ISSN 0183-570X

ESSAI DU THOMSON TO 7

L'ORDINATEUR INDIVIDUEL

LES LANGAGES : UNE NOUVELLE TOUR DE BABEL ?

Essais : Thomson TO 7, PHC-8000, Bus-80, Calcstar, ZX-M

L'ordinateur de la paroisse

Index des 200 programmes publiés en 1982

Jeux : Rubik's cube, martingale, orgue

le magazine de l'informatique pour tous

janvier 1983 n°44

Belgique: 162 FB · Suisse: 8 FS · Canada: 3.95 $ 20 F

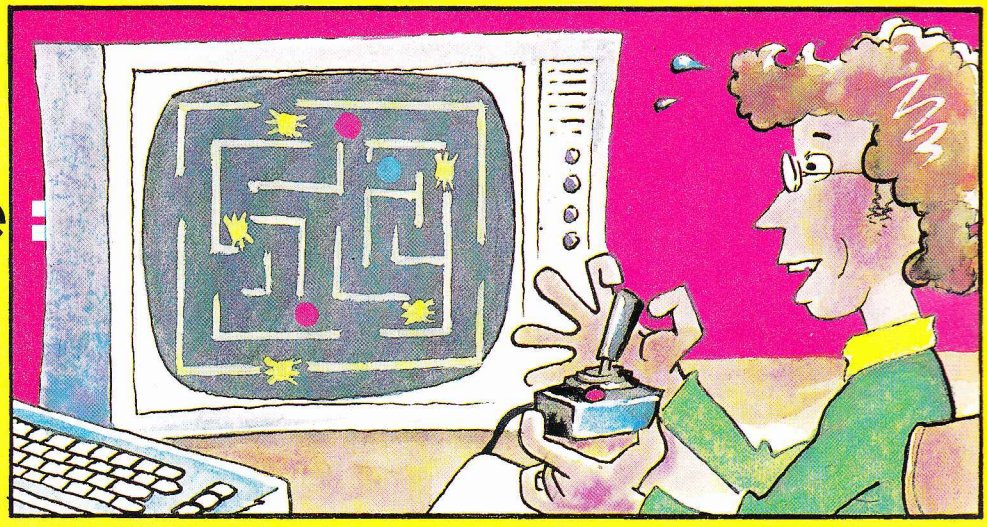
M.2946 - 44 - 20F

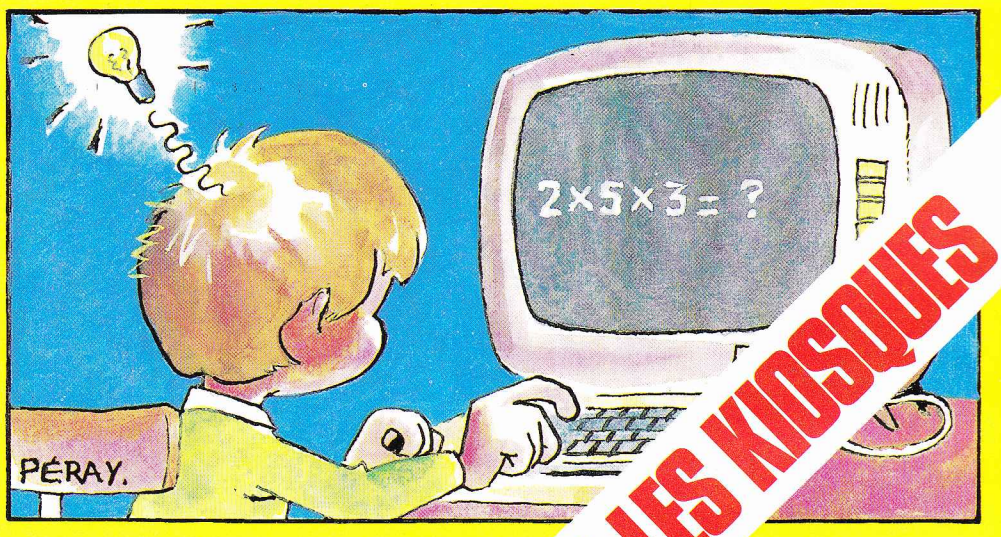
ISSN 0183 - 570 X

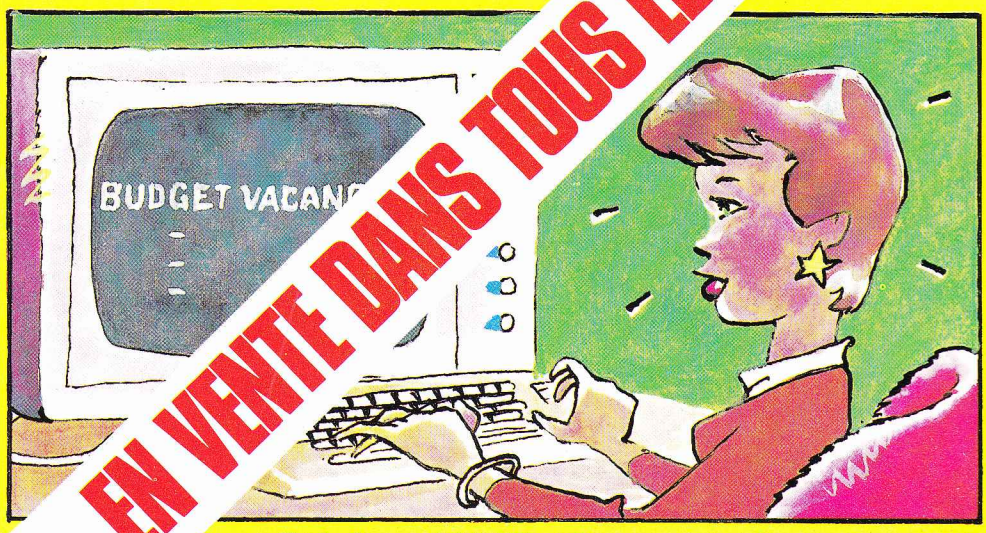
l'Ordinateur de Jeux

numéro spécial hors-série de
L'ORDINATEUR INDIVIDUEL

Jeu
Éducation
Vie pratique
votre ordinateur à la maison







Au banc d'essai :
10 ordinateurs familiaux

Noël :
les jeux dont vous rêvez

Apprendre en s'amusant :
le b. a. ba du Basic
le mastermind codeur

Comment gagner ?
les astuces des champions

Testés pour vous :
les jeux vidéo

n° 43 bis
décembre 82
15 FF

M 2946 - 43 bis - 15 F

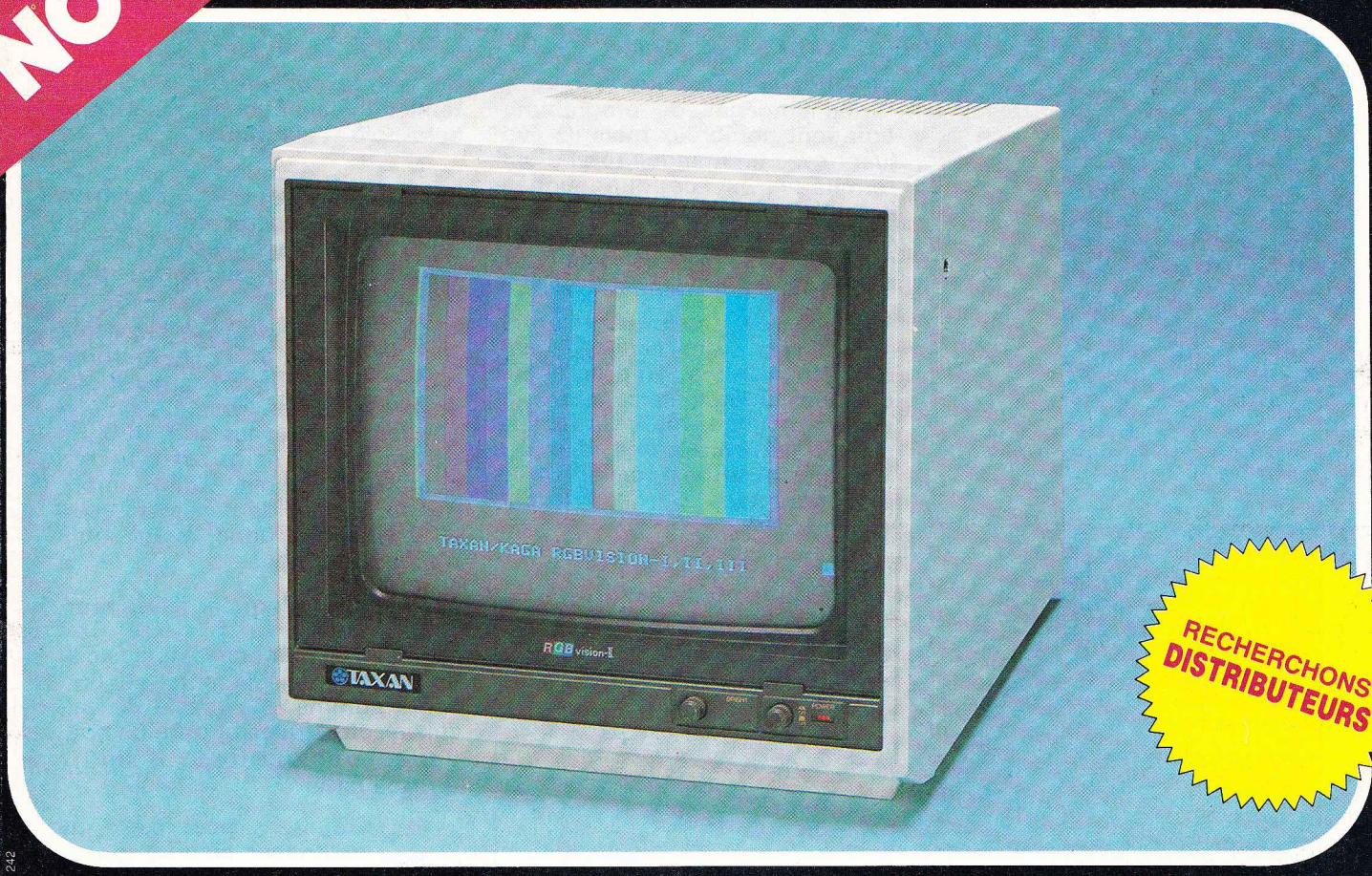
EN VENTE DANS TOUS LES KIOSQUES

Belgique 120 FB - Canada 3 $C - Suisse 5 FS

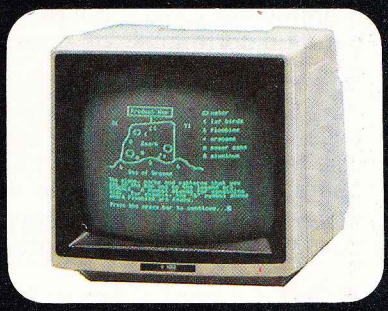
NOUVEAU

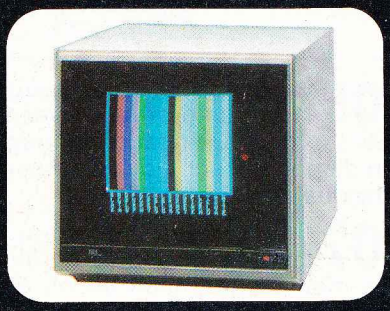
TAXAN/KAGA

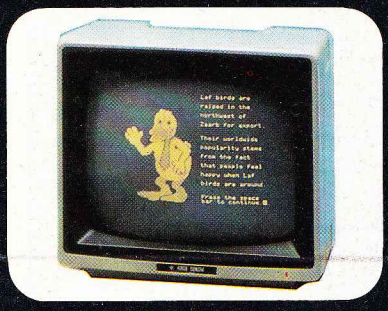
MONITEUR COULEUR 12" RGB VISION



RECHERCHONS DISTRIBUTEURS







CARACTÉRISTIQUES :

- Le moniteur idéal pour tout mini ou micro- ordinateur avec entrée RGB.
- Totalement compatible avec les ordinateurs individuels Apple III et IBM sans aucune interface complémentaire.
- Cartes interfaces « RGB » II disponibles pour compatibilité Apple II.

PERFORMANCE ET QUALITÉ

Une gamme complète de moniteurs couleur et monochrome à des **PRIX ATTRACTIFS**

Moniteurs 12"	Prix HT *
Couleur	
RGB Vision I	2.690,00 F
RGB Vision II	3.480,00 F
Carte RGB pour Apple II	590,00 F
Monochrome	
Écran vert	1.090,00 F
Écran ambre	1.250,00 F
Écran noir et blanc	1.090,00 F

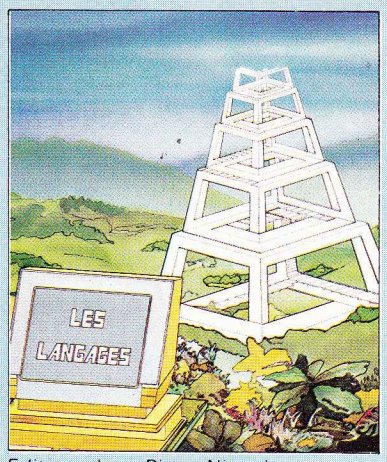
* prix janvier 1983

ERN PÉRIPHERIQUES ET SYSTÈMES

237, rue Fourny - Z.A. de BUC
78530 BUC - Tél. : (3) 956.00.11 - Télex : 698 627 F

Référence 101 du service-lecteurs (page 69)

MEGAM 242



Editeur : Jean-Pierre Nizard.
Rédacteur en chef : Bernard Savonet.

REDACTION

Rédacteur en chef adjoint : Jean-Pierre Brunerie.
Rédaction : Martine Solirenne *(secrétaire de rédaction),* Thierry Courtois *(rédacteur),* Pierre Formé *(rédacteur),* Antoine Jennet *(rédacteur),* Christian Tortel *(rédacteur),* Michelle Aubry *(assistante).*

Conseillers techniques : Christian Boyer, Daniel-Jean David, Xavier de La Tullaye, Yves Leclerc, Alain Pinaud, Benoît Thonnart.
Correspondants : Paul F. Jeffry (Etats-Unis), Riccardo Ettore (Belgique), Philippe Gysel (Londres), Jean-Louis Marx (Japon).

PUBLICITE-VENTE ADMINISTRATION

Editeur : Jean-Pierre Nizard.
Publicité : Marie-Christine Seznec, *assistée de* Fatma Boulila.
Administration : Maryse Marti *assistée de* Floriane Geneste.
Promotion : Brigitte Millé.
Abonnements, vente au numéro : Eliane Garnier, *assistée de* Muriel Watremez.

REDACTION-VENTE PUBLICITE

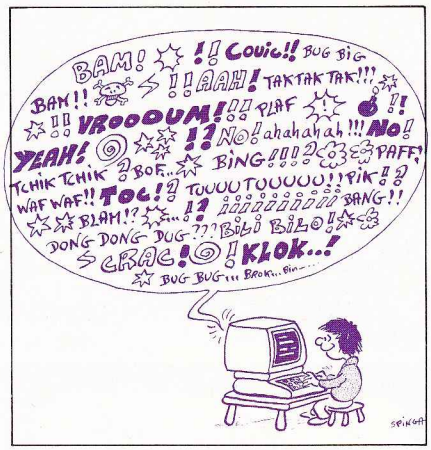
France et Etranger :
39 rue de la Grange-aux-Belles
75484 Paris Cedex 10
Tél : (01) 238 66 10
Télex : 230 589 EDITEST
Belgique :
3 avenue de la Ferme Rose
B-1180 Bruxelles
Tél : (02) 345 90 10
Suisse :
27 route du Grand-Mont
CH-1052 Le Mont-sur-Lausanne
Tél : (021) 32 61 77

Abonnements : page 68

L'ORDINATEUR INDIVIDUEL

1 En couverture, les langages : les langages de programmation sont nombreux mais ils évoluent avec un manque total d'uniformité ; c'est ce que suggère Thierry Di Sarro avec une tour de Babel inachevée. Notre dossier (pages 101-116 et 161-171) aborde ce thème rarement traité.

101 Présentation des langages : quel langage choisir ? Nous avons tenté de définir quelques critères majeurs pour résoudre cette question. Un tableau synoptique donne une idée des langages acceptés par les OI les plus courants.



105 Biographie du langage Ada : un entretien avec le créateur de ce langage, langage non disponible sur les ordinateurs individuels.

107 Interpréteurs et compilateurs : des programmes-maîtres que nous avons décortiqués pour vous ; un aspect théorique peu souvent abordé.

112 Présentation d'un système d'Enseignement assisté par ordinateur : nous avons rencontré Maurice Peuchot, qui est le concepteur d'un système original : Ego, qui fonctionne sur Apple et sur Micral.

117 La martingale sera de mise cet hiver : comment tenter de récupérer son enjeu et faire de bonnes mises grâce à un programme de jeu conçu pour Commodore.

120 Essai matériel ; le PHC-8000 de Sanyo. Cet ordinateur portatif japonais n'est pas encore vendu en France. Son prix au Japon équivaut à 8 000 FF pour une configuration très complète : écran vidéo, imprimante, lecteur de cassette. Vous le connaîtrez donc avant qu'il n'arrive chez nous.

124 Essai logiciel : Calcstar. Dans la famille des « star » voici un nouveau « calc ». Ce logiciel de Micropro se présente comme une feuille de calculs électroniques et offre de nouvelles possibilités. Il coûte 2 445 FF ttc.

129 Bourse : pour ceux qui maîtrisent bien ce qui se passe à la Bourse, voici un programme sur ZX-81, qui permet de gérer un portefeuille de titres.

133 Un Apple dans la pyramide d'un pharaon : difficile à lire et plus encore à déplacer, le livre de pierre de l'Egypte pharaonique. Quant à le reproduire... les typographes égyptologues y perdent leur... égyptien.

136 Au banc d'essai : le TO 7 de Thomson. ... Ordinateur qui se dit familial, le TO 7 permet effectivement de jouer, dessiner et faire de la musique et du Basic. Il coûte 4 500 FF ttc avec le module Basic.

Ont collaboré : Airy André, Olivier Arbey, Pierre Barnouin, France-Lise Bélair, Jean-Pierre Blanger, Serge Boisse, Xavier Bonfils, Nicole Bréaud-Pouliquen, Jeanne Bronner, Bernadette Bruneau-Couprié, Ramon Cereois Macia, Jacques Chaillot, Jean-François Challeton, Nicole Clorennec, Jacques Deconchat, Arnaud Debrevel, Myriam Fitoussi, Philippe François, Jean-Marc Geib, Henri Giacomel, Scott Gordon, François Hache, Emmanuel Halberstadt, Denis Jégonday, Roland Jost, Marc Kawam, Alain Kienlen, Philippe Kouyoumdjian, Pascal Lévêque, Claude Novakowski, Denis-Henri Petit, Joseph Pino, Michel Robin, Gilles Romano, Nicole Sitbon, Aimé St-Vryn, Alain Sanchez, M. Soulas, Pierre-Bernard Soulier, Olivier Ternam, Geneviève Ussel, Patrice Wellhoff. **Illustrations :** Eric Berthier, Thierry Di Sarro, Pascal Dupuis, Armand Krief, Alain Mangin, Alain Prigent, Jean-Christophe Renaux, Jean-Yves Sénélar, Nicolas Spinga.

n° 44 janvier 1983

sommaire

**L'OI n° 45 de février 1983 sera en kiosque dès le début février.
Avis aux amateurs !**

144 Le Rubik's cube ? Un jeu d'enfant avec votre PC-1211 : ce jeu n'est plus à présenter. Un programme sur PC-1211 vous aidera à remettre dans l'ordre les faces du cube.

148 L'OI ne fait pas le moine : lorsque le clocher pointu de l'église accueille un Sharp, l'heure de l'informatique individuelle a sonné. Voici, dans un village du sud de la France, une paroisse informatisée.

150 Jouer de la musique sur un clavier Qwerty : de nombreux OI permettent de composer et de jouer de la musique ; ce programme pour Apple 2 « imite » l'orgue.

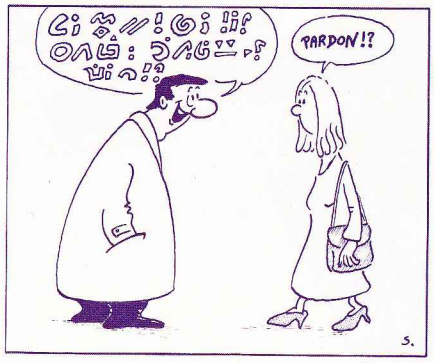
152 Essai matériel : le Bus-80 arrive en France. Fabriqué en Grande-Bretagne, il est à l'origine de nombreuses cartes et de kits qui permettent d'assembler un ordinateur.

156 Les jeux de L'OI : des jeux pour compter, dessiner et jouer au billard.

158 Essai logiciel : ZX-M. Gérer des fichiers sur cassette n'est pas toujours aisé ; le logiciel de gestion multifichiers ZX-M sera le bienvenu pour vous y aider. Conçu pour le ZX-81 de Sinclair, il coûte 215 FF ttc.

161 Evolution des langages : une étude analytique des langages (des plus anciens aux plus récents).

168 Forum des langages : introduction au langage de programmation C. Mieux vaut prendre les devants : voici la présentation d'un langage d'un type particulier qui n'est pas encore disponible sur OI mais qui le sera prochainement.



Ce numéro contient en encart un bulletin d'abonnement et de cartes-réponses paginées 67 et 68. Entre les pages 2 et 3 figurent dans les exemplaires destinés aux lecteurs de Belgique seize pages spéciales numérotées I à XVI et dans ceux pour la Suisse seize pages numérotées I à XVI.

L'Ordinateur Individuel
est une publication du
groupe **tests**

Notre publication contrôle les annonces commerciales avant insertion pour qu'elles soient parfaitement loyales. Elle suit les Recommandations du Bureau de Vérification de la Publicité. Si, malgré ces précautions, vous aviez une remarque à faire, vous nous rendriez service en écrivant au BVP, BP 45 08, 75362 PARIS CEDEX 08.

OJD DIFFUSION CONTRÔLÉE PUBLICITÉ 1982

BVP Bureau de Vérification de la Publicité.

Cependant, les Petites Annonces Classées relèvent de la seule responsabilité de notre publication. Vos remarques à leur sujet doivent donc nous être adressées directement.

Directeur de la publication
Jean-Luc Verhoye
CINQUIEME ANNEE

L'ALTERNATIVE EUROPEENNE:

BMI
BOROMÉE MULTISYSTÈME INFORMATIQUE

IMPORTATEUR EXCLUSIF

Système **compatible** avec la plus grande bibliothèque de logiciels disponibles de nos jours.

CARACTERISTIQUES TECHNIQUES STANDARD :

- microprocesseurs **6502** (compatible APPLE II *) et **Z80** (compatible CP/M *)
- **128 K**
- système ROM 2 K, plus supports pour 10 K d'utilisation ROM additionnels
- prise vidéo **RVB** et vidéo composite (PAL ou NTSC)
- **40** ou **80** colonnes
- haute résolution graphique
- sortie **parallèle**
- interface **RS-232c**
- **6 slots** compatibles **APPLE II ***

PRIX DE LANCEMENT :

14985 F. HT

tarif au 01/01/1983.

17 bis, rue Vauvenargues - 75018 Paris
Tél. : 229.19.74 - Télex : 280 150 F

* APPLE marque déposée APPLE Computer Inc.
CP/M marque déposée Digital Research Inc.

Demande de documentation à retourner

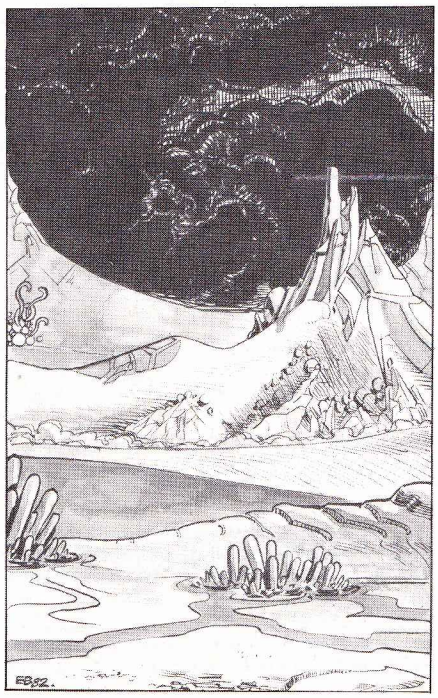
Nom .....
Adresse .....
CP .....
Tél. .....

Référence 105 du service-lecteurs (page 69)

il était une fois... des langages

Les langages en informatique sont une véritable tour de Babel. Ils sont innombrables et pourtant ils se ressemblent tous. Si par malchance (nous y reviendrons plus tard) ils ont le même nom, Basic par exemple, vous pouvez être pratiquement sûrs que deux Basic donnés seront différents et pourtant semblables. Paradoxal ? Sans doute mais, si l'on se fait l'avocat du diable, peut-être est-ce voulu afin de mettre en évidence ce qui importe le plus en informatique : aborder un problème avec logique et cohérence.

On a tendance à classer les langages en deux grandes catégories : ceux qu'on a l'impression de comprendre au premier coup d'œil – langages écrits à l'aide de mots anglais ou français rapidement et directement compréhensibles (même si ces mots peuvent être source de contresens ou de non-sens) – et les autres, langages de prime abord incompréhensibles soit parce qu'ils utilisent des symboles ma foi bien compliqués, soit parce qu'ils emploient des mots inventés de toute pièce ou des contractions de mots, contractions parfaitement *sibyllines* pour le non-initié.



Il était une fois...

Pour en revenir à GOTÒ, prononcez-le donc VATEN !

Vous n'aurez aucune peine – si vous êtes un peu angliciste – à reconnaître que, dans la première catégorie, on retrouve des langages comme Basic, Fortran, Cobol, Pascal, etc.

Prenons un terme comme Go To par exemple. Basic, Cobol et Fortran l'utilisent. Dans les trois cas il signifie « aller vers » ou « aller à ». L'équivalent en Basicois, Basic français préparé par L'OI, est Vaten – prononcez phonétiquement !

Pour en revenir à GoTo, sachez donc que vous pourrez l'utiliser indifféremment en Basic, Cobol et Fortran en ne prenant qu'une précaution : dans le cas de Fortran et Cobol, il faut laisser un espace entre Go et To. C'est tout (mais c'est nécessaire !).

Des langages « intelligibles », d'autres incompréhensibles

Un détour rapide par la deuxième catégorie évoquée précédemment nous permet d'y retrouver les divers langages machines spécialisés qui comportent diverses « joyeusetés » telles que RCL, contraction de ReCaLL, qui signifie « rappeler », INC, qui sont les trois premières lettres de INCrémenter, etc.

ALP est encore moins intelligible directement et ce langage, qui présente par ailleurs de nombreuses qualités, est « parlé » quasiment uniquement à l'aide de symboles totalement incompréhensibles pour le non-

Les OI et leurs langages

Langage / Ordinateur	Apl	Basic interprété	Basic compilé	Comal	Cobol	Forth	Fortran	Lisp	Logo	Pascal	Pilot	PL/1	CP/M	Prix
Apple 2		*	*			*	*	*	*	*			*	Jusqu'à 12 000 FF
Atari 400		*												
Atari 800		*				*				*	*			
Atom		*				*				*				
CBM 4000		*	*	*		*		*		*	*			
Commodore 64		*		*		*		*		*			*	
Dragon		*												
Goupil 2		*	*			*	*	*		*			*	
New-Brain		*												
Dai		*												
HX-20		*												
Sharp MZ 80B		*	*			*							*	
Spectrum		*												
TI-99/4 A		*							*	*				
TO-7		*												
TRS Color		*												
TRS-80-1		*	*		*		*							
TRS-80-3	*	*	*		*		*							
Vic-20		*				*					*			
Victor		*				*								
ZX-81		*												
Aim 65/40		*				*								de 12 000 à 30 000 FF
Alphatronic		*	*		*		*	*		*				
Canon TX 25		*	*		*									
CBM 8000		*	*	*		*		*		*	*		*	
Commodore 9000	*	*	*	*	*	*	*	*		*	*			
HP 85/86		*			*		*			*			*	
Nec PC 8000		*	*		*		*			*			*	
Osborne 1		*	*										*	
Philips P2000		*								*				
Sanco 8000		*											*	
Sord M23		*	*		*		*			*				
Xerox 820		*											*	
Zénith 100		*	*		*		*			*			*	
CP/M	*	*	*	*	*		*	*		*		*		

*Ada n'est pas cité, car il n'est encore disponible sur aucune machine. Lse ne figure pas non plus : on ne peut l'avoir que sur Logabax et sur Léanord. Les machines ayant CP/M en option (une * dans la ligne CP/M) ont la possibilité d'avoir les langages existant sous CP/M (une * dans la colonne CP/M).*
Les machines de plus de 30 000 FF, en principe, ont la possibilité d'accéder à tous les langages, notamment par CP/M.
Dans tous les cas, il est recommandé à l'utilisateur de vérifier, avec son distributeur, si tel langage est disponible, à quel prix, et comment il fonctionne.

initié (un initié peut-il vraiment relire facilement un programme qu'il a écrit un mois auparavant !)

Mais faisons-nous ici l'avocat de la première catégorie, les langages « intelligibles » et ce essentiellement pour le débutant.

Nous ne prétendons pas savoir quel est le langage le plus simple pour un débutant. Basic ? Pascal ? Lse ? Basicois ? ou encore Forth ? Ce qui importe, c'est que ces langages soient effectivement disponibles sur des ordinateurs *à bas prix*. Comment ne pas comprendre l'interrogation du débutant : « et si ça ne marche pas, si je n'y arrive pas, que vais-je faire de cet ordinateur ? »

Qui dit facilité d'écriture dit aussi facilité d'apprentissage

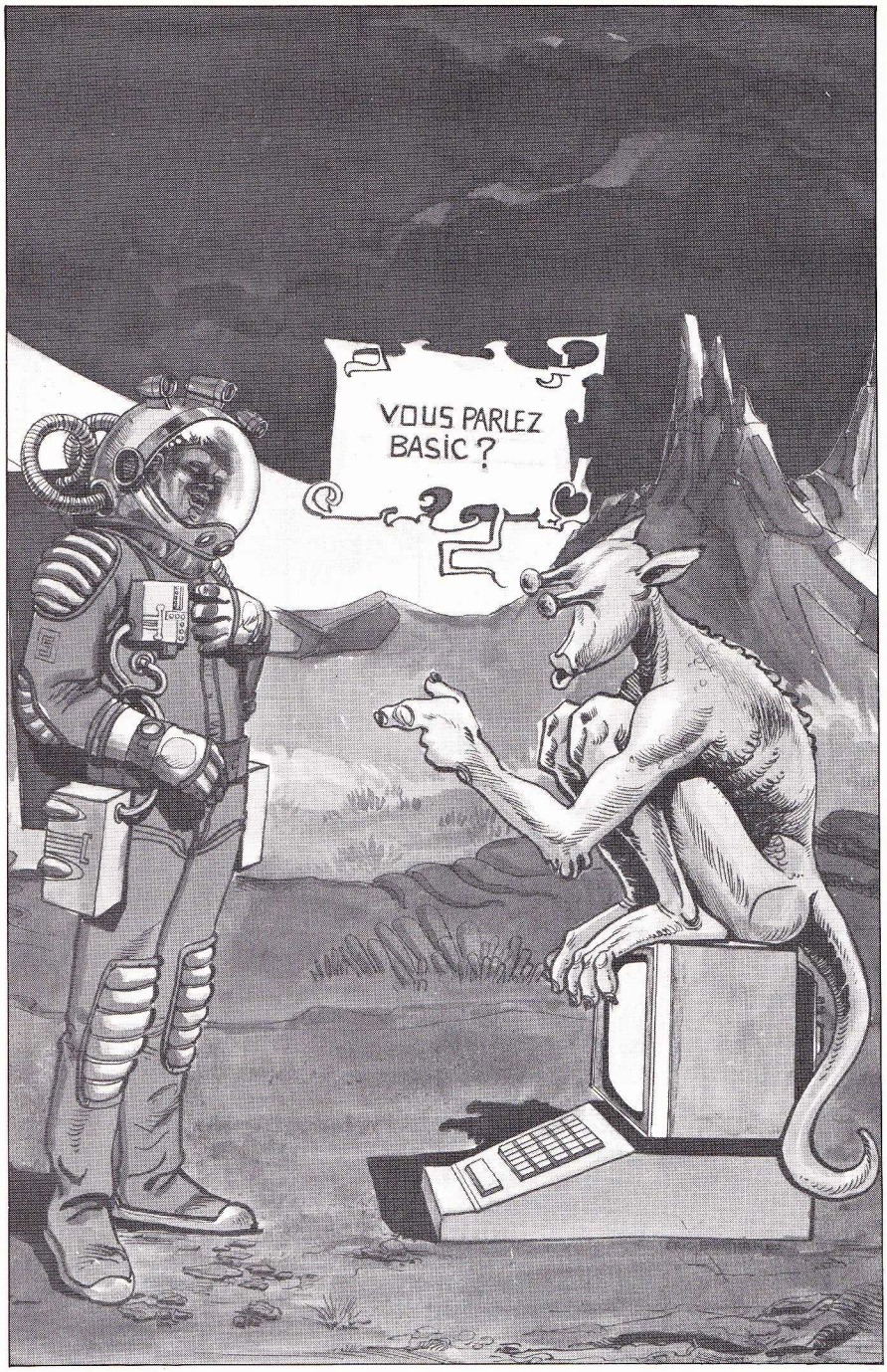
Ce qui importe ensuite, ce n'est pas tant la présence de tel ou tel langage, mais bien les *facilités d'écriture* dont on dispose : l'éditeur de textes est un point fondamental à considérer lors du choix d'un ordinateur. Est-il souple ? Est-il de type pleine page ? etc. Rien de plus désagréable que de devoir retaper entièrement une ligne. Rien de plus agréable, à l'opposé. que de pouvoir *générer automatiquement* à l'aide de l'éditeur des lignes de programmes (oui, c'est possible !)

Au niveau de la facilité d'apprentissage, Basic détient la première place ; il permet en effet de s'initier à la programmation de façon progressive, avec un niveau de difficulté croissant (car on est loin du premier Basic des années soixante, qui n'avait qu'un jeu très réduit d'instructions).

En tête Basic, suivi de près par les Basic français, Lse et Pascal

Actuellement Basic, dans ses versions les plus évoluées, est très puissant : se reporter aux nombreuses applications professionnelles développées avec ce langage. Un autre avantage est qu'un utilisateur peut s'arrêter dès qu'il sent qu'il a atteint ses limites : il peut ainsi pratiquer toute sa vie un Basic simple mais opérationnel.

Mais, pour le critère choisi, cette première place est à partager avec les Basic français, qui commencent à apparaître, et avec Lse. Ce dernier langage n'est malgré tout pas encore très diffusé, car il n'est disponible que sur quelques ordinateurs français ; mais cela ne lui enlève pas ses qualités !



.... des langages.

A un degré moindre, Pascal est aussi très abordable, à condition de bien assimiler les règles de la programmation structurée. Pour ceux qui savent déjà programmer et surtout pour eux, un autre critère peut être pris en considération : c'est la modularité des programmes. Rappelons que la programmation modulaire permet de disposer de modules répondant à plusieurs variantes d'un même programme et de composer à volonté des programmes adaptés à des applications voisines (mais non identiques).

Le programmeur doit structurer Basic de lui-même, tant bien que mal

Pascal répond très bien à cette exigence : un programme peut être constitué de blocs indépendants et pouvant être emboîtés (grâce aux instructions Function... End ou Procedure... End). Nous insistons sur cette possibilité d'appel de sous-programmes, sans laquelle il n'y a pas de pro-

L'Ordinateur Individuel a présenté dans Forum des Langages *la plupart des « grands » langages connus. Ce tableau vous permettra de retrouver les articles parus sur ce sujet. Les deux articles de base présentés en tête sont des références nécessaires pour aborder le choix d'un langage.*

Thème	Titre de l'article	Référence de L'OI
Articles de base	La programmation structurée	n° 17 p. 84
	Les langages de programmation des ordinateurs	n° 26 p. 127
Ada	Parlons d'Ada	n° 27 p. 150
Apl	Un langage plein de caractères : Apl	n° 16 p. 88
Basic	Commençons avec le b.a. ba du Basic	n° 4, pp. 32-35
	La récursivité du Basic	n° 8, pp. 53-54
	Les tours de Hanoï Basic	n° 9, pp. 60-62
	Basic, Lse ou Basicois	n° 21 p. 87
Basicois	Basicois	n° 10, p. 78
	Comment fonctionnent les programmes Basicois	n° 11 pp. 71-72
	Basicois	n° 12, pp. 75-77
	Basicois sur Sharp MZ-80	n° 16 p. 92
Cobol	Cobol	n° 17 p. 89
Comal	Comal (mi-Basic, mi-Pascal)	n° 32, p. 143
Forth	Un langage puissant : Forth	n° 22, p. 93
Fortran	Le Fortran IV	n° 15 p. 91
Langage machine	Langage machine et assembleur 6502 (I)	n° 24 p. 93
	Langage machine et assembleur 6502 (II)	n° 25, p. 78
	Langage machine et assembleur Z-80 (1)	n° 18, p. 91
	Langage machine et assembleur Z-80 (2)	n° 19, p. 91
	Langage machine et assembleur Z-80 (3)	n° 20, p. 100
	Langage machine et assembleur Z-80 (4)	n° 21, p. 88
	Langage machine et assembleur Z-80 (5)	n° 22, p. 97
Lisp	Lisp	n° 9, pp. 62-68
Logo	Logo	n° 6, pp. 51-60
	Logo	n° 14, p. 72
Lse	Le langage Lse pour l'enseignement	n° 4, pp. 50-52
	Lse	n° 6, pp. 51-60
	Lse	n° 7, pp. 57-64
	Lse	n° 8, pp. 54-57
	Lse	n° 9, p. 59
	Du nouveau à propos du Lse	n° 23, p. 99
Pascal	Pascal	n° 7, pp. 53-56
	Pascal et les ordinateurs individuels	n° 13, pp. 60-67
	Le Pascal des débutants (I)	n° 25, p. 78
	Le Pascal des débutants (II)	n° 26, p. 120
	Le Pascal des débutants (III)	n° 27, p. 125
	Le Pascal des débutants (IV)	n° 28, p. 108
	Le Pascal des débutants (V)	n° 29, p. 115
PL/1	PL/1 et ses procédures	n° 30, p. 152

grammation évoluée et performante (en vitesse d'exécution et en place mémoire).

Dans ce domaine, le langage Basic est moins pratique que les langages nés après lui : ses sous-programmes ne sont pas réellement indépendants et ne partent pas de noms symboliques.

De plus, comme ce n'est pas un langage structuré, c'est au programmeur qu'incombe la tâche de le structurer (tant bien que mal) ! D'où un effort supplémentaire.

Choisissez vous-même votre langage, car c'est vous l'utilisateur

Tout aussi important est le problème de la portabilité.

Le langage que vous avez (ou aurez) choisi est-il très répandu ?

Est-il prudent d'opter pour un langage nouveau, qui vous aura séduit pour différentes raisons (structure, rapidité, originalité, etc.), si peu d'ordinateurs acceptent l'interpréteur ou le compilateur correspondant ou encore s'il n'existe qu'une bibliothèque de programmes très réduite (consulter à ce sujet l'encadré de la page 102 qui donne une bonne idée des divers langages que l'on peut utiliser sur les ordinateurs individuels les plus courants).

N'oubliez pas qu'en dehors des applications très précises comme l'enseignement, les calculs scientifiques et la robotique pour lesquelles un langage spécifique est recommandé, ce sont les langages universels très répandus qui vous procureront le plus de satisfactions.

Lorsque vous serez devenu un programmeur astucieux, vous saurez adapter sur votre système, au prix de quelques modifications, un programme Basic (par exemple) écrit dans une autre version.

Le dossier que nous vous présentons va vous permettre de voir comment on crée un langage et quelle est l'évolution des langages depuis vingt ans. N'oubliez pas toutefois que, en définitive, c'est *vous* qui devez choisir car c'est *vous* l'utilisateur final.

Jean-Pierre Brunerie
Thierry Courtois

biographie du langage Ada

Ada passe auprès des meilleurs spécialistes pour Le langage des années 1980 à 2000. Il a été élaboré par un informaticien français — Jean Ichbiah — pour le compte du Department of Defense des Etats-Unis. Comment est né Ada ? Quelles sont ses caractéristiques ? Pour le savoir, nous avons rencontré Jean Ichbiah...

L'ordinateur individuel : *Avant d'aborder plus précisément les caractéristiques d'Ada, pouvez-vous préciser comment on en vient à envisager la conception d'un langage ?*

Jean Ichbiah : « En ce qui me concerne, ce fut le résultat d'une rencontre entre deux domaines qui m'ont toujours intéressé : les langues et l'informatique. Pour les langues, j'en parle quatre et j'en comprends cinq.

Mon intérêt pour l'informatique, en revanche, n'est venu que plus tard : après des études d'ingénieur aux Ponts et Chaussées. Je devais alors entrer au ministère des Transports, mais j'ai entrepris deux années d'informatique dans une université américaine. »

Qu'est-ce qui vous a attiré dans l'informatique ?

Ce qui attire, je crois, tous les informaticiens : le goût de l'inconnu, la curiosité ? La première fois que j'ai vu des ordinateurs, je me suis demandé comment cela pouvait bien marcher... même chose quand j'ai vu une planche à voile pour la première fois. Et ça m'a donné envie d'en faire. »

Et de retour en France ?

« Après avoir intégré le ministère des Transports, j'ai demandé à être détaché à la CII. C'est à ce moment-là que j'ai commencé a travailler sur un premier langage : Simula. A cette époque, la CII rencontrait les mêmes problèmes que toute entreprise en forte expansion : tenter de gérer le quotidien tout en se donnant les moyens de s'adapter aux évolutions du futur. La direction percevait très bien cette dernière nécessité. Nous disposions donc d'une certaine autonomie. »

C'est ce moment-là que vous avez commencé à percevoir les principes mis en œuvre pour l'élaboration d'Ada ?

« C'était une première approche, mais c'est surtout en concevant le langage Lis, de 1972 à 77, qu'il m'a été possible de préciser un certain nombre de choses. Avec Lis, il s'agissait de concevoir un langage informatique débouchant sur des logiciels plus sûrs et présentant de plus grandes facilités de maintenance. »

Peut-on dire de Lis qu'il est l'ancêtre d'Ada ?

« Tout à fait. Un certain nombre de notions utilisées dans l'Ada, — je pense à la notion de visibilité et de paquetage — et déjà envisagées pour Lis, n'avaient finalement pas été retenues, surtout parce qu'à l'époque nous n'avions pas été capable de trouver les arguments pour les justifier. Cela est important, car des principes dont nous sentions confusément l'intérêt sans pouvoir le formuler précisément ont mis plusieurs années avant de pouvoir être explicités clairement. »

Après Lis, comment en êtes-vous arrivé à Ada ?

« A la lecture de Strawman (l'homme de paille) (1). C'est le premier rapport élaboré au « Department of Defense (DOD) — le ministère américain de la Défense — par le « Groupe de travail pour l'élaboration d'un langage de haut niveau ». Le DOD était — et est resté — le plus gros consommateur de logiciels au monde. Ses systèmes informatiques gèrent des problèmes extrêmement divers et utilisent en conséquence des langages adaptés à chaque cas. Un besoin d'homogénéisation s'y était fait donc sentir. Il était apparu assez rapidement que la solution se trouvait dans l'élaboration d'un langage de programmation universel. »

Que s'est-il passé à partir de Strawman ?

« En lisant Strawman, j'ai été frappé par les concordances entre mes conceptions personnelles et les objectifs définis par le rapport : portabilité, lisibilité et « maintenabilité ». J'ai donc pris contact avec le colonel Whitaker, au DOD. »

Vous avez effectué ces démarches au nom et pour le compte de la CII ou à titre personnel ?

« A cette époque, la CII avait bien grandi, changé de patron et je bénéficiais toujours d'une certaine indépendance, car les résultats commerciaux de Lis s'étaient montrés satisfaisants, mais l'intérêt d'élaborer un nouveau langage de programmation n'y semblait pas évident pour tout le monde. C'est donc à titre personnel que j'ai entrepris cette démarche. Ce n'est qu'après les premiers résultats que la CII s'intéressa de plus près à ce projet. »

Vous n'étiez pas seul, je suppose, à vous intéresser au projet du DOD ?

« Non, au départ quelque dix

(1) Le Department of Defense élabora de 1971 à 1978 cinq rapports : Strawman (l'homme de paille), Woodenman (l'homme de bois), Tinman (l'homme d'étain), Ironman (l'homme de fer) et enfin Steelman (l'homme d'acier).

sept organismes proposèrent leurs projets, mais au bout de quelques mois quatre restèrent en course. Il s'agissait des sociétés Softech, SRI International de Californie, Intermetrics et CII-HB. Ce fut ce dernier projet qui l'emporta un an plus tard. »

D'où vient le nom d'Ada ?

« Chacun des quatre projets retenus s'était vu attribuer un nom de couleur : vert, rouge, jaune et bleu. Or le DOD voulait trouver un nom avant l'adoption du projet définitif. J'avais proposé qu'on conserve son nom de couleur au langage qui serait finalement retenu. Cette idée ne souleva pas un grand enthousiasme dans les milieux militaires américains, qui ne voulaient pas courir le risque d'avoir un langage qui s'appellerait « rouge ». On chercha donc autre chose. C'était à l'époque l'année de la femme, et l'idée d'attribuer au projet un prénom féminin sembla opportune à beaucoup. On lui donna finalement le nom d'Ada en souvenir de la comtesse Ada Augusta (2). »

Pour revenir au langage Ada, quel type de comparaison peut-on établir avec Pascal ?

« Pascal est un langage incomplet. Tant dans sa version première que dans la version UCSD. Il présente néanmoins plusieurs points communs avec Ada, ne serait-ce que la notion de types de données. De plus, Wirth (l'auteur du langage Pascal) collabora avec le groupe de travail qui élabora le projet final d'Ada, et ses apports furent très constructifs. Mais Pascal est un langage non terminé. La preuve, il n'existe pas – à ma connaissance – de grands programmes écrits en Pascal. Pascal reste un langage limité à quelques applications... Ce serait un bon langage d'initiation à Ada. »

... et par rapport à C ?

« C'est un peu la même chose : à l'égard de Pascal, c'est un langage incomplet. »

... et à Logo ?

« Logo est un générateur de programme : il ne sert pas à construire des logiciels. C'est donc un tout autre domaine. Peut-être appartient-il à cette génération de langages qui succèdera à Ada ? »

Lorsqu'on lit un programme en Ada, on ne peut s'empêcher d'y relever certaines lourdeurs syntaxiques. Qu'en pensez-vous ?

« Il est vrai qu'Ada peut sembler verbeux et plus lourd que certaines instructions en Basic même. Mais c'est au bénéfice de la lisibilité des programmes. Lorsqu'on observe l'œil de quelqu'un qui lit un programme, on remarque qu'il s'arrête sur les abréviations. Les abréviations sont très utiles pour l'écriture des programmes, mais beaucoup moins pour la lecture. Je me rappelle d'une recommandation de Woodenman qui précisait : *" La facilité de lecture a une précédence absolue sur la facilité d'écriture. "* C'est dans cette optique qu'a été écrit Ada. »

Verra-t-on un jour un système d'exploitation écrit en Ada ?

« Oui, certainement. Il y en a eu en Cobol, ou en PL 1. Mais un SE en Ada n'aura pas nécessairement à utiliser la notion de « tâches » (task). Un SE est un système qui sert à protéger du parallélisme, parce que c'est un outil délicat et qu'il faut l'ôter des mains du programmateur. Le rôle d'un SE, c'est de définir un modèle de « tâche », de définir son propre type de parallélisme : c'est un « bout de langage » en parallèle. Et si un SE décide de ne pas utiliser le parallélisme, c'est son droit. Ada n'oblige pas à préciser comment on veut que le système traite le programme. »

Y aura-t-il un jour une version française d'Ada ?

« Non, ça poserait beaucoup de problèmes et on n'y gagnerait rien : il faudrait commencer par changer tous les mots réservés, ce qui est assez complexe. De plus, certains d'entre eux sont déjà des mots français (type, procédure, déclare, exception, etc.). La seule chose qui semble intéressante, c'est l'élaboration d'un manuel en français... C'est en projet. »

Verra-t-on bientôt Ada sur un ordinateur individuel ?

« Sur un huit bits ? Non, le huit bits sera mort quand les premiers compilateurs Ada sortiront. Pour les seize bits, en revanche, c'est envisageable : ils disposent de suffisamment de mémoire pour recevoir un compilateur Ada, et sont suffisamment puissants pour commencer à en apprécier les avantages. »

Vous pouvez citer des noms ?

« Non, je n'ai pas d'information là-dessus, mais on peut raisonnablement s'attendre à ce qu'IBM fasse quelque chose. Etant donné le mystère dont ils s'entourent systématiquement, il est probable que la veille de la sortie de leur compilateur, on n'en saura toujours rien. Mais durant la phase d'élaboration d'Ada, j'ai lu des articles écrits par des gens d'IBM sur Ada : à plusieurs reprises, certaines observations dont nous ne percevions pas immédiatement la portée se révélèrent par la suite fort judicieuses. Ce genre de détails m'amène à penser qu'il doit y avoir chez IBM une équipe spécialisée sur Ada. On peut donc en déduire qu'il en sortira quelque chose... »

Quel est le meilleur atout d'un langage, pour s'imposer ?

« Le principal atout d'un langage, c'est ses possibilités de communications, le nombre de ses utilisateurs. Si Ada prend l'essor que je lui souhaite, le mérite n'en revient que pour 1 % à ses concepteurs et pour 99 % au DOD. Je souhaite que l'histoire de Cobol se renouvelle sur Ada : en son temps, Cobol avait également été choisi par le DOD, qui l'avait ensuite imposé à toutes les sociétés avec lesquelles il sous-traitait. Le DOD est la seule institution qui soit suffisamment importante pour créer une communauté autour d'un langage. Même IBM qui avait essayé avec PL 1 a essuyé un échec : IBM n'est pas assez importante... »

Vous avez toujours travaillé sur et pour des gros systèmes. Mais, que pensez-vous de l'informatique individuelle ?

« C'est un mouvement très intéressant qui est souvent mal perçu par les informaticiens traditionnels. Sans doute parce qu'il s'agit d'un mouvement de vulgarisation ; mais son intérêt principal réside surtout – me semble-t-il – dans le mouvement de standardisation qui l'accompagne : on entre dans une boutique, on achète un logiciel de ceci ou de cela, on passe à la caisse et on part avec son produit sous le bras... C'est surtout ça qui me semble intéressant : l'informatique individuelle, c'est l'épicerie de l'informatique... »

Propos recueillis par Pierre Formé

(2) Lady Lovelace, qui traduisit l'œuvre du mathématicien Menabrea et l'agrémenta de ses propres commentaires sur les dissemblances de la machine analytique et de la machine à différence, fut la collaboratrice de John Babbage dont elle rassembla et transmit les écrits complétés, eux aussi, de ses annotations. De par son travail sur cet ancêtre de nos ordinateurs que fut la machine analytique, Lady Lovelace peut être considérée comme le premier programmeur de l'histoire.

interpréteurs et compilateurs les mécanismes démontés

L'amateur d'informatique individuelle trouve plus fréquemment des publications sur les circuits logiques (interfaces, circuits d'extension, etc.) que sur la réalisation de compilateurs et d'interpréteurs. Ce dernier aspect est pourtant passionnant. Le défaut majeur de ce travail est sa complexité ; pour s'en rendre compte, il suffit d'essayer d'analyser, au moyen d'un désassembleur, le fonctionnement de l'interpréteur Basic d'une MEM. Néanmoins, l'évolution des techniques facilite ces réalisations et il est probable que, dans quelques années, de nombreux passionnés se lanceront dans ce genre d'activités.

Utiliser le langage de programmation Basic sur un ordinateur individuel, c'est agréable, simple et souvent très efficace. Cela contribue au succès des petits systèmes individuels : introduction du programme sous une forme assez accessible (organisation des calculs, structures, calculs, etc.), exécution et, très fréquemment, détection des erreurs, reprise du texte au moyen de l'éditeur, relance du programme, pause et modifications.

Le programme mis au point, on est parfois déçu par la lenteur de certains traitements et, si le temps de calcul est inacceptable, il est possible de recourir à un compilateur pour transformer le programme d'origine (source) en un code plus performant, plus ou moins proche du langage machine.

L'utilisation du Basic (souvent en MEM) est très commode, parce que les instructions sont décodées et exécutées les unes à la suite des autres, par un programme appelé interpréteur. En cas d'anomalie, il est alors facile de déterminer l'endroit précis où ce programme s'est arrêté, d'afficher et éventuellement de modifier la valeur de chaque variable, et de répartir où l'on veut. C'est l'avantage d'un système interactif.

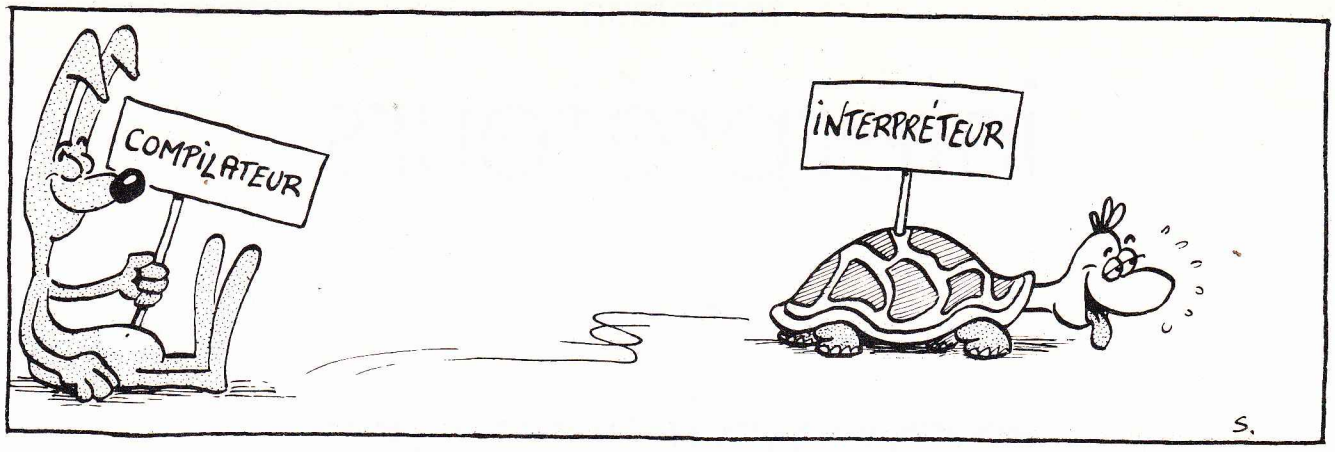
Le compilateur : du programme source au programme objet

Une autre manière de procéder, pour arriver aux mêmes résultats, consiste à traduire tout le programme source écrit en Basic (ou autre langage de haut niveau) en codes (programme objet), à traitement plus rapide. Le logiciel qui transforme en programme objet le texte source (écrit en langage symbolique) est le compilateur.

Celui-ci procède généralement en plusieurs étapes. Dans certains cas, la compilation donne un code intermédiaire qui doit être ensuite interprété par un autre programme ; ce mini-interpréteur est alors très performant puisqu'il traite un programme ultra-simplifié. Le Tiny-Pascal et même certains compilateurs Pascal procèdent de cette manière : le texte source est traduit en « P-codes » en une seule étape et ce code est traité par un simulateur de « P-machine ». Ce logiciel est choisi ici pour illustrer les techniques de compilation présentées ci-après.

Dans d'autres cas, le code résultant de la compilation (beaucoup plus proche du langage machine que le cas précédent) est traité par un programme (éditeur de liens), qui construit le programme « absolu », directement exécutable à partir de ce code relogeable et de la librairie des sous-programmes spécialisés.

On commence à trouver, dans les revues consacrées à la mini-informatique, la description complète de compilateurs et interpréteurs (Pascal, Basic, APL, etc.). Plus généralement, on constate que les amateurs entreprennent aujourd'hui des travaux de programmation qui n'étaient réservés qu'aux spécialistes ; les éléments décrits ci-après de-



vraient décider quelques passionnés à tenter l'aventure.

Un compilateur (ou interpréteur) est un programme écrit en « langage d'implantation » (langage symbolique classique, langage spécifique ou assembleur) : l'emploi de langages très évolués (comme le Pascal) simplifie considérablement le travail. Le programme source est traité par un compilateur qui donne un code exécutable. Cette transformation peut être faite en plusieurs étapes. Ainsi, une version du Tiny-Pascal a été écrite en Basic et permet donc, avec une lenteur extrême, de traduire en P-codes n'importe quel programme écrit en Tiny-Pascal. Ces instructions en langage intermédiaire sont soit exécutées par un interpréteur simplifié écrit en assembleur soit traduites en code machine par un traducteur (qui peut être écrit en Basic par exemple). Ensuite, on peut réécrire le premier compilateur en Tiny-Pascal et le compiler, toujours avec la version Basic, le recompiler avec la nouvelle version et comparer les P-codes obtenus.

Cette version en Tiny-Pascal peut être utilisée pour construire un compilateur plus évolué permettant, en procédant par améliorations successives, de se rapprocher du langage Pascal complet, ce qui constitue l'un des avantages du langage structuré.

Le Pascal ou Tiny-Pascal est également utilisable pour écrire divers compilateurs ou interpréteurs. On sait bien que les programmes compilés sont moins performants que ceux écrits directement en langage assembleur, mais il est bien plus facile d'écrire ces programmes compliqués en langage symbolique.

Pour écrire un compilateur (ou interpréteur), il faut d'abord définir rigoureusement le langage à traiter : le vocabulaire, les règles de syntaxe et de sémantique ; la théorie montre que s'il répond à certains critères caractérisant des classes de langage, le compilateur est alors construit à partir d'algorithmes d'analyse classiques.

Sans exposer la théorie des langages de programmation, il est cependant indispensable de présenter les définitions et propriétés de base pour comprendre l'élaboration des algorithmes d'analyse. Ces éléments théoriques sont assez difficiles à aborder, mais ils sont très utiles parce qu'ils simplifient considérablement la tâche.

Compilateur : définir son vocabulaire, sa syntaxe et sa symbolique

La principale difficulté est de décrire de manière finie des langages infinis. Dans ce qui suit, on considère le vocabulaire (V), qui est l'ensemble des symboles du langage. L'ensemble de toutes les séquences composées des symboles de V (et de la chaîne vide : $\varnothing$ est notée V^*. A la notation $V+$ correspond $V^* - \{\varnothing\}$. La longueur (nombre de symboles) d'une suite x est notée $|x|$.

Une grammaire (G) est définie par :

. un ensemble fini de symboles Vt appelés « terminaux » ; par exemple : IF, THEN, A, B, C, etc. ;

. un ensemble fini de symboles Vn appelés « non terminaux » ; par exemple : < début de boucle > , < fin de boucle >, < programme > , etc...

. un symbole particulier de Vn appelé symbole de départ S (ou axiome) ; par exemple : < programme > ;

. un ensemble fini (P) de règles, ou productions de la forme :
$(x, y \in V^* \mid V^* = Vt \cup Vn)$;

On peut donner quelques exemples :

1) < début de boucle > FOR < identificateur > = < départ > TO < arrivée > .

Le symbole non terminal intitulé < début de boucle > utilise un format de syntaxe : FOR < identificateur > = < départ > TO < arrivée > .

Sur cette règle viennent s'en greffer d'autres :
< départ > → < constante > ,
< identificateur > → < lettre > ,
< identificateur > → < identificateur > < lettre > ,
< identificateur > → < identificateur > < chiffre > .

On distinguera dans l'expression « début de boucle » les symboles terminaux et non terminaux :
FOR < *identificateur* > = < *départ* > **TO** < *arrivée* >

Ce qui est en gras représente le vocabulaire terminal, ce qui est en italique le vocabulaire non terminal.

2) D'une manière plus abstraite, on peut réduire les règles syntaxiques à des expressions simples. Par exemple :

S → AB
A → x
A → y
B → z
B → w

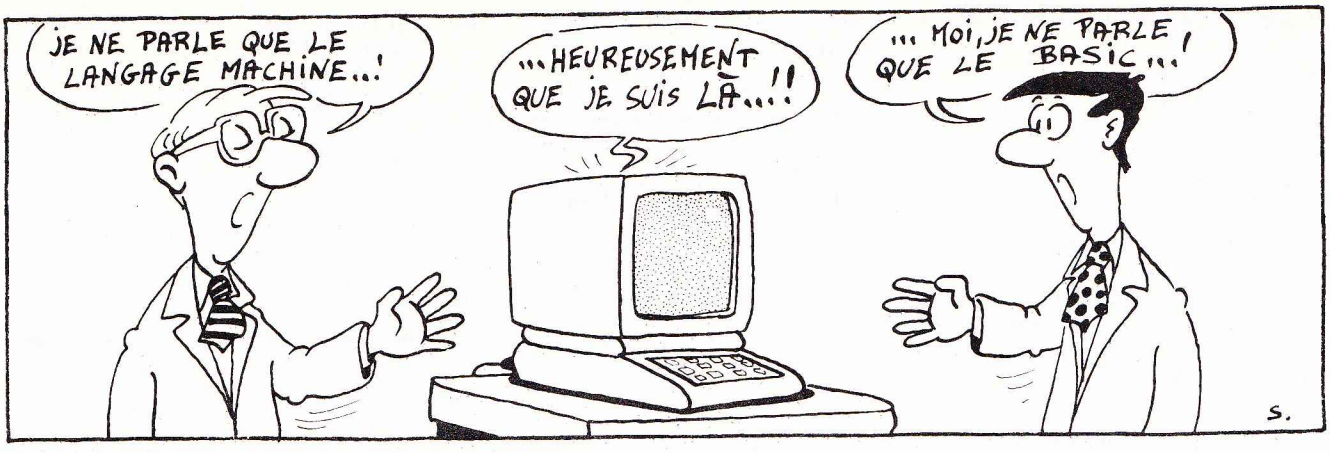
Ce minilangage ne comprend que quatre phrases générées par les règles ci-dessous :

règle x z : S → A B
A → x
B → z

règle y z : S → A B
A → y
B → z

règle x w : S → A B
A → x
B → w

règle y w : S → A B
A → y
B → w



x y n'appartient pas au langage : il n'est pas possible de le trouver dans la suite notée par convention $S \xrightarrow{+} xz$;
x est dit partie de gauche de la règle ;
y est dit partie de droite de la règle ;

Les conventions habituelles requièrent l'emploi de :
- lettres majuscules A-Z pour les éléments de Vn,
- lettres minuscules a-z pour les éléments de Vt,
- lettres italiques *a* - *z* pour les séquences de symbole.

Un langage peut présenter des symboles non terminaux à gauche :

$S \rightarrow xA$
$A \rightarrow z$
$A \rightarrow yA$

Ce langage génère une infinité de phrases de la forme :
x y y... y... y z.

Si $a \rightarrow b$ est une production et $x\,a\,z$ une séquence, alors :
$x\,a\,z \rightarrow x\,b\,z$ est une dérivation immédiate. C'est-à-dire une suite $a_0, a_1, a_2, a_3 \ldots, a_{n-1} \xrightarrow{*} a_n$. Elle est notée $a_o * a_n$, $n \geqslant 0$ (ou $a_o \rightarrow a_n$ si $n \geqslant 1$).

Toute séquence *r* dérivable à partir du symbole de départ S, c'est-à-dire $S \xrightarrow{*} r$ est appelée forme sententielle. Toute forme sententielle constituée uniquement de symboles terminaux est appelée sentence. Le langage L (G) généré par une grammaire G est l'ensemble de toutes les sentences, c'est-à-dire :

$$L(G) = \left\{ r \in V_t^* \mid S \xrightarrow{+} r \right\}$$

Chomsky a défini des classes (ou types) de langage où chacune est sous classe de la précédente.

Type 1 ou « sensible au contexte » : les productions sont limitées à la forme :

$$a\,A\,z > a\,b\,z \qquad (b \neq \varnothing)$$

Type 2 ou « non sensible au contexte » : la partie de gauche ne comporte qu'un symbole non terminal unique :

$$A \rightarrow b$$

type 3 ou grammaire régulière : les productions sont de la forme régulière à droite :

$$A \rightarrow a$$

ou bien régulière à gauche :

$$A \rightarrow B\,a$$

Dans le cas de grammaire non sensible au contexte, il y a dérivation à droite si le symbole non terminal le plus à droite est transformé de la manière ci-après. Soit une dérivation : $a_o, a_1 \ldots a_n$. A chaque étape on doit avoir :

$x\,A\,z \rightarrow x\,y\,z$ avec $z \in V^*_t$
$x_{i-1} \quad x_i$ $x \in V^*$
$A \in Vn$
$y \in V^*$

(action sur l'élément non terminal le plus à droite). L'analyse à droite LR d'une forme sententielle r est une séquence inverse des productions pour la dérivation à droite de *r*. De la même manière, on définit l'analyse à gauche LL.

Deux méthodes d'analyse : syntaxique ou descendante

Parmi les « stratégies » d'analyse syntaxique, on distingue l'analyse descendante (top down), qui va du symbole de départ S vers le texte à analyser et l'analyse ascendante (botom up), qui a la démarche inverse.

Au point de vue horizontal (si l'on peut dire), on distingue l'analyse de droite vers la gauche.

Des méthodes d'analyse ascendante utilisent des relations de précédence entre les symboles : précédence d'opérateurs, simple précédence, précédence étendue, contexte limité, etc.

Les méthodes d'analyse descendantes sont moins nombreuses et sont souvent déterministes, c'est-à-dire que l'analyseur va directement à l'analyse correcte, sans marche arrière. On y trouve les analyseurs LL. Pour éviter le retour arrière dû à un mauvais aiguillage dans l'exploration des règles (back tracking), on impose des exigences supplémentaires pour les règles de syntaxe. La grammaire obtenue est dite LL (k), ce qui signifie : analyse de la gauche (left) vers la droite, utilisant des productions à gauche (left) et en décidant l'analyse à partir de k (k) symboles situés à droite de celui qui vient d'être traité. La grammaire LL (1) est très utilisée pour les compilateurs Pascal et en particulier pour le Tiny-Pascal. Les deux règles pour éliminer le retour arrière sont les suivantes (Wirth 76).

1) Pour une production donnée :

$$A \rightarrow m_1 \mid m_2 \mid \ldots m_n ;$$

les ensembles des symboles de tête first m_i de toutes les phrases qui peuvent être générées à partir des m_i doivent être disjoints :
$\text{first}(m_i) \cap \text{first}(m_j) = \varnothing$
$\forall i \neq j$
Exemple :

$S \rightarrow A\,B$
$A \rightarrow x\,A$
$A \rightarrow y$
$B \rightarrow xB$
$B \rightarrow z$

first (A) = first (xA) = x.
first (B) = first (xB) = x.
donc ici first (A) ∩ first (B) ≠ ∅
L'intersection n'est pas vide, elle est égale à x, (éléments communs aux ensembles first (A) et first (B).

2) Lorsqu'une dérivation conduit à une chaîne vide telle que :
$\forall A \in Vn$ on ait : $A \xrightarrow{*} \varnothing$
L'ensemble des symboles initiaux doit être disjoint de l'ensemble des symboles qui suivent n'importe quelle séquence générée à partir de A :
first (A) ∩ follow (A) = ∅ .
où la fonction follow (A) est calculée en considérant pour chaque production Pi de la forme :
$X \rightarrow m\,A\,n$

la réunion de tous les S_i = first (n_i) (n_j)

Exemple : S → Ax
A → x
A → ∅
first (A) = x.
follow (A) = x.
Ici, la règle n'est pas respectée.

Ces deux règles excluent la récursivité à gauche, mais les productions concernées peuvent être facilement transformées pour l'éviter en remplaçant la récursion par une itération.

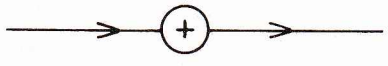
Il faut se limiter à ces quelques éléments de la théorie qui montrent l'aspect de la formalisation des notions de syntaxe des langages de programmation.

Une notation couramment retenue pour décrire les règles de syntaxe est celle utilisée pour définir l'Algol 60 et appelée Backus Nour Form (BNF), noms des auteurs du rapport de base. Les éléments syntaxiques ou mots du vocabulaire non terminal, exprimés dans la langue courante, sont délimités par les symboles < et > .

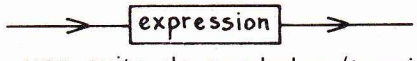
Les règles de syntaxe ou productions comprennent, comme il a déjà été dit, une partie de gauche et une partie de droite séparées par le symbole : « : : = » qui se lit « c'est » ou « est défini par ».
Exemple : < chiffre > : : = 0
< chiffre > : : = 1
< chiffre > : : = 9
ce qui peut s'écrire de manière plus condensée :
< chiffre > : : =
0|1|2|3|4|5|6|7|8|9
au moyen du symbole «|» qui signifie « ou bien ».

En outre, on fait parfois appel aux accolades « { » et « } » pour indiquer une répétition des symboles délimités 0 ou plusieurs fois.
Exemple : < chaîne de caractères > : : = ' < caractère > { < caractère > }'

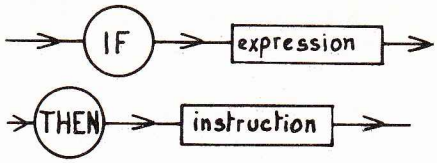
Un autre procédé consiste à utiliser des diagrammes suivant les règles ci-après ;
. les symboles terminaux sont inscrits dans des cercles, par exemple :



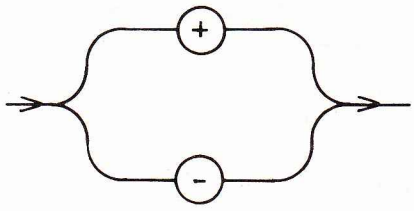
. les symboles non terminaux sont inscrits dans des rectangles, par exemple :



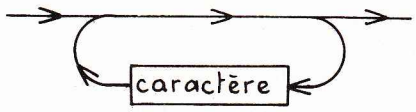
. une suite de symboles (terminaux et /ou non terminaux) est représentée par une chaîne de cercles et /ou rectangles, par exemple :



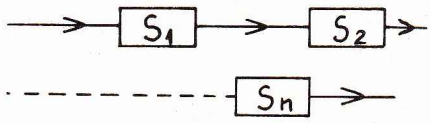
. un même élément syntaxique (non terminal) pour plusieurs symboles est représenté par des « circuits » en parallèle, par exemple : < signe > : : = + |–



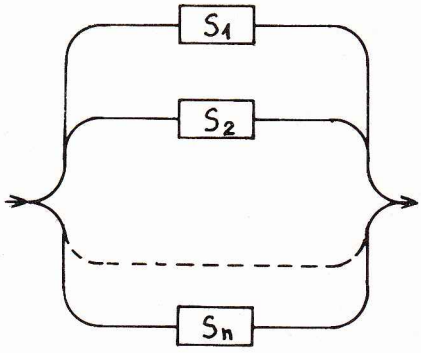
. enfin, la répétition est représentée par une boucle, par exemple : { < caractère > }



Outre l'intérêt de représenter clairement les règles de syntaxes, ces diagrammes facilitent très sérieusement l'établissement du programme d'analyse syntaxique. Si l'on écrit le compilateur en Pascal (ou Tiny-Pascal), une séquence de symboles non terminaux si :

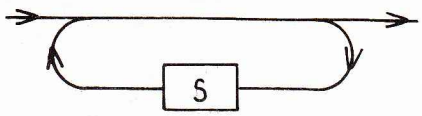


BEGIN T (S1) ; T (S2) ;....T (SN) END
où T (Si) désigne le traitement ou la procédure relative au symbole Si.
. le choix d'un élément syntaxique correspondant au diagramme suivant :



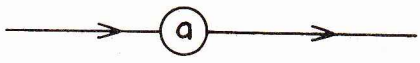
est traduit par :
CASE SYM OF
L1 : T (SI) ;
L2 : T (S2) ;
.....
LN : T (SN)
END

où L_i est le premier symbole de S_i.
. une boucle est traduite de la manière suivante :

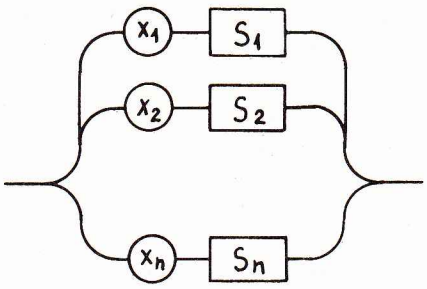


WHILE SYM = SYM = SYMS DO T (S) ;

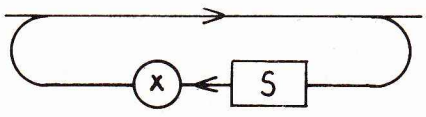
et lorsqu'on aboutit à un symbole terminal :



IF SYM = SYMA THEN GETSYM ELSE ERREUR ;
où GETSYM est la procédure pour passer au caractère suivant,
et enfin pour :

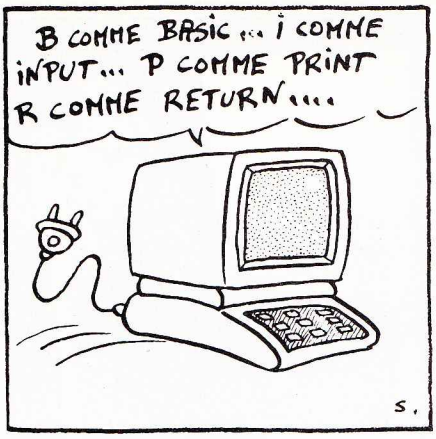


IF SYM = 'X1' THEN BEGIN GETSYM ;
T (S1) END ELSE
IF SYM = 'X2' THEN BEGIN GETSYM ;
T (S2) END ELSE
IF...
IF SYM = 'XN' THEN BEGIN GETSYM ;
T (SN) END ELSE
ERREUR ;
et



WHILE SYM = 'X' DO BEGIN GETSYM ;
T (S') END.

On voit bien maintenant l'avantage d'une théorie élaborée et




```
LIT d : T : = T+1 ; S (T) : = d                          (empilage d'une donnée)
LOAD a,l : T : = T+1 ; S (T) : = S (base (1) + a)        (chargement sauvegarde du sommet de la pile)
STO a,l : S (base (1) + a) : = S (T) ; T = T-1
JMP a : P : = a                                          (saut inconditionnel)
JPC a : IF S (T) = vrai THEN P : = a ; T : = T-1         (saut conditionnel)
OPR : T : = T-1 ; S (T) : = S (T) < OPR > S (T + 1)      (opération)
CALL a,l : S (T + 1) : = base (1) ; S (T + 2) : = B ;    (appel de sous-programme)
S (T + 3) : = P ; B : = T + 1 ; P : = a
RET : T : = B-1 ; P : = S (T + 3) ; B : = S (T + 2)      (retour)
avec d : donnée ; a : adresse relative, 1 : niveau et base (1), fonction de calcul d'adresse :
b1 : = B ; WHILE 1 > ∅ DO
          BEGIN B1 : = S (B1) ; 1 : = 1 : = L + 1 END
base : = b1
```

l'emploi du langage structuré pour écrire l'analyseur d'un compilateur (ou interpréteur). Le Tiny-Pascal est, de ce point de vue, un excellent outil. Voilà donc quelques éléments d'une méthode (Wirth 76) parmi beaucoup d'autres.

L'analyse syntaxique est la partie principale d'un compilateur mais, en amont, il y a l'analyse lexicographique avec l'établissement de tables de variables, de fonctions, de points d'entrée... et en aval, la génération et parfois l'optimisation de codes.

On continue avec le Tiny-Pascal, qui constitue un exemple simple et un outil efficace. Avant d'indiquer le principe de génération du P-code, langage intermédiaire pour le Pascal, il est bon de décrire sommairement la P-machine, qui fonctionne comme un automate à pile, c'est-à-dire avec une mémoire organisée en pile (dernier entré - premier sorti), que les habitués des calculatrices Hewlett-Packard connaissent bien. Les instructions se présentent, en quelque sorte, en notation polonaise généralisée.

Brièvement, la P-machine, calculateur hypothétique simulé par un programme, comprend quatre registres :
T : registre du sommet de la pile de travail ;
B : registre d'adresse de base pour les calculs d'adresse en cours d'exécution ;
P : registre d'adresse du programme ;
I : registre d'instruction.

La mémoire est partagée en deux zones : une zone fixe de stockage du programme en P-codes et une zone de travail : la pile des données qui contient toutes les variables de la procédure active. Et comme il n'est pas possible d'entrer dans le détail, seules les principales instructions sont décrites (encadré du haut).

A titre d'exemple, voici le code généré pour IF < expression > THEN < instruction 1 > ELSE < instruction 2 > ;
. calcul de l'expression et résultat au sommet de la pile.
JPC L2 : saut à l'instruction 2 si la valeur de l'expression est fausse.
. calcul de l'instruction 1.
JMP L3 : saut à la suite du programme.
L2 : calcul de l'instruction 2.
L3 : suite du programme.

Ce P-code généré est interprété dans le cas du Tiny-Pascal ou peut être traduit et optimisé en code machine pour être exécuté directement.

L'utilisateur du Tiny-Pascal (TRS-80 ou Apple), après examen du texte source du compilateur, voit bien comment fonctionne le compilateur et éventuellement comment le modifier.

Réaliserez-vous vos compilateurs et vos interpréteurs ?

Après cet exposé vraiment succint, mais qui malgré tout montre les difficultés, on pourrait se demander pourquoi se lancer dans la réalisation de tels programmes puisqu'ils sont disponibles.

Depuis les débuts de l'informatique, de nombreux compilateurs et interpréteurs ont été réalisés et par suite de l'évolution des matériels et des langages, il faudra en écrire encore beaucoup. Mais la tâche se simplifie grâce aux progrès de la théorie des langages, des algorithmes, qui ont fait leurs preuves, et des langages symboliques évolués.

Dans les prochaines années, de nombreux non spécialistes (ou amateurs) réaliseront de tels logiciels pour disposer de langages spécifiques, ou bien utiliseront les techniques de compilation pour élaborer des programmes très performants et l'on espère bientôt voir des publications relatives à ces travaux, qui concernent un domaine des plus complexes de l'informatique et qui sera prochainement entre les mains de beaucoup de passionnés de petits systèmes individuels.

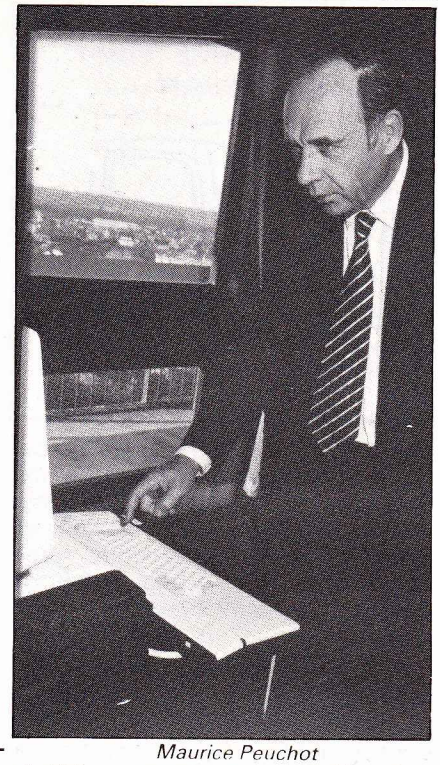
Claude Nowakowski

Bibliographie

F.R.A. Hopgood
Techniques de compilation.
Dunod, 1970
F.L. Bauer** et **J. Eickel
Compiler construction
Spinger - Verlag, 1975
N. Wirth
Algorithms + Data structure = Programs
Prentice Hall, 1976 (*)
D. Gries
Compiler construction for digital computers
John Wiley, 1971
D.W. Barron
Pascal - The language and its implementation.
John Wiley, 1981 (*)
W.M. Mckeeman, J.J. Horning** et **D.B. Wortman
A compiler generator
Prentice Hall, 1970 (*)
B.W. Liffick (Ed)
The Byte Book of Pascal
Byte magazine, 1979 (*)
P. Warme
Basex
Byte, 1979 (*)
L. Noblin
Formalisation des notions de machine et de programme
Gauthier Villars, 1969 (*)

(*) avec liste de compilateurs ou interpréteurs.

Ego
système d'enseignement assisté par ordinateur



Maurice Peuchot

L'enseignement assisté par ordinateur (EAO) commence sérieusement à faire parler de lui. Çà et là, des expériences naissent dans les lycées, les collèges et même dans des écoles primaires. De plus, et c'est un aspect que l'on oublie souvent, l'EAO concerne également les entreprises. Quoiqu'il en soit, toutes ces expériences sont encore isolées ; il n'y a pas encore d'unification entre les méthodes, les langages et les systèmes mis en œuvre. En ce qui concerne les langages et les systèmes, le problème est loin d'être résolu : c'est à qui arrivera à imposer Lse ou Basic (en anglais ou en français), Pilot ou Tudor, Plato ou... un autre système sophistiqué. Nous nous sommes entretenus avec Maurice Peuchot, qui a conçu un système d'EAO.

L'ordinateur individuel : *Maurice Peuchot, vous êtes le concepteur d'un système d'EAO, dont nous ne savons encore que peu de choses. Pourriez-vous nous retracer l'historique d'Ego, depuis sa création jusqu'à maintenant ?*

Maurice Peuchot : « L'idée est née il y a vingt ans. Au cours d'un voyage aux États-Unis, alors que je faisais des recherches sur l'enseignement assisté.

A l'époque, j'étais chez IBM-France comme ingénieur spécialisé chargé du contact avec les universités. J'assurais de plus un cours de logique mathématique à l'Institut des sciences de l'ingénieur de Nancy.

Dans ces années-là, l'EAO n'existait pas encore : ce n'était en vérité que de l'enseignement programmé avec des questions-réponses générant un vrai ou un faux ou encore des QCM (questions à choix multiple).

A mon retour des Etats-Unis, je me suis livré à une réflexion sur l'enseignement — j'étais auparavant professeur de mathématiques — afin de transmettre des connaissances, mais différemment d'un magnétophone !

Actuellement, l'EAO est conçu plutôt comme un contrôle des connaissances que comme un enseignement.

On peut dire qu'une leçon (ou un cours) comprennent trois éléments : la présentation d'une information pédagogique, l'aide à l'assimilation et enfin le contrôle des connaissances. En fait l'EAO s'occupe du premier et du troisième point..., et c'est ce à quoi il ne faut pas se limiter. »

La logique mathématique et la pédagogie sur un OI

Pourquoi fait-on de l'EAO ? Cela apporte-t-il une plus-value pédagogique ?

« Si l'on croit que l'EAO consiste à faire tourner automatiquement les pages d'un livre, en dispersant à travers le texte quelques questions de contrôle, alors il est évident que l'on peut utiliser systématiquement cet EAO-là dans tous les cas de figure. Mais quel en est l'intérêt ou la justification ?

Si cette utilisation n'est qu'un gadget, elle est peu durable, car il n'y a pas d'aide à l'assimilation des connaissances. Or, c'est cet élément qui doit compter. En général, pour aider à l'assimilation des connaissances, il faut dialoguer avec l'élève, les réponses « vrai-faux » sont insuffisantes et il n'y a pas d'enseignement sans dialogue !

C'est ainsi que je me suis mis à la recherche d'une méthode utilisant la langue naturelle ; mais, disons-le, l'analyse syntaxique, dans sa totalité, n'est pas possible : il y a trop d'exceptions et la langue française n'est pas formalisable.

Si, dans un système, on rentre les règles de syntaxe du français, une analyse syntaxique permet de vérifier plus tard, si une phrase est bien formée, c'est tout ! »

Mais qu'en est-il de son sens ?

Ici intervient la sémiotique (ou théorie générale des signes) ainsi que la sémantique (ou étude du langage considéré du point de vue du sens). Mais cela n'est pas formalisable, car chacun s'invente un code au moment où il utilise les mots.

Errare humanum est, sed ego pauper non est

J'ai donc choisi de privilégier l'analyse sémiotique et syntaxique : c'est la théorie de logique mathématique d'analyse du langage.

Nous en arrivons à la théorie des I-grammaires qui est, avec la théorie de logique mathématique déjà citée, le fondement même du système Ego. La recherche et la mise au point d'un premier noyau de mon système ont été réalisées entre 1963 et 1967, et il devint opérationnel à la fin de 1968. Entre 1969 et 1971 j'ai poursuivi mes recherches au Canada, pour le compte de l'Education nationale du Québec.

A partir de 1971, s'est effectué un aller-retour constant entre le programme mis à l'essai et le programme théorique, les critiques entraînant à chaque fois des améliorations.

Je citerai un problème, souvent rencontré en EAO, qui est celui des rôles respectifs de l'enseignant et de l'information : demander à un enseignant d'être un programmeur, c'est absurde, car l'enseignant n'est pas programmeur et le programmeur n'est pas pédagogue, sa démarche s'oriente vers l'enseignement programmé et il en résulte une incompréhension. On ne peut pas non plus les faire travailler côte à côte, car ils n'ont ni le même langage, ni les mêmes objectifs : l'un privilégie le code opération et l'autre, l'élève.

C'est pourquoi il convient de rendre l'activité informatique transparente à l'enseignant qui est chargé des cours. Nous verrons comment Ego s'emploie à satisfaire cette exigence. »

Dans quels pays votre système a-t-il été expérimenté ? Est-il utilisé couramment dans l'enseignement ? Dans quelles conditions ?

« A part le Canada, des essais ont eu lieu aux Etats-Unis, aux Pays-Bas, en Belgique et en France ; c'est pratiquement une œuvre collective qui a intégré l'expérience et les critiques d'enseignants de ces pays.

Depuis le premier noyau développé lorsque j'étais chez IBM-France, le programme a évolué en plusieurs versions successives, les dernières étant sur ordinateur individuel et dotées de graphiques et de la simulation. »

De quoi se compose votre système ? Quels sont les matériels et les langages utilisés ?

« Ego est écrit en Pascal et fonctionne depuis 1979 sur un Apple 2 ayant 64 Ko de MEV et la carte Pascal, et sur un R2E Micral 8 ou 16 bits (ce dernier ayant 256 Ko de MEV et deux disquettes de 600 Ko). »

Quelles sont les applications privilégiées du système Ego et qu'est-ce qui le distingue des autres systèmes d'EAO ?

« A vrai dire, il n'y a pas d'application privilégiée pour ce système, qui peut prendre en charge l'enseignement de n'importe quelle discipline, par exemple présenter un cours de philosophie. Ce qui ne veut pas dire que je pense que l'enseignement de la philosophie par ordinateur apporte quelque chose de plus à l'élève !

En d'autres termes, c'est l'enseignant qui doit choisir en fonction du contexte : population scolaire, niveau, nature de la discipline, telle ou telle partie du cours à privilégier afin d'aider les élèves à surmonter les obstacles rencontrés grâce à l'élaboration de véritables travaux dirigés par ordinateur. Actuellement, les applications développées sur mon système sont surtout du domaine scientifique : mathématiques, physique, cependant les exercices de grammaire française et anglaise sont abordés et je crois à un développement important pour ces disciplines.

Mais l'EAO, s'il est utilisé dans les lycées, collèges ou universités, est aussi employé pour la formation en entreprise. Dans l'entreprise, on peut tout présenter, toutefois, il faut savoir doser les différents outils pédagogiques utilisés : je ne vois pas l'intérêt de la connexion de l'audio-visuel à l'ordinateur ; vouloir tout raccorder par des fils à l'ordinateur, c'est risquer de sombrer dans le gadget. »

Mais l'utilisation d'un programme ne risque-t-elle pas d'amener l'enseignant à une paresseuse uniformité de son cours ?

Non, si – comme c'est le cas – l'EAO vient en renfort et non pour se substituer à l'enseignant lui-même : ce dernier fait son cours comme à l'habitude, et ce n'est que si à un moment donné, dans une discipline, l'élève est en difficulté, que l'EAO lui vient en aide sous la forme de modules d'aide à l'assimilation des matières. »

Un module permettant un dialogue avec l'élève

Qu'entendez-vous par « module » dans le système Ego ?

Un module est une leçon (ou une partie) dialoguée. Grâce au paramétrage, aussi bien des valeurs numériques que des mots ou expressions, cette leçon se présente sous des aspects différents selon le contact, le passé de l'élève et bien d'autres facteurs.

Ce qu'il faut, c'est que dans un ensemble de modules interconnectables, la connexion s'actualise en fonction du comportement de l'élève. Lorsqu'un élève passe d'un ensemble au suivant, cela signifie, que cet élève a parfaitement assimilé la notion enseignée dans cet ensemble.

En effet, il ne sert à rien de dire à un élève qu'il s'est trompé si la *cause* de l'erreur n'a pas été identifiée et si, partant, aucun *remède* ne lui est offert. J'irai même plus loin en affirmant qu'il n'est pas recommandé de dire à un élève qu'il s'est trompé. Il est de beaucoup préférable, lorsque l'on constate une erreur de sa part, de l'amener à découvrir lui-même qu'il a commis une erreur, en le dérivant vers certains modules

(suite p. 116)

Lorsqu'on appelle Ego, un menu principal proposera plusieurs modes de travail : le mode calculette, le mode auteur et le mode élève.

Le mode calculette permet de faire tous les calculs que l'on désire : sin, cos, tg, arctg, logarithmes népérien et décimal, modulo arrondi, troncations hautes et basses, etc. On peut ainsi introduire n'importe quelle équation (y compris des équations paramétriques) et tracer des courbes en donnant les coordonnées. Il est même possible de tracer deux courbes simultanément *(photo 1).*

Le mode élève propose la liste des modules : un module est une leçon dialoguée ; cela peut être un problème de mathématiques, de physique ou un exercice de français ou d'anglais.

Premier exemple : un automobiliste part de la ville V1 à destination de la ville V2. Combien de fois devra-t-il faire le plein du réservoir de son véhicule, sachant que le réservoir est vide au départ de la ville V1 ?

Elève : Je ne peux pas répondre.

Ego : Pourquoi ?

Elève : Il manque des éléments.

Ego : Je n'ai pas compris. (1)

Elève : Il manque des renseignements.

Ego : C'est vrai, questionnez-moi...

Elève : Il faut la distance entre les deux villes.

Ego : Distance V1, V2 : 1 268 km.

Elève : Quelle est la contenance du réservoir ?

Ego : Trente-trois litres.

Elève : Quelle est la longueur du trajet ?

Ego : J'ai déjà répondu à cette question.

Elève : Combien de litres cette voiture consomme-t-elle aux cent kilomètres ?

Ego : 8 litres.

Elève : Il fait (1 258 × 8/(100 × 33) fois le plein.

Ego : Votre résultat est arithmétiquement juste, TROP JUSTE ! Relisez l'énoncé.

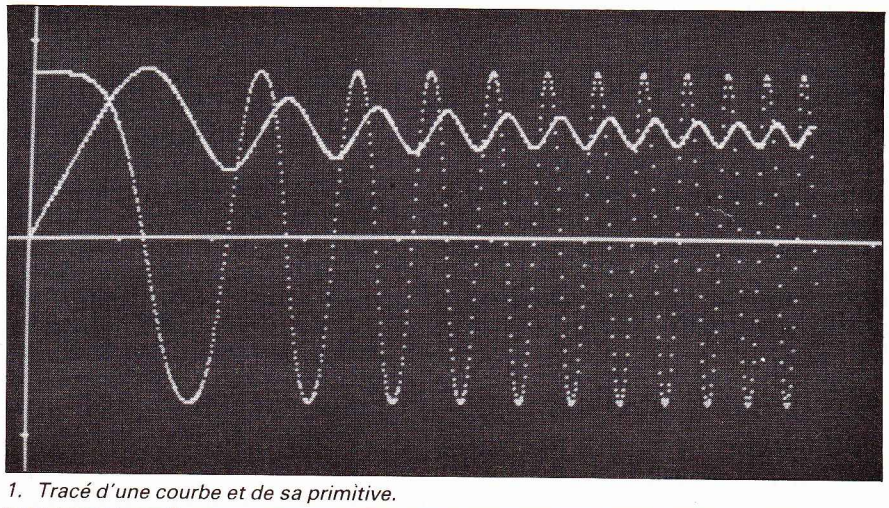
Elève : Alors THA (1 268 × 8 (100 × 33) fois. (2)

Ego : Très bien.

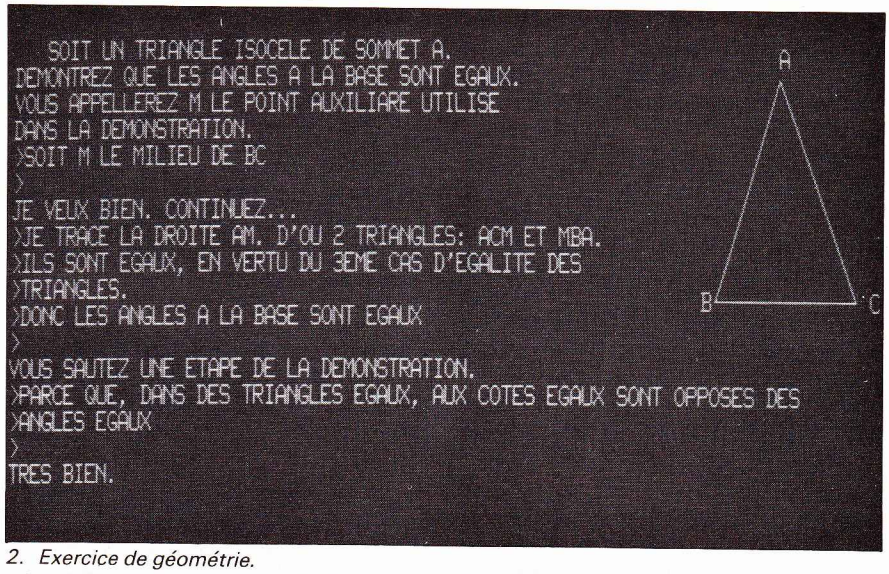
Cet exemple prouve que :

. Ego accepte et analyse des messages rédigés en langue naturelle ;

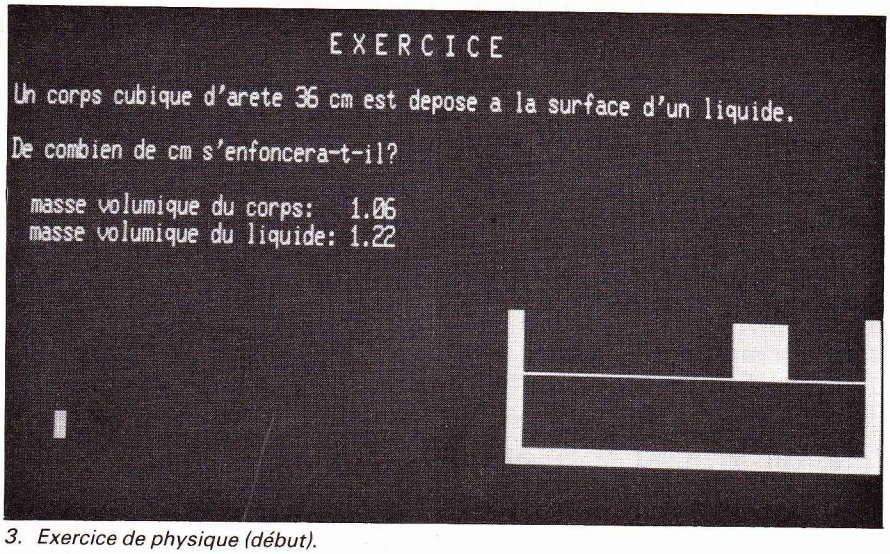
. les données changent chaque fois que cet exercice est activé ;



1. *Tracé d'une courbe et de sa primitive.*



2. *Exercice de géométrie.*



3. *Exercice de physique (début).*

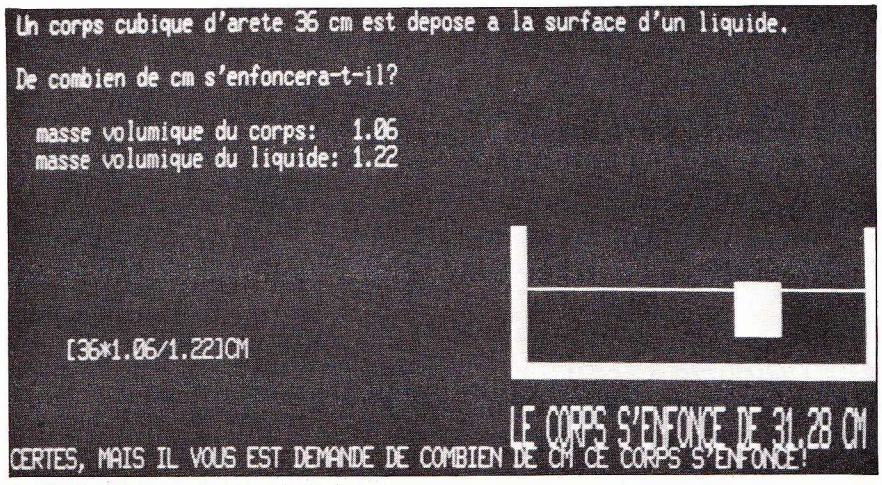
(1) « L'incompréhension » de ce message a été voulue afin d'expliquer que :
. Ego dit : « je n'ai pas compris » ;
. Ego enregistre ce message et son contexte ;
. l'auteur peut obtenir la liste des messages incompris et améliorer le module.

. il y a mémorisation du passé de l'élève ; par exemple, Ego n'accepte une réponse que si l'élève a formulé les trois questions dont les réponses complètent l'énoncé.

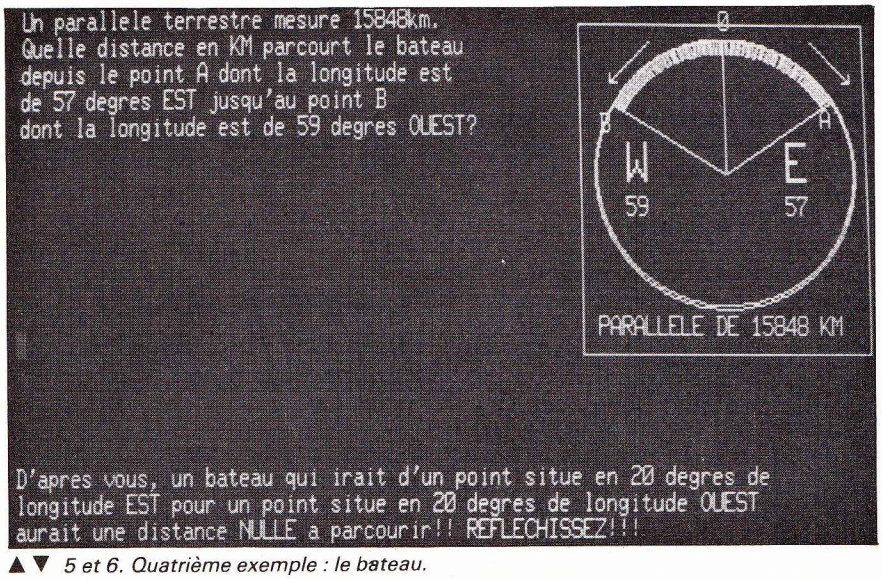
Enfin, la création de cet exercice a demandé quarante minutes de travail en tout.

Deuxième exemple : exercice de géométrie, le triangle isocèle (photos 2).

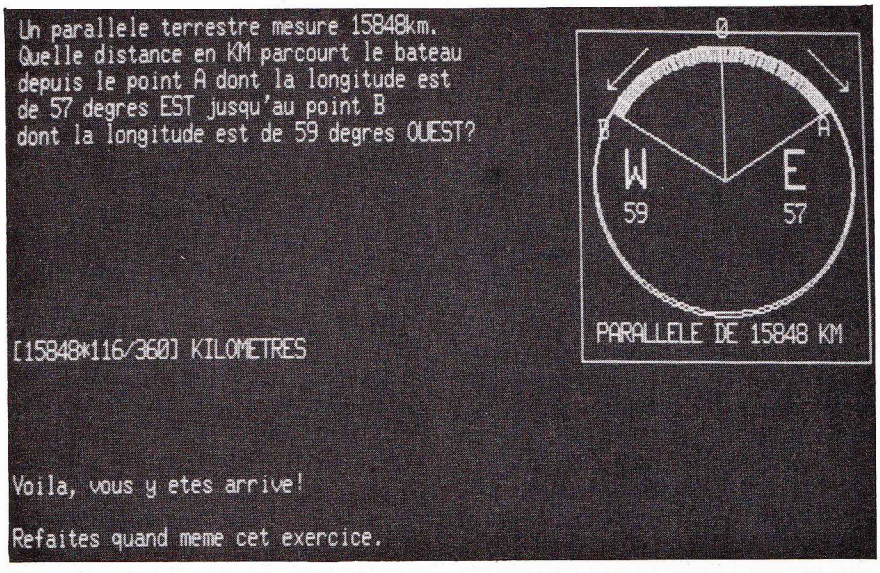
Cet exemple montre que : l'élève formule sa démonstration en langue française libre ; tout son raisonnement est analysé et une défaillance en un point quelconque de ce raisonnement est détectée et signalée. Cela a demandé une heure de travail au total.



▲ *4. Exercice de physique (suite).*



▲▼ *5 et 6. Quatrième exemple : le bateau.*



Troisième exemple : graphisme interactif (Archimède) (photos 3 et 4).

Possédant un corps cubique, on donne sa masse et celle de l'eau.

1) La masse du corps est supérieure à celle de l'eau. Voici le dialogue qui s'établit entre l'ordinateur et l'élève.

Système : Qu'advient-il du corps ?

Elève : Il flotte.

Système : Impossible.

Elève : Il coule.

Le système montre, en simulation, le corps qui descend jusqu'au fond du récipient.

2) La masse du corps est inférieure à celle de l'eau.

Elève : Il coule.

Système : Vous trouvez que ce corps coule alors que sa masse volumétrique est de 1,06 et celle du liquide de 1,22 ?

Elève : > Alors il flotte.

Système : Certes, mais de combien s'enfonce-t-il ?

L'élève entre alors une valeur et, en simulation, le corps s'enfonce selon la valeur donnée.

Quatrième exemple : Bateau (photos 5 et 6) ; cet exemple montre que :

. tout comme pour l'automobiliste, les données sont tirées aléatoirement ;

. un graphique *à l'échelle* apparaît sur l'écran ;

. l'élève est aidé et petit à petit conduit à la bonne démarche s'il éprouve des difficultés ;

. l'élève peut donner la formule conduisant au résultat plutôt que ce résultat lui-même.

Remarque générale : Bien sûr, un *vrai module* d'enseignement pourrait intégrer en les combinant toutes les possibilités précédentes et bien d'autres encore. Précisons enfin qu'une heure d'étude de l'élève implique en général en amont dix à quinze heures de création de la part de l'enseignant.

(2) Outre les opérations « classiques », Ego offre l'arrondi à l'entier le plus près (RND), inférieur (TBA) et supérieur (THA).

dans lesquels un dialogue s'engagera avec lui.

En Ego, un « cours » apparaît donc comme un ensemble de modules interconnectables dont les connexions dépendent également du passé de l'élève. Peu importe le chemin suivi, ce qui compte c'est l'assimilation de la discipline enseignée, même si quelques incursions dans des disciplines étrangères ont été nécessaires. En effet, il faut éviter le décloisonnement des disciplines afin de déboucher ainsi sur une pédagogie naturelle.

En fait, ce sont les obstacles qu'il faut traiter par ordinateur, qui apparaît ainsi comme un précepteur. Dans ce cadre, on peut dire que toutes les matières sont abordables.

Bien des points distinguent Ego des autres systèmes d'EAO : tout comme ceux-ci, Ego permet des exercices de renforcement, d'entraînement et aussi de contrôle. Mais Ego dispense totalement les enseignants d'acquérir la moindre connaissance en informatique ou en langage de programmation.

En outre, l'élève peut dialoguer librement avec l'ordinateur et ce en langue française. Ainsi, en simulation, l'élève modifie les paramètres, examine les réponses et poursuit l'exercice. »

Des exemples concrets de dialogue Ego-élève sont donnés dans l'encadré des pages 114 et 115 portant sur des exercices d'arithmétique, de géométrie et de physique.

L'ordinateur prend l'aspect d'un précepteur moderne

Ce qui vous semble important, c'est donc le dialogue de l'élève et du système ?

« Oui, un élément significatif de ce dialogue est que le système comprend toute réponse formulée en français ! De plus, il procède à une analyse du raisonnement de l'élève (et répond en conséquence), conserve les réponses de celui-ci et en tient compte pour la suite.

Plutôt que de parler d'EAO, on devrait employer le terme TDO, car ce sont de véritables travaux dirigés par ordinateurs. »

Quelles doivent être les compétences de l'enseignant ? Lui faut-il des connaissances supplémentaires ou de nouvelles habitudes ?

« L'enseignant, nous l'avons dit, ne doit pas être tributaire de l'informatique. Il crée ses cours, les expérimente et les modifie sans faire appel à un quelconque langage de programmation.

Avec mon système, il doit préparer son cours dans ce que l'on appelle le mode auteur et utilise, pour ce faire, des symboles assimilables en une journée. Je considère qu'il faut environ huit jours pour maîtriser parfaitement le mode auteur. Cela ressemble un petit peu à de la programmation – au sens de codage – mais en bien plus simple. La seule connaissance supplémentaire est la dactylographie... avec deux doigts ! »

Une nouvelle forme d'enseignement... des exercices dialogués

Combien y a-t-il de symboles dans le mode auteur ?

« Environ une dizaine et une dizaine de plus pour les graphiques. Je précise que ces symboles sont des symboles abréviateurs remplaçant des phrases entières. A la limite, on pourrait fort bien s'en passer.

Bien que le nombre de ces symboles soit très réduit, il est possible de développer en Ego des exercices dialogués très élaborés : dialogue en langue naturelle, paramétrage des données, graphiques interactifs, itératifs et déplaçables. Quant aux nouvelles habitudes, certes, il y en a ! Mais ces habitudes sont essentiellement d'ordre pédagogique et s'acquièrent naturellement avec l'expérience.

Pour donner un exemple, il faut entre une demi-heure et une heure et demie pour constituer un module Ego, en mode auteur ; si l'on compare avec un exercice présenté selon la méthode classique, il faudra à l'enseignant plus de deux heures pour corriger les copies et beaucoup plus pour l'EAO classique ».

De plus, Ego permet d'obtenir un « feed back » et donc une amélioration permanente de la méthode d'enseignement, renforçant ainsi le fait que l'ordinateur individuel (je considère en effet que l'EAO est plus adaptée à ces machines qu'aux gros systèmes) peut libérer l'enseignant de certaines tâches et lui donner du temps libre pour faire, par exemple, travailler les élèves par petits groupes. »

Ego, un système perpétuellement en évolution

Quelles sont, à l'heure actuelle, les limites de votre système ?

« On peut considérer plusieurs aspects distincts. Tout d'abord il y a des limites techniques liées à la structure des ordinateurs employés ainsi qu'au langage, l'élève ne peut pas rédiger un message de plus de deux cent cinquante-cinq caractères, mais c'est une limite de Pascal et non du système. De toutes façons, dans la pratique cela compte peu.

Plus important est le problème de la méthode d'analyse de la langue naturelle, qui n'est pas entièrement résolu : certes, l'analyse syntaxique permet la prise en compte d'un grand nombre de mots, mais il faut rester dans des limites raisonnables, c'est-à-dire travailler dans un contexte précis.

Des limitations dues à l'imagination des utilisateurs

Dans un cas extrême, toute incohérence serait traitée comme un rejet et non plus par une analyse. Pour la langue française, les résultats obtenus sont plus que satisfaisants, mais il faut ajouter que plus une langue est structurée, plus la théorie des l-grammaires est efficace ; de plus, les limitations seront souvent dues à l'imagination des utilisateurs.

J'ajouterai que le système Ego, s'il est aujourd'hui opérationnel, n'est pas pour autant achevé.

Bien entendu, je poursuis ces recherches et je peux dire qu'un important perfectionnement de l'analyse du raisonnement est pour bientôt.

Propos recueillis par Thierry Courtois

la martingale sera de mise cet hiver

La martingale ? Non, pas cette courroie qui empêche un cheval de donner de la tête mais ce « truc » que les joueurs avertis cherchent toute leur vie dans l'espoir d'un bénéfice important. Elle peut être à votre portée, sur un CBM.

La simplicité de la martingale proposée permettra, même au débutant, de l'utiliser et de réaliser des gains considérables. Terminées les parties à armes inégales avec le grand frère plus expérimenté qui emploie des algorithmes compliqués. Nous verrons bien comment il se comportera face à notre nouveau « truc ».

Mais que recouvre la martingale ? Au sens le plus simple du terme cela consiste, quand on a perdu sur un coup, à miser le double sur le coup suivant pour récupérer son enjeu. Si l'on perd encore on double à nouveau la mise et ainsi de suite.

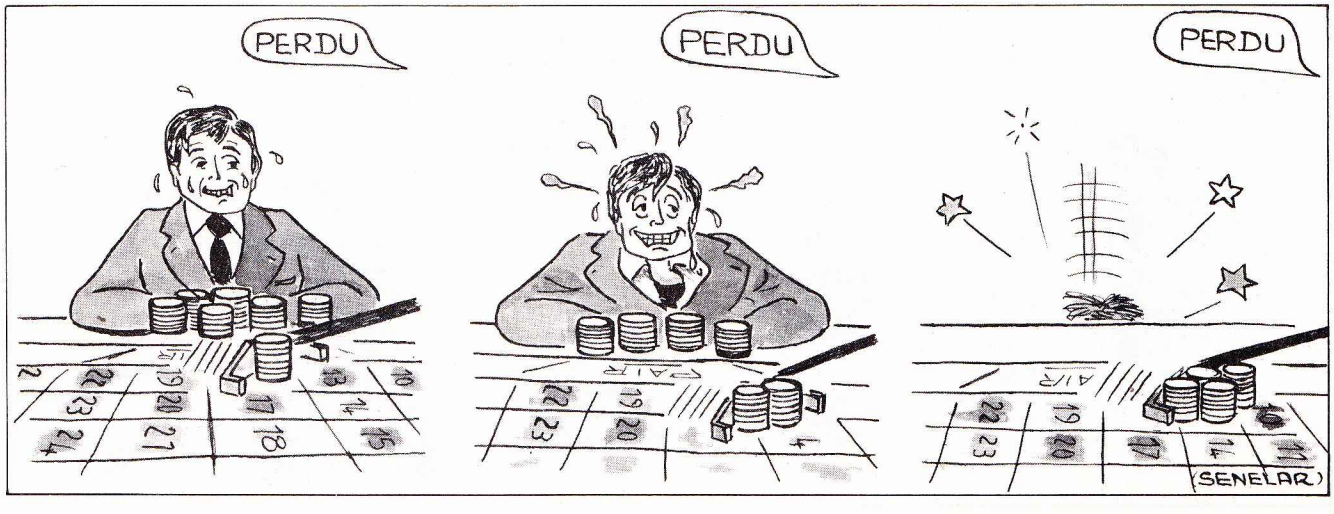
Ce procédé comporte, bien entendu, quelques petites restrictions car il n'est praticable, sans transposition, que pour les jeux binaires tels que pile ou face, la roulette (rouge/noir, pair/impair, passe/manque), etc.

Le programme « Tableau d'exploitation d'une martingale » éclaire le procédé utilisé. La première colonne (N) indique le numéro du coup, la deuxième (EN) l'enjeu, la troisième (PT) la perte totale, la quatrième (GC) le gain sur le coup et enfin, la cinquième colonne (GT) le gain total.

Les trois premières colonnes envisagent donc ce qui se passe si vous perdez à tous les coups ; les deux dernières analysent ce qui se produit si, dans cette série noire, un gain est réalisé.

Vous avez perdu ? Misez le double sur le coup suivant

Faites tourner ce programme : vous obtenez un tableau qui s'arrête au 22e coup, ce qui est amplement suffisant pour notre étude, d'autant qu'une série de



Programme du tableau d'exploitation d'une martingale

```
 1 REM * TABLEAU D'EXPLOITATION D'UNE MARTINGALE *
 2 REM AUTEUR : MICHEL ROBIN
 3 REM COPYRIGHT L'ORDINATEUR INDIVIDUEL ET L'AUTEUR
 4 REM************************************************
 10 EN=1
 20 PRINT" N    EN          PT          GC          GT"
 30 PRINT
 40 IF N<1 THEN 60
 50 EN=EN*2
 60 N=N+1
 70 PT=PT+EN
 80 GC=EN*2
 90 GT=GC-PT
 100 PRINT N;TAB(4);EN;TAB(14);PT;TAB(25);GC;TAB(36);GT
 110 IF N=22 THEN 110
 120 GOTO 40
READY.
```

Programme de la simulation d'une martingale au pile ou face

```
 1 REM * SIMULATION D'UNE MARTINGALE AU PILE OU FACE *
 2 REM AUTEUR : MICHEL ROBIN
 3 REM COPYRIGHT L'ORDINATEUR INDIVIDUEL ET L'AUTEUR
 4 REM************************************************
 10 REM ## TIRAGE P/F ##
 20 PRINT"■"
 30 N=1
 40 X=RND(1)
 50 Z=Z+1
 60 IF X>=.5 THEN 200
 100 :
 110 REM ## TRAITEMENT PILE ##
 120 CB=CB+N:A$="P"
 130 PRINT Z,A$,N,CB
 140 GOTO 30
 200 :
 210 REM ## TRAITEMENT FACE ##
 220 CB=CB-N:A$="F"
 230 PRINT Z,A$,N,CB
 240 N=N*2
 250 GOTO 40
READY.
```

22 coups reste un phénomène assez rare.

Vous pouvez, bien entendu, continuer beaucoup plus loin en modifiant la ligne 110, mais votre écran sera alors trop petit et vous verrez très vite apparaître le message d'erreur « OVERFLOW » qui indique un dépassement de capacité. La ligne 110 arrête la boucle mais évite aussi qu'apparaisse sur l'écran un BREAK très inesthétique.

Vous pouvez aussi doubler la mise au troisième coup

Le tableau met en évidence ce qui se produit au nième coup gagnant si vous avez perdu les n-1 coups précédents. Le résultat est dans la dernière colonne de ce tableau : il marque toujours un gain d'une mise. Théoriquement, le procédé est infaillible !

Une autre démarche, à peine plus compliquée et très intéressante, consiste à ne doubler la mise qu'au troisième coup, soit : 1, 1, 2, 4,..., etc. Cette méthode a l'énorme avantage de modérer une progression trop rapide et le désavantage minime d'égaliser les pertes et les gains, au lieu de laisser en fin de jeu un bénéfice d'une mise.

Vous avez là un procédé infaillible (théoriquement)

Pour tester ce procédé, il suffit de modifier la ligne 40 du premier programme en remplaçant le chiffre 1 par le chiffre 2, ce qui nous donne : 40 IF N < 2 THEN 60.

Lors de l'exécution, on cons-

Programme de représentation graphique des résultats

```
 1 REM * REPRESENTATION GRAPHIQUE DES RESULTATS *
 2 REM AUTEUR : MICHEL ROBIN
 3 REM COPYRIGHT L'ORDINATEUR INDIVIDUEL ET L'AUTEUR
 4 REM*************************************************
 10 REM ## TIRAGE P/F ##
 20 PRINT"■"
 30 N=1
 40 X=RND(1)
 50 Z=Z+1
 60 IF X>=.5 THEN 200
 100 :
 110 REM ## TRAITEMENT PILE ##
 120 CB=CB+N:A$="P"
 130 GOSUB 500
 140 GOTO 30
 200 :
 210 REM ## TRAITEMENT FACE ##
 220 CB=CB-N:A$="F"
 230 GOSUB 500
 240 N=N*2
 250 GOTO 40
 500 :
 510 REM ## DECOMPOSITION DECIMALE DE CB ##
 520 CA=ABS(CB)
 530 M=INT(CA/1000)
 540 C=INT((CA-1000*M)/100)
 550 D=INT((CA-1000*M-C*100)/10)
 560 U=CA-M*1000-C*100-D*10
 600 :
 610 REM ## HISTOGRAMME CB ##
 620 PRINT A$""Z
 630 PRINT"■■■■■■■■";
 640 :
 650 FOR S=0 TO M
 660 :  IF S>0 AND CB>0 THEN PRINT "M";
 670 :  IF S>0 AND CB<0 THEN PRINT "■M";
 680 NEXT S
 700 FOR T=0 TO C
 710 :  IF T>0 AND CB>0 THEN PRINT "C";
 720 :  IF T>0 AND CB<0 THEN PRINT "■C";
 730 NEXT T
 800 FOR V=0 TO D
 810 :  IF V>0 AND CB>0 THEN PRINT "D";
 820 :  IF V>0 AND CB<0 THEN PRINT "■D";
 830 NEXT V
 900 FOR W=0 TO U
 910 :  IF W>0 AND CB>0 THEN PRINT "U";
 920 :  IF W>0 AND CB<0 THEN PRINT "■U";
 930 NEXT W
 950 PRINT
 960 RETURN
READY.
```

tate que la dernière colonne donne systématiquement un gain total nul, c'est-à-dire qu'hormis le premier coup il ne se produit ni gains ni pertes. Bien entendu, si le premier coup est gagnant, la martingale n'est pas développée.

Vérifions cette théorie par un programme de simulation. Nous étudierons seulement le procédé ; si le second vous intéresse, il vous sera très facile d'extrapoler.

Considérons le programme « Simulation d'une martingale à pile ou face ». Le joueur mise systématiquement sur pile. Quatre colonnes vont se remplir : la première, correspondant à Z, donne le numéro du coup ; la deuxième (A$) indique si pile ou face est sorti ; la troisième précise la mise (N) ; la quatrième (CB) fait le bilan total des gains et des pertes.

Vous pourrez très bientôt vous en mettre plein les poches

Que voit-on ? La quatrième colonne va constamment se gonfler, avec quelques régressions et passages au négatif, mais inéluctablement le capital grossit. Vous pourrez vous en mettre plein les poches (fictivement, hélas !).

Il est intéressant de matérialiser visuellement l'évolution de la martingale. On remplace alors les instructions des lignes 130 et 230 par l'instruction GOSUB 500 et on tape le sous-programme « Représentation graphique des résultats », composé de deux parties.

Ne rêvez pas trop ! Ce sont de bonnes idées, mais concrètement...

La première partie, des lignes 500 à 560, décompose le capital en unités, dizaines, centaines et milliers ; la deuxième, à l'aide de boucles, reproduit les éléments de cette analyse sur une ligne horizontale. C'est-à-dire qu'un histogramme de l'évolution de vos gains et pertes va se matérialiser sous vos yeux, précédé à chaque fois du numéro du coup et de son résultat (pile ou face).

Les gains sont représentés en impression directe, les pertes en vidéo inversée. La symbolisation par des lettres n'est peut-être pas très esthétique, mais elle est commode pour les vérifications. Rien ne vous empêche, d'ailleurs, d'utiliser telle ou telle représentation graphique de votre préférence.

Il ne vous reste plus qu'à contempler ce capital qui se gonfle, se gonfle imperturbablement. Quelques passages au négatif viennent troubler la fête, mais ne retardent que d'un court instant l'inéluctable accroissement de vos gains.

Une réflexion sommaire sur ces résultats vous permettra de comprendre pourquoi les martingales fonctionnent si bien en théorie et si mal dans la pratique. Alors rêvez un peu, mais ne vous précipitez pas immédiatement vers le casino le plus proche.

Un OI vous amusera et sera moins onéreux. Il reste encore tant de programmes de simulation à réaliser pour lesquels la martingale donnera les meilleurs résultats. Alors, à vous de jouer !

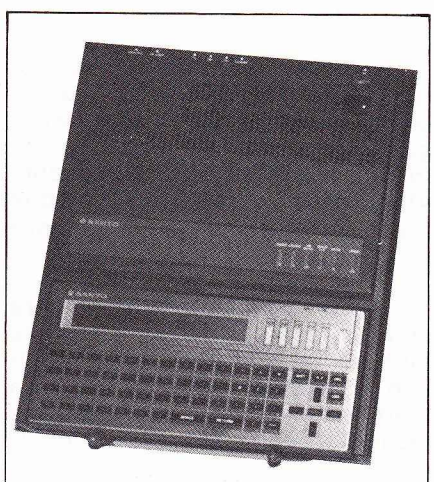
Michel Robin

essai : PHC-8000 de Sanyo

Le PHC-8000 de Sanyo a déjà une longue histoire derrière lui puisque les premiers prototypes ont été présentés à Tokyo au mois de mai 81. Dix-huit mois plus tard, j'ai enfin pu essayer cet ordinateur de poche dont on attendait beaucoup. Je l'avoue, je n'ai pas été déçu. Commercialisé uniquement au Japon pour l'instant, le système complet (attaché-case, ordinateur, interface, modem, imprimante et lecteur de cassette) coûterait 300 000 yens (8 000 FF t t c.)

Retraçons rapidement l'histoire de cet ordinateur de poche que certaines mauvaises langues avaient fini par appeler « l'ordinateur fantôme ».

Au mois de mai 81, les premiers prototypes du PHC-800 étaient présentés lors du Micro Computer Show, la plus grande exposition d'informatique individuelle au Japon (L'OI n° 42). Le PHC-800 apparaissait aussi à l'étranger (en particulier au Sicob 81). Mais la commercialisation tardait. Difficultés techniques ou amélioration des caractéristiques ?



Un ordinateur de poche dans un attaché-case.

Au Micro Computer Show 82 à Tokyo cette commercialisation était officiellement annoncée pour le PHC-800, devenu entre temps PHC-8000.

J'ai ensuite pu essayer ce nouvel ordinateur de poche (PHC sont les initiales de Personal Handheld Computer, ordinateur personnel que l'on peut tenir à la main). Sa configuration correspond à une version destinée au marché japonais et comprend le PHC-8000, l'interface PHC-8010, un lecteur de cassette et, moins classique pour un ordinateur de poche, un écran vidéo et une imprimante quatre-vingts colonnes !

Le PHC-8000 mesure 210 × 95 × 27 mm et pèse 450 grammes. Il est muni d'un écran à cristaux liquides de vingt-quatre caractères et d'un clavier Qwerty. On retrouve les caractères japonais katakana qui dédoublent le clavier Qwerty. Chaque touche peut avoir trois ou quatre significations différentes selon qu'on utilise la touche SHIFT (pour les katakanas) ou conjointement les touches SHIFT et KANA. La touche LOCK permet de verrouiller l'un de ces modes.

A côté de l'écran à cristaux liquides se trouvent cinq touches de fonction qui, nous le verrons, facilitent considérablement l'utilisation de cette machine.

Le PHC-8000 vient directement s'emboîter dans l'interface PHC-8010. Ce boîtier est assez imposant, mais donne des caractéristiques assez extraordinaires à l'ensemble PHC-8000 + PHC-8010.

N'attendons pas plus longtemps. Tournons l'interrupteur de la vidéo et branchons l'interface PHC-8010. Celle-ci met automatiquement en marche le PHC-8000.

Bizarre ! Le PHC-8000 semble réagir (l'inscription « PHC-8000 Handheld Basic » apparaît sur l'écran à cristaux liquides), mais rien sur la vidéo bien que l'interrupteur soit sur « ON ». On m'explique, car la documentation dont je dispose est rédigée en japonais (!), qu'il faut d'abord presser les deux touches « CNTL », « A » (Control A). Le message suivant est transcrit sur l'écran :

« DISPLAY IS CRT/LCD (L) »

Il faut donc choisir entre « C » pour CRT et « L » (LCD : Liquid cristal display ; écran à cristaux liquides), car le PHC-8000 affiche soit sur l'écran à cristaux liquides soit sur la vidéo. Tapons « C » RETURN et la vidéo s'allume. Le message suivant s'affiche alors :

« HAND-HELD BASIC VERSION 1.8, copyright by SANYO Elec. Co, Ltd OSAKA JAPAN »

Ce message amène deux remarques : tout d'abord il s'agit d'un Basic développé par Sanyo. Pour les ordinateurs de poche, aucune norme n'a encore été adoptée. Ce Basic est cependant très

Les trois modes d'utilisation du PHC-8000 : bloc-notes électronique, terminal de communication et ordinateur de poche parlant Basic.

proche des Basic dits classiques. De plus, la « VERSION 1.8 » est celle d'un Basic qui a déjà mûri puisqu'on en serait à la huitième amélioration de la première version (d'où, peut-être, ce retard de commercialisation ?).

Sur les ordinateurs de poche, j'ai l'habitude d'entrer le petit programme suivant lorsque j'ai besoin d'aller prendre un café (ou un thé, on est au Japon !) :

```
10 FOR I = 1 TO 10000
20 NEXT I
30 PRINT « c'est (déjà) fini ».
```

Le thé n'a pas le temps de refroidir car la « boucle est bouclée » en trente-trois secondes. Impressionnant de rapidité ce PHC-8000 !

Le Basic semble assez complet : il tient en 24 Ko de mémoire morte ! Il faut noter les deux commandes AUTO et RENUM qui permettent respectivement de numéroter et renuméroter les lignes de programmes Basic. AUTO 20, 1000 commencera à numéroter à partir de 1000 par pas de 20.

DEF FN permet de définir des fonctions avec passage de paramètres. Par exemple :

```
10 DEF FN A (X, Y) = X * 2 + Y *
   3 + 1
100 P = FN A (3, K)
```

calculera : P = 3 * 2 + K * 3 + 1.

On dispose d'un IF-THEN-ELSE, ce qui est bien pratique. L'instruction MOTOR allume le petit voyant Motor sur le boîtier du PHC-8010 et débranche la télécommande du minicassette. Qui n'a jamais été obligé d'enlever la prise de télécommande pour rembobiner la cassette sur laquelle venaient d'être sauvés programme ou données ?

Une interface RS 232C permet de communiquer avec un autre ordinateur

Les fonctions scientifiques et de traitement de chaînes de caractères ne semblent appeler aucun commentaire.

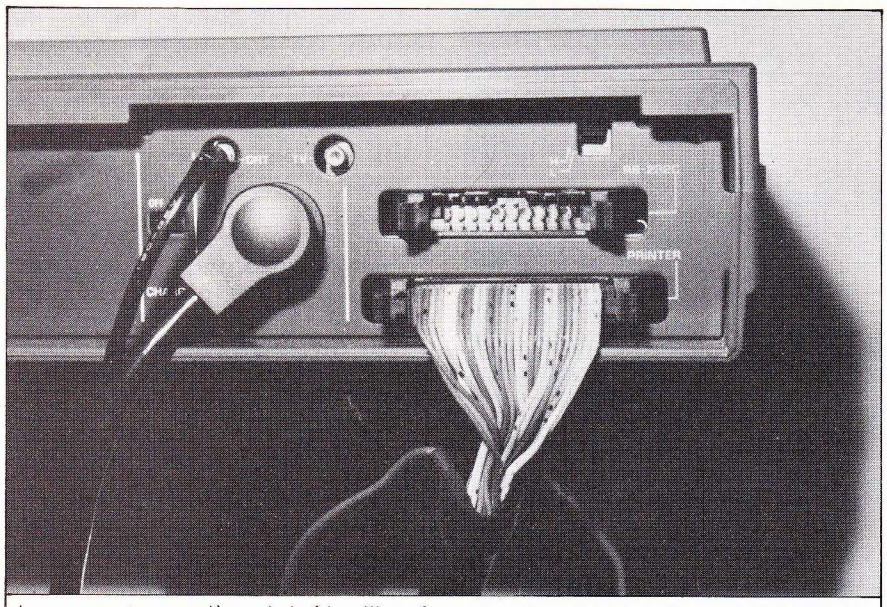
Chaque instruction peut s'abréger (Ex. : PRINT USING peut s'écrire P. US.).

N'insistons pas sur ce Basic, car il y a (bien) plus intéressant sur cette machine. Sortons du Basic en tapant « MON » pour revenir au moniteur.

En appuyant sur les touches « ↑ » et « ↓ » on peut choisir entre les modes D. T. (Data Terminal) et MEMO.

Branchons-nous d'abord sur le mode MEMO. Il permet de « prendre des notes » : on peut enregistrer des lignes de texte comportant jusqu'à 256 caractères. L'entrée se fait comme sur une machine à écrire : « RETURN » provoque un saut au début de la ligne suivante.

Il s'agit d'un petit système de traitement de texte simple d'utilisation : fonctions d'effacement, d'insertion de caractères ou de lignes ; à remarquer une fonction



Les connecteurs arrières du boîtier d'interface.

de recherche d'une chaîne de caractères : SEARCH (L'OI) va chercher ces quatre caractères dans le texte entré. Je regrette qu'il n'y ait pas de caractères français accentués sur ce clavier japonais, car le système est vraiment simple et pratique.

Appuyons sur la touche « BREAK » ; un « FINISH ! » apparaît sur l'écran. On repasse sous le moniteur.

Passons maintenant sous le mode DATA TERMINAL. C'est sous ce mode que j'ai passé la majeure partie de mon temps. Avec le PHC-8000, l'informatique passe à une autre dimension car on entre de plain-pied dans le monde de la communication.

Le PHC-8000 dispose d'une interface RS 232 C, qui lui permet de communiquer avec un autre ordinateur par une connection directe (câble) ou « indirecte » (utilisation d'un modem et d'une ligne téléphonique). Cette même interface RS 232 C pourrait également servir pour se connecter à une imprimante, mais ce n'est pas nécessaire, car l'on dispose d'une autre interface spécifique !... Il y a encore au dos de l'appareil une connexion pour la télé, une autre pour un écran vidéo et l'interface lecteur de cassette. Beaucoup d'ordinateurs de table ne disposent de ces interfaces qu'en option.

En mode DATA TERMINAL apparaît le message :

« EDIT FUNCTION KEY : – »

Ce n'est pas très explicite ; on aurait pu attendre quelques lignes d'explication, mais il ne faut pas oublier que l'affichage se fait soit sur un écran vidéo, soit sur l'écran à cristaux liquides du PHC-8000. On ne dispose dans ce cas que d'une ligne de vingt-quatre caractères.

En fait le système est extrêmement simple et pratique : c'est le moment d'utiliser les cinq touches de fonction (dédoublées par l'utilisation conjuguée de la touche SHIFT).

La touche F2 fait passer le PHC-8000 en éditeur de texte avec les mêmes possibilités d'insertion, d'effacement et de recherche de caractères que sous le mode MEMO. On peut garder plusieurs textes en mémoire, chacun d'entre eux ayant pour titre un nom de fichier. La touche « BREAK » permet de « sortir » de l'éditeur de texte et de revenir au menu (affichage de « EDIT FUNCTION KEY »).

Divers voyants du boîtier d'interface : les quatre de gauche servent à la transmission de données par modem.

Le mode « F3 » autorise le listage sur une imprimante (ici une imprimante quatre-vingts colonnes, trois fois plus grosse que le PHC-8000 et son interface) du contenu d'un des fichiers précédents.

La touche « F5 » permet de sauver sur cassette le contenu d'un fichier :

F5 TEXT SAVE
SAVE FILE : A4
SAVE FILE NAME : L-O-I
CASSETTE RECORD : [Y]
SAVE : L-O-I
OK !

Ces instructions sont à suivre si l'on désire sauver sur cassette le fichier créé sous le mode « éditeur de texte » (fichier référencé A4) et lui donner le nom « L-O-I ». Le PHC-8000 « mâche » le travail de l'opérateur et demande de vérifier que le lecteur de cassette est bien en mode d'enregistrement. Le message final « OK ! » indique que tout s'est déroulé correctement.

Il est tout aussi facile et tout aussi sûr de recharger ce fichier à partir de la cassette (utilisation de la touche F4). Il est très aisé également de transférer un fichier sur un autre (touche « SHIFT », « F2 » : instruction COPY), de concaténer (ajouter-fusionner) un fichier à un autre fichier (touches SHIFT F3 : instruction APPEND) ou d'effacer un fichier (touches SHIFT F4 : instruction CLEAR).

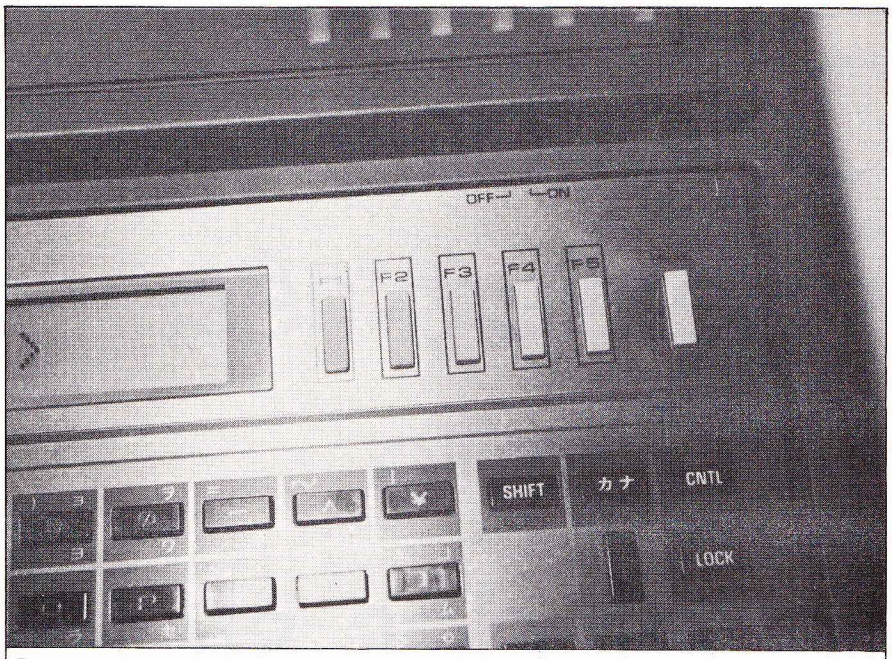
Un ensemble compact, puissant, portable et compétitif

Il faut avoir vu toutes ces possibilités avant d'attaquer le mode F1 :

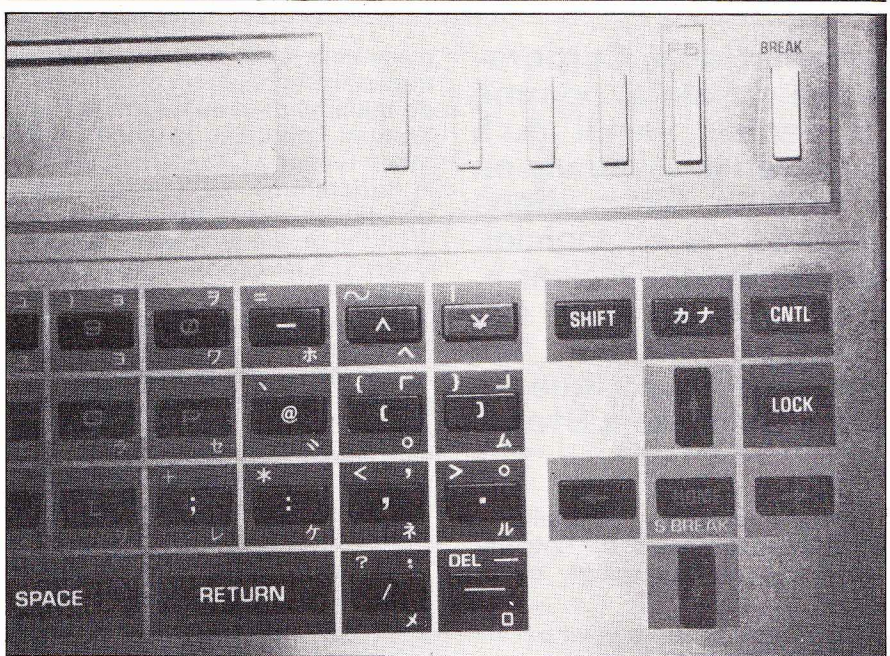
« F1 : TO COMMUNICATION MODE »

Ce mode de communication donnerait toute sa puissance au PHC-8000. C'est lui qui permet d'envoyer, par l'intermédiaire de l'interface RS 232 C, le contenu d'un fichier à un autre ordinateur. Le PHC-8000 est destiné à être transporté dans un attaché-case avec son interface PHC-8010, un modem, une petite imprimante et un lecteur de cassette. On a alors un ensemble portable et puissant pour transmettre à distance des données. Je n'ai hélas pas pu tester, lors de mon essai, ces possibilités de communication qui utiliseraient les touches de fonction pour accéder aux sous-menus :

F1 : TO EDIT MODE ; pour revenir



Des touches de fonction bien pratiques pour l'utilisateur qu'elles guident lors des menus successifs.



La partie droite du clavier permet de contrôler l'affichage sur l'écran à cristaux liquides et/ou sur un poste de télévision.

directement au menu principal.

F2 : TO ON-LINE ; message envoyé par l'interface RS 232 C.

F3 : STATUS ; pour entrer et définir tous les paramètres qui concernent la liaison RS 232 C : la vitesse de transmission (Baud Rate), la parité (Parity), la longueur d'un caractère, le bit d'arrêt (Stop Bit), le caractère qui indique la fin d'une ligne (Duplex).

F4 : pour définir les périphériques connectés à l'interface PHC-8010 : télévision couleurs, imprimante, lecteur de cassette.

La touche BREAK permet de sortir de tous ces modes et sous-modes d'utilisation. Il est possible, par ailleurs, grâce à la fonction AUTO MESSAGE (touche SHIFT + F1) de définir un certain nombre de messages envoyés avant le contenu d'un fichier. De plus, le PHC-8000 dispose également d'une instruction PROTECT pour les fichiers.

Le PHC-8000 est entièrement paramétrable, c'est-à-dire qu'on peut définir tous les facteurs d'utilisation de la machine. J'ai été très impressionné par la compacité de cet ensemble et par sa « docilité » pour un utilisateur non professionnel : on comprend le fonctionnement de cet ordinateur en un minimum de temps. L'emploi des touches de fonction facilite beaucoup l'utilisation.

PHC-8000 est géré par un processeur CMOS 8 bits (référencé NSC 800), qui tourne à 2 MHz. La mémoire morte fait 24 Ko et la mémoire vive CMOS 4 Ko. Le boîtier d'interface PHC-8010 contient 14 Ko de mémoire morte et 2 Ko de mémoire vive (celle-ci peut être étendue à 22 Ko). On peut ajouter 16 Ko de mémoire vidéo en option : le PHC-8000 afficherait alors seize couleurs différentes.

Un modem adapté à cet ensemble existe. Je l'avais vu lors de la dernière exposition de Tokyo. Sa référence est 8011. Une mini-imprimante destinée à être intégrée dans un attaché-case avec le modem, le PHC-8000 et le boîtier d'interface devrait être prochainement commercialisée au Japon.

Une nouvelle génération ? Peut-être pas, mais des caractéristiques inédites

Le prix du PHC-8000 est particulièrement compétitif puisqu'il est de 69 800 yens (1 900 FF) au Japon (seul pays où cet ensemble est actuellement commercialisé, précisons-le encore une fois).

Il est intéressant d'acheter, en même temps que le PHC-8000, le boîtier d'interface PHC-8010, qui donne à cet ensemble toute sa puissance (123 200 yens au Japon, soit environ 3 300 FF).

Le système complet (attaché-case contenant PHC-8000, PHC-8010, modem 8011, mini-imprimante et un lecteur de cassette) devrait valoir moins de 300 000 yens au Japon (8 000 FF environ). Ce système vise un créneau d'utilisateurs professionnels. Il permettra par exemple à un représentant d'envoyer en fin de journée l'ensemble de ses commandes du jour ou à l'homme d'affaires de faire parvenir rapidement un compte rendu de réunion au siège de sa société.

Faut-il voir en ce PHC-8000/PHC-8010 le précurseur d'une nouvelle génération d'ordinateurs de poche ? Peut-être, mais en tout cas ce nouvel ordinateur présente actuellement des caractéristiques inédites fort attirantes.

Jean-Louis Marx

Calcstar un des derniers-nés de la famille des calc et des star

Ce logiciel de Micropro International Corporation s'assimile à la famille des « Calc », comme les fameux Visicalc et Supercalc, ainsi qu'à T/Maker II. Il effectue sur l'écran des calculs en temps réel sous forme de tableau. Nous allons passer en revue les possibilités de ce logiciel, issu également de la grande lignée des « Star », comme Wordstar, Datastar (voir L'OI nº 41) créés par Micropo. Voyons si Calcstar, qui coûte 2 445 FF ttc, est aussi performant que les autres membres des familles « Calc » et « Star ».

Calcstar fonctionne (j'allais dire « bien sûr ») sous CP/M, système d'exploitation de disquettes (SED) donc sur la plupart des équipements possédant des disquettes et le système d'exploitation de disquettes (SED) CP/M de Digital Research avec un processeur 8080 ou Z-80. Il est livré dans un classeur blanc à lettres roses, contenant une disquette (dans mon cas de 13 cm) et une abondante documentation « in french ! ». Celle-ci, visiblement traduite d'un document anglais, a un style technique un peu sec, mais est largement suffisante pour apprendre à se servir de toutes les fonctions de Calcstar.

Le classeur comprend plusieurs parties :

. une introduction décrivant les possibilités de Calcstar, en particulier en liaison avec d'autres logiciels tels que le Basic, Wordstar et Mailmerge, Datastar, etc. ;

. la mise en œuvre, l'installation et le « lancement » de Calcstar, ainsi que la description de sa structure ;

. l'exploitation du logiciel par lui-même : on parcourt pas à pas les fonctions offertes par Calcstar.

Une documentation abondante, en français mais incomplète

A mon avis, il manque deux choses dans la documentation : un résumé des fonctions sous forme de carte, genre Visicalc ou Supercalc, et une structure plus didactique, avec un résumé de ce qui vient d'être étudié à chaque fonction ou groupe de fonctions.

Il n'y a rien de tel que l'apprentissage sur le tas ; j'attaque donc le vif du sujet en faisant une copie de la disquette originale et je place celle-ci en lieu sûr, précaution indispensable en cas d'erreur de manipulation au cours de l'utilisation. Impatient, je lance Calcstar en tapant « CS retour chariot », le programme se charge, mais rien ne se passe sur l'écran. En effet j'ai oublié qu'il fallait adapter Calcstar à une configuration particulière, qui est le Galaxy 1 de Gemini Computer, c'est-à-dire un système Z-80 avec 64 K-octets de mémoire, deux unités de disquettes 13 cm, le CP/M 2.2 comme SED, un écran de vingt-cinq lignes par quatre-vingts caractères ; je possède également une imprimante Epson MX80F/T type II.

Je lance donc le programme d'installation ; celui-ci me pose la question suivante :
PREMIERE INSTALLATION DE CALCSTAR 2 O/N

Je réponds oui et j'ai alors devant moi une liste de configurations dont aucune ne correspond à la mienne. Le programme « Install. com » me demande donc de configurer Calcstar en lui donnant des renseignements qu'il ne possédait pas dans sa banque de données : les valeurs hexadécimales pour les déplacements de curseur, l'effacement d'écran, le retour arrière, etc. Je connaissais ces données, sinon la seule ressource aurait été de consulter le revendeur de mon matériel.

La documentation présente quelques imprécisions en ce qui concerne les codes à entrer, si l'on veut directement changer les

valeurs de Termcap. Sys, le fichier qui contient les caractéristiques du système employé. Heureusement, il n'est pas nécessaire d'intervenir directement dans Termcap. Sys, le programme Instcs. Com le fait pour nous.

Par curiosité, j'ai relancé le programme d'installation et j'ai prétendu que ce n'était pas ma première installation, pour voir... Quelle ne fut pas ma surprise de voir s'afficher les options suivantes :

OPTIONS DE MENUS D'INSTALLATION CALCSTAR

A – MODIFICATION D'UNE INSTALLATION EXISTANTE
B – INSTALLATION A PARTIR D'UN FICHIER WORDSTAR

Pourquoi ne peut-on prendre les valeurs d'installation de Wordstar (que je possède) que lors d'une deuxième installation ? La logique m'échappe.

Une fois l'installation finie, je « lance » Calcstar, ce qui prend quelque temps : trois fois plus de temps que Supercalc, c'est-à-dire quatorze secondes environ.

Et là, victoire, ça marche ! C'est-à-dire que j'ai devant moi le tableau électronique en deux dimensions avec deux « tableaux de bord » : un en haut, qui m'indique comment déplacer le curseur (sauf s'ils ont été reconfigurés) et un résumé des commandes ; l'autre en bas donne un tas de renseignements : le nom du fichier sur lequel on travaille, la position du curseur sur le tableau, la position pointée par Calcstar. Il fournit aussi la direction des calculs (H-B ou G-D), le type de contenu de la cellule pointée par le curseur : texte, expression arithmétique, le contenu de la cellule et les différents messages et commandes.

Six colonnes de dix caractères par dix ou quinze lignes

Le tableau par lui-même possède six colonnes de dix caractères par dix lignes. Je découvre qu'il est possible, grâce à la commande « ;* » de changer le format de l'écran et d'obtenir quinze lignes en supprimant le tableau du haut. Je trouve que l'écran n'est pas utilisé au mieux, comparé à Supercalc, sur lequel on voit vingt et une lignes avec plus de renseignements dans la partie inférieure (sauf le nom du fichier traité, qui n'est pas affiché dans Supercalc).

Les titres COL et LIG sont affichés pour indiquer les colonnes et lignes, ce qui me paraît tout à fait superflu et gâche de la place inutilement. De même deux lignes sont tirées, l'une en haut du tableau, l'autre en bas ; des lignes de données auraient sûrement été plus utiles. Enfin, globalement, on note une mauvaise exploitation de l'écran.

Commençons à entrer des données, du texte : le message texte s'affiche et Calcstar ne demande pas de préciser que j'entre du texte, il le « sait » ! Mon texte possède plus de dix caractères, il sera tronqué.

Possibilité d'agrandissement de la colonne où se trouve le curseur

Pas de dépassement possible sur la colonne suivante si elle est vide (comme dans Supercalc). Dommage ! Je ne m'avoue pas vaincu, je regarde si je peux agrandir la colonne : c'est possible ; avec la commande « ;F », seule la colonne sur laquelle se trouve le curseur sera modifiée, contrairement à Visicalc, qui modifie toutes les colonnes.

Le traitement des erreurs est simple : il faut utiliser la touche de retour arrière, avant d'avoir entré la donnée ; après, nous n'avons pas le choix, il faut retaper toute la ligne de commande ; il n'y a pas de commande « edit », ce qui est pénible pour les dactylos à deux doigts, comme moi.

Jusqu'à présent, je n'ai pas été impressionné par les performances de Calcstar. Voyons si en poussant l'utilisation un peu plus loin, nous allons trouver des fonctions intéressantes.

Dix-neuf commandes disponibles quand on est en format dix lignes

En haut de l'écran, lorsque l'on se trouve en format « dix lignes », se trouve la liste des commandes disponibles ; elles sont au nombre de dix-neuf et sont précédées de « ; ». Ce sont :

A (uto), qui sélectionne les cellules à utilisation fréquente pour un accès rapide ;
C(opie), qui copie le contenu d'une ou de plusieurs cellules dans une ou plusieurs cellules ; les changements relatifs de numéro de cellule sont globaux et non pas sélectifs, comme dans Visicalc ou Supercalc ;
D(efface), qui efface une entrée, une colonne, une ligne ou le tableau en entier, mais pas un fichier sur disquette, comme Visicalc ou Supercalc ;
E, qui reconfigure le tableau de telle manière que la cellule pointée soit la cellule en haut et à gauche ;
F(ormat) qui change la largeur des colonnes (sélectivement), la précision des chiffres entrés et les cellules en mode auto ;

L'ORDINATEUR INDIVIDUEL

Essai logiciel : Calcstar

Nous avons aimé :	Qualité de la documentation	Facilité d'utilisation	Performance	A l'usage (confort, sécurité, rapidité)
passionnément	☐	☐	☐	☐
beaucoup	☐	●	☐	☐
un peu	●	☐	●	●
pas du tout	☐	☐	☐	☐

Matériel nécessaire

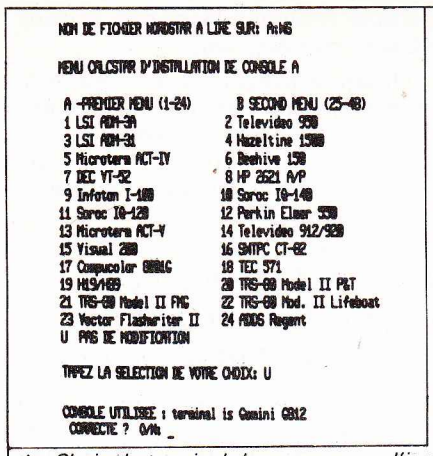
- **Tous systèmes CP/M.**
- **Unité de disquettes 13 cm.**

Prix du logiciel

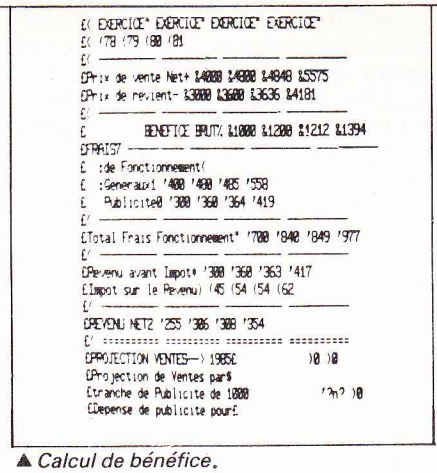
2 445 FF ttc.

Adresse du diffuseur

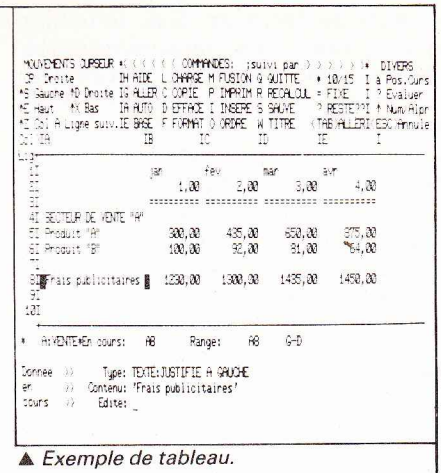
- **Micropro International France**
2, rue Nicolas-Ledoux
Paris Silic 206
94518 Rungis Cedex



▲ *Choix de terminal du programme d'installation.*



▲ *Calcul de bénéfice.*



▲ *Exemple de tableau.*

G, qui déplace le pointeur à une cellule donnée (la touche « Tab » remplit la même fonction) ;
H, qui affiche deux menus d'aide ; cette fonction serait très utile si elle faisait apparaître les renseignements voulus selon l'endroit où l'on se trouve dans l'entrée des données, mais la commande « H » donne deux écrans de renseignements, quel que soit le moment de l'appel ; il faut dire également qu'il n'y a qu'un seul moment où l'on puisse consulter les renseignements, c'est lorsque Calcstar attend une commande ou une entrée ; au risque de me répéter, je dois dire que là encore Supercalc est bien supérieur, car il n'y a pas beaucoup de conditions pendant lesquelles on ne puisse faire un appel à l'aide avec un simple «?» et, dans ce cas, on a accès uniquement à l'information nécessaire, sans passer par la lecture de toutes les commandes ;

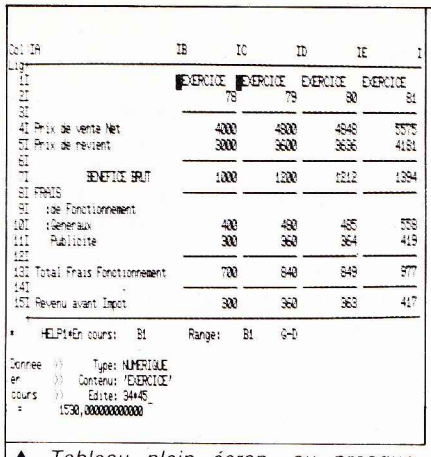
I(nsertion), qui insère une ligne ou une colonne ;
L, qui charge un fichier Calcstar précédemment enregistré ; là, il n'y a pas la possibilité de consulter la « directory » : il faut se souvenir du nom du fichier, sinon, il faut sortir de Calcstar et consulter la « directory » sous CP/M et rappeler Calcstar ;
M, qui permet la fusion d'un fichier sauvegardé précédemment avec les données du tableau ; la commande «;L» efface le contenu du tableau avant de charger le nouveau fichier ;
O(ption), qui donne une option de direction des calculs, soit verticalement soit horizontalement ;
P, qui imprime les données sur une imprimante ou un fichier qui pourra être traité plus tard par Wordstar ou tout autre logiciel de traitement de texte ; à chaque impression, Calcstar demande la vérification des options de configuration (longueur de page, de ligne, etc.) ; un simple retour chariot conserve la valeur précédente ; cela peut apparaître comme une sécurité ou un inconvénient ; pour ma part, je préfère qu'on me donne le choix entre la configuration et l'impression. Les titres de colonnes et lignes ne sont pas imprimés, leur impression peut être utile certaines fois mais, à mon avis, ce n'est pas un inconvénient majeur. Cette commande permet d'imprimer ou de tranférer sur un fichier la totalité du tableau ou seulement une partie.

Calcstar, un outil pour la prévision linéaire, scientifique ou autre

Q(uitte), qui est la sortie du programme Calcstar avec vérification ;
R(ecalcule), qui recalcule une entrée ou la totalité des entrées ; le calcul ne se fait que sur commande, ou lors d'une entrée et pour cette entrée seulement, contrairement aux autres « Calc », où le calcul se fait globalement avec l'option de s'effectuer à chaque entrée nouvelle ou sur commande ; ce manque de flexibilité de Calcstar ne favorise pas les exercices du genre « qu'est-ce qui se passe si », lorsque l'on veut étudier un modèle mathématique avec plusieurs options et voir les implications ;
S(auvegarde), qui sauvegarde le contenu du tableau ou d'une partie du tableau sur disque ; il n'y a pas la possibilité de consulter la « directory », mais Calcstar dira si un fichier existant déjà, n'utilise pas deux fois le même nom ;
W, qui visualise les têtes de colonnes et de ligne de la cellule pointée ;
= :, qui verrouille les en-têtes (lignes ou colonnes) ;
? :, qui indique la quantité de mémoire disponible en nombre de cellules ;
* :, qui change le format de tableau de dix lignes à quinze ou vice versa ;

Les fonctions arithmétiques disponibles sont les suivantes :
1) les classiques : +,−,*,/, qui sont les signes d'addition, de soustraction, de multiplication et de division.
2) les fonctions systèmes non moins classiques :
ABS : valeur absolue,
EXP : exponentielle,
LOG : logarithme décimal,
LN : logarithme népérien,
MAX : valeur maximale d'une série de valeurs,
MIN : valeur minimale d'une série de valeurs,
MOY : moyenne des valeurs dans une série de valeurs,
NBR : nombre de valeurs dans une série de valeurs,
SOM : somme d'une série de valeurs,
RACINE : racine carrée.

Les abréviations sont en français, c'est bien ; encore faut-il se rappeler qu'elles sont différentes du Basic, pour certaines.
3) Les fonctions originales :
REGR, qui calcule la régression linéaire des deux séries spécifiées et retourne la moyenne de la deuxième série (variables dépendantes) ;
PROJ (projection), qui insère une valeur de variable indépendante dans l'équation de régression et retourne la valeur probable (la variable dépendante) ;
DEPD, qui insère une valeur de variable dépendante dans l'équation de régression, l'effectue et retourne la meilleure estimation de variable indépendante ;
PENTE, qui retourne la valeur de la pente de l'équation de régression qui peut être utilisée pour évaluer la variable indépendante .

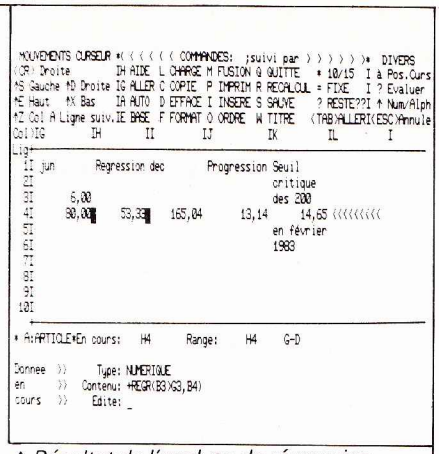


Col IA	IB	IC	ID	IE	I
Lig	EXERCICE	EXERCICE	EXERCICE	EXERCICE	
1I		79	79	80	81
2I					
3I					
4I Prix de vente Net	4000	4800	4848	5575	
5I Prix de revient	3000	3600	3636	4181	
6I					
7I BENEFICE BRUT	1000	1200	1212	1394	
8I FRAIS					
9I :de Fonctionnement					
10I :Generaux	400	480	485	558	
11I Publicite	300	360	364	419	
12I					
13I Total Frais Fonctionnement	700	840	849	977	
14I					
15I Revenu avant Impot	300	360	363	417	

* HELP1+En cours: B1 Range: B1 G-D

Donnee en cours = 1530,0000000000 : Type: NUMERIQUE; Contenu: 'EXERCICE'; Edite: 3445_

▲ *Tableau plein écran, ou presque... Exemple d'utilisation « calculateur ».*



^OUVEMENTS CURSEUR *(((((COMMANDES: ;suivi par))))))* DIVERS
(CR) Droite IH AIDE L CHARGE M FUSION @ QUITTE * 10/15 I à Pos.Curs
^S Gauche ^D Droite IG ALLER C COPIE P IMPRIM R RECALCUL = FIXE I ? Evaluer
^E Haut ^X Bas IA AUTO D EFFACE I INSERE S SAUVE ? RESTE??I ↑ Num/Alph
^Z Col A Ligne suiv. IE BASE F FORMAT O ORDRE W TITRE (TAB)ALLER(ESC)Annule

Col IG	IH	II	IJ	IK	IL	I
Lig						
1I	jun	Regression dec		Progression	Seuil	
2I					critique	
3I	6,00				des 200	
4I	80,00	53,33	165,04	13,14	14,65	
5I					en février	
6I					1983	
7I						
8I						
9I						
10I						

* A:ARTICLE*En cours: H4 Range: H4 G-D

Donnee en cours = +REGR(B3:G3,B4) : Type: NUMERIQUE; Contenu: +REGR(B3:G3,B4); Edite: _

▲ *Résultat de l'analyse de régression.*

La puissance de ces fonctions permettra de faire de la prévision financière avec une grande flexibilité et simplicité ; ces fonctions seraient très longues à mettre en place sur Visicalc ou Supercalc. C'est là le grand intérêt de Calcstar.

Un mot sur la régression linéaire : C'est un terme de statistique qui s'applique à la relation entre deux suites de nombres. Une équation caractérise cette relation ; cette équation est une droite calculée par la méthode des moindres carrés.

Visitrend possède des fonctions similaires.

Prenons un exemple : les ventes mensuelles pour un certain produit.

Nous avons deux suites :

. les mois : 1, 2, 3, 4, 5, 6 ;

. les ventes : 20, 30, 50, 60, 80, 80.

La représentation graphique de la corrélation entre ces deux suites donne le graphe de l'encadré ci-dessous.

Pour obtenir l'équation correspondant à une régression linéaire, on prend la droite obtenue en minimisant la somme des distances des points (à cette même droite, bien sûr). En d'autres termes, cette droite va représenter une tendance et l'on va pouvoir faire des projections en l'extrapolant. Calcstar permet de faire cela en peu de temps et d'efforts.

Voyons comment Calcstar peut nous aider à faire de la prévision financière, scientifique ou autre.

Pour plus de lisibilité, nous allons réserver la première ligne et la première colonne pour les titres. Nous allons donc aller dans la cellule B1, opération qui se fait soit en déplaçant le curseur, soit en appuyant sur la touche « TAB » suivie de « B1 » et « RETURN », soit encore en entrant la commande «;G», suivie de « B1 » et « RETURN ». Le curseur doit entourer la cellule « B1 ». A ce stade, on peut entrer « JAN » pour le mois de janvier, les courageux pourront l'entrer en entier ! Le mois de « Février » sera entré dans « C1 », « Mars » dans « D1 », etc.

Dans la première colonne nous entrerons les titres. Ainsi, le texte suivant pourra être entré dans la cellule « A3 » : «s numéro du mois ».

Dans la cellule « A4 » : « quantité d'articles vendus », on s'aperçoit que seuls les dix premiers caractères sont affichés ; pour remédier à cela, il suffit d'entrer la commande « ;F » suivie de W et 27, qui est la largeur nécessaire à la colonne pour l'affichage du titre le plus long.

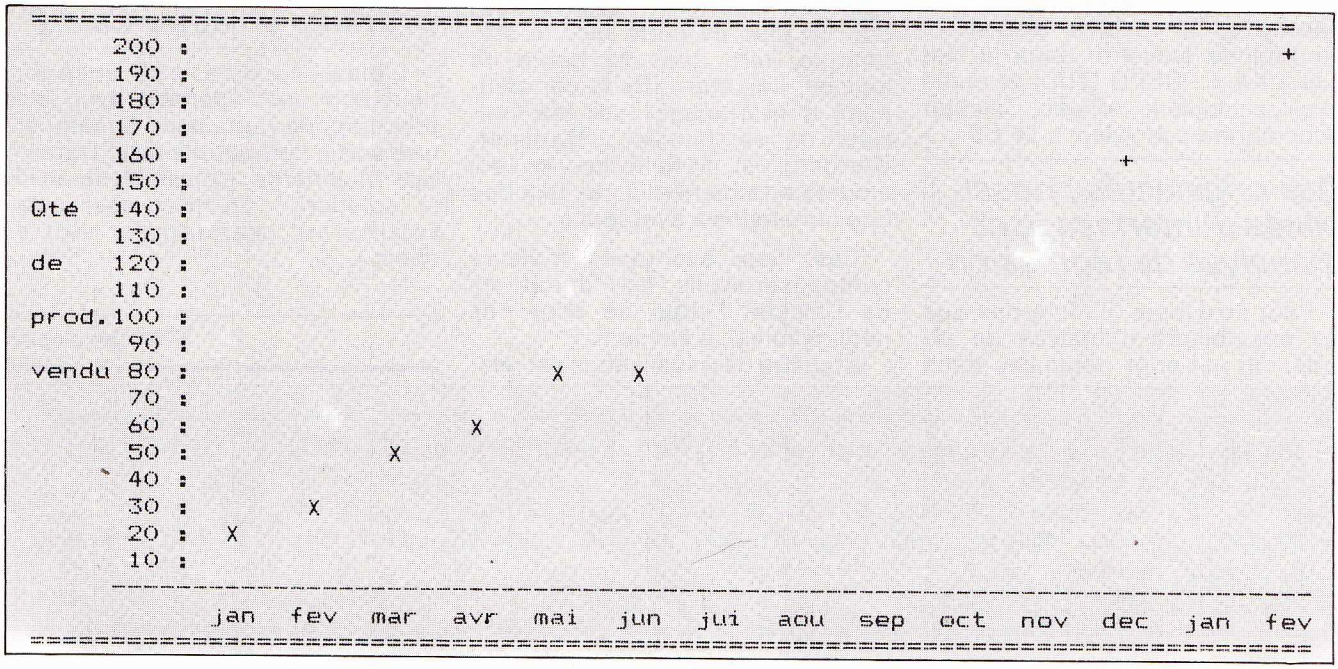
Maintenant, on peut entrer les deux suites : les numéros des mois de « B3 » à « G3 » et les quantités dans « B4 » à « G4 ».

Une compatibilité d'interfaçage avec Datastar, Wordstar et Mailmerge

Nous avons tous les éléments pour commencer notre travail de prévision. En « H4 », nous pouvons entrer l'équation de régression, qui doit comporter la référence aux deux suites, la suite indépendante (ici les mois) et la suite dépendante (les quantités vendues).

La syntaxe sera la suivante : + REGR (B3 > G3, B4).

Les deux suites comportent le même nombre d'éléments, il n'est donc pas nécessaire de préciser la fin de la deuxième suite.



Exemple d'utilisation des commandes de régression linéaire

	jan	fev	mar	avr	mai	jun	Regression dec		Progression	Seuil critique des 200 en février 1983
Numéro du mois	1.00	2.00	3.00	4.00	5.00	6.00				
Quantité vendue	20.00	30.00	50.00	60.00	80.00	80.00	53.33	165.04	13.14	14.65

Le nombre qui sera dans la cellule correspondra à la moyenne de la deuxième suite, ce qui a priori ne nous servira point, Calcstar nous donne un moyen de l'éliminer. Mais le plus important suit : en entrant dans la cellule « 14 » : + PROJ (12) nous saurons quelle est la prévision de vente pour le mois de décembre. Tout cela ne tient pas compte des effets saisonniers et n'est qu'une extrapolation, mais la même chose en Basic ou Supercalc demande des connaissances mathématiques et du temps de programmation.

Nous pourrons également connaître la pente de la droite de l'équation de régression, qui donne une idée de la progression des ventes dans notre cas.

En entrant dans la cellule « J4 » : + PENTE (), Calcstar nous donnera la progression des ventes sur la période de référence.

Si maintenant on veut savoir à quel moment on franchira un cap, on utilisera la commande : DEP.D (xxx) en indiquant le chiffre critique à la place des « xxx ».

Dans notre exemple, si nous voulons savoir à quel mois nous franchirons la barre des deux cents unités par mois, pour savoir à quelle époque il faudra augmenter la capacité de production par exemple, il faudra entrer la commande suivante dans la cellule « K4 » : DEPD (200) et après avoir recalculé le tableau, Calcstar nous donnera la valeur : 14.65.

Des originalités : possibilités d'interfaçage et d'analyse de régression

Pour être juste, il n'y a pas que les possibilités d'analyse de régression qui sont originales dans ce logiciel ; il y a également son interfaçage avec d'autres logiciels type « Star » : Datastar et Wordstar et Mailmerge (ce dernier fonctionnant avec Wordstar).

Datastar et Mailmerge travaillent sur des fichiers qui sont composés de « champs » variables séparés par une virgule ; Calcstar ne peut pas lire de tels fichiers mais peut en générer.

Notre exemple est alors représenté dans l'encadré ci-dessous.

Cette possibilité est spécifique à Calcstar mais ni Visicalc, ni Supercalc ne peuvent entrer des données dans Datastar ou dans Mailmerge.

Calcstar possède également, comme Visicalc et Supercalc, une sortie ASCII qui permet d'utiliser les tableaux ainsi produits dans des rapports écrits (nous avons utilisé cette possibilité).

Un très bon outil mais avec quand même des limitations

En conclusion, Calcstar est un outil inestimable pour les personnes ayant besoin de calculs de régression linéaire dans des domaines divers comme les sciences, les finances, l'agriculture, etc. En plus de cette faculté statistique, rare dans ce genre de logiciel, Calcstar offre la compatibilité d'interfaçage avec les logiciels de sa famille : Datastar, Wordstar et Mailmerge, ce qui ajoute certainement à ses facultés une dimension « système ».

Ces deux avantages majeurs compenseront-ils les limitations de Calcstar ? (qui ne sont pas négligeables), à savoir :

. la fenêtre de visualisation est relativement petite, malheureusement,

. il n'y a pas d'édition des cellules, ce qui signifie des pertes de temps, espace perdu entre les cellules, ne permettant pas de traits ni de textes continus entre les cellules,

. le texte est « bloqué » à une cellule, en d'autres mots si l'on veut des longs commentaires, il faut l'entrer dans des cellules différentes en tenant compte des espaces entre cellules,

. la fonction d'« aide » est présente mais limitée,

. la reconfiguration de l'écran lors du déplacement du curseur hors de l'écran est longue et confuse : en effet, la fenêtre est repositionnée trois cellules plus loin, ce qui demande un peu de gymnastique intellectuelle,

. il n'y a pas de possibilité d'imprimer les lettres et chiffres des colonnes et lignes.

. beaucoup de fonctions manquent : les fonctions trigonométriques ainsi que les fonctions Npv et Lookup,

. il n'y a pas de fenêtres dédoublées permettant de visualiser deux échantillons de données en même temps.

Et peut-être bientôt la famille Calc produira un autre Calc..?

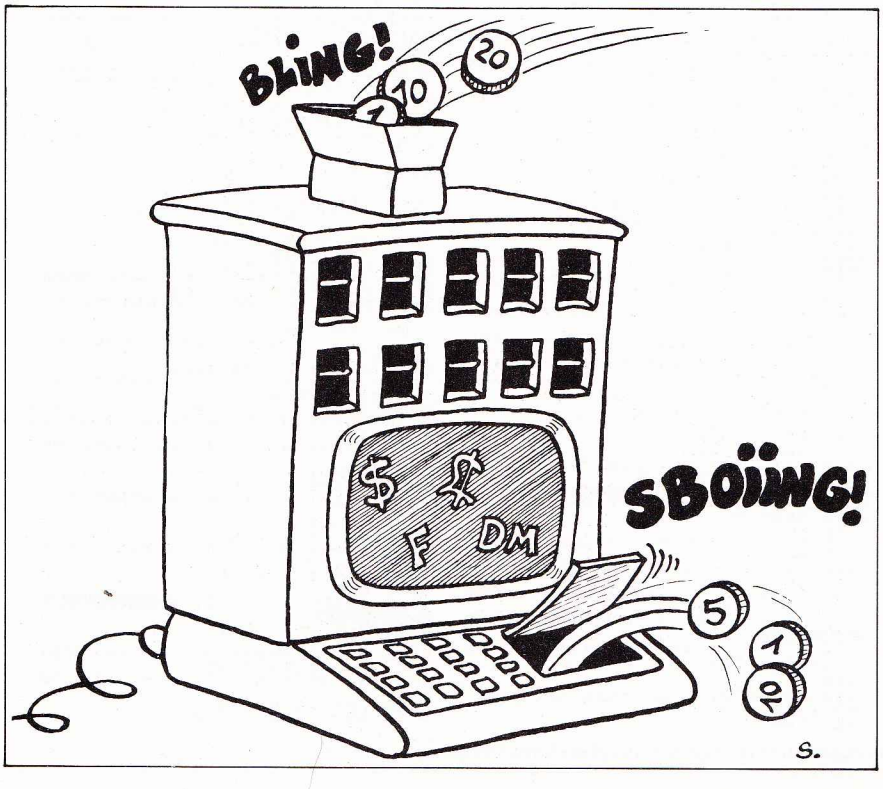
L'idéal n'existe pas encore ; peut-être un Supercalc avec les fonctions de régression linéaire et une sortie compatible avec Datastar, Mailmerge, Dbase II serait-il la solution ? En tout cas, cela faciliterait grandement notre choix !

Philippe Gysel

```
=================================================================================
jan,fev,mar,avr,mai,jun,Regression,dec,Progression,Seuil,
,,,,,,,,,,critique,
 1.00, 2.00, 3.00, 4.00, 5.00, 6.00,,,,,des 200,
 20.00, 30.00, 50.00, 60.00, 80.00, 80.00, 53.33, 165.05, 13.14, 14.66,
,,,,,,,,,,en février,
,,,,,,,,,,1983,
=================================================================================
```

enfin une valeur sûre pour gérer votre portefeuille de titres

Pourquoi ne pas vous aider de votre Sinclair ZX-81, muni de son extension 16 Ko MEV et éventuellement de l'imprimante, pour gérer votre portefeuille de titres ? Vous pourrez obtenir grâce à ce programme, outre la comptabilisation des valeurs détenues (jusqu'à trente valeurs différentes numérotées de 1 à 30), le calcul de la rentabilité globale de votre portefeuille, ainsi que celui de chaque valeur.



Au préalable, il vous faudra entrer en mémoire près de 9 K-octets, en prenant bien soin de décrire chaque ligne telle qu'elle figure dans la liste du programme, notamment pour tout ce qui est affichage de messages alphanumériques (même nombre d'espaces, etc.) et pour tout ce qui concerne la tabulation écran ou imprimante.

La toute première fois que vous utilisez le programme vous devez faire RUN, puis, après l'affichage du menu, vous devez nécessairement choisir l'option 5, c'est-à-dire la remise à zéro des fichiers mouvements.

Tapez « oui » en toutes lettres ; cela est indispensable, car certaines variables doivent être déclarées au cours de cette portion de programme. Vous pouvez alors mettre à jour votre fichier valeur.

Vingt caractères sont disponibles pour chaque valeur

Pour les utilisations ultérieures (le programme ayant été sauvegardé précédemment par l'exécution de la ligne 9020 SAVE « BOURSE »), après chargement du programme contenant les données de votre portefeuille, l'exécution se poursuit automatiquement par un GOTO à la ligne 100.

Si au cours de vos transactions, il arrive que le programme s'arrête sur un compte rendu d'er-

Liste des variables

A	Divers
ACH	Total des entrées
C (N)	Cours de la valeur de type N
CPN	Total des coupons encaissés
D$ (N)	Date de la dernière opération sur la valeur de type N
DIFF	Différence entre prix de revient et valeur du portefeuille
E$	Date d'opération
M (N)	Prix de revient global de la valeur de type N
O$ (X)	Type d'opération (entrée, sortie, etc.)
P	Cours de négociation
PRG	Prix de revient global d'une valeur
PRGT	Prix de revient global du portefeuille
PRU	Prix de revient unitaire d'une valeur
PTF	Valeur du portefeuille
PVD	Plus-value dégagée sur une opération
PVT	Total des plus-values dégagées
Q (N)	Nombre de valeurs de type N détenues
R	Quantité négociée
S$	Sauvegarde provisoire du libellé code opération
T	Montant d'une opération
V$ (N)	Libellé de la valeur de type N
VAR	Variation en pourcentage entre deux montants
VTE	Total des sorties
W	Compteur de lignes lors d'édition à l'écran
X$	Divers
Y	Numéro de colonne pour édition
Z	Utilisé pour la tabulation

reurs, un GOTO 100 en mode commande permet de reprendre l'exécution du programme, en affichant à nouveau le menu (surtout ne pas faire RUN).

Une fois le programme initialisé, procédez à la mise à jour du fichier valeur (option 1 ; vingt caractères sont disponibles pour le libellé de chaque valeur ; le cours à mettre est celui de la date d'évaluation souhaitée).

Le programme de libellé des codes réalisé selon vos souhaits

Ce fichier valeur peut être alors visualisé sur écran ou imprimé par le choix de l'option 2 du menu. Pour l'impression, la date peut être entrée sous toutes les formes (par exemple : 31 août 1982).

Pour effectuer la saisie des opérations, prenez l'option 3 du menu. Vous disposez en tout de trois codes d'opérations :

Programme de gestion d'un portefeuille de valeurs (début)

```
AUTEUR : MR ALAIN KIENLEN


   5 REM BOURSE
  10 DIM V$(30,20)
  20 DIM D$(30,6)
  30 DIM O$(3,7)
  40 DIM Q(30)
  50 DIM M(30)
  70 DIM C(30)
  75 LET O$(1)="ENTREE"
  77 LET O$(2)="SORTIE"
  80 LET O$(3)="COUPON"
 100 CLS
 101 PRINT TAB 13;"[REVERSE]"
 110 PRINT AT 4,0;"1 POUR M.A.J.
 FICHIER VALEUR"
 120 PRINT AT 7,0;"2 POUR EDITIO
N FICHIER VALEUR"
 130 PRINT AT 10,0;"3 POUR JOURN
AL DES MOUVEMENTS"
 140 PRINT AT 13,0;"4 POUR EDITI
ON D UN RELEVE"
 150 PRINT AT 16,0;"5 POUR REMIS
E A ZERO DES                FICHIERS M
OUVEMENTS"
 170 PRINT AT 20,0;"9 SI FIN"
 180 IF INKEY$<>"1" AND INKEY$<>
"2" AND INKEY$<>"3" AND INKEY$<>
"4" AND INKEY$<>"5" AND INKEY$<>
"9" THEN GOTO 180
 190 LET A=CODE INKEY$-28
 195 CLS
 200 GOTO 1000*A
1000 CLS
1001 PRINT TAB 3;"[REVERSE] [REVERSE] [REVERSE]
[REVERSE]"
1010 PRINT AT 4,0;"QUEL N0 VOULE
Z METTRE A JOUR ?  (99 SI FIN)"
1020 INPUT N
1025 IF N=99 THEN GOTO 100
1030 PRINT AT 7,0;"NUMERO ";N
1040 PRINT AT 9,0;"LIBELLE: ";V$
(N)
1050 PRINT AT 11,0;"COURS: ";C(N
);" FRANCS"
1060 PRINT AT 13,0;"FRAPPER 1 PO
UR MODIFIER LE                LI
BELLE                        2 PO
UR MODIFIER LE COURS         9 SI
 FIN"
1070 IF INKEY$<>"1" AND INKEY$<>
"2" AND INKEY$<>"9" THEN GOTO 10
70
1075 LET A=CODE INKEY$-28
1080 IF A=9 THEN GOTO 1000
1085 PRINT AT 21,0;"VOTRE REPONS
E: ";A
1090 IF A=1 THEN GOTO 1100
1095 IF A=2 THEN GOTO 1200
1100 INPUT V$(N)
1105 PRINT AT 21,0;"
     "
1110 GOTO 1030
1200 INPUT C(N)
1205 PRINT AT 21,0;"
     "
1210 GOTO 1030
2000 CLS
2010 PRINT TAB 3;"[REVERSE] [REVERSE] [REVERSE]
[REVERSE] [REVERSE]"
2020 PRINT AT 7,0;"1 POUR L IMPR
IMANTE"
2030 PRINT AT 10,0;"2 POUR L ECR
AN"
2040 PRINT AT 13,0;"9 SI FIN"
2050 IF INKEY$<>"1" AND INKEY$<>
"2" AND INKEY$<>"9" THEN GOTO 20
50
2055 LET A=CODE INKEY$-28
2060 IF A=9 THEN GOTO 100
2080 IF A=2 THEN GOTO 2500
2085 CLS
2090 PRINT AT 10,0;"INDIQUER LA
DATE"
2095 INPUT X$
2100 LPRINT ;"****************
**************"
2110 LPRINT TAB 3;"EDITION DU FI
CHIER VALEUR"
2115 LPRINT TAB 6;"AU ";X$
2117 LET X$=""
2120 LPRINT ;"****************
**************"
2130 LPRINT ;"CODE";TAB 10;"LIBE
LLE";TAB 26;"COURS"
2140 LPRINT
2150 FOR N=1 TO 30
2160 IF V$(N)="
     THEN GOTO 2180
2165 GOSUB 9050
2165 GOSUB 9030
2170 LPRINT TAB Z;N;"*";V$(N);"*
";TAB (28-Y);C(N)
2180 NEXT N
2185 LPRINT ;"FIN D EDITION"
2190 LPRINT ;"****************
**************"
2200 GOTO 2000
2500 CLS
2505 LET W=0
2510 PRINT TAB 3;"EDITION DU FIC
HIER VALEUR"
2520 PRINT ;"****************
**********"
2530 PRINT ;"CODE";TAB 10;"LIBEL
LE";TAB 26;"COURS"
2540 PRINT ;"****************
**********"
2550 FOR N=1 TO 30
2560 IF V$(N)="
     THEN GOTO 2580
2565 LET W=W+1
2566 GOSUB 9080
2567 GOSUB 9050
2570 PRINT AT W+4,Z;N;"*";V$(N);
"*";TAB (28-Y);C(N)
2574 IF W=16 THEN GOSUB 9998
2575 IF W=16 THEN LET W=0
2580 NEXT N
2584 PRINT
2585 PRINT "FIN D EDITION"
2587 PRINT
2590 IF INKEY$="" THEN GOTO 2590
2600 GOTO 2000
3000 LPRINT ;"****************
**************"
3005 LPRINT TAB 5;"[REVERSE] [REVERSE] [REVERSE]
[REVERSE]"
3010 LPRINT ;"****************
**************"
3012 CLS
3015 LET S$=""
3020 PRINT AT 2,0;"CODE VALEUR?
(99 SI FIN)"
3025 INPUT N
3025 IF N=99 THEN LPRINT "FIN D
EDITION"
3027 IF N=99 THEN GOTO 100
3030 PRINT AT 2,0,V$(N);" CODE:
";N
3035 PRINT AT 4,0;"SITUATION PRE
CEDENTE DU ";D$(N)(1 TO 2);".";D
$(N)(3 TO 4);".";D$(N)(5 TO 6)
3040 PRINT AT 5,0;"**************
*******"
3045 PRINT AT 6,0;"QUANTITE DETE
NUE: ";Q(N)
3050 PRINT AT 7,0;"PRIX REVIENT
GLOBAL: ";M(N);
3051 IF M(N)=0 THEN LET PRU=0
3052 IF M(N)=0 THEN GOTO 3060
3055 LET PRU=(INT (M(N)/Q(N)*100
))/100
3060 PRINT AT 8,0;"PRIX REVIENT
UNITAIRE: ";PRU
3065 PRINT AT 10,0;"CODE OPERATI
ON ? ";
3066 IF INKEY$<>"1" AND INKEY$<>
"2" AND INKEY$<>"3" THEN GOTO 30
68
3070 LET X=CODE INKEY$-28
3075 PRINT O$(X)
3076 PRINT AT 11,0;"AUTRE LIBELL
E ?"
3077 INPUT L$
3078 IF L$="" THEN GOTO 3081
3079 LET S$=O$(X)
3080 LET O$(X)=L$
3081 PRINT AT 11,0;"DATE ?
     ";
3082 PRINT AT 21,5;"."
3085 INPUT E$
3086 PRINT AT 21,5;" "
3087 IF LEN E$<6 THEN GOTO 3080
3090 PRINT AT 11,5;E$
3095 PRINT AT 10,0;"QUANTITE ?
     "
3100 INPUT R
3101 IF X=2 AND R>Q(N) THEN GOTO
 9300
3105 PRINT AT 10,0;O$(X);":";R;"
LE ";E$(1 TO 2);".";E$(3 TO 4);
".";E$(5 TO 6)
3106 IF S$<>"" THEN LET O$(X)=S$
3110 PRINT AT 11,0;"**************
*******"
3115 PRINT AT 12,0;"AU COURS DE
     "
3120 INPUT P
3125 PRINT P;" FF"
3130 PRINT AT 13,0;"MONTANT NET:
     ";
3135 INPUT T
3140 PRINT T;" FF"
3145 PRINT AT 15,0;"SAISIE CORRE
CTE O/N ?"
3146 FOR Z=1 TO 5
3147 NEXT Z
3148 PRINT AT 15,0;"[REVERSE] [REVERSE] [REVERSE]
[REVERSE] [REVERSE]"
3149 FOR Z=1 TO 5
3150 NEXT Z
3155 IF INKEY$<>"O" AND INKEY$<>
"N" THEN GOTO 3145
3160 IF INKEY$="N" THEN GOTO 301
2
3165 LET D$(N)=E$
3170 GOTO 3100+X*100
```

```
MENU

1 POUR M.A.J. FICHIER VALEUR

2 POUR EDITION FICHIER VALEUR

3 POUR JOURNAL DES MOUVEMENTS

4 POUR EDITION D UN RELEVE

5 POUR REMISE A ZERO DES
  FICHIERS MOUVEMENTS

9 SI FIN
?
```

Menu général.

```
MISE A JOUR FICHIER VALEUR

QUEL N0 VOULEZ METTRE A JOUR ?
(99 SI FIN)
NUMERO 1
LIBELLE: ESSAI
COURS: 0 FRANCS
FRAPPER 1 POUR MODIFIER LE
          LIBELLE
        2 POUR MODIFIER LE COURS
        9 SI FIN


VOTRE REPONSE : 2
```

Menus pour chaque option.

```
EDITION DU FICHIER VALEUR

1 POUR L IMPRIMANTE

2 POUR L ECRAN

9 SI FIN
```

```
EDITION D UN RELEVE

1 POUR L IMPRIMANTE

2 POUR L ECRAN

9 SI FIN
```

Programme de gestion d'un portefeuille de valeurs (suite et fin)

```
3200 LET Q(N)=Q(N)+R
3205 LET M(N)=M(N)+T
3206 IF Q(N)=0 THEN LET PRU=0
3208 IF Q(N)=0 THEN GOTO 3220
3210 LET PRU=(INT (M(N)/Q(N)*100
))/100
3220 LET ACH=ACH+T
3225 GOTO 3600
3300 LET Q(N)=Q(N)-R
3305 LET M(N)=M(N)*Q(N)/(Q(N)+R)
3310 LET M(N)=(INT (M(N)*100))/1
00
3315 LET PVD=T-(R*PRU)
3320 LET VTE=VTE+T
3325 LET PVT=PVT+PVD
3330 GOTO 3600
3400 LET CPN=CPN+T
3600 PRINT AT 15,0;"SITUATION PO
STERIEURE"
3605 PRINT AT 16,0;"************
*********"
3610 PRINT AT 17,0;"NOMBRE DE TI
TRES: ";Q(N)
3615 PRINT AT 18,0;"PRIX REVIENT
 GLOBAL: ";M(N);" FF"
3620 PRINT AT 19,0;"PRIX REVIENT
 UNITAIRE: ";PRU;" FF"
3625 IF X=2 THEN PRINT AT 21,0;"
PLUS VALUE DEGAGEE: ";PVD;" FF"
3628 PRINT AT 0,5;"

3630 COPY
3635 IF INKEY$="" THEN GOTO 3635
3640 GOTO 3010
4000 CLS
4005 PRINT TAB 7;"[illegible]
[illegible]"
4010 PRINT AT 7,0;"1 POUR L IMPR
IMANTE"
4015 PRINT AT 10,0;"2 POUR L ECR
AN"
4020 PRINT AT 13,0;"9 SI FIN"
4025 IF INKEY$<>"1" AND INKEY$<>
"2" AND INKEY$<>"9" THEN GOTO 40
35
4030 LET A=CODE INKEY$-28
4035 IF A=9 THEN GOTO 100
4037 PRINT AT 16,0;"DATE D EDITI
ON?"
4038 INPUT X$
4040 IF A=1 THEN GOTO 4500
4100 CLS
4105 LET PTF=0
4110 LET U=0
4115 PRINT TAB 8;"[illegible]
[illegible]"
4120 PRINT TAB 6;"AU ";X$
4125 FOR N=1 TO 30
4130 IF Q(N)=0 THEN GOTO 4190
4135 LET MT=Q(N)*C(N)
4140 GOSUB 9100
4145 PRINT AT U+2,0;"*";V$(N)
4147 PRINT AT U+2,Y;MT
4150 PRINT Q(N);" A ";C(N);" FF"
4155 LET PRU=(INT (M(N)/Q(N)*100
))/100
4160 PRINT ;"PRU: ";PRU;" PRG:";
M(N)
4165 PRINT ;"DER. OP. LE ";D$(N)
(1 TO 2);" ";D$(N)(3 TO 4);".";D
$(N)(5 TO 6)
4170 LET PTF=PTF+MT
4175 LET U=U+5
4179 IF U=20 THEN PRINT "APPUYER
 POUR LA SUITE"
4180 IF U=20 THEN GOSUB 9998
4181 IF U=20 THEN PRINT AT 21,0;
"
4185 IF U=20 THEN LET U=0
4190 NEXT N
4205 LET MT=PTF
4210 GOSUB 9100
4220 PRINT AT 21,15;"TOTAL:";TAB
 Y;PTF
4230 GOSUB 9998
4235 GOTO 4000
4500 LPRINT TAB 8;"[illegible]
[illegible]"
4501 CLS
4503 LPRINT
4505 LPRINT TAB 6;"AU ";X$
4510 LPRINT
4515 LET PTF=0
4516 LET PRGT=0
4520 LPRINT
4525 FOR N=1 TO 30
4530 IF Q(N)=0 THEN GOTO 4590
4535 LET MT=Q(N)*C(N)
4540 GOSUB 9100
4545 LPRINT ;"*";V$(N);TAB Y;MT
4546 LPRINT ;"
"
4550 LPRINT Q(N);"  A ";C(N);" F
F"
4555 LET PRU=(INT (M(N)/Q(N)*100
))/100
4560 LPRINT ;"PRU: ";PRU;" PRG:"
;M(N)
4565 LPRINT ;"DER. OP. LE ";D$(N
)(1 TO 2);" ";D$(N)(3 TO 4);".";
D$(N)(5 TO 6)
4567 IF M(N)=0 THEN GOTO 4570
4568 LET VAR=(MT/M(N)-1)*100
4569 LPRINT ;"VAR. ";(INT (VAR*1
00))/100;" PC"
4570 LET PTF=PTF+MT
4571 LET PRGT=PRGT+M(N)
4575 LPRINT
4590 NEXT N
4605 LET MT=PTF
4610 GOSUB 9100
4615 LPRINT
4620 LPRINT TAB 15;"[illegible]";TAB Y
;PTF
4625 LET MT=PRGT
4630 GOSUB 9100
4635 LPRINT
4640 LPRINT ;"PRIX REVIENT TOTAL
";TAB Y-2;"-";TAB Y;PRGT
4645 LPRINT TAB 21;"-----------"
4650 LPRINT
4655 LET MT=PTF-PRGT
4656 LET DIFF=MT
4657 LET MT=ABS MT
4660 GOSUB 9100
4662 IF DIFF<0 THEN LET Y=Y-1
4665 LPRINT ;"VARIATION EN FF:";
TAB Y;DIFF
4667 IF PRGT=0 THEN GOTO 4680
4670 LET VAR=(PTF/PRGT-1)*100
4675 GOSUB 9210
4680 IF PRGT=0 THEN GOTO 4680
4685 GOSUB 9100
4687 IF PVT<0 THEN LET Y=Y-1
4690 LPRINT ;"PLUS VALUES DEGAGE
ES:";TAB Y;PVT
4693 IF PRGT=0 THEN GOTO 4705
4695 LET VAR=(PVT/PRGT)*100
4700 GOSUB 9210
4705 LET MT=CPN
4710 GOSUB 9100
4720 LPRINT ;"COUPONS ENCAISSES"
;TAB Y;CPN
4722 IF PRGT=0 THEN GOTO 4735
4725 LET VAR=(CPN/PRGT)*100
4730 GOSUB 9210
4735 LPRINT
4740 LET MT=ABS (DIFF+PVT+CPN)
4745 GOSUB 9100
4748 IF DIFF+PVT+CPN<0 THEN LET
Y=Y-1
4750 LPRINT ;"TOTAL.........";T
AB Y;DIFF+CPN+PVT
4753 IF PRGT=0 THEN GOTO 4765
4755 LET VAR=((DIFF+CPN+PVT)/PRG
T)*100
4760 GOSUB 9210
4765 LPRINT
4770 LPRINT ;"TOTAL DES ENTREES:
";ACH
4775 LPRINT ;"TOTAL DES SORTIES:
";VTE
4780 LPRINT
4785 LPRINT "**********
**********"
4790 GOTO 4000
5000 CLS
5005 PRINT AT 8,0;"OUI OU NON VO
ULEZ VOUS FAIRE LA"
5010 PRINT AT 10,9;"REMISE A ZER
O ?"
5012 INPUT X$
5013 IF X$<>"OUI" THEN GOTO 100
5015 LET ACH=0
5020 LET VTE=0
5025 LET CPN=0
5027 LET PVT=0
5030 PRINT AT 12,0;"SI VOUS AVEZ
 MIS OUI AU LIEU DE NON...",,,,"
TANT PIS C EST FAIT."
5035 GOSUB 9998
5040 GOTO 100
9000 CLS
9005 PRINT AT 8,3;"SAUVEGARDEZ V
OTRE FICHIER"
9010 PRINT AT 12,0;"FRAPPEZ UNE
TOUCHE  APRES AVOIR  PREPARE VOTR
E MAGNETOPHONE"
9015 GOSUB 9998
9020 SAVE "BOURS[illegible]"
9025 GOTO 100
9050 LET Z=1
9060 IF N>=10 THEN LET Z=0
9070 RETURN
9080 LET Y=0
9090 IF C(N)>=10 THEN LET Y=1
9091 IF C(N)>=100 THEN LET Y=2
9092 IF C(N)>=1000 THEN LET Y=3
9093 IF C(N)>=10000 THEN LET Y=4
9094 RETURN
9100 LET Y=28
9101 IF MT>=10 THEN LET Y=27
9102 IF MT>=100 THEN LET Y=26
9103 IF MT>=1000 THEN LET Y=25
9104 IF MT>=10000 THEN LET Y=24
9105 IF MT>=100000 THEN LET Y=23
9106 IF MT>=1000000 THEN LET Y=2
2
9107 RETURN
9210 LPRINT ;"SOIT ";(INT (VAR*1
00))/100;" PC"
9215 RETURN
9300 PRINT AT 15,0;"0TE VENDUE >
 0TE POSSEDEE,      OPERATION RE
FUSEE."
9305 GOSUB 9998
9310 GOTO 3012
9998 IF INKEY$="" THEN GOTO 9998
9999 RETURN


  8860 OCTETS DE PROGRAMME.
```

```
**********************************
        JOURNAL DES MOUVEMENTS
**********************************

L ORD.INDIVIDUEL      CODE: 1

SITUATION PRECEDENTE DU   .  .
**********
QUANTITE DETENUE: 0
PRIX REVIENT GLOBAL: 0
PRIX REVIENT UNITAIRE: 0

ACHAT  :100 LE 12.06.82
**********
AU COURS DE : 280 FF
MONTANT NET: 28253.37 FF

SITUATION POSTERIEURE
**********
NOMBRE DE TITRES: 100
PRIX REVIENT GLOBAL: 28253.37 FF
PRIX REVIENT UNITAIRE: 282.53 FF


**********************************

L ORD.INDIVIDUEL      CODE: 1

SITUATION PRECEDENTE DU 12.06.82
**********
QUANTITE DETENUE: 100
PRIX REVIENT GLOBAL: 28253.37
PRIX REVIENT UNITAIRE: 282.53

ATTRIB.:10 LE 15.07.82
**********
AU COURS DE : 0 FF
MONTANT NET: 0 FF

SITUATION POSTERIEURE
**********
NOMBRE DE TITRES: 110
PRIX REVIENT GLOBAL: 28253.37 FF
PRIX REVIENT UNITAIRE: 256.84 FF

**********************************

L ORD.INDIVIDUEL      CODE: 1

SITUATION PRECEDENTE DU 15.07.82
**********
QUANTITE DETENUE: 110
PRIX REVIENT GLOBAL: 28253.37
PRIX REVIENT UNITAIRE: 256.84

COUPON :110 LE 30.07.82
**********
AU COURS DE : 8 FF
MONTANT NET: 880 FF

SITUATION POSTERIEURE
**********
NOMBRE DE TITRES: 110
PRIX REVIENT GLOBAL: 28253.37 FF
PRIX REVIENT UNITAIRE: 256.84 FF

**********************************

L ORD.INDIVIDUEL      CODE: 1

SITUATION PRECEDENTE DU 30.07.82
**********
QUANTITE DETENUE: 110
PRIX REVIENT GLOBAL: 28253.37
PRIX REVIENT UNITAIRE: 256.84

VENTE  :40 LE 18.08.82
**********
AU COURS DE : 325 FF
MONTANT NET: 12805.27 FF

SITUATION POSTERIEURE
**********
NOMBRE DE TITRES: 70
PRIX REVIENT GLOBAL: 17979.41 FF
PRIX REVIENT UNITAIRE: 256.84 FF

PLUS VALUE DEGAGEE: 2531.67 FF
**********************************
FIN D EDITION
```

Programme d'édition d'un journal des mouvements.

```
        RELEVE DE TITRES
        AU 31 AOUT 1982

*L ORD.INDIVIDUEL              24514
70  A 350.2 FF
PRU: 256.84 PRG:17979.41
DER. OP. LE 18.08.82
VAR. 36.34 PC

                  TOTAL        24514

PRIX REVIENT TOTAL:     - 17979.41
                          ----------
VARIATION EN FF:             6534.59
SOIT 36.34 PC
PLUS VALUES DEGAGEES:        2531.67
SOIT 14.08 PC
COUPONS ENCAISSES             880
SOIT 4.89 PC

TOTAL.........               9946.26
SOIT 55.32 PC

TOTAL DES ENTREES: 26253.37
TOTAL DES SORTIES: 12805.27

**********************************
```

Programme d'un relevé de titres.

. les entrées : code 1,
. les sorties : code 2,
. les coupons :code 3.

Le programme vous demande cependant si vous souhaitez modifier le libellé correspondant à ces codes, par exemple « achat » au lieu d'entrée et « vente » au lieu de sortie. Le nouveau libellé ne doit pas comporter plus de six caractères. Si vous tapez seulement Newline, le libellé standard apparaît, par exemple : entrée.

Le ZX-81 est une valeur sûre, si vous savez l'utiliser !

• La date, dont l'introduction est demandée à la suite du libellé, doit, cette fois-ci, être émise sous la forme JJ MM AA (par exemple : 3 mai 1982 = 030582).

Afin de ne pas fausser les fichiers « mouvements », il y a lieu, pour toute opération telle qu'une sortie de droits (qui n'est pas une vente) accompagnée d'une attribution d'action(s), de sortir les droits avec le code entrée et d'inscrire qualité et montant en « négatif ».

Il en est de même pour toute annulation d'opération enregistrée par erreur.

Ci-dessous, voici maintenant la décomposition du programme.

N'oubliez pas que votre Sinclair est une valeur sûre, mais qu'il ne fait que ce que vous lui demandez de faire.

Alain Kienlen

Programme

Lignes 10 à 80 :	initialisation des tableaux.
Lignes 100 à 200 :	affichage du « menu » général.
Lignes 1000 à 1210 :	mise à jour du fichier valeur.
Lignes 2000 à 2600 :	édition, sur l'écran ou sur l'imprimante, du fichier valeur.
Lignes 3000 à 3640 :	journal des mouvements, destiné à mettre à jour les quantités de titres détenus et à enregistrer les montants globaux des entrées, sorties et coupons encaissés.
Lignes 4000 à 4790 :	édition, sur l'écran ou sur l'imprimante, d'un relevé des valeurs détenues, indiquant différents renseignements.
Lignes 5000 à 5040 :	remise à zéro des fichiers mouvements, (entrées, sorties, coupons et plus values) à effectuer lors de la première utilisation du programme et à chaque fin d'année.
Lignes 9000 à 9025 :	sauvegarde du programme et des données de votre portefeuille. Vous y accédez par l'option 9 du menu général.

un Apple dans la pyramide d'un pharaon

Difficile à lire, et plus encore à déplacer, le livre gravé dans les stèles des pyramides et autres monuments de l'Egypte pharaonique ! Certes, Jean-François C., ce fut du bon boulot ! Mais franchement, ne rêvez-vous pas de lire la langue des scribes dans le texte d'origine, seul moyen d'une instruction sérieuse. Ne vous arrive-t-il pas de pester ? Quand les tourniquets de livres de poches proposeront-ils enfin des ouvrages en hiéroglyphes ? Si ce n'est pour demain, c'est peut-être pour très bientôt. Les sciences humaines aussi se mettent à l'informatique. Et, plus encore que celle des égyptologues de la vieille école, imaginez la surprise, l'étonnement de l'âme errante du pharaon Pépi Ier (sic) quand, un beau jour de l'été 1980, il vit arriver, dans sa chambre funéraire, une équipe du Cnrs avec un Apple 2, une caméra vidéo en laisse.

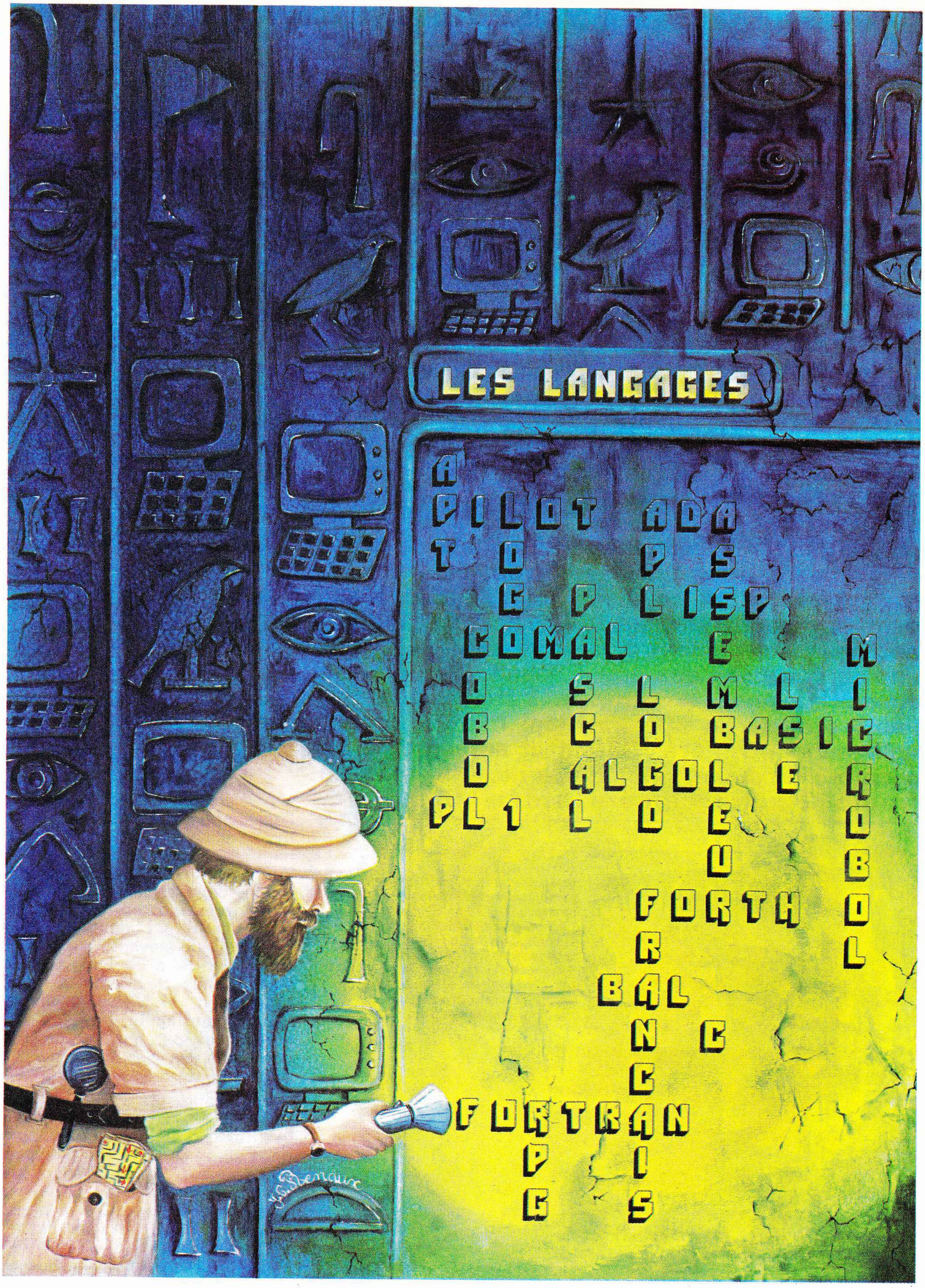
◀ *Au premier plan, les manettes qui servent à déplacer le point lumineux sur l'écran. En fond d'écran, quand on fait la collecte des hiéroglyphes, on trouve l'image vidéo (absente ici) du document d'origine. On trace, grâce aux manettes, le contour du hiéroglyphe qui est mémorisé sur une disquette de l'Apple.*

Gutenberg s'en serait arraché les cheveux. Pensez ou plutôt imaginez : un alphabet de sept mille signes qui peuvent s'écrire aussi bien de gauche à droite que dans le sens contraire, voire de haut en bas. Sans parler des fantaisies de « mise en page » du scribe qui n'hésite pas, si en fin de texte la place lui fait défaut, à emboîter les glyphes, à en graver plusieurs quand il n'y a d'espace que pour un seul.

Bref, version graphique, c'est la « malédiction des pharaons » qui s'abat sur l'égyptologue tentant de reproduire, d'analyser et d'éditer le pharaonique abécédaire de l'Egypte ancienne. Avec les tailles différentes des signes qu'il relève, ce sont quelque sept mille graphismes qu'il lui faut traiter.

Pourtant ce travail... de romain, les très sérieux égyptologues de l'université d'Oxford ou de l'Institut français d'archéologie orientale du Caire ont osé l'entreprendre : sept mille signes ont été fondus dans le plomb, et farfouillant dans les boîtes qui les contiennent, tout quidam autorisé peut voir de savants messieurs vouer leur vie à l'humble rôle de typographes de Ramsès, Toutânkhamon, et Machin-Anakton. Une à deux lignes peuvent être composées par jour, et environ une page les semaines qui ne tombent pas un 11 novembre !

D'où le prix (de 200 à 1 000 F) et la rareté des livres reproduisant ces textes hiéroglyphiques dont nous sommes tous très friands... Le « nous » désignant plus spécialement les chercheurs en sciences humaines du Cnrs (Centre national de la recherche scientifique), qui, au début des années soixante-dix, tentèrent l'usage de l'outil informatique pour saisir


LES LANGAGES
APT
PILOT
ADA
LOGO
APL
LISP
COMAL
COBOL
PASCAL
ASSEMBLEUR
LSE
MICROBOL
BASIC
ALGOL
PL1
LOGO
FORTH
FRANCAIS
BAL
C
FORTRAN
RPG

plus rapidement les informations, les traces écrites laissées par les civilisations passées.

Non sans difficulté, car pour un professeur (Jean L., membre de l'Institut), qui dès 1968 déclare que *« les savants de l'an 2000 seront des informaticiens »*, le temps est encore proche où un ancien professeur d'égyptologie au Collège de France interdisait l'usage des photocopieuses dans son département, jugeant que le premier apprentissage de sa science devait passer par un exercice de copiste !

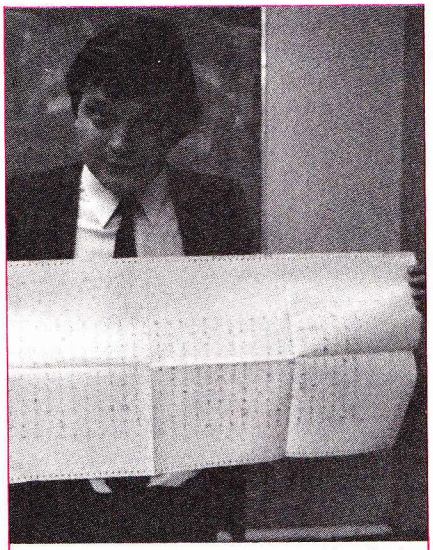
C'est dans les années 75/76 que l'égyptologie prend le tournant informatique, particulièrement avec Michaël H. Malgré ses détracteurs il y croit, pour avoir vu des traceurs dessiner des cartes de géographie autrement plus complexes que des hiéroglyphes. Pour les fournir à une mémoire, il s'attelle tout d'abord à relever à la main quelques points sur leur contour, assez proches pour qu'en les joignant il obtienne des segments de droite suffisamment rapprochés pour effacer les angles.

Il s'arrête bientôt. Relever ainsi les abscisses et ordonnées des glyphes afin d'en instruire plus tard une mémoire électronique, prend environ une heure par unité graphique. Combien pour l'ensemble ? La prévision la plus basse est de l'ordre de sept mille heures pour un vocabulaire « de base ». Et le temps coûte très cher !

Les systèmes en temps partagé retiennent tout d'abord l'attention des chercheurs : connectée à plusieurs terminaux, l'unité centrale permet les opérations de saisie de données. (Pourquoi immobiliser un gros ordinateur pour une tâche effectuée lentement par l'opérateur humain ?)

Mais bien vite il est possible d'envisager une évolution des traitements en utilisant des ordinateurs individuels. Ainsi, A. G., P. M. et d'autres chercheurs concoctent le système suivant : une caméra vidéo est connectée à un Apple 2. Sur son moniteur apparaît à la place du fond d'écran uniforme l'image du hiéroglyphe, agrandie par un macro-objectif aux dimensions de l'écran. A l'image produite par la caméra sont mélangés des signaux vidéo pour permettre la saisie des formes affichées sur l'écran.

L'explication du montage est donnée dans la brochure PSH (Programmation et sciences de l'homme) : *« Nous avons réalisé une carte interface qui se glisse dans l'un des « slots » de l'Apple, mais comme aucun connecteur ne porte les signaux vidéo, non plus que ceux de synchronisation, nous avons dû les prendre sur la carte de l'Apple. En sortie de notre carte interface, on trouve un câble relié à l'alimentation de la caméra et une prise pour le moniteur vidéo qui, dans ce mode d'utilisation, n'est plus raccordé à la prise standard de l'Apple. »* Montage, est-il précisé, qui *« ne gêne en rien les possibilités de fonctionnement de l'ordinateur »*.



▲ *Michaël H. présentant les hiéroglyphes de la paroi ouest de la pyramide de Pépi 1er.*

Le couplage caméra/ordinateur réalisé, il est possible, explique Michaël H., de déplacer un point lumineux sur l'écran, de le superposer à un endroit précis du contour de l'image. Pour enregistrer les coordonnées de ce point courant, on crée alors une interruption par l'intermédiaire du clavier de l'Apple.

Autrement dit, le logiciel de saisie écrit en Basic « étendu » – Applesoft – utilise les instructions de graphique haute résolution pour l'affichage, et les manettes permettant le déplacement de points lumineux sur l'écran.

A chaque position de ces deux manettes correspond un couple de valeurs numériques (x, y). Il suffit d'afficher un point sur l'écran à l'emplacement des coordonnées (x, y). A moins qu'on ne spécifie, en appuyant sur une touche du clavier, que l'on désire enregistrer ce point, un déplacement de la position des manettes génère un nouveau couple de coordonnées (x', y') : on efface alors l'ancien point (x, y) pour afficher le nouveau (x', y').

Translation, rotation, symétrie, etc., ce logiciel permet de dessiner n'importe quelle figure curviligne, et de constituer un catalogue de dessins que l'on peut afficher à la demande – en spécifiant son numéro d'ordre sur l'écran, éventuellement sur une table traçante. Plus de cinq cents figures de trente points en moyenne peuvent être enregistrées sur une seule disquette Apple.

Ce montage et ce logiciel ont permis, en 1980, d'enregistrer les trois cent quatorze hiéroglyphes de la paroi ouest de la pyramide de Pépi Ier. Temps moyen d'enregistrement par hiéroglyphe : cinq minutes.

Il y a, en Egypte, neuf pyramides « inscrites » qui comportent chacune douze parois dont l'impression nécessiterait environ (9 × 12) cent huit ans pour en obtenir un texte scientifique, selon les traditionnelles méthodes qui consistaient à en faire la saisie par des paléontologues, qui ne pouvaient qu'accumuler les longs séjours pour recopier sur leur papier millimétré le graphe millénaire des scribes.

Quinze jours suffisent, là où parfois une année de travail était nécessaire.

En quinze, voire trente minutes, la « récolte » pourra être imprimée : les typographes travaillaient – travaillent toujours – de deux à trois mois pour obtenir un tel résultat.

En dehors de ce travail de collecte et d'édition, ce sont de nouvelles possibilités d'introspection du champ linguistique qu'autorise l'informatique : elle permet ainsi d'afficher sur les consoles les divers sens des mots et les chaînes sémantiques, sans passer par le sisyphéen labeur qui consiste à s'user les doigts dans la consultation de milliers de fiches. Et parfois sans résultat, pour peu que la direction de départ soit erronée !

Mais cette légèreté n'est pas du goût de tous. Encore nombreux sont les sceptiques dans les sciences humaines. Pourquoi s'engloutir dans la manipulation de litanies de fichiers, purgatoires désuets ? La raison y trouve-t-elle toujours des excuses à ses douillets enlisements ? Amis papyrologues, les papyrus vous saluent bien. Bonsoir !

Pierre Bernard Soulier

Le TO7 au banc d'essai

L'ordinateur TO 7 est bâti autour du processeur 6809 (8 bits). Il possède une MEV de 22 Ko dont quatorze pour l'écran et huit pour l'utilisateur, et coûte 3 500 F ttc.

L'aspect du TO 7 est agréable à l'œil ; Thomson le présente comme un ordinateur familial. De ligne assez futuriste, il ira très bien dans un salon, près d'un téléviseur.

La partie frontale en matière plastique de couleur grise est légèrement inclinée, de ce fait la position des mains de l'utilisateur sur le clavier est reposante.

Monobloc, le TO 7 comprend l'unité centrale, le clavier et le lecteur de cartouches. C'est un appareil assez mince, mais large, car il comporte à l'extrême gauche le casier de chargement des cartouches programme, appelées encore « Mémo 7 ». Ces cartouches contiennent en mémoire morte tout le logiciel du TO 7 (sans lequel il ne peut rien faire). Malgré cela le radiateur est trop saillant à l'arrière droit de l'ordinateur et il faut prendre garde de ne pas s'accrocher la main (il est brûlant et ses arêtes vives piquent).

Et pourquoi toujours cette même couleur grise que l'on trouve sur des dizaines d'appareils ? De plus, la conception « tout en plastique » est-elle un gage de robustesse ?

Le clavier de type Azerty comporte cinquante-sept touches sensitives en gris (clair et foncé) sur fond noir.

Si l'on soulève une trappe au-dessus de ce clavier, on découvre l'emplacement du crayon optique, un organe d'entrée inhabituel qui, vous le verrez, intervient pour beaucoup dans l'attrait de cet appareil.

A droite de cette trappe se trouve une touche (il fallait appuyer dessus pour savoir que c'en était une !) appelée « initialisation » : c'est l'équivalent du RESET ; sur le côté droit du TO 7, l'interrupteur marche-arrêt et, sur le côté gauche, une prise DIN pour relier un magnétophone à cassettes. Sur la face arrière et de gauche à droite on peut voir : le cordon d'alimentation et le porte-fusible, un câble de raccordement à un téléviseur équipé de la prise Péritélévision et, juste au-dessus, une fiche d'alimentation 12 V pour un codeur-modulateur que l'on doit utiliser dans le cas où le téléviseur n'est pas muni de la fiche Péritélévision.

Viennent ensuite quatre connecteurs ; le premier pour l'extension mémoire et les trois autres pour des boîtiers d'extension (contrôleur de jeux, contrôleur de communication et interface-contrôleur de disquette).

Les dimensions du TO 7 restent raisonnables : longueur 45,5 cm, largeur 26 cm, hauteur maximale 7,5 cm pour un poids de 3,4 kg.

Après raccordement du cordon secteur et du câble Péritélévision, il suffit de mettre sous tension le téléviseur, puis le TO 7 (veiller à ce qu'une cartouche soit en place avant la mise sous tension). Un petit voyant rouge s'allume à l'avant dans le coin droit de l'ordinateur et un menu (en couleurs évidemment) apparaît sur l'écran. Soulignons qu'il n'est pas nécessaire de régler (ou de dérégler !) le téléviseur grâce à l'emploi extrêmement simple et pratique de la prise Péritélévision.

Le menu offre un choix dépendant de la cartouche enfichée. Pour le Basic, on peut soit le lancer, soit demander le réglage du stylo lumineux. La réponse s'effectue par le clavier ou en pointant avec le stylo la case affichée. La possibilité de régler la sensibilité du stylo est très intéressante, car certains paramètres peuvent varier d'un récepteur à l'autre. Il suffit simplement de pointer sur l'écran la case indiquée et le programme s'adapte automatiquement à la luminosité présente.

La sortie sur écran se fait donc sur un téléviseur couleurs : sa définition graphique est de 320 × 200 points, soit un affichage de vingt-six lignes de quarante caractères en mode texte. On peut obtenir jusqu'à huit couleurs.

Le clavier, de type Azerty (également disponible en Qwerty), comprend cinquante-sept touches dont quatorze touches de fonction. Ces touches sont sensitives mais manquent de sensibilité (ce qui risque de froisser les anglicistes et n'autorise pas une frappe très rapide). Les caractères passent en minuscules sur appui des touches CNT + ESPACE et un voyant en indique le mode. Il est possible d'accéder aux accents français, au prix, il est vrai, de

trois à cinq appuis successifs par symbole accentué... !

Une sonnerie interne au TO 7 fait entendre un « bip » lorsque l'on appuie sur une touche ; on peut régler l'intensité de ce signal au moyen de la commande volume du téléviseur. Soit, mais si l'on effleure la touche, il ne se passe rien, pas de « bip », et rien à l'écran ; il faut donc appuyer avec insistance ou frapper sur la touche, ce qui n'est pas très agréable. On aurait souhaité des touches réagissant à une faible impulsion. De plus, cette touche SHIFT n'est accompagnée d'aucun signal sonore et l'on ne sait si elle a été prise en compte qu'en regardant l'écran et le caractère affiché.

Parmi les touches importantes, nous avons CNT (contrôle), qui modifie le sens d'une touche et apporte donc une série de commandes supplémentaires. RAZ (Remise A Zéro) efface l'écran et remet le curseur en haut et à gauche (c'est le Clear-home français).

Ne cherchez pas les touches SHIFT, du moins sous ce nom ; elles sont représentées par une large touche avec un point jaune au milieu. Les touches STOP, EFF (effacement) et INS (insertion) ont une action qui varie selon le programme. En Basic par exemple, STOP est équivalent à BREAK et permet d'arrêter le déroulement d'un programme. A l'extrême droite du clavier, on remarque les quatre touches pour déplacer le curseur. L'habituelle touche ENTER ou RETURN s'appelle ici ENTREE et elle est jaune d'or !

Toutes les touches sont à répétition, ce qui signifie qu'en laissant le doigt sur l'une d'elles le code correspondant est généré continuellement. A l'usage, il s'avère parfois qu'elles ont aussi une fâcheuse tendance à « rebondir », c'est-à-dire à frapper plusieurs fois le même caractère sur un seul appui.

Conclusions partielles

- Présentation agréable.
- **Lecteur de cartouches programme.**
- **Clavier français accentué,** mais peu exploitable à des fins professionnelles.
- Un **organe d'entrée original : le crayon optique.**
- Radiateur de dissipation de la chaleur trop saillant.
- Clavier avec quelques rebonds.

▲ *Le clavier Azerty du TO 7.*

Le Basic de Microsoft mais adapté au TO 7

Le Basic de TO 7 n'est pas résident : il se charge à partir d'une cartouche « Mémo 7 » de 16 Ko. Il s'agit en fait d'une version de Microsoft (5.1 d'après le manuel et 1.1 d'après la cartouche et le libellé !), mais conçue pour le TO 7. Les commandes et les instructions sont donc bien connues sauf celles qui sont propres à cet appareil et c'est principalement sur cette catégorie que nous nous attarderons.

La commande SCREEN suivie d'un à trois chiffres permet de changer la couleur des caractères, du fond ou encore du pourtour de l'écran, par exemple SCREEN 0,2,6 fera écrire en noir sur fond vert avec un pourtour bleu clair.

Le TO 7 possède huit couleurs représentées par un chiffre (de 0 à 7). Restons dans le sommaire des couleurs et du dessin : c'est un ordinateur familial, nous le savons, et de ce fait il favorise beaucoup les dessins et leur animation. Rappelons que cet ordinateur a une résolution graphique de 320 × 200 points.

L'instruction P SET (numéro de colonne, numéro de ligne) a pour effet d'allumer un point sur l'écran (avec la couleur que l'on veut). En faisant boucler un programme, on pourra donc tracer des lignes horizontales et verticales.

L'instruction LINE parvient au même résultat mais, au lieu de travailler point par point, elle trace une ligne entre deux points. Vous pourrez ainsi dessiner un carré ou une ligne brisée.

L'instruction BOX dessine d'emblée un carré ou un rectangle à l'écran avec la couleur choisie ; BOXF a le même rôle, mais colorie entièrement la figure.

Il est aussi possible de se programmer son propre jeu de caractères avec l'instruction DEFGRS, qui permet de définir l'ensemble des points composant un caractère. Ce dernier sera affiché par PRINT GRS (x).

Venons-en à la grande originalité du Thomson : son crayon optique, et voyons quelles sont les instructions qui permettent de l'utiliser.

Les instructions INPUT PEN C, L et PSET (C,L) allument un point de colonne C et de ligne L quand on touche à l'écran avec le crayon optique. En faisant boucler le programme, on pourra dessiner une suite de points. Mais pour obtenir un trait continu, plus adapté au dessin, on recourra à l'instruction LINE :
LINE (A,B) - (C,D) joint les points (A,B) et (C,D) ; ajoutons à cela une boucle et miracle : vous pouvez maintenant dessiner sur votre téléviseur comme sur une feuille de papier ! Pas tout à fait aussi facilement, car il faut dessiner vite pour avoir de beaux arrondis.

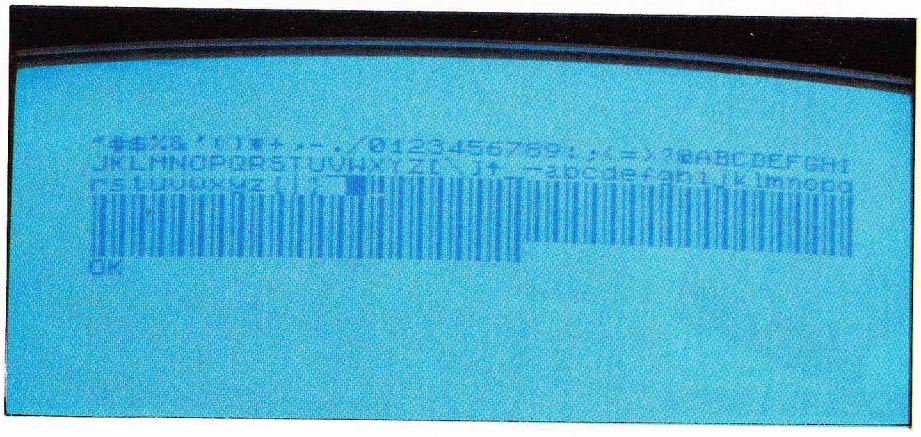
Toujours sur l'écran, nous pouvons doubler les caractères en hauteur et/ou en largeur grâce à l'instruction ATTR B. Cette même commande permet aussi de « marquer » (d'inhiber l'écriture sur) certaines zones de l'écran.

Examinons à présent les possibilités sonores de l'appareil : le TO 7 est muni d'un synthétiseur de sons qui permet de composer de la musique sur cinq octaves grâce à une notation très simple : les notes sont connues par leur nom (do, ré, mi, etc.), et l'instruction PLAY suivie des notes choisies permet de composer des musiques par un programme. Il est également possible de définir sept rectangles sur l'écran correspondant à la gamme et de jouer un air en « pianotant » avec le crayon optique comme sur un xylophone !

Voilà ! Dans l'ensemble, des possibilités sonores très classiques mais surtout des fonctions graphiques nouvelles et très étendues. De quoi s'amuser mais aussi créer des dessins et des compositions musicales dont le degré de sophistication dépendra de votre talent.

Quittons le domaine artistique pour examiner les autres instruc-

▼ *Affichage des caractères disponibles.*



tions non standards de ce Basic.

L'instruction CONSOLE réserve une fenêtre de travail sur l'écran (les deux premières lignes).

L'instruction PEN définit des zones rectangulaires sur l'écran ; elle est utilisée avec ON PEN... GOTO ou ON PEN... GOSUB, qui branchent le programme en fonction de la zone désignée par le crayon lumineux.

INPUTWAIT ressemble fort à un INPUT, mais avec un temps limité pour entrer les données. Notons encore quelques instructions particulières : comme UNMASK, qui démasque les zones de la fenêtre de travail qui avaient été masquées par ATTRIB ; MOTOROFF, qui provoque le défilement ou l'arrêt d'une bande sur cassette, EXEC, qui permet d'accéder à un sous-programme écrit en assembleur et déjà chargé mémoire centrale.

Pour tout le reste nous retrouvons un Basic Microsoft très complet. ERL, ERR, ON ERROR... GOTO, RESUME sont disponibles, de même que PRINT USING, IF... THEN... ELSE, PEEK et POKE, TRON et TROFF, etc. De plus, le Basic du TO 7 autorise des tableaux à deux dimensions.

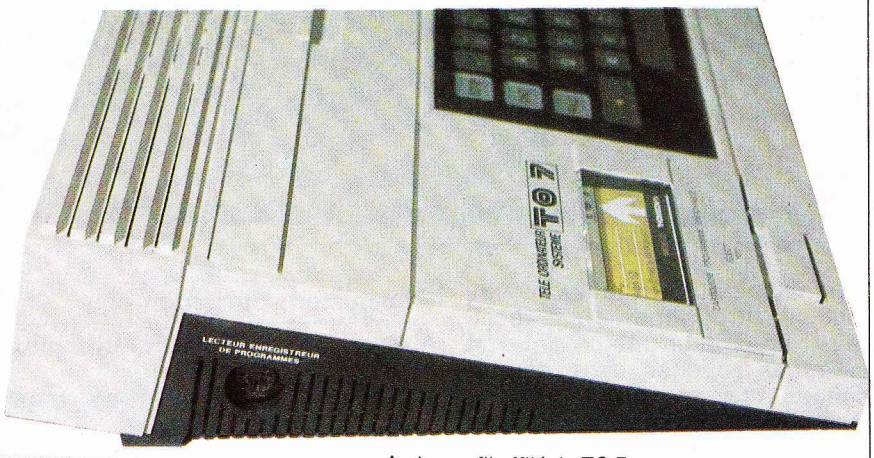
Et les fonctions mathématiques ? Disons qu'elles permettent les calculs les plus courants, mais qu'elles ne sont pas au grand complet : citons les absents : AN (X) pour donner, en radians, arctangente de X et l'instruction RANDOMIZE, qui permet de faire varier la séquence des nombres aléatoires.

Hormis l'absence de STRING $ (nombre de fois, chaîne) et de SPACE $ (X), les instructions travaillant sur les chaînes de caractères sont toutes là.

Le Basic du TO 7 permet de définir les types de variables (entrées, double ou simple précision, chaîne) : nous avons ainsi DEF SNG, DEFINT, DEF DBL et DEF STR (six chiffres en simple précision, seize en double). Quant à DEFUSR, il permet de définir le point d'entrée d'un sous-programme en langage machine.

En revanche, les instructions WHILE... WEND (qui exécute les instructions comprises entre ces deux bornes tant que l'expression logique est vraie) et SWAP (permutations des contours de deux variables) ne sont pas disponibles.

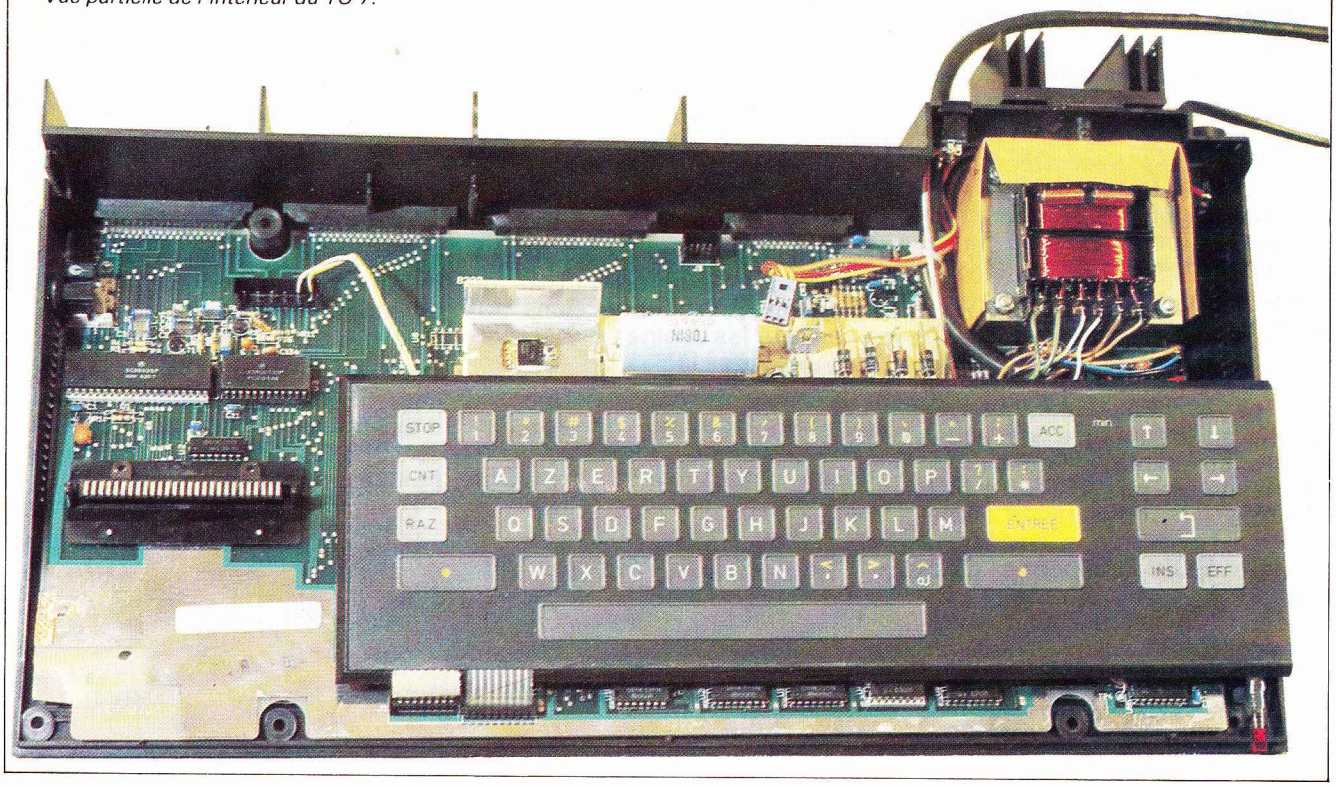
Le TO 7, animé par le processeur 6809, a une bonne vitesse d'exécution : le test habituel FOR I = 1 TO 10000 s'exécute en dix-huit secondes.



▲ *Le profil effilé du TO 7. Au premier plan on distingue le lecteur de cartouche Mémo 7. A gauche on peut voir la prise de raccordement du classique magnétophone à cassettes.*

◀ *Le crayon optique et son utilisation. Une image inhabituelle à l'écran. Le trait fin (encore hésitant) a été réalisé grâce au logiciel graphique Pictor, vendu séparément.*

Vue partielle de l'intérieur du TO 7.



Signalons aussi que la mise au point des programmes est facilitée par un éditeur très pratique : à l'aide des touches de fonctions fléchées, on se promène librement sur l'écran. Lorsque l'on modifie une ligne, il n'est pas nécessaire de reconduire le curseur jusqu'au bout avant de valider la ligne. Pendant l'édition ou l'entrée d'une ligne, on peut aussi utiliser les touches INS et EFF pour insérer ou effacer des caractères.

Les messages d'erreur, au nombre d'une vingtaine, sont bien expliqués dans le manuel Basic et nous évitent de tourner en rond en ne sachant que faire !

L'une des originalités de cet ordinateur est le stylo lumineux, qui offre un grand degré d'interactivité entre l'homme – l'enfant surtout – et la machine. Malheureusement, ce dispositif nous semble inadapté à l'écran de télévision couleurs, qui représente un certain danger pour la vue si l'on n'observe pas une distance raisonnable entre les yeux et l'écran (et qui ne sera sûrement pas respectée par l'enfant dessinant à l'aide du stylo lumineux). Pourtant, en fournissant cet objet en standard et en lui donnant une participation active dans la plupart de ses logiciels, Thomson ne semble pas avoir envisagé ce problème primordial sur lequel il nous a paru bon d'insister un peu.

Conclusions partielles

- **Basic de Microsoft, complet et rapide.**
- **Fonctions graphiques très étendues** (et **originales**) : Possibilités sonores intéressantes.
- **Bon éditeur « pleine page ».**
- Attention ! **Danger pour les yeux.**

Une documentation en français mais un peu trop succinte

La documentation du TO 7 est très disparate : elle comprend deux parties :
. un cahier de vingt pages de petit format intitulé « mode d'emploi »

Carte d'identité du matériel

Configuration de notre essai

- Système Thomson TO 7, modèle VC 90-001, numéro de série 900 Po 197.
- Un câble péritélévision intégré en sortie directe.

Présentation

- Le Thomson TO 7 est un ordinateur de table comprenant un clavier Azerty non accentué de cinquante-huit touches sensitives (quarante-cinq pour les signes, espaces et caractères, et treize pour les fonctions), un lecteur de cartouches programme enfichables, et un crayon optique. Construit autour d'un 6809, ce modèle comprend 8 Ko de MEV pour l'utilisateur, 14 Ko pour la vidéo et 6 Ko pour le moniteur. Les cartouches programme ont un maximum de 16 Ko de MEM.
- L'affichage sur un écran de télévision permet une résolution de 320 × 200 ou 25 lignes de 40 caractères, et huit couleurs.
- Les interfaces d'origine permettent de raccorder un magnétophone à cassettes et un téléviseur couleurs (avec prise péritélévision).

Accompagnement

- Une brochure de mode d'emploi et de présentation du matériel de vingt pages, en français.
- Le Basic Microsoft est vendu séparément sous forme d'une cartouche programme, dans un coffret contenant également le manuel d'introduction au Basic (prix : 800 FF ttc, dont 75 FF ttc pour le manuel Basic).

Prix
3 700 FF ttc.

Garantie
Un an, pièce et main d'œuvre.

et contenant les caractéristiques générales et quelques schémas (d'autres cahiers de même format seront réunis dans un classeur) ;

. un manuel de Basic de 186 pages ; c'est un véritable livre, publié aux éditions Cedic, qui s'adresse aux débutants ; il passe en revue un grand nombre d'instructions du Basic Microsoft (mais pas la totalité), avec de nombreux exemples et des dossiers humoristiques ; cet ouvrage, conçu de façon didactique et attrayante, rattrape un peu l'absence de documentation technique, mais ne la fait pas oublier.

Cette documentation est en français, mais on ne pourra s'empêcher de la trouver trop succinte.

Conclusions partielles

- **Bon ouvrage d'initiation au Basic.**
- **Documentation technique squelettique,** voire quasi inexistante.

Une mémoire vive de 28 Ko pouvant être portée à 32 Ko

Sept vis permettent de retirer le coffret supérieur du TO 7 et de faire apparaître les composants du système que l'on peut classer ainsi : la carte logique, le clavier relié à cette carte par deux petits câbles « film mince », l'alimentation (série) sur circuit imprimé logée sous le clavier et son transformateur (derrière le radiateur).

Le clavier est du type « film », moulé dans une plaque collée sur bâti de matière plastique. Le carter supérieur que nous venons de retirer comporte un ensemble mécanique (permettant l'introduction des cartouches mémoire), le stylo lumineux, la « sonnerie » (buzzer) et l'interrupteur d'initialisation. Il est relié à la carte principale par deux petits câbles enfichables.

Le stylo lumineux est constitué d'une cellule sensible à la lumière et d'un interrupteur miniature qui ferme un contact lorsque l'on exerce une pression sur le stylo.

La carte logique a la belle apparence d'un circuit multicouches avec sérigraphie et supporte soixante-dix circuits intégrés dont nous citerons les plus importants : MC6809 (processeur), MC6821 (adaptateur d'interface parallèle PIA), quelque chose à vingt-quatre pattes..., qui doit être une mémoire morte (il en faut bien une !), et un circuit quarante pattes (SC80826 Motorola) qui semble destiné à gérer les interfaces d'extension.

La mémoire vive est composée de quatorze boîtiers, soit une capacité globale de 28 K-octets. Attendu que 8 Ko sont seulement disponibles pour l'utilisateur (ils peuvent être portés à 32 en ajoutant une option extérieure), les 20 Ko restants doivent être réservés au logiciel et à la vidéo (page écran, générateur de caractères, attributs, etc.).

Contrairement à notre attente, la gestion vidéo n'est pas réalisée par un contrôleur spécialisé, mais par un système à base de composants discrets.

A gauche de la carte, un support de vingt-sept contacts est destiné à recevoir le connecteur des cartouches de logiciels, ces dernières n'étant ni plus ni moins qu'un circuit imprimé enfichable sur lequel est fixée une mémoire morte. Les extensions évoquées par le constructeur viendront directement s'enficher à l'arrière sur les quatre connecteurs de la carte logique prévus à cet effet.

Conclusions partielles

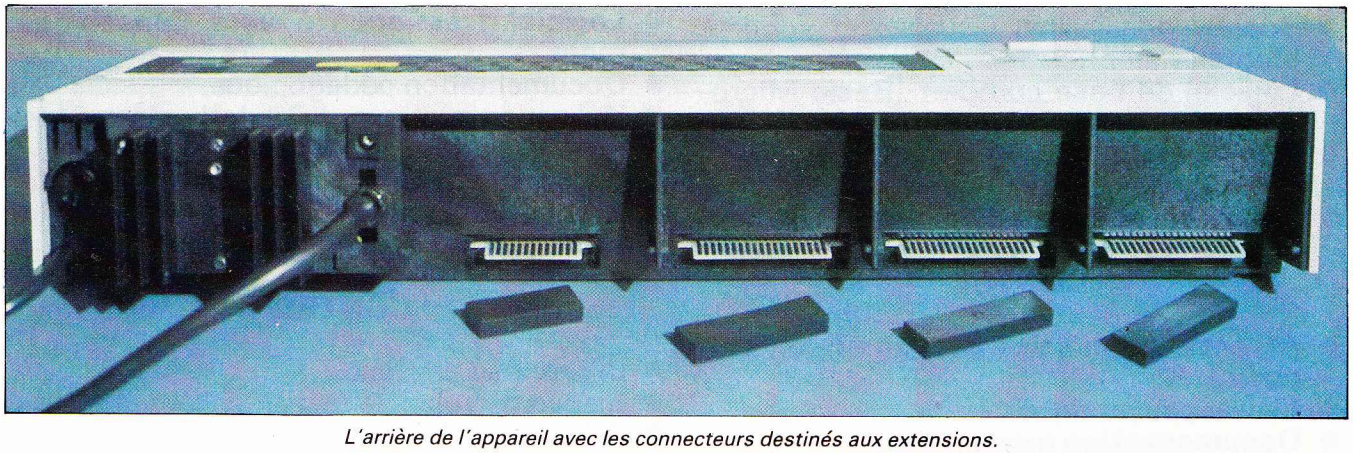
- **Belle réalisation technique,** pour laquelle le constructeur semble avoir voulu écarter tout risque « technologique ».

De nombreuses et diverses extensions prévues

Parmi les extensions matérielles, Thomson a prévu deux types d'imprimantes (thermique et à impact), un lecteur enregistreur de programmes sur cassettes (cassétophone) et sur disquettes 13 cm (jusqu'à quatre unités), des interfaces série et parallèle, des manettes de jeu et, pour ceux qui ne possèdent pas de téléviseur muni de la prise péritélévision, un codeur-modulateur Secam ou Pal permettant le raccordement direct sur la prise d'antenne. Néanmoins, nous ne conseillons pas l'usage de ce dispositif, qui est coûteux et loin d'apporter la qualité offerte par le dispositif standard. Les raccordements aux réseaux Antiope et Télétel sont aussi envisagés, du moins lorsque ceux-ci seront à la portée de tous.

Bien entendu, les programmes font également partie des promesses et des sociétés diverses s'y emploient actuellement. Le catalogue logiciel actuel nous propose déjà onze jeux sur « Mémo 7 » (cartouche mémoire) et une vingtaine de didacticiels en cinq collections : informatique, mathématiques, sciences, langues et français.

Nous avons pu tester l'une de ces cartouches : Pictor, qui en effet, par un menu affiché en bas de l'écran, de dessiner toutes sortes de figures en couleurs à l'aide du stylo lumineux qui vous donne la faculté d'« écrire » sur l'écran avec presque autant de facilité que sur une feuille de papier à dessin.



L'arrière de l'appareil avec les connecteurs destinés aux extensions.

conclusions

L'ordinateur français TO 7 de Thomson nous semble promis à un bel avenir : son prix relativement faible, sa conception simple et saine, son bon logiciel de base (Microsoft) et ses grandes possibilités d'extension, tant matérielles que logicielles, font qu'il se trouve très bien placé face à ses concurrents.

Le seul point faible pourrait être la petitesse de sa bibliothèque logicielle (dûe aussi à l'apparition récente de son processeur), mais la situation semble avoir été prise en mains pour que cela ne soit que de durée passagère.

Avec cet ordinateur, Thomson vise surtout les applications domestiques et l'enseignement (son clavier se prête peu aux applications professionnelles). Ce sera donc l'ordinateur familial par excellence.

Quant au stylo lumineux, il s'agit d'une bonne idée, mais il serait nécessaire qu'il ne soit pas fourni en standard et réservé à un emploi occasionnel, ou de l'utiliser en respectant une certaine distance (cours de l'enseignement), car son emploi avec un poste de télévision couleurs peut présenter des risques pour la vue, en particulier celle des enfants.

Thierry Courtois
Alain Pinaud
Jean-Pierre Brunerie

LE POUR ET LE CONTRE

UTILISATION PERSONNELLE

POUR

- **Rapport performances/prix intéressant.**
- Système esthétique et peu encombrant.
- **Mise en œuvre facile.**
- Bonnes possibilités visuelles et sonores (Péritélévision).
- **Logiciel de base complet (Basic Microsoft).**
- **Documentation pédagogique** (ouvrage d'initiation au Basic).
- **Nombreuses extensions possibles.**

CONTRE

- **Attention aux yeux avec le stylo lumineux !**
- Clavier peu pratique.
- **Documentation incomplète.**

UTILISATION DANS L'ENSEIGNEMENT

POUR

- **Système simple.**
- **Bonnes possibilités visuelles et sonores.**
- Gros caractères.
- **Logiciel très orienté vers l'enseignement.**
- Documentation pédagogique.
- Utilisation pratique du stylo lumineux (de loin).
- Logiciel de base à peu près complet.
- Nombreuses extensions matérielles.
- Faible prix sans récepteur couleurs.

CONTRE

- Clavier peu pratique.
- Risque de vol.
- Documentation incomplète.

TO 7

Le point de vue du constructeur

Avec le TO 7, Thomson a mis sur le marché, dès novembre 1982, un nouveau produit électronique dans la lignée des téléviseurs et des chaînes Hi-Fi ; il s'agit d'un ordinateur familial simple d'utilisation mais techniquement évolué, entièrement conçu, développé et fabriqué en France.

Plus qu'un ordinateur, c'est un véritable « système » ; l'unité centrale possède toute les interfaces pour recevoir les extensions et les périphériques, les logiciels sur mémoire morte (Mémo 7), sur cassettes et sur disquettes.

Avec ses logiciels de jeux et d'assistance à l'enseignement (microdidacts) et ses logiciels de gestion (familiale et professionnelle) le système TO 7 est complet et permettra de constituer progressivement « la chaîne » informatique et télématique de conception entièrement française adaptée à nos habitudes et à notre culture.

Nous insistons sur quelques points qui font du système TO 7 un ordinateur grand public aux possibilités tout à fait séduisantes.

1) La définition graphique du TO 7 est de 64 000 points (320 × 200). C'est à notre connaissance la meilleure définition dans cette gamme de produit. Cette caractéristique permet d'avoir une très grande qualité d'image et un graphisme réaliste et précis.

2) Le crayon optique intégré du TO 7 est l'outil de dialogues le plus convivial qui existe. L'utilisateur converse directement avec son téléviseur, sans la barrière de la frappe du message au clavier. De plus, ce crayon optique est programmable en Basic, ce qui donne aux amateurs une dimension supplémentaire pour écrire leurs programmes.

3) Le Basic du TO 7 est complet et puissant : c'est la version 5 de Microsoft. Il dispose d'une grande quantité d'instructions, la syntaxe est riche en possibilités, enfin l'éditeur « plein écran » offre une facilité supplémentaire pour la mise au point des programmes.

4) Le lecteur de Mémo 7 est intégré à l'unité centrale ; il se présente comme un lecteur de cassette. Il permet de lire... les Mémo 7, programmes écrits en assembleur et stockés sur mémoire morte. Ces programmes peuvent être des jeux, des didacticiels, des programmes de gestion mais aussi des langages. Basic est disponible sur Mémo 7, Logo le sera aussi dès avril prochain, d'autres langages sont à l'étude.

5) Vous vous interrogez sur la robustesse de l'habillage en plastique ; en fait, il s'agit d'une matière plastique appelée noryl renforcé (chargée de fibres en verre) dont la solidité a été testée. Les touches du clavier ne sont pas très sensibles mais nous l'avons voulu ainsi ; car, pour un usage grand public, il n'est pas fragile et évite les fausses manipulations. Il est vrai qu'il n'a pas été conçu pour la saisie.

Nous vous signalons que le TO 7 possède aussi les caractères semi-graphiques Télétel.

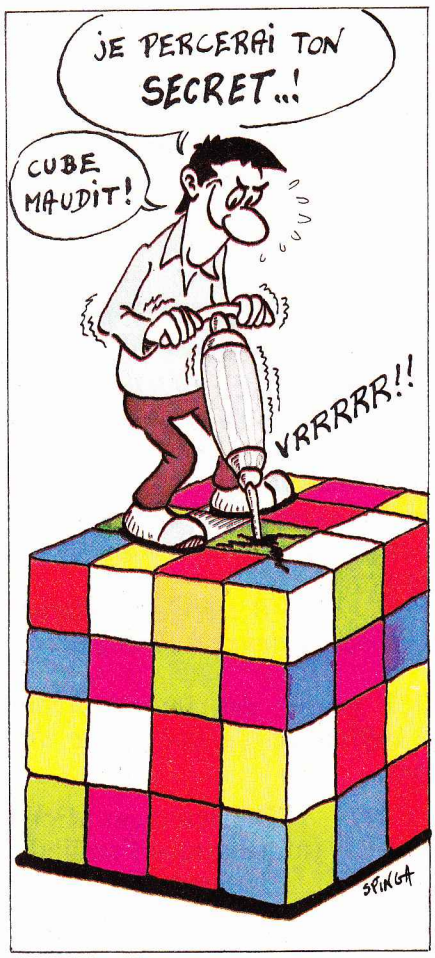
Nous estimons, pour répondre à L'Ordinateur Individuel, que le téléviseur ne peut être tenu pour responsable d'éventuels « maux de vue » pour les trois raisons suivantes :

- *les téléviseurs aujourd'hui ont fait beaucoup de progrès : stabilité des couleurs, qualité de convergence des tubes ;*
- *l'image fournie en RVB (mode moniteur) est d'une qualité bien supérieure à l'image télédiffusée ;*
- *enfin le TO 7 génère des images ayant 64 000 points de définition, ce qui donne des images très précises.*

Département Micro
Informatique Familiale
Thomson - S.D.R.M.
67 Quai Paul Doumer
92402 Courbevoie Cédex

le Rubik's cube ? un jeu d'enfant avec votre PC-1211 !

Apprenez à votre ordinateur de poche PC-1211 à remettre en ordre le fameux Rubik's cube. Voici un programme qui utilise toutes les ressources de votre petite machine favorite et qui vous permettra, de plus, d'améliorer vos performances !



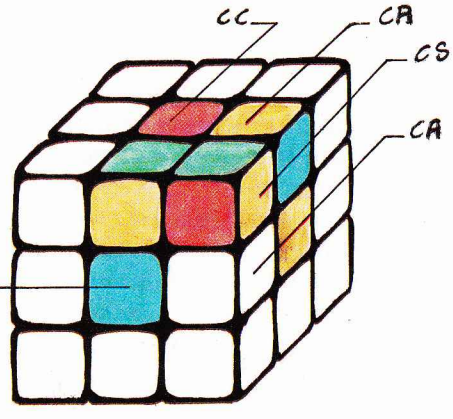
Si vous êtes un amateur du Rubik's cube et si vous possédez une Sharp PC-1211 avec interface cassette et imprimante, le programme présenté permettra à votre machine de remonter le cube, en partant d'une position quelconque.

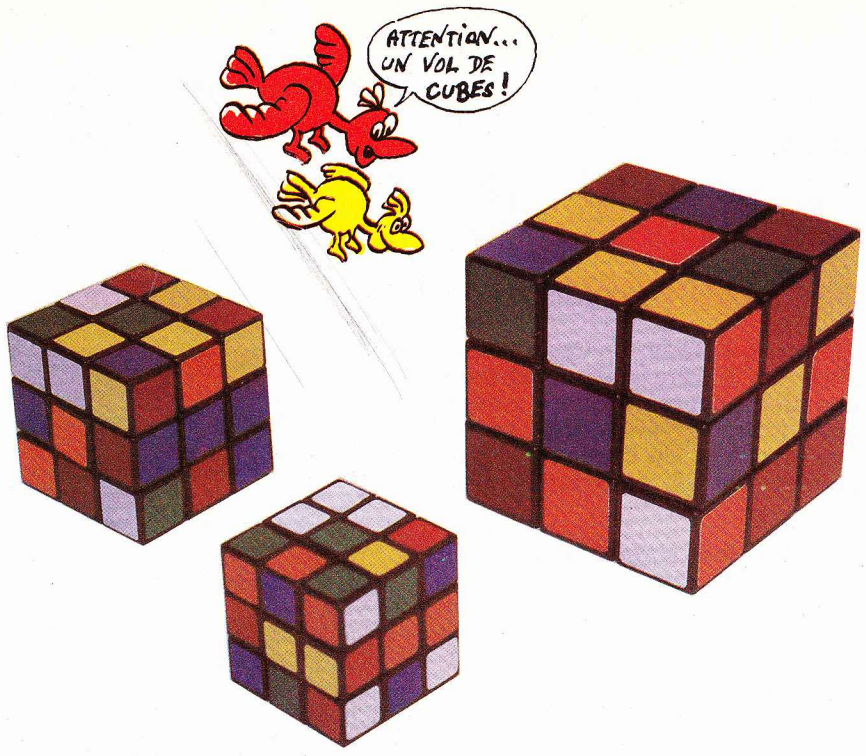
Sur un ordinateur individuel, un tel programme ne présente pas de difficulté, mais pour votre ordinateur de poche c'est une performance méritoire. Hélas, n'espérez pas battre avec votre PC-1211 un record de vitesse : il lui faudra, en moyenne, une demi-heure pour remonter le cube — exploit qui lui assurerait la dernière place au championnat de Rubik's cube ! En tout cas, si vous épluchez ce programme, vous vous rendrez compte des possibilités de votre machine et vous approfondirez peut-être vos connaissances sur le cube...

Le programme en question nécessite 4 247 pas (en comptant pas de programme et mémoires de données). Cela dépasse évidemment la capacité du Sharp-TRS de poche, aussi ce programme est-il découpé en trois parties, qui sont enchaînées l'une à l'autre grâce à l'instruction CHAIN dont la puissance apparaîtra clairement à cette occasion.

Avant de commenter ce programme, rappelons quelques faits simples concernant le cube lui-même (que l'on se rassure cependant, une personne manipulant le cube pour la première fois pourrait fort bien utiliser le présent programme, sans avoir à en saisir tous les détails).

Ce cube est théoriquement formé de vingt-sept petits cubes : six n'ont qu'une facette visible (ce sont les cubes centraux ou CC), douze ont deux facettes visibles (ce sont les cubes arêtes ou CA), huit ont trois facettes visibles (ce sont les cubes sommets ou CS) et le vingt-septième (à l'intérieur) n'existe pas. Les CC ne quittent jamais leur place, et servent ainsi de repère : la « face rouge » du cube sera celle dont la facette centrale est rouge.





Le programme des pages suivantes comprend en fait un fichier de données (baptisé CUBE) et trois programmes nommés respectivement CUBE 1, CUBE 2 et CUBE 3. Il nous faut d'abord écrire, puis enregistrer sur cassette ces quatre parties dans l'ordre : fichier CUBE, CUBE 1, CUBE 2 et CUBE 3. C'est là un travail long et fastidieux !

Pour vous aider, quelques conseils. Tout d'abord, si votre lecteur de cassette a un compteur, n'oubliez pas d'inscrire la place exacte de chaque partie. Pour vérifier qu'une partie a bien été écrite, le plus simple est d'en imprimer la liste et de la comparer à la liste ci-jointe. Enfin, une vérification très simple (dont l'efficacité vaut bien celle de la preuve par neuf !) consiste à voir si le programme que vous venez d'écrire occupe bien le nombre de pas requis ! A cet égard, les renseignements ci-après peuvent être utiles.

. Le fichier CUBE comprend les mémoires souples A(35) à A(112) ; ainsi A(35) = 12, etc.

. CUBE 1 nécessite 718 pas de programme ; si vous appuyez sur MEM Enter, il doit donc apparaître :

706 Steps 88 Memories.

Prenez le risque ! Mélangez soigneusement votre cube...

Ce programme utilise 86 mémoires souples :

A(27) à A(112)

ce qui laissera 18 pas disponibles.

. CUBE 2 nécessite 733 pas de programme ; vous devrez lire :

691 Steps 86 Memories.

Ce programme a besoin des mêmes mémoires souples que CUBE 1, vous laissant 3 pas disponibles !

. CUBE 3 nécessite 1 140 pas de programme ; vous devez lire :

284 Steps 35 Memories.

Ce programme emploie 35 mémoires souples [A(27) à A(61)], vous laissant 4 pas disponibles.

. Le fichier CUBE ne sera utilisé que dans les programmes CUBE 1 et CUBE 2.

Vous avez terminé l'enregistrement ? Vous voici prêt à commencer ! Placez votre cassette au début du fichier (faites NEW, c'est plus prudent), vérifiez vos branchements, en particulier l'imprimante doit être en mode PRINT : n'oubliez pas d'appuyer deux fois sur ON, réglez le niveau d'écoute au maximum, enclenchez la touche PLAY, et en route ! Voici ce que vous avez à faire.

1) Transférez le fichier CUBE grâce aux instructions suivantes (en mode RUN) :

INPT # CUBE Enter.

2) En mode PRO écrivez le programme :

1 CHAIN CUBE 1,

puis, en mode RUN, faites RUN 1 Enter. Le programme CUBE 1 est alors transféré de la cassette à l'ordinateur. Ce transfert effectué, l'exécution commence (automatiquement).

Le PC-1211 imprime : ANNONCEZ LES COULEURS, puis FACE AVANT ? et s'arrête, attendant un renseignement.

3) Mélangez votre cube, puis posez-le devant vous, de façon arbitraire. Indiquez alors la couleur de la face avant. Ainsi, dans l'exemple de la figure 1, faites : BLEU Enter.

Votre ordinateur de poche enregistre les six couleurs de face

L'OP imprime ensuite FACE DROITE ?, faites alors : JAUNE Enter, et opérez ainsi pour les six faces. Votre OP enregistrera ainsi les six couleurs de face (mémoires A$,..., F$) ; cette phase est nécessaire, car la répartition des couleurs des faces change d'un cube à l'autre.

4) Cela fait, votre machine vous demande de lui indiquer (successivement) les positions des douze CA. Ainsi lorsqu'elle imprimera JAUNE VERT (cf. figure page précédente) et affichera : ?, faites : BLEU Enter, puis ROUGE Enter.

Cela signifie que la facette jaune du CA « jaune vert » se trouve sur la face bleue du cube et que sa facette verte est sur la face rouge du cube. Noter que l'ordre est important : dans l'exemple de la figure de la page précédente, pour le CA JAUNE ROUGE, vous devrez faire :

ROUGE Enter puis JAUNE Enter.

Suspense, l'OP indique : « début de la solution » !

5) L'étape 4 achevée, le PC-1211 vous demande de lui indiquer les positions des huit CS, suivant le même principe que précédemment ; lorsque l'on imprimera (par exemple):

VERT JAUNE ROUGE, et affichera ?, faites :

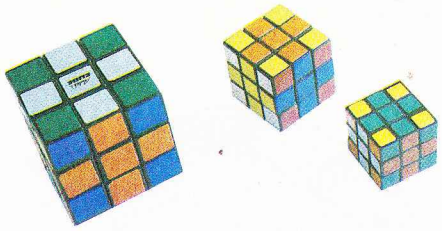
ROUGE Enter (affichage : ?), puis :

JAUNE Enter (affichage : ?), et enfin :

BLEU Enter.

Opérez de même pour les huit CS.

6) Votre OP connaît maintenant l'état exact de votre cube, cela lui suffit. Il imprime alors : « début de la solution », il enchaîne avec CUBE 2, le transfère à l'ordinateur puis en démarre l'exécution. A la fin de CUBE 2, il enchaînera avec CUBE 3, le transfèrera et l'exécutera. Cela fait, après quelques BEEP, il s'arrêtera en vous exprimant (imprimant) sa satisfaction.



7) Après l'étape 5) vous n'avez plus rien à faire : absentez-vous une demi-heure. A votre retour, vous trouverez imprimées toutes les rotations de faces qui vous permettront de remonter le cube.

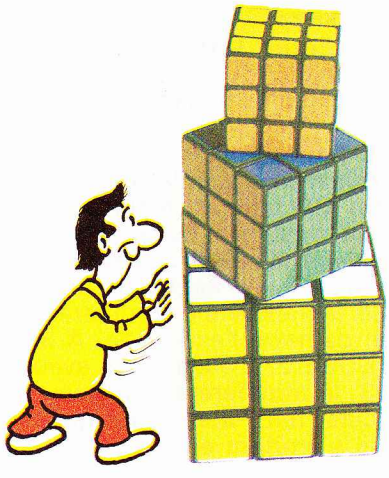
Ainsi :
1 BLEU (ou simplement BLEU) vous dit de faire subir à la face bleue du cube un quart de tour, dans le sens des aiguilles d'une montre (à main droite) ;
— 1 BLEU désigne le même quart de tour, mais en sens inverse.

Attention ! Une erreur de manipulation et l'on repart à zéro

Pour des raisons de place, il n'est pas possible d'analyser dans cet article ce programme en détail. Voici simplement quelques indications.

. Le programme CUBE 1 a pour seul but d'indiquer à l'ordinateur la position de départ.
. Le programme CUBE 2 s'occupe de remettre successivement à leurs places les douze CA (bien entendu l'ordinateur doit calculer aussi les déplacements des CS). Pour y arriver, on utilise une succession de sept rotations de faces que l'on appellera générateur.

Deux remarques importantes :
— les générateurs sont séparés



FICHIER CUBE

```
35.:12.
36.:13.
37.:14.
38.:15.
39.:23.
40.:24.
41.:26.
42.:35.
43.:36.
44.:45.
45.:46.
46.:56.
47.:363537013.
48.:353738132.
49.:383635201.
50.:-353941045.
51.:-403539104.
52.:394140451.
53.:-364243264.
54.:-433936402.
55.:-443740315.
56.:-404544573.
57.:-464238623.
58.:373836320.
59.:383635201.
60.:393642026.
61.:374045157.
62.:394140451.
63.:-364243264.
64.:-433936402.
65.:384446376.
66.:454143546.
67.:-464238623.
68.:373836320.
69.:394140451.
70.:374045157.
71.:414035510.
72.:423844237.
73.:424339640.
74.:-443740315.
75.:-404544573.
76.:-444642762.
77.:-433936402.
78.:-404544573.
79.:414346467.
80.:423844237.
81.:424339640.
82.:384446376.
83.:454437731.
84.:-444642762.
85.:414035510.
86.:394140451.
87.:424339640.
88.:-433936402.
89.:454437731.
90.:454143546.
91.:434645675.
92.:414035510.
93.:424339640.
94.:414346467.
95.:454437731.
96.:-404544573.
97.:464541754.
98.:424339640.
99.:414346467.
100.:454437731.
101.:454143546.
102.:464541754.
103.:424339640.
104.:-444642762.
105.:464541754.
106.:-464238623.
107.:-444642762.
108.:464541754.
109.:434645675.
110.:454437731.
111.:-444642762.
112.:464541754.
```

```
**************

    CUBE 1

  2:PRINT "ANNON
    CEZ LES COUL
    EURS:"
  4:INPUT "FACE
    AVANT?";A$
  6:INPUT "FACE
    DROITE?";B$
  8:INPUT "FACE
    DU HAUT?";C$
 10:INPUT "FACE
    DU BAS?";D$
 12:INPUT "FACE
    GAUCHE?";E$
 14:INPUT "FACE
    ARRIERE?";F$
 16:FOR W=35TO 4
    6
 18:G=INT (A(W)/
    10),H=A(W)-1
    0G:PRINT A$(
    G),A$(H):
    INPUT U$,T$
 20:GOSUB "A":I=
    Q:U$=T$:
    GOSUB "A":V=
    1:IF I>QLET
    V=-1,P=I,I=Q
    ,Q=P
 22:T=I*(9-I)/2+
    Q+29,T=T+(I=
    5)-(I<4)*(Q+
    I>7),A(W)=VT
    :NEXT W
 26:FOR W=0TO 7
 28:G=INT (W/4),
    H=INT (W/2-2
    G),I=W-4G-2H
    ,G=1+5G,H=2+
    3H,I=3+I
 30:PRINT A$(G);
    " ";A$(H);"
    ";A$(I)
 32:INPUT U$,T$,
    S$:GOSUB "A"
    :Ü=Q:U$=T$:
    GOSUB "A":P=
    Q
 34:U$=S$:GOSUB
    "A":U=Q
 36:L=0:IF L>3
    LET L=7-L,U=
    2^(Q-4)
 38:M=P:IF M>3
    LET M=7-M,U=
    U+2^(P-4)
 40:N=Q:IF N>3
    LET N=7-N,U=
    U+2^(Q-4)
 42:Z=SGN ((40-5
    L)*(4P-5M)*(
    40-5N)*(M-L)
    *(N-M)*(N-L)
    )
 44:V=Z*((N=1)-(
    M=1)),V=V-3*
    INT (V/3)
 46:A(27+W)=U+V/
    10:NEXT W:
    PRINT "DEBUT
     DE LA SOLUT
    ION:"
 48:CHAIN "CUBE
    2"
 50:"A":FOR X=1
    TO 6
 52:IF A$(X)=U$
    LET Q=X
 54:NEXT X:
    RETURN
```

```
**************

    CUBE 2

  2:W=35
  4:IF W=46GOTO
    28
  6:U=ABS (A(W))
    :IF U=WLET W
    =W+1:GOTO 4
  7:GOSUB "D":
    GOTO 6
  8:"D":V=A(W*(9
    1-W)/2+U-969
    ),G=ABS V:
    GOSUB "A"
 10:P=10P+Q,N=10
    N+M,L=10L+K
 12:T=35,G=P:
    GOSUB "B":S=
    Y,G=N:GOSUB
    "B":R=Y,G=L:
    GOSUB "B":Q=
    Y
 14:A(Q)=-A(Q),A
    (S)=N*SGN (V
    A(S)),A(R)=-
    P*SGN (VA(R)
    )
```

```
16:T=27,G=J:
   GOSUB "B":S=
   Y,G=1:GOSUB
   "B":R=Y,G=H:
   GOSUB "B":Q=
   Y
 8:FOR Y=8TO 10
20:T=A(A(Y+9)
   (10-Y)/10,V=
   INT T,Z=10*(
   T-V),Z=Z-3*
   INT (Z/3),A(
   A(Y+9))=V+Z/
   10:NEXT Y
22:A(S)=A(S)+H-
   J,A(Q)=A(Q)+
   J-H,T=7-H-J:
   GOSUB "C":Q=
   S,T=J-I:
   GOSUB "C":R=
   S,T=I-H
24:GOSUB "C":
   BEEP 2:PRINT
   A$(Q):PRINT
   -1;A$(R):
   PRINT -1;A$(
   Q):PRINT -1;
   A$(S)
26:PRINT A$(Q):
   PRINT A$(S):
   PRINT A$(R):
   PRINT "
   *****":
   RETURN
28:W=35
30:IF W=46GOTO
   38
32:U=W+1:IF A(W
   )LET W=W+1:
   GOTO 30
34:IF A(U)LET U
   =U+1:GOTO 34
36:GOSUB "D":
   GOSUB "D":
   GOTO 32
38:FOR Y=35TO 1
   12
40:A(Y)=0:NEXT
   Y:CHAIN "CUB
   E 3"
42:"A":FOR X=8
   TO 16
44:A(X)=G-10*
   INT (G/10),G
   =INT (G/10):
   NEXT X:
   RETURN
48:"B":Y=T
50:IF INT (ABS
   (A(Y)))=G
   GOTO 54
52:Y=Y+1:GOTO 5
   0
54:RETURN
```

```
56:"C":S=3.5-
   SGN (T)*(0.5
   +LOG (ABS (T
   ))/LOG 2):
   RETURN
```

CUBE 3

```
 2:G=142011,H=3
   60262,I=1232
   01,J=240450,
   K=235100,L=3
   16402,M=6446
   71,N=-122312
 4:O=131321,P=2
   14512,Q=4631
   52,R=546230,
   S=427310,T=R
   ,U=-314621,V
   =415732
 6:W=322640,X=5
   17622,Y=M:
   FOR Z=7TO 25
 8:A(Z+28)=A(Z)
   :NEXT Z:G=A(
   50),H=-51673
   1,I=533760,J
   =-646452,K=A
   (41),L=63754
   1
10:M=626751,N=-
   635762:FOR Z
   =7TO 14
12:A(Z+47)=A(Z)
   :NEXT Z:FOR
   W=27TO 32
16:V=INT (A(W))
   :IF V=W-27
   GOTO 32
18:G=A(W*(67-W)
   /2+V-506),V=
   SGN G,G=VG:
   GOSUB "A"
20:G=I:GOSUB "B
   ":P=Y,G=J:
   GOSUB "B":Q=
   Y,G=K:GOSUB
   "B":R=Y
22:A(P)=A(P)+J-
   I+H/10,A(Q)=
   A(Q)+K-J+(3-
   H-V)/10,A(R)
   =A(R)+I-K+(3
   +V)/10
24:FOR Y=16TO 1
   8
26:S=A(A(Y)),T=
   INT S,U=10*(
   S-T),U=U-3*
   INT (U/3),A(
   A(Y))=T+U/10
   :NEXT Y
28:Y=7-L:BEEP 2
   :PRINT V;A$(
   M):PRINT V;A
   $(L):PRINT -
   V;A$(M):
   PRINT -V;A$(
   Y)
```

```
30:PRINT V;A$(M
   ):PRINT -V;A
   $(L):PRINT -
   V;A$(M):
   PRINT V;A$(Y
   ):PRINT "
     *****":
   GOTO 16
32:NEXT W
34:FOR W=27TO 3
   4
36:T=A(W),A(W)=
   10*(T-INT T)
   :NEXT W
38:M=2631,N=462
   3,O=1233,P=4
   217,Q=6535,R
   =6327,S=5137
40:FOR W=27TO 3
   3
42:V=A(W),A(W)=
   0:IF V=0GOTO
   54
44:G=A(W-14):
   GOSUB "A":G=
   V+A(H+27),A(
   H+27)=G-3*(G
   >2)
46:IF V=2LET U=
   K,K=7-J,J=7-
   U,I=7-I
48:BEEP 2:PRINT
   A$(K):PRINT
   -1;A$(J):
   PRINT -1;A$(
   K):PRINT -1;
   A$(I):PRINT
   -1;A$(J)
50:Y=7-J:PRINT
   A$(I):PRINT
   -1;A$(Y):
   PRINT -1;A$(
   I):PRINT A$(
   J):PRINT A$(
   I)
52:PRINT A$(K):
   PRINT A$(J):
   PRINT -1;A$(
   K):PRINT A$(
   Y):PRINT "
     *****"
54:NEXT W:BEEP
   8:PRINT "JE
   SUIS GENIAL,
   NON?":END
56:"B":Y=27
58:IF INT (A(Y)
   )=GGOTO 62
60:Y=Y+1:GOTO 5
   8
62:RETURN
64:"A":X=8
66:A(X)=G-10*
   INT (G/10),G
   =INT (G/10):
   IF GLET X=X+
   1:GOTO 66
68:RETURN
```

les uns des autres dans l'impression de la solution : cela en rend la lecture plus commode ;
— à la fin de CUBE 2, les CA doivent être rangés, c'est-à-dire que chaque face doit présenter une « croix » unicolore.

. Le programme CUBE 3 ne s'occupe plus que des CS, qu'il doit successivement placer (un à un) puis ranger. Ce programme est plus simple et plus rapide que CUBE 2.

. A chaque instant, les positions des CS sont stockées dans les mémoires A (27) à A(34), celles des CA dans les mémoires A(35) à A(46) (sauf dans CUBE 3).

Miracle ! Mais oui, c'est vrai, le cube est remonté !

. Le fichier CUBE sert essentiellement dans CUBE 2 : à un moment donné, il permet à l'ordinateur de choisir le générateur à effectuer et fournit toutes les indications utiles concernant ce générateur.

. Une erreur de manipulation ou une erreur dans la description de l'état du cube vous obligera à repartir de zéro ! En revanche, si le transfert de l'un des programmes (par exemple CUBE 3) n'est pas effectué correctement — cela arrive ! —, ce n'est pas grave ; recommencez au début de ce programme en écrivant (mode PRO) :

1 CHAIN " CUBE 3 " Enter,

puis en faisant (mode RUN) RUN 1 Enter.

Et maintenant, bon courage !

Emmanuel Halberstadt

l'OI ne fait pas le moine

Un village du sud de la France, semblable à des milliers d'autres. Un village où la vie et les traditions se perpétuent, au rythme des saisons. Ici, pourtant, la tradition et l'informatique font bon ménage, on pourrait même dire « font bon mariage », puisque Monsieur le Curé gère les affaires de la paroisse à l'aide d'un petit ordinateur. Comment le Père H. en est-il venu là ? C'est toute une histoire.

Père H. : « J'ai 57 ans. Ma passion pour l'informatique a été tardive : je n'y suis venu que par curiosité intellectuelle. »

L'Ordinateur Individuel : *Qu'entendez-vous par là ?*

« Eh bien, pendant mes études secondaires et durant ma formation au séminaire, j'ai toujours été très intéressé par les mathématiques et la physique. Je n'étais d'ailleurs pas le seul. Plusieurs de mes camarades séminaristes partageaient cet intérêt pour les sciences exactes. »

A voir les appareils électroniques qui nous entourent, il semblerait que vous n'en soyez pas resté là !

« Effectivement, vers 1950 j'ai suivi des cours d'électronique par correspondance et j'ai monté mon premier poste de TSF à lampes. »

Vous avez obtenu de bons résultats ?

« Bien sûr ! D'ailleurs, j'ai continué : je me suis attaqué à un poste en modulation de fréquence et j'ai réalisé ma première télévision qui me servait accessoirement d'oscilloscope. »

Et après cela ?

« Je me suis consacré à ma paroisse sans trop investir dans mes loisirs scientifiques. Je n'y suis revenu qu'en 1970 pour réaliser le montage d'une télévision couleurs. »

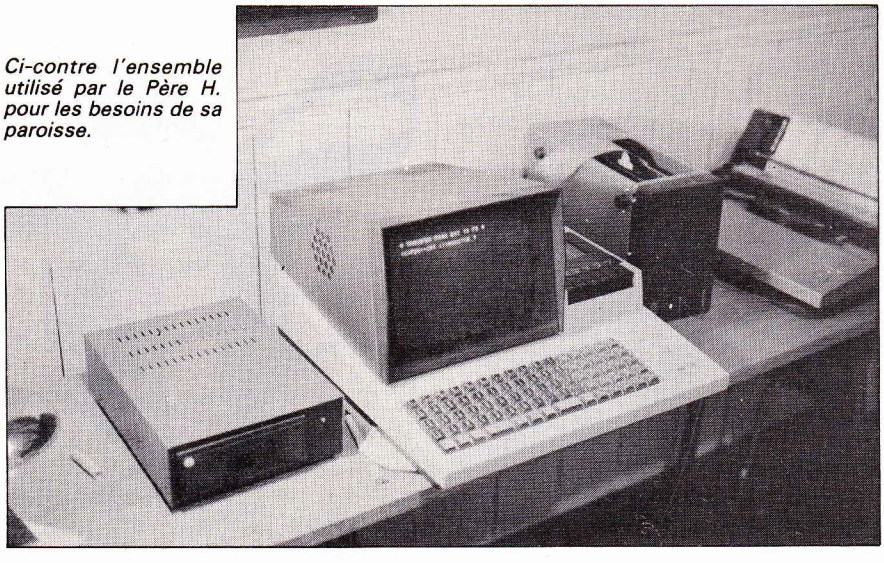
Cela a dû vous demander un certain investissement ?

« Oh non ! J'ai exclusivement utilisé des pièces de récupération de différentes marques, ce qui m'a d'ailleurs posé pas mal de problèmes. »

Lesquels ?

« Tout simplement l'accord entre les différents composants, qui n'était pas évident à réaliser. En particulier, certains transistors n'étaient pas assez puissants et je n'avais pas d'instruments de mesure UHF, ce qui m'obligeait à tâtonner. Mais enfin ma télévision couleurs fonctionne bien. »

Ce sont là toutes vos réalisations ?



Ci-contre l'ensemble utilisé par le Père H. pour les besoins de sa paroisse.

« Non. En 1975, j'ai fabriqué ma chaîne hi-fi de A à Z. J'ai dessiné et réalisé tous les circuits imprimés. Ma chaîne a de bons résultats en temps de montée et en bande passante ; de plus, je n'ai aucun bruit parasite. »

Et l'informatique dans tout ça ?

« Tout simplement un ami m'en a parlé et ma curiosité l'a emporté. »

Vous vous êtes « lancé » ?

« Oui. Tout d'abord, je me suis renseigné puis je me suis acheté un Sharp PC-1221 en 1981. En utilisant la notice et certaines publications (dont L'OI), je suis arrivé à apprendre le Basic. »

Vous n'avez donc pas disséqué votre Sharp pour comprendre comment il fonctionne ?

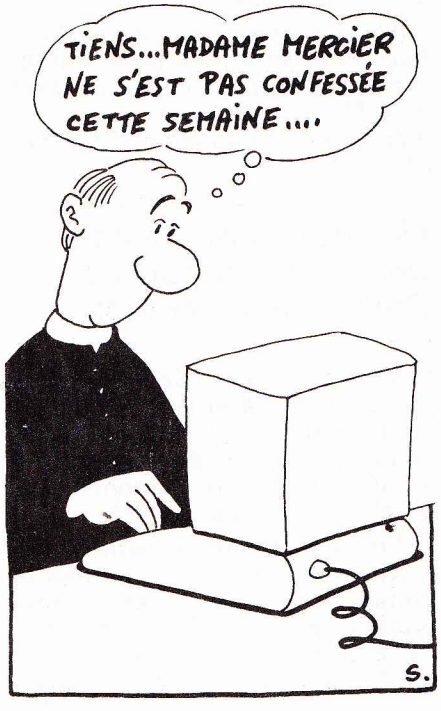
« Non. Je me suis dit qu'il valait mieux comprendre l'utilisation de l'ordinateur avant toute tentative d'en construire un. Mais maintenant que je sais comment cela fonctionne, je crois que je m'attaquerai bientôt à la construction d'un ordinateur, pour renouer avec ma passion de l'électronique. »

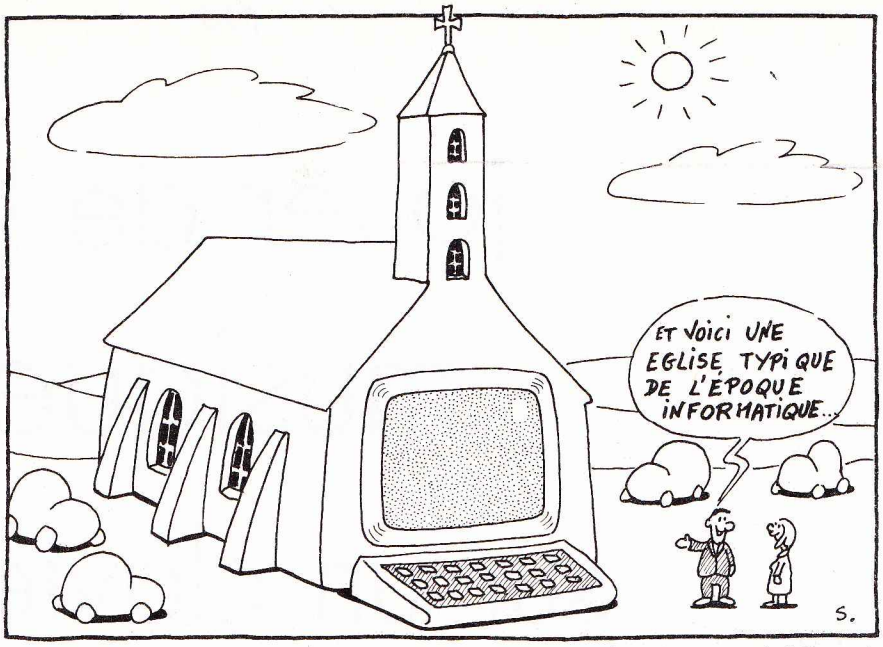
A voir le matériel de cette pièce, vous avez franchi une étape de plus.

« Effectivement. Le PC-1211 était trop limité. Alors je me suis acheté un MZ-80 K un mois plus tard, une unité de disquettes standard et une imprimante P3. »

Comment utilisez-vous ce matériel ?

« D'abord, j'ai constitué un fichier d'adresses pour le journal paroissial. J'ai eu des problèmes avec le tri. Au début, pour cinq cents adresses il me fallait sept heures, puis deux heures, puis vingt minutes ; maintenant je suis arrivé à deux minutes en utilisant une méthode logarithmique. »





Quels sont les messages sonores de fin de traitement ?

« Ce sont des petites mélodies des Beatles et autres. Ça m'amuse et puis c'est pratique. »

Que faites-vous d'autre ?

« Je gère mes cours de catéchisme. J'ai un fichier des enfants avec leurs dates de naissance, de baptême, de confirmation et de communion, ainsi que leur classe à l'école et le cours de catéchisme qu'ils suivent. Ça me permet de m'y retrouver car, vous savez, il y a tellement d'enfants que parfois je m'y perds. »

Vous l'utilisez comment ?

« Je trie des listes, par cours, par année de naissance, par classe à l'école et je fais des tris croisés : c'est très pratique pour avoir des informations rapides. »

Vous faites autre chose avec votre OI ?

« Bien sûr ! La comptabilité de ma paroisse est informatisée. J'ai un journal des recettes et dépenses, un traitement de mise à jour des postes de comptabilité générale et un détail des écritures pour chacun de ces postes. D'ailleurs, j'envoie très régulièrement un état de comptabilité sur liste à l'évêché. »

Et cela ne pose pas de problème ?

« Bien sûr que non ! L'imprimante écrit bien et puis c'est beaucoup plus détaillé que ce que je leur transmettais avant. »

Au point de vue langage, qu'utilisez-vous ?

« J'ai laissé tomber le Basic Sharp qui est trop limité ; j'utilise maintenant un Basic commercialisé par une société de la région lyonnaise. Je me suis mis à l'assembleur aussi. La capacité mémoire du MZ-80 K est tellement restreinte que je dois utiliser des tas de routines en assembleur, qui me font gagner de la place et du temps. »

Vous n'avez pas trop de problèmes avec l'assembleur ?

« Oh il suffit de s'y mettre ! Et puis, j'ai un utilitaire de conversion décimal-hexadécimal et binaire sur le PC-1211. Mon seul vrai problème est la conversion en langage machine que j'effectue à partir de la cassette. Ça irait beaucoup mieux si j'avais un deuxième lecteur de disquette mais c'est encore trop coûteux pour moi. »

Que souhaiteriez-vous maintenant ?

« Dans l'ordre : un deuxième lecteur de disquette, un compilateur Basic pour Sharp, un assembleur sur minidisquette. »

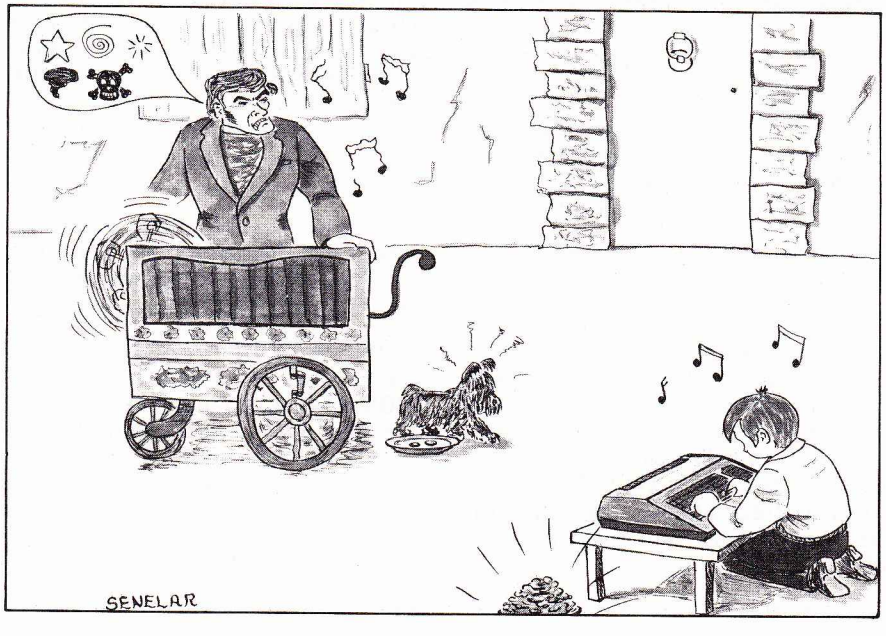
Ce sont là tous vos désirs ?

« En fait, un Apple avec carte 6809 me comblerait, mais il ne faut pas rêver ; aussi je crois que je vais bientôt me construire un ordinateur en kit. »

Propos recueillis par Arnaud Debreuil

jouer de la musique sur un clavier... Qwerty

On dit que la musique adoucit les mœurs..., mais c'est aussi vrai pour l'informatique. Pas besoin d'être premier prix de Conservatoire de Musique pour utiliser le programme que nous vous proposons (particulièrement bien conçu, il faut le dire !). Les amateurs d'originalité y trouveront aussi leur compte. Avez-vous déjà joué de la musique sur un clavier... Qwerty ? Ce n'est certes pas aussi facile que sur un clavier d'orgue, mais l'on s'habitue très vite ! Essayez.



Le principe de ce programme, réalisé sur Apple 2, est simple. Un morceau de musique se compose d'une succession de notes. A chaque note correspond un timbre (un son), une durée et un numéro d'ordre dans le morceau. Chaque air composé est conçu par le programme comme un tableau où R est le numéro d'ordre de chaque note, NO% (R) son timbre, et DU% (R) sa durée.

Pour le reste, c'est un jeu d'enfant. Pour jouer une note, enfoncez une touche alphabétique (sauf le « M »). Pour jouer un dièse, enfoncez simultanément la touche CTR (contrôle).

Une note, une touche, un dièse, une touche Bref, un jeu d'enfant !

Mais attention ! Ça ne marche pas pour le si # et le mi #, qui ne produisent aucun son, et pour cause, puisque si # = do et mi # = fa. Il faut donc actionner les touches suivantes : do et fa.

Simultanément, un petit compteur, en vidéo inverse, affiche le nombre des notes jouées — très pratique en cas de fausse note ! Par exemple, si la note numéro 10 est fausse, actionnez la touche

PROGRAMME ORGUE

```
0  ONERR  GOTO 9500
10  REM   INITIALISATION
20  GOSUB 5000
30  HTAB 20: VTAB 12: GET A$
40  IF A$ = "<" THEN 6000
50  IF A$ = "@" THEN 7000
60  IF A$ = "#" THEN 9000
70  IF A$ = "/" THEN R = R - 1: INVERSE : HTAB 35: VTAB 23: PRINT
    R * (R > = 0);: NORMAL : PRINT " ": IF R < 0 THEN R = 0: GOTO
    30
80  REM SI T%(CODE ASCII)>0 ALORS T=VALEUR DE T%(CODE ASCII)
90 T = 0: IF T%( ASC (A$)) > 0 THEN T = T%( ASC (A$))
100  REM    SI A$ EST UN CHIFFRE ALORS DUREE DE LA NOTE=-T%(ASC(A
    $))
110  IF T%( ASC (A$)) < 0 THEN J =  - T%( ASC (A$)): INVERSE : VTAB
    23: HTAB 4: PRINT A$: NORMAL : HTAB 20: VTAB 12: GOTO 30
120  REM SI T% N'EST NI UN CHIFFRE NI UNE LETTRE ALORS GET A$
130  IF T%( ASC (A$)) = 0 THEN 30
135  INVERSE : HTAB 35: VTAB 23: PRINT R + 1: NORMAL
140 R = R + 1:NO%(R) = T:DU%(R) = J: POKE 768,NO%(R): POKE 769,DU
    %(R): CALL 770: GOTO 30
5000  REM  S/PROG EN LANG.MACHINE
5010  POKE 770,173: POKE 771,48: POKE 772,192: POKE 773,136: POKE
    774,208: POKE 775,5: POKE 776,206: POKE 777,1: POKE 778,3: POKE
    779,240: POKE 780,9: POKE 781,202
5020  POKE 782,208: POKE 783,245: POKE 784,174: POKE 785,0: POKE
    786,3: POKE 787,76: POKE 788,2: POKE 789,3: POKE 790,96
5030  REM NO%(T)=VALEUR DE LA      NOTE X
5040  REM  DU%(T)=DUREE DE CETTE  NOTE
5050  DIM NO%(7000),DU%(7000),T%(95)
5060  REM INITIALISATION DE T%(X)
5070  FOR T = 49 TO 57:T%(T) = ((T - 48) * 30 - 15) *  - 1: NEXT

5080 T%(81) = 255:T%(17) = 242:T%(87) = 228:T%(23) = 215:T%(69) =
    203:T%(82) = 192:T%(18) = 181:T%(84) = 171:T%(20) = 161:T%(8
    9) = 152:T%(25) = 144:T%(85) = 136:T%(73) = 128
5090 T%(9) = 121:T%(79) = 114:T%(15) = 108:T%(80) = 102:T%(65) =
    96:T%(1) = 91:T%(83) = 85:T%(19) = 81:T%(68) = 76:T%(4) = 72
    :T%(70) = 68:T%(71) = 64:T%(7) = 60:T%(72) = 57:T%(8) = 54:T
    %(74) = 51:T%(75) = 48:T%(11) = 45
5100 T%(76) = 43:T%(12) = 40:T%(90) = 38:T%(26) = 36:T%(88) = 34:
    T%(67) = 32:T%(3) = 30:T%(86) = 29:T%(22) = 27:T%(66) = 25:T
    %(78) = 24
5110  HOME : PRINT "CHARGEMENT D'UNE MUSIQUE : <": PRINT "SAUVEGA
    RDE D'UNE MUSIQUE : #": PRINT "FIN D'UNE MUSIQUE : @ ": PRINT
    "EFFACEMENT DE LA DERNIERE NOTE : / "
5120  POKE 34,5: POKE 35,20
5130  VTAB 23: HTAB 11: PRINT "=> DUREE  NOTE # <="
5140  VTAB 12: HTAB 20: RETURN
5240  RETURN
6000  REM  CHARGEMENT
6010  PRINT : PRINT : INPUT "NOM DE LA MUSIQUE A CHARGER: ";NM$: IF
    NM$ = "" THEN  PRINT  CHR$ (4)"CATALOG": GOTO 6010
6020 D$ =  CHR$ (4)
6030  PRINT D$"OPEN"NM$
6040  PRINT D$"READ"NM$
6050  INPUT R
6060  FOR T = 1 TO R
6070  INPUT NO%(T),DU%(T)
6080  NEXT
6090  PRINT D$"CLOSE"NM$
7000  FOR T = 1 TO R: POKE 768,NO%(T): POKE 769,DU%(T): CALL 770:
    NEXT
8000  HTAB 1: INPUT "VOULEZ-VOUS (Q)UITTER LE PROGRAMME,     (C)O
    NTINUER VOTRE MUSIQUE ? ";N$: IF N$ = "Q" THEN  TEXT : HOME
    : END
8010  IF N$ = "C" THEN  HOME : VTAB 12: HTAB 20: GOTO 30
8020  PRINT  CHR$ (7): GOTO 8000
9000  REM  SAUVETAGE
9010  PRINT : PRINT : INPUT "NOM DE LA MUSIQUE A SAUVER : ";NM$: IF
    NM$ = "" THEN  PRINT  CHR$ (4)"CATALOG": GOTO 9010
9020 D$ =  CHR$ (4)
9030  PRINT D$"OPEN"NM$
9040  PRINT D$"WRITE"NM$
9050  PRINT R
9060  FOR T = 1 TO R
9070  PRINT NO%(T): PRINT DU%(T)
9080  NEXT
9090  PRINT D$"CLOSE"NM$
9100  GOTO 8000
9500  HOME : GOTO 30

]
```

« / » : le compteur revient en arrière à chaque pression.

La durée des notes est déterminée par les touches numériques. En début de programme, la durée par défaut correspond à la touche « 9 ». Dès qu'une touche numérique est enfoncée, le chiffre apparaît alors, en vidéo inverse, sur un second compteur où l'on verra ensuite tout nouveau chiffre sélectionné.

Les grandes orgues de Notre-Dame ? Non, mais presque !

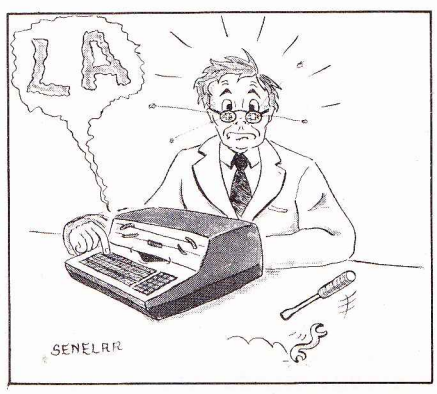
On peut bien sûr réentendre à tout moment une musique en cours de composition en enfonçant la touche « @ ». Une fois terminée, on la mémorisera sur disque, grâce à la touche « # ».

Enfoncez la touche « < », composez le nom d'une musique, et vous l'entendrez aussitôt.

Ce petit programme est donc fort complet et très intéressant. Bien sûr, ce ne sont pas les grandes orgues de Notre-Dame... Mais rien n'empêche d'en améliorer encore les performances : étendre le clavier à plusieurs octaves ou choisir certaines tonalités d'instruments, par exemple...

Mais en attendant, mélomanes, à tous vos claviers !...

Airy André



le bus arrivera-t-il en France ?

Le 80-Bus n'est pas une ligne de bus impérial sillonnant les routes de Sa Majesté la reine d'Angleterre, mais le nom d'un bus de système informatique développé au cours de ces dernières années en Grande-Bretagne, possédant des caractéristiques uniques qui ont séduit beaucoup de constructeurs anglais. Les cartes disponibles sont nombreuses et permettent de construire votre ordinateur en jouant au Meccano.

A l'origine, le 80-Bus s'appelait en fait bus Nascom : le Nasbus. Nascom vous dit peut-être quelque chose ; c'était le nom d'une société anglaise qui a créé deux ordinateurs, le Nascom 1 et le Nascom 2. Son succès trop rapide l'a poussée à la faillite et, après de longues hésitations, Nascom a été racheté par Lucas Logic, une autre société anglaise qui, à son tour, a créé le Nascom 3.

L'idée essentielle était d'utiliser des cartes bon marché, simple face, possédant tous les signaux nécessaires au processeur Z-80, avec possibilités d'extension à un éventuel processeur 16 bits plus un certain nombre d'autres signaux originaux.

La société Gemini Microcomputers Limited a défini plus précisément ce bus et a supprimé la possibilité de brancher un processeur 16 bits en utilisant les lignes d'adressage pour réaliser un adressage étendu avec le processeur Z-80. En effet, avec un logiciel et des cartes mémoires adhoc, il est possible, avec ce bus, d'adresser jusqu'à deux mégaoctets !

Une autre des particularités du 80-Bus est le signal RAM DIS qui donne automatiquement la priorité aux mémoires mortes sur les mémoires vives, ce qui permet de « superposer » des MEM et MEV.

A part la suppression de la possibilité du branchement d'un processeur 16 bits, le 80-Bus est totalement compatible avec le bus Nasbus, ce qui permet à tous les possesseurs de Nascom d'utiliser les cartes du 80-Bus.

Un grand nombre de fabricants se sont penchés avec intérêt sur le 80-Bus et ont produit un grand nombre de cartes, kits en tout genre, qui offrent aux esprits aventureux et bricoleurs un univers extensible.

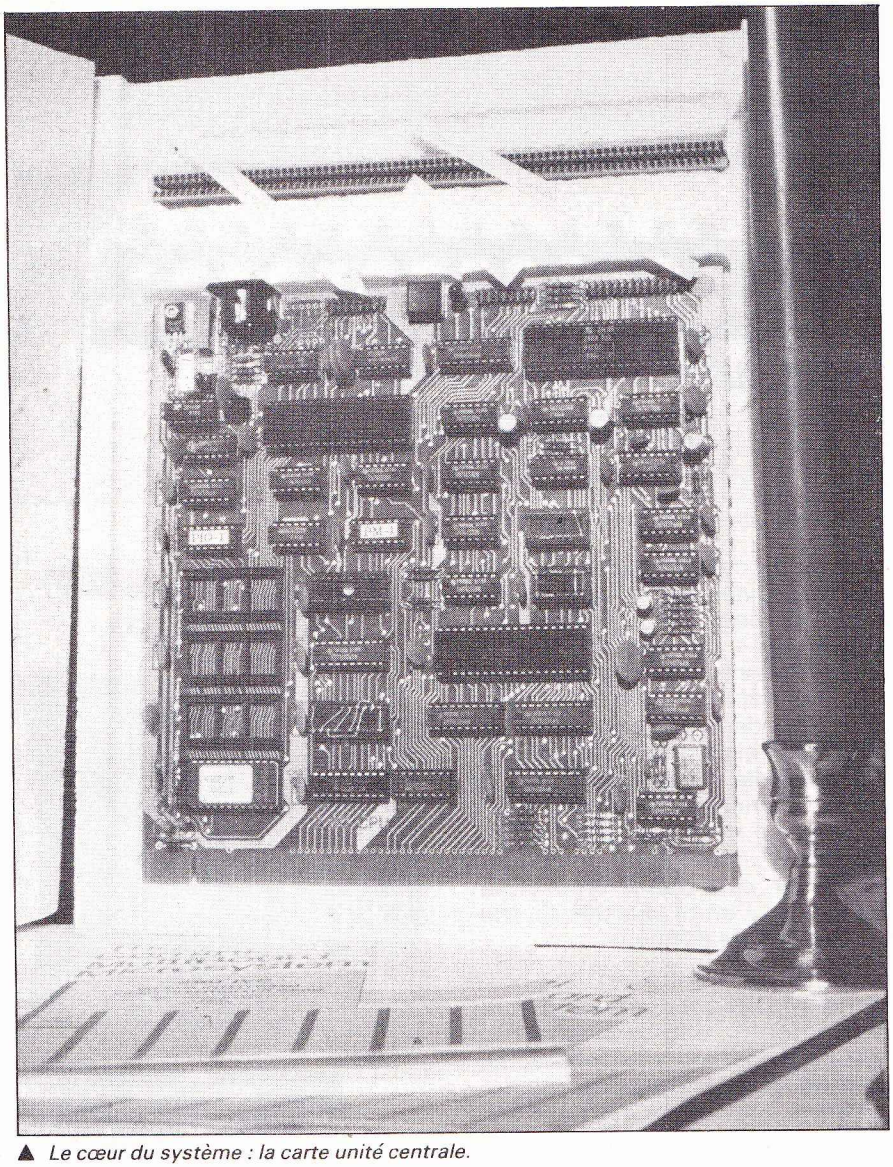
La société Gemini a créé une ligne de cartes qu'elle nomme « Multiboard » (multicartes) et qui comporte les cartes suivantes : une carte CPU (GM811) sans MEV ; une carte CPU (GM813) avec 64 K-octets de MEV ; une carte vidéo (GM812) ; une carte mémoire de 64 K-octets avec possibilité de pagination (GM802) ; une carte mémoire MEM/Eprom pour des mémoires mortes jusqu'à 40 Ko ; une carte contrôleur pour des lecteurs de disquettes (13 cm ou 20 cm) (GM809) ; une carte entrée-sortie (GM816) qui offre six ports, huit bits, un timer sur quatre canaux et une horloge en temps réel.

De nombreuses cartes sont disponibles ou encore à l'étude

D'autres cartes sont par ailleurs à l'étude : une carte série (GM818) avec quatre canaux séries avec vitesse de transmission « baud rate » programmable ; une carte de communication (GM836) ; un ensemble de disques Winchester (GM835).

D'autres fabricants anglais proposent des cartes pour ce standard :

. I/O Research propose la carte graphique couleur « Pluto » 640 × 576 bits et un processeur 16 bits ! Pour seize couleurs, ils ont également développé une carte de conversion analogique-numérique.

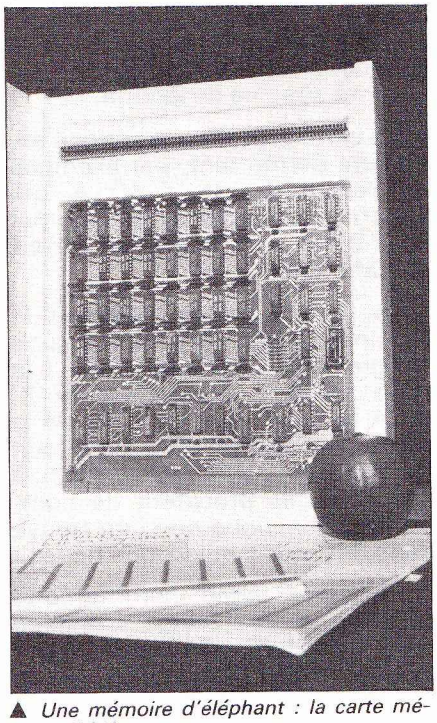


▲ *Le cœur du système : la carte unité centrale.*

. Microcode présente une carte de 32 K-octets de mémoires statiques avec batterie de secours.
. Ev Computing vend une carte contrôleur pour entrée-sortie avec standard IEEE488.
. Arfon Microélectronics a créé une carte synthétiseur de parole ; elle peut « prononcer » cent quarante mots... en anglais.
. Lucas Logic a développé une carte graphique en couleurs moins performante que la carte « Pluto » de I/O Research, mais aussi trois fois moins chère.

Map-80 propose une carte mémoire de 256 Ko paginable, qui utilise toute la puissance d'adressage du 80-Bus et permet d'avoir huit de ces cartes dans le système, c'est-à-dire deux mégaoctets.

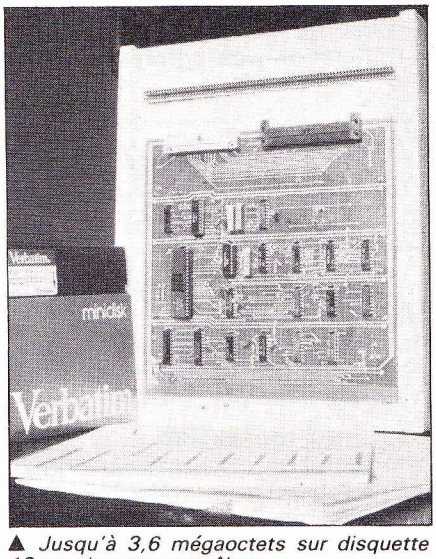
Toutes ces cartes sont accompagnées de nombreux autres équipements, genre alimentation traditionnelle ou à découpage, des cartes mères, programmateur d'Eprom, horloge en temps réel, des lecteurs de disquettes, des claviers, etc.



▲ *Une mémoire d'éléphant : la carte mémoire 64 Ko.*

La société Gemini a assemblé un certain nombre de ces cartes et en a fait un ordinateur appelé Galaxy 1, qui comprend les cartes CPU/MEV, la vidéo et la carte contrôleur. Les caractéristiques deviennent : deux processeurs Z-80 (un sur la carte CPU et un sur la carte vidéo), 64 K-octets de MEV ; 800 K-octets sur les deux lecteurs de disquettes ; écran de vingt-cinq lignes sur quatre-vingts caractères ; une entrée-sortie série ; une entrée-sortie parallèle (deux ports) ; une entrée-sortie cassette et une interface pour un crayon lumineux (light pen).

Le logiciel fourni comprend bien sûr le CP/M 2.2 et des programmes spécifiques comme le Gemzap, qui est un assembleur adapté à la carte vidéo, extrêmement rapide ; le Gempen, un éditeur de texte très pratique, avec possibilité d'imprimer un texte sur plusieurs colonnes ; le Gemdebug, qui est un outil de développement remplaçant avantageusement le débogueur du CP/M : le DDT ; le Comal-80, qui est un logiciel très puissant combinant les avantages du Basic et du Pascal (voir la description dans L'OI numéro 40).



▲ *Jusqu'à 3,6 mégaoctets sur disquette 13 cm : la carte contrôleur.*

Habitant en Grande-Bretagne, j'ai décidé de me lancer dans l'aventure « multicartes ». Ce système m'a séduit par son côté progressif ; on peut partir d'un minibudget pour arriver à un système très puissant : quatre lecteurs de disquettes (même huit si l'on est gourmand) ; un écran intelligent vingt-cinq lignes par quatre-vingts caractères ; un clavier ergonomique (Qwerty) ; des possi-

bilités d'entrée-sortie puissantes ; deux mégaoctets de MEV utilisables en disque virtuel et des possibilités d'extension très intéressantes limitées toutefois au processeur 8 bits.

J'ai donc commencé par la configuration minimale, c'est-à-dire une carte mère avec cinq positions, une alimentation « maison », une boîte permettant des extensions, un clavier, une carte unité centrale, une carte vidéo, une carte mémoire de 64 Ko et un moniteur noir et blanc ; le tout m'a coûté l'équivalent de 7 000 FF.

Les possibilités étaient déjà intéressantes : en effet, le moniteur est un pseudo-CP/M appelé RP/M, qui peut donc recevoir la plupart des logiciels CP/M en cassette et possède un certain nombre des fonctions d'un excellent débogueur avec changement des positions mémoires, affichage de positions mémoires sur l'écran avec leur contenu, déplacement de zones de mémoires, manipulation des entrées-sorties, etc.

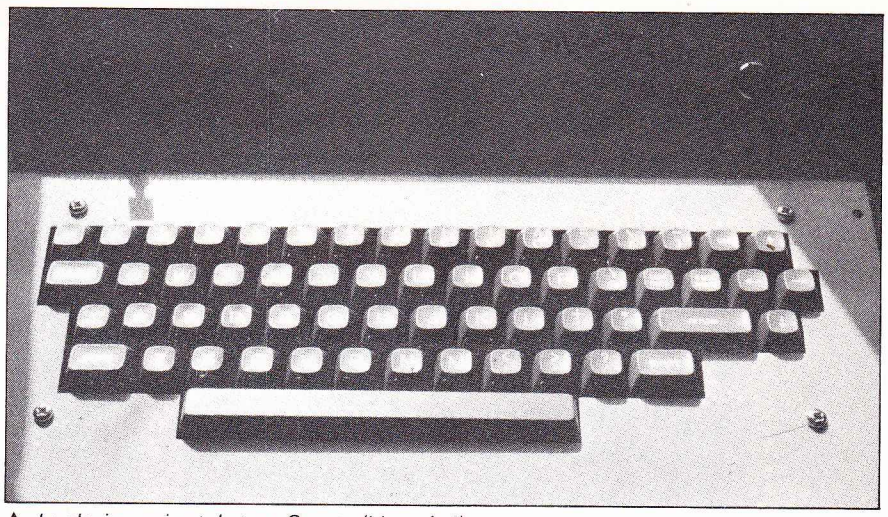
Trois circuits intégrés intelligents : Z-80 A, Z-80 PI/O et 8250

Le RP/M possède l'édition sur l'écran, c'est-à-dire le déplacement du curseur sur l'écran pour modifier des lignes de Basic sans avoir, par exemple, à les retaper.

Le cœur du système, la carte CPU (unité centrale : Central Processing Unit), est très dense et comporte trois circuits intégrés (CI) intelligents (voir photos) : le Z-80 A ; le Z-80 P I/O, qui procure deux ports parallèles de 8 bits avec différents types de communication, avec/sans interruptions, unidirectionnel, bidirectionnel, etc ; le 8250, un circuit qui gère les communications série procurant une sortie cassette à 1 200 bauds, une sortie série RS 232 C ou à boucle de courant ; la vitesse de transmission est programmable, la logique pour la gestion d'un modem est incluse, etc.

Il existe sur la carte trois supports vides pour des mémoires supplémentaires MEM ou MEV. De nombreux types peuvent être accommodés. La mémoire qui se trouve sur la carte peut être éliminée par logiciel.

Une prise clavier est prévue,



▲ *Le clavier, qui est de type Qwerty (bien sûr !).*

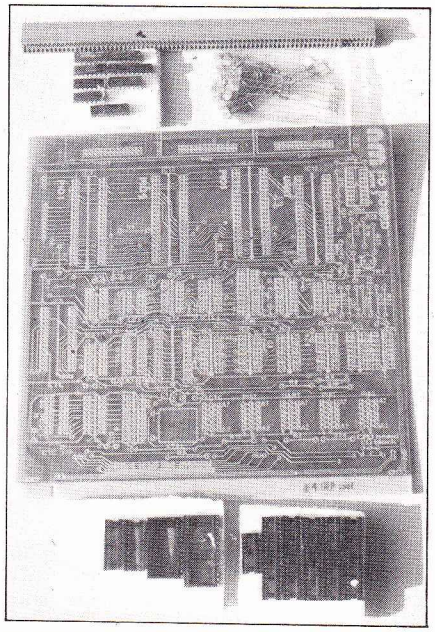
bien qu'il soit préférable de brancher le clavier sur la carte vidéo, ce qui présente comme avantage d'offrir un « tampon » (buffer) à l'entrée clavier, qui ajoute au système la rapidité d'entrée. Dans beaucoup de systèmes, si la carte unité centrale est occupée, il n'est pas possible d'entrer quoi que ce soit au clavier ; avec la carte vidéo qui met en mémoire jusqu'à cent vingt-huit caractères provenant du clavier, cela donne amplement le temps à l'unité centrale de traiter pas mal de données.

La prise cassette est une prise standard DIN avec lecture et enregistrement; elle gouverne un relais qui coupera ou rétablira l'alimentation du moteur du magnétophone. La vitesse est de 1 200 bauds, l'enregistrement se fait par blocs de cent vingt-huit caractères avec l'inclusion de blocs de vérifications et du nom du fichier à chaque bloc, ce qui donne une sécurité absolue de lecture.

La carte prévoit une horloge extérieure permettant, par exemple, de remplacer le Z-80 A qui « tourne » à quatre mégahertz par un Z-80 B qui « tournerait » à six mégahertz.

La vidéo est l'équivalent d'un terminal intelligent. Deux circuits intégrés donnent l'intelligence à la carte vidéo : un processeur Z-80 A (indépendant de celui de la carte unité centrale) et un circuit HD46505S (similaire au 6845) ; à eux deux, ils procurent de nombreuses fonctions que l'on trouve dans des terminaux sophistiqués.

La carte vidéo communique avec les autres cartes à travers trois ports E/S, elle possède ses propres MEM, indépendantes de l'espace mémoire de l'unité centrale ; cela offre deux avantages :



▲ *Certaines cartes sont disponibles en kit telle la carte entrée-sortie.*

ne pas limiter l'espace mémoire de l'unité centrale et accélérer les opérations. En effet, les traitements peuvent se faire parallèlement, le processeur principal n'est pas occupé par la gestion de l'écran. On peut même envisager de faire traiter deux programmes différents par les deux unités centrales, car la carte vidéo possède de la MEV utilisable par l'utilisateur.

La carte vidéo possède deux générateurs de caractères, l'un est une MEM de deux K-octets comportant cent vingt-huit caractères, un jeux de caractères ASCII classiques anglais, c'est-à-dire sans accent et avec le signe « # » remplacé par le signe de la livre sterling. L'autre est une MEV qui peut être programmée par l'utilisateur, être remplie des caractères inverses des caractères contenus dans le générateur en MEM

ou enfin recevoir des formes graphiques ; toutes ces options sont sélectionnables par logiciel.

Le logiciel de cette carte procure quelque cinquante fonctions qui, normalement, se trouvent sur des terminaux sophistiqués, donc chers. Ce sont, par exemple, l'effacement de la partie de l'écran au-delà du curseur ; l'insertion/effacement d'un caractère avec déplacement du reste de la ligne ou de l'écran ; l'effacement/insertion d'une ligne ; le changement du format de l'écran (25 × 80) ou (16× 49 = Nascom) ou tout autre format définissable pour l'utilisateur par logiciel ; la vidéo inverse/normal ; le « blocage » d'une certaine partie de l'écran, très pratique pour les titres ; les commandes graphiques : POINT, SET, RESET, etc.

Les possibilités graphiques sont limitées à 160 × 75 cellules mais, sachant que les caractères sont définissables, il y en a beaucoup plus : j'ai vu un programme qui dessinait des composants électroniques avec une grande précision.

Un crayon lumineux (« light pen ») peut être relié à une prise spécialement prévue, malheureusement le logiciel n'a pas été développé pour l'instant.

Une prise clavier permet de brancher un clavier ASCII. La carte vidéo procure une entrée à « tampon » de cent vingt-huit caractères, ce qui apporte un confort d'utilisation non négligeable.

Cette carte peut être utilisée avec un Nascom. Toutefois, des modifications de logiciel devront être apportées.

Revenons à la mémoire d'éléphant : la carte 64 Ko et plus... Les cartes mémoires auraient pu être de banales cartes 64 Ko, mais Gemini leur a ajouté la possibilité de les « paginer », c'est-à-dire que l'on peut mettre plusieurs cartes mémoires (MEM ou MEV) et les « commuter » dans le système quand on en a besoin en envoyant un certain octet à un port E/S : OFFH.

Cette possibilité peut avoir plusieurs applications. Lors du développement de programmes, on peut envisager de créer le texte source sur une page et de faire les essais sur une autre, évitant ainsi que les essais catastrophiques ne détruisent également la source. Une autre application, très populaire, est le disque virtuel, c'est-à-dire l'utilisation de MEV à la place et en tant que disquettes : cette solution présente donc un gain de temps lors de l'accès disque.

Le CP/M traite les MEV supplémentaires de la même manière que les disquettes ; il est ainsi possible de copier les disquettes en mémoire virtuelle ou de copier les mémoires virtuelles sur disquettes par les commandes CP/M classiques : PIP.COM. L'utilisation de programmes CP/M devient un plaisir, la recherche de mots sur un texte de 100 Ko un jeu d'enfant...

Une carte de 256 Ko est proposée par Map-80 ; elle peut être « paginée » par 64 Ko ; en plus de cela, grâce au 80-Bus ces cartes pourront être sélectionnées ou désélectionnées par les lignes d'adresse supplémentaires et permettront une puissance mémoire MEV de deux mégaoctets. Est-ce une alternative aux disques Winchester ?

Des caractéristiques intéressantes, des cartes de haute technicité

J'ai été satisfait de pouvoir utiliser l'un des Basic les plus puissants qui existent sans avoir de disquettes : le Basic-80 de Microsoft, que j'ai pu acheter en cassette. Il fait 24 Ko, c'est un peu long à charger à 1 200 bauds mais ça en valait la peine. Rapidement, l'attrait des disquettes est devenu irrésistible ; j'ai donc acheté une carte contrôleur puis un lecteur de disquette, son alimentation, son coffret et le CP/M 2.2 pour l'équivalent de 6 000 FF.

32 mégaoctets sur disquettes 13 cm sont possibles grâce à la carte contrôleur. Cette carte, contrairement aux autres, n'a pas une densité de population de circuits très forte. Elle est équipée d'un contrôleur 1797 de Western Digital, qui permet la simple et la double densité. Elle peut être utilisée pour des lecteurs 13 cm ou 20 cm. On peut y relier de un à quatre lecteurs. Elle peut être adressée à des ports différents permettant de mettre deux cartes dans le système, c'est-à-dire de pouvoir relier huit lecteurs 13 cm ou 20 cm. Ces lecteurs peuvent être de différents types : trente-cinq ou quarante pistes simple ou double face pour disquettes 13 cm, quatre-vingts pistes simple ou double face pour disquettes 13 cm, soixante-dix pistes simple ou double face pour disquettes 20 cm.

La fiabilité de lecture et d'enregistrement est assurée par une précompensation d'écriture et un circuit de reconstitution des données par phase à boucles fermées (phase locked loop data recovery circuitry).

En utilisant les lecteurs quatre-vingts pistes simple face qui sont recommandés par Gemini, on peut obtenir 400 Ko par lecteur ; avec deux cartes contrôleur, il est possible de relier huit lecteurs au système ; on obtient alors 3,2 mégaoctets en ligne. Si l'on utilise des lecteurs double face, on peut aller jusqu'à 6,4 K-octets, mais cette dernière solution n'est pas supportée par le logiciel fourni.

Le clavier que je possède comprend cinquante-neuf touches, y compris les touches de déplacement de curseur, contrôle, DEL-ESC, etc. Il ne possède pas de touches de fonction ni de touches type « calculatrice ».

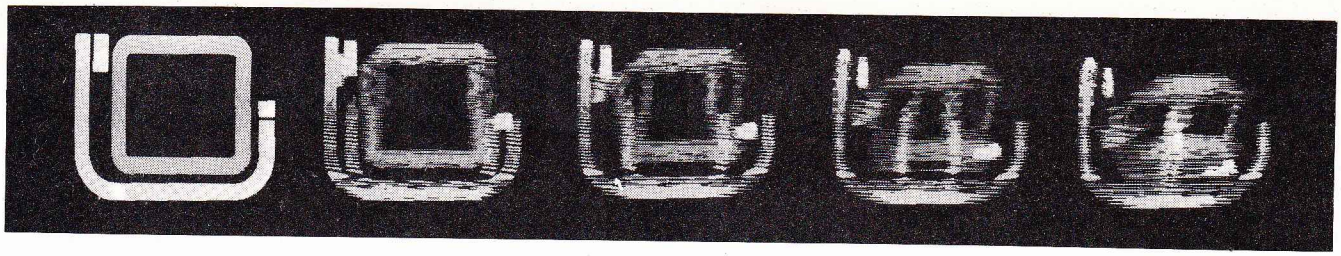
Gemini propose un nouveau clavier avec son coffret. Ce clavier comprend quatre-vingt-sept touches avec dix touches de fonction, un clavier séparé type « calculatrice », des touches curseur programmables, donnant deux octets lorsqu'elles sont enfoncées (très intéressant pour Wordstar).

Les claviers possèdent tous les deux une fonction « repeat » et deux touches peuvent être enfoncées presque simultanément ; les touches seront prises en compte : cela ajoute au confort du système.

En résumé, je pense que le 80-Bus possède des caractéristiques intéressantes qui le distinguent des autre bus genre S-100. Chaque carte n'a pas besoin de posséder son régulateur propre. Les cartes sont de haute technicité et procurent de nombreuses possibilités et un grand confort d'utilisation avec ses deux Z-80 et son clavier à « tampon » ; le tout supporté par un logiciel adapté et puissant, comportant l'édition d'écran sous CP/M, possibilité rare sinon unique qui confère au CP/M une réponse à tous ses détracteurs.

Seules certaines cartes sont vendues en France mais on peut espérer que les Français auront bientôt à leur disposition ce système prometteur...

Philippe Gysel



voulez-vous jouer aux jeux de L'OI ?

Votre journal grandit et s'étoffe : c'est à vous, amis lecteurs, qu'il le doit, à vos idées, à vos suggestions, à vos programmes. Voilà pourquoi nous vous proposons ce mois-ci d'ouvrir davantage la rubrique des jeux de L'OI. Si certains thèmes vous ont paru particulièrement séduisants, si vous avez pu réaliser un programme de qualité ou si vous avez des idées originales, écrivez-nous en précisant sur l'enveloppe « jeux de L'OI ». Les meilleures idées seront présentées dans cette rubrique. Faites-nous vite goûter la saveur de vos découvertes.

Les questions posées présentent divers degrés de difficulté, que nous essayons de vous indiquer (très subjectivement) par les sigles suivants :

débutant

plutôt simple

assez difficile

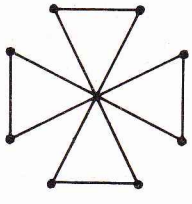
pour les longues soirées d'hiver

303

Ecrire un programme qui comptabilise sur une disquette le nombre de pressions effectuées sur telle ou telle touche de votre machine. Ce programme devra être capable d'éditer des statistiques en fin d'utilisation (en tenant compte des résultats obtenus, vous pourriez alors envisager une disposition plus ergonomique de votre clavier).

304

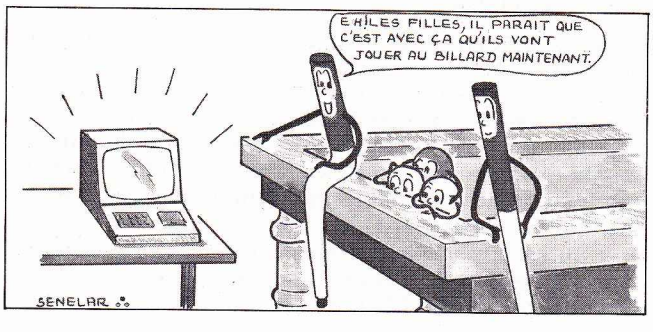
Un problème de théorie des graphes : le théorème de l'amitié. « Si, dans un groupe de n personnes, deux personnes quelconques ont un et un seul ami commun faisant partie du groupe, alors l'une des personnes est amie de toutes les autres, et n est impair ». Par exemple, pour n = 9, on obtient :

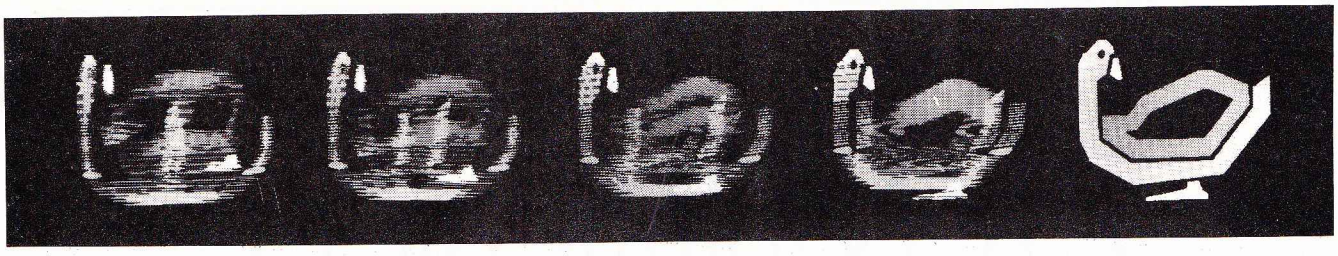


Ecrire un programme qui dessine tous les graphes possibles pour n donné.

305

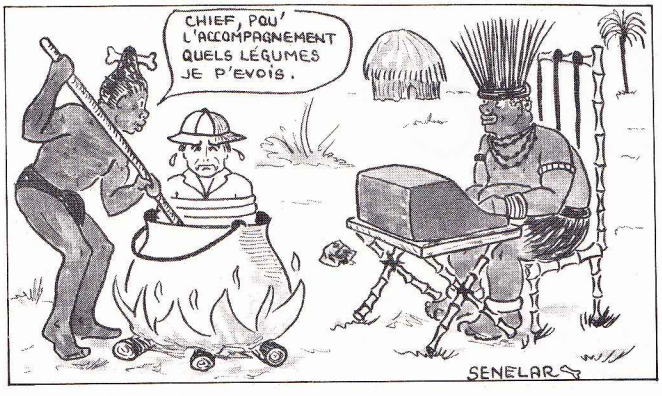
Avez-vous déjà essayé de jouer au billard avec votre ordinateur ? Pour le Noël de vos enfants, vous prendrez sans doute le temps de peaufiner un programme qui simule un jeu de billard, avec les rebords, l'effet, la commande de l'orientation de la queue par les manettes de jeu, et la possibilité de choisir la force à donner à chaque coup. Vous pourrez aussi introduire un compteur, et, pourquoi pas, prévoir la possibilité pour un joueur de faire un accroc, et cela de manière aléatoire.





306

Votre femme s'intéresse peu à votre ordinateur ? Faites-lui et faites-vous plaisir avec un programme de calcul des temps de cuisson pour la viande : nature du morceau, épaisseur, type de cuisson désirée, etc.



307

Redécouvrez le plaisir de compter en simulant, sur votre ordinateur, le fonctionnement d'un boulier classique (avec éventuellement des variantes : russe, japonais ou autre) ; le programme devra afficher les nombres que vous entrez sur le boulier et effectuer les calculs sous vos yeux, en décomposant les mouvements effectués.

308

Vous utilisez dans vos programmes un certain nombre de pointeurs ou indicateurs, qui peuvent prendre les deux états logiques vrai ou faux selon qu'une certaine condition est ou non satisfaite. Ecrivez un programme qui affiche en permanence dans un coin de l'écran l'état de ces pointeurs.

309

Votre ordinateur va dessiner sur l'écran un certain nombre de cercles, tangents par un côté. Il donne le sens de rotation du premier cercle. Le jeu consiste à trouver le sens de rotation du dernier, dans un temps minimum.

310

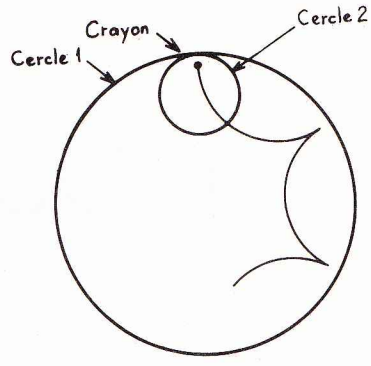
A partir de la même idée, emboîtez les cercles les uns dans les autres.

311

Toujours des cercles : connaissez-vous le jeu appelé spirographe ? On dispose de disques et de réglettes, pouvant s'engrener les uns sur les autres, les rapports d'engrenage étant connus.

Cela permet de tracer des figures géométriques plus ou moins complexes sur une feuille de papier ; pouvez-vous transformer votre ordinateur en spirographe ?

Rappel du principe :



Le cercle 2 roule sans glisser à l'intérieur du cercle 1 (on pourra définir au départ le nombre de tours).

312

Un peu d'optique : on considère un faisceau lumineux, matérialisé par une famille de rayons, convergeant initialement en un point O. On intercepte ce faisceau à l'aide d'un dioptre plan, perpendiculaire à l'axe du faisceau. Votre programme devra permettre de visualiser la « caustique » du dioptre (partie de l'espace où va s'accumuler la lumière).

313

La formule $n^2 - n + 41$ donne quarante et un nombres premiers successifs si l'on remplace n par 0, 1, 2, etc. De même la formule $n^2 - 79n + 1601$ en donne quatre-vingts. Pouvez-vous écrire un programme cherchant et essayant des formules qui donnent beaucoup de nombres premiers ?

314

Vous allez dessiner sur l'écran une bouche, puis vous écrirez un programme d'animation de cette bouche qui devra articuler un certain nombre de mots qui apparaîtront en bas de l'écran.

315

Vous pouvez, à partir du programme précédent, imaginer un jeu : la bouche articule un mot que vous ne voyez pas. Il faut essayer de le trouver.

316

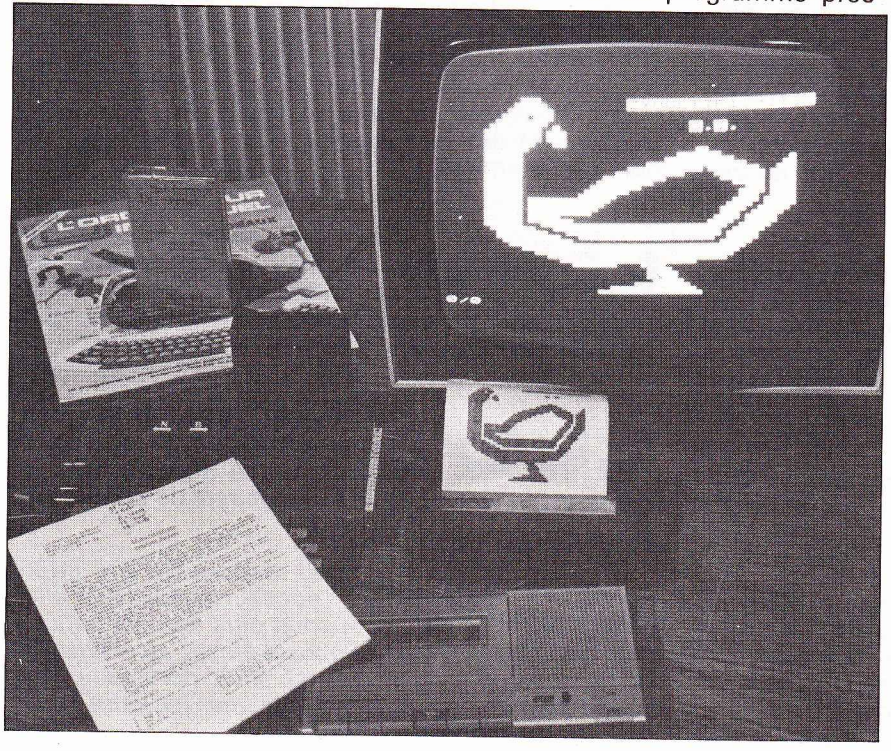
Si vous disposez d'une interface vocale, essayez d'aller beaucoup plus loin : la bouche va prononcer les mots que vous entrez au clavier. Si votre appareil est au point, allez donc voir une bonne agence de publicité : cela peut toujours servir mais aussi, signalez-le nous !

ZX Multifichiers logiciel pour ZX-81

ZX Multifichiers (ou ZXM) que l'on peut encore assimiler à son grand frère Visifile, logiciel pour Apple 2 (L'OI n°42, novembre 1982) est un logiciel à la mesure du ZX-81, servant à créer des fichiers d'adresses, à gérer des stocks, à classer des données, etc. Il coûte 215 FF ttc.

ZX Multifichiers se présente sous la forme d'une cassette et d'une notice d'accompagnement de dix pages dactylographiées.

Les deux faces de la cassette sont enregistrées, la deuxième face étant une copie du programme, utile si vous n'arrivez pas à lire la première face.



ZXM est un programme presque exclusivement écrit en langage machine, ce qui en fait un programme très performant quant à la rapidité d'exécution. Le langage machine, occupant un peu plus de trois K-octets, est implanté à la première ligne Basic au niveau d'une REM, le reste du programme n'étant composé que de dix lignes Basic.

La notice précise que ZXM est à vocation professionnelle (mais dans quelle mesure peut-on considérer le ZX-81 comme un ordinateur professionnel, malgré un rapport qualité/prix, me semble-t-il, très performant). Cette notice est bien commentée et permet, en prenant un exemple de création de fichier, de découvrir pas à pas les nombreuses options que ZXM propose et qu'il faudra apprendre à bien maîtriser pour tirer parti de toutes les possibilités offertes.

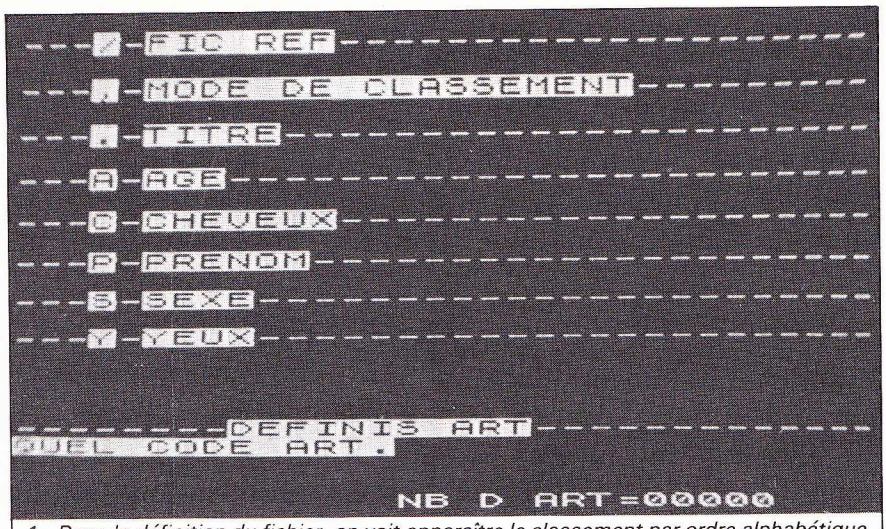
ZX-M dispose d'un menu comprenant, en tout, onze commandes

ZXM commence par vous proposer le menu principal composé de onze commandes, dont cinq sont directement liées à la création du fichier. Prenons un exemple : la création d'un fichier de famille.

. Par la *commande I,* nous allons définir les articles : prénom, âge, sexe, yeux, cheveux. On peut ainsi définir trente-six articles référen-

ciés de zéro à neuf et de A à Z, pour lesquels nous donnons à chaque fois un nom (P. « PRENOM » : P référence de l'article et « PRENOM », nom de l'article). Au fur et à mesure que nous définissons les articles, ceux-ci s'affichent à l'écran en ayant été au préalable classés par ordre alphabétique (ou plutôt dans l'ordre zéro à neuf et A à Z).

. Nous avons donc, avec ces cinq articles, défini une fiche. Chaque prénom, ainsi que les données s'y rapportant (sexe, âge, etc.), que nous rentrerons au moyen de la *commande A,* constitueront une fiche. L'ensemble des fiches sera donc notre fichier.



1. Dans la définition du fichier, on voit apparaître le classement par ordre alphabétique (A pour âge, C pour cheveux, etc.)

On définit le format de l'affichage du nom d'un article

La commande F sert à définir les formats d'affichage afin d'obtenir une bonne présentation des résultats à l'écran ou sur l'imprimante. On peut créer trente-six formats, référencés aussi de zéro à neuf et de A à Z, auxquels on peut affecter un nom.

. Continuons notre fichier en utilisant la *commande F.* A la question « référence fichier », tapons « ø » pour le référencer. Une sous-commande « . » (point) nous permet de lui donner un nom... Appelons-le « FICHIER DE FAMILLE ».

Une autre sous-commande de F, « , » (virgule) sert à définir le mode de classement et l'espacement entre les lignes affichées. Répondons alors « P2 » ; P va définir le classement par ordre alphabétique des prénoms (article P) et 2 est l'espacement entre deux prénoms (ici 2 signifie passage à la ligne suivante et saut d'une ligne).

Chaque article peut maintenant être repris pour définir le format dans lequel on désire l'affichage du nom de l'article. Ainsi pour P (PRENOM) nous avons pris le code format 4øAW2, qui signifie que PRENOM va s'inscrire en toutes lettres à partir de la ligne 4 au niveau de la colonne ø avec un nombre de caractères (affectation de PROCUSTE) maximal de dix (A = 10 ; B = 11, ..., T = 31), W précisant l'affichage en vidéo normale (B pour vidéo inverse).

ZX-M propose une sélection cumulative et une sélection soustractive

Dans notre exemple, le format des autres articles a été défini selon le même principe.

. La *commande R* sert à sélectionner l'un des trente-six formats (0 à 9 ou A à Z).

. La *commande S* permet de traiter et sélectionner toutes les données patiemment entrées. Son utilisation entraîne l'affichage d'un menu de sélection avec cinq options. L'option ø efface la sélection en cours. L'option 5 reprend l'ensemble des fiches en mémoire. L'option M entraîne un retour au menu principal. L'option 1, appelée choix additionnel, sélectionne en plus des fiches résultant de la sélection précédente (celles dont on précise le nouveau critère de sélection).

Ainsi, après avoir mis à zéro la sélection dans notre exemple (option ø), l'option 1, avec pour critère de sélection le sexe féminin, permet de retrouver à l'affichage par la *commande I* l'ensemble des articles « féminins ». Si, à nouveau, on reprend l'option 1 avec pour critère de sélection le sexe masculin, la sélection résultante donnera l'ensemble des articles « masculins » et « féminins ». L'option 1 est donc une sélection *cumulative.*

PRENOM	AGE	SEXE	YEUX	CHEVEUX
ANNE-MARIE	57	FEMI	BRUN	NOIRS
BENOIT	29	MASC	VERT	NOIRS
BERNADETTE	24	FEMI	BLEU	BLOND
CATHIA	25	FEMI	BRUN	MARRO
DANIEL	49	MASC	BRUN	MARRO
EDITH	22	FEMI	BRUN	MARRO
EMMANUELLE	27	FEMI	BRUN	BLOND
FRANCOISE	27	FEMI	BLEU	BLOND

PRENOM	AGE	SEXE	YEUX	CHEVEUX
JEAN-PIERR	30	MASC	BLEU	NOIR
JOEL	26	MASC	BRUN	MARRO
KATARINE	25	FEMI	BLEU	BLOND
PATRICK	28	MASC	BLEU	NOIR
PIERRE	61	MASC	BLEU	NOIR
THERESE	48	FEMI	BRUN	MARRO
XAVIER	4	MASC	BLEU	BLOND
YVAIN	2	MASC	VERT	BLOND

PRENOM	AGE	SEXE	YEUX	CHEVEUX
YVAIN	2	MASC	VERT	BLOND
XAVIER	4	MASC	BLEU	BLOND
EDITH	22	FEMI	BRUN	MARRO
BERNADETTE	24	FEMI	BLEU	BLOND
CATHIA	25	FEMI	BRUN	MARRO
KATARINE	25	FEMI	BLEU	BLOND
JOEL	26	MASC	BRUN	MARRO
EMMANUELLE	27	FEMI	BRUN	BLOND

PRENOM	AGE	SEXE	YEUX	CHEVEUX
FRANCOISE	27	FEMI	BLEU	BLOND
PATRICK	28	MASC	BLEU	NOIR
BENOIT	29	MASC	VERT	NOIRS
JEAN-PIERR	30	MASC	BLEU	NOIR
THERESE	48	FEMI	BRUN	MARRO
DANIEL	49	MASC	BRUN	MARRO
ANNE-MARIE	57	FEMI	BRUN	NOIRS
PIERRE	61	MASC	BLEU	NOIR

PRENOM	AGE	SEXE	YEUX	CHEVEUX
EDITH	22	FEMI	BRUN	MARRO
BERNADETTE	24	FEMI	BLEU	BLOND
CATHIA	25	FEMI	BRUN	MARRO
KATARINE	25	FEMI	BLEU	BLOND
EMMANUELLE	27	FEMI	BRUN	BLOND
FRANCOISE	27	FEMI	BLEU	BLOND
THERESE	48	FEMI	BRUN	MARRO
ANNE-MARIE	57	FEMI	BRUN	NOIRS

PRENOM	AGE	SEXE	YEUX	CHEVEUX
EDITH	22	FEMI	BRUN	MARRO
CATHIA	25	FEMI	BRUN	MARRO
EMMANUELLE	27	FEMI	BRUN	BLOND
THERESE	48	FEMI	BRUN	MARRO
ANNE-MARIE	57	FEMI	BRUN	NOIRS

PRENOM	AGE	SEXE	YEUX	CHEVEUX
EMMANUELLE	27	FEMI	BRUN	BLOND
THERESE	48	FEMI	BRUN	MARRO
ANNE-MARIE	57	FEMI	BRUN	NOIRS

Menu

```
ZXM

A. . .INSERTION
U. . .DERNIER ART.
D. . .DETRUIS ART.
R. . .SELECT. FICH.
S. . .SELECT. ART.
P. . .AFFICHAGE
T. . .TOTAL/MOYENNE
I. . .DEFINIS ART
F. . .FORMAT FICH.
B. . .NB D OCTETS
V. . .SAVE

NB D ART=00000
```

Reprenant le lot de fiches ainsi sélectionnées, si maintenant nous choisissons l'option 2, appelée tri sur une clé, avec pour critère de sélection le sexe féminin, les fiches sélectionnées ne comportent plus que des articles « féminins ». L'option 2 est donc une sélection *soustractive* par rapport aux fiches des sélections précédentes.

L'option 1 et l'option 2 permettent ainsi des sélections en cascade en fonction de différents critères. Ces deux options sont très complètes, car elles permettent d'extraire d'un fichier les fiches comportant des critères très précis, que l'on définit lors de l'affichage d'un sous-menu commun aux deux options.

L'exemple ci-contre le montre, où l'on extrait d'abord du fichier toutes les femmes (chaîne « féminin » pour l'article sexe), puis, de cette sélection, les femmes qui ont les yeux bruns (chaîne « bruns » pour l'article yeux) et enfin, parmi celles-ci, les femmes d'un âge supérieur à 25 ans (AGE >25). Les options du sous-menu autorisent des sélections de fiches par rapport à des chaînes alphanumériques, des sous-chaînes des valeurs numériques supérieures, égales ou inférieures à un argument que l'on précise.

Un classement par ordre alphabétique retouché Sinclair

D'autres options sont encore disponibles dans le menu général : la possibilité de modification d'un article, sans effacement, la *commande T*, qui donne le total et la moyenne d'un article dans une sélection définie, la possibilité de connaître à tout moment le nombre d'octets qui seront encore libres pour la création de fiches, etc.

Il faut noter, toutefois, une imperfection de ZXM : dans notre exemple, si par la commande F nous choisissons un mode de classement par âge et qu'il y ait des âges inférieurs et supérieurs à dix ans, le classement sera erroné. Cela tient au classement fait par ordre alphabétique (alphabétique Sinclair, bien entendu) ; quatre ans se situera donc dans le classement entre trente-neuf et quarante ans.

Un bon programme, rapide, peu gourmand en espace mémoire

Un remède serait d'écrire à la place de « 4 » pour l'âge, « Δ4 » (espace, 4). Mais si vous demandez alors le total et la moyenne des âges, ZXM vous répond que l'âge écrit ainsi n'est pas une valeur numérique. A noter aussi, à ce niveau, que ce commentaire, contrairement aux autres, a été omis par le traducteur anglais-français, car il est écrit NON-NUMERIC.

En conclusion, ZXM est un très bon programme, très rapide dans son exécution, ne faisant que peu appel au Basic. Il nécessite de se familiariser avec ses nombreuses commandes et options avant d'envisager son utilisation professionnelle (?). Etant écrit en langage machine, il n'est pas trop gourmand en espace mémoire, et vous permet de traiter près de cent vingt fiches comprenant en moyenne cent caractères chacune, avec 16 Ko de MEV. Tous les résultats peuvent être par ailleurs obtenus sur imprimante.

Benoît Thonnart

le dernier salon où l'on cause

Du langage machine au générateur de programmes : ce voyage, aucune agence de tourisme ne l'avait inscrit à son catalogue. L'OI relève le défi. Comment est né l'assembleur ? Pourquoi Fortran et Cobol sont-ils devenus des frères ennemis ? Quel est le rôle du Pentagone dans la création d'Ada ? Toutes ces questions ont leur réponse dans cet historique. De fil en aiguille, de compilateur en interpréteur et autres, embarquez vite pour voir défiler quelques-uns des quelque quatre cents langages informatiques.

Sans programme qui lui spécifie instruction par instruction ce qu'il doit faire, l'ordinateur est incapable de quoi que ce soit. Mais on est alors frappé par la diversité et le monde des langages de programmation existant. Cela est dû en partie à la pluralité des applications, mais aussi à des conditions historiques.

Examinons tout d'abord le seul langage qui soit véritablement « compris » par un ordinateur : le langage machine, binaire. Tous les gros ordinateurs possèdent un ensemble d'interrupteurs et de voyants permettant d'entrer une donnée binaire là où l'on veut dans la mémoire, et de la visualiser. L'inconfort de cette méthode est apparu très tôt et on l'a bien vite remplacée par l'utilisation d'un moniteur hexadécimal, au moins.

L'hexadécimal est au binaire ce que la sténographie est à la dactylographie. Cela montre d'emblée la nécessité de disposer, pour faciliter l'usage de l'ordinateur, d'un ensemble de programmes fournis par le constructeur (ou un autre) : le système d'exploitation.

Les interrupteurs et les voyants existaient sur les premiers ordinateurs individuels ; ils ont disparu depuis. Mais la programmation en hexadécimal est elle-même très incorfortable. Or, lorsque l'on prépare un programme en langage machine, il est naturel d'envisager chaque instruction en termes de nature de l'opération et désignation de l'opérande concerné (paramètre ou donnée entrant dans une opération logique). Cette manière naturelle de décrire les instructions, à peine formalisée, est permise par ce qu'on appelle le langage assembleur symbolique. Actuellement, dire « je programme en langage machine », sous-entend « en assembleur ».

Voici un petit exemple en assembleur, (pour le processeur 6809), avec, en regard de chaque instruction, sa traduction en hexadécimal. ▼

Assembleur	Hexadécimal	
ADTAB CLR A	2000 4F	; sous-programme
CLR B	2001 5F	d'addition des éléments d'un tableau.
LD X # DEBUT	2002 8E 1000	; début du tableau, adresse 1000
BOUCLE ADD D X++	2005 E3 81	
CMP X # FIN	2007 8C 1500	; fin du tableau, adresse 1500
BLS BOUCLE	200A 23 F9	
ST D SOMME	200C FD OF FE	; range la somme, adresse OFFE
ATS	200F 39	; retour de S/P.

Traditionnellement, une instruction assembleur comporte quatre zones séparées par au moins un blanc : l'étiquette qui sert à repérer l'instruction pour les sauts (dans l'exemple précédent « BOUCLE »), le code opération (par exemple ADD pour addition) quelquefois complété comme ci-dessus par l'indication d'un registre (par exemple ADD D addition dans le registre D), la zone opérande et la zone commentaire (dans l'exemple précédent introduite par un point-virgule).

Selon l'assembleur dont on dispose, la zone opérande peut faire intervenir des formalismes plus ou moins sophistiqués (dans notre exemple, le # qui spécifie l'adressage immédiat et le, X++ qui spécifie un adressage indexé avec double incrémentation en donnent une idée), mais l'essentiel est que l'assembleur permet l'emploi de noms symboliques, que le programmeur peut choisir aussi parlants que nécessaire.

Où l'assembleur est considéré comme un abus de langage

Naturellement, et cela devient indispensable dès que l'on quitte le niveau du langage machine binaire, un texte en assembleur comme le précédent ne peut être exécuté directement par la machine. Il faut qu'il soit traduit en binaire. Cette tâche est dévolue à un programme fourni (ou qui devrait l'être) avec le système d'exploitation et que l'on appelle assembleur (l'emploi de ce terme pour désigner le langage, alors qu'il est proprement réservé au traducteur, est un abus de « langage » consacré par la tradition). En fait, on n'utilise pas que l'assembleur pour préparer le programme ; on passe par la séquence classique :

. utilisation d'un *éditeur* pour créer — ou corriger — le texte en assembleur (on dit le « texte source ») et le ranger sur disque (fichier source) ;

. utilisation de l'*assembleur* qui, à partir de fichier source, crée le texte objet (=traduit) et le range sur disque ; l'assembleur produit en outre une liste qui contient le texte traduit en hexadécimal (cf précédemment) et les diagnostics d'erreurs éventuels ;

. utilisation d'un *chargeur* pour amener le programme objet en mémoire où il sera exécuté ; cette séquence se retrouvera avec les langages évolués.

La traduction effectuée par l'assembleur est simple : il y a correspondance biunivoque entre instructions-assembleur et instructions-machine. Pour chaque instruction, l'assembleur doit d'abord reconnaître le code opération et le traduire par le motif binaire correspondant, qu'il trouve dans une table. La traduction des symboles de la zone opérande est plus délicate ; là encore, l'assembleur gère une table des variables, qui établit la correspondance entre le nom et l'adresse de chaque variable. Mais la traduction est compliquée par le fait que certains assembleurs permettent l'emploi de véritables expressions qu'il faut interpréter et par le fait des *références en avant :* lors de la traduction d'une instruction, l'adresse correspondant à un symbole peut n'être pas encore connue ; c'est le cas par exemple d'une instruction de saut plus loin dans le programme.

Encore une notion intéressante : dans l'exemple précédent, l'assembleur peut savoir que le symbole Debut désigne l'adresse 1000 (hexa), car on lui indique dans une instruction qui ne sera pas traduite en langage machine mais sera utilisée pour aider à la traduction. Cela s'appelle une *directive* et se retrouve dans les langages évolués sous le nom de « déclarations ». Par exemple, le programme précédent serait précédé des directives :

```
SOMME = $ OFFE
DEBUT = $ 1000
FIN = $ 1500
* = $ 2000 ; fixe le départ du programme.
```

Chaque fois qu'il rencontre la macro, l'assembleur la développe

Les assembleurs se sont rapidement perfectionnés et ont acquis des possibilités très élaborées comme l'assemblage conditionnel et les macro-instructions. Ces macro-instructions sont très intéressantes : elles permettent d'étendre virtuellement le jeu d'instructions de la machine. Supposons que nous ayons une séquence d'instructions qui revient à tout bout de champ, par exemple la sauvegarde des registres D, X et Y à partir de la case mémoire M, cela s'écrit en 6809 :

```
ST D M
ST X M+2
ST Y M+4
```

Et, bien que l'on dispose d'un macro-assembleur, on peut définir une macro-instruction par :

```
ST D M
ST X M+2
ST Y M+4
ENDM
```

A chaque fois qu'on aura besoin de sauver les registres, on

écrira la ligne suivante :
SAUVE ZONE 1
un peu plus loin SAUVE ZONE 2, etc.

Ainsi, chaque fois qu'il rencontre la macro-instruction, l'assembleur la développe, c'est-à-dire qu'il implante le texte complet de la macro-instruction, en y remplaçant le paramètre par la valeur fournie. Ainsi pour SAUVE ZONE 1, on obtiendrait :

+ ST D ZONE 1
+ ST X ZONE 1+2
+ ST Y ZONE 1+4

Les + indiquent traditionnellement qu'il s'agit du développement d'une macro-instruction.

Les macro-instructions remplissent une fonction un peu analogue à celles des sous-programmes, à cela près que c'est au niveau de l'écriture qu'elles interviennent et non à celui de l'implantation mémoire. Mais les macro-instructions sont d'un intérêt très grand, elles permettent d'étendre le jeu d'instructions au moins au niveau de l'écriture : pour une instruction qui vous manque, vous définissez une macro-instruction formée de la séquence qui la simule.

Un autre aspect est de considérer que les macro-instructions permettent d'avoir une vue plus synthétique des séquences d'instructions et ce point de vue va se retrouver maintenant avec les langages évolués.

De l'assembleur symbolique aux langages évolués

En effet l'assembleur symbolique pallie un grand nombre des inconvénients du langage machine binaire, mais il n'est qu'une étape et il lui reste des inconvénients importants, puisqu'il est en correspondance étroite avec le langage machine. Les deux principaux sont :
. le caractère extrêmement élémentaire des instructions, ennemi de toute pensée synthétique ;
. le fait de devoir envisager les traitements du point de vue de la machine et donc de devoir renoncer à toute portabilité : deux machines différentes auront des langages différents et donc tout programme devra être réécrit si l'on veut passer d'une machine à une autre.

Très rapidement, le besoin d'un langage plus synthétique et plus proche de l'application a été ressenti. Il était même permis de rêver à la programmation des ordinateurs en langage naturel. On appelle *langages évolués* ces langages de programmation qui, sans vouloir être la langue naturelle, ont pour ambition d'être plus synthétiques que l'assembleur et plus proches des problèmes et des notations (mathémaques ou autres) de l'utilisateur.

Le premier est apparu dans le commerce vers 1955-1957... C'est le Fortran (développé par Backus). Il est dirigé principalement vers les applications scientifiques, étant surtout formé d'un très bon traducteur de formules mathématiques (dont il tire son nom : FORmula TRANSlator, soit traducteur de formules) et de quelques instructions d'entrées-sorties et de ruptures de séquence.

En Fortran, la définition de R comme somme de P et Q s'écrit presque comme en mathématiques R = P + Q, alors qu'en assembleur il faudrait déjà au moins trois instructions :
LD A P
ADD A Q
ST A R
et, la notation étant mathématique, elle est la même quelle que soit la machine.

Bien sûr, des expressions arithmétiques plus complexes sont possibles et le rendement par rapport à l'assembleur est encore plus grand.

Fortran a reçu des améliorations successives : la version Fortran II est apparue en 1959 et elle introduisait les sous-programmes ; la version Fortran IV (1963) améliorait (et systématisait) les entrées-sorties et elle introduisait un semblant de traitement de chaînes de caractères ; la version Fortran 77 (1977) essaie de moderniser le langage. Fortran se caractérise toutefois par un certain nombre de limitations, de contraintes d'écriture (l'écriture n'est pas libre dans les colonnes, les noms des fonctions en Fortran II doivent se terminer par F, la première lettre du nom d'une variable complique son type arithmétique, etc.) et par, il faut le dire, une certaine incohérence logique dans la définition, qui fait apparaître chaque instruction comme une sorte de « macro-instruction » indépendante des autres. Les limitations du langage ont principalement pour but de faciliter la tâche du programme traducteur, qui s'appelle un « compilateur ».

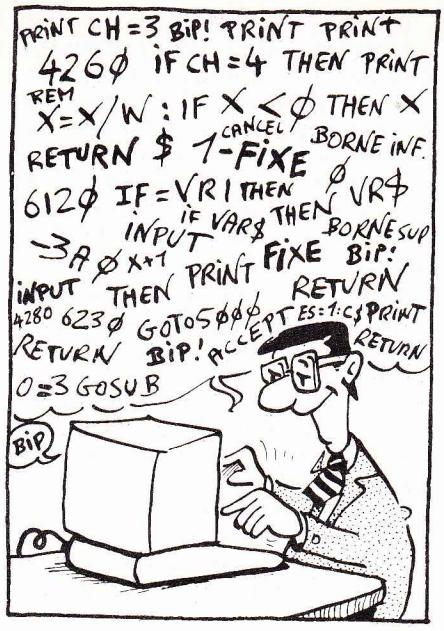
Les étapes d'introduction d'un programme sont celles décrites plus haut, le compilateur remplaçant l'assembleur. En outre, sur les machines de cette époque, au lieu de constituer un fichier source sur disque à l'aide d'un éditeur, on fournissait le programme sous forme d'un paquet de cartes perforées.

Fortran est l'archétype des langages de cette première époque, où les propriétés du langage sont, en définitive, celles du compilateur. Certaines caractéristiques du langage sont introduites non pour faciliter l'écriture des programmes d'application ou pour donner une cohérence linguistique au langage, mais pour faciliter la tâche – qui reste toutefois complexe – d'écriture des compilateurs.

Quant aux applications, les limitations de Fortran se faisaient surtout sentir pour la gestion avec, notamment, pas de traitement satisfaisant des chaînes de caractères et traitement de fichiers trop rudimentaires : la gestion des fichiers en accès direct est, dans tous les Fortran, une extension hors norme fournie à sa manière par le conducteur, s'il le veut bien ; il n'y a a fortiori pas de séquentiel indexé. Au contraire, Fortran est satisfaisant dans le domaine scientifique, où il a obtenu un immense succès. C'est pourquoi, peu de temps après Fortran, apparaissait en 1959 un langage pour la gestion : Cobol (COmmon Business Oriented Language).

Les deux frères ennemis Fortran et Cobol

Cobol se caractérise par un aspect beaucoup plus verbeux que Fortran, mais cet aspect a aussi des éléments positifs : les textes sont moins concis, mais moins secs qu'en Fortran. Ils ressemblent plus à de l'anglais naturel (c'est moins rébarbatif). On trouve en Cobol des phrases du genre ADD A TO B ou encore ADD P Q Giving R (qui peut se dire aussi Compute R = P + Q et est la traduction Cobol de notre exemple d'addition vu plus haut en Fortran et en assembleur 6809). Il y a même en Cobol des mots-clés supplétifs, c'est-à-dire facultatifs et qui ne sont là que pour mieux faire des phrases. Par exemple, on peut écrire soit Access Mode Is Random soit Access Mode Random.



Mais la caractéristique nouvelle de Cobol est sa puissance expressive, qui est beaucoup plus grande que Fortran pour ce qui est de la description des données à manipuler. Pour exprimer en Cobol qu'on trouve successivement sur une carte de données : le nom d'un individu formé du nom de famille et du prémon, sur deux fois vingt caractères, et son numéro de sécurité sociale (lequel a la structure connue : sexe, année de naissance, mois, département, etc.), on écrira :

```
01 CARTE.
02 NOM.
03 NOM FAM PIC X (20)
03 PRENOM PIC X (20)
02 NUM-SS
03 SEXE PIC 9.
03 AN PIC 99.
03 MN PIC 99.
03 DEP PIC 99.
03 COMMUNE PIC 999.
03 NUMERO PIC 999.
```

Une telle description est impossible en Fortran. Elle a été reprise pour les structures en PL/1 et les Record en Pascal. La description des données occupe en Cobol une division aussi importante que celle qui décrit les traitements.

Une autre notion apparue avec Cobol, qui prépare la transition vers la deuxième génération, est que le langage a été défini avec une norme, qui a évolué, mais reste une norme. Il n'est plus question de s'en remettre entièrement au bon vouloir des auteurs de compilateurs.

Aspect négatif de Cobol : son apparition a marqué un net clivage entre l'informatique dite scientifique et l'informatique dite de gestion. Cela est très nuisible. L'informatique est une, et la séparation en deux spécialités ne pouvant pas se comprendre (les uns parlent Fortran et les autres Cobol) n'a pas été facteur de progrès. Cela est en passe de se résorber maintenant, l'informatique individuelle ayant tendance à mettre tout le monde d'accord avec le langage commun Basic.

La deuxième génération de langages Algol, Apl et PL/1

Le premier langage ayant l'ambition d'une définition cohérente en tant que langage est Algol (Algorithmic Language apparu en 1958, et révisé en 1960 et 1968). La syntaxe est plus satisfaisante pour l'esprit que celle de Fortran, mais il est en définitive moins agréable d'emploi. Partout où l'on pourrait mettre une affirmation, on peut mettre une alternative, ce qui finit par être complexe.

Par exemple en Fortran (stylisé) on écrira :
Si (jour) dire « bonjour » ; si (nuit) dire « bonne nuit » ;
Si (homme) dire « monsieur » ; si (dame) dire « madame»
et en Algol (stylisé de même) on écrira :
dire « bon » si (jour) alors « jour » sinon « nuit » ;
si (homme) alors « monsieur » sinon « madame ».

Algol avait aussi la grave lacune de ne pas avoir normalisé d'instructions d'entrées-sorties. Elles ont donc été laissées à l'initiative des contructeurs, d'où une certaine anarchie. Algol a surtout eu une audience universitaire, n'ayant pas été soutenu par les constructeurs, qui ont préféré Fortran, mais il a pris une revanche posthume en étant l'inspirateur de Pascal.

De cette même époque datent Apl et PL/1. Apl est un peu marginal. Au départ, il s'agissait d'une simple notation mathématique inventée par Iverson (membre d'IBM) qu'on a eu l'idée de transformer en langage de programmation. On obtient un langage aux opérateurs innombrables, très concis, puissant et très satisfaisant pour l'esprit. Mais les notations finissent par apparaître très hiéroglyphiques et cela malgré la puissance de ce langage, due notamment au fait que toute opération s'étend aux tableaux sans qu'il soit besoin d'écrire des boucles.

Par exemple :
÷ M signifie inverser la matrice M,
□ ← V (↓ V ← □) signifie lire un tableau V et l'imprimer classé en ordre décroissant.

Apl n'a pas obtenu une grande diffusion. En effet, Apl est trop « intelligent » pour le programmeur moyen et, sauf dans quelques réalisations, notamment celle du Commodore 9000, Apl n'a pas de traitements de fichiers satisfaisants. En outre, il n'a pas été soutenu par son promoteur IBM qui, à l'époque, soutenait plus PL/1.

PL/1 a été développé de 1963 à 1966 par IBM avec l'ambition de présenter à la fois les avantages de Fortran et de Cobol (donc de réunifier l'informatique). Il s'agissait aussi pour IBM de montrer que la puissance de ses ordinateurs 360 permettait la réalisation d'un compilateur extrêmement complexe comme un langage aussi élaboré que PL/1 le nécessite.

L'ambition a été en grande partie réalisée : il est vrai que PL/1 est un langage parfaitement universel, apte à la fois au traitement scientifique et à la gestion. Du côté gestion, il possède un traitement adéquat des chaînes de caractères, permet les mêmes définitions de données structurées que Cobol et a accès à toutes les organisations de fichiers. Du côté scientifique, il recouvre les possibilités de calcul de Fortran, et a même plus de fonctions mathématiques. Sur un point, il est supérieur à Fortran : en Fortran, selon son problème, le programmeur doit opter pour les réels simple précision ou les réels double précision, et accepter la précision fournie par sa machine pour chacune de ces deux options ; en PL/1 le programmeur demande la précision qu'il souhaite pour sa variable, indépendamment de la machine.

Au moment où PL/1 semble triompher arrivent les Basic

Certains avaient annoncé que PL/1 supplanterait complètement Fortran. Il n'en a rien été parce qu'à la même époque, les mini-ordinateurs sont apparus. Or la richesse du langage PL/1 exigeait des compilateurs énormes (au minimum 100 K-octets de mémoire centrale), ce que les mini-ordinateurs étaient loin de posséder

alors que Fortran peut se contenter d'un compilateur sur 8 K-octets, fonctionnant en recouvrement. Fortran est alors devenu le langage par excellence pour les mini-ordinateurs dans le domaine scientifique (1967-1970).

A partir de 1965, et l'arrivée des mini-ordinateurs a accentué le phénomène, l'enseignement de la programmation s'est répandu considérablement. On a alors ressenti le besoin d'un langage aux possibilités réduites mais très facile à apprendre. Fortran, par exemple, est facile à apprendre mais il possède quand même certaines barrières anti-débutants comme le Format dans les instructions d'entrées-sorties (si vous voulez imprimer un nombre, il faut dire combien vous voulez de chiffres, il n'y a pas moyen en Fortran classique de dire « imprimez comme cela vient »).

Basic a été au départ défini, à l'université de Dartmouth, aux Etats-Unis, comme étant un sous-ensemble simplifié et réduit de Fortran et, de fait, le Basic de Dartmouth présentait beaucoup de limitations par rapport à Fortran. Basic (« élémentaire » en anglais) est le signe de « Beginners' All-purpose Symbolic Instruction Code » soit codage symbolique des instructions d'usage général pour les débutants ; il affiche donc bien sa vocation. Parmi ses simplifications, pour demander l'impression d'un résultat, il suffit de demander Print A ou même Print (expression arithmétique). Dans sa première version, tous les mots-clés de Basic étaient en trois lettres (par exemple Dim au lieu de Dimension), toute instruction devait commencer par un mot-clé (d'où l'existence de Let pour introduire l'instruction d'affectation). Ces éléments ont pour but de faciliter le travail du traducteur ; ils ont disparu des versions étendues.

De compilateur en interpréteur Basic s'accroche

Un autre élément, qui a fortement contribué à la popularité de Basic est que, au moment de son apparition, alors que les autres langages disposaient d'un compilateur, Basic utilisait un interpréteur. Rappelons qu'un compilateur traduit tout le programme source en bloc tandis qu'un interpréteur traduit et exécute les instructions du programme source une par une. Avec un compilateur, on doit, pour chaque correction à apporter au programme, passer par la séquence vue plus haut : édition, compilation, chargement, tandis qu'avec un interpréteur la mise au point des programmes est plus rapide. L'écriture d'un interpréteur étant plus simple, cela a rendu Basic accessible aux OI, d'où la domination que ce langage exerce maintenant sur le marché.

Mais cet élément n'est pas caractéristique du langage. De fait, tous les langages peuvent avoir, sur une machine donnée, à la fois un interpréteur (pour la mise au point) et un compilateur (pour voir une exécution rapide une fois que le programme est au point).

Le langage Basic insiste, s'étend et puis s'enracine

Nous avons employé le mot Basic au pluriel. En effet, à partir de la version Dartmouth, il a été d'une part créé une version encore plus restreinte (arithmétique entière) à l'usage des ordinateurs individuels bas de gamme « Tiny Basic », et de toutes sortes de versions étendues que l'on retrouve sur les OI courants. Ces versions sont très nombreuses et chaque constructeur prétend avoir la plus étendue.

Ces extensions portent Basic à un niveau largement égal à Fortran et même supérieur dans certains domaines importants comme le traitement des chaînes de caractères. Le seul domaine où Basic reste inférieur à Fortran est dans son mécanisme de sous-programmes. Un sous-programme Basic est appelé par un numéro, ce qui est moins parlant qu'un nom symbolique et surtout il n'y a pas de transmission de paramètres.

Par exemple, si vous avez écrit un sous-programme qui inverse la matrice A pour donner AA, pour inverser la matrice B en BB il faut recopier B en A, appeler le sous-programme puis recopier AA dans BB. En Fortran, on écrit Call In Mat (B, BB).

La diversité des versions de Basic nuit à la portabilité. De même, pour l'existence en Basic d'instructions d'appel de sous-programmes en langage machine (Sys ou Usr) et d'instructions d'écriture ou de lecture directe en mémoire (Peek et Poke).

Voici deux versions Basic du programme assembleur vu au début de cet article, qui calcule la somme des éléments d'un tableau d'entiers 16 bits. La première version a l'esprit « langage évolué » : le tableau est désigné par un nom A % et on ne se préoccupe pas de l'adresse :

```
10 DIM A %(640)
20 S % = 0
30 FOR I = 1 TO 640
40 S % = S % + A %(I)
50 NEXT I
```

La seconde version est plus proche du langage machine. Elle se préoccupe des adresses réelles où sont rangés les éléments du tableau (de 1 000 à 1 500 hexa soit 4 096 à 5 376) et le résultat (4 094 et 4 095) :

```
20 S % = 0 : DEBUT = 4 096 :
FIN = 5 376
30 FOR X = DEBUT TO FIN
40 S % = S % + PEEK (X) * 256
+ PEEK (X + 1)
50 NEXT X
60 POKE 4094, S %/256 : POKE
4095, S % ANN 255
```

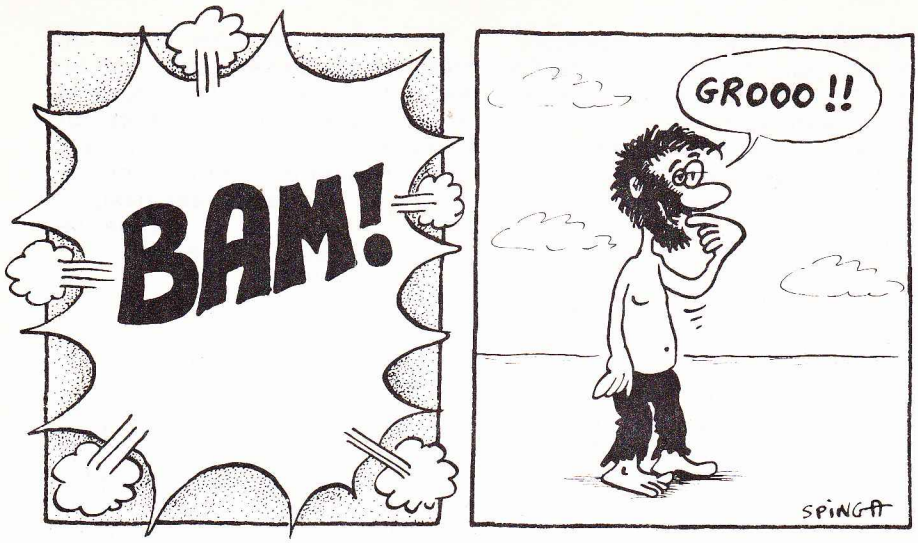
La seconde version est moins portable et moins générale que la première : tous les utilisateurs d'OI savent combien il est difficile de transposer les Poke astucieux d'une machine à l'autre.

La tendance des langages évolués est plus représentée par la première version, qui permet de s'abstraire des détails de la machine, mais l'accès à des ressources intimes de la machine peut être intéressant dans certains cas.

Et l'informaticien créa la programmation structurée

La programmation structurée est née, non pas des caractéristiques de Fortran, Cobol ou Basic, mais de l'utilisation que les programmeurs en faisaient. Avec les facilités qu'il offre, Basic permet de s'asseoir à son clavier et de se mettre à programmer sans avoir prévu une organisation rationnelle de son programme. Mais, si Basic le permet, il n'y oblige pas. Aucun langage n'oblige à programmer de façon désorganisée. Cependant, le fait est là : 90 % des premiers programmes de l'histoire de l'informatique étaient mal organisés et mal documentés, et donc impossibles à comprendre pour y faire une modification.

La programmation structurée est une discipline qui s'inspire de



ces réflexions pour conduire à des programmes bien organisés, bien documentés, sans erreur et faciles à relire. Les règles de la programmation structurée seraient presque des lapalissades, s'il n'était établi qu'elles ne sont pas naturellement suivies par les programmeurs. Rappelons ces règles, codifiées entre autres par Dijkstra en 1969 :

. tout programme doit être décomposé en sous-programmes très courts ;

. un sous-programme peut et doit n'être construit qu'à partir des trois « briques » suivantes : la séquence (inconditionnelle), l'alternative (de « si... alors... sinon... »), l'itération (boucles « tant que... faire... » ou « répéter... jusqu'à... ») ;

. le « GOTO » doit être évité (corollaire de la règle précédente) : c'est en effet à cause des GOTO que lorsqu'on relit un bout de programme on ne sait d'où on vient et donc on ne le comprend pas complètement ;

. le programme doit être abondamment commenté.

Il est clair que l'on peut faire de la programmation structurée avec n'importe lequel des langages vus jusqu'à maintenant. Ils offrent tous un moyen d'insérer des commentaires, ils ont tous un mécanisme de sous-programmes plus ou moins perfectionné, ils permettent tous de simuler les « briques » de la programmation structurée. Mais voilà, les programmeurs ne le font pas tant parce que, une fois que l'on a mis au point un programme, il est pénible de le reprendre pour le commenter, que par inertie et tradition de l'enseignement. Il est frustrant de se contraindre à respecter les briques de la programmation structurée quand le langage permet de s'en écarter et de faire des astuces. C'est pourquoi on a pensé qu'il était utile d'introduire un langage directement compatible avec la programmation structurée et conduisant naturellement les programmeurs à la pratiquer. Ainsi est né Pascal créé par Wirth en 1970.

Outre la programmation structurée pour laquelle il est construit en dérivant d'Algol, Pascal prend en compte un élément très important : il fait une place aussi large aux déclarations qui décrivent les données qu'aux instructions, (algorithmiques) qui écrivent les traitements. C'est la stricte application de l'axiome qui forme le titre d'un livre de Wirth ; Algorithms + data structures = programs. Et de fait, Pascal exige que toute donnée soit préalablement déclarée et que son type soit spécifié. Cette dernière disposition oblige le programmeur à penser et organiser son programme avant de s'asseoir à son clavier ; ce n'est pas une perte de temps, contrairement à ce que croient certains.

Le type d'une donnée définit sa structure, l'arithmétique qui s'applique à elles, éventuellement une contrainte d'intervalle. Pascal permet même à l'utilisateur d'inventer des types de données applicables à ses besoins, par exemple :

```
TYPE JOUR = (LUNDI, MARDI,
MERCREDI, JEUDI, VENDREDI,
SAMEDI, DIMANCHE) ;
VAR TODAY : jour ;
```

Un autre impératif qui a présidé à la définition de Pascal est la portabilité. Pour cela les concepteurs ont été jusqu'à fournir le compilateur, mais comme le langage avait quelques limitations, un autre compilateur est apparu en concurrence avec le Pascal Zurich, le Pascal Ucsd (University of California et San Diego). Ce dernier a notamment un meilleur traitement des chaînes de caractères.

Pascal a eu beaucoup de succès en milieu universitaire : il est le langage idéal pour enseigner l'informatique. Il a été très tôt accessible aux processeurs. Il est le deuxième langage le plus fréquemment employé sur les OI après le Basic et est plus compliqué à mettre en œuvre que Basic, mais il impose une meilleure organisation de programmes.

Le langage Comal ? Mieux que Basic !

En réponse à Pascal, certains Basic ont reçu des extensions fournissant les briques de la programmation structurée : c'est le IF... THEN... ELSE par exemple, et un mécanisme d'appel de sous-programmes avec transmission de paramètres. L'un d'entre eux, Comal (COMmon Algorithmic Langage), a été introduit au Danemark en 1980. Nous pensons que c'est un tort de l'avoir appelé d'un autre nom que Basic, car les utilisateurs craignent d'avoir à apprendre un nouveau langage alors que c'est en fait un Basic, très bien conçu. On peut le considérer aussi comme un Pascal ayant gardé la simplicité du Basic, vu qu'il est dépourvu des problèmes de types de données.

Nous en arrivons maintenant à l'époque contemporaine marquée par :

. l'arrivée de Ada du côté des langages puissants,

. l'éclosion de langages orientés vers une structure particulière de données (Lisp et Forth),

. l'arrivée de « langages de non-programmation » comme les logiciels ouverts et les générateurs de programmes (The Last One).

Mais oui, Pascal est bien le père d'Ada

Le Department of Défense des Etats-Unis constatant que, dans les organismes dépendant de lui, on utilisait plus de quatre cents langages différents, lança en 1978 un appel d'offres pour la définition d'un langage véritablement universel. C'est le langage proposé par une équipe en grande partie française dirigée par Jean Ichbiah qui a été choisi. Il a reçu le nom Ada en l'honneur de Ada Augusta comtesse de Lavelace, fille de Lord Byron et qui, dans

ses relations avec l'inventeur Charles Babbage, a mérité le titre de premier programmeur de l'histoire (cf l'article « Biographie du langage Ada » dans ce numéro).

Nous n'insisterons pas sur les caractéristiques de Ada qui ont déjà été décrites dans ces colonnes : Ada est le véritable prolongement de Pascal dont il reprend la problématique des types, en supprimant les limitations ; il va jusqu'au bout du possible en matière de modularité des programmes et de sécurité et permet les traitements simultanés et – caractéristique sur laquelle on insiste moins – est à la fois le langage le plus abstrait et le plus proche du problème de l'utilisateur, et le langage qui permet le meilleur contrôle de ce qui se passe au niveau machine grâce aux attributs et aux spécifications de représentation.

... et Ada, comtesse de Lovelace et fille de Lord Byron

Le point sur lequel nous voulons insister, c'est le processus de la définition de Ada. Un cahier des charges a été d'abord établi (en plusieurs étapes avec retouches successives) définissant bien les diverses qualités et potentialités souhaitées. Ensuite le langage a été défini en fonction de ce cahier des charges, indépendamment de toute contrainte de machine. Ce n'est qu'ensuite que l'on s'est mis à écrire les compilateurs, et d'ailleurs, vu l'énormité du langage, aucun compilateur complet n'est encore totalement générationnel à l'heure où nous écrivons.

En tout cas, cela est très représentatif de l'attitude moderne que l'on a vis-à-vis des langages : d'abord définir un langage cohérent en fonction des besoins et ensuite s'occuper du compilateur par opposition au premier Fortran, qui était le récueil des propriétés de son compilateur.

La deuxième tendance actuelle est celle des langages bâtis autour d'une structure de données particulière. Lisp est bâti autour de la structure de *liste chaînée.* Il n'est pas si récent que cela, ses premières esquisses remontant à 1968 environ. La caractéristique fondamentale de Lisp est sa possibilité de définir des opérations emboîtées qui reviennent à étendre le langage dans la direction que l'on veut. La deuxième caractéristique est sa grande aptitude à la manipulation d'expressions symboliques, qui, combinée à la première, fait de Lisp un langage idéal pour l'intelligence artificielle.

Le second langage représentatif de cette tendance est Forth (1970). Son nom veut dire « quatrième génération ». Forth est, lui, centré autour de la notion de piles, ce qui lui confère une grande aptitude à la récursivité. Une particularité est que Forth contient, dans son vocabulaire de base, les instructions du langage machine. Une deuxième caractéristique fondamentale est la possibilité de définir de nouveaux mots-clés du langage à partir des anciens, donc d'étendre le langage à volonté. Couplé avec la première possibilité, on voit que cette extensibilité n'est pas au détriment de la rapidité puisque les nouvelles opérations se font en langage machine. Cela explique le succès de Forth dans certains domaines, notamment certains contrôles de processus.

La troisième tendance actuelle est un peu paradoxale puisqu'il s'agit, en somme, *de ne pas programmer.*

Un premier élément est l'utilisation grandissante de *logiciels ouverts.* Ces logiciels ouverts sont à mi-chemin entre un langage de programmation qui permet de programmer n'importe quelle application et un logiciel d'application qui s'adresse à une application bien particulière. Ces logiciels permettent à l'utilisateur de spécifier son application parmi une classe d'applications définie.

Par exemple, Visicalc permet de gérer des tableaux de calculs sur écran ; il s'adapte donc à tout ce qui peut se présenter sous forme de tableaux de résultats mais pas aux problèmes de recherche dans des fichiers, donc tout ce qui est bilan, échéancier, etc. C'est l'utilisateur qui spécifiera si le tableau est un bilan, un échéancier de prêt, une prévision des ventes, etc., et qui indiquera quelles informations on met dans les cases et quelles relations il y a entre les cases.

Pour les problèmes de fichiers, outre Visicalc, il existe d'autres logiciels ouverts comme Ozz, Manager, Silicon, Office, Dbms, Versafile, etc.

Les générateurs de programmes ont une autre approche. Ils se présentent comme un questionnaire posé à l'écran, auquel l'utilisateur répond en spécifiant son problème : « Voulez-vous des fichiers ? » ou « Est-ce un nouveau fichier ? », « Quelles zones ? », « Entrée au clavier ? », « Voulez-vous un branchement ? », etc.

Ensuite, le générateur vous fournit le programme correspondant en Basic. Le plus typique de ce système s'appelle The Last One (le *dernier* programme jamais écrit, puisqu'il écrit les autres !). Il y a deux limitations :

. les programmes constitués font partie d'une classe définie, en gros la gestion de fichiers ;

. la réponse au questionnaire suppose de la part de l'utilisateur une certaine connaissance de l'informatique.

Enfin, il se pose la question de savoir si le programme obtenu est optimisé ou non.

Des langages pour éviter la programmation !

Au cours de cette revue chronologique, nous n'avons parlé que d'un nombre très réduit des quelque quatre cents langages de programmation et plus, actuellement en vie. Nous nous sommes concentrés sur les plus utilisés à un titre ou à un autre, en essayant de dégager les raisons qui avaient présidé à leur apparition. Quantitativement, Basic exerce une domination très nette, due à sa facilité d'accès sur ordinateurs individuels. Les principaux oubliés de notre description sont Simula (1967), langage de simulation et les langages d'écriture de systèmes. Ada est peut-être appelé à se substituer à eux. Il faudrait aussi penser aux langages d'enseignement Logo et Pilot.

Nous n'avons prédit la disparition d'aucun langage. A plusieurs reprises, le remplacement de Fortran était prévu, mais chaque fois, un événement arrivait pour lui donner un second souffle.

C'est pourquoi il nous semble que les nouveaux langages n'expulseront pas Basic, qui est trop agréable à mettre en œuvre. En revanche, il est possible que la version Comal de Basic s'impose par ses perfectionnements acquis sans sacrifier la simplicité de Basic. Vers le haut de gamme, Ada et ses sous-ensembles devraient s'imposer.

Daniel-Jean David

introduction au langage de programmation C

Il est des langages de programmation dont on parle beaucoup..., et nous ne vous ferons pas l'affront de les citer. D'autres, en revanche, sont très peu connus : parce qu'ils sont récents, donc encore peu répandus sur les OI, et parce que l'on ne trouve que quelques rares publications à leur sujet. Le langage « C » est l'un d'entre eux, et nous nous proposons donc de vous initier à ce langage de programmation général écrit et développé à l'origine sur le système d'exploitation Unix, sur un DEC-PDP/11, par Dennis Ritchie.

Le langage C est un langage tout à fait portable qui est disponible sur les « minis » et aussi sur un certain nombre d'ordinateurs individuels (Whitsmith C, Tiny C, etc.). Le langage de haut niveau est bien parti pour remplacer... l'assembleur !

Bien qu'il ait été conçu pour l'écriture de systèmes d'exploitation, c'est un langage de programmation général qui peut servir à l'écriture de programmes scientifiques, de gestion de bases de données ou encore de traitement de texte.

Les deux types de données disponibles sont de différentes sortes.

Les caractères :
char a, carac ;

Cette instruction déclare les variables « a » et « carac », qui seront des caractères disponibles sur la machine.

. Les entiers :
int entier ;
. Les nombres réels en virgule flottante, simple précision :
float flotte ;
. Les nombres réels en virgule flottante, double précision :
double deux fois ;

Des variables internes, externes ou globales

De plus, il existe un certain nombre de qualificatifs qui peuvent précéder les variables du type int : « short » et « long », qui définissent des entiers de longueurs différentes, et « unsigned », qui définit des nombres positifs ; on déclarera :
short int petit ;
long int girafe ;
unsigned int naturel ;

Pour faciliter la lecture, on peut assigner des valeurs aux variables dès leur déclaration :
float initial = 3.12e5 ;

Ces variables peuvent être internes (locales), externes ou globales. Les variables internes sont :
. automatiques (comme les valeurs locales des langages traditionnels ; à la sortie de la fonction, la variable perd sa valeur) ;
. statiques : elles conservent les valeurs pour un retour éventuel à la fonction.
int c ;
static int bouge-pas ;

Le langage « C » offre aussi deux autres types de classe de stockage : les spécificateurs « register » et « extern ». Les variables déclarées « extern » indiquent qu'elles ont été déclarées à l'extérieur du fichier courant.

Exemples : voici le contenu de deux fichiers, fich1.c fich2.c :

```
/* fichier 1:
    fich1.c Demonstration
    "extern" */
char nom[50];
unsigned int age;
   fonction1 (.....) { ..... }

   fonction2 (.....) { ..... }
/* fichier 2:
    fich2.c
    Demonstration "extern" */
 extern char nom[];
 extern unsigned age;
  fonction3 (.....) { ..... }
  main (.....)    /*
  programme principal */
  { ..... }
```

Comme nom et âge sont déclarés en tant que variables externes, on peut compiler fich 2.c séparément, le compilateur sachant que ces variables sont déclarées dans un module différent. De plus, il ne crée pas les variables et ne leur attribue pas de place en mémoire.

Nous remarquons aussi que, dans fich 2.c, il n'est pas utile de rappeler la longueur de la chaîne nom.

Le type « register » est l'un des plus intéressants, et « C » est aujourd'hui le seul langage de haut niveau qui offre ce type de stockage. La déclaration :

register int indice ;

indique au compilateur que la variable indice est une variable qui sera fréquemment utilisée. Dans la mesure du possible, les variables déclarées « register » seront placées dans les registres de la machine, ce qui permettra un gain de temps et de place mémoire.

Une grande facilité de manipulation des pointeurs

En plus de ces types de données, il est possible de déclarer des types dérivés que l'on peut créer avec des tableaux, des pointeurs, des structures, des unions et des fonctions.

Un tableau est déclaré de la manière suivante :
char nom [50] ;
int tab [30] ;
static float mat [3] [5] [7] ;

Ces instructions déclarent, dans fich 1.c et fich 2.c, un tableau statique « tab » de trente valeurs entières, un tableau tridimensionnel « mat » de taille 3*5*7 de valeurs relatives et une chaîne « nom » de cinquante caractères (déjà rencontrée dans fich1.c et fich 2.c). Les chaînes en « C » sont donc stockées caractère par caractère dans un tableau de type « char ». Pour marquer la fin de la chaîne il faut que le dernier élément soit le code ASCII (zéro) symbolisé en « C » par \0.

Ainsi, « Ordinateur Ind. » sera écrit :

.....

!O !r !d !i !n !a t !e !u !r !
!I !n! d !\0 !

.....

Une autre originalité du langage « C » est la facilité de manipuler des pointeurs. Un pointeur est une variable qui contient l'adresse d'une autre variable. Voici un exemple emprunté au livre *The C programming Language* de B.W. Kerningham et D.M. Ritchie (les concepteurs de ce langage), qui montre l'un des avantages de ce type.

Il s'agit d'écrire une fonction « PERMUT » (attention, « C » différencie les majuscules des minuscules) qui permute le contenu de deux variables : si a = 5 et b = – 32, il faut qu'au retour les valeurs soient a = – 32 et b = 5. On pourrait essayer de l'écrire de la manière suivante :

```
PERMUT (k,l)      /* MAUVAIS */
int k, l ;
{
        int temp ;
        temp = k ;
          k = l ;
          l = temp ;
}
```

PERMUT ne produira pas le résultat escompté : l'appel PERMUT (a, b) transmettra les valeurs de « a » et « b » par valeur. Pour obtenir le bon résultat, il faut transmettre à la fonction les pointeurs (références) des variables. Ainsi en C :

int ent ;
int * point ;

déclarent un entier (ent) et un pointeur (point). L'instruction :

point = & ent ;

assigne au pointeur (point) l'adresse de la variable (ent). Le résultat de l'opérateur monadique & est un pointeur de l'objet référé.

Il va de soi qu'il est impossible de se référer à l'adresse d'une variable déclarée « register ». Ainsi, si nous déclarons :
int ent 2 ;
et que nous assignons :
ent 2 = * point ;
nous obtenons un résultat équivalent à :
ent 2 = ent ;

La version correcte de notre programme s'écrira de la façon suivante :

```
PERMUT(ptr-k,ptr-l) /*
      Bonne version */
int *ptr-k, *ptr-l; {
      int temp;
      temp = *ptr-k;
      *ptr-k = *ptr-l;
      *ptr-l = temp;
```

Pour permuter a et b, il faut écrire : PERMUT (&a, &b).

Il existe en « C » une grande relation entre les tableaux et les pointeurs. L'expression tab (i + 1) est équivalente à *(tab + 1) ; tab est en fait un pointeur du premier élément du tableau, c'est pourquoi tab [10] réserve dix places de tab [0] à tab [9]. C'est la cause de certaines bogues : on peut dépasser les limites sans qu'aucune erreur ne soit détectée et cela produira des résultats imprévus.

En général un programme « C » se présentera comme ci-dessous:

```
int decl,gener;
char table[20];

 .......

fonction1(ars_f1)
déclaration ars_f1

{
        déclaration variables locales de f1
        corps de la fonction fonction1
}

fonction2(ars_f2)
déclaration ars_f2

{
        déclaration variables locales de fonction2
        corps de la fonction fonction2 }

main(ars_main)
déclaration ars_main
{
        déclaration des variables locales de main
```

```
corps de main (fonction principale)
```

Au début du fichier sont déclarées les valeurs globales du programme, ainsi les fonctions « fonction 1 », « fonction 2 » et « main » pourront s'y référer. Nous voyons ensuite la déclaration des fonctions : une première ligne présente son nom et ses arguments, la ligne suivante définit le type des arguments.

Des constructions de contrôle fondamentales pour la structuration

Dans le corps de chaque fonction on trouve en premier les déclarations des variables locales, puis la fonction proprement dite. Nous remarquons les accolades qui délimitent les fonctions, elles sont en fait synonymes (à peu de choses près) du BEGIN/END de Pascal.

Pour ma part, j'estime que { et } sont bien plus lisibles, d'autant plus que certains éditeurs d'écran vous montrent chaque fois que vous tapez } l'accolade que vous refermez.

Le langage « C » offre les constructions de contrôle fondamentales requises pour des programmes structurés. On peut citer le traditionnel « if » présent dans toutes ses variantes, « while » dans deux possibilités :

```
while (il_pleut > 0) {
        parapluie = STOP;
        }
```

ou avec test en fin de boucle :

```
do {
printf("Quel age as-tu ma
  belle ?");
scanf("%d",&age);
 } while (age>28) ;
```

qui est en fait un FAIRE TANT QUE mais avec test en fin de bloc. Dans cet exemple la question : « Quel âge as-tu ma belle ? » sera posée tant que la valeur « âge » sera supérieure à 28. Le langage « C » fournit aussi un mécanisme de sélection de cas :

```
switch (ch[i]) {
    case 'a' : case 'e' :
    case 'i' :
    case 'o' : case 'u' :
        ctr ++;
        }
```

qui rappelle le CASE (et non le pari) pascalien. D'autre part, la syntaxe des boucles FAIRE ---- POUR --- JUSQU'A est très puissante en « c ». La structure générale est la suivante :

for (exp 1 ; exp 2 ; exp 3) {
bloc 1 }

Exemple, le bloc :

```
for (i=0; i< 10; i++) {
        bloc1
}
```

exécutera bloc1 dix fois. [i=0] est l'initialisation ; [i<10] est le test de la boucle, et [i++] (i = i + 1) l'instruction à exécuter à chaque itération.

Il va de soi que [exp1] et [exp3] peuvent être formées de plusieurs instructions, ce qui permet d'écrire des boucles « for » très chargées comme :

for (i=0, k=18, j=20, b='a' ; (i<8 & et b= ='z') ; i++,k--)

Considérons maintenant l'ensemble des entrées-sorties

« C » est relativement petit et peut être défini rapidement. Un compilateur « C » simple et compact s'écrit facilement. Le revers de cette médaille est que « C » ne pourvoit pas des instructions d'entrée-sortie ni de traitement de fichiers.

L'utilisateur pourra accéder à ces fonctions spécialisées en invoquant des bibliothèques standards destinées à cet effet. De ce point de vue, nous pouvons dire que « C » est un langage de bas niveau. Par exemple, pour afficher un texte, on se sert de la fonction « printf » [print formated], et pour lire des données « scanf ».

scanf («%d %d %d», &i, &j, &k) ;
printf («\ni = %d\tj = %d\tk = %d.\n», i, j, k) ;

Le premier argument de « scanf » est une chaîne qui représente le format des données à lire ; ici, trois valeurs entières doivent être lues. Les arguments suivants doivent être des pointeurs aux variables à lire. Il faut insister sur ce point, car l'on a souvent tendance à l'oublier et ce sont encore de longues soirées perdues. [scanf («%d»,i) peut être correct si « i » est déjà un pointeur...].

Print f est moins problématique. Le premier argument est le texte à afficher, le premier %d provoque l'impression de la première variable, le second de la deuxième, etc. On peut aussi ajouter des caractères de contrôle comme \n, qui provoque un passage à la ligne ou \t un déplacement du tabulateur.

Il faut bien comprendre que ces fonctions n'appartiennent pas au langage, mais à la librairie d'entrées-sorties standards.

La présence d'un processeur rend les programmes très lisibles

Autre analogie avec l'assembleur, la présence d'un préprocesseur qui permet la définition de macros. Toutes les instructions du préprocesseur sont précédées du signe # en début de ligne :
define LONGUEUR 40
indique au processeur que toute occurrence du mot LONGUEUR

est à remplacer par la valeur 40, comme par exemple dans la déclaration :

char chaîne [LONGUEUR] ;

On peut écrire des macros plus complexes comme :

define MAX (A,B) (A)> (B) ? (A) : (B))

et, lors du prétraitement, la ligne :

num = MAX (num,j) ;

La ligne suivante arrivera au compilateur :

num = (num > j ? num : j) ;

Le préprocesseur permet de rendre les programmes plus lisibles. Notons une autre façon d'écrire des instructions de la forme :

SI ----ALORS ----SINON ----

qui devient en « C » :

exp1 ? exp2 : exp3

Dans cette expression conditionnelle exp1 est d'abord évaluée. Si le résultat obtenu est différent de zéro (donc vrai), c'est exp2 qui est évalué, sinon ce sera exp3.

Toujours à propos de la parenté du langage « C » avec l'assembleur, on peut noter les expressions :

i++, i--

déjà vues précédemment, qui seront traduites en assembleur par INC et DEC.

La syntaxe de « C » autorise des expressions comme :

j += 2 ;

qui est équivalent à :

j = j + 2

qui est très utile pour des expressions comme :

tab[j[k][1]] [j[k][2]] = tab[j[k][1]] [j[k][2]] + 2

que l'on écrit en « C » :

tab[j[k][1]] [j[k][2]] + = 2

Pour finir enfin avec cette présentation [non exhaustive] du langage « C », présentons les opérateurs logiques qu'il offre :

&	ET	diadique
|	OU inclusif	diadique
^	OU exclusif	diadique
<<	déplacement à gauche	diadique
>>	déplacement à droite	diadique
~	(complément à 1)	monadique

Exemple :

n = n & 0177

remet tout à zéro sauf les derniers bits de n.

A l'aide de I [OU inclusif] on peut inversement masquer des bits :

x = x I0177

Avec les opérations de déplacement on peut écrire le macro

#define allume (num-bit, var) ((var) L+ 1 << (num-bit))

qui allume le bit n.

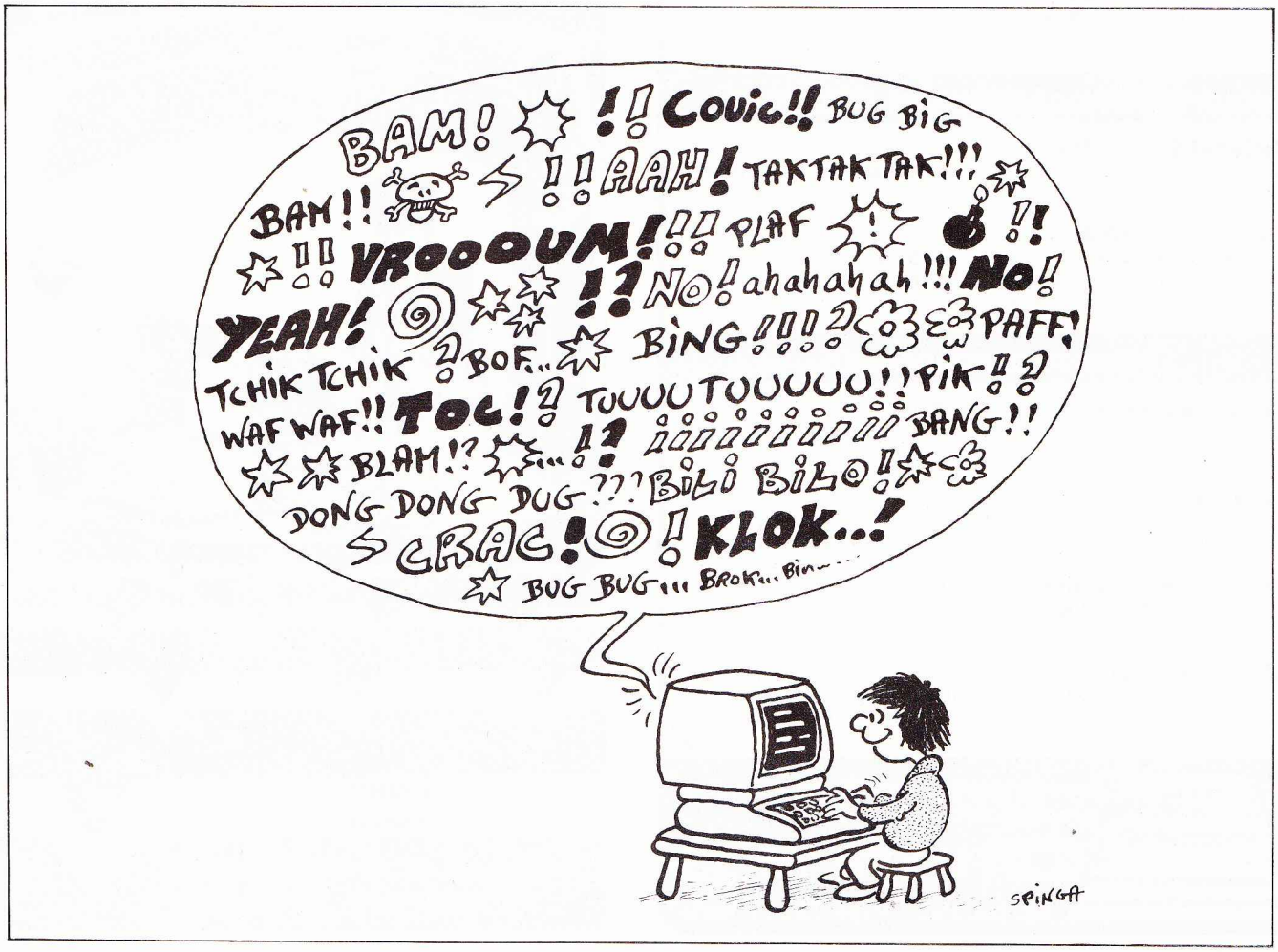
Le langage C, un langage puissant, proche de l'assembleur

Voilà terminé ce rapide et non exhaustif tour d'horizon du langage de programmation « C ».

Beaucoup d'intéressantes possibilités telles que type de f [définition de types] ou struct [définition de structures] n'ont pu être abordées.

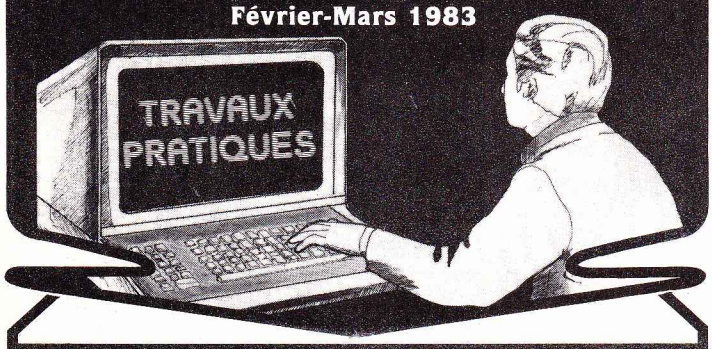
Nous y reviendrons mais nous pouvons, dès à présent, affirmer que « C » est un langage puissant, proche de l'assembleur.

Marc Kawam



SEMINAIRES MICRO-ORDINATEURS

Février-Mars 1983



POUR UTILISATEURS NON SPÉCIALISTES

CHOIX D'UN MICRO-ORDINATEUR :
2 jours - 14 et 15 février 1983. Le choix d'un matériel - Le choix d'un logiciel - La démarche à suivre.

VISICALC :
2 jours - 24 et 25 février / 17 et 18 mars 1983. Un progiciel remarquable pour la gestion d'entreprise. Principes de mise en œuvre, travaux pratiques sur micro-ordinateur par groupe de deux personnes.

LA BUREAUTIQUE :
2 jours - 17 et 18 février / 21 et 22 mars 1983. Le bureau du futur : ce qu'il faut savoir.

INTRODUCTION AUX BASES DE DONNÉES :
2 jours - 7 et 8 mars 1983. Les systèmes de gestion de bases de données. Cas pratique sur D Base 2.

POUR LES PROGRAMMEURS DÉBUTANTS

PREMIERS PAS EN BASIC :
3 jours - 21, 22 et 23 février / 8, 9 et 10 mars 1983. Premières notions et concepts du BASIC. Travaux sur micro-ordinateur.

FICHIERS EN BASIC :
2 jours - 28 février et 1er mars / 14 et 15 mars 1983. Pour réaliser des programmes mettant en œuvre les fichiers. Travaux pratiques sur micro-ordinateurs.

POUR PROFESSIONNELS

Le système d'exploitation UNIX :
2 jours - 14 et 15 février 1983. Caractéristiques, langages et utilitaires d'UNIX. Evaluation du système.

Le langage C :
2 jours - 16 et 17 février 1983. Pour les professionnels qui s'intéressent aux techniques de développement des logiciels de base.

Le système d'exploitation MS-DOS :
1 jour - 2 mars 1983. Origines - Objectifs - Langages et utilitaires - Commandes. Evaluation de MS-DOS.

LA TÉLÉMATIQUE :
3 jours - 21, 22, 23 février 1983. Un cours solide et bien construit pour maîtriser les techniques et les composants d'un réseau.

Inscriptions immédiates et programme détaillé : F.D.S.

(1) 588.76.53

10, rue Henri Pape 75013 Paris

Référence 158 du service-lecteurs (page 69)

CORIBEL Informatique

MICRO-ORDINATEURS POUR LES PARTICULIERS ET LES ENTREPRISES

Des prix bas sur:

LA GAMME COMMODORE : VIC 20 - CBM 4000
CBM 8000 - P 84 - P 500 - B 700 - BX 700
LA GAMME XEROX : 820 - 820 I - 820 II - 510 - 515
LES MACHINES A ECRIRE ELECTRONIQUES
XEROX : 610 - 615 - 620 - 625
LES MICRO PORTABLES PANASONIC

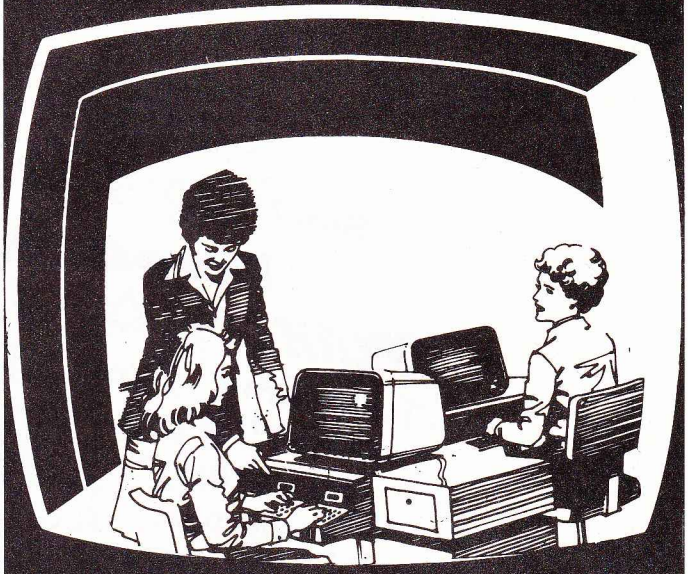
Un Service Clients COMPLET :

TOUS LES LOGICIELS STANDARDS disponibles sur les micro-ordinateurs XEROX et COMMODORE.
LOGICIELS SUR MESURE : Analyse des besoins en informatique - Cahier des charges - Mise en route, maintenance des matériels et des logiciels

Grand choix de livres d'informatique et de gestion.

FORMATION : Stages rémunérés en informatique.
Séances flash gratuites d'initiation.
Club micro-informatique.

GRAPHIKA



CORIBEL Informatique

95, av. du Général Leclerc,
75014 Paris tél. 543 72 14

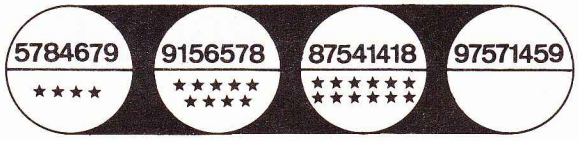
Référence 159 du service-lecteurs (page 69)

EXERCEZ VOTRE Q.I.

Ces quelques questions logiques sont extraites du concours mensuel du magazine des Jeux de réflexion Q.I., Jeux et Tests.

1

Quel nombre d'étoiles doit apparaître dans le dernier cercle ?



2

Quel est l'intrus dans cette suite logique de nombres ?

A/33857 B/34714 C/35428 D/35856 E/36715

3

Complétez le cercle vide avec l'un des nombres suivants :

A/734 B/682 C/399

4

Quel mot peut poursuivre la série ?

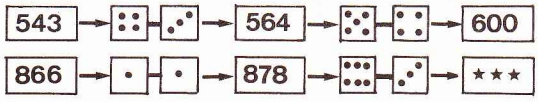
ACHAT - AGENT - ADMIS - ARGOT
ARGUS - ALPHA - APPEL - ACTIF

1/AVENT 2/ACCRU 3/ALORS

5

Remplacez les trois étoiles par le nombre qui convient.

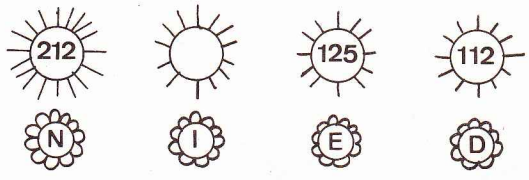
A/750 B/795 C/950



6

Quel nombre manque dans le deuxième soleil ?

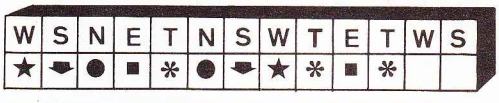
A/315 B/239 C/139 D/156



7

Quels symboles doivent apparaître, dans l'ordre, dans les deux cases vides ?

1/un point et une étoile 2/une flèche et une étoile 3/une étoile et une flèche.



Référence 175 du service-lecteurs (page 69)

UN P'TIT RHUME? UNE GROSSE FIÈVRE?

ISTC REMET RAPIDEMENT SUR PIED VOS SYSTEMES INFORMATIQUES.

Bien entendu, nous ne souhaitons pas que votre micro-ordinateur tombe en panne, mais nous avons quand même prévu cette éventualité.

LE SERVICE RÉPARATION

Le service réparation d'ISTC, installé sur 1500 m^2 au cœur de Paris est équipé en hommes et en matériels afin d'assurer rapidement la réparation de votre système informatique, unité centrale et périphérique. Nous avons en stock les pièces correspondant à toutes les grandes marques . **Unités centrales : APPLE, COMMODORE, GOUPIL, IBM, ISTC, ITT, REE-MICRAL, SHARP, SIRIUS, TRS, VGS.**
Imprimantes : CENTRONICS, DATAROYAL, DIABLO, EPSON, FACIT, OKI, QUME, SILENTYPE, TALLY. Moniteurs : NEC, PHILIPS, SANYO, SSV, THOMSON.
Table traçante : WATANABE. Visu : FALCO, GT 100, HAZELTINE, TVI. (marques déposées)
Nous savons où trouver les autres. Nous assurons le dépannage de ces systèmes quel que soit le lieu où vous les avez achetés.

Aucune surprise en ce qui concerne le coût de la réparation; un devis gratuit et immédiat vous est donné avant réparation, lorsque vous nous apportez le matériel défaillant. Il tient compte du prix des pièces à changer et d'un taux horaire fixe pour la réparation et le test. La réparation est bien sûr garantie.

LES MEILLEURS DÉLAIS D'INTERVENTION

Nous savons combien il est pénible d'être séparé de son micro-ordinateur. C'est pourquoi nous vous assurons les meilleurs délais d'intervention : un matériel déposé le matin à notre comptoir - 3, rue Ste-Félicité, PARIS 15e, sera repris le soir même.

Si vous êtes en province, il faudra juste ajouter le délai d'acheminement du transporteur que vous aurez choisi.
Si vous êtes moins pressé, nous pouvons aller chercher votre matériel.

Et puis, pourquoi attendre la panne pour s'assurer? ISTC propose des contrats de maintenance adaptés à votre problème, comprenant par exemple des visites préventives régulières.

Parce que nous sommes depuis 10 ans dans le domaine de la micro-informatique, nous savons que la maintenance est affaire de spécialistes. On ne bricole pas dans des domaines aussi sérieux et aussi techniques.

Alors, confiez la santé de vos micro-ordinateurs à ceux qui ont fait évoluer la micro-informatique professionnelle.

ARCANE Communication

Informatique Systèmes TéléCom
3, rue Ste-Félicité - 75015 PARIS
Tél. : (1) 532.80.01 - Télex 201 297 INSTEL

Référence 181 du service-lecteurs (page 69)

les TRucs du TRS-80

Rectificatif

Dans L'OI nº 39, le programme de l'article intitulé « deux trucs d'un coup » a été quelque peu malmené. Après analyse, et afin d'éviter des migraines inutiles à ceux qui auraient tenté de déchiffrer ce programme, voici la version revue et corrigée. ▼

```
403E            00100          ORG     403EH
403E E5         00110          PUSH    HL
403F F5         00120          PUSH    AF              ; on sauve HL et AF
4040 3EB9       00130          LD      A,0B9H          ; on change LOAD
4042 32E641     00140          LD      (41E6H),A       ; en CLOAD
4045 3EC3       00150          LD      A,0C3H          ; on met un JP RETOUR
4047 32B541     00160          LD      (41B5H),A       ; a l adresse ou on
404A 21540D     00170          LD      HL,RETOUR       ; retourne apres CLOAD
404D 22B641     00180          LD      (41B6H),HL      ;
4050 F1         00190          POP     AF              ;
4051 E1         00200          POP     HL              ; on restitue AF et HL
4052 C31F2C     00210          JP      2C1FH           ; on saute a CLOAD
4055 3EC9       00220 RETOUR   LD      A,0C9H          ; retour normal apres
4057 32B541     00230          LD      (41B5H),A       ; un CLOAD
405A 21E841     00240          LD      HL,41E8H        ; tampon d'entree
405D 368E       00250          LD      (HL),8EH        ; code du RUN
405F 23         00260          INC     HL              ;
4060 3600       00270          LD      (HL),0          ; fin du tampon
4062 2B         00280          DEC     HL              ;
4063 2B         00290          DEC     HL              ;
4064 C35A1D     00300          JP      1D5AH           ; saut a l'interpreteur
4189            00310          ORG     4189H           ; adresse de LOAD
4189 3E40       00320          DEFW    403EH           ; on branche en 403EH
06CC            00330          END     6CCH            ; retour au BASIC
```

Bonnes adresses et messages personnalisés

Les utilisateurs du Basic niveau 3 (extension du Basic commercialisée sur cassettes par Microsoft), pour peu qu'ils se livrent — grâce à quelques PEEK — à l'exploration de la mémoire vive, y feront quelques intéressantes découvertes.

. De 16 900 à 17 318 : tableau ASCII des vingt-quatre messages d'erreur, séparés par un zéro.
. De 18 090 à 18 151 : un message copyright.
. De 18 879 à 18 948 : un message Microsoft.
. De 20 388 à 20 401 : le message « undefined line » pour la renumérotation.
. De 21 065 à 21 479 : la table des assignations (14 octets libres par assignation, plus un zéro).

Cette découverte permet d'envisager un petit programme qui permettra de disposer de messages d'erreurs personnalisés et de ses assignations favorites.

On peut, bien sûr, modifier à volonté les messages des DATA de la ligne 20. On ajoutera ensuite les vingt-six assignations (séparées par des virgules). Pour assigner un NEW LINE (ENTER sur TRS), on entrera une flèche-curseur de passage à la ligne : le programme assure la conversion.

Sur VGS, on ajoutera également le code 23 qui provoque le message ? SN, et correspond en fait à « DISK Basic Feature » (ou L3 error sur TRS).

Sur VGS et TRS, on ajoutera aussi le code 24 qui est provoqué par un appel FN non défini.

▼ **Francis Pierot**

```
10 RESTORE : AD = 16900 : FOR A = 0 TO 23 : READ R$ : FOR I = 1 TO LEN (R$) : PRINT MID$ (R$, I, 1);: POKE AD, ASC (MID$ (R$, I, 1) : AD = AD + 1 : IF AD >17317 THEN 16 ELSE NEXT: POKE AD, 0 : PRINT : AD = AD + 1 : NEXT
11 PRINT « MESSAGES ERREUR -OK »
12 AD = 21065: A1 = AD: FOR A = 1 TO26 : READ R$: IF LEN (R$) >14 THEN R$ = LEFT $ (R$, 14)
13 FOR I = 1 TO LEN (R$): POKE AD, ASC (MID$ (R$, I, 1)) : IF MID$ (R$, I, 1) = CHR$ (10) THEN POKE AD, 13
14 PRINT MID$ (R$, I, 1);: AD = AD + 1: NEXT : PRINT: POKE AD, 0: A1 = A1 + 16: AD = A1: NEXT
15 CLEAR 50: NEW
16 PRINT « ATTENTION, MESSAGES-ERREUR TROP ENCOMBRANTS » : STOP
20 DATA PAS DE FOR, SYNTAXE, PAS DE GOSUB, MANQUE DATA, INCORRECTE, OVERFLOW, PLUS DE PLACE, LIGNE INCONNUE, INDICE INCORRECT, TABLEAU REDIMENSIONNE, DIVISION PAR ZERO
```

Truc TRèS court

Entrez le petit programme suivant :

```
10 A$ = « 1E10 0 »
20 X = VAL (A$)
```

Faites un RUN, et vous aurez un :? OU ERROR.

Faites un LIST : le deuxième guillemet de la ligne 10 a disparu..., mais le plus drôle se produit lorsqu'on fait un deuxième RUN. Celui-ci entraîne l'affichage successif de :

```
SN ERROR IN 10485
READY
UL ERROR IN 10485
READY
```

Cherchez l'erreur !

Régis Saleur

Un INPUT qui accepTe les fRactionS

Dans un même INPUT, un programme peut être amené à accepter des nombres entiers ou fractionnaires.

Malheureusement, les variables entrées ne peuvent être que des valeurs numériques calculées ou des chaînes de caractères, mais pas des fractions non calculées (exemple : 2/3, 4/5, etc.).

Le petit programme qui suit propose un assez bon moyen de résoudre cette difficulté, en remplaçant l'instruction INPUT par un équivalent qui accepte les expressions numériques et alphanumériques : USR (0). INPUT X est donc remplacé par X = USR (0).

```
ORG 7FE0H
CALL 36 36IH
INC HL
CALL 1BC0H
INC HL
CALL 2337H
RET
END
```

Ce sous-programme en langage machine doit, bien sûr, être appelé en mode programme.

On notera alors l'affichage — dans certains cas seulement — de messages d'erreurs : X = USR (0) <enter>donnera lieu à un : UL error ; et SIN (1) <enter>provoquera un SN error. Mais ces messages n'empêchent pas la frappe et l'exécution du programme.

Un bon programme pouvant en cacher un autre, le suivant permet de tester le fonctionnement du précédent et fournit un exemple d'utilisation.

```
10 M = 32736
20 POKE 16561,223 : POKE 16562,127
30 POKE 16526,224 : POKE 16527,127
40 FOR J = 0 TO 11
50 READ K : POKE M + J, K
60 NEXT J
70 CLEAR 50
80 DATA 205, 97, 3, 35, 205, 192, 27, 35, 205, 55, 35, 201
90 X = USR (0) :PRINT X
100 GOTO 100
```

On peut remplacer X par X$ ou X#. Mais en double précision, il faudra alors taper 2#/3 pour entrer 2/3, et 1,2 #pour 1,2 (par exemple).

J. Grappy

Variables et fichiers : double truc

Cette petite expérience, qui concerne les possesseurs du NewDos 80, leur révélera une bizarrerie du Basic (ou du reste ?).

On s'aperçoit qu'il est possible d'imprimer une variable non déclarée dans un fichier séquentiel indexé. Aucune erreur n'est détectée ni par le DOS ni au niveau du Basic.

En revanche, le « directory » de la disquette est altéré, ce qui empêche une bonne utilisation par la suite. Le petit programme ci-dessous illustre cette... « particularité ».

```
30 OPEN « O », 1,
« FICH/DAT : 1 »
40 PRINT #1, A
50 CLOSE
```

Si on y ajoute : 20 A = 0, l'erreur (de non détection d'erreur) ne se produit plus.

Toujours avec une variable (mais déclarée, cette fois), on peut utiliser la syntaxe de l'instruction CMD de la manière suivante : CMD A$. Cette utilisation, assez souple, tranche avec le CMD « instruction » traditionnel.

Cette astuce permet beaucoup d'applications, notamment l'attribution d'une protection à un fichier dont les noms et le mot de passe changent.

La variable A$ (dans l'expression CMD A$) est ainsi obtenue par concaténation du mot de passe, du nom du fichier et d'autres éléments, comme suit :

A$ = « ATTRIB, » + B$ + « ,ACC = PASSWORD, UPD = » + C$ + « ,INV »

B$ = nom du fichier
C$ = mot de passe.

Patrick Gassman

Sauvé !

Il peut arriver qu'à la suite d'une mauvaise initialisation, le secteur ∅ du « directory » soit endommagé. Si vous disposez du TRS DOS 2.3 B ou de NewDos 80 II, aucun problème : on fait Backup.

Si vous n'avez pas ces systèmes, la procédure de Backup se soldera par l'apparition d'un message de rejet :
« BACKUP REJECTED DISKETTES CONTAIN DIFFERENT PACK IDS ».

Tandy recommande, dans ce cas, d'effacer la disquette grâce à un démagnétiseur. En effet, Backup 2.3 vérifie l'égalité des noms et des mots de passe des deux disquettes. Si la disquette destination est formatée, il faut que les noms et les mots de passe des disquettes source et origine soient identiques.

Un examen approfondi de la documentation permet de découvrir le nom et le mot de passe principal : il s'agit respectivement de TRS DOS et de PASSWORD.

Il suffit alors de prendre une disquette quelconque, contenant ou non TRS DOS, et de la formater. On utilisera respectivement comme nom et comme mot de passe TRS DOS et PASSWORD (ou ceux de la disquette à copier).

Après formatage de la disquette, le Backup sera enfin possible.

Jacques Beyris

Ne pas perdre la main

Lors de la mise au point de programmes en langage machine ou en cas de mauvaise adressage dans un USR (X), il arrive que le processeur se perde dans les profondeurs de la MEV.

Le petit montage ci-dessous permet de reprendre la main sans avoir à couper le contact ni à perdre le contenu de la mémoire. Il suffit de relier à un bouton-poussoir les deux points du circuit imprimé.

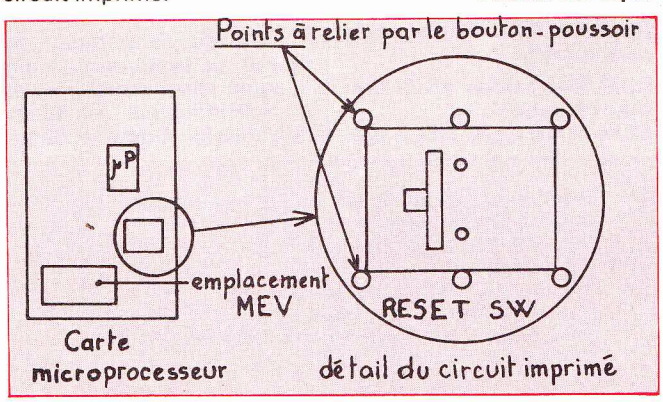
Lorsque l'on presse ce bouton, le processeur est réinitialisé et le message * SHARP MONITOR SP 1002 * apparaît.

Pour retrouver le Basic sans perdre le programme, il faut taper :
GOTO $ 1250.

Pour retrouver le langage machine, il faut taper :
GOTO $ 1260.

Dans les deux cas, si l'on désire faire un NEW, on tapera :
GOTO $ 1200.

Pascal Lévêque



Corriger la bogue du TAB

Sharp Basic (SP 50 25) fait un « Data Error » si l'on essaie de programmer ;

```
20 PRINT TAB (79) ;
« SHARP ».
```

On peut résoudre ce petit problème grâce à une ligne du programme ci-après.

```
POKE 10211, 127.
```

On dispose alors d'une marge de zéro à 256 caractères pour utiliser la commande TAB. Par exemple, faites donc :

```
20 PRINT TAB (256) ;
« SHARP ».
```

Scott Gordon

l'a.b.c. du p.e.t.

Quatre instructions de plus en Basic

Voici un petit programme en langage machine qui ajoute quatre instructions au Basic.

Ces quatre nouvelles instructions permettent de travailler facilement en résolution 4 000 points ; ce module occupe moins d'un demi K-octet ; il est ici implanté en bout de mémoire d'un 8 Ko sous Basic 2.0.

Une fois le module correctement recopié, il faut sauver le programme sur cassette à partir du moniteur langage machine par S « BASIC + », 01, 1E28, 2000.

Ensuite, on charge le programme à partir du Basic par un LOAD « BASIC + » et on tape un SYS 7720 pour exécuter le bootstrap (adresses 1E28 à 1E4C) qui réinitialise correctement la carte de la mémoire et modifie la routine de saisie de caractère du Basic.

Le curseur réapparaît alors et vous êtes maintenant dans le Basic modifié, c'est-à-dire acceptant les quatre nouvelles instructions de syntaxe suivantes :

: [, abscisse, ordonnée : qui allume les points d'abscisse et d'ordonnée fournies ;

:], abscisse, ordonnée : qui éteint le point considéré ;

!, abscisse 1, ordonnée 1, abscisse 2, ordonnée 2 : qui trace la droite approchée entre les deux points ;

@,abscisse 1,ordonnée 1, abscisse 2, ordonnée 2 : qui efface la droite approchée entre les deux points.

Les abscisses doivent être comprises entre 0 et 79 ; les ordonnées entre 0 et 49.

Les abscisses et les ordonnées peuvent être des constantes, des variables ou des expressions. On peut écrire, par exemple : !,A + B, C + D/2, SIN (E), LOG (F),pourquoi pas !

Les entêtes ne sont pas obligatoires en mode direct, mais le sont en mode programme. Il faut écrire par exemple 100 :[, 12, 12 ou 230 A = B + 2:], A, B

Ce module utilise peu d'adresses absolues (les poids forts sont soulignés), il est donc facilement transférable.

Il utilise trois routines internes du Basic 2.0: D123 pour afficher le message ILLEGAL QUANTITY ERROR, F460 pour saisir un paramètre après une virgule et 0070 pour saisir les caractères du texte source et qui est ici modifiée.

Il faudrait peut-être changer ces adresses (encadrées dans le texte hexa) pour le Basic 4.0.

Pour pouvoir mélanger des graphismes et du texte, ce module ne fait rien si la case considérée est occupée par un symbole alphabétique. On peut donc ainsi créer des décors évolutifs derrière un texte sans se préoccuper du texte lui-même, qui ne sera pas effacé.

```
1E28 A9 50 85 34 85 30 85 7A
1E30 A9 1E 85 35 85 31 85 7B
1E38 A9 4C 85 79 A9 03 85 2A
1E40 85 2C 85 2E A9 04 85 2B
1E48 85 2D 85 2F 60 EA EA EA
1E50 C9 5B F0 14 C9 5D F0 16
1E58 C9 21 F0 3A C9 40 F0 3C
1E60 C9 3A B0 03 4C 7D 00 60
1E68 A9 00 85 BD F0 04 A9 01
1E70 85 BD 20 70 00 20 60 F4
1E78 8A C9 50 90 03 4C 23 D1
1E80 85 C0 85 C2 20 60 F4 8A
1E88 C9 32 90 03 4C 23 D1 85
1E90 C1 85 C3 4C DB 1E A9 00
1E98 85 BD F0 04 A9 01 85 BD
1EA0 EA EA EA EA 20 70 00 20
1EA8 60 F4 8A C9 50 90 03 4C
1EB0 23 D1 85 C0 20 60 F4 8A
1EB8 C9 32 90 03 4C 23 D1 85
1EC0 C1 20 60 F4 8A C9 50 90
1EC8 03 4C 23 D1 85 C2 20 60
1ED0 F4 8A C9 32 90 03 4C 23
1ED8 D1 85 C3 D8 A9 01 85 C7
1EE0 85 C8 A5 C2 38 E5 C0 85
1EE8 C2 10 0B 49 FF 18 69 01
1EF0 85 C2 A9 FF 85 C8 A5 C3
1EF8 38 E5 C1 85 C3 10 0B 49
1F00 FF 18 69 01 85 C3 A9 FF
1F08 85 C7 A5 C7 85 C9 A5 C8
1F10 85 CA A5 C2 C5 C3 10 10
1F18 85 CC A5 C3 85 C2 A5 CC
1F20 85 C3 A9 00 85 CA F0 04
1F28 A9 00 85 C9 A5 C2 38 E5
1F30 C3 85 CB A5 C2 4A 49 FF
1F38 18 69 01 18 65 C3 85 CC
1F40 A5 C0 85 BA A5 C1 85 BB
1F48 A9 00 85 BC 85 BE A9 31
1F50 38 E5 BB 85 BB 46 BA 26
1F58 BE 46 BB 26 BE 06 BB 06
1F60 BB 06 BB A5 BB 06 BB 26
1F68 BC 06 BB 26 BC 18 65 BB
1F70 85 BB A5 BC 69 80 85 BC
1F78 A6 BE A9 01 85 BE E0 00
1F80 F0 05 06 BE CA 90 F7 A4
1F88 BA B1 BB A2 00 DD F0 1F
1F90 F0 08 E8 E0 10 90 F6 4C
1F98 B7 1F A5 BD D0 07 8A 05
1FA0 BE 18 AA 90 0A A5 BE 49
1FA8 FF 85 BE 8A 25 BE AA EA
1FB0 BD F0 1F A4 BA 91 BB A5
1FB8 CC 10 15 18 65 C3 85 CC
1FC0 A5 C0 18 65 CA 85 C0 A5
1FC8 C1 18 65 C9 85 C1 10 13
1FD0 38 E5 CB 85 CC A5 C0 18
1FD8 65 C8 85 C0 A5 C1 18 65
1FE0 C7 85 C1 C6 C2 A5 C2 10
1FE8 03 4C 76 00 4C 40 1F EA
1FF0 20 7E 7B 61 7C E2 FF EC
1FFF 6C 7F 62 FC E1 FB FE A0
```

Deux petits exemples de programme pour expérimenter ces nouvelles fonctions :

```
100 FOR I 0 TO 78
STEP 2:!, 40, 0, I, 25: @,
40, 49, 78-I, 26: NEXT
110 FOR I = TO 78 STEP 2:
@ , 40, 0 I, 25:!, 40, 49,
78-I, 26: NEXT
120 GOTO 100
```

```
100 A = 40: B = 25: X = 1;
Y = 1
110 :!, 0, 0, 0, 49:!, 0, 49,
79, 49:!, 79, 49, 79, 0:!,
79, 0, 0, 0
120 :], A, B: A = A + X: B
= B + Y: [, A, B
130 IFA = 78 ORA
= 1 THENX = – X
140 IFB = 48 ORB = 1
THENY = – Y
150 GOTO 120
```

Jean-Marc Geib

Lire tous les disques

Dans la gamme Commodore, il existe actuellement trois unités de doubles disquettes disponibles en France :

CBM 3040, DOS 1.0
CBM 4040, DOS 2.1
CBM 8050, DOS 2.5

Il y a des problèmes d'incompatibilité entre ces différents matériels. Dans certaines applications de gestion utilisant plusieurs types de disquettes, il peut s'avérer souvent bien utile de disposer d'un programme capable de reconnaître chaque type et d'éviter ainsi bien des erreurs.

Comment savoir si un programme travaille sur telle ou telle unité de disquettes ?

Pour répondre à cette question, nous allons lire le directory ($ 0 ou $ 1). Le premier caractère lu de ce fichier déterminera sur quelle unité a été formatée la disquette :

CBM 3040 Code ASCII premier caractère : 1 ;
CBM 4040 Code ASCII premier caractère : 65 (= « A ») ;
CBM 8050 Code ASCII premier caractère : 67 (= « C »).

Le programme en bas de page vous montrera mieux ce phénomène.

Cette méthode ne détermine pas directement sur quelle unité de disquettes on travaille mais sur quel matériel a été formatée la disquette.

On peut avoir des problèmes entre 3040 et 4040.

S'il est possible de lire à l'aide d'un 4040 une disquette formatée sur 3040 et réciproquement, il est dangereux d'écrire, plus particulièrement dans le cas où on aurait à écrire une disquette 4040 à l'aide d'un 3040. La méthode que nous venons de présenter est alors particulièrement nécessaire.

Comment un programme peut-il reconnaître une disquette ?

Lors du formatage, on fournit au système un nom (jusqu'à seize caractères) et un identificateur ID (deux caractères). Dans le direc-

```
10 REM DETERMINATION UNITE DE DISQUETTES
20 REM DENIS-HENRI PETIT
100 OPEN80,8,15,"I0"
110 OPEN87,8,7,"$0"
120 GET#87,A$:CLOSE87
130 IF A$=CHR$(1) THEN PRINT"CBM3040":END
140 IF A$="A" THEN PRINT"CBM4040":END
150 PRINT"CBM8050":END
READY.
```

	CBM 3040/4040	CBM 8050
Nom de la disquette ID	143 à 158 161, 162	5 à 20 323, 24

tory, ils occupent l'emplacement ci-dessus :

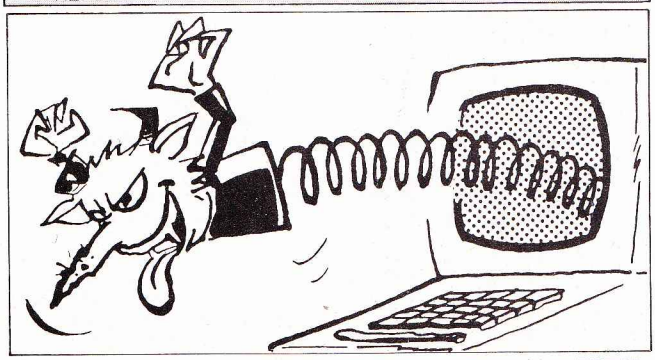
Lorsque le nom comprend moins de seize caractères, il est complété des « SHIFT SPACE » (code ASCII : 160). Il en est de même pour l'ID. A ce propos, il est déconseillé de mettre des « SHIFT SPACE » dans le nom ou l'ID d'une disquette.

Exemple : nous allons lire le nom et l'ID dans le programme ci-dessous.

On peut, bien sûr, combiner les programmes 1 et 2. Dans ces programmes on utilise le drive 0. Pour le drive 1, il faut remplacer I0 et $ 0 par I1 et $ 1.

Denis-Henri Petit

```
10 REM LECTURE NOM & ID D'UNE DISQUETTE
20 REM DENIS-HENRI PETIT
100 N=5:REM POUR CBM8050 SINON N=143
110 OPEN80,8,15,"I0"
120 OPEN86,8,6,"#0"
130 FOR I=1 TO N-1:GET#86,A$:NEXT I
140 N$="":FOR I=1 TO 16
150 GET#86,A$:IF A$=CHR$(160) THEN170
160 N$=N$+A$
170 NEXT I
180 GET#86,A$,A$
190 I$="":FOR I=1 TO 2
200 GET#86,A$:IF A$=CHR$(160) THEN220
210 I$=I$+A$
220 NEXT I
230 PRINT"■NOM:■ ";N$,"■ID■ ";I$
240 END
READY.
```



Puissances sur PC-1211

On ne compte plus le nombre de malheureux qui, calculant à la main, ont souffert des puissances dépassant la capacité de la plupart des calculatrices et ordinateurs individuels.

Les heureux possesseurs du PC-1211 voient aujourd'hui leur calvaire terminé. En effet, ce programme permet d'avoir, avec tous les chiffres significatifs, des nombres tels que 2^{100}, 199^{199}, 99^{99}, etc.

Le programme a été conçu de manière à utiliser le plus de mémoires possibles, sur les 204 que possède notre PC-1211. C'est pourquoi les entrées des données se font de la plus simple façon qui soit avec l'instruction « input » suivie de la variable (ligne 20).

Il y a, bien sûr, une limite de calcul : celle-ci est déterminée (ligne 20) par rapport au nombre de chiffres que peut contenir le Sharp.

Pour tout a^b, il est facile d'obtenir le nombre de chiffres par cette formule : int (b × log a) + 1 — (int étant la partie entière).

Pour a > 99 et a < 1 000 (c'est-à-dire un nombre de trois chiffres), l'affichage du résultat se fera par groupe de sept chiffres.

Les huit premières mémoires étant prises par le programme, il reste exactement 164 mémoires à sept chiffres = 1 148 chiffres. D'où int (b × log a) + 1 devra être inférieur à 1 148 (a et b sont nécessairement des entiers positifs).

Comment entrer les données au clavier ? On compte d'abord le nombre de chiffres de « a » sans les zéros éventuels (ainsi 210 ne fait que deux chiffres).

A ce stade des opérations, il convient d'adapter le programme au calcul de la puissance à l'aide du tableau ci-dessous.

Pour "a" (sans les zéros) avec :	1 chiffre	2 chiffres	3 chiffres	4 chiffres	5 chiffres
ligne 20	> 1 476	> 1 312	> 1 148	> 984	> 820
ligne 30	> 9	> 99	> 999	> 9 999	> 99 999
ligne 80-90	E-9 et E9	E-8 et E8	E-7 ; E7	E-6 ; E6	E-5 ; E 5

```
10    :  "M"CLEAR:D=1 :C=1
20    :  INPUT P,H,B:I=F:IF INT (H*LOG F)+1>
         1148 GOTO 20
25    :  IF E=0 PRINT 1:END
30    :  IF F>999 GOTO 20
40    :  FOR A=1 TO D
50    :  A(8+A)=A(8+A)*F
60    :  NEXT A
70    :  FOR A=1 TO D
80    :  E=A(8+A)*E-7
90    :  A(8+A)=(E-INT E)*E7
100   :  A(9+A)=A(9+A)+INT E
110   :  NEXT A
120   :  D=D+SGN INT E
130   :  G=G+1
140   :  IF H>G GOTO 40
150   :  B=B*H : BEEP 3
160   :  FOR C=D TO 1 STEP -1
170   :  PRINT A(C+8):NEXT C
180   :  PRINT "ET 5;B;"ZEROS
```

Suivant le nombre de chiffres de « a », on va apporter quelques modifications qui vont permettre d'avoir un maximum de chiffres significatifs, donc a^b sera dans tous les cas (ou presque) optimal.

Au-delà de 5, il convient de s'armer d'un bon crayon et de beaucoup de courage. Sans oublier beaucoup de papier pour tous les calculs !

Après Shift M, le premier point d'interrogation est le a de a^b (enlever les zéros). Le deuxième correspond à b, et le troisième est le nombre de zéros de a. Si l'on obtient un quatrième point d'interrogation, c'est que la puissance demandée est trop élevée pour le PC-1211.

Exemple : pour 20^9, je supprime le zéro. Il reste un chiffre : 2. Je me reporte au tableau dans la colonne « 1 chiffre ». J'effectue en mode PRO les modifications aux différentes lignes. Puis je fais Shift M en mode DEF.

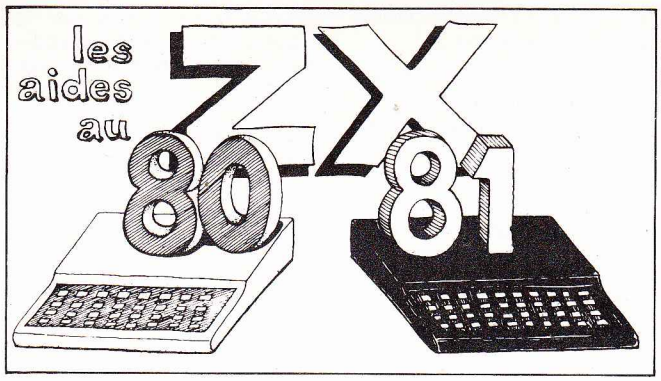
J'entre : 2 - « Enter », puis 9 - « Enter », puis 1 (il y a 1 zéro pour 20). Quelques secondes d'attente et j'obtiens 512. « Enter » - « et 9 zéros » . Donc 20^9 = 512 000 000 000.

A titre de vérification, 2^{100} = 1 267 650 600 228 229 401 496 703 205 376.

Si b est élevé, il faudra attendre assez longtemps, pour obtenir un résultat exact à l'unité près.

Pour vérifier ou revoir un résultat on fera : RUN 160. Si l'affichage ne donne par exemple que sept chiffres au lieu de neuf, on le complète à gauche par autant de zéros que nécessaire (ici deux zéros). Bon amusement !

Philippe Kouyoumdjian



Calculez votre biorythme

Voici un petit programme sur ZX-81 permettant de calculer :
. le biorythme d'un mois complet ;
. le jour de la semaine correspondant à une date donnée ;
. le nombre de jours écoulés entre deux dates.

Quelques rappels sur le biorythme. Trois cycles régissent notre vie :

. le cycle physique (PHY) qui dure vingt-trois jours ;
. le cycle émotionnel (EMO) qui dure vingt-huit jours ;
. le cycle intellectuel (INT) qui dure trente-trois jours.

Le jour de notre naissance, ces trois valeurs sont à zéro. Elles varient de façon sinusoïdale à partir de ce jour entre + 1 et − 1. Une valeur positive est favorable au cycle concerné, une négative est défavorable.

On calcule aussi la moyenne de ces trois valeurs (MOY). Au bout de 21 252 jours (soit cinquante-huit mois et une vingtaine de jours), les « pendules » sont de nouveau à zéro (21 252 = 23 × 28 × 33).

Le programme calcule le nombre de jours α écoulés depuis votre naissance et le pourcentage de biorythme de chaque cycle :
sin (2 × π × α /23) pour PHY,
sin (2 × π × α /28) pour EMO,
sin (2 × π × α /33) pour INT.

Multiplier par 100, prendre la partie entière et rediviser par 100 limite la réponse au centième.

Attention, car si vous transcrivez ces formules sur une autre machine, la fonction INT peut donner des résultats différents. Sur la ZX-81, la fonction INT correspond à la véritable fonction mathématique « partie entière », dont la définition est : le plus grand entier inférieur au nombre. Donc INT (− 2,5) = − 3,
car − 3 < − 2,5 < − 2.

En revanche, sur d'autres machines (comme la TI-58) :
INT (− 2,5) = − 2.

Donc, il peut y avoir un décalage (particulièrement pour les jours de la semaine). Les formules viennent de Gauss et du manuel de la TI-58 (module de base).

Une dernière remarque : le programme ne dessine pas le biorythme à l'écran, les valeurs sont données numériquement. Vous devrez donc tracer votre biorythme sur une feuille.

François Hache

```
    1 REM "BIORYTHME"
    2 REM AUTEUR FRANCOIS HACHE
    3 PRINT "APPUYEZ SUR LA TOUCH
E : "
    4 PRINT TAB 6;"1 POUR AVOIR V
OTRE BIORYTHME"
    5 PRINT TAB 6;"2 POUR CONNAIT
RE LE JOUR DE LA SEMAINE CORRESP
ONDANT A UNE DATE DONNEE"
    6 PRINT TAB 6;"3 POUR CONNAIT
RE LE NOMBRE DE JOURS ECOULES EN
TRE DEUX DATES"
    9 PAUSE 4E4
   10 POKE 16437,255
   12 IF INKEY$="1" THEN GOTO 17
   13 IF INKEY$="2" THEN GOTO 300
   14 IF INKEY$="3" THEN GOTO 500
   15 CLS
   16 GOTO 3
   17 CLS
   18 PRINT "ENTREZ VOTRE DATE DE
 NAISSANCE"
   20 GOSUB 1000
   30 IF M<=2 THEN LET X=W
   40 IF M>2 THEN LET X=Z
   50 PRINT
   60 LET J=1
   70 PRINT "ENTREZ LE MOIS ET L
ANNEE DU MOIS DONT VOUS VOULEZ L
E BIORYTHME"
   80 GOSUB 1030
   90 IF M=1 OR M=3 OR M=5 OR M=7
 OR M=8 OR M=10 OR M=12 THEN LET
 N=31
   95 IF M=4 OR M=6 OR M=9 OR M=1
1 THEN LET N=30
  100 IF M=2 THEN LET N=28+(A/4=I
NT (A/4))
  110 IF M<=2 THEN LET Y=W
  120 IF M>2 THEN LET Y=Z
  130 LET F=Y-X
  140 CLS
  150 PRINT "BIORYTHME DU MOIS :
";M;"/";A
  160 PRINT
  170 PRINT
  180 PRINT "J";TAB 4;"PHY";TAB 1
1;"EMO";TAB 18;"INT";TAB 24;"MOY
"
  190 PRINT
  199 REM CALCUL DU BIORYTHME DU
MOIS
  200 FOR K=0 TO N-1
  210 LET P=INT (100*SIN (2*PI*(K
+F)/23))/100
  220 LET E=INT (100*SIN (2*PI*(K
+F)/28))/100
  230 LET I=INT (100*SIN (2*PI*(K
+F)/33))/100
  240 LET M=INT (100*(P+E+I)/3)/1
00
  250 PRINT K+1;TAB 3;P;TAB 10;E;
TAB 17;I;TAB 24;M
  260 NEXT K
  299 STOP
  300 CLS
  305 GOSUB 1000
  310 IF M<=2 THEN LET H=W
  320 IF M>2 THEN LET H=Z
  329 REM DETERMINATION DU JOUR D
E LA SEMAINE
  330 LET D=H+(INT (1-H/7)*7)
  340 IF D=7 THEN LET K$="SAMEDI"
  350 IF D=1 THEN LET K$="DIMANCH
E"
  360 IF D=2 THEN LET K$="LUNDI"
  370 IF D=3 THEN LET K$="MARDI"
  380 IF D=4 THEN LET K$="MERCREDI"
  390 IF D=5 THEN LET K$="JEUDI"
  400 IF D=6 THEN LET K$="VENDRED
I"
  410 PRINT AT 20,0;"LE ";J;"/";M
;"/";A;" EST UN ";K$
  499 STOP
  500 CLS
  510 PRINT "ENTREZ LA PREMIERE D
ATE"
  520 GOSUB 1000
  530 IF M<=2 THEN LET X=W
  540 IF M>2 THEN LET X=Z
  550 PRINT
  560 PRINT "ENTREZ LA DEUXIEME D
ATE"
  570 GOSUB 1000
  580 IF M<=2 THEN LET Y=W
  590 IF M>2 THEN LET Y=Z
  600 LET F=ABS (Y-X)
  610 PRINT AT 15,0;"  LE NOMBRE
DE JOURS ENTRE CES  DEUX DATES E
ST : ";F
  999 STOP
 1000 REM CALCUL DU FACTEUR
 1001 PRINT "JOUR : ";
 1010 INPUT J
 1020 PRINT J
 1030 PRINT "MOIS : ";
 1040 INPUT M
 1050 PRINT M
 1060 PRINT "ANNEE : ";
 1070 INPUT A
 1080 PRINT A
 1090 LET W=365*A+J+31*(M-1)+INT
 ((A-1)/4)-INT (3*(INT (((A-1)/10
0)+1))/4)
 1100 LET Z=365*A+J+31*(M-1)-INT
 (0.4*M+2.3)+INT (A/4)-INT (3*(IN
T (A/100)+1)/4)
 1110 RETURN
```

Rectificatif

Dans les aides au ZX, page 198 du n° 39 de L'OI, traitant de la routine de formatage « PRINT USING », une anomalie a été découverte pour les nombres décimaux ayant un zéro immédiatement à droite après la virgule. Par exemple, le nombre 743,06 doit être lu 743,60.

Voici le programme modifié qui supprimera ce défaut.

BT

```
PRINT USING

   1 INPUT X
   2 LET X$=STR$ INT ((ABS X-INT
 ABS X)*10**D+.5)
  10 LET X$=("-" AND SGN X=-1)+S
TR$ INT ABS X+"."+("0000")(LEN X
$ TO D-1)+X$
  20 PRINT (" "+X$)(LEN X$
+8-I TO )
  30 GOTO 1
```

Allongez vos PRINT

Vous avez dû remarquer que, dans un PRT A$+B$+C$+D$+E$, E$ ne peut comporter plus de deux caractères si A$... D$ en contiennent sept sous peine d'erreur – 2.

La solution consiste à faire : PRT A$+B$+C$; D$+E$...

Jacques Chaillot

Pour aller à la ligne

Si à la suite d'un PRT se terminant par un point-virgule, vous désirez passer à la ligne, il est tout à fait possible, bien que ce ne soit pas dit dans le manuel, de faire : 10 PRT, sans argument.

Serge Boisse

Chiffrement avec clef « une fois »

Ce programme, pour lequel il est nécessaire d'utiliser la cassette, permet de chiffrer et déchiffrer rapidement jusqu'à cent seize lettres ou chiffres. Le cryptogramme se compose de groupes de dix chiffres avec point décimal central (un groupe pour six lettres claires).

Les clefs sont constituées par des blocs de 186 nombres aléatoires compris entre 3176 782 336 (= 3616 + 1E9) et 9 999 999 999, et stockées sur cassette et dans les mémoires A (4) à A (189).

Les valeurs initiales sont les suivantes.

1) Sur la cassette « chiffrement », chaque bloc de clefs est accompagné des valeurs initiales : A..Z = 10..35 et A0 = 3, et du programme de chiffrement en P0 et P1.2.

2) Sur la cassette « déchiffrement » on trouve avec chaque bloc de clef les valeurs :

$ = « 0123456789ABCDEFGHIJKLMN », B$ = « OPQRST »,

C$ = « UVWXYZ », et Y = 3, ainsi que le programme de déchiffrement en P0 et P1.

La procédure est la suivante : brancher le magnétophone à cassette et l'imprimante, choisir sur la cassette la page correspondant à la clef à utiliser, et taper « LOAD ALL » (clefs + variables + programmes), puis « F1 P0 ».

Chiffrement : tapez le message en clair en frappant EXE après chaque lettre ou chiffre. La lettre « K », tapée comme séparatif, sera remplacée par un espace ou déchiffrement.

Le message « 6 LETTRES » est affiché au départ, puis toutes les six lettres.

En fin de message, taper quelques lettres nulles si nécessaire, jusqu'à l'apparition de ce message.

Ensuite, taper « F1 P1 » pour déclencher l'impression du cryptogramme.

Déchiffrement : on tape les groupes de dix chiffres du cryptogramme avec leur point décimal central ; une fois cela terminé, « F1 P1 » imprime en clair.

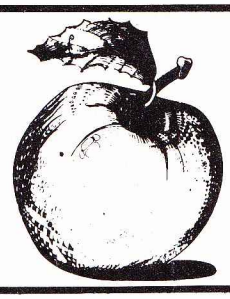
Les nombres aléatoires seront choisis par RAN. Ce procédé, dit avec clef « une fois », est parfaitement hermétique.

Pierre Barnouin

```
CHIFFREMENT
P0: 95 STEPS
LIST
   1 INP "6 LETTRES"
     ,A3
   2 A0=A0+1:A1=0:A2
     =0
   3 A1=A1+1:A2=36*A
     2+A3
   4 IF A1<6:INP A3:
     GOTO 3
   5 A(A0)=A(A0)-A2:
     GOTO 1

P1: 40 STEPS
LIST
   1 SET F5:MODE 7
   2 FOR Z=4 TO A0
   3 PRT A(Z)/1E5;
   4 NEXT Z:MODE 8
```

```
DECHIFFREMENT
P0: 133 STEPS
LIST
   1 INP Z:Y=Y+1:Z=A
     (Y)-Z*1E5:A$(Y)
     =""
   2 FOR X=1 TO 6:W=
     INT (Z/36):V=Z+
     1-36*W:Z=W
   3 A$=B$:IF V>30;A
     $=C$:V=V-6
   4 $=MID(1,24)+A$:
     A$(Y)=MID(V,1)+
     A$(Y)
   5 NEXT X:GOTO 1
P1: 26 STEPS
LIST
   1 MODE 7:FOR Z=4
     TO Y:PRT A$(Z);
     :NEXT Z:MODE 8
```



l'apple épluché

Concaténation de deux programmes

La concaténation de deux programmes Applesoft ne peut normalement s'opérer que par l'intermédiaire du programme « RENUMBER » figurant sur la disquette DOS 3.3 SYSTEM MASTER. Ce programme étant lui-même un programme Basic (enfin apparemment...) il faut commencer par le charger.

On utilisera ensuite ce petit programme en langage machine. Débutant à l'adresse 768 ($300), il n'interfère aucunement avec le programme Basic en mémoire.

Après avoir chargé le premier programme à concaténer, on modifiera le pointeur de début de programme pour qu'il soit égal à celui de la fin de programme.

On chargera ensuite le deuxième programme à concaténer et l'on remettra enfin le pointeur de début de programme à sa valeur initiale. On aura réalisé la concaténation des deux programmes.

Voilà pour le principe ! Dans la pratique, on s'assurera avant toute opération que les numéros de ligne des deux programmes à concaténer sont complémentaires. Sinon on risquerait de rencontrer certains problèmes ! Ensuite, après avoir assemblé le programme page 200 (de pré-

```
ADRS   CODE HEXA    DESASSEMBLEUR

300    A4 AF        LDY $ AF
302    A6 B0        LDX $ B0
304    4C 10 03     JMP   0310
307    86 68        STX $ 68
309    84 67        STY $ 67
30B    60           RTS
30C    00           BRK
30D    00           BRK
30E    00           BRK
30F    00           BRK
310    A9 03        LDA #$ 03
312    8D 29 03     STA $0329
315    C0 00        CPY #$00
317    F0 09        BEQ  $322
319    88           DEY
31A    CE 29 03     DEC $ 329
31D    F0 E8        BEQ $ 307
31F    4C 15 03     JMP $ 0315
322    A0 FF        LDY #$FF
324    CA           DEX
325    4C 19 03     JMP $0319
328    00           BRK
329    00           BRK
32A    00           BRK
32B    00           BRK
32C    00           BRK
32D    00           BRK
32E    00           BRK
32F    00           BRK
330    A9 01        LDA #$01
332    A2 08        LDX #$08
334    85 67        STA  $67
336    86 67        STA  $68
338    60           RTS
```

férence avec un assembleur ou au moniteur) et l'avoir sauvé (« BSAVE CONCATENATION, A$300, L$17), on chargera le premier programme à concaténer, on tapera « CALL 768 ». Après avoir chargé le deuxième programme on tapera ensuite « CALL 782 ».

Olivier Ternam

Bonne adresse, bonne formule

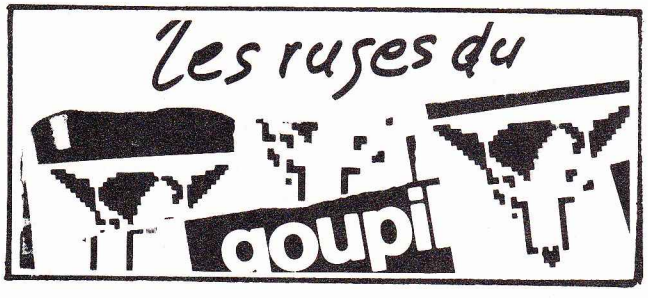
Dans la rubrique L'Apple épluché (L'OI nº 20), S. Goluch proposait trois formules permettant le calcul des adresses mémoire de la page 1 du texte. En réalité une seule formule est nécessaire :
PS = 1024 + X + 128 * Y –984 * INT (Y/8).

Un POKE, PS, CA (CA = code ASCII du caractère à afficher) visualisera ce caractère au point de coordonnées X et Y (0 ⩽ X ⩽ 39, 0 ⩽ Y ⩽ 23).

Suivant la valeur de CA (entre 0 et 255) le caractère affiché sera en mode FLASH, INVERSE ou NORMAL :
0 ⩽ CA ⩽ 63 : INVERSE
64 ⩽ CA ⩽ 127 : FLASH
127 ⩽ CA ⩽ 191 : NORMAL
192 ⩽ CA ⩽ 255 : NORMAL

La page deux de texte, difficilement utilisable en Applesoft, est facilement accessible en Basic entier. La formule à utiliser sera :
PS = 2048 + X + 128 * Y-984 * (Y/8).

Roland Jost



Majuscules ou minuscules sur Goupil

Beaucoup d'utilisateurs attendaient avec impatience une solution pour l'entrée des chaînes de caractères au clavier. En effet, comment savoir à l'avance si l'utilisateur va entrer son message (ou sa réponse) en majuscules ou en minuscules ? Il était donc nécessaire de tester tous les cas possibles.

```
1ø FOR I = 1 TO 25          25 CLIGNOTEMENTS
2ø POKE 59425,163           ETEINDRE LA DIODE
3ø FOR J = 1 TO 120 :       BOUCLE DE
   NEXT J                   TEMPORISATION (VITESSE)
4ø POKE 59425,16ø           ALLUMER LA DIODE
5ø NEXT I                   RETOUR
```

Il suffit de « forcer » l'utilisateur à entrer tous ses messages soit en majuscules, soit en minuscules à l'aide des POKE suivants :

. pour forcer le clavier en majuscules : POKE 58385, øø et si l'on veut effectuer cette opération entièrement, il est nécessaire d'allumer la petite diode de la touche majuscule : POKE 59425, 16ø.

. de même, pour forcer le clavier en minuscules : POKE 58385,255 et pour éteindre la diode : POKE 59425,163.

Le fait d'allumer ou d'éteindre cette diode n'est pas obligatoire, mais permet de ne pas tromper l'utilisateur sur l'état de son clavier, car ces opérations que l'on peut effectuer par programme remplacent l'action de la personne qui est devant le clavier.

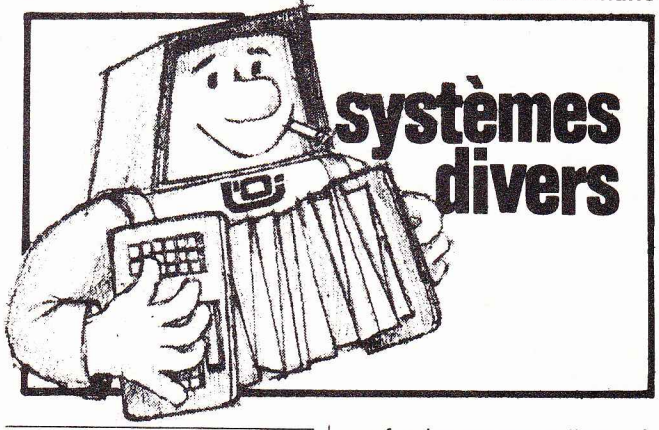
Le petit programme ci-dessus est un exemple amusant d'utilisation de ces astuces, car il permet de faire clignoter la diode de la touche majuscule.

Toujours à propos du clavier, une astuce bloque le clavier des lettres (y compris la touche ↵) : POKE 59425,ø et une autre bloque le clavier numérique, les différentes flèches, les caractères de traitement de texte et ceux qui se trouvent en haut du clavier ([*] / \... etc.) : POKE 59441, ø.

Et, pour terminer, voici un « truc » pour supprimer l'affichage des caractères sur l'écran : POKE 59411,ø et pour revenir en mode normal : POKE 59411, 127.

Ces différentes astuces peuvent être utilisées sur tout Goupil 5 pouces 16 × 64 et pour tous les Basic .

Gilles Romano



Prenez vos distances

Voici un petit bricolage très simple, qui permet de faire faire de substantielles économies aux « DAIstes » désireux de ne pas avoir de troubles visuels en travaillant sur leur matériel.

En effet, le DAI est relié au téléviseur par un cordon (prise péritel côté télévision, prise DIN six broches, mâle, côté DAI) mesurant 1,20 m (cordon livré avec le DAI). Multisoft, cependant, fournit des cordons (du même type) de 2,40 m et 3,60 m de long, mais il faudra alors débourser pour ce dernier la modique somme d'environ 445 FF.

La solution la plus économique, en fait, est de confectionner une rallonge à l'aide de câble blindé (six fils + masse) et de deux prises DIN six broches (une mâle + une femelle) et de la placer entre le DAI et le cordon d'origine.

une prise DIN six broches mâle	6 F
une prise DIN six broches femelle	6 F
n mètres de câble blindé	n × 7 F
Total pour une rallonge de 4 m (longueur totale ainsi obtenue : 5,20 m)	40 F

Attention, les composants ne sont pas très faciles à se procurer, mais le montage est très simple.

FICHIERS sur ATOM

```
Fichier d'adresses - Fichier téléphone - Fichier membres d'un CLUB - Listes
diverses ...   (TRI alphabétique sur 5 caractères).

  Ce FICHIER est compatible cassette ou disquette. Sa capacité dépend de la
taille mémoire utilisable. Le programme est placé en #2900 et le FICHIER :
  - de #8200 à #97FF pour un ATOM non étendu (5,6 Koctets) -
  - de #3C00 à #7FFF pour un ATOM  étendu    (17,4 Koctets) -

          menu

   I = INPUT au clavier
   S = SAVE Fichier
   L = LOAD Fichier
   R = RECHERCHE d'une fiche
   C = CARACTERISTIQUES du Fichier
   E = EFFACEMENT TOTAL du Fichier
   M = MODIF/KILL d'une fiche
   P = PRINT de tout le Fichier
   T = TRI alphabétique
   A = AJOUTER une ou plusieurs fiches
   Q = QUIT = FIN

            VOTRE CHOIX ?
```

LIST du programme pour un ATOM non étendu

(on utilise la mémoire visue graphique)

```
>L.
   10V=#8202;M=#2800;Z=#21C;REM TRI ALPHABETIQUE SUR 5 LETTRES
   20DIMLL100;REM A MODIFIER SI NECESSAIRE
   30oP.$12'
   40P."   menu"'" I = INPUT AU CLAVIER"'
   50P." S = SAVE FICHIER"'" L = LOAD FICHIER"'
   60P." R = RECHERCHE"'" C = CARACTERISTIQUES DU FICHIER"
   70P." E = EFFACEMENT TOTAL"'" M = MODIF/KILL D'UNE FICHE"'
   80P." P = PRINT FICHIER"'" T = TRI ALPHABETIQUE"'" A = AJOU"
   90P."TER"'" Q = FIN"';IN.'"          VOTRE CHOIX "$Z
  100IF$Z="I"G.i
  110IF$Z="S"G.s
  120IF$Z="L"G.l
  130IF$Z="M"G.m
  140IF$Z="P"G.p
  150IF$Z="E"G.e
  160IF$Z="A"G.a
  170IF$Z="T"GOS.h;G.t
  180IF$Z="R"G.r
  190IF$Z="C"G.c
  200IF$Z="Q";E.
  210G.o
  220iP.$12"ENTREES AU CLAVIER"'"MAX=5630 CARACTERES"'
  230IN.'"NOMBRE DE CHAINES "$Z;A=VALZ;?#8200=A
  240IN."LONGUEUR MAX. D'UNE CHAINE "$Z;S=VALZ;?#8201=S;B=1
  250IF S%A>5632 P.$7$7;G.i
  255b@=2;J=A;F.I=B TO J;P.I
  260IN.$V;IF L.V>S-1;?#8200=I-1;P.'"ERREUR"$7$7$7;GOS.h;RUN
  270V=V+S;N.;GOS.h;G.o
  275pGOS.g;@=2;GOS.u;GOS.v;P.$14"    TITRE"'$27"@";GOS.v;?230=0
  280F.I=#8202TO F S.S;P.1+(I-#8202)/S" "$I';GOS.v;N.
  290zP.'$3$7$7"     TAPEZ UNE TOUCHE ";LI.#FFE3;G.o
  300t!(F+S)=-1;J=-1;F.I=#8202TO F+S S.S;J=J+1
  310LLJ=(?I-65)*810000+(?(I+1)-65)*27000+(?(I+2)-65)*900
  320LLJ=LLJ+(?(I+3)-65)*30+?(I+4)-65;N.
  330E=0;J=-1;F.I=#8202TO F S.S;J=J+1;IFLLJ<=LL(J+1);G.n
  340$M=$I;$I=$(I+S);$(I+S)=$M;E=1
  350nN.;IF E=1;G.t
  360G.o
  370sP.$12"SAUVEGARDE DU FICHIER"'"(APRES TRI ALPHABETIQUE)"''
  380P."UTILISEZ LA TOUCHE COPIE"'';@=0
  390P."NOM DU FICHIER (7 CARACTERES) :"';IN.$M
  400?#8200=A;?#8201=S
  410P.'" *SAVE"""$M"""8200 "&(#8202+A*S)'$11;E.
  420lP.$12"CHARGEMENT DU FICHIER"'
  430P."NOM DU FICHIER (7 CARACTERES) :"';IN.$M
  440P."UTILISEZ LA TOUCHE COPIE"'" *LOAD"""$M"""8200"'
  450P.'" G.o"'$11$11$11;E.
  460rP.$12"RECHERCHE D'UN NOM"';IN."QUEL NOM "$M;@=2;X=1
  470xL=L.M;Y=0;GOS.h;F.I=#8202TO(#8202+A*S)S.S
  480F.J=0TO L-1;IF M?J=I?J Y=Y+1
  490N.;IF Y=L P.1+(I-#8202)/S" "$I';C=I
  500Y=0;N.;IF X=0 G.y
  510G.z
  520mGOS.u;P.$12"   MODIFIER UNE FICHE"';IN."QUEL NOM "'$M
  530X=0;G.x
  540yP."ENTREZ LA MODIFICATION :"'"   ( K=KILL LA FICHE)"'
  550P.1+(C-#8202)/S;IN.$C;IF $C="K";GOS.k
  560G.z
  570eF.I=#8200TO#97FFS.4;!I=#20202020;N.;!#8200=#D0100;GOS.h
  580GOS.u
  590c@=0;GOS.h;GOS.u;P.$12"LE FICHIER CONTIENT :"'A
  600P." FICHES DE "S" CARACTERES"';G.z
  610hA=?#8200;S=?#8201;F=#8202+(A-1)*S;R.
  620aGOS.u;P.$12"AJOUTER DES FICHES"'
  630IN."COMBIEN DE FICHES A AJOUTER "$Z;D=VALZ;B=A+1
  640A=A+D;V=F+S;F=#8202+(A-1)*S;?#8200=A;G.b
  650kA=A-1;?#8200=A;GOS.h;F.I=C TO F S.S;$M=$(I+S);$I=$M;N.;R.
  660g?230=0;GOS.h;%P=1
  665IN."IMPRIMANTE (O/N) "$Z;IF$Z="O"P.$2;%P=0
  670uIF A=0 P."FICHIER VIDE"';LI.#FFE3;RUN
  680R.
  690vIF%P=0;F.X=1TO90;P."-";N.;P.'
  700R.
   22/9/82          F2PE
>
>P.&TOP
    3322>
```

Alors amis DAIstes, rangez votre portefeuille et sortez votre fer à souder ! Résultats garantis !

Henri Giacomel

Gestion de fichiers sur Atom

◀ Voici, pour votre Atom, un petit programme de gestion de fichiers, utilisé sur minidisquette et qui est également utilisable sur cassette sans aucune modification. Il permet d'abandonner les GET BGET, SPUT, SGET, etc., en écrivant directement en mémoire.

Le tri et la recherche sont rapides, ce qui est bien utile pour gérer un fichier d'adresses, de téléphone ou de listes diverses.

Joseph Pino

Astuces pour New-Brain

Voici un « truc » destiné aux « New-Brainistes ».

Pour tous les possesseurs de cet ordinateur avec clavier Azerty, la notice (pour clavier Qwerty) ne convient pas toujours. Le Shift/ Escape est remplacé par Contrôle/0. Pour stopper un Verify, Load, Save, «*» sera remplacé par « ».

Pour vérifier le restant de mémoire disponible, il faut faire : ? FREE (ce qui n'est pas évident sur la notice).

Le meilleur moyen pour visualiser les 512 caractères disponibles est d'écrire le petit programme :

```
10 FOR I = 1 to 255
20 PUT 27, i, 26
30 NEXT I
40 END
```

1er type de caractères
puis agir sur Contrôle/W, puis b (2e type de caractères) ;
puis agir sur Contrôle/ W, puis H (3e type de caractères) ;
puis agir sur Contrôle/ W, puis J (4e type de caractères).

D'autre part, à la suite de l'essai du New-Brain (L'OI n° 42), on peut faire les remarques suivantes :

. en mode graphique, il est possible de dessiner un cercle par l'instruction PLOT ARC (D, O), avec D = longueur de l'arc et 0 rotation de l'angle au centre.

. l'instruction PEEK (X) (OX 65535) fonctionne très bien avec un « PRINT » ou « ? » devant le PEEK.

Alain Sanchez

Formation continue à la micro-informatique

PHOTO GUNHILD BULL

Tous nos informaticiens viennent de l'informatique traditionnelle, et en maîtrisent totalement les langages classiques : Assembleur, COBOL, FORTRAN... Ils utilisent leur professionnalisme et les méthodes de l'informatique pour réaliser des **applications professionnelles en micro-informatique.** Nous vendons des micro-ordinateurs sans programme. Nous vendons aussi des micro-ordinateurs avec les programmes. Il s'agit de programmes réalisés par la société KA, dont nous **garantissons la qualité** et le bon fonctionnement.
Nos formateurs enseignent l'informatique. L'enseignement de la micro-informatique nécessite des **formateurs professionnels,** suffisamment de **matériel** pour que **chacun puisse pratiquer,** un **support de cours** couvrant non seulement l'enseignement diffusé, mais permettant au participant de **s'auto-former** après le stage. Nous avons déjà accueilli de nombreux stagiaires, d'horizons et de centres d'intérêts divers : chefs d'entreprise, universitaires, professions libérales, informaticiens, musiciens compositeurs, retraités, cadres de grandes entreprises, revendeurs de micro-ordinateurs...

Nous proposons 5 possibilités :

■ **Stage de 2 jours bases de données.**
Comment utiliser les progiciels :
• bases de données
• manipulateurs de nombres et générateurs de tableaux
• générateurs d'états imprimés
Application pratique (un 48 K + un lecteur de disquettes pour deux participants).
Après ce stage, on peut générer, à partir de progiciels, un programme totalement adapté à son application en moins d'une journée de travail.
Ce stage nécessite de connaître la manipulation de l'APPLE II, ou d'avoir suivi au minimum la journée d'initiation.

Dates 21-22 mars
27-28 juin
Prix 2000 F h.t.

■ **Stage de 1 semaine de programmation BASIC.**
Il débute par la journée d'initiation.
Le stage permet d'assimiler la logique de programmation et de l'appliquer (un micro-système 48 K pour 2 participants).
En fin de stage, on sait établir un programme de gestion de fichier avec consultation en temps réel.
Ce stage ne nécessite pas de connaissance de départ en informatique.

Dates
du 17 au 21 janvier
du 21 au 25 février
du 14 au 18 mars
Prix 3850 F h.t.

■ **Stage 3 jours disquettes.**
Consacré à l'organisation, à la programmation et à l'exploitation de **fichiers sur disquettes magnétiques,** à travers l'étude du Disk Operating System APPLE II.
Travaux pratiques sur micro-systèmes (un 48 K + un lecteur de disquettes pour deux participants).
Ce stage nécessite :
• soit d'avoir suivi le stage de 1 semaine de programmation au préalable ;
• soit d'avoir une bonne connaissance théorique et une sérieuse pratique de BASIC de l'APPLE II.

Dates du 28 février au 2 mars
du 25 au 27 juillet
Prix 3080 F h.t.

■ **Journées de sensibilisation et stages de formation à Paris et en Province.**
Ils sont organisés à la demande
• d'une instance régionale telle, par exemple, une Chambre de Commerce ;
• d'un organisme de formation dans le cadre d'un cycle plus vaste de formation ;
• d'une entreprise.
La société KA installe le matériel pour la durée de la formation, assure la formation et fournit les supports de cours.

■ **Journée d'initiation** - Dates : 17 janvier, 21 février, 14 mars. Prix 700 F h.t.

Le nombre de places pour chaque stage est strictement limité, à la fois pour la qualité de l'enseignement et par les contraintes du matériel.
Deux animateurs sont présents pour aider les participants à la réalisation de leurs programmes.
Un support de cours très complet est remis à chaque participant.
Pour la journée d'initiation et pour les stages, les déjeuners sont pris en commun et compris.

l'informatique douce*

Renseignements et inscriptions à KA - Programme détaillé sur demande.
212 rue Lecourbe 75015 Paris - Tél. 533.13.50.
Le calendrier 82/83 est disponible.

* "L'informatique douce" est une marque déposée de la société KA

Référence 182 du service-lecteurs (page 69)

"qualimetric" la souplesse

Le symbole "qualimetric" témoigne de la parfaite adéquation des supports magnétiques BASF avec votre ordinateur. Du FlexyDisk au disk pack, BASF vous offre la souplesse d'utilisation de produits de pointe. Pour BASF, la qualité est une spécialité. Quant à la sécurité, elle est indispensable.

qualimetric

BASF la qualité sur mesure

Chaque support magnétique BASF est mis au point spécifiquement, fabriqué soigneusement et sévèrement contrôlé. Une garantie ? la position particulière de BASF dans le monde : spécialisée en chimie et en physique, maîtrisant les relations complexes matériel/supports, autonome pour les matières premières et les méthodes. La qualité de haut niveau est la vocation première de BASF.

Compagnie Française BASF
Dépt Matériels Supports Informatiques
140, rue Jules-Guesde
92303 Levallois
Tél. (1) 730 55 00

BASF BASF

Orchestra

Pour toutes précisions sur la société ou le produit présenté ci-dessus : Référence 102 du service-lecteurs (page 69)

ENCORE PLUS DE MEMOIRE

C'est vrai !
Le micro-ordinateur AVT 2
est entièrement compatible
avec l'Apple *

SPECIFICATIONS

- 64 K de RAM standard extensible par cartes de 256 K (1 Mbyte maximum)
- Microprocesseur 6502
- 16 K de ROM (mémoire morte)
- Sortie vidéo composite N/B
- Carte couleur RGB
- Affichage 24 lignes de 40 colonnes en N/B ou couleur
- Affichage graphique N/B 280 x 192 ou 280 x 160 avec 4 lignes de texte
- Affichage graphique 16 couleurs 40 x 48 ou 40 x 40 avec 4 lignes de texte
- Affichage graphique 6 couleurs 280 x 192 ou 280 x 160 avec 4 lignes de texte
- Clavier complet détachable 65 touches
- 7 connecteurs compatibles Apple * pour carte d'extension
- 2 lecteurs de disque 5 1/4 en option
- Interface cassette et poignée de jeux
- 4 sorties «Annunciator»

* Apple et Apple II sont des marques déposées de Apple Computer Inc.

AVT comp 2

IMPORTATEUR EXCLUSIF:

GENERALE ELECTRONIQUE SERVICES
68 et 76 avenue Ledru Rollin
75012 PARIS
Tél. : 345 25 92
Télex: 215 546 F GESPAR

Pour toutes précisions sur la société ou le produit présenté ci-dessus : Référence 103 du service-lecteurs (page 69)