignes. Toute ligne tapée, et conclue par RETURN, est introduite en mémoire. On pourra les corriger à volonté, soit en remontant l'écran si la ligne s'y trouve encore affichée, soit en utilisant la commande EDIT. Une fois la ligne affichée, il n'est pas nécessaire de la retaper entièrement pour effectuer une correction. Comme s'il s'agissait d'un simple texte écrit, il suffit de positionner le curseur au bon endroit, de taper INS pour insérer des caractères, DEL pour en détruire ou toute autre touche pour remplacer des caractères par de nouveaux, puis de terminer par RETURN. Chaque ligne Basic pourra contenir jusqu'à 255 caractères.

Le contrôle des erreurs de frappe s'effectue lors de la première tentative d'exécution : une erreur s'affiche et la ligne fautive est automatiquement éditée, le curseur clignotant sur l'instruction erronée, prêt à recevoir la correction. La fonction DELETE détruit en bloc un nombre de lignes déterminé et RENUM permet une renumérotation automatique des numéros de ligne.

Un très bon éditeur, donc. Et des touches de contrôle augmentent encore sa puissance, par exemple, CTRL Z pour détruire tous les caractères d'une ligne à partir de la position du curseur, CTRL→ et CTRL← pour passer d'un écran virtuel à l'autre, etc.

Sur le plan de la rapidité de traitement, l'Epson n'est visiblement pas doté d'un turbo : 26 secondes pour compter bêtement jusqu'à 10000 (par une boucle vide FOR... NEXT), c'est honorable, mais pas exceptionnel. Ceci noté, n'oublions pas que la permanence de la mémoire est basée sur une technologie particulière de construction des circuits (CMOS) dont les temps d'accès sont

Liste des mots clés du Basic du PX-8

ABS
ALARM
ALARM$
ASC
ATN
AUTO
AUTO START

BEEP

CALL
CDBL
CHAIN
CHR$
CINT
CLEAR
CLOSE
CLS
COMMON
CONT
COPY
COS
CSNG
CSRLIN
CVI
CVS
CVD

DATA
DATE$
DAY
DEF FN
DEF INT
DEF SNG
DEF DBL
DEF STR
DEF USR
DELETE
DIM
DSKF

EDIT
END
EOF
ERASE
ERL
ERR
ERROR
EXP

FIELD
FILES
FIX
FOR STEP NEXT
FRE

GET
GOSUB...RETURN
GO TO

HEX$

IF...THEN...ELSE
INKEY$
INP
INPUT
INPUT#
INPUT$
INSTR
INT

KEY
KEY LIST
KILL

LEFT$
LEN
LET
LINE
LINE INPUT
LINE INPUT#
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LOGIN
LPOS
LPRINT
LPRINT USING
LSET

MENU
MERGE
MID$
MKI$
MKS$
MKD$
MOUNT

NAME
NEW

OCT$
ON ERROR GOTO
ON...GOSUB
ON...GOTO
OPEN
OPTION BASE
OPTION COUNTRY
OPTION CURRENCY
OUT

PCOPY
PEEK
POINT
POKE
POS
POWER
PRESET
PRINT
PRINT USING
PRINT#
PRINT# USING
PSET
PUT

RANDOMIZE
READ
REM
REMOVE
RENUM
RESET
RESTORE
RESUME
RIGHT$
RND
RSET
RUN

SAVE
SCREEN
SGN
SIN
SOUND
SPACE$
SPC
SQR
STAT
STOP
STOP KEY
STR$
STRING$
SWAP
SYSTEM

TAB
TAN
TAPCNT
TIME$
TITLE
TRON
TROFF

USR

VAL
VARPTR

WAIT
WHILE...WEND
WIDTH
WIND
WRITE
WRITE#

souvent un peu plus lents. Ainsi, dans sa catégorie, le PX-8 est un ordinateur des plus rapides.

La capacité de travail d'un Basic se mesure aussi à la précision de ses calculs. Le PX-8 travaillera aussi bien en décimal (base 10), qu'en octal (base 8) ou en hexadécimal (base 16). Les constantes numériques sont de trois types : les entiers (de +32767 à −32768), les réels en notation fixe et les réels en notation scientifique.

Les variables disponibles sont alphabétiques (255 caractères de capacité) ou numériques, en entière, simple ou double précision. La simple précision affiche six chiffres significatifs (les calculs étant effectués sur sept), la double précision en affiche seize.

Dimensions sans limites

Leurs noms pourront comprendre jusqu'à 40 caractères alphanumériques différents ! Les tableaux de variables (ou matrices de chiffres) ne sont pas limités en dimensions : DIM A(3000) ou DIM A(1,1,1,1,1,1,1,1) sont parfaitement valides, si la taille de la mémoire disponible le permet, évidemment. L'ordre ERASE récupérera de l'espace mémoire en détruisant individuellement certains tableaux inutiles.

Mathématiquement parlant, le PX-8 est très correctement fourni : fonctions scientifiques comme ABS, LOG, EXP, trigonométriques, SIN, COS, TAN, et ATAN, opérateurs de relations tels >, <, <=, etc. On définira des fonctions mathématiques avec DEF FN, et elles pourront ensuite être employées directement, presque comme de simples instructions Basic.

Les manipulations des chaînes alphabétiques s'effectuent avec les fonctions classiques "+" (concaténation), les opérateurs de relation (>, <, =,...) et les fonctions RIGHT$, LEFT$ et MID$ d'extraction de sous-chaînes (droite, gauche et centrale), ASC et CHR$ (conversions entre codes ASCII et caractères). La fonction INSTR$ recherche dans une chaîne de caractères la présence éventuelle d'une sous-chaîne précise et LEN retourne le nombre de caractères d'une chaîne.

On pourra ordonner un démarrage automatique d'un programme dès l'allumage du PX-8 avec AUTO START "C:BASICA:NOMDUPROGRAMME" +CHR$(13) ce qui correspond au choix du Basic, puis du programme NOMDUPROGRAMME, le tout ponctué de RETURN. Le langage-machine est accessible avec PEEK, POKE, CALL et DEF USR ; le microprocesseur du PX-8 est un Z80. L'horloge interne de l'ordinateur est accessible avec TIME$ (heure, minutes, secondes) et DATE$ (mois, jour, année).

Le contrôle des erreurs s'effectue avec ERROR qui en simule une, ERL qui indique le numéro de la ligne en question et ERR qui précise le numéro de code de l'erreur. Avec ON ERROR GOTO nnn et RESUME, on programmera des routines personnalisées de traitement des erreurs.

Epson ne s'est pas contenté de reprendre les traditionnelles instructions Basic. Sur nombre d'entre elles, des améliorations ont été apportées. La fameuse boucle FOR... NEXT est perfectionnée en admettant pour NEXT une souplesse de références : NEXT seul ou NEXT I,J,K sont corrects. Le test IF... THEN (*si... alors*) admet un ELSE (*sinon*). Ici, on trouve aussi une boucle inverse de FOR... NEXT : WHILE xxx (tant que xxx est vrai exécuter les lignes d'instructions suivantes), terminée par un WEND (vérifier la condition). L'ordre INPUT permet l'introduction de données dans une série de variables. Les cinq touches de fonction (dix en combinaison avec SHIFT) situées sous l'afficheur sont programmables en Basic. On pourra donc y affecter, avec KEY, des expressions Basic souvent employées.

L'écran du PX-8 peut être géré graphiquement, soit en affichant des caractères graphiques du jeu ASCII, soit en y dessinant point par point. LINE trace une ligne droite entre deux points, PSET allume un point, PRESET l'éteint, et POINT teste son état.

Disposition remarquable, le magnétophone à microcassettes intégré à l'ordinateur est entièrement commandable avec le Basic : retour arrière, avance, lecture et écriture. Même le compteur de tours est accessible ! En définitive, cette cassette se comporte exactement comme une disquette d'accès ralenti.

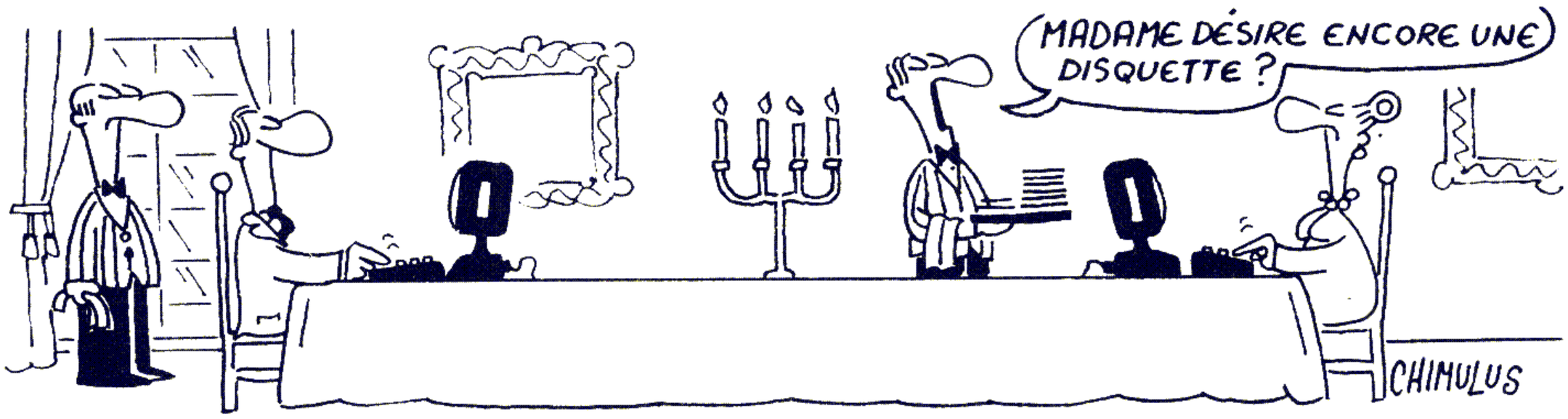
La revue détaillée de toutes les instructions Basic du PX-8 apporterait nombre d'idées intéressantes d'améliorations du Basic. Bien sûr, la compatibilité avec des Basic antérieurs en souffre mais lorsque, comme ici, le progrès est réel, vive l'incompatibilité !

Si seul le Basic, en raison de ses performances, intéresse l'utilisateur, l'Epson PX-8 reste sans doute cher. Mais, si cet utilisateur est intéressé par le traitement de texte portatif Wordstar et un tableur, il disposera alors en plus d'un Basic capable de réaliser toutes sortes d'applications personnelles et semi-professionnelles avec une grande souplesse. La conjugaison de ces outils justifie alors le prix de 12 600 FF.

Jean-Christophe KRUST

Fiche technique de l'Epson PX-8

Constructeur : *Epson*
Distributeur : *Technology Resources*
Prix public : *11 400 FF avec Basic et CP/M ; 12 600 F avec, en plus, Wordstar, Calc et Scheduler*
Processeur : *Z80*
Mémoire vive : *64 Ko (après chargement du Basic, il reste 14,4 Ko) ; extension possible de 120 Ko*
Langage : *Epson Basic 1.0 sur mémoire morte à enficher (fournie avec la machine)*
Variables : *entières de −32768 à 32767 ; 6 chiffres significatifs pour la simple précision ; 16 chiffres significatifs pour la double précision*
Précision : *calculs sur 7 chiffres significatifs*
Nombre de mots clés du Basic : *157*



QUAND ON CHERCHE SES MOTS

LA recherche d'une chaîne de caractères dans un texte, fonction classique pour un bon traitement de texte, paraît simple. Pourtant, comme toujours en informatique, il n'existe pas de solution unique. Parmi les algorithmes de recherche de chaîne dans un texte, voyons celui auquel chacun pense, et quelques autres...

C'est dans le système d'exploitation d'un ordinateur que nous avons découvert notre premier algorithme de recherche. Cet utilitaire a pour but de localiser un couple d'octets en mémoire centrale. L'algorithme utilisé en dit long sur les programmes-système livrés par certains fabricants d'ordinateurs puisqu'il ne fonctionne que dans des cas bien définis. On comprend pourquoi les services informatiques reçoivent tous les mois, et pour chaque machine, des fascicules d'une centaine de pages décrivant les dernières erreurs découvertes dans les programmes-système. Comme ce programme est écrit en Assembleur, nous l'avons converti en notation algorithmique (voir encadré ci-contre).

```
t ← 1 ;
trouvé ← faux ;
tant que t < lontex et non trouvé faire
    début
        si texte (t) = mot (1)
           alors
            début
               t ← t+1 ;
               si texte (t) = mot (2)
                 alors
                     trouvé ← vrai
            fin ;
        t ← t + 1
    fin ;
si  trouvé alors t ← t – lonmot ;
```

Dans cette première version de l'algorithme (qui n'est pas structuré) et comme pour les suivantes, *texte* est un tableau ou une fonction qui retourne le caractère du numéro indiqué dans le texte. De la même façon, *mot* est un tableau représentant les différents caractères du mot à rechercher. Enfin *lontex* et *lonmot* représentent respectivement la longueur du texte et celle du mot.

Mais c'est une bogue !

Nous pouvons tout d'abord remarquer que cet algorithme est faux. En effet, si l'on recherche un couple de caractères tel que **du**, rien n'est trouvé si la séquence **ddu** apparaît dans la liste des symboles. Voyons pourquoi : dès la détection du premier **d**, la valeur de *t* est incrémentée afin que le caractère suivant soit vérifié. Comme il s'agit encore d'un **d**, la valeur de **trouvé** n'est pas modifiée et la recherche reprend au caractère **u**, laissant ainsi passer la chaîne recherchée.

Si cet algorithme est faux, cela provient du fait qu'il est à la fois mal écrit

QUAND ON CHERCHE SES MOTS

et mal paramétré. Il est par ailleurs inutilisable si l'on désire rechercher des chaînes d'un seul ou de plus de deux caractères. Nous pouvons donc le récrire en en conservant l'esprit, mais en le rendant plus général et en le structurant.

```
t ← 0 ;
trouvé ← faux ;
tant que t < lontex et non trouvé faire
    début
        m ← 1 ;
        t ← t + 1 ;
        tant que t < lontex et m < lonmot et texte (t) = mot (m) faire
            début
                m ← m + 1 ;
                t ← t + 1
            fin
        si m > = lonmot et texte (t) = mot (m) alors trouvé ← vrai
    fin ;
si trouvé alors t ← t − lonmot + 1 ;
```

Comme dans la première version, l'erreur de cet algorithme provient du fait que, dans le cas où le début de la chaîne cherchée apparaît (mais pas sa fin), il faut non pas continuer l'analyse des caractères, mais revenir en arrière pour la reprendre à l'endroit où nous avions cru détecter la chaîne cherchée.

Cette deuxième version étant plus structurée, l'algorithme devient plus facile à corriger. C'est ainsi que nous obtenons un nouvel algorithme :


```
t ← 0 ;
trouvé ← faux ;
tant que t < lontex et non trouvé faire
    début
        m ← 1 ;
        tant que t + m < lontex et m < lonmot et texte (t+m) = mot (m) faire
            m ← m + 1
        si m > = lonmot et texte (t+m) = mot (m)
            alors trouvé ← vrai
            sinon t ← t + 1
    fin ;
si trouvé alors t ← t + 1 ;
```

Cet algorithme de recherche, qui vient assez naturellement à l'esprit, présente deux grands défauts. Le premier est qu'il n'est pas efficace, car à chaque échec dans la recherche il impose un retour en arrière presque égal à la longueur du mot. Le second inconvénient apparaît si le texte n'est pas la mémoire centrale, et s'il est, par exemple, transmis à distance. Dans ce cas, les retours en arrière peuvent devenir un lourd handicap.

Les améliorations à apporter à cet algorithme de base sont donc d'une part une plus grande efficacité en cas d'échec dans la recherche, et d'autre part des retours en arrière moins fréquents et d'une moindre ampleur.

Ces améliorations mettent en œuvre une phase d'analyse préalable du mot à rechercher. Une meilleure connaissance de ce mot facilite sa recherche par l'ordinateur. Cette analyse est en fait très rapide car le mot est le plus souvent assez court par rapport à la longueur du texte. Le résultat de l'analyse est mémorisé dans un tableau qui possède autant d'éléments que le mot comporte de symboles.

L'algorithme de Knuth, Morris et Pratt travaille de cette façon. Il mémorise, pour chaque élément du tableau, le nombre de positions à parcourir en arrière si un échec apparaît sur le symbole en question. Comme le faisait l'algorithme élémentaire, celui-ci fait

glisser le mot à rechercher sur le texte, en essayant de trouver une correspondance entre les deux. Si un échec apparaît, c'est-à-dire si deux caractères sont différents, le mot est déplacé vers la droite non pas d'une seule position comme avec notre premier algorithme, mais d'un nombre égal à la valeur mémorisée dans la table.

Reculer pour ne pas sauter

Ainsi, cette table indique la valeur du décalage dont il faut déplacer le mot. Si nous considérons que les valeurs du tableau *décalage* ont été correctement calculées, notre algorithme est le suivant :

```
t ← 1 ;
m ← 1 ;
tant que t < = lontex et m < = lonmot faire
    début
        tant que m > 0 et alors texte (t) < > mot (m) faire
            m ← décalage (m) ;
        t ← t + 1 ;
        m ← m + 1
    fin ;
si m > lonmot
        alors t ← t – lonmot ;
        sinon mot__non__présent ;
```

(Algorithme de Knuth, Morris et Pratt)

Il convient de remarquer une petite particularité dans cet algorithme : il utilise une instruction très utile, qui était implicitement présente dans le langage Algol et qui est réapparue explicitement en Ada. Il s'agit du connecteur **et alors**. Cet opérateur fonctionne comme un **et** logique, avec cette différence que l'expression suivant le **et alors** n'est pas évaluée si la première partie est fausse.

Cette propriété est astucieuse : si l'expression *m* > 0 n'est pas vérifiée, il est inutile d'évaluer *mot (m)* < > *texte (t)*, puisque de toute façon le **et** logique fournira un résultat faux. Cela permet une construction plus simple des programmes. Dans le cas présent, si **m** était inférieur ou égal à zéro, l'évaluation de la seconde partie de la condition provoquerait une erreur, l'élément 0 n'existant pas.

Dans le même ordre d'idées, il existe le connecteur **ou alors** avec lequel la seconde partie de la condition n'est évaluée que si la première donne un résultat faux.

Pour mieux comprendre cet algorithme, il convient de voir comment il fonctionne. Pour cela, nous allons prendre l'exemple de la recherche du mot *ababc*. Admettons que notre texte comporte la suite de symboles *abababc*.

Au début de l'analyse, les quatre premiers caractères correspondent bien au mot que nous recherchons. Par contre, arrivé au cinquième, le système considère qu'il y a échec, puisque le caractère du texte est un *a* mais qu'un *c* est présent dans le mot.

Alors que l'algorithme élémentaire repart du second caractère du texte pour rechercher une nouvelle coïncidence avec le mot, celui-ci va tirer parti du fait que la suite *ab* a déjà été trouvée. C'est ainsi que l'analyse du texte va repartir non pas à partir du deuxième caractère du texte, mais du cinquième pour vérifier s'il s'agit d'un *a*.

Il ne reste plus qu'à établir l'algorithme qui permet de calculer les valeurs mémorisées dans le tableau *décalage*. Le voici :

```
pour i ← 1 à lonmot faire
    décalage (i) ← 0 ;
pour i ← 2 à lonmot faire
    début
        j ← 0 ;
        répéter
            j ← j + 1 ;
            si décalage (i + j – 1) = 0 alors décalage (i + j – 1) ← j ;
        jusqu'à mot (j) < > mot (i + j – 1) ou alors i + j – 1 > = lonmot
    fin ;
```

Ainsi, dans notre exemple précédent qui effectue la recherche de la chaîne *ababc*, le tableau *décalage* mémorise les valeurs suivantes : 0,1,1,2 et 3.

L'algorithme de Boyer et Moore effectue la recherche d'un mot en analysant non pas tous les caractères du texte, mais seulement quelques-uns soigneusement choisis. De plus, les caractères du mot sont comparés de la droite vers la gauche, et non de la gauche vers la droite comme dans les algorithmes précédents. Recherchons par exemple quelles conclusions tirer en comparant le dernier caractère du mot avec un caractère du texte que nous appellerons *car*.

- Si *car* est le dernier caractère du mot, les *lonmot* – 1 caractères précédents du texte sont comparés avec ceux du mot pour savoir si celui-ci est ou non présent à cet emplacement.
- Si *car* n'existe pas dans le mot, il est possible de passer immédiatement *lonmot* positions dans le texte, pour le comparer au dernier caractère du mot. En effet, il est inutile d'examiner en détail tous les symboles du mot puisqu'un au moins ne convient pas.
- Si *car* n'est pas le dernier caractère du mot, on peut avancer dans le texte autant de fois qu'il est nécessaire pour amener *car* en correspondance avec le même caractère du mot.

Comme nous le constatons, cet algorithme permet d'avancer très rapidement dans le texte puisqu'il arrive dans certaines conditions que seul un caractère sur *lonmot* soit analysé. Toutefois cette méthode se complique assez rapidement dès que l'on entre dans les détails. Ainsi, lorsque *car* correspond avec le dernier caractère du mot, les symboles précédents sont comparés jusqu'à ce que l'on découvre que le mot est présent dans le texte ou qu'un caractère ne convient pas. Dans ce dernier cas, il est nécessaire d'avancer à nouveau dans le texte, mais seulement à partir de cette position d'échec.

Finalement, cet algorithme souffre du nombre de cas particuliers qu'il doit traiter, ce qui fait qu'il devient rapidement assez complexe et par conséquent moins efficace qu'il n'en a l'air.

D'une façon générale, les éditeurs de texte, qui ont tous une fonction de recherche de chaîne, utilisent l'algorithme de Knuth, Morris et Pratt. Bien entendu, il peut se trouver sous différentes formes plus ou moins optimisées. En fin de compte, nous n'avons pas cherché ici à découvrir les algorithmes les plus efficaces, mais plutôt à voir comment, à partir d'un problème simple, il est possible d'imaginer des méthodes de résolution très différentes les unes des autres.

Thierry CHAMORET

QUELS CURIEUX CARACTÈRES !

ON peut, avec l'Amstrad, redéfinir des caractères. Il n'en faut pas plus pour combler d'aise les amoureux de la cédille, du circonflexe et du tréma. C'est aussi l'occasion de créer des véhicules spatiaux, des bestioles venues d'un autre monde, et tout ce qu'il vous plaira. Si le champ est vaste, il manque un outil pratique de création. Le voici.

Comment sont constitués les caractères sur le CPC 464 ? La documentation fournie à l'achat de la machine est assez claire sur ce point, et nous nous contenterons d'en rappeler les grandes lignes.

Chaque caractère affichable est inscrit dans une matrice carrée de 8 points sur 8. S'il existe 256 caractères définis, seuls 224 d'entre eux sont vraiment affichables à l'écran : ceux qui portent les numéros 0 à 31 correspondent en effet à des codes de contrôle. Les numéros 32 à 255 sont, eux, tout à fait simples d'emploi, puisque pour afficher le caractère numéro 65 (il s'agit de la lettre A), il suffit d'un PRINT CHR$(65).

A noter que, ce caractère étant disponible directement au clavier, il est plus simple de l'obtenir par un PRINT "A" ; mais comme tous les caractères existants ne figurent pas sur le clavier, cette dernière solution n'est pas toujours utilisable. Par exemple, PRINT CHR$(249) fait apparaître un petit bonhomme qui n'est représenté sur aucune touche.

La redéfinition des caractères passe par l'emploi des instructions SYMBOL, et SYMBOL AFTER. Cette dernière commande indique au système le nombre de caractères qui peuvent être redé-

```
10 KEY 139,"MODE 1:LIST"+CHR$(13)
20 REM*****************************
30 REM     * CREACAR * CPC 464 *
40 REM     (C)JP LALEVEE ET LIST
50 REM*****************************
60 :
70 DIM C(7):SYMBOL AFTER 32
80 R$="O":WHILE R$="O"
90 :
100 FOR I=0 TO 7:C(I)=0:NEXT:REM MISE A 0 DU TABLEAU
110 :
120 REM ++++++++ PRESENTATION +++++++++
130 CLS:MODE 1:BORDER 10
140 PEN 2:PRINT " "; STRING$(38,"-")
150 LOCATE 10,2:PEN 1:PRINT"CREATEUR DE CARACTERES":PEN 2
160 PRINT " "; STRING$(38,"-")
170 :
180 FOR I=0 TO 8
190 MOVE 112,288-I*16:DRAW 112+8*16,288-I*16,1
```

```
200 MOVE 112+I*16,288:DRAW 112+I*16,288-8*16,1
210 NEXT
220 WINDOW#1,1,40,21,25:PAPER#1,2:CLS#1
230 WINDOW#2,25,31,10,13:PAPER#2,1:CLS#2
240 PEN#2,3:LOCATE 1,1:PRINT#2,"NO:";
250 :
260 REM --------- DEFINITIONS ---------
270 NC=0:WHILE NC<33 OR NC>255
280 PEN#1,0:INPUT#1,"NUMERO DU CARACTERE ";N$
290 NC=VAL(N$)
300 IF NC<33 OR NC>255 THEN PRINT#1,"CETTE VALEUR EST ILLEGALE !"
310 WEND
320 CLS#1:PRINT#2,NC
330 PEN#2,0:LOCATE#2,4,3:PRINT#2,CHR$(NC)
340 PEN#1,3:PRINT#1,"UTILISEZ LES FLECHES POUR VOUS DEPLACER"
350 PEN#1,0:PRINT#1,"        COPY POUR ALLUMER UN POINT."
360 PRINT#1,"        DEL  POUR ETEINDRE UN POINT."
370 PRINT#1,"        'F'  POUR TERMINER."
380 REM +++++++++ PGM PPAL ++++++++++++
390 X=0:Y=0:GOSUB 730
400 :
410 D$="":WHILE D$<>"F"
420 D$="":WHILE D$="":D$=INKEY$:WEND
430 IF D$=CHR$(127) THEN GOSUB 800:REM TOUCHE DEL
440 IF D$=CHR$(241) THEN GOSUB 670:Y=Y+1
450 IF D$=CHR$(240) THEN GOSUB 670:Y=Y-1
460 IF D$=CHR$(243) THEN GOSUB 670:X=X+1
470 IF D$=CHR$(242) THEN GOSUB 670:X=X-1
480 IF D$=CHR$(224) THEN GOSUB 870:REM TOUCHE COPY
490 GOSUB 720:REM DESSINE LA CROIX
500 WEND
510 :
520 REM --------- FIN DU DESSIN--------
530 CLS#1
540 PEN#1,0:PRINT#1,"CARACTERE";NC;":"
550 FOR I=0 TO 7
560 PEN#1,3:PRINT#1,C(I);
570 NEXT
580 PRINT#1:PRINT#1:PRINT#1,"UN AUTRE CARACTERE (O/N)?"
590 R$="":WHILE R$<>"N" OR R$<>"O":R$=INKEY$:WEND
600 WEND
610 :
620 REM +++++++ FIN DU PROGRAMME ++++++
630 MODE 1:PRINT"AU REVOIR !"
640 END
650 :
660 REM -------- EFFACE CROIX ---------
670 PX1=120+X*16:PY1=280-Y*16
680 MOVE PX1-4,PY1:DRAW PX1+4,PY1,0
690 MOVE PX1,PY1-4:DRAW PX1,PY1+4,0
700 RETURN
710 :
720 REM -------- DESSINE CROIX --------
730 IF X<0 THEN X=0:ELSE IF X>7 THEN X=7
740 IF Y<0 THEN Y=0:ELSE IF Y>7 THEN Y=7
750 PX1=120+X*16:PY1=280-Y*16
760 MOVE PX1,PY1-4:DRAW PX1,PY1+4,3
770 MOVE PX1-4,PY1:DRAW PX1+4,PY1,3
780 RETURN
790 :
800 REM -------- EFFACE POINT ---------
810 LOCATE X+8,Y+8:PRINT " "
820 MOVE 112+X*16,288:DRAW 112+X*16,288-8*16,1
830 MOVE 112,288-Y*16-16:DRAW 112+8*16,288-Y*16-16,1
840 IF (C(Y) AND 2^(7-X))>0 THEN C(Y)=C(Y)-2^(7-X)
850 GOTO 930
860 :
870 REM -------- ALLUME POINT ---------
880 LOCATE X+8,Y+8:PRINT CHR$(207)
890 MOVE 112,288-Y*16-16:DRAW 112+8*16,288-Y*16-16,1
900 MOVE 112+X*16,288:DRAW 112+X*16,288-8*16,1
910 IF (C(Y) AND 2^(7-X))=0 THEN C(Y)=C(Y)+2^(7-X)
920 :
930 REM ------ DESSINE CARACTERE ------
940 SYMBOL NC,C(0),C(1),C(2),C(3),C(4),C(5),C(6),C(7)
950 LOCATE#2,4,3:PRINT#2,CHR$(NC)
960 RETURN
970 END
```

Analyse du programme

Ligne 10 : comme à l'accoutumée, nous nous préparons le moyen de reprendre rapidement le contrôle de la situation lors de la mise au point du programme, par un simple appui sur la petite touche ENTER.

Ligne 70 : nous utiliserons un tableau de 8 cases correspondant aux 8 lignes d'un caractère. Le dimensionnement est limité à 7 puisque la case C(0) est utilisable et vient donc s'ajouter à C(1), C(2), C(3),..., C(7). Pour pouvoir redéfinir à peu près tous les caractères, SYMBOL AFTER est suivi du nombre 32, ce qui nous laisse 224 caractères reprogrammables. Un tel luxe n'est pas indispensable en l'occurrence, mais vous permettra de constater l'aspect étrange du programme si vous modifiez par exemple le caractère 32 : l'espace.

Lignes 180 à 210 : la grille de travail 8×8 apparaît à l'écran.

Lignes 220 à 240 : nous définissons 2 fenêtres d'affichage ; l'une contiendra le dessin du caractère en cours de traitement, l'autre sera réservée aux commentaires et aux résultats.

Lignes 260 à 310 : le programme s'enquiert du numéro du caractère à modifier, puis il en contrôle la validité.

Lignes 320 à 370 : ici s'affiche le mode d'emploi du programme.

Lignes 420 à 490 : en fonction des touches pressées par l'utilisateur, l'exécution est dirigée vers les différents sous-programmes de traitement. Les codes CHR$() correspondent aux touches DEL, COPY, et aux 4 touches de déplacement du curseur.

Lignes 530 à 570 : si l'utilisateur appuie sur F, l'affichage des 8 valeurs qui codent le caractère est effectué sur la fenêtre 1.

Lignes 580 à 640 : il est alors possible de redéfinir un autre caractère.

Lignes 660 à 700 : si l'utilisateur se déplace à l'aide des touches curseur, la croix rouge qui indique la position actuelle dans la grille est effacée.

Lignes 720 à 780 : elle est ensuite redessinée, et l'on prend garde à ne pas sortir du cadre de travail (grâce au test de X et Y qui représentent les coordonnées nouvelles).

Lignes 800 à 850 : en cas d'appui sur DEL, un point situé aux coordonnées X et Y est effacé. Il faut alors reconstituer la grille de travail qui vient de se trouver amputée. Enfin, la valeur numérique venant de changer sur la ligne Y en cours, on met à jour l'élément C(Y).

Lignes 870 à 910 : en cas d'appui sur COPY, on effectue une action similaire ayant pour but d'indiquer l'apparition d'un nouveau point dans la grille.

Lignes 930 à 960 : enfin, lors de chaque modification, on redessine le caractère affiché dans la fenêtre 2, ce qui permet de conserver sous les yeux l'aspect exact du nouveau caractère.

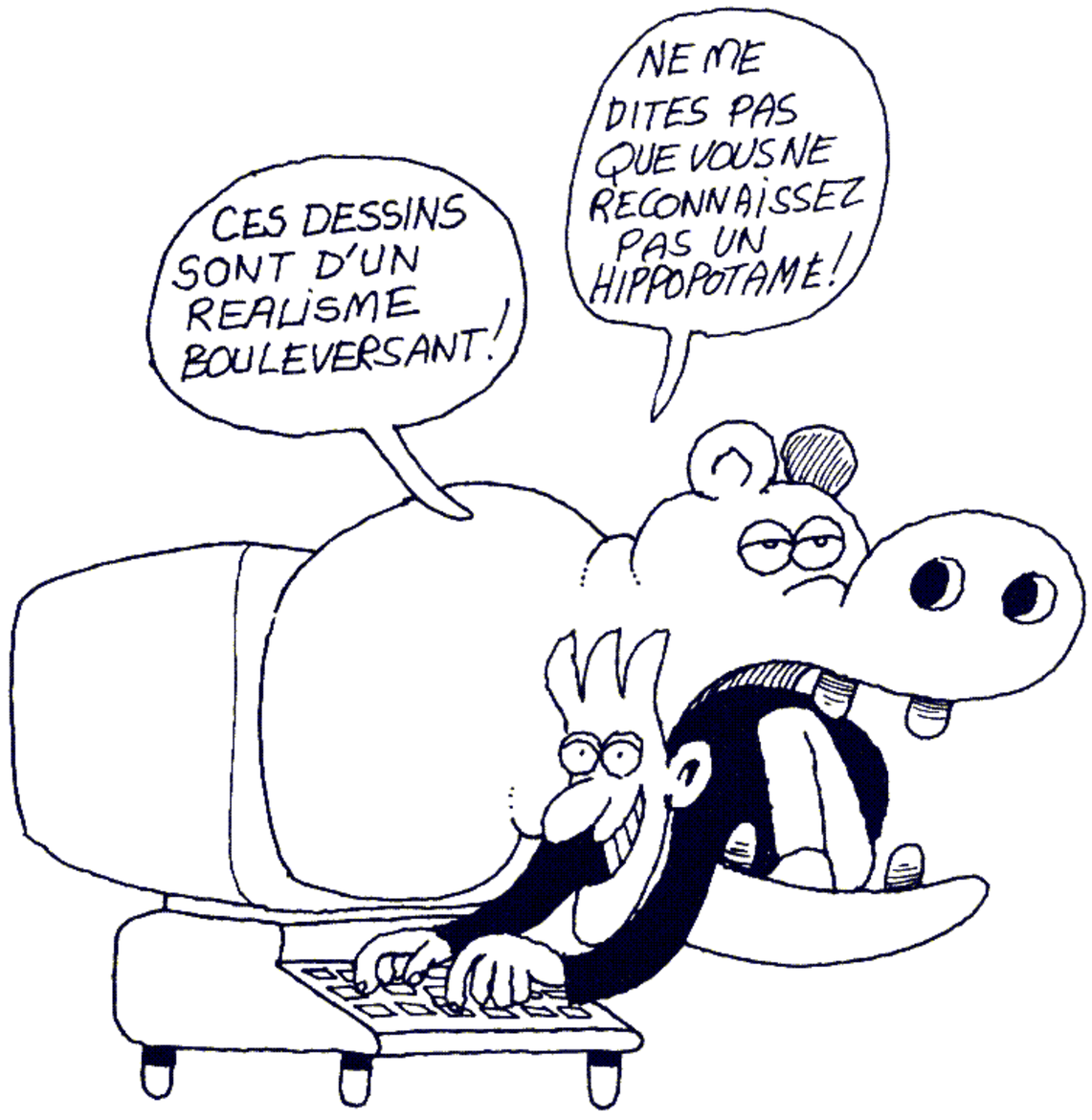
QUELS CURIEUX CARACTÈRES !

finis. Elle exige un argument numérique représentant le numéro du premier caractère qui pourra être modifié. Si vous n'utilisez pas cette commande, vous pourrez tout de même en modifier 16 : ceux qui portent les numéros 240 à 255. La commande SYMBOL est plus difficile à exploiter, car elle exige 9 paramètres pour fonctionner. Le premier représente simplement le numéro du caractère à redéfinir. Les 8 suivants représentent les valeurs numériques qui codent le caractère, calculées ligne à ligne à partir de la matrice 8×8.

Soyez paresseux

Ce programme que nous vous proposons vous épargnera de fastidieux calculs pour déterminer la valeur des 8 paramètres. Le caractère en cours de redéfinition sera visible sur l'écran en permanence, ce qui vous permettra de juger immédiatement de la qualité du résultat. Quand vous en serez satisfait, un simple appui sur la touche "F" provoquera l'affichage des 8 valeurs numériques attendues, que vous n'aurez plus qu'à recopier.

Si vous deviez faire ce travail manuellement, il vous faudrait dessiner une grille de 8 lignes sur 8 colonnes, et noircir les cases correspondant au dessin du caractère. Vous transformeriez ensuite chaque ligne en un nombre binaire, composé de 8 chiffres (un octet) correspondant eux-mêmes aux 8 points de la ligne. Un point noirci serait transformé en 1 ; un point intact devenant un 0. Enfin, ce nombre binaire serait à transformer en son équivalent décimal ou hexadécimal. Voyez le schéma de la lettre A qui résume la situation.



Heureusement, nous allons nous débrouiller autrement. C'est le programme, et lui seul, qui se chargera de mener à bien ce travail ingrat : à vous les joies de la création, à lui les basses besognes. Voilà une conception très plaisante de l'informatique, non ?

Tel qu'il vous est proposé, le programme vous évitera à coup sûr bien des erreurs. Mais on peut toujours modifier ou améliorer un logiciel. Comment faire évoluer celui-ci ? Il est relativement facile de le compléter en faisant en sorte que l'on puisse conserver sur fichier (cassette ou disquette) les codes des caractères redéfinis et les récupérer à volonté depuis tout autre programme Basic de votre cru.

Une autre possibilité, plus complexe celle-là, serait de traiter une plus vaste grille rassemblant un grand nombre de caractères et de réaliser un panneau graphique complet. Notre programme pourrait s'intégrer tout naturellement dans un tel projet.

Jean-Pierre LALEVÉE

Codage d'un caractère sur huit octets

128	64	32	16	8	4	2	1	
0	0	0	1	1	0	0	0	= 24 (16+8)
0	0	1	1	1	1	0	0	= 60 (32+16+8+4)
0	1	1	0	0	1	1	0	= 102 (64+32+4+2)
0	1	1	0	0	1	1	0	= 102 (64+32+4+2)
0	1	1	1	1	1	1	0	= 126 (64+32+16+8+4+2)
0	1	1	0	0	1	1	0	= 102 (64+32+4+2)
0	1	1	0	0	1	1	0	= 102 (64+32+4+2)
0	0	0	0	0	0	0	0	= 0

En haut du schéma, la correspondance décimale des bits. Sur la droite, huit nombres, chacun sur un octet, pour coder la lettre "A".

EN BASIC LES 32 BITS SONT RAPIDES, SANS PLUS

LES microprocesseurs 32 et 16 bits ont sur les 8 bits l'avantage théorique de la rapidité. Mais cet avantage se retrouve-t-il quand la machine fonctionne en Basic ? Pas vraiment.

Les programmes-tests

Test 1 - Boucle vide
```
10  FOR I=1 TO 10000
20  NEXT I
30  END
```

Test 2 - Sous-programmes
```
10  FOR I=1 TO 10000
15  GOSUB 100
20  NEXT I
30  END
100 GOSUB 110
110 RETURN
```

Test 3 - Matrice
```
 5  DIM A(10,10)
10  FOR I=1 TO 10
12  FOR J=1 TO 10
13  FOR K=1 TO 100
15  A(I,J)=K
17  NEXT K
18  NEXT J
20  NEXT I
30  END
```

Test 4 - Opérations sur les chaînes de caractères
```
 5  A$=« LISTEST »
10  FOR I=1 TO 10000
15  B$=LEFT$(A$,2)
      +MID$(A$,3,3)
      +RIGHT$(A$,2)
20  NEXT I
30  END
```

Test 5 - Arithmétique
```
10  FOR I=1 TO 10000
15  J=I*7+3/I
20  NEXT I
30  END
```

Test 6 - Calcul scientifique
```
10  FOR I=1 TO 10000
15  J=SIN (LOG(I))
20  NEXT I
30  END
```

Test 7 - Affichage
```
10  FOR I=1 TO 10000
15  PRINT CHR$(11) ;
      « LISTEST » ; I
20  NEXT I
30  END
```

Test 8 - Tracé d'une ligne graphique
```
10  FOR I=1 TO 10000
15  LINE(0,0)-(319,199)
20  NEXT I
30  END
```

Test 9 -Écriture de fichiers
```
 5  A$=« LISTEST »
 6  OPEN « O », # 1,
      « FICHIER »
10  FOR I=1 TO 10000
15  PRINT # 1, A$
20  NEXT I
25  CLOSE
30  END
```

Test 10 - Lecture de fichiers
```
 6  OPEN « I », # 1,
      « FICHIER »
10  FOR I=1 TO 10000
15  INPUT # 1, A$
20  NEXT I
25  CLOSE
30  END
```

La sortie sur Macintosh du nouveau Basic Microsoft en version 2 binaire, plus rapide que la version 1 testée dans *LIST* 7 donnait l'occasion de comparer les vitesses des machines 32 et 8 bits : Macintosh et QL chez les premières, Dai et BBC chez les secondes. Et nous en avons profité pour rajouter l'arbitrage d'un ordinateur 16 bits : le sérieux IBM-PC.

Les vieux 8080 et 6502 seraient-ils écrasés ou balayés par les flambants 68000 ? Nous en sommes en fait très loin, et les victoires des Dai et BBC dans les tests 1, 2, 3, 4, 7 et 10 prouvent que les 8 bits ont encore de beaux jours devant eux... Quant au malheureux IBM-PC, il ne gagne pas le moindre test.

Cela dit, la vitesse d'exécution d'un Basic n'est qu'une caractéristique parmi d'autres. Pour certaines applications, elle ne rentre pas en ligne de compte.

Résultats des dix tests

Ordinateur	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Processeur
Macintosh	**5**	17	30	96	**43**	191 15 chiffres	560	**345** 500×250	**69** disquette	68 disquette	68000 32/16 bits
Sinclair QL	19	68	66	119	54	**145** 8 chiffres	1269	2293 512×256	135 microdrive	99 microdrive	68008 32/8 bits
Dai	**5**	30	**21**	87	87	799 6 chiffres	427	2518 336×256	77 disquette	**62** disquette	8080 8 bits
BBC	7	**15**	32	**41**	57	405 9 chiffres	**114**	468 640×256	1944 cassette	1944 cassette	6502 8 bits
IBM-PC	13	36	54	97	73	290 7 chiffres	334	908 640×200	110 disquette	110 disquette	8088 16/8 bits

Les temps sont exprimés en secondes. En caractères gras, les meilleurs résultats.

GARDER LE SECRET

UN système de chiffrement est utilisé pour coder des messages que l'on souhaite garder secrets. Si, tout en étant fiable, il réduit le nombre d'octets des textes à transmettre, il devient particulièrement intéressant pour les ordinateurs. C'est le cas de celui qui est présenté ici dans un programme en Basic.

Le système de chiffrement que nous vous présentons remplit une double fonction : il assure le secret de vos communications tout en réduisant de 40 % le nombre d'octets des textes à transmettre. Normalement prévu pour des transmissions par fil ou un stockage sur bande ou disquette, ce procédé peut aussi servir pour une sortie sur imprimante. Cette variante se prête mieux à une démonstration et c'est pourquoi nous l'avons choisie, bien que l'obligation de transcrire chaque octet par deux chiffres hexadécimaux masque l'économie réalisée.

Entrons dans le vif de la technique. La fonction abréviative est assurée par un premier codage : l'espace et les 13 lettres les plus fréquentes sont représentés par un chiffre héxadécimal unique, autre que 1 ou 2 ; les autres lettres, les chiffres et 9 signes de ponctuation, par 2 chiffres hexadécimaux dont le premier est toujours 1 ou 2.

Le résultat de ce codage initial est découpé en groupes de quatre chiffres hexadécimaux correspondant chacun à un « entier » au sens Basic du terme, c'est à dire un nombre compris entre − 32768 et + 32767 et codé sur deux octets. Ces entiers subissent une première substitution, puis à la fin de la saisie, une transposition, constituée par un mélange pseudo-aléatoire réversible, et une deuxième substitution avant sortie pour transmission, stockage ou impression.

Le déchiffrement effectué dans l'ordre inverse les opérations réciproques de celles du chiffrement.

Une « clef secrète » numérique X, convenue entre les correspondants, la date et l'heure Y du message, ainsi que sa longueur entrent dans le calcul des « graines » servant à initialiser des séquences RND. Ces dernières fournissent les clefs individuelles FN K, définies ligne 20, qui déterminent chaque substitution ou transposition élémentaire par le biais de l'opérateur XOR (ou exclusif). Ce dernier a été choisi pour la symétrie commode des opérations réciproques : si A XOR B = C, alors C XOR B = A.

Tant pis pour les fautes

Mis à part le codage initial, l'utilisation d'entiers et celle de XOR, les principes de chiffrement sont analogues, pour l'essentiel à ceux du procédé exposé dans le n° 5 de *LIST* (page 28), mais on peut cette fois utiliser les chiffres et la ponctuation dans le clair.

```
R10 CLS:PRINT"Préparer l'imprimante":DEFSTR A-D:DEFINT E-M:DEFDBL X-Z
15 D="#A##CDEILNOPRSTU BFGHJKMQVWXYZ.,':/-()?0123456789":P=65535!
20 DEF FN A=RIGHT$("000"+HEX$(F),4):DEF FN K=P*(RND-.5):INPUT"CLEF";X
25 PRINT"1 Chiffrement,  2 Déchiffrement ?"
30 ON VAL (INKEY$) GOTO 35,75:GOTO 30
35 C=TIME$:Y=VAL(MID$(DATE$,4,2)+LEFT$(C,2)+MID$(C,4,2)+MID$(C,7))
40 Z=RND(-X/Y):DIM F(2500):PRINT"Taper le texte, puis #"
45 C=INKEY$:IF C="" THEN 45
50 E=INSTR(D,C):IF E THEN PRINT C;:B=B+HEX$(E-2) ELSE BEEP
55 IF LEN(B)>3 THEN F(L)=FN K XOR VAL("&H"+LEFT$(B,4)):B=MID$(B,5):L=L+1
60 IF E-1 THEN 45 ELSE Z=RND(-X*Y/L):LPRINT Y,L:FOR I=0 TO L-1
65 GOSUB 70:F=F(I) XOR FN K:LPRINT FN A+" ";:NEXT:END
70 M=I+ABS(FN K)MOD(L-I):SWAP F(M),F(I):RETURN
75 INPUT"Date.Heure";Y:INPUT"Longueur";L:DIM F(L),H(L):Z=RND(-X*Y/L)
80 FOR I=0 TO L-1:F(I)=I:NEXT:PRINT"Taper le crypto":FOR I=0 TO L-1
85 B=B+INKEY$:IF LEN(B)<4 THEN 85
90 F=VAL("&H"+B):IF B<>FN A THEN BEEP:B="":GOTO 85
95 PRINT B+" ";:B="":GOSUB 70:H(F(I))=F XOR FN K:NEXT:Z=RND(-X/Y)
100 FOR I=0 TO L-1:F=H(I) XOR FN K:C=C+FN A
105 IF LEN(C)<2 THEN NEXT:LPRINT,Y,L:END
110 E=2+(C<"1")+(C>="3"):G=VAL("&H"+LEFT$(C,E))
115 C=MID$(C,E+1):LPRINT MID$(D,G+2,1);:GOTO 105
```

Cryptographie
Programme pour Epson PX-8
Auteur Pierre Barnouin

Un exemple de message codé, puis décodé.

```
Clef: 123456789
 20143315       21
E5CF 2723 062D C203 9AB1 00E7 745C 1DFF A256 B728 E63A 8663 1CD2 A1FA CBAE 19E0
4B0D 83ED DA79 622B 94ED

L'HISTOIRE NE SUIT PAS UN SEUL COURS, ET L'ESSENTIEL N'EST JAMAIS CONCLU.
 20143315       21
```

Mesurés en octets, les textes en français usuel sont abrégés de 40 %.

Le contrôle de saisie écarte les minuscules et certains signes au chiffrement, et n'accepte que des groupes de quatre chiffres hexadécimaux au déchiffrement. La saisie s'effectue au kilomètre sans possibilité de correction, mais les fautes de frappe auront peu d'inconvénients. La saisie du crypto en blocs de quatre chiffres hexadécimaux par INPUT$(4) minimise le risque d'un décalage qui obligerait à recommencer le déchiffrement.

Côté sécurité, la transposition et les substitutions se protègent mutuellement de façon très efficace, et l'écrasement des fréquences par le précodage vient encore compliquer la tâche des décrypteurs, qui devront disposer de gros moyens informatiques et d'une parfaite connaissance des séquences RND sur l'ordinateur utilisé. Pour un usage professionnel, on les priverait de ce dernier atout en remplaçant la fonction FN K très simple définie ligne 20, par un ensemble de fonctions élaborant des entiers aléatoires de façon beaucoup plus sophistiquée.

Vous pourrez d'ailleurs personnaliser ainsi à votre gré ce programme, écrit en Basic Microsoft sur un Epson PX-8.

Voici quelques précautions utiles si vous manipulez des informations « sensibles. »

Ce n'est pas par découragement que les vilains petits curieux ont renoncé à semer dans vos bureaux les « punaises » qui leur rapportaient vos conversations. Ils disposent aujourd'hui de détecteurs qui recopient sans problème, à travers murs et planchers et à quelques dizaines de mètres tout ce qui s'affiche à grands flots d'électrons sur l'écran du tube cathodique de votre moniteur.

L'écran plat à cristaux liquides est du même coup devenu un « must ». Le Grid Compass fait fureur chez les militaires et on s'arrache les D.G. One dans certains services.

Tout aussi peu discrètes, et à proscrire absolument, sont les liaisons infrarouges de périphériques. Remettez des cordons s'il y a lieu à votre Apricot ou à votre IBM et attendez de pied ferme la nouvelle vague de « punaises » qui analyseront avec subtilité le crépitement des imprimantes !

Pierre BARNOUIN

Le programme ligne à ligne

Lignes 10 à 30 : Initialisation commune et choix

Lignes 35 à 55 : Chiffrement
- Initialisation (lignes 35 et 40)
- Saisie du clair, contrôle et encodage (lignes 45 et 50)
- Substitution n° 1 (ligne 55)
- Fin de saisie (ligne 60)
- Transposition, substitution n° 2 et impression (ligne 65)

Ligne 70 : Sous-routine commune de transposition

Lignes 75 à 115 : Déchiffrement
- Initialisation (lignes 75 et 80)
- Saisie du crypto sous contrôle (lignes 85 et 90)
- Transposition et substitution n° 2 (ligne 95)
- Substitution n° 1 (ligne 100)
- Décodage (ligne 110)
- Impression du clair (lignes 105 et 115)

D'un Basic à l'autre

L'instruction SWAP (ligne 70) échange les contenus des deux variables F(M) et F(I). Elle peut être remplacée par : W = F(M):F(M) = F(I):F(I) = W

L'opérateur XOR (lignes 65, 95, 100) est un « ou exclusif. » Il prend la valeur vraie (généralement – 1) si l'une seulement des deux assertions qui l'entourent est vraie, il est faux (généralement 0) dans le cas contraire.

ASS-DESAS

UN UTILITAIRE LANGAGE-MACHINE POUR LE CANON X-07

LES insatisfaits du Basic, les boulimiques de la petite fenêtre, les apôtres du NSC 800... bref, tous ceux qui possèdent un Canon X-07 et veulent en tirer plus encore, trouveront dans Ass-Desas un logiciel fait pour eux.

Sous un nom en forme de jeu de mots, l'*Ass-Desas* cache un Assembleur-Désassembleur destiné aux usagers du Canon. Mais pas à n'importe quels usagers ! De par ses caractéristiques et sa structure, ce logiciel sur cassette édité par Logi'stick exige une capacité mémoire minimum de 16 Koctets. Aussi, ceux qui — finances obligent — ne disposent pas d'une version « gonflée » de la petite machine tireront les conclusions qui s'imposent... Les autres peuvent envisager le chargement du programme. Si, en outre, ils disposent d'une interface TV et même d'une imprimante, alors tout est possible : le grand confort, en quelque sorte.

Peu épaisse, succincte même, la documentation rassemble sept feuilles sommairement agrafées qui forment un petit manuel de 28 pages et qui doivent tout vous apprendre sur la manipulation du logiciel. Les premières pages constituent une introduction au langage-machine, destinée à ceux qui ne se sont jamais essayés à cette langue barbare (idiome Z80, naturellement ; le processeur du X-07, le NSC 800, ressemblant fort au Z80). Les explications données sont évidemment des plus restreintes, et le novice fera bien d'équiper sa bibliothèque d'ouvrages plus conséquents qui prépareront le terrain.

C'est aussi sur ce sage conseil que s'achèvent les explications données. Les pages suivantes se décomposent en trois parties essentielles. Les premières contiennent le mode d'emploi de l'Assembleur, puis celui du Moniteur-Désassembleur. La dernière partie comporte neuf pages : il s'agit d'un tableau complet des mnémoniques du Z80 et de leur symbolisation.

Éditeur plein écran

La cassette contient quatre programmes : l'Assembleur, le Moniteur-Désassembleur, un Assembleur-Désassembleur rassemblant les deux premiers en un seul fichier, et un Désassembleur destiné à l'usage de l'écran vidéo TV.

La mise en mémoire de l'Assembleur ne pose pas de problème : un court programme Basic sert de chargeur aux codes de l'Assembleur proprement dit

qui réside après lui sur la bande. Le chargement s'effectue en moins d'une minute ; c'est donc assez rapide. L'Assembleur seul occupe les adresses &H3400 à &H3D11. Il laisse en fait plus de 11 Ko disponibles pour le travail suivant, si vous aviez 16 Ko au départ.

Pour l'entrée du programme-source, la technique « éditeur Basic » a été retenue. En fait, vous entrez au clavier la suite des mnémoniques composant le source comme vous le feriez pour un programme Basic. Vous devez donc numéroter les lignes et vous bénéficiez de tous les avantages de l'éditeur plein écran. Ceci implique toutefois quelques contraintes : la première ligne du source doit contenir un « (» et rien d'autre, tandis que la dernière ligne ne contiendra qu'un «) ». Ces deux caractères servent d'indicateurs de début et de fin du programme destinés à l'Assembleur que l'on lance par un RUN 60000. Le chargeur Basic contenant cette ligne devra donc être conservé en mémoire : pas de NEW avant d'écrire la routine !

Cinq codes d'erreur

L'Assembleur accepte les commentaires en fin de ligne, après le caractère « # ». Dommage que l'écran ne soit pas plus grand !

Il est possible d'utiliser des étiquettes (labels) de deux caractères au plus. L'Assembleur les reconnaît à la présence du signe « : » qui les précède obligatoirement : par exemple, nous pouvons définir l'étiquette « :L1 ».

Les mnémoniques doivent respecter la syntaxe Zilog. Il est en particulier indispensable de ne jamais omettre l'espace qui précède le(s) paramètre(s), car l'Assembleur ne signalerait pas l'erreur, et assemblerait avec un code erroné. C'est gênant.

Lorsqu'une référence est faite à une étiquette, son nom est signalé par un « L » qui le précède. Avec l'étiquette définie précédemment, « :L1 », nous pourrions donc écrire par exemple : "JR NZ,LL1".

Les paramètres en question peuvent être fournis en décimal, hexadécimal ou négatifs ($FF, 255, −$1 et −1 sont équivalents). Si les valeurs données sont aberrantes, l'Assembleur ne signalera pas, là non plus, l'erreur commise.

Lors de la phase d'assemblage, cinq codes d'erreur sont susceptibles d'apparaître, en fonction des défauts détectés au sein du programme-source. S'il s'agit de l'erreur 0, c'est bon signe : aucune erreur n'a été détectée. Il ne reste plus alors qu'à soumettre la routine à l'épreuve du feu.

Dans le cas contraire, le numéro de la ligne fautive qui est affiché en même temps vous aidera dans vos corrections.

La cassette contient deux versions du Moniteur-Désassembleur. L'une d'elles est destinée à l'usage d'un téléviseur.

Le programme Moniteur offre cinq possibilités à l'utilisateur : désassemblage, édition du contenu-mémoire, visualisation des pointeurs, sauvegarde et chargement langage-machine. Un menu permet de sélectionner l'option désirée.

Contrairement à l'Assembleur, le Désassembleur peut être chargé n'importe où en mémoire. Le choix de l'adresse d'implantation est donc le premier problème que les explications fournies dans le manuel n'aideront pas facilement à résoudre.

Le désassemblage s'effectue ligne à ligne sur l'écran, mais on peut aussi obtenir qu'il soit listé sur imprimante. L'édition mémoire affiche huit octets seulement sur l'écran incorporé (48 sur un moniteur TV). L'affichage peut se faire en décimal ou en hexadécimal. Cette dernière possibilité offrant en prime l'affichage du caractère ASCII correspondant au code. En manipulant un curseur à l'aide des touches habituelles, il est possible de modifier le contenu d'une adresse en changeant simplement la valeur affichée, ou le code ASCII associé.

Avec le moniteur, la zone mémoire pour fichiers est accessible, et modifiable à volonté. Cela pourra sans doute conduire à des résultats amusants !

Enfin, les routines SAVE et LOAD permettent la sauvegarde et la récupération sur cassette de vos routines, ou de la zone mémoire de votre choix, comprise entre les adresses spécifiées. L'option LOAD permet au besoin de recharger en mémoire à une adresse différente de l'adresse d'implantation initiale.

L'*Ass-Desas* est un logiciel intéressant dont sauront faire bon usage tous ceux que tente le langage-machine. Ses performances d'ensemble sont correctes et suffisantes pour faire beaucoup de choses... Comme pour tous les logiciels du genre, l'utilisateur devra bien connaître sa machine pour tirer le meilleur parti des possibilités offertes. Les débutants devront s'attendre à bien des peines, que la documentation insuffisante ne pourra aplanir. Ils devront en tout cas se documenter sérieusement avant de s'y frotter. Les autres savent déjà où ils vont et pourront faire beaucoup de petites routines !

Robin BOIS

Le logiciel en quelques lignes

Nom : *Ass-Desas*
Ordinateur : *Canon X-07*
Forme : *cassette*
Edité par : *Logi'stick*
Distribué par : *DDI*
Prix public : *150 FF*
Principales orientations : *assembleur, moniteur, désassembleur*

UN BASIC ÉTENDU POUR ZX SPECTRUM

COMME on le sait, le Basic de Sinclair est original à bien des égards, et notamment par ses lacunes. Pratiquement aucun traitement des erreurs, par de PRINT USING, d'ELSE, de DEL... Ère Informatique propose une extension du Basic sur cassette qui apporte une quinzaine d'instructions supplémentaires.

Le Basic étendu proposé par Ère Informatique comporte 15 nouvelles instructions, très facilement accessibles (un point d'exclamation précède chaque mot clé), et qui complètent heureusement le Basic du Spectrum.

Signalons tout d'abord des instructions que l'on ne trouve que dans certains langages très structurés, et qui rendent possible une écriture plus élégante des programmes (en évitant l'emploi de GOTO par exemple). On trouve en particulier la structure WHILE...WEND (Tant que - condition - Fin de tant que), la structure REPEAT...UNTIL... (Répète - instructions - Jusqu'à - condition). On trouve encore le ELSE en complément à IF...THEN... et des instructions de traitement des erreurs, du type ON ERROR... et RESUME, qui offrent plusieurs options. Ainsi RESUME 0 provoque un retour à la ligne où l'erreur s'est produite, RESUME NEXT permet le retour à la ligne qui suit l'erreur, RESUME nnn fait reprendre à la ligne nnn. Le PRINT peut désormais être accompagné d'une instruction de formatage (USING) qui fait ou non référence à une chaîne de caractères définie ailleurs (on écrira, par exemple : A$ = "###.##", puis PRINT USING A$, nnn). Tout cela apporte déjà un net « plus » au Basic.

Mais d'autres instructions viennent simplifier le travail d'écriture et de mise au point des programmes. C'est le cas d'AUTO (numérotation automatique), de DEL (suppression sélective des lignes), de RENUM (renumérotation de tout ou partie d'un programme).

Deux nouvelles instructions graphiques

Le mode TRACE est disponible pour afficher les numéros des lignes exécutées avec des options qui le rendent particulièrement performant. On peut par exemple fair exécuter un programme Basic ligne par ligne ou faire imprimer les numéros de ligne et les variables rencontrées par le programme (ce qui évite d'encombrer l'écran). Une petite curiosité au passage : la présence de l'ordre FIN dont l'intérêt devient évident quand on fait appel à l'ordre ON ERROR nnn.

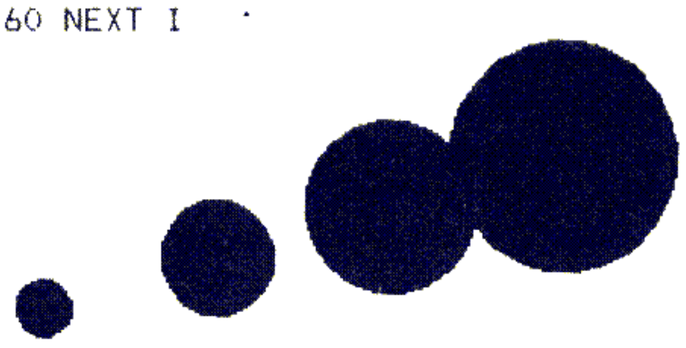
En effet, les erreurs une fois traitées, le programme est en quelque sorte protégé des fautes de l'utilisateur. Mais ces erreurs risquent tout de même de provoquer un plantage du programme. Or, le STOP est lui aussi considéré comme une erreur. L'utilisation de FIN est donc indispensable pour arrêter le programme.

Ce n'est pas tout. Deux autres instructions, présentes sur de nombreux appareils, mais pas sur le Spectrum, sont désormais disponibles, et permettent des effets remarquables : il s'agit de GIANT et PAINT.

Avec GIANT, on écrit en caractères

Exemple d'exécution

```
10 REM ESSAI DE PAINT
20 INK 6: PAPER 1: CLS
30 FOR I=1 TO 4
40 CIRCLE 60*I-40,20*I,10*I
50 PAINT 60*I-40,20*I,6
60 NEXT I
```



agrandis à n'importe quel endroit de l'écran (après avoir positionné le curseur d'écriture grâce à PRINT AT). Contrairement à ce qui se passe dans la plupart des micro-ordinateurs, on peut préciser à l'aide de deux paramètres la taille de l'agrandissement. Cette instruction originale, utilisée dans une boucle, rend possible des effets de « zoom » assez étonnants. Il est même possible d'agrandir avec cette instruction les caractères graphiques définis par l'utilisateur (j'en ai fait l'expérience), mais cela n'est pas précisé dans le manuel et ce n'est peut-être pas conseillé.

Plus classiquement, PAINT colorie une surface fermée désignée par l'un quelconque de ses points.

Les possesseurs de « microdrives » apprécieront très certainement le fait que la notice indique la manière de procéder pour copier le programme sur cartouche, ce qui le rend plus rapide à charger. Le logiciel est enregistré en ligne 0 ; il est donc possible de sauvegarder en une seule fois, sans manœuvre particulière, à la fois le Basic étendu et un programme réalisé dans ce mode.

Le logiciel en quelques lignes

Nom : *Basic étendu*
Ordinateur : *ZX Spectrum*
Forme : *cassette*
Edité et distribué par : *Ère Informatique*
Prix public : *180 FF*
Principales orientations : *extension du Basic, traitement d'erreur, édition de programmes, mode TRACE...*

La notice (14 pages au format de la cassette...) contient le minimum d'explications utiles pour tirer parti des 15 nouvelles instructions. Adaptée aux versions 14 et 48 Ko du Spectrum, cette extension du Basic coûte 180 FF ttc.

Jacques DECONCHAT

LES COUPS D'ŒIL DE LIST

XPER POUR LE COMMODORE 64

SI XPer travaille sur des fiches, il se démarque pourtant beaucoup des logiciels de gestion de fichiers : il met en œuvre certains des mécanismes propres aux systèmes-experts. Ses applications sont variées (chaque utilisateur en inventera), mais elles restent étroitement limitées par la mémoire de l'ordinateur.

Commençons par signaler que nous avons testé XPER sur C.64, mais qu'il en existe des versions adaptées à l'Apple II et à L'IBM-PC. Ce logiciel est vendu sous la forme d'une disquette. Un bon point pour la documentation d'accompagnement : un livret de 156 pages, complet, lisible et auquel ne manquent ni la table des matières, ni l'index.

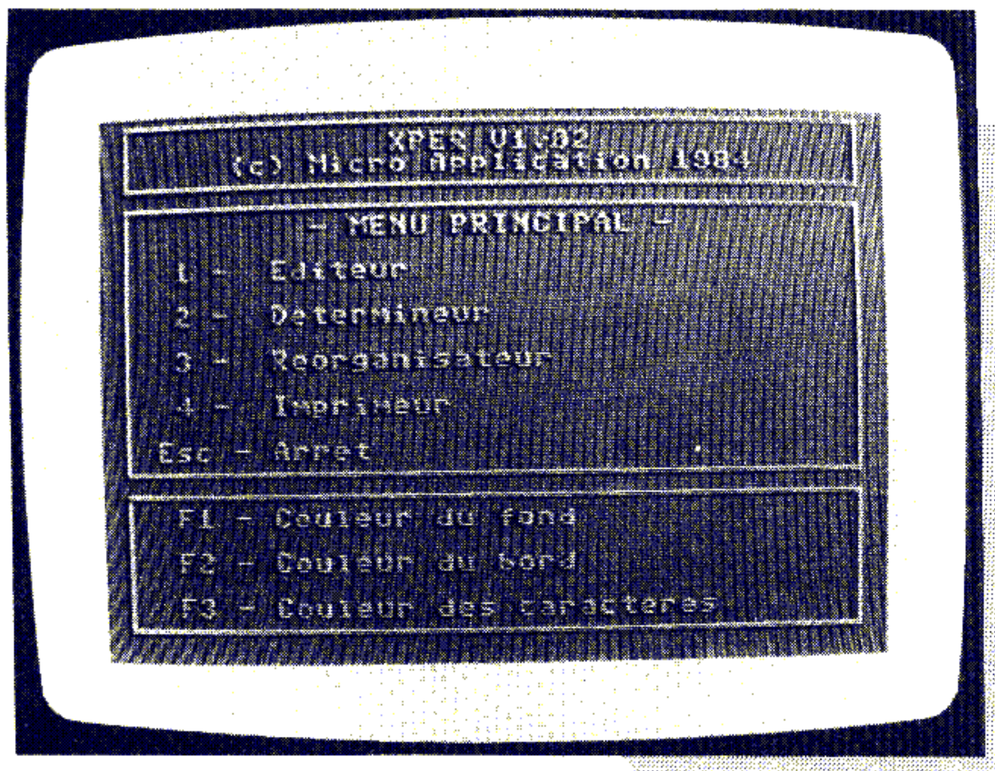
Disquette obligatoire

XPer se veut (documentation dixit) le premier logiciel d'IAO disponible sur micro-ordinateur. Un « AO » (Assisté par Ordinateur) de plus dans l'univers de l'informatique : sachez que le I signifie « Identification ». Si vous ne vous sentez pas mieux renseigné pour autant, sachez qu'il s'agit en fait d'un système de gestion de bases de données... Mais au fait, il en existe déjà bien d'autres ! Alors, qu'est-ce qui fait l'originalité de celui-là ?

La documentation se compose de trois parties essentielles : la première permet à l'utilisateur de se familiariser avec les concepts qui sous-tendent l'existence d'un système du type d'XPer. Huit pages sont consacrées à cette première approche. La seconde partie est un cours complet en 9 leçons qui permet à l'utilisateur de connaître le fonctionnement du logiciel dans ses grandes lignes. Enfin, les 60 dernières pages rassemblent la documentation de référence du logiciel.

L'installation d'XPer sur le commodore 64 nécessite un lecteur de disquettes, d'une part parce que le logiciel est proposé sur ce type de support, et d'autre part parce qu'on ne pourrait imaginer de faire fonctionner une base de connaissances (variante de la base de données, comme nous le verrons) un tant soit peu sérieuse en utilisant un magnétophone. L'utilisation d'une imprimante n'est, elle, pas indispensable ; mais si vous en avez une, ce sera parfait.

XPer est partagé sur la disquette en 4 programmes distincts qui correspon-



Au menu principal, le choix du programme, mais aussi celui des couleurs

dent aux principales fonctions du logiciel : édition, recherche, modification et impression. Le chargement de chaque programme est généralement long, ce qui est dû au volume de chacun et à la lenteur bien connue des disquettes du Commodore.

Sur le 64, XPer accepte de traiter des fichiers contenant au maximum 100 individus, chacun d'eux pouvant être décrit par 40 variables, elles-mêmes découpées en 200 modalités. L'espace mémoire disponible est de 4688 octets. Un fichier exemple fourni avec le programme rassemble 38 individus, 17 variables et 43 modalités, tout en laissant 3495 octets disponibles. L'espace mémoire semble donc suffisant, même s'il n'est pas immense à première vue.

La création d'une base de connaissances s'effectue sous le contrôle de l'*éditeur*. Il s'agit de définir le titre, les variables de description des individus et leurs modalités, pour remplir un tableau dit « croisé » qui les rassemblera. Pour notre premier essai, nous avons choisi de créer une base de données « vins », histoire de joindre l'utile à l'agréable... Nos individus (Gevrey-Chambertin, Mouton-Rotschild) se sont donc vu décrire en termes de variables (robe, corps) et modalités (rouge, gouleyant...). Ce genre de travail peut être relativement fastidieux, surtout si la cave est bien garnie ! Mais l'éditeur possède des facilités qui simplifient l'ouvrage avec une aide disponible à tout moment. A l'issue de ce premier essai, notre base de connaissances minimale rassemble 8 individus, 3 variables et 11 modalités. La sauvegarde du fichier consomme seulement 6 blocs sur la disquette, et cela en 6 fichiers distincts.

Le programme *détermineur* est à la base de toute exploitation du fichier. Ici encore, une familiarisation minimale permet de se débrouiller dans les commandes qui s'enchaînent. Le détermineur est capable de vous dire quel est le vin qui figure dans le fichier **et** qui correspond aux critères choisis : rosé, produit en Alsace, ayant de la cuisse. A cela s'ajoute la possibilité de savoir à tout moment pourquoi tel vin a été éliminé, et quels sont les critères ayant provoqué son rejet. Voilà donc une première différence avec une base de données traditionnelle.

Un logiciel complexe

De plus, lors de la création de la base de connaissances, il est possible d'utiliser un système de filiation entre des variables « mères » et « filles » qui permet de simplifier les critères de détermination ou de mieux structurer les données.

Autre possibilité peu courante, XPer peut calculer un taux de similitude entre individus. Pour rester dans le domaine de l'oenologie, cela revient à dire, par exemple que vous pourrez savoir par quel autre nectar réconfortant remplacer le vin qui fait défaut le jour où vos invités sont particulièrement difficiles...

Encore possible, le regroupement d'individus correspondant à certaines caractéristiques de votre choix. Le fichier « félins » qui sert de support didactique fourni avec le logiciel permet, grâce à ce type de groupement, de révéler que tous les félins africains de moins de 80 kg, qui grimpent aux arbres ont des griffes rétractiles, des canines de petite taille, et ronronnent (caressez-les pour voir !). On le démontre au prix d'un petit jonglage avec diverses commandes qui deviennent peu à peu plus complexes : nous en sommes à la 6e leçon. Nous n'avons pas utilisé notre base personnelle pour ce type d'essai ; sa trop petite taille, ses imprécisions et ses lacunes n'auraient pu que nous indiquer que tous les vins blancs viennent d'Alsace ! Toutefois, seule notre paresse aurait été mise en cause au vu de cette conclusion : l'analyse d'une mauvaise base de données ne peut donner que de mauvais résultats.

XPer enfin peut vous aider dans le choix des critères de sélection en éliminant de lui-même les critères non discriminants, c'est-à-dire ceux qui ne permetent pas de différencier plusieurs individus à un instant déterminé de la sélection. Si vous avez une imprimante, toutes les opérations effectuées par le déterminateur pourront être suivies pas à pas.

Deux derniers programmes enfin, appelés *réorganisateur* et *imprimeur*, permettent d'effectuer une liaison entre des bases de données ayant des individus ou des critères communs ; et d'effectuer des modifications dans un fichier déjà créé. Enfin, le listage sur papier du contenu intégral de la base de connaissances est une possibilité qui pourra se révéler précieuse.

XPer n'est donc pas un gestionnaire de fichiers comme les autres, les capacités que nous avons découvertes l'attestent. Au chapitre des critiques nous retiendrons que la taille des programmes rend les temps de chargement très longs, bien qu'à la vérité, on n'en change pas constamment. La complexité du logiciel, dont nous n'avons fait qu'effleurer les possibilités, rend indispensable un apprentissage sérieux. Le manuel offre 9 leçons bien construites, mais l'utilisateur devra découvrir aussi en pratiquant.

XPer paraît donc susceptible de supporter de nombreuses applications. Son usage se conçoit mieux avec des fichiers importants ; aussi son emploi sur des micro-ordinateurs aux capacités limitées peut sembler d'un intérêt discutable. Tout le problème est : XPer pourquoi faire ? Gageons que ses utilisateurs ne manqueront pas d'idées.

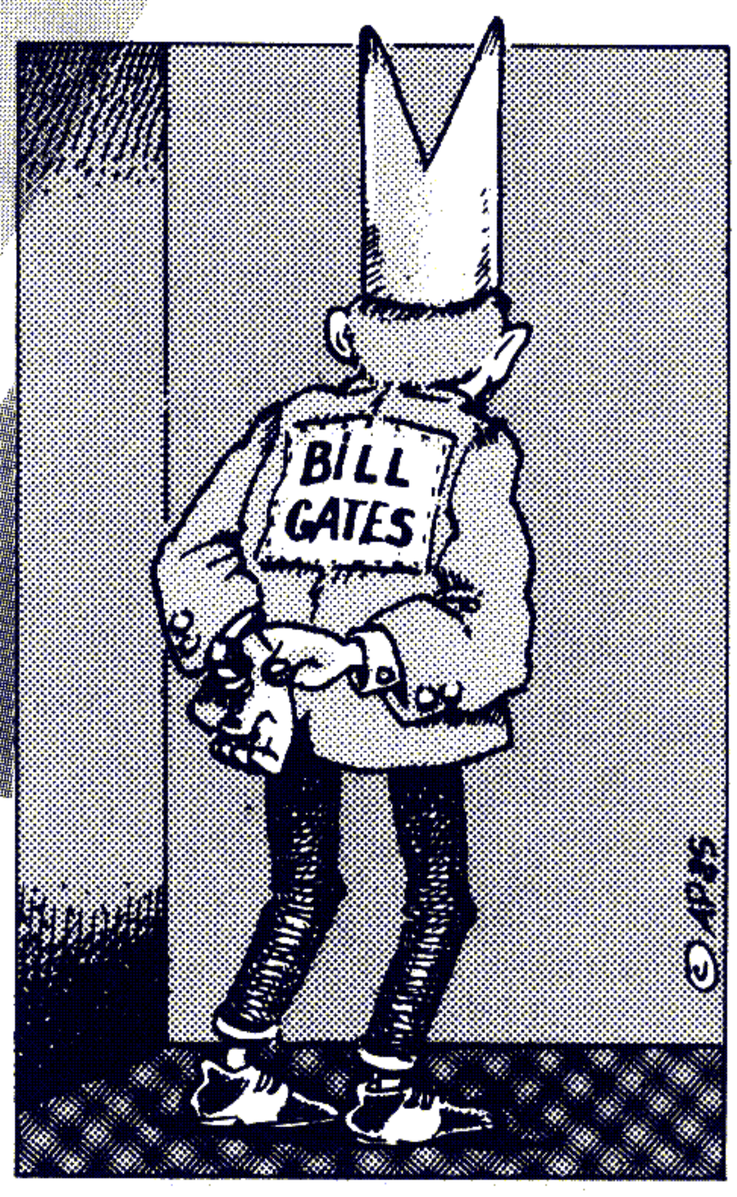
Jean-Pierre LALEVÉE

Le logiciel en quelques lignes

Nom : *XPER*
Ordinateur : *Commodore 64*
Forme : *disquette*
Edité et distribué par : *Micro Application*
Auteur : *Jacques Lebbe*
Prix public : *950 FF*
Principale orientation : *gestion de bases de connaissance*

UN ÉDITEUR IDÉAL

LE chemin qui va du Basic aux logiciels de traitement de texte est court. En le suivant, il devient possible de traiter une liste de programme comme s'il s'agissait d'un texte : se « promener » n'importe où dans cette liste, la corriger, la transformer très facilement. En sens inverse, le chemin est aussi court. Et on retrouve sa liste Basic corrigée.



Ah, Bill Gates, que nous te voulons de mal ! Quand tu écrivis ton célèbre Basic Microsoft, prévoyais-tu les crises de nerfs que vaudraient, à des milliers de maladroits et/ou perfectionnistes, les instructions par le biais tortueux desquelles on peut, après avoir subi toutes les étapes pénitentielles d'une lente cérémonie expiatoire et purificatrice, être admis à l'honneur de modifier une ligne de programme ? Certes non. Aujourd'hui, les nouveaux Basic ont des éditeurs commodes, parfois même pleine page, qui ont remplacé plus qu'avantageusement les tortures savantes que nous connûmes pour notre honte.

Bien plus, nous disposons maintenant de traitements de texte standard. Dans la plupart des cas, ils permettent d'opérer simplement, même dans les Basic anciens encore soumis, en principe, aux contorsions diaboliques et primitives issues du vieux système Microsoft qui nous ouvrit les portes de la programmation. Prenons le cas des langages de base des TRS-80, des Apple II et III, et du Dai (sans doute les lecteurs de *LIST* pourront-ils facilement allonger la liste) : pour chacun d'eux, il existe une manipulation élémentaire qui permet, à l'aide d'un traitement de texte classique,

Sur TRS-80 modèles III ou IV

Matériel : modèles III ou IV avec Basic Microsoft usuel et traitement de texte SuperScripsit.

1. Ecrire le programme sous SuperScripsit, le sauver en lui donnant un nom de fichier (par exemple, PGM:0).

2. Quitter le document (par Ctrl Q), appeler le transcripteur « Scripsit vers Ascii », ce qui fabrique un document Ascii avec, par exemple, le nom de fichier PGMBAS:0.

3. Passant en Basic, faire LOAD "PGMBAS", puis SAVE "PGMBAS". Le programme est alors normalement enregistré comme s'il avait été, depuis le début, écrit avec la procédure Basic usuelle ; on peut alors le renuméroter, l'exécuter, etc.

4. Partant maintenant d'un programme Basic (PGMBAS, par exemple), on peut l'amener sous SuperScripsit pour le triturer tout à loisir avec la souplesse maximale en se livrant simplement à la manœuvre inverse de la précédente : charger par LOAD "PGMBAS", faire SAVE "PGM :0",A (ce qui le met en Ascii). Puis, par l'utilitaire de transcription d'Ascii vers Scripsit, lui donner un nom de fichier Scripsit (PGM:0) et l'appeler alors à l'écran comme un texte ordinaire.

UN ÉDITEUR IDÉAL

Sur Apple II ou III

Matériel : Apple II+, IIe, IIc, III avec Applesoft ou Business Basic — selon le cas — et traitement de texte AppleWriter, ou autre.

1. Ecrire le programme Basic sous AppleWriter, lui donner un nom de fichier comme MYFILE compatible avec le DOS (SOS pour Apple III, par exemple), le sauvegarder puis, tout simplement, faire EXEC MYFILE. Il est alors enregistré en mémoire comme un vrai programme Basic et peut donc être exécuté, renuméroté, etc.

2. Pour montrer comment opérer de manière inverse, prenons par exemple un programme TOTO occupant les lignes 10 à 2500. On doit le transformer en un fichier texte utilisable sous AppleWriter (ou autre) en lui accolant un bout de programme supplémentaire dont le but est d'effectuer cette métamorphose.

• *Sur les Apple II :*

```
10000 D$ = CHR$(4)
10010 PRINT D$; "OPEN
      TOTO.TEXT"
10020 PRINT D$; "WRITE
      TOTO.TEXT"
10030 POKE 33,30
10400 LIST 0,2500
10500 PRINT D$; "CLOSE TOTO.
      TEXT"
10600 TEXT:END
```

• *Sur Apple III :*

```
10000 FILES$ = "TOTO.TEXT"
10100 CREATE FILES$,TEXT
10200 OPEN#1 AS OUTPUT,FILES$
10300 OUTPUT#1
10400 OUTREC=255
10500 LIST 0 TO 2500
10600 CLOSE#1
10700 END
```

d'écrire et de modifier un programme Basic et, surtout, de pouvoir prendre une ancienne liste afin de la retravailler. Voici donc quelques procédures adaptées à ces différents cas. Elles devraient nous épargner, à tous, de longues soirées d'énervement !

Comme on le voit par les quelques exemples proposés, les procédures à utiliser ne sont guère difficiles à mettre en place... quand on les connaît (encore faudrait-il que les différents constructeurs veuillent bien signaler largement ce genre de possibilités). L'utilité de ces procédures est évidemment encore plus grande lorsqu'il s'agit de programmeurs professionnels et/ou occasionnels ayant à reprendre fréquemment telle ou telle partie de programme dans des ensembles plus vastes : ils passeront par exemple alternativement sous traitement de texte pour faire les modifications indispensables — sélection d'un « bloc » déterminé, changement systématique d'une occurrence par une autre, etc. —, puis reviendront au Basic pour la rénumérotation systématique et ainsi de suite. Le gain de temps peut être alors vraiment significatif. Toute extension à cette liste « à malices » sera évidemment la bienvenue.

André WARUSFEL

Lancer ces programmes par RUN 10000 : notre TOTO est alors disponible sous forme accessible à AppleWriter ou tout autre traitement de texte usuel.

Sur Dai

Matériel : Dai avec DAItext.

1. Ecrire le programme, l'éditer en totalité, puis BREAK, BREAK.

2. Passer en UTILITY et relever les adresses de début et fin de fichier.

3. Placer DEBUT en #A2 et #A3, FIN+2 en #A4 et #A5.

4. Réserver de la place, sous Basic, par un CLEAR suffisant.

5. Compiler le programme par POKE #135,2. Lister, ou exécuter, c'est prêt !

Inversement, pour insérer dans DAItext un programme Basic déjà écrit (par exemple, TOTO), il faut : 1. Editer le programme EDIT, BREAK, BREAK.

2. Sous UTILITY, noter les adresses de DEBUT et FIN, en #A2/#A3 et #A4/#A5.

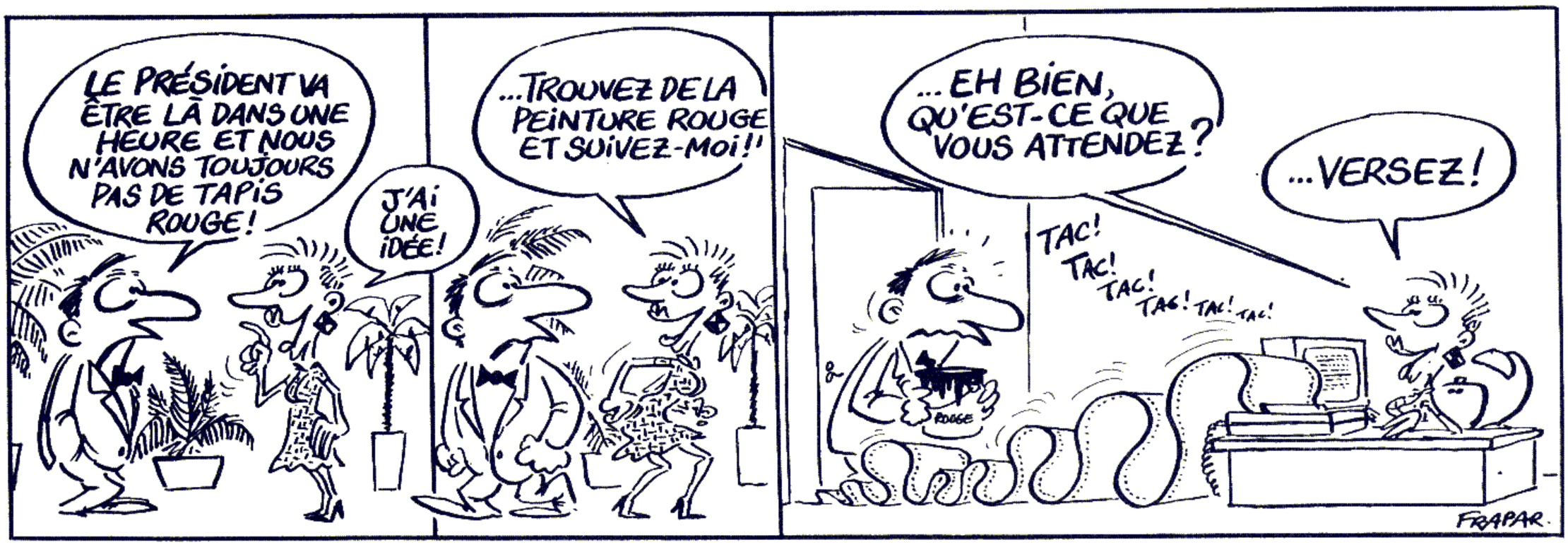
3. Déplacer le fichier (commande MOVE), de façon qu'il commence en #3000 et repérer la nouvelle FIN de fichier. Sauver ce fichier sur mémoire de masse par W3000 < FIN > NOM.

4. Charger DAItext v.20 et le lancer, si la version n'est pas « auto-start ». Appeler l'option MODEM, puis la commande « recevoir ». Sortir de DAItext par RESET, puis recharger le fichier NOM (R NOM).

5. Réajuster les pointeurs #A2/#A3 et #A4/#A5 aux bonnes valeurs (#3000, nouvelle FIN+2). Faire G(o)19BB qui a pour effet de transformer le fichier EDIT en fichier DAItext.

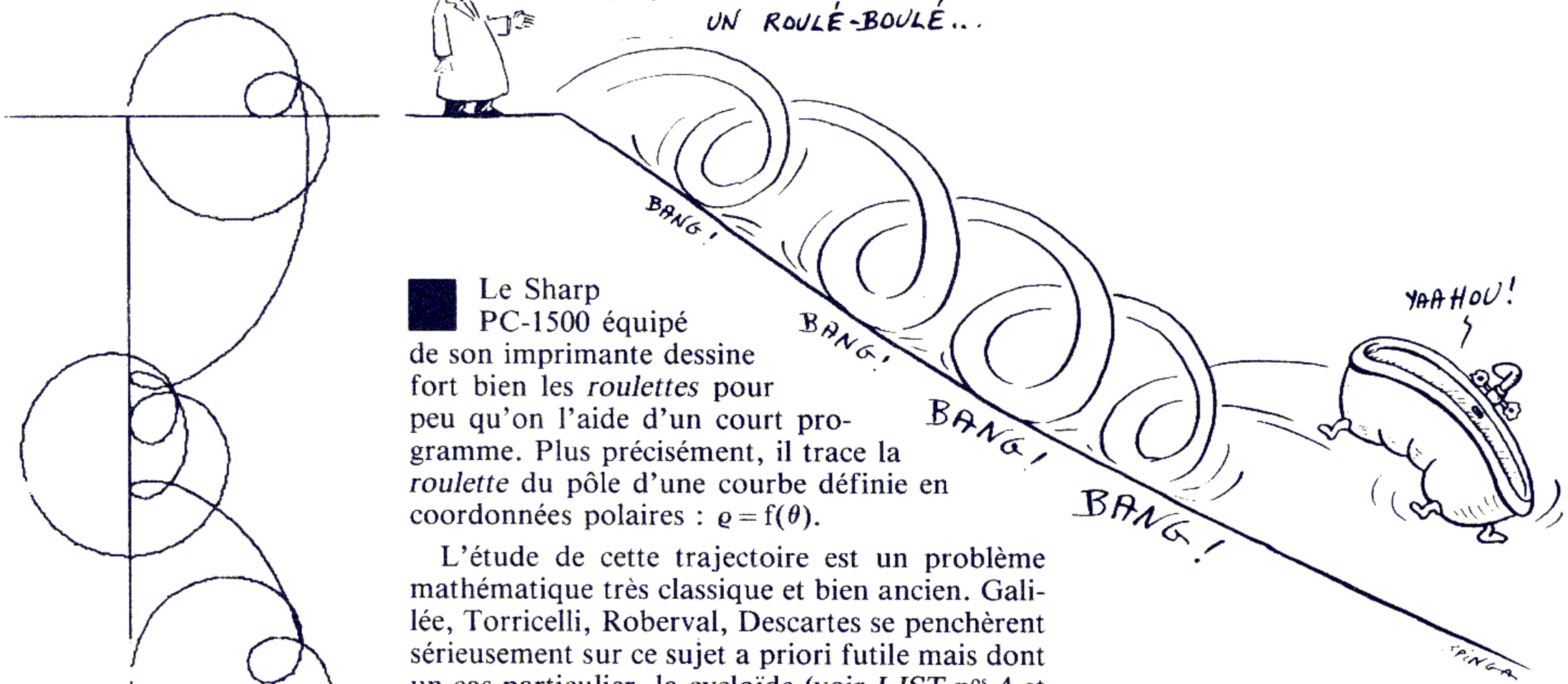
Ce dernier est prêt à être manipulé par DAItext.

Ces procédures sont utilisées pour transmettre ou recevoir, par voie télématique, des fichiers autres que du texte (binaire, Basic, Pascal, etc.).



SUR DES ROULETTES

LA roue bien ronde roulant sans glisser sur un sol plat n'est qu'un cas idéal certes, mais particulier. Comme on avait commencé à le voir dans LIST n° 6, on peut imaginer bien d'autres formes de roues, et observer les « roulettes » ainsi obtenues, c'est-à-dire les trajectoires décrites dans le plan fixe par tel point donné de la courbe en mouvement.



Le Sharp PC-1500 équipé de son imprimante dessine fort bien les *roulettes* pour peu qu'on l'aide d'un court programme. Plus précisément, il trace la *roulette* du pôle d'une courbe définie en coordonnées polaires : $\varrho = f(\theta)$.

L'étude de cette trajectoire est un problème mathématique très classique et bien ancien. Galilée, Torricelli, Roberval, Descartes se penchèrent sérieusement sur ce sujet a priori futile mais dont un cas particulier, la cycloïde (voir *LIST* nos 4 et 6), servit en fait de banc d'essai pour les nouvelles méthodes de calcul infinitésimal.

Le programme (page suivante) étudie le tracé de la courbe sur l'arc [T0, T1] et T représente l'angle courant θ (on est en coordonnées polaires). On pourra utiliser un programme indépendant de tracé de courbes pour connaître la forme et la valeur du maximum du rayon vecteur sur l'intervalle. Ce rayon fournira l'échelle du dessin. Bien évidemment, on aura intérêt à éviter les valeurs non définies ou infinies de ϱ.

Après avoir lancé l'exécution d'un RUN ou DEF A, cinq questions sont posées par le programme :

- l'équation de la roue, qu'on écrit sim-

```
ROULETTE
PRECI.:  40
NbROUL:  3
MAX PI:  1.5
BORNES: -6.3  ,  6

 30: "I"PI=.5+COS I:RETURN
```

SUR DES ROULETTES

Roulette
Programme pour PC-1500 et CE-150
Auteur Christophe Masurel

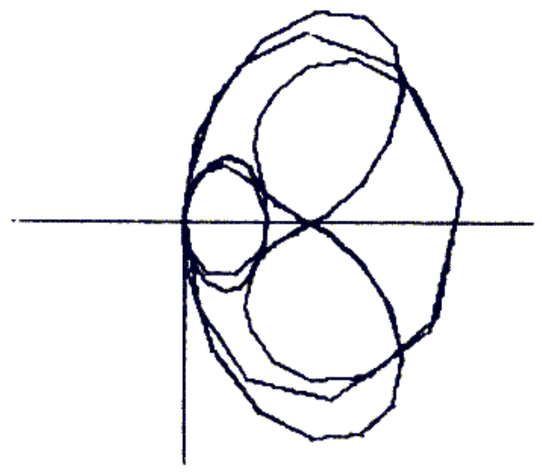

```
10:REM ROULETTE
20:GOTO "A"
30:"f"PT=--------
   --------------
   --------------
   --------------
   --------------
   -------
50:"E"RESTORE "f"
   :I=(PEEK &78BE
   -128)*256+PEEK
   &78BF+6:INPUT
   "EQUATION ROUE
   =?";J
60:J=31679:FOR I=
   ITO I-10+PEEK
   (I-7):IF PEEK
   J<>13POKE I,
   PEEK J:J=J+1:
   NEXT I
70:POKE I,58,241,
   153,13:RETURN
80:"A"GOSUB "E"
90:"S"RADIAN :
   CLEAR :INPUT "
   PARTITION:N=";
   N,"Nb ROUES=";
   NR,"MAX DE PT=
   ";PM
100:INPUT "BORNE0=
   ";T0,"BORNE1="
   :T1
110:TEXT :CSIZE 1:
   COLOR 0:LPRINT
   "ROULETTE":
   LPRINT "PRECI.
   : ";N:LPRINT "
   NbROUL: ";NR:
   LPRINT "MAX PT
   : ";PM
120:LPRINT "BORNES
   : ";T0;" , ";
   T1:LPRINT
125:INPUT "COEFF.
   GLISSEMENT: Q=
   ";Q:LPRINT "Q=
   ";Q
130:LLIST "f"
140:REM  INITIALIS
   ATION
150:NR=NR-1:CO=105
   /PM
160:R=(T1-T0)/(2*N
   ):H=2*R
170:REM REPERE-AXE
   S
180:GRAPH :
   GLCURSOR (70,-
   60):SORGN :
   LINE (-70,0)-(
   150,0):LINE (0
   ,-100)-(0,0)
190:J=1:C1=INT ((N
   +1)/NR):COLOR
   2
200:DIM L(3),M(3)
   INPUT "COEFF.G
   LISSEMENT:Q=";
   Q
210:T=T0-R:GOSUB "
   f":M(0)=PT
220:T=T0:GOSUB "f"
   :M(1)=PT
230:TV=M(1)*R/(M(1
   )-M(0))
240:IF M(1)-M(0)<0
   LET VE=Π+ATN (
   TV):GOTO "T"
250:VE=ATN (TV)
260:"T"X=-CO*M(1)*
   COS (VE):Y=CO*
   M(1)*SIN (VE)
270:GOSUB "L":
   GLCURSOR (Y,-X
   ):T=T0
280:FOR I=1TO N:T=
   T+R:GOSUB "f":
   M(2)=PT
290:T=T+R:GOSUB "f
   ":M(3)=PT
300:FOR K=1TO 3
310:L(K)=∫(M(K)*M(
   K)+(M(K)-M(K-1
   ))*(M(K)-M(K-1
   ))/(R*R))
320:NEXT K:S=L(1)+
   4*L(2)+L(3):SO
   =SO+R*S/3
330:DE=(M(3)-M(2))
   /R:TV=M(3)/DE
340:IF M(3)-M(2)<0
   LET VE=Π+ATN (
   TV):GOTO "R"
350:VE=ATN (TV)
360:"R"X=CO*(Q*SO-
   PT*COS (VE)):Y
   =CO*PT*(SIN (V
   E)):LINE -(Y,-
   X)
370:IF I=C1LET J=J
   +1:C1=INT (J*N
   /NR+.5):AT=T:
   GOSUB "L"
380:GLCURSOR (Y,-X
   ):M(0)=M(2):M(
   1)=M(3):NEXT I
   :LINE (0,-X)-(
   0,0):TEXT
390:INPUT "AUTRES
   VALEURS?(O,N)"
   ;R$:IF R$="O"
   GOTO "S"
400:END
410:"L"COLOR 2:RO=
   T+VE:T=T0:
   GOSUB "f":U=X+
   CO*PT*COS (T-R
   O):V=Y+CO*PT*
   SIN (T-RO)
420:GLCURSOR (V,-U
   ):T=T0-R
430:FOR Z=1TO N:T=
   T+H:GOSUB "f"
440:U=X+CO*PT*COS
   (T-RO):V=Y+CO*
   PT*SIN (T-RO)
450:LINE -(V,-U):
   NEXT Z:T=AT:
   COLOR 1:RETURN


STATUS 1                1501
```

Les lignes 30 à 70, réalisant l'autoprogrammation, sont extraites de *l'Ordinateur de poche* n° 11, page 36.

Quelques courbes particulières

- 1/2 + SIN T
- SIN T
- 1/2 + COS T
- 4/5 + COS T
- 1/SQR(1 − 0.5*SIN(2*T)^2)
- 1/(4 + COS(3*T))
- SQR(2 + COS(2*T))
- 3 + COS(3*T)
- 1/(SQR(1 + SIN(2*T)) + SQR(1 − SIN (2*T)))
- (SQR(1 + SIN(2*T)) + SQR(1 − SIN (2*T)))



```
ROULETTE
PRECI.:   40
NbROUL:   3
MAX PT:   1.3
BORNES:  -6.285  ,  0


 30:"f"PT=SIN T+.5*SIN (2*T)+(1/3)*
    SIN (3*T):RETURN
```

plement en toutes lettres sous la forme d'une fonction de la variable T (auto-programmation, voir *l'Ordinateur de poche* n° 11),

- la partition, comprise entre 20 et 100, soit simplement le nombre de points du tracé (précision),
- le nombre de positions, 3 au minimum, de la roue qui roule,
- les bornes T0 et T1 de l'intervalle (les T extrêmes de l'arc, en radians),
- le coefficient Q de glissement (entre 0 et 1).

Grâce à ce dernier coefficient, on peut introduire un glissement de la « pierre » sur le « sol ». Pour la valeur 1 aucun glissement ne se produit : le tracé est celui de la *roulette*. Pour 0 la roue patine sur place : c'est la *glissette*. Et pour 0 < Q < 1, on passe par les cas intermédiaires.

Christophe MASUREL

UNE PROCÉDURE PASCAL

SAISIE D'UNE VARIABLE LOGIQUE

QUEL que soit le programme utilisé, il y a toujours, à un moment ou à un autre, un choix à indiquer, une réponse par oui ou par non à donner. Autant programmer une fois pour toutes la procédure qui se chargera d'afficher la question et de contrôler la réponse. Elle trouvera sans aucun doute sa place dans votre bibliothèque de sous-programmes.

Chaque fois qu'un programme pose une question appelant une réponse par oui ou non, et les occasions ne manquent pas (confirmation d'un ordre, pour ne citer que ce cas), on peut mémoriser la réponse dans une variable booléenne. Examinons une fonction adaptée à ce type de problèmes. Elle est déclarée de la façon suivante :

function d__accord (question : string ; defaut : boolean) : boolean ;

Le premier paramètre *(question)* correspond au texte de la question qui doit être posée ; le second *(defaut)*, à la réponse qui sera prise en considération si l'on ne désire pas y répondre. Le programme poursuivra dans ce cas avec la réponse par défaut. La réponse retournée par la fonction sera donc soit la réponse entrée au clavier par l'utilisateur, soit la réponse par défaut.

Dans la liste du sous-programme telle qu'elle apparaît page suivante, plusieurs réponses sont possibles. Le caractère O, majuscule ou minuscule, permet de répondre « oui ». Même remarque pour Y et y qui correspondent au « yes ». Ainsi, les réponses fournies par un utilisateur habitué aux programmes du système d'exploitation (dont les dialogues sont souvent en américain) seront acceptées sans problème. Cela présente, accessoirement, l'avantage de généraliser les programmes à la langue anglaise.

N et n seront utilisés pour les réponses négatives. Une pression sur la touche correspondant à cette lettre fait donc retourner à la fonction, dans des conditions que nous verrons plus loin, la valeur fausse. Les touches « Return » et « Escape », la barre d'espacement, ainsi que la lettre Q (majuscule et minuscule) permettent de ne pas répondre. Elles sont utilisées lorsque l'on désire retourner (Return) à un endroit du programme, s'échapper (Escape) d'une fonction ou quitter (Q) un traitement. Dans tous ces cas, c'est-à-dire quand l'utilisateur ne donne pas de réponse explicite, c'est la valeur par défaut, spécifiée par le programme, qui est prise en compte.

Quand on ne répond ni oui ni non

Comme nous le voyons, cette fonction, bien que simple en apparence, rend plus souple l'utilisation des programmes en prévoyant les réponses qui sont habituellement données, ou que, dans le doute, on a intérêt à donner.

La valeur par défaut, qui est le paramètre de cette fonction, ne doit pas être négligée. En effet, en permettant d'éviter la réponse, c'est cette valeur qu'améliore l'ergonomie d'un programme.

Il est difficile de savoir dans quels cas cette valeur devra être vraie et dans quels autres elle sera fausse. Deux argu-

Programme de saisie d'une variable logique

```
function d__accord (question : string ; defaut : boolean) : boolean ;
const    sonnette = 7 ;      (Code ASCII du « beep » sonore)
         retour   = 13 ;     (Code ASCII de la touche (return))
         sortie   = 27 ;     (Code ASCII de la touche (escape))
var      car      : char ;
         reponse  : boolean ;
         autorise : boolean ;
         oui, non,
         def      : set of char ;
begin
  oui : = ['O', 'o', 'Y', 'y'] ;
  non: = ['N', 'n'] ;
  def : = [chr (retour), ' ', 'Q', 'q', chr (sortie)] ;
  write (question) ;
  repeat
    read (keyboard, car) ;
    if eoln (keyboard) then car : = chr (retour) ;
    autorise : = car in oui + non + def ;
    if not autorise then write (chr (sonnette))
  until autorise ;
  reponse : = car in oui ;
  if (car in oui) = (car in non) then reponse : = defaut ;
  if reponse then write ('Oui') else write ('Non') ;
  writeln
  d__accord : = reponse
end ;
```

ments interviennent. Le premier concerne la réponse qui est habituellement donnée : la valeur par défaut doit dans ce cas correspondre à la réponse la plus courante. Il peut s'agir par exemple d'une saisie répétitive d'articles telle que l'illustre l'extrait de programme suivant :

```
repeat
  saisir__article(art) ;
  enregistrer(art) ;
until not d__accord('Désirez-vous continuer la saisie ?',true) ;
```

Le second argument porte sur la sécurité des données ; la valeur par défaut doit alors correspondre à la réponse qui, comme dans l'exemple ci-dessous, ne provoquera pas d'effacement ni de destruction.

```
if d__accord('On détruit cet article ?', false)
  then
    detruire__article(numero)
  else
    begin
      writeln;
      writeln('Destruction non effectuée.')
    end ;
```

CAR in OUI	CAR in NON	CAR in DEF	VALEUR RETOURNEE
0 (faux)	0 (faux)	0 (faux)	(refusé)
0 (faux)	0 (faux)	1 (vrai)	1 (vrai)
0 (faux)	1 (vrai)	0 (faux)	0 (faux)
0 (faux)	1 (vrai)	1 (vrai)	0 (faux)
1 (vrai)	0 (faux)	0 (faux)	1 (vrai)
1 (vrai)	0 (faux)	1 (vrai)	1 (vrai)
1 (vrai)	1 (vrai)	0 (faux)	0 (faux)
1 (vrai)	1 (vrai)	1 (vrai)	1 (vrai)

Le sous-programme de saisie d'une variable logique se compose de trois sous-ensembles. Initialisés dès l'activation de la fonction, ils permettent de spécifier les caractères qui peuvent être utilisés pour répondre oui, non ou la valeur par défaut. Les adaptations éventuelles se feront donc aisément. Il est par exemple possible d'élargir le champ des réponses « oui » et « non » en tolérant les fautes de frappe de l'utilisateur. Il suffira de prendre également en compte les touches avoisinant celles que l'on a réservées aux réponses.

La fonction prévoit les cas où un caractère entré par l'utilisateur se trouve dans plusieurs des ensembles OUI, NON et DEF. Cette situation pose un problème car on peut hésiter pour savoir quelle est la réponse à prendre en compte. D'une manière générale, cela se résout logiquement. La table de vérité ci-contre indique quelle est la valeur qui sera retournée suivant que le caractère entré fait partie ou non des ensembles OUI, NON et DEF. L'étude détaillée du programme montre que ce contrôle ne nécessite que deux lignes.

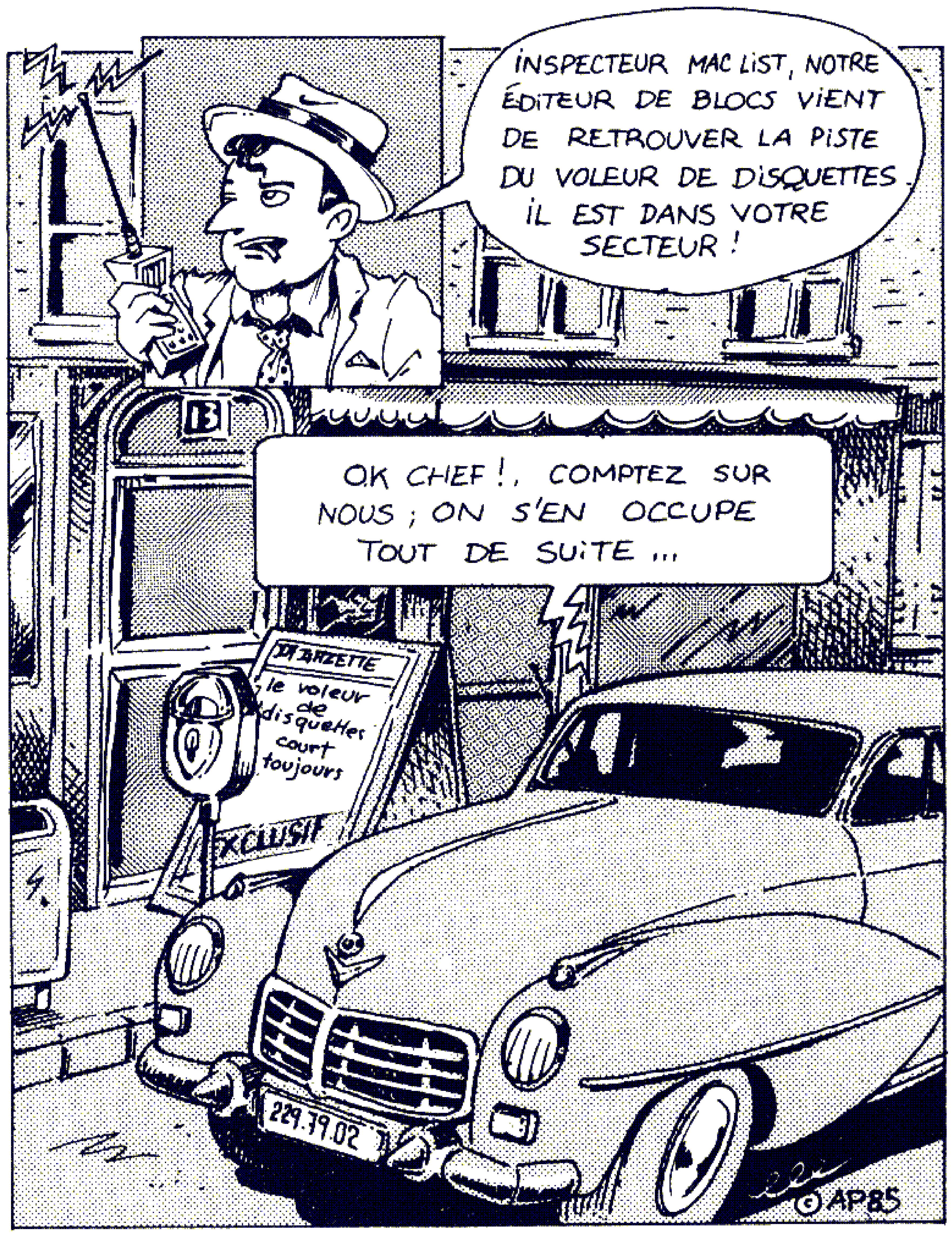
C'est ainsi qu'avec une simple fonction, les programmes deviennent à la fois plus simples, plus rapides à écrire et plus agréables à utiliser.

Thierry CHAMORET

LES DISQUETTES COMMODORE

LES BLOCS A L'AFFICHE

UNE bibliothèque d'utilitaires pour disquettes serait peu de chose sans un éditeur de blocs. Un tel instrument permet, après avoir recherché le contenu de n'importe quel secteur d'une piste, de l'afficher à l'écran, d'y apporter toutes les modifications souhaitées et de réécrire l'ensemble à l'endroit requis. C'est vraiment un outil fondamental.



Editeur de blocs
Programme pour Commodore 64
et lecteur de disquette
Auteur Jean-Pierre Lalevée
Copyright *LIST* et l'auteur

Dans le domaine des disquettes Commodore, poursuivons nos investigations, particulièrement bénéfiques aux utilisateurs de disquettes sur Vic, C.64 ou autres modèles plus anciens. La série d'utilitaires proposée jusque-là (voir *LIST* 3, 4, 6 et 7) est complétée aujourd'hui par un éditeur de blocs complet. C'est dire que l'étape est importante.

L'éditeur de blocs est un instrument à manier avec précautions : il ne faudra l'utiliser que sur les doubles des disquettes. Avec des disquettes originales, le risque est trop grand d'en détruire le

```
100 REM **************************
110 REM * UTILITAIRE DISQUE N.5 *
120 REM *      MODISECTEUR       *
130 REM **************************
140 :
150 REM EDITEUR: PERMET DE VISUALISER ET MODIFIER LES BLOCS SUR DISQUETTE
160 :
170 DEFFNA(N)=N-INT(N/16)*16+1
180 DEFFNB(N)=N/16+1
190 C0$=CHR$(.):X$="0123456789ABCDEF"
200 FORI=1TO39:EF$=EF$+" ":NEXT:EF$="[illegible]"+EF$+"]"
210 DIM TS(35),C1$(31),A$(31)
220 FOR I=1 TO 17:TS(I)=20:NEXT
230 FOR I=18 TO 24:TS(I)=18:NEXT
240 FOR I=25 TO 30:TS(I)=17:NEXT
250 FOR I=31 TO 35:TS(I)=16:NEXT
260 OPEN 80,8,15,"I0" '
270 :
280 GOSUB 530
290 CLOSE 82:PRINT#80,"I":CLOSE80
300 END
310 :
320 REM ************************ ERREUR DISQUE ********************************
330 INPUT#80,E1,E2$,E3,E4:IF E1<21 THEN 360
340 PRINT"ERREUR DISQUE :":PRINT E1","E2$","E3","E4:CLOSE82:CLOSE80
350 FOR I=0 TO 1000:NEXT
360 RETURN
370 :
380 REM *********************** CONV.ASCII>HEXA *******************************
390 N=ASC(N$+C0$):Z$=MID$(X$,FNB(N),1)+MID$(X$,FNA(N),1):RETURN
400 :
410 REM ********************** CONV.HEXA>DECIMAL ******************************
```

contenu, en partie ou entièrement. Un programmeur averti en vaut deux !

Chacun des blocs inscrits sur une disquette occupe très exactement un secteur de celle-ci, soit 256 octets. Il est facile de positionner la tête de lecture sur la piste et le secteur choisis afin de récupérer tous les octets qui s'y trouvent. Encore faut-il pouvoir lire ces octets. Pour être aussi parlant que possible, l'éditeur devra fournir d'une part les octets sous forme hexadécimale, et d'autre part leur traduction sous forme de caractères ASCII. Il devra aussi les disposer à l'écran de telle sorte que les modifications soient possibles.

La plupart des machines disposent de 40 colonnes. C'est peu, et en tout cas insuffisant pour tout afficher à la fois. Il faudra donc partager les 256 octets en deux pages de 128 octets, et se donner le moyen de commuter les pages à volonté sur l'écran.

Une solution d'une grande simplicité

Il va de soi qu'avec un Vic, la version de l'éditeur devra subir des modifications avec un découpage du bloc en quatre pages commutables. Mais avec 80 colonnes, aucun problème.

Pour modifier à loisir le contenu des pages, une solution idéale existe. Elle consiste à écrire, en Basic ou en langage-machine, une routine simulant un éditeur plein écran qui permet de se promener sur la page affichée et d'y effectuer toutes sortes de modifications. Mais cette solution est vraiment difficile à mettre en place.

Nous avons choisi, ici, une solution d'une grande simplicité : un programme en Basic qui emploie une série d'instructions INPUT. Ceci oblige à faire très attention à la frappe au clavier, aucun accès direct à l'octet choisi n'étant possible.

D'autre part, il faut veiller à ne pas introduire de virgules ou de deux-points, faute de quoi ils seraient refusés avec l'affichage d'un message d'erreur détruisant ce qui est à l'écran (mais sans conséquence pour la disquette et son contenu). Si une telle erreur était commise, le mieux serait alors de presser sur la touche STOP (ou STOP/RESTORE), et de relancer le programme avec RUN. Quelle perte de temps !

Les valeurs doivent être tapées en hexadécimal. Mais lorsqu'il s'agit de caractères alphabétiques ou numéri-

```
420 F=LEN(N$):N=.
430 FORJ=1TOF:G=ASC(MID$(N$,J,1))-48:IFG>9THENG=G-7
440 N=16*N+G:NEXT:RETURN
450 :
460 REM ********** CREATION DES CHAINES **********
470 FORI=.TO255STEP8:K=I/8:A$="":C$=""
480 FORJ=1TO8:GET#82,N$:GOSUB390
490 C$=C$+Z$+" ":IFN$<"@"ORN$>"←"THENN$="."
500 A$=A$+N$:NEXT:C1$(K)=C$:A$(K)=A$:NEXT
510 RETURN
520 :
530 REM ********** AFFICHAGE SECTEURS **********
540 PRINT"■","  ■ MODISECTEUR "
550 INPUT"■■ PISTE                    ■■■■■■■■■■■■■■■■■■■■■■";NP
560 IF NP<1 OR NP>35 THEN 540
570 PRINT"■"TAB(13);:INPUT"/ SECTEUR      ■■■■■";NS
580 IF NS<. OR NS>TS(NP)THEN 570
590 :
600 REM ********** LECTURE SECTEUR **********
610 OPEN 82,8,2,"#":GOSUB 330
620 FORK=.TO15:C1$(K)="":NEXT
630 :
640 PRINT#80,"U1:"2;0;NP;NS
650 GOSUB 330:IF E1 THEN 540
660 PRINT#80,"B-P"2;0:GET#82,AB$:GET#82,AH$
670 PRINT#80,"B-P"2;0
680 GOSUB 330
690 :
700 REM ********** AFFICHAGE ECRAN **********
710 PRINT"■","  LECTURE DE BLOC ■    "
720 PRINT"■      * PISTE ■"NP"■■ ■ / SECTEUR ■"NS"■■ ■ *"
730 PRINT"■■■■■■■","  UN INSTANT SVP ":GOSUB 460
740 CLOSE 82:E=0:GOTO 790
750 GET R$:IF R$=" "THEN 880
760 IF R$="F"THEN 1350
770 IF R$<>"←"THEN 750
780 E=E+16:IF E>16 THEN E=0
790 PRINT"■■■■"
800 FOR I=E TO E+15:K=I*8
810 PRINT"■";:IF K<100 THEN PRINT" ";:IF K<10 THEN PRINT" ";
820 PRINTK;"■■ ■ ";C1$(I);A$(I)
830 NEXT I
840 PRINTEF$:PRINT"  ■SPC■ MODIF/ECRIT * ■ ← ■ PAGE * ■ F ■ FIN"
850 GOTO 750
860 :
870 REM ********** MODIFICATION **********
880 PRINT EF$:PRINT"  ■ M ■ODIFIER * ■ ← ■PAGE * ■ D ■ISQUETTE"
890 GETR$:IFR$="D"THEN1170
900 IFR$="←"THEN840
910 IFR$<>"M"THEN890
920 :
930 PRINT"■","■MODIFICATION DE BLOC"
940 PRINT"■■■■":FOR I=E TO E+15
950 INPUT"■■■■";C1$(I)
960 NEXT I
970 :
980 PRINTEF$:PRINTTAB(8)"■PATIENTEZ UN INSTANT SVP"
990 :
1000 REM ********** TRAITE CHAINES ENTREES **********
1010 PRINT"■■■■"
1020 FORI=ETOE+15:C$="":A$="":PRINTTAB(4);"■";
1030 FORL=1TO24STEP3:P$=MID$(C1$(I),L,2)
1040 IFLEFT$(P$,1)="."THENN$=RIGHT$(P$,1):GOSUB390:P$=Z$:GOTO 1060
1050 N$=P$:GOSUB 420:N$=CHR$(N)
1060 C$=C$+P$+" "
1070 IFN$<"@"ORN$>"←"THENN$="."
1080 A$=A$+N$:NEXT:C1$(I)=C$:A$(I)=A$:NEXT
1090 :
1100 REM ********** EST-CE OK ??? **********
1110 PRINTEF$:INPUT"> TOUT EST OK (O/N) ";R$:R$=LEFT$(R$,1)
1120 IF R$="O"THEN 790:REM OK
1130 IF R$="N"THEN 930:REM A REFAIRE
1140 GOTO 1110:REM ERREUR
1150 :
1160 REM ********** REECRITURE DU SECTEUR **********
1170 PRINT"■","■ RECOPIE SUR DISQUE ■ "
1180 PRINTEF$:PRINT"> PUIS-JE RECOPIER (O/N) ?"
1190 GETR$:IFR$="N"THEN840
1200 IFR$<>"O"THEN1190
1210 PRINTEF$:PRINT"      ■ OK... RECOPIE EN COURS "
1220 OPEN82,8,2,"#":GOSUB 330
1230 PRINT#80,"B-P"2;0:GOSUB330:IFE1THEN1160
1240 PRINT#80,"B-P"2;0
1250 FORI=.TO31:K=I*8:IF(IOR16)=16THENPRINT"■■■■"
1260 PRINT"■";:IF K<100 THEN PRINT" ";:IF K<10 THENPRINT" ";
1270 PRINT K;"■■:■■";C1$(I);A$(I)
1280 FORL=1TO24STEP3:N$=MID$(C1$(I),L,2):GOSUB420:PRINT#82,CHR$(N);:NEXT:NEXT
1290 GOSUB 330:IF E1 THEN 1160
1300 :
1310 PRINT#80,"U2";2;0;NP;NS:GOSUB 330:IF E1 THEN STOP
1320 CLOSE82
1330 :
1340 REM ********** AUTRE SECTEUR ? **********
1350 PS=ASC(AB$+C0$):SS=ASC(AH$+C0$)
1360 IF PS<1 OR PS>35 THEN NS=NS+1:GOTO 1380
1370 NP=PS:NS=SS
```

Image d'un bloc en cours de traitement.

```
          MODIFICATION DE BLOC

     * PISTE  20  / SECTEUR  2  *
  0> 14 0C 20 05 13 14 20 05 ........
  8> 04 09 14 05 20 01 20 54 .......T
 16> 0F 15 0C 0F 15 13 05 20 ........
 24> 10 01 12 20 0C 01 20 13 ........
 32> 0F 03 09 05 14 05 20 22 ........
 40> 4C 41 4E 47 41 47 45 20 LANGAGE.
 48> 45 54 20 49 4E 46 4F 52 ET.INFOR
 56> 4D 41 54 49 .O .N .. 22 MATIQUE.
 64> 2E 1F 20 20 49 0C 20 13 ....I...
 72> 05 20 10 12 05 13 05 0E ........
 80> 14 05 20 13 0F 15 13 20 ........
 88> 0C 01 20 06 0F 12 0D 05 ........
 96? 20 04 27 15 0E 05 20 05 ........
104? 13 13 09 13 13 05 20 10 ........
112? 0F 03 08 05 14 14 05 20 ........
120? 10 0C 01 13 14 09 06 09 ........

     PATIENTEZ UN INSTANT SVP
```

Trois caractères ASCII au lieu de trois nombres hexadécimaux, pour remplacer le mot INFORMATIQUE par le mot INFORMATION. Ils sont précédés par un point.

Champ ASCII

Champ HEXA

Numéros des premiers octets de chaque ligne (ici, page 1 du bloc).

```
1380 IF NS>TS(NP) THEN NS=0:NP=NP+1
1390 IF NP>35 THEN PRINT"■FIN DU DISQUE ATTEINTE ":GOTO 1460
1400 :
1410 PRINTEF$:PRINT"SUITE: PISTE■";NP"■■ ■/SECT■";NS;"■■ ■ >OK ■■O■■■■";
1420 INPUTR$:R$=LEFT$(R$,1)
1430 IFR$="O"THEN 600
1440 IFR$<>"N"THEN 1410
1450 :
1460 PRINTEF$:INPUT"UN AUTRE SECTEUR (O/N) ■■O■■■■";R$
1470 IF LEFT$(R$,1)="O"THEN 540
1480 RETURN
1490 :
1500 END
READY.
```

Explications du programme

Lignes 170 à 260 : initialisations diverses avant déroulement du programme principal. Les lignes 220 à 250 peuvent exiger des modifications avec les disquettes du type 3000 ou 8000 sur lesquelles le nombre de pistes/secteurs n'est pas exactement le même.

Lignes 320 à 360 : routine classique de recherche des erreurs disque.

Lignes 380 à 440 : deux routines de conversion fréquemment employées par le programme. Les sous-programmes sont placés en tête de programme, ce qui permet de gagner quelques centièmes de seconde lors de chaque appel !

Lignes 460 à 510 : dernière routine, celle qui transforme les 256 octets en 32 chaînes placées en tableau.

Lignes 530 à 580 : début du programme principal, introduction des numéros de piste et de secteur à analyser.

Ligne 610 : réservation d'un buffer pour le transfert des données en provenance de la disquette.

Ligne 640 : positionnement de la tête de lecture à l'endroit choisi.

Lignes 660 et 670 : les deux premiers octets qui représentent les numéros de piste et de secteur suivants sont extraits. Cela permettra d'indiquer à l'utilisateur où se trouve la suite du bloc qu'il a choisi de traiter.

Lignes 780 à 850 : l'affichage de la première page se fait sur l'écran. L'appui sur "←" permet le changement de page, tandis que la barre d'espace est utilisée pour sélectionner le mode "MODIFICATION". Enfin, "F" fait sortir du mode affichage, pour passer à la suite...

Lignes 870 à 960 : la routine de modification offre trois options, la modification proprement dite en mémoire centrale, le retour au mode affichage et la réécriture du bloc sur la disquette. La ligne 950 prend les chaînes sur l'écran après modification(s).

Lignes 1000 à 1080 : ici, les chaînes entrées sont converties en caractères qui seront affichés avec elles à droite de l'écran.

Lignes 1100 à 1140 : en cas d'erreur, le retour à la case départ est permis.

Lignes 1160 à 1320 : la réécriture du bloc sur la disquette peut avoir lieu ; elle se déroule avec affichage simultané des chaînes à l'écran, page après page. La ligne 1230 repositionne le pointeur de bloc (dans le buffer réservé) sur le premier octet. La ligne 1310 permet l'écriture sur la disquette, via le buffer, en direction de la piste et du secteur en cours. (U2 est l'équivalent de B-W, avec la même syntaxe d'utilisation.)

Lignes 1340 à 1480 : l'utilisateur peut poursuivre ses investigations sur le bloc suivant, qui est soit celui dont les numéros de piste et de secteur ont été mémorisés, soit celui qui lui succède immédiatement sur le support physique. Ce sera le cas si les numéros de piste ou de secteur sont illégaux. Le programme s'arrête si le choix est négatif.

ques, on peut éviter des recherches fastidieuses de codes hexadécimaux en introduisant le caractère désiré dans les lignes d'INPUT. La première contrainte est alors d'inscrire un point devant le caractère ; la seconde, de ne pas taper de virgules, deux-points ou guillemets (pour la raison énoncée plus haut). La figure ci-contre *(Image d'un bloc en cours de traitement)* aide à mieux comprendre ce processus.

Attention à ne pas tout perdre

Dans tous les cas, il ne faut pas oublier de frapper les espaces qui servent de séparateurs entre les codes.

Lorsque la série d'INPUT est terminée, le programme demande une confirmation. Si une erreur subsiste, il est encore possible de remonter en haut de la page et de corriger. L'appel de la page suivante s'effectue par l'appui sur la flèche gauche (←), et le processus se poursuit selon le même principe.

Dès que la page a été entièrement traitée, le programme effectue une analyse des lignes et affiche la nouvelle traduction ASCII, ce qui permet une vérification plus efficace. La modification n'est cependant effective que dans la mémoire centrale. Sur la disquette, la modification intervient seulement si le choix en est fait, auquel cas l'écriture est irréversible, et le contenu antérieur du secteur traité irrémédiablement perdu. Heureusement, vous avez pris soin de conserver à l'abri un double de votre disquette ! Il ne faut pas oublier, en outre, d'enlever l'étiquette de protection en écriture sur la disquette de travail.

Entièrement en Basic, ce programme est un peu lent. Bien évidemment, il aurait été plus rapide s'il avait été écrit en langage-machine. Notre choix a été avant tout de rester simple et compréhensible. Mais il vous est toujours possible d'ajouter une routine permettant l'écriture des blocs sur une imprimante, en vue de conserver une trace du travail effectué et du contenu des disquettes.

Jean-Pierre LALEVÉE

PASCAL EST CONTAGIEUX : BASIC SE STRUCTURE (II)

DANS le précédent numéro de LIST, nous avions évoqué les avantages du Basic structuré. Nous avions insisté sur l'utilité des fonctions et des procédures. Ce mois-ci, nous nous penchons sur les structures itératives et sur les tests.

Dans pratiquement tout programme se trouvent des instructions devant être exécutées deux fois ou plus. Le plus souvent, on dispose de telles instructions dans une boucle dont le nombre de répétitions est contrôlé par une ou plusieurs variables nommées indices de boucle. La boucle ne sera parcourue, c'est-à-dire les instructions qui la composent ne seront exécutées que si les indices vérifient certaines conditions programmées sous forme de tests donnant le résultat *vrai* ou *faux*.

On peut démontrer que **toute** boucle peut se programmer à l'aide des deux structures suivantes, et d'elles seules :
- FAIRE instructions JUSQU'A condition VRAIE (voir organigramme 1) ;
- TANT QUE CONDITION VRAIE, FAIRE instructions (voir organigramme 2).

En Basic, ces deux types de boucles s'écrivent respectivement : REPEAT... instructions UNTIL condition, et WHILE condition DO... instructions... WEND (WEND servant simplement à indiquer la fin des instructions à exécuter).

A ce stade de l'explication, on peut se demander ce qu'il en est de la boucle FOR...NEXT. L'a-t-on oubliée ? En fait, cette boucle est inutile, puisqu'elle n'est qu'un cas particulier de REPEAT...UNTIL. Pour s'en convaincre, on examinera les deux programmes suivants qui produisent les mêmes résultats. En dessinant l'organigramme de l'un et de l'autre, il sera facile de se convaincre qu'ils sont identiques du point de vue algorithmique.

```
10 FOR I=0 TO 10 STEP .5
20 : PRINT I
30 NEXT I
```

et

```
10 I=0
20 REPEAT
30 : PRINT I : I=I+.5
40 UNTIL I>10
```

A titre d'illustration des différents types de boucles, on se reportera au programme calculant les valeurs exactes des coefficients du binôme pour n et p inférieurs à 9999999 (liste n° 1).

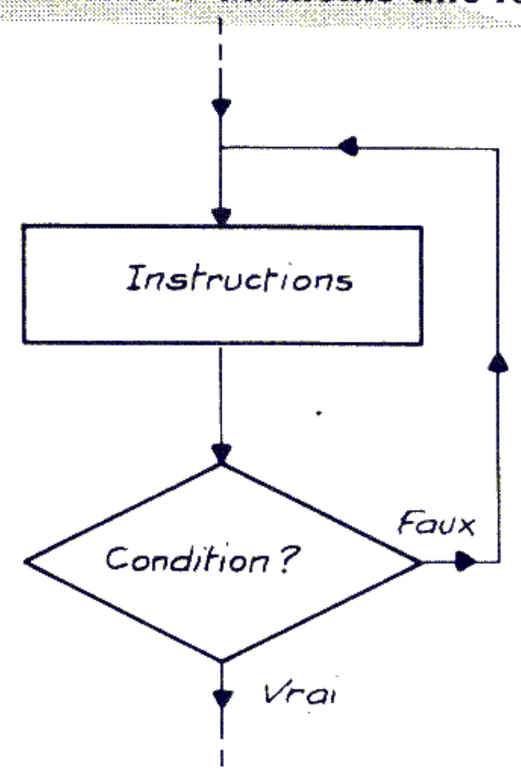
Pour vos boucles, attention aux nœuds

Ce programme peut être adapté à tout type de Basic à condition de remplacer les appels de procédures par des GOSUB, les ENDPROC par RETURN et de programmer directement les boucles REPEAT et WHILE par des tests et des GOTO. Si le lecteur a bien compris le fonctionnement des deux organigrammes présentés plus haut, cela ne devrait lui poser aucun problème.

On remarquera que, pour P=0, les boucles WHILE-WEND des lignes 290 et 420 ne sont pas parcourues. Étudions plus particulièrement la seconde d'entre elles. Le Basic, explorant le programme, arrive ligne 420 avec P=0 et J=2, trouve une condition fausse et doit donc chercher le WEND correspondant, qui se trouve à la ligne 520. Il devra donc explorer les lignes suivant la ligne 420, sans se laisser tromper par le WEND de 500, associé au WHILE de 480. Il me semble que c'est ce problème de recherche, *a priori* compliqué, qui

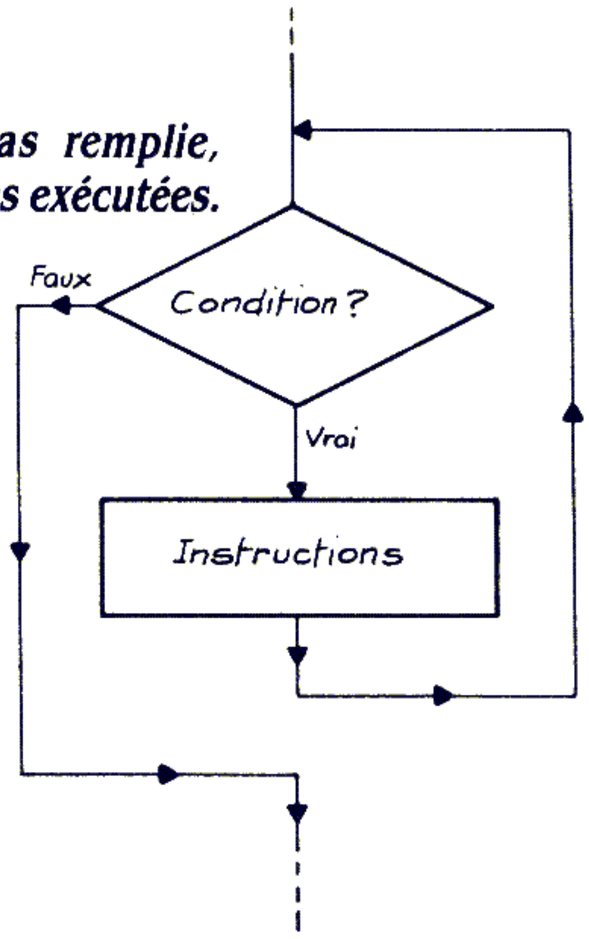
Organigramme 1 :

REPEAT...UNTIL
Dans tous les cas, les instructions seront exécutées au moins une fois.



Boucle désormais inutile à nourrir programmée pour s'en aller mourir loin de sa tribu

Organigramme 2 :
WHILE...DO...WEND
Si la condition n'est pas remplie, les instructions ne sont pas exécutées.



Liste 1

Calcul des coefficients d'un binôme
Programme en Basic structuré (MZ-700)
Auteur Bernard Kokanosky


```
10 PROC"Init"
20 REPEAT
30 : PROC"n,p=?"
40 : PROC"Num=n(n-1)...(n-p+1)"
50 : PROC"C=Num/p!"
60 : PROC"Resultat"
70 : PRINT:PRINT"Termine !":PRINT
80 UNTIL 0
90 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
100 DEF PROC"Init"
110 DIM A(255):H=1000000:CLS
120 PRINT"CALCUL DES C(n,p)"
130 PRINT"------------------":PRINT
140 ENDPROC
150 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
160 DEF PROC"n,p=?"
170 REPEAT
180 : INPUT"n=";N
190 UNTIL N>0 AND N=INT(N)
200 REPEAT
210 : INPUT"p=";P
220 UNTIL P>=0 AND P=INT(P) AND P<=N
230 PRINT "C(";N;",";P;")=";
240 IF P>N/2 LET P=N-P
250 ENDPROC
260 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
270 DEF PROC"Num=n(n-1)...(n-p+1)"
280 M=0:A(0)=1:J=N-P+1
290 WHILE J<=N DO
300 : R=0
310 : FOR K=0 TO M
320 :   A=A(K)*J+R
330 :   R=A DIV H:A(K)=A MODULO H
340 : NEXT K
350 : IF R>0 LET M=M+1:A(M)=R
360 J=J+1
370 WEND
380 ENDPROC
390 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
400 DEF PROC"C=Num/p!"
410 J=2
420 WHILE J<=P DO
430 : R=0
440 : FOR K=M TO 0 STEP -1
450 :   A(K)=A(K)+R*H:R=A(K) MODULO J
460 :   A(K)=A(K) DIV J
470 : NEXT K
480 : WHILE A(M)=0 DO
490 :   M=M-1
500 : WEND
510 : J=J+1
520 WEND
530 ENDPROC
540 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
550 DEF PROC"Resultat"
560 A$=STR$(A(M)):L=LEN(A$)
570 IF L>3 A$=IN$(A$,L-3,L-2," ")
580 PRINT A$;
590 IF M>0 PROC"Suite"
600 ENDPROC
610 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
620 DEF PROC"Suite"
630 FOR J=M-1 TO 0 STEP -1
640 : A$=STR$(A(J)):L=LEN(A$)
650 : A$=" "+STRING$(6-L,"0")+A$
660 : PRINT IN$(A$,4,5," ");
670 NEXT J
680 ENDPROC
```

Remarque : **IN$ (A$,n,p,B$) est la chaîne obtenue en remplaçant dans A$ les caractères de rang n + 1, n + 2,...,p − 1 par B$. En Basic standard, il suffit de remplacer IN$ par LEFT$ (A$,n) + B$ + RIGHT$ (A$,LEN(A$) − p + 1).**

PASCAL EST CONTAGIEUX : LE BASIC SE STRUCTURE (II)

Etat 1

	zone PROC 20

Etat 2

Etat 3

		zone REPEAT 30

Etat 4

	zone PROC 40	zone REPEAT 30

Etat 5

zone REPEAT 180	zone PROC 40	zone REPEAT 30

Etat 6

	zone PROC 40	zone REPEAT 30

Etat 7

zone FOR 450	zone WHILE 420	zone PROC 60	zone REPEAT 30

Les états successifs de la pile Basic

explique l'absence quasi générale de la structure WHILE-WEND dans les différentes versions du Basic standard.

Ce genre de problème ne peut pas se poser avec FOR-NEXT ou REPEAT-UNTIL puisque le test se faisant en fin de boucle, sa position est connue. On peut cependant se demander comment le Basic procède pour retrouver le début de la boucle. Pour donner une idée simplifiée du fonctionnement interne d'un Basic structuré, je prendrai l'exemple de celui que j'ai écrit pour le Sharp MZ-700.

Voici une carte simplifiée de la mémoire de l'ordinateur lors du fonctionnement d'un programme :

Basic	Programme de l'utilisateur	Noms des variables	Valeurs des variables		Pile Basic	Pile du micro-processeur
Les deux limites tracées en pointillé (valeurs des variables, et pile Basic) ne sont pas fixes : elles se déplacent durant l'exécution du programme.						

Le fonctionnement interne du Basic structuré est presque exclusivement basé sur la pile Basic que je nommerai pile tout simplement. Le Basic y stocke des résultats intermédiaires lors de ses calculs (nous ne nous intéresserons pas à cet aspect) et surtout des « zones FOR », des « zones REPEAT », des « zones WHILE », des « zones PROC ou FN ». Nous allons suivre le Basic lors de l'exécution du programme.

• Ligne 10 : appel de la procédure « Init ». Le Basic cherche donc cette procédure et la trouve en ligne 100. Il stocke dans la pile un pointeur indiquant l'endroit où il devra revenir. Pour simplifier, disons qu'il stocke 20, c'est-à-dire le numéro de la ligne suivante. On saute alors à la ligne 100, et la pile Basic est dans l'état n° 1 (voir ci-contre).

• Ligne 100 à 140 : exécution de la procédure. L'instruction ENDPROC est rencontrée à la ligne 140. Le Basic consulte alors la pile, trouve bien une zone PROC (en son absence, on aurait eu droit à une erreur), reprend le pointeur 20, élimine la zone PROC et retourne en ligne 20. La pile est alors vide (état 2).

• Ligne 20 : en rencontrant REPEAT, le pointeur stocke dans la pile le pointeur situé derrière REPEAT (ici, la ligne 30) et il passe à cette ligne 30. La pile est dans l'état 3.

• Ligne 30 : saut à la ligne 160 avec une pile dans l'état 4.

• Ligne 170 : REPEAT de nouveau, et donc passage à la ligne 180 avec une pile dans l'état 5. On entre N = 5, par exemple, en réponse à l'INPUT.

• Ligne 190 : la condition du test étant vérifiée, la zone REPEAT est éliminée et le Basic passe à la ligne 200 avec une pile dans l'état 6. Si l'on avait entré – 5, la condition n'aurait pas été satisfaite et le Basic, reprenant le bas de la pile, y aurait trouvé une zone REPEAT lui permettant de revenir à la ligne 180.

Chacun est invité à continuer un peu cette étude et à vérifier, par exemple, que lors de l'exécution de la ligne 450 (procédure appelée par la ligne 50), la pile est dans l'état 7. On remarquera que, pour la zone WHILE, c'est 420 qui doit être stocké puisque le Basic doit réévaluer la condition à chaque tour de boucle. De plus, dans une zone FOR, il faut stocker aussi le nom de la variable (indice de la boucle), la valeur finale et le pas d'incrément de la boucle.

Il faut soigner ses sorties

Ce mode de fonctionnement explique pourquoi, en Basic structuré, toute sortie de boucle par un GOTO et toute sortie d'une procédure autrement que par ENDPROC sont interdites : les sorties de ce type laissent en bas de la pile une zone parasite et le programme s'arrête tôt ou tard sur un message d'erreur.

Un Basic standard admet généralement sans broncher une sortie de ce genre car son fonctionnement ne repose pas sur une seule pile, mais sur plusieurs piles distinctes contenant respectivement la zone FOR, la zone GOSUB et, si elles existent, les zones REPEAT et WHILE. Le programmeur prend cependant le risque de voir son programme se comporter de façon bizarre sans qu'un quelconque message d'erreur ne puisse l'alerter.

Étudions maintenant les tests qui permettent d'orienter le déroulement du programme selon les valeurs prises par certaines variables. Même dans les Basic structurés, on trouve rarement autre chose qu'IF...THEN...ELSE dont le grand défaut est que toutes les instructions doivent être écrites sur la même ligne, ce qui conduit souvent à des programmes illisibles, un comble pour un langage structuré !

Parfois, il existera une structure plus complexe : IF...ELSIF...ELSIF... ELSE...ENDIF pouvant s'étendre sur un nombre quelconque de lignes et bien délimitée par IF (début) et ENDIF (fin). En pratique, ELSE peut être absent, comme d'ailleurs ELSIF qui peut aussi apparaître un nombre quelconque de fois. Bien entendu, les programmes sont alors assez gourmands en mémoire, mais il y est beaucoup plus facile d'y

fichier séquentiel sur cassette), apparaît un laid "PRESS PLAY ON TAPE", qu'on le veuille ou non, à moins que, sentant venir le vent, l'utilisateur ne se soit précipité, index frémissant, pour enfoncer la touche lecture avant qu'on ne le lui demande.

L'astuce va donc consister à demander à l'utilisateur *avant* l'OPEN d'appuyer sur la touche ; et pendant qu'on y est, à le lui demander *en français* :

```
5000 PRINT "APPUYEZ SUR LA TOUCHE PLAY DU MAGNETOPHONE"
5010 WAIT 1, 16, 16
5020 PRINT "D'ACCORD"
5030 OPEN ...
```

et la vieille dame du Quai Conti (Non ! pas Yourcenar, les quarante !) d'applaudir bien fort.

Etalonner le programme

Une autre utilisation de cette astuce consiste, par exemple dans un programme qui gère un seul fichier sur cassette, à mettre le fichier sur la même cassette que le programme, à distance respectueuse. Les choses se passent ainsi : le programme principal, dans la partie consacrée à l'écriture du fichier, demande de rembobiner la cassette jusqu'à son début, puis d'appuyer sur STOP. Il demande ensuite d'appuyer sur la touche « marche avant rapide ». Dès que cette touche est enfoncée, il lance un chrono et, lorsque ce chrono atteint le temps nécessaire pour que le programme principal soit dépassé sur la cassette, il coupe le moteur et demande d'appuyer sur la touche STOP du magnétophone avant de procéder à l'écriture.

```
49999 REM *** POSITIONNEMENT DE LA CASSETTE ***
50000 PRINT "METTEZ LA CASSETTE A SON DEBUT"
50010 PRINT "PUIS FRAPPEZ RETURN."
50020 GET R$ : IF R$ < > CHR$(13) THEN 50020
50030 PRINT "ENFONCEZ LA TOUCHE 'AVANCE RAPIDE'"
50040 PRINT "DU MAGNETOPHONE."
50050 WAIT 1, 16, 16
50060 PRINT "D'ACCORD"
50070 TI$ = "000000"
50080 IF TI < XX THEN 50080
50090 POKE 192, 7 : POKE 1, PEEK (1) AND 223 OR 32
50100 PRINT "ENFONCEZ LA TOUCHE 'STOP'"
50110 PRINT "DU MAGNETOPHONE.""
50120 WAIT 1, 16
50130 PRINT "D'ACCORD"
50140 REM *** ECRITURE DU FICHIER ***
50150 PRINT "ENFONCEZ LES TOUCHES ENREGISTREMENT"
50160 PRINT "ET LECTURE DU MAGNETOPHONE."
50170 WAIT 1, 16, 16
50180 PRINT "D'ACCORD"
50190 OPEN 1, 1, 1, "ZOZO"
etc.
```

WAIT : attendez qu'on s'explique !

WAIT n'est pas une instruction présente sur toutes les machines, et si elle est présente dans le Basic de Commodore, elle n'est pas souvent utilisée. Il est vrai que c'est une instruction booléenne en diaaaable, quoâ !

Elle fonctionne selon la syntaxe WAIT A,B,C où A représente une adresse et B et C deux nombres compris entre 0 et 255, C étant facultatif et valant 0 par défaut. En fait, l'ordinateur comprend : IF (PEEK(A) XOR C) AND B est différent de 0 THEN je continue, sinon, je boucle et j'attends obstinément qu'il cesse d'être nul comme ça.

XOR, c'est l'opération de OU exclusif ; AND, c'est le ET logique.

A	B	A XOR B	A AND B
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Sur le Commodore 64, le bit 4 de l'octet 1 est consacré à la détection d'une touche enfoncée sur le magnétophone, et le bit 5 au contrôle du moteur.
Si on veut attendre que toutes les touches du magnétophone soient relevées, il faut que le bit 4 de l'octet contenu dans l'adresse 1 soit à 1.

Essayez d'enfoncer la touche PLAY de votre lecteur de cassette et de faire PRINT PEEK(1) : vous obtenez 7 en temps normal. Si PEEK(1) vaut 7, comme la représentation de 7 sur un octet est 00000111 et celle de 16 est 00010000, alors 7 AND 16 devient sur un octet :

```
7                00000111
16               00010000
7 AND 16         00000000
```

Le résultat de 7 AND 16 est nul : WAIT 1, 16 fait boucler le programme.
Appuyez sur la touche STOP et refaites PRINT PEEK(1) : vous obtenez 55.
Si PEEK(1) vaut 55, on a sur un octet :

```
55               00110111
16               00010000
55 AND 16        00010000
```

Le résultat de 55 AND 16 est non-nul, le programme peut continuer.
Si, en l'état actuel des choses, on tombe sur un WAIT 1, 16, 16 on a :

```
55               00110111
16               00010000
55 XOR 16        00100111
(55 XOR 16) AND 16   00000000
```

Le résultat est nul et le programme boucle. Mais si on enfonce une touche, les bits 4 et 5 passent à zéro (l'octet étant formé de huit bits numérotés : 76543210) et l'on a :

```
7                00000111
16               00010000
7 XOR 16         00010111
(7 XOR 16) AND 16    00010000
```

Le résultat est non-nul et le programme peut continuer.

L'important dans l'histoire est donc d'étalonner le programme ainsi modifié pour fixer la valeur représentée par XX à la ligne 50080. Cette valeur est comparée à celle de TI. Rappelons que chez Commodore, deux variables réservées sont consacrées à l'horloge interne. L'une, TI, représente l'heure exprimée en *jiffies* ou 1/60e de seconde. On peut la lire, mais pas lui affecter de valeur : essayez de faire en mode direct TI=0 ou TI=12, vous aurez du SYNTAX ERROR autant que vous voudrez. Pour remettre TI à zéro, il faut passer par la

MISEZ P'TIT OPTIMISEZ

FONCTIONS HYPERBOLIQUES

SI jongler avec la pile opérationnelle de votre HP-41C, 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 !

Fonctions hyperboliques
Programme pour HP-41C
Auteur Alain Goubault de Brugière
Copyright *LIST*et l'auteur

```
01♦LBL "CH"
02 CF 00
03 GTO 00
04♦LBL "SH"
05 SF 00
06♦LBL 00
07 XEQ 03
08 E↑X
09 1/X
10 FC?C 00
11 CHS
12 ST- L
13 RDN
14 2
15 ST/ L
16 GTO 01
17♦LBL "TH"
18 XEQ "GD"
19 SIN
20 X<> L
21 GTO 01
22♦LBL "SE"
23 XEQ "GD"
24 COS
25 X<> L
26 GTO 01
27♦LBL "GD"
28 XEQ 03
29 E↑X
30 ATAN
31 STO L
32 CLX
33 4
34 XEQ 02
35 ST- L
36 CLX
37 2
38 ST/ L
39 XEQ 01
40 RTN
41♦LBL "ASH"
42 XEQ 03
43 SF 00
44 ATAN
45 GTO "AGD"
46♦LBL "ATH"
47 XEQ 03
48 SF 00
49 ASIN
50 GTO "AGD"
51♦LBL "ACH"
52 XEQ 03
53 SF 00
54 1/X
55♦LBL "ASE"
56 FC? 00
57 XEQ 03
```

Cosinus, sinus, tangente et fonctions inverses, c'est bien. Modes degrés, radians ou grades, ne sont pas mal non plus... Mais les fonctions hyperboliques, c'est mieux ! Pourquoi ? Parce que la HP-41C ne les possède pas d'origine, bien sûr, et nous offre là le prétexte d'une optimisation.

Mon programme offre cinq fonctions hyperboliques (sinus, cosinus, tangente, amplitude et sécante) et leurs inverses — rien que de très classique — mais n'utilise aucun registre de mémoire. Seule la pile opérationnelle est employée *et totalement sauvegardée*. Les anciens Y, Z et T demeurent et la valeur initiale de l'argument (ex-X) peut être récupérée dans Lastx : tout se passe comme si les fonctions nouvelles étaient microprogrammées, au temps de calcul près. Une fois assignées à des touches du clavier, elles donneront une parfaite illusion.

Coût total du programme : 215 octets, Labels compris (NDLR : c'est un record de... longueur pour *Misez p'tit* !). Mode d'emploi : aucun. Chaque fonction hyperbolique s'appelle de son nom d'un XEQ, son argument étant en X où se trouvera aussi le résultat au retour.

Alain GOUBAULT de BRUGIÈRE

QUI DIT MIEUX ?

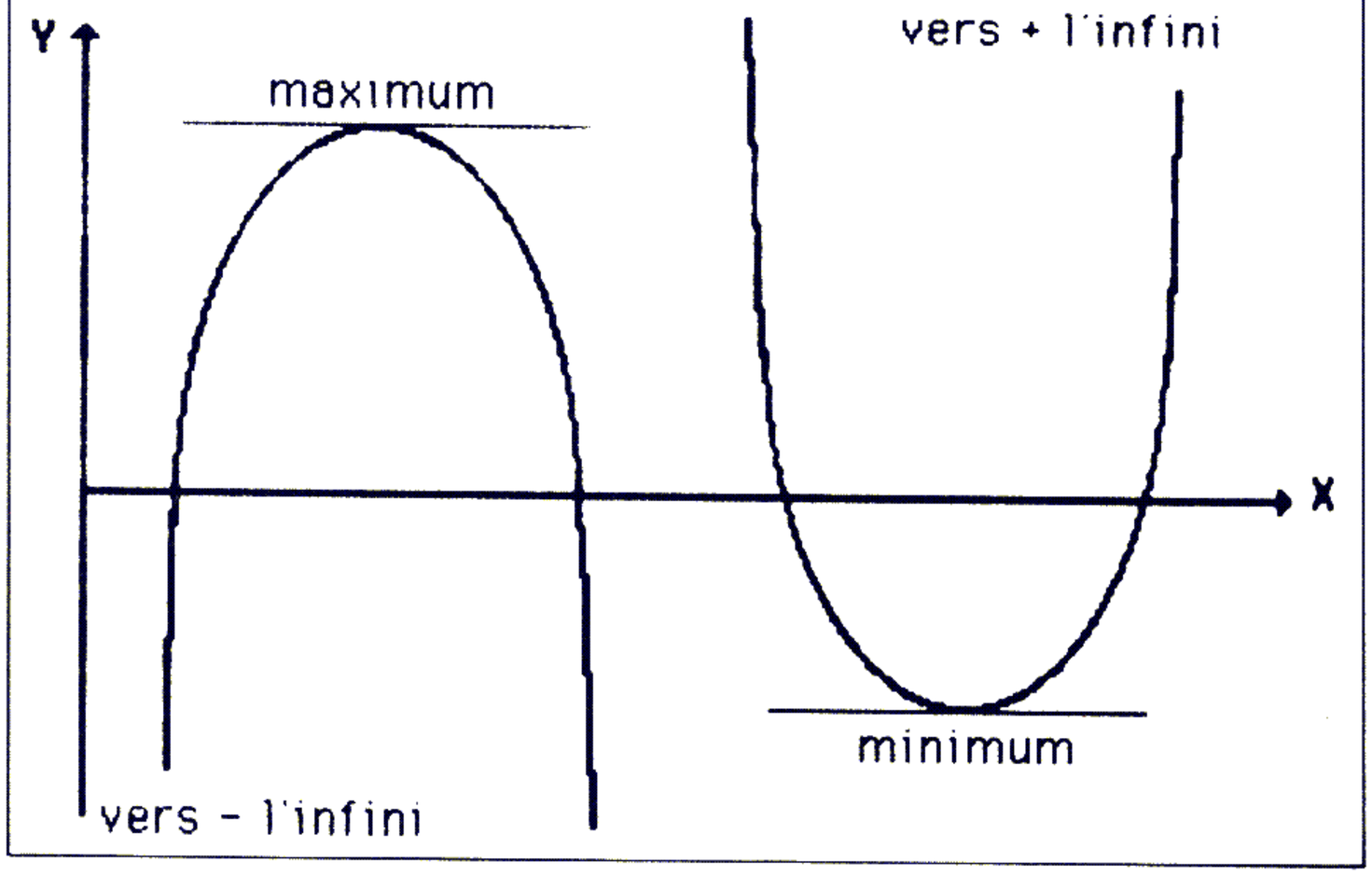
EXTREMUM OPTIMUM EST. Telle pourrait être la devise de *Misez p'tit* : l'extrême est optimum... *LIST* n° 8 a lancé en défi la recherche de l'extremum d'une équation du second degré : trouver l'X qui maximise ou minimise, sans distinction, le résultat Y (voir graphe).

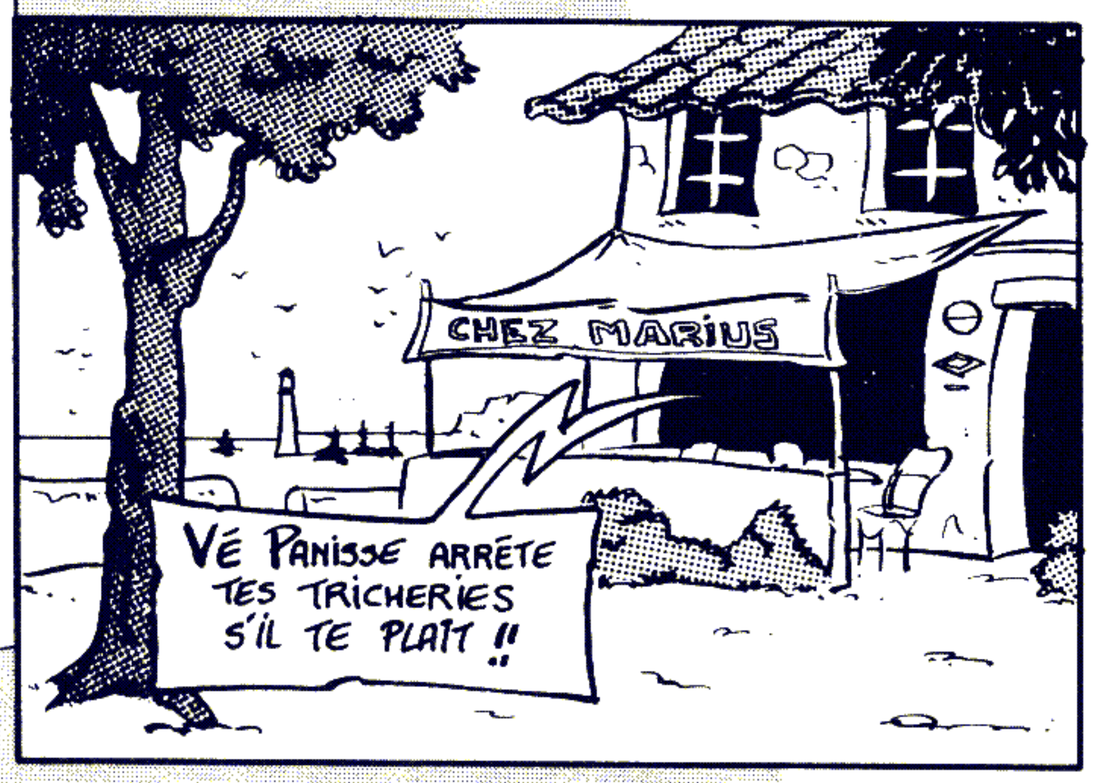
Comme promis, sur le même thème, voici le défi suivant : saurez-vous déterminer si l'extremum trouvé est un maximum ou un minimum ? Le programme de référence, d'Olivier Byrde, occupe 14 octets (sur 10 pas de programme) et trouve l'extremum de $3X^2-4X+5$, soit 2/3, en 32 centièmes de seconde. S'il s'agit d'un maximum, le flag 0 est armé, sinon (c'est le cas) baissé. Les coefficients sont introduits en Pile dans l'ordre c, b, a mais a, b, c serait plus agréable... Bien sûr, aucun registre de mémoire n'est employé.

On détaillera le mois prochain, le programme vainqueur, sa méthode et — surtout — la signification des dérivées première et seconde illustrée grâce à la résolution d'un tel problème de recherche d'extremum. Une bonne et ultime occasion de réviser à l'approche des examens...

Jean-Christophe KRUST

```
58 SF 00
59 ACOS
60*LBL "AGD"
61 FC?C 00
62 XEQ 03
63 STO L
64 CLX
65 2
66 XEQ 02
67 ST+ L
68 CLX
69 4
70 ST/ L
71 X<> L
72 TAN
73 LN
74 X<> L
75*LBL 01
76 CLX
77 ANUM
78 X<> L
79 RTN
80*LBL 02
81 ST* L
82 CLX
83 FS? 43
84 PI
85 FC? 43
86 180
87 RTN
88*LBL 03
89 CLA
90 ARCL X
91 END
```





LE ROUGE ET LE NOIR SUR LE TAPIS VERT

SI vous n'êtes pas un pilier de casino, le mot « martingale » ne fait pas partie de votre vocabulaire usuel. Le dictionnaire le définit comme suit (au figuré) : « système de jeu qui prétend assurer un bénéfice par une augmentation progressive de la mise ». Et si c'était vrai ? Rien ne vous empêche de le vérifier vous-même sans risquer un sou.

A la roulette, il y a 37 cases numérotées de 0 à 36. Parmi ces cases, 18 sont rouges et 18 noires ; la dernière (le zéro) n'est ni rouge ni noire. Si un joueur mise rouge et que le rouge sort, il reçoit le double de sa mise. Mais si le noir ou le zéro sort, sa mise lui est confisquée par la banque.

Si ce joueur applique la martingale la plus simple, après chaque coup perdant, il rempile en doublant systématiquement sa mise précédente. Comme le rouge sort tout de même de temps en temps, en moyenne presque une fois sur deux dans le jeu décrit, il est assuré de recevoir le double de sa dernière mise après 1, 2, 3, ..., X coups.

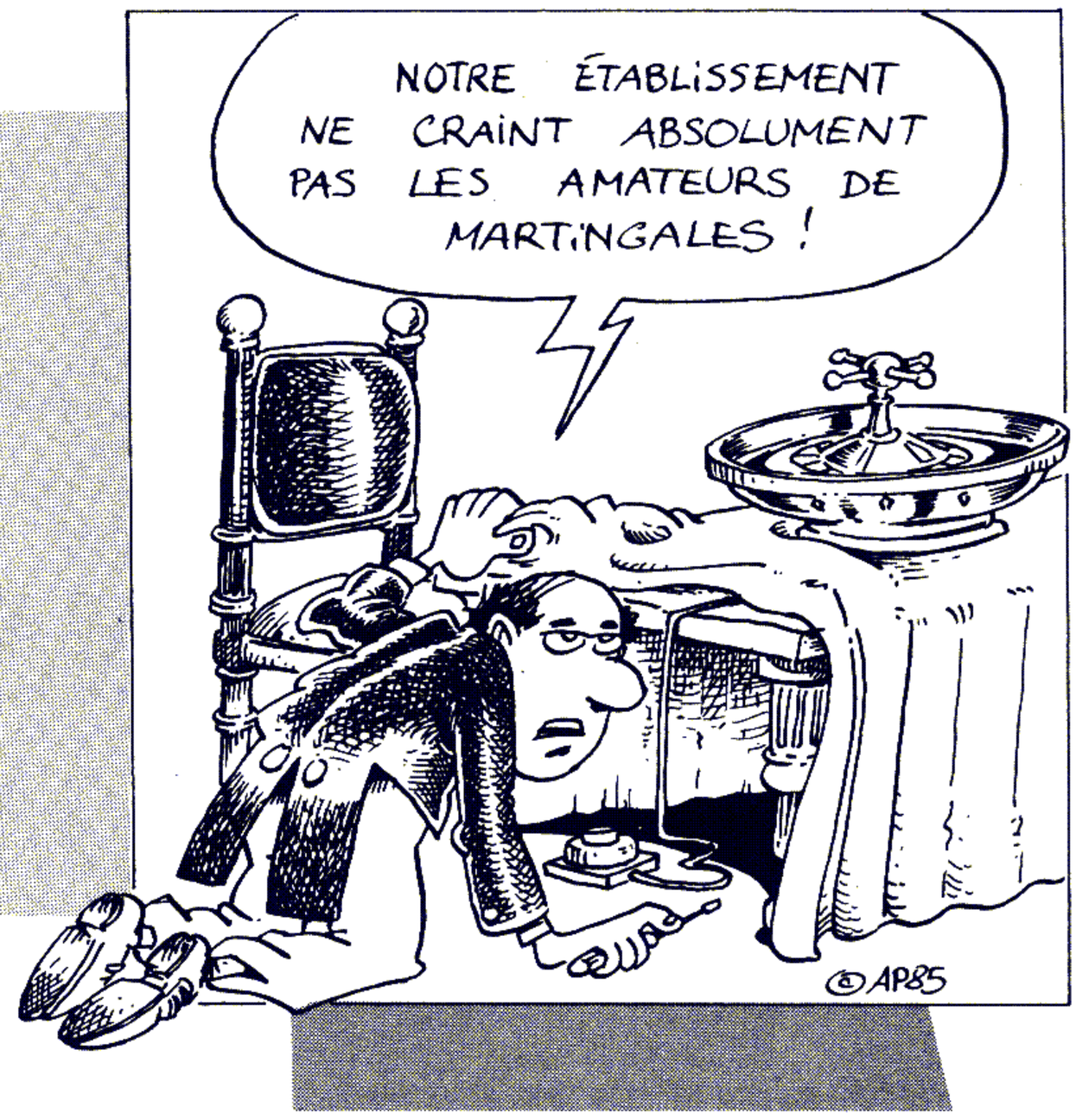
Pour y voir plus clair, analysons la série suivante : noir - noir - rouge, avec une mise initiale d'une unité (une fois 10 francs ou 1 fois 1 000 francs selon l'épaisseur de votre portefeuille). Nous avons donc misé 1 + 2 + 4 = 7 francs et nous recevons 2 × 4 = 8 francs, soit un bénéfice égal à la mise initiale. Et il en sera de même si, par malchance, le rouge ne sort qu'au seizième coup !

Et si je triplais la mise chaque fois que je perds ? Ce serait alors sensationnel : à la première sortie du rouge, le gain serait égal à la demi-somme de la mise initiale et de la mise finale, soit, pour la série déjà citée, 5 fois la mise initiale... Mais il y a évidemment un os. Pour l'expliciter, reprenons la série noire, la série toujours possible de 16 sorties du noir successives qui exigerait une mise totale de $1 + 2 + 4 + 8 + 16 + (...) + 65\,536 = 2 \times 2^{16} - 1 =$ 131 071 fois la mise initiale. Même si vous disposez de cette somme (il y a certainement un prince du pétrole parmi nos lecteurs), le croupier opposera son veto en faisant valoir le règlement interne qui prévoit une mise maximum.

Simuler la perte d'une chemise

Par ailleurs le casino doit couvrir ses frais et surtout remplir les caisses de l'État. En plus des 36 cases rouges ou noires, il y a toujours au moins une case zéro, de sorte que la probabilité de gain est au mieux de 18/37 (au lieu de 1/2), soit une perte statistique de 1/37 des mises cumulées.

Le programme proposé est une simu-



La martingale
Programme pour TRS 80 mod. 1 ou 3
Auteur Roger Brousmiche


```
 1 '
2 '   L A   M A R T I N G A L E
3 '
4 '   CONFIGURATION: MODELE 1 OU 3
5 '   4K A 48K, DISQUE OU CASSETTE
6 '
7 '   (C) LIST ET ROGER BROUSMICHE
8 '
10 RANDOM : CLEAR 100 : CLS
20 PRINT TAB(16)"R O U L E T T E   B I N A I R E" : GOSUB 360
30 '----------entree des donnees
40 PRINT : PRINT "MISE INITIALE = 1", : MT=0
50 INPUT "EN POCHE ";E
60 INPUT "MULTIPLICATEUR DE LA DERNIERE MISE EN CAS DE PERTE ";K
70 INPUT "LIMITE DU GAIN PAR PARTIE ";LA : LI=E+LA
80 INPUT "NOMBRE DE PARTIES ";NP : M=0 : C=0 : G=0
90 INPUT "NOMBRE DE 'ZERO' ";NX : NZ=36+NX : B1=18
100 INPUT "AFFICHAGE (P)ARTIEL OU (T)OTAL, ENTREZ 'P' OU 'T' ";P$
110 IF P$="T" THEN PRINT "APPUYEZ SUR LA BARRE D'ESPACE POUR GELER L'AFFICHAGE"
120 '----------corps du programme
130 N=0 : MI=1 : M=M+1 : EP=E : IF M>NP GOSUB 270 : GOTO 40
140 N=N+1
150 B=RND(NZ) : IF B>B1 THEN B=0 : GOTO 180
160 B=1 : EP=EP+MI : GOSUB 220 : IF EP>=LI GOSUB 240 : GOTO 130
170 MI=1 : GOTO 140
180 IF MI>=EP GOSUB 240 : GOSUB 270 : GOTO 40
190 EP=EP-MI : GOSUB 220 : MI=K*MI : IF MI>EP THEN MI=EP+1 : GOTO 180
200 GOTO 140
210 '----------affichages
220 MT=MT+MI : C=C+1 : IF P$="T" PRINT N;B,INT(MI+.5),INT(EP-E+.5),INT(EP+.5)
230 ZZ$=INKEY$ : IF PEEK(14591) THEN 230 ELSE RETURN
240 G=G+EP-E : PRINT C;"COUPS",M;"PARTIE(S)","GAIN";INT(G+.5),"MISES";INT(MT+.5) : FOR I=1 TO 500 : NEXT
250 RETURN
260 '----------fin du jeu
270 PRINT "Mise initiale = 1 , en poche ="E", multiplicateur ="K
280 PRINT "Limite gain ="LA","NP"parties ,"NX"zero(s)"
290 PRINT C"coups , mises totales =";INT(MT+.5);
300 IF G<0 THEN PRINT ", perte finale ="; ELSE PRINT ", gain final =";
310 PRINT ABS(INT(G+0.5))
320 PRINT "Pertes statistiques =";INT(MT*NX/NZ+0.5)
330 IF M<NP PRINT "Vous avez du quitter le jeu au cours de la partie #";M
340 PRINT : INPUT "<ENTER> pour un autre essai, ou <BREAK> pour terminer ";ZZ$ : RETURN
350 '----------presentation
360 PRINT STRING$(64,136)
370 PRINT "SI VOUS ETES RICHE, VOUS POUVEZ ESSAYER :"
380 PRINT "EN POCHE = 10 000 (FOIS LA MISE INITIALE)"
390 PRINT "MULTIPLICATEUR DE LA DERNIERE MISE EN CAS DE PERTE = 2"
400 PRINT "NOMBRE DE PARTIES = 30 (UNE PAR JOUR PENDANT UN MOIS)"
410 PRINT "LIMITE DU GAIN PAR PARTIE = 20 (FOIS LA MISE INITIALE)"
420 PRINT "ET VOUS SEREZ PROBABLEMENT ENCORE UN PEU PLUS RICHE..."
430 PRINT : PRINT "MAIS SI VOUS ETES PAUVRE, AVEC EN POCHE 100 FOIS LA MISE"
440 PRINT "INITIALE, VOUS DEVREZ PEUT-ETRE ECOURTER VOS VACANCES..."
450 RETURN
```

lation à l'écran des heurs et malheurs du joueur qui a opté pour une martingale de son choix. Il suffit d'entrer le pactole dont on dispose en début de jeu, le coefficient multiplicateur de mise, le nombre de parties à jouer, le gain maximum par partie et le nombre de zéros de la roulette. L'écran affiche alors le numéro d'ordre du coup suivi de 1 ou 0 (gagné ou perdu), de la mise, du gain et de la somme restant en poche.

Le déroulement des opérations étant très rapide, on peut geler l'affichage en appuyant sur la barre d'espacement. Les résultats globaux sont affichés à la fin de chaque partie.

En résumé, le scénario-type est le suivant : vous êtes en vacances et vous décidez de passer une heure chaque jour au casino local avec en poche une somme fixe. Vous commencez par miser un montant minimum que vous incrémenterez selon la martingale choisie jusqu'au premier gain.

Vous recommencerez avec la mise minimum et ainsi de suite jusqu'à ce que le bénéfice de la journée atteigne un montant prédéterminé.

Et vous reviendrez le lendemain, sauf si, en cours de martingale, vous n'avez plus assez de plaques pour rempiler. Dans ce cas, vous rentrerez tranquillement chez vous en jurant de ne plus jamais mettre les pieds dans un casino. Plus jamais, c'est promis ?

Roger BROUSMICHE

LE BASIC DU CANON X-07

APRÈS plus d'un an d'existence, le Canon X-07 n'a pratiquement pas vieilli. Ce véritable portable reste quasiment seul dans sa gamme de prix. Bien entouré (ses périphériques comprennent une interface vidéo), il est doté d'origine d'un Basic qui vaut le coup d'œil.

Voici donc un engin qui fait partie des machines amusantes à utiliser. Ses possibilités d'extension sont un atout que bien d'autres pourraient lui envier. La version de base dispose de 8 Koctets de mémoire vive, mais divers modules peuvent étendre cette capacité jusqu'à des valeurs plus conséquentes. L'écran à cristaux liquides est certes un peu étroit (4 lignes de 20 caractères chacune), mais il peut être complété par un affichage sur téléviseur plus confortable. Port cassette, haut-parleur intégré, port série, port parallèle, port d'extension, coupleur optique de transmission... Mais qu'en est-il du Basic ?

Ce Basic, intégré à la machine, est disponible dès la mise sous tension. Le premier message qui apparaît d'ailleurs à cet instant indique son origine : Microsoft encore une fois. Il s'agit d'une version adaptée au NSC 800, le cœur de la machine (avatar du Z 80, version faible consommation : piles obligent), et il occupe 28 Koctets. Une taille prometteuse !

Les calculs directs se font selon la syntaxe habituelle du Basic. Ici, on ne retrouve pas certaines facilités offertes par d'autres machines de poche : la présence du PRINT ou du « ? » devant les calculs est impérative. Par contre, le petit animal est assez doué en calcul : il offre sans manipulation spéciale 14 chiffres significatifs, ce que la taille de l'écran rend encore plus impressionnant.

Cette qualité dans les calculs est due au fait que tous sont effectués en double précision, d'où une consommation mémoire plus importante. La notation scientifique couvre l'éventail de 1E – 62 à 1E + 62, ce qui n'est pas si mal.

La bibliothèque des fonctions destinées aux manipulations arithmétiques et scientifiques de tous poils est tout à fait similaire à celle de n'importe quel autre ordinateur individuel. Les calculs trigonométriques se font exclusivement en radians. Les fonctions particulières peuvent être créées par DEF FN. DEFINT et DEFSNG permettront l'économie de mémoire que la double précision a parfois tendance à gaspiller. Parmi les fonctions classiques : FIX fournit la partie entière d'une expression, mais sans l'arrondir ; EXP rend l'exponentielle de l'argument associé, LOG rend le logarithme naturel (il n'y a pas de log décimaux), MOD rend le reste et ¥ (Yen) le quotient des divisions entières.

La machine effectue à la demande des calculs ou des conversions en octal ou hexadécimal. Deux fonctions logiques peu courantes sont enfin disponibles, qui sont XOR (OU exclusif) et EQV (équivalent).

Quatre types de variables

Les variables maintenant ? Rien de neuf. Les noms des variables peuvent comporter beaucoup de caractères, mais seuls les deux premiers sont significatifs. QWERTY et QWYTRE sont un seul et même identificateur. D'aucuns trouveront cela gênant, mais n'oublions pas la taille de l'écran, la consommation de mémoire et l'augmentation du temps de frappe ! Les variables chaînes contiennent peu de caractères, sauf si, par un CLEAR judicieux, vous leur réservez la place suffisante. Une chaîne est susceptible de contenir alors jusqu'à 255 vrais et bons caractères.

STRING$ permet de constituer une chaîne de longueur définie. Les traitements de chaînes sont classiques et n'appellent aucun commentaire particulier, si ce n'est l'existence de la fonction

INSTR qui permet de rechercher l'occurrence d'une chaîne dans une autre et qui fait souvent défaut sur des machines bien plus grosses... Les variables peuvent être définis (par DEF xxx) comme entières, réelles simple ou double précision, ou encore comme chaînes, et reconverties en d'autres types par CINT, CSNG et CDBL. Enfin, DIM autorise l'usage de variables indicées.

Si l'écran est petit, il a le mérite d'être adressable point par point. Aussi, des fonctions graphiques existent, et elles sont faciles à employer : CIRCLE, PSET, PRESET, LINE permettent de dessiner tout ce que l'on veut. POINT permet de tester l'état d'un point de l'écran. De plus, 62 caractères sont redéfinissables (FONT$). A noter que le jeu normal de caractères disponibles au clavier comporte de nombreux caractères semi-graphiques, majuscules, minuscules (accentuées !), lettres grecques,

Liste des mots clés du Basic du X-07

ABS
ALM$
AND
ASC
ATN
BEEP
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLOAD?
CLS
COLOR
CONSOLE
CONT
COS
CSAVE
CSNG
CSRLIN
DATA
DATE$
DEFDBL
DEFFN
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DIR
ELSE
END
EQV
ERASE
ERL
ERR
ERROR
EXEC
EXP
FIX
FN
FONT$
FOR
FRE
FSET
GOSUB
GOTO
HEX$
IF
INIT#
INKEY$
INP
INPUT
INPUT#
INSTR
INT
KEY$
LEFT$
LEN
LET
LINE
LINE INPUT
LINE INPUT#
LIST#
LIST
LLIST
LOAD
LOAD?
LOCATE
LOG
LPRINT
MID$
MOD
MOTOR
NEW
NEXT
NOT
OFF
ON
OR
OUT
OUT#
PEEK
POINT
POKE
POS
PRESET
PRINT
PRINT#
PSET
READ
REM
RESTORE
RESUME
RETURN
RIGHT$
RND
RUN
SAVE
SCREEN
SGN
SIN
SLEEP
SNS
SQR
START$
STEP
STICK
STOP
STR$
STRIG
STRING$
TAB
TAN
THEN
TIME$
TKEY
TO
TR
USING
USR
VAL
VARPTR
XOR
TROFF
TRON

Quelques-uns des caractères disponibles d'origine sur le X-07

symboles katakana (écriture japonaise). Un cache plastique est d'ailleurs livré avec le X-07, destiné à être placé sur le clavier, pour indiquer la position de ces caractères. Tous sont inscrits dans une matrice de 6×8 points. Les octets qui définissent un caractère ont donc 6 bits ; nul n'est parfait...

La gestion de l'écran en mode caractères est aisée avec LOCATE, qui permet de positionner le texte, et PRINT USING, qui formate les éléments affichés.

Les entrées de données sur la console disposent de quelques possibilités : TKEY rend 0 ou −1 si la touche choisie est enfoncée. STRIG est un peu similaire. STICK permet d'utiliser le pavé de commande du curseur comme un joystick. INPUT et INKEY$, plus classiques, sont évidemment présentes. Si l'on ajoute à cet éventail la possibilité de redéfinir 12 touches de fonction, et d'afficher leur contenu à l'écran en permanence grâce à CONSOLE, on s'aperçoit qu'il ne manque pas grand-chose.

La numérotation des lignes s'étend de 0 à 65529. Lignes qui peuvent comporter plus de 80 caractères, c'est-à-dire dépasser la capacité de l'écran, mais à condition d'utiliser l'interface téléviseur. Dans le cas contraire, 79 caractères sont le maximum. Attention, en cas d'erreur, une ligne aussi longue ne pourra plus être éditée : il faudra alors la retaper. En fait, la taille maximum réellement utilisable est de 60 caractères, soit 3 lignes sur l'afficheur. L'éditeur plein écran associé au pavé des quatre touches curseur facilite les corrections et la mise au point des programmes qui bénéficie d'autres fonctions utiles : TRON et TROFF bien sûr, mais aussi ON ERROR, RESUME, ERR et ERL, qui sont de précieux auxiliaires de débogage.

Accès au langage-machine

Les traitements de conditions disposent du ELSE : c'est un bon point. Par contre, les WHILE et UNTIL qui commencent à se répandre dans de nombreux Basic manquent à l'appel, et c'est dommage. Le fameux CONSOLE, dont nous avons déjà parlé, permet de redéfinir le déroulement d'écran, la répétition des touches, le BEEP qui accompagne leur enfoncement, le mode clavier (normal, graphique, numérique), et j'en passe... Pour la musique maintenant, sachez que BEEP est paramétrable en fréquence et durée : le haut-parleur

LE BASIC DU CANON X-07

interne a un son quelque peu « informatique », mais c'est mieux que rien.

Ceux dont le langage-machine est le péché mignon seront servis avec les EXEC, POKE, PEEK, USR et autre VARPTR qui vous ouvrent toutes grandes les portes du système. Seule manque la littérature sur son organisation interne : le manuel est trop succinct sur ce point.

Le X-07 peut limiter ses services à ceux d'une horloge intelligente, à mettre dans votre porte-documents ou au pied d'un lit : il sonnera à l'heure dite et le jour prévu. On peut lui demander la date à tout moment : il répond dans sa langue maternelle en fournissant l'année, le mois et le jour. TIME$, DATE$ et ALM$ sont les maîtres mots de ces fonctions. Le dernier est le plus complexe à programmer, mais il autorise toutes les fantaisies, du style : me réveiller à 11 h 30 le dimanche et à 7 h les autres jours, ou encore : sonner toutes les trois minutes pour faire cuire les œufs à la chaîne. Bref, n'importe quoi !

Puisque le X-07 est électriquement autonome, et parce qu'il consomme très peu, les fonctions d'horloge continuent leur travail même lorsque la machine est arrêtée. C'est pour la même raison que les concepteurs de la machine ont prévu de conserver en mémoire, dans une zone spéciale dont la taille est fixée à volonté par FSET, les fichiers qui y auraient été placés. La commande DIR (le DIRectory des disquettes) permet à tout moment de connaître le contenu de cette zone : titres et espace occupé. SAVE et LOAD permettent les transferts entre la mémoire de travail et cette zone de fichiers. Ces transferts s'effectuent à grande vitesse puisqu'aucun élément mécanique n'entre en jeu. Il s'agit donc en quelque sorte d'un « micro disque virtuel ».

Un périphérique à la fois

Lors de la mise en route, tout programme de la zone de travail reste prêt à l'exécution. Aussi, START$ pourra imposer à la machine d'exécuter celui de votre choix aussitôt qu'elle est mise sous tension.

Les transferts de données en direction du port cassette se font par CSAVE et CLOAD (vérification possible grâce à CLOAD?). Pour l'imprimante, LLIST et LPRINT sont étendus de LPRINT USING, ce qui est bien sympathique ma foi ! Bien entendu, diverses autres instructions sont destinées aux autres périphériques connectables sur les nombreux ports disponibles. Ainsi PRINT#, INPUT#, LINE INPUT#, SNS, INP, OUT, OUT#, qui sont utilisables après INIT#, équivalent de l'open plus connu.

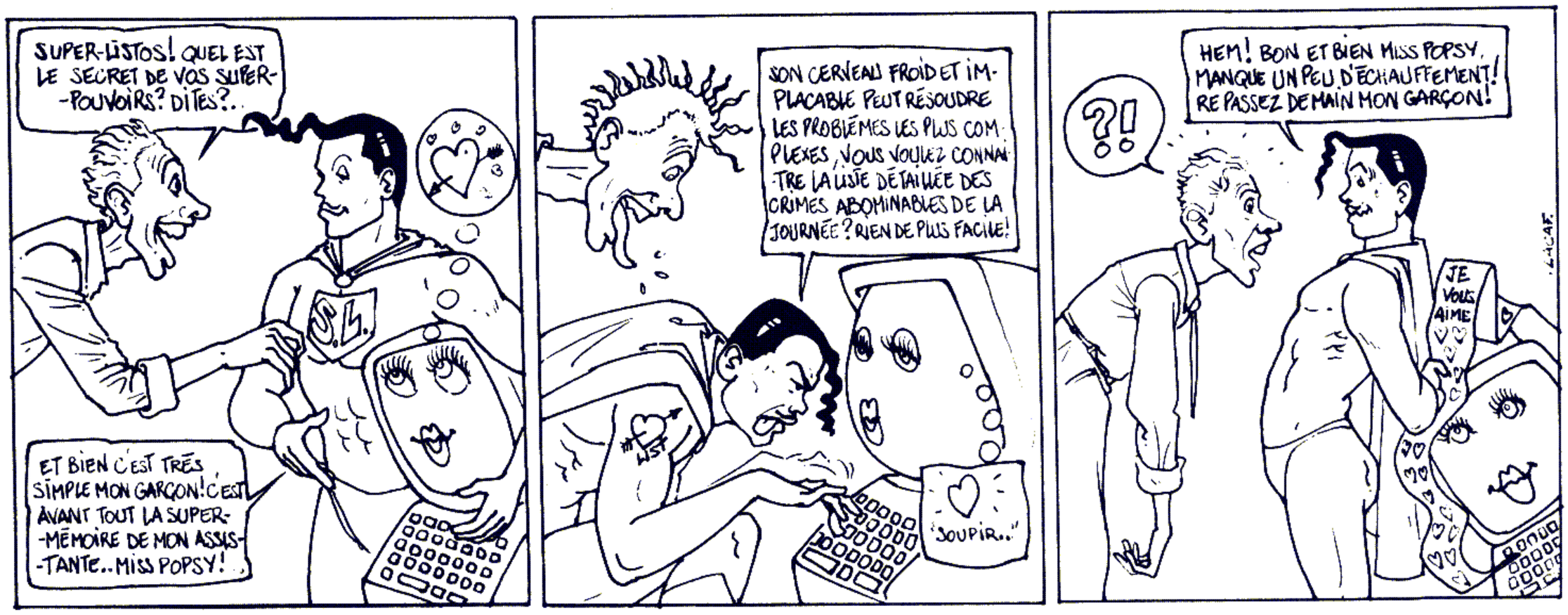
Mais le X-07 a une particularité : il ne gère qu'un seul périphérique à la fois, ce qui fait que l'instruction CLOSE n'existe pas. En effet, soit un nouvel INIT# ouvre un autre fichier, et referme automatiquement le précédent ; soit la fin du programme est rencontrée, et cette fermeture s'effectue. Les divers périphériques sont reconnus par une désignation associée à INIT : KBD représente le clavier, OPT le coupleur optique, CASI le magnétophone en lecture, etc.

Ce tour d'horizon vous aura montré que le Basic du X-07 dispose de caractéristiques intéressantes, qui savent tirer parti, ou compenser la petite taille de la machine. Certaines fonctionnalités sont originales, et les lacunes sont rares. Les possibilités de ce petit bijou le hissent au rang de machines plus chères et nettement plus encombrantes.

Robin BOIS

Fiche technique du Canon X-07

Constructeur : *Canon*
Prix public : *1850 FF ttc en version de base*
Processeur : *NSC 800*
Mémoire vive disponible : *8 Ko, extensibles*
Mémoire morte : *20 Ko, extensibles*
Langage : *Basic d'origine Microsoft*
Variables numériques : *de 1E-62 à 1E+62*
Précision : *14 chiffres significatifs en double précision*
Nombre de mots clés du Basic : *127*



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

AZIMUTEZ, AZIMUTEZ...

L'utilisation intensive du magnétophone sur votre Commodore 64 vous expose à de désagréables péripéties dont la plus sournoise est le défaut d'alignement de la tête de lecture, problème qui survient toujours au moment le plus inopportun conformément à la loi de Murphy, que je salue ici encore !

L'effet de ce défaut est désastreux, puisqu'il conduit à l'impossibilité de relire vos propres cassettes, ainsi que les cassettes commerciales. A noter que ces dernières sont généralement enregistrées en « Turbo », ce qui les rend particulièrement sensibles au problème d'azimutage sus-nommé. Or la correction de ce défaut est à la portée de tout un chacun et en particulier des lecteurs qui seront munis du petit utilitaire que voici :

```
0 REM AZIMUTAGE MAGNETO
  PHONE C.64—— R.BOIS——
1 FORI=0TO20:READA:POKE49152
  +I,A:NEXT:SYS49152
2 DATA 173,142,2,208,15,173,13,220,
  41,16
3 DATA 240,2,169,15,141,24,212,76,0,
  192,96
4 END
```

L'usage de ce programme est des plus simples, puisqu'après l'avoir mis

en mémoire et lancé, il suffit de jouer finement du tournevis tout en prêtant une oreille attentive (qui vous sera rendue par la suite). En effet, l'utilitaire de réglage permet d'entendre sur le haut-parleur dont est assurément muni le téléviseur associé à votre C.64, le contenu de toute cassette placée dans le magnétophone. Quant au tournevis, vous l'aurez choisi de petite taille, cruciforme, et si possible en matériau non magnétique. Il vous servira à tourner légèrement la vis de réglage qui maintient la tête de lecture récalcitrante du magnétophone.

Cette vis demeure cachée sous le capot supérieur du magnétophone, mais un trou a été prévu par le constructeur, à proximité immédiate de la trappe réservée aux cassettes, au-dessus de la touche REWIND. Si vous utilisez un ancien modèle, le trou existe, mais il est caché sous le bandeau métallique qu'il vous faudra percer vous-même au-dessus du mot F.FWD. Mais démontez le capot pour effectuer cette opération ! La vis de réglage doit se positionner exactement sous le trou lorsque le magnétophone est en lecture : profitez-en pour y insérer votre tournevis.

N'importe quelle cassette peut être placée dans le magnétophone (Bee-Gees, Bach ou d'autres Rocker's), mais le mieux est de préparer une bande spéciale sur laquelle seront enregistrées quelques minutes d'un son continu assez aigu.

Ce son, qui sera entendu par le haut-parleur, sera plus ou moins fidèle en fonction de la rotation que vous imprimerez à votre tournevis. Cherchez donc doucement la position qui donne le meilleur son, et vous rendra donc une tête correctement azimutée. Pour arrêter le programme, pressez SHIFT: vous voici maintenant avec un magnétophone quasi neuf, et une nouvelle chaîne quasi HIFI (hum !). Content de cette opération, votre magnétophone vous le rendra au centuple !

Robin BOIS

PC-1500

FAITES VOTRE CHOIX

Après l'article de Michel Dupont à propos des menus sur ordinateurs de poche *(LIST* 6), voici un programme plus spécialement destiné au PC-1500. Le principe de ce menu consiste à déplacer verticalement une fenêtre (l'écran du micropoche) sur une liste d'options. L'utilisateur peut à volonté faire monter ou descendre cette fenêtre grâce aux touches ↓ et ↑ dont le code ASCII est respectivement 10 et 11.

Lorsqu'il a fait son choix, il lui suffit de presser sur la touche ENTER et le sous-programme retenu est exécuté. Pour en avoir la démonstration, tapez le programme proposé puis demandez DEF A. Actionnez les touches ↓ et ↑ de manière à explorer tout le menu. Faites apparaître « Salade » à l'affichage, appuyez sur ENTER et le programme « Laitue » est exécuté.

Marc DANGLA

Un menu
Programme pour PC-1500
Auteur Marc Dangla
Copyright *LIST* et l'auteur

```
 5:"A":CLEAR :
   WAIT 0
10:PAUSE " * * *
   * * MENU * * *
    * * "
20:A=2:PRINT " -
   Entree":GOTO "
   B"
30:A=3:PRINT " -
   Viande":GOTO "
   B"
40:A=4:PRINT " -
   Legume":GOTO "
   B"
50:A=5:PRINT " -
   Poisson":GOTO
   "B"
60:A=6:PRINT " -
   Salade":GOTO "
   B"
70:A=7:PRINT " -
   Fromage":GOTO
   "B"
80:A=8:PRINT " -
   Dessert":GOTO
   "B"
90:A=9:PRINT " -
   Digestif":GOTO
   "B"
100:A=10:PRINT " -
    Vin":GOTO "B"
110:A=11:PRINT " -
    Etc...":GOTO
    "B"

300:"B"B$=INKEY$ :
    IF A<>11AND
    ASC (B$)=10
    GOTO ((A+1)*10
    )
310:IF ASC (B$)=11
    GOTO ((A-1)*10
    )
320:IF ASC (B$)=13
    WAIT :GOTO ((A
    *100)+1000)
330:GOTO (A*10)

 1200:PRINT " + Cr
      udites":END
 1300:PRINT " + To
      urnedos":END
 1400:PRINT " + Ep
      inards":END
 1500:PRINT " + Tr
      uite":END
 1600:PRINT " + La
      itue":END
 1700:PRINT " + Ca
      membert":END
 1800:PRINT " + Ta
      rte":END
 1900:PRINT " + Ea
      u de vie":
      END
 2000:PRINT " + Bo
      urgogne":END
 2100:PRINT " + Fi
      n":END
```

DES PROGRAMMES AU JOUR... ET A L'HEURE

Sur Apple IIe, il est possible de dater ses programmes par l'intermédiaire d'un utilitaire de la disquette système. Sur Apple IIc, on ne peut rien faire !

Or il est assez facile depuis Basic de dater ses programmes. Pour cela, quatre adresses sont concernées. Il s'agit de 49042 et 49043 pour l'heure et les minutes, de 49040 et 49041 pour la date. En ce qui concerne l'heure le problème est très simple : il suffit de poker l'heure en 49043 et les minutes en 49042.

Pour la date, c'est un peu plus compliqué, car deux adresses sont utilisées pour trois données. La méthode de calcul est la suivante :

- si le mois est strictement plus petit que 8, en 49041, poker *année *2* et en 49040, poker *mois*32 + jour ;*
- si le mois est supérieur ou égal à 8, en 49041, poker *(année*2) + 1* et en 49040, poker *(mois – 8)*32 + jour*

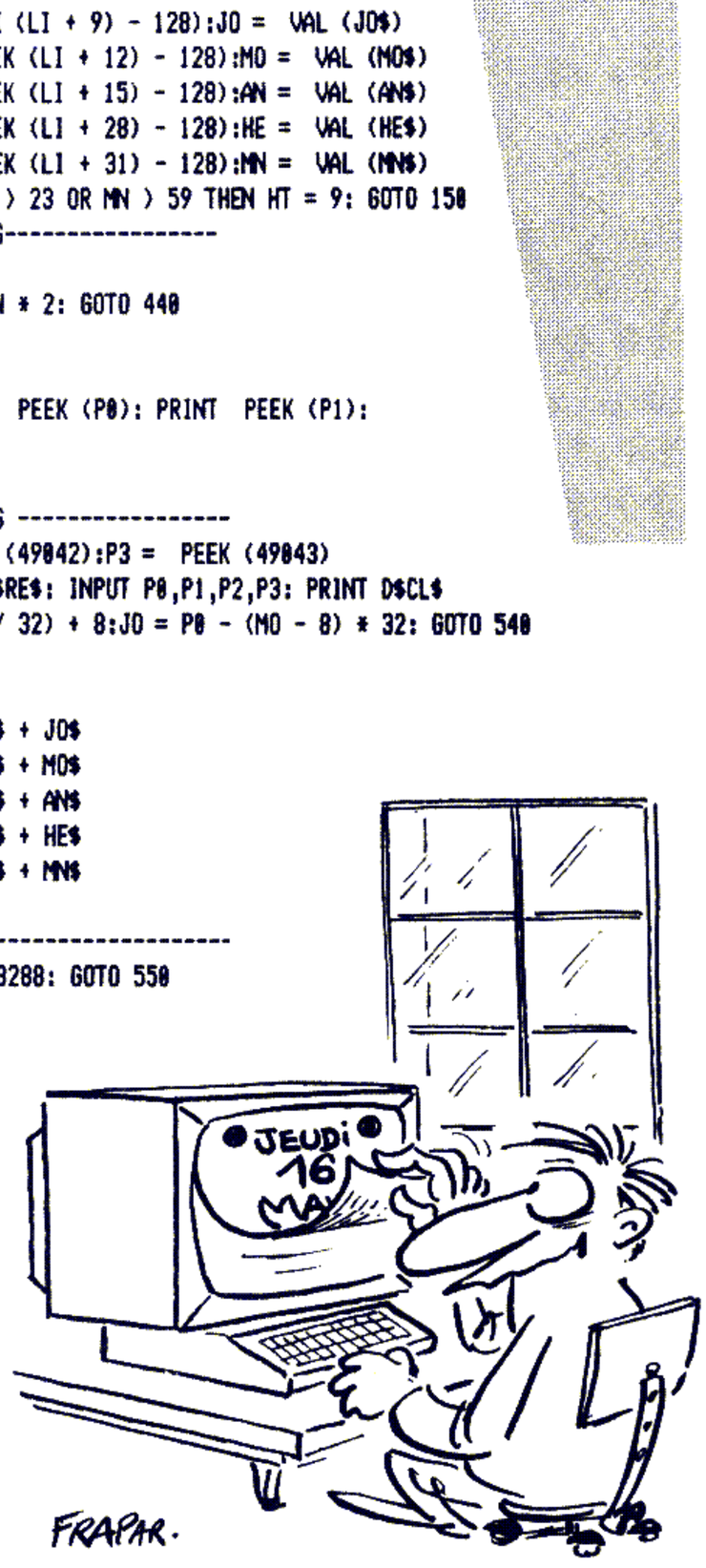
Prenons un exemple : le 24/09/85. Nous sommes alors dans le deuxième cas, où le mois est supérieur à 8 :

- en 49041, poker (85*2) + 1, soit 171 ;
- en 49040, poker (9 – 8)*32 + 24, soit 56.

Prodos va chercher ces valeurs lui-même, puis il les inscrit sur le directory de la disquette. Le programme ci-dessus (placé en *startup*) vous permet à chaque « boot » de mettre la date et l'heure à jour. De plus, il crée automatiquement un petit fichier séquentiel, évitant ainsi de ressaisir la date complète.

Calendrier
Programme pour Apple IIc
Auteur Michel Aubry
Copyright *LIST* et l'auteur

```
100  REM      ----------------------CALENDRIER PRODOS 1----------------------------------------
101  REM  -----                                                  ----------------------------
103  PRINT  CHR$ (21)
104  ONERR  GOTO 1520
105 O$ = "0":SE$ = "/":SP$ = ":":DA$ = "DATE ":HR$ = "     HEURE ":D$ =  CHR$ (4):TI$ = "- CALENDRIER
    PRODOS -":SO$ = "----------------":AI$ = "Fleches (droite) (gauche) (return)"
110 NF$ = "FICH.CALENDRIER":OP$ = "OPEN" + NF$:RE$ = "READ" + NF$:WR$ = "WRITE" + NF$:CL$ = "CLOSE" + NF$
115 HT = 9:AV = 1:LI = 1192
130  HOME : VTAB 3: HTAB 10: PRINT TI$: HTAB 13: PRINT SO$: VTAB 20: HTAB 5: PRINT AI$
140  VTAB 10: HTAB 4: GOSUB 500: PRINT DA$JO$SE$MO$SE$AN$HR$HE$SP$MN$;
150  HTAB HT: GET A$
160  IF A$ =  CHR$ (13) THEN 310
162  IF A$ <  >  CHR$ (21) THEN 200
165  IF HT = 10 OR HT = 13 OR HT = 29 THEN AV = 2: GOTO 190
170  IF HT = 16 THEN HT = 28: GOTO 150
180  IF HT = 32 THEN 150
190 HT = HT + AV:AV = 1: GOTO 150
200  IF A$ <  >  CHR$ (8) THEN 230
203  IF HT = 12 OR HT = 15 OR HT = 31 THEN AV = 2: GOTO 220
205  IF HT = 28 THEN HT = 16: GOTO 150
210  IF HT = 9 THEN 150
220 HT = HT - AV:AV = 1: GOTO 150
230  IF ( ASC (A$) < 48 OR  ASC (A$) > 57) THEN 150
240  PRINT A$;:DP = 1: GOTO 165
300  REM  ----------------------PEEK PAGE TEXTE -------------------
310 JO$ =  CHR$ ( PEEK (LI + 8) - 128) +  CHR$ ( PEEK (LI + 9) - 128):JO =  VAL (JO$)
320 MO$ =  CHR$ ( PEEK (LI + 11) - 128) +  CHR$ ( PEEK (LI + 12) - 128):MO =  VAL (MO$)
330 AN$ =  CHR$ ( PEEK (LI + 14) - 128) +  CHR$ ( PEEK (LI + 15) - 128):AN =  VAL (AN$)
340 HE$ =  CHR$ ( PEEK (LI + 27) - 128) +  CHR$ ( PEEK (LI + 28) - 128):HE =  VAL (HE$)
350 MN$ =  CHR$ ( PEEK (LI + 30) - 128) +  CHR$ ( PEEK (LI + 31) - 128):MN =  VAL (MN$)
360  IF JO > 31 OR JO < 1 OR MO > 12 OR MO < 1 OR HE > 23 OR MN > 59 THEN HT = 9: GOTO 150
400  REM  ------------------ECRITURE ADRESSES PRODOS----------------
410 P0 = 49040:P1 = 49041:P2 = 49042:P3 = 49043
420  IF MO < 8 THEN  POKE P0,MO * 32 + JO: POKE P1,AN * 2: GOTO 440
430  POKE P0,(MO - 8) * 32 + JO: POKE P1,AN * 2 + 1
440  POKE P3,HE: POKE P2,MN
450  IF DP = 1 THEN  PRINT D$OP$: PRINT D$WR$: PRINT  PEEK (P0): PRINT  PEEK (P1):
     PRINT MN: PRINT HE: PRINT D$CL$: HOME : NEW
460  FOR I = 1 TO 100: NEXT : HOME : NEW
500  REM  ------------------LECTURE ADRESSES PRODOS ----------------
510 P0 =  PEEK (49040):P1 =  PEEK (49041):P2 =  PEEK (49042):P3 =  PEEK (49043)
515  IF P0 = 0 AND P1 = 0 THEN  PRINT D$OP$: PRINT D$RE$: INPUT P0,P1,P2,P3: PRINT D$CL$
520  IF (P1 / 2 -  INT (P1 / 2)) THEN MO =  INT (P0 / 32) + 8:JO = P0 - (MO - 8) * 32: GOTO 540
530 MO =  INT (P0 / 32):JO = P0 - MO * 32
540 AN =  INT (P1 / 2)
550 JO$ =  STR$ (JO): IF  LEN (JO$) < 2 THEN JO$ = 0$ + JO$
560 MO$ =  STR$ (MO): IF  LEN (MO$) < 2 THEN MO$ = 0$ + MO$
570 AN$ =  STR$ (AN): IF  LEN (AN$) < 2 THEN AN$ = 0$ + AN$
580 HE$ =  STR$ (P3): IF  LEN (HE$) < 2 THEN HE$ = 0$ + HE$
590 MN$ =  STR$ (P2): IF  LEN (MN$) < 2 THEN MN$ = 0$ + MN$
600  RETURN
1500  REM  -----------------TRAITEMENT ERREUR-----------------------
1520  IF  PEEK (222) = 5 THEN  PRINT D$CL$: CALL  - 3288: GOTO 550
1530  PRINT "ERREUR N"; PEEK (222)
```



Explications du programme

Ligne 103 : force le mode 40 colonnes sur Apple IIc.

Lignes 130 à 240 : gestion du curseur, permet également d'éviter des saisies erronées.

Linges 300 à 360 : lecture des données directement dans la mémoire écran, permet de frapper <RETURN> à n'importe quel endroit de la ligne.

Lignes 400 à 460 : algorithme de calcul des valeurs à poker dans les adresses concernées, écrit également ces valeurs dans un fichier s'il y a eu des modifications (drapeau DP).

Lignes 500 à 600 : algorithme de lecture, si les adresses ont une valeur nulle (cas à l'allumage) lecture du fichier.

Michel AUBRY

UNE LIGNE SUR DEUX

Comme le manuel du CPC 464 l'indique, l'ordinateur, quand une liste sort sur l'imprimante, envoie un passage à la ligne — CHR$(10), alias "line feed" —, et une retour-chariot — CHR$(13) — après chaque ligne imprimée. Or sur la plupart des imprimantes, le passage à la ligne se fait automatiquement, ce qui conduit à un double espacement des lignes, et donc à une perte de papier.

```
10 ' UTILITAIRE DE LISTING IMPRIMANTE
20 ' SUPRIME LE LINE FEED DU CPC 464
30 ' ENTRE DEUX LIGNES DE PROGRAMMES.
40 ' LANCER LE PROGRAMME PUIS TAPER
50 ' WIDTH 255 [ENTER] SUIVIT DE LIST #8
60 '
70 '
80 CLS:MEMORY 42999
90 FOR AD=48427 TO 48429
100 READ VALEUR
110 POKE AD,VALEUR
120 NEXT AD
130 FOR AD=43000 TO 43008
140 READ VALEUR
150 POKE AD,VALEUR
160 NEXT AD
170 DATA 195,248,167,254,10,32,1,135,207
,242,135,201
180 '
190 '     PROGRAMME ASSEMBLEUR
200 '
210 ' CP 10    ;Compare A avec 10
220 ' JRNZ 1   ;Branche si diff. de 10
230 ' ADD A,A  ;Double A a 20
240 ' RST 08
250 '
260 '
```

Garder ses lignes
Utilitaire pour Amstrad CPC 464
Auteur Bruno Fiter
Copyright *LIST* et l'auteur

La solution permettant d'éviter ce gâchis consiste à utiliser une courte routine en Assembleur pour détecter le CHR$(10) et le changer. Le "jumpblock" écrivant un caractère est en mémoire vive (&8D2B) ; il peut donc être modifié (lignes 90-120) en un saut à l'adresse 43000 qui contient le programme suivant : CP 10 - JRNZ,1 - ADD A,A - RST 08.

Et le mode d'emploi ? Mais il est très simple. On exécute d'abord l'utilitaire puis on charge le programme à lister et l'on tape WIDTH 255 et LIST#8.

Bruno FITER

LES DESSOUS DE LA SOUSTRACTION

Une soustraction n'étant jamais qu'une addition dont un des termes est négatif, les observations que nous avons relevées dans *LIST* 7 (page 74) au sujet de l'addition de deux nombres restent valables pour la soustraction, sur un PC-1211. Si l'on pose A = 0.000 030 000 487 et B = 0.725 alors A − B = −0.724 969 999 513 ; mais A − B + 0.72 provoque à l'affichage −0.004 969 999 52, approximation médiocre si l'on doit exploiter tous les chiffres significatifs. En revanche, − B + A + 0.72 fournit le résultat parfait −0.004 969 999 513. On voit donc que dans l'écriture d'une différence, il est toujours souhaitable de mettre en tête le plus grand des deux nombres en valeur absolue.

Le cas des sommes algébriques avec signes mixtes est un peu plus complexe. Soit Z = 0.987 000 000 6, A = 0.000 000 000 353 2 et B = 0.000 000 000 118 9 ; alors Z + A − B = 0.987 000 000 834 3.

Si l'on applique le précepte précédent, pour obtenir les 10 derniers chiffres significatifs de Z + A − B, on introduit Z + A − B − 0.98 et le résultat est 0.007 000 000 835, ce qui n'est pas trop mal. Mais essayons de faire mieux par permutation des termes.

En fait, les six combinaisons possibles de l'ordre d'introduction des variables, agrémentées ou non de la mise entre parenthèses des deux der-

Tableau des résultats : une même opération sous plusieurs formes

Ordre d'introduction des données	Résultat
Z+A−B−0.98	0.007 000 000 835
Z−B+A	 835
A+Z−B	 832
−B+Z+A	 843
A−B+Z	 83
−B+A+Z	 83
Z+(A−B)	 834
Z+(−B+A	 834
A+(Z−B)	 83
−B+(Z+A)	 84
A+(−B+Z)	 84
−B+(A+Z)	 84

niers termes— la mise entre parenthèses des deux premiers termes ne change rien aux résultats — conduisent à six valeurs différentes de la somme algébrique recherchée (voir le tableau des résultats).

On voit que les meilleurs résultats s'obtiennent en écrivant Z+(A−B) ou Z+(−B+A). On peut donc optimiser la précision en mettant en queue et entre parenthèses le groupe de variables de faible valeur absolue unitaire par rapport à la variable (ou au groupe de variables) de tête. Sur le plan de la théorie, cette démarche est tout à fait logique, car la machine calcule en bloc l'expression entre parenthèses et par conséquent, quel que soit le nombre de termes sous parenthèses, la dépréciation de la précision n'est jamais affectée que par *une seule* opération, et non par autant d'opérations que de termes.

Cette procédure est particulièrement bénéfique dans le cas des expressions comportant de nombreux termes. Par exemple, affectez (en mode calcul) aux variables Z et A à F les valeurs suivantes :

Z = 0.987 6
A = 0.000 000 000 353 62
B = 0.000 000 001 448 97
C = 0.000 000 025 245 78
D = 0.000 000 317 029 36
E = 0.000 000 512 226 75
F = 0.000 035 471 844 96

Le calcul manuel de Z−A−B−C−D−E−F donne 0.987 563 671 850 56, soit 0.987 563 671 851 avec 12 chiffres significatifs.

A chacun sa réaction

Le PC-1211 fait les calculs sur 12 chiffres. Il en stocke 10 et en affiche 10. Votre ordinateur a peut-être d'autres limites. Il peut donc réagir bizarrement face à d'autres calculs (même des additions). Faites-nous part des résultats les plus étranges qu'il vous propose.

La machine fournit les résultats suivants : Z−A−B−C−D−E−F−0.98 = 0.007 563 671 855, valeur médiocre ; et Z−(A+B+C+D+E+F)−0.98 = 0.007 563 671 851, valeur exacte.

La mise entre parenthèses des derniers termes a donc permis de récupérer quatre unités sur la dernière décimale.

En résumé, si l'on veut obtenir une somme algébrique avec le maximum de précision :

- introduire en premier les nombres présumés les plus grands en valeur absolue,
- introduire ensuite les nombres par ordre décroissant de valeur absolue — si toutefois on en connaît la taille — et mettre entre parenthèses les variables d'ordre de grandeur comparable.

Pierre Ladislas GEDO

DEVINETTE

Quelle que soit la case de départ imposée, le Dr Jivaro, chevauchant sur son fidèle PX-8 aux lignes généreuses, favori du derby d'Epson, jalonne en un clin d'œil les 64 sauts qui lui permettraient de parcourir tout l'échiquier sans retomber deux fois sur la même case.

Connaissant son sens très strict de l'économie, et son goût pervers pour les instructions « à tiroirs », dites combien il a utilisé d'instructions, et subsidiairement, d'octets ? (La réponse se trouve à la page 81).

Pierre BARNOUIN

BASIC

TOURS DE HANOÏ

Rappelons brièvement en quoi consiste ce casse-tête créé par le mathématicien Lucas il y a 100 ans : huit disques, ou plus, de diamètre décroissant de la base au sommet, sont rangés en pile à gauche de l'écran. Il faut reconstituer cette pile à droite de l'écran, en utilisant une pile centrale intermédiaire, et en transférant les disques d'une pile à l'autre, un par un, sans jamais poser un disque sur un autre plus petit que lui.

En Pascal, c'est une application classique de la récursivité. En Basic, ce terme est synonyme de sérieuses complications et de gouffre pour la mémoire des petits systèmes. Nous ramènerons donc prudemment tous les mouvements à un unique cas de figure, par un choix judicieux des variables et procéderons par itération (voir programme page suivante).

L'initialisation occupe la ligne 1 et le corps du programme la ligne 2. La ligne 3 réalise l'unicité des cas de figure, que traite la ligne 4. La ligne 5 efface le disque ôté à une pile et la ligne 6, qui est aussi utilisée pour l'initialisation, affiche le disque ajouté à une autre.

Les « tours » sont numérotées de 0

Tours de Hanoï
Programme pour PX-8
Auteur Pierre Barnouin


```
1 SCREEN 3,,0:CLS:FOR I=1 TO 8:H(0)=I:T(0,I)=I:GOSUB 6: NEXT
2 WHILE H(2) < 8 : GOSUB 3:WEND:END
3 Q=(2*Q+2*R)MOD 3:R=(Q+1)MOD 3:IF T(Q,H(Q))<T(R,H(R)) THEN
  SWAP Q,R
4 H(R)=H (R) + 1:SWAP T(Q,H(Q)), T(R,H(R))
5 LOCATE 5+26*Q,9-H(Q):PRINT SPC (16) ; : H(Q) = H (Q)-1
6 K=9-T(R,H(R)):LOCATE 26*R-K+13,9-H(R):PRINT STRING$(2*K,
  140);:RETURN
```

à 2, les disques sont toujours enlevés à la tour Q et ajoutés à la tour R. H (I) est la hauteur de la tour I et T(I,J) correspond au disque situé à l'étage J de la tour I. Plus précisément le rayon de celui-ci est K = 9 – T (I,J). Les huit étages de la tour de gauche mettent 75 secondes à effectuer les 255 mouvements qui permettent de les empiler à droite, sur l'écran d'un Epson PX-8.

Suivant le matériel utilisé, vous devrez peut-être combiner des « périphrases » pour remplacer WHILE... WEND, MOD, ou SWAP, mais la reconstitution de l'algorithme que traduit le programme ne devrait pas vous donner trop de mal. Précisons qu'il s'agit ici d'une règle simple et non d'une formule algébrique.

Pierre BARNOUIN

PC-1500

CRIC, CRAC...

Alors ! Un mois après la publication (*LIST* 8, page 75) du programme Cric-crac, un coffre-fort pour Sharp PC-1500, vous ne l'avez pas encore forcé ? Tss...

La solution pour trouver la bonne combinaison du coffre et déjouer les systèmes de brouillage et d'alarme n'était pourtant pas inaccessible :

- la combinaison du coffre se trouve grâce aux notes produites par chaque chiffre ; aux notes les plus élevées, correspondent les bons chiffres ;
- toutes les 15 secondes, le brouillage se met en route et la combinaison est automatiquement changée ;
- l'alarme se déclenche 1 mn 30 après le démarrage du programme, il faut donc percer la combinaison rapidement ;
- enfin, cette alarme se déclenche aussi si l'on presse 11 fois successivement la même touche (une même roue a fait un tour complet).

Voici achevée votre première leçon de perceur de coffre-fort, faites-en profiter votre entourage, maintenant que vous savez...

Frédéric CHARLES

HP-41C APPLICATION SYNTHÉTIQUE

ET POURTANT, ILS TOURNENT

A quoi peuvent bien servir les registres de mémoire supplémentaire apportés par les modules d'extension X-Function et X-Memory ? A diverses choses, nous apprend la documentation, mais pas à exécuter des programmes qui s'y trouvent stockés.

Quoi de plus stimulant pour le programmeur qu'une interdiction ? Là encore, la programmation synthétique étend les possibilités de base de la HP-41C en permettant l'exécution d'un programme directement en mémoire étendue.

Préparation : le Cric doit être assigné à une touche du clavier (traditionnellement à Σ +). La méthode d'assignation est rappelée en encadré. Un premier programme doit se trouver en mémoire centrale. Peu importe sa longueur. Quant au programme stocké dans l'X-mémoire, il doit être le pre-

Assigner le Cric à Σ +

Faire un Memory LOST : OFF puis ON avec la touche ← pressée.
ASN BEEP Σ +
ASN PACK Σ –
(PRGM)
ENTER↑
GTO..
CATALOG 1 immédiatement suivi de R/S. Il s'affiche END
(alpha)

←	il s'affiche	4094 ENTER
BST	attendre	4093 ENTER
BST		4092 0
BST		4091 LBL 00
BST		4090 BEEP
←		4089 LBL 03
←		4088 LBL 08
√x		4089 ^{T}C

(prgm)
GTO..

Le Cric (affichage de XROM 05,03) est assigné à la touche Σ + et PACK à Σ –

suite page 80

mier et débuter par LBL B (car c'est d'un XEQ B qu'on l'exécutera). On ne doit y trouver aucun LBL A ou LBL a.

Ci-prêt, en mode de calcul, faire GTO.. pour compacter la mémoire et positionner correctement le pointeur de programme. Puis, évidemment en mode *User,* exécuter le Cric (touche Σ+, affichage de XROM 05,03). Enfin, une pression de la touche 1/x conduit à l'exécution du premier programme conservé en X-mémoire à condition, bien sûr, qu'il débute par LBL B (car 1/X en mode User correspond à XEQ B).

Tout ne se passe pas, malheureusement, aussi facilement qu'en mémoire centrale. En particulier, il n'est pas possible d'user de sauts (GTO) comme par exemple un GTO à un autre programme de la mémoire étendue. A moins qu'un lecteur parvienne à dépasser cette nouvelle difficulté ?

Stéphane LASTRAJOLI

THOMSON

LANCEMENT AUTOMATIQUE SUR DISQUETTE

Voici une procédure simple pour lancer automatiquement un programme enregistré sur disquette (TO7,TO7-70,MO5). Vous recopiez le Dos-Basic sur une disquette vierge (faire Backup 0 et suivre les instructions).

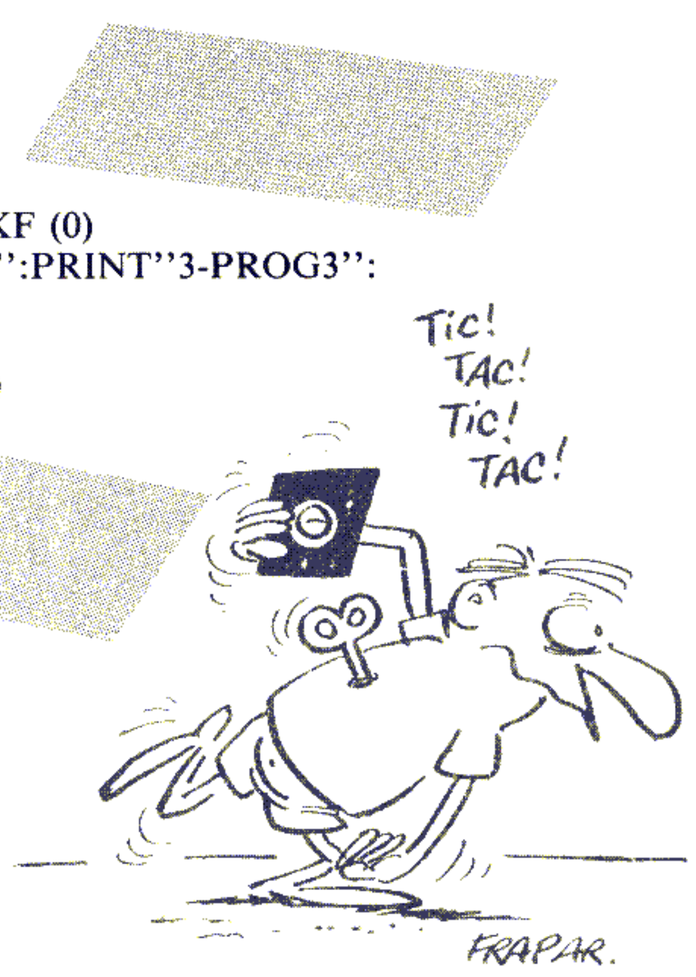
Ensuite, écrivez un programme que vous appellerez AUTO-BAT, chaque fois que vous frapperez la touche 2 à l'initialisation, le Dos-Basic puis le programme AUTO-BAT se chargeront. Le programme AUTO-BAT sera ensuite lancé automatiquement.

Supposons que ce programme vous donne la liste des programmes enregistrés sur la disquette et affectés d'un numéro. Il vous suffira alors de taper le numéro choisi pour lancer le programme correspondant.

Si, à la fin de chaque programme, vous avez ajouté RUN "AUTO.BAT", alors, vous reviendrez au menu offert par AUTO.BAT. En cas de problème nécessitant une réinitialisation du TO7, appuyez sur la touche 2 du menu général et vous relancerez AUTO.BAT.

Par exemple, si quatre programmes — PROG1, PROG2, PROG3, PROG4 — sont enregistrés sur la disquette numéro 23, vous pouvez écrire le programme AUTO-BAT suivant :

```
 10 CLS
 20 PRINT"Disquette N°23"
 30 PRINT"Nombre de blocs libres" ; DSKF (0)
 40 PRINT"1-PROG1":PRINT"2-PROG2":PRINT"3-PROG3":
    PRINT"4-PROG4":PRINT"5-FIN"
 50 A$=INPUT$(1):CH=VAL(A$)
 60 ON CH GOTO 100, 200, 300, 400, 500
100 RUN"PROG1"
200 RUN"PROG2"
300 RUN"PROG3"
400 RUN"PROG4"
500 CLS:END
```



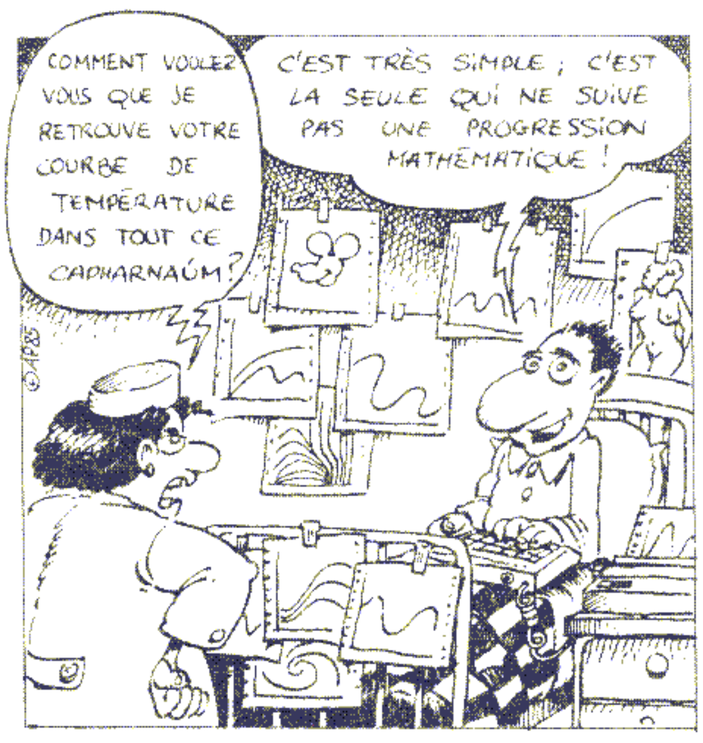
Ce petit programme n'est qu'un canevas, vous pourrez y ajouter tous les raffinements possibles : choix au crayon optique, descriptifs des programmes, etc.

Jean-Paul CARRÉ

PB-700

ENCORE DES COURBES

L'étude d'une fonction mathématique est tout de même plus aisée en présence de sa courbe représentative. Avec le PB-700 et un programme adapté, on pourra la voir à l'écran ou sur le papier de l'imprimante. Auparavant, le programme aura étudié la fonction entre deux limites fixées, calculé l'échelle et placé les axes.



Les commentaires du programme n'apparaissent pas dans la liste sous forme de REM, mais ils peuvent être détaillés. A la ligne 15, se situe le calcul du champ d'étude de la fonction et celui du pas. Si les bornes d'étude sont respectivement négative et positive, cette ligne calcule aussi la position de l'axe des ordonnées. Si les deux bornes sont négatives, la longueur d'étude est la valeur absolue de la borne inférieure de l'axe des X (ligne 20), auquel cas on ne voit que la partie négative de l'axe des abscisses, tracée à droite de l'écran (V = 159). Si les deux bornes sont positives, l'axe est à gauche de l'écran (V = 0, ligne 25), la longueur d'étude correspondant à la plus grande valeur des X, S.

La boucle FOR...NEXT (lignes 35 à 60) étudie la fonction point par point, recherchant son maximum (ligne 50) et son minimum (ligne 55). Le calcul de l'échelle s'effectue à la

```
  1 REM COURBE UNIVERSELLE
  5 CLEAR :CLS :W=16
 10 INPUT "Petit X";R,"Grand X";S
 15 D=ABSR+ABSS:U=D/160:IF R<0 THEN IF
S>0 THEN U=R-U:GOTO 35
 20 IF ABSR>ABSS THEN D=-R:U=159:GOTO
30
 25 D=S:U=0
 30 U=D/160
 35 FOR X=R TO S STEP U
 40 GOSUB 200
 50 IF Y>A THEN A=Y
 55 IF B<Y THEN B=Y
 60 NEXT X
 70 E=(ABSA+ABSB)/32
 75 IF B=0 THEN W=31
 80 IF A=0 THEN W=0
 85 CLS :DRAW(0,W)-(159,W):DRAW(U,0)-(
U,31)
100 FOR X=R TO S STEP U
110 GOSUB 200
130 Y=W-INT(Y/E)
140 IF Y>=0 THEN IF Y<32 THEN DRAW(INT
(X/U+U),Y)
150 NEXT X
160 IF INKEY$="" THEN 160
170 PRINT "Ech 1/";E;:END
200 Y=SIN(2*X)
210 RETURN
```

ligne 70, 32 étant le nombre de points de la verticale de l'écran du PB-700.

S'il n'y a pas de valeur négative de X, l'axe se trouve en bas de l'écran (W = 31, ligne 75) ; s'il n'y a pas de valeur positive, l'axe des X se trouve en haut de l'écran (W = 0, ligne 80).

Le repère est affiché (ligne 85), puis la fonction est tracée point par point (lignes 100 à 150). La ligne 160 attend qu'une touche soit frappée, elle empêche un terrible READY P0 de venir tout effacer.

Avant d'utiliser ce programme, il faut introduire, à la ligne 200, l'équation de la fonction dont on souhaite la courbe représentative. Après avoir entré les bornes d'étude de cette fonction, le repère s'affiche puis la courbe apparaît. L'appui sur RETURN donne alors l'échelle.

Enfin, il faudra faire attention à l'ensemble de définition de la fonction dont on souhaite la courbe : si l'intervalle d'étude introduit n'est pas inclus dans cet ensemble, un message d'erreur (erreur mathématique) est inévitable. C'est notamment le cas si X = 0, avec la fonction Y = 1/X.

Olivier DUFAILLY

Courbes
Programme pour PB-700
Auteur Olivier Dufailly

QL

COPIE DE SÉCURITÉ

Voici un utilitaire qui permet d'éviter la fastidieuse série de « Copy mdv1 identificateur to mdv2 identificateur » quand on doit copier entièrement un microdrive. Le programme lit automatiquement la liste des segments dans le directory, il calcule si l'espace libre est suffisant et, si oui, il recopie un par un les segments.

Pour améliorer le programme, on

RÉPONSE A LA DEVINETTE DE LA PAGE 75

Deux instructions seulement, pour 186 octets. Le parcours à 64 sauts est un parcours fermé, qui pourra rester toujours le même, et qu'il suffira d'afficher à partir de la case désignée. La première instruction enregistre simplement ce parcours fermé choisi à l'avance.

Reste à demander quelle est la case de départ, à enregistrer la réponse, à élaborer le résultat et à l'afficher. La deuxième instruction règle tout ça d'une seule foulée : du Basic acrobatique en quelque sorte !

```
1 A$="g5e6f4h3g1e2c1a2b4d3e1g2h4f3e5g6h8f7d8b7a5c6d4b3a1c2e3f1
h2g4h6g8e7f5d6c8a7b5a3b1d2c4b6a8c7d5f6e8g7h5g3h1f2e4c3d1b2a4c5a7
b8d7f8h7":PRINT"Case départ?",MID$(A$+LEFT$(A$,126),INSTR
(A$,INPUT$(2)),130)
```

Pierre BARNOUIN

```
100 REMark Franck-Olivier Lelaidier et Augustin Garcia 'duplik'
    version 2.0
110 CLS#0:CLS
120 CSIZE 3,1:AT 5,10:PRINT "DUPLIK":CSIZE 2,0:PAUSE 500:CLS
130 AT 5,0:PRINT " METTRE L'ORIGINAL DANS mdv1"
140 AT 6,0:PRINT " METTRE UN MICRO-DRIVE DANS mdv2"
150 REMark ------------------------------
160 REPeat test
170 AT 9,0:INPUT "VOULEZ-VOUS FORMATER LE MICRO-DRIVE ? ( O/N ) ";b$
180 IF b$="o" OR b$="n" OR b$="N" OR b$="O" THEN
190    IF b$="o" OR b$="O" THEN
200       AT 11,0:INPUT "SON NOM: ";nom$
210       FORMAT "mdv2_"&nom$
220    END IF
230    a=1:IF verif>0 THEN EXIT test
240    AT 14,10:PRINT "PAS ASSEZ DE PLACE":STOP
250 END IF
260 END REPeat test
270 REMark ------------------------------
280 CLS
290 copie
300 CLS:PRINT TO 10;"COPY TERMINEE":AT 10,10:DIR mdv2_
310 STOP
320 REMark ------------------------------
330 DEFine FuNction verif
340 REMark ------------------------------
350 OPEN_NEW #3,mdv2_provi2_temp
360 DIR #3,mdv2_
370 CLOSE #3:OPEN_IN #3,mdv2_provi2_temp
380 INPUT #3,dir2$:INPUT #3,dir2$
390 dir2=dir2$(1 TO 3)
400 CLOSE#3
410 DELETE mdv2_provi2_temp
420 REMark ------------------------------
430 OPEN_NEW #3,mdv2_provi1_temp
440 DIR #3,mdv1_
450 CLOSE #3:OPEN_IN #3,mdv2_provi1_temp
460 INPUT #3,dir1$:INPUT #3,dir1$
470 REMark ------------------------------
480 nb_lect=0
490 DIM a$(40,20)
500 REPeat lect
510 IF EOF(#3) THEN EXIT lect
520 INPUT #3,a$(nb_lect)
530 nb_lect=nb_lect+1
540 END REPeat lect
550 CLOSE #3
560 DELETE mdv2_provi1_temp
570 dir1=dir1$(1 TO 3)
580 dir11=dir1$(5 TO 7)
590 REMark ------------------------------
600 RETurn dir2-(dir11-dir1)
610 END DEFine verif
620 REMark ------------------------------
630 DEFine PROCedure copie
640 REMark ------------------------------
650 FOR i=0 TO nb_lect-1
660 PRINT TO 10;"SAVING ";a$(i)
670 COPY "mdv1_"&a$(i) TO "mdv2_"&a$(i)
680 END FOR i
690 END DEFine copie
```

pourra permettre à l'utilisateur de choisir les segments qu'il veut copier.

Augustin GARCIA
Franck-Olivier LELAIDIER

Le cœur du programme

Lignes 350-410 : Création d'un fichier temporaire sur le nouveau microdrive. Ecriture dans ce fichier de la liste des segments et des caractéristiques du microdrive. Lecture du nombre de secteurs libres sur mdv2 et suppression du fichier temporaire.

Lignes 430-480 : Nouvelle création de fichier temporaire sur mdv2. Ecriture dans ce fichier de la liste des segments et des caractéristiques du microdrive (source).

Lignes 480-490 : Initialisation du nombre de lectures à faire ainsi que du tableau indicé a$.

Lignes 500-540 : Boucle de lecture. Tant que ce n'est pas la fin du fichier, on récolte dans a$ le nom de segment. Incrémentation du nombre de lectures.

Lignes 570 et 580 : Lecture du nombre de secteurs libres ainsi que du nombre total de secteurs sur mdv1.

Ligne 600 : La fonction retourne au programme principal le paramètre dir2 – (dir11 – dir1), c'est-à-dire le nombre de secteurs libres sur mdv1 moins le nombre de secteurs à copier.

Lignes 630 à 690 : Les fichiers qui n'apparaissent pas dans la liste des segments ne sont pas recopiés. La procédure ne fait rien d'autre que ce que l'utilisateur aurait fait à la main. Elle récupère dans le tableau le nom de segment et le recopie.

Recopie de microdrive
Utilitaire pour QL
Auteurs Augustin Garcia et
Franck-Olivier Lelaidier

OÙ AVAIT-ON LA MÉMOIRE ?

Dans l'article de Sylvain Coron (*LIST* 7, pages 75 et 76), une erreur s'est glissée juste à la dernière phrase. Il fallait lire (ou plutôt nous aurions dû écrire) : « La routine est transposable **en mémoire** à condition… » et non pas : « La routine est transposable sur d'autres machines… » Toutes nos excuses à nos lecteurs et à Sylvain Coron.

ZX 81

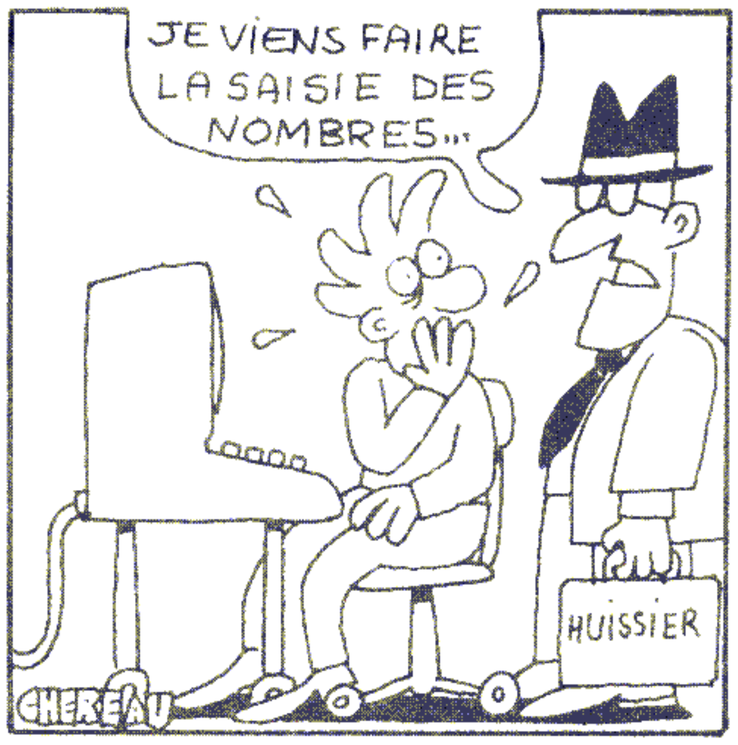
SAISIE DE NOMBRES

Lors d'un INPUT, le ZX 81, comme son successeur le ZX Spectrum, n'autorise la saisie que sur la dernière ligne de l'écran.

Le petit sous-programme assez élémentaire qui vous est proposé ci-après permettra la saisie de nombres à un emplacement plus standard. Il pourra également être utilisé pour des saisies de texte si les conventions faites pour Annuler, Valider ou Finir sont modifiées.

Les nombres entrés au clavier sont placés dans un tableau A largement dimensionné (ici 100 éléments) ; le sous-programme pourra être appelé en plusieurs endroits du programme principal, en faisant simplement GOSUB SAISIE. Les nombres seront, dans ce cas, enregistrés dans le tableau à la suite les uns des autres. La méthode employée permet de travailler sur des nombres de longueur quelconque.

La touche A peut être frappée à tout moment pour Annuler une entrée erronée ; la touche V permet au contraire de Valider la donnée introduite. Pour finir, taper F comme Fin : le programme de démonstration affichera alors la liste des nombres saisis.




```
   10 REM SAISIE DE NOMBRES SUR Z
X  81
   20 DIM A(100)
   30 LET SAISIE=1000
   40 LET D=1
   50 GOSUB SAISIE
   60 REM VERIFICATION DE LA SAIS
IE
   70 CLS
   80 FOR F=1 TO D-1
   90 PRINT A(F)
  100 NEXT F
  110 STOP
 1000 REM SAISIE
 1010 CLS
 1020 PRINT AT 3,6;"SAISIE DE NOM
BRES";AT 5,0;"A POUR ANNULER";AT
 6,0;"V POUR VALIDER";AT 7,0;"F
POUR FINIR"
 1030 LET N$=""
 1040 LET R=1
 1050 PRINT AT 10,6;"NOMBRE ?
                    "
 1060 PRINT AT 10,14+R;"-"
 1070 PRINT AT 10,14+R;CHR$ 128
 1080 LET F$=INKEY$
 1090 IF F$="A" THEN GOTO 1030
 1100 IF CODE F$>27 AND CODE F$<3
8 THEN GOTO 1140
 1110 IF F$="V" THEN GOTO 1180
 1120 IF F$="F" THEN RETURN
 1130 GOTO 1060
 1140 PRINT AT 10,14+R;F$
 1150 LET N$=N$+F$
 1160 LET R=R+1
 1170 GOTO 1060
 1180 LET N=VAL N$
 1190 LET A(D)=N
 1200 LET D=D+1
 1210 GOTO 1030
```

Saisie de nombres
Programme pour ZX 81
Auteur Yvon Pérès


```
SAISIE DE NOMBRES

A POUR ANNULER
V POUR VALIDER
F POUR FINIR

NOMBRE ? 12345
```

Ce programme n'accepte que les nombres entiers, mais il sera facile de l'adapter au cas des nombres décimaux ou négatifs, en modifiant simplement la ligne 1100 (prévoir F$ = "." ou F$ = "−").

Yvon PÉRÈS

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...

28

La ronde des variables

Nous avons vu, dans le premier numéro de *LIST* (jeu 3), que pour échanger le contenu de deux variables, I et J, il n'est pas indispensable d'utiliser une variable auxiliaire. En effet, au lieu d'écrire :

K = I
I = J
J = K

on peut utiliser, sous réserve des débordements de mémoire, les instructions suivantes :

I = I + J
J = I − J
I = I − J

Nous vous proposons d'aller plus loin et d'écrire, toujours sans utiliser de variable auxiliaire K, deux séquences d'instructions qui échangent les contenus de *trois* variables A, B et C.

1. Trouver la première séquence qui échange les contenus de A, B et C par une rotation à gauche : (A, B, C) devient (B, C, A). Le contenu de A va dans B, celui de B dans C et celui de C dans A.

2. Trouver la deuxième séquence qui échange les contenus par une rotation à droite : (A, B, C) devient (C, A, B). Le contenu de A va dans C, celui de B dans A et celui de C dans B.

29

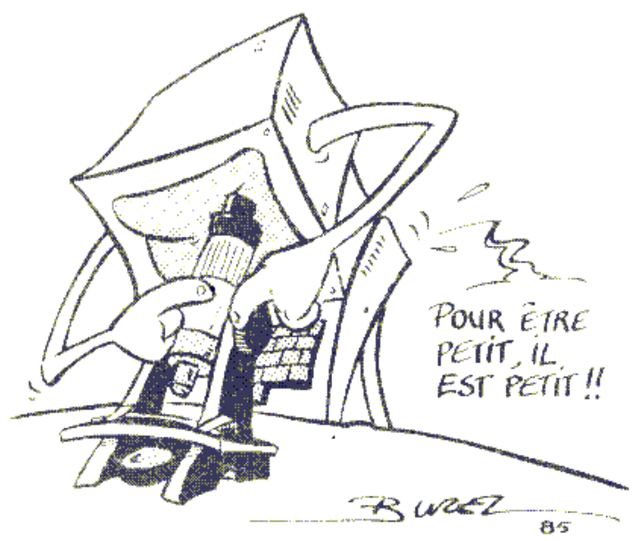
En peu d'instructions

Notre casse-tête 17, (voir *LIST* 6, page 80) proposait de construire la matrice unité, le plus rapidement possible. Notre objectif est maintenant d'obtenir le même résultat avec le plus petit nombre d'instructions possible. En APL, nous pouvons l'écrire en une seule ligne : (N,N) ⍴ (1,N ⍴ 0)

En effet, l'instruction (1, N ⍴ 0) définit une séquence de N + 1 valeurs, la première valant un, et les N suivantes étant égales à zéro. Le tableau (N,N) est donc rempli avec un certain nombre de fois cette séquence de nombres.

Dans des langages tels que le Basic,

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



nous sommes obligés d'utiliser les boucles explicites. Ainsi, nous devons par exemple écrire :

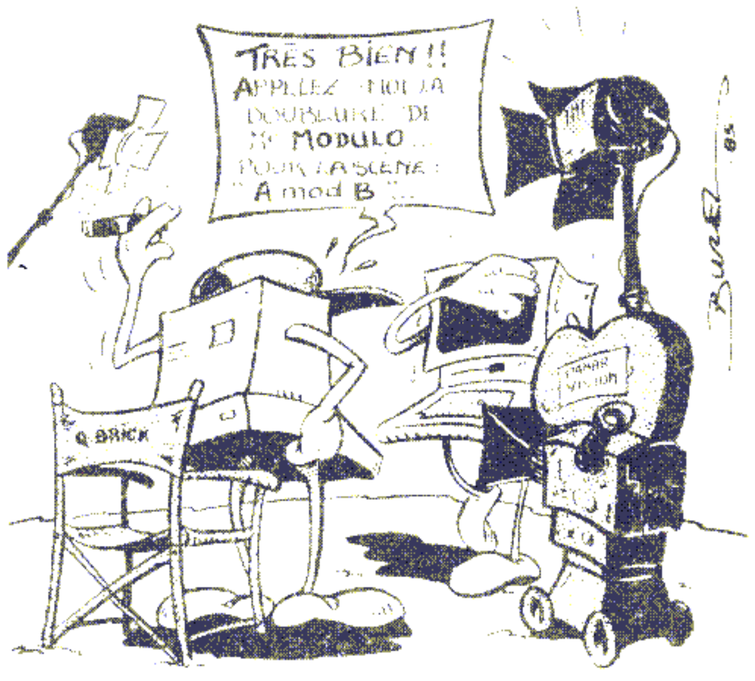
```
10 for L=1 to MAX
20    for C=1 to MAX
30       if L=C then goto 60
40       A(L,C)=0
50       goto 70
60       A(L,C)=1
70    next C
80 next L
```

Mais il est possible de modifier ce programme pour qu'il ne dépasse pas cinq lignes, tout en fournissant le même résultat. Toutefois, il est interdit d'utiliser des instructions qui ne sont pas présentes dans toutes les implantations de Basic, comme le ELSE de IF-THEN-ELSE ou le fait que les variables venant d'être déclarées sont automatiquement initialisées à zéro, ou les instructions matricielles qui remettent à zéro tous les éléments d'un tableau.

30

Modulo

Vous connaissez bien l'opérateur modulo : A MOD B fournit le reste de la division entière de A par B. Il est très pratique, et nous l'utilisons très souvent. Toutefois, il existe de nombreux Basic qui ne possèdent pas cet opérateur. Pour pouvoir en disposer, vous devrez écrire une fonction qui, en une ligne, fournira le même résultat.



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 !

23

Un programme anticonvivial

Un utilisateur du programme « anticonvivial » proposé ne peut pas manquer de faire des remarques. En voici dix, il peut y en avoir d'autres.

1. Lorsque ce programme est exécuté, il n'indique pas le traitement qu'il réalise, c'est-à-dire la moyenne de valeurs comprises entre 0 et 20.

2. L'utilisateur ne sait pas que la première donnée à entrer est le nombre de valeurs.

3. La saisie du nombre de valeurs n'est pas contrôlée. Si ce nombre dépasse 100, une erreur se déclenchera au moment de l'entrée de la 101e, alors que cela pourrait être testé dès le début du programme.

4. Le nombre de valeurs dont il faut faire la moyenne doit être entré au début du programme, ce qui oblige à les compter. L'ordinateur pourrait se charger de ce travail, l'entrée d'une valeur sentinelle (par exemple, −1) indiquant la fin de la liste.

5. Le programme ne fonctionne que si le nombre de valeurs ne dépasse pas 101. Cette limite aurait pu être facilement étendue si les valeurs n'étaient pas stockées dans un tableau, mais totalisées au fur et à mesure de la saisie.

6. En cours de saisie, on ne sait pas quel est le rang de la valeur à entrer. Ceci risque d'être très gênant si l'utilisateur est interrompu dans son travail.

7. Il n'est pas possible de corriger une valeur. A la moindre erreur, c'est-à-dire si une mauvaise valeur a été entrée, il faut arrêter le programme, et saisir à nouveau tous les nombres depuis le début.

8. Le programme effectue le contrôle des valeurs seulement une fois qu'elles ont été toutes entrées. Il aurait été préférable d'effectuer ce contrôle au moment de la saisie.

9. Il n'est pas indiqué que la valeur affichée, à la fin de l'exécution du programme, est la moyenne des valeurs entrées.

10. Le message « Valeur erronée » n'indique pas le numéro de la valeur rejetée par le programme.

24

La parité des mots

Le programme suivant met 0 dans la variable appelée Parite, si le nombre de bits à 1 dans N est pair. Il y met 1, dans le cas contraire.

```
10 Parite=0
20 Parite=(Parite + N MOD 2)
   MOD 2
30 N=N DIV 2
40 IF N<>0 THEN GOTO 20
```

Ici, l'opérateur MOD fournit le reste de la division entière, alors que DIV fournit le résultat de cette même division. Si votre ordinateur ne dispose pas de telles fonctions, vous pouvez utiliser le programme suivant :

```
10 Parite=0
20 Parite=Parite + N − 2*INT(N/2)
30 N=INT(N/2)
40 IF N<>0 THEN GOTO 20
50 Parite=Parite − 2*INT(Parite/2)
```

Il faut noter que cette méthode de contrôle par bit de parité ne permet de

SOLUTIONS DU NUMÉRO PRÉCÉDENT

détecter que les erreurs sur un nombre impair de bits. Si l'on désire une méthode de détection plus sûre, les codes de Hamming peuvent être utilisés. Avec deux bits (au lieu d'un seul bit de parité), il est possible de détecter les erreurs sur deux bits, et de corriger les erreurs sur un bit.

25

En toute logique

L'algèbre logique est régie par les règles suivantes :

la commutativité

A et B = B et A
A ou B = B ou A

l'associativité

A et (B et C) = (A et B) et C
A ou (B ou C) = (A ou B) ou C

la distributivité

A et (B ou C) = (A et B) ou (A et C)
A ou (B et C) = (A ou B) et (A ou C)

l'idempotence

A ou A = A
A et A = A

la complémentarité

A ou (non A) = 1
A et (non A) = 0

l'involution

non (non A) = A

l'absorption

A ou (A et B) = A
A et (A ou B) = A

les théorèmes de De Morgan

non (A ou B) = (non A) et (non B)
non (A et B) = (non A) ou (non B)

En appliquant ces différentes règles, les équations logiques se simplifient de la façon suivante :

1. non (non B ou A) = (non A) et B
2. (A et B) ou (A et (non B)) = A
3. (A ou B) et (A et B) = A et B
4. non ((A ou B) et B) = non B
5. non ((non A) et (non B)) = A ou B
6. ((non A) et (non B)) ou B = (non A) ou B
7. ((non A) et B) ou (A et B) = B
8. ((non A) et (non B)) ou ((non A) et B) = non A
9. ((non A) et (non B)) ou (A et (non B)) ou (A et B) = A ou (non B)
10. A et (A ou B) et (non B) et (A ou (non B)) = A et (non B)

26

Lequel des trois

1. La vitesse de rotation d'un disque souple est de 300 tours par minute : c'est la valeur couramment rencontrée pour les lecteurs de disquettes de 5 pouces.

2. La vitesse de calcul de Mark 1 était de 12 multiplications par minute : il réalisait la multiplication de deux nombres de 10 chiffres en cinq secondes.

3. En 1981, le nombre moyen de lignes de listes imprimées dans les entreprises utilisant un ordinateur était de 55500 par an et par salarié : cette valeur étonnante représente plus de 200 lignes imprimées par jour et par personne.

4. La productivité moyenne d'un programmeur professionnel travaillant sur un grand projet est de 5 instructions par jour : ce résultat provient d'une étude de Barry W. Boehm, *Software and its Impact*.

5. En 1977, il y avait environ 50000 micro-ordinateurs installés dans le monde.

27

Recherche du DU

Le programme de recherche de la séquence "DU" donne un résultat faux pour une chaîne telle que "ADDUCTION D'EAU" : il ne trouvera pas le "DU" apparaissant au milieu de "ADDUCTION". En effet, l'algorithme utilisé n'est pas valable lorsque la sous-chaîne à rechercher est précédée de son propre premier caractère (ici, un D).

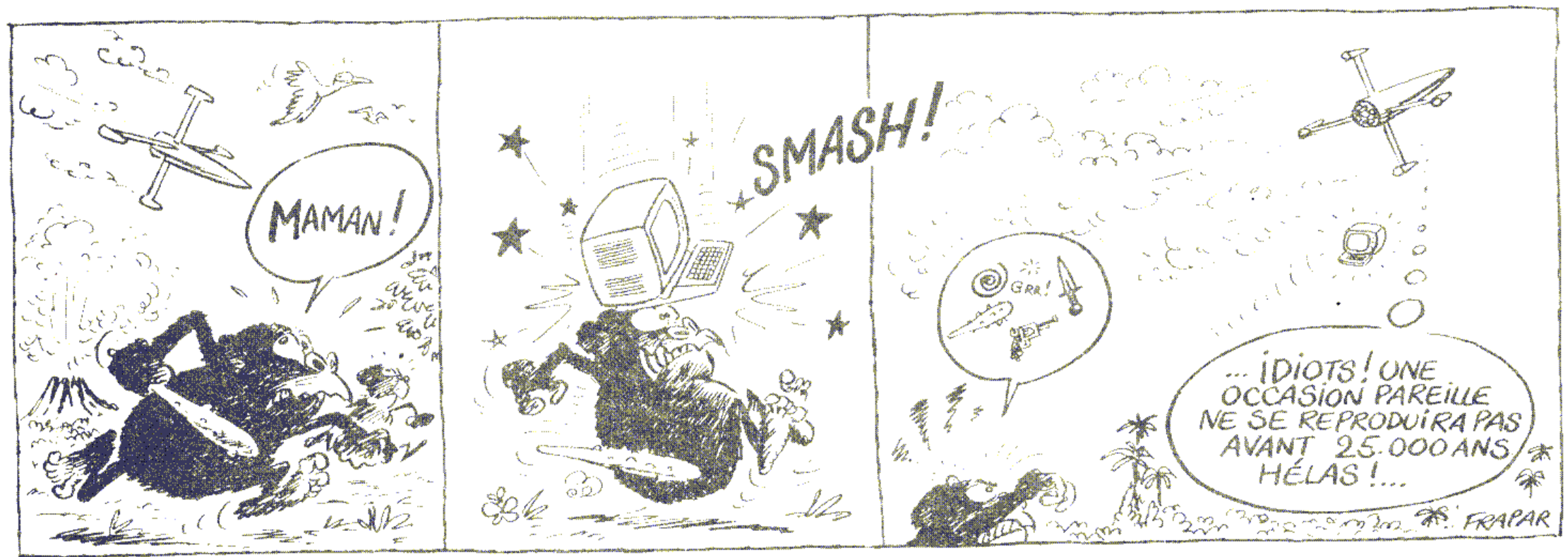
Mais cette — petite — erreur d'algorithme en cachait une grosse : les OR des lignes 50 et 70 doivent être remplacés par des AND. Pour la ligne 70, elle était :

```
70 IF Chaine(I) < > "D" OR Chaine(I) < > "d" THEN GOTO 30
```

Elle devient :

```
70 IF Chaine(I) < > "U" AND Chaine(I) < > "u" THEN GOTO 40
```

(Vous trouverez dans ce numéro, page 37, une description détaillée des différents algorithmes de recherche de sous-chaînes.)



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