LTSI

LE JOURNAL DES AMATEURS DE PROGRAMMATION

nº 2

SEPTEMBRE 1984

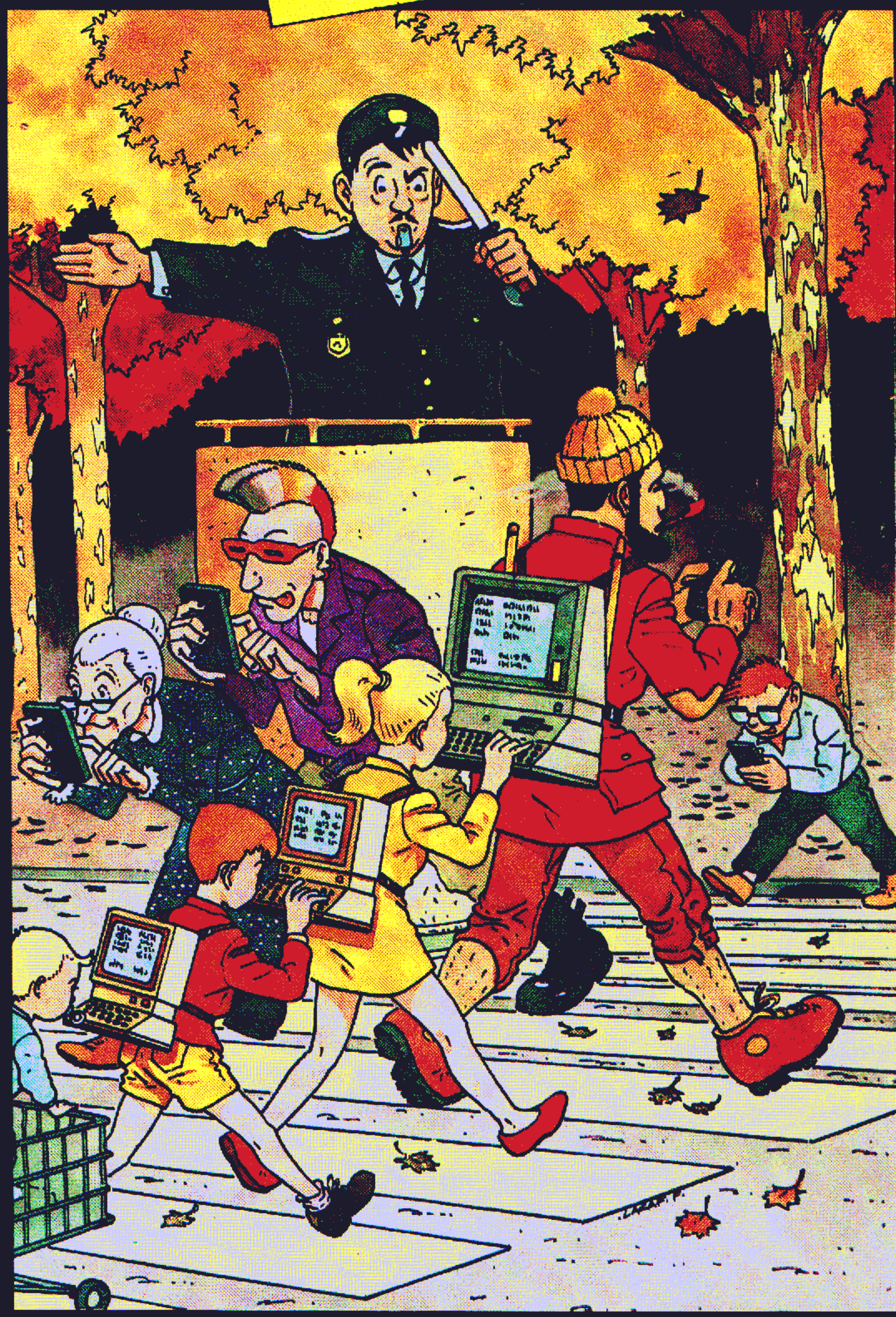
A l'essai : Le Basic du nouveau HP-71 B
Les matheux sont gâtés...

■ Tic-tac-toe,
un programme qui se perfectionne en jouant,
ou les balbutiements de l'intelligence artificielle

■ Histoire des langages de programmation :
FORTRAN, 30 ans déjà !

■ Trois logiciels à la loupe :
Basic graphique pour Oric
Editeur-assembleur pour C.64
De nouvelles fonctions pour PC-1500

■ Ordinateurs domestiques, ordinateurs de poche,
Assembleur, Basic, Forth, Logo, Pascal :
un trésor d'idées pour mieux programmer



M2712-2-20 F

l'Ordinateur de poche

Belgique : 166 FB - Canada : 2.95$C - Suisse : 6 FS

20 F

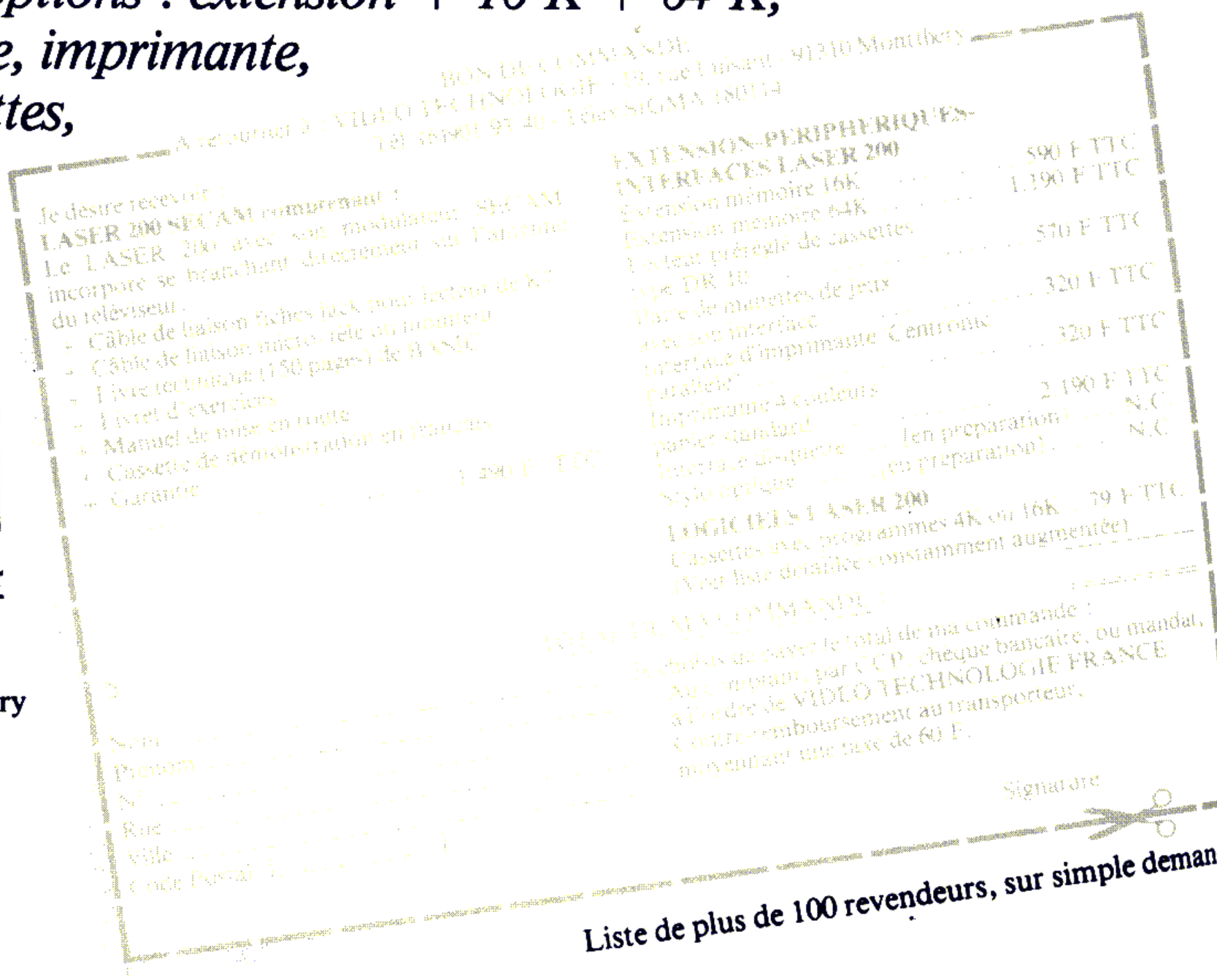
LE LASER 200

UN MICRO ORDINATEUR COULEUR SECAM

VRAIMENT TRÈS ÉTONNANT.

1490 F TTC

Microprocesseur Z 80 A • Langage Microsoft Basic • Affichage direct antenne télé SECAM • Clavier 45 touches pleine écriture, + clef d'entrée, + graphismes, + bip sonore anti-erreurs... • Texte + graphismes mixables 9 couleurs • Edition et correction plein écran • Son incorporé • Toutes options : extension + 16 K + 64 K, interface imprimante, imprimante, stylo optique, manettes, jeux, modem, disquettes...


Je désire recevoir :
LASER 200 SECAM comprenant :
Le LASER 200 avec son modulateur SECAM incorporé se branchant directement sur l'antenne du téléviseur.
- Câble de liaison fiches DIN pour lecteur de K7
- Câble de liaison prises télé du moniteur
- Livre technique (150 pages) de BASIC
- Lot d'exercices
- Manuel de mise en route
- Cassette de démonstration en français
- Garantie

1 490 F TTC

BON DE COMMANDE
A retourner à VIDEO TECHNOLOGIE FRANCE - 19, rue Luisant - 91310 Montlhéry
Tél. (6)901.93.40 - Télex SIGMA 180114

EXTENSION-PERIPHERIQUES-INTERFACES LASER 200

Extension mémoire 16K	590 F TTC
Extension mémoire 64K	1 190 F TTC
Lecteur prêteur de cassettes type DR 10	570 F TTC
Paire de manettes de jeux avec son interface	320 F TTC
Interface d'imprimante Centronics câblée	320 F TTC
Imprimante à couleurs parce standard	2 190 F TTC
Interface disquette	(en préparation) N.C
Stylo optique	(en préparation) N.C

LOGICIELS LASER 200
Cassettes avec programmes 4K ou 16K 59 F TTC
(Une base d'achat constamment augmentée)

MODE DE REGLEMENT :
Vous pouvez payer le total de ma commande :
Ci-joint par : C.C.P. chèque bancaire, ou mandat,
à l'ordre de VIDEO TECHNOLOGIE FRANCE
Contre-remboursement au transporteur,
moyennant une taxe de 60 F.

Nom
Prénom
N°
Rue
Ville
Code Postal

Signature

TECH

VIDEO TECHNOLOGIE FRANCE

19, rue Luisant - 91310 Montlhéry
Tél. (6)901.93.40
Télex SIGMA 180114

Liste de plus de 100 revendeurs, sur simple demande

eva w

la puissance

DRAGON 64 Prix : 3 600 F T.T.C. adaptateur PERITEL en option

Microprocesseur	6809 E
Mémoire	64 K RAM 16 K ROM (41 K avec 4 pages graphiques)
Ports	RS 232. 2 manettes, 1 cassette, 1 parallèle centronic
Sorties	Connecteur 40 lignes 6809 - PAL ou Peritel/UHF (son + vidéo) - 1 moniteur composite
Clavier	53 touches machine à écrire avec autorépétition
Affichage	Noir sur vert 16 x 32 - 24 x 51 sous OS 9 curseur bleu en mode 48 K
Graphique	16 x 32 9 couleurs - 32 x 64 9 couleurs 128 x 96 2 sets 2 couleurs - 128 x 96 2 sets 4 couleurs 256 x 192 2 sets 2 couleurs
Son	Par télé ou amplificateur
Basic	Microsoft® couleur étendu

LOGICIELS D 64 K de 750 à 1 250 F T.T.C.

OS 9	Système d'exploitation multitaches, multifonctions
Pascal	Langage P implemanté complet
C	Langage compilé linkable
Dynacale	Tableur professionnel
Stylograph	Traitement de textes + dictionnaire + liaison fichiers
RMS	Base de donnée
Basic 09	Basic structuré complet modulable
Flex	Système d'exploitation le plus répandu dans le monde

EXIGEZ LA CARTE DE GARANTIE DU REVENDEUR

BON DE COMMANDE ET DEMANDE DE DOCUMENTATION

Je désire recevoir : Une documentation ☐
Dragon 32 PERITEL ☐ UHF ☐
Dragon 64 PAL/Moniteur ☐ PERITEL (650 F) ☐
Lecteur de disquette ☐

ci-joint : Chèque bancaire ☐ Mandat ☐ Date :
Contre remboursement ☐ Signature
Frais à ma charge

NOM ..... Prénom .....

Adresse .....

Code postal ..... Ville .....

GOAL COMPUTER 15, rue de Saint-Quentin, 75010 PARIS

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

1 COUVERTURE
Comme on peut le voir, il n'y a pas d'âge pour s'intéresser à la programmation. Ce mois-ci, l'illustration de notre couverture est signée Fabien Lacaf.

21 A VOS CLAVIERS

22 POURQUOI, DIABLE, PROGRAMMEZ-VOUS ?
Nous avons posé la question à cinq personnes. Vous reconnaîtrez-vous dans les réponses que nous avons obtenues ?

23 LA GAZETTE DE LIST

33 TIC-TAC-TOE UN PROGRAMME MALIN
Ecrit en Basic standard, le programme joue contre vous ou contre lui-même. Et plus il dispute de parties, plus il devient difficile à battre...

36 L'HISTOIRE DES LANGAGES : LE FORTRAN
L'un des tout premiers langages évolués continue à être utilisé. Il a déjà trente ans, mais sa dernière version date de 1978.

39 ENCHAÎNONS, ENCHAÎNONS...
Les ordinateurs de la gamme Commodore ne connaissent pas la fonction CHAIN. Un moyen radical pour pallier cette absence.

41 A L'ESSAI : LE BASIC DU HP-71 B
Le dernier né des poquettes de Hewlett-Packard, malgré des dimensions très réduites, est doté d'un Basic étonnant. Il devrait faire le bonheur des amateurs de maths et de stats, s'ils peuvent se l'offrir...

44 A LA DÉCOUVERTE DES LOGARITHMES
Qu'est-ce qu'un log et comment le calcule-t-on ? Les quatre algorithmes présentés ici peuvent, bien entendu, être programmés. On les retrouve d'ailleurs parfois mis en œuvre dans le langage d'origine des ordinateurs.

47 POUR QU'UN PROGRAMME NE CHASSE PAS L'AUTRE
Basic Applesoft : un procédé pour faire cohabiter en mémoire vive un programme de menu et les autres programmes qu'il va chercher sur la disquette.

49 SI VOTRE PROCESSEUR EST UN Z80
Le plus grand nombre premier connu est long de 39 751 chiffres. Pour en retrouver la valeur exacte, quatre heures suffisent. La routine en langage-machine occupe 98 petits octets et permet par ailleurs d'explorer les nombres de Mersenne.

52 TROIS LOGICIELS A LA LOUPE

ORIC, BASIC ETENDU
Une cassette qui, pour un prix modeste, dope l'Oric-1 et l'Atmos. Les améliorations concernent principalement les graphismes.

ARROW 64 POUR COMMODORE 64
La cartouche contient un éditeur-assembleur assez performant et simple d'emploi. Les débutants regretteront tout de même la minceur de la notice d'emploi.

PC-UTIL 2 POUR PC-1500
Selon les modèles du poquette, 16 ou 19 fonctions supplémentaires dont une bonne partie intéresse l'aide à la programmation.

58 PASCAL, FILTREZ LES ENTRÉES...
Une erreur dans la saisie d'un nombre, et le programme déraille. Pour éviter cela, suivons la bonne procédure.

61 QUE LE GRAND CRIC ME CROQUE !
HP-41C : suite de l'initiation à la programmation synthétique.

N° 2 - SEPTEMBRE 1984

SOMMAIRE

63 *PASSIONNÉ PAR FORTH*
Progressons dans la découverte de ce langage original. Ce mois-ci, nous nous intéressons plus spécialement à la boucle DO...LOOP.

67 *THOMSON MO5 AU BOUT DU CRAYON*
Créez, en un clin d'œil, vos propres caractères graphiques.

68 *UN CATALOGUE AMÉLIORÉ POUR LE SPECTRUM*
Chaque cartouche des « microdrives » contient davantage d'informations qu'elle n'en délivre normalement. Comment aller les repêcher ?

73 *PC-1500 : FAIRE FACE AUX ERREURS*
Dans certains cas, on peut prévoir qu'un programme conduira normalement à des messages d'erreur. On doit alors faire en sorte que l'exécution n'en soit pas interrompue.

75 *LOGO N'EST PAS RÉSERVÉ AUX ENFANTS*
Petits cours et travaux dirigés sur ce langage dont les possibilités sont trop souvent méconnues.

79 *SI VOTRE PROCESSEUR EST UN 6502*
Oric, Atmos, Apple... Faites au moins une fois l'expérience du langage-machine. Qui sait si, à votre tour, vous ne deviendrez pas un mordu de cette forme de programmation ?

82 *PB-700 : LES ÉCRITS RESTENT*
Quand l'affichage fait plaisir à voir, on aimerait souvent en garder une trace. Deux utilitaires de recopie d'écran.

91 *LA BOÎTE A MALICES*
Prenez un programme et retirez-en les astuces, toutes les astuces, des plus grossières aux plus subtiles. Que reste-t-il ? Rien. Dans ce numéro, des ficelles pour TO 7, TI-66, TI-99/4A, PB-100, CBM 4000 et 8000, VIC, C.64, HP-41C...

98 *MISEZ P'TIT, OPTIMISEZ*
Grignotons les octets et les fractions de seconde (HP-41C). Un nouveau défi et les résultats du précédent : onze vainqueurs ex aequo !

100 *JEUX ET CASSE-TÊTE INFORMATIQUES*
Exercez votre logique et votre ingéniosité pour résoudre quelques petits problèmes simples en apparence.

Ce numéro contient en encart des bulletins d'abonnement paginés 19, 20, 85 et 86.

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

Ont collaboré à ce numéro : Olivier Arbey, Michel Arditti, François J. Bayard, Catherine Bellamy, Eric Bertrem, Paulette Besnard, Robin Bois, Christian Boyer, Gilles Bransbourg, Gérald Béjot, Jean-Paul Carré, Thierry Chamoret, Robert Daguesse, Jacques Deconchat, Jérôme Gaudin, Florence Gautier-Louette, Béatrice Ginoux Defermon, Max Hagenburger, Renée Koch, Jean-Christophe Krust, Jacques Labidurie, Marcel Laglasse, Jean-Pierre Lalevée, Bernard Lambey, Thierry Lévy-Abégnoli, Alain Mariatte, Christian Mildner, Pierrick Moigneau, Arnaud Peruta, Yvon Pérès, Denis Sebbag, Michel Susini, Benoît Thonnart, Eric Tière, André Warusfel, Marc Wisniewsky, Boris Yarénitch.

Illustrations : Philippe Burel, Philippe Delacroix, Frapar, Manuel Gracia, Bernard Helme, Fabien Lacaf, Alain Mangin, Alain Mirial, Alain Prigent, Jean-Marc Rubio, Nicolas Spinga.

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

VENTES
Diffusion NMPP : Sophie Marnez
Abonnements : Muriel Watremez
assistée de Sylvie Trumel, Cécilia Mollicone et Dominique Loridan

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

SINCLAIR s'impose par la passion des Sinclairistes. Ils sont 2 millions dans le monde à avoir découvert Sinclair. Les revues et les nombreux clubs en sont l'écho.

Fiche technique du ZX SPECTRUM

Unité centrale
Microprocesseur Z 80 A, 3,25 MHz.
RAM 16 K ou 48 K.
ROM 16 K.

Clavier
40 touches avec répétition automatique et témoin sonore. Système d'entrée de toutes les fonctions par mots-clefs.

Affichage
31 x 24 caractères, majuscules ou minuscules. Haute définition graphique 256 x 192 (49 152 points adressables individuellement).

Générateur de caractères
ASCII étendu (matrice 8 x 8). 21 caractères programmables. Possibilité de redéfinition de l'ensemble des caractères.

Couleurs et sons
8 couleurs. Haut-parleur intégré 130 demi-tons (10 octaves). Amplification par prise micro.

Langages
Basic intégré, Pascal, Assembleur et Forth en option.

Interface magnétophone
Vitesse de transmission : 1500 bauds. Sauvegarde de pages mémoire et tableaux séparés. Fonctions VERIFY et MERGE.

Ecran
Raccordement sur prise antenne pour récepteur PAL ou prise PERITEL pour récepteur SECAM.

Nous sommes à votre disposition pour toute information au 359.72.50.
Magasins d'exposition-vente :
Paris - 11 rue Lincoln 75008 (M° George V)
Lyon - 10 quai Tilsitt 69002 (M° Bellecour)
Marseille - 5 rue St-Saëns 13001 (M° Vieux-Port).

EXCEPTIONNEL
Ensemble Spectrum (48 K) + coffret de 8 logiciels vedettes au prix « Sinclair » de 1965 F.

Sinclair s'impose.

Sinclair s'impose par la richesse unique de sa gamme de logiciels et de par sa bibliographie incomparable.

Sinclair s'impose par sa capacité d'innovation et son souci de la perfection, à des prix abordables par tous.

Les 3 nouveaux périphériques du ZX SPECTRUM en sont la preuve. Découvrez-les d'urgence.

Le Microdrive ZX

Une prouesse technologique dans le domaine de la mémoire. Chaque microdrive utilise des bandes sans fin interchangeables, d'une capacité de 85 K octets. L'accès à la mémoire s'effectue en un temps record. Ainsi, un programme de 48 K octets se charge en 9 secondes. 8 microdrives peuvent être connectés au SPECTRUM, qui dispose alors d'une capacité de 680 K octets en ligne. C'est incomparable.

L'Interface ZX 1

Une extension qui transforme votre micro en géant. Elle permet, outre le raccordement des microdrives, de gérer des fichiers et de brancher des imprimantes de format courant. De plus, elle autorise l'établissement d'un réseau de communication à vitesse élevée, pouvant regrouper 64 SPECTRUM. Et toujours à un prix Sinclair.

L'Interface ZX 2

Avec elle, le plaisir est total. Elle lit instantanément les nouvelles cartouches ROM de jeu et permet le branchement simultané de 2 manettes de jeu.

Ce nouveau périphérique peut se brancher directement sur le microordinateur ou sur l'interface ZX 1.

Le ZX SPECTRUM constitue alors un incomparable système informatique. Sinclair s'impose.

Présent au Sicob Stand N° 146

sinclair
la micro-ordination

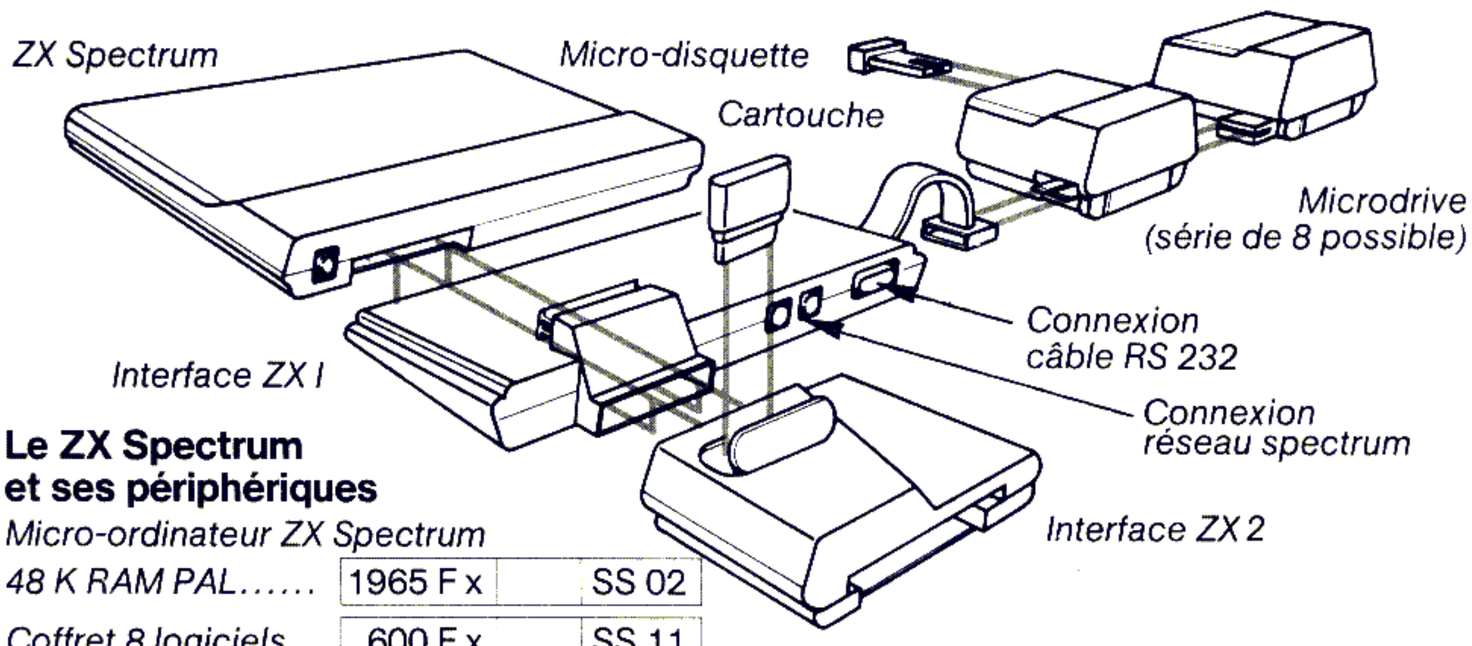
Bon de commande au verso.

ZX Spectrum. Un incomparable système informatique.

CRÉATEURS CONSEILS

Bon de commande

A retourner à Direco International - 30, avenue de Messine - 75008 Paris.



Le ZX Spectrum et ses périphériques

Micro-ordinateur ZX Spectrum

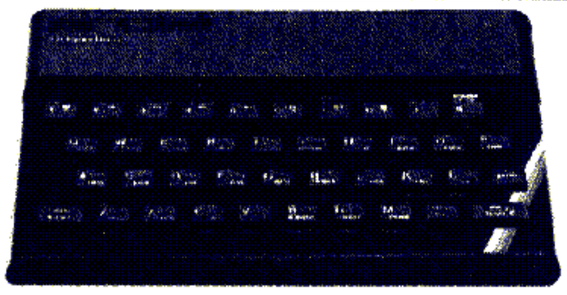
48 K RAM PAL.....	1965 F x		SS 02
Coffret 8 logiciels ..	600 F x		SS 11

Promotion Automne 84

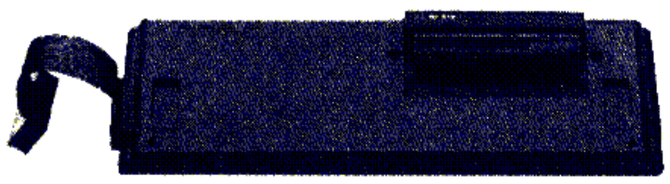
Spectrum PAL + coffret	1965 F x		SS 10

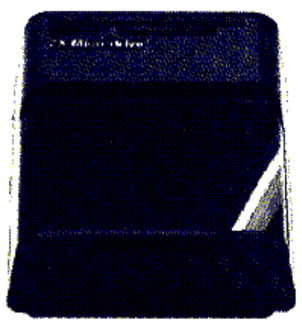
(offre valable jusqu'au 30.09.84)

Adaptateur Péritel..	360 F x		CS 05



Interface ZX 1	895 F x		SS 05
Câble RS 232	235 F x		SS 06





Microdrive ZX

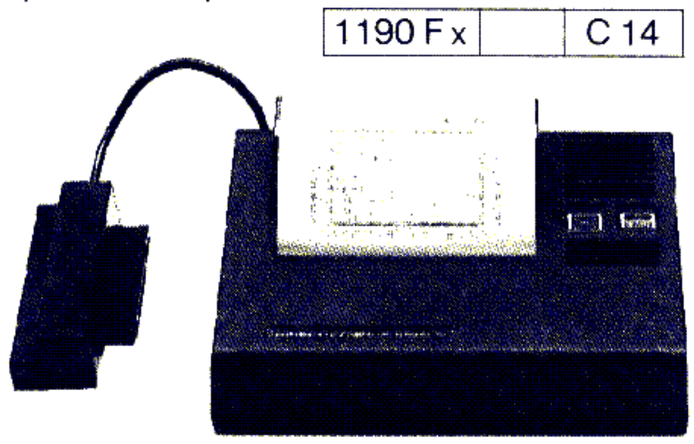
940 F x		SS 07	

Boîte de 4 microdisquettes vierges

316 F x		SS 09	

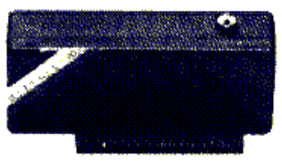
Imprimante Alphacom 32

1190 F x		C 14	



Boîte de 5 rouleaux de papier

150 F x		P 02	



Modulateur noir et blanc

190 F x		CS 04	

Manette de jeux Quickshot

140 F x		C 15	

Interface ZX 2

351 F x		SS 10	

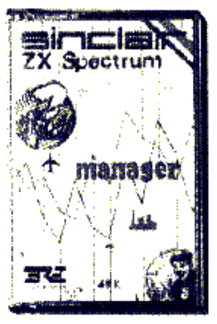
Les logiciels-cartouches

Pssst !	185 F x		RS 01
Jet Pac	185 F x		RS 02
Cookie	185 F x		RS 03
Trans Am	185 F x		RS 04
Space Raiders	185 F x		RS 05
Planetoids.....	185 F x		RS 06
Hungry Horace.....	185 F x		RS 07
Echecs.....	185 F x		RS 09
Backgammon	185 F x		RS 10

Les logiciels-cassettes

JEUX DE RÉFLEXION

Cobalt (simul. de vol)	95 F x		JS 01
Echecs.....	115 F x		JS 15
Othello.....	75 F x		JS 02
Manager.....	140 F x		JS 16



UTILITAIRES

Pascal.....	260 F x		US 01
ZX Trans.....	95 F x		US 03
Devpac (Ass/Desass)	160 F x		US 02

JEUX D'ARCADES

Jumping Jack	95 F x		JS 17
Zoom.....	95 F x		JS 18
Alchemist	95 F x		JS 23
Mined-Out.....	86 F x		JS 05
Androïdes.....	75 F x		JS 07

GESTION

Direction financière.	120 F x		GS 01
Gestion de fichier ...	115 F x		GS 02

TOTAL : ..... *F*

Indiquez dans chaque case la quantité commandée. Effectuez le calcul du total et inscrivez le résultat dans la case TOTAL.

Votre commande vous sera adressée sous 3 semaines.

Je paie par : chèque bancaire ..... ☐

CCP..... ☐

établi à l'ordre de Direco International, joint au présent bon de commande.
(aucun chèque n'est encaissé avant l'expédition du matériel).

contre-remboursement* ☐

** Contre-remboursement taxe PTT (14,20 F) pour toute commande de moins de 2000 F. Au-delà, barème Sernam.*

Nom _____

Prénom _____

Adresse _____

_____

Code postal ⌊_⌊_⌊_⌊_⌊_⌋ Tél. : _____

Signature (pour les moins de 18 ans, signature de l'un des parents):

LLS A

Au cas où je ne serais pas entièrement satisfait, je suis libre de vous retourner le matériel dans les 15 jours. Vous me rembourserez alors entièrement.

sinclair

la micro-ordination

Pour votre ordinateur, Boîtes à outils MégaO Poche.

Mine de rien, une mine d'idées.

La collection MégaO Poche met à votre disposition des petits programmes tout prêts qui vous permettront de résoudre les nombreux problèmes de la vie quotidienne, de simplifier votre comptabilité, votre tableau de bord, la tenue de vos fichiers etc...

Actuellement 14 titres :
Boîte à outils pour PC 1500
Boîte à outils pour FX 702 P
Boîte à outils pour Sinclair
Boîte à outils pour PC 1251
Boîte à outils pour PB 100 et FX 802 P,
TRS-80 PC-4
Boîte à outils pour le navigateur de plaisance
Boîte à outils pour ORIC - Tome 1
Boîte à outils pour ORIC - Tome 2
Boîte à outils pour Commodore 64
Boîte à outils pour TI-99/4A
Micro gestionnaire pour Sinclair
Micro compta pour Sinclair
Boîte à outils Micro-Stats NOUVEAU
Boîte à outils pour Micro-Manager NOUVEAU

MegaO POCHE

La collection MégaO Poche est diffusée par : P.S.I. DIFFUSION
B.P. 86 77402 Lagny-sur-Marne Cédex Tél. : (6) 006.44.35 Télex : PSIDIF 600 978 F

P.S.I.

Table de conversions en Francs belges et Francs suisses
35 FF = 250 FB - 12,20 FS

Envoyez ce bon accompagné de votre règlement à :
FRANCE : P.S.I. diffusion - BP 86
77402 Lagny-s/Marne Cédex - Tél. : (6) 006.44.35
BELGIQUE, LUXEMBOURG : P.S.I. Benelux
5, avenue de la Ferme Rose 1180 Bruxelles
Tél. : (2) 345.08.50
SUISSE : P.S.I. Suisse - Case postale - Route neuve 1
1701 Fribourg (Suisse) - Tél. : (037) 23.18.28 - CCP 17 56 84
CANADA : S.C.E. Inc - 65, avenue Hillside Montréal
(Westmount) - Québec H3Z1W1 - Tél. (514) 935.13.14
MAROC : SMER Diffusion - 3, rue Ghazza - Rabat - Tél. (7) 237.25

LG 9

DÉSIGNATION	NOMBRE	PRIX
	TOTAL	

par avion ajouter 8 FF (75 FB) par livre
pour la Suisse, ajouter 1,50 FS pour tous les ouvrages

Signature obligatoire pour paiement par carte de crédit ▶

☐ Paiement par chèque joint ☐ Paiement en FF par carte bleue VISA (à P.S.I. DIFFUSION uniquement paiement supérieur à 50 FF)

N° ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Date d'expiration ☐☐☐☐

NOM ..... PRÉNOM .....

RUE ..... N° .....

CODE POSTAL ..... VILLE .....

PSI : PROGRAMMEZ DE VOS PROPRES AILES

Vous avez du mal à résoudre certains problèmes pratiques de la programmation car vous manquez de méthode ?
Les ouvrages de la collection "techniques et méthodes" vous fourniront les moyens d'approcher un problème particulier sous un angle très général grâce à des techniques spécifiques.

TECHNIQUES DE PROGRAMMATION DES JEUX

par David Levy
256 pages - 110,00 FF

Tout le monde aime jouer, même si en fonction des goûts de chacun les préférences diffèrent grandement. D'un individu à l'autre on constate que la structure mentale qui permet de briller aux échecs, au bridge, au backgammon ou au poker est aussi particulièrement adaptée à la résolution de mots croisés ou à la programmation des ordinateurs. Dans le même ordre d'idée, la programmation sur ordinateur individuel des jeux intellectuels réserve de grandes joies à cette catégorie de joueurs. C'est à eux que s'adresse cet ouvrage qui aborde la transformation des stratégies et des tactiques des principaux jeux intellectuels en algorithmes de programmation.

LA PROGRAMMATION INVENTIVE

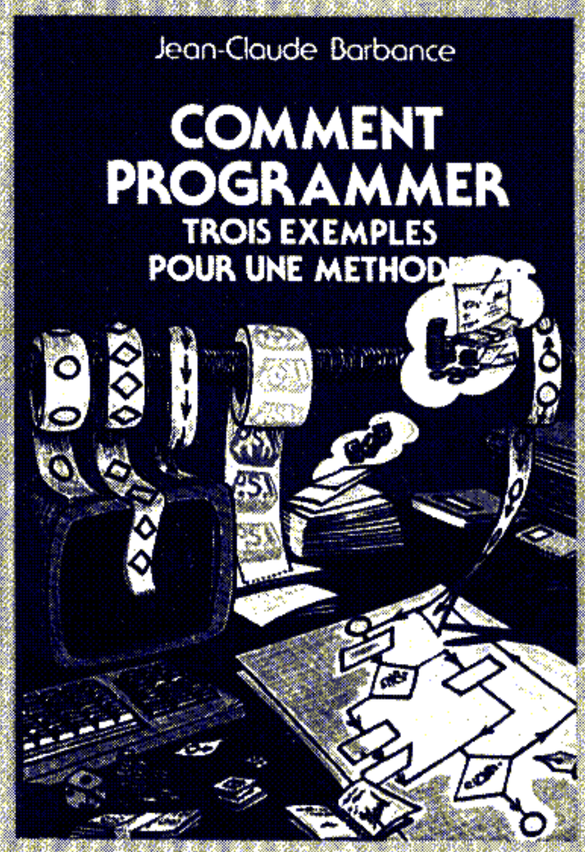
par Xavier de la Tullaye
160 pages - 95,00 FF

Voici une analyse du processus de raisonnement logique qui conduit à la programmation : en procédant par analogies et en s'appuyant sur des exemples simples (programme d'une journée, construction d'une étagère), l'auteur initie au maniement des outils de la programmation : l'ordinateur lui-même, l'algorithme, l'organigramme, l'écriture du programme. A la fin du livre il ne reste plus au lecteur qu'à faire preuve d'imagination pour réaliser des programmes.

MÉTHODES DE RÉALISATION DES PROGRAMMES

par Michel Benelfoul
104 pages - 80,00 FF

Cet ouvrage s'adresse aux utilisateurs d'ordinateurs individuels qui, après avoir appris le Basic, sentent le besoin d'une approche méthodique de la réalisation des programmes : définition du problème, étude de la solution, programmation, mise au point, maintenance. Une définition rigoureuse du vocabulaire informatique et de nouveaux concepts d'analyse, tirés des mathématiques modernes y sont exposés.



COMMENT PROGRAMMER

(Nouvelle Edition)
Trois exemples pour une méthode
par Jean-Claude Barbance
216 pages - 100,00 FF

Jean-Claude Barbance s'adresse ici à ceux qui connaissent déjà le Basic et désirent aborder les principes fondamentaux de la réalisation des programmes : définition d'objectifs, analyse, confection des organigrammes et mise au point. Les trois exemples traités dans ce livre sont écrits en Basic standard et le lecteur pourra aisément adapter les programmes à son ordinateur. Cet ouvrage a fait l'objet d'une mise à jour complète.

P.S.I.

COLLECTION "TECHNIQUES ET MÉTHODES"

P.S.I.

Table de conversions en France belges et France suisses
35 FF = 250 FB - 12,20 FS
80 FF = 465 FB - 19,10 FS
70 FF = 540 FB - 22,20 FS
80 FF = 620 FB - 25,30 FS
90 FF = 695 FB - 28,40 FS
100 FF = 770 FB - 31,50 FS
110 FF = 850 FB - 34,60 FS
120 FF = 925 FB - 37,60 FS
130 FF = 1000 FB - 40,60 FS

Envoyez ce bon accompagné de votre règlement à :

FRANCE : P.S.I. diffusion - BP 86
77402 Lagny-s/Marne Cédex - Tél. : (6) 006.44.35

BELGIQUE, LUXEMBOURG : P.S.I. Benelux
5, avenue de la Ferme Rose 1180 Bruxelles
Tél. : (2) 345.08.50

SUISSE : P.S.I. Suisse - Case postale - Route neuve 1
1701 Fribourg (Suisse) - Tél. : (037) 23.18.28 - CCP 17 56 84

CANADA : S.C.E. Inc - 65, avenue Hillside Montréal
(Westmount) - Québec H3Z1W1 - Tél. (514) 935.13.14

DÉSIGNATION	NOMBRE	PRIX
	TOTAL	

par avion ajouter 8 FF (75 FB) par livre
pour la Suisse, ajouter 1,50 FS pour tous les ouvrages

Signature obligatoire pour paiement par carte de crédit ►

☐ Paiement par chèque joint ☐ Paiement en FF par carte bleue VISA (à P.S.I. DIFFUSION uniquement paiement supérieur à 50 FF)

N° |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| Date d'expiration |_|_|_|_|

NOM _____ PRÉNOM _____

RUE _____ N° _____

CODE POSTAL _____ VILLE _____

LTM : 27/14.25

ORDI-5

le magazine de votre

SINCLAIR

SPECTRUM et ZX-81

Si vous utilisez un ordinateur SINCLAIR (ZX 81*, ZX 80* ou Spectrum*) ou si vous comptez en acheter un, sachez que la revue **ORDI-5** a été créée pour vous. Indépendante de tout constructeur ou importateur, **ORDI-5** vous fournit quatre fois par an des programmes, des conseils, des astuces, de nouvelles idées d'utilisation. **ORDI-5** teste pour vous en toute objectivité et indépendance les produits matériels et logiciels adaptables sur votre SINCLAIR. **ORDI-5** vous tient au courant de toutes les nouveautés susceptibles de vous intéresser. Commandez un numéro ou... **abonnez-vous**, vous économiserez 20 %.

*marques déposées

ORDI-5, pour tirer bien plus de votre SINCLAIR

BON DE COMMANDE

à retourner à ORDI-5, 8 rue Saint-Marc 75002 PARIS

Nom _____

Adresse _____

Pays _____ Code postal _____ Ville _____

☐ Je désire recevoir les 4 derniers numéros parus et m'abonner pour recevoir les 4 prochains numéros.
(France 160 FF ; Etranger** 180 FF ; par avion 320 FF).

☐ Je désire recevoir les numéros antérieurs suivants : _____
(prix d'un n° 25 FF ; Etranger** 30 FF ; par avion 40 FF).

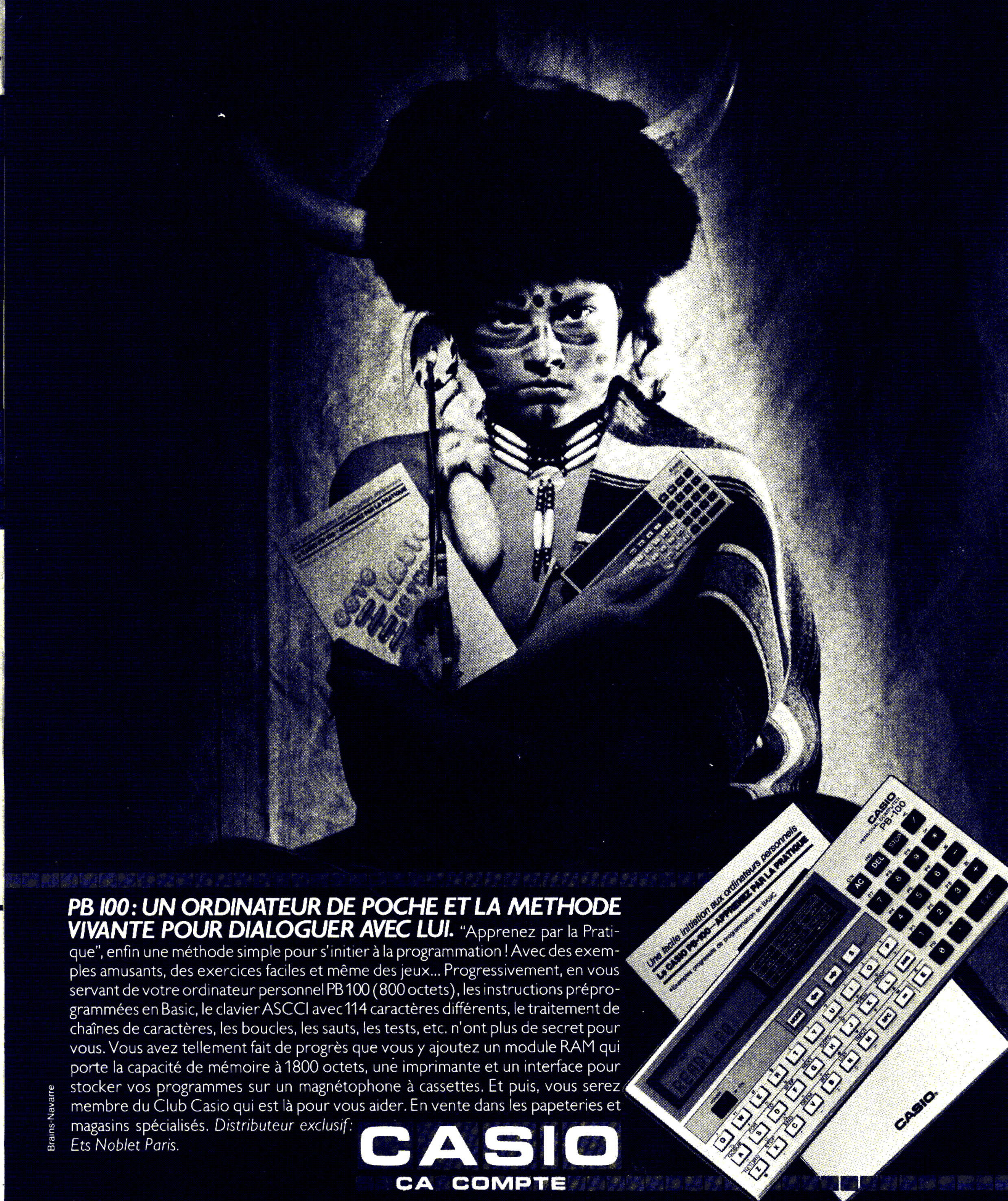
☐ Je désire m'abonner à ORDI-5 pour 1 an, 4 n^{os} à partir du n° _____
(tarif France 80 FF ; Etranger** 90 FF ; par avion 160 FF). (Actuellement ORDI-5 est trimestriel).

Ci-joint mon règlement indispensable par chèque bancaire ☐ chèque postal ☐ virement ☐

**Pour les pays autres que la France, utiliser un virement en FF compte Crédit Lyonnais Paris n° 30002 00402 8455 J. Les frais de virement sont à la charge de l'acheteur.

CASIO PB 100

LE BASIC PAS SORCIER



PB 100 : UN ORDINATEUR DE POCHE ET LA METHODE VIVANTE POUR DIALOGUER AVEC LUI. "Apprenez par la Pratique", enfin une méthode simple pour s'initier à la programmation ! Avec des exemples amusants, des exercices faciles et même des jeux... Progressivement, en vous servant de votre ordinateur personnel PB 100 (800 octets), les instructions préprogrammées en Basic, le clavier ASCCI avec 114 caractères différents, le traitement de chaînes de caractères, les boucles, les sauts, les tests, etc. n'ont plus de secret pour vous. Vous avez tellement fait de progrès que vous y ajoutez un module RAM qui porte la capacité de mémoire à 1800 octets, une imprimante et un interface pour stocker vos programmes sur un magnétophone à cassettes. Et puis, vous serez membre du Club Casio qui est là pour vous aider. En vente dans les papeteries et magasins spécialisés. *Distributeur exclusif: Ets Noblet Paris.*

Brains-Navarre

CASIO

CA COMPTE

édi tests : les dernières parutions...

Demandez le catalogue gratuit EDITESTS

(P.S.I. au SICOB 84 : C.N.I.T. - Sicob Boutique - stand n° 30)

Série « Système d'Exploitation »

Le système MemDos

par Pierre Clerc - 104 pages - 90,00 FF

Quel que soit le langage de programmation habituel de l'utilisateur, la connaissance de MemDos lui permettra une réflexion féconde sur les caractéristiques optimales d'un système d'exploitation. MémDos se superpose au Basic habituel des Apple II et III, CBM, ITT 2020, ... et bientôt IBM-PC. Utilisé pour le développement d'application de gestion, il permet d'adopter une structure rigoureuse tout en bénéficiant des commodités d'un Basic interprété.

Série « Productique »

Systèmes industriels d'intelligence artificielle

par Lucas Pun - 304 pages - 130,00 FF

Les ingénieurs, industriels, automaticiens et informaticiens qui ont à concevoir et à réaliser de plus en plus de « Systèmes Industriels d'Intelligence Artificielle » (SIIA) dans le cadre de la productique, trouveront dans cet ouvrage à vocation pédagogique les réponses à des questions telles que : Que sont les SIIA ? Comment en faire l'analyse et la synthèse ? Où et quand les appliquer ? Etc.

Introduction à la robotique

Enseignement, recherche, développement

par Pierre Lopez et Jean-Numa Foulc

310 pages - 170,00 FF - *Vient de paraître*

Elève de terminale ou élève ingénieur, chercheur, technicien ou ingénieur, le lecteur, motivé par la robotique et désireux de contribuer à la diffusion des robots dans les divers secteurs de la productique, trouvera dans le premier tome de « l'introduction à la robotique » la formation de base indispensable à la fabrication et à la maintenance de ces Systèmes Artificiels ayant une action intelligence sur l'environnement et appelés plus communément « robots ».

Série « Informatique »

Opérations arithmétiques dans les ordinateurs

par Ioan Dancéa

176 pages - 100,00 FF - *Vient de paraître*

Les concepteurs d'ordinateurs ont besoin de matériels de plus en plus rapides pour effectuer les quatre opérations arithmétiques (addition, soustraction, multiplication et division) et augmenter les performances de leurs systèmes. Cet ouvrage leur donne les différentes méthodes possibles, accompagnées de réalisations matérielles et illustrées de nombreux exemples. Sont par exemple abordés les différents schémas d'additionneurs et de soustracteurs, l'addition et la soustraction des nombres binaires signés. Ils permettront au lecteur d'approfondir ses connaissances des mécanismes de fonctionnement d'un ordinateur.

Série « Communication »

Les techniques de la télématique

par Jérôme Toussaint et Philippe Masson

120 pages - 90,00 FF

Le but de cet ouvrage est de présenter l'ensemble des techniques regroupées sous le terme général de télématique. Il ne s'adresse pas aux seuls spécialistes des télécommunications ou de l'informatique, mais à tous ceux qui veulent s'informer des évolutions à venir. Parmi les sujets traités, on trouvera entre autres le terminal Minitel et ses applications, le traitement de textes et la bureautique, la télécopie, la numérisation du téléphone et des réseaux publics de télécommunications.

et toujours...

- **Série « Système d'exploitation »**
  - . Le système Unix - 100,00 FF
  - . Le système CP/M pour Z-80 - 100,00 FF
  - . Le système CP/M pour 8080 - 110,00 FF
  - . Le système Pascal UCSD tome 1 - 90,00 FF
  - . Le système Pascal UCSD tome 2 - 100,00 FF
  - . Le concept Forth - 110,00 FF
- **Série « Langage de communication »**
  - . Le langage C - 90,00 FF
  - . Le langage APL - 90,00 FF
  - . Le langage ADA - 90,00 FF
  - . Le langage Fortran - 80,00 FF
- **Série « Communication »**
  - . Les réseaux locaux d'entreprise - 110,00 FF
- **Série « Dialogue Homme-machine »**
  - . Synthèse, reconnaissance de la parole - 130,00 FF
- **Série « Electronique numérique »**
  - . Choisir un système de développement pour microprocesseurs - 80,00 FF
  - . Les systèmes à microprocesseurs - 90,00 FF
  - . Microprocesseurs et circuits associés - 100,00 FF
  - . Les circuits programmables - 110,00 FF
  - . Mise en œuvre du BUS IEEE 488 - 90,00 FF

Envoyer ce bon accompagné de votre règlement à **P.S.I. DIFFUSION** ou pour la Belgique et le Luxembourg à **P.S.I. BENELUX** et pour la Suisse à **P.S.I. SUISSE** (voir table de conversions ci-contre).

P.S.I. DIFFUSION - BP 86
77402 Lagny-s/Marne Cedex - FRANCE - Tél. : (6) 006.44.35

P.S.I. BENELUX - 5, avenue de la Ferme-Rose
1180 Bruxelles - BELGIQUE - Tél. : (2) 345.08.50

P.S.I. SUISSE - Case Postale - Route Neuve
1701 Fribourg - SUISSE
Tél. : (037) 23.18.28 - C.C.P. 17-5684

au Canada : SCE Inc.
65, avenue Hillside Montréal
(Westmount) Québec H 32 1 W1
Tél. : (514) 935.13.14

au Maroc : SMER DIFFUSION
3, rue Ghazza - Rabat
Tél. : (7) 237.25

Table de conversions en Francs belges et Francs suisses
80 FF = 620 FB - 25,30 FS
90 FF = 695 FB - 28,40 FS
100 FF = 770 FB - 31,50 FS
110 FF = 850 FB - 34,60 FS
120 FF = 925 FB - 37,60 FS
130 FF = 1 000 FB - 40,60 FS
170 FF = 1 310 FB - 52,60 FS

LE 9

DÉSIGNATION	NOMBRE	PRIX
	TOTAL	

par avion ajouter 8 FF (75 FB) par livre
Pour la Suisse, frais de port pour tous les livres : 1,50 FS

Nom _____ Prénom _____

Adresse _____ N° _____

Code postal _____ Ville _____

☐ ***Je désire recevoir le catalogue EDITESTS gratuit***

☐ *Paiement par chèque joint* ☐ *Paiement en FF par carte bleue VISA (à P.S.I. DIFFUSION uniquement)*

N° _____ *Date d'expiration* _____

Signature (obligatoire pour paiement par carte de crédit)

A VOS CLAVIERS

DEUX MOIS, C'EST LONG...

FÉLICITATIONS pour votre nouveau journal, mais sur la couverture, j'ai lu « juillet-août ». *LIST* serait-il donc bimestriel ? Et si oui, envisagez-vous de le rendre mensuel ?

Eric HANUISE
Soignies Belgique

■ LIST *ne paraîtra pas tous les deux mois, mais à raison de* **dix** *numéros par an, comme c'était le cas de* l'Op. *Deux numéros doubles donc, juillet-août et janvier-février, pour nous permettre de souffler et de prendre un peu de recul.*

J'INVESTIGUE

BONJOUR à l'équipe de la rédaction,

Puisque l'Ordinateur de poche s'est déguisé en *LIST* (on a du mal à le reconnaître), une petite comparaison s'impose entre le premier *LIST* et le dernier *Op*.

Format de la revue : identique à deux ou trois millimètres près, je crois. Epaisseur : 70 pages pour *l'Op* (60 seulement pour le n° 22), et 100 pages pour *LIST*. On remarque par ailleurs une augmentation de la place consacrée à la publicité.

L'information générale occupe elle aussi une place plus importante (ça, c'est bon pour la culture informatique des lecteurs). Enfin, la revue consacre presque deux fois plus de place aux ordinateurs de table qu'aux ordinateurs de poche (Help !) malgré la bénéfique augmentation du nombre de pages.

Un lecteur investigateur (pour la rime)

P.S. Pourrait-on avoir des essais de logiciels pour ordinateurs de poche (ce sont les essais de logiciels d'aide à la programmation pour ordinateurs de table qui ont fait germer l'idée) ? Un gros merci d'avance.

■ *Oui, c'est juste,* LIST *n'est pas la réplique exacte de* l'Op. *Mais avec cette nouvelle formule, il nous paraît, quant à nous, que les utilisateurs de matériels de poche ne devraient pas perdre au change concernant tout ce qui touche à la programmation.*

A propos des essais de logiciels pour OP, nous comptons en publier. Vous trouverez d'ailleurs dans ce numéro, page 56, notre coup d'œil sur PC-UTIL 2 pour Sharp PC-1500.

FAITES CIRCULER...

SALUT les LISTeurs,

LE groupe TESTS a encore frappé. Après *l'Ordinateur Individuel, l'Op,* voici *LIST*. Dans le kiosque, ce titre m'a fait sursauter. Ce nom a évoqué en moi des dizaines de pages de programmes. Je suis un tapeur fou ! J'adore taper sur mon TRS 80.

Lorsque j'ai ouvert votre magazine, j'ai eu plusieurs bonnes surprises. La première c'est qu'il n'y a pas trop de pub. Un peu ça va, mais trop... Deuxième bonne surprise, il y avait des programmes pour mon TRS. Bien que Tandy ne fabrique plus le modèle 1, il est sûr que celui-ci a encore de très beaux jours devant lui.

La troisième bonne surprise est que vous parlez de l'assembleur. C'est un langage très complet qui permet un dialogue direct avec son ordinateur favori. Pour le TRS, j'ai quelques idées en tête, et si les lecteurs de *LIST* peuvent écrire des articles, ou quelque chose qui y ressemble, j'aimerais bien savoir s'il est possible de les faire sortir chez vous.

Fabrice Glibert
Paris 19e

DANS notre club, nous aimerions savoir si vous envisagez de plublier des articles concernant le DAI, notre ordinateur de prédilection ? En attendant votre réponse, je vous adresse mes sincères salutations.

Daniel MOULES
63 Saint-Germain Lembon

■ ***Bien entendu, le DAI n'est pas exclu : tout ce qui touche à la programmation intéresse notre journal. Et nous répétons ce que nous écrivions dans notre premier numéro : chez nos lecteurs, combien de trouvailles dorment dans des tiroirs ? Faites circuler, faites circuler vos idées.***

Langage-machine, assembleurs, Pascal, Logo, Forth, Basic, notation polonaise inverse ou algébrique, etc., trouvailles d'intérêt général ou destinées à un matériel particulier, algorithmes originaux, n'hésitez pas à nous écrire : critiques, suggestions, propositions d'articles...

JAI reçu *LIST* avant-hier et j'avoue être agréablement surpris, sauf peut-être pour la présentation un peu triste. J'ai tout d'abord été heureux de consater que les ordinateurs de poche avaient gardé une place de choix. Le nombre de sujets traités a considérablement augmenté pourtant !

J'ai particulièrement apprécié la variété des langages abordés (Forth m'a spécialement intéressé). Bien que programmant depuis deux ans, j'ai beaucoup appris dans le premier numéro de *LIST*.

Un petit renseignement maintenant, s'il vous plaît : quelle est l'adresse de DDI qui assure la diffusion des cassettes Logi'stick ?

Bruno DESAUNAY
14 Condé sur Noireau

■ ***Voici l'adresse que vous recherchez :***
DDI
Centre d'Affaires Paris-Nord
« Le Bonaparte »
93153 Le Blanc Mesnil

INDEX DES ANNONCEURS

BVP ..... **p. 14**
Casio ..... **p. 17**
Cassettes Le Témoignage ..... **p. 26**
Duriez ..... **p. 102**
Ecole Universelle ..... **p. 25**
Editrace ..... **p. 16**
ETMS ..... **p. 29**
Guide de l'Ordinateur Individuel ..... **p. 103**
Goal Computer ..... **p. 3**
Librairie Informatique d'Aujourd'hui ..... **p. 31**
L'Ordinateur Personnel ..... **p. 90**
L'Ordinateur de poche ..... **p. 30**
Loriciels ..... **p. 15**
Maubert Electronic ..... **p. 27**
PAC+ ..... **p. 25**
Petit Ordinateur Illustré ..... **p. 32**
PSI ..... **p. 11, 12, 13, 18, 88, 89**
Sémaphore ..... **p. 84**
Série III ..... **p. 28**
Sharp ..... **p. 24**
Sinclair ..... **p. 8, 9, 10**
Technology Resources ..... **p. 6, 7**
Vectron - Exelvision ..... **p. 104**
Vidéo Technologie ..... **p. 2**

POURQUOI DIABLE PROGRAMMEZ-VOUS ?

POURQUOI pianoter des heures durant (la nuit parfois) sur le clavier d'une machine ? Vous lirez ici cinq réponses à cette question. Mais peut-être aimez-vous la programmation pour d'autres raisons. Dans ce cas, si cela vous dit, écrivez à LIST. Essayez d'expliquer à votre tour « pourquoi diable programmez-vous ? » Quel démon vous a saisi ?

CETTE SATANÉE MACHINE

EH bien, oui, je le dis : je programme parce que je suis idiot ! Parce qu'un jour j'ai vu des gamins tapoter sur des claviers et faire apparaître sur l'écran des tas de choses. Furtivement, j'ai essayé d'en faire autant, et l'écran m'a retourné "ERROR". C'est par dépit, peut-être, que j'ai décidé alors de tout faire pour être en mesure, un jour, d'épater ces gamins, ou leurs frères cadets.

Depuis ce jour-là, je bosse comme un dingue. Plus de pêche à la truite, plus d'astronomie, plus de polars, je martèle mon clavier. Et à chaque fois que je progresse, que je pense tenir le bon bout, à chaque fois que je m'enfonce dans le duvet de la satisfaction, cette satanée machine m'envoie un message d'erreur en pleine figure ! Alors je sursaute, je jure, et le corps-à-corps avec mon ennemi préféré reprend.

Mais un jour, je l'aurai ! Un jour, je le dominerai, il pliera, il fera exactement ce que je veux. Alors ce jour-là, Ô ce jour-là, je pourrai...

Quelques secondes de patience : mon imprimante crépite, les résultats de mon dernier programme tombent... et vlan ! ERROR évidemment, nomdinpetibonome !

Christian MILDNER

PEUT MIEUX FAIRE

LA vitesse à laquelle les ordinateurs (même "lents") effectuent les calculs est prodigieuse. Je ne m'en suis jamais lassée. Ce sont principalement les nombres premiers que je passe dans ma moulinette informatique. Et peu importe que mes petites recherches demeurent inutiles. Ce qui me plaît, au fond, c'est d'obtenir en quelques jours — et grâce à la machine — des résultats qui m'auraient demandé plusieurs siècles de calcul à la main.

La programmation me paraît présenter une autre caractéristique peu commune : distiller l'intelligence. Sans être une lumière, petit à petit, en retouchant un programme, en l'améliorant, en le remettant cent fois sur le métier, on obtient un produit qui tourne de mieux en mieux.

A chaque retouche, on ajoute quelques astuces. Progressivement, le programme devient de plus en plus malin, de plus en plus habile. Au bout du compte, on finit par en tirer quelque fierté. Et l'on a toute la vie devant soi pour faire encore mieux.

Paulette BESNARD

DOMPTER LA MACHINE

AVANT tout, la programmation est une détente pour moi. Elle permet de concrétiser une idée que l'on a eue sur un sujet ou sur un autre.

Mais c'est aussi une école de patience où chacun peut tester et améliorer son niveau d'ingéniosité. Souvent, on doit lutter contre la machine logique qui se défend à coup de "SYNTAX ERROR", mais quel plaisir quand on l'a enfin maîtrisée et qu'elle obéit docilement... Car la récompense finale reste d'avoir créé un programme qui "marche bien".

Yvon PÉRÈS

OÙ COURS-JE ?

POURQUOI suis-je devenu programmeur ? Il me vient à l'esprit une bonne centaine de réponses possibles ! En voici quelques-unes.

Parce que, de formation littéraire, j'estime que cette discipline ne doit pas être abandonnée à l'impérialisme des maths. Ou bien parce que, professeur de sciences humaines, j'ai besoin dans mon travail de cet outil fantastique qu'est l'informatique, et que, tant qu'à faire, mieux vaut concevoir son logiciel d'éducation soi-même.

Pour le plaisir aussi de se dépasser en se posant un problème compliqué, et pour la joie de triompher de la difficulté. Je ne sais pas très bien après quoi je cours dans la vie, mais je sais que j'y cours très vite, et je sais surtout ce qui me rattrapera dès que je cesserai de courir !

Alain MARIATTE

LA CHASSE A LA BOGUE

CE qui me passionne dans la pratique de l'informatique, c'est de pouvoir analyser un problème, de me plonger dans un monde d'abstraction où seules valent la réflexion et la concentration.

Il y a toujours mille manières d'aborder un problème. Rechercher une solution meilleure que celle à laquelle j'avais songé, la peaufiner, la rendre aussi parfaite que je le peux, voilà ce qui m'apporte les plus grandes satisfactions.

Avec la méthode que j'applique, la phase de programmation proprement dite m'occupe peu de temps (elle a été précédée par une analyse précise). Malgré cela, elle m'apporte aussi des satisfactions. La première, bien entendu, est de voir que mon projet devient réalité. Mais il y a aussi le plaisir de traquer, de dépister, de "mettre la main" sur des erreurs. Vous savez, ces grains de sable qui détraquent une jolie mécanique et que, dans un programme, on appelle des bogues. Plus les bogues sont petites, plus elles sont difficiles à retrouver.

La programmation en langage-machine est un excellent terrain de chasse à la bogue. C'est même, à mon sens, un véritable sport.

Thierry LÉVY-ABÉGNOLI

LA GAZETTE DE LIST

UN LIVRE

Fichiers en Basic par l'exemple
Xavier Gaucherand
et Jean-Pierre Lamoitier
Editions Edimicro
Paris, 1984
Broché, 274 pages
Prix : 148 FF

SI vous avez "toujours voulu en savoir plus sur les fichiers sans jamais avoir osé le demander", ce livre peut vous aider. Accompagnés de beaucoup d'exemples, diagrammes et dessins, les trois premiers chapitres vous expliquent les principes de base de la gestion des fichiers. La table des matières permet d'utiliser cette première partie comme un ouvrage de référence, mais on regrettera l'absence d'un index alphabétique.

La deuxième partie illustre cette théorie à travers cinq exemples complets (explications, organigrammes et programmes adaptés à l'IBM/PC). Consacrer le chapitre suivant aux techniques de protection des fichiers est une très bonne idée. Les problèmes de sécurité informatique sont en effet trop souvent passés sous silence dans les livres destinés au grand public.

Les auteurs proposent en annexe un programme d'encryptage d'un fichier séquentiel, avec commentaires. Une deuxième annexe nous rappelle sous forme de tableau les commandes, instructions et fonctions de gestion de fichiers sous MBASIC. Enfin, les compléments pour TRSDOS et PC/DOS font que cet ouvrage est utilisable sans peine par une bonne partie des programmeurs amateurs.

JL ■

L'Oric-1 et l'Atmos prennent la parole

ORIC FRANCE annonce la commercialisation d'un synthétiseur vocal pour l'Oric-1 et l'Atmos.

Malgré un mode d'emploi en français, ce périphérique bavard conservera certainement l'accent anglo-américain du marché pour lequel il a été initialement conçu. Et cela, même quand on lui demandera de réciter du Molière...

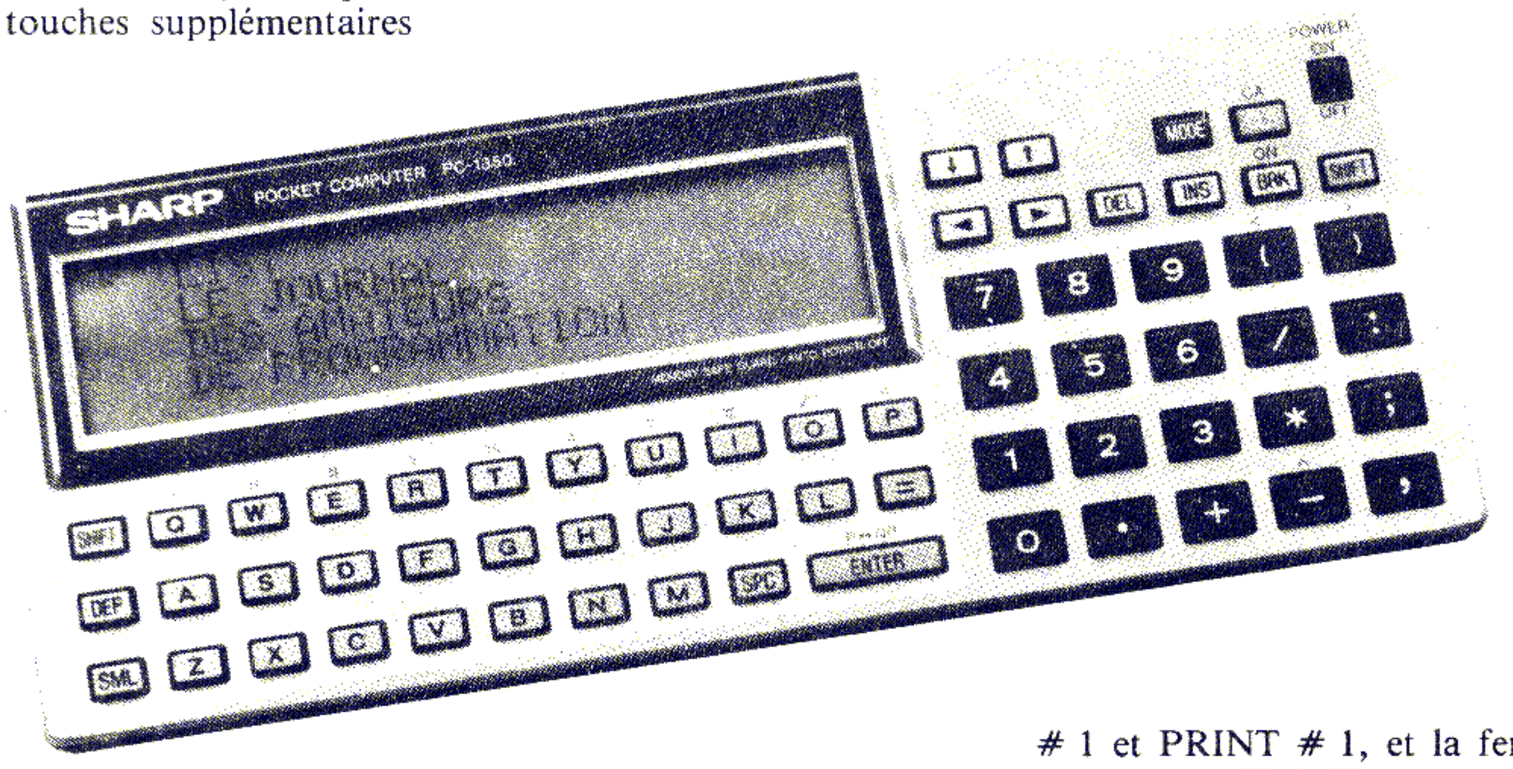
Prix de vente : 450 FF ttc (+ 100 FF pour le cordon de raccordement). ■

Un nouvel ordinateur de poche chez Sharp : le PC-1350

ENCORE un nouvel ordinateur de poche Sharp ! Après les PC-1211, 1212, 1500, 1245, 1251, 1255, 1261, 1401 (et j'en oublie sûrement), voici le PC-1350. La nouveauté la plus apparente est l'afficheur à cristaux liquides : quatre lignes de 24 caractères. C'est un gros atout face aux machines à une ou deux lignes.

Le clavier comporte quelques touches supplémentaires par rapport aux précédents Sharp. C'est ainsi que les caractères souvent employés tels que la virgule, les parenthèses, les deux-points, et surtout l'insertion INS et l'effacement DEL sont ici directement accessibles (sans appui préalable sur SHIFT).

Le nouveau PC a une mémoire vive disponible de 3 070 octets. Cela peut sembler faible, mais il est possible de rajouter une carte mémoire de 8 Koctets (CE-201M) ou de 16 Koctets (CE-202M). Nous disposions de la seconde, et donc de 19 454 octets. Une telle carte a un format "carte de crédit". Elle est plate (3 mm d'épaisseur), et dispose d'une pile intégrée : une fois la carte ôtée du 1350, elle conserve le programme que vous y aviez entré.



Le Basic est semblable à celui du 1261 et est exactement aussi rapide, soit deux à trois fois plus qu'un 1251. De nouveaux ordres ont été toutefois rajoutés à cause de l'afficheur graphique, avec ses 4 800 points adressables (150×32). Chacun des points peut être commandé par PSET et PRESET, alors que la fonction POINT permet de connaître leur état. De plus GCURSOR positionne un curseur à partir duquel on peut tracer des motifs graphiques avec GPRINT. Et la puissante instruction LINE permet de tracer au choix lignes, cadres, ou rectangles pleins, tout cela en continu ou en pointillé définissable !

Autre innovation, une interface RS-232 CMOS intégrée ajoute une nouvelle famille d'ordres Basic. Ainsi l'ordre OPEN permet de définir la vitesse de transmission (300, 600, ou 1200 bauds), la parité, 7 ou 8 bits, 1 ou 2 stop bits, etc. Les échanges de données se font par INPUT # 1 et PRINT # 1, et la fermeture par CLOSE.

Outre cette interface, un autre connecteur permet de brancher une imprimante thermique-interface cassette : le bloc CE-126P.

Tout ce beau matériel devrait être disponible au dernier trimestre 1984. Le prix du PC-1350 de base se situerait aux environs de 2 500 FF.

CB ■

UN LIVRE

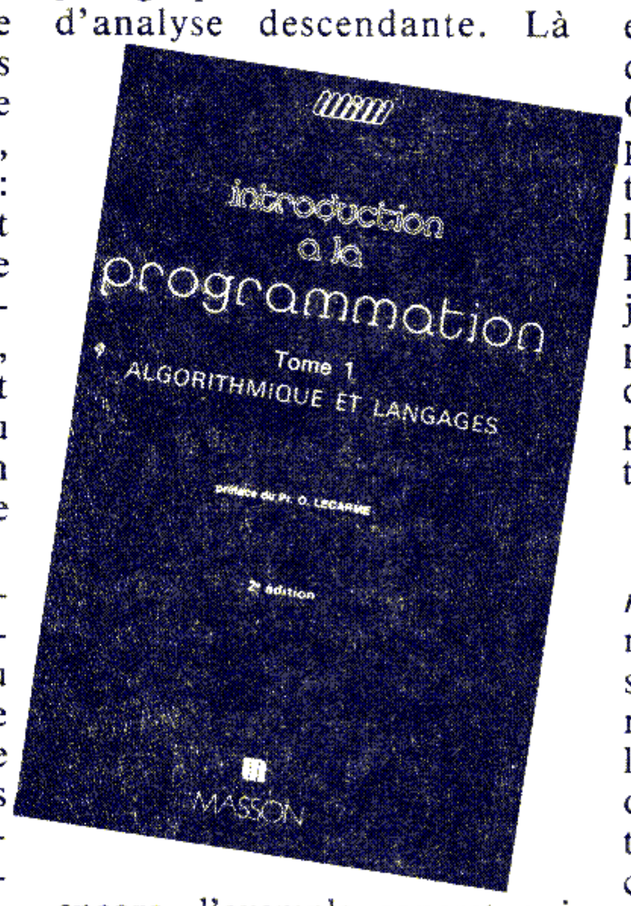
Introduction
à la programmation
Algorithmique
et langages
2ème édition
Joëlle Biondi
et Gilles Clavel
Éditions Masson
Paris 1984
Broché, 249 pages
Prix : 74 FF

INTRODUCTION à la programmation appartient à la série des manuels informatiques **Masson.** C'est un livre universitaire destiné aux étudiants de première année d'IUT (Institut Universitaire de Technologie). Il est donc nettement orienté vers les techniques de gestion, et quand la notion de langage est abordée, il s'agit surtout de Cobol, Fortran, Pascal. Dans le même ordre d'idées, le manuel propose, à chaque chapitre, une série d'exercices, dont le corrigé se trouve dans les dernières pages du livre.

Ceci ne doit nullement décourager le particulier désirant s'initier à l'informatique. En effet, les quatre premiers chapitres de l'ouvrage abordent les notions de base de cette discipline de façon très progressive, et sans « faire de l'informatique » tout de suite, ce qui est une excellente idée : après tout, comme il est dit vers le début, une recette de cuisine peut très bien être considérée comme un programme, alors que la cuisinière est l'équivalent du processeur (au sens d'entité capable de lire un énoncé et d'effectuer la tâche indiquée).

Ainsi, à l'aide de comparaisons imagées, de dessins simples et explicites, puis un peu plus tard d'un pseudo-langage facilement compréhensible (méta-Pascal francisé), les auteurs abordent les principaux objets manipulés en programmation : l'algorithme, l'affectation de variables, les constantes, les vecteurs (les tableaux unidimensionnels, si vous préférez), et enfin, les schémas conditionnels et itératifs.

J'ai beaucoup apprécié la quasi-absence de notation mathématique dans ces chapitres, sauf peut-être dans le paragraphe sur la notion d'analyse descendante. Là



encore, l'exemple concret qui apparaît, et les petits dessins assurent une compréhension aisée du concept.

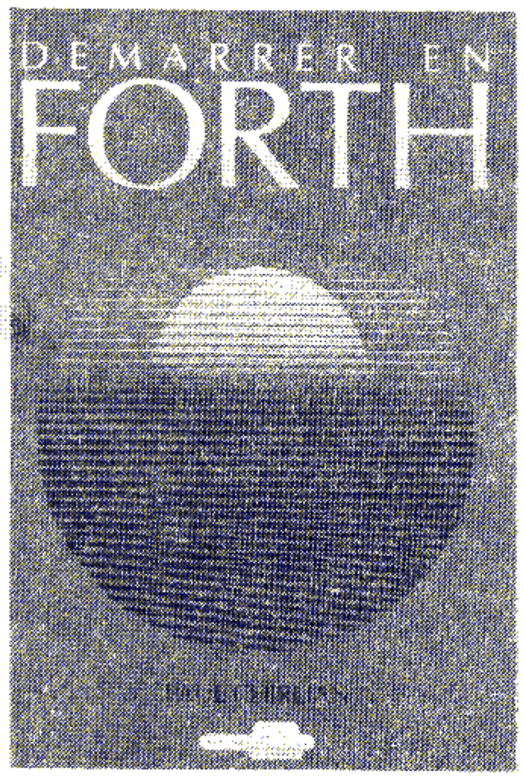
Les derniers chapitres s'intéressent au « processeur habillé », notion imagée désignant un ordinateur, c'est-à-dire un processeur plus un langage de programmation. Cette partie risque d'être moins lisible pour le particulier non-étudiant en gestion, en raison du choix des langages retenus. Cependant, chaque liste de programme illustrant le chapitre est donnée dans les trois langages : Cobol, Fortran, Pascal. L'amateur pourra toujours se référer à ce dernier, et, pourquoi pas, s'initier aux deux autres, même s'ils ne font pas partie de ses préoccupations essentielles.

Introduction à la programmation est principalement un manuel de cours d'IUT, mais ses qualités de clarté lui assureront aussi une audience parmi les amateurs d'informatique, quitte pour ces derniers à sauter les chapitres qui ne les concernent pas directement. Une des vertus de l'ouvrage, et non la moindre, est aussi de rester dans une gamme de prix « très sage », dans un domaine qui réserve rarement ce genre de bonne surprise.

AM ■

UN LIVRE

Démarrer en Forth
Paul Chirlian
Traduit de l'américain
par Claude Nowakowski
et Dominique Petit
Éditions du PSI
Lagny, 1984
Prix : 120 FF



LE bleu étant la couleur des ouvrages consacrés au Forth (allez savoir pourquoi !), vous ne serez pas surpris du bleu profond de ce volume de 256 pages denses.

Dès les premières pages, on rentre dans le vif du sujet. Un exposé rapide de la place de Forth dans les langages de haut niveau précise bien que Forth est différent de tous les autres langages parce qu'il a été conçu pour la rapidité et l'économie de mémoire, et qu'il combine les avantages des langages interprétés et des langages compilés.

L'importance consacrée par l'auteur au maniement des différentes « piles » (pile de données et pile retour, voire pile image dans certaines formes les plus récentes, telles le Forth-Pampuk de l'Hector HRX) place le débat là où il doit l'être : la notation, dite « polonaise inverse » ou encore « post-fixée » est la seule difficulté réelle de Forth. Il y a là une gymnastique à faire, des réflexes nouveaux à acquérir, des hiérarchies d'opérateurs à apprendre qui sont sans doute le seul frein que Forth ait rencontré jusqu'à présent.

Paul Chirlian entraîne son lecteur dans l'arithmétique simple-longueur et double-longueur, les contrôles d'entrées/sorties, la structuration des programmes et les indispensables tableaux, ainsi que le maniement des chaînes de caractères. Tout est dit, ou presque. Car on aborde encore, mais rapidement, la compilation et la greffe sur le dictionnaire des vocabulaires spécialisés.

Bien sûr, on peut rêver ! On pourrait signaler à PSI que les quelques erreurs de typographie qui se glissent dans tout ouvrage sont parfois spécialement gênantes quand on parle de Forth. Dans un exemple, le moindre point, le moindre espace compte. On pourrait aussi, de façon générale, attirer l'attention de tous les auteurs sur l'intérêt qu'il y a à distinguer, et pour les mêmes raisons, la ponctuation du « texte Forth » et celle du commentaire.

A côté d'exercices astucieux, mais donnés sans leurs solutions, on trouve encore de très nombreux programmes-exemples accompagnés d'explications « fouillées ».

Au total, un excellent manuel, et qui plus est, solide grâce à sa reliure cousue-collée. Même après de mauvais traitements, il ne laissera pas échapper de feuilles volantes.

CM ■

Un petit tour chez le libraire

Programmation : introduction théorique en vue de la pratique
F.H. Raymond
Editions Masson
Paris, 1984
Broché, 184 pages
Prix : 95 FF

ZX Spectrum, votre micro-ordinateur
Serge Pouts Lajus
Editions Cedic/Nathan
Paris, 1984
Broché, 128 pages
Prix : 35 FF

Dictionnaire de la Micro-Informatique
Français/Anglais
Edité par Franterm
Diffusé par Nathan
Paris, 1984
Broché, 136 pages
Prix : 146 FF

Programme interne du Commodore 64
Milton Bathurst
Editions DataCap
Diffusé par PSI
Belgique, 1984
Broché, 252 pages
Prix : 130 FF

Clefs pour l'Oric (Oric 1 et Oric Atmos)
Emmanuel Flesselles
Editions du PSI
Lagny, 1984
Reliure spirale, 116 pages
Prix : 100 FF

Assembleur du TRS 80
Daniel Ranc
Editions Techniques et Scientifiques Françaises
Collection Poche Informatique
Paris, 1984
Broché, 128 pages
Prix : 35 FF

Le module 16K d'Alice

IL y avait si longtemps qu'on l'attendait, on finissait par ne plus y croire. Il sera là à Noël, nous disaient-ils ; puis en février...

C'est finalement en juillet qu'il arrive, ce module de mémoire de 16 Koctets qui devrait enfin permettre à Alice de rejoindre le clan des ordinateurs familiaux confortables.

Il faut dire que la pauvre Alice, pourtant bien née, semblait un peu délaissée par ses deux géniteurs, M. Matra et Mme Hachette : les logiciels régulièrement annoncés ne sont pas toujours disponibles, et les acheteurs en étaient finalement réduits à faire eux-mêmes leurs programmes. Une démarche somme toute assez logique pour un appareil qui proposait une initiation à la programmation, mais tout de même, c'est un peu frustrant par rapport à ceux qui nous narguent avec leur *Grand Prix* ou leur *Glouton,* non ? Aussi, on se lance et on fait son propre programme d'*envahisseurs de l'espace* et patatras... Nous voilà bloqués par les tout petits 4 Koctets de mémoire disponible.

Alors 16 Koctets, quelle aubaine ! Et puis, sur le dépliant publicitaire, certaines promesses alléchantes : des instructions cachées du Basic, des possibilités de graphisme haute résolution, comment avoir des touches à répétition... De ce côté-là, il faut bien dire que la déception est sérieuse : le miniguide livré avec l'extension, intitulé "Aller plus loin avec Alice", n'a rien à voir avec le manuel d'origine. Il ne s'adresse pas du tout à un débutant, et surtout, les quelques renseignements qu'il dévoile comme à regret auraient pu, auraient dû, dirais-je, figurer normalement

dans une annexe en fin du manuel de base. Le fait d'avoir ou non l'extension mémoire n'intervient pratiquement pas dans les explications fournies.

Prenons quelques exemples, pour mieux comprendre : les trois instructions "cachées" du Basic sont EXEC (pour exécuter un programme en assembleur), CLEAR (pour réserver de la place à ce programme) et CLOADM (pour charger en mémoire un programme en assembleur). Les acheteurs de cassettes un peu curieux les avaient sans doute déjà découvertes depuis un certain temps, et elles sont tout à fait utilisables sans le module.

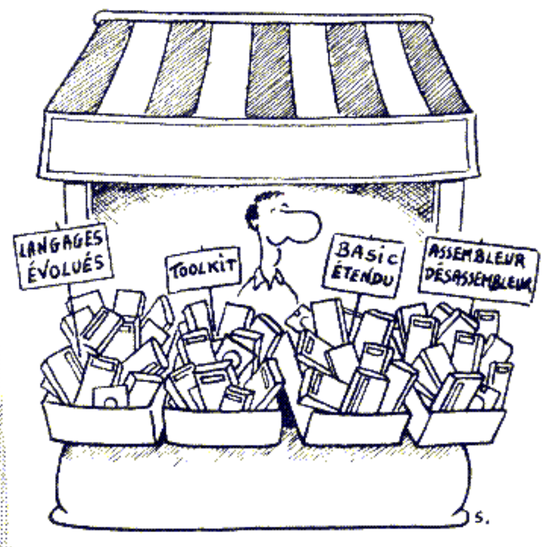
Quant au graphisme haute résolution... D'accord, il y a quelques renseignements intéressants sur le circuit générateur vidéo MC 6847, mais aucun ne concerne la gestion d'écran, et il n'y a pas un véritable programme de démonstration de la haute résolution (il y en a bien un qui porte ce titre, mais il se contente apparemment de faire des dessins très fantaisistes sur les quatre lignes du haut de l'écran et il disparaît dès que l'on cherche à sortir pour reprendre le contrôle).

Dans le même ordre d'idée, il n'est pas vraiment nécessaire de disposer du module 16K pour apprendre que l'on peut sauvegarder des données ou des tableaux alphanumériques tout simplement en les transformant en numérique par leur code ASCII.

Quelques petits renseignements utiles pourront être trouvés de-ci, de-là, mais ce manuel est tout de même bien décevant. Pour ma part, je ne trouve finalement qu'un mérite à cette extension mémoire : c'est d'être une extension mémoire, purement et simplement. Alice dispose alors de 19 Koctets de mémoire vive, ce qui permet de faire du travail sérieux.

On regrettera que Matra n'ait pas rédigé un vrai manuel, dans l'esprit du manuel de base qui était tout à fait adapté à l'appareil. Dommage aussi qu'à une époque où l'on peut trouver une extension mémoire de 64 Koctets pour environ 800 FF (celle de l'IBM PC), Matra choisisse de vendre la sienne à un prix de 595 FF. Cela pourrait bien rebuter nombre d'acheteurs potentiels. Si au moins, pour le prix, on pouvait disposer d'un véritable éditeur !

JD ■



Cassettes et disquettes

PC-Vision
Cassette pour PC-1500
et Tandy PC-2
Éditeur pleine page pour
ordinateur de poche
Édité par Pocket Soft
Distribué par X-LOG
Prix : moins de 300 FF

J'apprends le langage Forth
Deux cassettes pour Oric-1,
compatibles Atmos
Édité par ARG Informatique
Distribué par Innelec
Prix : 200 FF

J'apprends la CAO
Cassette pour Oric-1/Atmos
Initiation à la Conception
Assistée par Ordinateur
Édité par Loriciels
Prix : 180 FF

Éditeur musical
Cassette pour Oric-1/Atmos
Édité par Loriciels
Prix : 95 FF

Calc
Cassette pour PB-700
Tableau de calculs
Édité par Logi'stick
Distribué par DDI
Prix : 140 FF

Cassettes et disquettes

Supercode
Cassette pour ZX Spectrum 16 et 48 Ko
100 routines en langage-machine réutilisables dans les programmes en Basic ou en langage-machine
Édité et distribué par Sémaphore
Prix : 150 FF

D Bug
Cassette pour Oric-1/Atmos
Moniteur-debugger
Édité par No man's land
Distribué par Innelec
Prix : 140 FF

Master 64
Disquette pour Commodore 64
Extension du Basic
Édité par Micro Application Software
Distribué par Procep
Prix : 950 FF

Assembleur symbolique
Cassette pour Sega 3000
Édité par Loriciels
Prix : 260 FF

MCoder 11
Cassette pour ZX 81 16 Ko
Cassette pour ZX Spectrum 48 Ko
Traduit les programmes Basic en langage-machine
Édité par Personal Software Services
Distribué par Innelec
(Manuels en anglais)
Prix de chaque cassette : 120 FF

Tiny Forth
Disquette pour Commodore 64
Édité par Micro Application Software
Prix : 490 FF

Graphe
Cassette pour PB-700
Graphiques statistiques
Édité par Logi'stick
Distribué par DDI
Prix : 140 FF

Virgule
Disquette pour Commodore 64
Disponible sur cassette
Traitement de texte
Édité et distribué par Micro Application Software
Prix : 750 FF

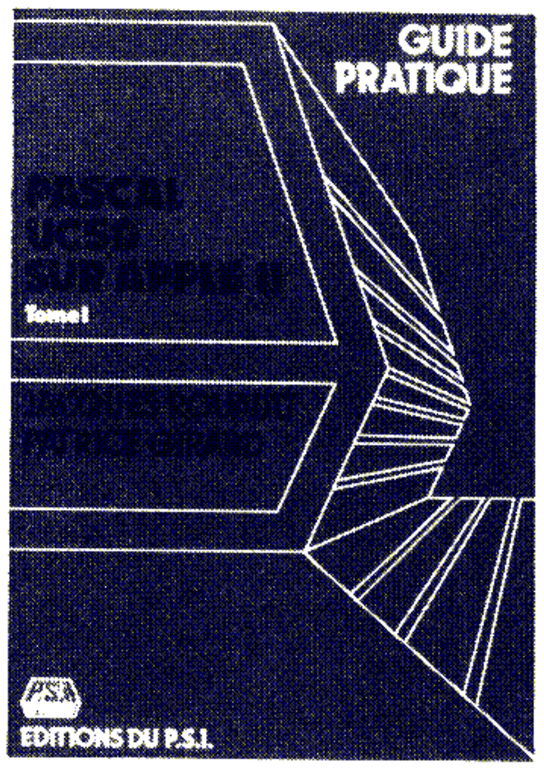
UN LIVRE

Pascal UCSD sur Apple II
Jacques Rouault et Patrice Girard
Éditions du PSI
Collection Guide Pratique
Lagny, 1984
Broché, 228 pages
Prix : 110 FF

Ce livre tient les promesses de son titre. Il traite en effet du Pascal, du système UCSD et de l'Apple II. Il est divisé en deux parties. La première expose la base du système UCSD, alors que la seconde traite du langage Pascal vu sous l'angle UCSD.

Il est surtout destiné aux débutants rebutés par la lecture (en anglais) du *Apple Pascal - Language Reference Manual*, mais qui ont utilisé le système UCSD pendant quelques heures. Toutefois, les utilisateurs connaissant bien ce système y trouveront des renseignements très intéressants et même de véritables utilitaires.

Le seul défaut de ce livre est certainement son manque de structure, ce qui est le comble pour un ouvrage exposant les principes d'un système d'exploitation écrit dans un langage structuré. Il vous faudra donc passer d'un chapitre à l'autre selon les notions que vous désirerez voir abordées, sans qu'un



sommaire détaillé ou un index soit là pour vous aider.

L'ordre de lecture qui nous paraît plus logique est celui-ci.

Les chapitres 1, 2 et 3 présentent le matériel, le système UCSD et le langage Pascal. Ils sont courts, mais fournissent une bonne introduction.

On peut ensuite passer directement à la seconde partie du livre, qui permet de découvrir le langage Pascal, sous son implémentation UCSD. De très nombreux exemples, ni trop simples, ni trop compliqués, illustrent ces notions. Ils devraient permettre au lecteur de progresser rapidement.

A chaque étape sont indiquées les limitations du système qui sont autant de petits pièges dans lesquels chacun finit par tomber. Par exemple, vous découvrirez en page 114 pour quelle raison le programme de calcul de racine carrée vous donne 51 comme racine de 68 137, alors que 51 * 51 = 2 601.

Les bogues du système sont également finement analysées. Elles paraissent parfois tellement incroyables que nous nous sommes fait un devoir de vérifier qu'elles existaient. Hélas, elles existent bel et bien : les programmeurs du Pascal UCSD pour Apple II pourraient lire ce livre avec profit, si ce n'est déjà fait...

Le chapitre 4 traite du compilateur, en décrivant ses différentes options. Ici encore, de bons exemples sont fournis. On comprend rapidement le fonctionnement des diverses commandes.

Enfin, le chapitre 5 présente de véritables utilitaires qui, s'ils sont faciles à utiliser, ne sont pas évidents à comprendre, puisqu'ils modifient des zones réservées au système. Bien qu'il s'intitule « Programme de mise en route », ce chapitre pourra très bien être lu en dernier, si l'on accepte quelques inconvénients mineurs tels que le manque de contrôle du « reset », le non-fonctionnement de deux touches de déplacement du curseur, ou la mauvaise initialisation de l'imprimante.

Ce livre doit, à notre avis, se placer dans la bibliothèque de l'utilisateur du système Pascal UCSD. Il ne remplace pas le manuel de référence du langage, mais il le complète utilement. Bien qu'il traite de la version 1.1 du système, il reste entièrement valable pour la nouvelle version 1.2 sortie récemment.

TC ■

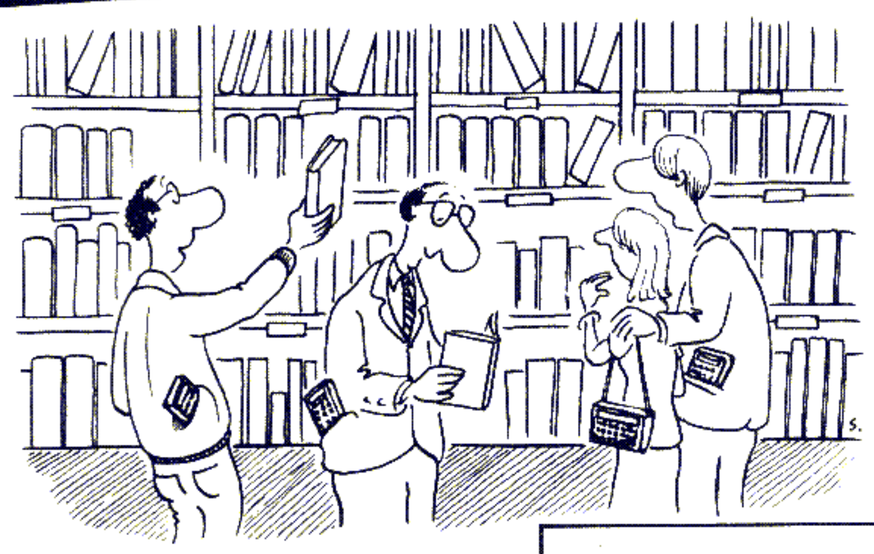
Deux ans de garantie, qui dit mieux ?

LA firme américaine **Texas Instruments** garantit désormais pendant deux ans (pièces et main-d'œuvre) toutes ses calculatrices, depuis la simple "4 opérations" jusqu'à la calculatrice haut de gamme. Et cela vaut pour tout achat effectué depuis le 1er mars 1984.

Tout produit défectueux devrait être purement et simplement remplacé — en échange standard — à l'endroit où il a été acheté. Plus de crainte donc, en principe, pour un clavier défaillant ou un afficheur qui flanche.

On ne peut que se féliciter de cette initiative. Souhaitons qu'elle soit suivie par les autres constructeurs. Dans le domaine des calculatrices ou des ordinateurs, il est rare en effet que la garantie s'étende sur plus de six ou neuf mois... ■

Un petit tour chez le libraire



Le langage machine du ZX Spectrum
Ian Stewart et Robin Jones
Editions Cedic/Nathan
Paris, 1984
Broché, 144 pages
Prix : 78,50 FF

Pratique du Commodore 64 et du portable S/DX-64
Henri Lilen
Editions Radio
Paris, 1984
Broché, 174 pages
Prix : 100 FF

Jeux sur Philips C 7420 Videopac + Extension Basic pour console de jeu
Christophe Bardon
et Benoît de Merly
Editions Edimicro
Paris, 1984
Broché, 200 pages
Prix : 98 FF

Le guide du Spectravidéo
Daniel Lemahieu
et Etienne Dubois
Editions du PSI
Lagny, 1984
Broché, 174 pages
Prix : 100 FF

Applications du Z80
James W. Coffron
Editions Sybex
Paris, 1984
Broché, 314 pages
Prix : 198 FF

Pratique du micro-ordinateur Tandy MC-10 Méthode progressive avec 20 programmes
Henri Lilen
Editions Radio
Paris, 1984
Broché, 158 pages
Prix : 100 FF

UN LIVRE

Le Basic
bien programmé
A.P. Stephenson
Traduit de l'anglais par
G. Jastrzeb
Editions Dunod
Paris, 1984
Broché, 120 pages
Prix : 65 FF

CONTRAIREMENT à ce que pourrait laisser penser le titre (*Le Basic bien programmé*), il ne s'agit pas là d'un manuel de perfectionnement, mais d'un livre d'initiation. La traduction du titre anglais (*Beginner's guide to basic programming*) nous a paru bizarre au vu du contenu. L'auteur, en effet, insiste sur le fait que, pour un débutant, avoir écrit un programme qui "tourne" est beaucoup plus important que d'avoir obéi aux règles de la programmation structurée. Il considère que vouloir obéir à ces règles dès l'apprentissage peut être source de découragement pour le programmeur novice.

Voilà donc l'idée qui sous-tend la progression du livre et elle peut se défendre s'agissant d'un manuel d'initiation (on en a parfois assez d'entendre les professionnels chanter les louanges de la programmation structurée). L'objectif premier du grand débutant est souvent le résultat immédiat.

L'auteur essaie de ne pas se référer à une machine ou à un Basic particulier, et il y réussit assez bien. La progression est habilement menée (des passages sur le fonctionnement interne de la machine donnent envie d'aller plus loin).

On passe rapidement (trop ?) sur les fichiers, mais il est vrai que, dans ce domaine, les différences d'un Basic à l'autre sont très importantes. Le dernier chapitre (petit guide pratique de programmation) paraît fort utile au stade de l'apprentissage.

En résumé, un livre qui pourrait convenir à plus d'un débutant en informatique.

JL ■

Du côté des clubs

Si tous les epsonistes voulaient bien...

DEPUIS le début de 1984, l'APBLUTH édite un petit bulletin de liaison destiné aux utilisateurs du HX-20 ou de tout autre modèle compatible.

Pour tout renseignement sur cette association :
APBLUTH
65 rue des Fleurs
73000 Chambéry

Ciel ! Un club à Tahiti...

SOUS les tropiques, au pays des vahinés et des lagons transparents, on pratique l'informatique avec autant d'intérêt qu'en France métropolitaine. Le CIEL (bleu, bien sûr), ou Club d'Informatique et d'Electronique de Loisirs, regroupe plus d'une centaine de personnes de 7 à 77 ans.

Le club est équipé de quatre « grosses » machines et de plusieurs autres plus modestes. Il se réunit (les lundi, mercredi et jeudi soir) dans les locaux d'une école primaire, à Tipaerui-Plage, en la bonne ville de Papeete. Qu'on se le dise dans la région. *LIST* adresse son salut à ses amis du bout du monde.

Pour tout renseignement :
Club CIEL
BP 4460
Tahiti
Polynésie française

Carnet rose

En Seine-Saint-Denis

A Drancy, le club ALIF paraît démarrer sur les chapeaux de roues. Il s'adresse à la fois aux débutants et aux amateurs chevronnés.

Micro-club ALIF
54 avenue Henri Barbusse
93700 Drancy
Tél. : 832 10 44

Dans le Gard

LE Club Oric de Saint-Chaptes accueille ses nouveaux adhérents sans leur demander de cotisation : il suffit d'y apporter sa propre passion. Les réunions ont lieu les dimanche après-midi.

Club Oric de Saint-Chaptes
rue des Mases
30190 St Chaptes
Tél. : (66) 81 25 88

Dans les Hauts-de-Seine

LE nouveau Club Informatique Régional de Courbevoie (alias CIROCO) a, lui aussi, le vent en poupe, et ce ne sont pas les projets qui lui manquent.

Il tiendra son assemblée générale le 11 septembre à 20 h 30, au stade municipal de Courbevoie. Tous les curieux y sont invités.

CIROCO
12 rue Carpeaux
92400 Courbevoie
Tél. : 333 38 83

Pour les amateurs de Forth

UN groupement d'utilisateurs du langage Forth, le GULF, a tenu sa première réunion fin juin à Paris.

A souligner une excellente disposition inscrite dans les statuts de cette association : tous les membres du GULF sont convenus de proscrire les copies illégales de logiciels commerciaux.

Comment ne pas approuver cette prise de position contre le piratage des programmes ?

GULF
14 place Gabriel Péri
75008 Paris
Tél. : 293 35 54

Si vous aimez les maths...

Si vous aimez les maths, par curiosité, pour le plaisir de découvrir des problèmes étonnants, trois ouvrages méritent votre attention.

Math'Circus
Martin Gardner
Traduit de l'américain par
Jean-Pierre Labrique
Bibliothèque Pour la Science
Diffusé par Belin
Paris, 1982
Relié, 144 pages
Prix : 62 FF

Math'Festival
Martin Gardner
Bibliothèque Pour la Science
Diffusé par Belin
Paris, 1981
Relié, 176 pages
Prix : 62 FF

Mathématiques venues d'ailleurs
Adapté du russe par
Jean-Michel Kantor
Éditions Belin
Paris, 1982
Broché, 160 pages
Prix : 72 FF

UN LIVRE

Programmer chez soi : le Basic
Ilya Virgatchik
Editions Marabout
Paris, 1983
Broché, 256 pages
Prix : 24 FF

VOICI un ouvrage d'initiation bon marché, plutôt classique et assez complet. L'auteur commence par décrire un petit système informatique (matériel et logiciel), puis il nous délivre un cours de Basic où l'on aurait aimé trouver un peu de cet humour qui permet d'apprendre en s'amusant.

Les instructions et les fonctions sont décrites une par une, agrémentées de quelques exemples un peu courts. On y trouve également une présentation des ordinogrammes.

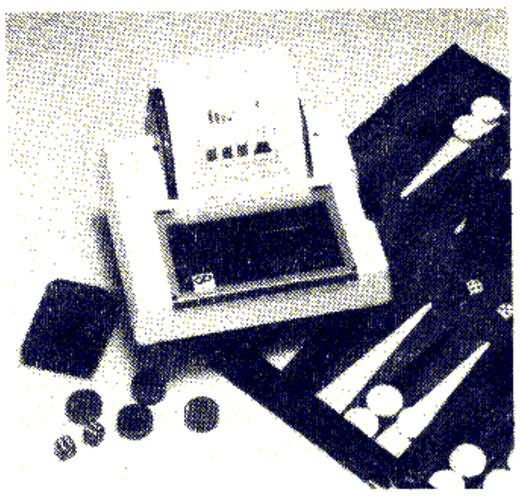
A la suite de ce cours sont proposés 22 proprammes de calcul, de logique et de gestion. Quelques pages évoquent différents Basics courants sur le marché, et un chapitre est consacré à la description de dix ordinateurs familiaux.

Dernier point, très intéressant, l'ouvrage se termine sur deux lexiques, l'un des termes de l'informatique et l'autre des mots clés du Basic.

JL ■

Des imprimantes pour les petits budgets

LA **Seikosha** GP-50A est une imprimante matricielle (5 points sur 8 et 46 colonnes) utilisant du papier ordinaire en rouleau. Elle fonctionne à la vitesse de 40 caractères à la seconde et permet l'impression en double largeur ainsi que les graphismes point par point. L'interface est au standard *Centronics*.



Il en existe une version spécialement adaptée au Spectrum (la GP-50S) qui n'imprime que sur 32 colonnes et à la vitesse de 25 caractères à la seconde.

Les dimensions de ce périphérique sont modestes (215 × 250 × 85 mm), et le prix l'est aussi : environ 1 400 FF ttc. ■

PROGRAMME AUTODIDACTE...

QUAND UN ORDINATEUR APPREND LE TIC-TAC-TOE

AU début, votre ordinateur ne connaît que les règles du jeu. C'est un piètre adversaire. Puis, peu à peu, à force de jouer, il s'améliore. Le programme ne se transforme pas, mais il tient compte des parties précédentes...

Si vous ne connaissez pas encore ce grand classique qu'est le jeu de tic-tac-toe, vous en apprendrez les règles en moins d'une minute. Elles ont été rappelées dans l'encadré ci-dessous et sont élémentaires. Les parties se déroulent à deux, assez souvent en cachette, au fond d'une salle de classe.

A défaut d'un adversaire humain, on peut aussi se mesurer à un ordinateur. Mais il y a mieux. On peut en effet faire en sorte que le programme tire les leçons des parties qu'il a déjà disputées et soit, en quelque sorte, capable d'un apprentissage. Le mécanisme utilisé est en fait très simple.

Au début, la machine joue en appliquant — bêtement — les règles du tic-tac-toe et elle constate la victoire de l'un des deux camps ou, éventuellement, le match nul. Puis elle modifie son jeu en fonction des résultats obtenus.

Un damier de tic-tac-toe est un carré de trois cases sur trois dont chacune peut être vide ou occupée par l'un ou l'autre des deux adversaires (nous les désignerons par X et Y). Un premier calcul nous indique donc un maximum théorique de 3^9, soit 19 683 positions différentes. En fait, beaucoup de ces positions ne peuvent pas être rencontrées au cours d'une partie réelle, et cela pour deux raisons au moins :

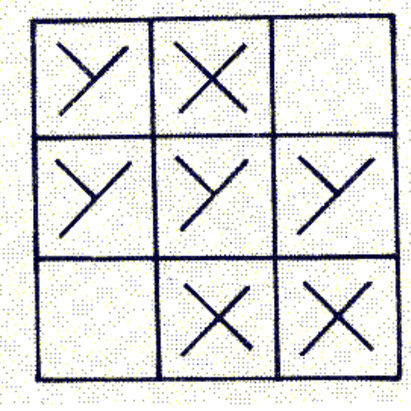
- les joueurs occupent une case à tour de rôle : la différence entre le nombre des X et celui des Y sur le damier ne peut qu'être égale à zéro ou à un ;
- la partie s'arrête dès qu'un joueur a gagné (La Palisse) ; on ne peut donc trouver à la fois un alignement de trois X **et** de trois Y.

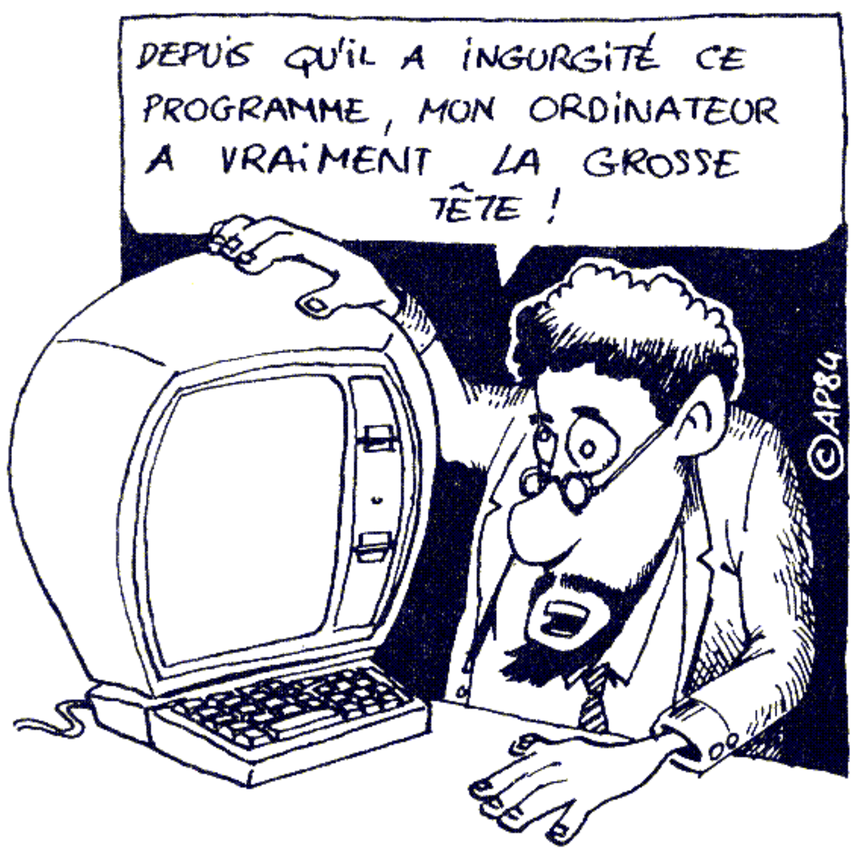
On pourrait d'ailleurs réduire encore le nombre des positions différentes en considérant les symétries existant entre certaines d'entre elles.

Pendant chaque partie, le programme mémorise les positions successives, puis il recherche la note attribuée à chacune de ces positions (1). Il augmente la note d'un point si cette position a finalement conduit à la victoire du camp qui l'a jouée, et il la diminue d'autant dans le cas contraire

Règles du tic-tac-toe

LE jeu se dispute sur un échiquier de trois cases sur trois. Il oppose deux joueurs qui déposent, chacun à son tour, un pion à sa couleur sur une case encore vide. On gagne en alignant trois pions de son camp soit horizontalement, soit verticalement, soit en diagonale. Sur la figure ci-dessous, la victoire est allée aux Y.





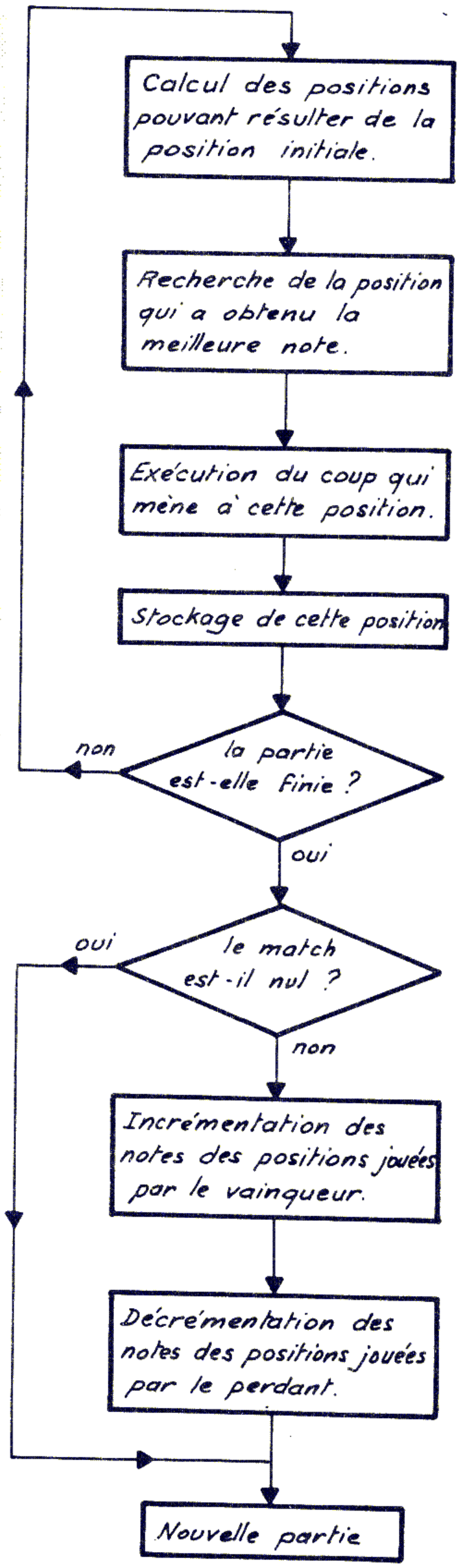
(1) Au début, toutes ces notes sont nulles, et le programme joue au hasard.

PROGRAMME AUTODIDACTE...

TIC-TAC-TOE

Tic-tac-toe, un programme qui progresse
Basic Microsoft
Auteur Thierry Lévy-Abégnoli


```
10 CLS:REM EFFACE L'ECRAN
97 REM-------------------------
98 REM INITIALISATION DEFINITIVE
99 REM-------------------------
100 DIM JE(8),C(1),CE(1,8),MX(9),M(9),N(9),POX(9841),NJ(1),J(8),X(
7),Y(7),Z(7),A$(2)
110 DATA 1,2,3,6,9,18,27,54,81,162,243,486,729,1458,2187,4374,6561
,13122
120 FOR I=0 TO 8:READ CE(0,I):READ CE(1,I):NEXT I
130 J(0)=0:J(1)=1:J(2)=0:J(3)=1:J(4)=0:J(5)=1:J(6)=0:J(7)=1:J(8)=0
140 C(0)=1:C(1)=-1:NJ(0)=1:NJ(1)=2
150 DATA 0,2,1,3,5,1,6,8,1,0,6,3,1,7,3,2,8,3,0,8,4,2,6,2
160 FOR I=0 TO 7:READ X(I):READ Y(I):READ Z(I):NEXT I
170 A$(0)=" ":A$(1)=" X":A$(2)=" Y":I=0
197 REM-------------------------
198 REM PRESENTATION DES OPTIONS
199 REM-------------------------
200 PRINT"VOULEZ-VOUS":PRINT"-  QUE JE JOUE CONTRE MOI (1)":PRINT"
-  JOUER CONTRE MOI (2)":INPUT A:A=A-1
210 IF A=1 THEN 250
220 IF A<>0 THEN 200
230 PRINT" POUR REVENIR AU MENU, PRESSEZ UNE TOUCHE, PUIS ATTENDEZ
 LA FIN DE LA PARTIE EN COURS"
240 GOTO500
250 INPUT "VOULEZ-VOUS COMMENCEZ (O OU N)";A$
260 IF A$="O" GOSUB 1500:GOTO 590
270 IF A$="N" THEN 500
280 GOTO 250
297 REM-----------------------------------
298 REM S/P INITIALISATION AVANT CHAQUE PARTIE
299 REM-----------------------------------
300 FOR I=0 TO 8:JE(I)=0:NEXT I
310 P=0:I=0:DR=0:V=0:Q=0
320 RETURN
497 REM-------------------------
498 REM DETERMINATION DU COUP JOUE
499 REM-------------------------
500 S=-1000:H=0:FOR I1=0 TO 8
510 IF JE(I1)<>0 THEN 550
520 CO=P+CE(J(I),I1)
530 GOSUB 1100
540 IF RP=S LET MX(H)=I1:H=H+1
545 IF RP>S LET S=RP:MX(0)=I1:H=1
550 NEXT I1:MC=MX(RND(H)-1):PRINT"JE JOUE EN";MC+1;":"
560 GOSUB 700
570 IF DR=1 THEN 630
580 IF A=0 THEN 620
590 INPUT "OU JOUEZ-VOUS ";MC
600 IF MC<1 OR MC>90 THEN 590
605 IF JE(MC-1)<>0 THEN 590
610 MC=MC-1:GOSUB 700
620 IF DR=0 THEN 500
630 GOSUB 300
640 IF A=0 THEN 660
650 GOTO 200
660 IF INKEY$<>"" THEN 200
670 GOTO 500
696 REM---------------
697 REM S/P COUP JOUE
698 REM---------------
699 REM------MODIFICATION DU JEU
700 P=P+CE(J(I),MC):Q=Q+CE(1-J(I),MC):JE(MC)=NJ(J(I)):GOSUB1500
710 GOSUB 900:IF V=1 THEN 760
720 IF I<8 THEN 750
730 PRINT"MATCH NUL":DR=1:RETURN
750 M(I)=P:N(I)=Q:I=I+1:M(I)=0:N(I)=0:RETURN
758 REM-----MODIFICATION DE SES 'CONNAISSANCES'
759 REM-----EN CAS DE VICTOIRE D'UN CAMP
760 C=C(J(I))
770 FOR I1=0 TO I STEP 2
790 CL=C:CO=M(I1):GOSUB 1300:CL=C:CO=N(I1):GOSUB 1300
```



Organigramme du programme quand il joue contre lui-même.

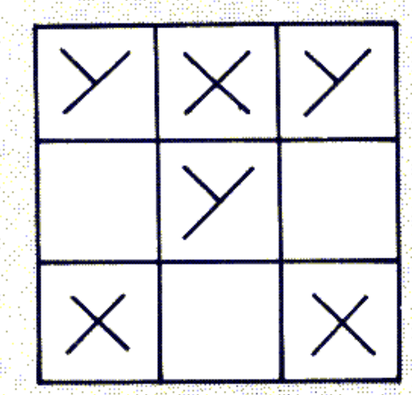
Représentation en mémoire d'une position

LE terrain de jeu est composé de neuf cases dont chacune peut être vide ou contenir soit un X, soit un Y. Nous choisirons de représenter un vide par le chiffre 0, un Y par le chiffre 1 et un X par le chiffre 2. Avec ces conventions, la position ci-dessous sera représentée par le nombre 121010202 en base 3, soit, en base 10 : $2\times3^0 + 0\times3^1 + 2\times3^2 + 0\times3^3 + 1\times3^4 + 0\times3^5 + 1\times3^6 + 2\times3^7 + 1\times3^8 = 11\,765$.

Si cette position a mené à la défaite du camp qui l'a jouée, l'élément de tableau numéro 11 765 est diminué d'une unité. Pour des raisons de place en mémoire, chaque élément du tableau SC(i) contient les notes de deux positions, l'accès à l'une ou l'autre se fait dans le sous-programme de la ligne 1100. Chaque élément SC(i) est compris entre -32768 et $+32767$ (entier codé sur deux octets) ; les deux notes S_j et S_{j+1} sont telles que $SC(i) = S_j + 256\times S_{j+1}$ avec S_j et S_{j+1} compris entre -128 et $+127$. Chaque note est donc finalement codée sur un octet. Une solution beaucoup plus simple aurait consisté à utiliser les fonctions PEEK et POKE, mais cela aurait rendu plus délicate l'adaptation du programme d'une machine à l'autre.

1	2	3
4	5	6
7	8	9

Numérotation des cases



Position étudiée

1	2	1
0	1	0
2	0	2

Codage de l'état des cases

```
800 CL=-C:CO=M(I1+1):GOSUB 1300:CL=-C:CO=N(I1+1):GOSUB 1300
820 NEXT I1
830 DR=1:RETURN
897 REM------------------
898 REM TEST DE VICTOIRE
899 REM------------------
900 FOR I2=0 TO 7:N=0
910 FOR I3=X(I2) TO Y(I2) STEP Z(I2)
920 IF JE(I3)=0 OR JE(I3)<>JE(X(I2)) LET I3=8:GOTO 940
930 N=N+1:IF N=3 LET V=1:I2=8:I3=8:PRINT"LES";A$(NJ(J(I)));" ONT G
AGNE !"
940 NEXT I3
950 NEXT I2
960 RETURN
1095 REM-----------------------------------------
1096 REM S/P LECTURE DE LA NOTE D'UNE POSITION:
1097 REM          EN ENTREE:POSITION CO
1098 REM          EN SORTIE:NOTE RP
1099 REM-----------------------------------------
1100 IT=INT(CO/2)
1110 T1=PO%(IT)
1120 Y=INT(T1/256)
1130 X=T1-256*Y
1140 IF X>127 LET X=X-256
1150 IF Y>127 LET Y=256-Y
1160 IF IT=CO/2 LET RP=X:RETURN
1170 RP=Y:RETURN
1295 REM-----------------------------------------
1296 REM MODIFICATION DE LA NOTE DE LA POSITION CO:
1297 REM    INCREMENTATION SI CL=1
1298 REM    DECREMENTATION SI CL=-1
1299 REM-----------------------------------------
1300 GOSUB 1100
1310 IF RP=-128 AND CL=-1 RETURN
1320 IF RP=127 AND CL=1 RETURN
1330 IF IT<>CO/2 LET CL=256*CL:GOTO 1380
1340 IF CL=1 THEN 1370
1350 IF RP=0 LET CL=255
1360 GOTO 1380
1370 IF RP=255 LET CL=-255
1380 PO%(IT)=PO%(IT)+CL:RETURN
1497 REM------------------
1498 REM AFFICHAGE DU JEU
1499 REM------------------
1500 PRINT"-------------"
1510 FOR I4=0 TO 6 STEP 3
1520 FOR I5=0 TO 2
1530 IF JE(I4+I5)=0 LET V$=STR$(I4+I5+1):GOTO 1550
1540 V$=A$(JE(I4+I5))
1550 PRINT"!";V$;" ";
1560 NEXT I5
1570 PRINT"!"
1580 NEXT I4
1590 PRINT"-------------"
1600 RETURN
```

(voir encadré ci-contre). En cas d'ex-æquo, il laisse la note telle qu'elle était.

Pour choisir ce qu'il va jouer, le programme passe en revue les différents coups légaux, il calcule les positions qui en résultent, puis il joue le coup qui a obtenu le plus de points jusqu'à présent. Si la plus forte note a été attribuée à plusieurs coups, il en choisit un au hasard pour le jouer.

Le programme peut, si vous le voulez, jouer contre lui-même et s'auto-perfectionner sans que vous n'ayez à intervenir. Il le fera au rythme d'environ trois parties à la minute. Quand vous déciderez de l'affronter ensuite, vous découvrirez un adversaire qui a fait des progrès.

Ce mode d'auto-apprentissage reste pourtant assez lent : le programme piétine un peu dans les premiers temps. Il s'aperçoit tout de même assez vite que jouer la case du centre mène souvent à la victoire. Après plusieurs heures, il devient assez difficile à battre, sauf quand il rencontre des positions nouvelles.

A ce stade, il faut d'ailleurs signaler un phénomène qui survient parfois. Quand le programme fait une partie nulle au cours de laquelle il n'a pas eu à choisir un seul coup au hasard, il n'apprend rien et rejoue indéfiniment la même partie. Il faut alors reprendre le contrôle et livrer une ou deux parties contre lui.

Si vous disputez vous-même les parties contre l'ordinateur, l'apprentissage sera d'autant plus rapide que vous serez un joueur habile. Pour vous en convaincre, faites une trentaine de parties en tenant le même camp et en jouant le mieux possible, puis changez de camp. Vous verrez que le programme a assimilé en partie votre façon de faire...

Thierry LÉVY-ABÉGNOLI

DE FORTRAN II A FORTRAN V

APRÈS le langage-machine, un langage évolué fait son apparition : Fortran. Essentiellement tourné vers les applications scientifiques, ce langage a subi de nombreuses évolutions. Après Fortran II, né en novembre 1954, Fortran III, IV et V vont suivre. Quel chemin...

Rappelé d'urgence à Paris, vous devez prendre demain le train Deauville-Paris.

Une première solution consiste à négocier, avec le chef de gare de Deauville, la place coin-fenêtre, dans le sens de la marche, du compartiment 3 de la voiture 14 du train de 8 h 47. La seconde consiste à faire confiance à la SNCF et simplement à réserver, par téléphone, une place en seconde classe dans le 8 h 47.

L'ancêtre est en pleine forme

La première méthode s'appelle : programmer en langage-machine. La seconde, qui laisse au système le soin de régler des détails sans grande importance, consiste à utiliser un "langage évolué". C'est à eux maintenant que nous allons nous intéresser, et d'abord au glorieux ancêtre, toujours fidèle au poste : Fortran.

LIST - PAGE 36

D'après René Moreau (dans *Ainsi naquit l'informatique* paru chez Dunod en 1982) il eut en vérité quelques précurseurs, dont par exemple le calcul d'expressions arithmétiques dû au mathématicien Rutishauser pour les besoins de l'analyse numérique (1952) ; il eut aussi quelques concurrents plus ou moins contemporains dont Mathmatic (de cette chère Grace Murray Hopper qui avait déjà inventé un assembleur) chez le grand rival d'IBM : le constructeur Univac. Mais il est incontestable que Fortran mérite le rang de premier, à cause de son immense succès commercial, de même qu'Apple restera dans l'histoire comme le créateur du micro-ordinateur en dépit des machines (françaises) plus anciennes Micral et Alcyane.

On comprend déjà sans doute que Fortran (FORmula TRANslator, traducteur de formules), avait essentiellement un but scientifique. La facilité qui consiste à écrire dans un programme une simple égalité comme : Z = 3 * COS (X – Y), cache un travail important de la part de la machine ; par exemple, il faut qu'elle sache d'abord calculer X – Y avant d'en prendre le cosinus. Une telle analyse semblait, au début des années cinquante, trop complexe pour pouvoir être exécutée en un temps convenable.

Pour l'époque, c'était un problème considérable car le coût immense du matériel rendait les programmeurs très sensibles à la vitesse du calcul. John Backus, déjà auteur de l'assembleur Speedcode d'IBM, sut persuader sa compagnie de créer, à la fin de 1953, un groupe chargé d'examiner si, comme lui-même le croyait, l'écriture d'un langage capable de faire ces analyses syntaxiques à une vitesse raisonnable était possible. Parfois, les historiens d'IBM ne citent même pas cette initiative qui devait pourtant jouer un rôle primordial dans la foudroyante ascension qui lui assure aujourd'hui le leadership incontesté en informatique. Le 10 novembre 1954 était publié le premier descriptif du langage, aussitôt suivi par l'écriture effective d'un compilateur et la mise au point de Fortran II.

Une descendance très nombreuse

Cette rapidité était due, en partie, aux qualités de l'IBM 704, notamment à ses registres pour index et surtout aux mémoires à tores de ferrites. Signalons en passant que cette machine vit aussi la première ébauche de ce qui deviendra plus tard l'*operating system* (système d'exploitation) qui soulagera tant le travail incroya-

blement ingrat des programmeurs de l'époque passant leur temps à courir en tous sens pour gérer "à la main" mille tâches à la fois.

On ne saurait faire la liste de toutes les innovations dues à Backus dès les premières versions de Fortran : la plus grande partie des langages modernes en est issue (à l'exception peut-être de Lisp et de quelques autres spécialisés en intelligence artificielle). Relevons simplement la notion de boucle (DO) et celle de GOTO (qui fera par la suite tellement couler d'encre...).

Le compilateur Fortran II devint commercial en avril 1957 dans la maison mère, mais Univac en sortira un en 1961 pour ses propres machines. Si Fortran III, né l'année suivante, resta dans les cartons, c'est surtout Fortran IV en 1962 qui assurera la première "vulgarisation" réussie de l'informatique.

Il finira ses jours le 3 avril 1978, lorsque l'American National Standard Institute adoptera la norme ANSI définissant l'actuel Fortran V, plus connu sous le sigle "77", qui corrige avec un certain bonheur la plupart des absences ou lourdeurs des premières versions. Modernisé, sûr de perdurer par sa bibliothèque de programmes immense (un peu comme l'Apple II dans le domaine des micros), ce langage historique est parti joyeusement pour une nouvelle carrière.

Puisque tout, ou presque, est né de lui, souvent on connaît déjà une bonne partie de Fortran, sans le savoir. En effet, le Basic dit "standard" en est, d'une certaine façon, un sous-ensemble simplifié. C'est donc par une comparaison père/fils que nous allons sommairement décrire les caractéristiques de base de la norme actuelle.

Essentielles : les étiquettes

Si les programmes sont également constitués de lignes, le numérotage en est très sporadique. En pratique, on constate que seules quelques marges de la liste comportent des étiquettes, formées d'entiers se succédant dans un ordre quelconque. Ces labels servent essentiellement aux fameux GOTO, aux boucles (où l'on ordonne d'effectuer l'instruction numérotée n), ainsi qu'aux formats de lecture et d'écriture que nous verrons plus loin. L'aspect extérieur des deux types de programmes est donc assez voisin, sans plus.

Les opérations mathématiques et les fonctions usuelles sont pratiquement les mêmes. La boucle FOR ... NEXT s'écrit DO n I a, b, s (c'est-à-dire exécuter l'ordre décrit sous l'étiquette n en faisant varier l'indice entier I de a à b, avec le pas s), suivi de CONTINUE pour passer de I à I+s.

Les modalités de tests sont également très voisines : vous ne serez pas surpris non plus par IF ... THEN ... ELSE ... END IF. A vrai dire, sans le savoir, vous essayez là le *Nouveau* Fortran. Le tout premier du nom ne connaissait qu'un IF "arithmétique" de la forme IF *(exp)* p,q,r avec trois étiquettes p, q, r à utiliser respectivement selon que l'expression sous examen était strictement positive, nulle ou négative. A cet IF ternaire s'était ajouté le IF "logique" dont voici un exemple :

IF (I.GT.J) n

(si I est plus grand que (greater than) J, faire n).

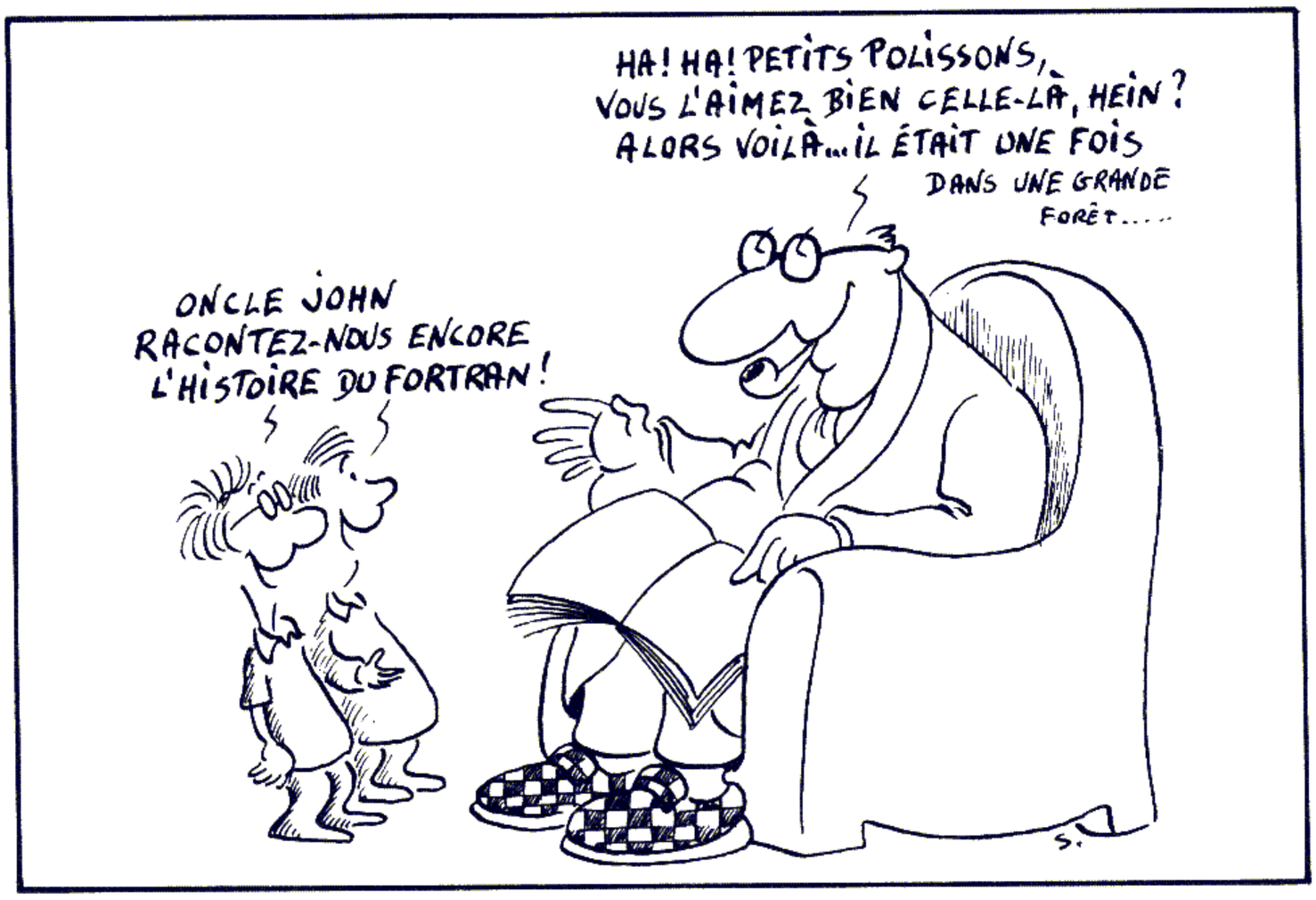
Signalons aussi un GOTO calculé : GOTO (p,q,r,s) *exp* est l'équivalent du Basic ON *exp* GOTO p,q,r,s : exécuter p si l'expression vaut 1, q si elle vaut 2 et ainsi de suite.

Peu doué pour traiter les textes

Jusqu'à présent donc, peu de différences entre les deux langages, surtout depuis le toilettage de 77. Il en est encore à peu près de même en ce qui concerne les tableaux (jusqu'à sept dimensions), voire de la gestion des fichiers, séquentiels ou à accès direct, maintenant que OPEN et CLOSE ont fait leur apparition... Le maître a copié l'élève.

Le traitement de chaînes de caractères est, comme on le sait, très différent d'un Basic à un autre. En Fortran, où il était très malaisé, pour ne pas dire inexistant, la norme ANSI a introduit des variables du genre A = 'BONJOUR' (en Basic A$ = "BONJOUR") et l'opérateur de concaténation //, comme dans 'CAR'//'NAVAL' ; mais il n'y a pas d'équivalent imposé des très commodes LEFT$, RIGHT$ ou STRING$ de Microsoft.

Peut-être croyez-vous, en fonction de ce qui précède, qu'un Basicophone peut donc entrer avec facilité dans un programme Fortran puisque tant de choses les rapprochent, surtout depuis 77. C'est malheureusement très approximatif, car il reste deux singularités profondes, à vrai dire causes de difficultés importantes pour les débutants, qui font que ce langage restera toujours d'un abord aride : les déclarations des variables et surtout celles des formats.



PETITE HISTOIRE DES LANGAGES

DE FORTRAN II A FORTRAN V

Les premières sont banales pour des Pascalophiles. Le Fortran utilise plusieurs types qu'ils reconnaîtront : les variables entières, réelles en simple précision, en double précision, les chaînes de caractères et les booléens (vrai ou faux logique). Particularité due à son origine mathématique, il possède aussi un type "complexe" avec ses opérations :

$(a+ib)*(c+id) = (ac-bd) + i*(ad+bc)$.

Comme en Pascal donc, il faut définir les types avec grande précision, car Fortran est très maniaque. Si vous lui demandez de calculer A/B, où A = 12 et B = 13, il verra que A et B sont entiers, et répondra donc par un entier. Comme 12/13 est compris entre 0,923 et 0,924 il répondra que A/B est égal à 0, ce qui pourrait avoir quelques conséquences fâcheuses... (Pour éviter cette catastrophe, calculez 12.0/13.0, ce qui fera tout rentrer dans l'ordre).

Au début de chaque programme, on déclarera donc clairement qui fait quoi, avec une exception heureuse : toutes les variables dont le nom commence par I, J, K, L, M ou N sont implicitement (sauf déclaration contraire) de type entier. Par exemple, si BELL est un indicateur renvoyant 1 si l'ordinateur peut émettre des sons et 0 en cas contraire, il sera souvent noté IBELL en Fortran, ce qui le rend entier. Plus généralement, on peut décréter par exemple (avec l'aide de l'instruction IMPLICIT) que tous les mots commençant par Z seront du type logique, etc.

Il reste à dire quelques mots du cauchemar qu'est le concept de format. En exergue du chapitre qu'il lui consacre dans son excellent cours de Fortran 77 (Éditions Masson), Patrice Lignelet cite avec humour Desmarets de Saint-Sorlin :

"Prophane, esloigne toy, j'entre dans ma fureur !"

Ces fameuses cartes perforées...

Calmons d'abord le jeu en dévoilant que, depuis 77, il existe heureusement un équivalent de INPUT,A sous la forme READ*,A (ainsi qu'un bien commode PRINT*,A). Mais avant ces formats libres, les entrées/sorties exigeaient de définir très précisément comment manipuler les données échangées. Toute une symbolique indique le type concerné — I pour entier, F pour réel... —, le nombre de chiffres avant et après la virgule ; la lettre X désigne le nombre de blancs à insérer, etc.

L'origine de cette rigueur typographique ne laissant rien au hasard est évidemment historique : on la trouve sur ces fameuses cartes perforées qui furent longtemps les seuls périphériques informatiques (même le banal écran de visualisation, apparu en 1954, mit de longues années à se généraliser).

Tous les anciens connaissent les célèbres 80 colonnes : de 1 à 5 pour l'étiquette, 6 pour un code d'impression, de 7 à 72 pour l'utilisation normale proprement dite, de 73 à 80 sans signification (jadis on y mettait un numéro d'ordre de la carte dans le paquet qui s'étalait trop souvent à terre, conformément aux lois de Murphy !). Par exemple, l'instruction READ(5,18)A,B,C, suivie de FORMAT(41X,F6.4,2X,I3,7X,E3.2) sur une ligne précédée de l'étiquette 18, doit être décryptée de la manière suivante :

- le 5 (de READ) indique que quelque chose sera lu sur le périphérique numéro 5 (c'était le lecteur de cartes sur les IBM 360) ;
- cette lecture sera faite selon le format explicité dans la ligne de label 18 ;
- dans le FORMAT, 41X indique que, au départ, 41 colonnes seront sautées (on passera donc directement à la colonne 42) ;
- avec F6.4, le réel lu sera mis en mémoire sur six chiffres dont quatre après la virgule : s'il s'agit de pi = 3,1415936535..., l'ordinateur retiendra donc 03,1415 ; il écrira ce premier nombre dans la mémoire associée à la première variable du READ, A ;
- 2X pour passer deux cases ;
- I3 pour lire un entier à trois chiffres à placer en B ;
- 7X fait passer sept colonnes ;
- E3.2 pour lire un réel à mettre dans C en écriture "scientifique" (E = exposant) avec trois chiffres dont deux après la virgule.

Une telle précision est évidemment fort utile, par exemple, pour l'édition d'états comptables impeccables, mais son usage systématique était loin d'être partout justifié. Il a découragé bien des programmeurs. Peut-être d'autres prendront-ils le relais.

André WARUSFEL

Un exemple en Fortran

VOICI un programme qui donne le maximum MA et le minimum MI d'une suite de 10 nombres entiers positifs inférieurs à 999999, écrits sur des cartes perforées placées dans le lecteur de la machine. Le premier de ces nombres est évidemment à la fois le maximum et le minimum initiaux. Par le jeu de neuf fois deux comparaisons successives, on édite le maximum et le minimum absolus sur l'imprimante (sortie n° 6).

```
      J=1
      READ(5,14)MA
      MI=MA
14    FORMAT(10X,I6)
20    J=J+1
      READ(5,14)K
      IF(K.GT.MA)MA=K
      IF(K.LT.MI)MI=K
      IF(J.LT.10)GOTO20
      PRINT(6,15)MA
15    FORMAT('LE MAXIMUM EST: ',I6)
      PRINT (6,17)MI
17    FORMAT('LE MINIMUM EST: ',I6)
      STOP
      END
```

LIST - PAGE 38

LA SIMULATION DE "CHAIN"

DANS le premier numéro de LIST, nous avons exposé une méthode permettant de chaîner des programmes. En voici une nouvelle. Elle tente de combler une lacune du Basic du Commodore en simulant l'instruction de chaînage CHAIN qui fait si cruellement défaut à cet ordinateur.

Lorsque vous sollicitez de votre machine l'exécution immédiate d'un ordre, vous commencez certainement par taper au clavier la ligne de commande. Alors, vous appuyez sur la touche RETURN. Par exemple : LOAD "titre", 8 suivi de RETURN.

Quand cette validation est effectuée, le système consulte la partie mémoire contenant la ligne (en l'occurrence, il s'agit de la mémoire écran), et tente d'exécuter l'ordre qu'elle est censée contenir. Si rien ne s'y oppose, l'exécution se produit aussitôt. Dans le cas contraire, c'est un message d'erreur qui apparaît.

Forts de cette constatation, nous pourrions imaginer d'inscrire par programme sur l'écran la ligne de commande, et de la valider aussitôt, si possible sans intervention manuelle. L'écriture d'une ligne de commande en mémoire d'écran est très facile à obtenir. On tapera, par exemple :

```
100 PRINT"LOAD";CHR$(34);
"titre";CHR$(34);",8"
```

Mais comment valider la ligne ainsi affichée ? En fait, le problème peut être résolu aisément. Lors de l'appui sur une touche, trois phénomènes se produisent :

- le code de la touche pressée apparaît dans le tampon-clavier ;
- un compteur spécialisé est incrémenté, indiquant le nombre de caractères présents dans le tampon-clavier ;
- le symbole correspondant à la touche pressée s'affiche à l'écran.

Le *tableau des adresses* indique, pour chaque machine utilisée, les adresses de mémoire concernées par les deux premiers phénomènes.

Dix octets seulement sont disponibles dans le tampon-clavier ; nous devrons donc nous en contenter. Mais notons toutefois que le remplissage du tampon-clavier et la mise à jour du compteur associé pourront s'effectuer aussi bien par l'appui direct sur les touches que par des POKEs judicieusement placés dans un programme. On peut l'observer en tapant, en mode direct, la ligne POKE 631,147: POKE 632,13:POKE 198,2 et en la validant par un appui sur RETURN.

Programmons RETURN

Pourquoi cela ? Nous avons placé, à la première adresse du tampon, le code de l'effacement d'écran (CHR$(147)). Nous avons placé à l'adresse suivante le code de RETURN (CHR$(13)) ; et enfin, nous avons mis à jour le compteur, en indiquant à l'adresse concernée la présence de deux caractères. L'appui sur RETURN a rempli le tampon-clavier, puis a provoqué sa remise à zéro et l'affichage des caractères qui s'y trouvaient. L'écran s'est donc effacé.

Tableau des adresses

	CBM 3/4/8000	VIC 20	C.64
Tampon-clavier	623 à 632	631 à 640	631 à 640
Pointeur	158	198	198

Pourquoi, dès lors, ne pas imaginer de positionner de cette façon le curseur sur la ligne à valider (que nous aurions auparavant inscrite à l'écran), puis d'envoyer tout simplement le code du RETURN dans le tampon-clavier ?

Composons le programme ci-dessous (attention, les adresses sont celles des VIC et C.64) :

```
10 PRINT "FOR Q=0 TO 20:PRINT Q;:NEXT"
20 POKE 631,19          :REM CODE DE 'HOME'
30 POKE 632,13          :REM CODE DE 'RETURN'
40 POKE 198,2           :REM 2 CARACTERES DANS LE TAMPON
```

Et lançons-le par le RUN habituel...

Parfait, ça marche : nous venons de faire exécuter un programme par un autre programme...

Disquette ou cassette ?

Ayant maintenant en mains tous les éléments de la réussite, résumons par un exemple efficace les secrets du problème, en essayant le court programme ci-dessous :

```
10 REM programme appelant
20 REM suite du programme
30 REM fin du programme
40 PRINT CHR$(147);"LOAD";
   CHR$(34);"PROG2";CHR$(34)
   ;",8"
50 PRINT:PRINT:PRINT:PRINT
60 PRINT"RUN"
70 POKE 631,19:POKE 632,13:
   POKE 633,13
80 POKE 634,13:POKE 635,13:
   POKE 636,13
90 POKE 198,6
99 END
```

Attention : si vous utilisez un magnétophone, le ",8" de la ligne 40 doit être remplacé par ",1". Sauvegardons ce premier programme sur disquette ou cassette, selon le cas, puis après un NEW, composons le programme qui sera appelé par le précédent :

```
10 REM programme appelé
20 FOR Q=0 TO 20:PRINT Q;:
   NEXT
30 END
```

Sauvegardons ce second programme, sous le titre PROG2. Cela étant fait, rappelons en mémoire le premier programme, et lançons-le par le RUN traditionnel. Comme prévu, les messages des lignes 40 et 60 s'affichent en haut de l'écran, tandis que la disquette (ou la cassette) se remet à fonctionner, chargeant — comme nous l'espérions — le programme suivant. Le RUN est ensuite validé sans intervention de notre part. Notez que nous avons pris la précaution de le placer à l'endroit exact de la réapparition du curseur après le LOAD. Un simple LIST nous confirmera la présence en mémoire du second programme, et la disparition totale du précédent.

Cette seconde méthode présente l'avantage d'éviter une manipulation délicate des pointeurs de mémoire. Elle permet aussi de charger successivement, et sans problème, des programmes de longueur quelconque. Sa mise en œuvre est donc très simple. Malheureusement, elle interdit totalement le passage de variables d'un programme à l'autre. Sauf si le programme appelé est un programme en langage-machine qui ne détruira pas les pointeurs de mémoire du programme appelant.

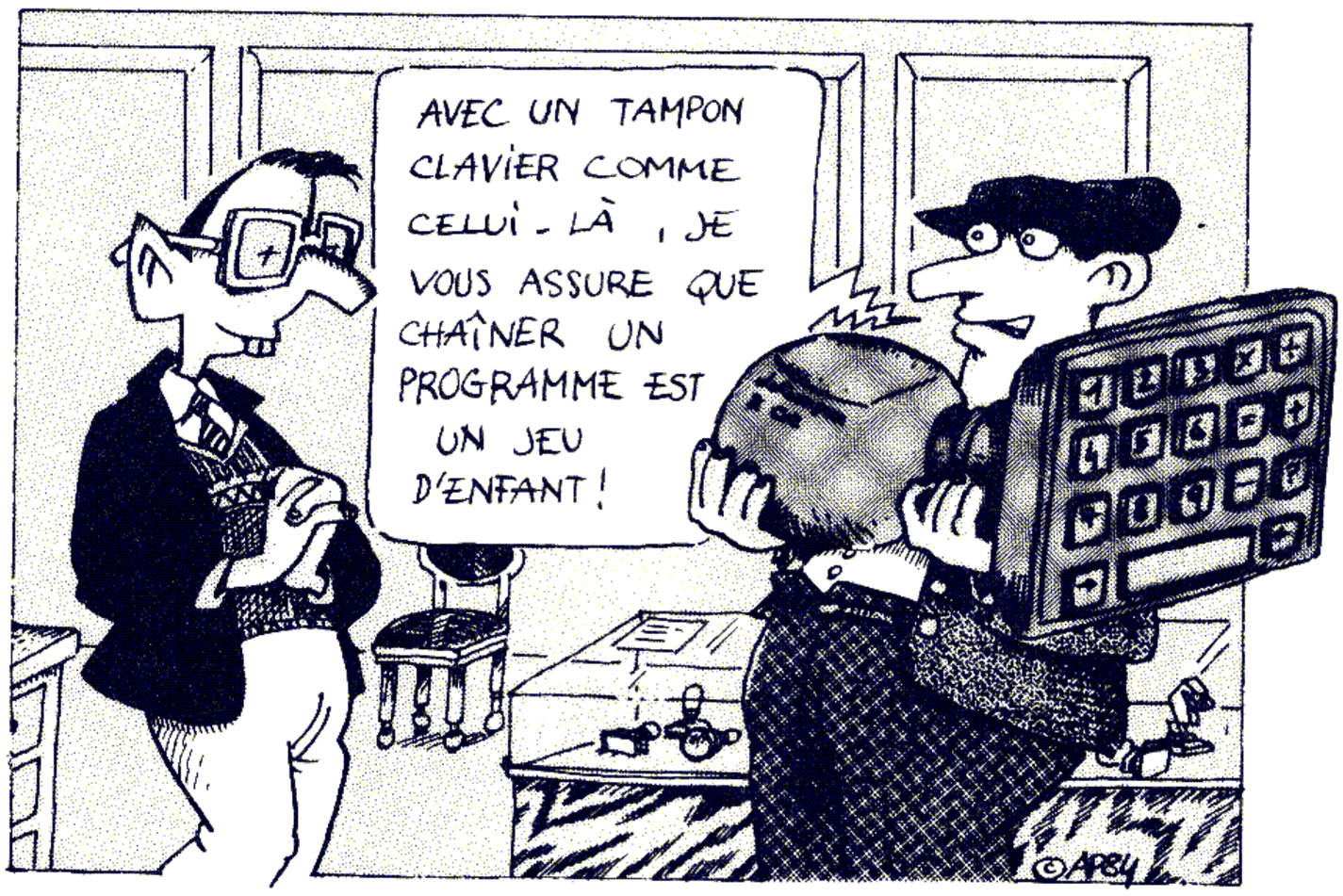
Sur un écran couleurs, si vous prenez la précaution de faire des affichages de la même couleur que le fond, ceux-ci deviendront invisibles, mais l'exécution continuera tout à fait normalement. Vous rendrez ainsi plus discret le processus de chaînage de vos programmes.

S'il s'agit enfin de chaîner un programme en langage-machine, vous pouvez utiliser un *programme appelant* ayant cette forme simplifiée :

```
10 REM (début du programme appe-
   lant)
20 C=C+1
30 IF C=1 THEN LOAD
   "prog1",8,1
40 REM (suite du programme)
50...
```

Il est important de ne pas omettre le ',1' de la ligne 30, faute de quoi "prog1" viendrait recouvrir et détruire le programme appelant, avec les inconvénients que vous devinez !

Le fonctionnement est alors le suivant : lors du premier lancement du programme, C prend la valeur 1. La ligne 30 est donc exécutée, et le programme en langage-machine "prog1" est mis en mémoire. Ceci étant fait, le programme redémarre automatiquement en ligne 10 sans qu'il soit nécessaire de taper un second RUN. Quant à "prog1", implanté suffisamment haut en mémoire, il n'a pas détruit les variables et C prend maintenant la valeur 2. La ligne 30 n'est donc plus exécutée, et le programme appelant se poursuit normalement.



Munis de ces éléments essentiels, vous pourrez "dynamiser" vos programmes, et ne plus trop souffrir du manque de mémoire de votre machine. A vous de jouer maintenant !

Jean-Pierre LALEVÉE

LE BASIC DU HP-71B

DU CONCENTRÉ D'ORDINATEUR

C'EST une calcultrice scientifique élaborée, un ordinateur de poche programmable en Basic, un petit joyau de Hewlett Packard qui a réussi à rassembler en un volume vraiment réduit une belle panoplie de possibilités. Mais cela se paie : 5 100 FF ttc pour la version de base.

Non, le mode CALC dans lequel sont réalisés tous les calculs n'utilise pas la notation polonaise inverse (très) chère au cœur des inconditionnels d'HP, mais une classique notation AOS : « *Je calcule comme c'est écrit, 1 + 1 vaut 2.*» Rentrée dans le rang, donc, pour le HP-71B ; la machine n'en sera que plus accessible car, il faut bien le dire, la notation algébrique semble plus ''naturelle'' à bien des utilisateurs.

Mais tant qu'à choisir cette notation, autant l'améliorer. On donne ici un bon point à HP qui semble avoir soigné tout particulièrement le système de gestion des erreurs : il est toujours possible de récupérer d'un coup l'expression qui vient d'être calculée et, éventuellement en cas d'erreur, de la corriger.

Mieux, un long calcul peut être exécuté pas à pas avec la touche SST : 1 + 3*SIN (2/3, donnera successivement les calculs de 2/3 puis (refermer la parenthèse) SIN (2/3), 3*SIN (2/3) et, enfin, le calcul général. Les résultats sont affichés sur 12 chiffres significatifs mais les calculs sont effectués avec 15 chiffres. L'exposant varie de −499 à 499 : l'abondance ne nuit pas. L'opérateur contrôle trois types d'arrondi : OPTION ROUND ZERO (vers 0), POS (à la valeur supérieure) et NEG (à la valeur inférieure). Quatre formats de sortie sont disponibles : FIX (fixe), SCI (scientifique), ENG (ingénieur) et STD (virgule flottante).

Une constatation s'impose donc : le HP-71B vise essentiellement des usages scientifiques et sa force réside dans sa puissance de calcul.

Une pile de cinq niveaux d'instructions est employée pour conserver les cinq derniers calculs effectués (on peut l'étendre jusqu'à 16 niveaux). C'est alors un jeu d'enfant que de retrouver d'anciennes expressions afin de les exécuter, éventuellement en les modifiant.

Des fonctions très spéciales

Si l'on vous dit CEIL (X), à quoi pensez-vous ? Certainement pas immédiatement au calcul du plus petit nombre entier supérieur ou égal à X... Deux fonctions ''partie entière'', INT et IP, retournent respectivement, la partie entière (mathématique) et la partie du nombre à gauche d'une virgule.

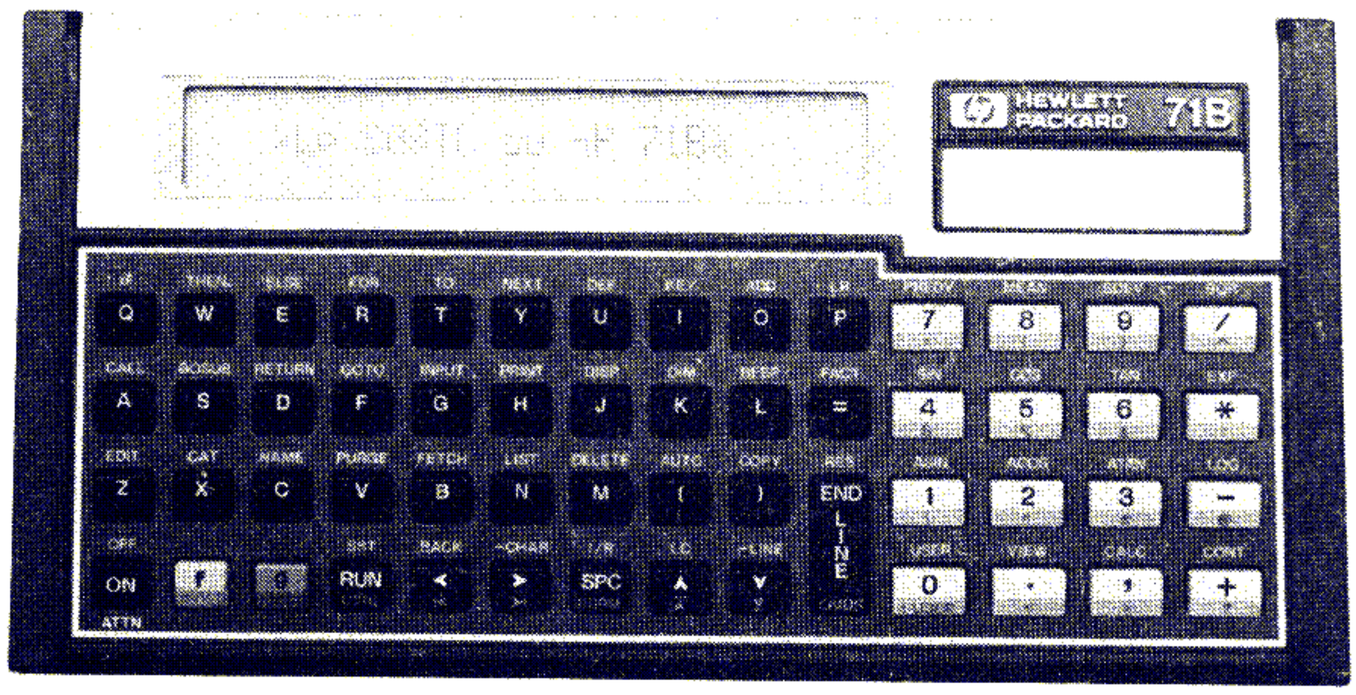
Savez-vous traiter une erreur mathématique à la mode du HP-71B ? Bien pratique cependant est la possibilité (déclarée avec l'instruction DEFAULT) de donner à une division par zéro... la valeur par défaut de 9,999...9 E 499 ou ''l'infini'' selon HP (Inf). De même, inutile d'hésiter, LOG(0) donne −Inf (moins l'infini).

La panoplie des fonctions statistiques est complète avec les opérations moyenne, écart-type, régression linéaire, corrélation, valeur probable... Et tout cela sur un tableau dont la taille peut comporter jusqu'à 15 variables ! Le statisticien trouve là l'instrument de poche sans doute le plus puissant à ce jour.

LIST A TESTÉ

LE BASIC DU HP-71 B

Des petites dimensions pour de grandes applications



C'est l'ordre EDIT qui permet d'accéder à l'éditeur de programmes Basic. La taille de la mémoire vive disponible est alors d'environ 16,5 Koctets. Chaque programme, car plusieurs peuvent cohabiter pacifiquement dans la mémoire, est défini par une ligne d'état comportant son nom, son type (on les verra tous), sa taille et sa date. Un dernier paramètre précise les conditions d'accès (protégé ou non,...) d'un utilisateur au programme.

Un Basic puissant

Les types des fichiers (on dira fichier de préférence au trop limitatif mot de programme) sont de différents ordres. BASIC pour les programmes en ... Basic, BIN et LEX pour le langage-machine (par exemple le module du langage Forth optionnel), DATA pour les données, TEXT (sans commentaire) et SDATA pour des fichiers à échanger entre HP-71B et HP-41C (via HP-IL).

Toutes les touches du clavier sont redéfinissables, et ces jeux de touches sont mémorisés dans des fichiers KEYS. Un mode spécial (USER) active les touches redéfinies (avec DEF KEY).

Dans le Basic lui-même, on retrouve toutes les fonctions classiques telles IF...THEN...ELSE, l'adressage symbolique (donner des noms à des lignes de programmes), ON...ERROR avec ERRL (n° de ligne erroné), ERRM$ (message d'erreur) et ERRN (n° de l'erreur). Dans le traitement des erreurs, donc, le HP-71B est bavard. Très bavard même car il ne présente pas moins de 97 messages d'erreur différents.

Fiche technique du HP-71B

Prix : environ 5 100 FF ttc
Dimensions : 190 × 97 × 12 mm
Poids : 340 g
Mémoire vive : 16 955 octets pour l'utilisateur (version de base) extensible par modules de 4 Ko (max. 33,5 Ko pour 4 modules). Mémoire continue répartie entre fichiers et variables au fur et à mesure de leur utilisation
Mémoire morte : 64 Ko
Alimentation : 4 piles bâtons alcalines micro de 1,5 V
Langage : Basic d'origine, Forth et Assembleur par modules enfichables
Affichage : fenêtre de 22 caractères sur une ligne virtuelle de 96 caractères
Pile de commande : 5 niveaux
Périphérique : interface HP-IL, lecteur de cartes magnétiques, interface RS 232 C, lecteur de cassette, imprimante, interface-vidéo, GPIO, HP-IB, série 80

Citons au passage les fonctions AND, OR, XOR et NOT d'algèbre booléenne, AUTO pour la numérotation automatique des lignes, RENUMBER qui renumérote un programme, KEYDOWN pour la saisie d'une pression de touche, PUT et KEY$, enfin, pour *envoyer* et saisir des caractères dans la mémoire tampon du clavier.

Les variables : une déception

Les mémoires du HP-71B, dites "variables", sont à la fois un point faible et fort. On regrettera la limitation des tableaux de chiffres à deux dimensions. Pire : une seule dimension pour les tableaux de caractères...

Heureusement, on dispose de variables locales, indépendantes pour chaque sous-programme qui, avec CALL, SUB et ENDSUB permettent de sauvegarder des contenus de variables avant d'exécuter des sous-programmes puis de les récupérer au retour.

Les noms de variables sont limités à une seule lettre ou une lettre suivie d'un chiffre. Calculez vous-même le nombre réduit de ces noms. Pour y remédier, on devra user à profusion des variables locales des sous-programmes. Cela rendra peut-être plus structurée la rédaction des programmes ; c'est bien, mais y être contraint est tout de même frustrant.

Ceci étant, le peu dont on dispose est très correctement géré. OPTION BASE définit pour des tableaux la valeur de départ de l'indice (0 ou 1) ; l'emploi d'une variable de tableau (par exemple : C(1,1) = 12) crée *automatiquement* un tableau de dimensions 10 × 10 et l'ordre DESTROY permet d'éliminer *sélectivement* certaines variables. Un tableau peut même être redimensionné ultérieurement sans perte de ses contenus.

Bien des caractères à l'affiche

Avec l'afficheur à cristaux liquides de 22 caractères, on retrouve l'efficacité. Chacune des 132 colonnes de 8 points de l'afficheur peut être gérée séparément et l'utilisateur peut se créer un jeu de caractères particuliers (codes 128 à 255). Enfin, la variable spéciale DISP$ contient en perma-

LES TESTS DE LIST

Ces tests ont pour vocation de tester la rapidité de travail du HP-71B. Bien sûr, ils sont forcément restrictifs, et la rapidité n'est pas toujours le critère premier de l'efficacité. C'est dans *LIST* n° 1 que le lecteur trouvera la justification du choix de ces tests.

Tous les tests portent sur une boucle Basic qui effectue 10 000 fois la même opération. Les temps donnés sont exprimés en secondes.

Test	
1 - Boucle vide	95
2 - Sous-programme	206
3 - Calcul matriciel	97
4 - Chaînes de caractères	266
5 - Arithmétique	262
6 - Calcul scientifique	823 (15 chiffres)
7 - Affichage	1 146
8 - Écriture fichier	420 (mémoire)
9 - Lecture fichier	200 (mémoire)

GOTO 500 exécute la ligne 500 cinq secondes plus tard.

Une batterie de 128 flags (drapeaux ou indicateurs binaires) gère l'état du système. Les trois quarts sont accessibles à l'utilisateur qui en use selon ses besoins, par exemple : le flag −1 supprime les messages d'erreur. Les flags 0 à 63 sont sans signification pour le système.

Malgré quelques restrictions tout à fait surprenantes, le Basic du HP-71 est l'un des plus puissants disponibles sur une machine de cette taille. Mais plus encore que le Basic, c'est la puissance de calcul qui se révèle d'un attrait exceptionnel, tant pour les mathématiciens que pour les statisticiens... fortunés.

Olivier ARBEY
Jean-Christophe KRUST

nence les 22 caractères présents sur l'afficheur.

Le jeu de caractères est très complet : outre les majuscules et les minuscules, on trouve des caractères accentués et des lettres grecques.

Enfin ! Un sommet est atteint dans la gestion des cristaux liquides. Avant, on se tordait le cou pour bien les lire, puis une petite molette est venue apporter une possibilité de réglage de l'intensité (contraste) de l'afficheur. Avec le HP-71B ce réglage est réalisé par voie logicielle avec l'instruction CONTRAST, et c'est très réussi.

Classiques, mais fondamentales pour les esprits curieux, sont les fonctions PEEK$ et POKE ; ce qui l'est moins c'est l'obligation de travailler sur des demi-octets inversés ! (4841 est en fait codé 4148).

La gestion des fichiers DATA, TEXT et SDATA réalisée en mémoire est assez performante. Leur taille s'ajuste automatiquement au fur et à mesure que l'on introduit les informations, données ou textes, qu'on peut évidemment relire ensuite à volonté.

Une horloge interne tient à jour deux variables spéciales TIME (heure) et DATE dont l'utilité est évidente. De plus, à la manière d'une HP-41C avec module TIME, une interruption temporelle peut être programmée. Ainsi, tel jour à telle heure, un programme peut s'exécuter automatiquement (y compris si le HP-71B est éteint). D'autres interruptions, tenant du compte à rebours, sont programmables en Basic : ON TIMER # 1,5

L'ALGO DU LOG

IMAGINEZ que la fonction logarithme de votre calculatrice ou de votre ordinateur ne marche plus. Comment feriez-vous pour l'obtenir tout de même en n'utilisant que les quatre opérations arithmétiques ? Cette question s'est d'ailleurs posée aux concepteurs de langages puisque les processeurs ne connaissent que les opérations de base. Aussi certains des algorithmes présentés ici sont utilisés à l'intérieur de vos ordinateurs ou de vos calculatrices.

Pendant des siècles, l'astronomie, la navigation, la physique... ont fait appel à des calculs très longs et très difficiles.

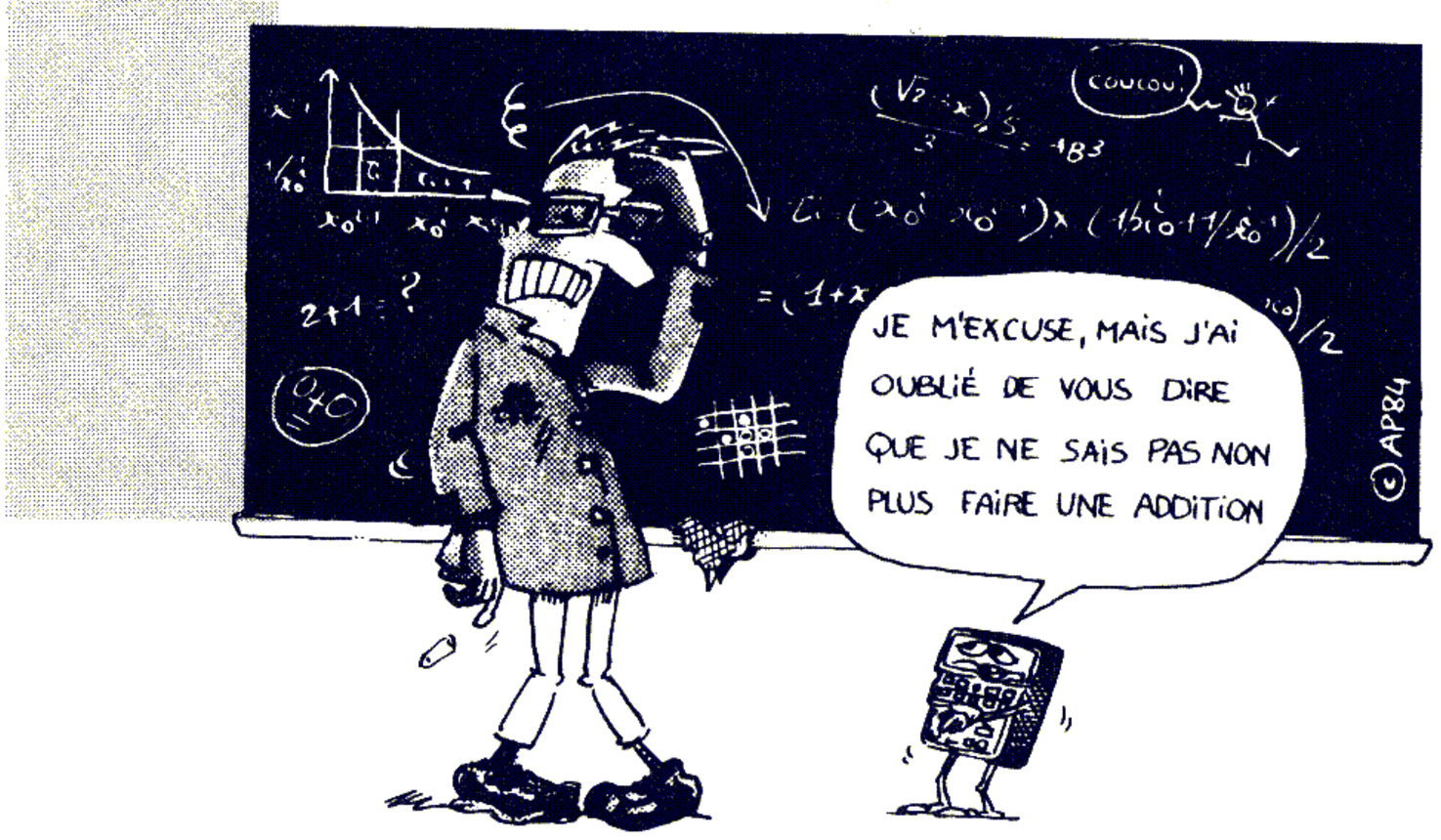
Utilisant ses heures de loisir à la recherche de méthodes nouvelles de calcul numérique, John Napier, baron de Merchiston (1550-1617), fit paraître en 1614 une *Description de l'admirable table de logarithmes* (Mirifici Logarithmorum Canonis Descriptio). Cet ouvrage va permettre de simplifier « admirablement » les calculs pénibles : *grâce aux logarithmes, les multiplications ou les divisions deviennent des additions ou des soustractions.*

Cette découverte est basée sur l'observation de points mobiles, les uns évoluant comme la suite algébrique 0, 1, 2, 3, ... et les autres comme la suite géométrique 1, 2, 4, 8, ... Napier a alors cherché un lien entre ces deux suites. Il l'a trouvé en constatant que chaque terme de la seconde suite était égal à 2 élevé à la puissance du terme de même rang de la première suite : $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, ...

Si on appelle 2 le « nombre de base », on peut dire ce qui précède autrement : chaque terme de la suite algébrique est le logarithme en base 2 du terme de même rang de la suite géométrique. Soit, 0 est le logarithme de 2 en base deux, 1 est le logarithme de 4 en base deux, 3 est le logarithme de 8 en base deux,...

En généralisant, dire que a est le logarithme de c en base b revient à dire que b à la puissance a est égal à c. Ce qui s'écrit mathématiquement : $a = \log_b c$ est équivalent à $b^a = c$.

Les propriétés du logarithme qui permettent de passer de la multiplication à l'addition ou de la division à la soustraction sont :
$\log_b (a \times c) = \log_b a + \log_b c$ et $\log_b (a/c) = \log_b a - \log_b c$.



Pour les calculs, deux bases de logarithme ont été retenues : la base dix (logarithme décimal) et la base e (logarithme népérien ou naturel). Ce nombre e (environ 2,7182818) possède des propriétés valables dans différents domaines mathématiques (en particulier le calcul infinitésimal).

Le logarithme népérien (ou naturel) permet de calculer tous les autres logarithmes. En effet, le logarithme en base b d'un nombre x est lié au logarithme népérien par la formule : $\log_b x = \ln x / \ln b$ (ln représente le logarithme népérien).

Le terme de « naturel » appliqué au logarithme népérien se comprend mieux lorsque l'on étudie la fonction logarithme népérien et sa fonction inverse, la fonction exponentielle. Cette dernière est la seule fonction d'un nombre qui ait un degré de variation par rapport à ce nombre égal à la fonction elle-même (en d'autres termes, c'est la seule fonction égale à sa dérivée).

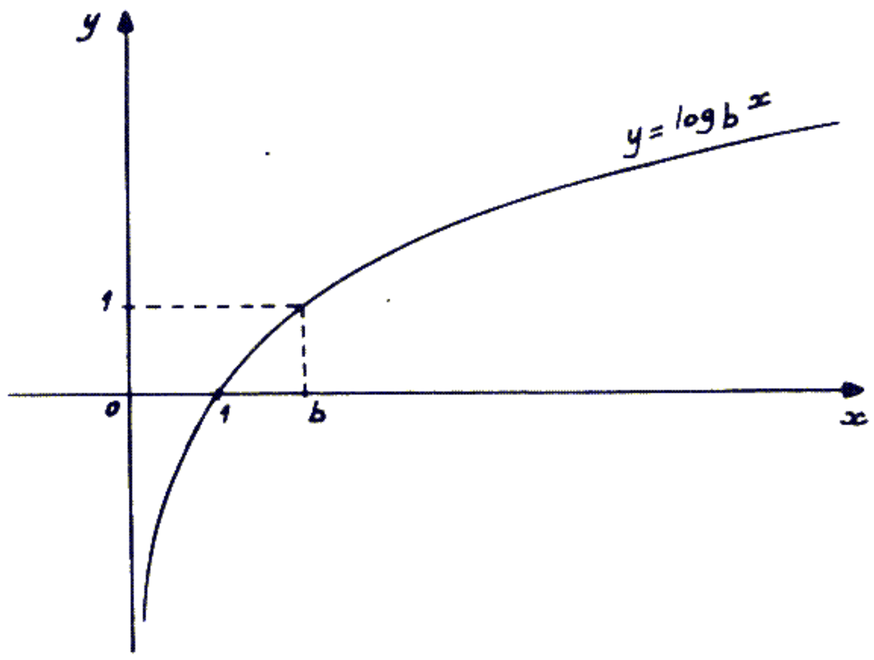
D'après la définition...

La première méthode que nous abordons peut être facilement comprise à partir de la définition : $y = \log_b x \Leftrightarrow x = b^y$ ($\Leftrightarrow$signifie « est équivalent à »). Cette méthode utilise l'écriture décimale de y, $y = y_0, y_1 y_2 y_3 \ldots$ en supposant ici que y est positif et donc que x est supérieur à 1. Nous avons : $x = b^{y_0 + y_1/10 + y_2/100 + y_3/1000 + \cdots}$ où y_0, y_1, y_2, y_3,... sont des chiffres variant de 0 à 9.

LIST - PAGE 44

Cela peut se récrire ainsi : $x = b^{y_0} \times b^{(1/10)(y_1+y_2/10+y_3/100+\cdots)}$.

Le second terme étant nécessairement inférieur à b, nous pouvons calculer y_0



Courbe représentative de la fonction logarithme

qui sera le plus petit nombre pour lequel $x_0 = x/b^{y_0}$ deviendra inférieur à b.

Une fois y_0 trouvé, en posant $x_0^{10} = (x/b^{y_0})^{10} = b^{y_1+y_2/10+y_3/100+\cdots}$, on s'aperçoit que y_1 se trouvera de la même manière que y_0. Puis y_2, y_3, etc.

Cela donne le court programme Basic suivant :

```
10 INPUT « Nombre et Base » ; X, B
20 Y = 0
30 IF X >=B THEN X=X/B:Y=Y+
   1 : GOTO 30
40 PRINT Y ;
50 X=X*X:Y=X:X=X*X:X=X*X:
   X=X*Y
60 GOTO 20
```

La ligne 50 est le calcul de x à la puissance 10 avec seulement quatre multiplications. La ligne 40 imprimera un par un tous les chiffres de $\log_b x$.

La précision dépend de la machine

Ce programme qui boucle indéfiniment peut donc vous donner une infinité de chiffres. Attention : les chiffres réellement significatifs sont en rapport avec la précision de votre machine. On peut estimer par exemple que si votre Basic calcule sur 8 chiffres, seuls les 6 premiers donnés par le programme sont bons. Quant aux suivants, ils peuvent être une excellente source de nombres aléatoires...

La méthode des premières tables

Comme nous l'avons déjà dit, cette méthode n'est valable que pour $x \geqslant 1$. On peut toutefois l'utiliser pour tout x compris entre 0 et 1 en sachant que : $\log_b x = -\log_b(1/x)$. Il vous suffit donc de rajouter la ligne :

```
15 IF X<1 THEN X=1/X : PRINT
« - » ;
```

Et pour être complet, il vaut mieux rajouter :

```
12 IF X<=0 THEN PRINT "Impossi-
ble" : END
```

Lorsque, en 1617, Henry Briggs (1556-1631) publia des tables de logarithmes à 14 décimales, il n'avait à sa disposition ni calculatrice ni, a fortiori, ordinateur. Ce travail colossal a été rendu possible grâce à une méthode basée sur la formule : $\log_b\sqrt{x_1x_2} = (\log_b x_1 + \log_b x_2)/2$.

Outre les quatre opérations, nous supposons donc que nous avons à notre disposition la racine carrée.

La méthode de Briggs est valable pour $1 \leqslant x \leqslant b$. Il démarre avec $x_1 = 1$ et $x_2 = b$. Ce qu'on peut encore écrire : $\log_b x_1 = 0$ et $\log_b x_2 = 1$. Puis il cherche à « serrer » le plus finement possible x entre x_1 et x_2.

Ainsi si $\sqrt{x_1x_2}$ est inférieur à x, alors x_1 croît en devenant $\sqrt{x_1x_2}$ (sinon x_2 décroît en devenant $\sqrt{x_1x_2}$). Bien sûr, $\log_b x_1$ (ou $\log_b x_2$) est remis à jour par la formule rappelée plus haut.

Lorsque x_1 et x_2 deviennent assez proches, on peut considérer que $\log_b x \approx \log_b \sqrt{x_1x_2}$. D'où le programme :

```
10 INPUT « Nombre et Base » ; X, B
20 X1=1 : L1=0 : X2=B : L2=1
30 X3=SQR(X1*X2):L3=(L1+L2)/2
40 IF X>X3 THEN X1=X3 : L1=L3
   ELSE X2=X3 : L2=L3
50 IF X2-X1>1E-8 THEN 30
60 PRINT (L1+L2)/2
```

Vous remarquerez que le 1E-8 de la ligne 50 n'est autre que la précision demandée. Cette précision doit être moins bonne que celle de la machine.

Pour que le programme puisse aussi marcher pour x>b, on utilise le fait que $\log_b b^k x = k + \log_b x$. Il suffit donc de rajouter ces deux lignes :

```
17 K=0
18 IF X>B THEN X=X/B :
   K=K+1 : GOTO 18
```

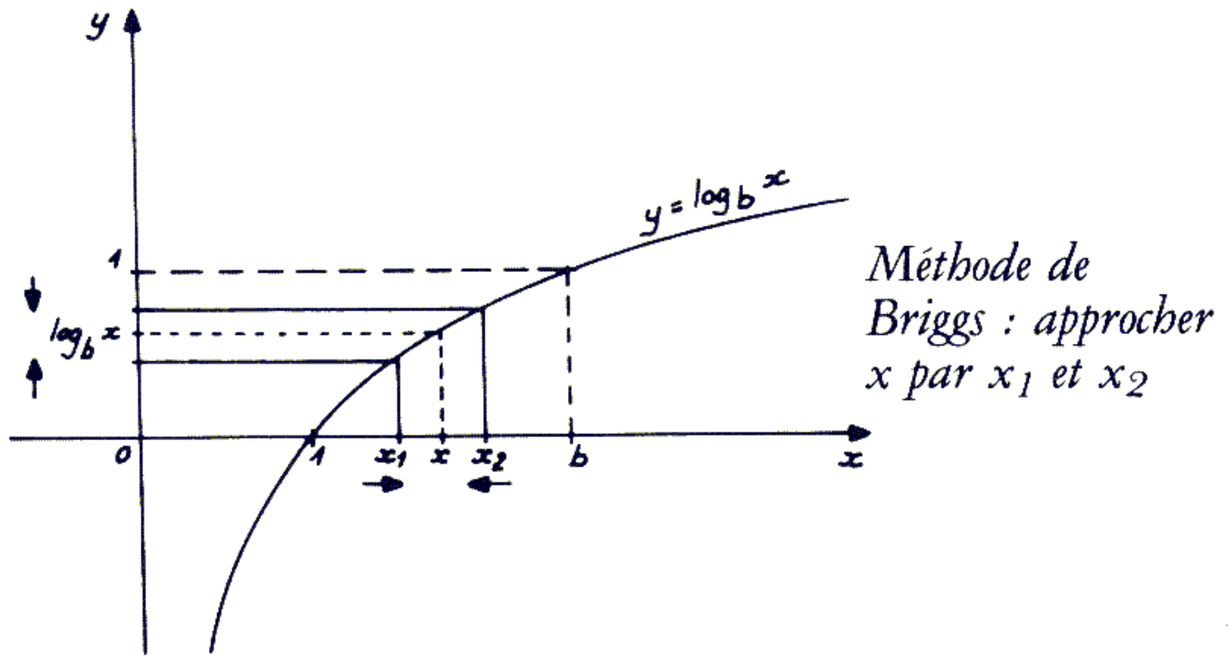
et de modifier la ligne 60 ainsi :

```
60 PRINT K+(L1+L2)/2
```

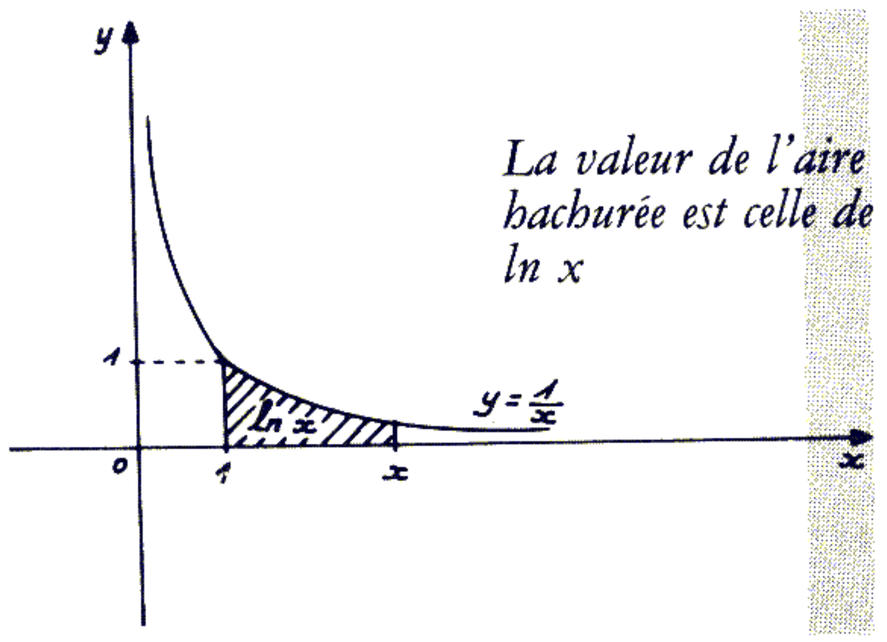
Et pour que le programme fonctionne aussi pour les x inférieurs à 1, les lignes 12 et 15 de la première méthode peuvent être utilisées.

En se servant d'une hyperbole

Vous savez peut-être que $\ln x = \int_1^x (1/t)dt$. Cela signifie en clair que la surface hachurée dans le dessin ci-dessous vaut exactement ln x.



Méthode de Briggs : approcher x par x_1 et x_2



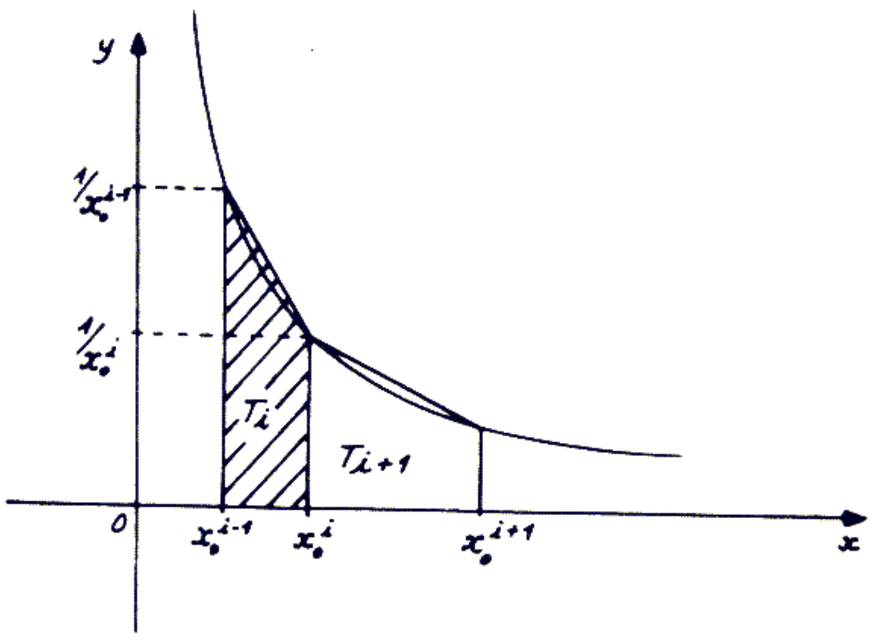
La valeur de l'aire hachurée est celle de ln x

ALGORITHMIQUE

L'ALGO DU LOG

Plusieurs méthodes utilisent cette propriété. Celle présentée ici encadre la courbe $y = 1/x$ entre cordes et tangentes. La surface est partagée en 2^n trapèzes aux abscisses 1, x_0, x_0^2, x_0^3, ..., $x_0^{2n} = x$. L'intérêt de ce partage est que les trapèzes ont même aire. Ainsi calculons l'aire du trapèze T_i limité par x_0^{i-1} et x_0^i et construit sur corde : aire de $T_i = (x_0^i - x_0^{i-1}) \times (1/x_0^i + 1/x_0^{i-1})/2 = (1 + x_0 - 1/x_0 - 1)/2 = (x_0 - 1/x_0)/2$.

L'aire du trapèze T_i étant indépendante de i, nous avons bien $T_1 = T_2 = \ldots = T_{2^n} = (x_0 - 1/x_0)/2$.



Les aires de T_i et de T_{i+1} sont égales

Nous n'allons pas expliquer le détail de la méthode, mais son principe vous a été présenté. Et vous avez maintenant les éléments nécessaires pour pouvoir l'approfondir si vous le désirez.

Cette méthode assez complexe donne pourtant un court programme :

```
10 INPUT « Nombre » ; X
20 Z=(X+1/X)/2 : Y=(X-1/X)/2
30 Z=SQR((1+Z)/2) : Y=Y/Z
40 IF ABS(1-Z)>1E-8 THEN 30
50 PRINT Y
```

Le Y en sortie est un logarithme népérien. Nous vous rappelons que si vous désirez le logarithme décimal, il suffit de mettre PRINT Y/2.3025851 en ligne 50 puisque $\log_b x = \ln x / \ln b$.

L'énorme intérêt de la méthode de l'hyperbole est son côté général. En effet, essayez le même programme en remplaçant uniquement le calcul initial de Z et Y de la ligne 20 par Z=0 : Y=2 (la ligne 10 ne sert plus, le calcul étant indépendant de X). Et qu'obtenez-vous ? Mais oui, le Y de sortie n'est autre que le très célèbre $\pi = 3{,}14\ldots$! Il n'y a plus guère de rapport avec les log !

Selon le contenu de la ligne 20, ce programme est capable de calculer *toutes* les fonctions réciproques trigonométriques et hyperboliques.

Ainsi 20 Z = SQR(1-X*X) : Y = X permet de calculer Arc sin x, ce qui explique mieux le π trouvé précédemment.

Et avec 20 Z = X : Y = SQR (1-X*X) vous obtiendrez Arc cos x.

Quant à 20 Z = SQR(1+X*X) : Y = X, cela donne Arg sh x.

A vous d'en découvrir d'autres !

Nous vous donnons maintenant l'une des méthodes les plus puissantes pour calculer les logarithmes. L'intérêt est dans l'absence complète de boucle, contrairement aux autres présentées jusqu'ici. Le résultat est donc obtenu directement et beaucoup plus rapidement.

Voici in extenso le programme, sans aucune explication, cette dernière étant complexe et sortant du cadre de cet article :

```
10 INPUT « Nombre » ; X
20 X1=3.1622777
30 X2=(X-X1)/(X+X1)
40 X3=X2*X2
50 X4=0.17076579*X3*X3*X3
     +0.063802242*X3*X3
     +0.12906789*X3
     +0.17338396
60 Y=(X4*X3*X3+0.28953773*X3
     +0.86858893)*X2+0.5
70 PRINT Y
```

Ici, le Y de sortie est un logarithme décimal. Cela vous aidera-t-il de remarquer que $X1 = \sqrt{10}$?

Sachez que cette méthode est réellement employée à l'intérieur de certains Basics.

Méthode complexe mais générale

Le programme listé n'est valable que pour $1 \leqslant x < 10$. Pour qu'il soit valable dans le cas général, on peut utiliser les astuces déjà vues pour les autres méthodes. C'est-à-dire rajouter :

```
12 IF X<0 THEN PRINT « Impossible » : END
15 IF X<1 THEN X=1/X : PRINT « - » ;
17 K=0
18 IF X>10 THEN X=X/10 : K=K+1 : GOTO 18
```

et remplacer la dernière ligne par :

```
70 PRINT K+Y
```

Si vous voulez des logarithmes népériens et non des décimaux, faites plutôt :

```
70 PRINT (K+Y)*2.3025851
```

Nous ne vous avons pas montré ici toutes les méthodes existantes pour calculer des logarithmes. Toutefois, les quatre présentées ci-dessus ont été choisies de la plus simple à la plus complexe, et de la moins précise à la plus performante.

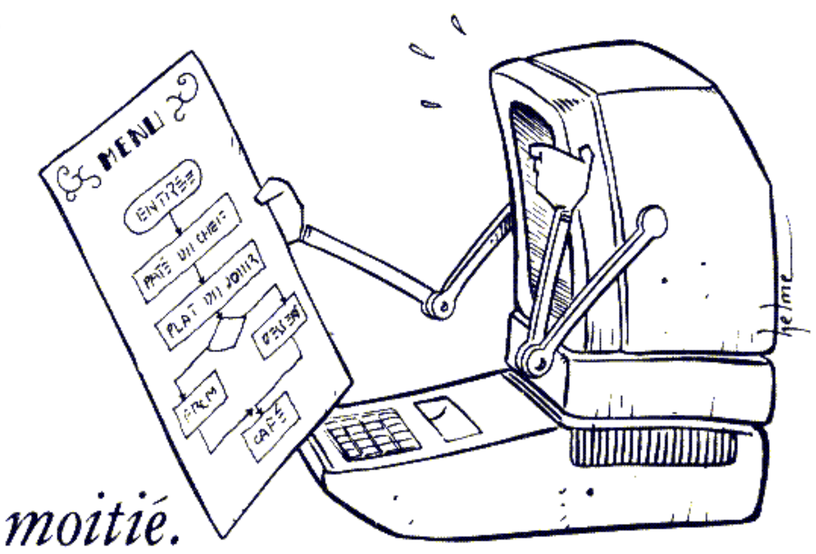
Maintenant vous saurez quoi faire si vous tombez en « panne de log ».

Boris YARENITCH

LIST - PAGE 46

RAPPELER LE MENU

QUAND on réalise, sur un ordinateur, une application complexe, il est d'usage de la découper en sous-ensembles auxquels on accède par un menu général. S'il s'agit de programmes indépendants, on a recours à un menu qui permet de choisir celui que l'on désire et qui le charge. Mais il faut encore que ce dernier, une fois exécuté, rappelle en mémoire le menu, d'où un temps d'attente important. Nous allons voir comment diminuer ce temps de moitié.



Comment faire pour qu'un programme de menu ne soit pas chassé par celui qu'il appelle ? Le principe de base est simple : il suffit de pouvoir faire cohabiter deux programmes en mémoire. Le premier sera le programme de menu. Vous aurez à le charger une seule fois et il restera en mémoire jusqu'à la fin de l'application. Il appellera vos différents programmes secondaires. Une fois ceux-ci terminés, ils relanceront votre menu, sans qu'il soit besoin d'aucun accès au disque.

Quand vous tapez la commande LOAD (ou RUN), l'interpréteur Basic ne charge pas votre programme n'importe où. Il se fie en effet au pointeur de début de programme (voir la table des pointeurs Basic dans l'encadré sur la mémoire-utilisateur), et il met à jour le pointeur de fin de programme. A l'exécution, il modifie les pointeurs d'occupation mémoire des variables. A vous de lui laisser croire qu'il est seul maître du jeu, tout en le mettant à votre guise !

Regardons tout d'abord les listes du programme « MENU » et du programme « CHOIX 1 ».

Les pointeurs de l'interpréteur Basic sont inscrits en mémoire sur deux octets, poids faible-poids fort. Dans le programme MENU, quand votre choix est fait, il faut donc mettre à la place du pointeur DP (début de programme) les valeurs du pointeur FP (fin de programme). Soit : POKE 103, PEEK (105) : POKE 104, PEEK (106).

Programme de menu
pour Applesoft
Auteur Jacques Labidurie


```
10  REM PROGRAMME MENU
11 :
100  HOME
110  VTAB 5: PRINT "1 - CHOIX 1":
      VTAB 7: PRINT "2 - CHOIX 2"
     : VTAB 9: PRINT "3 - CHOIX 3
     ": VTAB 11: PRINT "4 - CHOIX
      4"
120  VTAB 17: HTAB 10: PRINT "VOT
     RE CHOIX ? ";: GET K$: PRINT
     K$:K =  VAL (K$): IF K < 1 OR
     K > 4 THEN  PRINT  CHR$ (7):
      GOTO 120
130  POKE 103, PEEK (105): POKE 1
     04, PEEK (106): POKE  PEEK (
     105) +  PEEK (106) * 256 - 1
     ,0: PRINT  CHR$ (4)"RUNCHOIX
     " + K$
```

Programme CHOIX 1

```
10  REM PROGRAMME CHOIX1
11 :
100  HOME : VTAB 4: PRINT "PROGRA
     MME DE TRAITEMENT DU CHOIX 1
     "
110  VTAB 20: PRINT "TAPER UNE TO
     UCHE POUR REVENIR AU MENU ";
     : GET K$
120  POKE 105, PEEK (103): POKE 1
     06, PEEK (104): POKE 103,1: POKE
     104,8: RUN
```

Leurrer l'interpréteur

Ne pas oublier également que le Basic tient absolument à ce que l'octet précédant la valeur de ce pointeur soit à 0. Pour cela, pas de problèmes : POKE PEEK (105) + PEEK (106) * 256 − 1 , 0. Exécutez ensuite le RUN du programme de votre choix et l'interpréteur n'y voit que du feu. Il va prendre comme zone de chargement la place libre après votre programme MENU ; celui-ci sera donc conservé intact en mémoire centrale.

Il s'agit maintenant, à la fin de CHOIX 1 de revenir automatique-

ment à MENU et ce, sans avoir à le recharger à partir de la disquette. Il vous faut donc rétablir les pointeurs DP et FP comme ils étaient auparavant. Où trouver la fin du programme MENU ? Relisez quelques lignes plus haut, c'est simplement le début du programme CHOIX 1. Et quelle est la valeur à mettre dans DP ?

Si vous n'avez fait aucune manipulation hasardeuse avant de commencer, c'est bien évidemment la valeur standard d'implantation d'un programme Basic sur Apple II : $801 en hexadécimal soit 2049 en décimal. Nous trouvons donc à la ligne 120 d'une part la restauration de FP : POKE 105, PEEK (103) : POKE 106, PEEK (104) et d'autre part celle de DP : POKE 103, 1 : POKE 104, 8.

Si vous n'utilisiez pas cette valeur standard, il aurait fallu conserver la valeur du DP de MENU dans des zones ne risquant pas d'être détruites, donc non utilisées par Applesoft. Par exemple, les adresses 7 et 8. Donc, rajouter à MENU : POKE 7, PEEK (103) : POKE 8 , PEEK (104).

Bien sûr, il ne faut pas oublier, dans ce cas, de modifier la ligne 120 de CHOIX 1 : POKE 103, PEEK (7) : POKE 104 , PEEK (8).

Examinons les valeurs des différents pointeurs Basic après chaque pas de notre exécution. Au départ, le pointeur DP est à $801, et FP à $804. Il n'y a rien de présent en mémoire. Que remarque-t-on ensuite ?

Rappel sur la « mémoire-utilisateur »

Le Basic Applesoft découpe la mémoire-utilisateur en différentes zones :
- mémoire-programme ;
- mémoire des variables simples et pointeurs de chaînes ;
- mémoire des tableaux ;
- mémoire des chaînes de caractères.

Les trois premières zones sont implantées au début de la mémoire-utilisateur ($0800 en hexadécimal, soit 2048 en décimal) et s'étendent vers le haut de la mémoire.

La quatrième, à l'inverse, part de la plus haute adresse ($95FF = 38399) et s'étend vers le bas.

En page 0 ($0000 à $00FF) se trouvent les différents pointeurs permettant de connaître les implantations de ces zones. Toutes ces adresses sont codées sous la forme habituelle sur Apple : octet de poids faible, puis octet de poids fort.

Voici ces pointeurs :

Nom	Adresse en hexadécimal	Adresse en décimal
DP	$67.$68	103.104
FP	$AF.$B0	175.176
LOMEM	$69.$6A	105.106
DT	$6B.$6C	107.108
FT	$6D.$6E	109.110
DC	$6F.$70	111.112
HIMEM	$73.$74	115.116

DP ..... : Début du programme
FP ..... : Fin du programme
LOMEM ..... : Début de la zone variables simples et pointeurs de chaînes
DT ..... : Début de la zone tableau
FT ..... : Fin de la zone tableau
DC ..... : Début de la zone des chaînes de caractères
HIMEM ..... : Fin de la zone des chaînes + 1

La récupération de toutes ces valeurs à partir du Basic est très simple. Par exemple, pour obtenir l'adresse de début de programme, il suffit de demander PEEK (103) + PEEK (104) * 256 (n'oubliez pas la codification « poids faible - poids fort »). A noter aussi :
- l'octet DP – 1 doit toujours être à 0 ;
- un programme Applesoft se termine toujours par trois 0 ;
- LOMEM, enfin, pointe normalement sur le dernier zéro + 2.

La connaissance de ces pointeurs a de multiples utilisations. Vous pouvez ainsi restaurer un programme malencontreusement écrasé par un NEW malheureux, réserver une zone pour un sous-programme assembleur qui sera ainsi sauvé en même temps que le programme Basic, insérer dans votre programme Basic des instructions qui modifient le programme lui-même, etc.

Plan de la mémoire-utilisateur

DP FP	Mémoire Programme
LOMEM	Zone variables simples et pointeurs de chaînes
DT FT	Zone des tableaux
	Mémoire libre
DC	Zone des chaînes de caractères
HIMEM	

Mille choses à faire

Après le chargement de MENU, DP est inchangé, mais FP est positionné à $8E6. Les pointeurs de variables n'ont pas bougé. Vous pouvez vérifier tout cela sans difficulté en suivant les indications fournies dans l'encadré ci-contre.

Après modification des pointeurs et avant chargement de CHOIX 1, les pointeurs de variables ont évidemment bougé mais, surtout, le DP a été positionné à la valeur de FP, soit $8E6.

Après le chargement de CHOIX 1, le programme a bien été chargé en dessus de MENU. Enfin, après modification des pointeurs et avant le retour à MENU, DP a été remis à $801 et FP à sa valeur d'origine $8E6.

Maintenant que ces pointeurs n'ont plus guère de secrets pour vous, essayez d'imaginer des complications supplémentaires :
- faire coexister plus de deux programmes ;
- seulement deux, mais en conservant les valeurs des variables au retour ;
- mieux encore, deux ou plusieurs programmes utilisant la même zone de variables simples...

Et ce ne sont que des exemples. Il y a mille choses à faire en jouant avec les pointeurs.

Jacques LABIDURIE

LE PLUS GRAND NOMBRE PREMIER CONNU

LE plus grand nombre premier connu aujourd'hui est égal à $2^{132049} - 1$. Le petit Z80 ne prétend pas vérifier qu'il est premier mais peut en moins de quatre heures retrouver les 39 751 chiffres qui le composent !

Depuis septembre 1983, le plus grand nombre premier connu est $2^{132049} - 1$ (découvert par l'Américain Slowinski).

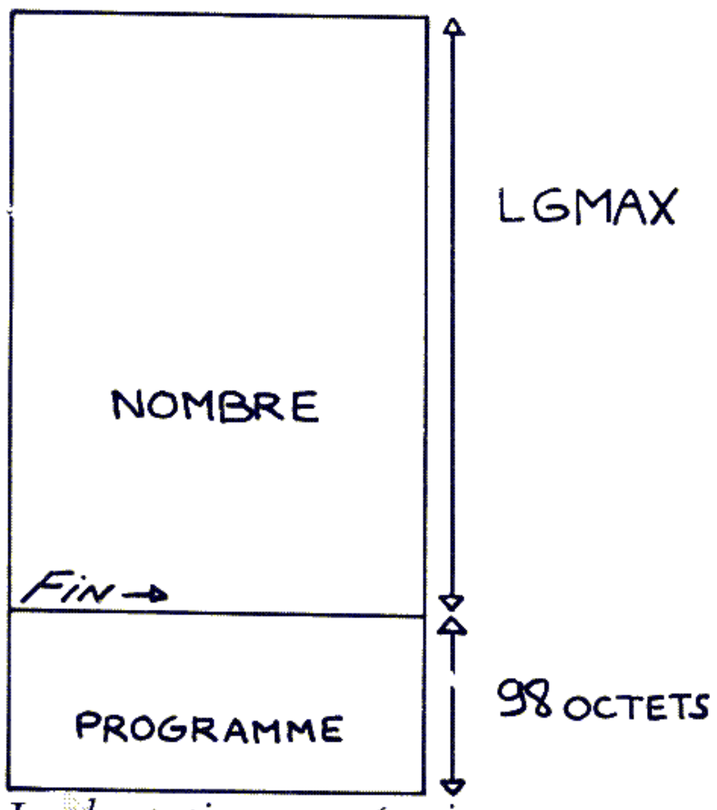
Ce nombre fait partie de la famille des nombres premiers de la forme $(2^n - 1)$, les nombres de Mersenne, du nom d'un moine français qui s'y intéressa le premier en 1644. Depuis, les records successifs du plus grand nombre premier ont presque toujours appartenu à cette famille, qu'ils aient été découverts par Euler au 18e siècle, Lucas au 19e siècle ou Slowinski l'année passée.

Nous ne vous proposons bien sûr pas de vérifier la primalité de $(2^{132049} - 1)$, ce qui nécessite plusieurs heures de calcul sur Cray 1 avec le test de Lucas-Lehmer. Mais par contre, avec un petit Z80, nous allons calculer la valeur exacte de ce nombre et obtenir les 39 751 chiffres qui le composent.

Le programme est assez court, 98 octets seulement, mais a besoin d'une grande zone de travail pour stocker le nombre. L'implantation générale en mémoire est représentée par le schéma de la page suivante.

Le nombre est calculé en binaire codé décimal, un octet contenant

ASSEMBLEUR DU Z80



Implantation en mémoire

deux chiffres. Dans le programme, LGMAX (longueur maximale du nombre) vaut 8 000 H (H comme hexadécimal), soit 32 768 en décimal. Chaque octet contenant deux chiffres, on pourra donc stocker des nombres de 65 536 chiffres : les nombres de Mersenne auront ainsi suffisamment de place pour aller jusqu'à $(2^{217705}-1)$.

Prévoir la durée des calculs

Pour connaître la durée des calculs, observons la boucle principale du programme :

LD	A,(HL)	7 cycles
ADC	A,A	4 "
DAA		4 "
LD	(HL),A	7 "
DEC	H L	6 "
DJNZ		13 "

Cette boucle est formée de 41 cycles.

Deux compteurs sur 8 bits (d'abord B, puis C) ont été préférés à un seul compteur sur 16 bits, BC, qui aurait ralenti le programme de moitié environ.

A chaque appel de DOUB, la boucle est effectuée CB fois, CB représentant la longueur en octets du nombre. Cette longueur varie bien sûr tout au long de l'exécution du programme, passant progressivement de 1 à (39 751/2) pour le nombre que nous

Les nombres de Mersenne

ACTUELLEMENT, on connaît 29 nombres de Mersenne, nombres premiers de la forme (2^n-1). Tous les n n'ayant pas été explorés entre 44 497 et 86 243, il n'est pas sûr que 2^{86243} soit le vingt-huitième nombre de Mersenne. De même entre 86 243 et 132 049.

Rang	Valeur du n de l'expression 2^n-1	Découvert par	Découvert en
1	2		
2	3		
3	5		
4	7		
5	13		
6	17		
7	19		
8	31	Euler	1772
9	61	Pervushin	1883
10	89	Powers	1911
11	107	Powers	1914
12	127	Lucas	1876
13	521	Lehmer et Robinson	1952
14	607	"	"
15	1 279	"	"
16	2 203	"	"
17	2 281	"	"
18	3 217	Riesel	1957
19	4 253	Hurwitz et Selfridge	1961
20	4 423	"	"
21	9 689	Gillies	1963
22	9 941	"	"
23	11 213	"	"
24	19 937	Tuckerman	1971
25	21 701	Nickel et Noll	1978
26	23 209	Noll	Février 1979
27	44 497	Nelson et Slowinski	Avril 1979
?	86 243	Slowinski	Janvier 1983
?	132 049	Slowinski	Septembre 1983

Quelques chiffres

2 147 483 647 découvert par Euler est resté pendant plus d'un siècle le plus grand nombre premier connu.

Actuellement les 500 premiers chiffres du plus grand nombre premier connu sont :

5127402762 6932072381 2785763620 3402218800 4658622706
9926831240 3841858231 2743056203 6107774949 9092908732
1255570932 0045159618 5805491533 7915698134 5993400430
1403420963 8765030513 9593110201 5314923539 8042745823
9674399280 7950474719 2259564935 4975511373 1084255868

1977969184 3458193759 8237719449 6938307582 9555852798
8434483984 4029268453 7504239767 6916772484 1506464109
1777251902 8191260065 7947401797 6932469836 7726983862
1511897039 5984248890 0061272076 0244591123 4087810954
6457346491 5541439258 4268514594 9023067551 5467717759

et les 501 derniers chiffres de ce nombre sont :

8553988330 0940172476 5419165472 8279987863 2042910294
5424906196 5847854995 1321771127 3013288607 3480123682
6849327926 0403717058 6585948941 8424977308 3050086907
5427501338 3432537497 1854459957 0712923244 1541861524
7764010289 0109682185 0407923309 9853896500 4936957667

8110501603 5501363245 8865243152 2439404697 5356744436
0461513964 3318420648 4240783585 7099023061 7213849139
8542760432 7412127556 7738115305 3892561883 9063766021
9368323673 6730822711 6789561494 3253264415 3240796400
4851093298 8337863164 4703566339 8521385784 5573006131 1

Le programme en Z80 vous permettra de calculer les 38 750 chiffres restants !

Nombres de Mersenne, calcul de
2 Λ DEL − 1
Assembleur du Z80
Auteur Christian Boyer


```
Adres. Codes        Commentaires

0000   P            FIN:    EQU   F7FFH
0000   P            LGMAX:  EQU   8000H

F800                        ORG   FIN+1
F800                ;
F800                ;
F800                ;
F800 2ED1           PUISS:  LD    L,209        ; DEL=132049
F802 110302                 LD    DE,515       ;    =515*256+209
F805 CD17F8                 CALL  INIT         ; Initialisation.
F808                ;
F808 CD35F8         PUISS1: CALL  DOUB         ; On double DEL fois.
F80B 2D                     DEC   L
F80C 20FA                   JR    NZ,PUISS1
F80E 1B                     DEC   DE
F80F 7A                     LD    A,D
F810 B3                     OR    E
F811 20F5                   JR    NZ,PUISS1
F813                ;
F813 CD55F8                 CALL  MOINS        ; On soustrait 1
F816 C9                     RET                ; et c'est fini!
F817                ;
F817                ;
F817                ;
F817 E5             INIT:   PUSH  HL           ; Nombre=1
F818 D5                     PUSH  DE
F819 01FF7F                 LD    BC,LGMAX-1
F81C 11FEF7                 LD    DE,FIN-1
F81F 21FFF7                 LD    HL,FIN
F822 3600                   LD    (HL),0
F824 EDB8                   LDDR
F826 21FFF7                 LD    HL,FIN
F829 3601                   LD    (HL),1
F82B D1                     POP   DE
F82C E1                     POP   HL
F82D                ;                          *
F82D 010001                 LD    BC,0100H     ; CB=long. du nombre=1
F830                ;
F830 7D                     LD    A,L          ; On retouche DEL
F831 B7                     OR    A
F832 C8                     RET   Z
F833 13                     INC   DE
F834 C9                     RET
F835                ;
F835                ;
F835                ;
F835                ;
F835 E5             DOUB:   PUSH  HL
F836 C5                     PUSH  BC
F837 21FFF7                 LD    HL,FIN
F83A 78                     LD    A,B
F83B B7                     OR    A
F83C 2801                   JR    Z,DOUB1
F83E 0C                     INC   C
F83F                ;
F83F 7E             DOUB1:  LD    A,(HL)       ; C'est
F840 8F                     ADC   A,A          ; ici
F841 27                     DAA                ; que
F842 77                     LD    (HL),A       ; ca
F843 2B                     DEC   HL           ; se
F844 10F9                   DJNZ  DOUB1        ; passe.
F846 0D                     DEC   C
F847 C23FF8                 JP    NZ,DOUB1
F84A                ;
F84A C1                     POP   BC
F84B 3006                   JR    NC,DOUB2     ; 1er octet=1 si carry
F84D 3601                   LD    (HL),1       ; et longueur CB=CB+1.
F84F 04                     INC   B
F850 2001                   JR    NZ,DOUB2
F852 0C                     INC   C
F853 E1             DOUB2:  POP   HL
F854 C9                     RET
F855                ;
F855                ;
F855                ;
F855 21FFF7         MOINS:  LD    HL,FIN
F858 37                     SCF
F859 7E             MOINS1: LD    A,(HL)       ; Nombre=nombre-1
F85A DE00                   SBC   A,0
F85C 27                     DAA
F85D 77                     LD    (HL),A
F85E 2B                     DEC   HL
F85F 38F8                   JR    C,MOINS1
F861 C9                     RET
F862                ;
F862                ;
F862                ;
F862                        END
```

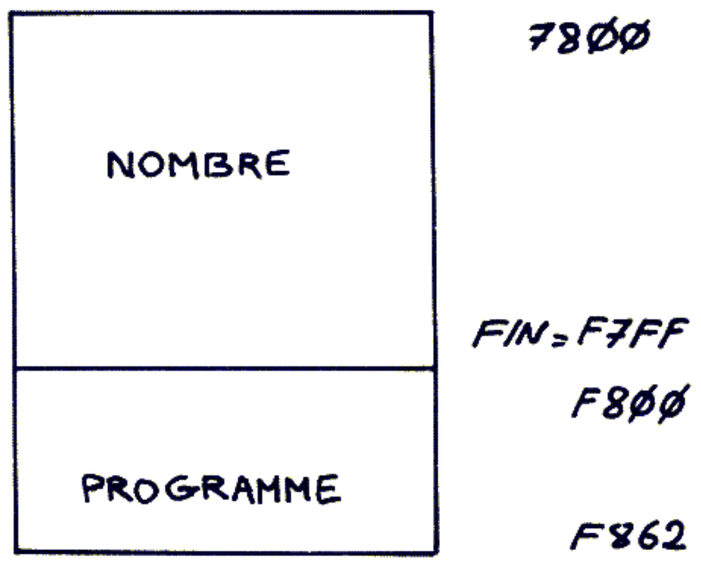
cherchons. On peut ainsi en déduire que, si le quartz a une fréquence de 4 MHz, la durée t du calcul de $(2^n - 1)$ est :

$t = 7{,}72 \times 10^{-7} \times n^2$ secondes.

Cette durée étant fonction du carré de n, sa croissance est très rapide. Pour $(2^{132049} - 1)$, il vous faudra attendre trois heures et quarante-cinq minutes, alors que pour $(2^{3217} - 1)$, huit secondes suffiront.

Et si l'on veut modifier...

Comment adapter ce programme à votre système Z80 ? Les *seules* valeurs à changer sont celles de FIN et LGMAX selon l'espace mémoire disponible. Dans le cas du programme proposé ici, l'occupation de l'espace est représentée ci-dessous :



Occupation de l'espace mémoire

Après l'avoir lancé (et avoir attendu patiemment la fin de son exécution !), ce programme vous donnera le nombre cherché, ses deux derniers chiffres étant en FIN.

Si vous voulez calculer un nombre de Mersenne autre que $(2^{132049} - 1)$, il vous suffit de modifier en conséquence les LD L et LD DE du début du programme. Et comme le dix-huitième nombre de Mersenne est déjà composé de près de mille chiffres, vous apprécierez certainement la rapidité de l'assembleur...

Christian BOYER

BASIC ÉTENDU POUR ORIC

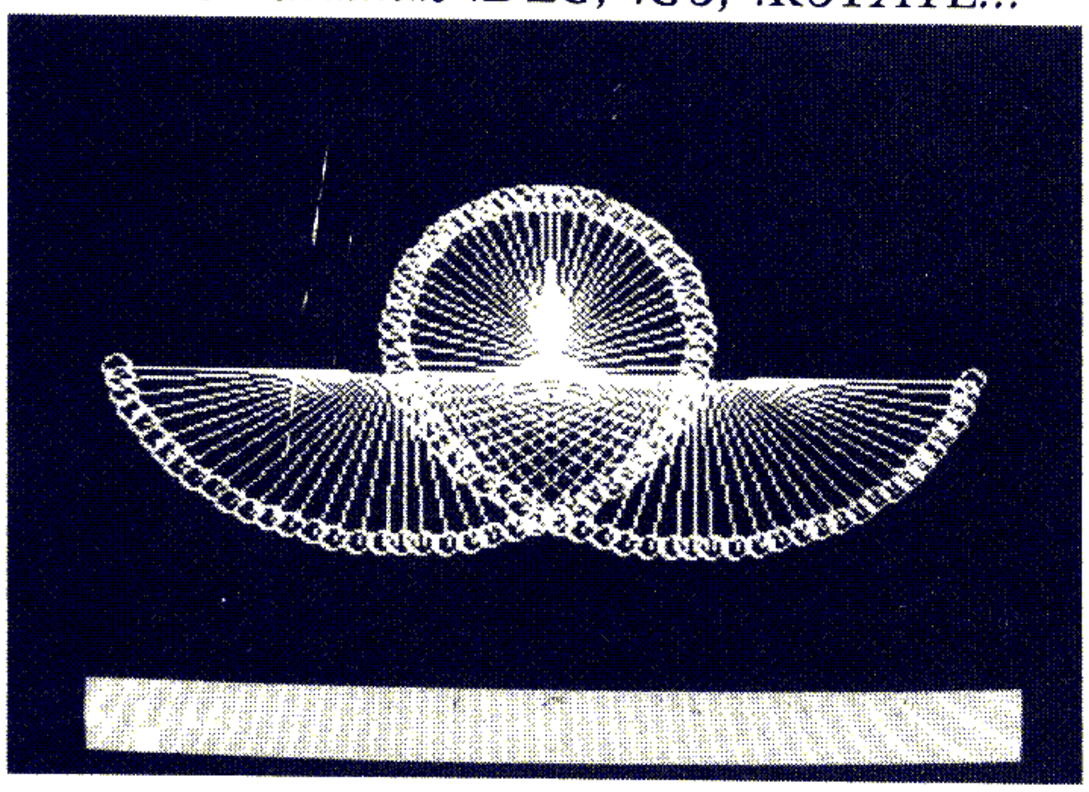
UNE seule et même cassette, « Oric, Basic étendu », enrichit à la fois le Basic de l'Oric-1 et celui de l'Atmos. Orientée surtout vers le graphisme, elle dote l'Oric-1 de treize fonctions supplémentaires alors qu'elle en apporte vingt-trois à l'Atmos. Elle est éditée et distribuée par Infogrames et coûte environ 160 FF ttc.

Le Basic de l'Oric-1 ressemble beaucoup à celui de l'Atmos. Pourtant, la cassette *Oric, Basic étendu* n'agit pas de la même façon sur ces deux ordinateurs. La face 1 enrichit l'Oric-1 de 13 nouvelles commandes Basic et occupe, pour cela, 2 Koctets de mémoire vive. Quant à la face 2 de cette cassette, elle fournit 23 nouvelles commandes à l'Atmos en accaparant 3 Koctets de mémoire vive.

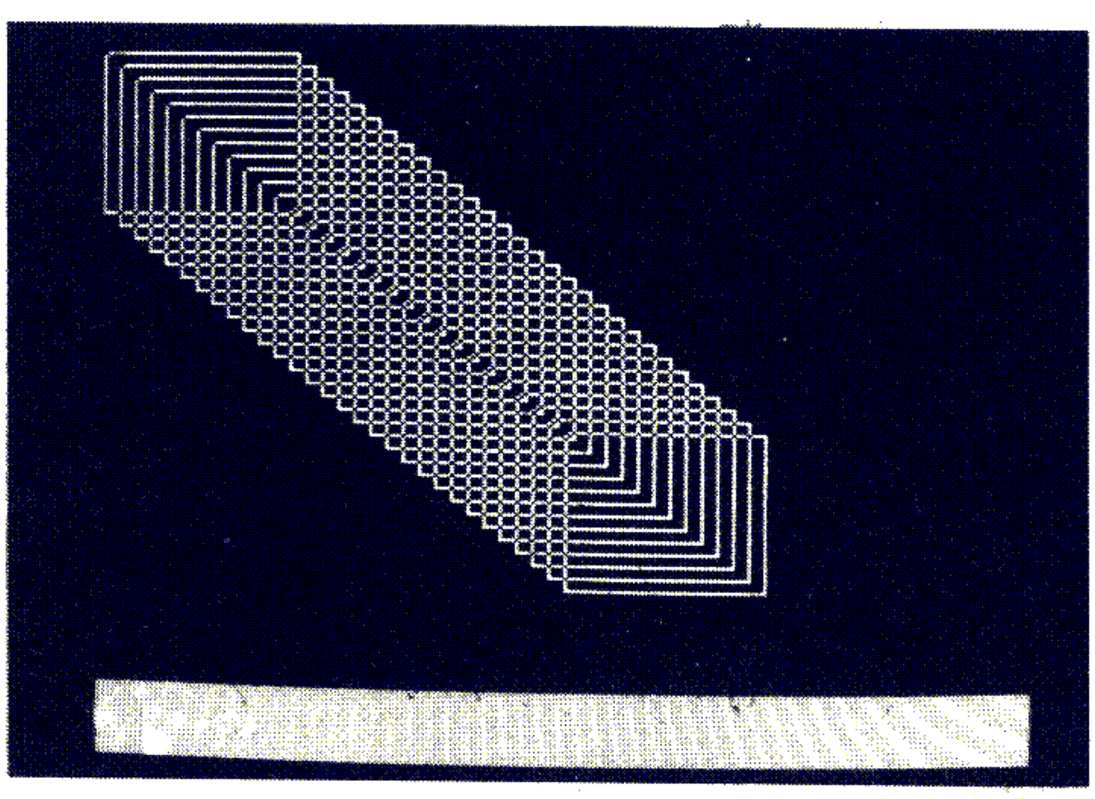
Cette extension du Basic est essentiellement graphique. Les commandes se présentent précédées d'un point d'exclamation. Certaines se retrouvent sur les deux machines. C'est le cas de !BOX qui trace des rectangles ou de !SCROLL qui déplace l'écran, en mode HIRES comme en mode TEXT. Avec cette dernière instruction, l'écran se déplace dans une direction choisie à l'intérieur d'une fenêtre définie par quatre paramètres : la ligne de début, la ligne de fin de fenêtre (de 0 à 26 en mode TEXT et de 0 à 199 en mode HIRES), la colonne de début et la colonne de fin de fenêtre (de 0 à 39 sous les deux modes). En mode HIRES, lorsque la fenêtre est importante, le *scrolling* est

Munie du Basic étendu la haute résolution fait ses preuves

Ici avec les instructions !DEG, !GO, !ROTATE...



et là, avec l'instruction !BOX.



assez long car le bloc mémoire est déplacé octet par octet.

L'instruction !CSET redéfinit des caractères et la réservation de huit octets consécutifs n'importe où dans la mémoire permet à !CHAR d'afficher les caractères en mode HIRES.

Avec !GO, on retrouve une instruction assez similaire à celle du logo (AVANCE, pour les francophones),

Le logiciel en quelques lignes

Nom : Oric, Basic étendu
Ordinateur : Oric-1 et Atmos
Forme : cassette
Édité et distribué par : Infogrames
Prix public : 160 FF ttc
Nombre de fonctions nouvelles : 13 sur Oric-1 et 23 sur Atmos
Principale orientation : graphismes
Autres orientations : gestion de cassette, son
Occupation mémoire : 2 Koctets sur Oric-1 et 3 Koctets sur Atmos

l'angle de direction étant fixé par !ROT (!ROTATE sur l'Oric-1) et initialisé par !DEG. D'autre part, la valeur de cet angle peut toujours être connue par ? & (0) car l'expression & (x) est traitée comme une variable. Grâce à l'instruction !WRITE, des chaînes de caractères sont directement affichées à l'écran, en mode HIRES. Attention cependant sur l'Atmos : toute tentative d'utilisation de cette instruction en mode TEXT bloque la machine. Il ne reste plus alors qu'à la débrancher... Espérons que cette erreur sera corrigée sur les prochaines versions.

Toutes les commandes graphiques du Basic étendu décrites jusque-là sont disponibles sur l'Oric-1 et sur l'Atmos. Pour des raisons inconnues, d'autres commandes sont réservées uniquement à l'Atmos. C'est le cas de !INVERSE qui remplace chaque couleur par sa couleur complémentaire. Pour les animations, il vaut mieux utiliser cette instruction en mode HIRES à cause de la lenteur d'exécution due, ici encore, à la taille de la page mémoire graphique.

L'instruction !CLS doit être suivie de deux paramètres. Elle efface l'écran en mode TEXT et place dans les deux premières colonnes les deux attributs spécifiés par les paramètres. En mode HIRES, une utilisation intéressante de cette instruction est le changement de couleur des trois lignes de commande.

Le Basic de l'Atmos s'étend même au son puisque !PING fournit un *ping* de fréquence déterminée, et !RISE ou !FALL, un son de fréquence croissante ou décroissante, assez peu harmonieux d'ailleurs.

Bravo pour les sauvegardes

Deux instructions très sérieuses et intéressantes se trouvent dans les deux extensions de Basic. Tout d'abord !RECALL sur Oric-1 (!OLD sur Atmos) repêche un programme effacé par erreur avec NEW. Quant à !ESAVE, elle sauve à la fois le programme utilisant le Basic étendu et le Basic étendu lui-même. Cette dernière instruction (suivie de la même syntaxe que CLOAD donne au Basic étendu tout son intérêt : grâce à elle, il n'y a pas à recharger la cassette à chaque fois que l'on utilise un programme écrit à l'aide du Basic étendu.

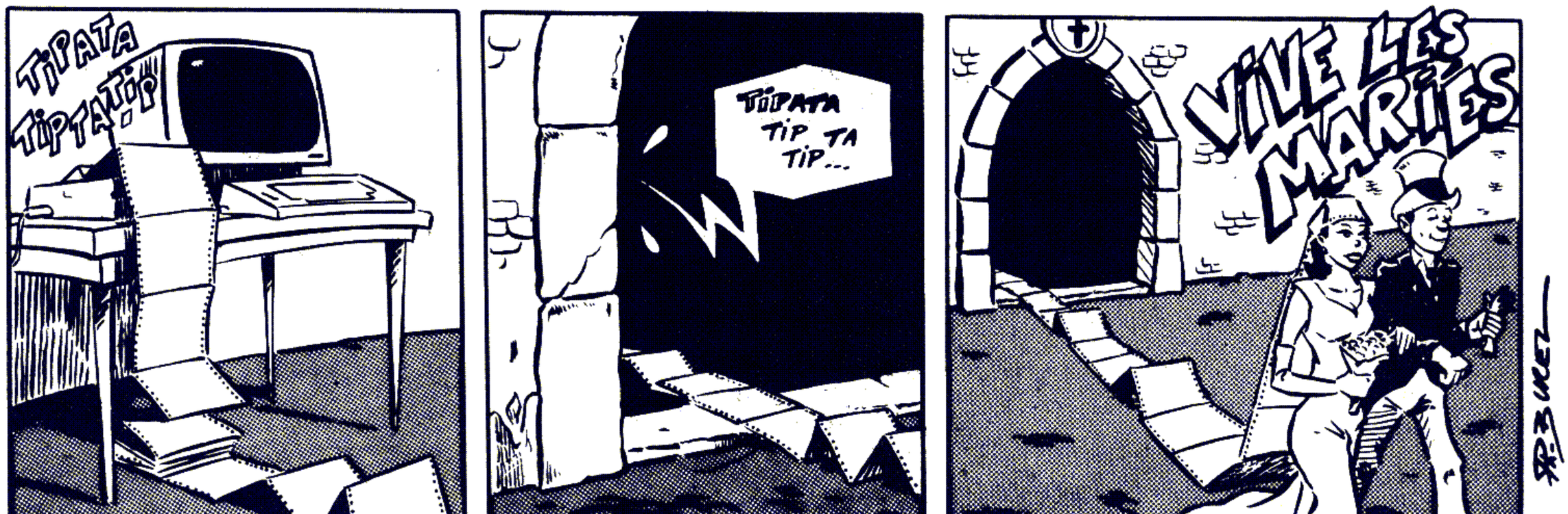
Pour ce qui est de la gestion des cassettes, on trouve, pour l'Oric-1, !VERIFY qui vérifie que le programme en mémoire a bien été sauvé, et !FIND qui retrouve le nom des programmes enregistrés sur cassette. Ces commandes rendent la gestion des cassettes de l'Oric-1 comparable à celle de l'Atmos. Mais !STORE fait toujours défaut.

Voulez-vous savoir l'heure ?

Le Basic étendu enrichit encore l'Atmos de fonctions gérant une horloge affichée sur la ligne de contrôle, en temps réel : !ONCK, !OFFCK et !TIME. Le temps n'est d'ailleurs pas vraiment réel, car certaines instructions, comme ZAP, le font dériver.

Enfin, !COPY permet une recopie d'écran sur imprimante, en mode TEXT uniquement, et !GET "DONNEE" laisse l'Atmos inactif aussi longtemps que DONNEE n'est pas introduit. En conclusion, les instructions graphiques sont intéressantes mais ne sont pas vraiment sophistiquées. La possibilité de sauver en même temps l'extension du Basic et son application est un « plus », absolument nécessaire lorsque le nombre de commandes supplémentaires est aussi important. C'est un point positif pour ce Basic étendu qui est, on peut le rappeler, surtout orienté vers les graphismes.

Denis SEBBAG



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...

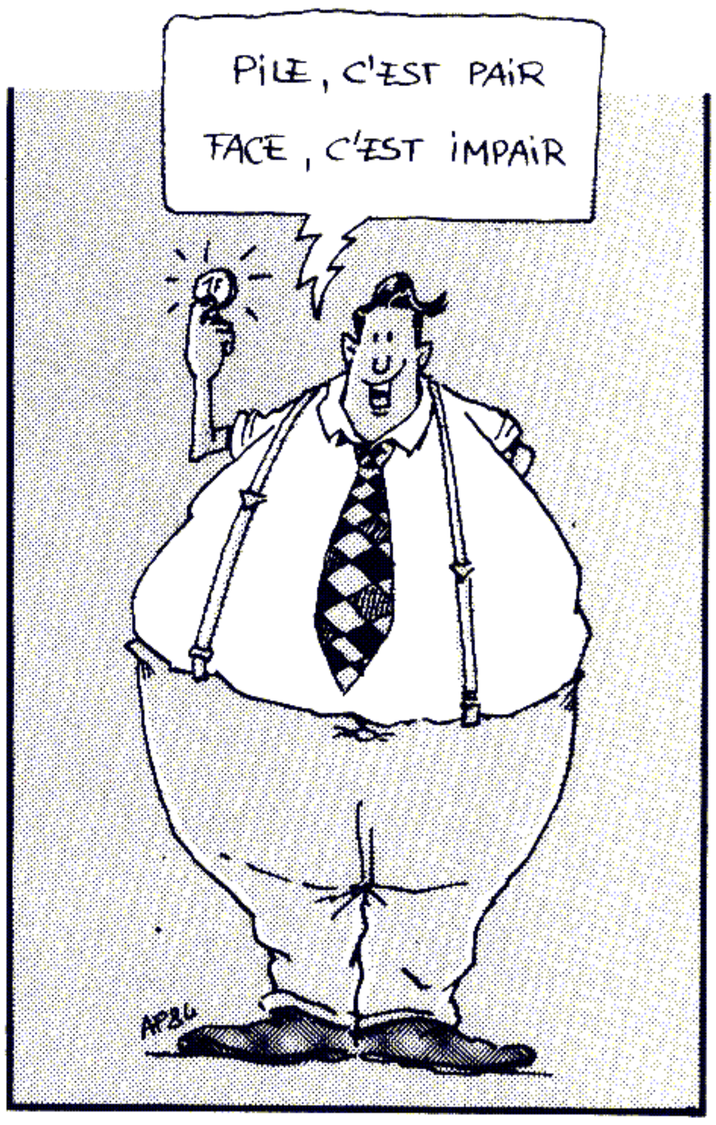
4

Court et structuré

La revue américaine *Creative Computing* a publié, dans son numéro d'août 1978, le programme suivant :

```
10 let X=0
20 print X,2^X
30 let X=X+1
40 if X=18 then 60
50 goto 20
60 end
```

Comme vous pouvez le constater, ce programme ne présente pas de grande difficulté, ni sur le plan des résultats ni sur le plan de la programmation. Au fait, qu'est-il chargé d'afficher à l'écran et comment le récrire pour qu'il soit d'une part plus court, et d'autre part, plus structuré, c'est-à-dire plus lisible ?



5

Pair ou impair

Le langage Pascal dispose de la fonction ODD(I) qui indique si l'entier I qui lui est passé en paramètre est pair (valeur FALSE, soit 0), ou impair (valeur TRUE, soit 1). Mais peut-être ne programmez-vous pas en Pascal. Dans ce cas, cette fonction est simple à écrire, puisqu'elle est égale au résultat de l'expression :

(i mod 2)

L'opérateur mod (modulo) qui est utilisé calcule le reste de la division entière. Ainsi si i est pair, le reste de sa division par 2 est égal à 0, mais il vaut 1 si i est impair. Encore faut-il disposer de cet opérateur mod. S'il est absent de votre ordinateur, il faudra utiliser une autre formule :

$(1 - (-1)^i)/2$

Cette expression retourne bien, en effet, la valeur 0 lorsque i est pair, et 1 lorsque i est impair. Toutefois, cette formule n'est, par sa complexité, pas très satisfaisante. Elle utilise en effet une élévation à la puissance qui met en œuvre le logarithme et l'exponentielle. Ce qui nécessite de nombreux et longs calculs. Vous n'aurez sans doute aucun mal à en trouver une plus simple, avec les fonctions couramment disponibles dans les différents langages.

---

* *Les solutions de ces jeux paraîtront dans le prochain numéro de* LIST.

6

L'inventaire incomplet

Comme de nombreux problèmes que nous soumettons à votre sagacité, celui-ci est réel, et provient d'un programme de gestion de stock d'une grande société de distribution. Pour chaque élément du fichier stock, il est mémorisé :

- le nombre d'articles commandés,
- le nombre d'articles livrés,
- le nombre d'articles vendus.

Le but est d'imprimer à la fin de chaque mois un inventaire, c'est-à-dire la liste de tous les articles qui ont fait l'objet d'au moins un mouvement, à savoir une commande, une livraison ou une vente. Le programme d'édition se présente de la façon suivante (ici, en Pascal) :

```
REPETER
   Lire_article_suivant
   SI Commandé + Livré + Vendu <> 0 ALORS
      Imprimer_article
   FIN_SI
JUSQU_A_Fin_de_fichier
```

Or, un mois, l'utilisateur a eu la surprise de ne pas voir apparaître sur la liste un article qui pourtant avait fait l'objet non seulement d'une commande, mais également d'une vente.

Pouvez-vous dire ce qui s'est passé, et comment corriger le programme pour que cela ne se reproduise pas ?

SOLUTIONS DU NUMÉRO PRÉCÉDENT

▸▸▸ 1

Remontons un peu dans le temps...

Pour calculer les numéros du mois et de l'année qui précèdent le mois de base, il existe deux jeux de formules, selon que le calcul de l'année est réalisé avant ou après celui du mois. Si le mois est calculé en premier, les formules sont les suivantes :

Mois = ((Mois + 10) mod 12) + 1
Année = Année − (Mois div 12)

Par contre, si l'année est calculée en premier, sa formule de calcul doit être modifiée ainsi :

Année = Année + ((Mois + 10) div 12) − 1
Mois = ((Mois + 10) mod 12) + 1

La première formule, plus simple, est bien entendu préférable.

▸▸▸ 2

Pas un bit de plus

Le nombre de valeurs différentes qu'il est possible de représenter avec B bits est égal à :

$N = 2^B$ où $B > 0$

Le nombre minimum de bits nécessaires pour représenter N valeurs distinctes est donc le plus petit entier B tel que :

$2^B >= N$ où $B > 0$

En prenant le logarithme des deux membres de l'inégalité, nous obtenons :

$\log (2^B) >= \log N$
où $N > 0$ et $B > 0$

Si nous notons PLAFOND(X) la fonction retournant le plus petit entier supérieur ou égal à X, cela revient à écrire :

B = PLAFOND (log N/log 2) où N > 1

qui est la formule désirée. Pratiquement, cette formule peut être programmée, compte tenu des problèmes d'arrondi, de la façon suivante :

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

où ENT(X) est la fonction retournant la partie entière du nombre réel X. Par exemple, avec N égal à 3, nous avons :

B = ENT (1,0986/0,6931 + 0,99999)
B = ENT (2,5850)
B = 2

puisqu'il faut effectivement deux bits pour mémoriser trois valeurs distinctes.

▸▸▸ 3

I et J, mais pas K

L'échange des valeurs de I et de J, sans passer par une variable auxiliaire, s'effectue simplement par les trois instructions suivantes :

I = I + J
J = I − J
I = I − J

Ce procédé ne provoque toutefois pas, sur certaines machines, le même résultat que le passage par une variable auxiliaire, et ceci dans deux cas.

Tout d'abord, lorsque les deux variables I et J ne sont pas du même ordre de grandeur. En effet, compte tenu du nombre de chiffres significatifs disponibles sur la machine utilisée, il peut arriver que I + J soit égal à I. Par exemple, l'addition de 1 E 30 et de 12 a en général pour résultat 1 E 30, c'est-à-dire que la valeur 12 est perdue par l'ordinateur. Dans ce cas, les valeurs de I et de J ne sont plus celles que l'on attendait.

Ce procédé est également mis en défaut lorsque les deux valeurs sont très grandes. Par exemple, pour un ordinateur pouvant stocker des nombres jusqu'à 32767, si I et J valent tous deux 32000, leur somme est égale, en général, à −1536 (32000 + 32000 − 2 * (32767 + 1)), et le résultat est ainsi faussé.

Toutefois, ces deux remarques ne mettent pas en cause la méthode employée (qui reste juste) mais son utilisation qui dépend des capacités de la machine utilisée. ■

Chez Duriez : 15 micros portatifs + 9 domestiques

bons pour le Service

ATARI, CANON, CASIO, COMMODORE, HEWLETT PACKARD, ORIC, SHARP, SINCLAIR, THOMPSON.

Avez-vous vu les **300 prix**

valables jusqu'au 20 septembre

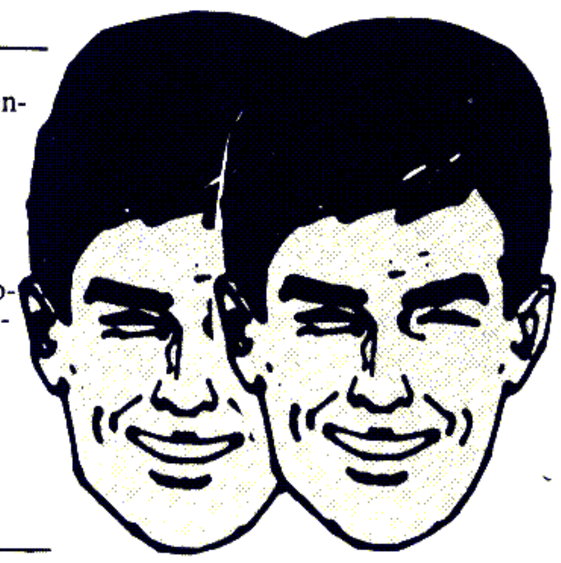
Charter© Duriez ?

POUR CHOISIR, pensez 2 fois.

1° Les performances de l'appareil ?

2° Les performances des programmes disponibles ?

Duriez fait des sélections pour vous éviter des regrets. Vous êtes tranquille.



AU CŒUR DU QUARTIER LATIN, Duriez vend en magasin et par poste à prix charter.©

Il publie régulièrement bancs d'essai et Catalogues condensés de caractéristiques techniques précises, sans délayage publicitaire, complétés par des appréciations et des tests Duriez sans complaisance.

Ce banc d'essai est gratuit en magasin, ou envoyé par poste contre 3 timbres à 2,10 Frs.

ATARI

600 XL Péritel	2500
800 XL Péritel	3380
Magnéto	890
Lecteur de disquette	3690
Imprimante courrier	3490
Traceur 4 couleurs	2590
Manette de jeu	120
Cordon magnéto	65
Coupleur optique	470
Inter. RS232 + cordon	725
Cordons imp. parallèle	295
Secteur	82
Carte Fichiers	530
Carte Graphique	530
Cassette Stat	298
Cassette Graph	298
Cassette Text	298

★★★★★★★★
Machines à écrire
- Photocopieurs
- Répondeurs téléphoniques
- Calculatrices
- Papeterie
- etc...

☐ Demandez le nouveau **catalogue général Duriez** contre 3 timbres à 2,10 F.

☐ **Duriez, 112 et 132 bld St-Germain 75006 Paris (M° Odéon, St-Michel)**
★★★★★★★★

CANON

X07 mémoire 8K	2170
Traceur 4 coul. X710	1850
X07 + X710	3900
Extension 8K	750
Carte mém. 4K XM100	412
Carte mémoire 8K XM101	850

CASIO

PB 700	1640
Traceur 4 coul. FA10	2280
PB 700 + FA10	3850
Extension 4KO R4	427
Magnéto intégré CM1	850
Interface FA4	865
Fx 702P	1050
Interface magnéto FA2	280
Imprimante FP10	610
Fx 802P	990
PB 100	635
Interface magnéto FA3	285
Imprimante FP12	600
FP 200	2990
Extension 8K	623
Cordon magnéto	85
Traceur 4 coul.	2280
Lecteur de disquettes	4430
Clavier numérique	512
Secteur	225
Cordon impri. parallèle	390
Extension CETL (ROM)	809

COMMODORE

Commodore 64 Pal	2750
Commodore 64 Péritel	3450
Commodore VIC 20 Pal	1550
Commodore VC 20 Secam	2100
Extension mémoire 3K	295
Extension mémoire 8K	416
Extension mémoire 16K	665

PERIPHERIQUES VIC20 et C64

Lecteur de cassettes	465
Lecteur de disque 1541	3380
Imprim. 50 cps MPS801	2690
Traceur 4 couleurs	1995
Interface RS232C	345
Manette de jeu	120
Crayon lumineux	475

LOGICIEL VIC 20

Super expender	430
Programmer's aid	350
Screen Master	415
VIC Forth	800

LOGICIEL C64

Utilitaire	
TOOL 64 (cart)	640
Master (disq)	950
64 Forth (cart)	659
Zoom Pascal (disq)	456
HES MON 64 (cart)	440
Professionnel	
HES Writer (cart)	498
Omnicalc (cart)	587
Stat 64 (cart)	490
Graph 64 (cart)	380
Multiplan (disq)	990
Vizawriter (disq)	1355
Super Base 64 (disq)	1600
Educatif	
Turtle graphic (cart)	659
Paint brush (cart)	223
Sinthy 64 (K7)	326
Turtle Toyland (disq)	365
Coco (disq)	440
Jeux	
Choplifter (cart)	495
Lode Runner (cart)	495
Attack of Mutant Camel (cart)	384
Lazer Zone (cart)	328
Gridrunner (cart)	328
Rootin tootin (cart)	365
Omegarace (cart)	215
Space rescue (disq)	495
Speed Bingo (cart)	215
Clowns (cart)	215
Kickman (cart)	215
Sea Wolf (cart)	215
Jupiter Lander (cart)	215
Radar rat race (cart)	215
Echec Grand Master (K7)	305
Kong (K7)	125
Scramble (K7)	125
Motor Mania (K7)	165
M TNT (cart)	329
Benji (cart)	236
The Pit (cart)	329

EPSON

PX 8	10300
Extension mémoire 60K	3300
Extension mémoire 120K	4660
HX 20	5800
Magnéto intégré	1100
Extension 16K	1200
Modem + cordon	1755
Cassette Intext	780

HEWLETT-PACKARD

HP 11C	810
HP 15C	1235
HP 12C	1235
HP 16C	1235
HP 41C	1765
HP 41 CV	2540
HP 41 CX	2880
HP 71	5100
Extension mémoire 4K	784
Lecteur de cartes magnétiques	1634
Interface	1318
Lecteur de cartes	1850
Lecteur optique	1190
Imprimante 82 143	3690
Accus rechargeables	390
Chargeur	155
40 cartes magnétiques	239
Papier therm. noir (6b.)	120
Mémoire quadruple	809
Module X fonction	809
Module temps	809
Module mémoire tampon	809

PERIPHERIQUES HPIL

Module HPIL pour HP41	1348
Lecteur de cassette digit.	4770
Imprim. thermique HPIL	4770
Interface TV	3450
Interface moniteur	2290
10 mini cassettes digit.	990

OLIVETTI

M10 mémoire 8K	5890
M 10 mémoire 24K	6990
Traceur 4 coul.	2090
Secteur	98
Cordon imp. parallèle	199
Cordon imp. RS 232	498

ORIC ATMOS

Oric Atmos 48 K	2330
Cordon Péritel + alimentation 12 V	95
Traceur 4 coul. + cordon	1510
Cordon magnéto (jack)	45
Cordon imp. parallèle	150
Modulateur noir et blanc	210
Modulateur coul. SECAM	530
Lecteur de disquettes 3"	3600
disquette 3"	69
Aigle d'or (K7)	180
Categoic (K7)	95
Xenon (K7)	120
Zorgon (K7)	120
Hobit (K7)	249
Forth (K7)	180
Anglais Assimil (K7)	440
Author (K7)	187
Oric Calc (K7)	187
Poly Fichier	180

SHARP

PC 1500 A	2065
Traceur 4 coul. CE 150	1990
PC 1500 A + CE 150	3990
Extension 8K CE 155	790
Ext. 8K Protégée CE 159	1000
Ext. 16K Protégée CE 161	1700
Interf. RS232/Parallèle	1990
Cable imp. parallèle	480
Clavier sensitif	1265
PC 1251	1215
PC 1245	790
PC 1401	1290
Interface magnéto	169
Imprimante CE 126P	790
Imp. + magnéto CE 125	1695

SINCLAIR

ZX 81	580
Extension 16K	360
Spectrum 48K Péritel	2325
Spectrum 48K Pal	1965
Interface Péritel	360

TEXAS INSTRUMENTS LOGICIEL

Jawbreaker II (cart)	250
Othello (cart)	188
Mash (cart)	250
The Attack (cart)	134
Star Trek (cart)	250
Return to Pirate I. (cart)	250
Tombstone City (cart)	188
Super Demon Attack (cart)	250
TI Invaders (cart)	188
Hopper (cart)	250
Mind Challenger (cart)	134
Burger Time (cart)	250

THOMSON

MO 5	2387
Lecteur de K7	598
TO7-70	3486
Lecteur K7	690
Extension 64K	1055
Contrôleur de communic.	850
Manettes jeux et son	580
Lecteur dis. avec cont.	3596
Memo Basic	480
Cordon imp.	290
Interface SECAM	530

LOGICIELS TO7

Basic vol. 1	195
Basic vol. 2	195
Basic vol. 3	195
Basic vol. 4	195
Basic vol. 5	195
Basic vol. 6	195
Atomium	350
Echo	260
Survivor	350
Logicod	295
Gemini	260
Crypto	295
Motus	295
Tridi	260
Trap	375
Pictor	495
Melodia	495
Sauterelle	125
Compléments et mult.	120
Carré magique	175
Horloge	125
Encadrement	120
Carotte malicieuse	175
Diététique	175
Allemand vol. 1	195
Allemand vol. 2	195
Mots croisés 1	195
Mots croisés 2	195
Budget familial	450
Carnet d'adresses	480
Gérer vos fichiers	525
Ronde des chiffres	125
Noix de coco	145
Carte de France	145
Mots en fleurs	185
Bibliothèque	490
Coktail 1	95
Coktail 2	95
Coktail 3	95
Calculatrice	360
Agenda	490
Portefeuille boursier	580
Mélimélo	437
Clé des champs	170
Quest (ROM)	325
Quest histoire géographie	66
Quest sport	66
Quest sciences	66
Signes dans l'espace	175
Système métrique	150
Pickman	120
Stock car	120
Yams	179
Loto	128
Ronde des formes	148

Je commande à Duriez : 132, Bd St-Germain, 75006 Paris.

☐ 1 Catalogue Duriez "Micros" (essais comparatifs des 20 micro-ordinateurs les plus vendus chez Duriez) contre 3 timbres à 2,10 F.

☐ Le(s) article(s) entouré(s) sur cette page photocopiée (ou cités ci-dessous).

Si changement de prix, je serai avisé avant expédition.

☐ Ci-joint chèque de .....

..... F

y compris Port et Emballage 40 F.

☐ Je paierai à réception (Contre-Remboursement) moyennant un supplément de 30 F + 40 F Port et Emballage.

J'aurai le droit, si non satisfait, de renvoyer sous 8 jours le(s) appareil(s) modules, cassettes ou ouvrages Duriez, qui me remboursera la somme ci-dessus, (sauf suppl. 30 F du C. Rb), port et emballage.

Mes Nom, Prénoms, Adresse (N°, Rue, Code, Ville) :

.....

.....

.....

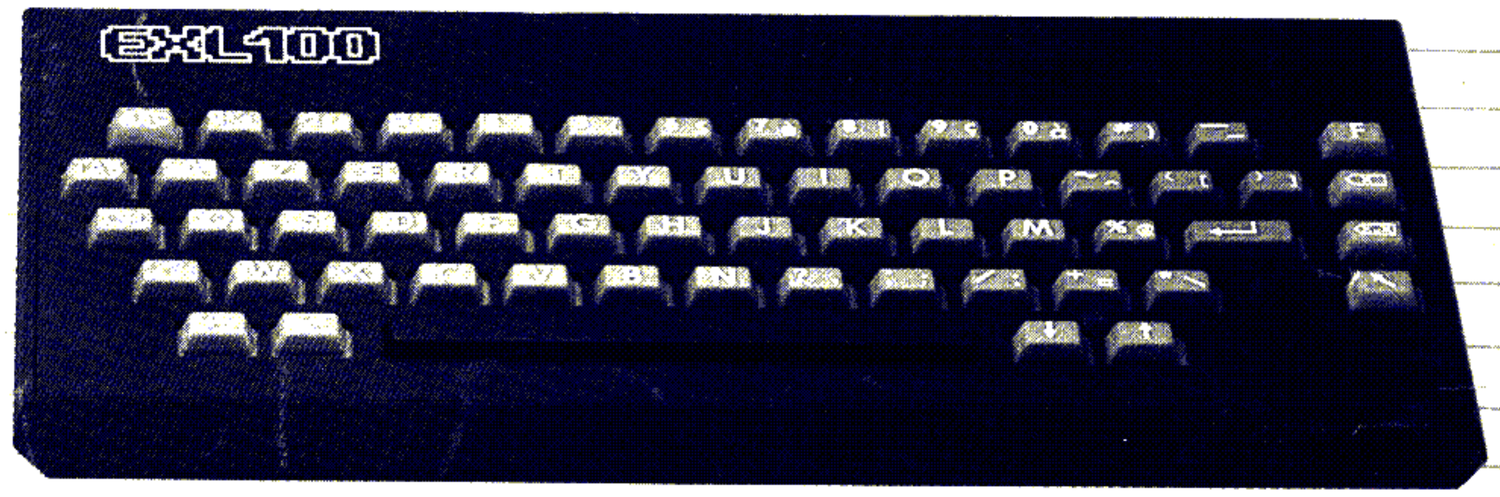
Date et Signature .....

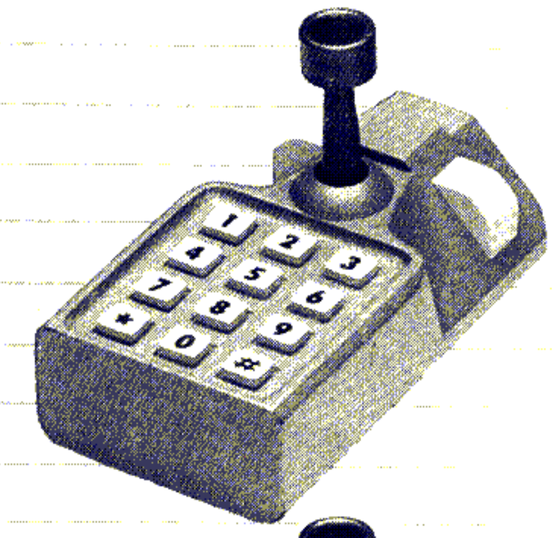
.....

LI Sept. 84 Média Conseil

Imprimerie SIMA, rue des Epinettes, ZI de Torcy Sud, 77200 Torcy. Printed in France. Directeur de la Publication : Jean-Luc Verhoye
Dépôt légal août 1984. Diffusion NMPP. Numéro de Commission paritaire : 63617

La réponse française !





Offre spéciale de lancement

Cette offre spéciale comprend :

- 1 unité centrale 32 K utilisateurs avec synthèse de la parole.
- 1 clavier azerty accentué, infrarouge.
- 2 manettes infrarouge.
- 1 module Basic (32 K).
- 1 manuel d'utilisation.
- 1 cordon Péritel.
- 1 cordon secteur.
- 1 livre complet de programmes prêts à l'emploi.

exelvision

Coupon-réponse à renvoyer à VECTRON 73, rue du Cherche-Midi - 75006 Paris
Tél. : (1) 549.14.50

Veuillez me faire parvenir, sans engagement de ma part, une documentation complète de votre offre spéciale "EXEL 100".

Nom :

Adresse :

Code postal :

Ville :

5

ARROW 64
UN ÉDITEUR-ASSEMBLEUR POUR COMMODORE 64

DESTINÉ au Commodore 64, le logiciel Arrow 64 est un éditeur-assembleur qui intéressera surtout les programmeurs en langage-machine.
Arrow 64 se présente sous forme de cartouche ; édité par Micro Application Software, il est accompagné d'un guide d'utilisation en français qui frappe par sa minceur.
Il coûte environ 640 FF ttc.

Le logiciel *Arrow 64* pour Commodore 64 devrait permettre, même aux débutants, de faire un bon usage des fonctions d'édition ou de l'assemblage de programmes. Encore faut-il ne pas avoir peur d'entreprendre des recherches personnelles car le guide d'utilisation est vraiment très mince.

Un rapide coup d'œil sur ce guide nous indique que les possibilités d'*Arrow* se subdivisent en quatre catégories distinctes, et complémentaires. Nous commencerons par les commandes de gestion du magnétophone.

Les trois commandes classiques de ce périphérique sont conservées : LOAD, SAVE et VERIFY. Mais *Arrow* leur en ajoute une quatrième : APPEND, tout en améliorant la vitesse d'une façon appréciable. Les transferts peuvent maintenant se faire à 3 600 Bauds, soit une multiplication par 6 environ de la vitesse habituelle de travail.

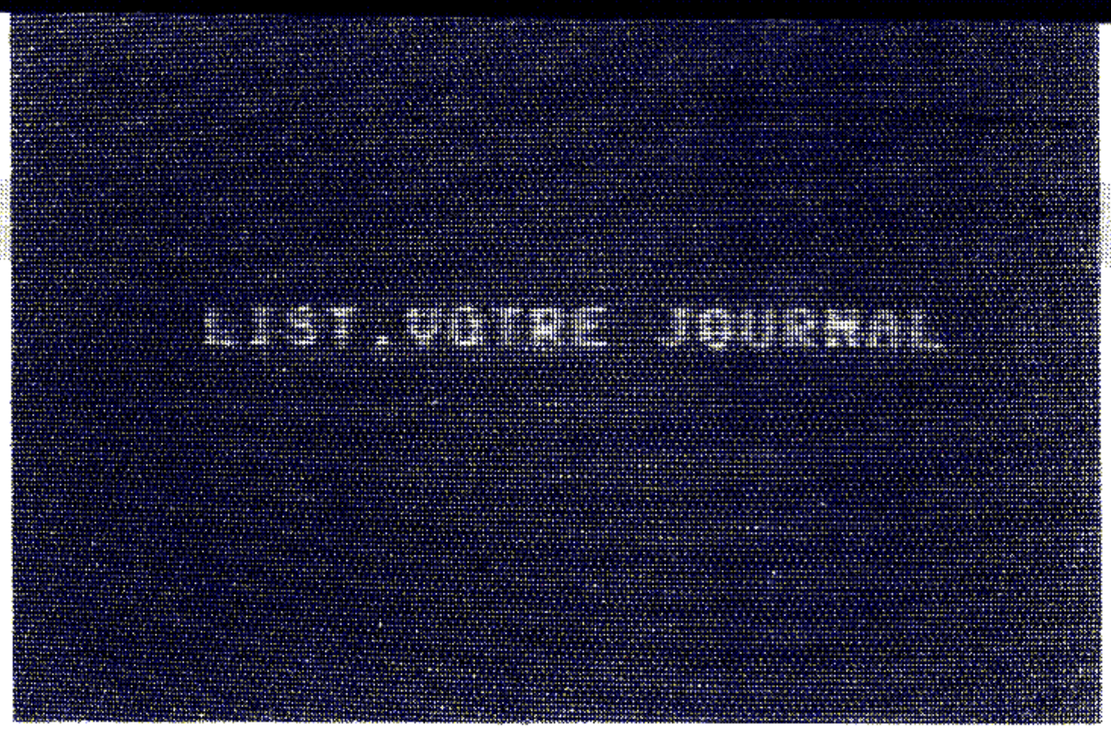
La syntaxe est d'une simplicité remarquable puisque les instructions de commande se réduisent à une seule lettre précédée de ←. Ainsi : LOAD ''titre'' devient : ←L ''titre''.

La fiabilité des enregistrements effectués à 3 600 Bauds reste excellente : nous n'avons eu aucune erreur de chargement au cours de tous nos essais, même pour de très longs programmes. Quant au gain obtenu en rapidité, il est extraordinaire, jugez-en : un programme de 23 Koctets est sauvegardé ou remis en mémoire depuis le magnétophone en une minute (y compris 10 secondes d'amorce, soit 50 secondes). A titre de comparaison, le même programme (qui occupe 90 blocs) demande également une minute pour être transféré depuis une disquette. Moralité : les

Un programme sans prétention assemblé par Arrow 64...

```
100 .O $C000 ; ADR D'IMPLANTATION
110 .C ; TABLE ETIQUETTES
120 .
130 ECRAN=$05EB ; 1ERE ADR ECRAN
140 COULR=$D9EB ; 1ERE ADR COULEUR
150 CADRE=$D020 ; COULEUR CADRE ECRAN
160 PRINT=$FFD2 ; ROUTINE PRINT
170 .
180 LDA $93
190 JSR PRINT ; EFFACE L'ECRAN
200 LDA $0E
210 JSR PRINT ; MODE MINUSCULES
220 .
230 LDX 17
240 BOUCL1 LDA MESSG,X
250 STA ECRAN,X
260 DEX
270 BPL BOUCL1 ; AFFICHE MESSAGE
280 .
290 LDY $0F
300 BOUCL2 TYA
BREAK
READY.
```

```
310 LDX 17
320 BOUCL3 STA COULR,X
330 DEX
340 BPL BOUCL3 ; CHANGE COULEURS
350 TYA
360 PHA
370 LDY $7F
380 BOUCL4 LDX $FF
390 BOUCL5 STX CADRE
400 DEX
410 BNE BOUCL5 ; TEMPORISE
420 DEY
430 BNE BOUCL4 ; ENCORE
440 PLA
450 TAY
460 DEY
470 BNE BOUCL2 ; 15 FOIS
480 .
490 RTS ; FIN
500 .
510 MESSG .B 'L','I','S','T',' ','V','O','T','R',
'E',' ','J','O','U','R','N','A','L'
520 .E
READY.
```



... et les effets graphiques s'arrêtent sur ce message.

disquettes Commodore ne sont pas aussi rapides qu'elles devraient l'être !

Lors de l'utilisation des commandes du magnétophone, vous obtenez en prime sur l'écran le nombre d'octets occupés par le programme ayant fait l'objet de la commande. Notons également, c'est important, que les programmes sauvegardés à grande vitesse avec *Arrow* ne peuvent plus être rechargés en mémoire si la cartouche n'est plus connectée.

La fonction APPEND permet de mettre bout à bout un programme présent en mémoire centrale et un programme enregistré sur cassette. Mais il faut veiller en ce cas à la numérotation du second programme, ses numéros de ligne devant être supérieurs à ceux du premier.

Une dernière commande permet de faire avancer la bande à grande vitesse jusqu'à une position prédéfinie. Voilà qui pourra s'avérer précieux pour retrouver un programme qui n'est pas en début de bande.

Éditeur-assembleur mais aussi calculatrice

Arrow offre un autre attrait, présenté à la suite par le manuel : des possibilités de calcul intéressantes. En effet, une *calculatrice* hexadécimale-décimale permet de faire, avec une grande facilité, les conversions de décimal à hexadécimal, et vice versa, avec additions et soustractions... mais malheureusement, pas de multiplications ni de divisions ! Le programme présent en mémoire lors de l'utilisation de cette calculatrice reste intact. Quant à la simplicité d'emploi, elle est parfaite.

Intéressons-nous maintenant aux quelques fonctions d'édition supplémentaires apportées par cette cartouche. Une fonction recherche une séquence en mémoire (FIND) et peut même la remplacer par une autre. Il est possible, par exemple, de rechercher et de remplacer toutes les occurences de la chaîne « TRUC » dans un programme Basic par la chaîne « MACHIN ». Le remplacement se fait ligne par ligne sous le contrôle de l'utilisateur, ce qui permet de ne l'effectuer qu'à bon escient, bien qu'en contrepartie cela le rende fastidieux...

La même fonction peut être également utile pour retrouver rapidement la variable Z$ dans un programme. Une fonction classique de numérotation automatique des lignes est disponible (AUTO). Une autre, de renumérotation (RENUM) est peu intéressante pour le Basic, car elle ne permet pas de renuméroter les adresses de saut (GOTO, GOSUB...).

Enfin, une fonction DELETE élimine facilement un programme ou un groupe de lignes indésirables.

Pour intéressantes qu'elles soient, il faut bien avouer que ces quelques fonctions ajoutées pour l'édition des programmes ne sont pas vraiment suffisantes. En Basic, une fonction pas à pas (STEP), une fonction TRACE et un vrai RENUM auraient été les bienvenus. Mais ne perdons pas de vue qu'*Arrow* est avant tout un utilitaire d'assemblage, et que les fonctions d'édition disponibles sont alors suffisantes pour cette activité.

En tant qu'utilitaire d'assemblage, ce logiciel offre d'abord un petit moniteur langage-machine, qui ressemble assez au TIM des ''gros'' Commodore. On peut, grâce à ce moniteur :

- visualiser en hexadécimal à l'écran une zone mémoire, et la modifier éventuellement ;
- faire exécuter un programme en langage-machine à partir d'une adresse spécifiée ;
- afficher et modifier le contenu des registres du processeur ;
- charger et sauvegarder sur support magnétique une zone mémoire située entre des adresses spécifiées ;
- enfin, rechercher une séquence d'octets en mémoire.

Ce moniteur est quand même un peu « juste » : il lui manque, par exemple, l'affichage des codes ASCII correspondant aux octets lors de l'affichage d'une zone mémoire. Plus gênante encore est l'absence d'un petit désassembleur qui serait une aide précieuse pour la recherche des erreurs.

Le logiciel en quelques lignes

Nom : Arrow 64
Ordinateur : Commodore 64
Forme : cartouche
Édité et distribué par : Micro Application Software
Prix public : 640 FF ttc
Principale orientation : assemblage de programmes
Autres orientations : édition de texte, gestion de cassette, calculatrice hexadécimale.

Un manuel bien peu consistant

Le mode d'emploi de l'éditeur-assembleur occupe onze des seize pages du manuel d'utilisation. Et c'est là que nos craintes concernant la minceur du livret se précisent... Un éditeur-assembleur est destiné (que les familiers du langage-machine ne nous tiennent pas rigueur de ces quelques mots d'explication !) à transformer un *programme-source,* écrit sous la forme d'instructions mnémoniques, en un programme langage-machine dit *code-objet,* exécutable. Les pratiquants de la chose savent combien est fastidieux le travail d'assemblage s'il est fait manuellement, avec un papier et un crayon. Et combien il est périlleux aussi, les risques d'erreur étant énormes !

Avec un assembleur comme *Arrow,* le *code-source* est composé de la même façon qu'un programme Basic, avec des numéros de lignes, et l'utilisateur conserve toutes les facilités de l'éditeur du Commodore. L'assembleur *Arrow* utilise la syntaxe habituelle pour les mnémoniques et les opérandes, au standard *Mos Technology*. Les données et adresses peuvent être fournies sour forme de symboles, expressions ou constantes (en décimal, hexadécimal ou binaire). Il est possible d'utiliser des étiquettes et d'introduire des commentaires dans le *programme-source*. Par ailleurs, des *pseudo-codes* permettent l'assemblage

d'octets (pour les messages par exemple), la réservation d'espace en mémoire, etc.

Lorsque le *programme-source* est composé, le lancement de l'assembleur provoque l'affichage à l'écran des résultats et, bien entendu, des erreurs éventuellement commises, qu'il est facile de corriger car le *source* reste intégralement présent en mémoire. L'imprimante peut être utilisée pour l'impression des résultats de l'assemblage. Celui-ci se déroule en deux passes successives, mais très rapidement exécutées. On voit là l'avantage de la cartouche sur la disquette. Une table des étiquettes utilisées dans le programme et une table des références seront, si on le veut, éditées en fin d'assemblage.

Une seule page du manuel fournit un exemple de la syntaxe à respecter pour les modes d'adressage et les types d'opérandes. Il s'agit d'une liste traitée comme un programme, mais aucun commentaire ne l'accompagne. Aucun exemple concret contenant un vrai programme fonctionnel ne figure dans ce « livre »... Un manque de pédagogie regrettable dont souffriront sans aucun doute les débutants.

Arrow est donc un éditeur-assembleur assez performant, et simple d'emploi. Il est bien dommage qu'un manuel d'utilisation un peu trop succinct oblige les débutants à tâtonner pour faire leurs premiers pas. La mise en œuvre, une fois ces premiers pas franchis, est heureusement très facile. De quoi passer des heures passionnantes !

Robin BOIS

PC-UTIL 2 : POUR DOPER LE BASIC DU PC-1500

IL est déjà puissant, le Basic de l'ordinateur Sharp PC-1500 (ou Tandy PC-2), mais l'abondance ne nuit jamais en matière de programmation. PC-Util 2, logiciel, sur cassette édité par Pocket-Soft, est surtout dédié aux programmeurs.

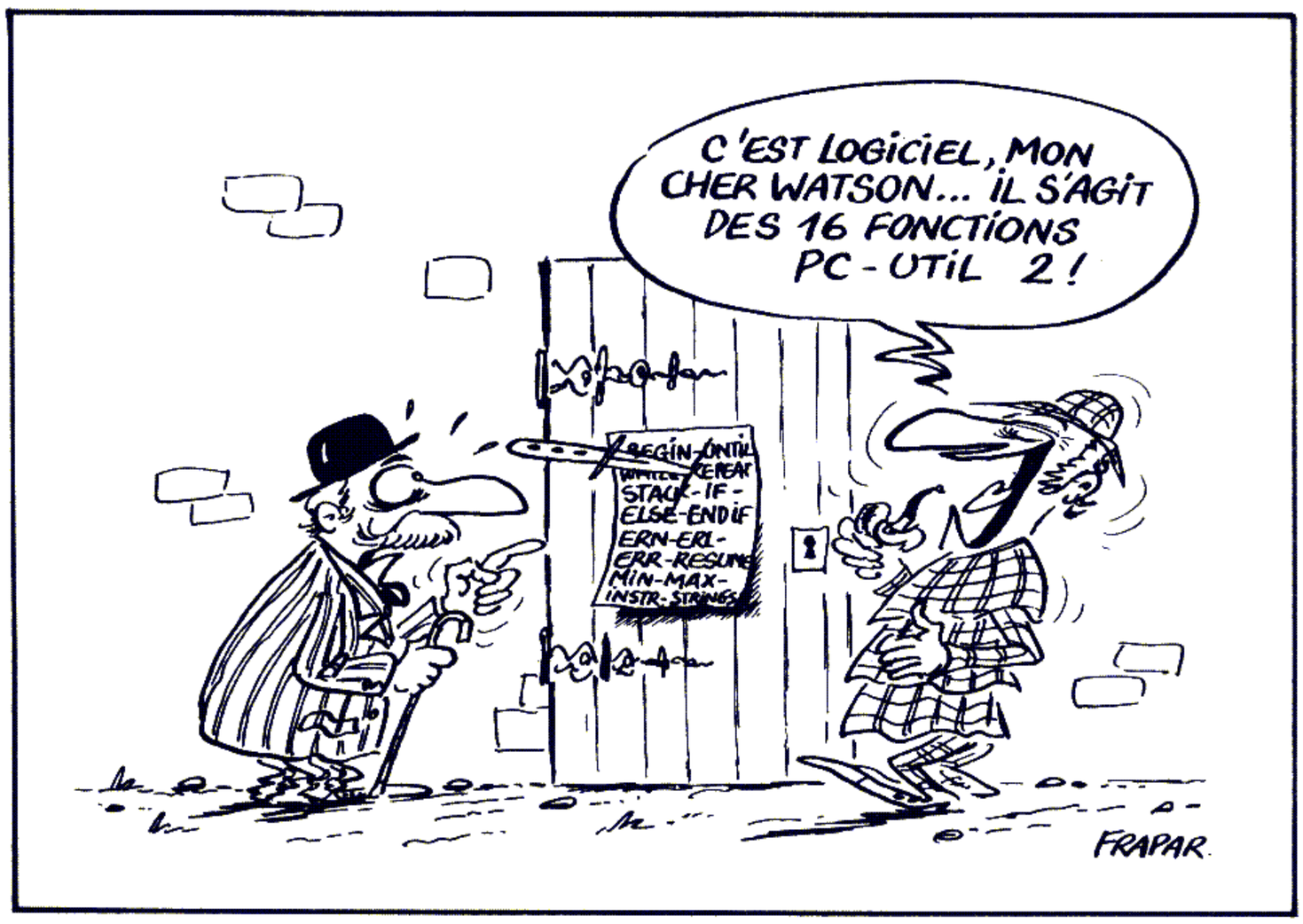
Selon que vous disposez d'un ordinateur de première ou de seconde génération (la mémoire morte diffère), la cassette d'extension du Basic ajoutera, à votre PC-1500, 16 ou 19 instructions (1). Il s'agit essentiellement d'instructions d'aide à la programmation dont certaines ne peuvent renier leur parenté avec des langages tels Pascal ou Forth beaucoup plus structurés que Basic.

Le logiciel est totalement rédigé en langage-machine et peut être employé avec toutes les configurations possibles du PC-1500. Un module de 4 Koctets de mémoire supplémentaire, ou plus, est indispensable.

En sus des nouvelles instructions, *PC-Util 2* transforme les claviers des machines de seconde génération : bip sonore à chaque pression de touche et « répétition » pour chacune d'elles. Une pression continue provoque la répétition de l'action, ce qui est bien utile avec les fonctions INS et DEL en particulier.

Les seize fonctions communes à toutes les machines sont les suivantes : BEGIN, UNTIL, WHILE, REPEAT, STACK, IF, ELSE, ENDIF, ERN, ERL, ERR, RESUME, MIN, MAX, INSTR et STRING$.

Les huit premières définissent des structures de programmation particu-



(1) Première ou seconde génération ? 16 ou 19 instructions ? Si PEEK &E2B9 donne le nombre 213 sur votre ordinateur, vous avez perdu, le vôtre est un « ancêtre » !

La logiciel en quelques lignes

Nom : PC-Util 2
Ordinateur : PC-1500
Forme : cassette
Édité par : Pocket-Soft
Distribué par : XLog
Prix public : 250 FF ttc
Nombre de fonctions nouvelles : 16 ou 19 selon la génération du PC-1500
Principale orientation : aide à la programmation
Occupation mémoire : 2 100 octets

lières illustrées dans le tableau des *instructions de programmation structurée.*

Les instructions ERN, ERL, ERR et RESUME perfectionnent la gestion des erreurs (on avait déjà le ON ERROR GOTO...). ERN donne le numéro de la dernière erreur constatée ou commise par le programme (par exemple, après une division par zéro, la variable ERN vaut 38). ERL renvoie le numéro de la ligne du programme où cette erreur s'est produite tandis que ERR force une erreur : ERR 38 fait croire au PC-1500 qu'il vient d'effectuer une division par zéro (arrêt du programme et affichage de ERROR 38 ON... ou branchement à un sous-programme si on a programmé ON ERROR GOTO...). Enfin, RESUME permet de retourner à l'exécution normale des instructions du programme qui suivent celle qui a provoqué l'erreur.

Par exemple, un ON ERROR GOTO 100 prépare le programme à se rendre à la ligne 100 à la moindre erreur ; faire A = 1/0, ou faire ERR 38 qui lui est strictement identique, provoque ce branchement en 100 où ERN donne le numéro de l'erreur (ici 38) et ERL le numéro de la ligne Basic de l'erreur. On peut programmer alors une routine de ce genre :

```
100 : IF ERN=38 PRINT"DIVISION PAR ZERO EN LIGNE "; ERL
101 : RESUME
```

Voilà un moyen d'expliciter — en français — tous les messages d'erreur. La dernière instruction, RESUME, renvoie à la suite de l'instruction fautive, ici 1/0 ou ERR 38.

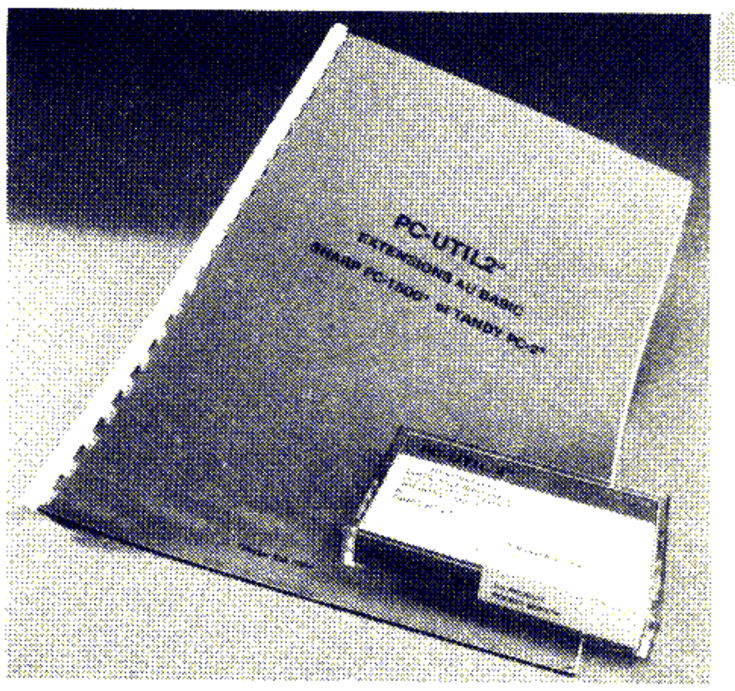
INSTR et STRING$ sont utiles aux manipulations de chaînes de caractères. La première trouve l'occurence d'une sous-chaîne dans une chaîne principale : trouver par exemple dans A$ qui contient « BONJOUR MONSIEUR », la sous-chaîne « MON » qui est dans B$, s'écrit INSTR (A$, B$, 1) et retourne le nombre 9 représentant la position de « MON » dans la chaîne principale. Le troisième paramètre, 1 dans l'exemple, est le numéro du caractère de départ de la recherche dans la chaîne, il peut varier de 1 à 80.

STRING$ (n, caractère ou code) renvoie n fois le même caractère ou code ASCII. C'est tout de même plus pratique que de l'écrire n fois ; STRING$ (70, "*") donne 70 fois le caractère "*".

MIN et MAX, comme leur nom l'indique immédiatement, trouvent entre deux nombres ou chaînes de caractères le plus petit et le plus grand.

Deux Koctets bien employés

Les ordinateurs de seconde génération disposent en outre de trois nouvelles fonctions : AUTO, PASS et UNPASS. PASS donne un mot de passe à un programme en empêchant son listage, et UNPASS supprime cette protection. Précisons immédiatement qu'elle ne résiste pas longtemps aux assauts des connaisseurs de la structure intime de l'ordinateur... Enfin, AUTO est réellement utile aux programmeurs puisqu'elle réalise la numérotation automatique des lignes d'un programme. Il n'est alors plus nécessaire de taper ces numéros à chaque nouvelle ligne de programme, ils le sont automatiquement après l'exécution de AUTO n° de départ, incrément.



La cassette et sa notice d'emploi

Toutes ces instructions nouvelles s'emploient normalement dans les programmes tant que *PC-Util 2* est en mémoire. Cela bloque donc 2 100 octets environ en permanence, mais ils sont bien employés. Si, avec les ordinateurs de première génération, il suffit d'écrire l'instruction en toutes lettres dans le programme, ce n'est pas possible avec les autres : les instructions sont affectées à seize touches particulières redéfinissables de « réserve » et sont par leur intermédiaire introduites dans le programme Basic.

Une mention « très bien » doit être attribuée aux fonctions de programmation structurée (voir le tableau) mais aussi à AUTO.

En conclusion, *PC-Util 2* est un bon utilitaire d'aide à la programmation du PC-1500.

Jean-Christophe KRUST

Instructions de programmation structurée

DANS la ligne qui suit, « action » est une suite quelconque d'instructions Basic, ainsi que « suite ». La « condition » est une expression arithmétique ou logique identique à celle des tests IF classiques et dont le résultat est soit vrai (vérifié, 1) soit faux (non vérifié, 0).

BEGIN : action : UNTIL condition : suite →
↑ continuer tant que la condition n'est pas vérifiée (vraie)

BEGIN : action : WHILE condition : action : REPEAT : suite →
↓ tant que condition est vraie
→ fin, dès que la condition n'est plus vérifiée

STACK remet à zéro la pile d'adresses avant toute boucle BEGIN...

IF # condition : action si condition vraie
: ELSE# : action si condition fausse
: ENDIF : suite du programme →

UNE PROCÉDURE PASCAL

FILTRER LES ENTRÉES

UNE petite erreur lors de l'entrée d'une donnée et le programme ne tourne plus rond. D'où l'intérêt de mettre en place une procédure qui vérifie la validité d'un nombre entré. En voici une, écrite en Pascal, qui ne laissera passer que les nombres autorisés.

Un nombre peut toujours être introduit sous la forme d'une chaîne de caractères. Si ce nombre correspond au format donné, on dit qu'il est *valide*. Sinon, il sera source d'erreurs. Pour éviter ces erreurs, une vérification s'impose. Elle passe, ici, par la création d'une fonction.

Cette fonction est déclarée :

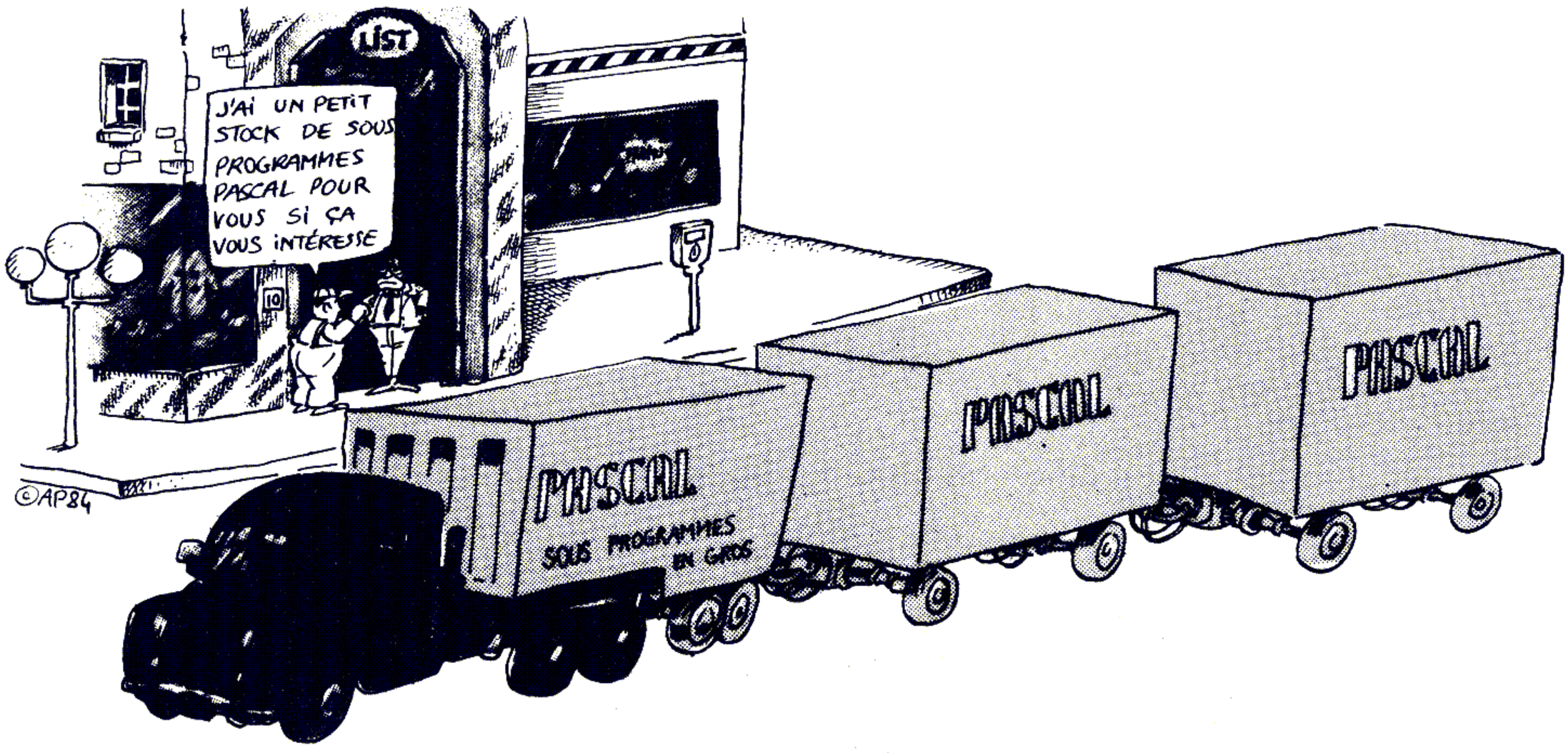
```
function nombre_valide (valeur : string ; f1, f2 : integer) : integer ;
```

La chaîne VALEUR contient le nombre dont il faut vérifier le format. Elle ne doit contenir que les signes « + » ou « − », les chiffres de 0 à 9, et un point (ou une virgule) décimal.

Les deux autres paramètres, F1 et F2, correspondent au format que doit respecter le nombre contenu dans VALEUR. F1 correspond à la longueur maximum du nombre, et F2 au nombre maximum de chiffres pouvant apparaître après la virgule.

Le premier paramètre, VALEUR, fournit donc à cette fonction la chaîne à analyser. Les deux autres, F1 et F2, permettent de spécifier le format auquel devra se trouver le nombre stocké dans VALEUR. Ce format est analogue au format des réels tel qu'il est défini en Pascal pour la procédure standard WRITE ; F1 correspond à la longueur maximale du nombre, y compris, éventuellement, le signe ainsi que la virgule ou le point décimal et F2 indique le nombre de chiffres après la virgule. Si F2 est égal à 0, le nombre attendu est un entier. Il ne doit par conséquent comporter ni virgule, ni point décimal. Dans le cas contraire, le nombre réel pourra comporter une virgule ou un point, mais ne devra pas avoir plus de F2 décimales.

Cette fonction retourne une valeur entière qui correspond à un code d'erreur. Si le nombre contenu dans



la chaîne est convenable, la fonction retourne la valeur 0. Si ce nombre n'est pas *valide,* la valeur retournée est un code d'erreur qui indique la raison pour laquelle ce nombre ne convient pas. Le tableau suivant fournit la liste de ces codes avec leur signification.

Valeur de nombre__valide	**Libellé de l'erreur**
1	Pas de nombre
2	Deux virgules ou points décimaux
3	Le signe doit être au début du nombre
4	Caractère illégal dans le nombre
5	Nombre trop grand
6	Nombre comportant trop de décimales

Trois exemples typiques permettent d'illustrer, dans des cas très différents, l'intérêt de cette fonction.

Pour une petite faute de frappe

Le premier, qui vient immédiatement à l'esprit, concerne les saisies numériques entrées au clavier sous forme de chaînes. La simple conversion d'une chaîne entrée en nombre, sans contrôle de validité, n'est pas satisfaisante : elle risque de ne pas interpréter l'erreur de la même façon que l'utilisateur. L'exemple le plus courant est celui où la lettre O a été saisie à la place du chiffre 0. L'affichage à l'écran semble correct, alors que ce caractère peut soit ne pas être pris en compte par la routine de transformation de la chaîne en nombre, soit être considéré comme un symbole indiquant la fin du nombre. Dans ce cas, la valeur affichée à l'écran est différente de la valeur mémorisée par l'ordinateur (à moins que l'on ne prenne soin de réafficher toutes les zones saisies après traitement par l'ordinateur). La fonction créée ici permet d'éviter une telle confusion.

Le second exemple est celui où les données saisies au clavier sont cruciales pour la suite du programme. Dans ce cas, il convient de ne tolérer aucune erreur de la part de l'utilisateur. Par exemple, la moindre faute dans l'entrée de la réponse à la question « quel est le numéro du fichier à détruire ? » doit être soigneusement signalée, même si l'on effectue des sauvegardes régulières des fichiers...

Le troisième exemple correspond à ce que l'on appelle le traitement différé ou traitement en « batch ». Prenons le cas d'un fichier qui a été saisi à l'aide d'un éditeur de textes, sans le moindre contrôle de validité des informations entrées. Ces données sont destinées à être prises en compte ultérieurement par un programme de liaison (une comptabilité, par exemple). Dans cette situation, il ne faut en aucun cas interpréter des données erronées en essayant de les corriger, car il n'est pas sûr que « l'intuition » du programme s'avère correcte à tous les coups. De plus, il peut non seulement s'agir d'une faute de frappe, mais également d'une erreur dans le programme qui ne lit pas correcte-

Fonction de vérification des entrées
Programme en Pascal
Auteur Thierry Chamoret
Copyright *LIST* et l'auteur

```
function nombre__valide (valeur : string ; f1, f2 : integer) : integer ;
var     i          : integer ;
        avanvir ,
        aprevir    : integer ;
        virgule    : boolean ;
        numerr     : integer ;
begin
  if length (valeur) <=0
    then
     numerr : = 1
    else
     begin
        numerr : = 0 ;
        avanvir : = 0 ;
        aprevir : = 0 ;
        virgule : = false ;
        i : = 0 ;
        repeat
          i : = i + 1 ;
          if valeur [i] in ['0'..'9']
            then
              if virgule
                then aprevir : = aprevir + 1
                else avanvir : = avanvir + 1
            else
              if valeur [i] in [',','.']
                then
                  if virgule
                    then numerr : = 2
                    else virgule : = true
                else
                  if valeur [i] in ['+', '-']
                    then
                      begin
                        if i > 1 then numerr : = 3 ;
                        avanvir : = avanvir + 1
                      end
                    else
                      numerr : = 4
      until (i >   = length (valeur)) or (numerr <> 0) ;
      if numerr = 0
              then
                begin
                  if (f1 > 0) and (f1 < avanvir + aprevir + ord (virgule)) then
                  numerr : = 5 ;
                  if (f2 > 0) and (f2 < aprevir) then numerr : = 6
                end
    end ;
  nombre__valide : = numerr
end ;
```

UNE PROCÉDURE PASCAL

ment la suite de caractères. Dans tous ces cas, un contrôle strict des données s'avère indispensable.

La mise en œuvre de la fonction de vérification est assez simple. Elle peut s'insérer de la façon suivante dans un programme :

```
case  nombre_valide (entrée, 6,2)
 0 :   valider_entrée (entrée) ;
 1 :    traiter_erreur ('Vous n'avez pas entré de valeur') ;
 2 :    traiter_erreur ('Ce nombre contient deux virgules décimales') ;
 3 :    traiter_erreur ('Un signe apparaît au milieu du nombre') ;
 4 :    traiter_erreur ('Caractère non numérique présent dans le nombre') ;
 5 :    traiter_erreur ('Cette valeur est trop grande') ;
 6 :    traiter_erreur ('Cette valeur comporte trop de décimales')
end ;
```

Pour déterminer si le nombre fourni dans la chaîne VALEUR rentre bien dans le format défini par F1 et F2, notre fonction utilise tout d'abord un indicateur VIRGULE qui signale si une virgule (ou un point décimal) a été rencontrée. De plus, deux compteurs AVANVIR et APREVIR mémorisent respectivement le nombre de chiffres avant et après la virgule.

Si la chaîne passée a une longueur nulle ou défectueuse, le code erreur 1 est retourné, indiquant qu'aucun nombre n'a été entré. Autrement, tous les caractères sont analysés jusqu'à la fin de la chaîne ou jusqu'à ce qu'une erreur soit détectée.

L'analyse des caractères s'effectue de la façon suivante :

- la reconnaissance d'un chiffre entraîne l'incrémentation de l'un des deux compteurs AVANVIR ou APREVIR, selon que l'on se situe dans la partie entière ou dans la partie décimale du nombre ;
- la rencontre d'une virgule ou d'un point décimal provoque, si l'indicateur VIRGULE est vrai, une erreur de code 2. En effet, il s'agit du cas où deux virgules ou points décimaux apparaissent dans le nombre. La variable logique VIRGULE est ensuite positionnée à la valeur VRAI, ce qui permet de savoir, pour la suite du traitement, que la partie décimale du nombre va être analysée ;
- si un signe est détecté, un test sur sa position dans la chaîne doit être réalisé. S'il n'apparaît pas comme premier caractère, l'erreur de code 3 sera alors signalée, un signe ne devant pas apparaître au milieu d'un nombre ;
- enfin, le caractère qui n'a été traité dans aucun des cas précédents entraîne une erreur 4, signalant qu'un caractère illégal est présent dans le nombre.

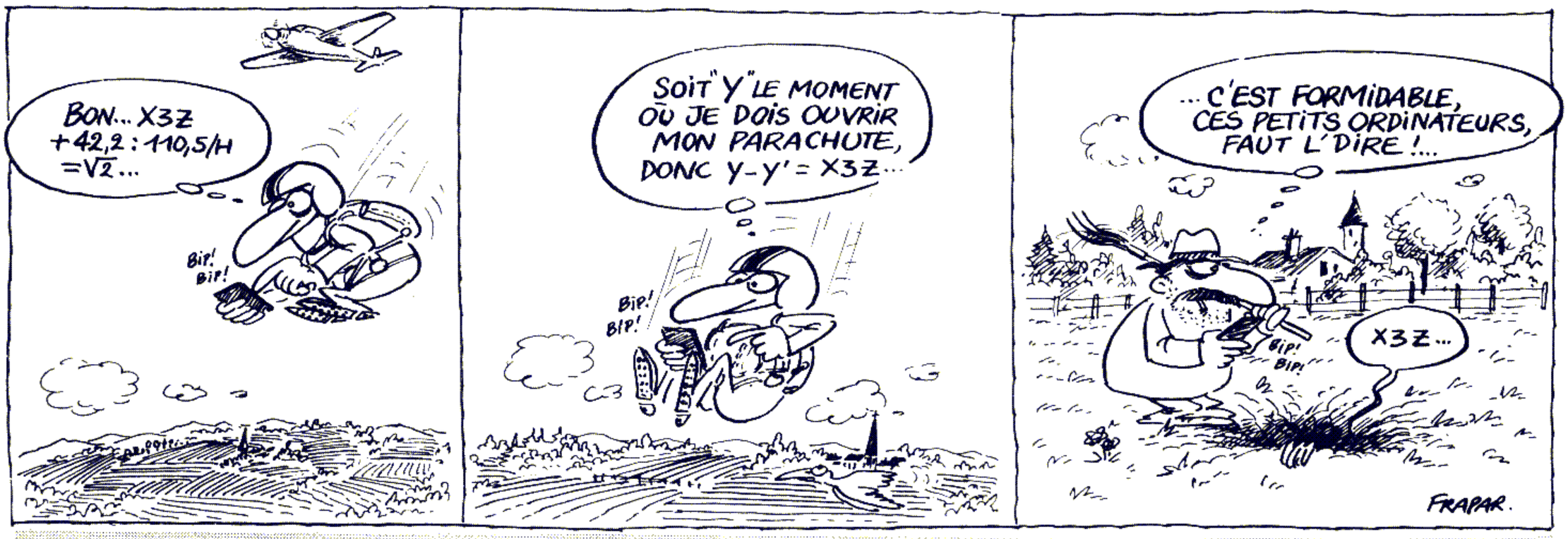
Alors seulement il convient de vérifier que le nombre ne dépasse pas la longueur maximale indiquée par F1. Elle est égale au nombre de chiffres avant (AVANVIR) et après (APREVIR) la virgule, augmenté de 1 si une virgule était présente dans le nombre. La fonction ORD, appliquée à une variable booléenne, retourne la valeur 1 si elle est vraie, et 0 si elle est fausse. Nous utilisons cette propriété pour incrémenter, sans avoir besoin de faire un test logique, la longueur totale du nombre entré. S'il dépasse la longueur fixée par le format, le code d'erreur retourné est 5.

La bonne note ici, c'est zéro

La vérification de la partie décimale du nombre est ensuite effectuée. Si le nombre comporte trop de décimales, c'est le code d'erreur 6 qui apparaît.

Il ne reste plus, alors, qu'à souhaiter que la seule valeur retournée (ou du moins la plus fréquente) soit le 0, signe que le nombre introduit est valide...

Thierry CHAMORET



AU CENTRE DE LA HP-41 C

LA PROGRAMMATION SYNTHÉTIQUE : ONZE REGISTRES POUR TOUT FAIRE

UNE « bogue », en jargon informatique,
est une erreur de programmation.
Les programmeurs de chez HP ont mis grand soin
à faire de la HP-41 C une machine sophistiquée, évolutive, sérieuse...
Mais la *bogue est là, heureuse, énorme et subtile.*
Loin de diminuer l'intérêt du calculateur, elle en multiplie les attraits.
La porte s'ouvre grande sur un monde merveilleux où tout est possible.

Dans notre premier article (*LIST* n° 1), nous avons entrouvert la porte sur de nouvelles méthodes de programmation de la HP-41 C.

Sans se jeter immédiatement dans l'analyse théorique du processus de création d'instructions synthétiques, on en a énoncé la logique. Parmi les onze nouveaux registres (M, N, O, P, Q, ⊢ , a, b, c, d et e), les sept derniers nous sont encore inconnus : leur rôle est fondamental. Aussi, sans attendre, regardons-les de plus près.

Les registres d'état

Le tableau (page suivante) illustre, pour chaque registre d'état, composé de sept octets, son ou ses rôle(s) dans la gestion par la HP-41 C de sa propre mémoire. Leur manipulation est rendue possible par l'emploi de fonctions synthétiques telles RCL, STO, X < >, etc. Mais, pour les employer on doit connaître les codages des données dans les registres d'état.

Les trois registres M, N, O et trois octets du registre P ont été examinés en détail la dernière fois. Ils forment tout simplement, ensemble, le grand registre alpha dont la capacité est de 24 caractères (3 * 7 + 3).

L'autre partie du registre P, possède un emploi alternatif. Ses contenus varient en fonction des besoins de la HP-41 C. On y trouve parfois des lettres alphabétiques du registre alpha (lettres 25 à 28 a priori considérées comme perdues), ou des informations diverses lors d'opérations spécifiques comme les CATALOGs ou l'introduction de chiffres au clavier.

Le registre Q est aussi un tampon de stockage temporaire de données. En l'occurrence, il s'agit de messages alphabétiques. Par exemple, lors de l'exécution de XEQ ᵀ ABCDEF (même si aucun label de ce nom n'existe), on trouve dans Q les lettres FEDCBA qui ne sont autres que celles du nom du label, mais inversées. Ces lettres sont codées en ASCII (codes hexadécimaux 41 à 5A pour les lettres A à Z) ce qui donne dans le registre Q le nombre : 0,64 54 44 34 E - 59

Les codes 41 à 46 des lettres ABCDEF y sont pourtant tous, mais inversés (doublement) 64 pour 46, 54 pour 45, etc. Le code 42 est caché sous le signe E-, et le code 41 est bien exprimé par 59 car 59 = 100 - 41 et c'est ainsi que sont codés les exposants. Ce nombre est bien « pathologique », on en examinera les dessous dans le détail ultérieurement.

Le registre ⊢ possède une partie temporaire et code aussi les assignations des touches primaires (non '' shiftées ''). En fait, les assignations sont codées plus loin en mémoire, et le registre ⊢ se contente de noter si une touche est *assignée ou non.* Comme on a 36 touches, la HP-41 C utilise 36 bits du registre ⊢ (soit 4 octets et demi). A chaque bit correspond une touche ; un bit est strictement la même chose qu'un drapeau : s'il est armé, la touche possède une assignation (le contenu est recherché dans la table), sinon il n'y a pas d'assignation et la fonction exécutée est la fonction standard.

De même, le registre e possède 36 bits chargés de noter l'état d'assigna-

tion des touches secondaires (" shiftées").

Les registres a et b sont en totalité réservés pour stocker des adresses de la mémoire. Essentiellement, celles des sous-programmes employés (adresse où l'exécution doit retourner à la rencontre de RTN).

Quatre octets sont nécessaires pour coder une adresse car la mémoire de la HP-41 C est découpée en registres de sept octets chacun. Toute adresse de la mémoire peut donc être définie avec quatre octets : trois pour le numéro du registre et un pour le numéro d'octet. La table des registres d'état de la figure ci-contre occupe les adresses 005 à 00F (registres 6 à 16 de la mémoire), les cinq précédents (000 à 004) codant par ailleurs les registres T, Z, Y, X et L de la pile opérationnelle.

Les registres d'état

Adresse	Registre	n° d'octet 6	5	4	3	2	1	0
00F	e	assignations des touches secondaires					temp.	n° de ligne
00E	d	registre des flags						
00D	c	ΣREG	temp.		constante		ad. R00	ad. END
00C	b	SSP. n° 3	adr. SSP n°2		adr. SSP n° 1		pointeur d'adr.	
00B	a	adr. SSP n° 6		adr. SSP n° 5		adr. SSP n° 4		adr...
00A	⊦	assignations des touches primaires					temp.	
009	Q	temp. alphabétique						
008	P	temp.					alpha : lettres 22 à 24	
007	O	registre alpha : lettres 15 à 21						
006	N	registre alpha : lettres 8 à 14						
005	M	registre alpha : lettres 1 à 7						

Sources : *L'Ordinateur Individuel* n° 24 à 28 et 31, et *Synthetic programming on the HP-41 C* (en français aux Éditions du Cagire).

Attention aux trous de mémoire

Les quatre derniers octets du registre b codent l'adresse du pointeur de programme : celle de l'instruction que l'on est en train d'exécuter. En jouant avec cette adresse, rien n'empêche d'exécuter des programmes, par exemple, dans le module X-FUNCTIONS ou, même, dans les registres R00 à RNN (à suivre...).

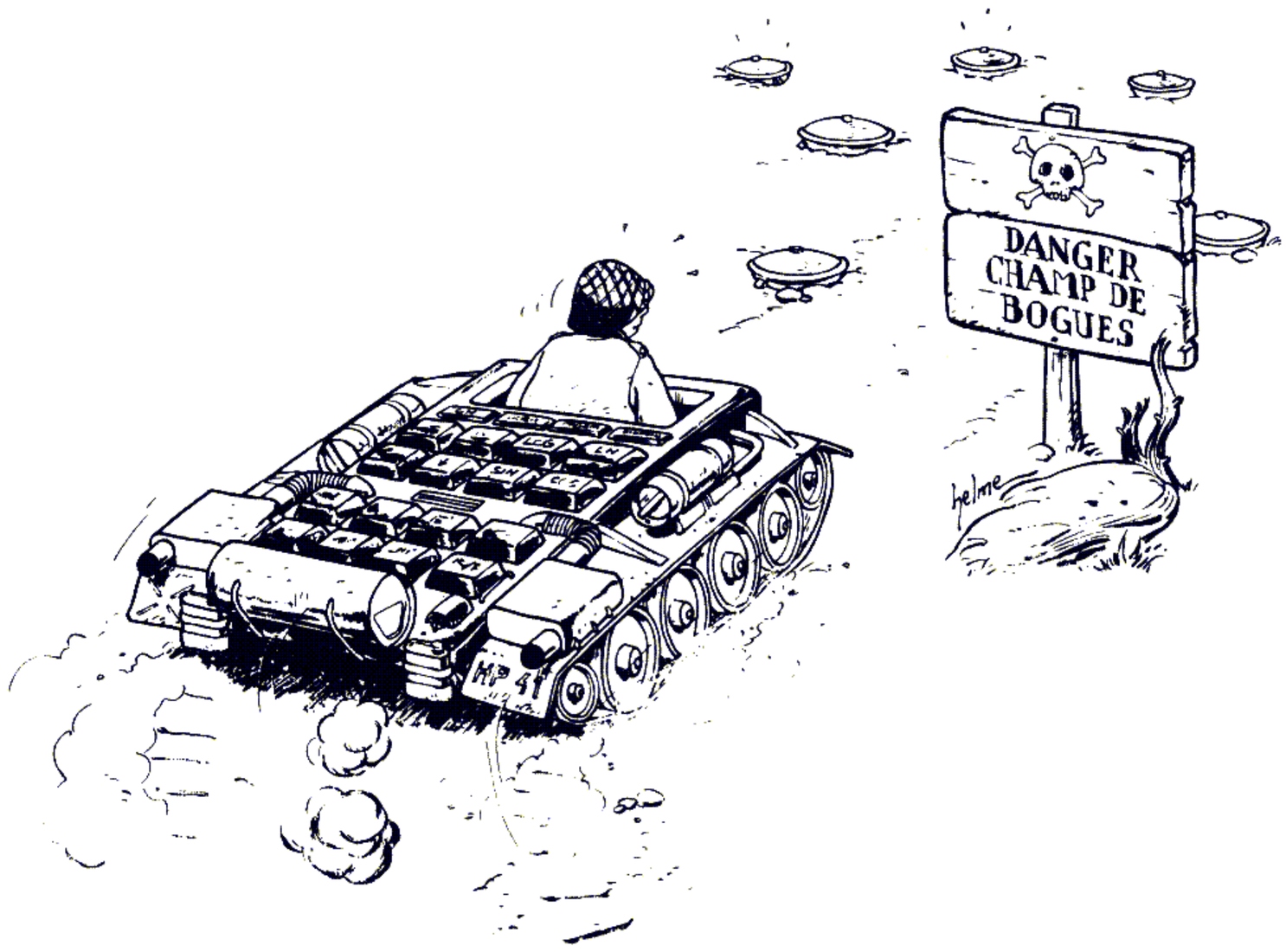
Un dernier registre, c, est plus délicat d'emploi : les "memory lost" sont légion si l'on n'y prend garde. C'est à cause de la constante codée sur 12 bits en son milieu. Cette constante, 169, ne doit pas être modifiée ! Périodiquement, la HP-41 C vérifie qu'elle s'y trouve et, si par malheur ce n'est pas le cas, provoque purement et simplement un "memory lost".

ΣREG est l'adresse (sur 12 bits) du premier registre statistique. R00 est l'adresse du registre 00 et END celle du END final.

Enfin, les 12 premiers bits du registre e codent le numéro de la ligne du programme sur laquelle se trouve le pointeur.

Quand on connaît la carte et les instructions synthétiques appropriées, un peu d'imagination permet de programmer des applications bien surprenantes. A vous de jouer maintenant.

Jean-Christophe KRUST



QUE LE GRAND CRIC ME CROQUE !

Pour assigner le Cric - voir *LIST* n° 1 - à une touche du clavier, il faut une HP-41 C nue de toute extension. Seul le module TIME est autorisé.

N'est-ce pas trop demander aux possesseurs de HP-41 CX qui possède en standard le module X-FUNCTIONS ? A moins de jouer du scalpel, on ne peut l'ôter !

Une solution existe pourtant : après le *memory lost*, exécuter un SIZE 272. Ensuite, tout fonctionne à merveille.

Arnaud PERUTA

A LA POINTE DU CRAYON

UNE fonction du Basic du MO5, DEFGR$, permet de définir des caractères originaux. Son utilisation n'est pas vraiment simple. Pour la remplacer, c'est tout un programme.

Création de caractères
Programme pour MO5
Auteur Eric Tière


```
10 TUNE:CLEAR,,128
20 DIM TA(7,7),G(7)
30 V=1:L=16
40 '**** DESSIN DE LA GRILLE ****
45 SCREEN 4,6,6:CLS:PRINT CHR$(20)
50 A=0:B=0:C=0:D=8*L
60 LINE (A,B)-(C,D)
70 LINE (D,C)-(B,A)
80 A=A+L:C=C+L:IF A<=8*L THEN 60
82 LOCATE 25,8:PRINT"Dessin";V
85 '**** POSITION ET REMPLISSAGE D'UNE C
ASE ****
90 INPUTPEN X,Y
94 IF X<0 THEN 90 ELSE IF X>=8*L THEN 15
0
95 IF Y<0 THEN 90 ELSE IF Y>=8*L THEN 15
0
96 PLAY "L1DO"
100 X%=INT(X/L):Y%=INT(Y/L):X=X%*L:Y=Y%*
L
110 BOXF (X,Y)-(X+L,Y+L)
120 TA(X%,Y%)=1
121 G(Y%)=0:FOR I=0 TO 7:G(Y%)=2*G(Y%)+T
A(I,Y%):NEXT I
122 DEFGR$(V)=G(0),G(1),G(2),G(3),G(4),G
(5),G(6),G(7)
123 LOCATE 25,10:PRINT GR$(V)
124 FOR I=0 TO 7:LOCATE 17,2*I+1:PRINT G
(I):NEXT I
140 GOTO 90
145 '**** DESSIN FINI ****
150 PLAY "L24DOREMI"
160 LOCATE 0,18:INPUT "Memorisation (O/N
) ";O$
170 IF O$="O" THEN V=V+1
320 PRINT:INPUT "Autre dessin (O/N) ";O$
330 IF O$<>"O" THEN 410
335 '**** RAZ DES VARIABLES: TABLEAU ET
CARACTERE ****
340 FOR I=0 TO 7
350 FOR J=0 TO 7
360 TA(I,J)=0
370 SCREEN,,J
380 NEXT J,I
390 FOR I=0 TO 7:G(I)=0:NEXT I
400 GOTO 40
410 '**** FINI ****
420 CLS:PRINT "On recapitule"
430 FOR I=1 TO V-1
440 PRINT I;": ";CHR$(128+I)
450 NEXT I
460 PRINT:END
```

Le Basic du MO5 permet à l'utilisateur de définir ses propres caractères grâce à la fonction DEFGR$, et c'est intéressant.

Intéressant, certes. Mais si, comme nous, vous ne voyez pas directement que la définition d'un "as de pique" est DEFGR$(1) = 8,28,62,127,127, 28,62, alors cette fonction devient inutile. En revanche, le programme proposé ici la remplace tout en la simplifiant : plus besoin de feuille quadrillée, ni de calculette !

Le fonctionnement en est fort simple. Après le réglage du crayon optique, une grande grille de huit cases sur huit s'affiche. Elle représente la matrice d'un caractère. Pointez alors les cases que vous voulez noircir et les coordonnées apparaissent au fur et à mesure. Lorsque vous jugerez avoir fini la composition de votre caractère, pointez à l'extérieur de la grille.

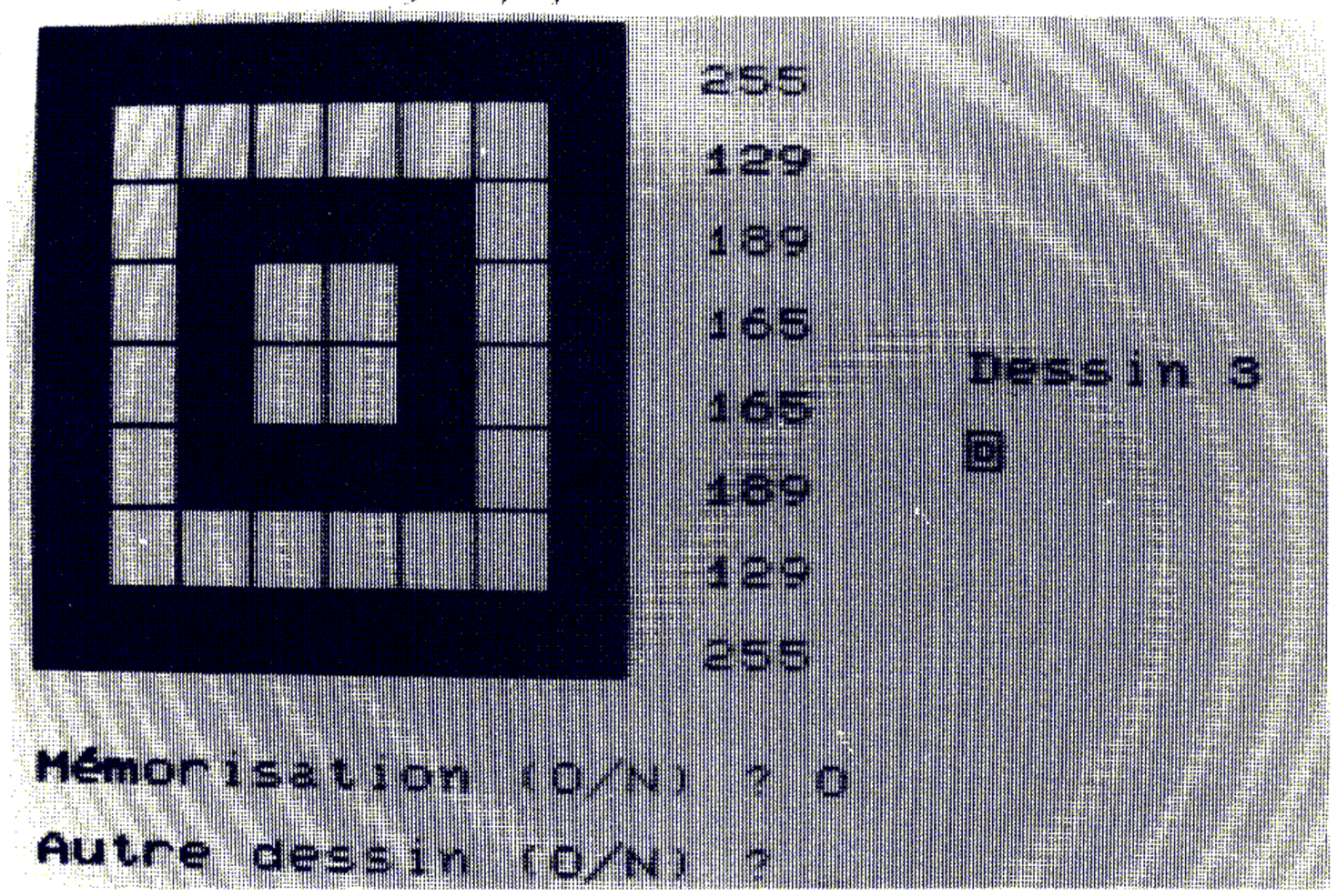
Le programme vous demande alors si vous désirez mémoriser votre œuvre : elle apparaît en taille normale et ses poids dans DEFGR$ sont automatiquement calculés et affichés.

Le programme boucle autant de fois que vous le désirez. Les caractères qui auront été mémorisés seront ensuite directement utilisables par GR$(i) ou CHR$(128+i).

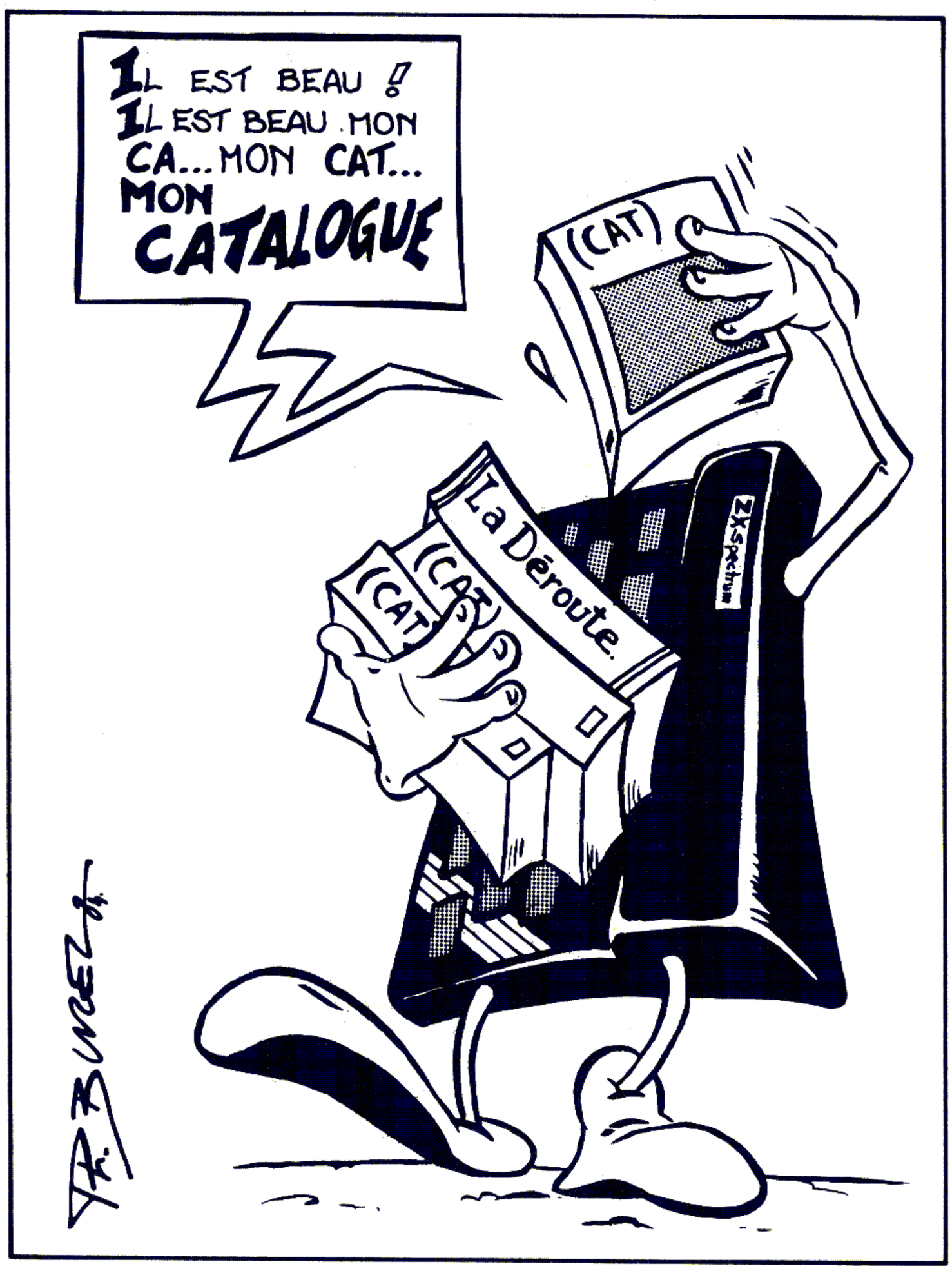
On peut imaginer l'amélioration de ce programme par l'ajout de nouvelles "commodités". Pourquoi, par exemple, ne pas sauvegarder sur cassette les caractères dont vous êtes fiers et gérer ensuite ce fichier ? On peut aussi prévoir une gomme qui efface les cases noircies par erreur. Le programme serait un peu plus compliqué, mais l'utilisation en serait tellement simplifiée...

Éric TIÈRE

Un nouveau caractère créé à la pointe du crayon optique



DEMANDEZ LE CATALOGUE



CHACUNE des cartouches destinées au Spectrum contient davantage d'information qu'elle n'en délivre normalement. Comment aller les repêcher ?

Avec le programme que nous vous proposons vous allez pouvoir étendre la fonction de CATALOGUE (CAT) normalement présente en relisant les préambules, stockés sur bande, concernant les programmes ou les blocs de données. Effectivement la fonction CAT, sous la direction de l'interface ZX 1, permet seulement de relire les noms de programmes ou de blocs de données enregistrés sur la cartouche. En prime le nombre de Koctets restant disponibles est affiché ou imprimé. Néanmoins les autres informations précisant si l'on a affaire à un programme, à un tableau numérique, à un tableau de caractères ou à un bloc d'octets sont absentes.

A moins d'être méticuleux et d'avoir le temps de noter sur un cahier le contenu précis de chaque cartouche, il vous arrivera rapidement de ne plus savoir à quoi correspond tel ou tel nom. A fortiori s'il s'agit d'un bloc d'octets, comment retrou-

ver son origine d'implantation et sa longueur ?

Tout possesseur de microdrives a reçu au moins une fois le compte rendu d'erreur "WRONG FILE TYPE" lors de la tentative de chargement d'un programme, parce que le nom correspondait à un bloc d'octets et non pas à un programme.

Le nouveau CATalogue

Cependant toutes ces précieuses informations sont écrites sur la cartouche. Un préambule ou HEADER en anglais est enregistré au début de chaque sauvegarde (SAVE*"m";1; "nom" pour un programme, suivi ou non de LINE, CODE, SCREEN$, ou DATA). Ce préambule de 9 octets se décompose ainsi :

• l'octet 1 définit le type de sauvegarde : 0 pour un programme, 1 pour un tableau numérique, 2 pour un tableau de caractères, 3 pour un bloc d'octets (on notera que ce sont les mêmes conventions dans les sauvegardes magnétocassettes) ;

• les octets 2 et 3 représentent la longueur en octets de la sauvegarde ;

• les octets 4 et 5 représentent l'adresse de l'origine de la sauvegarde ;

• les octets suivants concernent le type 0, c-à-d les programmes Basic.

C'est ainsi que les octets 6 et 7 indiquent la longueur du programme seul (les octets 4 et 5 représentaient alors la longueur du programme et des variables).

Quant aux octets 8 et 9, ils représentent la ligne Basic où doit s'autolancer le programme après chargement.

L'utilitaire qui fait l'objet de cet article se compose de deux parties : une interface Basic et une routine en langage-machine longue de 60 octets. Pour ce qui est de la routine en langage-machine, vous avez deux possibilités pour l'implanter . On peut la placer en DATA dans le programme principal dont on remplace la première ligne par 1 FOR n=32000 TO 32059 : READ a : POKE n,a : NEXT n. Dans ce cas, on écrira les DATAs en ligne 180, par exemple.

L'autre possibilité est de sauvegarder le code sous la forme d'un enregistrement sur cartouche. Pour créer cet enregistrement, on utilisera le programme suivant :

```
1 FOR n=32000 TO 32059 : READ a : POKE n,a : NEXT n
2 SAVE*"m";1;"michead" CODE 32000,60
3 VERIFY*"m";1;"michead" CODE
4 DATA ...,...,...,...,...
```

Dans les deux cas, les DATAs sont les codes décimaux présentés à la suite du désassemblage de la routine.

Le programme Basic

Avant d'appeler la routine, on "pokera" le nom à rechercher sur la cartouche à une adresse où il pourra être retrouvé par celle-ci. On procédera de même pour la longueur du nom.

Après l'appel de la routine, les 9 octets du préambule correspondant à l'enregistrement ont été copiés dans

Programme de décodage du préambule

```
  1 CLEAR 30000: LOAD *"m";1;"michead" CODE 32000,60
  5
 10 REM  Decodage du HEADER
 20 REM  ++++++++++++++++++++
 30
 40 PRINT  AT 10,2;"Decodage du preambule (HEADER)"; AT 12,10;"pour soit :"; AT
14,12;"programme BASIC"; TAB 12;"tableau de variables"; TAB 12;"bloc d'octets"
 50
 60 INPUT "nom sur la cartouche : ";a$
 65 LPRINT
 70 LET a= LEN a$: POKE 31989,a: FOR n=1 TO a: POKE 31989+n, CODE a$(n): NEXT n
 80 RANDOMIZE  USR 32000
 90 LET org=31980: LET type= PEEK org
100 LPRINT ("Programme " AND type=0)+("Tableau numerique " AND type=1)+("Tablea
u de caracteres " AND type=2)+("Bloc d'octets " AND type=3);"<";a$;">"
110 LPRINT "     Longueur en octets "+("Programme + Variables" AND  NOT type)+(
"Code" AND type);" : "; PEEK (org+1)+256* PEEK (org+2)
120 LPRINT "     Origine du bloc : "; PEEK (org+3)+256* PEEK (org+4)
130 IF type THEN  GO TO 60
140 LPRINT "     Longueur du programme sans les variables : "; PEEK (org+5)+256
* PEEK (org+6)
150 LET a= PEEK (org+7)+ PEEK (org+8)
160 IF a <> 65535 THEN  LPRINT "     Ligne d'autolancement : ";a
170 GO TO 60
```

Programme source de la routine de décodage

```
  10 *H     HEADER
  20
  30          ORG   32000
  40
  50 NOM      EQU   31990
  60 DSTR1    EQU   23766
  70 NSTR1    EQU   23770
  80 CURCHL   EQU   23633
  90 HEADER   EQU   31980
 100 LONG     EQU   31989
 110
 120 ;Creation des variables systeme
 130          RST   8
 140          DEFB  #31
C150
 160 ;Sauvegarde de l'adresse de retour au Basic
 170          EXX
 180          PUSH  HL
 190          EXX
 200
 210 ;Selection du microdrive 1
 220          LD    A,1
 230          LD    (DSTR1),A
 240
 250 ;Longueur du nom du bloc
 260          LD    A,(LONG)
 270          LD    L,A
C280          XOR   A
 290          LD    H,A
 300          LD    (NSTR1),HL
 310          LD    HL,NOM
 320          LD    (NSTR1+2),HL
 330
 340 ;Ouverture du fichier
 350          RST   8
 360          DEFB  #22
 370          PUSH  IX
 380          POP   HL
 390          LD    (CURCHL),HL
 400          LD    B,9
 410          LD    HL,HEADER
 420
C430 ;Lecture des 9 octets du HEADER
 440 BOUCLE   PUSH  BC
 450          PUSH  HL
 460          CALL  #15E6
 470          POP   HL
 480          LD    (HL),A
 490          INC   HL
 500          POP   BC
 510          DJNZ  BOUCLE
 520
 530 ;Fermeture du fichier
 540          RST   8
 550          DEFB  #23
 560          LD    A,2
C570          CALL  #1601
 580          EXX
 590          POP   HL
 600          EXX
 610          RET
 620
 630 ;...fin....
```

les adresses allant de 31980 à 31988. Les lignes Basic à partir de 90 entreprennent le décodage de ces octets en fonction des critères cités précédemment.

La routine en langage-machine

L'assemblage de la routine a été réalisé avec l'assembleur GENS2 de Hisoft. Concernant les mnémoniques, les nombres précédés d'un dièse représentent des nombres hexadécimaux, les autres correspondent à des nombres décimaux.

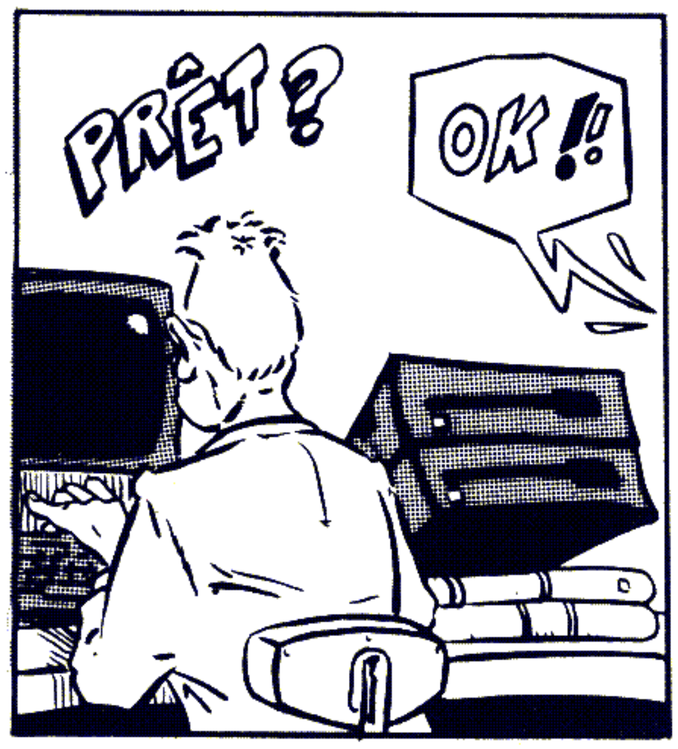
Les pseudo-instructions d'assemblage, au début de la liste, initialisent les labels des différents pointeurs :

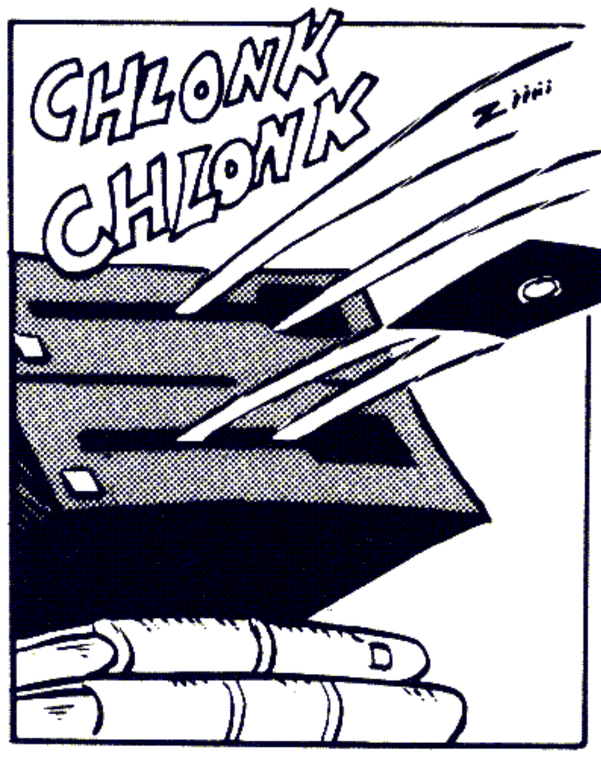
- NOM pointe sur le début du nom poké par le Basic ;
- LONG est la longueur du nom ;
- HEADER pointe sur le début des 9 octets de préambule lus sur la bande.

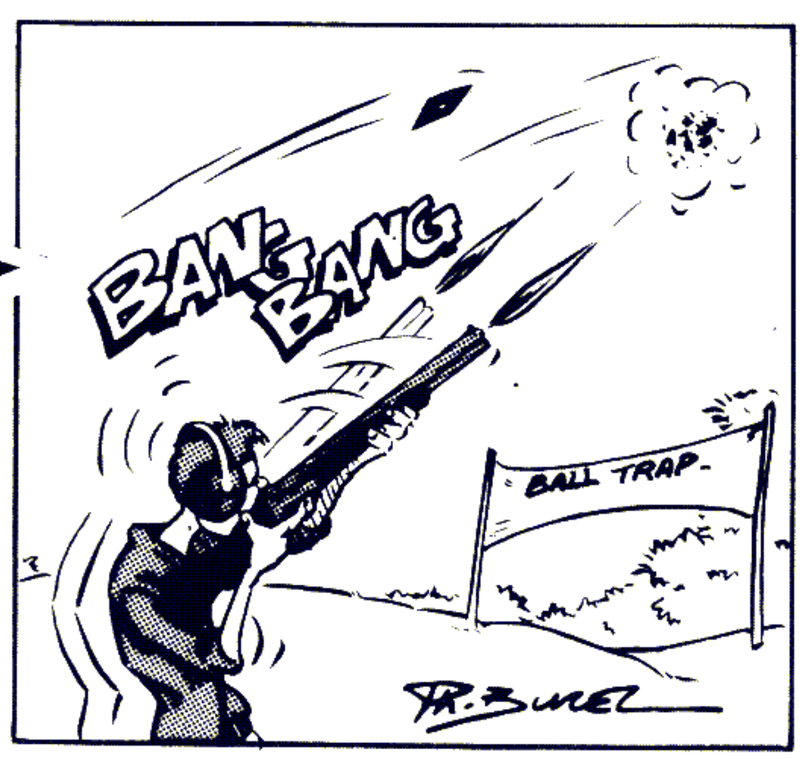
Les variables système utilisées sont :

- CURCHL, pointeur des canaux ;
- DSTR1, numéro d'unité de microdrive ;
- NSTR1, longueur du nom du fichier.

Ouvrons une parenthèse pour décrire comment s'effectue l'appel de routines présentes dans la mémoire morte de l'interface ZX 1. Quand la mémoire morte du Spectrum est sélectionnée, la mémoire morte de l'interface ZX 1 est hors-circuit et réciproquement. (La ligne ROMCS du bus de sortie du Spectrum est utilisée pour cette opération.)







Le programme source une fois assemblé

```
Pass 1 errors: 00
    HEADER

                    10 *H    H
                       EADER
                    20
7D00                30
                       ORG          32000
                    40
7CF6                50 NOM
                       EQU          31990
5CD6                60 DSTR1
                       EQU          23766
5CDA                70 NSTR1
                       EQU          23770
5C51                80 CURCHL
                       EQU          23633
7CEC                90 HEADER
                       EQU          31980
7CF5               100 LONG
                       EQU          31989
                   110
                   120 ;Creati
                       on des varia
                       bles systeme

7D00 CF            130
                       RST          8
7D01 31            140
                       DEFB         #31
                   150
                   160 ;Sauveg
                       arde de l'ad
                       resse de ret
                       our au Basic

7D02 D9            170
                       EXX
7D03 E5            180
                       PUSH         HL
7D04 D9            190
                       EXX
                   200
                   210 ;Select
                       ion du micro
                       drive 1
7D05 3E01          220
                       LD           A,1
7D07 32D65C        230
                       LD           (DSTR1
                       ),A
                   240
                   250 ;Longue
                       ur du nom du
                        bloc
7D0A 3AF57C        260
                       LD           A,(LON
                       G)
7D0D 6F            270
                       LD           L,A
7D0E AF            280
                       XOR          A
7D0F 67            290
                       LD           H,A
7D10 22DA5C        300
                       LD           (NSTR1
                       ),HL
7D13 21F67C        310
                       LD           HL,NOM

7D16 22DC5C        320
                       LD           (NSTR1
                       +2),HL
                   330
                   340 ;Ouvert
                       ure du fichi
                       er
7D19 CF            350
                       RST          8

7D1A 22            360
                       DEFB         #22
7D1B DDE5          370
                       PUSH         IX
7D1D E1            380
                       POP          HL
7D1E 22515C        390
                       LD           (CURCH
                       L),HL
7D21 0609          400
                       LD           B,9
7D23 21EC7C        410
                       LD           HL,HEA
                       DER
                   420
                   430 ;Lectur
                       e des 9 octe
                       ts du HEADER

7D26 C5            440 BOUCLE
                       PUSH         BC
7D27 E5            450
                       PUSH         HL
7D28 CDE615        460
                       CALL         #15E6
7D2B E1            470
                       POP          HL
7D2C 77            480
                       LD           (HL),A

7D2D 23            490
                       INC          HL
7D2E C1            500
                       POP          BC

7D2F 10F5          510
                       DJNZ         BOUCLE
                   520
                   530 ;Fermet
                       ure du fichi
                       er
7D31 CF            540
                       RST          8
7D32 23            550
                       DEFB         #23
7D33 3E02          560
                       LD           A,2
7D35 CD0116        570
                       CALL         #1601
7D38 D9            580
                       EXX
7D39 E1            590
                       POP          HL
7D3A D9            600
                       EXX
7D3B C9            610
                       RET
                   620
                   630 ;...fin
                       ....

Pass 2 errors: 00

Table used:     97  from    203
```

Codes hexadécimaux de la routine

CF 31 D9 E5 D9 3E 01 32 D6 5C 3A F5
7C 6F AF 67 22 DA 5C 21 F6 7C 22 DC
5C CF 22 DD E5 E1 22 51 5C 06 09 21 EC
7C C5 E5 CD E6 15 E1 77 23 C1 10 F5 CF
23 3E 02 CD 01 16 D9 E1 D9 C9

Codes décimaux de la routine

207 49 217 229 217 62   1 50 214 92
 58 245 124 111 175 103 34 218 92 33
246 124 34 220 92 207 34 221 229 225
 34 81 92   6   9 33 236 124 197 229
205 230 21 225 119 35 193 16 245 207
 35 62   2 205   1 22 217 225 217 201

LES CARTOUCHES DU SPECTRUM

DEMANDEZ LE CATALOGUE !

L'instruction RST 8 sert sur le Spectrum de base à la gestion des comptes rendus d'erreur dont la liste est reproduite en page 215 du manuel de programmation. La valeur du DEFB, qui suit l'instruction RST 8, incrémentée d'une unité donnera le compte rendu correspondant. Les DEFB du Spectrum de base vont de 0 à 26 (0 à 1A en hexadécimal).

Si des valeurs comprises entre 27 et 50 (1B et 32 en hexadécimal) sont mises dans les DEFB suivant l'instruction RST 8, la mémoire morte de l'interface ZX 1 est sélectionnée et la routine correspondant au code est appelée. Il en est de même pour le DEFB 255 (FFh) qui donne le compte rendu 'Program finished'.

Les valeurs des DEFB comprises entre 51 et 254 (33 et FE en hexadécimal) ne correspondent à rien et renvoient le compte rendu 'Hook code error'. *Hook code* n'est pas facile à traduire ; disons que les Hook codes accrochent la mémoire morte de l'interface ZX 1. Pour la suite, nous continuerons de les appeler Hook codes, puisque c'est ainsi qu'ils sont dénommés dans les comptes rendus.

Lors de l'appel d'une routine contenue dans la mémoire morte de l'interface, les registres ne sont pas sauvegardés et doivent donc être mis sur la pile avant l'appel, pour être récupérés au retour. Notamment le registre H'L' doit être préservé, en général, car il contient l'adresse de retour au Basic.

Exemple d'utilisation

```
Bloc d'octets <chess>
     Longueur en octets Code : 36540
     Origine du bloc : 27001

Bloc d'octets <chess2>
     Longueur en octets Code : 360
     Origine du bloc : 32501

Programme <echec>
     Longueur en octets Programme + Variables : 169
     Origine du bloc : 23813
     Longueur du programme sans les variables : 169
     Ligne d'autolancement : 10

Programme <run>
     Longueur en octets Programme + Variables : 253
     Origine du bloc : 23813
     Longueur du programme sans les variables : 253
     Ligne d'autolancement : 11

Programme <turc>
     Longueur en octets Programme + Variables : 151
     Origine du bloc : 23813
     Longueur du programme sans les variables : 151
     Ligne d'autolancement : 10

Bloc d'octets <turc1>
     Longueur en octets Code : 6912
     Origine du bloc : 16384

Bloc d'octets <turc2>
     Longueur en octets Code : 14885
     Origine du bloc : 50651

Programme <turc3>
     Longueur en octets Programme + Variables : 25857
     Origine du bloc : 23813
     Longueur du programme sans les variables : 24827
     Ligne d'autolancement : 225
```

Dans notre routine de décodage du préambule nous utilisons trois de ces Hook codes.

Le premier correspond au DEFB 31h. Ce Hook code sert à créer les nouvelles variables système indispensables à la gestion de l'interface ZX 1. Donc, avant de se servir des lecteurs de cartouche par l'intermédiaire du langage-machine, cette instruction permet de s'assurer que les nouvelles variables système ont bien été créées.

Le fichier dont le nom a été introduit par le programme Basic de décodage est ensuite ouvert grâce au Hook code 22h, équivalent à la fonction Open File. L'ouverture du fichier sert à rechercher le premier enregistrement physique correspondant au nom donné. Ce premier enregistrement est lu puis copié dans la mémoire-tampon du canal. Les 9 octets du préambule peuvent alors être copiés à l'emplacement que nous avons choisi (adresse 31980), avant la fermeture du fichier.

Cette fermeture est obtenue grâce au code 23h, puis le canal de sortie est ouvert à nouveau sur le téléviseur. L'adresse de retour au Basic est alors restaurée.

Les résultats

Vous pouvez voir ci-contre un exemple de relecture d'une cartouche. Le programme Basic dirige les sorties vers l'imprimante, mais vous pouvez tout aussi bien diriger celles-ci vers le téléviseur en changeant chaque instruction LPRINT par PRINT. Vous pouvez de même les enregistrer sur cartouche par un PRINT dièse.

Vous avez maintenant les principes de base du fonctionnement des microdrives (1). Essayez de réaliser des programmes qui les utilisent et — pourquoi pas ? — proposez-les nous.

Benoît THONNART

(1) Pour en savoir plus, vous pouvez consulter utilement l'ouvrage du Dr Ian Logan, Spectrum microdrive book, *publié par* Melbourne House Publishers Ltd, Church Yard, Tring, Hertfordshire HP 23 5LU.

GESTION DES ERREURS

DES messages d'erreur en français, simuler les fonctions ERN, ERL et RESUME... C'est possible, en Basic, avec le programme ERR.

Le Basic du PC-1500 est incomplet en ce qui concerne la gestion des erreurs qui peuvent intervenir lors de l'exécution d'un programme. Dans la plupart des cas, on obtient un message du genre de celui-ci :

ERROR 38 IN 100

C'est un peu laconique, non ? L'erreur 38, après consultation du manuel, s'avère être une division par zéro. Donc, en ligne 100 du programme cette opération incorrecte est réalisée.

Mais le programme est stoppé. Dans certains cas, des erreurs font partie du domaine du possible et le programme doit pouvoir les traiter sans pour autant être stoppé définitivement lorsque l'erreur survient.

C'est le rôle de l'instruction "ON ERROR GOTO..." qui provoque un branchement à la ligne "..." dès qu'une erreur se produit dans l'exécution du programme. Ainsi ces lignes :

```
 1 : ON ERROR GOTO "ERR"
10 : A = A/0
...
1000 : "ERR" PRINT "ERREUR"...
```

prépare l'exécution de la routine de gestion d'erreur "ERR" (ligne 1) ; provoque une erreur 38 (ligne 10) ; et affiche le message "ERREUR".

On peut maintenant programmer, après la ligne 1000, un sous-programme de gestion d'erreur. Deux données nous manquent : le numéro de l'erreur (ERN) et le numéro de la ligne (ERL) où elle s'est produite.

L'encadré ci-dessous donne des expressions Basic qui calculent ces valeurs.

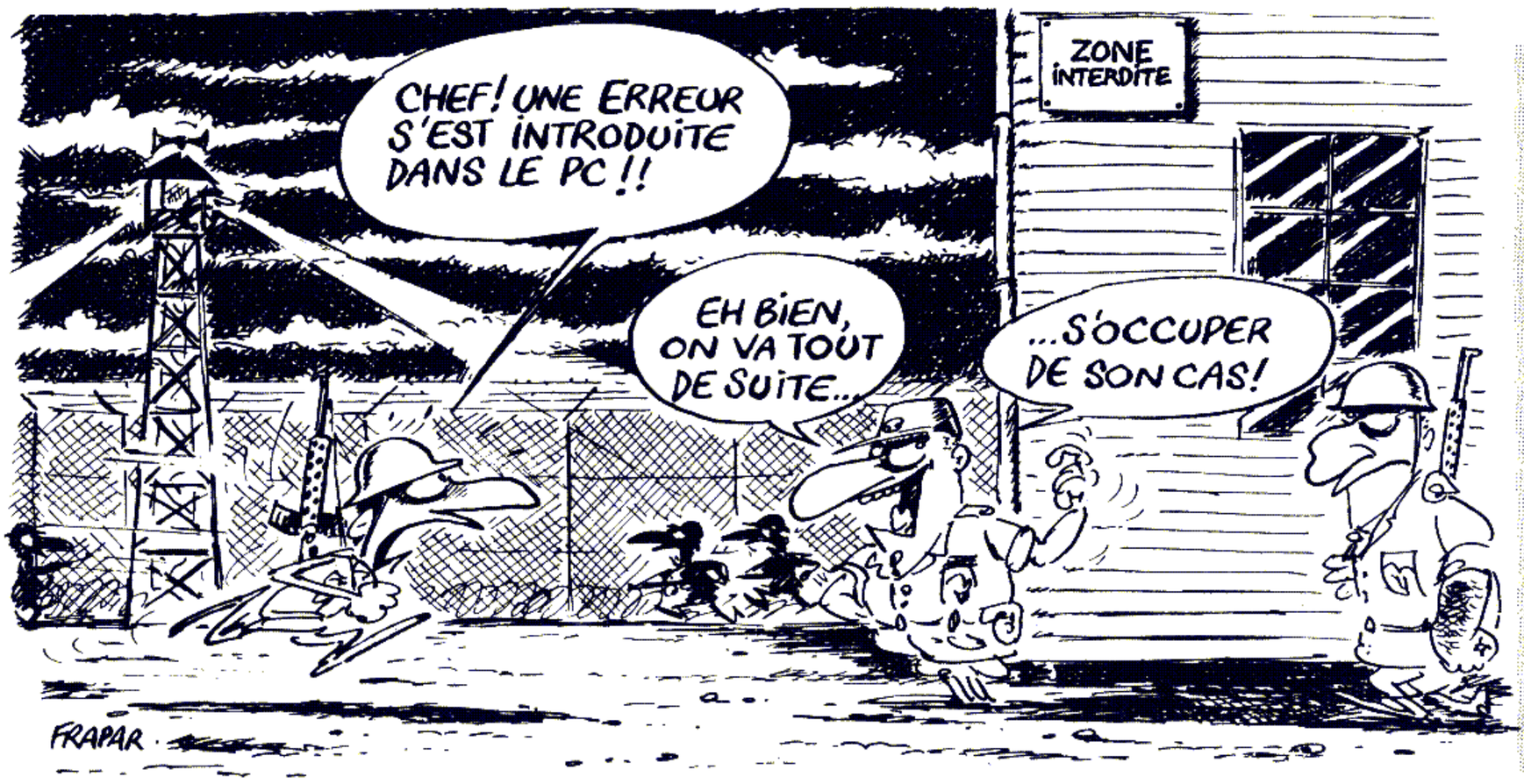
Les données en Basic

N° de l'erreur	PEEK &789B
N° de ligne	PEEK &78B4*256 + PEEK&78B5
Adresse de l'erreur	PEEK &78B2*256 + PEEK&78B3

Et l'on y trouve l'essentiel de notre routine de gestion des erreurs. Si on connaît le numéro, rien de plus simple que d'afficher des messages d'erreur en français, par exemple :

```
IF PEEK &789B = 38 PRINT "DIVISION PAR 0"
```

Quant au numéro de la ligne, on préfèrera l'expression STATUS 4 qui, exécutée en toute première instruction de la ligne "ERR" (voir le programme), retourne le numéro de la ligne contenant la dernière instruction exécutée, donc, celle qui contient l'erreur.



UN UTILITAIRE POUR LE PC-1500

```
 1:ON ERROR GOTO
   "ERR



1000:"ERR"A=
     STATUS 4:
     BEEP 1:PAUSE
     "ERREUR";
     PEEK &789B;"
      EN";A
1001:B=PEEK &789B
1002:IF B=0PAUSE
     "DONNEE INCO
     RRECTE"
1003:IF B=2PAUSE
     "NEXT INCORR
     ECT"
1004:IF B=4PAUSE
     "READ SANS P
     LUS DE DONNE
     ES"
1005:IF B=6PAUSE
     "VARIABLE NO
     N DECLAREE"
1006:IF B=7PAUSE
     "VARIABLE DE
      TYPE INCORR
     ECT
1007:IF B=38PAUSE
     "DIVISION PA
     R ZERO"
1008:REM AUTANT D
     E MESSAGES Q
     UE SOUHAITE
1010:RESTORE A:A=
     (PEEK &78BE-
     128)*256+
     PEEK &78BF-1
     :A=A+PEEK A+
     1:A=PEEK A+
     PEEK (A+1):
     GOTO A
```

Nous pouvons analyser le programme "ERR", reproduit ci-contre. Entre la ligne n° 1 et la ligne 1000 on peut programmer n'importe quoi. Par exemple, afin de tester le programme, les lignes :

10 : A = A/0 : REM ERREUR
25 : BEEP 5 : END

Lors de l'exécution, la ligne 10 provoque l'erreur. Le programme "ERR" va successivement afficher :

ERREUR 38 EN 10
DIVISION PAR ZERO

avant de renvoyer l'exécution, *automatiquement, à la ligne qui suit immédiatement* celle de l'erreur, ici la ligne 25.

En 1000, on saisit (STATUS 4) le numéro de la ligne fautive et, en consultant aussi la variable système &789B, on affiche le premier message d'erreur. Puis, de 1001 à 1008, six types d'erreurs sont analysés. Ce choix est arbitraire, on peut très bien traduire en français tous les messages d'erreur en fonction de leur numéro. Ceci conduit à l'affichage du second message.

Enfin, la fonction Basic RESUME — absente dans le PC-1500 et qui fait se poursuivre l'exécution à la ligne suivant l'erreur — est réalisée à la ligne 1010.

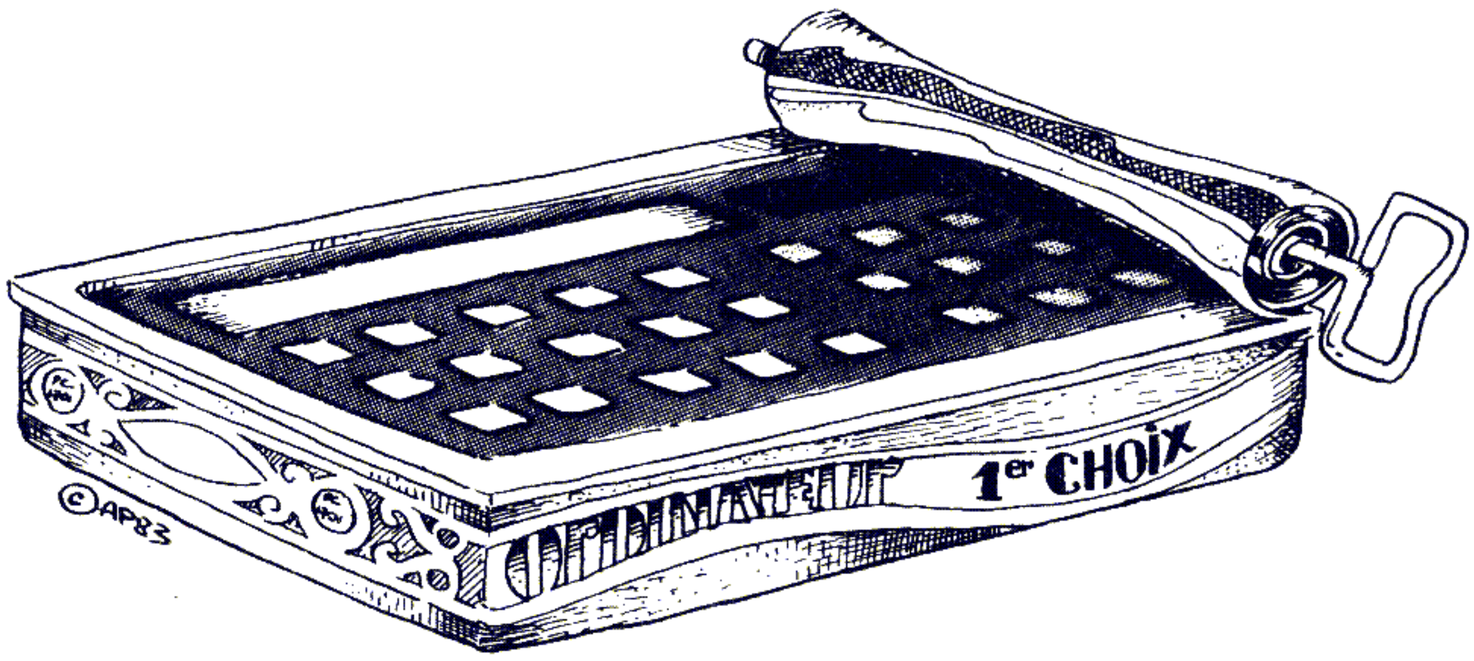
Une variable système (&78BE-BF), celle du pointeur de RESTORE est consultée après un RESTORE. Elle retourne l'adresse du code de longueur de la ligne où l'erreur s'est produite.

Connaissant cette adresse et la longueur de la ligne, on peut, à l'aide d'une simple addition, trouver l'adresse du numéro de la ligne suivant celle où l'erreur s'est produite.

Il ne reste alors plus qu'à calculer ce numéro de ligne dans A avant d'y diriger l'exécution d'un simple GOTO A.

Nous savons donc maintenant déterminer complètement un processus de gestion des erreurs du PC-1500, les applications sont — outre la simple francisation des messages d'erreur — nombreuses et trouveront sans peine leur place dans vos programmes.

Jean-Christophe **KRUST**



PETITS COURS ET TRAVAUX DIRIGÉS

EN Logo, rien ne distingue les données des programmes : une procédure peut être définie et traitée comme une variable. En particulier, être créée ou modifiée par une autre. Une procédure peut donc évoluer en fonction d'événements externes. On la dit alors "intelligente", par analogie avec l'homme qui s'éduque.

En Logo, une procédure est introduite par le mot POUR suivi d'un nom et de l'ensemble éventuel des paramètres formels, chaque paramètre étant précédé par deux-points (:). Par exemple :

```
POUR DOUBLER :A
```

La procédure se termine par le mot FIN. Ainsi définie, elle ne peut pas être modifiée par une autre. Seul le programmeur y aura accès, grâce à l'éditeur de texte dont les commandes sont interactives. Pour écrire des procédures modifiables, il suffit de leur donner une structure d'objet Logo, c'est-à-dire de les transformer en listes.

Un premier problème se présente : transformer en liste une procédure déjà définie avec POUR. On utilise alors la primitive TEXTE — ou DEFINITION — en donnant un nom à la liste créée.

```
POUR TRAIT :N
AVANCE :N
FIN
DONNE "T TEXTE :TRAIT
AFFICHE :T
[[ N ][ AVANCE :N ]]
```

Ce programme appelle quelques remarques :

- la liste T créée est une liste de listes ; "T est son nom ; son contenu est le texte de la procédure TRAIT et c'est pourquoi nous écrirons DONNE "T TEXTE :TRAIT et non "TRAIT ;
- la première liste est celle des paramètres, elle est obligatoire ; en leur absence, elle est vide mais existe ; les noms des paramètres ne sont plus précédés d'aucun caractère ;
- chaque instruction est en elle-même une liste ;
- le mot FIN n'apparaît plus ;
- il n'est pas interdit d'écrire DONNE "TRAIT TEXTE :TRAIT.

La liste T est maintenant un objet Logo, manipulable par les primitives PREMIER, SAUFPREMIER, DERNIER, SAUFDERNIER, MOT, PHRASE, LISTE, INSEREPREMIER, INSEREDERNIER. Écrivons par exemple une procédure pour modifier T en lui ajoutant un paramètre et une instruction.

```
POUR MODIFIER
DONNE "T (LISTE [ N D ] DERNIER :T [ DROITE :D ])
FIN
AFFICHE :T
[[ N D ][ AVANCE :N ] [ DROITE :D ]]
EDITE T
POUR T :N :D
AVANCE :N
DROITE :D
FIN
```

Un autre problème réside dans l'écriture d'une procédure sous forme de liste. On utilisera pour cela la primitive DEFINIS (à condition de respecter les remarques faites ci-dessus, avec la primitive TEXTE). On aurait pu écrire directement DEFINIS "T [[N][AVANCE :N]]

Dans ce cas, il faut faire attention : le nom de la procédure est toujours précédé d'un guillemet (") alors qu'il ne l'était pas avec POUR.

RAPPEL DE LA SYNTAXE LOGO

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

En Logo, un nombre est écrit tel quel, éventuellement précédé d'un signe. Un mot est toujours précédé de guillemets (on ne referme pas les guillemets à la fin du mot). Une liste est encadrée de crochets carrés []. Les noms de variables, qui ne sont pas liés à leur contenu, sont des mots. Le contenu de la variable "A est :A.

Quant à transformer une procédure écrite avec DEFINIS en une procédure écrite avec POUR, cela semble bien inutile : les procédures ont le même statut vis-à-vis de l'éditeur, quel que soit leur mode de définition. A l'édition, celle qui est écrite avec DEFINIS apparaît sur l'écran avec les mots POUR et FIN, et se corrige de la même façon ; à la fin, elle reprend sa structure de liste. Il n'est donc pas nécessaire de définir une primitive pour transformer la forme DEFINIS en la forme POUR. D'ailleurs, en mémoire, les procédures sont implantées comme des listes de listes.

Créons un mini-langage

Par exemple, la modification d'une procédure en fonction du hasard se présente comme suit :

```
POUR ITINERAIRE
SI HASARD 2 = 0 ALORS DEFINIS "P [[ ][ AV 10 DR 5 ]] SINON DEFINIS "P [[ ][ RE 10 GA 5 ]]
SI LISCAR = "A ALORS EXECUTE :P SINON AFFICHE [ TAPER A SVP ]
ITINERAIRE
FIN
```

L'utilisateur doit taper sur la lettre A, sinon un message s'affiche. Lorsqu'il frappe A, le tirage au hasard donne soit 0 et dans ce cas la procédure P est définie comme AV 10 DR 5, soit donne 1 et la procédure P est RE 10 GA 5. A la frappe, P est exécutée (remarquer le deux-points qui le précède) puis redéfinie en fonction d'un nouveau tirage au sort. La procédure ITINÉRAIRE est récursive (récursion terminale). Il faut remarquer aussi la liste vide obligatoire des paramètres de P.

Logo travaille selon deux modes : le mode PILOTAGE dans lequel les instructions sont exécutées directement mais ne sont pas mémorisées, et le mode PROCEDURAL qui mémorise les instructions sans les exécuter. Nous pouvons définir un troisième mode qui exécute directement les instructions, et les mémorise en vue d'une éventuelle définition de procédure à la fin. Pour ce faire, il convient d'établir un "mini-langage" : APPRENDS introduit le mode, FIN annonce la fin des instructions, ANNULE détruit la dernière instruction exécutée.

Bien évidemment, ce type d'écriture de procédure ne permet pas d'introduire des paramètres. Mais des procédures correspondantes existent :

- "PROC est le nom provisoire de la liste des instructions qui constitueront éventuellement le corps d'une procédure ;
- LISLISTE permet d'entrer une ligne au clavier et de lui donner un nom si nécessaire ; cette ligne a une structure de liste ; elle doit être terminée par un "retour-chariot" ;
- "REP est le nom donné à la liste entrée au clavier ;
- "NOM est le nom donné à la procédure créée.

Ce qui permet d'établir la liste suivante :

```
POUR APPRENDS
VIDEECRAN
DONNE "PROC [[ ]]
PILOTE
AFFICHE [ VOULEZ-VOUS CONSERVER VOTRE TRAVAIL ? ]
AFFICHE [ SI OUI, TAPEZ SON NOM, PUIS "RETOUR" ]
AFFICHE [ SI NON, TAPEZ "RETOUR" ]
DONNE "NOM PREMIER LIS LISTE
SI NON :NOM = "DEFINIS :NOM :PROC
FIN
POUR PILOTE
DONNE "REP LISLISTE
SI :REP = [ FIN ] STOP
SI :REP = [ ANNULE ] ALORS ANNULER SINON EXECUTE :REP DONNE "PROC INSEREDERNIER :REP :PROC
PILOTE
FIN
```

LA PROCÉDURE DESSIN

```
POUR DESSIN
AV 2 DR 5
SI CAP = 0 ALORS STOP
DESSIN
AV 10 GA 5
FIN
```

Supposons la tortue au centre, CAP au NORD et observons les effets de chaque ligne de la procédure :

AV 2 DR 5 : le CAP prend la valeur 5.

SI CAP = 0 ALORS STOP : le CAP n'étant pas égal à 0, on continue.

DESSIN: on abandonne la procédure initiale qui n'est pas terminée puisqu'il reste à faire AV 10 GA 5 (qui appellera la procédure DESSIN considérée comme une autre procédure).

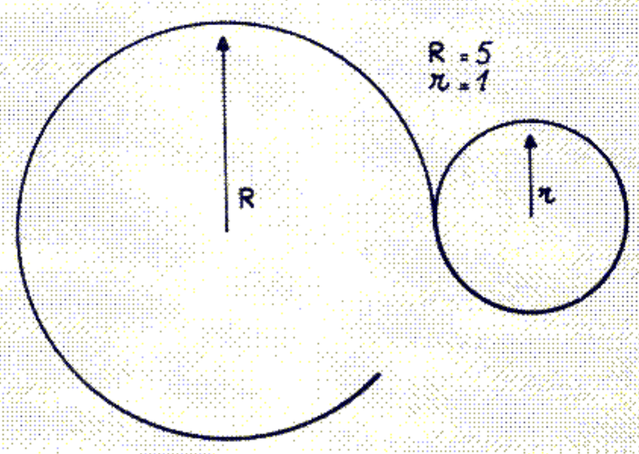
AV 2 DR 5 : le CAP prend la valeur 10.

SI CAP = 0 ALORS STOP : on continue.

DESSIN : abandon de la procédure DESSIN non terminée pour appeler DESSIN, et ainsi de suite...

Au 72e appel, le CAP prend la valeur 0. Donc STOP et retour à la 71e procédure DESSIN qui exécute maintenant AV 10 GA 5 et s'arrête sur le mot FIN. Ce mot provoque le retour à la 70e procédure DESSIN qui, etc.

Au STOP de la 72e procédure DESSIN, il reste à faire 71 fois l'instruction AV 10 GA 5. Ce qui donne le dessin suivant :



Pour refermer le cercle, il suffit d'inclure dans DESSIN, la 72e instruction AV 10 GA 5 :

```
POUR DESSIN
AV 2 DR 5
SI CAP = 0 ALORS AV 10 GA 5 STOP
DESSIN
AV 10 GA 5
FIN
```

LOGO : PETITS COURS ET TRAVAUX DIRIGÉS

```
    POUR ANNULER
    AFFICHE PHRASE [J'AN
NULE ] DERNIER :PROC
    DONNE "PROC SAUFDER
NIER :PROC
    VIDEECRAN
    EXECUTE :PROC
    FIN
```

Il faut savoir que EXECUTE :REP peut générer un message si la ligne introduite n'est pas exécutable. Ce qui entraînera un retour à Logo. Il faudra alors taper à nouveau APPRENDS pour réinitialiser "PROC. Tout n'est pas perdu pour autant : taper PILOTE permet de continuer *sans* perdre les instructions déjà mémorisées. Une meilleure solution consiste à "ATTRAPER" les erreurs et à les envoyer à une procédure chargée à la fois de les gérer et de retourner au contrôle de la procédure dans laquelle l'erreur s'est produite, sans arrêter l'exécution.

Et maintenant, quelques colles

Dans le numéro 1 de *LIST*, quatre procédures à méditer vous étaient proposées : DESSIN, AA, BB, CC (les commentaires qui les concernent se trouvent dans les encadrés ci-contre). En voici de nouvelles :

```
    POUR A :P
    COPIEDEF MOT ". :P :P
    DONNE "T TEXTE :P
    DONNE "V (LISTE PREMIER
:T)
    DONNE "V (LISTE :V
(PHRASE [ ENTREE DANS ] :P))
    B PREMIER :T
    C SAUFPREMIER :T
    DEFINIS :P :V
    FIN

    POUR B :W
    SI :W = [ ] STOP
    DONNE "V LISTE :V (LISTE
(AFR [ LA VALEUR DE ] PRE
MIER :W "EST) (MOT ": PRE
MIER :W))
    B SAUFPREMIER :W
    FIN

    POUR C :X
    SI :C = [ ] STOP
    DONNE "V LISTE :V (LISTE
"AFR PREMIER :X)
    DONNE "V LISTE :V D
    DONNE "V LISTE :V PRE
MIER :X
    C SAUFPREMIER :X
    FIN

    POUR D
    DONNE "Z LISLISTE
    RETOURNE "
    FIN

    POUR E :Q
    COPIEDEF :Q MOT ". :Q
    EFFACE MOT ". :Q
    FIN
```

Les questions que nous vous soumettons sont : que fait l'enchaînement des quatre premières procédures et que fait la dernière ?

Pour mieux répondre, il faut savoir que :

• COPIEDEF "MOT1 "MOT2 prend le texte de la procédure "MOT2 et le recopie dans une nou-

LES PROCÉDURES AA, BB ET CC

```
POUR AA :L1
SI :L1 = [] ALORS AFFICHE :L1 STOP
BB :L1 []
FIN

POUR BB :L2 :L3
SI :L2 = [] ALORS AFFICHE :L3 NIVEAUSUP
CC :L2 :L3 []
FIN

POUR CC :L4 :L5 :L6
SI :L5 = [] BB SAUFPREMIER :L4 PHRASE PREMIER :L4 :L5
SI PREMIER :L4 > DERNIER :L5 ALORS BB SAUFPREMIER :L4
(PHRASE :L5 PREMIER :L4 :L6) SINON CC :L4 SAUFDERNIER :L5 DER
NIER :L5 :L6
FIN
```

Ces procédures exécutent un tri par ordre croissant — ou un tri alphabétique pour certaines versions Logo capables de comparer deux mots.

Elles sont nécessaires pour introduire et initialiser deux listes vides : la première contiendra la liste triée et la seconde sera une liste intermédiaire contenant la fin de la précédente.

Les éléments de la liste à trier sont pris un par un et enlevés au fur et à mesure. Le tri sera terminé lorsque la liste sera vide. L'élément pris est-il supérieur au dernier de la liste déjà triée ? Si oui, il est mis à la fin et l'élément suivant est pris. Sinon, le dernier élément de la liste triée est transféré au début de la liste intermédiaire et la recherche reprend avec la liste ainsi amputée. Les transferts ont lieu jusqu'à ce que l'élément soit placé et la liste triée est concaténée à la liste intermédiaire.

Il est bien rare d'avoir à trier une liste vide. La ligne qui teste L1 dans la procédure AA est donc inutile. Il reste encore une "bogue". Elle se produit si deux éléments de la liste à trier sont égaux. Il ne faut donc pas tester si l'élément à placer est supérieur au dernier élément de la liste triée. Voici donc les procédures corrigées :

```
    POUR AA :L1
    BB :L1 []
    FIN

    POUR BB :L2 :L3
    SI :L2 = [] ALORS AFFICHE :L3 NIVEAUSUP
    CC :L2 :L3 []
    FIN

    POUR CC :L4 :L5 :L6
    SI :L5 = [] BB SAUFPREMIER :L4 PHRASE PREMIER :L4 :L5
    SI NON PREMIER :L4 < DERNIER :L5 ALORS BB SAUFPREMIER :L4
(PHRASE :L5 PREMIER :L4 :L6) SINON CC :L4 SAUFDERNIER :L5 DER
NIER :L5 :L6
    FIN
```

PETITS COURS ET TRAVAUX DIRIGÉS

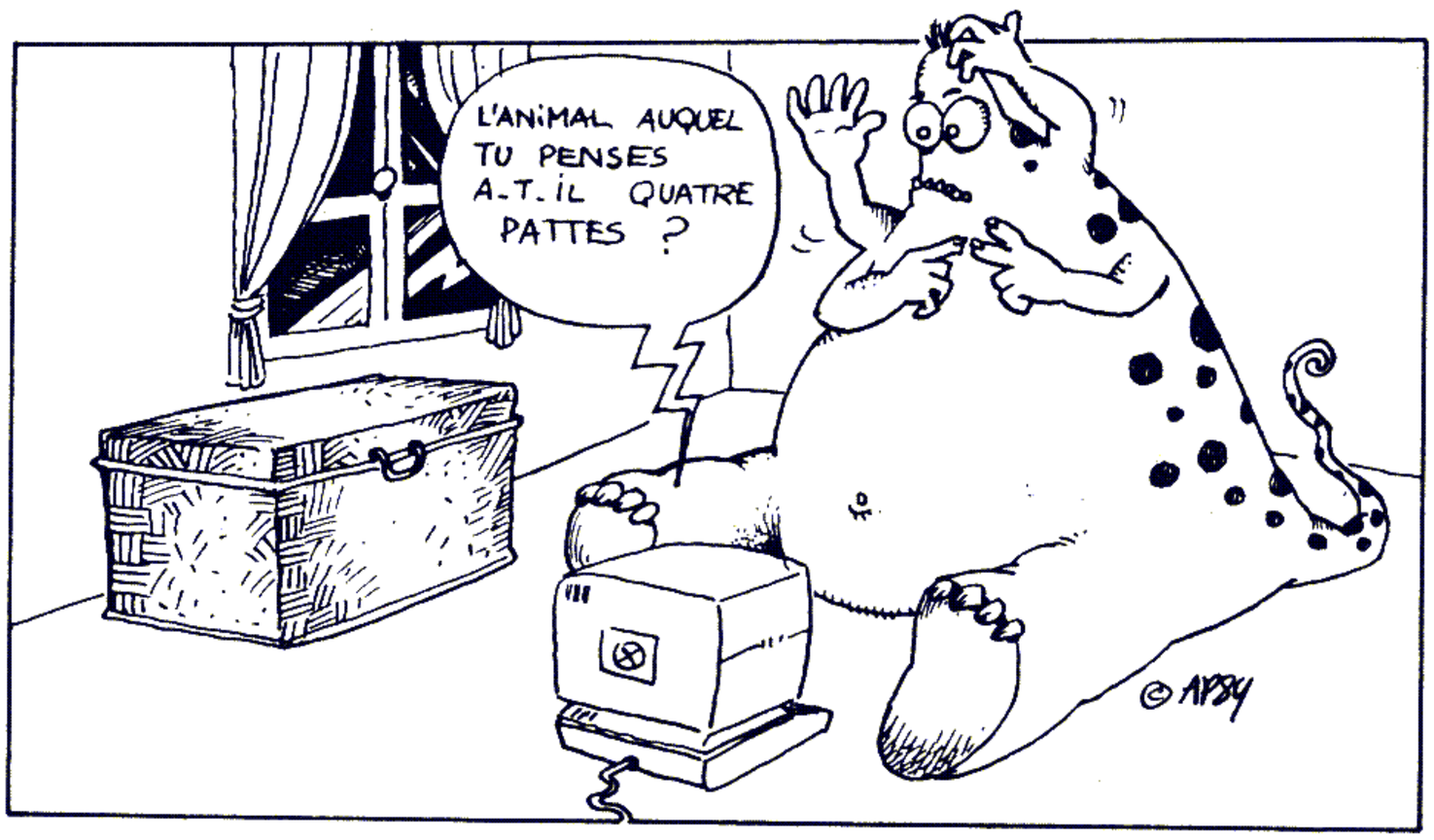
velle procédure dont le nom est "MOT1 ;

• le parenthésage permet de donner un nombre quelconque d'arguments aux primitives qui, sans lui, en nécessiteraient deux ;

• AFR est identique à AFFICHE mais ne provoque pas de retour à la ligne ;

• RETOURNE revient à la procédure appelante en retournant un résultat ; par exemple, RETOURNE "VRAI retourne le mot VRAI ;

• EFFACE détruit une procédure de l'espace de travail.



C'est une vache ? Non, non et non !

Le programme "intelligent" qui est généralement donné en exemple, se nomme "ANIMAUX. Sauriez-vous le retrouver sachant qu'il possède les caractéristiques suivantes (les majuscules représentent les messages affichés par l'ordinateur et les minuscules, les réponses faites par l'utilisateur) :

PENSE A UN ANIMAL. JE VAIS LE DEVINER
TON ANIMAL A-T-IL QUATRE PATTES ?

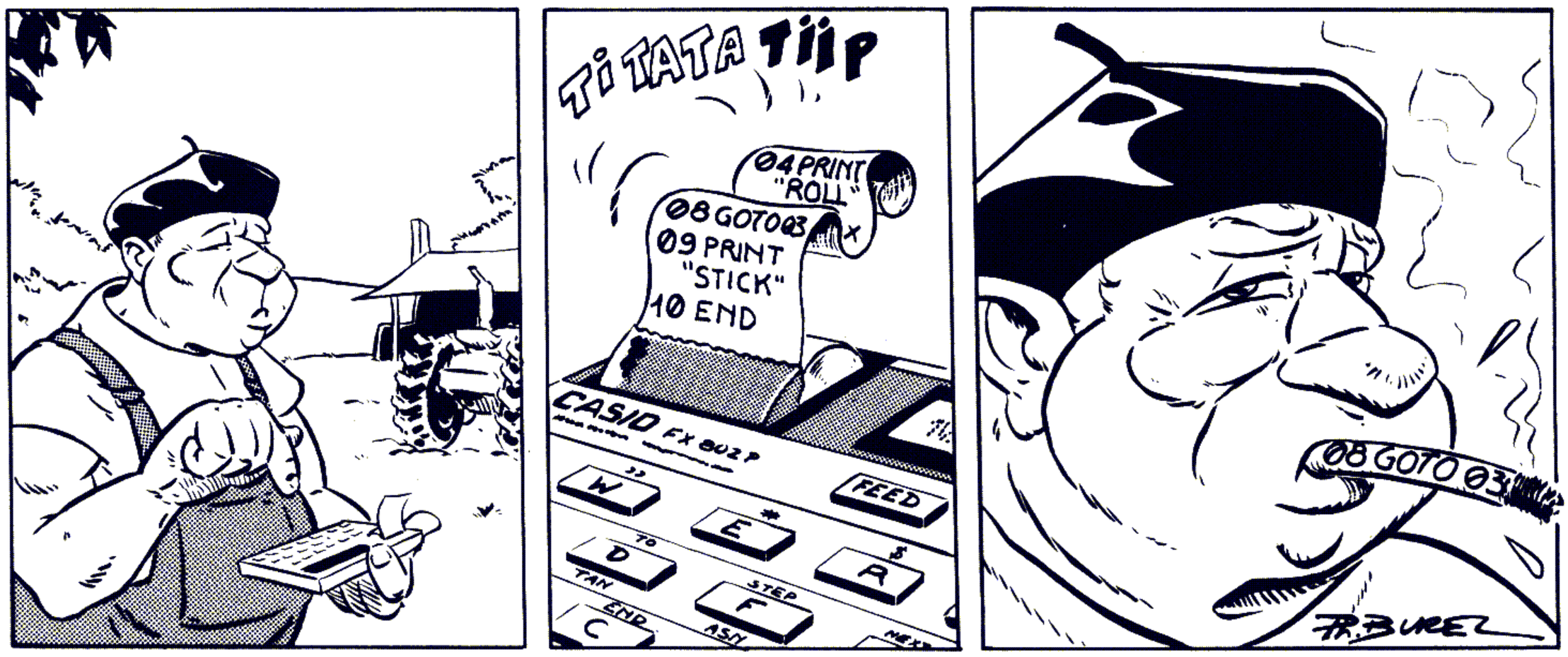
Supposons que la réponse de l'utilisateur soit oui.

C'EST UNE VACHE
non
A QUEL ANIMAL PENSAIS-TU ?
un cheval
QUELLE QUESTION DOIS-JE POSER POUR DISTINGUER UNE VACHE D'UN CHEVAL ?
court-il vite ?
OK J'AI COMPRIS, RECOMMENÇONS

Au coup suivant, si la réponse à la question TON ANIMAL A-T-IL QUATRE PATTES ? est oui, la prochaine question sera TON ANIMAL COURT-IL VITE ? Et si la réponse est oui encore, le programme proposera UN CHEVAL. Et si ce n'est pas un cheval, il faudra encore taper la question à poser pour distinguer le cheval de l'animal proposé. A chaque nouveau tour, le programme s'enrichit. Bien "éduqué", il devient presque intelligent.

Vous avez certainement des idées pour écrire des procédures qui en créent d'autres, qui s'auto-détruisent, qui "mûrissent" en fonction de conditions extérieures... Un conseil : n'allez pas trop vite, ne visez pas trop loin, ne cherchez pas encore à faire de votre ordinateur un bon adversaire aux échecs !

Robert DAGUESSE



NOIR ET BLANC ET VICE-VERSA

COMME la plupart des ordinateurs, ceux dont le processeur est un 6502 (Oric ou Apple par exemple) ne se programment pas seulement en Basic. Vous pensez peut-être que le langage-machine est trop difficile à apprendre ? Avant d'y renoncer, faites tout de même un petit essai...

Il existe plusieurs bonnes raisons de programmer en langage-machine :

- mieux comprendre "le langage du processeur" et donc le fonctionnement des ordinateurs ;
- trouver une solution à la lenteur d'exécution de certains ensembles d'instructions du Basic ;
- compléter le langage Basic.

Les fonctions techniques les plus courantes existent déjà comme ordres du Basic, on les appelle des "routines". Ce sont, par exemple, les instructions qui permettent de travailler avec les périphériques : CLS efface l'écran, GET A$ met dans A$ un caractère obtenu sur le clavier, ZAP émet un son élaboré, etc. Mais il y a d'autres fonctions qui ne se trouvent pas dans le Basic, il faut donc les programmer soi-même.

Le Basic, disponible quand on met en route l'ordinateur, contient heureusement des instructions qui permettent de créer et d'utiliser des "routines" en langage-machine. Sur l'Oric, par exemple, ce sont : POKE, CALL, USR, etc.

En langage-machine, on peut pratiquement faire tout ce que l'on veut mais à deux conditions. Il faut :

- savoir à quel endroit implanter la routine en langage-machine ;
- déterminer dans le moindre détail tout le travail à faire effectuer par la routine.

Sauvegarder avant de lancer

C'est le deuxième point qui est le plus difficile à réaliser, car on ne dispose pas d'un "détecteur" d'erreurs comme en Basic : à la moindre bogue, il ne reste plus qu'à débrancher et à tout reprendre à zéro.

La prudence conseille donc de toujours faire une sauvegarde d'un programme avant de l'exécuter : SAVE "PROGRAMME", puis RUN.

C'est une routine d'inversion vidéo sur Oric qui nous servira d'exemple. Il s'agit d'inverser un dessin blanc sur fond noir, en dessin noir sur fond blanc et réciproquement, en mode graphisme haute résolution.

Première question donc : où placer la routine ? Sûrement pas dans la partie utilisée par le programme en Basic ou par l'écran, de #500 à #BFE0, la routine serait "écrasée". Au-dessus de ces adresses, il n'y a pas de place. Mais à partir de #400, l'Oric laisse un peu de place pour une petite routine.

Inversion point par point

En graphisme haute résolution, l'image de l'écran est constituée de 8 000 cellules de 6 pixels chacune. Chaque cellule est un caractère en mémoire centrale dont 6 bits sur 8 correspondent à un pixel, c'est-à-dire un point sur l'écran. Si l'un des six bits dans le caractère est à 1, le point correspondant brille. Sinon, il est éteint.

Le problème consiste donc à inverser la valeur de tous les bits de ces caractères en veillant à ne pas toucher aux deux bits de poids fort (ce qui aurait des effets inattendus).

On pourrait utiliser l'ordre Basic FILL pour chacune des 8 000 cellules mais cela prendrait plusieurs minutes. Une routine en langage-machine fera le même travail en un cinquième de seconde.

Le processeur de l'Oric, le 6502, possède un registre A pouvant contenir n'importe quel caractère.

POUR LES UTILISATEURS DE L'APPLE

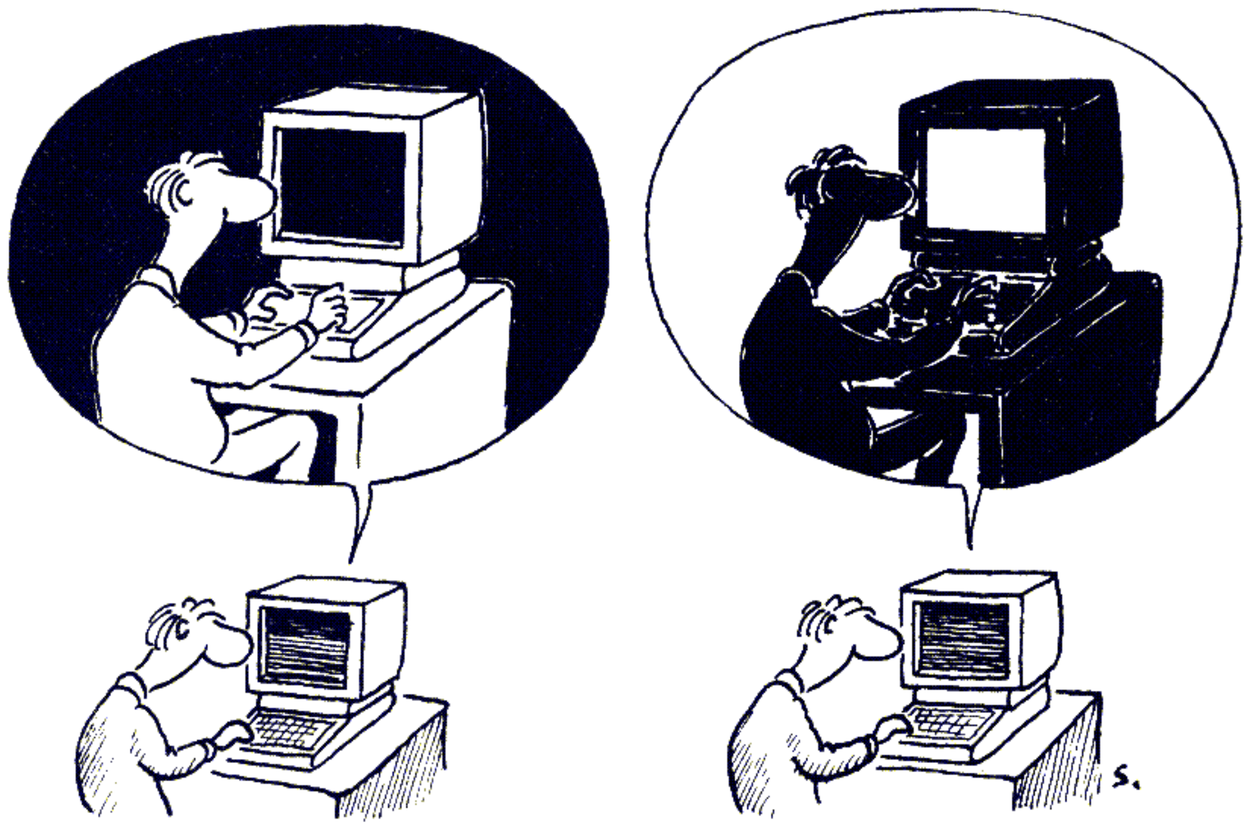
L'Apple, lui aussi, contient un 6502, et le mode graphisme haute résolution utilise 40 cellules pour chaque ligne.

Mais chaque cellule se compose de 7 pixels, ce qui donne pour une ligne 40×7 = 280 points (le huitième bit donne la couleur des 7 points). Par conséquent, l'instruction EOR doit utiliser un masque de 8 bits à 1.

De plus, l'image de l'écran en mémoire centrale va de #2000 à #4000, et le programme devient :

```
A9  20  LDA #20        premier bloc écran dans A
85  07  STA 07         range dans 07 (octets disponibles)
A0  00  LDY #00        début de bloc dans Y
84  06  STY 06         range dans 06
B1  06  LDA (06),Y     charge le caractère désigné par 06 et Y
49  AF  EOR #FF        masque d'inversion de 7 bits
91  06  STA (06),Y     range à la même place
C8      INY            caractère suivant
D0  F7  BNE 308        si Y non égal zéro remonte de 7 (le LDA)
E6  07  INC 07         bloc suivant (+1)
A5  07  LDA 07         charge
C9  40  CMP #40        et compare #40=dernier bloc écran
D0  EF  BNE 308        si non remonte de 17 (le LDA)
60      RTS            retour au programme principal
```

Dans la mémoire de l'Apple, il y a de la place disponible pour nos routines à partir de #300.



On utilise l'instruction EOR X pour combiner chacun des bits du registre A avec les bits du caractère X (appelé masque) selon une relation booléenne exclusive : chaque bit de X à 1 inverse le bit correspondant de A. Par exemple :

si X =	0	0	1	1	1	1	1	1
et A =	0	0	1	0	1	0	1	0
alors A =	0	0	0	1	0	1	0	1

Dans A, on obtient ainsi l'inverse des six bits de poids faible.

Un registre de huit bits, autrement dit d'un octet, ne peut contenir que les valeurs de 0 à 255. C'est, et de loin, insuffisant pour les 49 152 caractères.

Une première solution consiste à découper la mémoire en "pages" de 256 caractères, elles-mêmes numérotées de 0 à 192. On réserve une page unique, pour les échanges entre pages. La page zéro ne demande pas à être numérotée car elle va de 0 à 255 dans la mémoire.

L'utilisation de la page zéro fait gagner du temps et de la place puisque c'est un "adressage court" mais tous les caractères ne sont pas disponibles : les 50 premiers à l'exclusion de 18, 19, 26, 27 et 28 sont utilisables mais, à partir de 52, il vaut mieux s'abstenir.

Une deuxième solution est de découper l'information en blocs de 256 caractères, l'adresse de chaque bloc étant mise en page zéro. On utilise alors un registre pour compter les caractères du bloc.

Le 6502, processeur de l'Oric, est pourvu d'un registre Y dont la valeur s'ajoute au contenu de l'adresse lors des échanges avec le registre A.

L'écran est découpé en plusieurs blocs de #A0 à #BF (32 blocs adressés par 02 de la page zéro). Chaque bloc est constitué de 256 caractères de

Inversion vidéo
Programme en langage-machine 6502
Version pour Oric-1
Auteur Max Hagenburger


```
1000 REM routine inversion vidéo
1010 A = #400
1020 D$="A9A08502A0008401B1014
     93F9101C8D0F7E602A502C9BF
     D0EF60"
1030 FOR I=1 TO LEN(D$) STEP 2
1040 V=VAL("#"+MID$(D$,I,2))
1050 POKE A,V :A=A+1
1060 NEXT
1095 RETURN
```

A n'importe quel endroit du programme, on peut procéder à un appel à la routine d'inversion vidéo par

```
110 GOSUB 1000 :REM implantation
    routine
120 HIRES
....
210 CURSET 100,100,1
220 FOR I=1 TO 76 STEP 3 :CIR
    CLE I,1 :NEXT
230 FOR I=1 TO 9
240 CURSET 10+CV,10,0
250 CHAR ASC(MID$("INVER
    SION",I,1)),0,1
260 CV=CV+6 :ZAP
270 FOR J=0 TO 1 :CALL #400
    :WAIT 25 :NEXT
280 NEXT
....
990 END
```

Faites SAVE"INVERSION" puis RUN pour voir !

Pour implanter des routines plus importantes (dépassant 33 octets), on fera :

```
1020 REPEAT :READ D$
....
1070 UNTIL D$="GG"
1080 DATA A9A08502A0008401...
....
1090 DATA ZZ
```

LANGAGE MACHINE DU 6502

LA ROUTINE EN ASSEMBLEUR

```
LDA #AO     charge le numéro du premier bloc dans A
STA 02      range dans 02 (page zéro)
LDY #00     charge le numéro du premier caractère du bloc dans Y
STY 01      range dans 01
            (l'adresse du 1er caractère de l'écran est dans 01/02)
LDA (01),Y  charge le caractère désigné par 01 et compte par Y
EOR 3F      inverse les 6 bits (6 points d'écran)
STA (01),Y  range le résultat à la même place (de l'écran)
INY         ajoute +1 au registre Y (caractère suivant)
BNE -9      si Y non = 0 remonte de 8 octets (sur LDA)

INC 02      ajoute +1 sur 02 (bloc suivant)
LDA 02      charge le numéro du bloc dans A
CMP #BF     compare avec #BF
BNE -17     si non égal remonte de 17 octets (sur LDA),
            n'oubliez pas que Y est à zéro (début de bloc)
RTS         retour au programme principal (Basic)
```

Remarque : en plus des calculs, le processeur doit faire des choix. Pour cela, il dispose de tests et d'instructions de comparaison, mais aussi d'instructions de branchement suivant le résultat du test.

Les tests les plus fréquents consistent à décompter (les caractères d'un bloc...) et la comparaison est en fait une soustraction interne du 6502.

C'est pourquoi on utilise le "branchement si non égal à zéro", BNE pour recommencer un traitement.

#00 à #FF (adresse de début dans 01, mais comptés dans le registre Y).

On inverse chaque caractère dans le registre A. On a donc recours à deux répétitions comme on le voit dans l'èncadré ci-dessus.

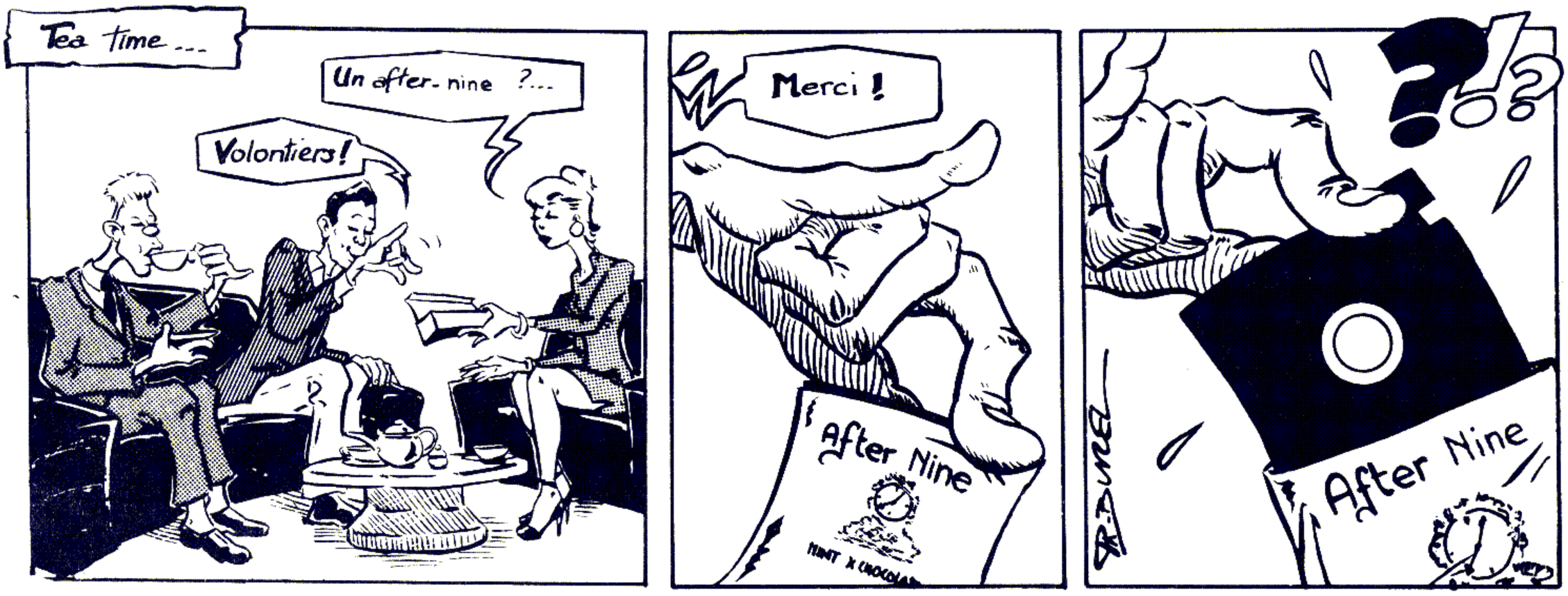
Implanter la routine

L'instruction POKE permet de mettre une valeur dans un caractère de la mémoire. Pour implanter les valeurs de la routine, vous pouvez vous reporter au manuel de l'Oric-1 (page 140) ou à celui beaucoup plus étoffé de l'Atmos. Nous allons choisir une méthode facilement généralisable pour les routines plus importantes.

Le langage-machine du 6502 possède 58 instructions qui sont codées en 1, 2 ou 3 octets suivant qu'il n'y a pas d'adresse, une adresse en page zéro (1 octet) ou une adresse longue (2 octets). Dans notre exemple : LDA #A0 sera codé A9 A0 ; STA 02 = 85 02 ; LDY #00 = A0 00 ; STY 01 = 84 01 ; LDA (01),Y = B1 01 ; E0R #3F = 49 3F ; STA (01),Y = 91 01 ; INY = C8 ; BNE −7 = D0 F7 (#100−7 = #F7 si > #80 BNE va vers l'arrière, sinon vers l'avant) ; INC 02 = E6 02 ; LDA 02 = A5 02 ; CMP #BF = C9 BF ; BNE−17 = D0 EF et enfin RTS = 60.

Vous trouverez, avec les quelques lignes d'implantation de la routine, un programme de démonstration. J'espère que cela vous donnera l'envie de découvrir à votre tour le langage-machine.

Max HAGENBURGER



UNE ROUTINE SUR PB-700

COPIE D'ÉCRAN

PROFITANT de quatre lignes d'affichage, le PB-700 présente parfois de bien charmants dessins. Mais ils sont à l'écran. Pour en obtenir une trace imprimée, deux conditions s'imposent : programmer une routine de copie de l'écran et... disposer de l'imprimante-table traçante, FA-10.

Avec certains ordinateurs, une copie imprimée de l'écran s'obtient grâce à une instruction unique, mais avec le PB-700, c'est tout un programme. Ou plutôt deux programmes, l'un proposant une copie de l'écran point par point, l'autre la permettant ligne par ligne. Au choix.

La copie point par point utilise le principe de l'instruction POINT : POINT (X, Y) est égal à 1 si le point de l'écran de coordonnées X et Y est activé, et à 0 sinon. L'écran étant une matrice continue de 160 sur 32 points (soit 5120 points au total), X pourra varier de 0 à 159 et Y de 0 à 31.

Le programme va analyser un par un tous ces points grâce à deux boucles emboîtées :

- FOR X = 0 TO 159
- FOR Y = 0 TO 31

Chaque fois que l'expression POINT (X, Y) décèlera un point allumé sur l'écran, le PB-700 donnera l'ordre à la table traçante de dessiner un petit carré représentant ce point. Les coordonnées de ce carré étant proportionnelles à celles du point de l'écran, c'est-à-dire à X et à Y, l'affichage sera fidèlement reproduit sur le papier.

La mise en œuvre du programme utilise deux des dix zones de programmation. Dans l'une d'elles se trouve le programme de départ, c'est-à-dire celui qui réalise le tracé sur l'afficheur et dans l'autre, se trouve la routine de copie d'écran. Ici, les zones choisies sont P0 et P5.

La jonction entre les deux se fait en plaçant dans le programme principal une instruction de saut à la zone P5 (GOTO PROG 5) ou bien en considérant cette zone comme un sous-programme (GOSUB PROG 5). Le tracé est alors automatique, et à la fin de la routine une instruction de retour doit renvoyer au programme principal, en P0.

On va plus vite à la main...

Cependant, cette routine présente un défaut sérieux : l'exécution en est très lente. Il faut environ une dizaine de minutes pour que tous les points soient analysés et tracés. Ceux qui n'ont pas suffisamment de patience pourront se reporter directement au programme de copie ligne par ligne.

Il s'agit en fait d'une recopie manuelle de l'écran. La table traçante dessine un cadre qui représente l'afficheur. Puis le PB-700 demande l'introduction, ligne par ligne, de ce qu'il faut placer dans ce cadre. On a alors le choix entre deux options : C comme Caractères ou G comme Graphismes. Si la ligne à tracer ne comporte que des caractères directement accessibles au clavier, tels que majuscules, minuscules, chiffres ou ponctuation, il faut choisir l'option C ; on tape directement la ligne et une pression sur ↵ l'imprime aussitôt dans le cadre.

En revanche, si la ligne comporte un caractère ne pouvant être obtenu qu'avec son code ASCII précédé de CHR$, il faut choisir l'option G. Ce sera nécessaire par exemple pour les symboles « couleurs de cartes », pour les caractères japonais ou « katakana », etc. On donnera alors le nombre de signes de la ligne, puis on entrera, un par un, le code de chacun d'eux. Et la ligne sera imprimée automatiquement après un appui sur ↵.

Plus rapide, cette méthode est aussi plus « fatigante » puisqu'il faut entrer des codes qu'on ne connaît pas toujours. Quant au résultat graphique, il est bien différent d'une méthode à l'autre. Il ne reste plus qu'à choisir le plus joli. Et là, à chacun ses goûts.

Pierrick MOIGNEAU

La marche à suivre ligne par ligne

Vérifier que le *plotter* (table traçante) est prêt à fonctionner : touche PLOTTER sur ON, touche ON LINE/LOCAL sur ON LINE. En faisant RUN↵, on déclenche le tracé du cadre ; sur l'afficheur apparaît cette question :

LIGNE N° 1 / Caractères (C) ou Graphismes (G) ?

En répondant C, l'affichage devient :

ENTREZ VOTRE LIGNE.

En tapant, par exemple, C EST UN ESSAI suivi de↵, les lettres sont aussitôt tracées à l'emplacement de la première ligne dans le cadre. La question de départ revient ensuite :

LIGNE N° 2 / Caractères (C) ou Graphismes (G) ?

Si cette fois la réponse est G, une autre question apparaît :

NB DE SIGNES ?

Il ne faudra pas dépasser 20. Chaque code pourra alors être rentré à la suite de la demande :

Code du caractère no xx ?

Par exemple, si le premier signe est un as de pique, il faut taper 232↵.

Quand tous les codes sont introduits, la ligne est imprimée à sa place dans le cadre. De même pour les lignes suivantes.

Copie d'écran point par point
Programme pour PB-700 et FA-10
Auteur Pierrick Moigneau

En zone P0

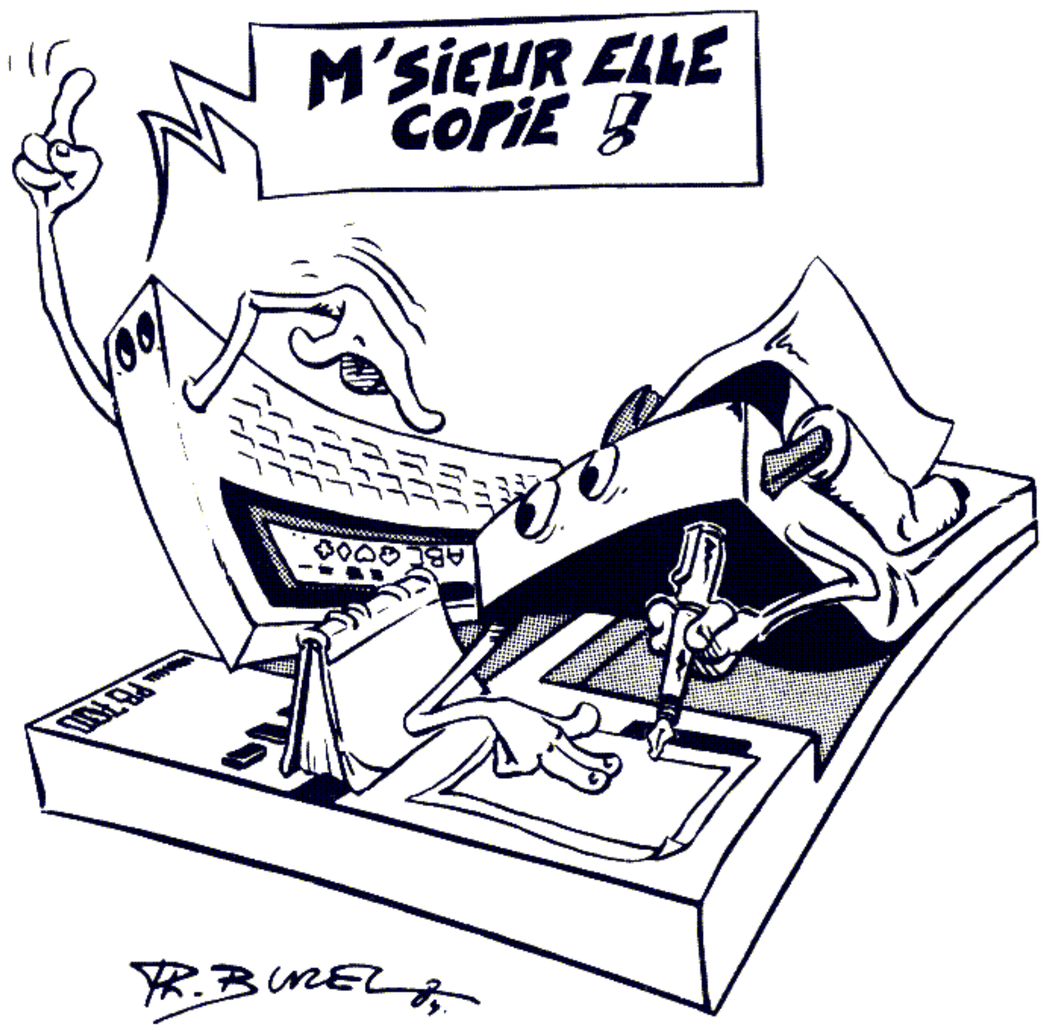
```
 5 REM....Programme principal...
10 PRINT "EXEMPLE DE COPIE..."
15 PRINT "D ECRAN...*/* point par poi
   nt"
20 DRAW(2,28)-(156,28)-(156,30)-(2,30
   )-(2,28)
30 GOSUB PROG 5
```

En zone P5

```
 10 REM--- SS-PRGM  COPIE D ECRAN ---
 20 REM ---INITIALISATION ET TRACE DU
    CADRE ---
 30 LPRINT CHR$(28);CHR$(37):LPRINT "S
    2"
 40 LPRINT "A0,0,96,-20"
 45 REM --- ANALYSE DE L AFFICHEUR ---
 50 FOR X=0 TO 159
 60 FOR Y=0 TO 31
 70 IF POINT(X,Y)=0 THEN 100 ELSE 80
 75 REM --- TRACE DU POINT ---
 80 LPRINT "M";X/1.6666;",";-(Y/1.666)
    -1.5
 90 LPRINT "P";CHR$(46)
100 NEXT Y:NEXT X
110 RETURN
```

Exemple d'exécution point par point :

```
EXEMPLE DE COPIE...
D'ECRAN...*/* point
par point
```



Copie d'écran ligne par ligne
Programme pour PB-700 et FA-10
Auteur Pierrick Moigneau


```
 5 REM --- Copie d ecran ligne par li
   gne ---
10 CLEAR :CLS :DIM A$(3)*20,E(20)
15 REM --- Mode graphique ---
20 LPRINT CHR$(28);CHR$(37)
25 REM --- Trace du cadre ---
30 LPRINT "A0,0,96,-20"
35 REM --- Definition des espaces et
   de la taille des caracteres ---
40 LPRINT "Z4":LPRINT "S2"
45 REM --- ENTREE DES LIGNES ---
50 FOR B=0 TO 3
60 CLS :PRINT "LIGNE NO";B+1,"Caracte
   res(C) ou graphismes(G)?";
70 D$=INKEY$:IF D$="" THEN 70 ELSE IF
    D$="G" THEN 200
80 CLS :INPUT "*ENTREZ VOTRE LIGNE:",
   A$(B)
85 REM --- Position du debut deligne
   ---
90 C=-4-(B*5)
100 LPRINT "M1,";C
105 REM --- Impression ---
110 LPRINT "P";A$(B)
120 NEXT B:END
190 REM --- Cas des graphismes ---
200 CLS :INPUT "NB DE SIGNES";J:FOR I=
    1 TO J
210 PRINT "Code du caractere no";I;:IN
    PUT E(I)
220 A$(B)=A$(B)+CHR$(E(I)):NEXT I
250 GOTO 90
```

Exemples d'exécution ligne par ligne :

```
...Exemple de copie
d ecran sur PB-700.
ABC時分秒♤♡▤▩
```

```
*Exemple de recopie
d ecran sur PB-700
          ABC時分秒 ♤♡◇♧▤
```

ddc SÉMAPHORE INTERNATIONAL

LILLE BRUXELLES GENEVE LILLE BRUXELLES GENEVE

DDC SEMAPHORE INTERNATIONAL est incontestablement le groupe le plus performant en matière d'adaptation et distribution de logiciels de qualité en français. Ecrivez-nous !

Utilitaires, langages et programmation en français pour Spectrum

* TASWORD II (Traitement de texte accentué) Toutes les fonctions professionnelles.M/D. CHF68 255FF 1999FB x =....
* TASPRINT (5 polices de caractères Suppl.)M/D. Doublez les possibilités de votre imprimante. CHF39 145FF 990FB x =....
* BUDGET,gestion financière familiale complète CHF35 95FF 790FB x =....
* GESTION STOCKS (Essentiellement pratique)M/D. Entrée, sortie recherche et analyse budgétaire. CHF45 169FF 1290FB x =....
* COMPTABILITE PROFESSIONNELLE Journaux achats/ventes. Banque. Facturation. Téléphonez-nous.
* SUPERCODE (boîte à outils,120 routines c/m ,microdrives) CHF50 180FF 1290FB x =....
* BASIC étendu (10 instructions BASIC supplémentaires) CHF35 100FF 990FB x =....
* Escargot-LOGO (graphisme tortue) CHF50 180FF 1290FB x =....
* Sémaphore MICRODRIVE FORTH 1.1 (livré sur 2 cartouches + manuel français de 288p.) Editeur, graphismes/sons, E/S, double précision... Nombreux blocs d'expansion disponibles, enfin un vrai FORTH sur Spectrum. CHF150 500FF 3490FB x =....

Jeux de réflexion, adresse et arcade pour Spectrum (doc. franç)

* SUPERECHECS 3.0 - 10 niveaux de jeu, le plus intéressant et puissant sur Spectrum. CHF35 100FF 990FB x =....
* MONSTRES EN ENFER...Un cauchemar parmis les créatures du démon. CHF29 85FF 795FB x =....
* METEORIDES. La meilleure version micro du jeu d'arcades "ASTEROIDES" CHF29 85FF 795FB x =....
* ROBON. Combattez pour survivre sur la planète ZETON II 9 niveaux. Sons et graphismes CHF29 85FF 795FB x =....
* MEGAPEDE. Pas moins de 9 niveaux de jeu passionnant avec sons et graphismes haute résolution. CHF29 85FF 795FB x =....
* REPULSAR. Seul à vos manettes pour défendre votre cité. Effets sonores et graphisme H.R. CHF29 85FF 795FB x =....
* LES OISEAUX DE FEU. Vous êtes la dernière chance de l'humanité. Ne craquez pas... CHF32 95FF 895FB x =....
* OSTRON. Chevalier du futur, combattez les Seigneurs des Ténèbres... CHF32 95FF 895FB x =....
* UGH! Revivez l'âge des cavernes (complet avec ptérodactyles et tyrannosaures) CHF32 95FF 895FB x =....
* STARBLITZ. Guerre-éclair dans l'espace. CHF32 95FF 895FB x =....
* FLIPPER. Un réalisme superbe! Un vrai flipper dans votre salon... CHF29 85FF 795FB x =....
* STOPPEZ LES MISSILES! Le premier jeu d'arcade pacifiste, non-sexiste et satirique! CHF29 85FF 795FB x =....
* CHAINE DE MONTAGE. Vous n'aurez rien à envier à Stakkanov! CHF29 85FF 795FB x =....

Jeux d'adresse et d'arcade en français pour ORIC 48 et ATMOS

* SUPER METEORES. Traversez les champs de météores, attention aux bosses et aux lasers ennemis! CHF35 95FF 895FB x =....
* GALAXIANS. Des vaisseaux étrangers pénètrent votre secteur ...Alerte rouge! CHF35 95FF 895FB x =....
* LA REVANCHE DE DRACULA ... une visite de château toute tranquilité, Brrrr CHF35 95FF 895FB x =....
* LA RAGE D'ACHERON. Gagnez la guerre des étoiles à bord de votre Oric ou ATMOS CHF35 95FF 895FB x =....
* LE GEANT GLACE. Délivrez le pays des sortilèges cryoniques de Morgar. CHF35 95FF 895FB x =....

Jeux d'adresse et d'arcade en français pour COMMODORE 64

* SUPERECHECS 3.0 - 10 niveaux de difficulté CHF40 125FF 995FB x =....
* ZOIDS Excellent jeu d'arcade... CHF40 125FF 995FB x =....
* REVELATION. Echappez-vous des cavernes de l'enfer en combattant les monstres de l'Apocalypse CHF40 125FF 995FB x =....
* JEEPERS CREEPERS. L'araignée qui tisse plus vite que son ombre, attention aux oiseaux et autres insectivores... CHF40 125FF 995FB x =....
* BUCKSQUAD Action soutenue! CHF40 125FF 995FB x =....
* QUARK ATTACK Concentration... CHF40 125FF 995FB x =....

BON DE COMMANDE A RENVOYER A -- DDC SEMAPHORE INTERNATIONAL --
FRANCE: 104 Rue Nationale -- 59800 Lille tlx. 129552 f
SUISSE: CH 1283 La Plaine -- Genève tel. 22 / 541195
BELGIQUE: 1 Rue du Planiau -- 1301 Bierges-Lez-Wavre
tel. 02 / 6540611 - 02 / 6539553 tlx. 65946 b
OUVERTS LE DIMANCHE (sur rendez-vous téléphonique)

*** Je verse la somme de .....CHF / FF / FB au compte
- LILLE: BNP 85 Rue Nationale No 224420-12 DDC Séma. Intern.
- GENEVE:CCP 12-24798-3 ou soc.de ban.suisse C0-192.930-1
- BRUXELLES: Société générale 271-0165791-50 DDC Sémaphore int.
*** je joins un chèque barré de .....CHF/FF/FB au nom de DDC
*** je vous envoie un mandat poste de .....CHF / FF / FB.

Logiciels et progiciels Spectrum + documentation en français

* OMNICALC 2. Le tableur professionnel pour Spectrum, rapide et puissant (M/D + histogrammes). CHF50 185FF 1290FB x =....
* MASTERFILE (16 ou 48K) Gestion de fichiers définissables par l'utilisateur.CHF59 195FF 1490FB x =....
* MF-PRINT utilitaire d'impression 80 col. pour MASTERFILE CHF29 85FF 795FB x =....
* DLAN Utilitaire d'affichage de texte, d'écrans. Réalisez votre publicité vidéo. CHF40 125FF 995FB x =....
* HISOFT PASCAL. Initiez-vous à ce puissant langage informatique. CHF113 420FF 2990FB x =....
* ASSEMBLEUR CHF32 95FF 895FB x =....
* DESASSEMLEUR CHF32 95FF 895FB x =....
* COMPILATEUR virgule flotante Convertit vos programmes BASIC en langage machine CHF50 185FF 1290FB x =....
* COUNTRIES OF THE WORLD. Cartes du monde, et caractéristiques de 169 paysCHF29 85FF 799FB x =....
* JOUEUR DE BRIDGE. Jouez un vrai jeu de bridge contre votre Spectrum ou Atmos CHF35 95FF 895FB x =....
* PENETRATOR. Pénétrez la base ennemie... CHF40 125FF 995FB x =....
* CHEQUERED FLAG. Simulation formule 1 CHF29 85FF 795FB x =....
* MAZIACS. Trésors et labyrinthes maudits. même prix x =....
* SIMULATEUR VOL (PSION). Avion de tourisme. " " x =....
* VOL DE NUIT (HEWSON). Avion de tourisme CHF32 95FF 895FB x =....
* CONTROLE TRAFIC AERIEN (HEWSON). CHF32 95FF 895FB x =....
* PILOTE DE CHASSE (Dig.Int.)Pilotez un F.15 CHF35 100FF 895FB x =....
* BOMBARDIER (Night Gunner - Dig. Int.) Protégez votre appareil des chasseurs ennemis... CHF35 100FF 895FB x =....
* WHEELIE. Eclatez-vous en "Zedixaki 500" CHF29 85FF 795FB x =....
* SCUBA DIVE. Trésors et dangers marins CHF29 85FF 795FB x =....
* TRAIN GAME. Contrôle du trafic ferroviaire CHF29 85FF 795FB x =....
* TRON. Protégez-vous des octets et pixels empoisonnés. CHF29 85FF 795FB x =....
* TUNNEL 3D. Survivez aux obstacles de ce souterrain diabolique. CHF32 95FF 895FB x =....

- Imagine:
* * ARCADIA, AH DIDDUMS, MOLAR MAUL, JUMPING JACK, STONKERS ZIP-ZAP, ZZOOM, PEDRO, ALCHEMIST, CHF22 75FF 525FB x =....

- Quicksilva:
* * FRED, SNOWMAN, DRAGONSBANE, BUGABOO, MINED OUT, FRENZY ASTROBLASTER, 3D ANT ATTACK, CHF25 80FF 595FB x =....

- Howson Consultants:
* * 3D LUNATTACK, 3D SEIDDAB ATTACK, D-LITHIUM LIFT, KNIGHT DRIVER, 3D SPACE WARS, CHF29 90FF 795FB x =....

* GRANDE OFFRE DE LA RENTREE pour utilisateurs de ZX 81: 4 cassettes de jeux sinclair zx 81 . CHF40 125FF 995FB x =....

PUBLICATIONS.

* DECOUVRIR LE SPECTRUM. Le complément idéal du manuel Sinclair. CHF24 79FF 549FB x =....
* 16/48 MAGASINE sur CASSETTE en français 4 numéros par an,le numéro CHF15 55FF 390FB x =....
1 an (4 numéros) CHF55 199FF 1390FB x =....
* REVUE MICRO EUROPE pr Sinclair , CHF 3 10FF 70FB x =....
* REVUE MICRO-QL (Sinclair) le No, CHF 3 10FF 70FB x =....
* DOCUMENT.DDC Sémaphore International CHF6 20FF 150FB x =....

MATERIELS.

* SPECTRUM 48 K PAL (Garantie 6 Mois) CHF495 1960FF 12500FBx =..
* Interface No1 (RS/232 + M.D. CHF218 890FF 5690FB x =....
* ZX 81 (PAL) + Manuel Français + Microclavier + 16 k .. 4 cassettes de logiciel Sinclair. CHF325 1190FF 7990FB x =...
* Microclavier pour ZX 81 (Touches en relief) CHF40 125FF 995FB x =....
* Cable RS/232 pour interface No1 CHF68 235FF 1880FB x =....
* Microdrive ZX CHF218 890FF 5690FB x =....
* Cartouches vierges pour microlecteurs (4 pièces) CHF85 315FF 2115FB x =....
* Interface programmable + Manette de jeux DDC-S. CHF159 590FF 3990FB x =....
* Manette de jeu incassable CHF49 140FF 1150FB x =....
* Synthétiseur de voix Currah CHF119 450FF 2990FB x =....
* Extension Spectrum 16/48k CHF130 510FF 3500FB x =....
* Imprimante ALPHACOM 32 CHF290 1100FF 7490FB x =....
* Rouleaux pour ALPHACOM 32 (5 pcs) CHF35 139FF 850FB x =....
* Clavier professionnel ZX81/Spectrum Dk Tronics avec pavé numérique et barre d'espace CHF190 760FF 4500FB x =....

6 MOIS D'ECHANGE STANDARD = GARANTIE DDC Sém.----------
Tous nos prix s'entendent TTC TOTAL: CHF FF FB.....
Port, Emballage et Envoi recommandé : CHF3 7FF 50FB.....

TOTAL PORT COMPRIS CHF FF .FB.....

Nom: . . . . . Prénom: . . . . . Pays: . . . . . .

Rue: . . . . . N°:. . . Code postal: . . Localité: . . . .

le journal des amateurs de programmation

Si programmer un ordinateur est devenu pour vous un loisir, un plaisir... une passion, sachez que **LIST** a été créé pour vous. **LIST** vous aide à tirer davantage de votre matériel, à vous perfectionner dans la conception des programmes qui "tourneront" sur votre machine.
LIST vous initie aux langages informatiques et sélectionne les meilleurs livres pour progresser. **LIST** vous informe de l'actualité et vous fournit trucs, astuces et idées pour mieux programmer...
Pour être sûr de ne rater aucun numéro et pour recevoir **LIST** chez vous, **abonnez-vous** !

FAITES 40F D'ECONOMIE !

LIST, LE PLAISIR DE PROGRAMMER

20F chez votre marchand de journaux

BULLETIN D'ABONNEMENT

à retourner à **LIST**
(service Abonnements)
5, place du Colonel-Fabien
75491 Paris Cedex 10

Nom : _____
Adresse : _____
Ville : _____
Code postal : |_|_|_|_|_| Pays : _____

Veuillez m'abonner pour 10 numéros au prix avantageux de **160 F*** au lieu de 200 F. Je fais ainsi une économie de 40 F sur le prix de vente au numéro.
Je joins mon règlement indispensable libellé à l'ordre de **LIST**.

* Belgique : **1330 FB** ; Suisse : **50 FS** ; Canada : **30 $ C** ; autres pays : **210 FF**.
Par avion : Afrique francophone : **245 FF**, Amérique, autre Afrique, Océanie : **305 FF**; Asie : **355 FF**.
Belgique : Soumillon, 28, av. Massenet, 1190 Bruxelles.
Versement Société Générale 2100405 835-39.
Suisse : 19, route du Grand-Mont, CH 1052, Le Mont-sur-Lausanne, versement Caisse d'Epargne et de Crédit, 10-2418 Le Mont CH 1052, compte courant n° 650 156-7.
Canada : LMPI, 9345, rue de Meaux, St Léonard (Québec), H 1R 3H3, Canada.
Autres pays : 5, place du Colonel-Fabien, 75491 Paris Cedex 10.

Allez plus loin avec P.S.I.

Des langages à découvrir...

Ces ouvrages vous permettront d'apprendre des langages passionnants comme le Logo, le Pascal, le C... Chaque ouvrage vous donnera une vue générale du langage, ses éléments, ses instructions, et les moyens d'exécuter des programmes.

Certains langages sont plus ardus que d'autres et nécessitent parfois une bonne connaissance de l'informatique et la maîtrise d'un autre langage, comme le Basic par exemple.

Référez-vous à la couleur de la série de chaque ouvrage pour en mesurer le degré de difficulté.

Programmer en L.S.E.
par S. Berche et Y. Noyelle
128 pages - **80,00 FF** - série verte*

Langages de programmation
par S. Berche et Claude Lhermitte
128 pages - **80,00 FF** - série verte*

Programmer en Logo
par F.X. Testard-Vaillant et J.-P. Régourd
208 pages - **100,00 FF** - série bleue*

Programmer en Assembleur
par Alain Pinaud
144 pages - **90,00 FF** - série bleue*

Démarrer en Forth
par Paul Chirlian
160 pages - **120,00 FF** - série bleue*

Programmer en Forth
par Alain Pinaud
160 pages - **90,00 FF** - série bleue*

Programmer en Fortran
par Daniel Jean David
128 pages - **80,00 FF** - série bleue*

Lisp sur Apple
par Nicole Bréaud-Pouliquen
112 pages - **80,00 FF** - série bleue*

Programmer en Pascal
par Daniel Jean David
et J.-L. Deschamps
160 pages - **90,00 FF** - série rouge*

Pascal UCSD sur Apple II
par Jacques Rouault et Patrice Girard
212 pages - **110,00 FF** - série rouge*

Programmer en C
par Claude Nowakowski
136 pages - **80,00 FF** - série rouge*

(*) **Série verte** : initiation
Série bleue : perfectionnement
Série rouge : approfondissement
Série noire : maîtrise de la méthode

Le Basic des initiés :

Le Basic de A à Z

par Jacques Boisgontier
176 pages - 110,00 FF
Une initiation au Basic vous permet d'assimiler très rapidement les notions fondamentales de la programmation. L'ouvrage se poursuit par : un dictionnaire des mots clefs du Basic Microsoft, TRS-80 et CP/M, permettant de retrouver rapidement la syntaxe d'une instruction, suivi de programmes de synthèse et de programmes utilitaires.

Le Basic et ses fichiers
(tome 1)
par Jacques Boisgontier
144 pages - 90,00 FF
Cet ouvrage s'intéresse à la programmation des applications utilisant des fichiers sur disquettes ou sur disques. La version de Basic retenue est le 5 de Microsoft, fonctionnant sous CP/M. Les utilisateurs de TRS-80 et d'ordinateur à microprocesseurs Z-80 et 8080 sont donc directement intéressés.

Le Basic et ses fichiers
(tome 2)
par Jacques Boisgontier
160 pages - 90,00 FF
Ce second tome est essentiellement consacré à des programmes soit utilitaires comme le générateur de saisie d'écran ou le tri rapide, soit de gestion comme la facturation ou la paie.

Le dictionnaire du Basic
par David A. Lien
480 pages - 195,00 FF
Référence de base et outil de travail indispensable, cet ouvrage contient la liste des 500 mots les plus importants du langage Basic « parlé » par les ordinateurs les plus diffusés, ainsi que leur définition et des exemples d'utilisation. Il procure les moyens d'adapter les différents « dialectes » Basic à chaque ordinateur permettant ainsi de l'utiliser au maximum de ses possibilités.

Logic Basic
par P. Sénicourt et M. Massiou
224 pages - 130,00 FF
Finie la programmation empirique en langage Basic : voici une méthode complète de programmation structurée, illustrée d'exemples concrets !
Après avoir expliqué la démarche d'analyse, les auteurs décrivent les nouvelles structures logiques à utiliser. Les principes de dessin des organigrammes et de rédaction des programmes sont repris dans un logiciel d'analyse logique qui diagnostique les incohérences des programmes écrits en Logic Basic. Chargé sur un ordinateur individuel, cet analyseur, écrit en langage Basic Microsoft, récompensera l'utilisateur par l'édition d'un organiprogramme en traçant sur le programme lui-même l'organigramme d'origine.

Le Basic Microsoft
par Ken Knecht
168 pages - 100,00 FF
Voici un ouvrage d'initiation au langage Basic le plus répandu : le Basic Microsoft 5.0.
Après avoir présenté le vocabulaire indispensable pour apprendre à programmer en Basic, l'auteur vous explique, exemples à l'appui, comment utiliser les chaînes de caractères, l'éditeur, les variables, les boucles, les fichiers, les opérations airthmétiques en Basic, etc. Cet ouvrage est complété par un index détaillé.

Clefs pour un micro

Destinés à se trouver en permanence à portée de la main, ces ouvrages vous permettront de retrouver rapidement les informations dont vous avez besoin lorsque vous vous trouvez devant votre ordinateur. Les « Clefs » c'est : la liste des instructions Basic commentées, les mnémoniques, les points d'entrée de la ROM Basic, des explications sur les variables systèmes. Et c'est également : les caractéristiques des principales extensions, une liste d'astuces pour mieux utiliser l'écran, les cassettes et les programmes en langage machine.

NOUVEAU

Clefs pour le Commodore 64
par Daniel-Jean David
128 p., **100,00 FF**

Clefs pour l'Apple II
par Nicole Bréaud-Pouliquen
144 pages - **100,00 FF**

Clefs pour Oric
(Oric 1 et Atmos)
par Emmanuel Flesselles
124 pages - **100,00 FF**

Clefs pour le TRS-80
(tome 1)
par Rémy Pineau
192 pages - **120,00 FF**

Clefs pour le TRS-80
(tome 2)
par Rémy Pineau
200 pages - **120,00 FF**

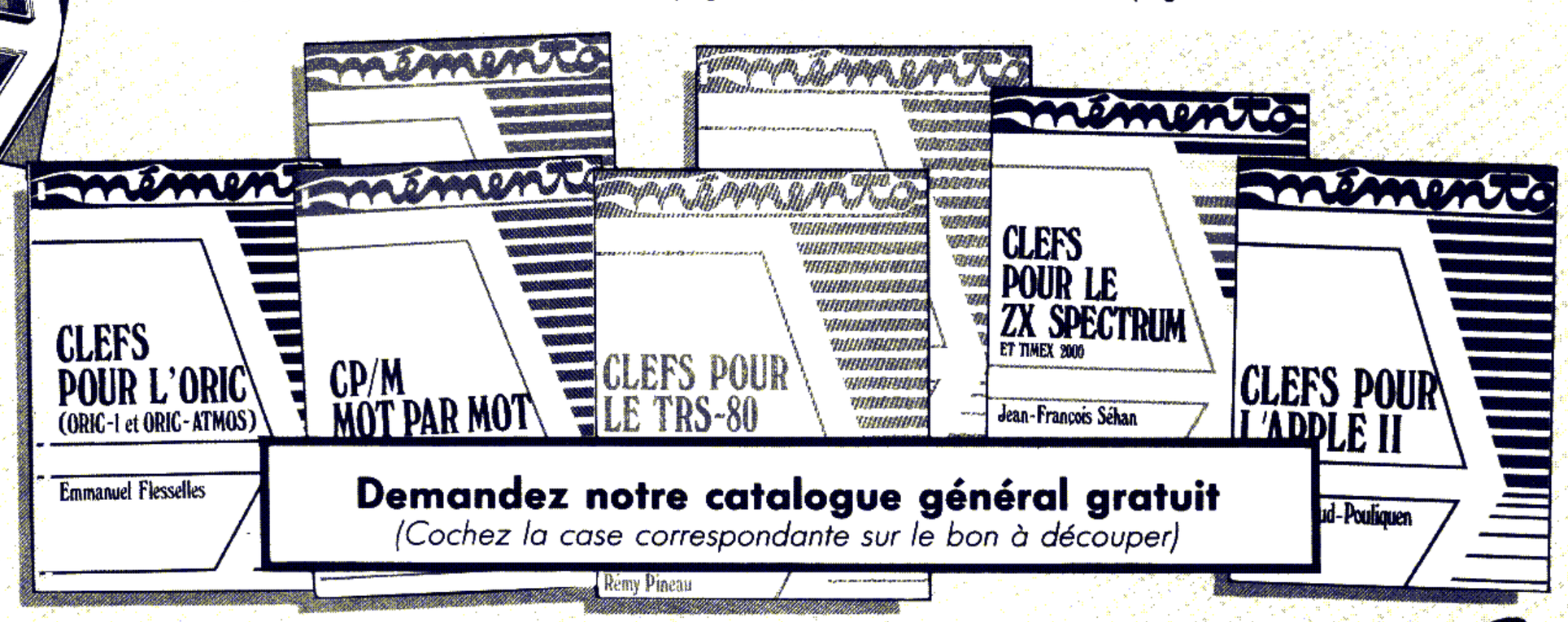
Clefs pour le ZX Spectrum
par Jean-François Sehan
112 pages - **90,00 FF**

Clefs pour ZX 81
par Jean-François Sehan
96 pages - **90,00 FF**

PC/DOS mot par mot
par Yvon Dargery
120 pages - **90,00 FF**

CP/M mot par mot
par Yvon Dargery
112 pages - **90,00 FF**

Clefs pour le Vic
par Daniel-Jean David
120 pages - **90,00 FF**



Demandez notre catalogue général gratuit
(Cochez la case correspondante sur le bon à découper)

P.S.I. DIFFUSION BP 86
77402 Lagny-s/Marne Cedex
FRANCE
Téléphone : (6) 006.44.35

P.S.I. BENELUX
5, avenue de la Ferme-Rose
1180 Bruxelles - BELGIQUE
Téléphone : (2) 345.08.50

P.S.I. SUISSE
Case Postale - Route Neuve
1701 Fribourg - SUISSE
Téléphone : (037) 23.18.28
C.C.P. 17 - 5684

au Canada : SCE Inc.
65, avenue Hillside Montréal
(Westmount) Québec H 32 1 W1
Téléphone : (514) 935.13.14

au Maroc : SMER Diffusion
3, rue Ghazza - Rabat
Téléphone : (7) 237.25

Envoyer ce bon accompagné de votre règlement à P.S.I. DIFFUSION ou pour la Belgique et le Luxembourg à P.S.I. BENELUX et pour la Suisse à P.S.I. SUISSE (voir table de conversions ci-contre).

DÉSIGNATION	PRIX
TOTAL	

LL 9

par avion ajouter 8 FF (75 FB) par livre

Pour la Suisse, frais de port pour tous les livres : 1,50 FS

Nom _____

Adresse _____

_____

_____ Code postal |_|_|_|_|_|

Ville _____

☐ ***Veuillez me faire parvenir votre catalogue général gratuit***

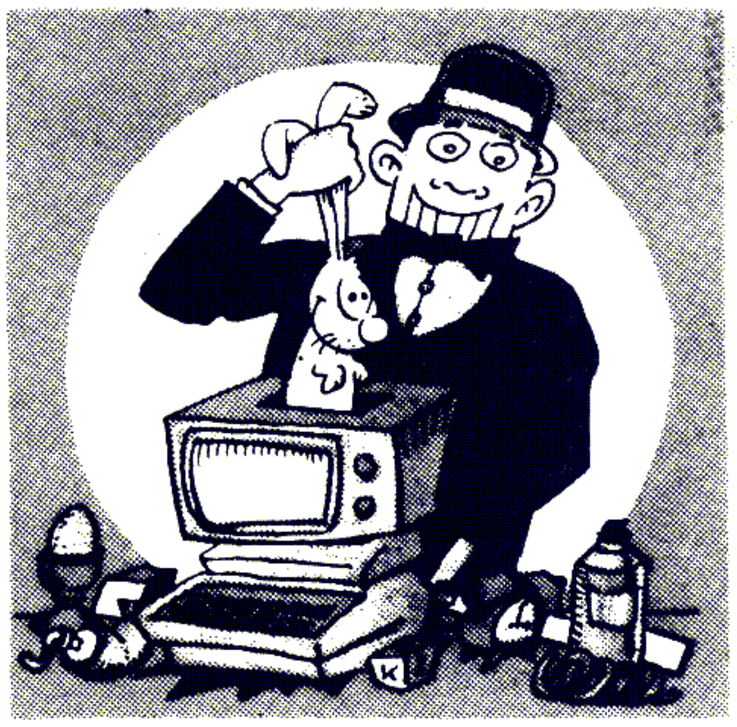
☐ *Paiement par chèque joint*

☐ *Paiement en FF par carte bleue VISA (à P.S.I. DIFFUSION uniquement)*

N° |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| *Date d'expiration* |_|_|_|_|

Signature (obligatoire pour paiement par carte de crédit)

Table de conversions en Francs belges et Francs suisses
80 FF = 620 FB - 25,30 FS
90 FF = 695 FB - 28,40 FS
100 FF = 770 FB - 31,50 FS
110 FF = 850 FB - 34,60 FS
120 FF = 925 FB - 37,60 FS
130 FF = 1 000 FB - 40,60 FS
195 FF = 1 500 FB - 60,00 FS



LA BOÎTE A MALICES...

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

TI-66

DEUX IDÉES A CREUSER...

Quand on tape un programme en position normale (255, 31), le temps de réponse est relativement long ; la frappe rapide est impossible. Ceci est dû au nombre de pas de programme. Plus il est grand, plus le temps de réponse augmente. Il faut donc changer de position. Essayez de taper rapidement en Part 32 : +1 = Pause RST, puis rentrez le même programme en Part 58, par exemple. C'est beaucoup mieux, non ?

Autre chose, moins utile certes, mais bizarre. Effectuez sur votre machine la séquence 2nd Part 37 2nd CP 2nd Part 64 2.61 STO 63 RST LRN. Pressez maintenant sur SST et un 2 s'inscrit à l'affichage ; SST une seconde fois et deux étoiles apparaissent. Cette "instruction" est inconnue du manuel. En remplaçant 2.61 par 2.71 dans la séquence précédente, on obtient un autre affichage anormal : *IN. Peut-être y a-t-il quelque chose à découvrir de ce côté-là...

Eric BERTREM

UN ÉCRAN VERT A PEU DE FRAIS

Beaucoup d'amateurs se contentent, en guise de « moniteur », d'un téléviseur noir et blanc plus ou moins ancien (et plus ou moins bricolé).

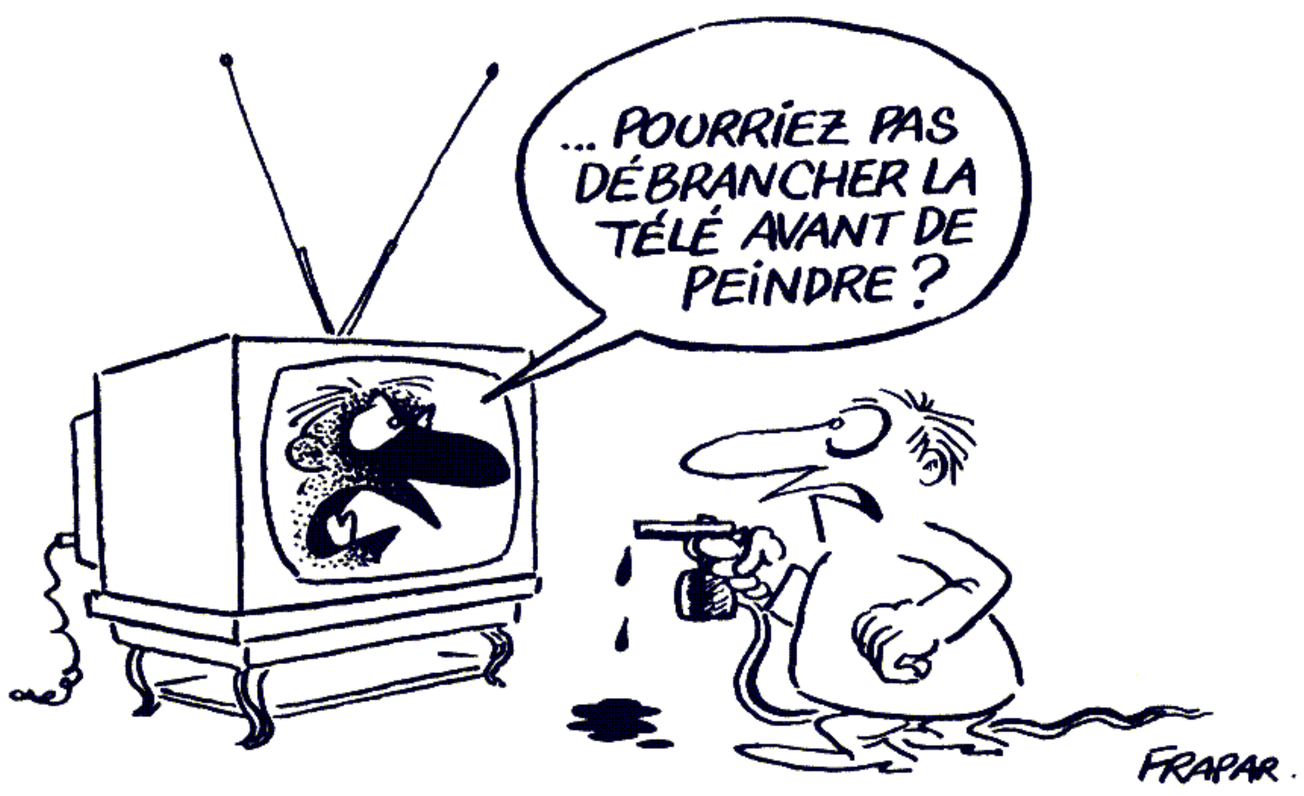
Il est assez facile de le transformer en écran vert ou ambre, à condition que ledit téléviseur soit définitivement dévolu aux activités informatiques. Il suffit... de peindre l'écran !

La meilleure façon de procéder est sans doute d'utiliser un pistolet à

peinture et de la laque transparente colorée (vernis « vitrail » que l'on trouve chez les marchands d'articles de dessin). On trouve aussi des vernis du même type conditionnés en bombes. Il faut opérer dans un local aussi dépoussiéré que possible et passer au moins trois couches de vernis dilué à l'aide d'acétone ou d'un solvant plus volatile que le white-spirit. Comme cela sèche très vite, il est possible de tout faire dans la même séance. Avec un peu de soin et une paire d'heures de travail, vous voilà détenteur d'un écran vert aussi agréable que les « vrais », et bien moins cher !

Cela dit, soyez prudent. Si vous n'avez jamais utilisé cette technique, prenez d'abord conseil dans votre entourage auprès de quelqu'un qui la connaît bien. Le pistolet (même à peinture !) peut avoir des effets désastreux...

Marcel LAGLASSE



TO7

UN PETIT RENUM

Le Basic du TO7 en version de base, si musclé soit-il, ne compte pas l'instruction RENUM (elle est fournie par le SED[1] aux heureux possesseurs d'un lecteur de disquettes). Cette instruction, très pratique, permet de renuméroter les lignes d'un programme de façon automatique.

Voici un petit programme qui implante en mémoire une routine en langage machine renumérotant les lignes de programme de 10 en 10. Après avoir indiqué le nouveau numéro de la première ligne en faisant POKE 34560, (L@256) puis POKE 34561, (LMOD256) où L représente le nouveau numéro, on lance le RENUM par la commande EXEC 32000.

```
100  CLEAR, 31999
110    FOR I=0 TO 22
120      READ OCT
130      POKE 32000+I, OCT
140    NEXT
150  DATA 142, 101, 245, 16, 190, 135, 0, 238, 132, 48, 2, 16, 175, 132, 31, 49,
     49, 42, 238, 132, 38, 243, 57
```

Après RUN, le petit RENUM est implanté et l'on peut donc effacer les six lignes de Basic.

Cet utilitaire, très court, ne corrige évidemment pas les branchements par GOTO ou GOSUB qu'il faudra donc rectifier après coup.

Jean-Paul CARRÉ

(1) SED : système d'exploitation de disquettes ; DOS en anglais.

ÉCRIRE : ÇA USE, USE...

Un petit truc pour les possesseurs de PC-1500 et surtout de l'imprimante CE-150 : comment économiser les quelque 45 francs nécessaires au réapprovisionnement de votre périphérique favori, bref, remplissez vous-même vos stylos !

Une encre de remplacement paraît tout à fait adaptée : celle des feutres classiques (genre Baignol & Farjon ou autre). Après avoir ouvert le petit stylo, ôter la feutrine usée et la remplacer tout simplement avec un bout de celle du feutre martyr. J'en suis à mon troisième rechargement de ce type sans aucun problème. De toute façon seul le stylo usé peut souffrir de l'opération (à l'ouverture par exemple), mais il était perdu pour l'informatique... Alors sans peur ni remords !

Et ceci vaut aussi bien sûr pour les imprimantes des ordinateurs Canon X-07, Casio PB-700, Sanco TP-8300, bref toute imprimante « tablette-traçante » qui équipe désormais un nombre croissant d'ordinateurs de poche. Attention toutefois, car cette manœuvre vous fait sans doute perdre la garantie qui couvre votre imprimante.

Gérard BEJOT

HP-41C

FONCTIONS CLASSIQUES

Les ordinateurs Hewlett-Packard possèdent la réputation méritée d'offrir à leurs utilisateurs de très nombreuses fonctions. La HP-41 C n'échappe pas à cette règle. Mais ses utilisateurs sont souvent à la recherche de nouvelles fonctions synthétiques ou provenant de modules... Et pourtant les bonnes vieilles fonctions du CATALOG 3 sont toujours là. Pourquoi les oublier ?

Ainsi pour SIGN, fonction méconnue de la HP-41 C, et cependant bien utile... SIGN, en mode programme, remplace le nombre placé en X par « 1 » si ce nombre est positif ou nul,

ou par un « −1 » s'il est strictement négatif. Quant au nombre, il se retrouve en LAST X.

Mais à quoi peut donc bien servir cette fonction ? Pendant des années, je ne m'en suis pas servi jusqu'au jour où j'ai découvert que cette fonction permet surtout de remplacer « 1 » quand on sait que le nombre en X est positif. L'avantage est énorme, surtout au sein d'une boucle répétitive. En effet, l'instruction « 1 » est exécutée en 7/100e de seconde contre 2/100e seulement pour l'instruction SIGN. Par conséquent, lorsqu'une boucle tourne 50 fois au sein d'un programme, vous gagnez 2,5 secondes... ce qui n'est pas mal pour le remplacement d'un octet par un autre.

Certaines dispositions de la pile permettent même des gains d'octets, quand par exemple le nombre placé en X devient gênant et qu'il faudrait de toute manière l'éliminer par un RDN ou par un CLX. Dans ce cas, SIGN remplace RDN 1.

Et CLX ? Combien de fois devons-nous utiliser des zéros : pour initialiser une mémoire, un registre de la pile, etc. Il suffit de remplacer « 0 » par « CLX », lorsque le nombre en X n'est pas précieux : 1,75/100e de seconde au lieu de 8/100e de seconde.

Mais cette même fonction peut servir à autre chose : par exemple, remplacer RDN. Cela peut paraître étonnant, mais CLX et RDN peuvent avoir un résultat strictement équivalent au sein d'un programme. Examinons ces deux petites routines : RDN RCL 01 et CLX RCL 01, et nous constaterons qu'elles sont identiques... sauf le fait que RDN perd 0,5/100e de seconde par rapport à CLX. Voilà un bon moyen d'accélérer certains programmes...

L'instruction CLX nous réserve encore une surprise : écrivons le petit programme suivant :
LBL « LIST », 1, ENTER, ENTER, ENTER, CLX, 3, END.

A la fin du programme, la pile se présente comme suit : T, Z et Y contiennent 1, et 3 se trouve en X.

Maintenant, intercalons un LBL 01 — instruction apparemment sans incidence — entre CLX et 3. Nous obtenons : LBL LIST, 1, ENTER, ENTER, ENTER, CLX, LBL01, 3, END.

N'est-ce pas le même programme ? Et pourtant... Nous avons maintenant 1 en T et en Z, 0 en Y et 3 en X. Que s'est-il passé ? A vous de le deviner... pour mieux comprendre CLX.

Autre fonction qui peut être très performante en certaines situations : %. Ainsi, si vous devez diviser un nombre par 100. Vous écrivez : 100 / (coût : quatre octets ; durée : 16,75/100e de seconde).

Alors, pourquoi ne pas remplacer ceci par : 1% ? Avec seulement deux octets, et 10/100e de seconde, l'avantage n'est-il pas évident ?

Passons maintenant à un parfait exemple de double emploi : exécutez R↑ ou RCLT, vous obtenez le même résultat : la pile se transforme ainsi :

a → T		b → T
b → Z		c → Z
c → Y	devient	d → Y
d → X		a → X

De plus, RCLT coûtant deux octets, il devient nécessaire de bannir cette fonction de tous vos programmes.

Mais cet exemple de double emploi est-il si parfait ? RCLT peut-il devenir irremplaçable en certaines occasions ? A vous de réfléchir... et si vous ne trouvez pas, voici une colle : à partir d'une pile ↷→T, ↷→Z, a→Y et b→X, obtenir une pile : a→T, b→Z, b→Y et a→X en quatre octets. Si l'on veut utiliser R↑, nous pouvons effectuer ENTER↑, ENTER↑, R↑ et STOT, mais cela prend cinq octets. Tandis qu'avec RCLT la solution tient en quatre octets seulement : ENTER↑, ENTER↑ et RCLT.

Encore une occasion de constater l'intérêt qu'il y a de posséder beaucoup de fonctions, même d'emplois proches.

Un petit problème pour finir : en X, un nombre à multiplier par 2 et toute une pile truffée de précieux nombres rendant impossible la séquence 2×. Alors, écrivez tout simplement ST+X...

Gilles BRANSBOURG

TI99/4A

MACRO-INSTRUCTIONS

Pour mieux gérer l'écran avec un TI99/4A, il faut concevoir de nouvelles instructions. Elles sont programmées ici sous la forme de petites routines, logeables en fin de programme. Pour y accéder, un GOSUB suffira en recopiant bien les REM. Bien sûr, le passage des variables est total, vous pourrez utiliser celles qui sont incluses dans les routines en dehors de leur exécution.

PRINT AT écrit le contenu de T$ sur la ligne L, à partir de la colonne C (le saut de la ligne est respecté).
CADRE trace un... cadre aux dimensions spécifiées par deux coins opposés du rectangle (L1,C1 et L2,C2). Cette fonction ne doit pas être utilisée pour générer un tableau, car le cadre occupe l'emplacement d'une ligne et d'une colonne à chaque case ; par contre, cela est très utile pour générer des fenêtres d'affichage intégrées à l'écran : on peut les colorer en les remplissant de caractères transparents dont le fond aura la couleur voulue, SPACE (ASCII 32) ou bien OBLITERATION (ASCII 127) par exemple.

LUTIN est très pratique pour déplacer du texte — ou du graphisme — à travers l'écran sans modifier le contenu de celui-ci : en superposition en quelque sorte. Pour être efficace, cette routine doit s'inscrire dans le programme principal en petits morceaux ; elle porte le caractère de code S à la ligne L, colonne C, puis le déplace à la vitesse horizontale VC et à la vitesse verticale VL (VC et VL prennent uniquement les valeurs +1 ou −1).

VALIDATE est très intéressant puisqu'il permet de saisir du texte n'importe où sur l'écran, et d'effectuer des commandes sélectionnées avec les touches de fonction : vous avez la possibilité de définir la réac-

Un PRINT à paramètres sans "scroll" vertical

```
100 REM  ----------------
110 REM  PRINT T$ AT(L,C)
120 REM  ----------------
130 FOR I=1 TO LEN(T$)
140 CALL HCHAR(L,I+C-1,ASC(SEG$(T$,I,1)))
150 IF I+C<32 THEN 180
160 C=1-I
170 L=L+1
180 NEXT I
190 RETURN
```

Avez-vous pensé à encadrer vos messages ?

```
100 REM  ------------------
110 REM  CADRE(L1,C1,L2,C2)
120 REM  ------------------
130 CALL CHAR(95,"00000000000000FF")
140 CALL CHAR(124,"0101010101010101")
150 CALL CHAR(125,"FF")
160 CALL CHAR(126,"8080808080808080")
170 CALL HCHAR(L1-1,C1+1,95,C2-C1-1)
180 CALL HCHAR(L2,C1+1,125,C2-C1-1)
190 CALL VCHAR(L1,C1,124,L2-L1)
200 CALL VCHAR(L1,C2,126,L2-L1)
210 RETURN
```

On a toujours besoin de ce SUPER-INPUT

```
100 REM  ----------------
110 REM  VALIDATE AT(L,C)
120 REM  ----------------
130 CALL KEY(5,R,E)
140 IF R>31 THEN 180
150 IF R<0 THEN 130
160 ON R GOTO 130,130,130,130,130,130,130,
    130,130,130,130,130,130,130,130
170 REM  PRINT R
180 CALL HCHAR(L,C,R)
190 C=C+1
200 IF C<32 THEN 130
210 L=L+1
220 C=1
230 GOTO 130
```

Un lutin averti en vaut deux

```
100 REM  --------------------
110 REM  LUTIN (L,C,S,VL,VC)
120 REM  --------------------
140 L=L+VL+24*((L<2)*(VL<0)-(L>23)*(VL>0))
150 C=C+VC+32*((C<2)*(VC<0)-(C>31)*(VC>0))
170 CALL GCHAR(L,C,R)
180 CALL HCHAR(L,C,S)
200 CALL HCHAR(L,C,R)
210 GOTO 140
```

Rangez soigneusement vos pages d'écran : afin de mieux les modifier

```
100 REM  ----------------------
110 REM  ECRAN T$(PAGE,LIGNE)
120 REM  ----------------------
130 DIM T$(6,24)
140 FOR L=1 TO 24
150 FOR C=1 TO 32
160 CALL GCHAR(L,C,R)
170 CALL HCHAR(L,C,30)
180 T$(P,L)=T$(P,L)&CHR$(R)
190 CALL HCHAR(L,C,R)
200 NEXT C
210 NEXT L
```

80 colonnes à la une, c'est plus confortable !

```
10Ø REM --------------
11Ø REM 8Ø COL(T$,L,I)
12Ø REM --------------
13Ø DISPLAY AT(L,I):SEG$(H$(P,L),I,28)
14Ø CALL KEY(5,R,E)::I=I-(R=8)*(I>1)-(R=9)
15Ø IF R>31 THEN H$(P,L)=SEG$(H$(P,L),1,I)&
   CHR$(R)&SEG$(H$(P,L),I+1,255)::I=I+1
16Ø GOTO 13Ø
```

La variable FACT contient son propre résultat

```
100 REM  --------
110 REM  FACT (N)
120 REM  --------
130 F=LOG(N/EXP(1))*N/LOG(10)+(LOG(2*PI*N))/(2*LOG(10))
140 FA=10^(F-INT(F))
150 FAC=(1/12+(1/288-1/373/N)/N)/N
160 FACT=(1+FAC)*FA
170 PRINT FACT;"E";INT(F)
180 RETURN
```

Table de l'alphabet

Ø97	ØØØØ38Ø47C847CØØ
Ø98	ØØ8Ø8ØF884847ØØ
Ø99	ØØØØØ7880807ØØ
1ØØ	ØØØ8Ø87888887CØØ
101	00003844F8403800
102	0018202078202000
103	00030483C084830
104	0040405864444400
105	1000301010102800
106	10003010101050Ø0
107	0040405060504800
108	30101010101028Ø0
109	0000685454545400
110	0000506848484800
111	0000344844241800
112	0000182424782020
113	00001824241A0404
114	0000586440404000
115	0000384030887000
116	0010103810100C00
117	0000484848483C00
118	0000424448502000
119	0000A4A4A4A85000
120	0000442810284400
121	0000442414082810
122	00003C0810207C00

Pour les voyelles accentuées, remplacez les 4 premiers zéros par :

- grave = 2Ø1Ø
- aigu = Ø81Ø
- circonflexe = 3844
- tréma = 24ØØ

tion de chaque touche du clavier (un argument de 3 dans CALL KEY au lieu de 5 bloque le clavier en majuscules) grâce à ON GOTO que vous devrez paramétrer à votre guise.

Après avoir inscrit votre texte sur l'écran, vous serez sans nul doute heureux de le sauver dans la variable adéquate : T$(PAGE,LIGNE), ou mieux T$(P,L). ECRAN stocke en 55 secondes le contenu de l'écran en y déplaçant un pseudo-lutin au fil de sa lecture. Vous saurez facilement apporter la modification qui restreint la taille de l'écran aux 24 lignes de 28 caractères utilisées par l'ordre PRINT.

Les utilisateurs du Basic étendu profiteront de la routine 80 COL (80 colonnes) en l'adaptant à leur goût ; celle-ci transforme une ligne donnée (L) en fenêtre d'affichage de 28 caractères sur une taille limitée (excusez du peu)... à 192 caractères. L'utilisation intensive de DISPLAY AT, qui est instantané, permet de rafraîchir l'affichage à un rythme élevé, d'où son double intérêt.

Si ces éléments vous permettent de réaliser un début de traitement de texte ou d'image — avec ECRAN vous pouvez stocker plusieurs pages d'écriture puis les mixer à l'aide de SEG$(T$,X,Y) —, vous aurez sans doute besoin de créer un alphabet avec minuscules accentuées... reportez-vous plutôt à la table que nous vous proposons ; cela vous évitera de longs moments pour traiter l'interlignage et l'accolement des caractères. Une fois ceux-ci programmés par CALL CHAR, on les obtient directement au clavier.

Pour finir, la petite routine FACT(N), résolument non-graphique, calcule la factorielle de N. On peut la délocaliser en faisant précéder les lignes 130 à 160 par DEF ; FACT prend alors automatiquement la valeur de n factorielle, au même titre qu'une variable.

Ces outils en main, attelez-vous à quelques réalisations pratiques, en préservant la portabilité de ces routines ; ce qui est acquis doit le rester...

Michel ARDITTI

UN TEST FACILE A ÉVITER

Au cours du déroulement d'un programme, il arrive souvent qu'une variable doive « basculer » entre deux valeurs différentes : elle sert alors d'indicateur ou de drapeau.

Ainsi, si la variable V contient la valeur a, alors elle doit recevoir une autre valeur, b, et si elle contient b, alors elle doit recevoir a. La traduction en Basic de cette opération est :

```
10 IF V=a THEN V=b : GOTO 30
20 V=a
30 Suite du programme
```

Mais, la traduction la plus économique, la moins gourmande en octets, est :

```
10 V=a+b-V
20 Suite du programme
```

Par exemple, si V doit « basculer » entre les deux valeurs a=0 et b=1, il suffit qu'elle passe par la ligne 10 : si elle valait 1, alors 1−V vaudra bien 0 et si elle valait 0, alors 1−V vaudra bien 1.

Un petit exemple d'utilisation permet d'illustrer le gain d'octets.

Sur l'écran d'un FX-702P, on voudrait voir passer une étoile de gauche (position CSR 5) à droite (position CSR 18) et inversement. Une première solution, en 46 pas, est :

```
10 P=18 : WAIT 20
20 PRINT CSR P ; "*" : IF P=18 ;
   P=5 : GOTO 20
30 P=18 : GOTO 20
```

La deuxième solution n'occupe plus que 32 octets :

```
10 P=18 : WAIT 20
20 PRT CSR P ; "*" : P=23-P :
   GOTO 20
```

Ce n'est pas plus compliqué que cela.

Michel SUSINI

PB-100 ÉTONNANT

Les réactions bizarres de votre PB-100 vous intéressent. Alors, commencez par saturer sa mémoire. Si vous ne disposez pas de l'extension, six lignes pleines en P1 suffiront (62 étoiles, par exemple) suivies de deux lignes du même type en P0. Si vous disposez de l'extension, il faudra plus de vingt lignes en P1. Une fois le remplissage effectué, détruisez la zone P0 par un CLEAR et presque simultanément, éteignez la machine. Rallumez-la et listez P1.

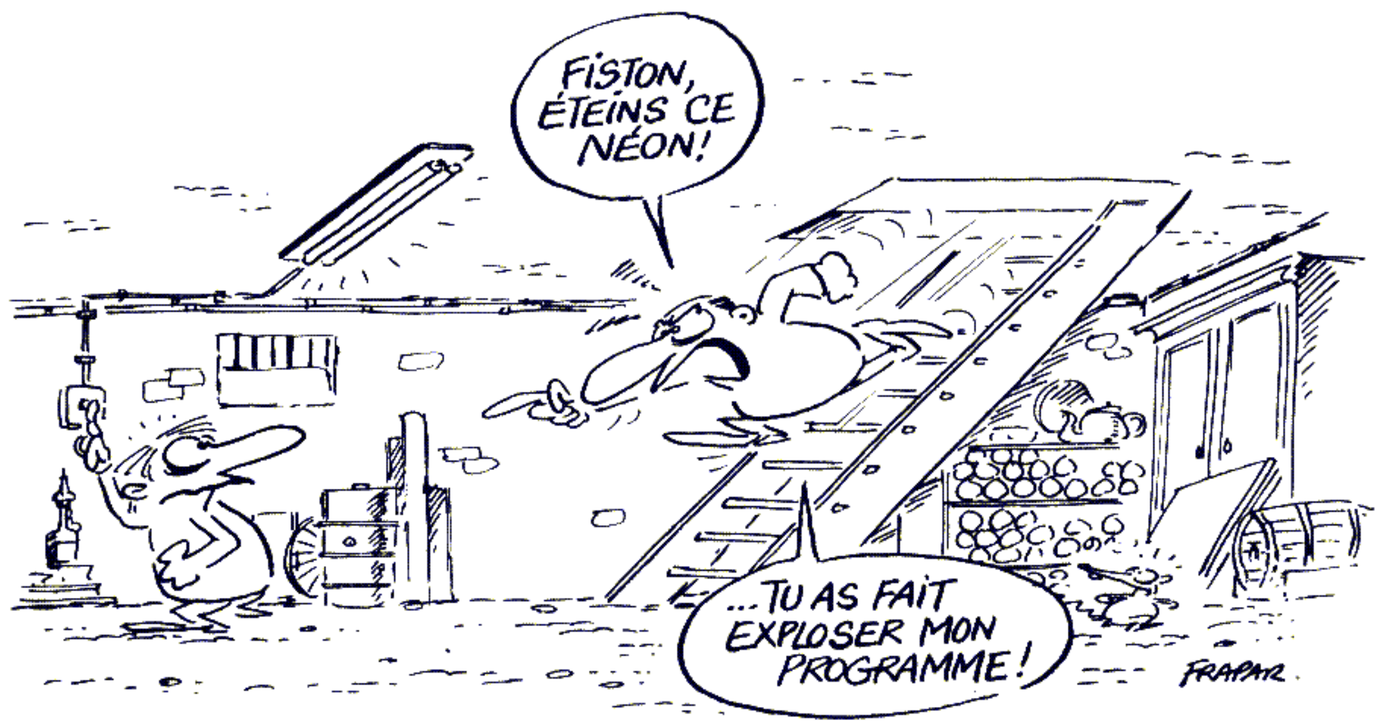
Selon que vous le ferez en mode 0 ou en mode 1, le résultat sera déjà différent. De plus, avec un peu de chance, vous verrez des choses bizarres à la première ligne. Tout dépend du temps qui s'est écoulé entre la pression sur EXE et l'extinction du PB-100. L'explication de ce phénomène est simple. Quand on demande la destruction de la zone P0, l'ordinateur transfère les zones suivantes en début de mémoire programme. En éteignant la machine, vous l'avez interrompue dans son travail. D'où les résultats bizarres.

Jérôme GAUDIN

ÉLIMINONS LES PARASITES

Le secteur véhicule des parasites et des micro-coupures qui s'avèrent bien gênantes lors de l'utilisation d'un micro-ordinateur, sans parler des franches pannes de courant. Pour ces dernières, à part l'alimentation de secours, il n'y a pas grand-chose à faire, non plus d'ailleurs que pour les micro-coupures, sinon... prier pour que les condensateurs de filtrage de l'alimentation soient assez grandement dimensionnés pour suppléer momentanément à l'absence de courant.

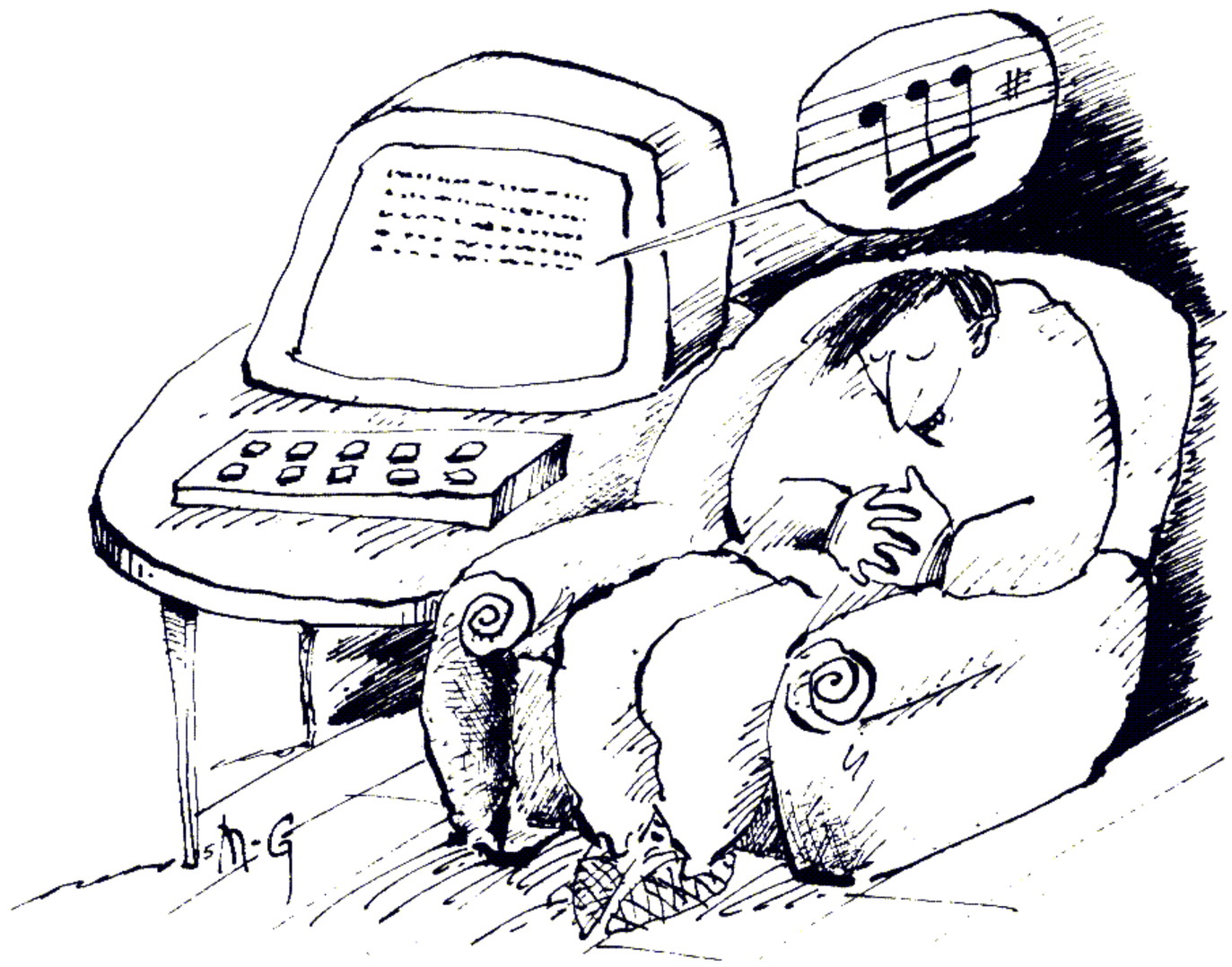
Restent les parasites. Vous savez : votre fils allume le tube au néon à la cave (le « ballast » est un peu poussif), juste en dessous du bureau, là où vous êtes en train d'entrer la 500e ligne d'un programme que vous n'avez pas encore sauvé sur mémoire de masse... Il y a une bonne chance pour que vous ayez droit à un magnifique RESET (et une conversation avec votre fils !).



Un moyen simple d'éviter ces désagréments est de monter, dans la prise secteur de l'ordinateur, un *suppresseur de parasites 250 V,* appelé encore SIOV, GMOV, etc. Demandez à un vendeur de composants électroniques : il vous procurera le modèle qui convient (coût : moins de 10 FF). Ce composant a la forme d'un disque d'un centimètre de diamètre. Il suffit de connecter les deux fils qui en sortent aux bornes de la fiche 220 V assurant l'alimentation de votre ordinateur. Le tout est assez petit pour se loger à l'intérieur d'une fiche et vous débarrassera des parasites qui peuvent traîner sur le secteur.

Alain MARIATTE

UNE VISITE EN FAMILLE



La série des CBM 4000 Grand Ecran comme la série des 8000 comporte une sonnerie incorporée en fin de ligne, comme sur une machine à écrire. A l'intérieur d'un programme, il suffit de faire PRINT CHR$(7) pour qu'elle tintinnabule, et CHR$ (135) pour qu'elle tintinnabule deux fois. EDEX 3.3 utilise cette particularité pour signaler la fin d'une opération, et quand l'ordinateur a de gros gros calculs à faire, ou des fichiers longuets à charger (surtout sur cassette !), c'est agréable de pouvoir faire autre chose en sachant que la bête vous avertira quand elle aura fini.

Mais on peut allonger la sauce en jouant de l'adresse 1004. A la mise sous tension, 1004 contient 16 : c'est le rythme auquel les sept notes qui composent la sonnerie sont débitées par la machine. Mettez-y une valeur plus grande (pas plus de 255, ça déborderait) par un POKE bien tempéré, et vous pourrez savourer note par note la douce mélodie du constructeur. Ce n'est pas du Satie, mais c'est nettement moins impératif que le drelin d'origine, et puis quoi ! il vaux mieux être attiré par la musique que de venir quand on vous sonne.

Des jeux et des touches

Pas de joyeustique, pas de paddeules, c'est depuis le clavier que vous dirigez bolides et missiles des jeux que vous inventez. Et vous testez le clavier soit par un GET soit par un PEEK judicieux à l'adresse de la dernière touche enfoncée (151 sur CBM, 203 sur VIC, 197 sur C.64). Mais vous vous heurtez aux raffinements de la machine : le tampon-clavier et l'anti-rebond.

L'anti-rebond fait que si vous maintenez la touche enfoncée, c'est comme si vous n'aviez appuyé qu'une fois. D'où crispation de l'index quand il faut se déguiser en machine à coudre si l'on veut faire trois pas à droite. La solution ? Faire croire à la machine que vous avez relâché la touche alors que c'est même pas vrai, heu-là ! L'adresse de la dernière touche enfoncée retourne le code de la dernière touche enfoncée, soit. Mais si on n'enfonce pas de touche, hein ? que retourne-t-elle ? Ben elle retourne un code qui veut dire qu'il n'y a pas de touche enfoncée (255 sur CBM, 64 sur VIC et C.64). Alors, de là à faire un POKE 151,255 ou un POKE 197,64 avant son GET, il n'y a qu'un pas. Essayez, vous verrez comme on est plus calme...

Quant au quiborde-beuffeure, pardon, au tampon-clavier, il fait que si dans la frénésie du danger on a frappé trois coups à gauche et qu'on s'aperçoive qu'il faut virer à droite, le programme, bête et discipliné, exécute les trois coups à gauche avant de virer à droite, mais c'est déjà trop tard, vous avez perdu ! Or, les machines les plus récentes ont un tampon-clavier à géométrie variable. Faites PRINT PEEK (1003) sur CBM 4000 GE et 8000, ou PRINT PEEK (649) sur C.64 (et aussi, je crois, sur VIC) : vous obtenez respectivement 9 et 10 ; c'est le nombre maximum de caractères que contient le tampon-clavier. Essayez FOR I=1 TO 10000 : NEXT, et pendant la boucle, appuyez une bonne vingtaine de fois sur une lettre. Arrêtez, attendez la fin de la boucle : 10 lettres apparaîtront sur l'écran. Faites POKE 1003,1 ou POKE 649,1, puis commencez l'expérience : une seule lettre apparaît. Voilà comment faire que votre jeu ne prenne qu'un ordre à la fois.

Maintenant, si votre meilleur copain a le dos tourné, faites POKE 649,0 : PRINT CHR$(147). L'écran s'efface, il y a un sympathique READY, le curseur clignote et... le clavier est mort. Sur VIC et C.64, RUN/STOP RESTORE vous tire d'affaire (à condition d'appuyer sur les deux touches à la fois) ; sur CBM, après un POKE 1004,0 on n'a plus qu'à allonger le bras pour atteindre l'interrupteur en criant vengeance.

Quel est le code ASCII de rien ?

A cette grande question, un certain nombre de bécanes répondent bêtement zéro. Le Commodore, depuis le PET jusqu'aux derniers-à-sortir-mais-qui-ne-sortiront-pas-à-moins-que..., reste intelligent : il vous signale que ce n'est pas possible en vous balançant dans les gencives un ILLEGAL QUANTITY ERROR humiliant. C'est particulièrement embêtant lorsqu'on lit un fichier où il peut y

avoir des chaînes vides. Alors il fallait faire des acrobaties du genre :

```
500 GET#88,C$ : IF C$ = "" THEN
A = 0 : GOTO 520
510 A = ASC (C$)
520 SUITE...
```

Or, les IF, quand on veut tasser cinquante instructions par ligne, ça vous met en l'air. Heureusement, si C$ n'est pas un caractère, mais une chaîne, ASC(C$) renvoie le code ASCII du *premier* caractère de la chaîne. Bon sang, mais c'est bien sûr ! Contentons-nous de faire :

```
500 GET#88,C$:A = ASC(C$ +
CHR$(0))
```

et ça colle : si C$ existe, c'est le premier caractère de C$ + CHR$(0), et on a son code, sinon, le premier caractère de la chaîne est CHR$(0), dont le code ASCII est... Je ne me souviens plus !

Tout ce qu'on impute au pet...

Toujours dans les fantaisies célèbres, l'INPUT du PET/CBM et celui du VIC.

Sur PET/CBM, un simple RETURN en réponse à un INPUT fait arrêter le programme. Solution connue, ouvrir un fichier clavier :

```
100 OPEN 1,0 : INPUT = 1,R$ :
CLOSE 1 : IF R$ = "" THEN 100
```

Mais on peut aussi faire croire à l'ordinateur qu'un fichier est ouvert sans l'ouvrir avec POKE 16,1 (sur 4000 ; sur 3000, c'est POKE 14,1 et sur PET, POKE 3,1). On n'a plus qu'à traiter les effets secondaires, absences de retour à la ligne et de point d'interrogation.

Sur VIC, INPUT « CECI EST UNE QUESTION DE PLUS DE 22 CARACTÈRES » ; N vous fera défiler du REDO FROM START autant que vous voudrez.

INPUT « ET CELA EST UNE QUESTION AUSSI LONGUE » ; R$ mérite d'être suivi d'un PRINT R$, pour voir. Essayez, vous m'en direz des nouvelles. Morale, si la question plus la réponse doivent dépasser une largeur d'écran, posez plutôt la question par un PRINT, allez à la ligne et faites votre INPUT sans message.

François J. BAYARD

CONVERSIONS D'ANGLES

Certains ordinateurs ne connaissent qu'une seule unité d'angle, le radian, et quatre fonctions trigonométriques : sinus (SIN), cosinus (COS), tangente (TAN) et arctangente (ATN).

Pour faire des calculs sur des angles connus en degrés ou en grades, il faut donc commencer par les convertir en radians. C'est facile si l'on sait qu'un demi-cercle forme un angle de π radians, de 180 degrés ou de 200 grades. Alors, un angle de A degrés est égal à un angle de $(A*\pi/180)$ radians, soit approximativement $(A*0{,}01745329)$ radians. De même, un angle de B grades est égal à un angle de $(B*\pi/200)$ radians, soit approximativement $(B*0{,}01570796)$ radians.

Le problème se complique (un tout petit peu) s'il s'agit de retrouver un angle dont on connaît le sinus ou le cosinus. Dans ce cas, l'angle peut être retrouvé, en radians, par la formule : $C = ATN\ (a/\sqrt{1-a^2})$ où a est le sinus de l'angle, donné au départ. Si c'est le cosinus qui est donné alors l'angle C en radians sera le résultat de $ATN\ (\sqrt{1-b^2}/b)$ où b est le cosinus donné au départ.

Il ne reste plus qu'à convertir l'angle C en degrés ou en grades, si c'est nécessaire.

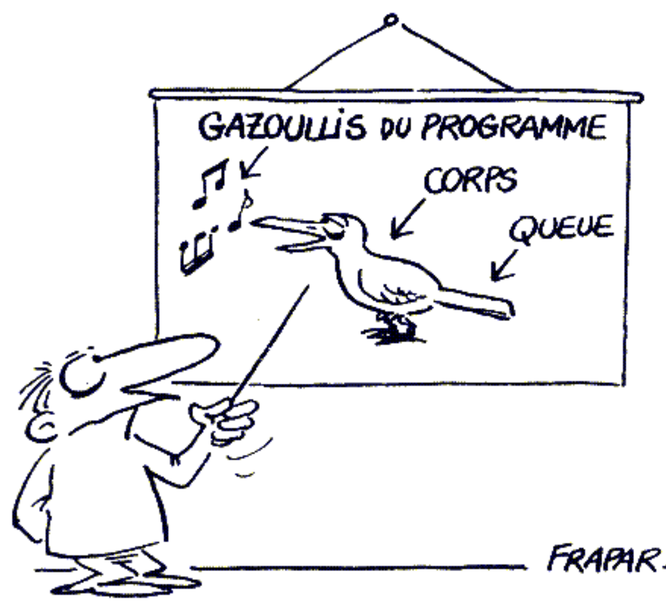
En degré, C devient $(C*180/\pi)$ soit approximativement $(C*57{,}29578)$ degrés et, en grades, C devient $(C*200/\pi)$ soit approximativement $(C*63{,}66197724)$ grades.

Ces résultats sont plus faciles à obtenir sur les ordinateurs de poche : toutes ces fonctions étant préprogrammées, quelques touches suffisent.

Anne-Sophie DREYFUS

CHARGEZ VOS PROGRAMMES EN QUATRIÈME VITESSE

Comme vous le savez sans doute, lors du chargement d'un programme, il se passe bien des choses dans l'ordinateur. Généralement, il lit l'avant-programme sur la bande, c'est-à-dire le sifflement continu qui précède le « gazouillis » du programme proprement dit, puis le corps de ce programme et éventuellement enfin, la « queue de programme » qui contient par exemple des indications permettant de vérifier que le chargement s'est bien déroulé.



La vitesse de transfert des octets est faible (300 à 1 800 bauds environ), ce qui fait que le processeur passe son temps à attendre les bits venant de la bande magnétique. Et pourtant, certains ordinateurs acceptent, une fois lue l'amorce de synchronisation, de « digérer » la bande à vitesse plus élevée. Pour ce faire, il faut évidemment disposer d'un magnétocassette à vitesse variable.

Essayez vite sur le vôtre et dites-nous le résultat de votre expérience. A titre indicatif, un DAI accepte de charger un programme trois fois plus vite que prévu.

Alain MARIATTE

TOUS POUR UN ONZE POUR TOUS

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 pourrait-il être l'ami du bien ?
Dans cette rubrique, les défis se succèderont : des programmes toujours plus courts, plus rapides...
Et les records tomberont !

Le programme à optimiser, défi lancé par Gilles Bransbourg dans *LIST* n° 1, est un classique du genre. Tout amateur de programmes de maths pour HP-41 C se doit d'en posséder une version dans sa mémoire : simplification de fractions. Après une lutte acharnée à coup d'octets, de millisecondes et d'astuces, nous n'avons pas un vainqueur mais... onze ex aequo !

Ces onze auteurs se partagent donc la satisfaction d'avoir réalisé "le" meilleur programme de simplification de fractions pour la HP-41 C :

E. Aubourg, A. Borsay, G. Bransbourg, F. Galichet, J. Hectus, A. Joux, A. Maucuer, A. Peruta, A. Terlinden, L. Tordjmann et O. Zimmer.

Il s'agissait de simplifier des fractions dont le numérateur et le dénominateur se trouvent en Y et en X. La méthode classique, employée justement par tous les lecteurs qui ont participé à *Misez p'tit*, est celle du calcul du PGCD (plus grand commun diviseur) des deux nombres en Y et en X. A la fin, la nouvelle fraction, simplifiée, est obtenue en divisant Y et X par ce PGCD.

L'algorithme du calcul est représenté (figure 1) en organigramme. Il est assez clair en lui-même en ce qui concerne les instructions du calcul, reste à examiner sa logique et, bien sûr, la manipulation de la pile opérationnelle.

Le calcul de Y modulo X est fondamental. Son résultat est nul si Y est divisible par X. A titre d'illustration le tableau de la figure 2 donne l'état de chacun des registres de la pile opérationnelle pour la simplification de la fraction 127/381.

```
01◆LBL "FRA
02 RCL Y
03 RCL Y
04◆LBL 00
05 MOD
06 LASTX
07 X<>Y
08 X≠0?
09 GTO 00
10 RDN
11 ST/ Z
12 /
13 END
```

Le programme (LBL T FRA) occupe 11 pas et 15 octets sans compter ni le LBL de tête ni le END final. Les temps d'exécution sont, pour les fractions données en exemple : 127/381 et PI/1, respectivement, de 0,305 seconde et 1,395 seconde.

Il sera difficile de mieux faire, n'est-ce pas ?

Jean-Christophe KRUST

Figure 1
Algorithme de simplification de fraction dressé par Arnaud Peruta. Y et X correspondent aux registres de la pile opérationnelle.

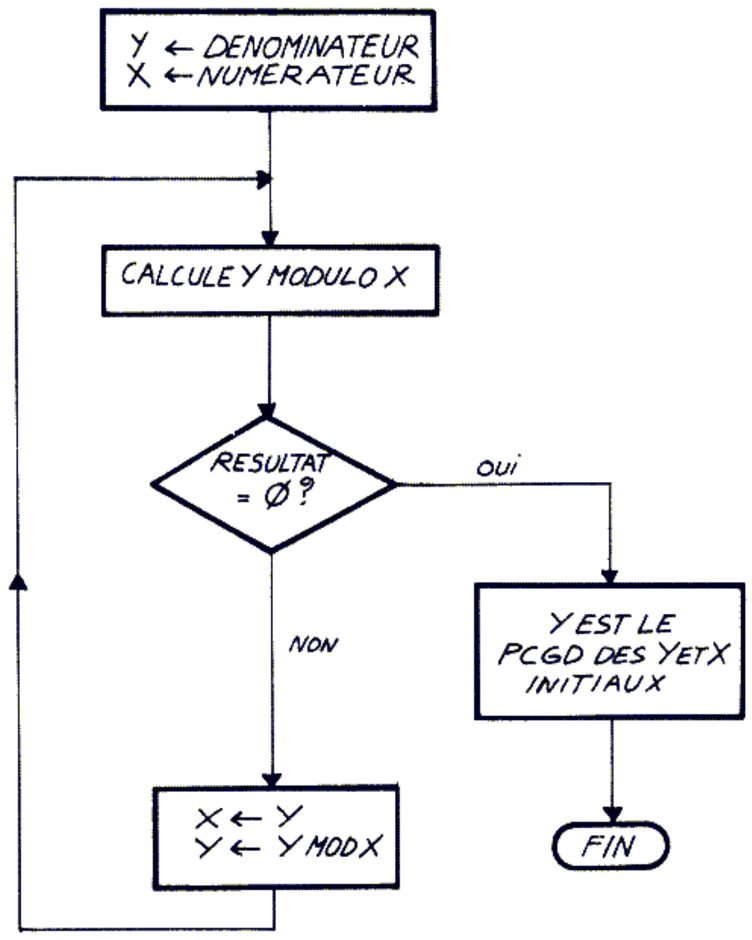


Figure 2
Etat des registres de la pile pour la simplification de 127/381

	X	Y	Z	T	Lastx
départ	*381*	*127*	–	–	–
RCL Y	**127**	*381*	*127*	–	–
RCL Y	**381**	127	*381*	*127*	–
LBL 00					
MOD	**127**	*381*	*127*	127	**381**
LASTX	**381**	127	*381*	*127*	–
X <> Y	**127**	**381**	*381*	*127*	–
X≠ 0 ?					
GTO 00 – LBL 00					
MOD	**0**	*381*	*127*	127	**127**
LASTX	**127**	0	*381*	*127*	–
X <> Y	**0**	**127**	*381*	*127*	–
X ≠ 0 ?					
RDN	127	*381*	*127*	0	–
ST/Z	*127*	*381*	**1**	0	–
/	**3**	1	0	0	–
fin					

En italique se trouvent 127 et 381, respectivement, numérateur et dénominateur de départ. En gras, le nombre résultat du calcul effectué par l'opération.

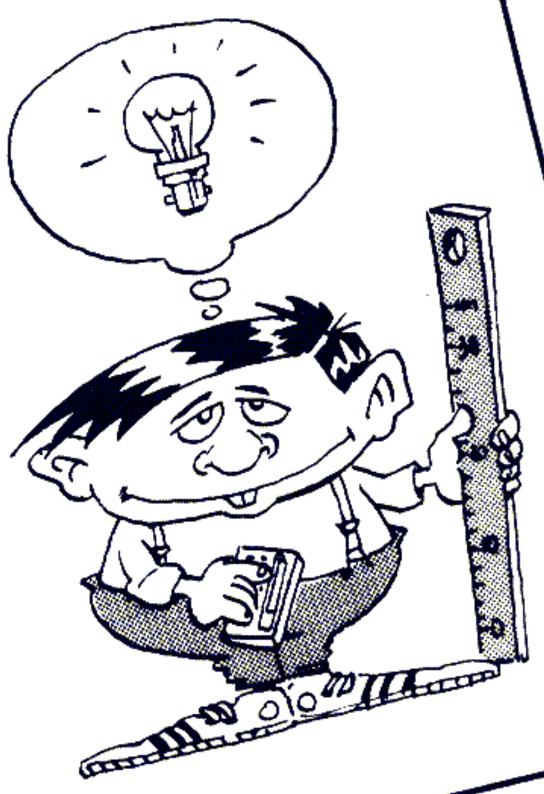
QUI DIT MIEUX ?

MODULO-CI, modulo-là... Dans un prochain numéro de *LIST*, la fonction MOD de la HP-41 C sera examinée sous (presque) toutes ses coutures. Mais dès à présent, le défi du mois (solution dans *LIST* n° 4) tourne autour de ce thème.

Considérons un très grand nombre, par exemple 5125. Quel est le reste de la division entière de 5125 par 3 ? Eh bien, il s'agit du chiffre 2. On dit que 5125 modulo 3 donne 2. De même 8152 modulo 3 vaut 1.

Mais essayez donc d'introduire le nombre 5125 dans la HP-41 C... Il faut ruser. Le défi du mois est donc de réaliser le programme qui simule la fonction MOD pour de très grands nombres. Les différents paramètres sont introduits séparément dans la pile opérationnelle (premier nombre, exposant et second nombre) et le résultat s'y trouve au retour.

Mon programme utilise 4 mémoires et 44 octets, ce qui est loin d'être idéal. Optimisons donc...



Gilles BRANSBOURG