LISTI

LE JOURNAL DES AMATEURS DE PROGRAMMATION

n°6

ISSN 0761-9936

JANVIER/FÉVRIER 1985

GROS PLAN SUR QUATRE LOGICIELS

- Gas-Kit pour C.64 : dessin et musique
- Un petit Logo pour les Oric
- Haute résolution sur ZX 81
- PC-Vision, super-éditeur pour PC-1500

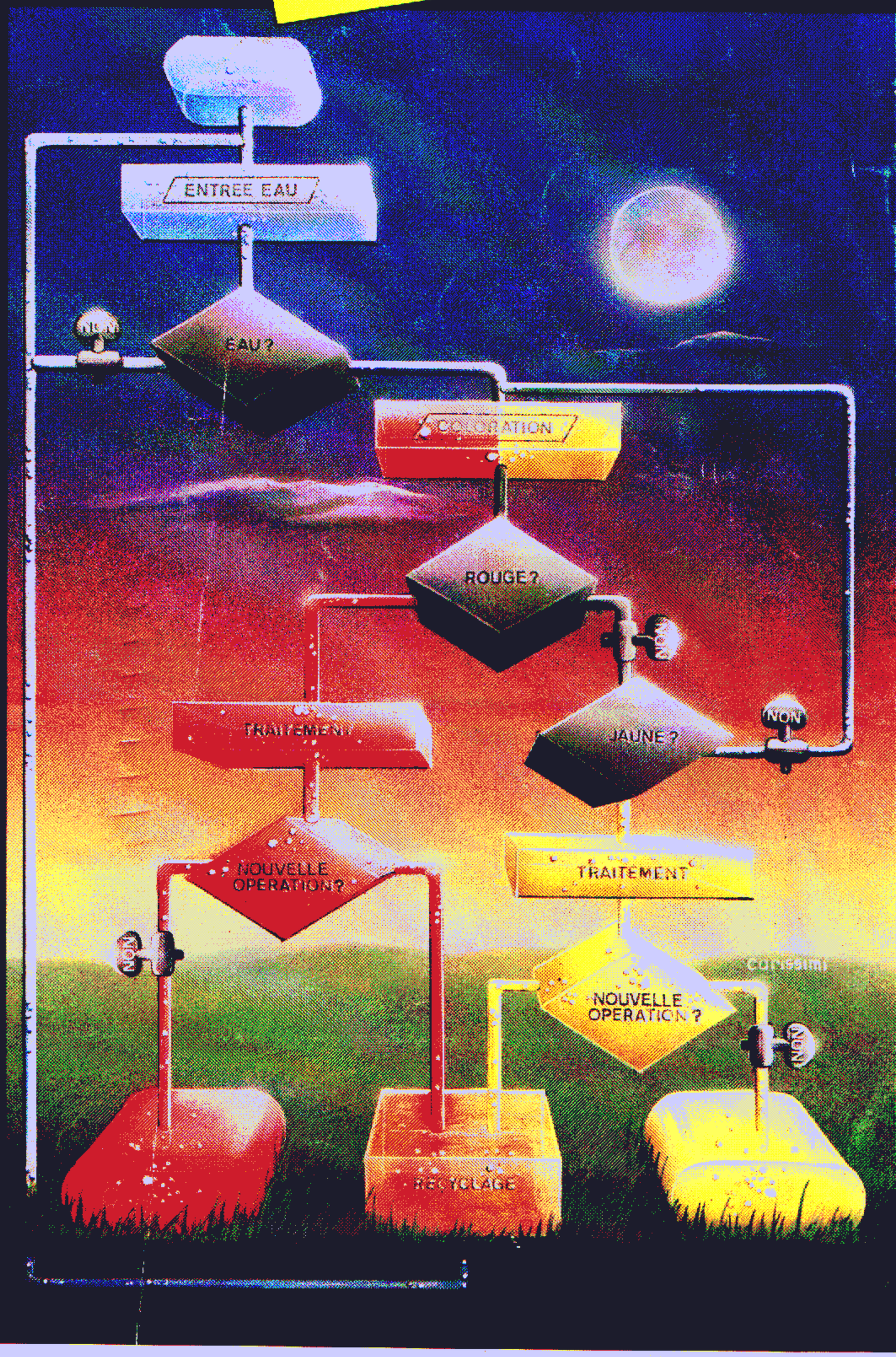
CURIEUX, CURIEUX

- Les « tits » menacent-ils le système binaire ?
- Quand les cycloïdes ne tournent plus très rond

BASIC A L'ESSAI

- L'Amstrad défie la concurrence
- Hector 2HR+ peut mieux faire

LE SOMMAIRE DE NOS CINQ PREMIERS NUMEROS



M2712-6-20F

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

20 F

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 6 - JANV/FÉV 1985

SOMMAIRE

RÉDACTION-RÉALISATION

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

Ont collaboré à ce numéro : *Pierre Barnouin, Robin Bois, Jacques Boisgontier, Gérard Bossuet, Roger Brousmiche, Thierry Chamoret, Eric Chauvel, Raymonde Coudert, Jacques Deconchat, Jean Drano, Michel Dupont, Florence Gautier-Louette, Pierre Ladislas Gedo, Josette Godefroy, Pierrick Greslin, Max Hagenburger, Jean-Christophe Krust, Jacques Labidurie, Jean-Pierre Lalevée, Pierre Langlois, Thierry Lévy-Abégnoli, Alain Mariatte, Pierrick Moigneau, Claude Nowakowski, Edouard Rencker, Denis Sebbag, Frédéric Vadez, André Warusfel.*

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

ÉDITION-PUBLICITÉ-PROMOTION

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

VENTES

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

Directeur de la publication : Jean-Luc Verhoye

LIST est une publication du groupe tests

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

Erratum : Nous prions les lecteurs de nous excuser pour l'erreur qui a été faite dans le numéro de novembre 1984 sur le prix du lecteur de disquette 1541 Commodore.

A VOS CLAVIERS

Écrivez à LIST
5 place du Colonel Fabien
75491 Paris Cedex 10

Toute l'équipe de LIST vous souhaite une bonne année 1985

Un seul else vous manque, et...

APROPOS du programme « Des lutins envahissent l'écran » (*LIST* 5, page 45), vous écriviez — je vous cite — que *la ligne 35 mérite d'être soulignée*. Effectivement, elle le mérite, mais pas pour les raisons que vous avancez. En fait, elle comporte une erreur !

Vous avez écrit : 35 IF X = −1 THEN X = 3 : IF X = 4 THEN X = 0. Si X vaut 4, le deuxième test ne sera pas pris en compte, car avec le premier test (IF X = −1), le programme aura déjà sauté à la ligne suivante. Il faut donc écrire :

```
35 IF X = −1 THEN X = 3
ELSE IF X = 4 THEN X = 0
```

A part cette erreur (déjà pardonnée), le programme est distrayant.

Sylvain CLÉMENT
45 Olivet

Vous êtes nombreux à nous avoir signalé l'erreur et fourni la version corrigée de cette maintenant fameuse ligne 35. Dans le texte de l'article, nous avions pourtant insisté à son propos en expliquant comment on pouvait l'écrire sur deux lignes :

```
35 IF X = − 1 THEN X = 3
36 IF X = 4 THEN X = 0
```

C'est en « recollant » les deux lignes que nous avons escamoté le ELSE. Quelle distraction !

INDEX DES ANNONCEURS LIST N° 6

Intéressant - mais

PERMETTEZ-moi de vous adresser la réflexion suivante : votre journal bénéficie du privilège d'être l'un des seuls à parler de programmation. Mais si toutefois vous vous écartez de ce sujet premier ou si vous n'allez pas au fond de celui-ci, il ressemblera à beaucoup d'autres.

Et c'est hélas ce que je constate parfois. Il est, par exemple, totalement hors sujet de publier des bancs d'essai de matériels informatiques (Atari, Alice, etc.), de parler de programmes commercialisés ou encore de compilateurs, et de publier des programmes sans donner d'explications. (Fenêtre pour ZX 81 : on aurait préféré avoir la liste du programme langage-machine désassemblé. Le programme marche très bien, mais comment ?)

C'est pourquoi je propose l'abandon des essais de matériels, des essais de programmes du commerce et l'étude en profondeur de chaque programme publié.

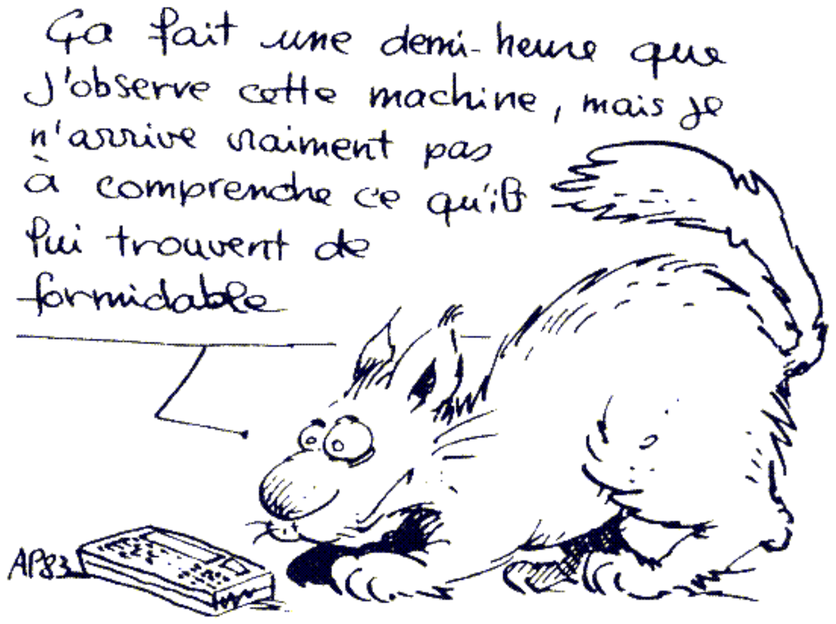
A part cela, je trouve les articles intéressants. Continuez dans cette optique.

Un programmeur fou

Il faut bien dire que quelque chose nous échappe dans votre raisonnement. En ce qui nous concerne, et nous pensons qu'il en est de même pour la majorité de nos lecteurs, nous nous intéressons à tout ce qui touche à la programmation. Bien sûr, il ne s'agit pas des gros systèmes de l'informatique lourde, mais des machines que les particuliers peuvent avoir chez eux. De ce point de vue, il nous paraît intéressant de tester le Basic de certains ordinateurs (et non le matériel lui-même, le Basic étant un langage de programmation), ou d'essayer tel ou tel logiciel qui représente une aide à la programmation.

Ce n'est pas parce que l'on programme soi-même que l'on écrit son propre interpréteur Basic ou Pascal, ou que l'on met au point, pendant ses heures de loisir, un compilateur de son invention. En ce sens, certains logiciels du commerce s'avèrent bien utiles pour le programmeur.

Quant à l'étude en profondeur de chaque programme publié, nous essayons de faire pour le mieux. Et ce n'est pas toujours facile : il y a beaucoup de matériels différents sur le marché, autant de syntaxes, des langages qui méritent une attention particulière, etc. Il est toujours difficile de satisfaire tout le monde.



A VOS CLAVIERS

De meilleures solutions pour les jeux

DANS chaque numéro de *LIST*, vous trouvez trois jeux et casse-tête informatiques dont une solution est publié dans le numéro suivant. Ce n'est pas toujours *la* solution. Vous êtes nombreux à en proposer d'autres, souvent meilleures. Nous vous soumettons ici celles qui concernent les jeux parus dans *LIST* 1.

Remontons un peu dans le temps... proposait, sans utiliser de test de comparaison, de calculer la valeur du mois et de l'année précédent celle de deux variables MOIS et ANNEE. La réponse proposait deux formules, selon que le mois ou l'année était calculé en premier. La première était celle-ci :

```
Mois   = ((Mois + 10) mod 12) + 1
Année  = Année - (Mois div 12)
```

Et la seconde, qui calculait l'année en premier :

```
Année  = Année + ((Mois + 10) div 12) - 1
Mois   = ((Mois + 10) mod 12) + 1
```

Jean-Claude Besse (de Paris) a trouvé une autre formule qui est particulièrement originale. En effet, l'expression 2 div (M+1) prend les valeurs 1 pour M = 1, et 0 pour M >= 2.

Il en tire donc les formules :

```
Année  = Année - (2 div (Mois + 1))
Mois   = Mois - 1 + (12 * (2 div (Mois + 1))
```

L'objet du jeu 2, *Pas un bit de plus*, était de trouver le nombre B de bits nécessaires pour mémoriser N valeurs différentes. Nous indiquions que, compte tenu des problèmes d'arrondi, la formule suivante pouvait être utilisée :

```
B = ENT ((log N/log 2) + 0,99999)
```

Vous avez été nombreux à réagir : cette formule s'avère inexacte à partir d'une certaine valeur de N, différente selon les ordinateurs.

En particulier, Denis Roegel (de Illkirch-Graffenstaden) s'est penché sur ce problème. Il nous signale qu'en prenant des valeurs telles que N = 524289, ou N = 1048583, on constate que le résultat fourni est erroné, ce qui montre que la formule proposée n'a pas la généralité nécessaire.

Le nombres récalcitrants sont les entiers X tels que $2^M < X < 2^{M+(1-P)}$ où M est un entier quelconque, et où P = 0.99999.

La plus petite valeur de M est donnée par :

```
M = - INT (LOG (2^(1-P) - 1) / LOG(2)
```

Ainsi, si P est égal à 0.99999, on obtient M = 18. Augmenter le nombre de décimales de P, en prenant par exemple P=0.999999 n'élimine pas le problème, mais le fait reculer.

Un autre lecteur nous indique encore deux solutions plus élégantes (et plus exactes) pour calculer le nombre de bits nécessaires à la mémorisation d'une variable. La première a l'avantage de pouvoir être écrite dans n'importe quel langage. Elle utilise les trois instructions suivantes :

```
A = LOG(N) / LOG(2)
B = INT(A)
IF A > B THEN B=B+1
```

Une autre formule, moins universelle, utilise la fonction « algébrique » INT. Elle est telle que INT (-3, 1) = -4. Si l'on dispose de celle-ci, notre calcul se réduit à une seule ligne :

```
B = - INT (-LOG(N) / LOG(2))
```

Jean-Claude Besse nous précise que le problème est tout simplement d'arrondir par excès la valeur de B = LOG(N)/LOG(2). Il suggère donc, à condition que l'on travaille en Basic ou dans un langage dérivé, d'utiliser dans notre formule l'expression (INT(C) < > 0) qui prend la valeur 0 si le reste est nul, et 1 si le reste est non nul.

Vous vous souvenez peut-être du jeu 3, *I et J mais pas K*, un casse-tête qui proposait de trouver un algorithme permettant d'échanger le contenu de deux variables I et J sans passer par une mémoire auxiliaire K.

Jean-Paul Carré (de Vaux le Penil) nous propose une démonstration rigoureuse de la validité de la formule. En notant X0 et Y0 les variables initiales, nous avons :

```
X = X + Y = X0 + Y0
Y = X - Y = X0 + Y0 - Y0 = X0
X = X - Y = X0 + Y0 - X0 = Y0
```

Ce qui montre bien que l'algorithme proposé est exact dans sa formulation.

Jean-Claude Besse a, lui, trouvé une autre solution qui est excellente par sa simplicité et sa pertinence. Il remarque en effet que les opérations de multiplication et de division peuvent être utilisées respectivement à la place des opérations d'addition et de soustraction. Nous pouvons donc écrire :

```
X = X * Y
Y = X / Y
X = X / Y
```

Cette nouvelle formule peut-être vérifiée en utilisant la méthode proposée par Jean-Paul Carré. En notant X0 et Y0 les variables initiales, nous avons :

```
X = X * Y = X0 * Y0
Y = X / Y = X0 * Y0 / Y0 = X0
X = X / Y = X0 * Y0 / Y0 = Y0
```

Ce qui montre bien la validité de l'algorithme. Bien entendu, de la même façon qu'en utilisant l'addition et la soustraction, des problèmes d'arrondi, de sur-débordement ou de sous-débordement peuvent se poser. Jean-Paul Carré nous indique que pour cette raison, K (la troisième variable, dit « de passage ») n'est pas morte. Cela est certain en pratique. Mais il faut bien remarquer que les algorithmes restent justes. C'est la différence entre ce que nous pourrions appeler l'informatique théorique et l'informatique appliquée.

Alors que Jean-Claude Besse échange les opérations additifs par des opérations multiplicatifs pour trouver une autre méthode d'échange de I et J, Pierre Barnouin (de Cabris) nous propose l'algorithme suivant :

```
X = X xor Y
Y = X xor Y
X = X xor Y
```

où XOR est le OU EXCLUSIF, qui retourne un bit à 1 lorsque les deux bits correspondants de ses opérandes sont différents, et 0 lorsqu'ils ont tous deux la même valeur.

Si l'on ne dispose pas de cet opérateur, il est possible d'écrire l'algorithme ainsi :

```
X = (X or Y) and not (X and Y)
Y = (X or Y) and not (X and Y)
X = (X or Y) and not (X and Y)
```

Bien entendu, cette méthode permute correctement les valeurs de X et de Y, sans risque de débordement, puisque les opérations sont effectuées bit à bit. Toutefois, pour les langages « algolistes » (Algol, Simula, Pascal, Modula,...) qui disposent de véritables variables logiques, l'opérateur booléen XOR ne fonctionne pas avec des entiers. D'ailleurs, cet opérateur n'opère dans ces cas là que sur un seul bit, ce qui rend la méthode impossible à utiliser. C'est pour cela que nous avions réservé cet algorithme pour un autre casse-tête, qui demanderait d'effectuer l'échange de deux variables logiques sans variable auxiliaire.

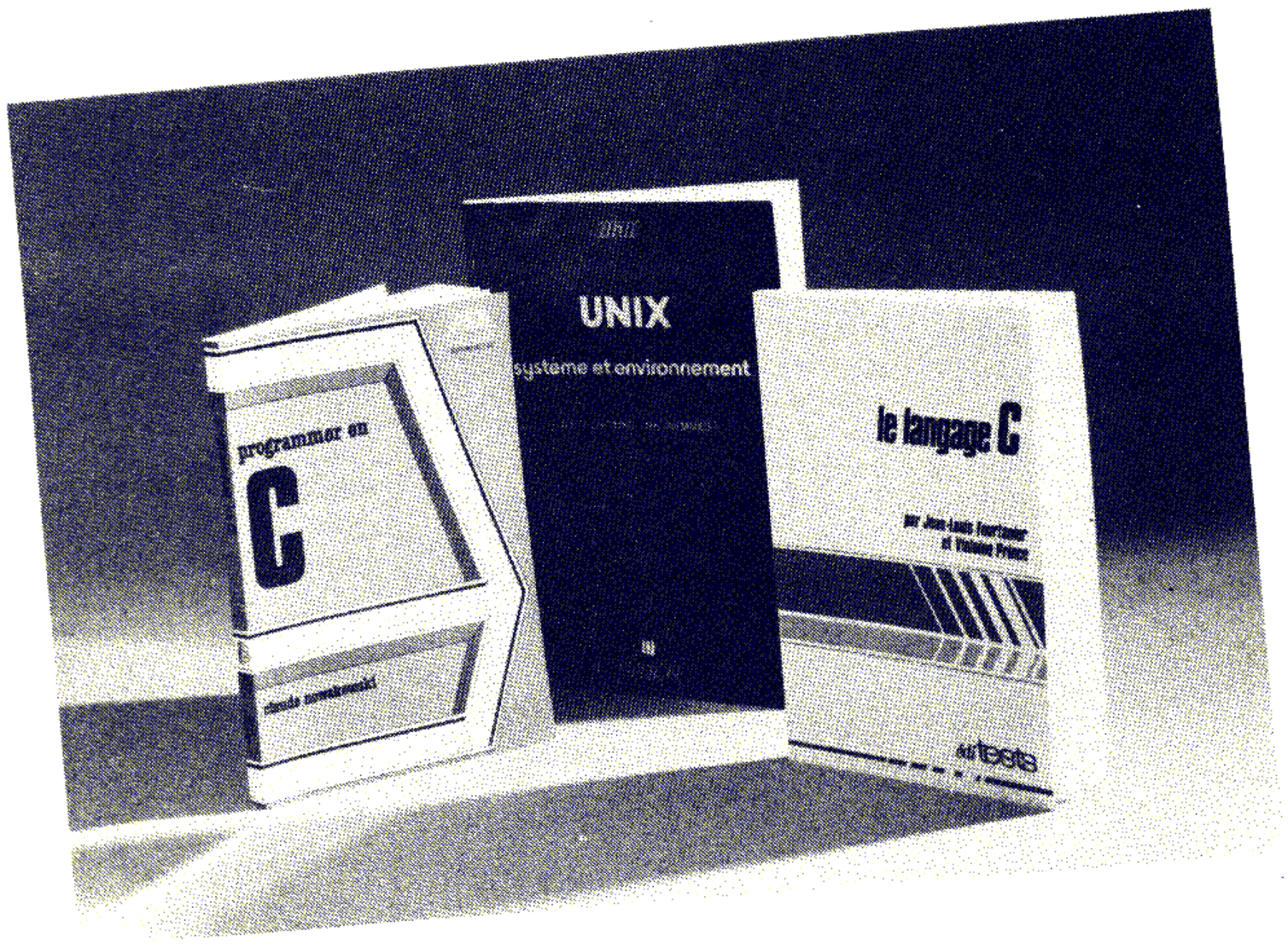
Comme Pierre Barnouin, Jean-Paul Carré nous fait remarquer que cette méthode d'échange peut se généraliser. Il nous cite la technique d'échange de deux variables du type chaîne de caractères :

```
X$ = X$ + Y$
Y$ = LEFT$ (X$, LEN(X$) - LEN(Y$))
X$ = RIGHT$ (X$, LEN(X$) - LEN(Y$))
```

Ainsi, sauf surprise, nous avons passé en revue l'échange de tous les types courants de variables.

LA GAZETTE DE LIST

TROIS LIVRES



Unix : système et environnement
Alain B. Fontaine
et Philippe Hammes
Editions Masson
Paris, 1984
Broché, 228 pages
Prix : 130 FF

Programmer en C
Claude Nowakowski
Editions du PSI
Lagny, 1984
Broché, 135 pages
Prix : 80 FF

Le langage C
Jean-Louis Fourtanier
et Violaine Prince
Editests
Paris, 1984
Broché, 109 pages
Prix : 90 FF

QUELLE n'aurait pas été la tête de Dennis M. Ritchie, l'un des pères d'Unix et de C, auteur avec Brian W. Kernighan d'un livre fondamental sur C (Masson trad. 1984), si on lui avait dit à la fin des années 60 que son langage et le système qui l'entoure feraient parler d'eux dans des revues grand public.

Pour les lecteurs de *LIST,* le premier titre n'est pas exactement du genre de leur bibliothèque coutumière, mais il devrait en passionner beaucoup. Les auteurs proposent d'abord un historique fourni (cas assez rare pour être signalé favorablement), mais surtout une vue d'ensemble d'Unix, l'un des deux ou trois systèmes de l'avenir — quand ce ne serait qu'à cause de la puissance d'AT&T maintenant très présente dans la petite informatique professionnelle après s'être longtemps contenté d'équiper les logiciels des PDP et autres VAX.

On comprend très bien (en partie grâce à la préoccupation des auteurs de remettre les faits dans le fil de l'histoire) pourquoi et comment Unix, certes aujourd'hui surtout système d'exploitation, était au départ conçu pour que les informaticiens puissent « produire, tester, documenter des logiciels » et reste donc surtout un « environnement de programmation », adapté au génie logiciel, présent à chacun des stades de l'évolution. Bien entendu ils présentent (brièvement il est vrai) C lui-même, mais aussi les rudiments d'un assembleur, d'un traitement de textes, etc.

Le deuxième livre, « Programmer en C », de Claude Nowakowski, est une introduction pédagogique et, comme le signale l'auteur lui-même, « à géométrie variable ». La règle du jeu choisie a été de ne pas transposer le Ritchie, mais de permettre — à partir du deuxième chapitre — une introduction progressive par étude de cas, c'est-à-dire, ici, par l'examen de programmes variés, du traitement de chaînes de caractères à quelques applications mathématiques simples. C'est très efficace. Une première annexe liste, de manière claire, les principales règles de syntaxe. La suivante, certainement fort utile, s'intitule « Bibliothèques » : ce sont des routines élémentaires d'une part pour le traitement de caractères, d'autre part pour le calcul scientifique — par exemple pour l'obtention de racines carrées, ou de puissances à l'aide de procédures itératives (le langage C, sorte de macro-assembleur, n'ayant naturellement pas comme le Basic des fonctions de ce type préprogrammées). Signalons aussi une revue des principaux

LA GAZETTE DE LIST

compilateurs disponibles (en mai 84 ; actuellement C arrive en masse sur les micros, au moins sur les 16 bits ; la liste s'allonge donc).

Troisième livre sur ce langage, chez un éditeur proche du précédent : un doublon ? Pas du tout ; bien entendu on y trouvera aussi de nombreux exemples, des routines de base... On pourrait bien situer le Fourtanier et Prince en disant qu'il a donné la priorité à une description, claire et précise, des principaux points de la syntaxe, mais c'est aussi en même temps un véritable guide d'aide à la programmation, avec des conseils explicitant telle ou telle possibilité, souvent des énoncés d'exercices et de nombreuses mises en garde, voire des exemples classiques d'erreurs explicitement présentes dans des programmes-pièges. C'est un ouvrage fait pour la consultation fréquente et facile, un de ceux que l'on glisse au-dessus de la pile immédiatement à droite du clavier !

Compte-rendu du fait que C n'est pas un langage de débutant et demande un certain effort intellectuel, on sent le besoin de plusieurs aides pour en obtenir une bonne maîtrise. Avec, encore une fois, le Kernighan et Ritchie comme bible « de base », il semble que ces trois ouvrages, chacun dans son style, fournissent de solides points d'appui. Munis de ces viatiques, le lecteur peut partir à la découverte de l'un des plus excitants domaines de la programmation moderne, à la fois intelligent et proche de l'informatique des informaticiens.

AW ■

Carte Minitel pour Canon X 07

INUTILE de s'énerver et de courir désespérément entre Minitel et ordinateur. Ne gaspillez plus votre énergie, vous pouvez désormais raccorder votre Canon X 07 au système Minitel ! Pour 300 FF ht.

L'*Electronic Cable Interface,* produit par la société **ECI,** vous permet de visualiser vos informations sur Minitel, de mémoriser des pages vidéotex, de mettre en œuvre le protocole vidéotex et de mémoriser les procédures d'accès aux banques de données.

Pour tous renseignements : (91) 78 92 75 ■

Porte-Parole : l'ordinateur bavard (extension pour Apple 64 K et 2e)

C'EST une carte, c'est un logiciel. C'est la parole donnée à l'ordinateur. Si vous aimez le vôtre plus que tout au monde, voici la seule chose qui lui manquait. Avantage non négligeable, l'Apple muni de la carte Porte-parole ne cause que si on lui demande. Et qui plus est, il ne raconte que ce que l'on souhaite entendre ! Que demander de mieux, hum ?

Que fait Porte-parole ? Sans attendre, commençons avec un exemple... parlant, bien entendu ! Une première application consiste à sonoriser le clavier (pas inutile pour en apprendre la disposition des touches, les non-voyants en particulier apprécieront). Un programme qui remplit cette fonction est d'ailleurs fourni avec la carte. Plus généralement, n'importe quel message, tapé au clavier en français, est traduit en langage phonétique. C'est en partant de cette représentation d'une phrase qu'est réalisée la synthèse du son.

La carte se connecte à l'ordinateur par l'intermédiaire du port n° 4 (et tant pis pour la carte Z-80 !), puis au petit haut-parleur grâce à un fil d'une cinquantaine de centimètres. Notons au passage que la documentation précise bien que cette dernière connexion s'effectue sur la prise située à droite, sous le clavier. Raté, sur l'Apple 2e cette prise est à gauche ! On peut aussi dévier le son vers un haut-parleur externe à l'aide d'une grosse prise mâle (genre HIFI ou antenne de TV). De nombreux supports de circuits intégrés sur la carte sont... complètement libres ; on peut ainsi ajouter à la carte des puces de mémoire morte ou vive dans lesquelles on stockera des données à synthétiser. Evidemment, cela vaut surtout pour des applications professionnelles.

Le premier programme de la disquette réalise la synthèse d'un texte court tapé au clavier, soit en clair, soit directement en écriture phonétique. Ceci fait, l'ordinateur prononcera le texte à sa façon (et pas toujours tel que nous le dirions nous-mêmes). Il est possible de modifier la transcription phonétique effectuée par l'Apple, afin d'affiner la prononciation. On dispose d'une série d'outils pour infléchir la voix, la rendre interrogative ou péremptoire, etc.

On devra aussi travailler des phrases difficiles comme « les poules du couvent couvent » car, d'évidence, l'un des deux derniers mots sera mal prononcé (ici le second). L'orthographe désopilante du français jouent des tours au programme : la finale *-ent* (troisième personne du pluriel) est prononcée "ant".

La sauvegarde des textes s'effectue sur disquettes, largement, car elle est loin d'être économique ! On se constituera des bibliothèques de mots ou groupes de mots (un fichier par texte).

Le second programme est un utilitaire permettant la réunion de plusieurs bibliothèques différentes en un catalogue : ces phrases pourront ensuite être prononcés à volonté à l'appel d'un quelconque programme en Basic ou en Edi-Logo (même éditeur que la carte Porte-parole).

Le fin du fin est la mise au point graphique des messages parlés. L'utilitaire fourni assure la représentation graphique des sons paramétrés : amplitude, fréquence, etc. que l'on fera varier un à un jusqu'à obtenir le son voulu. C'est du grand art que d'utiliser à fond cet ultime outil de mise au point.

Il cause bien, mais joue de la musique sans doute aussi bien. Moins compliquée que la parole, la note de musique, du moins si l'on se contente d'un timbre artificiel, est aisée à générer. Porte-parole ne s'en prive pas. C'est avec l'utilitaire graphique qu'on fixera les paramètres de la production des notes. Inutile de songer à écrire ainsi toute une symphonie, mais quelques notes de musique sont parfois bienvenues dans un programme.

Bien sûr, Porte-parole sait aussi faire du bruit. D'horribles bruits même, on se les concoctera à volonté avec l'utilitaire de mise au point. Six exemples sont donnés sur la face 2 de la disquette.

Edités par Ediciel (Matra-Hachette) la carte et le logiciel Porte-parole sont diffusés pour un prix de 1 995 FF ttc. Le logiciel seul coûte 600 FF ; il est destiné aux utilisateurs qui sont déjà en possession de la carte électronique. Cette dernière est en effet vendue avec Edi-Logo.

JCK ■

Olympiades de la micro-informatique

PILOTER un simulateur d'Airbus, dessiner sur un micro-ordinateur... ce sont deux exemples des épreuves proposées aux candidats des premières olympiades de la micro-informatique organisées par la ville de Toulouse du 3 décembre 1984 au 10 janvier prochain. Manifestation d'un nouveau genre, les concurrents (ils doivent être domiciliés dans l'agglomération toulousaine) seront répartis en quatre classes d'âge (moins de 12 ans, 12 à 16 ans, 16 à 20 ans, et plus de 20 ans) et pourront disputer une ou plusieurs des cinq épreuves proposées. Au programme :

- Un concours de création artistique sur micro-ordinateur Thomson grâce au logiciel Pictor.
- Une épreuve de simulation d'atterrissage aux commandes d'un Airbus.
- Un concours de programmation sur micro-ordinateur Thomson avec pour thème imposé un sport disputé lors des jeux Olympiques, d'été ou d'hiver.
- Chasse aux bogues. Il s'agit de relever le maximum d'erreurs glissées dans une liste de programme. Le premier qui réussit à faire fonctionner le programme a gagné.

Enfin, dernier casse-tête olympique : raccourcir le nombre de caractères d'un programme Basic sans, bien entendu, en modifier la substantifique moelle.

C'est le célébrissime Michel Platini en personne, parrain de ces premières olympiades, qui remettra les récompenses aux vainqueurs le 28 janvier 1985. ■

Une imprimante française pour 3 000 F

UNE imprimante "professionnelle", bi-mode (graphique-texte-vidéotex) bi-directionnelle, compacte, silencieuse, le tout pour 3 000 FF : une blague ?

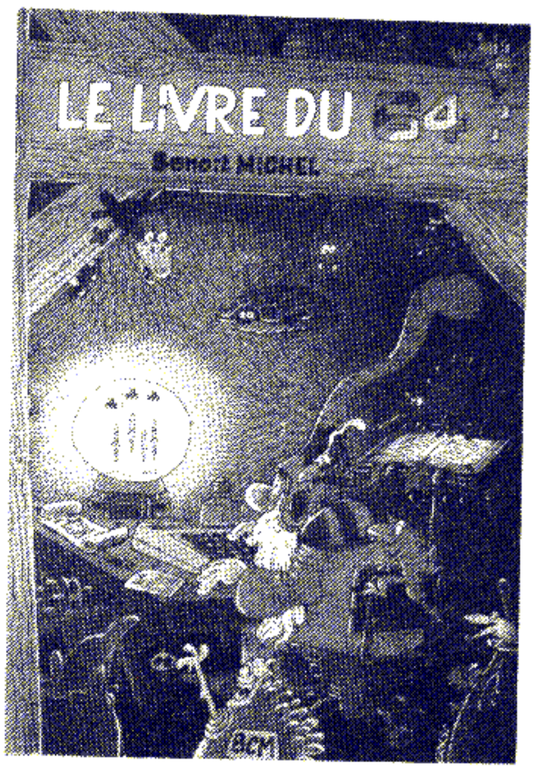
Si on ajoute qu'elle est entièrement française (de conception et de fabrication), personne, bien sûr, n'y croira. Et pourtant. Une fois n'est pas coutume, cette imprimante existe.

Elle est baptisée EXL et sort tout droit de l'association de deux sociétés françaises, Euroterminal et la CGCT. Elle « court » à raison de 100 caractères par seconde, se branche aussi bien sur un ordinateur que sur Minitel et se révèle à l'usage relativement silencieuse.

L'EXL n'est pas thermique (c'est une matricielle à aiguilles) et elle accepte 80/132 colonnes. Elle dispose aussi d'une interface parallèle, d'une entrée/sortie asynchrone RS 232 avec boucle de courant et gestion des protocoles X-On, X-Off et DTR. On annonce par ailleurs pour bientôt une version couleur. ■

UN LIVRE

Le livre du 64
Benoît Michel
Editions BCM
(PSI diffusion)
Banneux, 1984
Broché, 290 pages
Prix : 120 FF



TRÈS bon, excellent même. Peu de livres méritent ces qualificatifs. *Le livre du 64*, écrit par Benoît Michel avec l'aide d'un club informatique belge, est de ceux-là... Il se révèle être une vaste mine de renseignements et d'idées sur le Commodore 64 et sa programmation. L'auteur, une fois encore, a frappé fort. Il n'est pas un inconnu pour les « vieux » commodoristes qui ont su tirer profit de ses précédents ouvrages, parus sous le titre *Programmes internes du PET/CBM*, chez le même éditeur.

Le livre du 64 reprend, avec une présentation plus gaie, ce qui a fait le succès des précédents : il contient une étude fouillée de tout ce qui fait la puissance du C.64.

Après une description générale de l'architecture du système 64, le deuxième chapitre est consacré aux programmes internes. On n'y trouve pas le désassemblage systématique de la mémoire, mais une liste des points d'entrée des diverses routines utilisables, et la manière de les mettre en œuvre.

Les chapitres 3 et 4 sont les plus fournis : ils décrivent avec clarté et précision les différents modes graphiques et la gestion des lutins. Les amateurs de haute-résolution et de graphisme en général y trouveront pratiquement tout ce qu'il faut connaître sur le sujet. On aborde ensuite les sons, puis les entrées/sorties. Enfin, le dernier chapitre, contient une carte-mémoire simplifiée du C.64.

En annexe, sur une centaine de pages, sont fournis de nombreux programmes de démonstration de qualité assez inégale mais dont certains sont remarquables, soit par leur puissance, soit par leur concision.

Voici donc un livre très documenté, technique et clair. Les débutants y trouveront leur compte, mais le livre s'adresse plutôt aux programmeurs aguerris, qui veulent tirer plus de leur machine. Une petite connaissance (même très sommaire) du langage-machine est utile pour profiter de tout.

JPL ■

LA GAZETTE DE LIST

CASSETTES

Pour X-07

Edité par Logi'stick
Distribué par DDI
Compatible Péritel

Texte
Prix : 150 FF

Graphe
Prix : 150 FF

Fonctions + matrices (maths)
Prix : 150 FF

Pour Oric 1 et Atmos

Edité et distribué par Loriciels

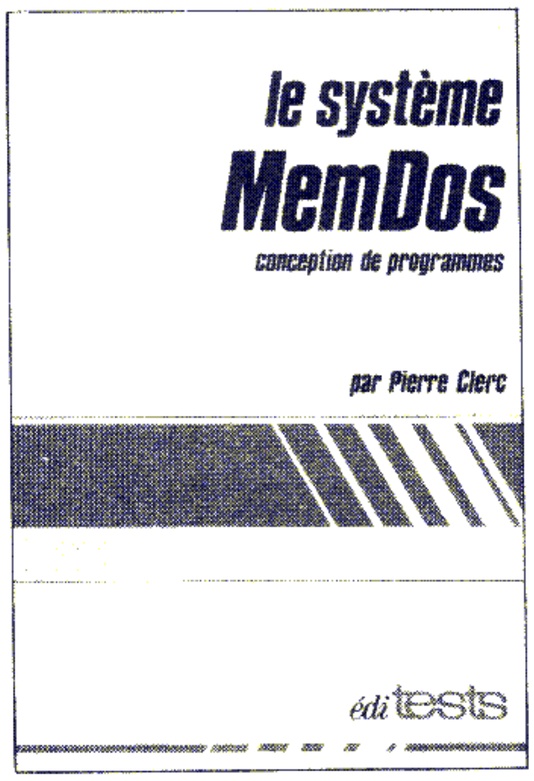
Assembleur symbolique
Prix : 260 FF

Gengraph
(générateur de caractères et de graphismes)
Prix : 140 FF

Lorigraph
(création graphique)
Prix : 290 FF

UN LIVRE

Le système MEM/DOS
Pierre Clerc
Editests
Paris, 1984
Broché, 102 pages
Prix : 90 FF



ENFIN ! vont dire bien des utilisateurs de MEM/DOS. Tous les possesseurs d'Apple ayant essayé de programmer des fichiers sous DOS 3.3. connaissent, au moins de nom, le MEM/DOS.

Il s'agit d'un outil d'aide à la conception de programmes, qui se présente sous la forme d'une carte enfichable dans l'un des ports d'extension de l'Apple II (disponible également sur CBM 8096, Sil'z et sans doute prochainement sur IBM PC). C'est un système d'exploitation qui, rajoutant des ordres au Basic, permet de gérer des fichiers séquentiels indexés multiclés et des masques d'écran (saisie, restitution, impression) avec une facilité que « l'ancêtre » DOS 3.3 ne laissait pas supposer.

Sa qualité fait que l'on ne rencontre pratiquement pas d'utilisateur qui n'en soit pas enthousiaste. Sauf... un petit détail : le manuel qui accompagnait la carte jusqu'à dernièrement était particulièrement « imbuvable ». Pour l'initiation, on n'ose même pas en parler. Vous dire l'impatience des utilisateurs. Le livre de Pierre Clerc tombe donc très bien et va sûrement contribuer au développement de ce système.

Une présentation du produit, de ses concepteurs Lafitte et Nesnidal, de la société Memsoft et une courte prospective nous prouvent qu'informatique ne rime pas toujours avec Amérique.

Ensuite, dans un style concis mais clair, l'auteur nous décrit l'organisation générale de ce système d'exploitation. Cela permettra à l'utilisateur de comprendre ce que MEM/DOS sait et ne sait pas faire. Nous avons spécialement apprécié l'emploi du mot exact et la précision des informations non diluées dans un verbiage pseudo-technique.

Le chapitre trois présente la syntaxe des ordres MEM/DOS sous forme de 39 fiches (une par ordre) claires et complètes (effet, syntaxe, particularités, exemples d'utilisation).

Enfin, un exemple concret et commenté nous fait découvrir, à travers un programme classique de gestion d'immobilisations, les fonctionnalités de MEM/DOS.

Nous n'avons trouvé dans cet ouvrage qu'un oubli (la description des ordres LOAD'' et SAVE''), mais nous regrettons beaucoup l'absence d'un index alphabétique. On se demande vraiment pourquoi tant d'auteurs de nos jours négligent ces index qui rendent la consultation d'un livre tellement plus commode. Cela dit, cette lacune est peu de chose comparée à l'aide que ce livre apportera aux programmeurs et aux futurs programmeurs MEM/DOS.

JL ■

Faites-vous connaître...

Si vous faites partie d'un club d'informatique, sans but lucratif, où l'on apprend et pratique la programmation et si vous recherchez de nouveaux adhérents, signalez-nous votre existence. En lisant votre adresse dans ces colonnes, beaucoup de lecteurs seront contents d'apprendre que votre club n'est pas trop loin de chez eux. ■

Initiation à l'informatique au prochain Sicob

FIDÈLE au rendez-vous, le prochain Spécial Sicob se tiendra du 6 au 11 mai 1985 au CNIT (Paris-la Défense).

Un succès en prévision puisqu'à cinq mois de l'ouverture de ce deuxième rendez-vous de printemps, 413 sociétés ont déjà confirmé leur participation. (En un an le Spécial Sicob a augmenté de près de 60 % sa superficie totale).

Deux journées (le 8 et le 11 mai) seront réservées au grand public. Et pour la première fois, on proposera des sessions d'initiation à la micro-informatique. En une heure trente (durée d'une session), les participants apprendront à mieux choisir un micro-ordinateur en fonction de leurs besoins. (Le coût d'une session est de 100 FF).

Quant à la boutique du Sicob, une suprise, elle déménage. Sicob Boutique quitte en effet la Défense pour s'installer au Palais des congrès (porte Maillot). Sicob Boutique ouvrira ses stands du 23 au 28 septembre prochain. De nombreux aller-retours en perspective pour les visiteurs curieux. ■

Interface parallèle graphique pour Commodore 64

LA société alsacienne **Neol** annonce une interface permettant de retirer n'importe quelle imprimante dotée d'une liaison parallèle au standard Centronics au bus série du Commodore 64.

Equipée de son propre micro-processeur, la *WW 9200/G* (nom de code de cette interface), assure la conversion de liaison (bus série - parallèle type Centronics), l'encodage des caractères ainsi que la reproduction du graphime utilisé par le Commodore 64. Elle ne nécessite aucun programme particulier et n'occupe aucun emplacement en mémoire.

Livrée avec cables et connecteur, la WW 9200/G coûte 992 FF ttc, au départ de Strasbourg bien entendu. ■

Nouveautés Apple

EN tête des hit-parades des logiciels depuis plus de six mois selon Apple, l'*Apple Works* sera disponible en français dès la fin décembre. A noter, en version tricolore, le logiciel de traitement de texte et de gestion de fichiers personnels fait peau neuve.

Innovation qui paraît aller de soi par rapport au modèle US, l'Apple Works français utilise les accents circonflexe, tréma, aigu, grave, ils sont tous là... Apple Works propose également des formats financiers et comptables libellés en francs, des dates et des formats horaires conformes, aux usages français, des virgules décimales...

Commercialisé au prix de 2 490 FF ttc, Apple Works est accompagné d'un ouvrage de référence de 330 pages et d'un manuel de travaux pratiques de 140 pages rédigé en français. Pour une fois, on ne pourra pas reprocher le manque de documentation. ■

Lecteur de disquettes sur batteries chez Hewlett-Packard.

DESTINÉS aux gros dévoreurs de mémoire, de nouveaux lecteurs de disquettes (dont un modèle fonctionnant sur batterie) ont été annoncés chez Hewlett-Packard.

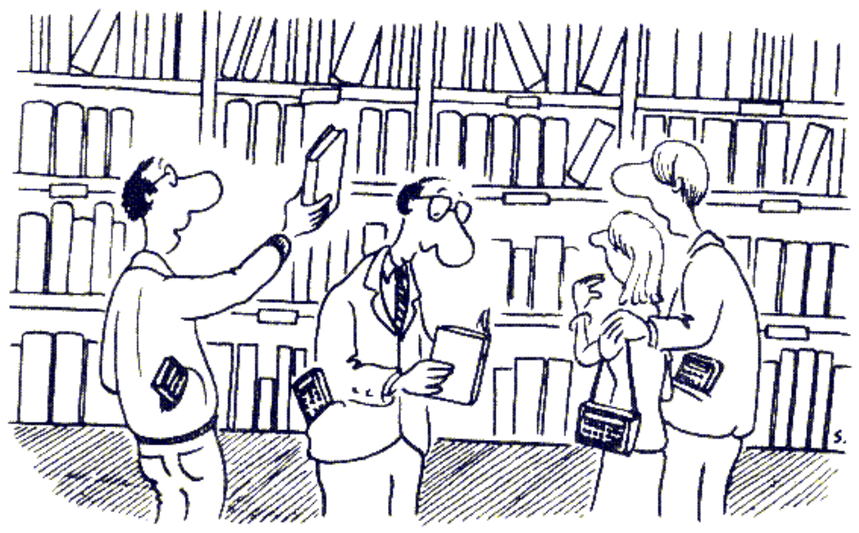
Trois configurations différentes seront prochainement disponibles. Parmi celles-ci :

• la HP 91 225, modèle simple, offre une capacité de stockage de 710 Koctets formatés, soit le triple en capacité de la version introduite en 1982 ;

• la HP 9114 A, modèle portable, fonctionne sur batterie avec disque 3 1/2 pouces.

Cette unité permettra par ailleurs, de réaliser des échanges de données entre le portable HP 110 et, bien entendu, leur modèle « de charme », le HP 150 à écran tactile. Ces nouvelles unités 3 1/2 pouces sont compatibles avec les unités simple face déjà en service. ■

UN PETIT TOUR CHEZ LE LIBRAIRE



Musique sur Commodore 64
Un guide pour programmer la musique et les sons
James Vogel et Nevin B. Scrimshaw
Editions Cédic/Nathan
Paris, 1984
Broché, 160 pages
Prix : 85 FF

Graphisme et CAO
Techniques et exemples de conception graphique
P. Darnis
J.-M. Van Thong
Editions Edimicro
Paris, 1984
Broché, 192 pages
Prix : 125 FF

Guide de l'informatique individuelle
Helen Varley et Ian Graham
Editions Hachette Informatique
Paris, 1984
Broché, 224 pages
Prix : 99 FF

Le Basic en 30 heures
Clive Prigmore
Les éditions d'organisation
Paris, 1984
Broché, 272 pages
Prix : 148 FF

L'assembleur facile du 6809
François Bernard
Editions Eyrolles
Paris, 1984
Broché, 168 pages
Prix : 89 FF

Pratique du micro-ordinateur Adam
Henri Lilen
Editions Radio
Paris, 1984
Broché, 176 pages
Prix : 100 FF

Programmer, c'est pas sorcier
Gary W. Orwig et William S. Hodges
Editions Belin
Collection Modulo
Paris, 1984
Broché, 180 pages
Prix : 140 FF

Dictionnaire de la micro-informatique
Charles J. Sippl
Traduit et adapté par Johanne de Luca
Editions Belin
Collection Modulo
Paris, 1984
Broché, 292 pages
Prix : 295 FF

Programmez vos jeux sur TI 99/4A
François Abella
Editions Sybex
Paris, 1984
Broché, 116 pages
Prix : 78 FF

Jeux graphiques sur Atmos
David Chane-Hune et François Darbois
Editions Edimicro
Paris, 1984
Broché, 186 pages
Prix : 95 FF

Choisir son micro-ordinateur
Yvon Dargery
Editions Cédic/Nathan
Collection Micromonde
Paris, 1984
Broché, 128 pages
Prix : 35 FF

Choisir votre micro-ordinateur
Ilya Virgatchik
Editions Marabout
Paris, 1984
Broché, 288 pages
Prix : 24 FF

Tours de Forth
Marc Petremann et Michel Rousseau
Editions Eyrolles
Paris, 1984
Broché, 184 pages
Prix : 98 FF

Activités avec le Commodore 64
David Lawrence
Editions Hachette Informatique
Paris, 1984
Broché, 190 pages
Prix : 95 FF

Programmeurs : à vos claviers !

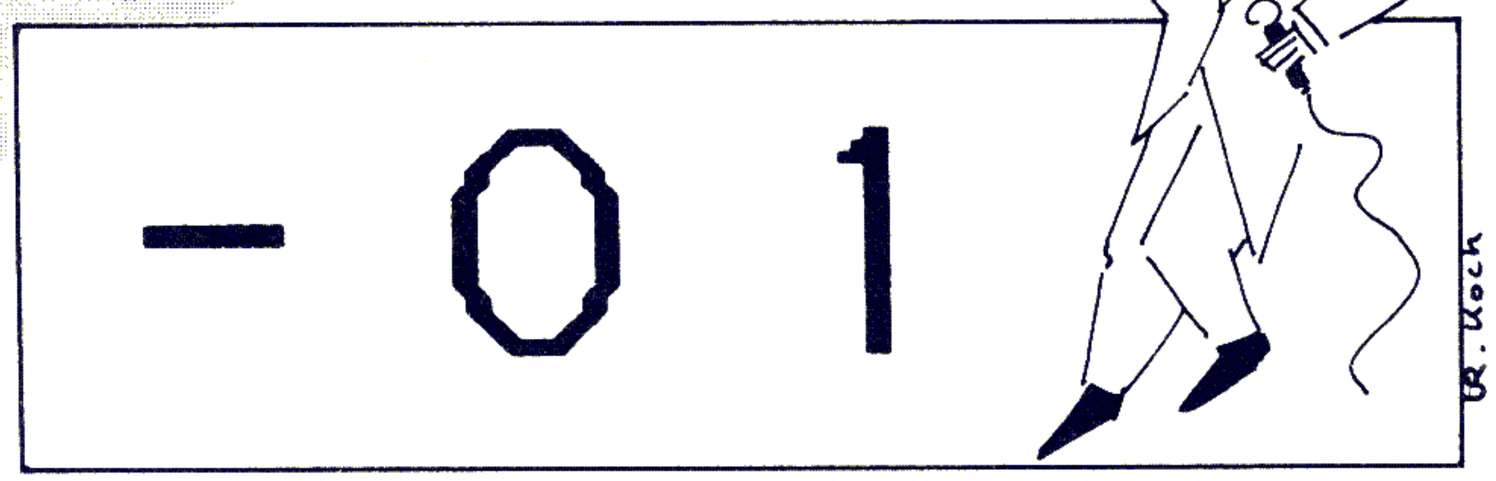
UN concours de scénarios de logiciels éducatifs ouvert aux élèves, aux enseignants et aux clubs d'informatique de l'enseignement secondaire public, c'est ce que « concocte » le ministère de l'Éducation nationale en liaison avec l'Encyclopédie Nationale des Sciences et Techniques.

Le ministère n'a pas encore rendu publiques les modalités du concours mais l'Education Nationale prend la chose au sérieux. Elle débloquera quelque 300 000 francs de crédits pour les candidats qui, de plus, pourront, s'ils le souhaitent, disposer de micro-ordinateurs fournis par l'E.N.

Les trois réalisations retenues par le jury final seront éditées. ■

LA GUERRE DES TROIS

CHACUN connaît la numération binaire et ses liens étroits avec nos ordinateurs. Le système ternaire dont nous allons évoquer l'utilisation en informatique nous est beaucoup moins familier. Il est assez séduisant pour l'esprit et c'est probablement plus qu'une curiosité mathématique.



Il y a système ternaire et système ternaire. Celui auquel nous avons songé est bien en base trois, comme son nom l'indique ; il s'écrit à l'aide de « tits » valant non pas 0, 1 et 2 contrairement à ce que l'on aurait pu attendre, mais -1,0 et $+1$, que nous transcrirons par $-$,0 et 1 (lire : *tiret*, *zéro* et *un*). C'est le choix de ces trois valeurs et la notation particulière qui en découle qui font l'originalité de ce système.

Tit s'impose par forfait

« Tit », construit sur le modèle de bit, est une contraction de « ternary digit ». Il s'est imposé par forfait, aucun monosyllabe acceptable n'ayant pu être tiré d'expressions telles que chiffre ternaire ou unité ternaire. Vive le tit, donc.

Quelques exemples sans attendre, pour mieux se fixer les idées ; les nombres de -4 à $+4$ s'écriront respectivement : $--$ (prononcer « tiret-tiret »), -0 (prononcer « tiret-zéro »), -1 (« tiret-un »), 0, 1, $1-$ (« un-tiret »), 10 (« un-zéro »), et 11 (« un-un »). Quant à 5, il s'écrira $1--$ qui se décompose en $+9-3-1$ ou, si l'on préfère, $+3^2-3^1-3^0$

Ainsi, rencontré dans des expressions comme « $-A$ » ou « $B-C$ », le signe du tiret (« – »), promu au rang de « tit » s'interprète comme $+(-1)\times$, exactement comme 2A vaut $(+2)\times A$.

Si cette convention très naturelle suffit à écarter toute ambiguïté, c'est que l'adoption de ce système ternaire rend obsolète la notion même de signe d'un nombre, et conduit à redéfinir strictement la soustraction comme une multi-

Mettons-nous au ternaire

Pour vous permettre de vous familiariser avec le système ternaire et sa notation, voici d'abord ses tables d'addition et de multiplication :

Addition

	–	0	1
–	–1	–	0
0	–	0	1
1	0	1	1–

Multiplication

	–	0	1
–	1	0	–
0	0	0	0
1	–	0	1

La multiplication par « – » change tous les 1 en – et réciproquement. Quant à la soustraction, c'est une multiplication par « – » suivie d'une addition.

En guise de répétiteur, vous trouverez ci-contre un court programme de conversion destiné à faciliter votre apprentissage. Il est limité aux entiers de -32768 à $+32767$ dans le sens décimal/ternaire, mais pratiquement illimité dans l'autre sens.

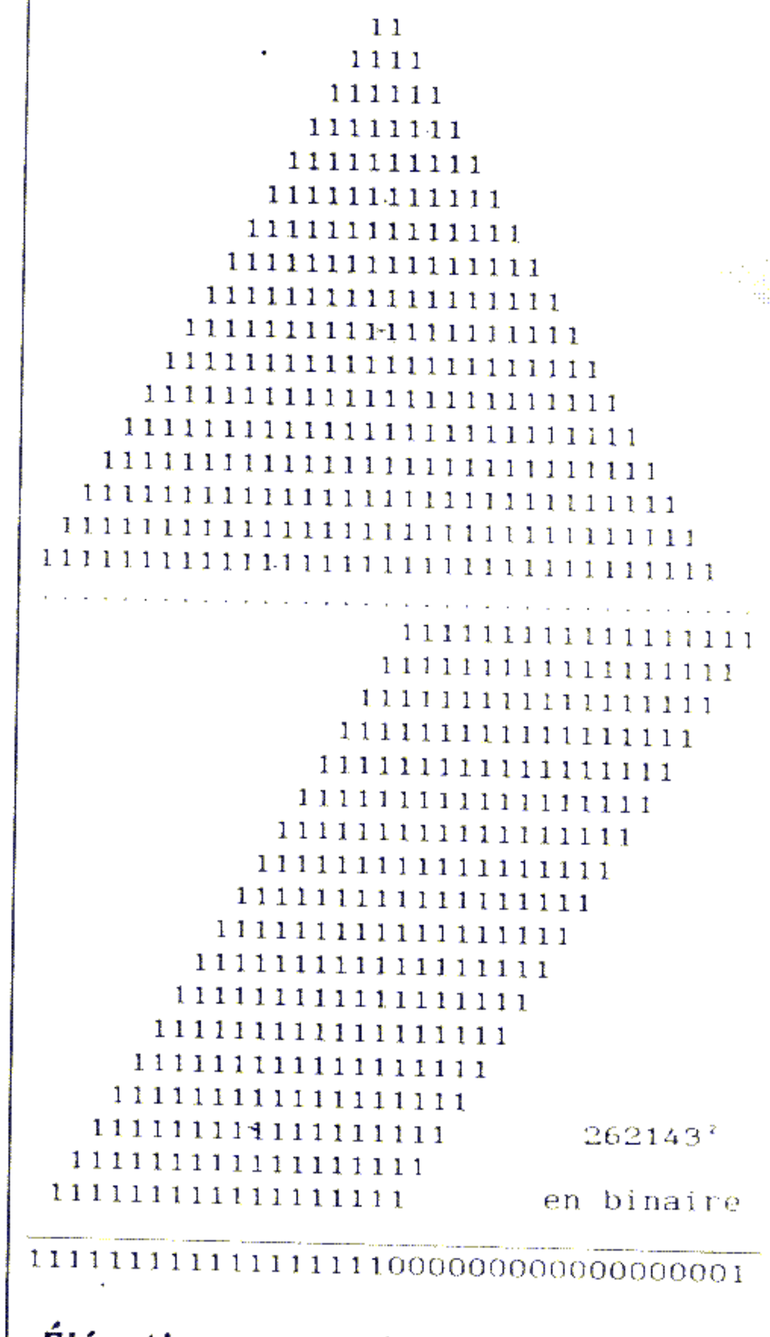
Écrit pour Canon X-07, il est assez facilement adaptable à d'autres matériels.

CONVERSION
DECIMAL / TERNAIRE

```
1 CLS:Z$(0)="0":Z$(1)="1":Z$(2)="-"
2 PRINT"DECIMAL <-> TERNAIRE(RETURN) s
i l'option convient pas"
3 INPUT"Nb Decimal";D:E=D:IFDTHEN8
4 INPUT"Nb Ternaire";T$:IFT$=""THENRUN
5 L=LEN(T$):FORI=0TOL-1:A=INSTR("-01",MI
D$(T$,L-I,1))
6 IFATHENE=E+(A-2)*3^I:NEXTELSERUN
7 INPUT"= en Decimal";D:K=(E=D):GOTO11
8 A=DMOD3:B=(A+3)MOD3:T$=Z$(B)+T$
9 D=D\3+(A=-2)-(A=2):IFDTHEN8
10 INPUT"= en Ternaire";X$:K=(X$=T$)
11 IFKTHENPRINT"EXACT"ELSEPRINT"ERREUR"
12 PRINTT$;"=",E:LINEINPUTT$:RUN
```

Et je retiens un... et je retiens un...

Les deux carrés (262143^2 et 265720^2) sont du même ordre de grandeur. Ils ont été choisis parce que leur calcul (en binaire et en ternaire) conduit à un maximum de retenues. Ces dernières forment un triangle au-dessus de la série des additions.



262143² en binaire

Élévation au carré de 262143 en binaire

265720² en ternaire

Élévation au carré de 265720 en ternaire

Conversion décimal-ternaire

Valeur décimale	Valeur ternaire	Valeur décimale	Valeur ternaire
0	0	36	1100
1	1	37	1101
2	1 -	38	111 -
3	10	39	1110
4	11	40	1111
5	1 - -	41	1 - - - -
6	1 -0	42	1 - - -0
7	1 -1	43	1 - - -1
8	10 -	44	1 - -0 -
9	100	45	1 - -00
10	101	46	1 - -01
11	11 -	47	1 - -1 -
12	110	48	1 - -10
13	111	49	1 - -11
14	1 - - -	50	1 -0 - -
15	1 - -0	51	1 -0 -0
16	1 - -1	52	1 -0 -1
17	1 -0 -	53	1 -00 -
18	1 -00	54	1 -000
19	1 -01	55	1 -001
20	1 -1 -	56	1 -01 -
21	1 -10	57	1 -010
22	1 -11	58	1 -011
23	10 - -	59	1 -1 - -
24	10 -0	60	1 -1 -0
25	10 -1	61	1 -1 -1
26	100 -	62	1 -10 -
27	1000	63	1 -100
28	1001	64	1 -101
29	101 -	65	1 -11 -
30	1010	66	1 -110
31	1011	67	1 -111
32	11 - -	68	10 - - -
33	11 -0	69	10 - -0
34	11 -1	70	10 - -1
35	110 -		

plication par « − » (= − 1) suivie d'une addition.

Avec la notion de signe sont ainsi balayées, bien entendu, les servitudes qui pesaient sur l'informatique binaire pour le stockage et la gestion du signe des nombres et exposants.

Deuxième avantage, incontestable lui aussi, il faut beaucoup moins de tits en ternaire que de bits en binaire pour stocker une même information. Il suffit, par exemple, de 16 tits pour écrire tout nombre compris entre −21523360 et +21523360 alors que l'on reste confiné entre −32768 et +32767 avec 16 bits.

Un nombre de 14 chiffres significatifs en base 10 compris entre -10^{40} et $+10^{40}$ peut s'écrire sur 34 tits, contre 56 bits (64 bits en décimal codé binaire). Avec 4 tits, d'autre part, on code 81 caractères, un cinquième tit offrant 162 caractères ou codes spéciaux supplémentaires : on obtient ainsi les 243 combinaisons d'un « pentet », qui joue le rôle de l'octet traditionnel. Et l'on peut même coder sur 3 tits seulement de l'alphabétique pur, limité aux 26 lettres et à l'espace.

Les tables d'opérations arithmétiques restent simples et l'avantage du ternaire devient écrasant dès qu'il s'agit de multiplication. Élevons au carré deux nombres tout à fait comparables et qui s'écrivent chacun uniquement avec des 1. Il faut en binaire 594 additions élémentaires comportant 306 reports de retenues pour calculer 262143^2, contre 192 additions et 72 retenues seulement pour 265720^2 en ternaire (voir l'encadré ci-dessus).

La supériorité de ce ternaire

On remarquera que, les deux fois, nous nous sommes placés dans les conditions **les plus défavorables.** Cela dit, et chacun pourra le vérifier s'il veut, avec des valeurs quelconques, les annulations mutuelles de tits de valeur opposée compensent la proportion plus faible de zéros, et le rapport des volumes d'opérations élémentaires reste du même ordre.

Au-delà des gains considérables à attendre en puissance de calcul, en capacité de stockage et en vitesse de transmission, la supériorité du ternaire s'affirmerait aussi en accélérant nettement les opérations de tri et de recherche sur arborescence. Il favoriserait également l'éclosion d'une logique à trois voies : <, = et >, ou encore VRAI, FAUX et INDÉCIS, plus souple et plus performante que la dichotomie binaire.

Que faut-il pour adapter l'ordinateur au ternaire ? Essentiellement des puces de 3, 9 et 27 tits à base de jonctions d'un type nouveau. On peut imaginer que les impulsions électriques passent dans un sens, dans l'autre, ou ne passent pas du tout. On peut imaginer aussi des microdomaines magnétiques à trois états stables : magnétisés dans un sens, dans l'autre, ou démagnétisés.

Pour l'instant, bien sûr, il n'existe pas de machines ternaires. Mais qui peut jurer de ce que réserve l'avenir ?

Pierre BARNOUIN

DEUX LANGAGES "PRO"

APL et PL/1

LES années les plus riches de l'histoire des langages se situent probablement aux alentours des années 60. Fortran, Algol, Cobol sortirent vers cette époque. Avant l'aventure du Basic, au cours des années 62-63, deux « canards sauvages » virent le jour : APL et PL/1. Ils sont encore bien vivants, surtout le premier d'entre eux qui jouit d'un grand prestige auprès des scientifiques. Le second, en qui tous les espoirs furent placés, a connu jusqu'à nos jours une carrière commerciale honnête, certainement en deçà des possibilités dont on l'avait chargé.

On ne se lève pas un beau matin en disant : tiens, si j'inventais un langage ? Le professeur canadien Kenneth Eugene Iverson n'échappa nullement à la règle. Dès 1956 — d'après une synthèse très documentée de Jean-Luc Stehlé — ce mathématicien d'Harvard, dans le cadre de recherches sur la théorie des algorithmes, cherchait à développer des notations générales et non ambiguës pour exprimer ses idées, notamment sur les problèmes de tris. Il en publia une première version en 1960 en rejoignant le centre de recherches Thomas J. Watson à Yorktown Heights, deux ans avant d'écrire son livre fondamental *A Programming Language* (chez Wiley and Sons). 1962 est donc considérée comme la date de naissance officielle d'APL, bien que la première implantation, due à Arthur Hellerman, ne date que de 1963 (sur un IBM 1620).

Il ne s'agissait donc, au départ, que d'un outil mathématique pour faciliter la communication entre hommes de sciences, très bien adapté à la description et l'analyse formelle d'ordinateurs.

Célèbre surtout dans les universités

Jean-Luc Stehlé indique la date de novembre 1966 comme étant celle d'une certaine normalisation du langage dans une forme très voisine de ses versions actuelles — qui bénéficièrent d'un enrichissement notable en 1973 avec la notion de variable partagée. Il fallut apporter des modifications assez nombreuses aux symboles d'Iverson pour les adapter aux claviers usuels des terminaux mais, pour le fond, c'est toujours l'esprit de ses notations qui subsiste. Certains auteurs regrettent qu'aucune norme ne soit venue mettre un peu d'ordre dans des implantations devenues aujourd'hui assez anarchiques.

Le fumet très universitaire qui se dégage d'APL le rendit tout de suite célèbre dans son milieu d'origine, mais maintenant il est même utilisé pour des manipulations de fichiers et de bases de données, comme d'autres langages pra-

Deux « canards sauvages » parmi les autres langages

tiqués par l'informatique professionnelle. Tous les commentateurs d'APL soulignent malignement qu'il fut longtemps l'objet de véritables guerres civiles, opposant laudateurs inconditionnels et adversaires impitoyables, mais il semble bien aujourd'hui qu'un certain consensus lui reconnaisse enfin, au-delà d'une originalité évidente, des qualités assez exceptionnelles.

S'il est vrai que les applications d'APL hors université ne sont pas aussi nombreuses que pour Cobol ou PL/1, on peut citer par exemple en France le cas de Citroën et, ce qui est sans doute plus frappant, l'utilisation qui en fut faite par les studios Walt Disney pour la réalisation d'une partie des « nouvelles images » du film *Tron*... Les implantations sur ordinateur individuel sont encore assez peu nombreuses : à côté d'une version « tiny » sur TRS-80, il faut surtout remarquer les versions adaptées aux différents PC d'IBM (naissance oblige), un interpréteur moyen exigeant souvent près d'une centaine de Koctets (ce qui le réservait plutôt jusqu'à présent aux minis). L'évolution des matériels devrait changer cet état de choses, surtout compte tenu de l'intérêt pédagogique que présente ce langage pour les scientifiques.

La position d'APL par rapport aux champions actuels de la modularité (« famille » Pascal, en particulier) est assez ambiguë. Contrairement à eux, il n'est pas compilé, mais interprété, et surtout très fortement interactif. Il n'est besoin de déclarer ni variables, ni dimensions de tableaux. Il n'a pas de mots réservés. Une caractéristique essentielle, qui explique son succès chez les mathématiciens purs et appliqués, c'est qu'on y traite des vecteurs, des nombres complexes et des matrices aussi simplement — au moins au niveau de la programmation, cela se payant par une lenteur d'exécution parfois déroutante — que des nombres ou des chaînes de caractères.

Sa principale qualité est sans aucun doute son extrême concision, source de fiabilité évidente mais hélas aussi responsable d'une relecture très pénible (cas toujours cité par les APLophiles, caractéristique de l'art avec lequel les notations d'Iverson font simple quand le fond des choses n'est pas compliqué : un programme de recherche de nombres premiers peut tenir en une seule instruction — voir par exemple la page 63 du livre de Daniel-Jean David, *Le langage APL*, édité par Editests).

Donnons deux exemples minuscules, mais significatifs :

• pour calculer une somme de nombres, une ligne du type +/ 10 11 12 13 14 15 donne automatiquement le total 75 ;

• pour garnir un tableau à une dimension X par les dix-huit premiers entiers supérieurs à 32, il suffit d'écrire X [32 + i18] qui équivaut à la boucle classique du Basic, FOR I = 0 TO 17 THEN X(I) = 33 + I : NEXT I (la lettre « i » est en fait une minuscule grecque ; APL adore cet alphabet auquel il emprunte également le rhô et le delta : une réminiscence légèrement snob et très « Ivy League »).

Langage conduisant à des listes élégantes mais bien trop souvent hermétiques, abhorrées par les censeurs vigilants à la Wirth, APL est pourtant très modulaire à sa façon, ce qui lui donne un air de modernité surprenant pour son âge. Il manipule essentiellement des fonctions (pas de GOSUB) que l'on peut mettre au point et utiliser séparément : « les fonctions primitives sont en quelque sorte les mots du langage » (Bernard Legrand, *Apprendre et appliquer le langage APL*, édité par Masson). Chaque programmeur crée à volonté ses propres fonctions, un peu comme en LISP, qui étendent considérablement la facilité d'écriture des instructions : il existe des opérateurs entre fonctions qui définissent une nouvelle fonction, comme un signe + crée un nombre x + y à partir de deux nombres x et y. Il y a très peu de boucles, dont la fausse simplicité transforme parfois en cauchemar la maintenance de 300 petites lignes de Basic...

Gagner des parenthèses

La syntaxe est réduite à un petit nombre de règles. L'évaluation des fonctions se fait de droite à gauche, ce qui surprend souvent le débutant ; ce n'est pourtant qu'une (bonne) habitude à prendre, un peu voisine des « représentations polonaises » et qui fait gagner parfois bien des parenthèses.

Mais c'est là le type d'argument qui provoque la fureur des inconditionnels anti-APL : ils prétendent non sans bon sens que ce « plus » est souvent un

DEUX LANGAGES « PRO » APL ET PL/1

« moins » aux conséquences parfois catastrophiques s'il cause une erreur pratiquement impossible à déceler. La simplicité des règles, combattue par la complexité des notations, fait que le temps d'apprentissage d'APL varie suivant les individus : chez les matheux, ça va en général très vite...

Qualités et défauts d'APL viennent, en grande partie, du fait qu'il est issu de la tête d'un seul homme. Ils sont assez difficiles à évaluer en toute objectivité ; les années à venir, qui verront probablement un développement notable de l'utilisation de ce langage, permettront de faire le point. Aux deux ouvrages signalés au hasard des lignes de cette brève présentation, ajoutons *Introduction à APL* de S. Pommier chez Dunod et, pour les Tandyistes, *L'APL sur TRS-80* de Claude Nowakowski aux éditions du PSI.

Si APL signifie « *un* langage de programmation », le sigle PL/1 renvoie plus orgueilleusement au langage de programmation « numéro 1 ». Quasiment contemporain d'APL, il résulta, comme Cobol ou Algol, de la création en septembre 1963 d'un groupe de travail, le Share Fortran Committee, né d'une initiative commune d'IBM et d'un groupe d'usagers. Il était chargé de concevoir un produit plus puissant, capable en particulier de traiter correctement des problèmes de gestion sans perdre la richesse scientifique de l'ancêtre. Dès mars 1964 naissait le NPL (New Programming Language) bientôt rebaptisé PL/1, à la demande du National Physical Laboratory britannique.

La bible de PL/1, longtemps le seul ouvrage disponible en France sur le sujet dès 1971 (*Le langage PL/1*, Marcus Dornbusch, chez Dunod) résume ainsi les buts de ses créateurs :

- être avantageusement comparable à Fortran, Algol, Cobol et Assembleur dans tout ce que ces langages ont d'efficace ;
- être modulaire, c'est-à-dire ici, utilisable par tout programmeur spécialisé dans son propre domaine de travail même s'il en ignore les autres possibilités ;
- pouvoir s'enrichir par l'addition de fonctions nouvelles ;
- contenir en lui-même des outils de mise au point.

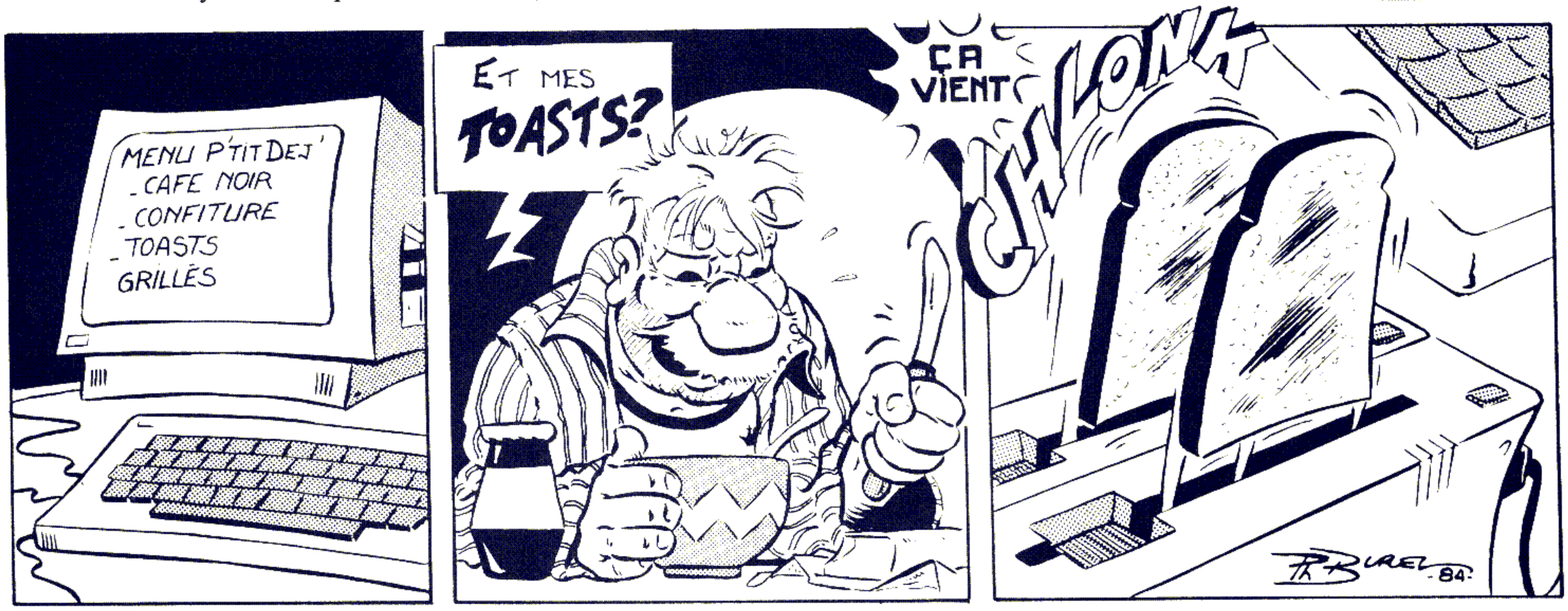
Une bonne structuration

La référence à Algol conduisit en particulier à une assez bonne structuration du langage, proche de ce qui fit le succès de Pascal. Même une gestion des incidents (ou exceptions), permettant à un programme de tourner en dépit d'un éventuel dépassement de capacité par exemple, reconnue ensuite comme fondamentale par les réalisateurs d'ADA, figure déjà dans PL/1. De Fortran, il gardait l'aptitude scientifique, les tableaux, la notion de format, les boucles, enrichissant les procédures de sous-programmes et fonctions, alors que Cobol, lui, fournissait un maniement commode des chaînes de caractères et des fichiers. Il permet, dans une large mesure, d'éviter le recours à des routines directes en machine.

Le démarrage de PL/1 fut malaisé, d'une part parce que le premier compilateur de 1966 était assez médiocre et semblait trop concis pour être vraiment efficace, d'autre part parce que l'ombre de la puissance de ses géniteurs effrayait ses utilisateurs potentiels. Quoique ses problèmes de jeunesse aient été grandement réglés depuis, la montée de systèmes plus « modernes », sa richesse et la complexité qui en résulte en ont restreint l'application. Bien qu'il ait été voulu comme *le* langage de la fameuse série 360, PL/1 n'a pas répondu entièrement à l'attente commerciale de ses concepteurs.

Aujourd'hui encore, il est pourtant toujours largement présent dans les secteurs de l'informatique professionnelle, dans des configurations assez riches pour le supporter. Sa puissance fait qu'il n'est pas étonnant qu'on n'en trouve pas de version pour micro-ordinateur. En tant que première tentative de définition d'un langage vraiment universel, son importance théorique ne peut être surestimée. D'une certaine manière, la définition d'ADA n'aurait pu être entreprise s'il n'avait pas existé ; presque tout ce qui y figure en dehors du noyau principal (évidemment issu de Pascal) sort pratiquement de PL/1 : c'est une preuve manifeste de qualité.

André WARUSFEL



L'EXEMPLE D'UNE BONNE CORRECTION

UNE partie non négligeable de la programmation consiste à corriger, mettre au point, améliorer les lignes que l'on a pondues. L'ensemble des fonctions qui permettent d'apporter les modifications désirées forment l'éditeur. Ce mois-ci, nous examinerons celui du PC-1251. Comme nous allons le voir, il est commode et très bien conçu : c'est un plaisir.

Avant de corriger des fautes dans un programme, il faut, évidemment, les y introduire. Ordinairement, cela tient de la génération spontanée : les erreurs apparaissent à l'insu du programmeur... Pour les besoins de l'exemple (une fois n'est pas coutume), nous les commettrons de façon délibérée.

Dans une liste pour PC-1251, nous écrirons donc la ligne : 250 IMPT « NOMBRE SUIVANT ? » ; S, ce qui demande 32 pressions de touche si la machine est déjà en mode PRO, mode où l'on peut introduire, lister ou corriger un programme.

Si le PC-1251 se trouve en mode RUN (exécution d'un programme ou calculs au clavier), on doit d'abord déplacer le commutateur sur la position PRO : une manipulation de plus, soit un total de 33 actions. Ce nombre tient compte des pressions sur la touche SHIFT et de l'appui final sur ENTER qui *valide* la ligne, c'est-à-dire l'inscrit en mémoire.

La ligne que nous voulons obtenir est : 250 INPUT « PREMIER NOMBRE ? » ; A. Pour entrer cette ligne, il faut, selon la façon dont on s'y prend, entre 30 et 33 pressions de touche si l'ordinateur est en mode PRO. Avec le Basic Sharp, en effet, l'ordre INPUT peut être dactylographié I., IN., INP., INPU. ou en toutes lettres, INPUT, soit 2, 3, 4 ou 5 frappes.

Notre exercice va donc consister à passer de 250 IMPT « NOMBRE SUIVANT ? » ; S à 250 INPUT « PREMIER NOMBRE ? » ; A.

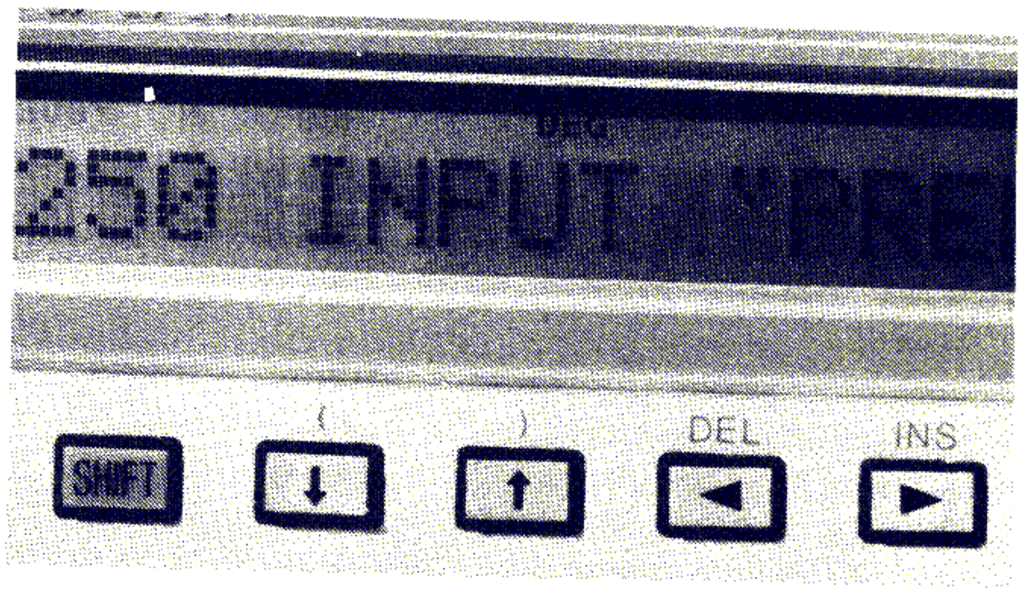
Il y a plusieurs façons de procéder. La première que nous allons décrire n'est pas, et de loin, la meilleure, mais elle a le mérite d'illustrer la plupart des fonctions d'édition disponibles sur la machine.

Un carré noir sur la faute

Nous commençons par lancer le programme (en mode RUN) : RUN ENTER. Instantanément, l'ordinateur affiche ERROR 9 IN 250. Une pression sur la touche ↑ affiche la ligne fautive, un pavé noir clignotant à l'endroit où l'erreur a été dépistée. Dans le cas présent, le pavé clignote sur le I de IMPT. Dès que l'on relâche la touche ↑, le message d'erreur réapparaît. Cette facilité, très appréciable, se retrouve sur les poquettes Sharp 1211, 1212, 1245, etc.

Pour corriger la ligne, nous devons maintenant passer en mode PRO. Une nouvelle pression sur ↑ et la ligne 250 revient (pavé clignotant au même endroit). Cette fois-ci, la ligne demeure à l'affichage.

Dans un premier temps, nous allons



Les cinq touches essentielles à la correction

L'EXEMPLE D'UNE BONNE CORRECTION

transformer IMPT en INPUT. Une pression sur la touche ▶ et le pavé se place sur M. On frappe simplement la lettre N qui remplace aussitôt le M. Dans le même temps, le pavé clignotant s'est décalé d'une position vers la droite, sur la lettre P. On frappe P et le pavé clignote alors sur la lettre suivante, T. Pour insérer à cet endroit le U manquant, trois touches : SHIFT INS (un rectangle vide apparaît entre INP et T) et U.

La ligne est devenue : 250 INPUT « NOMBRE SUIVANT ? » ; S.

Une correction plus rapide

Depuis le début de l'opération, on a pressé 7 touches. Nous allons maintenant insérer PREMIER et un espace devant NOMBRE, détruire SUIVANT et remplacer S par A à la fin de la ligne.

On déplace donc le pavé clignotant jusqu'au N de NOMBRE (2 fois ▶), on se ménage 8 espaces libres pour l'insertion (8 fois SHIFT suivi de INS) et l'on frappe PREMIER □ (le signe □ représente un espace).

Après 33 pressions de touche, la ligne est devenue : 250 INPUT « PRE MIER NOMBRE SUIVANT ? » ; S et le pavé clignote encore sur le N de NOMBRE. On le déplace jusqu'au S de SUIVANT (7 fois ▶), mot que l'on efface, ainsi que l'espace qui le suit (SHIFT DEL, répété 8 fois).

Depuis le début de la correction, on a déjà pressé 56 touches... Notre ligne est devenue : 250 INPUT « PREMIER NOMBRE ? » ; S.

Reste à remplacer S par A. Trois fois ▶, et le pavé clignote sur S. On frappe A, puis ENTER. La ligne 250 est désormais bien celle que l'on voulait : 250 INPUT « PREMIER NOM BRE ? » ; A.

Il nous aura fallu presser plus de 60 touches pour apporter les modifications prévues. Si l'on n'avait pas mis à profit les facilités de l'*éditeur* du 1251 et si l'on s'était « bêtement » contenté de retaper la ligne en entier, une trentaine de touches auraient suffi. Faut-il en conclure que les possibilités d'insertion, de destruction, de remplacement par recouvrement ne sont que des gadgets inutiles ? Certainement pas !

Deux remarques s'imposent ici. Tout d'abord, la ligne à corriger (exemple oblige) est spécialement truffée d'erreurs. Ensuite, et surtout, nous avons effectué les corrections de la façon la moins économique possible. Cela nous a permis de passer en revue la plupart des fonctions d'édition. Mais nous allons décrire maintenant une deuxième façon de faire plus rapide.

Nous repartons donc de notre ligne erronée : 250 IMPT « NOMBRE SUIVANT ? » ; S.

Après le message d'erreur, nous passons en mode PRO. Une pression sur ↑ et la ligne est affichée, pavé noir clignotant sur le I d'IMPT. On tape alors I. □ □ puis PREMIER NOMBRE, ▶▶▶▶ et A ENTER. La ligne a été corrigée en 27 pressions de touche.

A cela s'ajoute que les quatre touches fléchées (déplacement horizontal à l'intérieur d'une ligne, ou passage d'une ligne à l'autre) agissent aussi en répétition automatique. Il n'est donc pas indispensable d'appuyer 4 fois de suite sur ▶ pour déplacer le curseur de 4 positions vers la droite. Une pression prolongée sur la même touche produit le même effet, à condition que l'on ait suffisamment de réflexes pour relever le doigt au bon moment. Avec un peu d'entraînement, on y parvient presque à tous les coups. Si l'on prend en compte cette possibilité, notre ligne 250 est corrigée en 24 pressions de touche, soit 6 de moins que pour la réécrire en entier.

Répétons que l'exemple a été retenu parce qu'il permettait d'illustrer la plupart des fonctions dites d'édition : remplacement, suppression, insertion. Le plus souvent, quand il faut corriger une ligne de Basic, les modifications à apporter sont beaucoup plus légères et l'on a tout intérêt à ne pas retaper la ligne en entier. On s'aperçoit alors que le PC-1251 est très agréable d'emploi.

Si l'insertion (SHIFT INS) et la suppression d'un caractère (SHIFT DEL) agissaient en répétition automatique, l'ensemble serait presque parfait. Les cinq touches que l'on utilise sont judicieusement rassemblées, côte à côte, sur la première rangée du clavier.

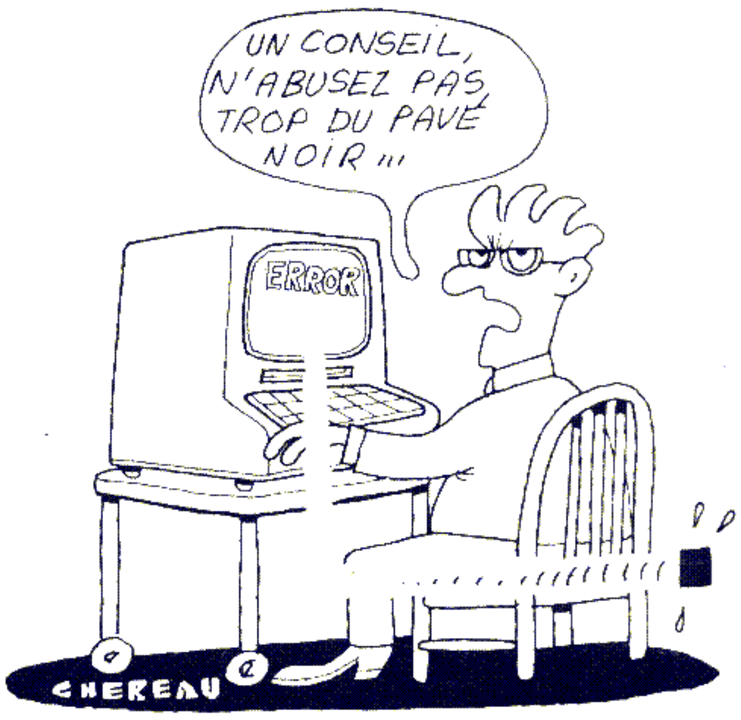
Souplesse et simplicité d'emploi

Pour finir, remarquons que la duplication des lignes de programme est obtenue très simplement. On positionne le curseur sur l'ancien numéro (250 dans notre exemple), on y substitue le nouveau (disons 965) et l'on presse sur ENTER. Il n'en faut pas plus pour obtenir une nouvelle ligne 965 identique à la ligne 250.

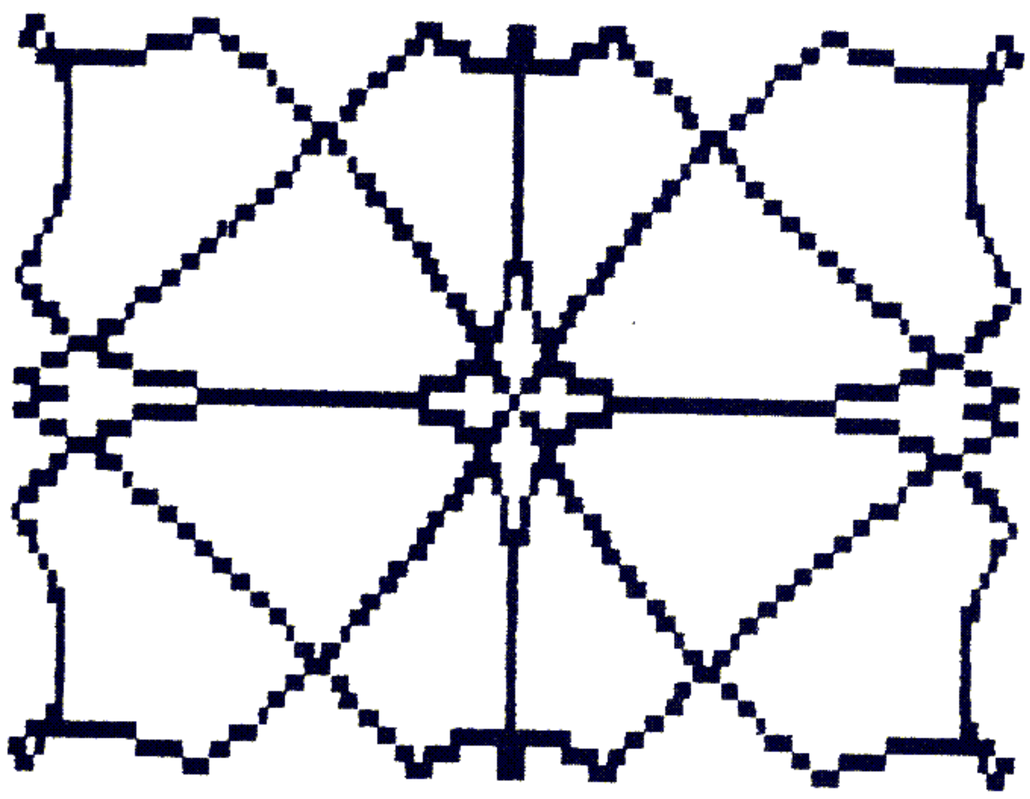
Souplesse et simplicité d'emploi sont, en résumé, les deux points forts de l'*éditeur* des PC-1211, 1212, 1245, 1251, etc. Certaines machines plus encombrantes sont, de ce point de vue, moins bien loties.

Je vous invite d'ailleurs à regarder quelles sont, en matière d'édition des programmes, les performances du matériel que vous pratiquez. Entrez donc la ligne 250 IMPT « NOMBRE SUIVANT ? » ; S et administrez-lui une bonne correction pour obtenir 250 INPUT « PREMIER NOMBRE ? » ; A.

Comptez le nombre de touches que vous devez frapper pour mener à bien cette opération...



Jean DRANO

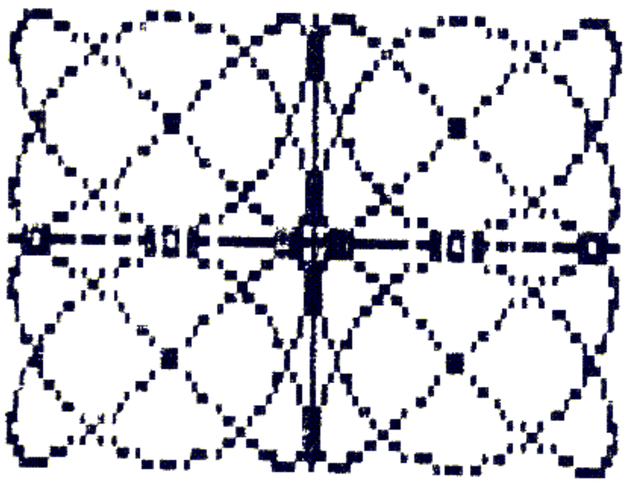


ROSACES EN TOUT GENRE

TEL qu'il est, le programme vous propose cinq dessins construits à partir de courbes. Il les calcule et les trace d'abord lentement, l'une après l'autre. Puis il passe à la vitesse supérieure, et l'écran de votre TRS s'anime. Mais vous pouvez aussi définir vos propres courbes...

L'association des calculs et du graphisme est à la base de ce que l'on appelle maintenant l'art informatique. Si l'on sait limiter sagement ses ambitions dans ce domaine, on obtient rapidement d'assez jolis résultats. Le court programme présenté ici est un bon exemple de ce qu'on peut réaliser en basse résolution sur un TRS-80 modèle 1 ou 3. Le caractère relativement standard du Basic employé devrait permettre une adaptation facile sur de nombreuses autres machines.

Dans un premier temps, cinq dessins constitués par des courbes plus ou moins simples sont tracés à l'écran et mémorisés. Ensuite, les courbes mémorisées s'enroulent et se déroulent en vitesse accélérée de manière cyclique. Pour mettre fin au spectacle, BREAK, RESET ou OFF.



Comme on le verra, les courbes se suivent et ne se ressemblent pas ; les matheux y reconnaîtront un cercle, une ellipse, une hypocycloïde, une astéroïde, etc., selon la valeur des quatre paramètres U, V, B et A lus dans les lignes de DATA.

Plusieurs courbes concentriques sont tracées (leur nombre dépend de la valeur des trois autres paramètres Z1, Z2 et Z3). Lorsque les coordonnées X ou Y sont en dehors du cadre, le tracé de la courbe se poursuit à partir du côté opposé (lignes 90 et 100).

Une courbe en deux temps

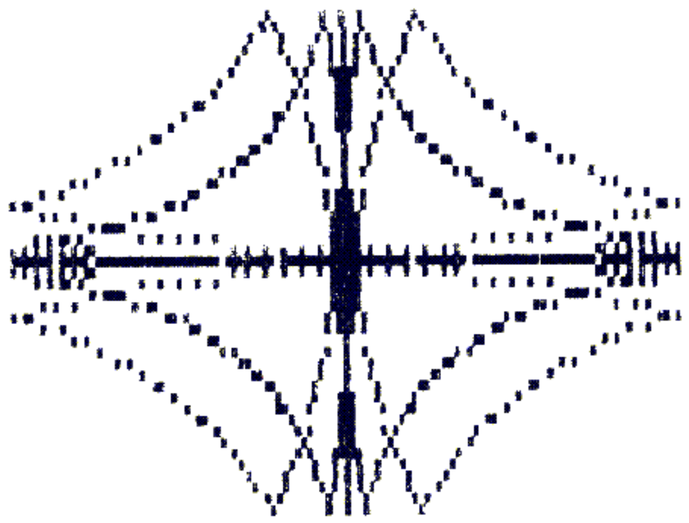
Dans la première phase de l'exécution du programme, le tracé de la courbe est lent : environ quatre points par seconde. La complexité des calculs en est la cause. Mais il n'en va pas de même par la suite. En effet, les coordonnées de chaque point sont sauvées dans un tableau AD (NP) (NP étant le nombre de points) indexé par le tableau DE (ND). La variable ND contient le nombre de dessins, de sorte qu'il devient

possible de redessiner à grande vitesse (50 points par seconde) les courbes mémorisées. L'animation de l'écran est alors étonnante.

On notera, à la ligne 120, l'astuce utilisée pour mémoriser dans une seule variable entière les coordonnées X et Y d'un point et, aux lignes 240 et 270, la méthode de restitution de ces deux valeurs.

En développant un peu le programme, il vous est loisible de sauver sur disque ou cassette la variable ND et les tableaux entiers DE (ND) et AD (NP) pour constituer une bibliothèque de dessins originaux, car avec un peu d'imagination, vous pouvez aussi modifier les paramètres qui déterminent les courbes à tracer...

Roger BROUSMICHE



Des chiffres et des dessins
Programme pour TRS-80 modèle 1 ou 3, 16 Ko
Auteur Roger Brousmiche

D'un Basic à l'autre

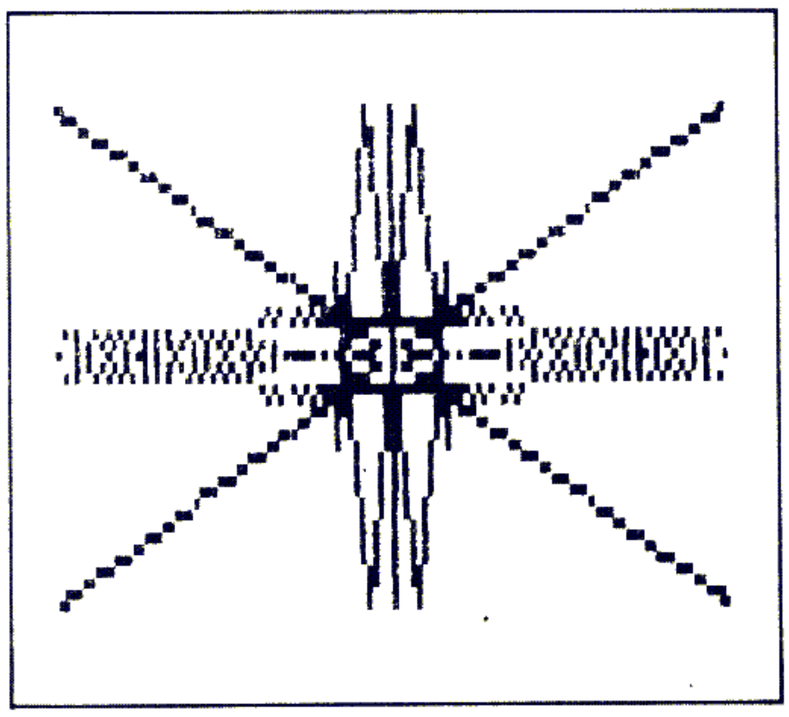
Les instructions graphiques diffèrent très souvent d'un Basic à l'autre. Si votre machine n'est pas un TRS-80, voilà ce qu'il vous faut savoir pour transposer le programme.

Sur les modèles 1 et 3, les trois principales instructions à retenir ici sont :
• SET : allumer un point (en fait un rectangle) ;
• RESET : éteindre un point ;
• POINT : tester si un point est ou non allumé.

Ces trois instructions sont toujours suivies de deux paramètres *x* et *y* qui désignent respectivement la ligne et la colonne du point choisi. Les lignes sont numérotées de 0 à 47, et les colonnes de 0 à 127, le point de coordonnées 0,0 se trouvant en haut et à gauche de l'écran. Ainsi, Set (15, 86) allume le rectangle à l'intersection de la 16e ligne et de la 87e colonne.

Enfin, deux rectangles contigus forment un carré.

```
1 '       DES CHIFFRES ET DES DESSINS
2 '       CONFIGURATION TYPE: MODELE 1 OU 3, 16K
3 '       ROGER BROUSMICHE 04/83
4 '
10 '        ***  INITIALISATION  ***
20 CLS : CLEAR 50 : DEFINT A-Y : ZP=ATN(1)*8
30 U=0 : V=0 : B=0 : A=0 : Z1=0 : Z2=0 : Z3=0
40 DIM AD(6750),DE(10) '10 DESSINS, 6750 POINTS MAX.
50 '     ***  COMPOSER DES DESSINS  ***
60 READ U,V,B,A,Z1,Z2,Z3 : IF Z3=9999 THEN 220
70 CLS : FOR Z=Z1 TO Z2 STEP Z3 : FOR Z0 = 0 TO ZP STEP ZP/A
80 X=Z*54*COS(U*Z0)+B+63.5 : Y=Z*22*SIN(V*Z0)+B+25.5
90 IF X)117 THEN X=X-108 ELSE IF X<9 THEN X=X+108
100 IF Y)47 THEN Y=Y-44 ELSE IF Y<3 THEN Y=Y+44
110 NP=NP+1 : SET(X,Y)
120 AD(NP)=256*Y+X : NEXT Z0,Z : ND=ND+1 : DE(ND)=NP
130 FOR I=1 TO 1000 : NEXT :GOTO 60
140 '     ***  DATA DESSINS  ***
150 DATA 1, 1, 1, 16, .7, .05, -.05
160 DATA 1, 1, 5, 500, 3, 2, -1
170 DATA 1, 1, 3, 16, 3, .1, -.05
180 DATA 3, 4, 3, 1300, 1.1, 1.1, 1
190 DATA 5, 7, 3, 1300, .9, .9, 1
200 DATA 0, 0, 0, 0, 0, 0, 9999
210 '     ***  DESSINER  ***
220 FOR I=1 TO ND
230 CLS : FOR J=DE(I-1)+1 TO DE(I)
240 SET(AD(J) AND 255,AD(J)/256) : NEXT J
250 FOR J=1 TO 1000 : NEXT J
260 FOR J=DE(I-1)+1 TO DE(I)
270 RESET(AD(J) AND 255,AD(J)/256) : NEXT J
280 NEXT I : GOTO 220
```



GAS-KIT 64
DESSIN ET MUSIQUE SUR COMMODORE 64

GRAPHISMES et sons ne sont pas les points forts du Basic Commodore 64. Pour accéder à ces domaines cachés, il faut un logiciel spécialisé. Gas-Kit 64 remplit son rôle : c'est un « kit » destiné aux Graphismes, Animation et Sons, comme son nom l'indique...

Gas-Kit 64, logiciel édité par Anirog et distribué en France par Infogrames, fait partie de ces utilitaires « créatifs » dont la nécessité se fait sentir un jour ou l'autre.

Si le titre peut paraître étrange, il renseigne (relativement !) sur les fonctions du logiciel : *Gas-Kit 64* est donc un *kit* destiné aux Graphics, Animation et Sounds (en anglais, mais faut-il traduire ?), sur C.64.

Voilà qui semble intéressant puisque d'un seul coup d'un seul, nos yeux et nos oreilles vont pouvoir — peut-être ! — être charmés. Voyons au prix de quels efforts...

La pochette plastique joliment décorée, qui s'ouvre comme un livre, renferme une cassette douillettement nichée dans l'alvéole prévue à cet effet, à l'abri d'un manuel d'instructions qui promet d'être passionnant. Ouvrons-le d'une main fébrile.

Alors, l'angoisse nous étreint de son bras glacé...

Une notice en anglais

Qu'est-ce à dire ? Un logiciel vendu en France, avec une notice en anglais ? Ciel ! Voici donc 30 pages d'écriture fine en dialecte grand-breton... Mais qu'ai-je fait de mon dictionnaire ? Bon, pas de dictionnaire dans mon environnement proche ; je joue le jeu : chargement immédiat du logiciel.

L'indication « turbo » sur le support magnétique laisse présager d'une vitesse de chargement confortable (mon magnétophone est-il bien réglé ?). Mais cette cassette a deux faces : cruel dilemme ! J'opte pour celle dont le titre est Show-Animation...

Après quelques instants de patience, le programme démarre seul, et ce que j'ai choisi à pile ou face apparaît à l'écran. Par chance, c'est un programme de démonstration ! Une image haute résolution (champêtre à souhait) s'affiche en même temps que quelques commentaires explicatifs (toujours en anglais). Quelques mouvements de lutins (*sprites*) animent le paysage, le tout ponctué par des séquences musicales. La démonstration graphique s'achevant en deux minutes environ, une démonstration musicale démarre, après changement du graphisme à l'écran.

L'aspect sonore de la démonstration est plus engageant que l'aspect graphique. La musique rend de beaux sons, tandis que les images fournies sont ternes et manquent de détails. L'animation des lutins semble par contre pleine de promesses.

Tous les messages et commentaires sont évidemment en anglais. Mon appréciation : sept sur dix. Le dernier

GAS-KIT 64 DESSIN ET MUSIQUE

message invite l'utilisateur à retourner la cassette après l'avoir rembobinée, pour charger les logiciels de l'autre face. Je m'exécute donc.

Pour la création graphique

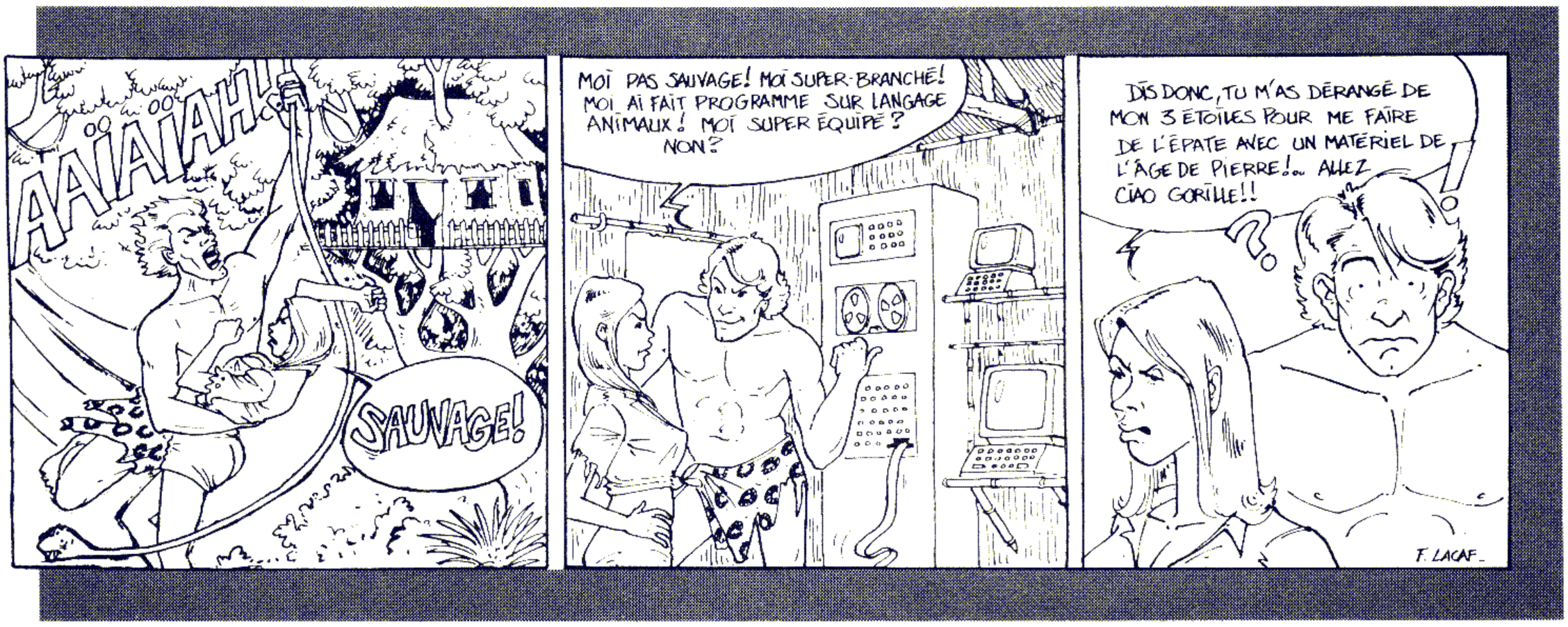
Le premier programme chargé porte le nom d'*Artkit*. Sa présentation indique qu'il s'agit d'un logiciel de création graphique. Toujours sans consulter la notice, je tente de comprendre. Un menu de huit fonctions permet un choix facile avec l'aide du clavier. Je choisis d'abord la fonction HELP, ce qui me permet d'apprendre les fonctionnalités nouvellement dévolues aux diverses touches, et m'indique que l'usage de la manette de jeux (ou *joystick*) est autorisé. J'ai enfin retrouvé mon dictionnaire : il va s'avérer très utile.

L'option CREATE fait réapparaître l'image de titre, que je m'empresse d'effacer en sélectionnant ERASE (''You really want to erase (y/n)?'' - Yes, yes, réponds-je). L'écran devient blanc, avec un point foncé au centre. Il est en position PENCIL. Les déplacements se font facilement, en laissant ou non une trace, avec les touches ou le « joystick ». Difficile tout de même de tracer des courbes régulières. Les lignes droites se dessinent sans problème. Une curiosité : le tracé des cercles définis à partir de valeurs numériques spécifiées par l'utilisateur, rayon et portion angulaire de circonférence. Inutile de dire que le résultat est rarement celui escompté...

Une image haute résolution, champêtre à souhait

Les couleurs ne sont pas vraiment faciles à manipuler : remplissages difficiles, précision insuffisante du mode BRUSH dont la taille est celle d'un pavé d'écran... Un entraînement sérieux semble indispensable.

La fonction SPRITE est plus intéressante et permet la création facile de lutins. Les fonctions SAVE et LOAD sauvegardent sur cassette et chargent les graphismes ainsi créés. Appréciation : six sur dix, je suis sévère.



Le second programme sur cette face de la cassette s'appelle *Demo* ; le suivant a pour titre *Democ*. Ne pouvant en découvrir l'utilité, je me vois obligé de consulter la notice d'utilisation. Heureusement, je n'ignore pas tout de la langue de Shakespeare... Mon dictionnaire et mes maigres connaissances me permettent d'extraire de ce livret touffu les informations indispensables.

Les manipulations ne sont pas vraiment simples. La première page du livret contient des « revised instructions » qui indiquent à l'utilisateur les actions à entreprendre pour recopier (sur deux cassettes réservées à cet usage) les éléments fondamentaux du système *Gas-Kit*.

Meilleur en musique qu'en dessin

Un tel logiciel aurait beaucoup gagné à être utilisable sur disquette. Les programmes *Demo* et *Democ* doivent être chargés à partir d'*Artkit*. Ils présentent la même image champêtre, haute résolution, que celle du Show rencontré précédemment.

J'ai omis d'essayer la fonction dite SEQUENCE CONTROL du même *Artkit*. L'essai, dirigé par la précieuse notice, est concluant. Cette fonction autorise le tracé « à répétition » de motifs définis par une chaîne de caractères.

Avec le quatrième et dernier programme de la cassette, *Composer*, vous pourrez créer des séquences musicales complètes. Le même système de menu ouvre un éventail de huit fonctions. PLAY permet de jouer après avoir sélectionné les diverses caractéristiques musicales de l'instrument à imiter. Les touches du clavier simulent celles d'un piano, le tout sur deux octaves.

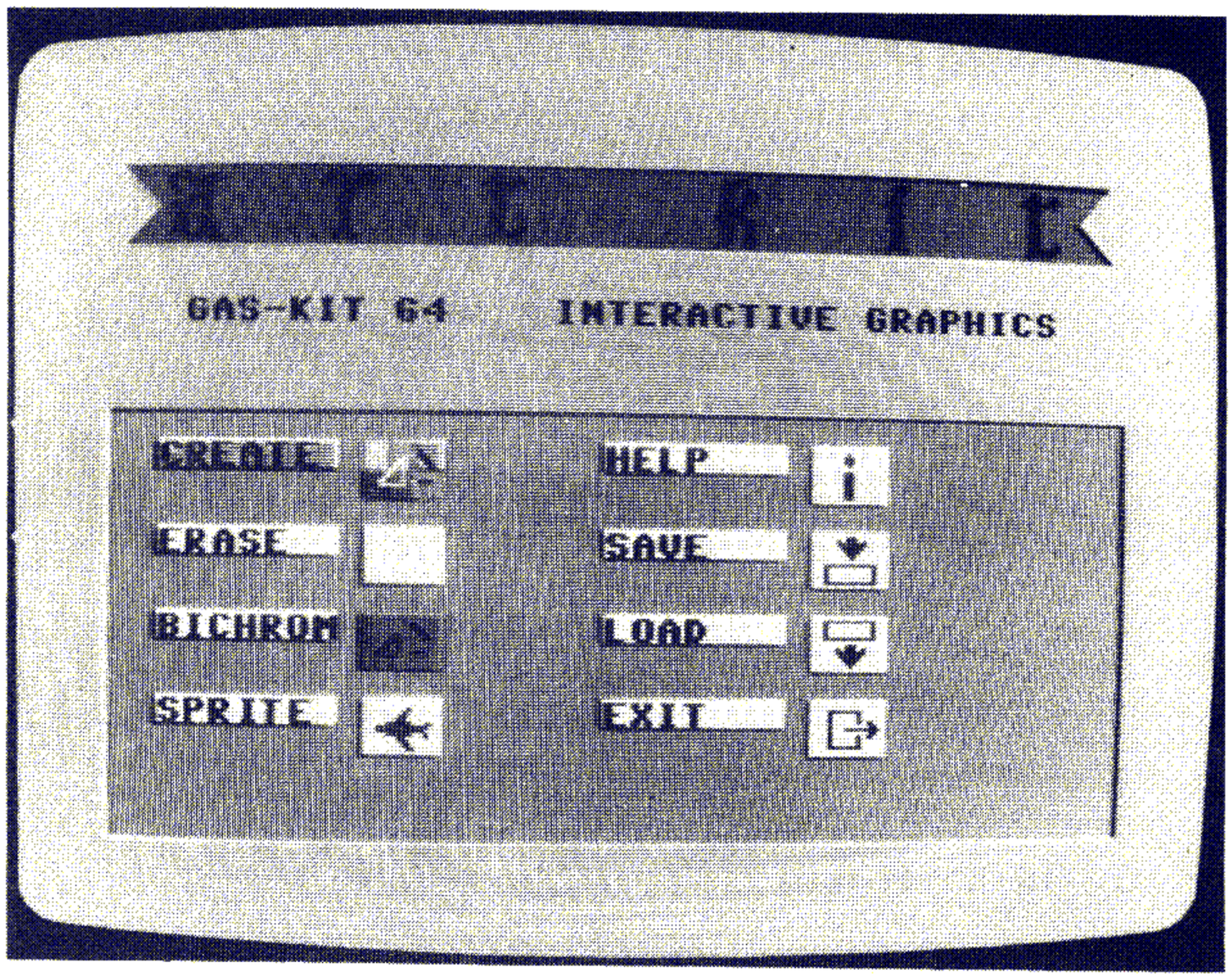
La séquence musicale jouée sur le clavier est mémorisée intégralement. REPLAY permet donc de la réécouter, en modifiant au besoin quelques paramètres (hauteur, timbre, vitesse, etc.).

STORE permet un stockage protégé en mémoire, avec un titre quelconque. Un catalogue est alors remis à jour ; il permet ensuite, grâce à FETCH, de choisir un morceau qui sera rejoué.

APPEND rassemble plusieurs titres en un seul dans la mémoire protégée.

WRITE et LOAD permettent un stockage et une récupération aisés sur cassette.

Cette partie de *Gas-Kit* est intéressante et bien plus facile d'emploi que la partie graphisme : huit sur dix.



Le menu proposé par Gas-Kit : huit fonctions

Un Basic étendu en prime

Artkit et *Composer* laissent en mémoire, lorsqu'on les quitte en utilisant EXIT, une intéressante extension au Basic du C.64. En effet, plus de vingt instructions nouvelles deviennent disponibles, dont la caractéristique commune est de débuter par un « @ ».

Cet ensemble se décompose en deux groupes : le premier groupe est formé des instructions qui permettent la mise en mémoire (au sein d'un programme Basic), des productions réalisées grâce à *Artkit* et *Composer* ; le second groupe permet la manipulation des productions ainsi récupérées.

Par exemple, GLOAD,"DEMO",1 remet en mémoire le dessin haute résolution ayant ce titre sur la cassette *Gas-Kit*. @HION provoque le passage de l'écran en mode haute résolution. @DOT,50,99 allume un point aux coordonnées spécifiées.

Pour les lutins, @DEF,2,7,0,2 allume le lutin défini comme portant le numéro 2, ayant été le septième mis en mémoire, de couleur noire, et disposé verticalement. @SPOS,*w*,*x*,*y* positionne le lutin numéro w aux coordonnées spécifiées, x et y.

En ce qui concerne enfin la musique, @PLAY, *tempo, hauteur,* rejoue un morceau de musique mis en mémoire.

Ces quelques exemples ne sont pas limitatifs, l'éventail des instructions nouvelles permet bien des fantaisies, en évitant au programmeur les longues séries de POKE fastidieux et souvent inefficaces. Appréciation : huit sur dix.

Le logiciel en quelques lignes

Nom : *Gas-Kit 64*
Ordinateur : *Commodore 64*
Forme : *cassette*
Edité par : *Anirog Software*
Distribué par : *Infogrames (Lyon)*
Prix public : *135 FF*
Principales orientations : *graphisme, animation graphique, sons*
Autre orientation : *extension du Basic*

Gas-Kit 64 est un logiciel complexe et complet, de qualité inégale dans ses différents volets d'utilisation. Nous avons vivement regretté cette détestable notice mal conçue et en anglais. Si vous en avez la possibilité, choisissez plutôt une version disquette du logiciel. Médiocre au niveau des graphismes, bon pour ses qualités de création musicale, et bon encore pour l'extension du Basic. Peut-être est-ce *Gas-Kit* qu'il vous faut ?

Robin BOIS

LOGOR
UN LOGO POUR ORIC-1 ET ATMOS

LOGOR n'est pas un véritable Logo, mais plutôt une première approche de ce langage, « une étape vers le jaillissement de l'esprit ». Pour les jeunes enfants et les classes de primaire, c'est un bon produit qui regroupe graphisme et musique. Il nous fait avancer à pas de tortue, avec beaucoup d'honnêteté, jusqu'à des limites clairement définies.

Six petites pages de présentation et d'explications suffisent à rendre le logiciel *Logor* parfaitement maîtrisable : pilotage de la tortue, procédures paramétrables, musique, etc.

Avec la tortue, un premier bon point : la longueur du pas et le nombre d'unités pour faire un tour complet sont programmables. En outre, la primitive ROND qui est absente du Logo classique, trace un rond en fonction d'un rayon donné.

La tortue donne sa position

Les modes LEVEPLUME, BAISSEPLUME, CACHETORTUE, MONTRETORTUE sont respectés, ainsi que la GOMME et l'INVERSEUR. En choisissant différentes couleurs, la tortue laisse même une trace sur un fond. Elle se programme en coordonnées relatives (AVANCE, RECULE, DROITE, GAUCHE) ou en coordonnées cartésiennes (POSITION en X Y, DIRECTION). Cependant les axes de coordonnées ne sont pas liés au centre de l'écran, et les valeurs programmables en X et en Y sont positives. Elles vont de 20 à 220 pour X, de 20 à 180 pour Y. Etonnant.

Le fait de pouvoir interroger la tortue sur sa position et sa direction est encore un bon point à mettre à l'actif de *Logor*. La tortue peut aussi écrire un texte à l'endroit où elle se trouve, ce qui permet de mettre facilement une légende à un graphique.

Quant à la musique, elle est programmable sur sept octaves en utilisant directement le nom des notes de la gamme chromatique. La durée des notes — croche, noire, blanche — et le SILENCE se programment aussi. La gamme est complétée par TIR, ZAP, PING, BOUM, un complément musical indispensable !

Les procédures sont limitées à 10. C'est une limite suffisante pour la majorité des utilisateurs (compte tenu d'une certaine expérience dans les écoles primaires).

En revanche, *Logor* bouscule la syntaxe de Logo sur les variables et les paramètres. Plus de caractère guillemet (") pour indiquer le nom, plus de deux-points (:) pour parler de contenu. D'ailleurs les variables ne sont pas définies par l'utilisateur, elles sont imposées : A contient une valeur angulaire, L une valeur linéaire. Il faut les initialiser en dehors des procédures, mais on peut les modifier à l'intérieur de celles-ci grâce à des expressions du type L + 5, A*3,... qui ne font plus intervenir l'affectation au sens informatique (L = L + 5 s'écrit ici L + 5). La manipulation des variables globales devient plus facile. Un petit « pied de nez » à l'informatique classique.

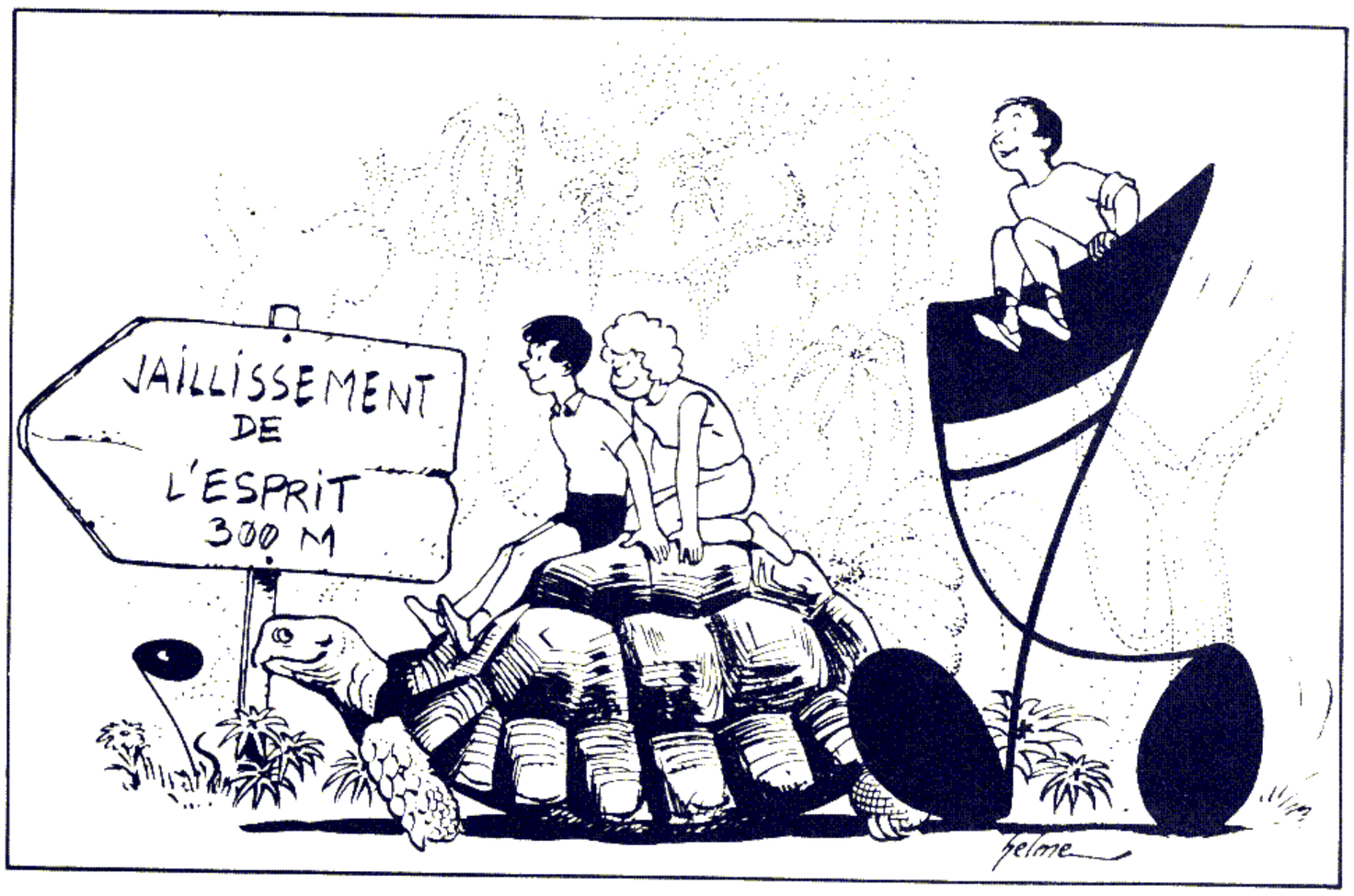
Ce logo n'est pas récursif

Avec *Logor*, le nom de la procédure est séparé de son corps par le caractère deux-points (:). Quant au RETURN, il est assimilé à une fin de procédure. Ce qui oblige à tout écrire sur une seule ligne (pouvant contenir jusqu'à 99 instructions, tout de même). Cette contrainte ne facilite pas les corrections.

Un gros point noir : l'auteur de ce logiciel semble confondre appel de procédure et récursivité. C'est sans doute ce dernier mot qui permettrait de don-

Le logiciel en quelques lignes

Nom : *Logor*
Ordinateurs : *Oric-1 et Atmos*
Forme : *cassette*
Edité et distribué par : *Infogrames*
Prix public : *160 FF*
Principale orientation : *première approche de Logo*



ner un « label Logo » au produit. *Logor* n'est pas récursif. Ce n'est pas bien grave dans le contexte de ce produit qui se limite à une initiation.

La possibilité de conserver les procédures écrites ou de les rappeler n'est indiquée nulle part.

Malgré une syntaxe peu adaptée à l'écriture de procédures, au passage des paramètres et surtout à l'absence de récursivité, *Logor* présente une bonne approche graphique et musicale des micromondes Logo. Les messages sont en français et clairs. *Logor* est un produit aux limites bien définies, les utilisateurs sont avertis. Amélioré dans le sens de la récursivité, il pourrait obtenir *le* label. Surtout que le microprocesseur et la taille mémoire de l'Oric permettent l'implantation d'un vrai Logo.

Robert DAGUESSE

LES COUPS D'ŒIL DE LIST

HAUTE RÉSOLUTION POUR ZX 81 16 Ko

HRG est un logiciel qui permet d'effectuer des tracés avec 256 x 220, soit 56 320 points. Même si toutes les combinaisons de points ne sont pas accessibles, et si l'on ne peut pas recopier l'écran sur l'imprimante, les résultats sont honorables.

Vous disposez d'un micro-ordinateur bon marché. Un ZX 81 pour tout dire. Même avec l'extension 16 Ko, ses possibilités demeurent bien limitées, évidemment. C'est vrai notamment pour le graphisme dont la résolution est très grossière. Vous aimeriez bien l'améliorer sur ce point. Allez, qu'à cela ne tienne !

Le ZX 81 affiche 22 lignes de caractères, chaque caractère étant défini dans une matrice de 10 sur 8 points, soit 80 points. Au total, l'écran utile comporte donc $22 \times 32 \times 80$ points, soit 56320 points. Oui, mais comment faire concrètement pour allumer ou éteindre l'un de ces points ? Première idée : il y a en mémoire morte une routine qui contrôle l'image envoyée sur l'écran. Détournons cette routine de telle sorte qu'au lieu d'envoyer 22 lignes, elle en envoie 220.

Chacune de ces nouvelles lignes est une série de 256 points (32 fois 8), autrement dit 32 caractères contigus comptant 8 points. Dès lors, pour obtenir n'importe quel motif en haute définition, il suffit de pouvoir éteindre ou allumer n'importe lequel de ces 8 points, ce qui fait 2^8. Soit 256 possibilités.

Dans l'idéal, pour exploiter vraiment un écran graphique de 56320 points, nous devons donc disposer, en mémoire vive, d'une table de 256 « caractères » : on déroutera le pointeur de la table d'origine pour accéder aux nouveaux caractères ainsi définis. C'est ce que fait *HRG* (comme Haute Résolution Graphique), ou plutôt ce qu'il ferait si, au lieu d'offrir 256 caractères, il n'en offrait pas 90 seulement.

Sept Koctets pour l'écran graphique

Ce programme pour ZX 81 version 16 Ko est écrit en langage-machine ; il occupe environ 850 octets dans une REM en ligne 0 et il utilise la méthode

HAUTE RÉSOLUTION ZX 81

décrite précédemment. L'écran graphique occupe 7 Ko de mémoire vive, ce qui laisse à l'utilisateur 8 autres Ko environ.

Pour simplifier la programmation en haute résolution, l'auteur d'*HRG* préconise d'entrer les adresses des routines utilisées dans des variables aux noms évocateurs. Ainsi : LET CLS = 16516 pour l'effacement de l'écran (qui sera dès lors appelé par RAND USR CLS), LET TEXT = 16519 pour le passage en mode texte, etc. Bien entendu, si vous êtes un peu fantaisiste ou si vous n'aimez pas l'anglais, rien ne vous empêche d'écrire LET POINTE = 16528 au lieu de LET PLOT = 16528, ou LET TRACE = 16531 au lieu de LET DRAW = 16531.

Des résultats convenables

On dispose en tout de 11 fonctions permettant l'effacement de l'écran, le passage en texte, le passage en HRG, l'affichage d'un caractère, d'un point, le tracé d'une ligne, le déroulement de l'écran vers le haut ou vers le bas, la sauvegarde et le chargement d'un écran graphique, l'inversion vidéo. Tout cela est bien, mais répétons-le, seules sont disponibles 90 combinaisons de 8 points (sur les 256 théoriquement possibles et nécessaires pour exploiter à fond une résolution de 56 320 points). D'autre part, aucune routine n'a été prévue pour la recopie d'écran sur imprimante ; il ne sera donc pas possible de garder des images des graphiques créés. Par ailleurs, l'utilisateur ne devra pas oublier de réserver la place de l'écran graphique avant le chargement du programme ou d'une page graphique, faute de quoi il devra débrancher son ZX.

Malgré ces limitations, *HRG* apporte, pour certaines applications, un net « plus » et il offre des résultats convenables. Il sera utile surtout pour visualiser des fonctions ou des courbes mathématiques ou certains effets spéciaux. Mais il ne faut pas pour autant en attendre des miracles ; en particulier, la haute résolution ne vaut que pour les programmes qui ont été écrits dans ce mode.

La notice (10 pages) signale correctement les possibilités et les limites de ce logiciel astucieux qui coûte 120 FF ttc.

Jacques DECONCHAT

Quelques exemples d'utilisation de la haute résolution fournis avec la cassette

```
  1 REM
  2 REM
  3 REM    HRG VERSION 7.0
  4 REM    DU  1983   DE
  5 REM
  6 GOSUB 1040
  7 REM  CE PROGRAMME DOIT
VOUS AIDER A COMPRENDRE LA
MANIERE D'UTILISER HRG 7.0
 10 RAND USR CLS
 40 RAND USR HRG
 50 REM  AXE DES X
 60 POKE 16508,110
 70 FOR X=0 TO 31
 80 POKE 16507,X
 90 POKE 16417,30
100 RAND USR PRINT
110 NEXT X
120 REM  AXE DES Y
130 POKE 16438,128
140 POKE 16439,220
150 POKE 16507,128
160 POKE 16508,0
170 RAND USR DRAW
180 REM  GRADUATIONS DES X
190 FOR X=1 TO 255 STEP 15
200 POKE 16438,X
210 POKE 16439,108
220 POKE 16507,X
230 POKE 16508,110
240 RAND USR DRAW
250 NEXT X
260 REM  GRADUATIONS DES Y
270 FOR X=205 TO 0 STEP -16
280 POKE 16438,127
290 POKE 16439,X
300 POKE 16507,129
310 POKE 16508,X
320 RAND USR DRAW
330 NEXT X
340 REM  COURBE SINUS EN PLOT
350 FOR X=0 TO 255 STEP 3
360 POKE 16507,X
370 POKE 16508,109+10*SIN X
024543,
380 RAND USR PLOT
390 NEXT X
400 REM  COURBE SINUS EN DRAW
410 POKE 16438,30
420 POKE 16439,110
430 FOR X=0 TO 225 STEP 3
440 POKE 16507,X+30
450 POKE 16508,109+10*SIN X
024543,
460 RAND USR DRAW
470 NEXT X
480 REM  INVERSION VIDEO
490 RAND USR INVR
500 FOR X=0 TO 40
510 NEXT X
520 RAND USR INVR
530 REM  SCROLL VERS LE HAUT
540 FOR X=1 TO 7
550 RAND USR HAUT
560 NEXT X
570 REM  SCROLL VERS LE BAS
580 FOR X=1 TO 7
585 RAND USR BAS
600 NEXT X
610 REM  COURBE HORIZONTALE
620 POKE 16508,1
630 FOR X=0 TO 255 STEP 3
640 POKE 16507,128+12*SIN X
024543,
650 IF X>128 THEN GOTO 700
660 POKE 16417,30
670 FOR Y=0 TO 31
680 POKE 16507,Y
690 POKE 16508,218
700 RAND USR PRINT
710 NEXT Y
720 POKE 16508,0
730 RAND USR PLOT
740 RAND USR HAUT
750 NEXT X
760 FOR X=0 TO 43
770 RAND USR HAUT
780 RAND USR HAUT
790 NEXT X
800 FOR X=0 TO 23
810 RAND USR BAS
820 RAND USR BAS
830 NEXT X
840 FOR X=0 TO 50
850 REM  FIGURE
860 NEXT X
870 RAND USR CLS
880 POKE 16438,0
890 FOR X=0 TO 255 STEP 3
900 POKE 16438,0
910 POKE 16439,255-X 255
920 POKE 16507,X
930 POKE 16508,0
940 RAND USR DRAW
950 NEXT X
960 REM  RETOUR AU MODE TEXTE
970 IF INKEY$="" THEN GOTO 970
980 RAND USR TEXT
990 STOP
1000 SAVE "RECAP
1010 IF PEEK 16388+256*PEEK 16389
=25344 THEN RUN
1020 PRINT AT 10,7;"PROGRAMME TAPE
HAUT"
1030 STOP
1040 REM  INITIALISATIONS
1050 REM  ---------------
1060 LET CLS=16516
1070 LET TEXT=16519
1080 LET HRG=16522
1090 LET PRINT=16525
1100 LET PLOT=16528
1110 LET DRAW=16531
1120 LET HAUT=16534
1130 LET BAS=16537
1140 LET SAVE=16540
1150 LET LOAD=16543
1160 LET INVR=16546
1170 RETURN
```

Le logiciel en quelques lignes

Nom : *HRG 7.0*
Ordinateur : *ZX 81 16 Ko*
Forme : *cassette*
Auteur : *Jean-Michel Cohen*
Edité par : *Sinclair*
Prix public : *120 FF*
Principale orientation : *graphisme haute résolution sur l'écran du ZX 81.*

PC-VISION

UN SUPER ÉDITEUR POUR SHARP PC-1500

PC-VISION est un utilitaire pour l'ordinateur de poche Sharp PC-1500 qui en transforme le mode de fonctionnement : éditeur pleine page, redéfinition du clavier et assignation de mots clefs du Basic notamment. Une fois en mémoire, PC-Vision se fait oublier ; il est même compatible avec la programmation en langage-machine. Cela étant, ce logiciel (écrit en langage-machine) occupe 1,5 Ko de mémoire vive.

PC-Vision fonctionne sur les PC-1500/1500 A et Tandy PC-2 muni d'une mémoire d'au moins 4 Ko. Seules les machines dotées des mémoires mortes ROM 2 à 5 peuvent l'utiliser (si, sur votre machine, PEEK &E2B9 donne 213 la mémoire morte est une ROM 1, dommage).

Le chargement en mémoire du programme n'est pas très compliqué : un petit « poke » pour définir l'écran virtuel (*PC-Vision* est d'abord un éditeur pleine page) suivi de CLOADM et c'est tout. Le programme se charge automatiquement.

Les touches du clavier sont redéfinies de manière plus fonctionnelle : souvent employés, les signes de la virgule et des deux-points sont accessibles directement grâce aux touches F5 et F6 (les touches F1 à F6 correspondent aux six touches situées au-dessous de l'afficheur). De même, les fonctions DEL et INS ont été relogées en F1 et F3. Ceci, ajouté à l'autorépétition des touches en cas de pression un peu prolongée, rend certaines manipulations bien plus aisées qu'auparavant : finies les interminables séries de SHIFT INS ou SHIFT DEL... Quant aux touches '', : DEL et INS'', elles donnent accès aux caractères spéciaux : '' _ | [et]''.

Des mots Basic sont affectés aux touches du clavier. On y accède en frappant SHIFT suivi d'un caractère. Ces fonctions sont : COLOR, CSIZE, USING, GOTO, GOSUB, RETURN, CSAVE, CLOAD, TEXT:, GRAPH:, RADIAN:, DEGREE:, LEFT$(, MID$(, RIGHT$(, INKEY$, PEEK&, POKE&, CALL&, SIN, COS, TAN, LN, EXP, LOG et 10^.

L'écran virtuel

L'écran du PC-1500 est limité à l'affichage de 26 caractères, mais chaque ligne peut en contenir jusqu'à 79. *PC-Vision* apporte un écran virtuel, c'est-à-dire la faculté de mémoriser un grand nombre de ces lignes. Par exemple, tapez, en mode RUN, la suite d'instructions :

```
RESOL/EQUATION      ↓
A =                 ↓
B =                 ↓
C =                 ↓
D = B*B-4*A*C       ↓
X1 = (-B+√D)/2/A    ↓
X2 = (-B-√D)/2/A    ↓
X1                  ↓
X2                  ↓
```

Vous venez de remplir les neuf premières lignes de l'écran virtuel. On peut remonter avec la touche ↑ et redescendre avec ↓. Mieux encore, il est possible de compléter les lignes 2 à 4, don-

nant ainsi des valeurs aux variables A, B et C (toujours avec ↓ entre chaque ligne). Jusque-là aucun calcul n'a été réalisé. On s'est servi de l'écran virtuel pour écrire du texte, comme on écrirait sur un carnet de notes ou d'adresses (c'est d'ailleurs une des utilisations de ce logiciel). La séquence de touches ENTER et ↓, répétée sur chaque ligne, va permettre d'effectuer ces calculs. En descendant ainsi de la première à la dernière ligne, on peut résoudre une équation du second degré en mode RUN. Ensuite, un peu comme avec un tableau de calcul, on pourra remonter modifier l'une ou l'autre des valeurs et recalculer le tout.

La définition des touches

L'écran virtuel est *théoriquement* illimité en nombre de lignes (seule compte la mémoire disponible) ; en revanche, le nombre de caractères par ligne ne peut pas excéder 79 (les mots clefs du Basic, comme SIN, comptent — après une pression sur ENTER — pour deux caractères). Il est possible de passer du mode PRO au mode RUN des lignes entières de fonctions. Les informations contenues dans cet écran virtuel demeurent en l'état tant qu'on ne les efface pas. Même l'extinction de l'ordinateur, qui n'affecte en rien le fonctionnement de *PC-Vision,* ne détruit pas son contenu.

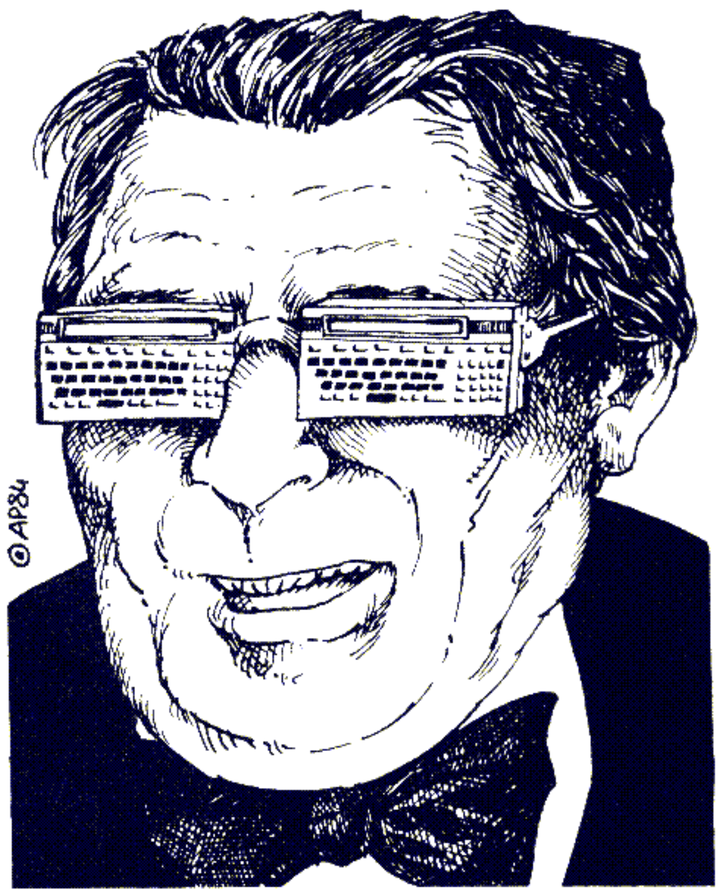
L'écran virtuel offre, en outre, une très intéressante possibilité de définition de touches, comme dans l'ancien mode RESERVE. Taper, par exemple sur une ligne de l'écran :

A : 2*PI/4 ↓

réserve les caractères 2*PI/4 au symbole A. Dès lors il suffira de presser RCL A pour les obtenir. On pourra stocker de cette manière toutes sortes de formules ou constantes scientifiques...

Pour l'édition proprement dite, la manipulation de ces lignes de texte, chiffres ou instructions, on dispose de fonctions spécifiques :

- DEF → et DEF ← provoquent un saut du curseur en fin ou en début de ligne ;
- DEF MODE effectue un saut du curseur, d'instruction en instruction ;
- DEF CL efface la ligne depuis le curseur jusqu'à sa fin ;
- DEF F1 supprime la ligne et affiche la suivante ;
- DEF F2 fait réapparaître une ligne malencontreusement effacée avec CL ;



- DEF F3 duplique une ligne en décalant les suivantes (c'est un moyen d'insérer de nouvelles lignes) ;
- DEF F4 suivi d'un caractère recherche la ligne commençant par l'étiquette correspondante ; ainsi, DEF F4 A trouve la ligne débutant par A), car un caractère suivi d'une parenthèse est une étiquette pour l'écran virtuel de *PC-Vision ;*
- DEF F5 suivi d'un caractère supprimera toutes les lignes comprises entre la ligne actuelle et la ligne étiquetée de ce caractère ;
- DEF F6 enfin recopie le contenu de l'écran sur l'imprimante si le commutateur PRINT de celle-ci est sur P.

Quand l'ordinateur est en mode PRO, DEF ENTER transfère une ligne de programme vers l'écran virtuel du *PC-Vision* ; on l'y traitera comme n'importe quelle autre ligne. En mode RUN, lorsque le curseur est au milieu d'une ligne, DEF ENTER la coupe en deux. SHIFT RCL positionne le curseur sur la première ligne de l'écran virtuel, tandis que DEF F4 ↓ permet d'atteindre la dernière ligne qui est symboliquement étiquetée "↓)", ce qui explique qu'on l'atteigne ainsi.

Parmi les autres fonctions d'édition, on retiendra aussi la possibilité de « couper-copier-coller » les lignes de texte ou de programme.

Le logiciel en quelques lignes

Nom : *PC-Vision*
Ordinateurs : *PC-1500 et 1500 A*
Forme : *Cassette*
Edité par : *Pocket-Soft*
Diffusé par : *XLog (21, rue du Général Foy, 75008 Paris)*
Prix public : *270 FF*
Principale orientation : *éditeur pleine page*

Avec son interface-cassette, le PC-1500 dispose de la fonction MERGE qui permet de faire cohabiter selon des règles très précises des programmes différents possédant pourtant les mêmes numéros de lignes. *PC-Vision* reproduit cette fonction MERGE sans requérir l'interface cassette (SHIFT n CL, où n est le numéro du programme distinct à créer), et permet aussi d'éditer l'un ou l'autre de ces programmes « mergés ». Le programme ainsi choisi peut être listé, modifié, exécuté, comme s'il se trouvait seul en mémoire (programme actif).

Deux modes distincts d'exécution d'un programme sont proposés ; on les choisit en allumant ou non l'indicateur RESERVE (SHIFT MODE). S'il est allumé, toutes les données introduites en réponse à un ordre d'INPUT sont stockées dans l'écran virtuel de *PC-Vision,* sinon l'exécution est normale.

Un code pour allumer la machine

Enfin, la touche OFF d'extinction de l'ordinateur est revue et corrigée : l'affichage, les textes de l'écran virtuel et, bien sûr, les programmes et les données sont conservés en l'état. A l'allumage, l'imprimante ne débite plus cinq inutiles lignes de papier. Une possibilité est offerte de protéger par un code l'utilisation de la machine : DEF OFF. Le PC-1500 ne reprendra du service qu'à la condition de presser simultanément les touches Q, A et ON. On peut choisir sa propre combinaison de touches (cette protection n'est pas sûre à 100 % mais dissuadera les non-initiés au système interne du PC-1500).

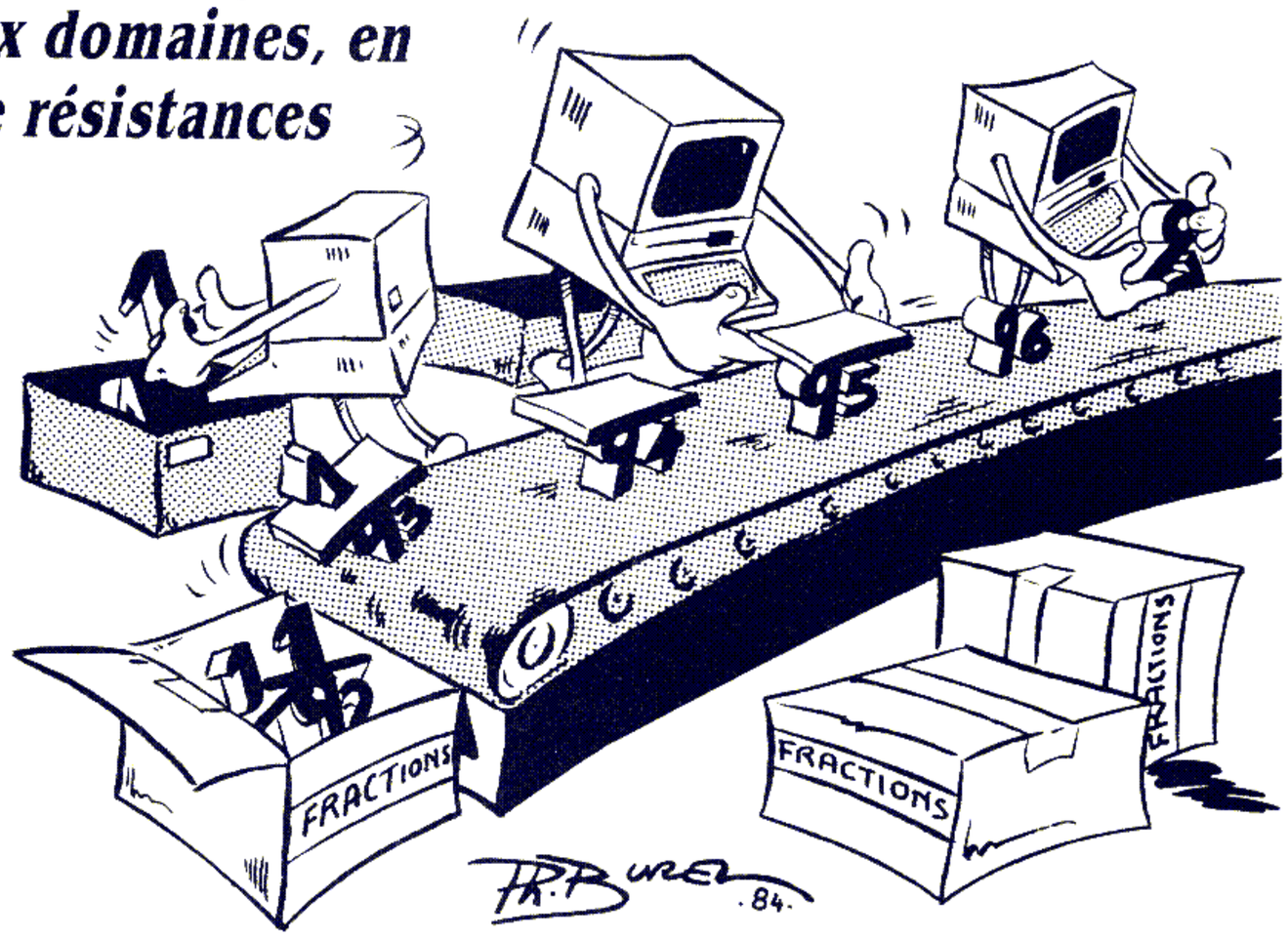
Deux critiques sont à formuler concernant *PC-Vision.* Une insuffisance d'abord : il ne faut pas tenter d'effacer une ligne de programme par DEF F1 sans qu'il en existe au moins une de réellement programmée, sous peine de « plantage ». Ensuite, la qualité de la documentation laisse vraiment à désirer : en français mais condensée sur 15 pages ; c'est complet, mais succinct.

On sait qu'en programmation, le confort de l'utilisateur dépend beaucoup de l'éditeur de la machine. D'origine, le PC-1500 est déjà bien loti de ce point de vue. Avec *PC-Vision,* il est encore meilleur. La cassette et sa brochure d'accompagnement sont vendues 270 FF ttc environ.

Jean-Christophe KRUST

FRACTIONS EN SÉRIE

CERTAINS petits problèmes appartenant au domaine des mathématiques gagnent à être abordés par le biais de la programmation. Il en est ainsi des fractions continues. Partant de l'algorithme d'Euclide, elles trouvent des applications dans de nombreux domaines, en particulier dans les réseaux de résistances électriques. Les algorithmes présentés ici conduisent à de petits programmes faciles à mettre au point.



En quatre lignes de Basic

Voici un utilitaire sans aucune prétention. Il répond à un très vieux rêve des anciens Grecs, qui croyaient que n'importe quel nombre était une fraction (on dirait aujourd'hui un rationnel).

Dans certains calculs, il peut être commode de remplacer un nombre x par une bonne approximation rationnelle A/B. Soit, trouver un rationnel (rangé dans la variable F) de telle sorte qu'il diffère de x de moins de 10^{-6}. On essaie le plus simple : B = 1, A = INT (F).

Tant que ça ne marche pas, on fait tourner une procédure élémentaire, celle que l'on appelle « développement en fraction continue » (les Grecs parlaient d' « antypharèse »). La démonstration de la convergence du procédé n'est pas très facile. Même si vous ne vous sentez pas capable de la faire ou de comprendre tout seul pourquoi ça marche, vous pouvez vous reporter au tome 2 de « Mathématiques pour l'Informatique Individuelle » de Daniel Jakubowicz et Hervé Lehning (Masson 1982), page 76.

```
10 INPUT F:B=1:C=1:D=0:G=F:E=INT(G):A=E
20 IF ABS(F-A/B)<10E(-6) PRINT A;"/";B:END
30 G=1/(G-E):E=INT(G):H=A*E+C:C=A:A=H
40 H=B*E+D:D=B:B=H:GOTO 20
```

Quoi qu'il en soit, l'efficacité et la rapidité de cet utilitaire sont excellentes.

André WARUSFEL

Ce qui suit doit être considéré comme un divertissement et non comme une « dose » de mathématique que certains avaient bien du mal à avaler lors de leurs études secondaires. Il s'agit, comme dans un jeu, de découvrir ce qui se cache derrière des formules apparemment bizarres.

Une suite d'allure curieuse

Un algorithme devenu célèbre, l'algorithme d'Euclide, permet de trouver le *plus grand commun dénominateur* (PGCD) de deux nombres entiers. Il existe sous plusieurs formes. Celle basée sur la division entière conduit à la fraction continue. Elle se présente de la manière suivante :

1. Soient deux nombres n_1 et n_2 dont on cherche le PGCD. (On pose $n_1 > n_2$.)

2. Si le nombre n_2 est égal à 0, aller en 4. Sinon, exécuter 3.

3. Calculs :
$q = \text{int}(n_1/n_2)$ où $\text{int}(n_1/n_2)$ est le quotient entier de n_1 par n_2 (par exemple, int(7/3) = 2, int(1/3) = 0, etc.)
$r = n_1 - n_2 * q$
$n_1 = n_2$
$n_2 = r$
Retour en 2.

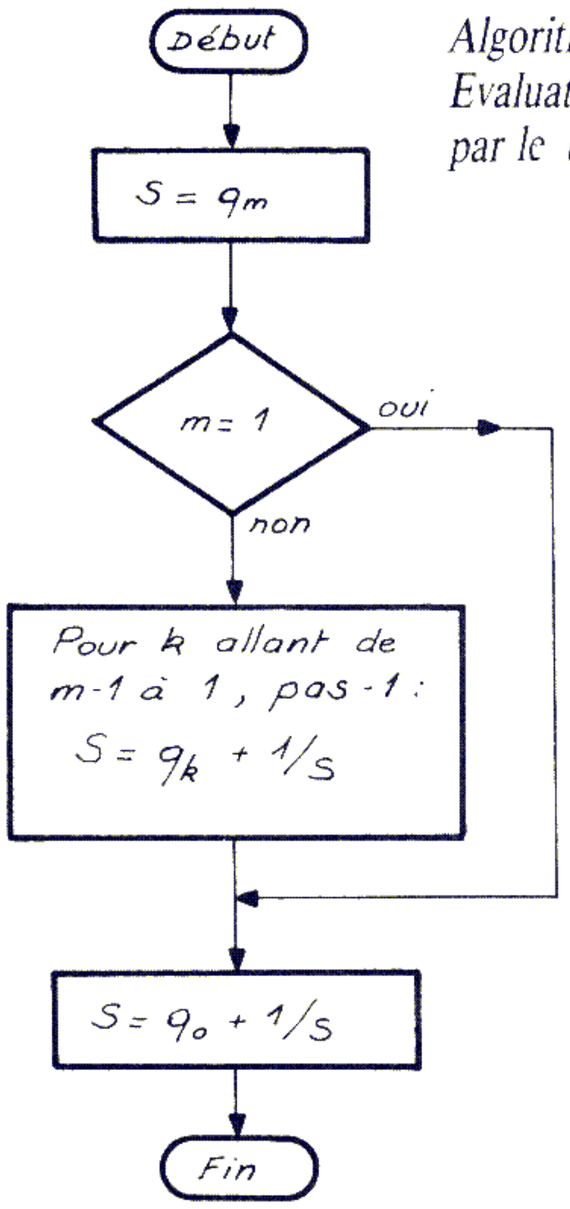
4. Le PGCD est n_1.

Si on désigne par q_i (pour i allant de 0 à m), les quotients obtenus successivement à chaque passage par les calculs de l'étape 3, alors le rapport n_1/n_2 peut s'écrire sous la forme d'une fraction continue : $n_1/n_2 = q_0 + 1/(q_1 + 1/(q_2 + 1/(q_3 + \ldots + 1/(q_{m-1} + 1/q_m)\ldots)))$

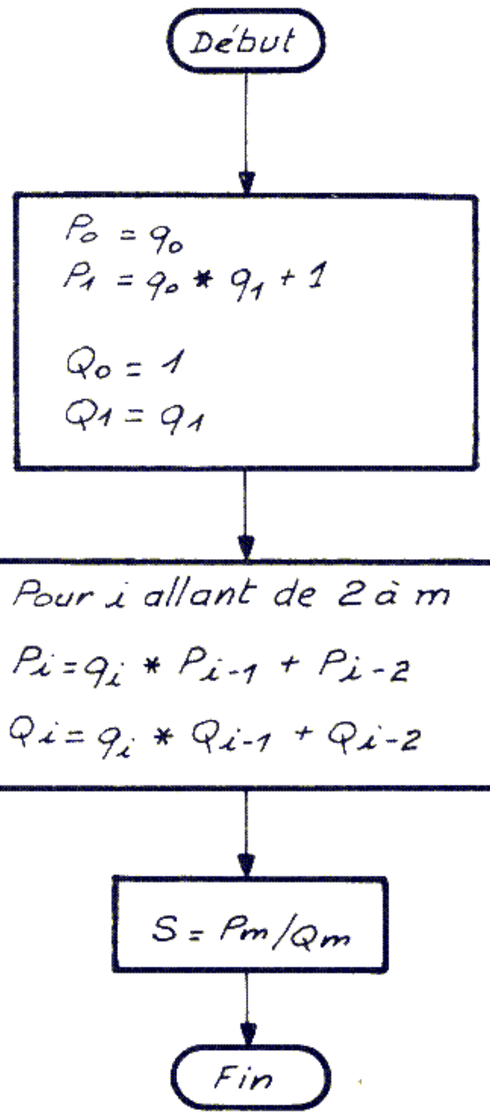
Sous une forme « condensée », cette fraction continue s'écrit : $n_1/n_2 = q_0 + [1/q_1, 1/q_2, \ldots, 1/q_{m-1}, 1/q_m]$

Pour l'évaluer, il suffit de partir du terme le plus bas, $1/q_m$, puis de remonter, $1/(q_{m-1} + 1/q_m)$, etc. L'algorithme 1 (page suivante) représente cette évaluation.

Comme à chaque étape, il est nécessaire de calculer l'inverse du résultat partiel S, l'exécution du calcul général est très ralentie. Un algorithme descendant est donc préférable : il augmentera



Algorithme 1
Evaluation d'une fraction continue par le bas



Algorithme 2
Algorithme descendant

la vitesse d'exécution (algorithme 2). L'évaluation ne demande plus alors qu'une seule division, P_m/Q_m, au lieu de la cascade de l'algorithme précédent.

Les fractions continues servent souvent à approcher un nombre irrationnel. Il arrive alors que les coefficients q_i qui les composent forment une suite infinie d'allure curieuse :

$\sqrt{2} \simeq 1 + [1/2, 1/2, 1/2, \ldots]$
$\sqrt{3} \simeq 1 + [1/1, 1/2, 1/1, 1/2, \ldots]$
$\sqrt{5} \simeq 1 + [1/4, 1/4, 1/4, 1/4, \ldots]$
$e \simeq 2 + [1/1, 1/2, 1/1, 1/1, 1/4, 1/1, 1/1, 1/6, \ldots]$

Les numérateurs qui apparaissent dans ces formes « condensées » sont tous à un. Ils peuvent prendre d'autres valeurs. Ainsi :

$\sqrt{7} \simeq 2 + [3/4, 3/4, \ldots]$

$\pi \simeq 2 + 2* [1/1, 2/1, 6/1, \ldots, m (m-1)/1, \ldots]$

La formule qui donne π est assez simple mais la convergence est très lente. Pour juger de l'intérêt de tels développements, il faudrait étudier la vitesse de convergence et le nombre de termes à prendre en compte pour une précision donnée. Surtout, il faudrait comparer cette méthode avec d'autres.

De telles formules peuvent être appliquées au calcul de la série $c_0x^0 + c_1x^1 + c_2x^2 + \ldots + c_nx^n$. On montre qu'elle se met sous la forme d'une fraction continue : $(c_0x^0 + c_1x^1 + \ldots + c_nx^n) = c_0 + [c_1x/1, -(c_2/c_1)\ x/(1 + (c_2/c_1)\ x), -(c_3/c_2)\ x\ /(1 + (c_3/c_2)\ x), \ldots]$

De même, l'expression $(1 + x)^m$ peut être mise sous la forme d'une fraction continue : $(1 + x)^m = 1 + [m\ x/1, (1-m)\ x/2, (1 + m)\ x/3, (2-m)\ x/2, (2 + m)\ x/5, (3-m)\ x/2, (3 + m)\ x/7, (4-m)\ x/2, \ldots]$

Et avec $x = 1$ et $m = 1/2$, on obtient bien le développement de $\sqrt{2}$.

Les fractions continues trouvent des applications dans l'interpolation de fonctions, mais aussi dans des domaines non mathématiques. En électricité, par exemple, le calcul de certaines résistances équivalentes (dans des réseaux de résistances) fait appel à ces fractions continues. En régime alternatif, les variables qui apparaissent sont même complexes. La fraction continue devient alors imaginaire. De quoi rêver !

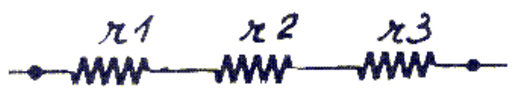
Claude NOWAKOWSKI

Pour les amateurs d'électricité

En électricité, la loi fondamentale du courant continu est la loi d'Ohm : $U = RI$ (U représente la différence de potentiel, R la résistance et I l'intensité du courant du circuit électrique).

Partant de cette relation, on montre que l'association de trois résistances en « série » r_1, r_2, r_3 (figure 1) a pour résistance équivalente $R = r_1 + r_2 + r_3$.

Figure 1
Réseau en série



Si les résistances sont montées en « parallèle » (figure 2), la résistance équivalente est donnée par :
$1/R = 1/r_1 + 1/r_2 + 1/r_3$

Figure 2
Réseau en parallèle

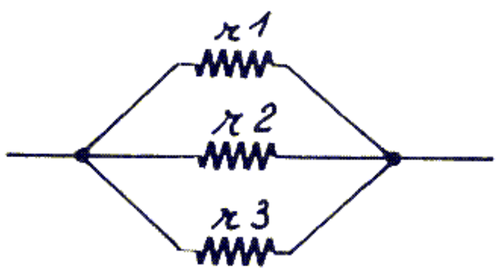
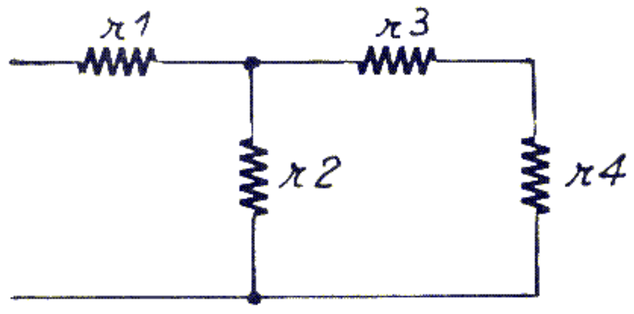


Figure 3
Réseau en échelle



Face à un réseau en échelle (figure 3), on trouve des résistances montées en série (r_1 et r_2, r_3 et r_4) et des résistances montées en parallèle (r_2 et r_{34}, où r_{34} est la résistance équivalente de r_3 et r_4). Ainsi :

$r_{12} = r_1 + r_2$
$r_{34} = r_3 + r_4$
$1/r_{234} = 1/r_2 + 1\ /\ r_{34} = 1/r_2 + 1\ /\ (r_3 + r_4)$
D'où la résistance équivalente R :
$R = r_1 + 1/(1/r_2 + 1/(r_3 + r_4))$

Si on mettait en dérivation sur r_4, un montage en série avec r_5 et r_6, tout se passerait comme si r_4 était remplacé par $1/(1/r_4 + 1/(r_5 + r_6))$

D'où la résistance équivalente R :
$R = r_1 + 1/(1/r_2 + 1/(r_3 + 1/(1/r_4 + 1/(r_5 + r_6))))$

Et, plus généralement, avec un réseau en échelle composé de n résistances :
$R = r_1 + [1/1/r_2, 1/r_3, 1/1/r_4, \ldots, 1/r_{n-1}, r_n]$

On observe ainsi que la résistance équivalente d'un réseau en échelle est donnée par une fraction continue !




MES MONSTRES M'OBÉISSENT


AVEC le programme pour Oric publié dans le précédent LIST, nous avons appris à créer des petits monstres. Aujourd'hui, nous allons leur insuffler la vie, ou du moins les animer. Il ne leur manquera plus que la parole.

Sur l'écran d'un téléviseur, le mouvement n'est qu'une illusion (optique, évidemment). L'animation consiste, en fait, à afficher des dessins les uns après les autres. Il faut donc que l'affichage soit assez rapide pour tromper l'œil : jusqu'à plusieurs milliers de points par seconde. Pour manipuler des centaines de caractères (six points chacun), on aura recours à une routine en langage-machine.

Autrement dit, on doit se ménager un véritable système d'exploitation des dessins.

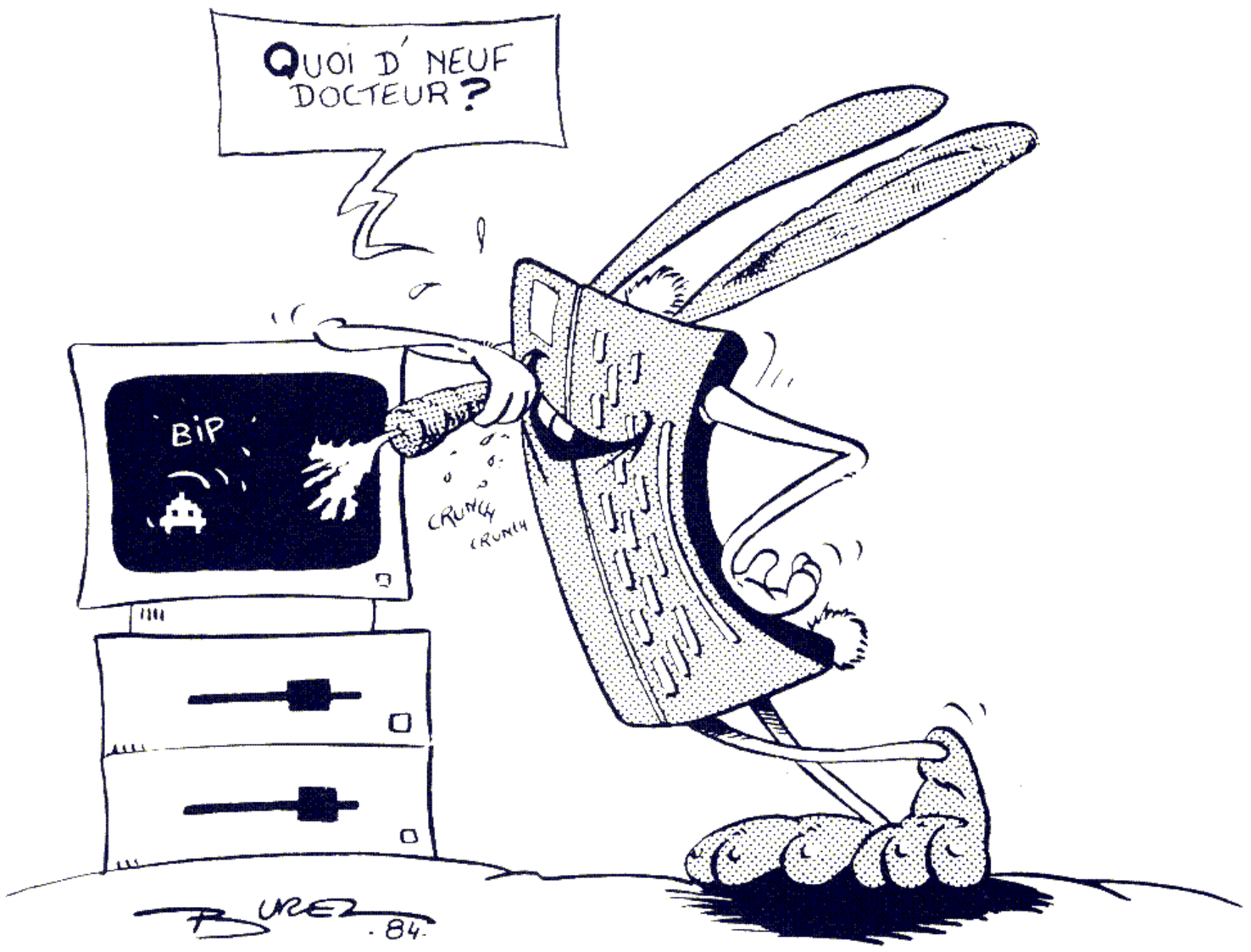
Au programme que nous avons présenté le mois passé, et qui était conforme aux principes de la programmation fonctionnelle, il suffit d'ajouter les nouvelles fonctions :

- G garde un dessin en mémoire,
- P place un dessin sur l'écran,
- S supprime un dessin de la mémoire,
- A anime les dessins indiqués,
- les quatre flèches spécifient la direction du mouvement,
- la barre d'espace produit une animation sur place,
- F spécifie le format d'un motif sur l'écran,
- ↑ F (ou Ctrl F) permet le retour au format précédent,
- L liste des dessins,
- ↑ S (Ctrl S) supprime tout,

Les petits monstres se déplacent

Il faut aussi que le dessin à animer soit d'un format suffisamment réduit. En effet, la mémoire ne peut contenir que deux dessins de la taille de l'écran, alors qu'elle peut stocker jusqu'à 300 différents monstres de petites dimensions (16 × 16) qui, de plus, pourront se mouvoir et traverser l'écran.

Il faut enfin avoir le moyen de gérer les dessins en mémoire : conserver ou supprimer, placer sur l'écran n'importe quel dessin, en donner la liste, etc.



Animez vos monstres
Extension du programme pour Oric
publié dans *LIST* 5, page 58
Auteur Max Hagenburger


```
70 IF GARD THEN GOSUB 1200
72 IF PLAC THEN GOSUB 1300
74 IF SUPP THEN GOSUB 1400
76 IF ANIM THEN GOSUB 1500
78 IF MOUV THEN GOSUB 1600
80 IF FORM THEN GOSUB 1700
82 IF CFOR THEN GOSUB 1800
84 IF LIST THEN GOSUB 1900
86 IF CSUP THEN GOSUB 2000
88 IF CCOU THEN GOSUB 2100

115 DDES=#5000 :REM debut dessins
117 IF PEEK(DDES)>0 THEN DOKE DDES,0
175 :DSS1=3200 :GOSUB 3300 'ss/prog

255 IF VAL(A$)>0 THEN VIT%=VAL(A$)
260 GARD=(A$="G")
265 PLAC=(A$="P")
270 SUPP=(A$="S") :CSUP=(A=19)
275 ANIM=(A$="A")
277 MOUV=(A=>8 AND A<=11 OR A$=" ")
280 FORM=(A$="F") :CFOR=(A=6)
285 LIST=(A$="L")
287 CCOU=(A=3)

1200 PRINT "Garde";
1210 :GOSUB DSS1
1220 IF EXIS THEN PRINT" change 0"H$;
:GET O$:IF O$<>"0" THEN RETURN
1225 IF EXIS THEN 1275
1230 LX=INT((X9-X0)/6)+1 :LY=Y9-Y0+1
1240 HV=ADRS+LY*LX+3
1250 IF HV>#97FF THEN PING:RETURN
1255 POKE ADRS,ASC(DES$)
1260 POKE ADRS+1,LX
1265 POKE ADRS+2,LY
1270 DOKE HV,0 '---
1275 POKE 1,LX :POKE 2,LY
1280 DOKE 4,ADRS+3
1285 CURSET X0,Y0,3
1290 :CALL #4F00
1295 RETURN :REM -----
1300 PRINT "Place";
1310 :GOSUB DSS1
1320 IF NOT EXIS THEN PING:RETURN
1330 POKE 0,63*(1-TRA)
1340 POKE 1,LX :POKE 2,LY
1350 DOKE 4,ADRS+3
1360 :CALL #4F2B
1395 RETURN :REM -----
1400 PRINT "Supprime";
1410 :GOSUB DSS1
1420 IF NOT EXIS THEN PING:RETURN
1430 PRINT :PRINT "  confirme 0"H$;
1440 GET O$:IF O$<>"0" THEN RETURN
1450 SUIV=ADRS+LX*LY+4
1460 DOKE 1,ADRS
1470 DOKE 3,SUIV
1480 :CALL #4F40
1495 RETURN :REM -----
1500 PRINT "anime ("ANIM$")"
1510 IF DEEK(DDES)=0 THEN PING:RETURN
1520 ANIM$="" :AANIM=1
1530 :GOSUB DSS1
1540 IF NOT EXIS THEN PING:POKE 617,2
:GOTO 1530
1550 ADRS(AAN)=ADRS :ANIM$=ANIM$+DES$
1560 LX(AAN)=LX :LY=(AAN)=LY
1570 AAN=AAN+1 :IF AAN>10 THEN RETURN
1580 PRINT :GOTO 1530
1595 RETURN :REM -----
1600 PRINT "animation";
1605 IF ANIM$="" THEN RETURN
1610 POKE 0,63*(1-TRA)
1615 REPEAT :C=PEEK(520)
1620 NA=NA+1 :IF NA=AANIM THEN NA=1
1625 IF C=132 THEN 1650
1630 :X=X+((C=172)-(C=188))*VIT%
1635 :IF X<X0 OR X>X9 THEN X=X0
1640 :Y=Y+((C=156)-(C=180))*VIT%
1645 :IF Y<Y0 OR Y>Y9 THEN Y=Y0
1650 POKE 1,LX(NA) :POKE 2,LY(NA)
1655 DOKE 4,ADRS(NA)+3
1660 CURSET X,Y,3 :POKE 526,0
1670 :CALL #4F2B
1675 MUSIC 1,5,RND(1)*12+1,8
1680 UNTIL C=56
1690 MUSIC 1,0,1,0
1695 RETURN :REM -----
1700 PRINT "Format:";
1710 :GOSUB INPT
1720 X=INT(X/6)*6 :XX=INT(XX/6)*6+5
1730 IF X>XX OR Y>YY THEN PING:RETURN
1740 T=0 :GOSUB CADRE
1750 X1=X0 :X8=X9 :Y1=Y0 :Y8=Y9
1760 X0=X :X9=XX :Y0=Y :Y9=YY
1770 T=1 :GOSUB CADRE
1795 RETURN :REM -----
1800 PRINT "Format echange";
1810 IF X1=0 THEN PING:RETURN
1820 T=0 :GOSUB CADRE
1830 R=X0:X0=X1:X1=R :R=X9:X9=X8:X8=R
1840 R=Y0:Y0=Y1:Y1=R :R=Y9:Y9=Y8:Y8=R
1850 T=1 :GOSUB CADRE
1895 RETURN :REM -----
1900 PRINT "Liste: ";
1910 ADRS=DDES+1 :LIS$=""
1920 IF DEEK(ADRS-1)=0 THEN 1980
1930 LIS$=LIS$+CHR$(PEEK(ADRS))
1940 ADRS=ADRS+PEEK(ADRS+1)*PEEK(ADRS
+2)+4 :GOTO 1920
1980 PRINT LIS$" ("ADRS-DDES-1")";
1990 GET A$
1995 RETURN :REM -----
2000 INPUT"Supprime tout 'O'";O$
2010 IF O$="O" THEN DOKE DDES,0
2095 RETURN :REM -----
2100 PRINT "'Couleur ";
2110 :GOSUB INPT
2120 IF X=XX OR Y=YY THEN RETURN
2130 FOR I=Y TO YY STEP SGN(YY-Y)
2140 FOR J=X/6 TO XX/6 STEP SGN(XX-X)
2150 P=40960+40*I+J :Q=PEEK(P)+128
2160 IF Q=>256 THEN Q=Q-256
2170 POKE P,Q
2180 NEXT J,I
2195 RETURN :REM -----

2840 PRINT"Txt 'EPYCo For Gar Pla Sup
Ani Lis"

3200 PRINT " dessin ? "; :GET DES$
3210 IF DES$<=" " THEN POP :RETURN
3220 PRINT DES$;
3230 EXIS=FALSE :ADRS=DDES+1
3240 IF DEEK(ADRS-1)=0 OR ADRS>#97FF
THEN RETURN
3250 IF ASC(DES$)=PEEK(ADR) THEN 3270
3260 ADRS=ADRS+PEEK(ADRS+1)*PEEK(ADRS
+2)+4 :GOTO 3240
3270 EXIS=TRUE
3280 LX=PEEK(ADRS+1):LY=PEEK(ADRS+2)
3295 RETURN :REM -----
3300 REM ss/prog assembleur
3310 IF DEEK(LB)=#02A6 THEN RETURN
3320 REPEAT :READ D$
3330 FOR I=1 TO LEN(D$) STEP 2
3340 V=VAL("#"+MID$(D$,I,2))
3350 POKE LB,V :LB=LB+1 :NEXT
3360 UNTIL D$="60"
3365 DATA A602A000B1109104C8C401D0F72
0134FD0F060
3370 DATA 18A501650485049002E60518A92
8651085109002E611CA60
3375 DATA A602A000B10445009110C8C401D
0F520134FD0EE60
3380 DATA A000B1039101C8D0F9E602E604A
504C997D0ED60
3385 DATA 60
3395 RETURN :REM ========
```

Les routines assemblées

```
adres  l.machine   assembleur    ;interpretation

4F00:  A6 02        LDX $02      ;X decompte les lignes
       A0 00     L1 LDY #$00     ;Y=0, debut de ligne
       B1 10     L2 LDA ($10),Y  ;charge 1 caractere ecran
       91 04        STA ($04),Y  ;et le range en memoire
       C8           INY          ;Y+1, caractere suivant
       C4 01        CPY $01      ;compare fin de ligne ?
       D0 F7        BNE L2       ;si non egal -> charge
       20 13 4F     JSR $4F13    ;ss/prog ligne suivante
       D0 EF        BNE L1       ;si X>0 -> debut de ligne
       60           RTS          ;retour programme Basic

4F13:  18           CLC          ;retenue (addit.) a zero
       A5 01        LDA $01      ;charge longueur ligne
       65 04        ADC $04      ;    +  adresse memoire
       85 04        STA $04      ;range l'adresse
       90 02        BCC L3       ;si retenue,
       E6 05        INC $05      ;  +1 sur adresse haute
       18        L3 CLC          ;retenue a zero
       A9 28        LDA #$28     ;charge largeur ecran 40
       65 10        ADC $10      ;    +  adresse ecran
       85 10        STA $10      ;range l'adresse
       90 02        BCC L4       ;si retenue,
       E6 11        INC $11      ;  +1 sur adresse haute
       CA           DEX          ;X-1 ligne suivante
       60        L4 RTS          ;retour ss/prog appelant

4F2B:  A6 02        LDX $02      ;
       A0 00     L5 LDY #$00     ;
       B1 04     L6 LDA ($04),Y  ;charge caractere memoire
       45 00        EOR $00      ; OU exclusif avec TRA
       91 10        STA ($10),Y  ;le place sur l'ecran
       C8           INY          ;
       C4 01        CPY $01      ;
       D0 F5        BNE L6       ;  (idem prog. 4F00)
       20 14 4F     JSR $4F14    ;
       D0 ED        BNE L5       ;
       60           RTS          ;

4F40:  A0 00     L7 LDY #$00     ;debut de bloc
       B1 03     L8 LDA ($03),Y  ;charge l'ancienne posit.
       91 01        STA ($01),Y  ;met a nouvelle position
       C8           INY          ;position suivante/bloc
       D0 F9        BNE L8       ;si meme bloc -> charge
       E6 02        INC $02      ;nouveau bloc suivant
       E6 04        INC $04      ;ancien bloc suivant
       A5 04        LDA $04      ;charge ancienne adresse
       C9 97        CMP #$97     ;si non limite memoire,
       D0 EF        BNE L7       ;  -> debut
       60           RTS          ;retour au Basic
```

• ^ C (Ctrl C) donne la couleur inverse,
• les chiffres de 0 à 9 représentent la vitesse de déplacement dans l'animation.

Chaque commande déclenche l'appel d'un sous-programme, par exemple IF ANIM THEN.GOSUB 1500, et la fonction est reconnue grâce à la valeur d'une variable logique (soit vraie, soit fausse) ; ainsi, ligne 275, on a ANIM = (A$ = "A"). La séquence d'instructions correspondant à la fonction proprement dite peut de cette façon être étudiée séparément du reste du programme (pour la fonction *anime*, les lignes 1500 à 1580).

On trouvera la partie en langage-machine stockée en data aux lignes 3365 à 3385. Pour simplifier le traitement, nous avons préféré traiter les cellules telles qu'elles représentent l'image de l'écran **en mémoire** et non pas point par point. Chaque cellule est un ensemble de six points de l'écran dans le sens horizontal. Ces mouvements selon cet axe iront donc de six en six points.

Avant l'appel de cette routine, le programme communique les paramètres suivants :
• la largeur LX par le registre 1 (adresse de la mémoire),
• la hauteur LY par le registre 2,
• l'adresse en mémoire par le double registre 4,
• l'adresse de l'écran par le double registre 10 (hexa), mais c'est aussi l'adresse du curseur qui est automatiquement gérée par CURSET,
• un masque enfin par le registre 0 qui donne une image normale ou inverse selon que la trace est ou non visible (variable logique TRA).

Bien sûr, grâce à la programmation fonctionnelle, rien ne vous empêche d'ajouter encore de nouvelles commandes à notre programme de monstres avec, par exemple, insertion de caractères géants, sauvegarde des dessins sur cassettes ou sur disquettes, etc.

Max HAGENBURGER

UNE PROCÉDURE PASCAL

GÉNÉRATION D'UN FICHIER

L'INITIALISATION d'un fichier dépend de la forme de celui-ci. Un programme pour chaque initialisation, ce serait trop. La procédure Pascal décrite ici est générale : elle s'applique à tous les fichiers, quelle que soit leur structure, elle est plus rapide qu'un sous-programme trop spécifique, elle optimise l'encombrement en fonction du nombre de blocs utilisés sur le support magnétique.

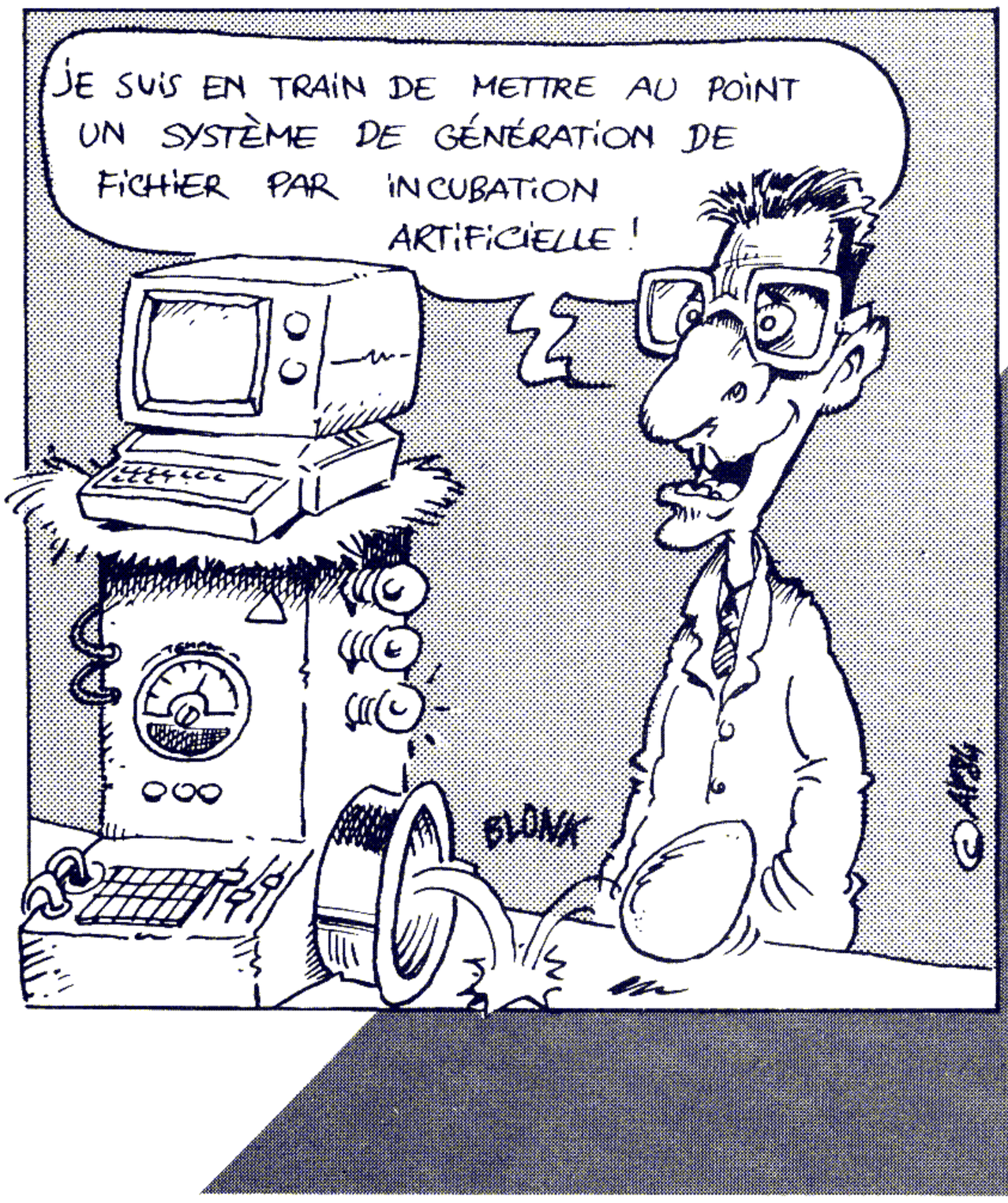
Au début de la procédure de génération d'un fichier vide, une fonction est déclarée. Elle possède trois paramètres qui indiquent les caractéristiques du fichier à créer.

Le premier, NOM, est une chaîne de caractères qui mémorise le nom du fichier. Ce nom doit être complet, c'est-à-dire qu'il doit spécifier sur quel périphérique (par exemple, le numéro du disque), cette création est effectuée.

Le second, NOMBRE, indique le nombre d'enregistrements que le fichier

UNE PROCÉDURE PASCAL

GÉNÉRATION D'UN FICHIER



Il arrive souvent que des fichiers soient créés et initialisés avec une longueur fixe, déterminée à l'avance. Cela permet de simplifier l'écriture des programmes et d'éliminer les problèmes posés par les fichiers de longueur variable (ceux qui s'étendent en fonction des besoins... jusqu'à être bloqués entre deux autres fichiers).

Valeur d'initialisation en fonction du type du champ d'enregistrement

Type du champ de l'enregistrement	Valeur utilisée pour l'initialisation
Logique (boolean)	False
Intervalle (e.g. : 0..255)	Première valeur de l'intervalle
Scalaire (exemple : (PRINTEMPS, ETE, AUTOMNE, HIVER))	Premier identificateur de l'énumération (exemple : PRINTEMPS)
Entier (integer)	0
Entier long (longinteger ou integer [N])	0
Réel (real)	0.0
Caractères (char)	Caractère NUL (chr(0))
Chaînes (string)	Chaîne vide de longueur 0 (''')

peut mémoriser. Cette valeur est du type entier, permettant de générer des fichiers de 1 à 32767 enregistrements. Rien ne s'oppose, si le système d'exploitation le permet, à une déclaration de ce paramètre en entier long. Ainsi, les fichiers créés pourront comprendre un plus grand nombre d'enregistrements.

Tous les types de fichiers

Le troisième paramètre, LONGUEUR, spécifie la longueur, en octets, d'un enregistrement. Les compilateurs disposent généralement d'une fonction standard permettant de déterminer cette valeur. Pour le système UCSD, un des plus répandus, cette fonction s'appelle SIZEOF, et a comme unique paramètre le nom du type (ou de la variable) déclaré avec le fichier.

La fonction retourne alors une valeur logique : elle est vraie si le fichier a été correctement généré et initialisé, elle est fausse si le fichier n'est pas créé.

Cette fonction permet de générer tous les types de fichiers, en initialisant normalement les différentes zones des enregistrements. La valeur de l'initialisation est variable selon les types des champs des enregistrements (voir tableau « Valeur d'initialisation en fonction du type du champ d'enregistrement »).

Grâce à la fonction déclarée ici, les fichiers sont créés au démarrage de chaque application. Et la rapidité du traitement fait que cette fonction peut encore être utilisée à chaque initialisation d'un nouveau fichier : l'ancien est alors détruit.

La création s'effectue d'une façon très simple. En voici un exemple avec un fichier d'articles. Il est écrit en Pascal UCSD, et utilise la pseudo-fonction SIZEOF qui retourne ici la longueur en octets du type ou de la variable représentant un enregistrement du fichier :

```
if creer_fichier ('#4:ARTICLES', 2000, sizeof (article))
    then
      write ('Creation correctement effectuee')
    else
      write ('Erreur : fichier non cree')
write (', tapez [return] pour continuer.');
readln;
```

Il faut noter que les fichiers enregis-

Procédure de génération d'un fichier vide

```
function creer_fichier (nom : string ; nombre : integer ; longueur : integer) : boolean
const    lonbloc = 512 ;
var      i          : integer ;
         correct    : boolean ;
         nbrbloc    : integer ;
         nbroctet   : integer [6] ;
         bloc       : file of packed array [1..lonbloc] of 0..255 ;
begin
  {$I-}
  nbroctet : = nombre ;
  nbroctet : = nbroctet * longueur ;
  nbrbloc : = trunc ((nbroctet + lonbloc-1) div lonbloc) ;
  rewrite (bloc, nom) ;
  correct : = ioresult = 0 ;
  for i : = 1 to lonbloc do bloc ^[i] : = 0 ;
  i : = 1 ;
  while (i < = nbrbloc) and correct do
    begin
      put (bloc) ;
      correct : = correct and (ioresult = 0) ;
      i : = i + 1
    end ;
  if correct
    then
      begin
        close (bloc, lock) ;
        correct : = correct and (ioresult = 0)
      end ;
  creer_fichier : = correct
  {$I+}
end ;
```

trés sur support magnétique sont souvent structurés en secteurs de 256 octets et en blocs de 512 octets. Le dernier secteur ou le dernier bloc, s'il n'est pas totalement utilisé, est entièrement réservé par le système pour le fichier considéré. Cette fonction tient compte de la possibilité de créer un ou plusieurs enregistrements supplémentaires sans que la longueur du fichier s'allonge sur le disque ou la bande magnétique. Dans ce cas, le nombre d'enregistrements réellement initialisés peut être supérieur à celui spécifié par le paramètre NOMBRE.

La fonction opère en deux temps : elle se charge de calculer les paramètres de création du fichier, puis elle procède à son initialisation.

La première opération détermine la longueur du fichier en nombre d'unités de mesure pour le disque, c'est-à-dire en nombre de secteurs ou de blocs. Le sous-programme calcule cette longueur en blocs de 512 octets. Pour qu'il puisse être utilisé avec des systèmes ayant le secteur comme mesure de structuration des disques, il faut mettre la constante LONBLOC à 256.

Le nombre d'octets nécessaires au fichier est déterminé, et mémorisé dans la variable NBROCTET. Le type de cette variable, INTEGER [6], sera éventuellement modifié selon les impératifs du compilateur utilisé. Il est nécessaire que cette variable puisse mémoriser un entier pouvant avoir jusqu'à six chiffres significatifs (fichiers de 976 Ko).

Un entier court dans un entier long

Avant tout calcul, le nombre d'enregistrements est stocké dans la variable NBROCTET. C'est presque toujours indispensable pour la suite. En effet, si le produit de NOMBRE par LONGUEUR est directement effectué, le résultat est faux sur de nombreux compilateurs : NOMBRE et LONGUEUR étant des entiers courts, l'expression est évaluée selon ce type. Puisque le résultat ne peut, en général, pas être mémorisé sous ce format (la limite du type INTEGER est égale à 32767), il se produit un débordement faussant le reste des opérations. Ce problème est contourné en mémorisant l'entier court dans un entier long, puis en effectuant l'opération.

Le nombre de secteurs ou de blocs est ensuite évalué. Il est approximativement égal au nombre d'octets divisé par la longueur d'un secteur ou d'un bloc. Comme il doit être entier, il faut tenir compte du cas où le dernier bloc est partiellement utilisé. Ce nombre est alors augmenté de un. La lecture du programme permettra de découvrir l'astuce qui effectue cela sans test, avec une seule expression arithmétique.

Le tampon servant à la génération du fichier est ensuite rempli avec des zéros. Puis le fichier est ouvert en création. Bien entendu, après chaque instruction d'entrée/sortie, la valeur de IORESULT, qui permet de savoir si une telle opération s'est bien déroulée, est contrôlée. Cette valeur ne peut être testée que grâce à l'option de compilation {$I-}, qui supprime ces contrôles habituellement effectués par le système d'exploitation.

Le reste du traitement consiste à créer le fichier. Le nombre de secteurs ou de blocs déterminé dans la première étape permet de réaliser cela. Ensuite, le fichier est refermé, et sa création est confirmée avec l'instruction CLOSE (BLOC, LOCK). Enfin, le résultat de la fonction permet de savoir si le traitement s'est déroulé correctement.

Cette fonction rend de nombreux services, grâce à son caractère général et à sa rapidité. Elle s'insérera donc en bonne place dans une bibliothèque de sous-programmes techniques.

Thierry CHAMORET

LE BASIC DE L'AMSTRAD

LE constructeur britannique Amstrad, avec son CPC 464, vient de casser les prix de l'informatique familiale. C'est une chose. Il va sans doute faire un malheur sur ce marché. Mais il y a beaucoup mieux : le Basic de sa machine est tout simplement l'un des meilleurs qui soient.

L'Amstrad CPC 464 est le premier ordinateur commercialisé par Amstrad. Cette firme britannique s'était cantonnée jusqu'à présent dans la fabrication de matériel acoustique grand public où elle occupe d'ailleurs une place très enviée outre-Manche. A l'inverse de la quasi-totalité de ses concurrents, le nouvel ordinateur familial possède dans son emballage d'origine tout ce qui est nécessaire à son fonctionnement et à son exploitation.

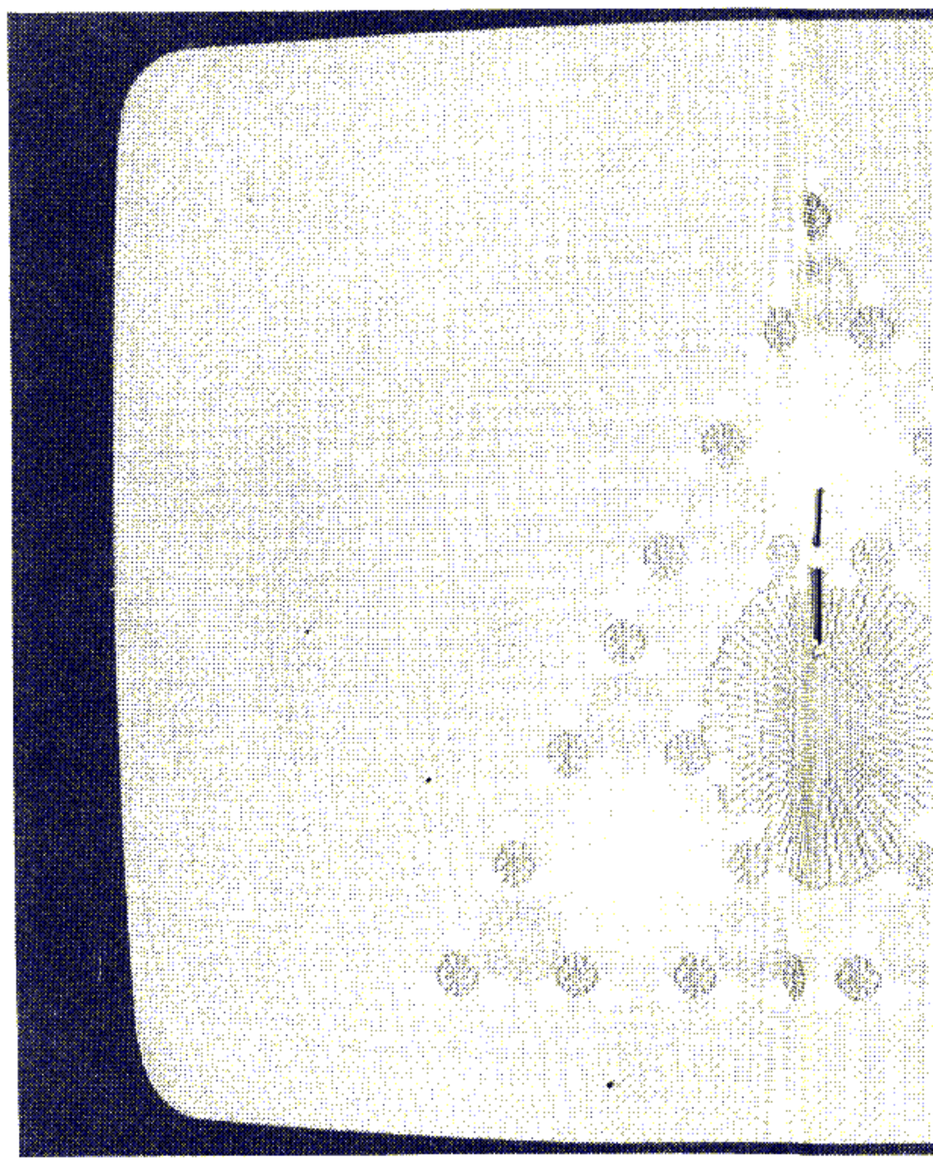
Tout d'abord, un bloc unité centrale comprenant un clavier QWERTY mécanique, agréable au toucher et très bien conçu (pavé numérique séparé et pavé de gestion du curseur). Certaines touches stratégiques se font remarquer en rouge, bleu ou vert. A la droite du pavé numérique se trouve un magnétocassette intégré qui, s'il n'est pas entièrement pilotable par logiciel, possède l'indispensable compteur.

Le moniteur (couleurs pour la version à 4 490 francs, noir et blanc pour celle à 2 990 francs) se connecte par l'intermédiaire de deux prises sur l'unité centrale : l'une pour l'alimentation, l'autre pour le signal vidéo. Seul le moniteur se connecte au secteur.

Les branchements sont donc très peu nombreux et quelques secondes après la sortie de son emballage, la machine est prête à fonctionner.

Malgré les apparences, l'éditeur de l'Amstrad n'est pas « plein écran ». Il y a pourtant un pavé de « gestion de curseur » qui permet de le déplacer dans les quatre directions n'importe où sur l'écran, mais les touches d'effacement sont alors inopérantes. En revanche, si l'on tape du texte et si l'on appuie sur l'énorme touche ENTER, le texte que l'on vient d'afficher est validé. Certaines fantaisies sont ainsi permises : si l'on déplace le curseur en plein milieu du message qui apparaît à la mise sous tension et si l'on tape 10 PRINT « COUCOU ! » suivi d'ENTER, la ligne 10 est effectivement créée. Veut-on la modifier ? Toute tentative utilisant directement le pavé de flèches se solde par un échec. On doit taper EDIT 10 (l'espace après EDIT comme après tout mot-clé du Basic Amstrad est obligatoire). On peut alors déplacer horizontalement le curseur ; la touche DEL supprime les caractères situés à sa gauche et la touche CLR les caractères situés à sa droite. De plus, la commande EDIT place automatiquement en mode insertion. Tout caractère frappé se fraie ainsi une place sans détruire ses voisins. Il paraît impossible d'écrire par recouvrement. En tout cas, je n'ai pas réussi à mettre en évidence un mode dans lequel les caractères tapés se substitueraient aux anciens. Ajoutons qu'il est possible de déplacer le curseur sur le numéro d'une ligne de manière à dupliquer celle-ci.

Il existe une deuxième méthode pour éditer une ligne : on déplace d'abord le curseur sur le début de la ligne à modifier en pressant sur Shift et l'on s'aperçoit que le curseur s'est dédoublé. En appuyant alors sur la touche COPY, on duplique la ligne à corriger. Sur le

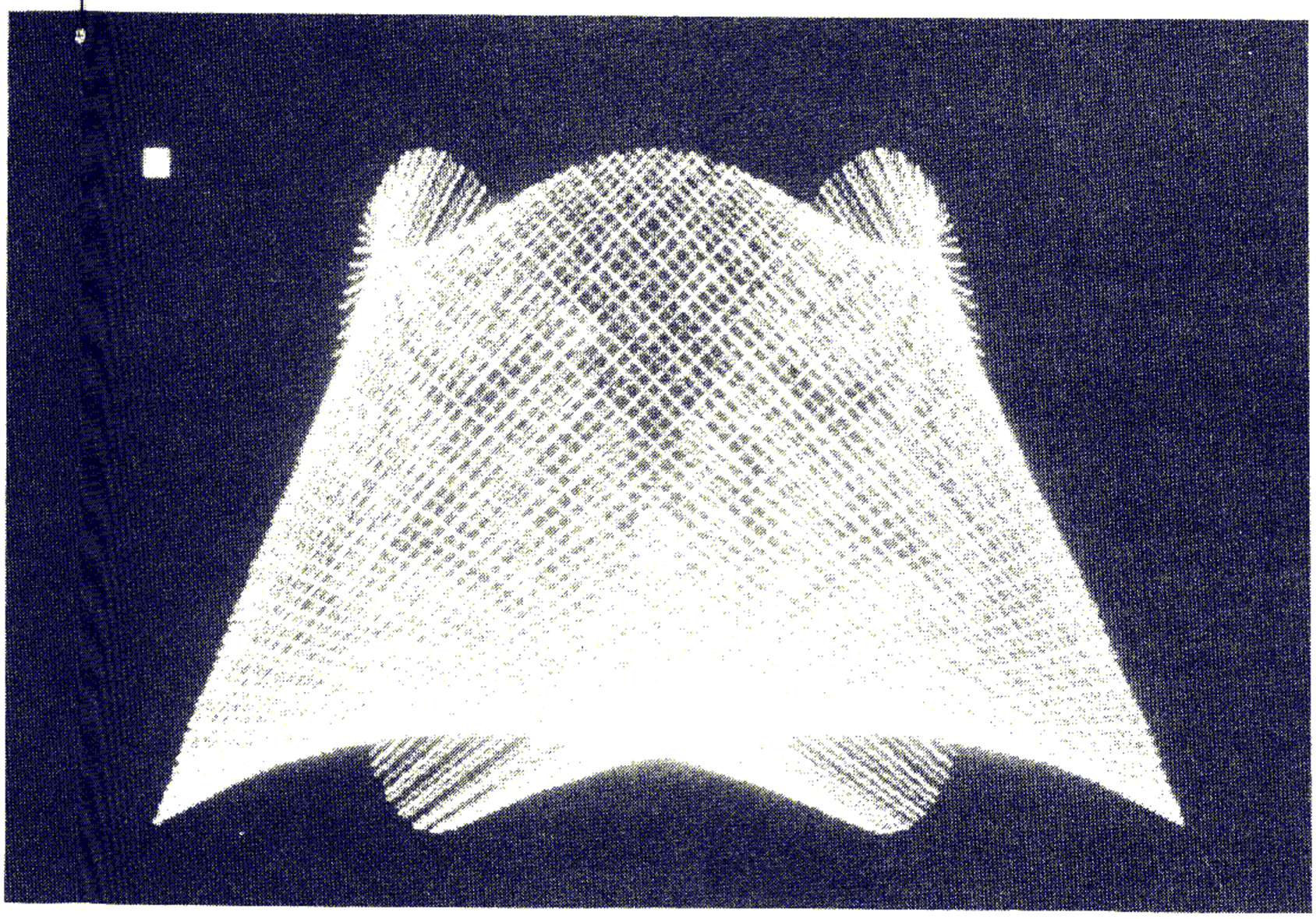


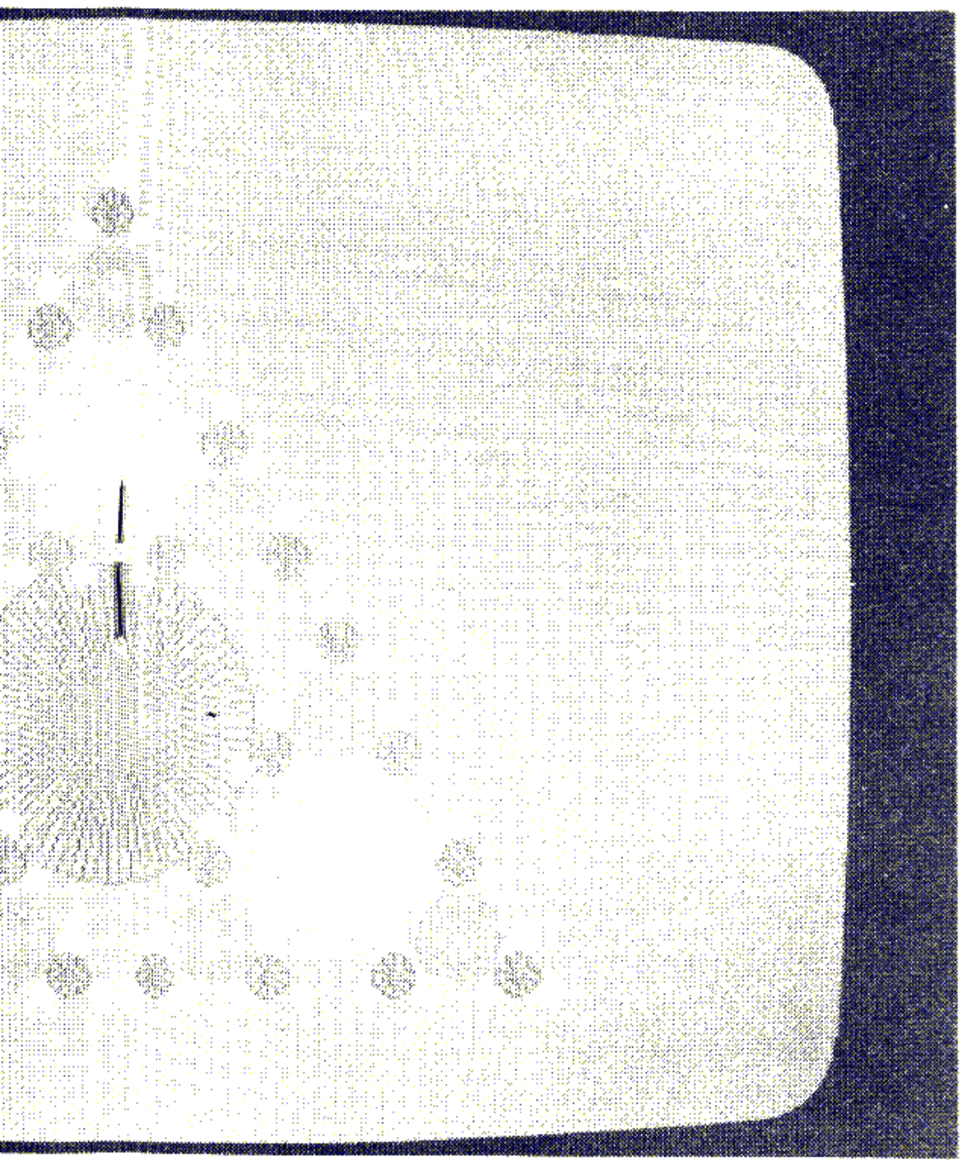
Une cassette de démonstration, livrée avec l'Amstrad, propose des dessins de toutes les couleurs

« clone » ainsi obtenu, qui s'inscrit à partir du premier curseur, on effectue toutes les corrections voulues, exactement comme en mode EDIT.

Les autres commandes d'édition sont classiques, mais les voir toutes réunies a de quoi réconforter : AUTO numérote automatiquement, DELETE détruit le groupe de lignes spécifié et

La démonstration joue sur la haute résolution graphique, un point fort de l'Amstrad





RENUM renumérote un programme en entier.

Ajoutons que le nombre maximum de caractères pour une ligne Basic est de 255.

Le Basic du CPC 464 autorise des noms de variables allant jusqu'à 40 caractères, tous significatifs : un véritable luxe. Le type chaîne de caractères peut être spécifié par l'instruction DEFSTR ou par le suffixe $. A côté de cela, il existe deux types de variables numériques : les variables entières codées sur deux octets et dont les valeurs peuvent être comprises entre −32768 et +32767 (suffixe % ou instruction DEFINT). Les variables « réelles » ont neuf chiffres significatifs. Notons que pour une telle précision, les calculs utilisant les fonctions mathématiques sont effectués très rapidement. C'est ce type de variables qui est choisi par défaut. Les valeurs traitées vont alors de −1,7E38 à +1,7E38. En valeur absolue, la plus petite valeur possible (à part zéro !) est 2,9E-39.

Les instructions CREAL et CINT permettent de convertir les variables d'un type à l'autre. La dimension d'une variable indicée n'est limitée que par le nombre maximum de caractères sur une ligne.

L'instruction ERASE annule des déclarations implicites ou explicites de variables, ce qui a pour effet de libérer la place mémoire correspondante. Cette instruction est très utile quand la résolution d'un problème nécessite la déclaration successive de plusieurs très grands tableaux qui ne pourraient pas tous cohabiter en mémoire simultanément.

Le trio READ, DATA et RESTORE répond présent à l'appel, RESTORE pouvant être suivi d'un numéro de ligne.

Traitement de faveur pour les chaînes

Les chaînes de caractères disposent d'un traitement de faveur : on trouve toutes les fonctions souhaitées, même la puissante INSTR qui détecte une chaîne dans une autre chaîne et en donne la position. Cette recherche peut d'ailleurs s'effectuer à partir d'une certaine position de la chaîne la plus longue : INSTR («COUCOU»,«COU») donne le résultat 1 et INSTR (3,«COUCOU»,«COU») donne le résultat 4.

En fait, si les concepteurs de ce Basic ont visiblement tenu à ne pas trop s'éloigner du Basic Microsoft qui les a beaucoup inspirés, ils ont enrichi la panoplie habituelle de plusieurs fonctions assez utiles : LOWER$ transforme toutes les majuscules d'une chaîne en minuscules, UPPER$ fait l'inverse ; SYMBOL permet de redéfinir jusqu'à 256 caractères et SYMBOLAFTER fixe le premier code ASCII dont le caractère correspondant est redéfinissable. Par défaut, cette valeur est 240 (16 caractères redéfinissables) mais l'exécution d'un SYMBOLAFTER 0 (ce qui signifie que même l'alphabet peut être redéfini) libère 1920 octets.

Le contrôle des structures s'effectue au moyen d'instructions classiques : GOTO (qui ne peut être suivi d'une expression algébrique), IF... THEN... ELSE, WHILE...WEND.

Le traitement des erreurs fait appel

LE BASIC DE L'AMSTRAD

aux non moins classiques ON ERROR GOTO ou GOSUB, ERR, ERL et RESUME. ERROR provoque l'erreur dont le numéro est spécifié, ce qui permet notamment de créer ses propres messages d'erreur. ON BREAK GOSUB (que l'on peut annuler par ON BREAK STOP) permet, comme on le devine, un branchement à un sous-programme dès lors que l'utilisateur a tenté d'interrompre le programme. Ces deux dernières instructions, associées à la possibilité de protéger un programme durant la sauvegarde et celle de le faire démarrer automatiquement après chargement, donneront un peu de fil à retordre aux « pirates ».

La façon la plus simple d'obtenir un son avec l'Amstrad est de taper PRINT CHR$(7). C'est un peu faible, me direz-vous. Oui, mais ce n'est que l'octet qui cache les Ko (ou quelque chose comme ça). Les possibilités offertes dans ce domaine sont immenses. Attention : tout ne va pas sans une certaine complexité, mais il y a vraiment de quoi faire. Le CPC 464 dispose de trois voies sonores simultanées pour la musique (ou le bruit quand on débute...). A chaque voie est associé un « réservoir » de sons et la machine joue jusqu'à ce que ce réservoir soit vide. Ces sons ont été préalablement définis par SOUND suivi de rien moins que huit paramètres qui déterminent l'état de la voie, la période, la durée, le volume du son, son enveloppe, etc. N'étant pas musicien et ne connaissant rien aux synthétiseurs, je me suis contenté de quelques modestes essais, pas toujours réussis, mais étonnants ! Les instructions ENV et ENT (suivies de 16 paramètres) permettent de travailler avec une grande finesse les caractéristiques physiques des sons synthétisés.

Par ailleurs, SQ(x) donne toutes les informations nécessaires sur l'état de la voie x.

Un petit tour sur les canaux

Toutes les entrées-sorties de l'Amstrad transitent par des canaux : les canaux 0 à 7 correspondent à autant de fenêtres-écrans dont l'emplacement et les dimensions sont au besoin redéfinis par WINDOW. Le canal 8 permet de piloter l'imprimante, et le canal 9 est dédié à la lecture et l'écriture de données sur le magnétophone.

Mais revenons à ces fenêtres. Pour choisir celle dans laquelle on veut écrire, on fait suivre l'instruction d'affichage par # numéro de canal : PRINT # 0, «BONJOUR» se passe de commentaires. Notons que le canal 0 est choisi par

Liste des mots-clés du Basic Amstrad

*	PRINT	GOSUB
+	PRINT USING	GOTO
–	RAD	HEX$
/	RANDOMIZE	HIMEM
↑	READ	IF...THEN...ELSE
ABS	RELEASE	INK
AFTER	REM	INKEY
ASC	REMAIN	INKEY$
ATN	RENUM	INP
AUTO	RESTORE	INPUT
BIN$	RESUME	INSTR
BORDER	RETURN	INT
CALL	RIGHT$	JOY
CAT	RND	KEY
CHAIN	ROUND	KEYDEF
CHAIN MERGE	RUN	LEFT$
CHR$	SAVE	LEN
CINT	SGN	LINE INPUT
CLEAR	SIN	LIST
CLG	SOUND	LOAD
CLOSEIN	SPACE$	LOCATE
CLOSEOUT	SPEEINK	LOG
CLS	SPEED KEY	LOG10
CONT	SPEED WRITE	LOWER$
COS	SQ	MAX
CPS	SQR	MEMORY
CREAL	STOP	MERGE
DATA	STR$	MID$
DEFFN	STRING$	MIN
DEFINT	SYMBOL	MOD
DEFREAL	SYMBOLAFTER	MODE
DEFSTR	TAG	MOVE
DEG	TAGOFF	MOVER
DELETE	TAN	NEW
DI	TEST	ON BREAK GOSUB
DIM	TESTR	ON BREAK STOP
DRAW	TIME	ON ERROR GOSUB
DRAWR	TROFF	ON ERROR GOTO
EDIT	TRON	ON SQ GOSUB
EI	UNT	ON...GOSUB
END	UPPER$	ON...GOTO
ENT	VAL	OPENIN
ENV	VPOS	OPENOUT
EOF	WAIT	ORIGIN
ERASE	WHILE...WEND	OUT
ERL	WIDTH	PAPER
ERR	WINDOW	PEEK
ERROR	WINDOW SWAP	PEN
EVERY	WRITE	PI
EXP	XPOS	PLOT
FIX	YPOS	PLOTR
FOR...STEP...NEXT	ZONE	POKE
FRE		POS

défaut et que PRINT peut être suivi de USING *format*, et que WINDOW SWAP, très spectaculaire, échange le contenu de deux fenêtres.

L'affichage du texte et du graphique utilise un procédé de plus en plus répandu (c'est celui adopté notamment par le Macintosh, le Sinclair QL et l'Hector). Il n'existe pas de véritable mode texte, les caractères étant affichés dans l'écran graphique. Il est donc possible de mélanger à volonté texte et dessins, mais cette souplesse se paie par une certaine lenteur quand il s'agit d'afficher du texte ; cela est sensible lors de l'exécution de l'ordre LIST.

Les possibilités graphiques du CPC 464 sont proprement époustouflantes pour une machine de cette catégorie. Nous ne ferons qu'en évoquer certaines.

Il existe trois modes sélectionnés par l'instruction MODE suivie de 0, 1 ou 2. Le mode 0 correspond à une définition graphique de 160×200 points (ou 25 lignes de 20 caractères) en seize couleurs. Le mode 1 est celui de la machine à la mise sous tension ; il permet l'affichage de 25 lignes de 40 caractères ou 320×200 pixels en quatre couleurs. Le mode 3 offre 640×200 pixels ou 25 lignes de 80 colonnes. Les caractères sont alors minuscules et s'ils restent parfaitement lisibles sur le moniteur couleur, c'est tout de même assez fatigant.

Notons que les trois modes graphiques utilisent la même zone-mémoire (16000 octets) pour stocker ce qui apparaît à l'écran.

Dans tous les modes, il n'existe aucune contrainte de proximité pour les couleurs : chaque point peut être allumé individuellement dans la couleur désirée. Les couleurs disponibles (2, 4 ou 16 suivant le mode) sont sélectionnées dans une palette de 27 par l'instruction INK qui permet de surcroît le clignotement (à vitesse variable) de certaines couleurs.

Impossible de passer ici tout en revue : l'Amstrad met à la disposition de l'amateur de graphisme une très jolie panoplie d'instructions (mais CIRCLE fait défaut).

Par ici les entrées

En plus d'INPUT et INKEY$, le Basic de l'Amstrad apporte quelques facilités au niveau des entrées :

LINE INPUT # numéro de canal, variables permet la lecture des données provenant d'un canal quelconque (même s'il s'agit du magnéto-cassette), INPUT n'étant capable de lire que le clavier.

KEY permet d'associer à certaines touches du clavier (notamment le pavé numérique) une séquence de frappe ; après avoir tapé : KEY 128 = "TRON" + "RUN" + CHR$(13) une pression sur la touche 128 (c'est le zéro du pavé numérique) fera passer en mode TRACE et lancera le programme (CHR$(13) correspond à l'enfoncement de la touche ENTER). Cette méthode est analogue au mode RESERVE des poquettes de chez Sharp.

KEYDEF attribue à une touche du clavier un nouveau code ; on peut ainsi, par exemple, transformer le clavier QWERTY en clavier AZERTY (il faut, bien sûr, coller de nouvelles étiquettes sur les touches !). Relevons, pour terminer, SPEEDKEY qui redéfinit l'attente initiale et la vitesse de répétition des touches, et JOY qui teste la position de deux manettes de jeu et l'état de leurs boutons de tir.

Comme très souvent en Basic (sauf sur certaines machines de poche), les fonctions mathématiques inverses sont délaissées et seule ATN (Arctangente) n'a pas été oubliée.

En trigonométrie, l'Amstrad peut travailler soit en degrés, soit en radians, la communication d'un mode à l'autre s'effectue par RAD ou DEG. Enfin, la pseudo-variable PI est présente et retourne la valeur 3,14159265.

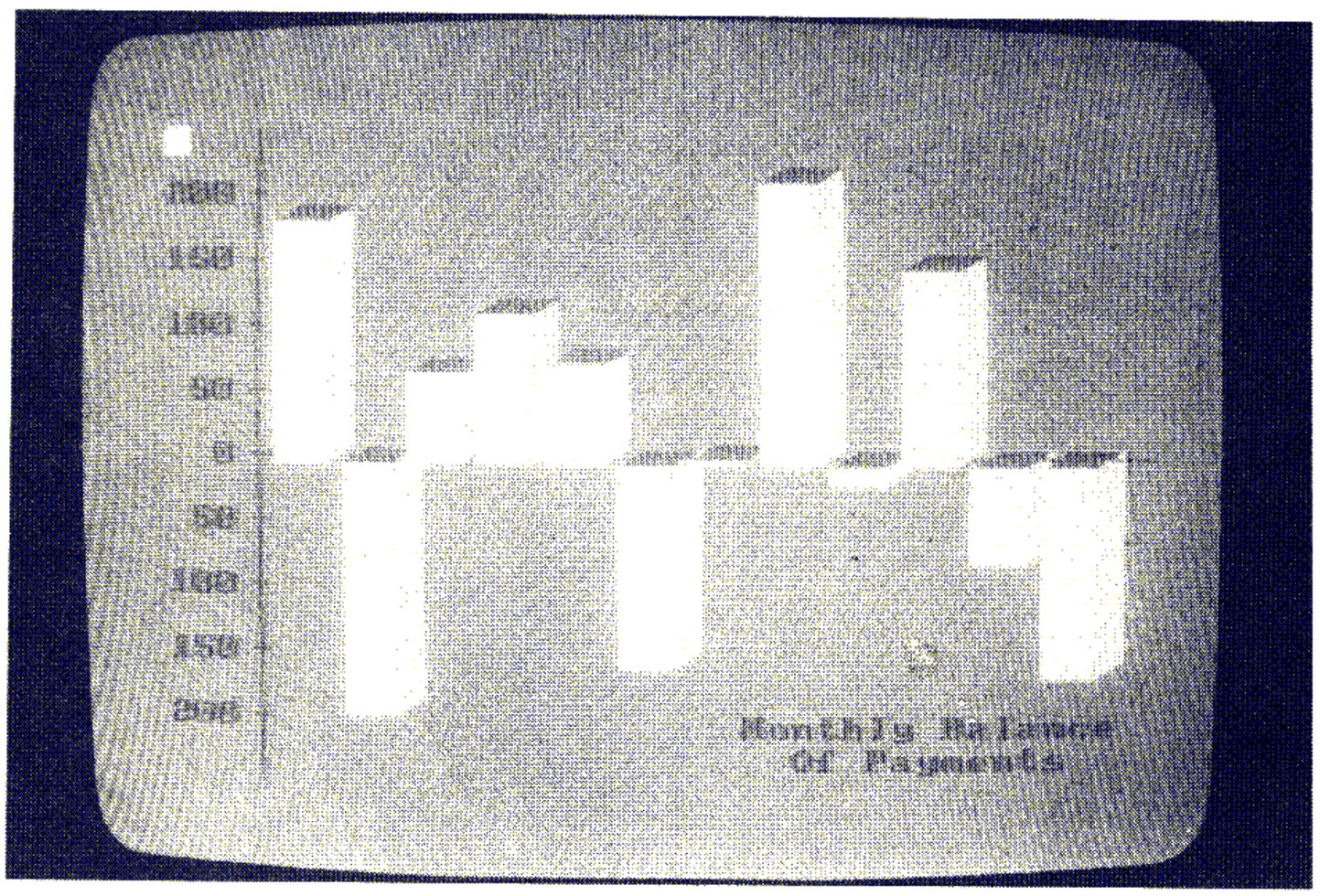
L'élévation à la puissance est obtenue par le signe ↑ et MOD donne le reste d'une division entière. Toujours à propos de la division, il faut distinguer le signe / qui effectue la division dans l'ensemble des réels et qui retourne un réel et \ qui divise un réel par un entier, le résultat étant un entier. MAX et MIN retournent la plus grande et la plus petite valeur d'un groupe d'expressions. Ainsi, MAX (1,2,3,8,12,0,4) donnera 12.

La fonction RND fournit des nombres « aléatoires » compris entre 0 et 1, et RANDOMIZE permet d'éviter la répétition des mêmes séquences. BIN$ et HEX$ effectuent respectivement les conversions décimal-binaire et décimal-hexadécimal.

La fonction ROUND arrondit une expression à partir de la position spécifiée avant ou après la virgule. Quant à UNT, il convertit un entier non signé en un entier compris entre −32767 et +32768 ; ainsi UNT(&FFFF) donne −1. Enfin DEFFN autorise la définition de nouvelles fonctions.

Le magnétophone à cassette intégré fonctionne avec des cassettes standard. La vitesse de transfert est soit de 1000 bauds, soit de 2000 bauds, le passage de l'une à l'autre s'effectuant par l'instruc-

La haute résolution au service des histogrammes.



LE BASIC DE L'AMSTRAD

tion SPEED WRITE 0 ou 1. L'ordinateur détecte automatiquement, au début de chaque lecteur, la vitesse à laquelle l'enregistrement a été effectué. Les instructions LOAD et SAVE offre plusieurs options :

- SAVE «X»,A sauvegarde le programme en mémoire sous forme ASCII ;
- SAVE «X»,P le sauvegarde en le protégeant (après lecture, impossible de le lister) ;
- SAVE «X»,B *adresse de départ, longueur* permet de sauvegarder un secteur complet de la mémoire sous forme binaire. Il est ainsi possible de mémoriser sur cassette l'image qui apparaît à l'écran.

Le catalogue de la cassette

CHAIN charge un programme en remplacement du précédent alors que CHAIN MERGE l'ajoute en mémoire en le renumérotant éventuellement pour éviter les chevauchements. Ces deux dernières instructions conservent l'état des variables. MERGE tout seul charge en « aveugle » un programme, la cohabitation avec l'ancien nécessite que le programmeur ait pris ses précautions.

Les instructions OPENIN, CLOSEIN, OPENOUT et CLOSEOUT ouvrent et ferment une zone de la mémoire vive (2 Ko) qui constituera un intermédiaire entre l'ordinateur et le cassettophone durant la sauvegarde ou le chargement de variables.

Signalons, pour finir, la commande CAT, qui dresse la liste des fichiers contenus par la cassette.

L'Amstrad est doté d'une horloge interne (non sauvegardée par batterie). La pseudo-variable TIME donne le temps écoulé (en 300e de secondes) depuis la mise sous tension. Cependant, toute lecture ou écriture cassette interrompt provisoirement le décompte du temps.

Les interruptions peuvent être gérées directement à partir du Basic : l'instruction AFTER...GOSUB provoque un branchement à un sous-programme après un laps de temps spécifié. Quatre comptes à rebours peuvent ainsi s'effectuer simultanément. L'exécution du programme reprend à l'endroit où elle a été suspendue dès qu'un RETURN est rencontré.

L'instruction EVERY...GOSUB déclenche l'exécution du sous-programme désigné périodiquement avec la fréquence spécifiée.

REMAIN *(n° de chronomètre)* teste l'état d'un des quatre chronomètres puis l'arrête (s'il est déjà arrêté, la fonction retourne zéro).

Comme tout Basic qui se respecte, celui de l'Amstrad donne accès aux ressources du système : PEEK (lecture d'un octet), POKE (écriture dans un octet) et CALL (appel d'un programme en langage-machine avec passage de plusieurs paramètres).

La pseudo-variable HIMEM donne le dernier octet auquel a accès le Basic (HIMEM est initialisée à 43903 et peut être modifiée par l'utilisateur).

Fiche technique de l'Amstrad CPC 464

Constructeur : *Amstrad*
Prix public : *2990 F (écran noir et blanc)*
4490 F (écran couleur)
Processeur : *Z 80 A*
Mémoire vive disponible : *43 Ko + 16 Ko graphique*
Mémoire morte : *16 Ko*
Langage : *Basic résident (158 mots-clés) et, en option, Pascal, Logo (disquette)*
Affichage : *25 lignes de 20 caractères.*
25 lignes de 40 caractères.
25 lignes de 80 caractères.
Résolution graphique : *jusqu'à 640 x 200 points*
Vitesse de transmission cassette : *1000 ou 2000 bauds.*
Précision : *9 chiffres significatifs*

Le Basic de l'Amstrad, dont nous sommes loin d'avoir fait le tour, est l'un des plus complets qu'il nous ait été donné d'essayer. Il soutient sans aucun problème la comparaison avec son concurrent direct, le Basic MSX. Si le clan MSX devait sortir gagnant de la compétition qui se prépare, ce serait sans doute grâce à une plus grande bibliothèque de logiciels, une importante variété de périphériques et surtout, grâce à la puissance commerciale de l'ensemble des constructeurs ayant adopté ce standard. Mais rien n'est joué pour l'instant, car l'Amstrad est une machine sensationnelle.

Thierry LÉVY-ABÉGNOLI

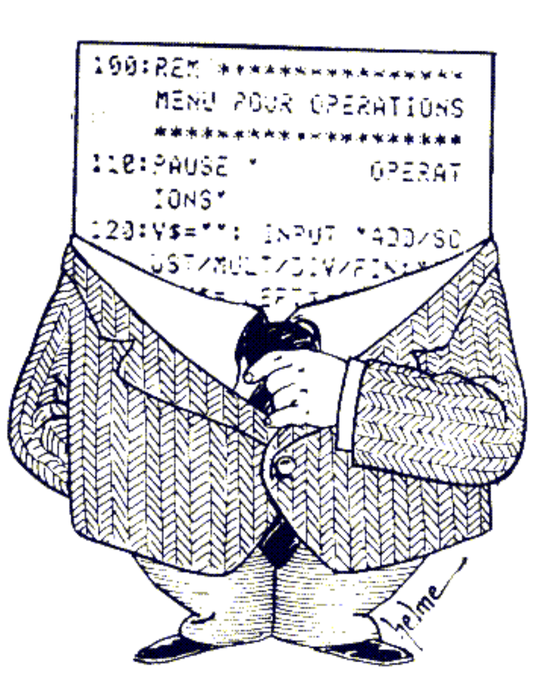
AU MENU : UN PROGRAMME A LA CARTE

POUR bien présenter un programme, rien de tel qu'un menu : tout y est clair et on choisit ce qui nous intéresse. Mais, sur la seule ligne d'affichage des ordinateurs de poche, il faut savoir abréger.

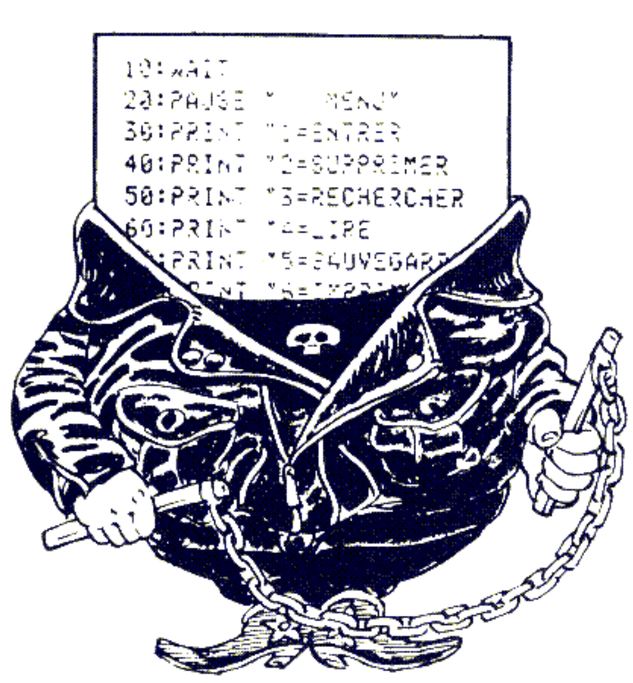
On peut se souvenir sans trop de nostalgie de ces premiers ordinateurs de poche dont la taille mémoire était un problème : il fallait économiser le moindre octet, souvent au détriment de la présentation et de la facilité d'utilisation du programme.

Aujourd'hui, leurs capacités de mémoire ont nettement augmenté, mais le style de programmation est encore resté le même : difficile d'utiliser un programme sans avoir lu auparavant son mode d'emploi. Alors, pourquoi ne pas profiter de la place souvent disponible pour concevoir des « menus » ? D'autant que depuis longtemps ils sont employés avec succès sur les ordinateurs de table. Le plus souvent, le programme se suffit ainsi à lui-même : peu ou pas de notice extérieure.

Sur les ordinateurs de table qui disposent de l'écran d'un téléviseur, le menu peut être constitué à partir d'une suite d'instructions PRINT avant un INPUT (comme le montre le programme 1). Mais avec la seule ligne d'affichage des poquettes, l'utilisateur n'a plus que celle de l'INPUT sous les yeux lors du choix. Il est



BONNE PRÉSENTATION



MAUVAISE PRÉSENTATION

Programme 1
Un exemple de menu à éviter pour les ordinateurs de poche

```
10:WAIT
20:PAUSE "    MENU"
30:PRINT "1=ENTRER"
40:PRINT "2=SUPPRIMER"
50:PRINT "3=RECHERCHER"
60:PRINT "4=LIRE"
70:PRINT "5=SAUVEGARDER
80:PRINT "6=IMPRIMER"
90:PRINT "7=TERMINE"
100:INPUT "1/2/3/4/5/6/7
    : ";C
110:IF C<1 OR C>7 GOTO 1
    00
120:IF C=7 PAUSE "   AU
    REVOIR": END
130:ON C GOTO "ENTR","SU
    PPR","RECH","LIRE","
    SAUVE","IMPRIM"
```

donc préférable d'employer des abréviations comme le montre le programme 2.

Programme 2
Un menu conçu pour les poquettes

```
100:REM ********************
    MENU POUR OPERATIONS
    **********************
110:PAUSE "       OPERAT
    IONS"
120:V$="": INPUT "ADD/SO
    UST/MULT/DIV/FIN:";V
    $:V$= LEFT$ (V$,1)
130:IF V$="A" GOTO "ADD"
140:IF V$="S" GOTO "SOUS
150:IF V$="M" GOTO "MULT
160:IF V$="D" GOTO "DIV
170:IF V$="F" PAUSE "
       AU REVOIR": END
180:GOTO 120
```

Cependant, si toutes les abréviations de l'INPUT ne peuvent pas tenir sur la ligne d'affichage, on procède alors par la méthode *help* : l'utilisateur demande une aide, sous la forme d'une explication, dès qu'il en a besoin.

Avec un SOS, par exemple, toutes les abréviations seront données en clair (programme 3), si on le désire. Sinon, le programme passe directement au traitement.

Jouez sur tous les niveaux

Vous remarquerez que deux CALL spécifiques au PC-1251 permettent de jouer avec l'affichage à cristaux liquides (lignes 600 et 620).

On peut aussi procéder par *arborescence* pour résoudre le problème des initiales multiples. Par exemple, un S pour Supprimer et un S pour Sauver peuvent apparaître à deux niveaux différents. Les commandes de niveau inférieur doivent être moins fréquemment employées, puisqu'elles sont quand même moins faciles à appeler : pour les sélectionner, il faut avoir donné autant de réponses qu'il y a de niveaux au-dessus d'elles. Le programme 4 donne l'exemple d'un menu avec deux niveaux d'arborescence. Ainsi, le S de la sauvegarde ne pourra être sélectionné que si, au premier niveau, le choix s'est porté sur PERI.

Pour encore plus de commodité, il faudrait pouvoir retourner au premier niveau après avoir sélectionné le niveau inférieur. Il suffit alors d'ajouter une option RET (comme retour) en ligne 90 et de rentrer la nouvelle ligne 125 IF K$ = "R" GOTO 20.

Dans le PC-1251, comme dans d'autres ordinateurs de poche, il est possible de stocker simultanément plusieurs programmes que l'on sélectionne par DEF. D'où l'idée de toujours avoir un menu en tête récapitulant tout ce que contient la machine (programme 5).

Bien sûr, tous ces menus demandent un effort supplémentaire lors de la programmation. Mais ensuite quelle commodité d'utilisation ! Et finies les interminables questions sur le fonctionnement de ce fameux programme que vous n'aviez plus utilisé depuis de si longs mois...

Programme 3
Un menu avec SOS : des explications peuvent être demandées au besoin

```
10:REM ********************
   MENU PRG FACTURATION
   **********************
20:WAIT 0: PRINT "C/M/F
   /V/SOS/TERMINE:":
   GOSUB 600
30:IF B=19 PAUSE "C= CA
   TALOGUE": PAUSE "M=
   MODIFICATION": PAUSE
   "F= FACTURE": PAUSE
   "V= VOIR": GOTO 20
40:IF B=20 PAUSE "
    AU REVOIR": END
50:IF B=22 GOTO "VOIR"
60:IF B=6 GOTO "FACTUR
70:IF B=3 GOTO "CATALOG
80:IF B=13 GOTO "MODIFI
90:GOTO 20
100:REM ******************
        ICI SE TROUVE
         LA SUITE DU
          PROGRAMME
590:REM ******************
   S/P INKEY$ & LCD"ON"
   **********************
600:A$="": CALL &11E0
610:A$= INKEY$
620:IF A$ CALL &11E5:
   BEEP 1:B= ASC A$-64:
   RETURN
630:GOTO 610
640:REM ******************
```

Programme 4
Un menu à deux niveaux

```
10:REM ********************
   MENU POUR AGENDA TEL
   **********************
20:WAIT 0: PRINT "ENTR/
   SUPPR/RECH/PERI/FIN"
   : GOSUB 700
30:IF K$="E" GOTO "ENT"
40:IF K$="S" GOTO "SUP"
50:IF K$="R" GOTO "REC"
60:IF K$="P" GOTO 90
70:IF K$="F" PAUSE "
      AU REVOIR": END
80:GOTO 20
90:WAIT 0: PRINT "LIRE/
   SAUVEGARDER/IMPRIME"
   : GOSUB 700
100:IF K$="L" GOTO "LIR"
110:IF K$="S" GOTO "SAU"
120:IF K$="I" GOTO "IMP"
130:GOTO 90
140:REM ******************
        ICI SE TROUVE
         LA SUITE DU
          PROGRAMME
690:REM ******************
   S/P INKEY$ & LCD"ON"
   **********************
700:K$="": CALL &11E0
710:K$= INKEY$
720:IF K$ CALL &11E5:
   BEEP 1: WAIT :
   RETURN
730:GOTO 710
740:REM ******************
```

Programme 5
Un menu récapitulatif

```
10:REM ********************
    MENU POUR "MERGE"
   **********************
20:"M" WAIT : PAUSE "
         MENU"
30:PRINT "DEF B=BIDULE
40:PRINT "DEF S=STCHIF
50:PRINT "DEF C=CHOSE
60:PAUSE "      AU REVO
   IR"
70:END
80:REM ******************
```

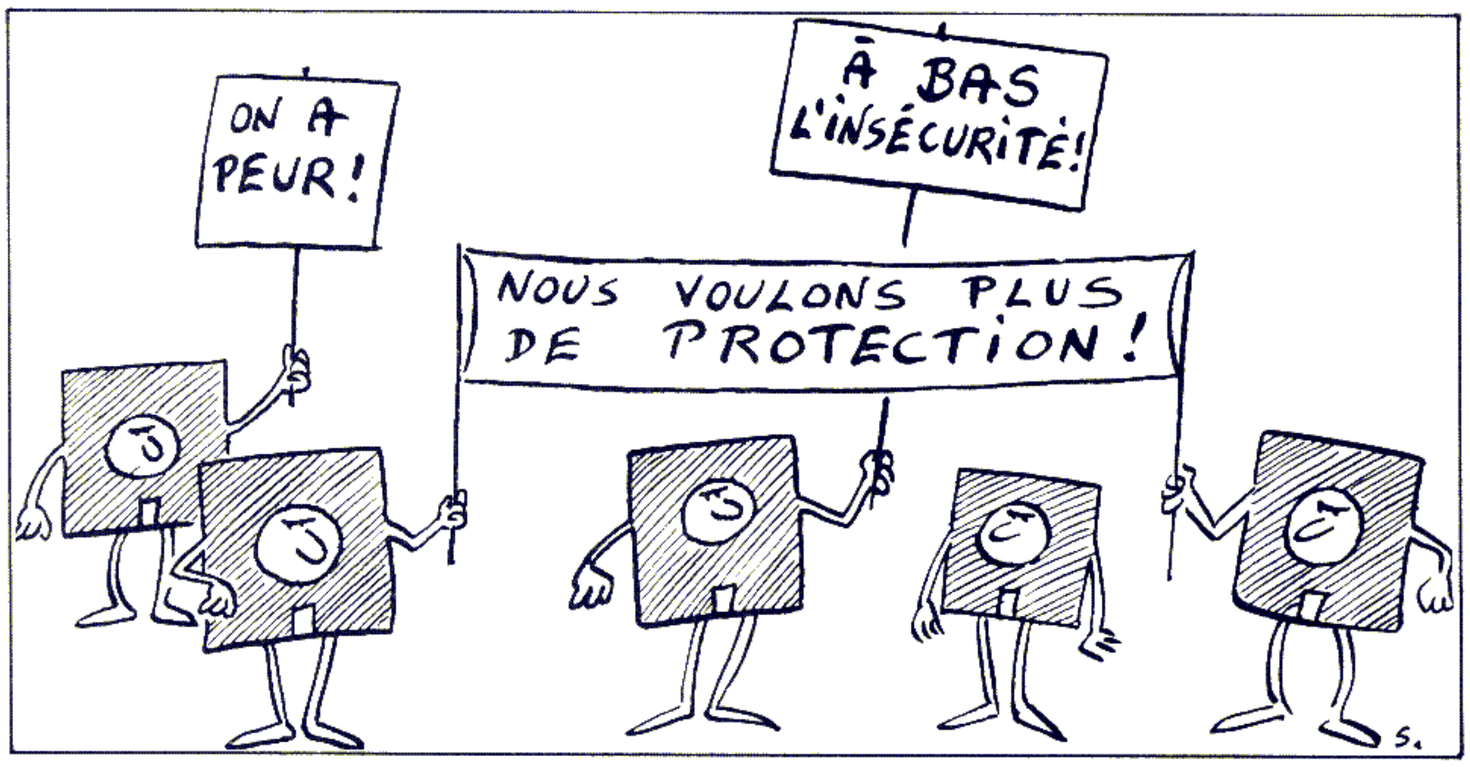
Michel DUPONT

FICHIERS DU SIXIÈME TYPE

EN tant qu'utilisateur du lecteur de disquette Commodore, vous aimeriez protéger vos fichiers contre toute destruction intempestive, sans pour autant mettre en place une étiquette sur l'encoche de vos disquettes... Une solution existe : la protection logicielle. Nous l'avons rencontrée !

Laissée en suspens dans *LIST 4*, l'étude de fichiers d'un type quelque peu spécial est abordée ici. Pourquoi spécial ? Tout d'abord parce que rien, dans le manuel d'utilisation des disquettes, n'est dit à leur sujet, et ensuite parce que les propriétés de ce nouveau genre de fichiers sont vraiment très particulières.

Avant de se lancer dans cette étude, il faut rappeler les bases de l'organisation des disquettes sur Commodore. Cinq types de fichiers différents peuvent coexister normalement sur une disquette. Quatre types sont majeurs : les fichiers séquentiels (SEQ), programmes (PRG), utilisateurs (USR), relatifs (REL).



L'octet indicateur

Le cinquième est un peu particulier puisque les fichiers de ce type n'apparaissent plus lors de l'affichage du catalogue de la disquette, et ne peuvent plus être rechargés en mémoire : il s'agit des fichiers détruits (DEL).

Le SED (Système d'Exploitation des Disquettes) reconnaît la catégorie à laquelle appartient chaque fichier grâce à une indication *ad hoc*, qui figure à côté du titre du fichier sur la piste 18 de la disquette (cette piste est celle du catalogue). Le tableau 1 établit qu'un seul octet est suffisant pour contenir cette

Tableau 1
L'octet souligné indique le type de fichier

```
PISTE 18 SECTEUR 1
  0 > 00 FF 82 11 00 49 4E 56 45 52 53 20 43 41 52 41  : .....INVERS CARA
 16 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00 02  : ................
 32 > 00 00 83 11 01 54 52 41 4E 53 46 45 52 54 20 47  : .....TRANSFERT G
 48 > 52 41 50 48 A0 00 00 00 00 00 00 00 00 00 00 03  : RAPH............
 64 > 00 00 84 11 03 43 4F 50 49 45 20 45 43 52 41 4E  : .....COPIE ECRAN
 80 > 20 48 2E 52 2E 00 00 00 00 00 00 00 00 00 00 03  :  H.R............
 96 > 00 00 00 11 05 54 41 42 4C 45 53 20 44 45 20 4D  : .....TABLES DE M
112 > 55 4C 31 A0 A0 00 00 00 00 00 00 00 00 00 00 08  : UL1.............
128 > 00 00 81 11 09 4D 55 4C 54 32 A0 A0 A0 A0 A0 A0  : .....MULT2......
144 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00 00  : ................
160 > 00 00 82 13 00 47 52 44 53 20 43 41 52 41 A0 A0  : .....GROS CARA..
176 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00 01  : ................
192 > 00 00 C2 13 02 45 4C 45 4D 45 4E 54 20 54 45 53  : .....ELEMENT TES
208 > 54 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00 02  : T...............
224 > 00 00 82 13 01 53 55 42 4C 4F 47 20 24 37 33 30  : .....SUBLOG $730
240 > 30 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00 09  : 0...............
```

FICHIERS DU SIXIÈME TYPE

indication du type, l'octet souligné. Il vaut 81 pour un fichier séquentiel, 82 pour un fichier programme, 83 pour un fichier utilisateur, 84 pour un fichier relatif et 00 pour un fichier détruit. Et quand on demande l'affichage du catalogue, le SED analyse l'octet indicateur, et rend son verdict sous la forme de trois lettres qui apparaissent à l'écran : SEQ, PRG, USR, REL, ou sous la forme d'une absence d'affichage si le fichier est de type DEL.

Rigoureusement indestructible

Le tableau 2 établit une correspondance entre les valeurs hexadécimales possibles (qui figurent dans le tableau 1) et leur écriture binaire. Ceci pour chaque type de fichier.

La lecture de ce tableau indique que les bits 0 à 2 sont suffisants pour définir le type du fichier. Par ailleurs, les bits 3 à 6 semblent devoir rester toujours à 0. Quant au bit 7, il a une fonction spéciale :

- s'il est à 0, il indique que le fichier est ouvert ;
- s'il est à 1, il indique que le fichier a été refermé.

Vous pourriez découvrir comme nous, par le plus grand hasard, que le bit 6 est doué de propriétés tout à fait remarquables.

En effet, le simple fait de positionner ce bit à 1 provoque la création d'un nouveau type de fichier... Demeurant visi-

Explications du programme

Lignes 190 à 210 : initialisations diverses, les cinq types connus de fichiers sont stockés en tableau.

Lignes 230 à 250 : le canal de commande est ouvert comme d'habitude sous le numéro logique 15, tandis que le canal de données portera le numéro 2.

Lignes 270 à 320 : pour qu'il n'y ait aucun doute sur la disquette à traiter, le programme lit et affiche au préalable son titre et son numéro d'identification.

Lignes 340 à 530 : le titre du fichier à protéger est demandé à l'utilisateur (le programme se termine en cas de non-réponse).

Quand le titre est défini, le programme le recherche sur la piste 18 qui contient le catalogue de la disquette.

L'utilisateur a la possibilité d'abréger le titre du fichier en ne tapant que les premiers caractères. En ce cas, c'est le premier titre correspondant à ces caractères qui sera proposé pour le traitement.

Lignes 550 à 590 : le titre trouvé est affiché, et l'octet de type est lu à son tour. La ligne 590 indique le cas échéant si le bit 6 de cet octet est déjà à 1, auquel cas le traitement pourra être une libération du fichier.

Lignes 610 à 670 : si tel n'est pas le cas, le type actuel du fichier est affiché, et l'utilisateur doit confirmer sa demande de traitement. La ligne 660 permet de positionner le bit à 1 pour établir la protection.

Lignes 690 à 730 : si la protection est déjà en place, l'utilisateur peut demander la libération du fichier, ce qui s'effectue en ligne 730 par remise à 0 du bit 6.

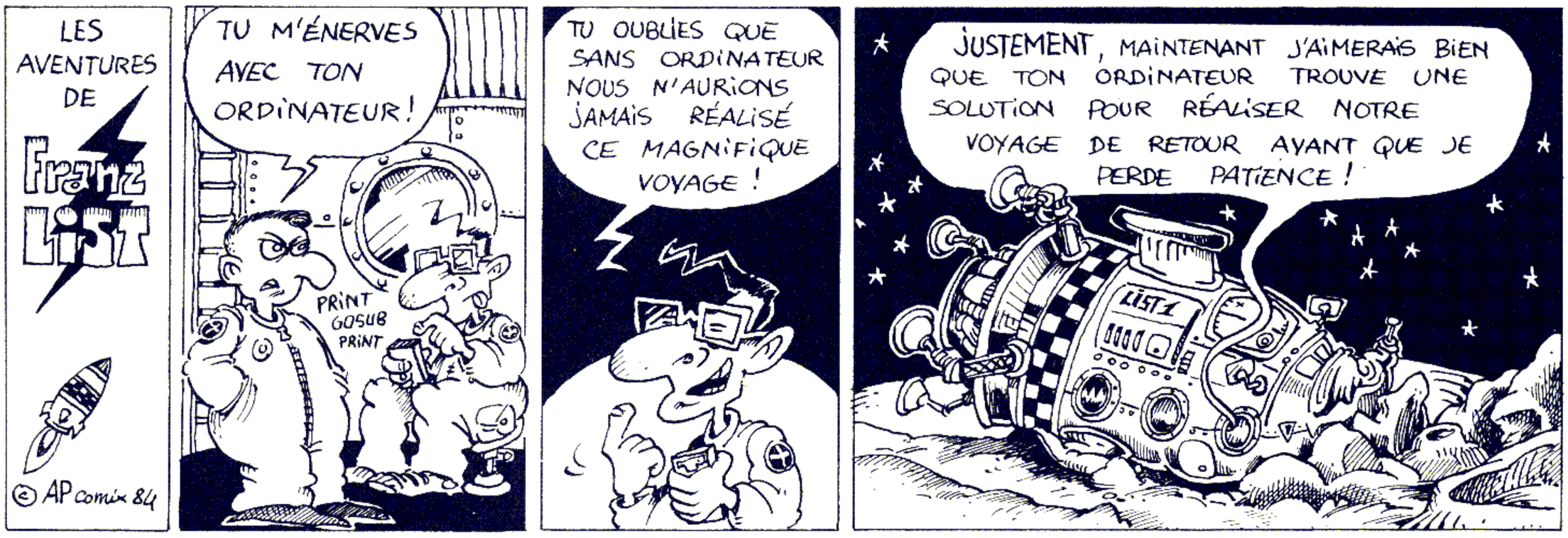
Lignes 750 à 800 : dans les deux cas, l'octet modifié est remis en bonne place sur la piste 18, tandis qu'un commentaire signale la fin du traitement.

Ligne 860 : le retour à la ligne 340 permet de traiter "dans la foulée" un autre fichier.

Lignes 880 à 910 : classique et tout à fait indispensable, voici le sous-programme de traitement des problèmes éventuels décelés sur la disquette au cours du travail.

Tableau 2
Représentation sur huit bits des types de fichier

Type de fichier	Représentation sur huit bits								Code hexadécimal
	7	6	5	4	3	2	1	0	
DEL	0	0	0	0	0	0	0	0	00
SEQ	1	0	0	0	0	0	0	1	81
PRG	1	0	0	0	0	0	1	0	82
USR	1	0	0	0	0	0	1	1	83
REL	1	0	0	0	0	1	0	0	84



```
100 REM ***************************
110 REM * UTILITAIRE DISQUE NO.3 *
120 REM *       ANTI-SCRATCH      *
130 REM ***************************
140 :
150 PRINT"␣","␣ ANTI-SCRATCH "
160 PRINT"␣PERMET DE PROTEGER DES FICHIERS CONTRE L'EFFACEMENT,"
170 PRINT"␣OU DE SUPPRIMER CETTE PROTECTION.
180 :
190 C0$=CHR$(0)
200 DIM A$(4):FOR I=0 TO 4:READ A$(I):NEXT
210 DATA DEL,SEQ,PRG,USR,REL:REM TYPES DE FICHIERS
220 :
230 REM *************** INITIALISATIONS DISQUE ****************
240 OPEN 15,8,15,"I0":GOSUB 890
250 OPEN 2,8,2,"#":GOSUB 890
260 :
270 REM ******************** TITRE ET ID **********************
280 PRINT#15,"U1";2;0;18;0:GOSUB 890
290 PRINT#15,"B-P";2;144:GOSUB 890
300 :
310 FOR I=0 TO 19:GET#2,A$:F$=F$+A$:NEXT I
320 PRINT"␣␣ TITRE: ";LEFT$(F$,16);"␣ ID: "RIGHT$(F$,2)
330 :
340 REM ********************* PROG PPAL ************************
350 INPUT"␣TITRE DU FICHIER A TRAITER ␣␣␣␣␣";T1$
360 IF T1$="*" THEN 930
370 IF LEN(T1$)>16 THEN 350
380 :
390 P=18:S=1
400 PRINT#15,"U1";2;0;P;S:GOSUB 890
410 PRINT#15,"B-P";2;0:GOSUB 890
420 :
430 REM ++++++++++++++++ EMPLACEMENT SUIVANT ++++++++++++++++++
440 GET#2,P$:PS=ASC(P$+C0$):REM PISTE
450 GET#2,S$:SS=ASC(S$+C0$):REM SECTEUR
460 J=1:REM COMPTEUR DU NOMBRE DE FICHIERS/BLOC
470 :
480 REM +++++++++++++++++++ TITRE FICHIER +++++++++++++++++++++
490 PRINT#15,"B-P";2;J*32+5
500 T2$="":FOR I=1 TO 16
510 GET#2,A$:T2$=T2$+A$
520 NEXT I
530 IF T1$<>LEFT$(T2$,LEN(T1$)) THEN 820
540 :
550 REM ++++++++++++++++++++ TYPE FICHIER +++++++++++++++++++++
560 PRINT">>␣";T2$;"␣:";
570 PRINT#15,"B-P";2;J*32+2
580 GET#2,X$:X=ASC(X$+C0$)
590 IF X AND 64 THEN PRINT" FICHIER DEJA PROTEGE !":GOTO 690
600 :
610 REM ++++++++++++ ETABLISSEMENT DE PROTECTION ++++++++++++++
620 PRINT"FICHIER DE TYPE ␣ ";A$(X AND 15);" "
630 PRINT:INPUT"ON PROTEGE (O/N) ";R$
640 IF R$<"N" OR R$>"O" THEN 630
650 IF R$="N" THEN 860
660 X=(X OR 64):C$="PROTEGE."
670 GOTO 750
680 :
690 REM +++++++++++++ SUPPRESSION DE PROTECTION +++++++++++++++
700 PRINT:INPUT"ON LIBERE (O/N) ";R$
710 IF R$<"N" OR R$>"O" THEN 700
720 IF R$="N" THEN 860
730 X=(X AND 191):C$="LIBERE."
740 :
750 PRINT#15,"B-P";2;J*32+2:GOSUB 890
760 PRINT#2,CHR$(X);:GOSUB 890
770 PRINT#15,"B-P";2;0:GOSUB 890
780 PRINT#15,"U2";2;0;P;S:GOSUB 890
790 PRINT"␣␣>>>>> FICHIER ";C$;"<<<<<"
800 GOTO 860
810 :
820 J=J+1:IF J<9 THEN 490
830 :
840 IF PS THEN P=PS:S=SS:GOTO 400:REM SUITE ?
850 PRINT"␣␣FICHIER NON TROUVE !"
860 GOTO 340:REM SUITE
870 :
880 REM ****************** S/P ERREURS DISQUE ******************
890 INPUT#15,E1,E$,E3,E4
900 IF E1>20 THEN PRINT"ERREUR DISQUE:":PRINT E1","E$","E3","E4:STOP
910 RETURN
920 :
930 CLOSE 2:CLOSE 15
940 END
READY.
```

Anti-scratch
Programme pour Commodore 64
et lecteur de disquette
Auteur Jean-Pierre Lalevée

Tableau 3
Le catalogue

```
0 "VIC 20           " V1 2A
2     "INVERS CARA"       PRG
3     "TRANSFERT GRAPH"   USR
3     "COPIE ECRAN H.R."  REL
8     "MULT2"             SEQ
1     "GROS CARA"         PRG
2     "ELEMENT TEST"      PRG<
9     "SUBLOG $7300"      PRG
628 BLOCKS FREE.
```

ble et disponible au catalogue, un fichier de ce type se charge tout à fait normalement en mémoire (si c'est un PRG, évidemment !). Il devient alors rigoureusement indestructible sous l'effet des commandes utilisées habituellement : SCRATCH "titre" ou PRINT#15, "S0: titre" resteront catégoriquement sans effet.

En toute sécurité

Lors de l'affichage du catalogue, cette nouvelle propriété sera signalée par le SED sous la forme d'un "<" inscrit en fin de ligne (tableau 3). La remise à 0 du bit 6 sera tout à fait suffisante pour supprimer cette protection logicielle, et tout pourra rentrer dans l'ordre.

Tel est donc le but du programme que nous vous soumettons. Ce troisième utilitaire (les deux précédents sont parus dans *LIST* 3 et *LIST* 4) vous permettra de modifier — mais avec toute la sécurité requise — un seul bit à la fois sur la disquette ! Il vous suffira d'indiquer au programme le titre du fichier à protéger ; le reste se fera automatiquement.

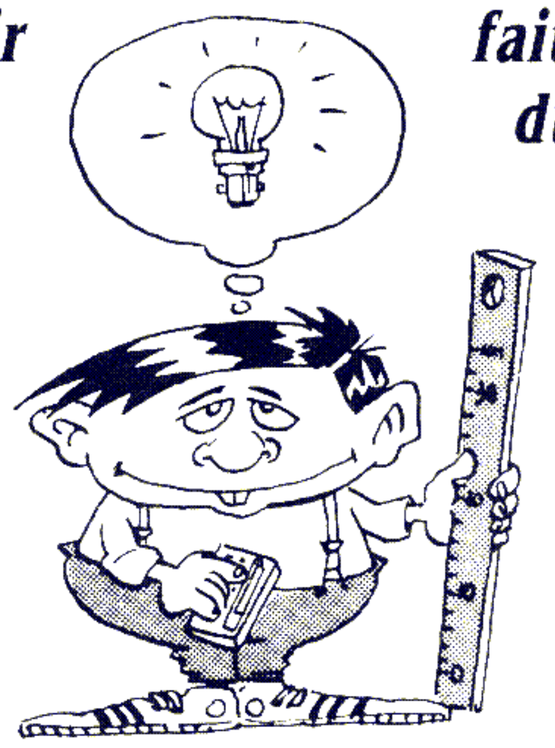
Cette bibliothèque d'utilitaires pour disquettes sera complétée une prochaine fois par l'étude d'un programme qui permettra de remettre au catalogue les fichiers détruits imprudemment ou par erreur.

On pourra s'attaquer ensuite au problème des fichiers restés ouverts en écriture, et dont le contenu était jusqu'à présent presque irrémédiablement perdu.

Jean-Pierre LALEVÉE

UN CENTRE INTROUVABLE

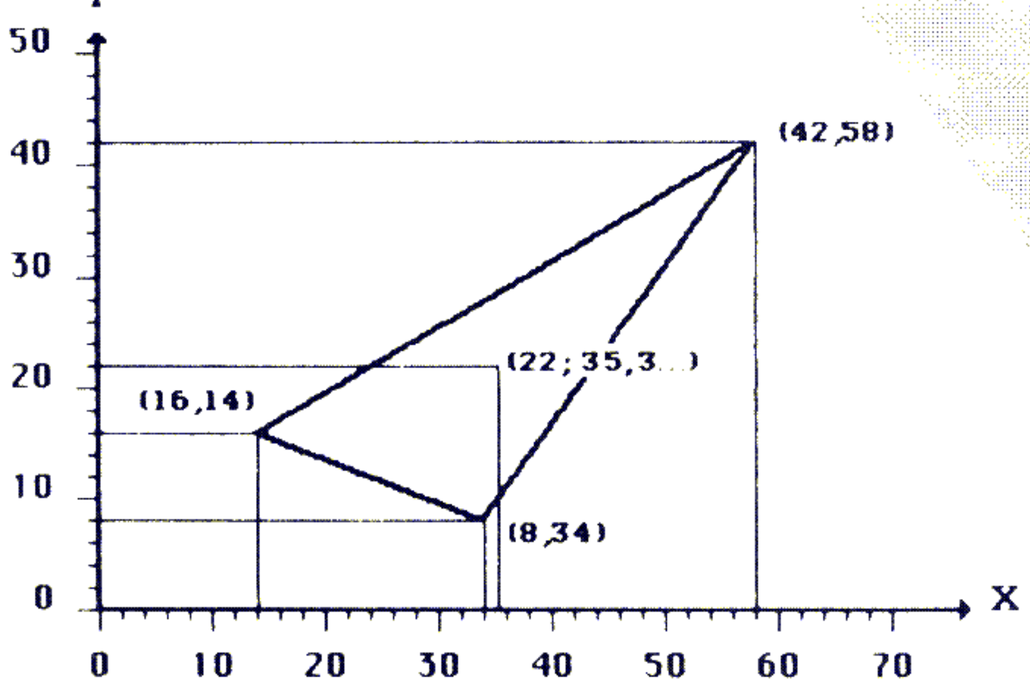
Si jongler avec la pile opérationnelle de votre HP-41 C, traquer la milliseconde perdue et rogner le moindre octet est votre pain quotidien... Ou si, à l'inverse, vous échappe parfois un peu de la subtile recherche des programmes en Notation Polonaise Inverse... Alors voici qui doit vous intéresser.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



Imaginez une aiguille dressée vers le ciel, un triangle de carton et l'exercice suivant : poser en équilibre le triangle sur la pointe de l'aiguille. Un seul point de la surface correspond à cet équilibre : le centre de gravité.

Trouver où est ce point, tel est le défi que Pierre Lison a lancé dans *LIST 4*. Connaissant les coordonnées X et Y chacun des trois sommets du triangle (voir le graphe) calculer celles du centre de gravité.

La formule de calcul est toute simple et n'a posé de problème à personne. En revanche, sa programmation a donné lieu à un festival de contorsions logicielles. On a même pu voir (ô scandale !) l'instruction STOP mise à toutes les sauces ! (Je tairai les noms...)



Centre de gravité d'un triangle dont on connaît les sommets par leurs coordonnées

Le bon algorithme

N'oublions jamais la règle première de nos optimisations : tout programme de Misez P'tit doit être rédigé *comme un sous-programme.* La totalité des données lui est fournie *avant* l'exécution, tous les résultats sont livrés ensemble en fin d'exécution. Jamais, donc, de STOP dans un programme optimisé. On doit considérer que *l'opérateur n'a*

pas à intervenir au cours de la résolution du problème.

Les coordonnés des trois sommets sont notées, dans un repère orthonormé classique (X1, Y1), (X2, Y2) et (X3, Y3). On calcule celles du centre de gravité en en effectuant simplement la moyenne : (X1 + X2 + X3)/3 et (Y1 + Y2 + Y3)/3.

La HP-41 C peut évidemment calculer cette moyenne avec ses fonctions statistiques : Σ + et MEAN. Outre le fait que ces instructions sont assez lentes, on ne les emploiera pas car elles affectent 5 registres de la mémoire autres que la pile opérationnelle. Par principe, donc, MEAN est exclue.

La solution est plus directe encore. Si trois nombres sont dans les registres X, Y et Z de la pile, leur moyenne est obtenue par : + + 3/, tout simplement.

Mais ici, pour le calcul du centre de gravité, on a six chiffres et non pas trois (trois fois X et Y). Comment faire ? Pas question, comme on l'a vu, de mettre de STOP dans la routine (cela donnerait + + STOP + + 3 STO/Z / en introduisant d'abord X1, X2 et X3 puis dans un second temps Y1, Y2 et Y3). Alors, on doit changer le mode d'introduction des coordonnées : ne séparons pas X de Y.

MOD sans MOD

Les autres calculatrices en Notation Polonaise Inverse commenceraient-elles à relever les défis tandis que les adeptes de la HP-41 se reposent sur leurs lauriers ?

Après une très sérieuse cure d'optimisation, le problème de la réalisation de la fonction modulo sans MOD, cas des « petites » machines, est revenu grâce à Bruno Piguet (Nantes).

De 19 octets et 2 mémoires, l'auteur est parvenu à seulement 10 octets. Voici son record. Qui dit mieux ?

```
ENTER
ENTER
R↑
ENTER
R↑
/
INT
R↑
X
−
```

L'algorithme employé est toujours A-B*INT(B/A) qui calcule — par définition de modulo — A MODULO B. Introduire A puis B, le résultat se trouve en X après l'exécution.

Deux résultats en un seul coup

Il suffit de programmer + + 3/ pour calculer d'un coup la moyenne des trois sommets exprimée sous une forme décimale (X,Y) soit (X1+X2+X3,Y1+Y2+Y3). Stop ! Bien sûr, cela ne fonctionne correctement qu'avec des coordonnées X et Y entières (on doit pouvoir les exprimer sous la forme X, Y) et il ne faudrait pas que la partie décimale du couple X, Y déborde sur les X à cause de retenues dans l'addition de Y1 + Y2 + Y3.

Réfléchissons. D'une part, tout nombre à virgule de la HP-41 C peut être exprimé sous la forme d'un entier à diviser par une puissance de dix (3,1 vaut 31/10). Rien n'empêche donc de traiter le cas des entiers, celui des décimaux n'en étant qu'une version particulière (tout sens mathématique perdu !). On divisera en fin de traitement les deux coordonnées du centre de gravité par la puissance de dix qui aura été nécessaire à leur expression entière.

Quant au débordement des Y sur la partie entière du nombre X,Y, cela ne peut advenir. En effet, cela se conçoit même intuitivement : si le plus grand des trois Y1, Y2 ou Y3 « tient » sous sa forme décimale (par définition puisqu'ils sont ainsi introduits), la moyenne (Y1+Y2+Y3)/3 ne peut que lui être inférieure ou égale (égale si Y1=Y2=Y3). Ainsi (X1,99+X2,99+X3,99)/3 donnera ((X1+X2+X3),99). CQFT (ce qu'il fallait... trouver).

Évidemment, si les coordonnées X et Y d'un des sommets au moins ne pouvaient s'exprimer dans les dix chiffres autorisés par la HP-41 C pour l'écriture d'un nombre, il faudrait traiter deux fois le problème — avec le même programme — et calculer d'une part (X1+X2+X3)/3 et d'autre part (Y1+Y2+Y3)/3. A suivre.

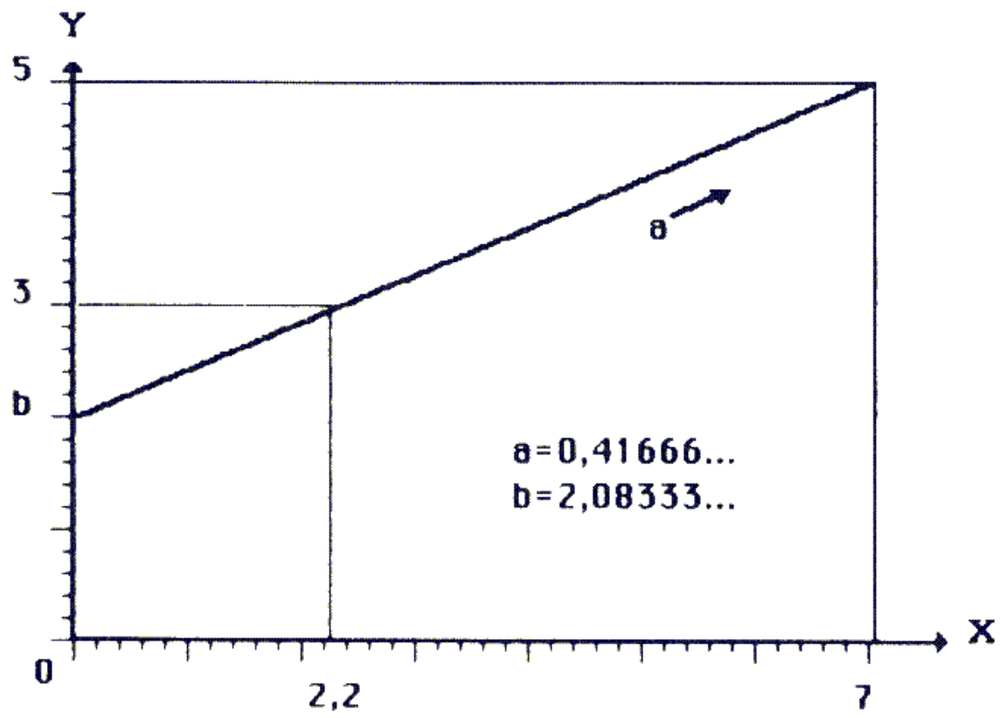
Jean-Christophe KRUST

QUI DIT MIEUX ?

ON sait que par deux points ne passe qu'une droite. D'ailleurs, si l'on a deux droites, on peut parier qu'elles ne sont qu'une.

Connaissant les coordonnées des deux points (X1, Y1) et (X2, Y2), retrouvez l'équation de la droite (il n'y en a qu'une...) qui, justement, passe par ces deux points.

On rappelle qu'une droite dans un repère (O, X, Y) est définie par l'équation Y = aX + b. Ce sont donc les deux coefficients a et b que le programme doit trouver sachant que Y1 = aX1 + b et, en même temps, Y2 = aX2 + b.



En 13 octets et sans aucun registre de mémoire, mon programme opère en 28 centièmes de seconde pour trouver l'équation de la droite passant par les points de coordonnées (13, 17) et (3, 11).

***NDLR** : et pas de STOP dans vos programmes !*

Pierre LANGLOIS

GRAND DÉFILÉ SUR PETIT ÉCRAN

Le terme « menu » est souvent employé en informatique pour désigner un ensemble de propositions parmi lesquelles l'utilisateur doit faire un choix. Dans la vie courante, le menu fait plutôt penser à un bon repas. Alors, si pour une fois un menu d'ordinateur nous mettait l'eau à la bouche, en défilant à l'écran ?

Le programme proposé ici est un prétexte pour illustrer le fonctionnement d'un *scrolling*. Un véritable menu (avec poulet, foie gras, moules, etc.) va défiler à l'écran d'un PB-700. Ce défilé (ou scrolling) pourra trouver d'autres applications : agenda, tarif, répertoire, etc.

Le fonctionnement du programme est des plus simples : à la mise en route, les trois premiers plats de la liste apparaissent, celui du haut étant désigné par un tiret. Une pression sur la flèche → fait descendre ce tiret ; au-delà du troisième plat, c'est l'écran qui se déplace sur la liste, vers le bas. A l'inverse, la touche ← effectue la remontée, vers le haut de la liste. On recrée ainsi le SCROLL que possèdent beaucoup d'ordinateurs de table.

Quand le tiret est en face du plat choisi, une pression sur la touche « retour chariot » renvoie à un sous-programme particulier à ce plat : cela peut être une recette, une explication, ou autre chose.

Faire plus concis

Avant de démarrer l'exécution, il est nécessaire d'apporter certaines adaptations à ce programme.

Ligne 6 : les valeurs données aux variables U et V correspondent respectivement au nombre de plats (ici 8) et au nombre de signes par plat (ici 18).

Lignes 16 à 19 : les DATA sont remplis avec les noms des plats ; ils doivent tous avoir le même nombre de signes pour éviter des affichages parasites pendant le défilement (compléter avec des espaces, si nécessaire). Chaque plat doit commencer par un chiffre qui est utilisé pour effectuer le branchement éventuel : par exemple, si le plat «7-TRIPES LYONNAISE» est choisi, le branchement a lieu vers la ligne 700, où se trouve le sous-programme particulier à ce plat (la recette, l'explication, la composition, ou autres).

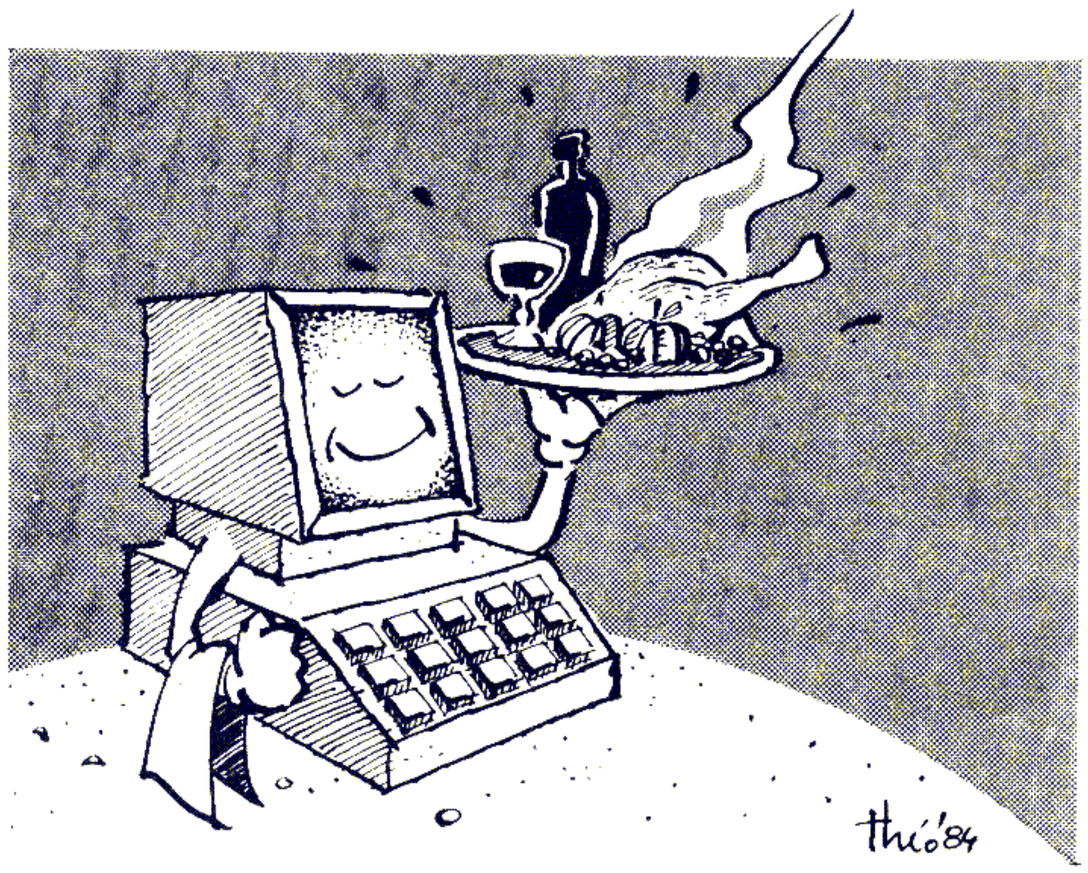
Traduction des codes CHR$ utilisés dans le programme

Valeur de CHR$	Touche
29	flèche vers la gauche ←
28	flèche vers la droite →
13	flèche « retour chariot » ↵
95	tiret –

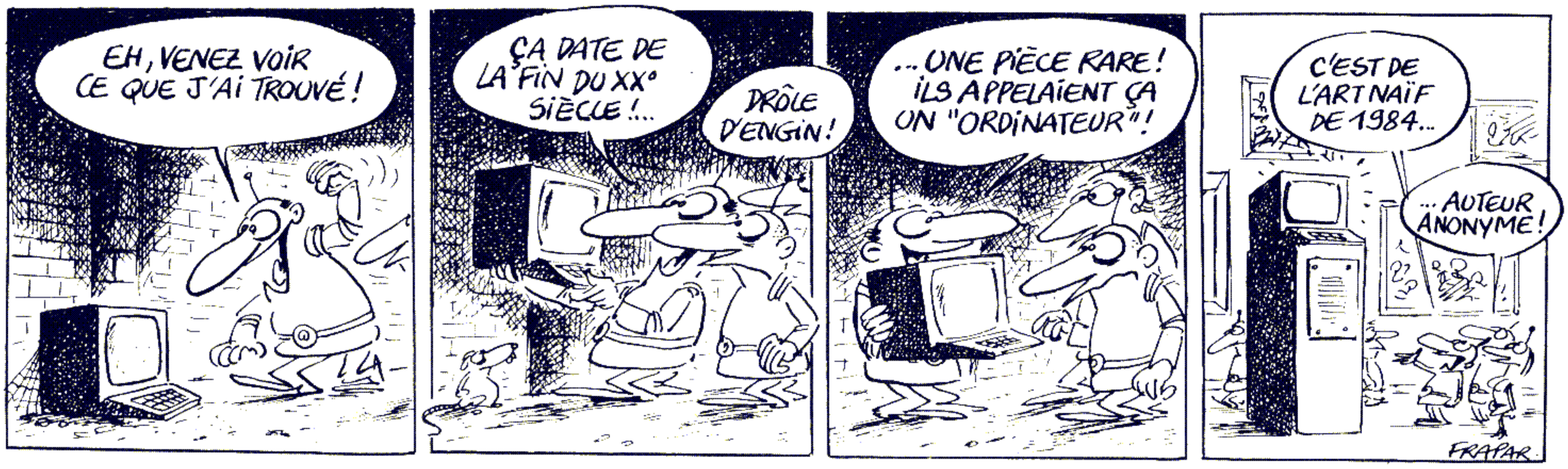
Enfin, il reste à supprimer toutes les REMs pour deux raisons : l'économie « d'huile de coude », et l'économie d'octets. Un utilitaire se doit d'être le plus court possible !

Benoît DE CABISSOLE

Scrolling
Programme pour PB-700
Auteur Benoît de Cabissole




```
 1 REM    SCROLLING
 2 REM    Initialisation du defilement
 3 REM  U=Nb de plats
 4 REM  V=Nb de signes par plat (18 m
   axi)
 5 CLEAR :CLS
 6 U=8:V=18
10 DIM A$(U)*V:J=0:L=U:FOR I=1 TO L:R
   EAD A$(I):NEXT I:GOSUB 20
12 REM  Branchement au plat choisi---
15 G=VAL(LEFT$(A$(I+1),1)):ERASE A$:C
   LS :GOSUB 100*G:RESTORE :GOTO 10
16 REM Definition des plats en Data--
17 DATA"1-POULET A LA GRQ.","2-RILLON
   S TOURAINE","3-MOULES ALA CREME"
18 DATA"4-BROCHET BEURRE B","5-FONDUE
    SAVOYARDE","6-FOIE GRAS CANARD"
19 DATA "7-TRIPES LYONNAISE","8-BOUDI
   N TRUFFE  "
20 REM ----- Debut du ss-programme sc
   rolling-----
21 REM
22 LOCATE 0,0:PRINT CHR$(95);A$(1):LO
   CATE 1,1:PRINT A$(2):LOCATE 1,2:PR
   INT A$(3)
25 I=J:GOSUB 95
30 IF K$=CHR$(13) THEN BEEP 1:IF I=>L
    THEN I=L-1:RETURN  ELSE RETURN
32 REM----Defilement vers le bas ----
35 IF K$=CHR$(28) THEN J=J+1:IF J>L T
   HEN J=L
38 REM---Defilement vers le haut ----
40 IF K$=CHR$(29) THEN 65 ELSE IF K$<
   >CHR$(28) THEN 25
45 IF J<3 THEN LOCATE 0,I:PRINT " ":L
   OCATE 0,J:GOTO 60
50 IF J+1>L THEN 25
55 LOCATE 0,2:PRINT " ":LOCATE 0,3
60 PRINT CHR$(95);A$(J+1):GOTO 25
65 J=J-1:IF J=-1 THEN J=0
70 IF J<2 THEN 90
75 IF J=L THEN CLS
79 REM ---  Nouvel affichage  ---
80 LOCATE 1,0:PRINT A$(J-1):LOCATE 1,
   1:PRINT A$(J)
85 LOCATE 0,2:PRINT CHR$(95);A$(J+1):
   GOTO 25
90 FOR H=0 TO 2:LOCATE 1,H:PRINT A$(H
   +1):NEXT H:GOTO 45
95 K$=INKEY$:IF K$="" THEN 95 ELSE RE
   TURN
98 REM
99 REM--- SUITE DU PROGRAMME  -------
   -(branchements)--------
100 PRINT "RECETTE1":GOSUB 950:RETURN
200 PRINT "RECETTE2":GOSUB 950:RETURN
300 PRINT "RECETTE3":GOSUB 950:RETURN
400 PRINT "RECETTE4":GOSUB 950:RETURN
500 PRINT "RECETTE5":GOSUB 950:RETURN
600 PRINT "RECETTE6":GOSUB 950:RETURN
700 PRINT "RECETTE7":GOSUB 950:RETURN
800 PRINT "RECETTE8":GOSUB 950:RETURN
950 REM  ---TEMPORISATION ---
960 FOR N=1 TO 50:NEXT N:RETURN
```



LE BASIC DE L'HECTOR 2 HR+

HECTOR existe en plusieurs modèles : Hector 1, Hector HRX, Hector 2 HR+. C'est le Basic de ce dernier qui nous intéresse ici. Sur cette version, il est résident en mémoire morte, mais les possesseurs de HRX (Forth résident) pourront se le procurer sous forme de cassette ou de cartouche. Notons que l'on trouve aussi une version plus étendue de ce Basic mais elle n'est disponible que sur disquette.

L'Hector dispose, une fois n'est pas coutume chez les familiaux, d'un excellent clavier, agréable au toucher et, qui plus est, AZERTY accentué. On regrettera simplement l'absence de pavé numérique et de bloc d'édition séparé. Bien présent en revanche, à la droite du clavier, un magnéto-cassette intégré qui donne toute satisfaction à un détail près : il n'a pas de compteur, ce qui, évidemment, rend problématique le repérage des programmes sur la bande.

S'il est un élément spécialement important pour le programmeur, c'est bien l'éditeur. Sans ce programme, la machine n'est plus guère utilisable. Cet intermédiaire indispensable a été particulièrement bichonné sur Hector. On y accède de deux façons : le mode Basic et le mode Edit. Avec le premier, que tout le monde connaît, il est possible d'entrer ou de corriger une ligne en utilisant la commande EDIT suivie d'un numéro de ligne. Celle-ci apparaît et nous pouvons alors insérer ou supprimer des caractères à l'intérieur de la ligne appelée. A signaler cependant qu'il est possible de dupliquer une ligne en changeant simplement son numéro.

Pour une utilisation plus originale

Si l'on tape EDIT seul, on entre dans un autre mode où les commandes classiques ne sont plus exécutables en mode direct, à l'exception de RUN. Nous disposons alors de nouvelles commandes très puissantes :

- RENUM qui renumérote tout ou partie d'un programme,
- AUTO qui numérote automatiquement les lignes,
- DELETE qui détruit tout ou partie d'un programme.

Voilà qui paraît bien classique, mais attendons la suite :

- EXTRACT permet, comme son nom l'indique, d'extraire une partie du programme en mémoire, les autres lignes étant supprimées (c'est, en somme, le contraire de DELETE) ;
- avec LOCATE, on localise une chaîne de caractères ;
- LAST fait réapparaître la dernière ligne ;
- enfin, MERGE et APPEND permettent de charger un programme en mémoire sans perdre celui qui s'y trouve déjà. Si MERGE ne fait que charger le second programme (en effaçant éventuellement les lignes du premier qui chevauchent celles du nouveau venu), APPEND fait mieux. Il le charge à la suite du premier en le renumérotant si nécessaire, dix numéros après la dernière ligne de celui-ci.

Les instructions de contrôle, quoique sans surprise, proposent des utilisations originales.

Si ON...GOTO est présent, l'instruction GOTO peut également être suivie d'une expression algébrique. Ainsi les séquences :
10 ON A GOTO 20,30,40,50 et

10 GOTO 10+10*A
sont équivalentes si A est un entier compris entre 1 et 4.

C'est aussi valable pour GOSUB.

La gestion des boucles FOR...NEXT est, quant à elle, assez curieuse. La variable suivant le NEXT n'est pas prise en compte par le Basic, son rôle est purement documentaire. Par exemple,
10 FOR I=1 TO 10
20 NEXT A
ne produit aucun message d'erreur. Il en est de même pour la séquence :
10 FOR I=1 TO 10
20 FOR J=1 TO 10
30 NEXT I
40 NEXT J

Notons d'autre part qu'une ligne 30 NEXT : NEXT : NEXT peut s'écrire 30 NEXT,,

Dernière curiosité au chapitre des boucles FOR...NEXT, la forme 10 FOR I,J,K=1 TO 10 est autorisée et équivalente à 10 I=0:J=0:FOR K=1 TO 10.

THEN peut être suivi de ELSE et PRINT USING permet de formater les affichages.

L'instruction REM est présente, bien sûr, mais là encore, les choses ne se passent pas comme d'habitude. En effet, le Basic d'Hector remplace systématiquement toutes les REMs par des apostrophes.

Les traditionnels READ, DATA, RESTORE sont également disponibles. RESTORE peut être suivi d'un numéro de ligne ou d'une expression algébrique, mais les lignes de DATA doivent contenir soit des constantes, soit des chaînes de caractères entre guillemets.

Puisque nous en sommes à parler des chaînes, abordons leur traitement. Il est des plus classiques : LEFT$, RIGHT$, MID$, STR$, ASC, VAL et LEN. En ce qui concerne MID$, notez que si le dernier paramètre n'est pas spécifié, tous les caractères sont pris en compte à partir de celui qu'indique le premier paramètre. Par exemple, MID$("ABC DE",4) extrait DE. Si cette séquence est équivalente à MID$("ABCDE",4,2), elle a l'avantage d'être plus rapide et plus concise, surtout quand la longueur de la chaîne n'est pas connue à l'avance.

A côté d'INKEY$ (saisie d'un caractère au clavier), on dispose d'INSTR$ (nombre de caractères) qui est une sorte d'INPUT attendant la frappe du nombre de caractères spécifiés. La touche ENTER est alors inopérante et la syntaxe est la même que pour INKEY$, c'est-à-dire du type A$=INSTR$(N).

Les fonctions mathématiques disponibles sont relativement peu nombreuses : SIN, COS, TAN, ATN, EXP, LOG, SGN, INT, SQR. Heureusement, il est possible d'en définir de nouvelles grâce à DEFFN. Une curieuse fonction PI (expression) donne le nombre PI multiplié par une valeur fournie : PI(2) donne 6,28319.

Les fonctions logiques sont AND, OR et NOT, et le hasard a sa place avec RND mais, comme il est d'usage chez Micronique, cette fonction n'utilise pas la syntaxe la plus courante. Ainsi RND(3,6) retournera un nombre réel compris entre 3 inclus et 6 exclu. RND est complété par l'instruction SEED qui force la valeur de la racine servant au calcul du nombre aléatoire. Pour une même valeur de SEED, la série de nombres aléatoires générés sera identique. La séquence SEED TIME(1) initialise la racine de façon aléatoire et elle équivaut au RANDOM de Microsoft.

Variables : pas de quoi se vanter

Hector ne connaît que deux types de variables : les variables alphanumériques, reconnues comme telles lorsqu'elles sont suivies du symbole $ (l'instruction DEFSTR n'existe pas ici), et les variables numériques (nous touchons là le point faible de la machine) codées sur quatre octets. Ces dernières ne fournissent que six chiffres significatifs qui sont compris entre 1,701 E38 et −1,701 E38. (Pas de variables de type entier, ni de réels en double précision.) Le nom des variables peut comporter autant de lettres que désiré, seules les deux premières sont significatives.

L'instruction DIM autorise la déclaration de tableaux dont le nombre des dimensions ne sera limité que par la longueur maximum d'une ligne (222 caractères).

Au chapitre de la manipulation des variables, nous découvrons des possibilités assez étendues. C'est ainsi que l'on peut procéder à plusieurs affectations simultanées : la séquence A,B,C,D (5,2)=8 est autorisée. D'autre part, l'instruction SWAP échange le contenu de deux variables alors que les fonctions MIN et MAX fournissent respectivement la plus petite et la plus grande valeur de deux expressions algébriques.

Les possibilités graphiques d'un ordinateur, surtout s'il s'agit d'un « familial », peuvent contribuer à son succès... ou à son échec commercial. Voyez l'Apple II qui, en son temps, fut le premier à offrir des possibilités élevées dans ce domaine. Micronique l'a bien compris et les progrès réalisés depuis l'Hector 1 sont énormes. La définition graphique du 2 HR+ est de 243×231 points en quatre couleurs à choisir avec COLOR et BRIGHT, dans une palette de 16 couleurs (en comptant la demi-luminosité).

L'instruction PLOT possède deux syntaxes. La première (de la forme PLOT X,Y,C) allume un point de couleur C aux coordonnées X,Y. La seconde (de la forme PLOT X,Y,L,H,C) allume un rectangle de couleur C, de longueur L et de hauteur H dont le coin supérieur gauche a pour coordonnées X,Y. LINE aussi possède deux syntaxes. LINE X1,Y1,X2,Y2,C allume une ligne de couleur C allant du point X1,Y1 au point X2,Y2. Alors que LINE X1,Y1,C trace une ligne commençant au dernier point spécifié par le précédent LINE et se terminant en X1,Y1.

Toujours au chapitre des instructions graphiques, citons la très puissante instruction SCROLL qui permet de déplacer l'ensemble de l'image dans toutes les directions (y compris en diagonale !)

FLASH enfin est destiné aux jeux d'action. Il fait « flasher » tous les points de l'écran allumés dans la couleur spécifiée en utilisant la complémentaire de cette couleur.

Tout en mode graphique

A l'instar du Macintosh et du Sinclair QL, il n'existe pas de véritable mode texte, l'affichage des caractères s'effectuant en mode graphique. Le système doit écrire dans plusieurs octets pour provoquer l'affichage d'une seule lettre. L'écriture du texte est donc assez lente. Mais bien des fantaisies sont permises ; il devient par exemple possible de placer les caractères au pixel près et de pratiquement superposer des lettres en utilisant plusieurs couleurs différentes.

C'est l'instruction CURSOR X,Y qui positionne le curseur à la position désirée, X étant compris entre 0 et 242, et Y entre 0 et 230.

SCREEN permet de redéfinir la fenêtre d'affichage : SCREEN X,Y,L,H définit une fenêtre dont le coin supérieur gauche est en X,Y, dont la largeur est L et la hauteur H. Cette opération s'effectue au niveau du pixel, le plein écran correspondant à un SCREEN 13,230, 230,230. WIPE annule le SCREEN en cours et PEN permet de définir la couleur d'un affichage effectué par l'instruction PRINT.

L'instruction OUTPUT rend possible

LE BASIC DE L'HECTOR 2 HR +

un affichage de texte à partir de la position spécifiée et même en dehors de la fenêtre spécifiée par SCREEN, ce que le couple CURSOR-PRINT n'autorise pas.

Enfin, avec la fonction POS, nous obtenons des informations concernant l'état de l'affichage et de l'imprimante :

- POS(1) indique le numéro de la colonne du dernier caractère affiché,
- POS(2) fait de même pour l'imprimante,
- POS(3) retourne l'abscisse du premier point suivant le dernier affichage,
- POS(4) retourne l'ordonnée du point supérieur gauche du prochain caractère à afficher.

Dans un autre domaine, il faut bien l'avouer, l'Hector n'est pas très musicien et seules deux instructions nous sont proposées. TONE X,Y produit un son de hauteur X et de durée Y. Une précision cependant : les deux paramètres ne sont pas indépendants et la durée sera constante si le produit de X par Y reste constant. L'exécution de TONE 10,5 et celle de TONE 20,5, par exemple, donnent des sons de durées différentes.

SOUND X,Y quant à lui, émet un bruit dont la nature est déterminée par X et Y. Il n'y a pas de règle évidente et seuls des tâtonnements vous permettront de trouver ce que vous cherchez.

Une horloge interne

Les sauvegardes sur cassette s'effectuent à la vitesse honorable de 1 500 bauds et elles semblent fiables. Concernant la gestion du cassettophone intégré, c'est sa portion congrue : SAVE et LOAD permettent lectures et sauvegardes de programmes. Les données ne sont sauvées que sous forme de tableaux numériques. Les éléments d'un tableau A(20) seront sauvés avec un SAVE*A et lus par LOAD*A.

TAPE allume et éteint le magnétophone alors que REWIND ne permet que la mise sous tension, la mise hors tension étant alors obtenue par la frappe de n'importe quelle touche. Comme son nom l'indique, cette instruction sert surtout à rembobiner la bande magnétique.

Hector 2 HR+ possède aussi une horloge interne gérée de manière fruste : TISET la remet à zéro et TIME(x) en autorise la lecture, x étant un multiplicateur. Ainsi, la séquence

```
10 PRINT TIME(50)
20 GOTO 10
```

affiche le nombre se secondes écoulées depuis la dernière mise à zéro. En remplaçant 50 par 3000, nous aurons les minutes, etc.

Deux autres instructions utilisent l'horloge interne : PAUSE (durée) qui suspend l'exécution du programme, et SPEED(x) qui en fait varier la vitesse. A la mise sous tension, SPEED vaut 0 et la vitesse est maximum.

Le Basic de l'Hector donne accès au langage-machine. Rien d'extraordinaire, mais le minimum y est.

Avec PEEK et POKE, vous pourrez lire et écrire en mémoire, avec OUT et INP, gérer les ports d'entrée-sortie du Z80 et avec USR appeler un programme en langage-machine. La syntaxe originale de cette dernière instruction la rend très souple d'emploi, USR x,y, par exemple, lance l'exécution d'un programme commençant à l'adresse x, la valeur y étant placée dans le registre DE du Z80.

USR x,t1,t2,t3,... effectue la même opération, à ceci près que DE pointe cette fois sur une table de paramètres contenant les valeurs t1,t2,t3,...

A la seule lecture de la liste des instructions disponibles, le Basic de l'Hector 2 HR+ peut paraître classique. Il recèle en réalité une bonne dose d'ingéniosité, certaines de ces instructions s'utilisant selon plusieurs syntaxes. Le revers de la médaille est que l'on s'écarte sensiblement du pseudo-standard inspiré, notamment, par Microsoft. L'adaptation de programmes écrits dans des Basic d'origine différente risque donc de poser certains problèmes.

Josette GODEFROY

Liste des mots du Basic de l'Hector 2 HR+

ABS	NOT
AND	ON
ASC	OR
ATN	OUT
BRIGHT	OUTPUT
CHR$	PAUSE
CLEAR	PEEK
CLS	PEN
COLOR	PI
CONT	PLOT
COS	POINT
CURSOR	POKE
DATA	POS
DEF	POT
DIM	PRINT
EDIT	READ
ELSE	REM
END	RESTORE
ERROR	RETURN
EXP	REWIND
FIRE	RIGHT$
FLASH	RND
FN	RUN
FOR	SAVE
FREE	SCREEN
GOSUB	SCROLL
GOTO	SEED
IF	SGN
INKEY$	SIN
INP	SOUND
INPUT	SPEED
INSTR$	SPC(
INT	STEP
JOY	STOP
LEFT$	STR$
LEN	SQR
LET	SWAP
LINE	TAB(
LIST	TAN
LLIST	TAPE
LOAD	THEN
LOG	TIME
LPRINT	TISET
MAX	TO
MID$	TONE
MIN	USING
NEXT	USR
NEW	VAL
	WIPE

Fiche technique de l'Hector 2 HR +

Constructeur : *Micronique*
Prix public : *4 390 FF*
Mémoire vive : *48 Ko*
Mémoire morte : *16 Ko*
Langages : *Basic résident, Forth, Pascal, Assembleur*
Affichage texte : *23 lignes de 38 caractères*
Vitesse cassette : *1 500 bauds*
Calcul numérique sur : *7 chiffres*

AVEC LA RÈGLE ET LE COMPAS

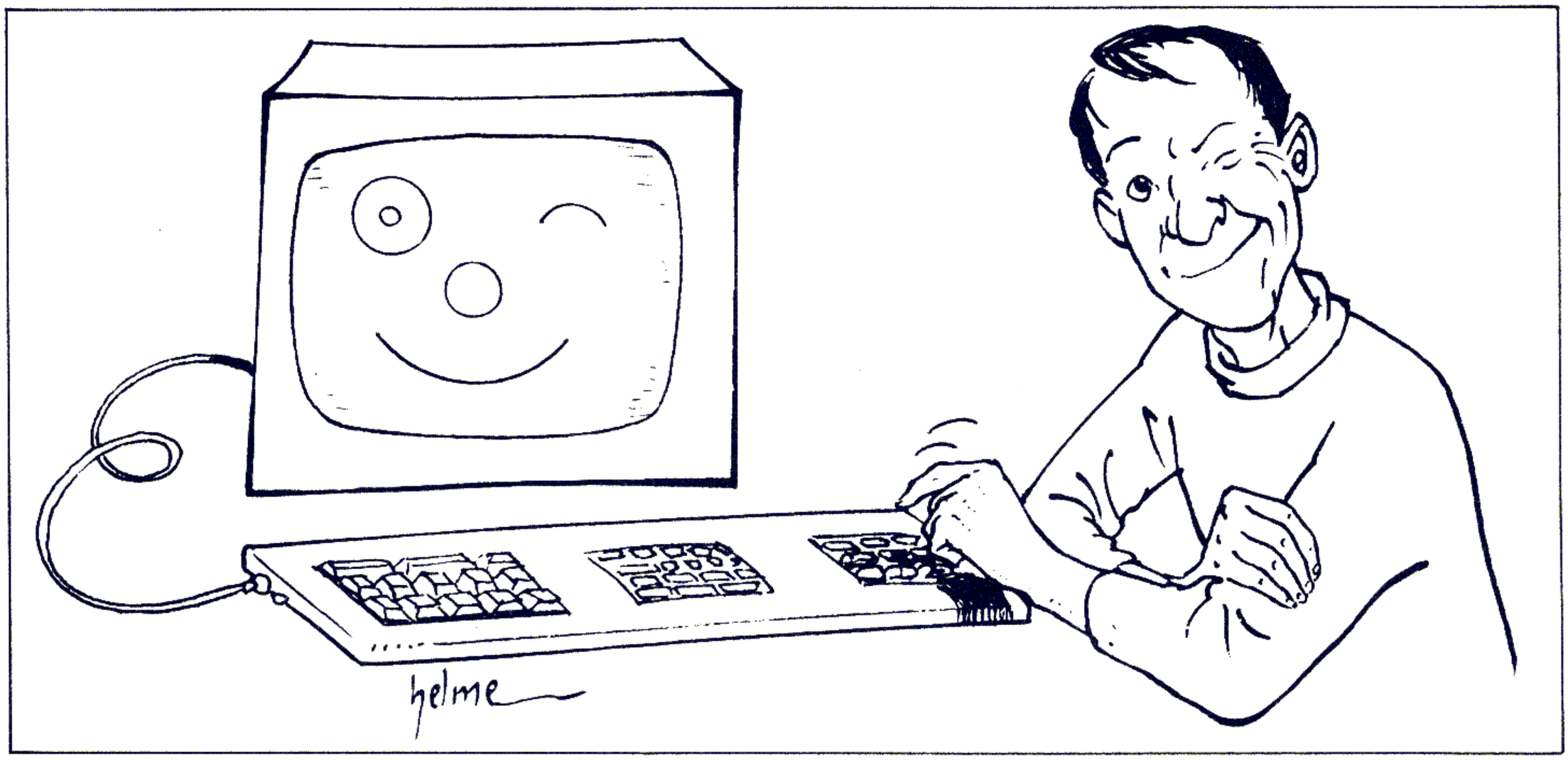
AVEZ-vous essayé le cubisme sur l'écran de votre Amstrad ? Ce programme vous fournit le minimum nécessaire : droites, rectangles, triangles et cercles, sans oublier la gomme.

Le mois dernier (*LIST* 5, page 76), nous avons vu comment dessiner sur une machine MSX. Voici un autre programme dessinateur de conception sensiblement différente et destiné cette fois-ci à l'Amstrad.

Chacune des commandes élémentaires (au total une quinzaine) est accessible en pressant une seule touche. Ainsi, pour dessiner une droite, on utilisera successivement trois commandes : on commence par une pression sur V (validation d'un point, c'est-à-dire ici du début de la droite) ; on déplace alors le curseur en utilisant les flèches et l'on obtient le tracé avec une pression sur la touche D (figure 1, page suivante).

Une autre façon de dessiner

S'il s'agit de dessiner un cercle (plein), on appuie sur V (validation d'un point, en l'occurrence le centre du cercle), puis on déplace le curseur jusqu'à l'un quel-



AVEC LA RÈGLE ET LE COMPAS

Dessinateur
Programme pour Amstrad
Auteur Jacques Boisgontier


```
10 '---------------- DESSINATEUR
20 '
30 ce=1:cf=0      ' couleur ecriture et fond
40 INK 0,26       ' fond blanc
50 INK 1,0        ' encre noire
60 PAPER 0:PEN 1    ' stylo 1
70 MODE 1
80 LOCATE 1,18:PRINT "CLAVIER MAJUSCULE"
90 LOCATE 1,20:PRINT "PREMIER POINT:fleches Puis 'V'"
100 LOCATE 1,21:PRINT "2EME POINT: fleches Puis:"
110 LOCATE 3,22:PRINT "D:droite   R:rectangle"
120 LOCATE 3,23:PRINT "C:cercle   Y:triangle(V pour 2 Points)"
130 LOCATE 3,24:PRINT "COULEUR: 1,2,3   F:FOND (GOMMER)"
140 LOCATE 3,25:PRINT "A:annulation derniere droite"
150 xa=200:ya=300                    ' Point Precedent
160 xb=xa:yb=ya
170 x=xa+20:y=ya
180 '---------------- CURSEUR CLIGNOTANT
190 t=TEST(x,y)
200 '
210 c$=INKEY$:IF c$<>"" THEN 250
220 PLOT x,y,ce:PLOT x,y,cf
230 GOTO 210
240 '
250 PLOT x,y,t
260 c=ASC(c$)
270 IF C=242 THEN X=X-2:GOTO 190
280 IF C=243 THEN X=X+2:GOTO 190
290 IF C=240 THEN Y=Y+2:GOTO 190
300 IF C=241 THEN Y=Y-2:GOTO 190
310 c$=UPPER$(c$):LOCATE 1,17:PRINT c$
320 IF C$="V" THEN PLOT x,y,ce:xb=xa:yb=ya:xa=x:ya=y
330 IF C$="D" THEN PLOT xa,ya,ce:DRAW x,y,ce:xb=xa:yb=ya:ya=y:xa=x
340 IF C$="C" THEN GOSUB 470
350 IF c$="R" THEN GOSUB 420:xa=x:ya=y
360 IF c$="Y" THEN GOSUB 550:xb=xa:yb=ya:xa=x:ya=y
370 IF c$="A" THEN PLOT xb,yb:DRAW xa,ya,cf:x=xb:y=yb:xa=xb:ya=yb
380 IF C$="F" THEN CE=0
390 IF C$>="0" AND C$<="9" THEN CE=VAL(C$)
400 GOTO 190
410 '------------- rectangle Plein
420 FOR y1=ya TO y STEP SGN(y-ya)
430 PLOT xa,y1,ce:DRAW x,y1,ce
440 NEXT y1
450 RETURN
460 '---------------------- Cercle Plein
470 r=SQR((xa-x)^2+(ya-y)^2):r2=r^2
480 FOR dx=-r TO r
490  dy=SQR(r2-(dx^2))
500  PLOT xa+dx,ya+dy,ce:DRAW xa+dx,ya-dy,ce
510 NEXT dx
520 RETURN
530 '---------------------- triangle Plein
540 ' valider 2 Points avec 'V' Puis un troisieme avec 'Y'
550 d=SQR((yb-ya)^2+(xb-xa)^2)
560 IF d=0 THEN RETURN
570 cx=(xa-xb)/d:cy=(ya-yb)/d
580 FOR dd=0 TO d STEP 0.5
590  x3=xb+dd*cx:y3=yb+dd*cy
600  PLOT x,y,ce:DRAW x3,y3,ce
610 NEXT dd
620 RETURN
630 '
640 '  Permet de dessiner des droites
650 '  des rectangles et cercles Pleins
660 '  ex: Pour tracer une droite:   1/aPPuyer sur 'V'
670 '                                2/dePlacer le curseur et aPPuyer sur 'D'
680 '
690 '      Pour tracer un cercle:    1/aPPuyer sur 'V'
700 '                                2/dePlacer le curseur et aPPuyer sur 'C'
710 '
720 '    Pour effacer,utiliser 'F' et 'R'.
730 '
740 '    Avec 'F' ,le curseur n'aPParait que sur les zones coloriees.
750 '    Pour faire aPParaitre a nouveau le curseur,fraPPer '1'.
760 '
770 '    Pour des figures discontinues,utiliser 'V'.
780 '
```

conque des points de la circonférence et on appuie sur C (figure 2).

Le triangle plein s'obtient en désignant les deux premiers sommets avec V puis le troisième avec Y. C'est le segment de droite reliant Y au deuxième sommet V qui pivote sur lui-même pour « remplir » le triangle.

Figure 1

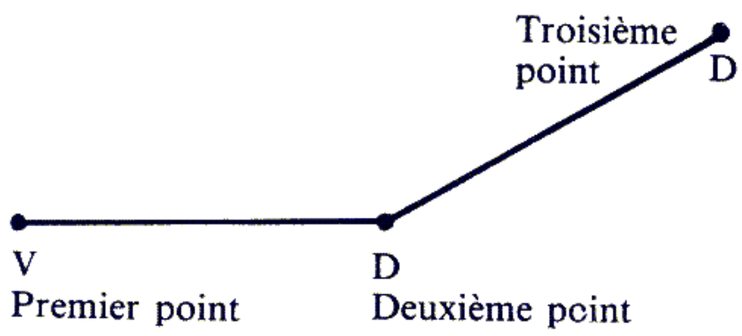
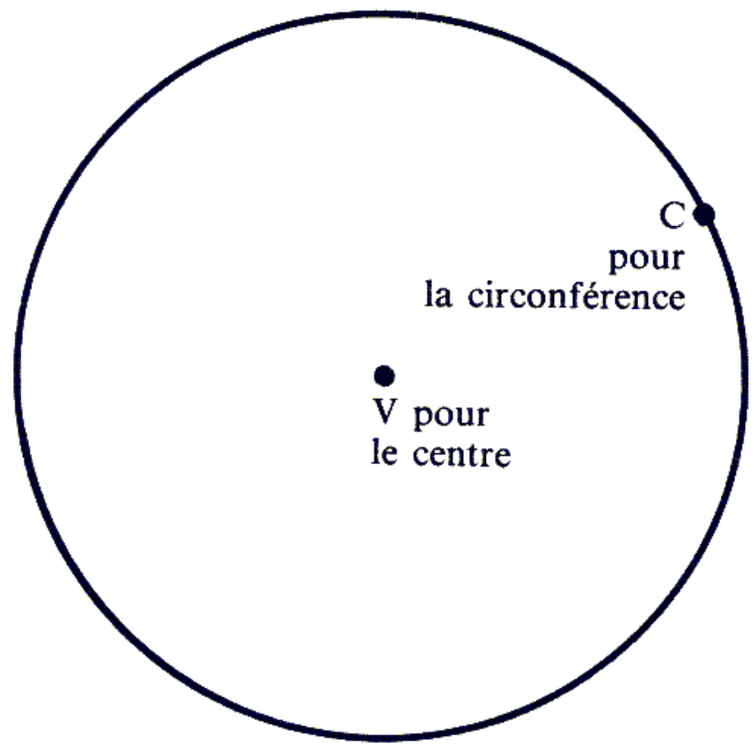
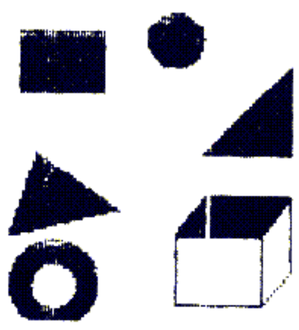


Figure 2





Grâce à ce triangle plein, on peut construire un grand nombre de figures, en fait toutes celles dans lesquelles n'entrent que des segments de droite (voir l'exemple du cube). Il remplace donc partiellement l'instruction PAINT qui existe sur d'autres matériels et notamment sur les MSX.

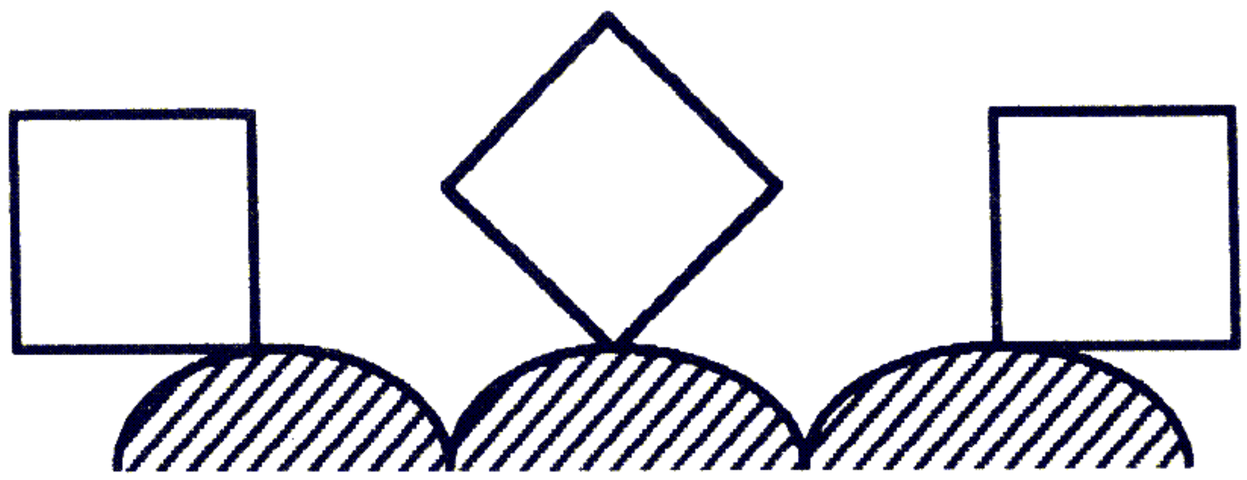
Même principe pour le tracé d'un rectangle plein : V pour valider l'un de ses angles et R dès que le curseur est situé sur l'angle opposé.

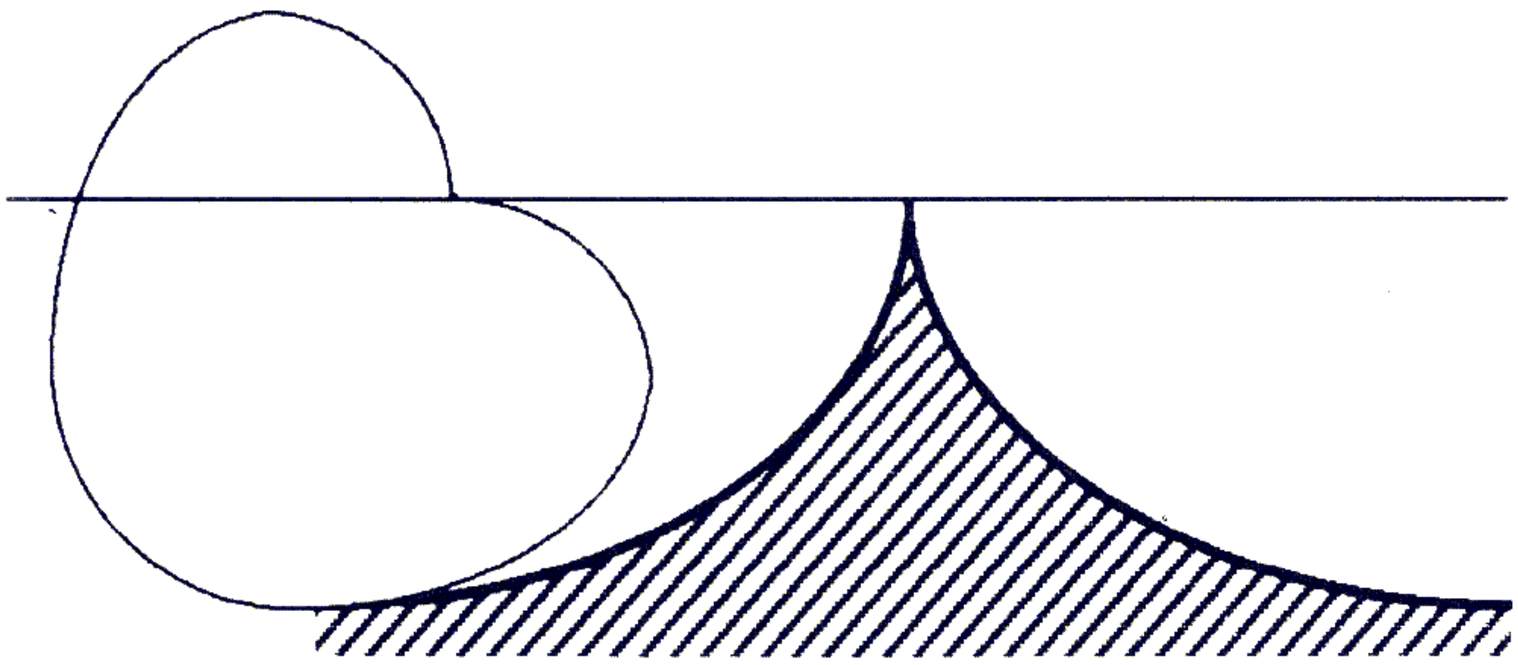
En utilisant la couleur de fond (touche F) pour colorier un motif, on efface bien entendu certaines parties du dessin, c'est la gomme. Enfin, une fois que le dessin est achevé, rien n'est plus facile que de l'archiver sur cassette en tapant SAVE "XXX",B,&C000,&4000 ; les trois caractères entre guillemets sont au choix de l'utilisateur : ils servent à identifier le fichier.

Jacques BOISGONTIER

LA ROUE NE TOURNE PLUS ROND

A la recherche des cycloïdes, nous avons découvert, dans LIST 4, ces courbes abstraites que décrivent des points de pièces rondes qui roulent. Mais le cercle lui-même n'est qu'une forme idéale de roue, et le sol bien plat une splendide utopie.



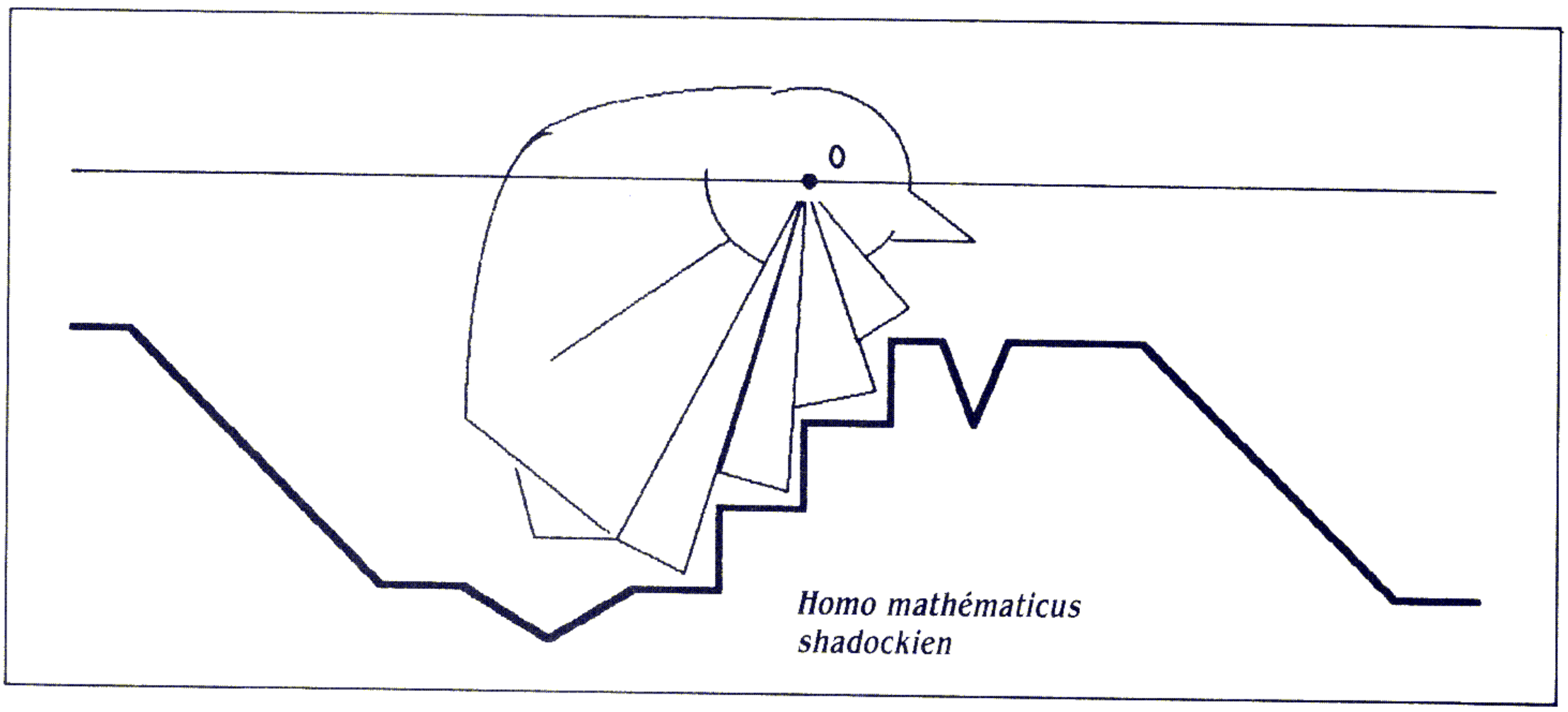


Imaginons une roue carrée ! Pourquoi pas ? Il existe bien un type de sol particulièrement adapté à son emploi. La figure ci-dessus en est l'illustration.

Plus loin encore, vers l'univers shadockien des mathématiciens pervers : voyez ce monstre spécialement conçu pour rouler sur un sol d'une très exemplaire irrégularité ! (ou régularité, c'est selon).

Notre propos est — plus sérieusement —, sans entrer dans le détail des explications mathématiques, de trouver le sol le plus adapté (son équation et le tracé

LA ROUE NE TOURNE PLUS ROND



Homo mathématicus shadockien

correspondant) à une roue de forme quelconque (sachant son équation).

Rapidement. On se donne donc une équation de roue : $\rho = f(\theta)$ en coordonnées polaires, on doit déterminer une autre courbe, celle du sol le plus adéquat. Cette adéquation s'apprécie « matériellement » par un roulis nul (type Rolls-Royce sur autoroute flambant neuve) et, plus mathématiquement, en posant que le pôle O (ou centre, voir figures) suive, dans le déplacement de la roue, une ligne droite horizontale.

Ce sol idoine possède une équation donnée par :

$$x = x_0 + \int_{T_0}^{T_1} \rho \quad dT$$

et, bien sûr, $y = \rho$

Le programme ci-contre pour Sharp PC-1500 utilise ces formules pour dessiner le sol, les positions de la roue et l'axe.

Il emploie une routine d'auto-programmation de formules mathématiques du PC-1500 *(l'Ordinateur de poche* n° 11). Il n'est donc nécessaire que d'introduire la formule de la courbe, directement, pour qu'elle soit programmée dans le Basic.

Trois modèles de courbes sont édités en illustration : la spirale d'Archimède, le cercle et une courbe spéciale.

Le programme trace les axes (ligne numéro 150), imprime les données de base (90 et 100), calcule l'intégrale de f (T) dT de T_0 à T_1 (200 à 220) puis le tracé du sol (230). Les différentes positions de la roue sont calculées (de 270 à 370) et tracées.

Modus operandi

Après RUN qui lance l'exécution, indiquer la partition, c'est-à-dire le nombre de points à tracer sur la courbe, puis le nombre de roues à dessiner à intervalles réguliers et proportionnels à

La roue ne tourne plus rond
Programme pour PC-1500
Auteur Christophe Masurel


```
10:REM LA ROUE NE
   TOURNE PLUS R
   OND
15:GOTO "A
20:"f"PT=--------
   --------------
   --------------
   --------------
   --------------
   -------
30:"E"RESTORE "f"
   :I=(PEEK &78BE
   -128)*256+PEEK
   &78BF+6:INPUT
   "EQUATION ROUE
   =?";J
40:J=31679:FOR I=
   ITO I-10+PEEK
   (I-7):IF PEEK
   J<>13POKE I,
   PEEK J:J=J+1:
   NEXT I
50:POKE I,58,241,
   153,13:RETURN
60:"A"RADIAN :
   CLEAR :INPUT "
   PARTITION:N=";
   N,"NB ROUES=";
   NR,"MAX DE PT=
   ";PM
70:INPUT "BORNE0=
   ";T0,"BORNE1="
   ;T1
80:GOSUB "E
90:TEXT :CSIZE 1:
   COLOR 0:LPRINT
   "PARTITION:";N
   :LPRINT "NB RO
   UES:";NR:
   LPRINT "MAX PT
   :";PM
100:LPRINT "BORNES
   :";T0;"   ,";T
   1:LPRINT :
   LLIST 20:LF 1
```

la distance de T_0 à T_1 (ci-après). Le maximum de points (max. de ρ) est, en quelque sorte, le coefficient d'échelle du dessin. Enfin, les bornes T_0 et T_1 définissent l'intervalle de variation de la variable T dont l'équation f (T) définit la roue.

Quant à cette équation, on l'introduira le plus simplement du monde, comme un message alphabétique (avec possibilité d'abréger d'un point les ordres du Basic) : SINT, par exemple pour la fonction de roue f (t) = sin t.

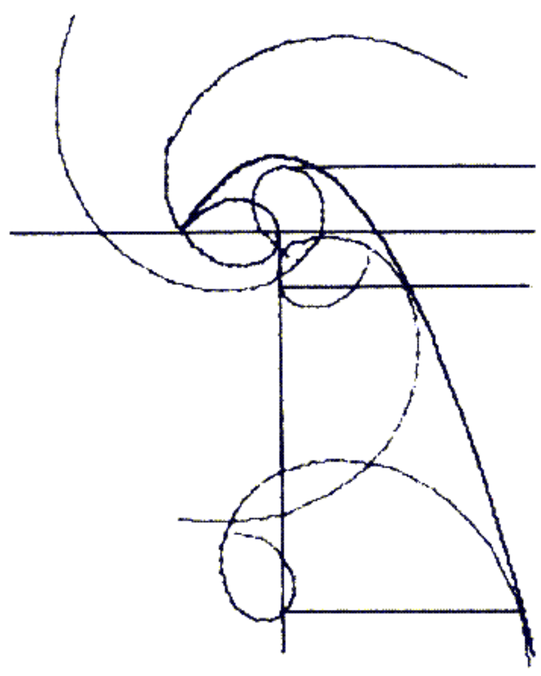
Avant toute chose, le programme écrit ces paramètres caractéristiques sur l'imprimante, on y trouvera donc (listes ci-contre) trois exemplaires tracés à ne manquer à aucun prix.

Que de roues n'est-on pas tenté d'imaginer... Des fermées, à tangente au sol continue — ou non (« angles vifs »). Et des roues non fermées telle la spirale d'Archimède. Le mathématicien que rien n'arrête — surtout pas le réel — saura travailler sur des roues ouvertes, des profils qui s'interpénètrent... et le PC-1500 tracera.

Christophe MASUREL

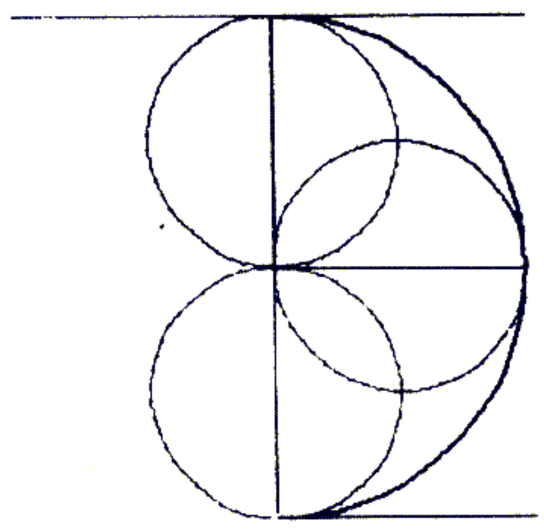
Spirale d'Archimède

```
PARTITION: 50
NB ROUES: 4
MAX PT: 40
BORNES:-1.57079632
7    ,  4


 20:"f"PT=10*T:
    RETURN
```



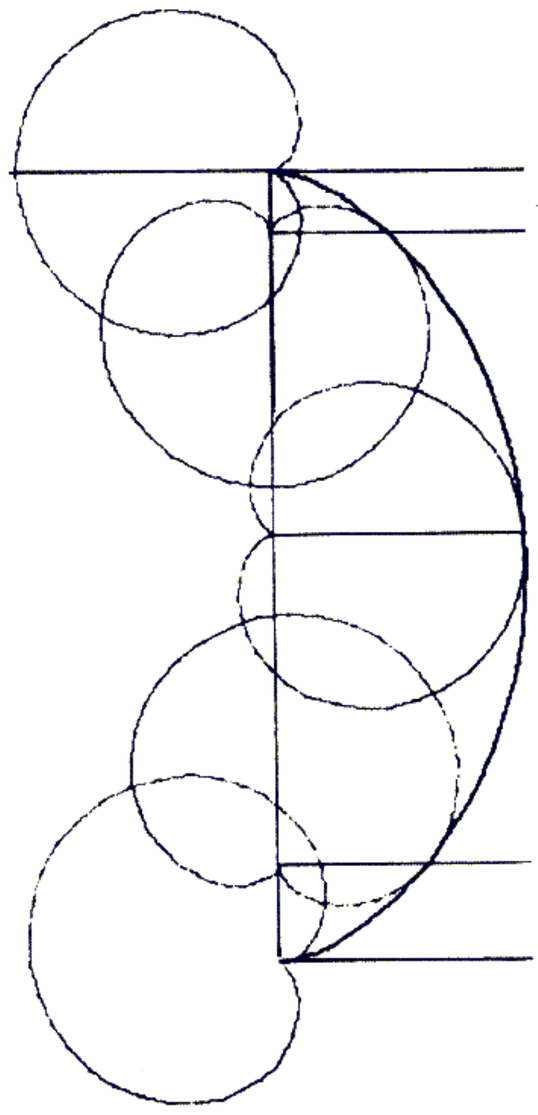
Déplacement du cercle

```
PARTITION: 50
NB ROUES: 3
MAX PT: 1
BORNES: 0    ,  3.14
1592654


 20:"f"PT=SIN T:
    RETURN
```



Courbe spéciale

```
PARTITION: 50
NB ROUES: 5
MAX PT: 10
BORNES: 0    ,  6.28
3185307


 20:"f"PT=10*(SIN
    (T/2)^2):
    RETURN
```

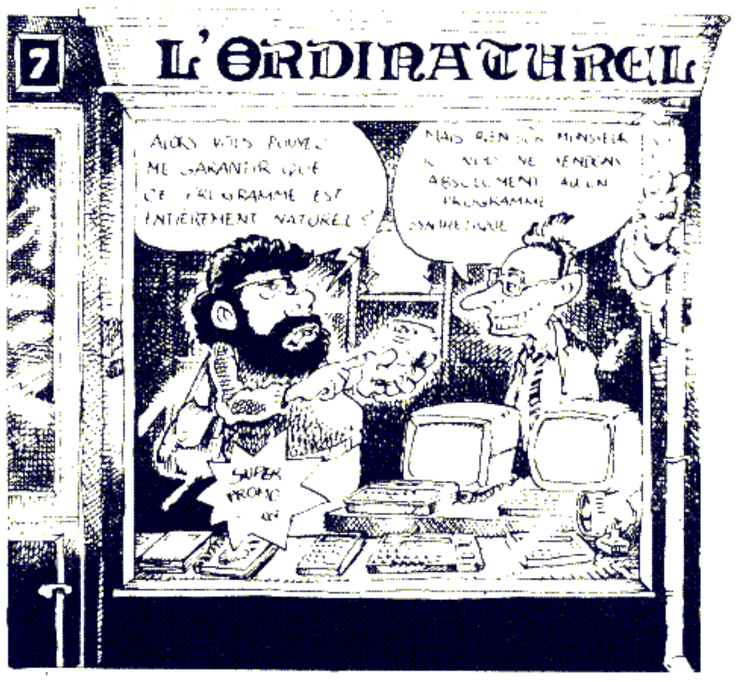


```
110:REM INITIALISA
    TION
120:NR=NR-1:Q=INT
    (N/NR):C1=Q:J=
    1:SO=0:IN=0:CO
    =105/PM
130:R=(T1-T0)/(2*N
    )
140:DIM IT(NR):IT(
    0)=0
150:GRAPH :
    GLCURSOR (110,
    -60):SORGN
160:LINE (-110,0)-
    (110,0):LINE (
    0,-100)-(0,0)
170:REM TRACE DU S
    OL
180:COLOR 1:T=T0:
    GOSUB "f":M0=P
    T:YO=M0*CO:
    GLCURSOR (YO,0
    )
190:FOR I=1TO N
200:T=T+R:GOSUB "f
    ":M1=PT
210:T=T+R:GOSUB "f
    ":M2=PT
220:S=M0+4*M1+M2:S
    O=SO+S:IN=SO*R
    /3
230:X=IN*CO:Y=M2*C
    O:LINE -(Y,-X)
240:IF I=C1LET IT(
    J)=IN:J=J+1:C1
    =J*Q
250:M0=M2:NEXT I
260:COLOR 0:LINE (
    0,-X)-(0,0)
270:REM TRACE DES
    ROUES
280:GLCURSOR (0,0)
    :COLOR 2
290:FOR K=0TO NR:T
    =T0:H=2*R:G=H*
    K*Q:GOSUB "f"
300:U=IT(K)*CO:
    LINE (0,-U)-(1
    10,-U)
310:U=(PT*COS G)*C
    O
320:U=(IT(K)-PT*
    SIN G)*CO:
    GLCURSOR (U,-U
    )
330:FOR O=1TO N:T=
    T+H:GOSUB "f"
340:U=PT*COS (G-O*
    H)*CO
350:U=(IT(K)-PT*
    SIN (G-O*H))*C
    O
360:LINE -(U,-U):
    NEXT O
370:NEXT K:TEXT :
    END
```

PROGRAMMATION SYNTHÉTIQUE DE LA HP-41 C

CES DIABLES D'INSTRUCTIONS

Les programmeurs de chez HP ont mis grand soin à faire de la HP-41 C une machine sophistiquée, sérieuse, évolutive... Tellement même qu'au-delà de sa face officielle, la programmation synthétique ouvre la porte aux manipulations les plus subtiles.



Il est bien loin le temps des pionniers de la programmation synthétique pour qui le Cric ou son cousin saxon Byte jumper étaient les seules clés d'accès au jardin synthétique. Le module d'extension X-FUNCTIONS simplifie considérablement le travail en programmation synthétique.

Grâce aux instructions ATOX, XTOA, AROT et la magnifique PASN, on va réaliser un programme dont la fonction est d'assigner « n'importe quoi » à n'importe quelle touche du clavier.

C'est avec ce programme, KASN, qu'on réalisera un clavier synthétique, c'est-à-dire où la plupart des fonctions sont assignées. Finies, alors, les longues manipulations nécessaires à la création d'instructions synthétiques : elles sont maintenant assignées et disponibles en permanence.

L'heure du repos n'a cependant pas encore sonné : il faut introduire en mémoire le programme KASN avec ses quatre instructions synthétiques, respectivement :

11	STO	M
27	RCL	M
45	x< >	C
49	x< >	C

On les programmera facilement en employant la méthode du Cric, décrite dans *LIST* n° 1. Observons au passage que l'imprimante écrit : [, \,], ↑, — et T les registres synthétiques M, N, O, P, Q et ⊢.

La logique des opérations

Lorsque KASN est programmé (et sauvegardé), il est impératif de s'assurer qu'il existe un nombre **pair** d'assignations déjà effectuées. Comme notre propos est de créer un clavier synthétique complet, on effacera toute assignation préalable avec CLKEYS (zéro, comme chacun sait, étant un nombre pair bien agréable).

Ci-fait, penchons-nous quelque peu sur la logique des opérations. Pour obtenir un clavier synthétique, avec KASN, il faut indiquer le numéro de la touche à assigner et les deux codes de chaque fonction.

Les codes des touches sont bien connus (voir la fonction PASN) mais les codes des fonctions de la HP-41 C, synthétiques ou non, le sont sans doute moins.

C'est dans *LIST* n° 4 que l'on trouve la table des codes des fonctions de la HP-41 C. Avec cette table, en assemblant à volonté les codes correspon-

KASN
Programme pour HP-41 C
Auteur F. VADEZ


```
01♦LBL "KASN"      27 RCL [
02 "T?"            28 192
03 PROMPT          29 XEQ 01
04 STOP            30 XEQ "KASN"
05 "PASN"          31♦LBL 00
06 PASN            32 PROMPT
07 X<>Y            33 STOP
08 PASN            34 CLA
09 192             35 ARCL 00
10 XEQ 01          36 X<>Y
11 STO [           37 XTOA
12 ATOX            38 X<>Y
13 ATOX            39 XTOA
14 ATOX            40 1
15 ASTO 00         41 AROT
16 "C1?"           42 RTN
17 XEQ 00          43♦LBL 01
18 ATOX            44 .
19 ATOX            45 X<> c
20 ASTO 00         46 RCL Z
21 "C2?"           47 X<> IND Z
22 XEQ 00          48 X<>Y
23 240             49 X<> c
24 XTOA            50 X<>Y
25 -1              51 RTN
26 AROT            52 END
```

dants, il est possible de composer n'importe quelle fonction :
144 117 pour RCL M,
145 117 pour STO M,
etc.

A noter que les arguments M, N, etc. peuvent être choisis dans les codes 245, 246, ..., 255. Ce choix ayant pour effet d'ajouter l'indirection :
144 245 pour RCL IND M,
145 117 pour STO IND M,
etc.

Enfin, pour ce qui est du clavier synthétique, le tableau 1 précise nettement les codes des touches et des fonctions à créer.

XEQ « KASN » et le programme demande : T?, soit les numéros des deux premières touches à assigner. En effet, on effectue deux par deux les assignations. Introduire alors, chacun séparé par R/S, les numéros des deux touches.
Par exemple :
11 R/S 12 R/S

La question suivante de KASN est C1?, qui demande les deux codes de la première fonction à assigner. Par exemple, répondre : 241 R/S 67 R/S.

De même, au message : C2?, ce sont les deux codes de la seconde fonction qu'il convient de préciser :
144 R/S 124 R/S

Pour réaliser les assignations suivantes, il faut indiquer les codes de touches et de fonctions dans l'ordre où ils se trouvent dans le tableau 1.

Les deux assignations faites, KASN (insatiable) en redemande. Poursuivre à volonté et stopper l'exécution du programme lorsque toutes les assignations sont faites.

Tableau 1
Codes du clavier synthétique

Touches		Codes		Fonctions		Codes
Σ +	1/x	11	12	CRIC	RCL b	241 67 et 144 124
y^x	$\sqrt{x}$	−12	13	STO b	TONE 7	145 124 et 159 37
x^2	LOG	−13	14	RCL c	RCL d	144 125 et 144 126
10^x	LN	−14	15	STO d	RCL e	145 126 et 144 127
e^x	x < > y	−15	21	STO e	RCL ⊢	145 127 et 144 122
CLΣ	R↓	−21	22	STO ⊢	RCL M	145 122 et 144 117
%	SIN	−22	23	STO M	RCL N	145 117 et 144 118
SIN^{-1}	COS	−23	24	STO N	RCL O	145 118 et 144 119
COS^{-1}	TAN	−24	25	STO O	RCL P	145 119 et 144 120
TAN^{-1}	ASN	−25	−32	STO P	STO Q	145 120 et 145 121
x = y?	BEEP	−51	−62	μ	x < > c	4 19 et 206 125
P→R	R→P	−63	−64	x < > d	x < > M	206 126 et 206 117
FIX	SCI	−72	−73	x < > N	x < > O	206 118 et 206 119
ENG	LASTx	−74	−83	x < > P	DEL	206 120 et 4 2

Les fonctions d'un seul octet, comme μ et DEL, ont toujours pour premier code « bidon » : LBL 03 de code 4. La plupart des commandes de la HP-41 C ont un code d'un octet. Le tableau 2 les précise.

Simple comme bonjour

La fonction μ que nous venons de créer est utile en programmation synthétique pour *programmer* des chaînes de caractères. Par exemple, faire XEQ ᵀBONJOUR. Après avoir lu NON EXISTENT, en mode programme, exécuter la fonction μ. Le chiffre 3 est programmé (sans intérêt) suivi de la chaîne :
ᵀBONJOUR

L'explication est simple. Pour chercher le LBLᵀBONJOUR, la HP a stocké temporairement son nom dans le registre Q. Il y demeure. La fonction μ programme le chiffre 3 suivi d'une chaîne composée des caractères contenus dans le registre Q.

La fonction mu

Soit en X une valeur numérique quelconque, par exemple l'état des flags de la HP-41 C donné par RCL d, faire STO Q, en charge le contenu dans le registre Q.

En mode PRGM, exécuter la fonction μ décharge ce contenu, qui est programmé *inversé* (si Q contenait les codes 1234, ce sont les codes 4321 qui sont programmés). Il faut réaliser une seconde inversion pour programmer la bonne constante représentant l'état initial de X.

Se placer sur la chaîne de caractères, en mode de calcul faire SST (la constante monte en alpha, registre M). Puis, RCL M pour ranger en X sa représentation numérique, de nouveau STO Q puis, enfin, la fonction μ (en mode PRGM).

Cette chaîne de caractères programmée contient toutes les informations sur les flags de la HP-41 C au moment de l'extraction par RCL d.

Dès lors programmer :
ᵀ La chaîne spéciale
RCL M
STO d
remettra tous les indicateurs de d, y compris les flags du système, en l'état initial. C'est, en quelque sorte, résumée en trois pas, toute une série de CF - - et SF - -.

Cette méthode de stockage de constantes numériques sous forme alphabétique est précieuse :
ᵀ⊀ ⊥⊔ E ⊠ ⊠ ¯ ⊠
RCL M
est bien plus rapide que son trivial équivalent :
1 2 3 4 5 6 7 8 9

JCK

Code	Fonction
0	CAT
1	@C
2	DEL
3	COPY
4	CLP
5	R/S
6	SIZE
7	BST
8	SST
9	ON
10	PACK
11	←
12	ALPHA
13	2 - -
14	SHIFT
15	ASN

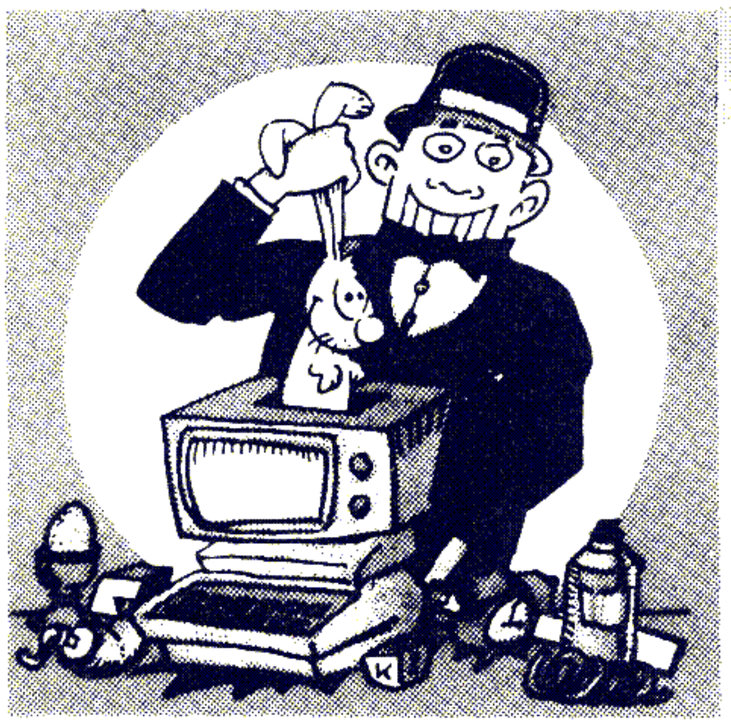
Il est recommandé de ne pas employer 2 - - et @ C sous peine de « plantage ».

Tableau 2
Codes des commandes d'un octet

Avec la fonction STO Q (que nous avons assigné au clavier, touche ASN) on pourra ranger en Q n'importe quel nombre tapé au clavier ou récupéré depuis un registre synthétique quelconque, puis le programmer en quelque sorte comme une constante.

Attention cependant, une inversion a lieu entre le nombre stocké en Q et ce même nombre programmé par la fonction μ. L'encadré ci-contre décrit la procédure à suivre pour programmer, par exemple, le stockage dans un registre synthétique d'une constante numérique.

Frédéric VADEZ
Jean-Christophe KRUST



LA BOÎTE A MALICES...

QUAND UN PROGRAMME RÉTRÉCIT...

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

LIST

L'interpréteur Basic ignore presque toujours les espaces du texte d'un programme, exception faite de ceux qui figurent à l'intérieur des chaînes de caractères, c'est-à-dire entre guillemets (ou entre guillemets et fin de ligne).

Pourquoi, dès lors, encombrer la mémoire de la machine avec ces blancs inutiles ? On a tout intérêt à écrire des programmes aussi compacts que possible, sans espaces ; cela réduit non seulement la consommation de mémoire dans l'unité centrale, mais aussi sur le disque ou sur la cassette.

Malheureusement, ce qui est bon pour la machine ne l'est pas nécessairement pour l'utilisateur. Nous sommes accoutumés à lire des mots séparés par des espaces sans lesquels un texte n'est qu'un charabia.

L'utilitaire DEPAK/BAS implante en mémoire auto-protégée une routine en binaire de 182 octets permettant, au choix, de « compacter », c'est-à-dire de supprimer les espaces inutiles, ou bien de « décompacter » en insérant des espaces dans un programme Basic quelconque. Pour « compacter », l'utilisateur entre au clavier PRINT USR (0) et tous les espaces (codes 20H/32) sont supprimés, sauf ceux qui sont inclus dans les chaînes de caractères, les commentaires (ou REM) et les lignes DATA.

Pour « décompacter », on entre PRINT USR (1) et des espaces seront, au besoin, insérés de part et d'autre de chaque deux-points et de chaque mot-clé du Basic, à l'exclusion des REM, DATA, des opérateurs arithmétiques et de la plupart des instructions dont l'argument se trouve entre parenthèses (MID$, INT, PEEK, etc.).

Les programmes ainsi reconditionnés peuvent être, à volonté, exécutés, listés, sauvés sur disque ou cassette, ou modifiés. On peut, par exemple, compacter avant la sauvegarde sur disque ou cassette, puis aussitôt décompacter


```
1 '-------------------------------------------------+
2 '  DEPAK/BAS - (C) 1984, LIST  et Roger BROUSMICHE  !
3 '  COMPACTAGE ET DECOMPACTAGE D'UN PROGRAMME BASIC  !
4 '  Configuration : modele 1 ou 3 , 16K a 48K        !
5 '                  disquette ou cassette.           !
6 '-------------------------------------------------+
10 CLS : CLEAR 500 : N=182 : U$= STRING$ (N,32)
20 V= VARPTR (U$) : A1=PEEK(V+1) : A2=PEEK(V+2)
30 AD=A1+256*A2 : M=AD-2
40 PRINT "* MEMORY SIZE ="AD : PRINT
50 PRINT "Il faut faire PRINT USR (0) pour COMPACTER
   un programme basic ," : PRINT
60 PRINT "    ou bien PRINT USR (1) pour le DECOMPAC
 TER." : PRINT
70 IF AD>32767 THEN AD=AD-65536
80 FOR I=0 TO N-1 : READ A : CS=CS+A
90 POKE AD+I,A : NEXT
100 IF CS<>24754 PRINT "ERREUR DATA..." : STOP
110 IF PEEK(16396)=201 THEN POKE 16526,A1 : POKE 16527,
   A2 :ELSE DEF USR =AD
120 '---------- protection memoire
130 B2=INT(M/256) : B1=M-256*B2
140 POKE 16561,B1 : POKE 16562,B2 : CLEAR 50
150 '----------
160 DATA 205, 127, 10, 217, 253, 42, 249, 64, 42, 164
170 DATA 64, 229, 221, 225, 126, 35, 182, 40, 90, 35
180 DATA 35, 35, 175, 182, 32, 97, 35, 221, 117, 0
190 DATA 221, 116, 1, 24, 232, 175, 217, 181, 217, 40
200 DATA 114, 126, 254, 58, 40, 16, 254, 210, 40, 12
210 DATA 254, 211, 40, 8, 254, 128, 56, 219, 254, 205
220 DATA 48, 215, 43, 126, 221, 190, 3, 40, 4, 254
230 DATA 32, 32, 12, 35, 35, 126, 254, 32, 40, 197
240 DATA 254, 147, 40, 193, 43, 253, 229, 253, 35, 253
250 DATA 229, 209, 213, 235, 183, 237, 82, 43, 229, 193
260 DATA 209, 225, 237, 184, 35, 54, 32, 24, 168, 253
270 DATA 34, 251, 64, 253, 34, 249, 64, 253, 34, 253
280 DATA 64, 217, 201, 126, 254, 147, 40, 4, 254, 136
290 DATA 32, 6, 35, 175, 182, 32, 251, 43, 254, 34
300 DATA 32, 149, 35, 62, 34, 190, 40, 143, 175, 182
310 DATA 32, 246, 43, 24, 136, 126, 254, 32, 32, 203
320 DATA 229, 229, 229, 253, 229, 225, 253, 43, 209, 183
330 DATA 237, 82, 229, 193, 225, 35, 237, 176, 225, 43
340 DATA 24, 181
```

pour obtenir une liste présentable. Mais attention : la suppression des espaces peut entraîner dans certains cas (très rares heureusement) des erreurs de syntaxe caractérisées. Ainsi le compactage de

IF F OR K THEN 1000

retournera IFFORKTHEN1000, soit la création d'un FOR incongru à cet endroit (comme quoi l'interpréteur Basic tient parfois compte des espaces...).

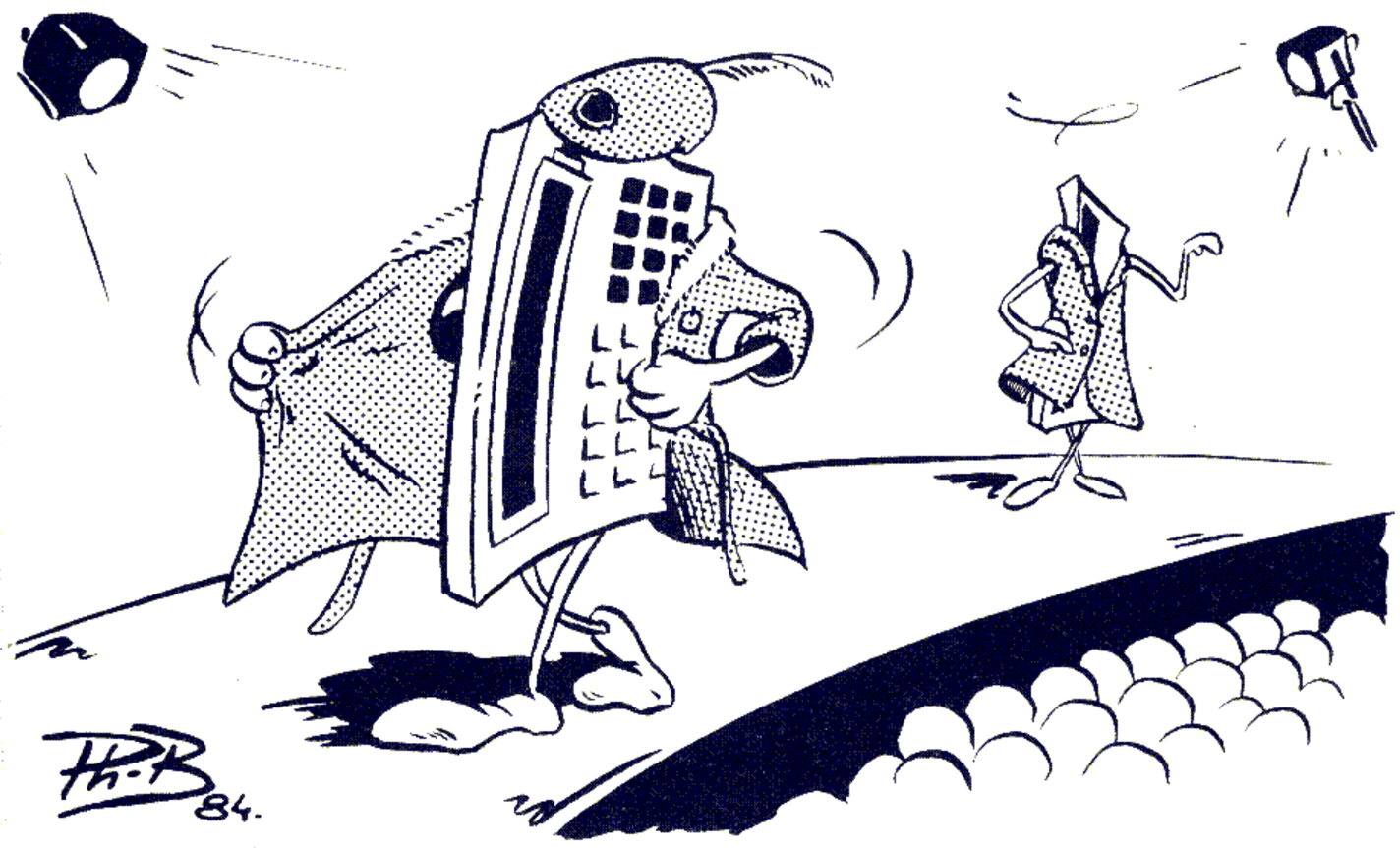
Après décompactage, les ELSE seront précédés d'un signe deux-points qui n'affecte que l'esthétique. Plus important : il ne faut **jamais** activer la routine binaire par une instruction J = USR (0) ou J = USR (1) qui peut provoquer l'altération du programme si la variable J est déjà définie en mémoire. Il est d'ailleurs beaucoup plus simple de faire PRINT USR (0) ou (1) ou (version abrégée) ?USR (0) ou (1) qui donne, en prime, la confirmation d'exécution.

Selon une technique courante, le MEMORY SIZE, quel qu'il soit, est déplacé vers le bas, et la routine est implantée dans l'espace mémoire rendu disponible. Elle reste active jusqu'à la réinitialisation du Basic, ou la redéfinition du DEF USR dans un autre programme...

Roger BROUSMICHE

PC-1251

CHANGEMENT DE MODES



On trouvera dans le tableau ci-dessous des indications sur la façon dont sont codés les différents modes à l'intérieur du PC-1251. Pour l'octet 63550, seul PEEK est opérant, mais on peut poker en 63548 et 63549.

PC-1251 : indicateurs et modes

Bit n°	7	6	5	4	3	2	1	0
Octet 63548					DE	G	P	DEF
Octet 63549					E	RAD	SHIFT	BUSY
Octet 63550						RSV	RUN	PRO
Valeur	128	64	32	16	8	4	2	1

Un exemple d'utilisation : DEF

```
10 Q = 63548
20 POKE Q,1 : PRINT "A, G, ="
30 "A" POKE Q,1 : PRINT "...SALUT..."
40 "G" POKE Q,1 : PRINT "...A..."
50 "=" POKE Q,1 : PRINT "...TOUS. "
```

La ligne 20 du programme met l'ordinateur en mode DEF sans que la touche correspondante n'ait été pressée.

Eric CHAUVEL

BONNES ADRESSES (2e partie)

Dans LIST 5, nous avions passé en revue la liste des principales variables-système stockées en page zéro de la mémoire vive. Voici maintenant ce qui concerne la page 2.

Les adresses énumérées sont exprimées en hexadécimal et elles sont le plus souvent identiques pour Oric-1 et pour Atmos. Quand elles diffèrent d'une machine à l'autre, il en est fait mention.

208 : contient l'adresse de la touche pressée. Les deux premiers bits correspondent au numéro de la ligne de la touche, les trois bits suivants au numéro de la colonne. A chaque touche est donc associée une valeur différente. Cette adresse ne détecte pas la pression des touches SHIFT et CTRL (voir ci-après).

209 : contient l'adresse de la touche spéciale (SHIFT ou CTRL) pressée, 164 ou 167 pour le SHIFT droit ou gauche, 162 pour CTRL. La valeur 56 indique qu'aucune touche n'a été pressée.

20C : 255 pour le mode majuscules et 128 pour le mode minuscules.

20E : utilisé comme compteur pour la répétition automatique des touches.

20A et 210 : contiennent respectivement le bit de colonne et le numéro de ligne de la *dernière* touche pressée. Ces octets sont très utiles quand on veut que l'action se maintienne même après pression de la touche en question. L'octet 210 a toujours son bit 7 à 1, contrairement à l'adresse 211 où l'on trouve le numéro de ligne en clair (de 0 à 7 donc).

219 : position horizontale du curseur HIRES.

21A : position verticale du curseur HIRES.

21F : 1 = HIRES, 0 = TEXT ; l'Atmos utilise cet octet pour savoir s'il doit ou non afficher les messages dans la ligne de status, de façon à ne pas altérer les graphismes HIRES, ce qui se produit sur l'Oric-1.

220 : prend la valeur 1 dans le cas d'un Oric 16 Ko (sinon 48 Ko).

228, 229 et 22A : contiennent, sur Oric-1, le saut à la routine d'interruptions (IRQ). On y trouve un JMP EC03.

22B, 22C et 22D : contiennent, sur Oric-1, le saut à la routine d'interruptions non masquables (NMI) que l'on appelle à tort « RESET ». Sur Oric-1, on y trouve JMP 430.

230 : sur Oric-1, retour d'interruptions IRQ, contient normalement un RTI que l'on peut remplacer pour ses besoins par un JMP pour détourner la fin des interruptions.

245, 246 et 247 : Atmos, saut à la routine d'interruptions IRQ (JMP EE22).

247, 248 et 249 : Atmos toujours, saut à la routine d'interruptions non masquables (JMP F882).

24A : identique à 230, mais sur Atmos.

24D : sur Atmos, vitesse de transfert cassette ; 0 = rapide, sinon lente.

25C et 25D : sur Atmos, contient après un *Verify* le nombre d'erreurs rencontrées.

261 : vecteur de saut aux routines « contrôles » (maj./min., double hauteur, etc.).

268 : ligne du curseur. Une modification de cet octet déplace effectivement le curseur, contrairement à l'octet 30 qui indique seulement sa position.

269 : numéro de la colonne du curseur.

26A : drapeaux pour différents contrôles. On trouve ainsi :

- bit 0, curseur on/off
- bit 1, affichage on/off
- bit 2, inutilisé
- bit 3, déclic du clavier
- bit 4, dernier caractère frappé ESCAPE
- bit 5, 38 ou 40 caractères par ligne
- bit 6, double hauteur

Dans un programme, pour agir sur les contrôles, il est beaucoup plus pratique de modifier directement cet octet 26A que d'utiliser des PRINT CHR$(), car on n'a pas à se soucier de l'état précédent des contrôles.

26B : couleur du fond modulo 16.

26C : couleur de l'encre.

271 : drapeau pour le clignotement du curseur (1 = normal et 0 = inverse).

272-273 : « timer » clavier (décrémenté à partir de 4).

274-275 : « timer » clignotement curseur (décrémenté à partir de 25).

276-277 : « timer » pour le WAIT, décrémenté tous les centièmes de seconde.

26D-26E : sur Oric-1 uniquement, adresse de la première ligne de l'écran, normalement BB80.

26F : sur Oric-1 uniquement, nombre de lignes à l'écran.

27F : sur Atmos, nom du programme sauvé sur cassette, terminé par un zéro, en ASCII.

293 : sur Atmos, nom du programme chargé depuis la cassette, terminé par un zéro, en ASCII.

2A9-2AA et 2AB-2AC : toujours sur Atmos, respectivement adresse du début et de la fin du programme chargé depuis la cassette.

27E (Atmos) : 0 = fichier chargé non AUTO, sinon fichier en AUTO.

27F (Atmos) : type du fichier chargé depuis la cassette ; 0 = Basic, 128 = bloc-mémoire, 64 = tableaux de variables.

2C0 : si le bit 0 est à 1, mode HIRES, sinon mode TEXT. Si le bit 1 est à 1, mode RELEASE, sinon mode GRAB. C'est cet octet qui détermine l'apparition du message « DISPLAY TYPE MISMATCH ERROR ? ». Il permet d'utiliser les commandes du mode TEXT quand on est en mode HIRES et vice-versa.

2DF : contient le code ASCII modulo 128 de la dernière touche pressée.

2E0 : contient 1 sauf si une condition de dépassement a eu lieu avec une commande graphique.

2E1 à 2E8 : registres de passage des paramètres pour les routines sonores et graphiques. Chaque paramètre occupe deux octets, en tant qu'entier, par exemple les octets 2E1 et 2E2 pour le premier, 2E3 et 2E4 pour le second, etc. Un DOKE, et non pas un simple POKE, est nécessaire pour introduire les paramètres des différentes routines. En langage-machine, ne pas oublier, même pour les paramètres inférieurs à 255, de traiter le deuxième octet, pour obtenir des valeurs correctes. A noter qu'après un MUSIC, 2E3 et 2E4 contiennent la période du signal émis.

Voilà pour aujourd'hui. Maintenant, l'Oric doit avoir moins de secrets pour vous, mais il en reste encore quelques-uns. Alors à bientôt.

Denis SEBBAG

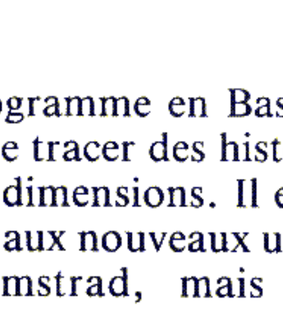
HISTOGRAMME EN TROIS DIMENSIONS

Histogramme 3D
Programme pour Amstrad
Auteur Jacques Boisgontier
Copyright *LIST* et l'auteur

Voici un programme en Basic permettant de tracer des histogrammes en trois dimensions. Il est avant tout destiné aux nouveaux utilisateurs de l'Amstrad, mais il s'adaptera aisément sur tout matériel disposant du graphisme haute résolution.

Si votre machine est dotée de l'instruction BOXF (boîte pleine), comme le MO5 ou les ordinateurs au standard MSX, le programme devient plus simple.

```
10 '---------------- histogramme 3D
20 MODE 2
30 xd=50:yd=50            ' origine
40 lg=30                  ' largeur
50 Pr=10                  ' Profondeur
60 itv=30                 ' intervalle batons
70 '
80 nh=5                   ' nombre de batons
90 '
100 h(1)=40
110 h(2)=30
120 h(3)=50
130 h(4)=10
140 h(5)=60
150 '----
160 FOR h=nh TO 1 STEP-1
170  xb=xd+(h-1)*itv                    ' origine baton
180  yb=yd+(h-1)*itv
190  FOR dx=1 TO Pr                     ' 1 baton
200   PLOT xb+dx,yb+dx+h(h)  ,1         ' Point gauche
210   IF dx>1 THEN PLOT xb+dx+lg,yb+dx,1:DRAW xb+dx+lg,yb+dx+h(h),1:GOTO 250
220   FOR dl=1 TO lg                    ' rectangle
230    PLOT xb+dx+dl,yb+dx,1:DRAW xb+dx+dl,yb+dx+h(h),1
240   NEXT dl
250   PLOT xb+dx+1,yb+dx+h(h):DRAW xb+dx+lg,yb+dx+h(h),0      ' effacement ligne
260  NEXT dx
270   PLOT xb+dx-1,yb+dx+h(h),1:DRAW xb+dx-1+lg,yb+dx+h(h),1 ' derniere ligne
280  TAG:PLOT xb+30,yb:PRINT h(h);
290 NEXT h
```



Dans l'exemple retenu, les cinq grandeurs représentées par l'histogramme sont déterminées aux lignes 100 à 140.

Jacques BOISGONTIER

FACILITONS NOS ÉCHANGES DE CASSETTES

Test de réglage de lecture de cassette
Programme pour Dai
Auteur Alain Mariatte

```
1    REM
10     REM ************************************************
20     REM ** TEST DE REGLAGE DE LECTURE DE CASSETTE **
30     REM ************************************************
40     REM
50     REM
60     REM
100    DIM A(0,0)
120    FOR I=1 TO 20
130    I$=STR$(I):L=LEN(I$):N$=LEFT$(I$,L-2):IF I<10 THEN N$=" "+N$
140    A$="    "+CHR$(137)+" "+N$+" "+"CASSETTE ADJUSTMENT TEST "+CHR$(136)
150    SAVEA A A$
160    NEXT
*
```

Comme tous les amateurs de micro-informatique, les daïstes aiment échanger leurs programmes. Cela permet l'enrichissement de leur bibliothèque de logiciels, la découverte de nouveaux trucs et astuces. Hélas, c'est souvent l'occasion d'homériques bagarres avec le potentiomètre de volume de leur magnéto-cassettes.

Pas d'échange sans réglage

D'abord, il faut en changer le réglage habituel pour l'adapter au niveau requis par la cassette du correspondant, et cela dérange les habitudes. Tant pis, c'est le prix à payer pour avoir le droit de découvrir les « trésors » certainement contenus dans ladite cassette. Avouons quand même qu'il est consternant de procéder à ce réglage, par tâtonnements successifs, en essayant de charger le premier programme. La loi universelle de Murphy nous enseigne que, dans ce cas précis, ce premier programme est justement le plus long de la cassette, si bien qu'il faut attendre de longues minutes avant que le fatidique « LOADING ERROR » final nous apprenne que le réglage n'était pas encore « tout à fait bon » !

Voilà pourquoi je propose à tous les créateurs de programmes sur Dai quelques malheureuses lignes de Basic. Elles sauvent 20 fois sur la cassette un tableau de variables avec le titre « CASSETTE ADJUSTMENT TEST » précédé d'un numéro d'ordre (1 à 20). Enregistré au début de la cassette, ce fichier sera testé par le destinataire grâce à la commande Basic

CHECK. Dès lors, il sera très facile d'ajuster les réglages du magnétophone pendant les 20 secondes de vérification de ce fichier d'essai. La numérotation renseignera sur l'avancement des travaux (un numéro manquant montrant que le réglage n'est pas absolument fiable). La mention « OK » derrière les 20 titres, au contraire, assurera une honnête probabilité de succès pour le reste de la cassette.

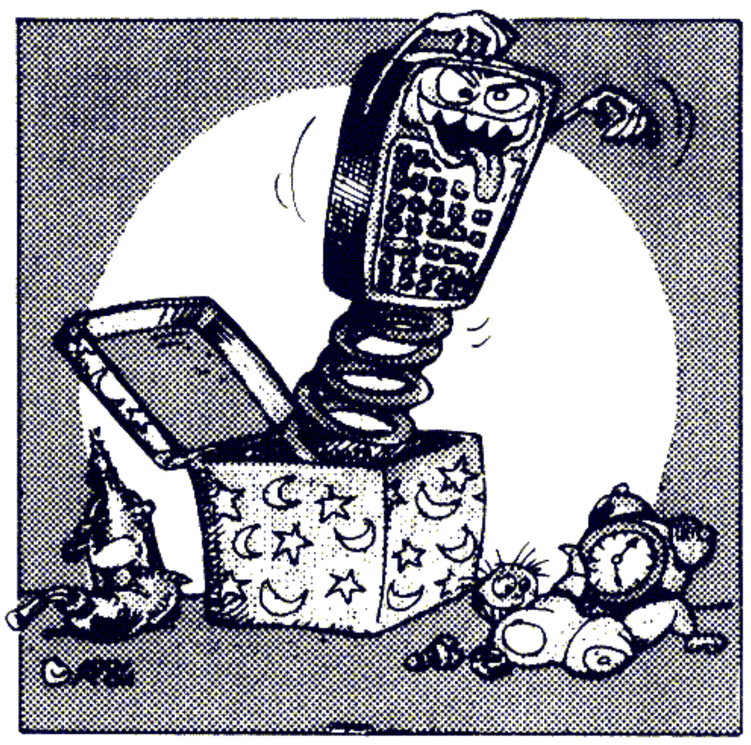
Si tous les échangeurs de programmes voulaient se donner la main... ce serait bien. Qu'ils se donnent un coup de main en acceptant de généraliser ce petit truc, et cela ne sera déjà pas si mal !

Lors d'un « check », les titres doivent apparaître à l'écran alignés comme ci-dessous et suivis de « OK » si le fichier est bon, de « BAD » dans le cas contraire.

```
 1 CASSETTE ADJUSTMENT TEST
 2 CASSETTE ADJUSTMENT TEST
 3 CASSETTE ADJUSTMENT TEST
 4 CASSETTE ADJUSTMENT TEST
 5 CASSETTE ADJUSTMENT TEST
 6 CASSETTE ADJUSTMENT TEST
 7 CASSETTE ADJUSTMENT TEST
 8 CASSETTE ADJUSTMENT TEST
 9 CASSETTE ADJUSTMENT TEST
10 CASSETTE ADJUSTMENT TEST
11 CASSETTE ADJUSTMENT TEST
12 CASSETTE ADJUSTMENT TEST
13 CASSETTE ADJUSTMENT TEST
14 CASSETTE ADJUSTMENT TEST
15 CASSETTE ADJUSTMENT TEST
16 CASSETTE ADJUSTMENT TEST
17 CASSETTE ADJUSTMENT TEST
18 CASSETTE ADJUSTMENT TEST
19 CASSETTE ADJUSTMENT TEST
20 CASSETTE ADJUSTMENT TEST
```

Alain MARIATTE

HIRISEZ VOS PROGRAMMES !



Si vous êtes utilisateur de TI-58/59, vous connaissez sans doute l'existence de la fonction HIR. Il s'agit d'une instruction multivalente, qui n'est pas révélée par les notices ou modes d'emploi, car elle est manifestement réservée aux calculs internes. En tout cas, c'est une fonction occulte, en principe interdite à l'utilisateur, mais que l'ingéniosité de quelques pionniers fouineurs a mise pratiquement à nu et à la disposition de tous.

Nous vous en rappelons l'essentiel et vous proposons (page suivante) un petit programme à titre d'exemple.

La fonction HIR, accessible uniquement en mode LRN, a pour objet de substituer aux instructions relatives aux mémoires (STO, RCL, SUM, Prd) d'autres instructions dont l'intérêt principal est de ne consommer que des pas de programme, mais pas de contrepartie de réservation de mémoire. Cette propriété permet de soulager la partition mémoire, voire de travailler en partition zéro (0 Op 17).

Six fonctions, huit mémoires

Ainsi, la fonction HIR donne accès à 6 fonctions et à 8 « mémoires » que nous appellerons registres HIR (en fait, ce sont des registres de gestion de parenthèses détournés de leur fonction). Pour introduire une instruction en mode HIR, il faut faire suivre HIR d'un nombre de deux chiffres occupant un seul pas de programme. Comme on le voit avec le tableau de correspondance ci-contre, le premier chiffre désigne la nature de la fonction, le deuxième (X) le numéro du registre. Par exemple HIR 14 est équivalent à RCL 4. Les registres adressables vont de 1 à 8, ou plutôt de 8 à 1, car il est impératif de les assigner dans l'ordre décroissant.

Comment écrire un programme en mettant à profit l'instruction HIR ? Mes mésaventures personnelles me poussent à vous conseiller vivement de l'écrire d'abord en langage clair et, une fois qu'il tourne parfaitement, de le transcrire en HIR. En effet, la mise au point d'un programme écrit directement en HIR se révèle pratiquement impossible, à tout le moins déraisonnable. N'oubliez pas en effet que si vous voulez contrôler la valeur d'une variable dans un programme écrit en HIR, la touche RCL est inopérante, que ce soit en mode calcul ou en mode LRN.

Tableau de correspondance

Instruction HIR	Instruction AOS
HIR 0X	STO X
HIR 1X	RCL X
HIR 3X	SUM X
HIR 4X	Prd X
HIR 5X	INV SUM X
HIR 6X	INV Prd X

Dans la colonne de gauche, X représente le numéro du registre interne. Dans celle de droite, il représente le numéro d'une mémoire de données.

Pour transcrire votre programme, utilisez les instructions en HIR selon les clés du tableau de correspondance, puis introduisez le programme en appliquant les règles suivantes :

• l'instruction HIR a pour code 82. Comme HIR n'existe pas au clavier,

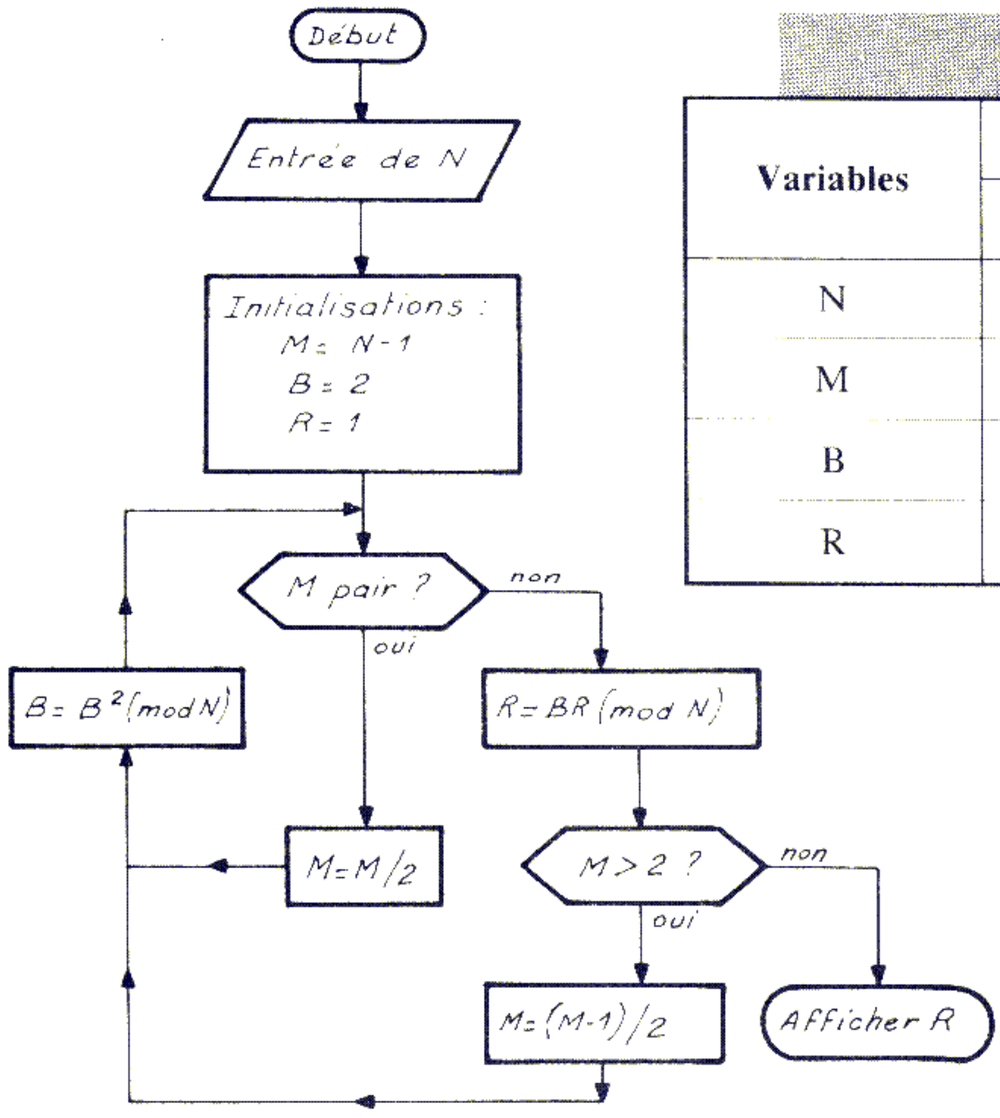
il faut entrer le code 82 en un seul pas de programme, ce qui ne peut pas se faire en introduisant successivement 8 et 2. Tapez donc STO 82 et annulez STO. Il reste 82, donc HIR. En pratique, il convient de presser successivement sur les touches STO 8 2 BST BST 2nd DEL SST .

• nous avons vu que HIR doit être obligatoirement suivi d'un code de deux chiffres (le premier étant éventuellement un zéro) également en un seul pas. Si ce code correspond à une touche, il suffit de l'actionner. Ainsi, pour introduire HIR 13, tapez HIR selon la technique exposée au paragraphe précédent, puis le faire suivre de C (13 est le code de C). Mais pour HIR 64, c'est plus délicat : vous pouvez, après HIR, introduire la série de touches correspondant au code 64 (2nd Prd 2nd Ind), soit taper STO 64 et annuler l'instruction STO, comme pour l'introduction de HIR.

N'oubliez surtout pas d'assigner les registres HIR par ordre décroissant, à partir de 8.

Pour illustrer le mécanisme de la pile de registres accessibles grâce à HIR, nous vous proposons un programme de test de Fermat écrit en AOS normal, puis transcrit en HIR.

Rappelons que le test de Fermat a pour objet de déterminer si un nombre N est composé ou non. Il suffit pour cela de calculer le reste R de la division de B^{N-1} par N, B étant un nombre quelconque, non divisible par N (on prend généralement B = 2). Si R est différent de 1, N est composé ; si R est égal à 1, N est présumé premier (avec une faible marge d'incertitude). En fait, le programme effectue le calcul de R et en affiche la valeur pour tout nombre entier N supérieur à 1 et inférieur à 10^6.



Variables	Registres	
	En AOS	En HIR
N	00	8
M	01	7
B	02	6
R	03	5

Pour faciliter la compréhension de la transcription du programme en HIR, nous le présentons sous forme de diptyque, écrit horizontalement, un peu à la manière de la disposition en Basic, avec les numéros des pas en tête de lignes. Le programme reproduit fidèlement les indications de l'organigramme ci-dessus. Notez que les multiplications et les élévations au carré en modulo N (signifiant que l'on ne retient que le reste de la division du résultat par N) sont effectuées par le même sous-programme E.

Le programme en HIR occupe deux pas de moins que le programme normal (79 pas contre 81) en raison de la compacité des instructions HIR 57 et HIR 67 par rapport à leurs homologues normaux INV SUM 7 et INV Prd 7. En outre, il tourne un peu plus vite : pour N = 1 037, le résultat (R = 815) est affiché en 30'9'' par le programme normal et en 28'5'' par le programme HIR.

Notez surtout que le programme HIR n'utilise que quatre variables et peut être introduit en partition zéro, laissant libres 401 pas sur TI-58. Mais il vous est parfaitement loisible de panacher les instructions en HIR et en AOS normal dans le corps d'un programme, à plus forte raison dans deux programmes distincts.

Pierre Ladislas GEDO

Test de Fermat

Programme en AOS normal	Transcription en HIR
(00-04) Lbl A CP STO 00	(00-04) Lbl A CP HIR 08
(05-09) − 1 = STO 01	(05-09) − 1 = HIR 07
(10-12) 2 STO 02	(10-12) 2 HIR 06
(13-15) 1 STO 03	(13-15) 1 HIR 05
(16-17) GTO D	(16-17) GTO D
(18-23) Lbl B 1 INV SUM 01	(18-22) Lbl B 1 HIR 57
(24-29) Lbl C 2 INV Prd 01	(23-27) Lbl C 2 HIR 67
(30-35) RCL 02 SBR E STO 02	(28-33) HIR 16 SBR E HIR 06
(36-46) Lbl D RCL 01 ÷ 2 = INV Int x=t C	(34-44) Lbl D HIR 17 ÷ 2 = INV Int x=t C
(47-52) RCL 03 SBR E STO 03	(45-50) HIR 15 SBR E HIR 05
(53-59) RCL 01 − 2 = x ⩾t B	(51-57) HIR 17 − 2 = x ⩾t B
(60-62) RCL 04 R/S	(58-60) HIR 15 R/S
(63-80) Lbl E × RCL 02 − (CE ÷ RCL 00) Int × RCL 00 = INV SBR	(61-78) Lbl E × HIR 16 − (CE ÷ HIR 18) Int × HIR 18 = INV SBR

L'utilisation de HIR et des registres internes de la calculatrice permet ici d'économiser 4 mémoires de données.

ALICE 32 ET ALICE 90

QUELQUES ADRESSES A L'USAGE DES EXPLORATEURS

La petite Alice de Matra dispose désormais d'une bibliothèque d'environ une dizaine d'ouvrages destinés aux amateurs débutants. Mais ces ouvrages ont été écrits pour l'Alice d'origine, et, si les programmes Basic ne posent aucun problème, la moindre utilisation d'un PEEK ou d'un POKE risque de perturber sérieusement le fonctionnement du système.

En effet l'Alice initiale disposait d'une mémoire vidéo située de l'adresse 16384 à l'adresse 16896 (512 octets, pour 16 lignes de 32 caractères). Beaucoup de programmes, même sur des livres portant la mention « ALICE 90 », explorent cette mémoire d'écran, qui ne correspond à rien sur les nouveaux appareils. Ceux-ci sont en effet équipés d'un circuit vidéo spécifique EF 9345 qui possède sa propre mémoire interne, et auquel on accède depuis le microprocesseur 6803 à l'aide d'une série de registres. Ce circuit offre d'ailleurs des possibilités non exploitées par le Basic, par exemple des caractères accentués, des possibilités de double largeur ou double hauteur, de clignotement, etc.

Les adresses de la nouvelle Alice

Sans entrer dans les détails, voici un certain nombre de POKEs conduisant sur la nouvelle Alice à des résultats assez spectaculaires.

POKE 12316,n permet de définir une fenêtre écran dans laquelle le programme sera listé et exécuté, le début de la fenêtre se trouvant à la nième position à partir de la droite.

POKE 12305,k définit la marge droite de la fenêtre.

POKE 12304,xx affiche des caractères très spéciaux en début de fenêtre.

POKE 12315,m agit sur la hauteur de la fenêtre.

POKE 12928,xx ou POKE 12929,yy provoquent des anomalies très intéressantes (je vous laisse les découvrir).

POKE 12930,xx change l'allure du curseur.

POKE 12307,p donne lui aussi des résultats étranges, selon la valeur de p : caractères accentués, agrandis, démarrage de l'imprimante, RESET inactif, etc.

Les cases 12317, 12402, 36142 réservent aussi quelques surprises aux hardis explorateurs. Mais le plus intéressant reste bien sûr une petite visite dans les arcanes du chip vidéo, implanté à des adresses allant de 48928 à 48943. Quelques petites boucles pour explorer ces cases ne manqueront pas de vous ramener quelques bonnes prises : nous attendons vos trouvailles.

Jacques DECONCHAT

SPECTRUM

LE BASIC SINCLAIR SERAIT-IL RÉCURSIF ?

On connaissait déjà les étonnantes possibilités du Basic Sinclair en matière d'évaluation d'expressions arithmétiques et d'utilisation des fonctions logiques, mais d'ici à trouver ce Basic récursif... Et pourtant, une utilisation judicieuse des diverses particularités de ce Basic permet vraiment de se poser la question sur son caractère récursif.

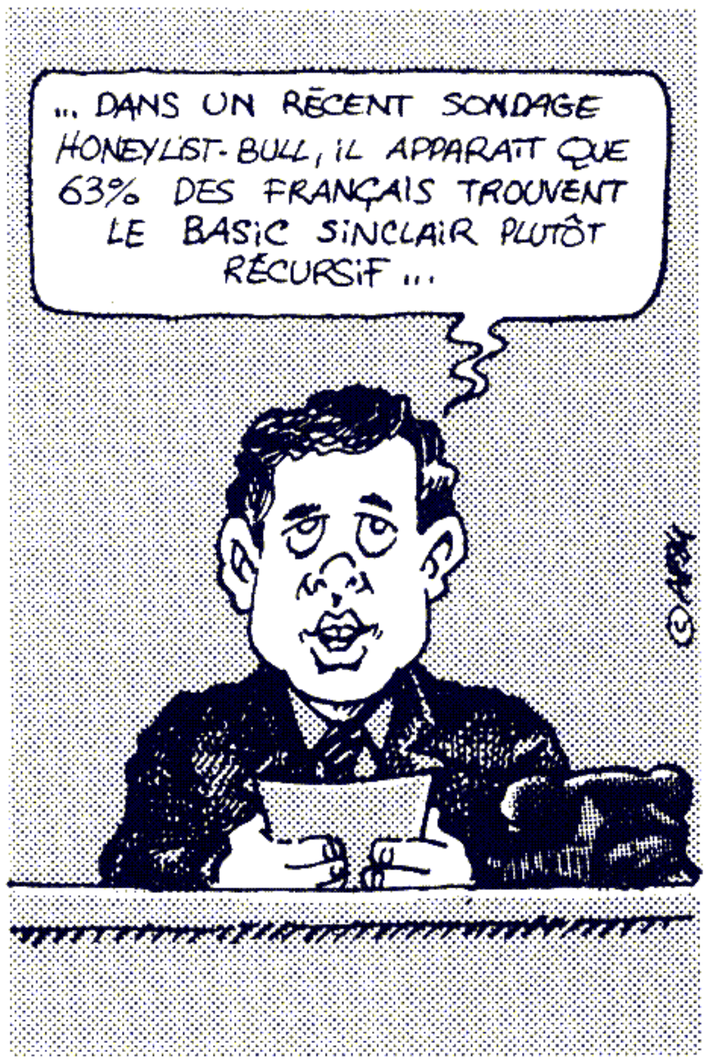
Prenons le célèbre exemple du calcul de N!. En mathématiques, la fonction factorielle est définie par récurrence :

$n! = n \times (n-1)!$ et $0! = 1$

Ceci permet d'écrire

$1! = 1 \times 0! = 1 \times 1 = 1$
$2! = 2 \times 1! = 2 \times 1 = 2$
$3! = 3 \times 2! = 3 \times 2 = 6$
$4! = 4 \times 3! = 4 \times 6 = 24$, etc.

Les langages informatiques récents permettent la définition récursive d'une fonction, mais ce n'est possible en Basic que par la gestion d'une pile, qui va « entasser » les appels successifs de la fonction et les restituer au fur et à mesure pour le calcul. Mais essayez donc sur votre Spectrum le curieux petit programme ci-dessous, et regardez le résultat.



Le mystère de cette récursivité ?

```
10 REM definition recursive de factorielle n
20 PRINT "DONNEZ UN ENTIER :";
30 INPUT N
40 PRINT N
50 DEF FN F(N)=VAL (("N*FN F(N-1)" AND N>=1)+("1" AND N=0))
60 PRINT : PRINT "F(";N;") = ";FN F(N)
```

Prenons un exemple qui illustre l'une des possibilités d'utilisation de la fonction AND sur le Spectrum :
LET A$ = ("AU REVOIR" AND TEST < = 5) + ("BONJOUR" AND TEST > 5). Si TEST < = 5 alors A$ = "AU REVOIR", sinon A$ = "BONJOUR"

Regardons maintenant la ligne suivante :
DEF FN f(n) = (n × FN f(n-1) AND n > = 1) + (1 AND n = 0)

Elle peut se lire, en décomposant :
IF n > = 1 THEN LET FN f(n) = n × FN f(n-1)
IF n = 0 THEN LET FN f(n) = 1

Malheureusement, l'écriture précédente ne fonctionnera pas telle quelle car, les deux membres de part et d'autre du AND étant des expressions numériques, le premier (avant le AND) sera d'abord évalué, et non la condition qui suit le AND. C'est ici qu'intervient une deuxième astuce : on transforme l'expression qui précède le AND en chaîne alphanumérique, pour se replacer dans les conditions du premier exemple. C'est donc seulement après l'évaluation de l'expression que l'on calculera la valeur numérique, à l'aide de la fonction VAL.

Pour les passionnés du Spectrum, voilà une merveilleuse occasion d'utiliser la fonction VAL$ pour définir par un procédé analogue des fonctions récursives alphanumériques. Une seule difficulté : faire très attention aux guillemets.

Frédéric BLONDIAU

ALGORITHME

DES DÉCIMALES A FOISON

Si vous ordonnez à votre ordinateur de calculer le nombre 987 654321/123456789, il vous répondra quelque chose comme : 8,000000073.

C'est relativement exact, mais cela ne fournit qu'une précision très limitée. Pas moyen de savoir réellement quelle est la neuvième décimale de ce rationnel : sans doute un 3, mais ce n'est pas sûr. Et encore, ce genre de résultat a été obtenu sur un ordinateur de poche (Sharp 1500 en l'occurrence) ; un TRS-80 ou un Apple en simple précision sont beaucoup moins fameux.

La précision multiple

La solution consiste en un calcul en précision multiple. Il arrive que l'on ait besoin de ce genre de méthode (par exemple à l'épreuve pratique de l'Ecole normale supérieure de la rue d'Ulm de 1984). Versons donc dans la « boîte à malices de *LIST* un utilitaire bien commode donnant autant de décimales que l'on veut d'un nombre a/b (aux limites près de la mémoire, ce qui laisse une bonne marge si l'on dispose de quelques Ko).

Multiplions d'abord a ou b par une puissance convenable de 100 pour avoir l'inégalité : $0 \leqslant a < 100\ b$ et définissons deux suites d'entiers (a_n) d'une part, et (q_n) d'autre part par les relations de récurrence simples : $a_0 = a$, $q_n = \text{INT}(a_n/b)$, $a_{n+1} = 100(a_n - bq_n)$.

On constate aisément que l'on a toujours $0 \leqslant a_n < 100\ b$. Le nombre rationnel a/b s'écrit alors $(q_0, q_1\ q_2 \ldots q_n \ldots)$, où les entiers successifs q_n sont écrits avec exactement deux chiffres décimaux (éventuellement nuls) chacun.

Donnons un exemple : pour calculer le développement de 111/61, on part de b = 61, a = 111, et on trouve les résultats suivants (en décimal) :

$a_0 = 111$	$q_0 = 1$
$a_1 = 5000$	$q_1 = 81$
$a_2 = 5900$	$q_2 = 96$
$a_3 = 4400$	$q_3 = 72$
$a_4 = 800$	$q_4 = 13$
$a_5 = 700$	$q_5 = 11$
$a_6 = 2900$	$q_6 = 47$
$a_7 = 3300$	$q_7 = 54\ldots$

Ainsi 111/61 = 1,81 96 72 13 11 47 54 09 83 60... (ne pas oublier un zéro devant le 9). Dans un cas simple comme celui-ci, un programme Basic très court convient avec l'instruction INT (partie entière), par exemple :

```
10 INPUT A,B
20 C = 100
30 Q = INT (A/B)
40 PRINT Q
50 A = C*(A-Q*B)
60 PRINT A
70 GOTO 30
```

Mais ceci tient évidemment à ce que les entiers de départ a et b étaient de petits nombres, pour lesquels le calcul de INT (A/B) pouvait se faire exactement ; ceci n'aurait pas convenu pour notre exemple initial dans lequel a = 987654321, b = 123456789, encore moins pour des nombres d'une trentaine de chiffres chacun.

Supposons par exemple qu'aucun des entiers a et b n'ait plus de trente chiffres. On fera éventuellement précéder ces entiers d'un zéro pour pouvoir les découper exactement en tranches de deux chiffres. La solution consiste alors à préparer par exemple quinze cases mémoires B(I), pour I de 1 à 15, seize cases mémoires A(I), pour

PB-700

TOP SECRET

Contrairement à ce qui est écrit dans le manuel du PB-700, il est possible d'insérer dans un programme les fonctions NEW et NEW ALL (de même que LIST et SYSTEM). Ainsi, pour empêcher les petits curieux d'accéder à des « programmes top secrets », il suffit d'introduire au début de ceux-ci, les lignes :

```
1  FOR A=0 TO 15
2  A$=INKEY$ : IF A$="J" THEN 10
3  NEXT A
4  NEW ALL
10 REM-PROGRAMME-
```

Les « petits curieux » qui lancent de tels programmes ne trouveront rien : tout a été effacé. Ils disposaient bien d'un petit peu plus d'une seconde pour appuyer sur J, mais ils ne connaissaient pas le code.

Alors, pensez à sauvegarder vos programmes sur cassette ou sur papier avant de laisser le PB-700 traîner entre toutes les mains.

Pierrick GRESLIN

Quotient de deux entiers à une précision donnée

Programme en Basic Microsoft (TRS-80)

Auteur André Warusfel


```
10        REM ENTREE DES DONNEES
20        REM
30 CLEAR 1000:DEFINT A-Z
40 PRINT"NOMBRE MAXIMUM DE TRANCHES"
50 INPUT "DE DEUX CHIFFRES ";T
60 DIM A(T):DIM B(T):DIM C(T):N=0
70 PRINT"ENTREZ LES CHIFFRES DE  A  A PARTIR"
80 PRINT"DE LA GAUCHE PAR TRANCHES DE DEUX:"
90 PRINT:A(0)=0:B(0)=0
100 FOR I=1 TO T
110     PRINT"A(";I;")= ";:INPUT A(I)
120 NEXT
130 PRINT
140 PRINT"ENTREZ LES CHIFFRES DE  B  A PARTIR"
150 PRINT"DE LA GAUCHE PAR TRANCHES DE DEUX:"
160 PRINT
170 FOR I=1 TO T
180     PRINT"B(";I;")= ";:INPUT B(I)
190 NEXT
200 PRINT
210       REM
220       REM CALCUL DE Q(N)
230       REM
240 K=INT((100*A(0)+A(1))/B(1))+1
250 IF K>=100 THEN K=100
260       REM
270       REM ESSAIS SUCCESSIFS DE VALEURS DE K
280       REM
290 K=K-1
300 FOR I=0 TO T
310     C(I)=K*B(I)
320 NEXT
330       REM
340       REM TOILETTAGE DU TABLEAU  C
350       REM
360 FOR I=T TO 1 STEP-1
370     R=INT(C(I)/100)
380     C(I-1)=C(I-1)+R
390     C(I)=C(I)-100*R
400 NEXT
410       REM
420       REM COMPARAISON DES TABLEAUX  A  ET  C
430       REM
440 I=-1
450 I=I+1:IF I=T+1 THEN 610
460 IF A(I)=C(I) THEN 450
470 IF A(I)<C(I) THEN 290
480 PRINT"Q(";N;")= ";K
490       REM
500       REM CALCUL DU NOUVEAU RESTE PARTIEL
510       REM
520 FOR I=T TO 0 STEP-1
530     A(I)=A(I)-C(I)
540     IF A(I)>=0 THEN 560
550     A(I)=A(I)+100:A(I-1)=A(I-1)-1
560 NEXT
570 FOR I=1 TO T
580     A(I-1)=A(I)
590 NEXT
600 A(T)=0:N=N+1:GOTO 240
610 PRINT"Q(";N;") ";K
620 PRINT"DIVISION TERMINEE EXACTEMENT":END
```

I de 0 à 15, seize cases mémoire C(I), pour I de 0 à 15, puis de mettre :

- les chiffres de b, par paquets de deux consécutifs, dans les mémoires B(1), B(2), B(3), etc.
- les chiffres de a, par paquets de deux consécutifs, dans les mémoires A(1), A(2), A(3), etc.
- des zéros dans les cases A(0) et les cases inoccupées (s'il en existe) provenant du fait que a ou b n'ont pas nécessairement une trentaine de chiffres, mais peut-être vingt-huit, vingt-quatre, etc.

On peut considérer que ces mémoires contiennent alors chacune l'équivalent d'un « chiffre » dans un système de numération dont la base serait cent et non dix. L'algorithme est exactement celui que l'on emploie habituellement dans une division « à la main » comme on l'apprend à l'école primaire, avec ses essais successifs, ses restes partiels, etc. Très connu de chacun d'entre nous, il est assez malaisé à expliciter complètement : le mettre sur machine est un bon exercice.

Nous supposerons qu'à un instant donné les mémoires A(I), pour I allant de 0 à 15, contiennent le nombre a_n ; b reste toujours, quant à lui, dans les B(I) pour I de 1 à 15. Soient x, y et z les contenus respectifs de A(0), A(1) et B(1) ; pour première valeur plausible de q_n, on prendra d'abord k, plus petit des deux nombres 99 et partie entière de (100x + y)/z. Les mémoires C(1) à C(15) vont alors recevoir les contenus des mémoires B(1) à B(15) multipliés chacun par k. Une opération de « nettoyage » par dégraissages successifs conduira, en partant de C(15), à débarrasser toute mémoire C(I) de multiples de 100 en les reportant — comme avec des retenues classiques en calcul décimal — dans les cases de numéro inférieur. C'est ainsi qu'il est très possible que la case C(0) en vienne à contenir quelque chose provenant de C(1).

Reste alors à comparer a_n, emmagasiné dans les A(I) et le produit bk, gardé dans les C(I). Si bk est strictement supérieur à a_n, c'est que k est trop grand et il faut essayer $k-1$ et ainsi de suite jusqu'à trouver q_n, plus grand entier h tel que $bh \leqslant a_n$. Dans les cases A(I) figure maintenant le reste partiel $a_n - bq_n$. Comme dans une division habituelle, on « descend une nouvelle tranche » en faisant translater le contenu des A(I) vers les A(I – 1), ce qui équivaut à multiplier par 100 et donne bien la relation de récurrence $a_{n+1} = 100(a_n - bq_n)$.

Voici un programme possible. Il demande le nombre maximum de tranches de deux chiffres — par exemple quinze — nécessaires pour représenter a et b, puis ces deux nombres que l'on doit entrer au clavier pour emplir dans l'ordre croissant les mémoires A(1), A(2)... puis B(1), B(2) et ainsi de suite. Il imprime successivement q_0, q_1, q_2, ... sans fin : c'est à l'opérateur de décider l'arrêt du processus.

Le Basic Microsoft utilisé (ici sur TRS-80) est très classique. On a cherché à structurer au maximum le programme, ce qui le rend sans doute un peu long, mais relativement clair à relire. On pourra essayer l'exemple donné au début de ce petit article (a = 987654321, b = 123456789), pour en calculer une vingtaine de décimales ; on trouve : 8, 00 00 00 07 29 00 00 06 63 39 00 06 03 68...

On y voit surtout beaucoup de zéros ! Ce qui ne veut pas dire que ces décimales soient si simples que cela à déterminer sans machine.

André WARUSFEL

LE JOURNAL DES AMATEURS DE PROGRAMMATION

COMPLETEZ VOTRE COLLECTION !

Il vous manque d'anciens numéros de LIST ? Utilisez le bon ci-contre pour les commander. Si vous en choisissez 3, vous n'en paierez que 2 !

N° 1 : Le Basic du MO5 ● Trois logiciels d'aide à la programmation pour C.64, TI 99/4A et TO7 ● Le point sur le ZX Spectrum ● La programmation synthétique de la HP-41C ● Une procédure tri en Pascal et Basic ● L'histoire des langages : l'Assembleur ● Logo, un langage puissant ● Comment utiliser Forth ● L'instruction FOR... NEXT ● Des auxiliaires précieux : les "Booléens" ● Dix programmes courts pour tester le Basic de votre machine ● Trucs et astuces sur X-07, ZX 81, Apple II, PC-1500, TO7, PB-100, Alice, FX-702 P, PC-1212 ●

N° 2 : Le Basic du HR-71 B ● Dopez votre Oric ● Un éditeur-assembleur pour C.64 ● 16 fonctions supplémentaires sur PC-1500 ● Des applications du langage machine pour processeur Z80 et 6502 ● Copie d'écran sur PB-700 ● Pourquoi diable programmez-vous ? ● L'histoire des langages : le Fortran ● Travaux dirigés en Logo et Forth ● Pascal : filtrez vos entrées ● Des ficelles pour TO7, TI-66, TI 99/4A, PB-100, CBM 4000 et 8000, Vic, C.64, HP-41C ●

N° 3 : Le Basic du BBC ● Un compilateur pour Spectrum ● Création graphique sur TO7 ● Un macro-assembleur pour C.64 ● La tablette graphique Koala Pad ● Tri Quicksort en Basic ● Pascal convertit les majuscules en minuscules ● L'histoire des langages : le Cobol ● Un mini-moniteur pour PC-1251 ● Tic-Tac-Toe : le programme imbattable ● La fonction Modulo sur HP-41 ● Un détecteur de nombres premiers ● Trucs et astuces sur Atari, FX-602 P, FX-702 P, PET/CBM, TI-57, ZX Spectrum ●

N° 4 : Les Basics du Coco 2 et des Atari ● Le compilateur "Turbo Pascal" pour Apple II ● Programmez votre TO7 en Assembleur ● Une table de conversion décimal-hexadécimal ● Interpréteur ou compilateur ? ● Quand le Basic devient récursif ● Conversion de chaînes en Pascal ● Haro sur les octets perdus de la HP-41C ● L'histoire des langages : le Pascal ● 20 décimales pour vos logarithmes ● En conversation avec Logo ● Rapidité d'exécution : 16 machines testées ● Des ficelles pour Alice, C.64, DAI, MO5, Oric, PC-1211, 1251, 1500, TO7, TRS-80 ●

N° 5 : Le Basic du standard MSX ● Le Basic et l'Assembleur d'Alice 90 ● Forth pour Spectrum ● Un assembleur pour le DAI ● Initiation au Basic sur TO7 ● Simon's Basic : 100 instructions supplémentaires pour C.64 ● Protégez vos messages sur X-07 ● Sous la loupe : un jeu d'envahisseurs pour PB-700 ● Programmez un écran graphique sur Oric et Apple ● Formatage des nombres en Pascal ● Paramétrez vos périphériques ● PC-1251 en musique ● Le Basic de 30 machines au banc d'essai ● Trucs et astuces sur Oric, PC-1251, TI-57 LCD, TO7, DAI, Apple II, ZX 81, MSX, HP-41C, Spectrum ●

Prix d'un numéro :
France : 20 FF
Belgique : 154 FB
Suisse : 6 FS
Canada : 2.95 $C
Autres Pays : 25 FF

Par avion :
Afrique francophone : 28 FF
Amérique, autre Afrique,
Océanie : 34 FF
Asie : 40 FF

Belgique :
SOUMILLION
28, av. Massenet
1190 Bruxelles
Tél. : 02/345 91 92
Versement à la SGB
210-040 5835-39

Suisse :
EDIMONT
19, r. du Grand-Mont
CH 1052
Le Mont sur Lausanne
Tél. : 21/32 15 65
Versement Caisse
d'Epargne et de Crédit
10.432 Le Mont CH 1052
compte courant
n° 650 156-7

Canada :
LMPI
9345 rue de Meaux
Saint-Léonard
Québec
H1R 3H3
Tél. : (514) 328 21 43

France et autres pays :
5, place du Colonel-Fabien
75491 Paris Cedex
Tél. : (1) 240 22 01

BON DE COMMANDE

à retourner à LIST
- Service Numéros -
5, place du Colonel-Fabien
75491 PARIS CEDEX 10*

Je désire recevoir les numéros de **LIST** cochés ci-dessous.

1	2	3	4	5

au prix de **20 FF*** l'unité.

Pour 3 anciens numéros, je bénéficie du prix de **40 FF** au lieu de **60 FF**.*

Je joins mon règlement indispensable à l'ordre de **LIST**.

Nom _____

Prénom _____

Adresse _____

_____

Code Postal _____

Ville _____

Pays _____

*Etranger : voir ci-contre.

LES JEUX ET CASSE-TÊTE INFORMATIQUES

de Thierry CHAMORET

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

16

Opérations DIV et MOD

Les opérateurs MOD et DIV ne sont pas présents dans tous les Basic. Ils s'appliquent à deux variables. Ainsi, A MOD B donne le reste de la division entière de A par B. Par exemple, 11 MOD 3 est égal à 2.

L'opération A DIV B retourne, quant à elle, le résultat de la division entière de A par B. 11 DIV 3 est donc égal à 3.

Connaissant ces opérateurs, vous trouverez, sans l'aide d'un ordinateur, les résultats des opérations suivantes :

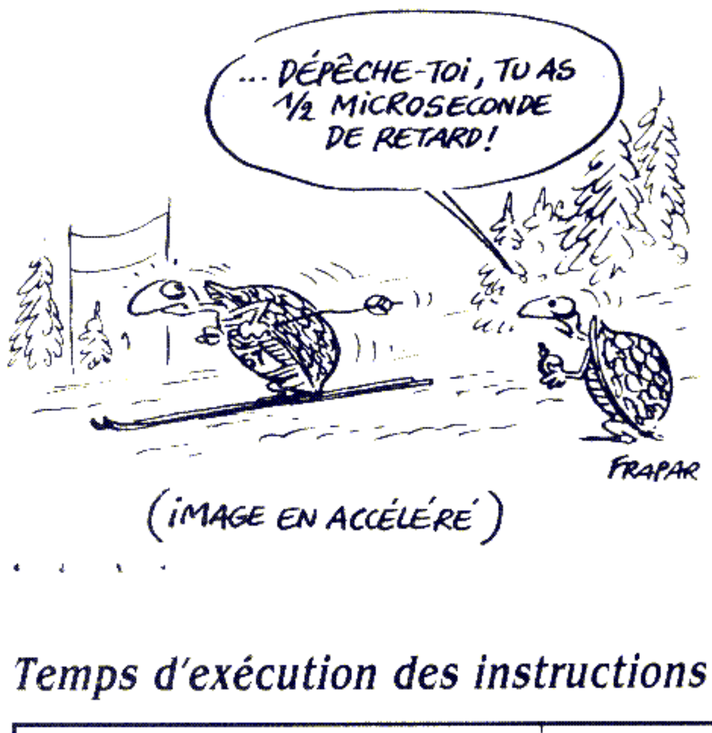
1. A MOD 1
2. A MOD A
3. A MOD -1
4. A MOD -A
5. A MOD 0
6. A DIV 1
7. A DIV A
8. A DIV -1
9. A DIV -A
10. A DIV 0

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

17

Construire la matrice unité

Le calcul matriciel définit des opérations non pas sur des nombres, comme en algèbre, mais sur des tableaux de valeurs, appelés *matrices*.



L'élément neutre de la multiplication des nombres vaut 1. Celui de la multiplication des matrices est la *matrice unité*, matrice carrée composée de uns dans la diagonale principale et de zéros ailleurs :

$$\begin{bmatrix} 1\,0\,\ldots\,0\,0 \\ 0\,1\,\ldots\,0\,0 \\ \ldots\ldots\ldots \\ 0\,0\,\ldots\,1\,0 \\ 0\,0\,\ldots\,0\,1 \end{bmatrix}$$

Une valeur placée à la ligne L et à la colonne C est égale à 1 si L = C et à 0 sinon.

L'initialisation d'un tel tableau dans un programme informatique n'est pas une chose bien complexe. Le programme suivant le réalise très bien (même dans le cas d'une matrice rectangulaire) :

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

Temps d'exécution des instructions rencontrées

Instructions	Exemples	Temps d'exécution
Passage sur une boucle for... next	for L = 1 to N / next L	850 microsecondes par itération
Affectation d'une valeur à l'élément d'un tableau	A(L,C) = 0 ou A(L,C) = 1	1 130 microsecondes
Comparaison de deux valeurs	if L = C then / else	810 microsecondes

Les temps d'exécution, en microsecondes (millionièmes de seconde), relevés sur notre ordinateur sont réunis dans le tableau des "Temps d'exécution des instructions rencontrées".

Par exemple, avec MAXLIG = 100 et MAXCOL = 100, le temps d'exécution de notre programme se calcule de la façon suivante :
100 fois la ligne 10, (100 × 100) fois la ligne 20, (100 × 100) fois la comparaison de la ligne 30 et (100 × 100) fois l'affectation de la ligne 30. Soit en microsecondes : (100 × 850) + (10 000 × 850) + (10 000 × 810) + (10 000 × 1 130), ce qui fait un total de 27 985 000 secondes, soit environ 28 secondes.

Comme nous le constatons, le temps de calcul pour initialiser notre matrice est loin d'être négligeable. Comme il est toujours préférable en informatique d'obtenir des programmes plus rapides, vous pourrez écrire un autre programme d'initialisation qui, en fonction des temps de calcul indiqués, s'exécutera en moins de 20 secondes.

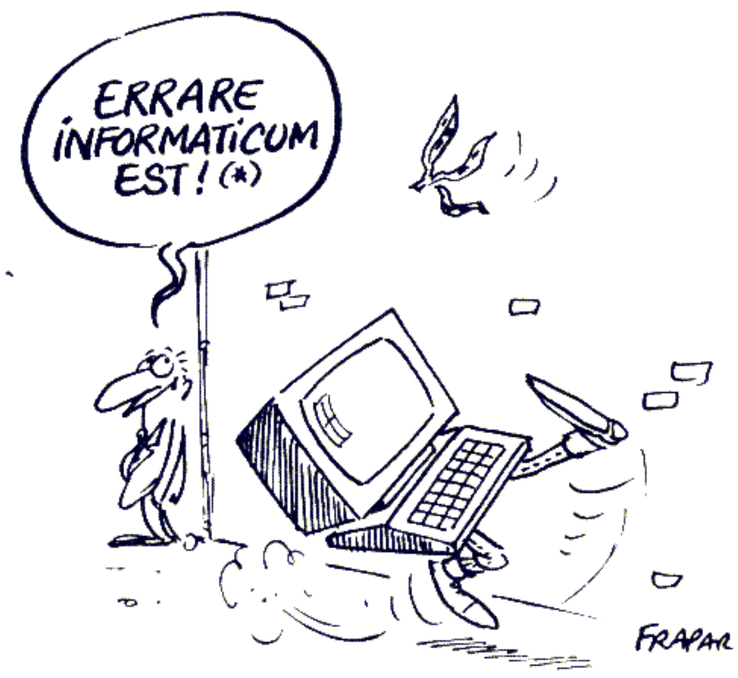
18

La fausse égalité

L'informatique n'est pas une science exacte. Mais toutes les erreurs rencontrées ne sont pas des erreurs de programmation. Contrairement à ce que l'on nous a toujours appris, l'ordinateur fait, lui aussi, des erreurs. Pour illustrer cette affirmation, vous trouverez certainement des valeurs de A, B et C qui ne vérifient pas l'égalité :

(A * B) * C = A * (B * C)

Et ce, sans qu'il y ait de débordement ou de sous-débordement dans les calculs.



(*) IL EST INFORMATIQUE DE SE TROMPER.

SOLUTIONS DU NUMÉRO PRÉCÉDENT

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

▸▸13

Un test impossible

Remplacer le test "if (A/B) < 1" par "if A < B" est une opération dangereuse : ces deux tests ne sont pas équivalents. En effet, dans le cas où l'une des valeurs A ou B est négative, ils ne donnent pas les mêmes résultats.

En prenant par exemple, A = 2 et B = −4, le premier test est vérifié (A/B vaut −0,5 et −0,5 < 1 est vrai) alors que le second ne l'est pas (2 < −4 est faux !).

Dans un programme qui utilise des valeurs négatives pour B, le second test ne renvoie donc pas les mêmes résultats que le premier. C'est pourquoi le remplacement de if (A/B) < 1 doit être effectué avec prudence. Plusieurs solutions peuvent être envisagées :
if (A < B) = (B > 0)
if ((A < B) and (B > 0)) or ((A > = B) and (B < = 0))
où *and* et *or* sont des opérateurs logiques.

D'autres solutions peuvent encore être découvertes...

▸▸14

Toujours plus court

La séquence d'instructions :
10 for I = 1 to N
20 if I − 2 * int (I/2) = 0
30 then E (I) = int (I/2) + 1
40 else E (I) = int (I/2)
50 next I
a pour objet de mettre la valeur 1 dans les cases 1 et 2 du tableau E, 2 dans les cases 3 et 4, 3 dans les cases 5 et 6, etc.

Elle peut être programmée beaucoup plus simplement, de la façon suivante :
10 for I = 1 to N
20 E (I) = int ((I + 1)/2)
30 next I

▸▸15

Une autre fonction aléatoire

Il existe deux méthodes pour obtenir un nombre aléatoire compris entre A et B à partir de la fonction RANDOM.

La première consiste à diviser la valeur de RANDOM par 32768. Ainsi, nous obtenons des nombres aléatoires compris entre 0 et 1, de la même façon que la fonction RND. Nous avons donc la formule :

(B − A + 1) * RANDOM / 32768.0 + A

La seconde méthode utilise l'opérateur *mod* qui fournit le reste de la division entière de deux nombres. Le reste de la division de RANDOM par (B − A + 1) est une valeur qui est toujours comprise entre 0 et (B − A). En ajoutant la valeur de A à ce reste, nous obtenons des nombres aléatoires entre A et B. Cette expression s'écrit : A + RANDOM mod (B − A + 1)

En analysant de près le comportement de cette dernière formule, on constate qu'elle ne donne pas, sur une longue série de valeurs, des nombres aléatoires uniformément distribués. En prenant A = 0 et B = 10000, la moyenne des nombres aléatoires n'est pas de 5000, mais de 4694,047. Cette dernière valeur théorique se confirme en faisant un grand nombre de tests. Sur cent millions de tirages que nous avons réalisés, nous trouvons par exemple une moyenne de 4694,748 pour la seconde formule, alors que la première obtient une moyenne de 4999,612, ce qui est très proche de 5000.

DANS QUEL NUMÉRO ÉTAIT-CE ?

INDEX récapitutalif des articles publiés dans les cinq premiers numéros de LIST.

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

Pour vous aider à retrouver en un clin d'œil les textes et les programmes qui vous intéressent, nous vous proposons aujourd'hui cet index récapitulatif.

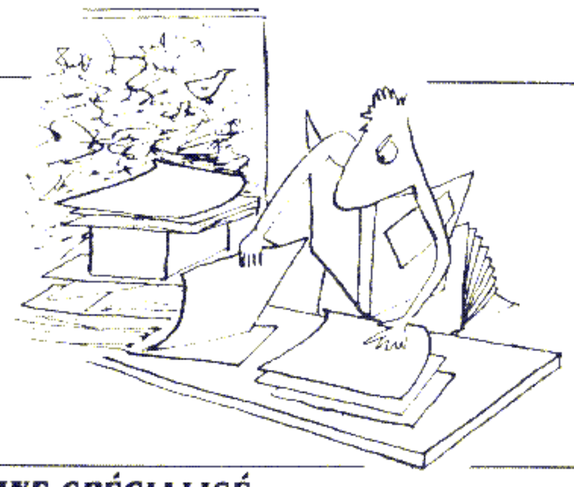
Comment fallait-il répertorier ces articles ? Nous avons distingué six grands axes : programmation, système, langages, coups d'œil sur des logiciels et essais de différents Basics. A l'intérieur de ces grandes catégories, nous avons de nouveau classé, le plus souvent par machines ou par langages.

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

Maryse GROS
Eliane GUEYLARD

PROGRAMMATION

PROGRAMMATION



BASIC

LANGAGE-MACHINE

PROCÉDURES PASCAL

LANGAGE MACHINE SPÉCIALISÉ

UTILITAIRES

SYSTEME

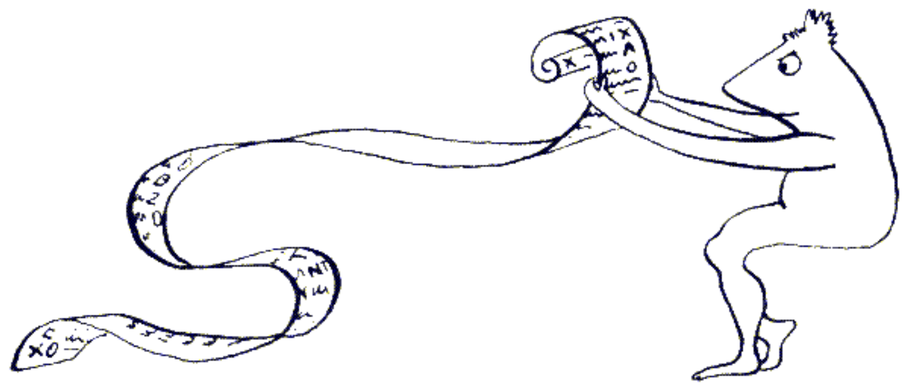
LANGAGES

L'HISTOIRE DES LANGAGES

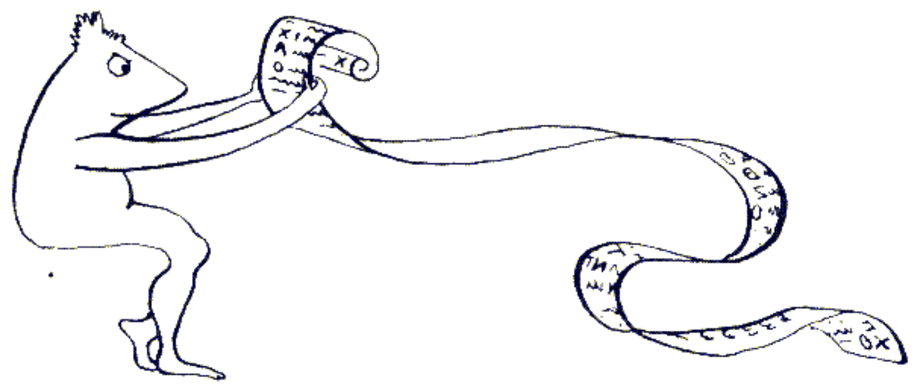
FORTH

LOGO

LE LANGAGE C



LES COUPS D'ŒIL DE LIST



Imprimerie SIMA, rue des Epinettes, ZI, de Torcy Sud, 77200 Torcy. Printed in France. Directeur de la publication : Jean-Luc Verhoye.
Dépôt légal décembre 1984. Diffusion NMPP. Numéro de commission paritaire : 63617