LIGNES TIC

LE JOURNAL DES AMATEURS DE PROGRAMMATION

n° 3

OCTOBRE 1984

A L'ESSAI

- Le Basic du BBC vaut le détour
- Tablette Koala Pad, dessinez vos rêves les plus fous

LANGAGES

- Pourquoi Cobol reste le plus utilisé
- Découvrez C, un luxe à votre portée

LOGICIELS A L'ÉPREUVE

- Un Basic de course pour Spectrum
- Caractor traduit vos dessins en programmes TO 7
- Un assembleur complexe et plus puissant pour C.64

M2712 3 20 F

l'Ordinateur de poche

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

20 F

LIST

LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 3 - OCTOBRE 1984

SOMMAIRE

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

Ont collaboré à ce numéro : Olivier Arbey, Michel Arditti, François J. Bayard, Catherine Bellamy, Robin Bois, Bernard Bouvier, Christian Boyer, Thierry Chamoret, Raymonde Coudert, Jacques Deconchat, Thierry Demoy, Pierre Ladislas Gedo, Florence Gautier-Louette, Jean-Christophe Krust, Jacques Labidurie, Jean-Pierre Lalevée, Bernard Lambey, Alain Lavenir, Yves Leclerc, Jean-Charles Lemasson, Thierry Lévy-Abégnoli, Olivier Magnan, Christian Mildner, Philippe Moralès, Claude Nowakowski, Robert Pulluard, Marc Rivet, Fabrice Rogister, Christophe Taverner, Benoît Thonnart, René Tissrand, André Warusfel, Marc Wisniewsky.

Illustrations : Boredom, Philippe Burel, Chimulus, Philippe Delacroix, Frapar, Bernard Helme, Renée Koch, Alain Mangin, Alain Mirial, Pasty, Alain Prigent, Nicolas Spinga.

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

VENTES
Diffusion NMPP : 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

LE MICRO-
A EC

CANON X 07 :
BRANCHEZ VOTRE MICRO-ORDINATEUR SUR VOTRE TELEVISEUR.

IMPRESSIONNANT, LE CANON X 07 POUR UN MICRO-PORTABLE ! UNE INTERFACE OPTIONNELLE VOUS PERMET DE LE BRANCHER SUR VOTRE TELEVISEUR ET DE VISUALISER AINSI TOUTES LES OPERATIONS INSCRITES SUR VOTRE X 07.

MAIS LE CANON X 07 N'EST PAS SEULEMENT LE PREMIER MICRO-PORTABLE A ECRAN, IL EST AUSSI LE PREMIER MICRO-MULTICARTES.

SA FORCE ? DES PETITES CARTES EXTRAORDINAIRES POUR REALISER ET CONSERVER VOS PROPRES PROGRAMMES, COMME VOUS L'ENTENDEZ... A LA CARTE.

PRATIQUE, IL PARLE EN BASIC, LE LANGAGE ORDINATEUR FACILE A APPRENDRE.

AVEC SES NOMBREUSES CASSETTES ET CARTES A PROGRAMMES AUSSI ELABORES QUE LA GESTION DE STOCK, LA FACTURATION, LA PAYE, LE TABLEUR,... CANON X 07 A EGALEMENT BIEN D'AUTRES ATOUTS.

GRACE A SES MULTIBRANCHEMENTS : MACHINE A ECRIRE, IMPRIMANTE, ORDINATEUR, MODEM ET MEME VOTRE TELEVISEUR... CE TOUT PETIT ORDINATEUR A TROUVE PLUS D'UN MOYEN POUR DEVENIR GRAND.

JE SOUHAITERAIS RECEVOIR VOTRE DOCUMENTATION COMPLETE SUR LE MICRO-ORDINATEUR X 07.

VOICI MON NOM, MON ADRESSE ET MON TELEPHONE :

NOM _____

SOCIETE _____

N° _____ RUE _____

VILLE _____

CODE POSTAL _____ TELEPHONE _____

DEMANDE D'INFORMATION A RENVOYER A CANON FRANCE, 93154 LE BLANC-MESNIL CEDEX, TELEPHONE 865.42.23.

Canon

CANON, HAUTE TECHNICITE, HAUTE SIMPLICITE

CANON EST PRESENT AU SICOB : ZONE 4A, STAND 4101

-PORTABLE
CRAN.

LES 'MÉMENTOS' P.S.I. : les clefs de votre ordinateur.

PSI au SICOB
du 19 au 28 septembre
CNIT
SICOB Boutique.
Stand n° 30

Des livres destinés à se trouver en permanence à côté de votre ordinateur individuel tandis que vous programmez?
Des livres qui renferment toutes les informations que vous avez besoin de retrouver rapidement: syntaxes des commandes, codes caractères, message d'erreur, codes machine, adresses utiles...?
Des livres qui vous donnent des "trucs" très précieux?
Ces livres sont les MEMENTOS P.S.I., pour une utilisation quotidienne de votre ordinateur.

Clefs pour l'Oric
Oric 1 et Atmos
par Emmanuel Flesselles
116 pages - 100,00 FF

Clefs pour le ZX Spectrum
par Jean-François Sehan
112 pages - 90,00 FF

Clefs pour le ZX 81
par Jean-François Sehan
96 pages - 90,00 FF

Clefs pour le Vic
par Daniel-Jean David
120 pages - 90,00 FF

Clefs pour l'Apple II
par Nicole Bréaud Pouliquen
144 pages - 100,00 FF

Clefs pour le Commodore 64
par Daniel-Jean David
128 pages - 100,00 FF

Clefs pour Visicalc
par Jean-Louis Marx et Alain Thibault
104 pages - 100,00 FF

Le Basic de A à Z
par Jacques Boisgontier
176 pages - 110,00 FF

CP/ M mot par mot
par Yvon Dargery
112 pages - 90,00 FF

CHEZ VOTRE LIBRAIRE OU EN BOUTIQUE SPÉCIALISÉE

LM 10

P.S.I.

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 1
1701 Fribourg
SUISSE
Tél. : (037) 23.18.28
CCP 17.56.84

Au Maroc
SMER DIFFUSION
3, rue Ghazza
Rabat
MAROC
Tél. : (7) 237.25

Au Canada
SCE Inc.
65, avenue Hillside
Montréal (Westmount)
Québec H321W1 - CANADA
Tél. : (514) 935.13.14

Table de conversions en Francs belges et Francs suisses
90 FF = 695 FB - 28,40 FS
100 FF = 770 FB - 31,50 FS
110 FF = 850 FB - 36,40 FS

Envoyer ce bon accompagné de votre règlement à
P.S.I. DIFFUSION
ou, pour la Belgique et le Luxembourg à
P.S.I. BENELUX
ou, pour la Suisse à
P.S.I. SUISSE

DESIGNATION	NOMBRE	PRIX
	TOTAL	

par avion : ajouter 8 FF (75 FB) par livre
Pour la SUISSE frais de port sur tout envoi : 1.50 FS

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

Signature (obligatoire pour paiement par carte de crédit)

AGR4PH

☐ Je désire recevoir régulièrement le catalogue P.S.I.

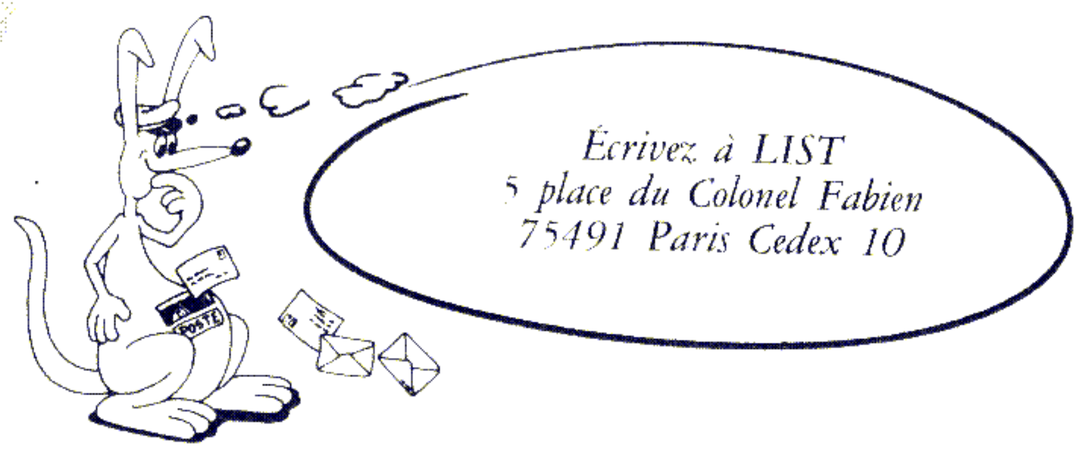
N° _____ Date d'expiration _____

NOM _____ PRENOM _____

rue _____ n° _____

Code postal _____ Ville _____

A VOS CLAVIERS



Toujours plus vite

A seule fin de montrer que j'ai pris plaisir à lire le numéro 1 de *LIST*, voici en quoi il m'a été utile pour mon FX-702 P.

Au départ, un convertisseur décimal-binaire avec un programme classique de divisions-soustractions successives par les puissances de 2, de 0 à 7.

L'ennui, c'était le temps d'exécution : plus de 4 secondes après l'entrée du nombre. L'idée des tests (page 87 de *LIST 1*), arrivait à point nommé pour une série de mesures. Sans entrer dans le détail, et uniquement pour ce qui concerne le programme, une boucle vide de 1 à 10 000 s'exécute sur le 702 en 2 mn 47 s. Bref (!), une vraie tortue. Comment faire plus rapide ? En supprimant la boucle, ça devrait aller mieux. Essayons...

C'est nettement plus long à écrire. Mais on gagne d'une bonne moitié en temps de calcul. Continuons. Voyons, en déclarant une matrice vide au départ, on peut supprimer les GOTO et les $ = $ + "O". Ce qui donne une troisième version de mon programme. Moins d'instructions dans la boucle de calcul, et donc gain de temps.

Peut-on faire mieux ? Oui, en paramétrant (*LIST*, page 66) avec une ligne à l'initialisation. Quatrième version. A l'exécution, on a 1,2 seconde de calcul maximum. Soit une division par 3 sur le programme initial. Je vous laisse tirer les conclusions.

Pour finir, j'ai également mis à profit l'astuce d'affichage de la boîte à malices (page 77) : comment "effacer" un blanc sur FX-702 P.

Jacques GUILLERON
29 Quimperlé

Pour trouver LIST

C'EST par l'intermédiaire de publicités dans des magazines que j'ai appris l'existence d'une nouvelle revue consacrée à la programmation.

J'ai essayé sans succès de me procurer *LIST* auprès de deux libraires qui, non seulement ne l'avaient pas mais encore ne le connaissaient pas.

Lors d'un passage à Paris, j'ai enfin pu trouver le premier numéro.

J'aimerais pouvoir l'acheter près de chez moi. M'est-il possible de commander *LIST* chez mon libraire ?

Annie AMELLE
03 Isserpent

■ *Bien sûr ! Il vous suffit pour cela d'indiquer à votre marchand de journaux le numéro NMPP de* LIST *qui se trouve sur le côté gauche de la couverture, c'est-à-dire M2712, afin que lui-même puisse le commander.*

Des avis partagés

Chers amis de *LIST*,

J'AI reçu les deux premiers numéros de votre (et mon) nouveau journal et je trouve que c'est une réussite.

En apprenant la nouvelle de ce changement, j'ai craint un instant de voir les ordinateurs de poche sacrifiés pour leurs frères aînés de table. Mais il n'en est rien et j'espère que mes craintes ne seront jamais justifiées.

Avec mes félicitations.

Luc THOMAS
50 Octeville

A la lecture de l'éditorial du numéro de mai de l'*Ordinateur de poche*, je me suis réjoui de l'ouverture aux ordinateurs familiaux. Il s'agissait d'enrichir le journal, non d'abandonner les ordinateurs de poche.

Quelle ne fut pas ma déception lorsque j'ai eu le n° 1 de *LIST* entre les mains. Je me trouve devant un magazine imprimé sur du papier jaune dont il faut faire trois fois le tour avant de trouver l'article qui nous intéresse, perdu au milieu de rubriques tout juste dignes d'une revue de vulgarisation. Je le regrette.

Claude LELIÈVRE
45 Châtillon-sur-Loire

JUSTE un mot au sujet de votre nouvelle formule *LIST* ; c'est vraiment la seule qui propose autant de diversité : programmation, essais, langages et poquettes. Tout ceci avec une pointe d'humour, ce qui ne gâte rien.

Eric CANOVAS
94 Saint-Maurice

LA transformation de l'*Ordinateur de poche* en *LIST* me fait craindre à plus ou moins longue échéance l'abandon des articles sur les petites machines au profit des ordinateurs familiaux.

D'autre part, je ne vois aucune différence entre votre revue et les autres magazines d'informatique. Quel dommage !

Denis WIRION
75011 Paris

APRÈS avoir apprécié l'*Ordinateur de poche*, j'ai constaté qu'avec *LIST*, vous n'aviez pas laissé de côté la

INDEX DES ANNONCEURS

A VOS CLAVIERS

HP-41 C. Cette machine fait des merveilles avec ses multiples combinaisons et ses trésors cachés.

Roger DELEPLANQUE
02 La Fère

DISPOSANT d'un Sharp PC-1251, je me permets de vous exprimer ma satisfaction en ce qui concerne votre revue.

Vincent FORICHON
59 Valenciennes

Nos dix tests et vos machines

JE suis heureux qu'une revue d'informatique pour les programmeurs soit enfin parue. Les magazines axés sur le matériel et les logiciels professionnels m'intéressent peu et les revues traditionnelles n'insistent pas assez sur la programmation.

J'ai apprécié en particulier vos dix tests. Je les ai appliqués à deux machines : l'extension ordinateur du Philips Vidéopac (G7400 + G7420) muni d'un Z 80 A et d'un Basic Microsoft, et mon Apple IIe.

Philippe LHOSTE
64 Orthez

JE viens de dévorer votre premier numéro. A propos des dix tests, Christian Boyer aurait dû ménager les nerfs des "Opéistes" en faisant aller les boucles de 1 à 1 000 au lieu de 1 à 10 000.

Quoi qu'il en soit, l'idée est bonne et j'espère que vous publierez les temps de très nombreuses machines. J'ai moi-même essayé les tests sur celles que je possède : PC-1500, PC-1245 et PB-100.

Jérôme GAUDIN
49 Angers

■ *Vous avez raison, mille tours suffisaient pour tester vos poquettes. Nous précisions d'ailleurs dans l'introduction de l'article que "pour certains ordinateurs", nos lecteurs devraient "procéder à des adaptations". Quant à présenter les temps d'autres matériels, nous avons bien l'intention de le faire aussi souvent que possible.*

Deux ans, c'est très, très long...

Chers amis de *LIST*,

PROFITANT de mon abonnement au successeur de l'*Op* (longue vie à *LIST* !), je me permets une petite suggestion qui sera sûrement appréciée par vos anciens et vos futurs lecteurs. La voici : pourriez-vous faire un index récapitulatif par machines et par rubriques des trois derniers numéros de l'*Op* dans le même style que celui que vous y aviez déjà publié.

Cela serait vraiment pratique et parachèverait la collection (complète dans mon cas) de l'*Op*.

Et bien sûr, faire de même avec *LIST* dès qu'un certain nombre de numéros seront parus.

Marc VAROQUI
57 Moulins-les-Metz

■ *Pour les 20 premiers numéros de l'*Op, *l'index récapitulatif se trouve dans le n° 21. Pour les nos 21, 22 et 23, nous ne prévoyons pas d'établir d'index pour le moment. La conception de* LIST *nous occupe déjà beaucoup. Mais peut-être un de nos lecteurs s'est-il constitué pour son propre usage l'index que vous recherchez. S'il veut bien vous l'adresser, il lui suffit de nous l'envoyer, nous vous le transmettrons.*

Nous reprendrons bien entendu l'idée pour LIST *(nous savons qu'elle a été très bien accueillie), mais nous n'attendrons pas 20 numéros avant de publier l'index des articles parus : deux ans, c'est vraiment très long !*

Basic étendu du TI 99/4A

C'EST avec intérêt que j'ai lu le numéro 1 de *LIST*. Toutefois, je suis surpris de voir un article sur le "Basic étendu" du TI 99/A4 à la fin duquel vous indiquez que l'on peut se procurer la cartouche et le manuel correspondant pour environ 500 F alors que, depuis plusieurs mois, j'essaie en vain de trouver ce module qui est effectivement intéressant.

Pouvez-vous m'éclairer et m'indiquer si l'on peut espérer le voir réapparaître sur le marché ?

Jean GOUZE
92 Boulogne-sur-Seine

■ *Si* Texas Instruments *a cessé la commercialisation du TI 99/4A et des logiciels s'y rapportant, le "Basic étendu" que nous présentions dans notre premier numéro devrait être disponible dès la fin du mois de septembre,* La Règle à Calcul *en reprenant la distribution.*

Le manuel joint à la version proposée par Texas Instruments *était en anglais. La cartouche est désormais accompagnée d'une notice en français, le tout pour 900 F.*

Les lecteurs de province pourront se procurer le logiciel par correspondance en ajoutant à leur règlement 30 F pour frais de port.
La Règle à Calcul
65/67 boulevard St-Germain - 75005 Paris - Tél. : 325 68 88

LA GAZETTE DE LIST

UN LIVRE

Programmation inventive
Xavier de La Tullaye
Éditions du PSI
Lagny, 1984
Broché, 160 pages
Prix : 95 FF

CE livre offre, avec un enthousiasme communicatif, une réflexion anti-pédante et très efficace sur l'activité du programmeur individuel. C'est un ouvrage d'initiation, certes, tourné en particulier vers les jeux et lisible par quiconque aurait déjà posé les doigts sur le clavier d'un Casio FX-702P, par exemple. Mais il apprendra sûrement aussi quelque chose aux plus chevronnés qui, après avoir lu ce livre, ne regarderont sans doute plus les mots *algorithme* ou *organigramme* tout à fait de la même manière.

A ce propos, l'auteur choisit son camp : il se veut un farouche défenseur de l'organigramme qui est parfois violemment critiqué par quelques théoriciens.

En refermant son livre, l'on s'aperçoit soudain que, sans douleur, à coups de blocs, d'arbres, de classements, d'architecture, nous savons tout sur la programmation str... (chut, le mot redoutable n'est écrit qu'une fois, en passant...). Un coup de force ! Avec, en plus, des programmes de tri à bulle, de chasse au sous-marin sur ordinateur de poche (et aussi en Basic Microsoft), et même de calcul du volume d'une sphère.

Un plaisir, quoi !

AW ■

Festival du logiciel
Triomphe pour un Transat...

DES idées sous le soleil. De la matière grise à foison au pied de la cité des Papes : le 2e festival du logiciel de Villeneuve-les-Avignon (certains l'écrivent : Villeneuve-lez-Avignon) a tenu la vedette près d'un mois durant auprès d'un public curieux, fidèle, pincé... Chauds les claviers !

De nombreux parrains à cette manifestation : le mensuel-ami *l'Ordinateur Individuel,* RTL, la Fondation de France, l'Agence pour l'Informatique, le Crédit Mutuel, etc. Bigre. La programmation sait déchaîner les passions.

Témoin, la passion selon Jean-Marc Sornin. 29 ans, géologue, il enlève le Grand Prix avec son *Transat,* pianoté à l'origine sur un... ZX81, il y a quatre ans. La version finale sur Apple est quand même plus musclée ! Quid ? un jeu. Pour passionné de courses à la voile, en solitaire face à l'écran. Vous quittez la Rochelle pour Halifax. A vous de déterminer le bon fret, celui qui n'alourdira pas trop les bateaux tout en garantissant aux équipages un approvisionnement suffisant en plein Atlantique.

Jean-Marc Sornin n'a pas prévu de courte paille en cas de disette. Son logiciel, en tout cas, ne vous laisse pas sur votre faim.

Le festival ne s'est pas privé de primer les professionnels. C'est *Pillage cosmique* d'Ediciel, jeu d'adresse sur Spectrum, qui a plu au jury de Villeneuve-les-Avignon. Ludique à l'honneur ? Pas toujours. Le prix de *l'Ordinateur Individuel* s'en est allé à Dominique Sola pour *Logodomi,* un didacticiel pour TI 99.

La liste des gagnants est encore longue. Préparez-vous. A l'année prochaine ! ■

UN LIVRE

Sharp PC-1500
Technical reference manual
Sharp corporation
Japon, 1983
Broché, 160 pages
Prix : 150 FF.

LE voici disponible ce « livre d'or » tant attendu, référence vitale pour tout fanatique du PC-1500 ! De la « carrosserie » aux logiciels, tout est parfaitement décortiqué au long des 160 pages du manuel qui n'a pas volé son titre de « référence technique ».

Sont étudiés le microprocesseur d'abord, riche de son jeu d'instructions, puis le port d'entrées/sorties LH 5811. Suivent les descriptions matérielles et logicielles de la petite bécane, agrémentées de quelques exemples de programme en langage-machine. En prime et en fin de manuel, les plans détaillés du PC-1500 et de ses périphériques. Le bricoleur pourra s'en donner

DU CÔTÉ DES CLUBS

Avis aux Oricophiles du Var

SI vous habitez Toulon ou ses environs, si vous possédez un Oric et que vous tient le démon de la programmation, vous pouvez échanger vos idées et vos découvertes avec d'autres amateurs.

Une quinzaine de jeunes de 16 à 20 ans sont déjà réunis au "Club Oric 83" ; ils vous y attendent à partir du 1er octobre courant.

Deux réunions mensuelles sont prévues, entre 14 et 16 heures, les premier et troisième samedis de chaque mois. Une participation annuelle de 30 F est demandée à chaque membre. N'hésitez pas à écrire ou à téléphoner.

Club Oric 83
7 avenue des Vignettes
83000 Toulon
Tél. : 16 (94) 41 42 23

Si vous êtes tenté par le Sharp MZ 700

CINQ passionnés ont créé à Boullay-Thierry dans l'Eure-et-Loir un club MZ 700. Ils proposent de faire découvrir à qui le leur demandera, les mystères de la programmation sur cet ordinateur. Vous pouvez vous joindre aux programmeurs qui correspondent déjà avec eux en leur écrivant à l'adresse suivante :

Monsieur LUCEAU
Club MZ 700
2 rue Saint-Lubin
28210 Boullay-Thierry
Tél. : 16 (37) 38 37 96

CARNET ROSE

Pour les mordus d'Alsace

DÈS la rentrée scolaire, habitants de Strasbourg et de sa région qui pianotez, vous pourrez vous retrouver dans un nouveau club.

Dans le cadre d'une association de quartier, le Micro-Club ARES propose aux débutants de les initier au Basic.

Informations, conseils aux nouveaux venus, mais aussi et surtout, rencontres entre branchés et mordus du microprocesseur.

Une participation de 60 F est demandée à chaque membre de l'association. Le montant de la cotisation à verser au Micro-Club n'a pas encore été fixé.

Les réunions auront lieu chaque jeudi à partir de 20 heures à l'*ARES*
10 rue d'Ankara
67000 Strasbourg
Tél. : 16 (88) 61 63 82

Dans la Haute-Garonne

NAISSANCE à Portet-sur-Garonne d'une association qui souhaite accueillir tous les passionnés d'informatique de poche.

L'Union pour la Promotion du Calculateur Programmable, c'est son nom, a pour but de promouvoir la pico-informatique chez les amateurs et les professionnels ainsi que de regrouper tous ses utilisateurs français.

L'association produit tous les deux mois un bulletin interne.

Pour de plus amples informations, vous pouvez contacter l'*U.P.C.P.*
18 avenue des Mimosas
31120 Portet-sur-Garonne

LA GAZETTE DE LIST

à cœur joie entre les schémas de « timing », les valeurs de résistances et autres tensions.

Il n'en demeure pas moins que ce manuel apparemment complet reste succinct et souvent avare de commentaires. Je conseille au débutant de préférer dans un premier temps une bonne bible du Z80 qui lui sera de la plus grande utilité pour aborder ce manuel Sharp, rebelle à l'initiation du néophyte.

JCL ■

DEUX LIVRES

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

LE manuel de Daniel Ranc sur le langage-machine et l'assembleur du TRS 80 fera le plaisir des "ludiques" plus sûrement qu'il ne servira les ambitions de ceux qui utilisent leur ordinateur pour écrire un nième 'biorythme". Il sera, plus généralement, la bible de tous ceux qui ont besoin de rapidité dans leurs programmes et, bien entendu, à la minorité agissante qui aime bien savoir "comment ça se passe à l'intérieur".

Il existe déjà beaucoup d'ouvrages, certains excellents, sur l'Assembleur et particulièrement sur le processeur du Z80. L'auteur s'attaque ici à une machine bien connue, le TRS 80, mais en se limitant malheureusement au Modèle I à cassette, aujourd'hui très dépassé. Court mais clair, il comporte quelques routines qui pourront se montrer utiles (classement par ordre alphabétique, gestion d'écran, affichage de chaînes...). On y trouvera notamment un bon chapitre sur la gestion des entrées/sorties.

C'est sans doute le manque de place qui a obligé l'auteur à rester incomplet. Bien entendu, le jeu d'instructions du Z80 n'y figure pas en totalité, mais il est facile de le glaner ailleurs (par exemple dans la documentation du programme Assembleur, éditée par EDTSAM et dont l'auteur présente rapidement le mode d'emploi). Une façon intéressante de faire la connaissance de l'Assembleur.

AW ■

CASSETTES ET DISQUETTES

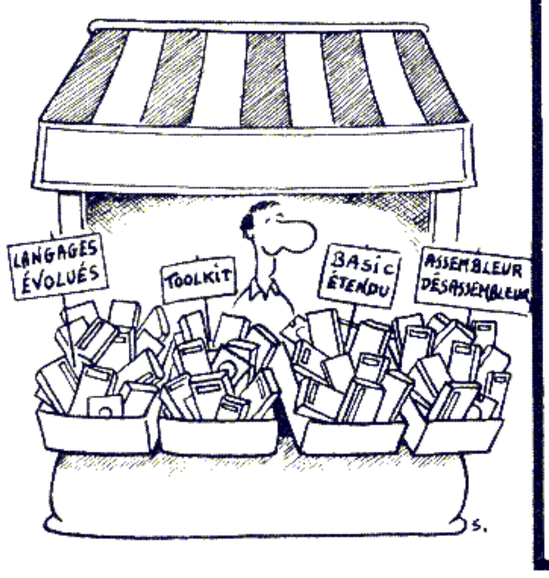
Moniteur désassembleur
Cassette pour Sega 3000
Édité par Loriciels
Prix : 180 FF

Forth
Cassette pour Spectrum
Édité par Artic
Distribué par Innelec
Prix : 295 FF

Zoom Pascal
Disquette pour Commodore 64
Édité par Micro Application Software
Prix : 490 FF

Oric Basic Plus
Cassette avec programme pour Oric-1 et pour Atmos
Basic étendu
Édité par ARG Informatique
Distribué par Innelec
Prix : 125 FF

Tasprint
Cassette pour Spectrum
Pour obtenir de nouvelles polices de caractères



Édité et distribué par Sémaphore
Prix : 105 FF

Spectrum Assembleur
Cassette pour Spectrum
Assembleur-désassembleur et moniteur
(Manuel en anglais)
Édité par Artic
Distribué par Innelec
Prix : 155 FF

Drag Bug
Cassette pour Dragon 32
Assembleur-désassembleur
Édité par Personal Software Services
Distribué par Innelec
Prix : 170 FF

Forth
Cassette pour Atmos
Édité par Tansoft
Distribué par Innelec
(Manuel en anglais)
Prix : 190 FF

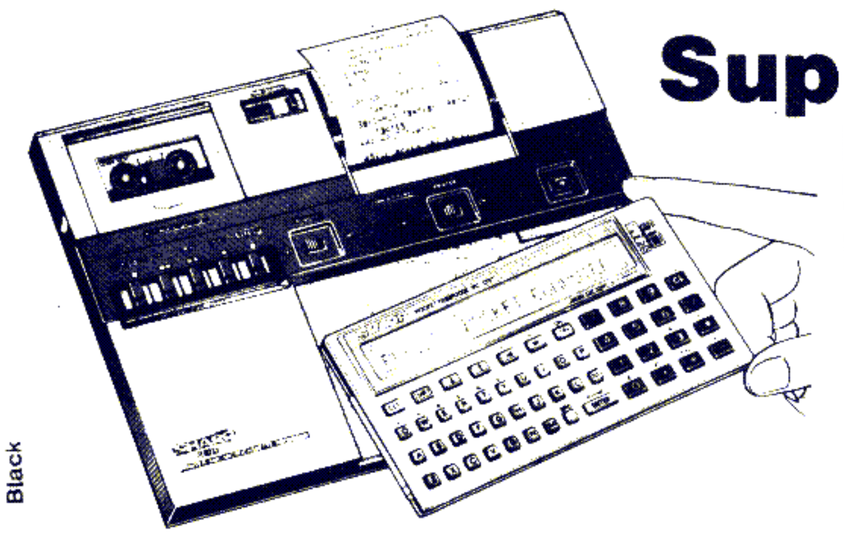
Moniteur 1.0
Cassette pour Oric-1 et Atmos
Assembleur-désassembleur
Édité par Loriciels
Prix : 140 FF

Pour mieux utiliser le Commodore 64
Méthodes pratiques
Jacques Boisgontier
Éditions du PSI
Lagny, 1984
Broché, 160 pages
Prix : 100 FF

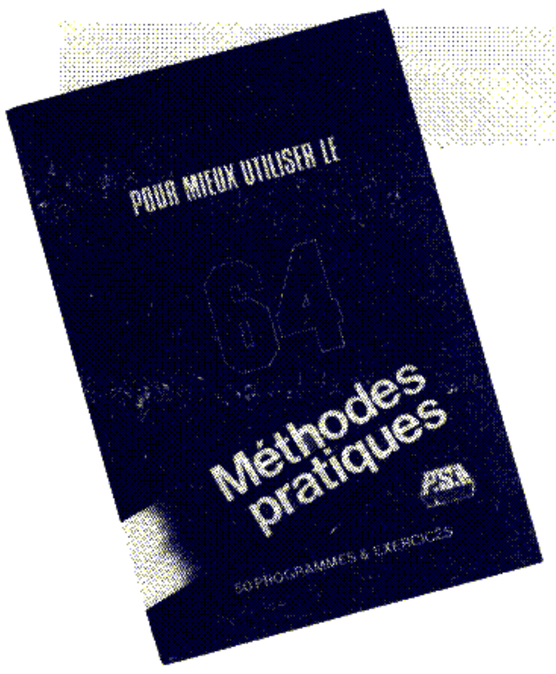
QUELQUE 50 programmes et exercices variés à mettre en œuvre sur Commodore 64 : les utilisateurs curieux de cet ordinateur-là apprécieront. Les "faux débutants" en Basic aussi. Faux débutants pour la bonne raison que cet ouvrage ne se veut pas vraiment un livre d'initiation à la programmation Basic.

Il exige quelques connaissances élémentaires, fussent-elles acquises ailleurs que sur le C.64.

On apprend pourtant beaucoup avec ces "méthodes pratiques" : une première partie passe en revue les instructions



LA GAZETTE DE LIST



Basic fondamentales, insiste sur les plus puissantes et les plus complexes, montre à l'usager les particularités propres au Commodore. Un exemple d'application, systématique, accompagne chaque instruction, un exemple de difficulté variable. Clarté et pédagogie gouvernent l'ensemble. Agréable à suivre.

A souligner : l'effort particulier accordé au graphisme haute résolution (avec ou sans extension "tool"), aux lutins ("sprites"), aux caractères programmables. Le tout servi de schémas clairs, bien conçus. Une excellente introduction au bénéfice des apprentis graphistes hésitant sur les voies à suivre. Mais une introduction seulement.

En revanche, silence (!) sur les possibilités musicales du C.64. L'auteur a peut-être tout simplement préféré faire l'impasse sur un thème méritant à lui seul un ouvrage entier.

Il a consacré, pour se rattraper, un nombre confortable de pages aux fichiers : sur cassette, sur disquette, séquentiels, relatifs, indexés. Les explications demeurent simples et claires, enrichies toujours de schémas bien pensés. Et de programmes d'application, bien sûr, pour chacun des thèmes abordés au long de cet important chapitre.

La deuxième partie, moins étoffée, rassemble quelques programmes de jeu. Essentiellement graphiques.

Enfin des programmes de gestion, simplifiés, forment une conclusion logique aux fichiers.

"Pour mieux utiliser..." s'acquitte de sa mission en complétant utilement la notice que Commodore livre avec ses machines. Les programmeurs du C.64, forts de leurs premières armes, y trouveront matière à progrès.

JPL ■

Interface parallèle pour Commodore 64 et VIC 20

POUR relier une imprimante équipée d'une liaison parallèle (type Centronics) à un C.64 ou à un VIC 20, nous avons trouvé une petite interface, à Strasbourg, plus précisément à Bischheim (4a, rue Nationale, 67800 Bischheim).

Son type : 92008. Elle se connecte, tout comme l'imprimante d'origine, sur le bus série « imprimante/floppy » du Commodore. Le port utilisateur est alors libre comme l'air.

Atout de la 92008 : sa mémoire-tampon de 8 000 caractères. Elle accélère confortablement l'exécution des programmes exigeant des impressions. Si tel n'est point votre souci, l'interface existe aussi sans cet ajout mémoire. Elle est alors du type 92000.

Livrée avec câbles d'entrée et sortie, elle est annoncée (pour la fin septembre) à 730 FF ttc environ dans sa version de base et les 8 Koctets de mémoire la portent à 910 FF ttc environ. ■

UN LIVRE

Spectrum Microdrive Book
Dr Ian Logan
Editeur Melbourne House,
Church Yard,
Tring, Hertfordshire
HP 23 5LU
Broché, 106 pages
Prix : 107 FF ttc

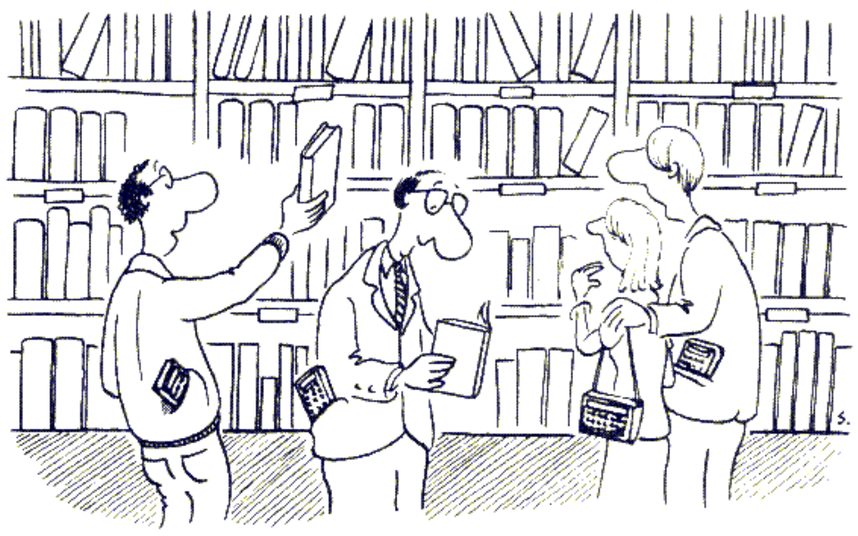
IAN LOGAN, Docteur Ian Logan, est un monsieur bien connu des « sinclairistes ».

Il a publié les désassemblages complets des programmes moniteurs du ZX81 et du ZX Spectrum. Il poursuit ici son « œuvre » avec ce Spectrum Microdrive Book, du reste point du tout réservé aux dites « microdrives », mais bien plutôt consacré à l'analyse des trois fonctions de l'interface ZX1.

Le « book » attaque fort logiquement par l'ossature matérielle de la ZX1 et la description de ses nouvelles fonctions. Mais le plat principal, ce sont bien les trois temps-clefs, la trilogie interface de la ZX1. Trois chapitres pour comprendre : « Microdrives », réseau et liaison RS 232.

Ian Logan décompose clairement la séquence des opérations que réalisent les nouvelles com-

UN PETIT TOUR CHEZ LE LIBRAIRE



Initiation à la programmation
Claude Delannoy
Editions Eyrolles
Paris, 1984
Broché, 192 pages
Prix : 90 FF

Le Traducteur Micro
Jean-Pierre Lamoitier
Lexique anglais-français des 1500 mots-clés de la micro-informatique
Editions Edimicro
Paris, 1984
Broché, 152 pages
Prix : 58 FF

Le Langage C
Jean-Louis Fourtanier et Violaine Prince
Editions Editests
Paris, 1984
Broché, 112 pages
Prix : 90 FF

Commodore 64, 66 programmes Basic
Stanley R. Trost
Editions Sybex
Paris, 1984
Broché, 202 pages
Prix : 78 FF

Programmer en C
Claude Nowakowski
Editions du PSI
Lagny, 1984
Broché, 136 pages
Prix : 80 FF

30 programmes pour Commodore 64
Dominique Lasseran
Editions Techniques et Scientifiques Françaises
Collection Poche Informatique
Paris, 1984
Broché, 128 pages
Prix : 35 FF

MO5 premiers programmes
Rodnay Zaks
Editions Sybex
Paris, 1984
Broché, 248 pages
Prix : 98 FF

Langage machine pour ZX81
Paul Sirven
Editions Radio
Paris, 1984
Broché, 176 pages
Prix : 75 FF

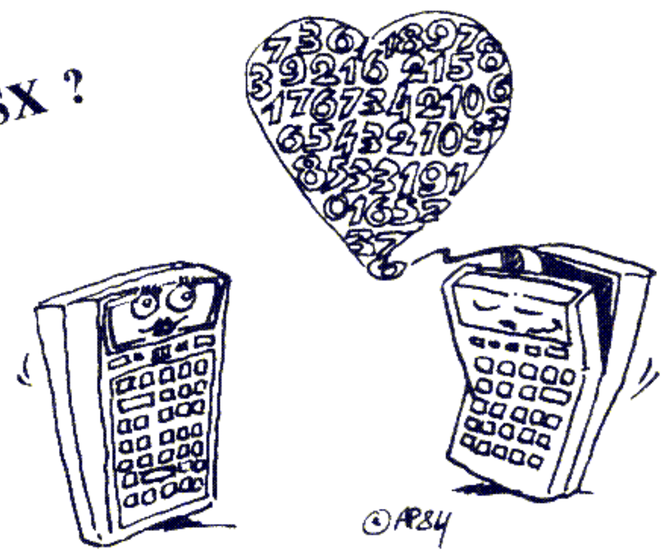
mandes. Une mention très bien pour les explications du contenu des « tampons » et les subtilités de « timing ».

Restait à se plonger dans l'utilisation du langage-machine avec l'interface ZX1. A ce stade, une surprise. Si Logan s'acquitte avec précision du décortiquage des fonctions de 23 « hook codes », un curieux « imprimatur » scelle le 24e, sans nul doute le plus intéressant, par une simple et pudique mention : « réservé à Sinclair Research Ltd » ! Explication de l'auteur sous forme de note : c'est Sinclair Research qui a expressément demandé que les détails propres à compromettre l'intimité, la « sécurité » des « microdrives » ne soient pas publiés. Force nous est donc de tenter de lire entre les lignes pour en extraire certains éléments, à peine suggérés !

Un « blanc » surprenant, impuissant toutefois à battre en brèche la valeur de l'ouvrage : il demeure essentiel pour qui veut utiliser au mieux son interface ZX1.

BT ■

Aimez-vous MSX ?



PAR MSX comprenez « nouvelle vague » : 20 constructeurs ont prêté serment d'allégeance au nouveau venu du standard familial 8 bits. Plus de 25 modèles recevront l'appui tactique de 220 logiciels ! Soit près de 60 000 machines construites chaque mois : ces précisions témoignent de l'envergure donnée à cette gamme d'ordinateurs.

Le pari : assurer une compatibilité totale entre tous les ordinateurs à venir. Pour cela, définir un standard matériel et logiciel ; en l'occurrence, le Basic choisi est une version étendue du MS.BASIC 4.5 inscrite en 32 Koctets de mémoire morte, utilisant une précision de 14 chiffres pour les calculs, des commandes graphiques complètes (Draw, Circle, Line, Paint, Color, Sprite, Screen...) et sonores (Play...).

Le matériel est bâti autour d'un processeur central Z80A et d'un processeur graphique TI TMS 9918A, celui-ci gérant 24 lignes de 32 ou 40 colonnes de texte (256 × 192 points) en 16 couleurs. Le générateur sonore GI AY-3-8910 standard connaît 8 octaves de fréquences.

L'originalité du système repose dans quatre slots d'extension mémoire : chacun accepte quatre pages de 16 Koctets de mémoire vive supplémentaires ou bien une extension de quatre slots secondaires. Donc un ordinateur MSX gérera un maximum de 16 × 64 Koctets de mémoire vive !

La norme MSX désigne aussi un système d'exploitation : MSX-DOS, qui possédera une interface de compatibilité avec CP/M 80 permettant l'accès à la plus vaste bibliothèque de programmes pour ordinateurs Z80. Une éventuelle compatibilité avec MS-DOS, le standard des 16 bits, serait envisagée. Il existe également un MSX Disk Basic de 16 Koctets de mémoire morte, indépendant de MSX-DOS.

Toutes ces caractéristiques préfigurent une génération de machines bas de gamme de hautes performances à usage essentiellement ludique, mais ouvertes à toutes les applications sérieuses non-professionnelles.

MA ■

TABLETTE GRAPHIQUE

KOALA PAD

PRODUIT d'Outre-Atlantique, la tablette graphique Koala Pad est importée en France depuis peu. Elle est pourtant un élément passionnant pour la mise en œuvre des capacités graphiques du Commodore 64. D'autres versions existent aussi pour Vic 20, Apple, Atari et IBM PC. La Koala Pad est distribuée par BIP. Son prix : environ 1 300 FF ttc.

Facilement extrait de son emballage, Koala (version pour C.64) se présente sous la forme de trois éléments distincts qui sont, par ordre de taille :

- la tablette graphique et son stylet,
- la documentation,
- la disquette contenant le logiciel de dessin.

L'examen de ce contenu révèle donc que la possession d'un lecteur de disquettes est indispensable.

Cette fameuse tablette, clé de voûte du système Koala se présente extérieurement comme un boîtier de plastique blanc. Sa face supérieure noire lui confère l'aspect d'une ardoise d'écolier ; ressemblance encore accentuée par la présence du stylet à pointe arrondie (en plastique également).

Toutefois, les formes futuristes, la présence de deux touches noires et d'un câble de liaison incongrus sur une ardoise d'écolier laissent deviner une utilisation quelque peu différente de celle imaginée au premier abord.

La tablette graphique a donc un aspect bien sympathique, sinon familier. Mais ses dimensions réduites (11 × 11 cm) ne sont-elles pas un inconvénient ? Sa précision est-elle suffisante ? A-t-elle une bonne sensibilité ?

Pour apporter une réponse à toutes ces questions, l'utilisation du logiciel fourni avec la tablette s'impose. Mais avant tout, consultons...

La documentation se réduit à un livret petit format de 25 pages traitant du mode d'emploi du logiciel, plus un dépliant de sept pages concernant l'utilisation de la tablette proprement dite.

D'emblée, c'est une mauvaise surprise : ces deux documents sont en anglais ; et malgré une fouille en règle de l'emballage, nulle traduction française, même sommaire, ne montre son nez... Moi qui espérais pouvoir confier l'ensemble aux mains inexpertes de ma fille, me voici condamné à lui apprendre d'abord l'anglais !

Mais, qu'à cela ne tienne : je vais auparavant travailler pour mon propre compte. Ces deux documents paraissent clairs et bien conçus... Dieu merci, j'ai conservé quelques restes d'une lointaine scolarité laborieuse !

Le logiciel Koala Painter, troisième et dernier élément, se présente sous la forme d'une disquette contenant le logiciel de dessin. A en juger par les photos d'écran qui illustrent l'emballage du Koala, ses possibilités semblent pour le moins intéressantes. Le chargement du programme fait d'abord apparaître sur l'écran une image haute résolution (pas mal !) tandis que la mise en mémoire se poursuit pendant près de deux minutes. Enfin, un cliquetis se fait entendre du côté de la disquette, indiquant que le logiciel est protégé contre la copie, et un menu — en anglais, encore ! — s'affiche alors à l'écran (voir photo ci-dessous). Nous voici donc à pied d'œuvre.

La tablette, connectée à l'ordinateur, remplit son office à merveille : une pression du doigt ou du bout du stylet sur la surface sensible fait apparaître sur l'écran une flèche que l'on déplace à volonté. Elle obéit, c'est le cas de le dire, au doigt et à l'œil. Le clavier du Commodore 64 devient parfaitement inutile : voilà qui fait irrésistiblement penser à la souris chère à Apple ! Une pression sur l'une des touches de la tablette et l'élément de menu est aussitôt sélectionné : commande de tracé, aspect du pinceau, choix des couleurs ou gestion de disquette. Simplicité et efficacité remarquables.

Une facilité étonnante

Après quelques tâtonnements et quelques coups d'œil sur la notice, le résultat prend forme et, ma foi, l'œuvre a belle allure ! Un détail qui cloche ? ZOOM sur le défaut. Une bavure inconsidérée ? OOPS immédiat. Une couleur à choisir, une ligne droite à tracer, un cercle, une boîte, une surface à remplir ?

Tout s'effectue en un clin d'œil, avec une facilité étonnante et le résultat est excellent. Huit formes et tailles de pinceaux sont disponibles. Quant à la palette, elle comporte 32 couleurs (les 16 couleurs de base du Commodore 64, plus les mêmes couleurs tramées). Parmi les 17 commandes que l'on peut sélectionner, signalons STORAGE, seule commande non graphique, qui permet de conserver et de récupérer sur disquette un dessin achevé ou à terminer plus tard. Cette même commande sert aussi à formater une disquette neuve.

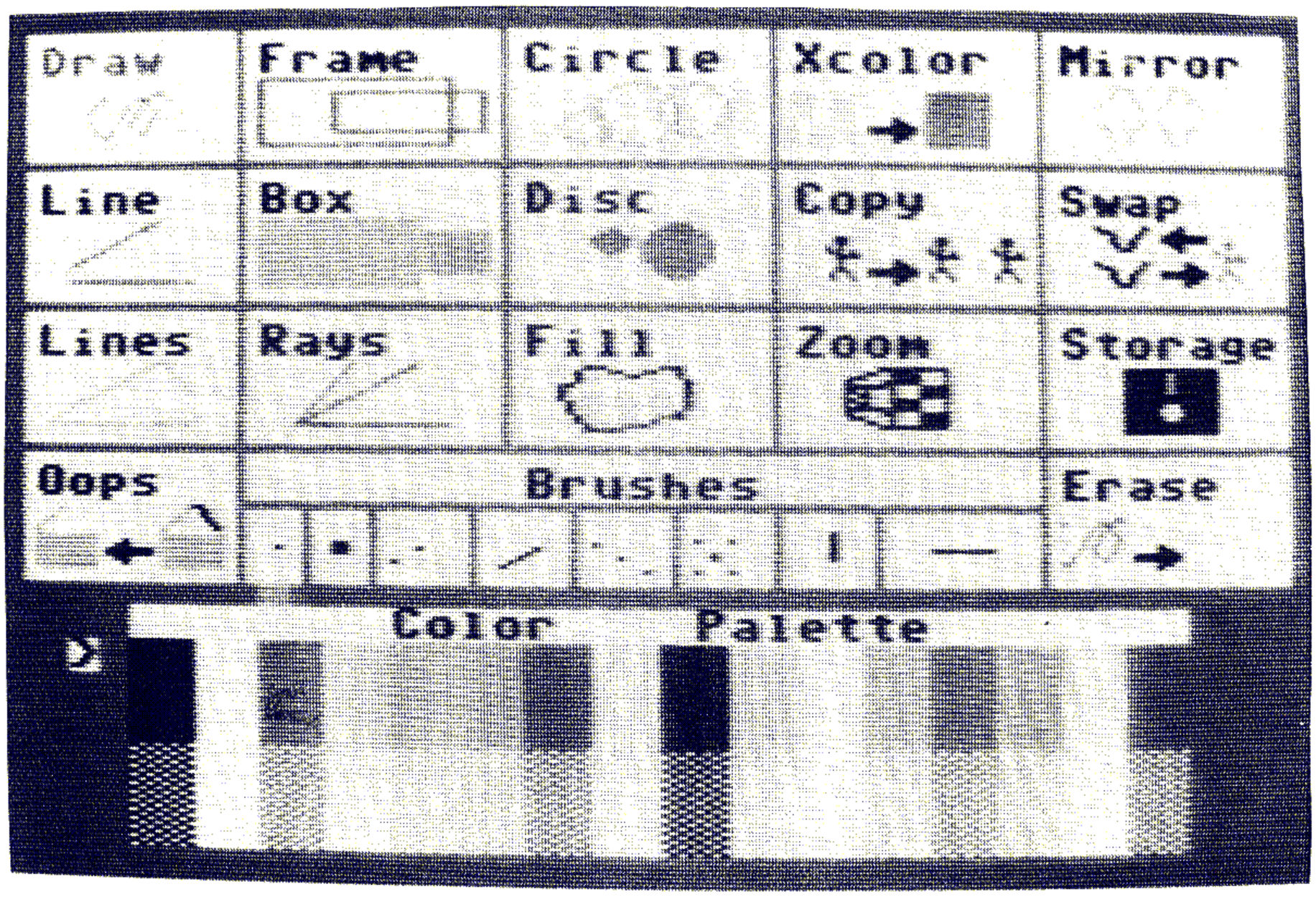
La disquette programme contient d'ailleurs quelques dessins déjà réalisés. Il est possible de les reprendre intégralement ou d'en recopier quelques éléments à insérer dans une œuvre personnelle.

Ces petits tours de passe-passe sont rendus possibles par l'existence de deux pages-dessin indépendantes l'une de l'autre et sélectionnables à volonté, les éléments de l'une pouvant passer sur l'autre grâce aux commandes SWAP et COPY.

Il suffit de quelques minutes de manipulation pour se familiariser avec ce nouveau périphérique. Chaque commande figurant au menu est accompagnée d'un dessin explicatif utile à qui ne connaît pas l'anglais (ou ne sait pas lire !). Et si la notice reste hermétique, quelques tâtonnements permettent de saisir rapidement comment fonctionne le tout.

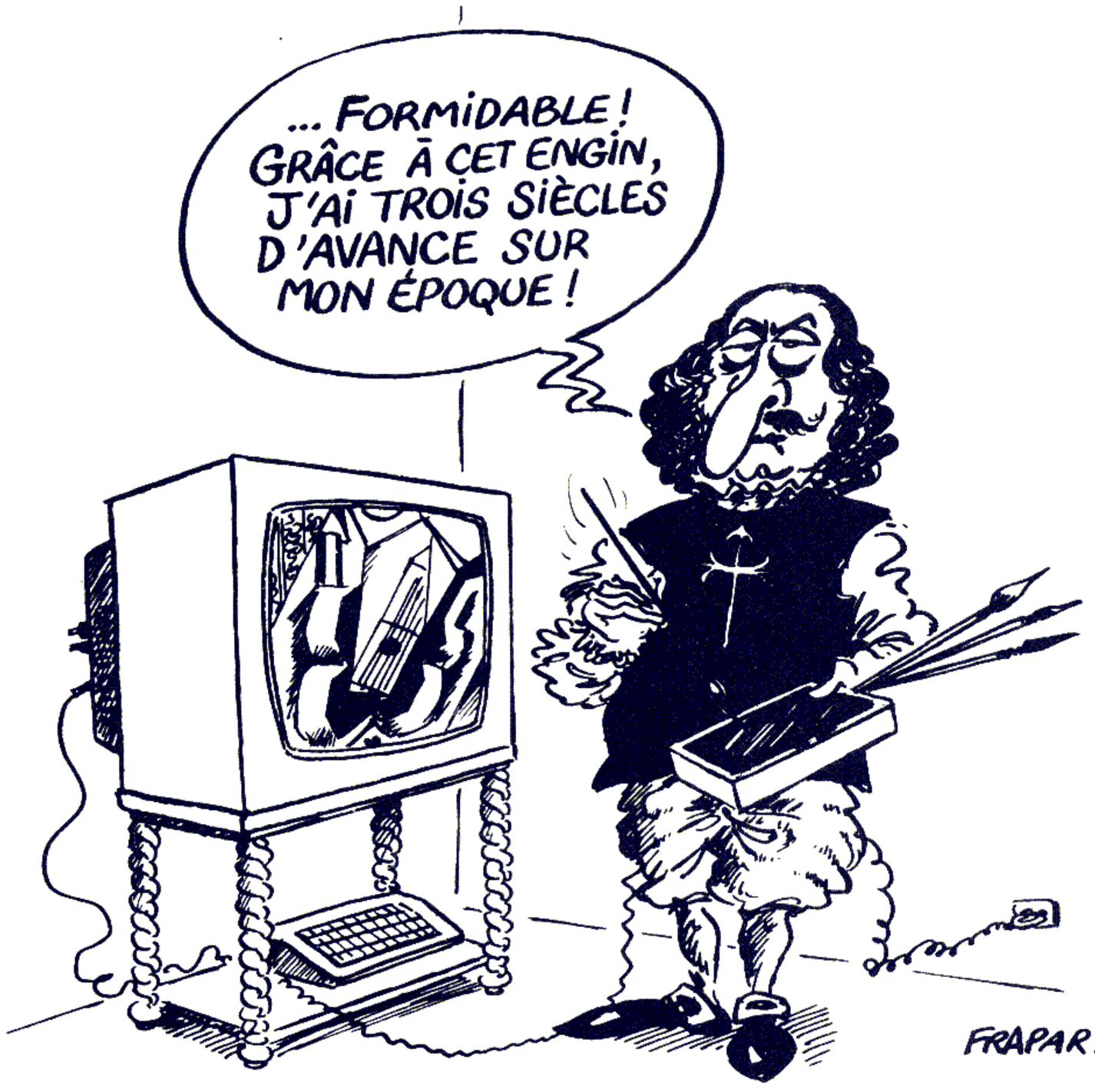
A l'usage, la taille relativement réduite de la surface tactile ne pose pas de problème : sa sensibilité, jointe aux performances du logiciel, rend son usage aisé.

En cas de perte du stylet, il est possible de dessiner avec le doigt, mais la



Le menu : tracer des droites, des cercles, des courbes, choisir sa couleur... Tout est prévu, même la gomme !

TABLETTE GRAPHIQUE KOALA PAD



précision s'en ressent. Ne jamais utiliser la tablette avec un outil pointu, en particulier s'il est métallique (crayons et stylos à bille sont à proscrire) : la surface sensible en souffrirait. La robustesse et la fiabilité semblent bonnes, mais nous n'avons pas martyrisé la tablette outre mesure...

Dessiner avec les manettes de jeu ?

Le dépliant d'utilisation indique que les adresses 54297 et 54298 du Commodore 64 sont utilisées par cette tablette. Cela signifie que le logiciel Koala Painter doit pouvoir fonctionner avec des "paddles". Je vous laisse toutefois imaginer les difficultés qu'il peut y avoir à tracer une ligne courbe en manipulant deux boutons à la fois ! Et d'ailleurs, le logiciel n'est pas vendu sans la tablette, alors...

En revanche, la tablette est susceptible de remplacer les boutons de jeu dans les logiciels qui font appel à eux. Voilà donc un éventail d'applications à explorer.

Les dessins créés et mémorisés sur disquette peuvent être rappelés et affichés à l'écran par un programme Basic simple dont la liste est fournie dans le manuel d'utilisation. Il faut noter toutefois qu'un seul dessin occupe 40 blocs sur la disquette. Le temps de chargement est donc long. On peut agrémenter ainsi un programme classique, mais il ne faut pas espérer réaliser un jeu d'aventure graphique un peu long : une disquette ne suffirait pas.

Et les enfants ? Ayant dû, par la force des choses, renoncer à apprendre l'anglais à ma fille, je me suis contenté de l'asseoir devant la tablette. Quelques explications ont suffi, juste assez pour découvrir qu'elle comprenait le Koala bien mieux que moi. Je me suis donc éclipsé discrètement et elle y est encore. Décidément, un jour ou l'autre, elle va m'obliger à faire l'achat d'un autre ordinateur...

Une très bonne aide à la création

Il faut souligner deux avantages majeurs de la tablette vis-à-vis du traditionnel crayon optique :

- la position horizontale est plus naturelle pour la main ; et la fatigue est bien moins grande qu'avec un crayon qui se déplace verticalement sur l'écran ;
- il est superflu de pousser la luminosité du téléviseur pour compenser, éventuellement, un manque de sensibilité du crayon, et l'utilisateur n'a pas à s'approcher trop près de l'écran.

Résultat : des yeux moins fatigués après un emploi prolongé.

En résumé, Koala fournit, pour le prix de 1 300 FF environ, le moyen de passer de longues heures créatives. La qualité mécanique de l'ensemble est bonne, et les possibilités du logiciel très intéressantes. L'aspect éducatif est évident : les enfants ne s'en plaindront pas. Mais Koala apportera aussi au programmeur un moyen commode d'agrémenter ses propres programmes. Les possibilités encore peu explorées de la tablette graphique sont une porte grande ouverte sur l'imagination. Dommage tout de même qu'elle soit si petite pour d'autres applications.

Et à quand la traduction des quelques pages de la notice ?

Robin BOIS

LE ZX 81 A SA FENÊTRE

LA douceur du climat californien y est peut-être pour quelque chose : les fenêtres sont très à la mode sur les petits ordinateurs professionnels. Chacun veut avoir les siennes, une façon comme une autre de prendre l'air ! Le ZX 81 relève le défi. Il offre une fenêtre mobile, de taille variable, pouvant être disposée à n'importe quel endroit de l'écran.

La consultation d'un tableau à deux dimensions est plus facile lorsqu'elle s'accompagne d'une fenêtre : elle fait défiler le contenu du tableau vers le haut, le bas, la droite ou la gauche.

Grâce à la routine proposée ici, la création d'une telle fenêtre est possible sur le ZX 81. Elle aura des dimensions quelconques et pourra être placée n'importe où sur l'écran. Le reste étant bien entendu encore utilisable.

On rentrera, en mode FAST, le programme 1. Il comprend l'implantation de la routine (lignes 1 à 29) et un exemple de fenêtre (lignes 30 à 240). A la suite, il faudra ajouter les deux lignes 1000 SLOW et 1001 RAND USR 16516 avant de le lancer par RUN.

L'écran affiche alors la longueur de la chaîne C$, soit 490. Si cette valeur est différente, il ne reste plus qu'à vérifier les lignes 3 à 11 qui définissent cette chaîne avant de refaire RUN.

Un appui sur CONT provoquera l'affichage de 9/29. La ligne 1 prend alors l'allure indiquée par la figure 1 et la liste ne se fait plus complètement car le caractère NEW LINE y est présent.

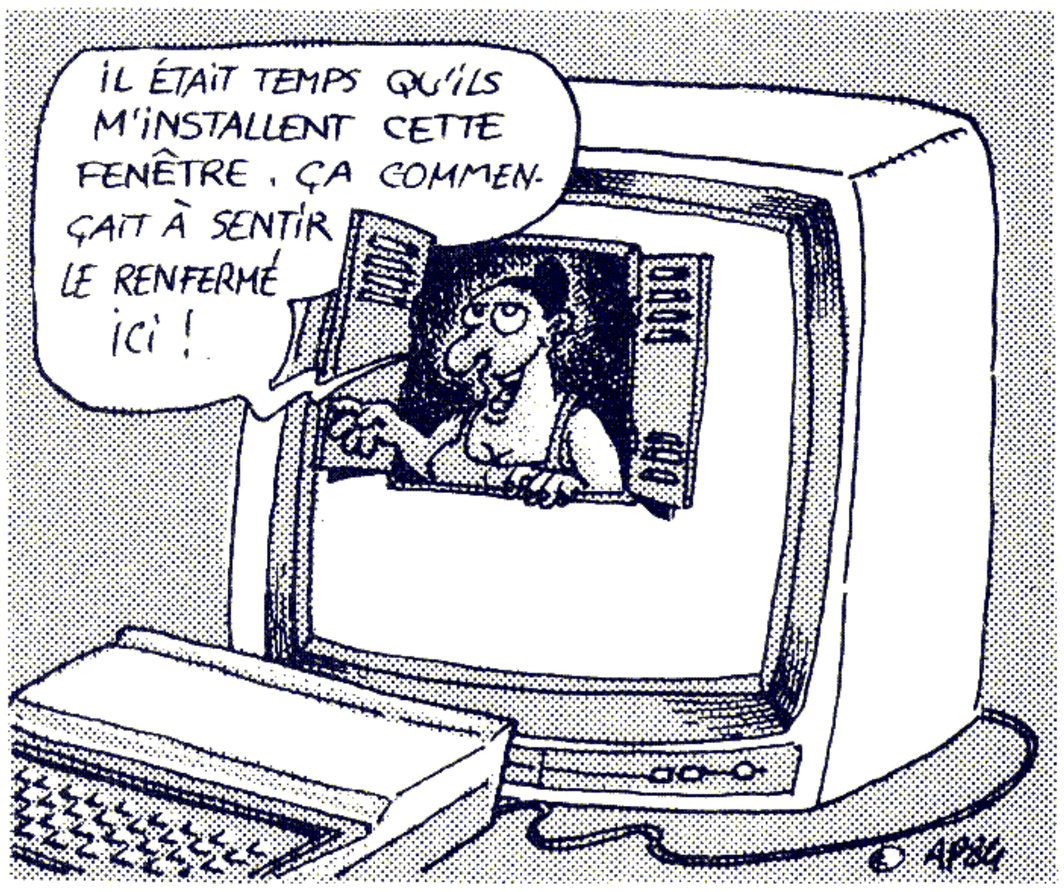
Appuyons une nouvelle fois sur CONT. Si la routine a été chargée correctement, l'écran aura l'aspect de la figure 2. Si l'ordinateur se bloque, il faut le débrancher, récupérer le programme (qui aura été sauvegardé sur cassette auparavant) et corriger l'erreur qui se situe dans une des lignes 1 à 29 du programme 1.

La recherche du message

Une fois la représentation de la figure 2 obtenue à l'écran (les lettres et leur position étant tirées au hasard, elles seront différentes), on peut utiliser les touches ↑ → ↓ ← pour chercher le message BONJOUR. La touche O permet de sortir de la routine.

Il est possible d'utiliser cette routine pour n'importe quel tableau à deux dimensions. Les lignes 2 à 240 du programme n'ont servi qu'à l'implantation de cette routine et à montrer son fonctionnement dans un exemple. Elles peuvent donc être supprimées.

Le nom du tableau est sans importance, mais il doit se trouver avant les variables, c'est-à-dire qu'il doit être



Programme 1
Programme pour ZX 81
Auteur René Tisserand
Copyright *LIST* et l'auteur

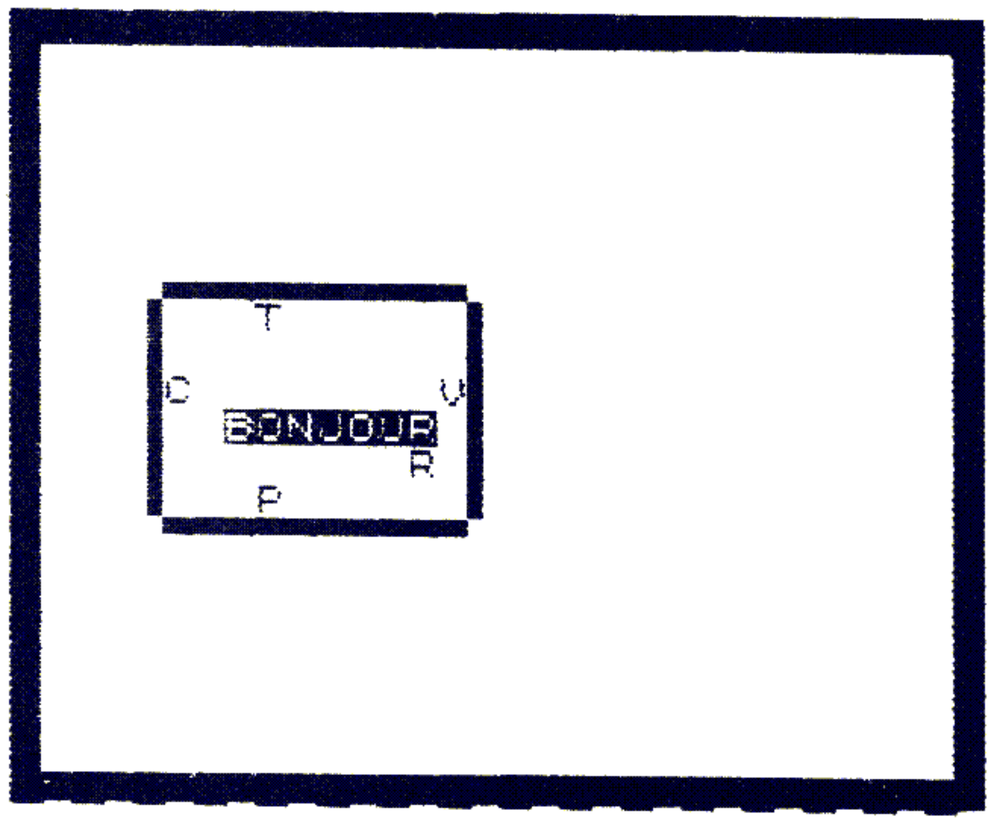
```
    1 REM 111111111111111111111111
1111111111222222222222222222222222
2222222223333333333333333333333333
3333333334444444444444444444444444
4444444445555555555555555555555555
5555555556666666666666666666666666
6666666667777777777777777777777777
7777777771234567890123456789012
    2 REM IMPLANTATION ROUTINE
    3 LET C$="0000186640004000080
500060A0000003A0000003600003C"
    4 LET C$=C$+"0608CB47280119 1F
CB23CB1210F4C9"
    5 LET C$=C$+"3A8D4032998 40E52A
0C40233A8A4011210 0CD9A40ED5B3B40
19EBE1ED4B8E40ED"
    6 LET C$=C$+"B03A9840 3D329840
2001C9EB3E76EDB1ED4B3B4009EBED4B
88400 9ED4B8E40ED"
    7 LET C$=C$+"42180 32A10400108
0009ED5B9040ED4B88407 9CD9A40ED4B
88407 8CD9A40ED4B"
    8 LET C$=C$+"94400 9CDA9403A99
4047 10FE3D20FA3A2540FEF728443A28
40FEDF282EFEEF28"
    9 LET C$=C$+"18FEF7 2805FEFBC8
18DCED4B94407 8B128B10BED43944018
AAED4B90402A9240"
   10 LET C$=C$+"ED42289F03ED4390
4018998ED4B90407 8B12890 0BED439040
1869ED4B94402A96"
   11 LET C$=C$+"40ED42CAEC4003ED
439440C3EC40"
   15 PRINT LEN C$
   16 STOP
   20 LET R=16514
   22 FOR X=1 TO LEN C$-1 STEP 2
   24 POKE R,16*(CODE C$(X)-28)+C
ODE C$(X+1)-28
   26 LET R=R+1
   28 NEXT X
   29 STOP
   30 REM TABLEAU 64,64
   32 CLEAR
   33 DIM A$(64,64)
   36 FOR F=1 TO 600
   38 LET A$(INT (RND*64)+1,INT (
RND*64)+1)=CHR$ (INT (RND*26)+38
)
   40 NEXT F
   42 LET A$(1,1)="▒"
   44 LET A$(1,64)="▒"
   46 LET A$(64,1)="▒"
   48 LET A$(64,64)="▒"
   50 LET A$(30,30 TO 38)=" BONJO
UR "
  100 REM CADRE
  105 SLOW
  110 PRINT AT 7,5;"                "
  120 PRINT AT 14,5;"                "
  130 FOR F=8 TO 13
  140 PRINT AT F,4;"▒";TAB 15;"▒"
  150 NEXT F
  200 LET B$="
                                    "
  210 PRINT AT 0,0;B$
  220 PRINT AT 21,0;B$
  230 FOR F=1 TO 20
  235 PRINT AT F,0;"▒";TAB 31;"▒"
  240 NEXT F
```

Avant de lancer le programme, ajouter les lignes 1000 SLOW et 1001 RAND USR 16516.

Figure 1
Ce qu'est devenue la première ligne de la routine

```
    1 REM    /?£ G▪ (   ▒4 ▪  ▪    CLEA
R  £▪  ?▪▒ACS  ?C▪ ;3ACS  7ACS  )(  PO
KE TAN U▒RNDM▒RND  FAST  E£RND?U▒R
ND)5  LN ▒RND  GOSUB  ?▒RND(  FOR  LP
RINT  GOSUB  ?▒RND  GOSUB  ▒U▒RNDXM▒
RND4▪ TAN  FOR  Y
```

Figure 2
Copie d'écran



Programme 2
Programme pour ZX 81
Auteur René Tisserand
Copyright *LIST* et l'auteur

```
500 REM TABLEAU B$(12*300)
510  CLEAR
511  DIM  B$(12,300)
518  POKE  16516,12
520  POKE  16520,44
521  POKE  16521,1
522  POKE  16522,16
523  POKE  16523,0
525  POKE  16525,8
526  POKE  16526,32
528  POKE  16528,0
530  POKE  16530,4
532  POKE  16532,0
534  POKE  16534,12
535  POKE  16535,1
537  POKE  16537,100
540  FOR  F=1  TO  300
541  LET  B$(INT  (RND*12)+1,INT
RND*300)+1)="*"
542  NEXT  F
544  FOR  F=1  TO  300
545  FOR  G=12  TO  12-INT  (RND*5)
STEP  -1
550  LET  B$(G,F)="▒"
560  NEXT  G
570  NEXT  F
600  LET  B$(1,1)="▒"
602  LET  B$(1,300)="▒"
603  LET  B$(12,1)="▒"
604  LET  B$(12,300)="▒"
1000  SLOW
1001  RAND  USR  16516
```

dimensionné juste après une instruction CLEAR (lignes 32 et 33).

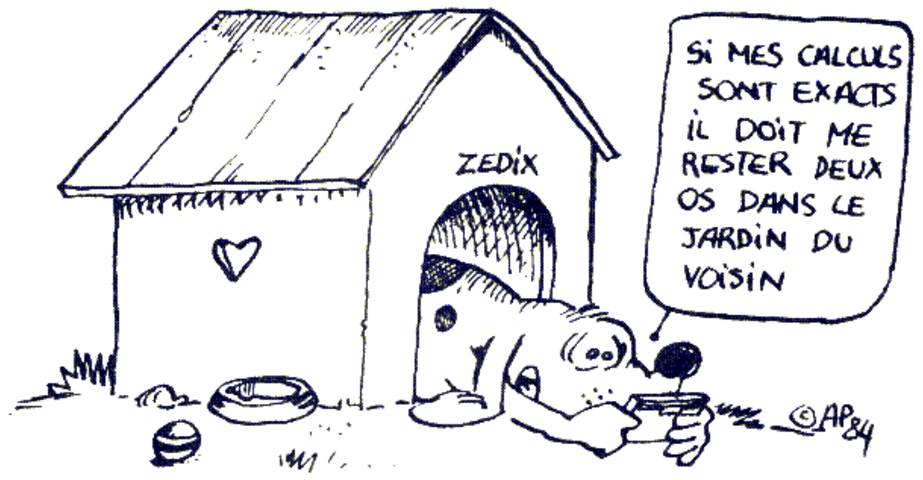
Il est également indispensable de modifier les variables (voir le tableau des variables). Certaines peuvent avoir une valeur supérieure à 255. Dans ce cas, il faudra diviser cette valeur par 256 et retenir le reste et le quotient comme adresses. Par exemple, si le nombre de colonnes du tableau est 300, comme 300 = (1 × 256) + 44, on fait POKE 16520, 44 et POKE 16521,1.

Le programme 2 est un exemple d'utilisation avec un tableau 12 × 300. La routine permet d'utiliser les 24 lignes de l'écran. Pour modifier la vitesse de défilement, on agira sur la variable TEM. La vitesse la plus grande est obtenue par POKE 16537,1.

Au cours de l'entrée d'un programme Basic, la liste peut rester bloquée au milieu du 1 REM. Pour y remédier, il suffit de supprimer le NEW LINE par POKE 16598,0. Dans ce cas, la routine ne peut plus être utilisée sinon l'ordinateur se bloque. Et pour l'utiliser à nouveau, il faut rétablir le NEW LINE par POKE 16598,118.

L'effet de défilement *(scroll)* est obtenu de la manière suivante :

- un sous-programme (FENET) recopie dans une fenêtre de l'écran, une région du tableau dépendant des variables PLT et PCT ;
- un programme principal (DEBUT) repère la partie du tableau à recopier, appelle le sous-programme FENET, teste les touches de déplacement (les flèches), modifie éventuellement PLT et NLT et recommence.



Ce programme principal boucle donc sur lui-même. Ainsi, sans appuyer sur aucune touche du clavier, le sous-programme sera appelé à chaque boucle et la fenêtre sera recopiée identique à elle-même.

Si, au contraire, à chaque boucle, on appuie sur la flèche vers le haut (↑), PLT sera modifié et la fenêtre sera décalée d'une ligne.

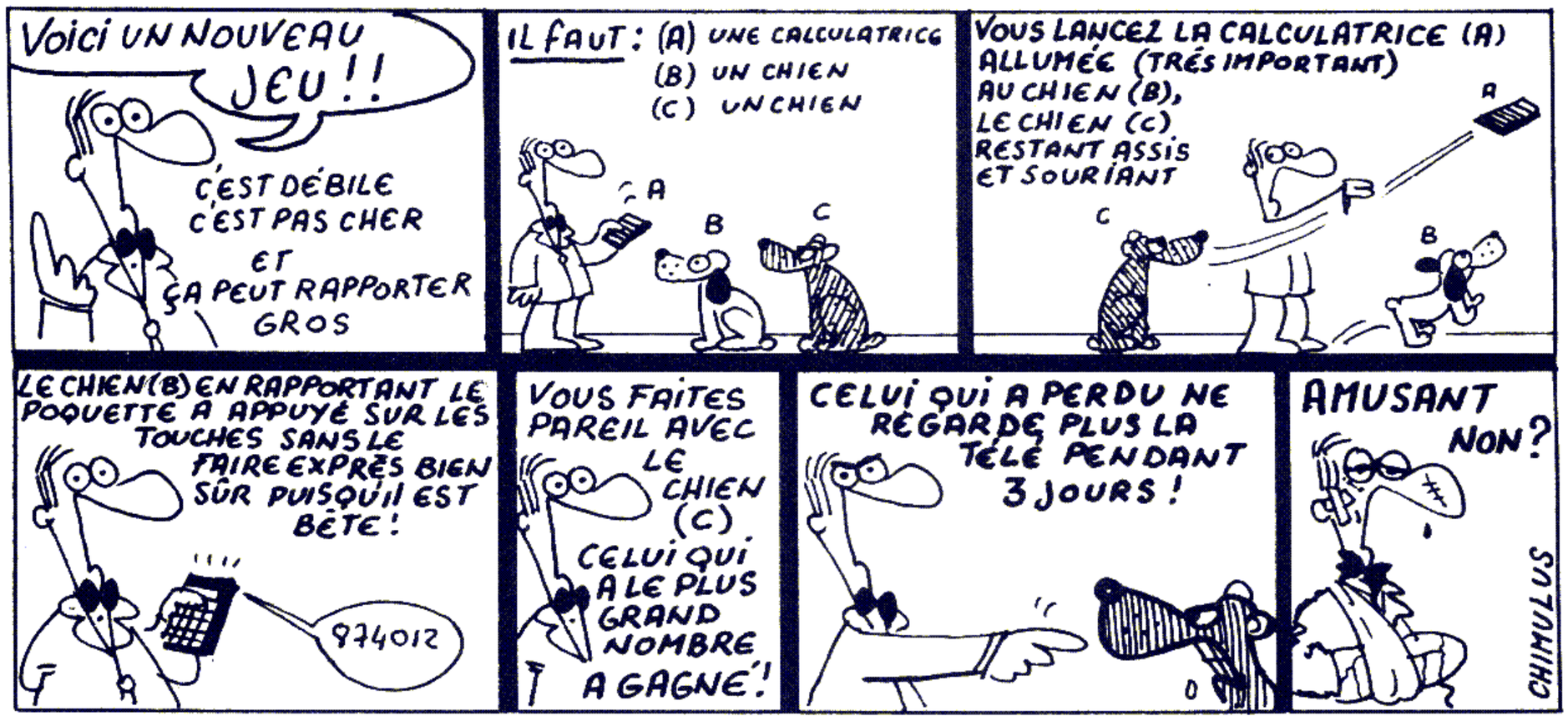
Pour éviter de recopier sur l'écran une partie de la mémoire ne faisant plus partie du tableau, des tests de « fin de tableau » ont été prévus avant modification de PLT et PCT.

Si la valeur de PLM a été mal calculée, on risque de « sortir du tableau ». Ce qui entraînera en général un brouillage de l'écran.

René TISSERAND

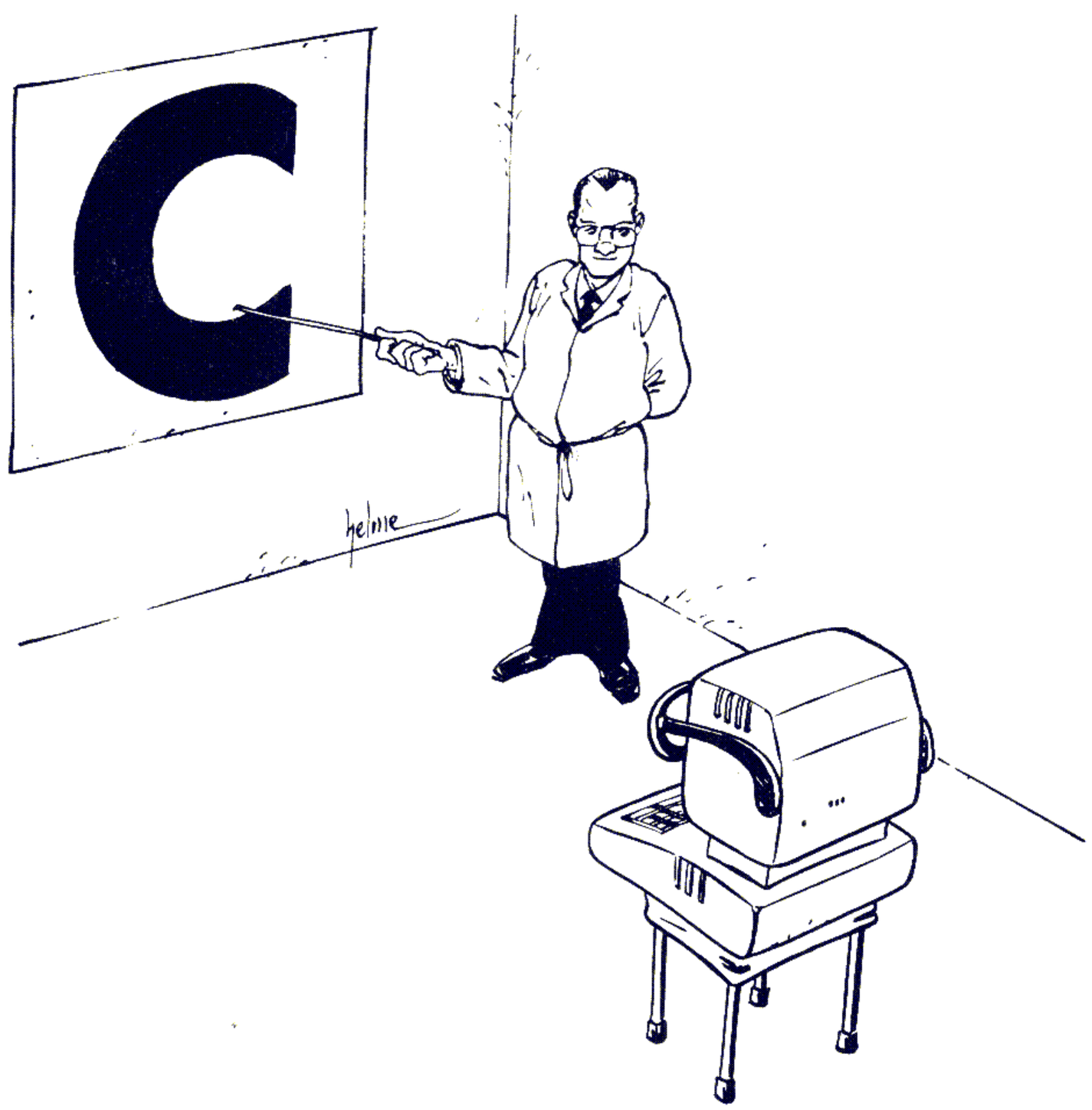
Tableau des variables

Variables	Nombre d'octets	Adresse décimale	Adresse hexadécimale	Valeur
Nombre de lignes du tableau (NLT)	2	16518	4086	suivant tableau
Nombre de colonnes du tableau (NCT)	2	16520	4088	suivant tableau
Première ligne de la fenêtre écran (PLE)	1	16522	408A	0 à 23
Première colonne de la fenêtre écran (PCE)	2	16523	408B	0 à 31
Nombre de lignes de la fenêtre (NLE)	1	16525	408D	24 maxi
Nombre de colonnes de la fenêtre (NCE)	2	16526	408E	32 maxi
Première ligne du tableau (PLT)	2	16528	4090	0
Première ligne maxi (PLM)	2	16530	4092	(NLT) – (NLE)
Première colonne du tableau (PCT)	2	16532	4094	0
Première colonne maxi (PCM)	2	16534	4096	(NCT) – (NCE)
Nombre de lignes fenêtre restant (LR)	1	16536	4098	
Durée de la temporisation (TEM)	1	16537	4099	1 à 255



UN NOUVEAU LANGAGE STRUCTURÉ

IL suffit d'un nouveau langage pour ranimer l'espoir d'un monde meilleur chez le programmeur aguerri. Le langage C, qu'on dit déjà « performant et portable » est prometteur sur ce point !



L'événement en matière de nouveau langage réside dans l'apparition de compilateurs C pour *petits systèmes*, c'est-à-dire de plus grande diffusion. Historiquement, C représente l'aboutissement d'une suite de langages destinés à l'écriture de compilateurs et de systèmes d'exploitation. Un compilateur est un programme volumineux, compliqué, avec des algorithmes variés (manipulation de chaînes de caractères, de fichiers, tris, calculs d'adresse, récursivité). C'est pourquoi C promet d'être idéal pour le calcul numérique, le traitement de texte, la gestion de fichiers, les banques de données, la comptabilité... L'espoir dont il est porteur est lié aussi à sa portabilité : on sait que celle-ci impose des lourdeurs de rédaction dues aux liaisons avec le système. Mais baste ! Achevons les présentations.

Une hiérarchie toute structurée

Le langage C est un langage structuré comme Algol et PL/1 ou Pascal, mais destiné à la compilation : son but est de fournir un code machine très efficace pour éliminer presque totalement l'emploi de l'assembleur.

Contrairement aux autres langages structurés, il apporte la possibilité de travailler à différents niveaux avec des instructions soit très élaborées, soit très proches du système.

La structure d'ensemble linéaire impose la hiérarchie suivante : un programme est composé de fichiers compilés occupés par des fonctions et leurs arguments ; une fonction est définie par

LIST - PAGE 25

UN NOUVEAU LANGAGE STRUCTURÉ

un en-tête puis un bloc ; l'en-tête est une déclaration descriptive (type, nom de la fonction, arguments, type des arguments) ; le bloc délimité par des accolades contient la déclaration des variables, les instructions, éventuellement d'autres blocs.

Variables, pointeurs et instructions...

On dispose de trois types de variables (entières = INT, réelles = FLOAT, de caractères = CHAR) avec le choix de quatre zones d'allocation mémoire : allocation dynamique des variables locales (AUTO), dans une zone fixe (EXTERN ou STATIC), ou encore dans un registre rapide du processeur (REGISTER). Ces variables sont simples ou composées, en tableaux ou structures (RECORD du Pascal). Les pointeurs qui représentent des adresses se comportent presque comme des variables vis-à-vis des opérateurs : p = 'GROG affecte au pointeur p l'adresse de la variable &GROG tandis que x = *p affecte à la variable « x » la valeur de la variable pointée par p. L'utilisation des pointeurs est donc nettement plus poussée qu'en Pascal. Ils sont d'ailleurs nombreux : les opérateurs arithmétiques (*, /, +, −, mod) logiques et de comparaison (&& (et), || (ou), == (=), != (≠),...), classiques. D'autres sont spécifiques à C : ++, −−, incrément et décrément de variables entières ; +=, −=, *= et /= opèrent directement sur la variable sans passer par l'accumulateur (ainsi x+=y correspond à x = x+y) ; *(type) expression* est appelé « cast » en C, il demande la conversion du résultat de l'expression suivant le type indiqué.

Parmi les instructions, celles de structuration pour effectuer les tests, boucles et aiguillages sont classiques (if, if... else, if else if, for, while, do while, switch case). Cependant, l'expression conditionnelle est originale : *expression* 1 ? *expression* 2 : *expression* 3 prend la valeur 2 si 1 est vraie, et 3 dans le cas contraire (exemple : (a>b) ? a:b calcule sup (a,b)).

L'instruction GOTO existe, mais elle est fortement déconseillée comme dans les autres langages structurés. Toute fonction est clôturée par RETURN, avec ou sans transmission de résultat. Il n'existe pas d'instruction d'entrée-sortie. En effet, ces actions sont réalisées au moyen de fonctions de la librairie standard qui accompagne le système (éditeur, compilateur) ; là encore, des fonctions très élaborées (par exemple des sorties avec conversions spécifiables ou formatage) côtoient des fonctions de bas niveau, très proches du système.

Enfin les instructions du préprocesseur (logiciel) sont reconnaissables au symbole # qui les précède. Il s'agit éventuellement de macro-définitions et de variables muettes (ou paramètres) et de directives de compilation (compilation conditionnelle).

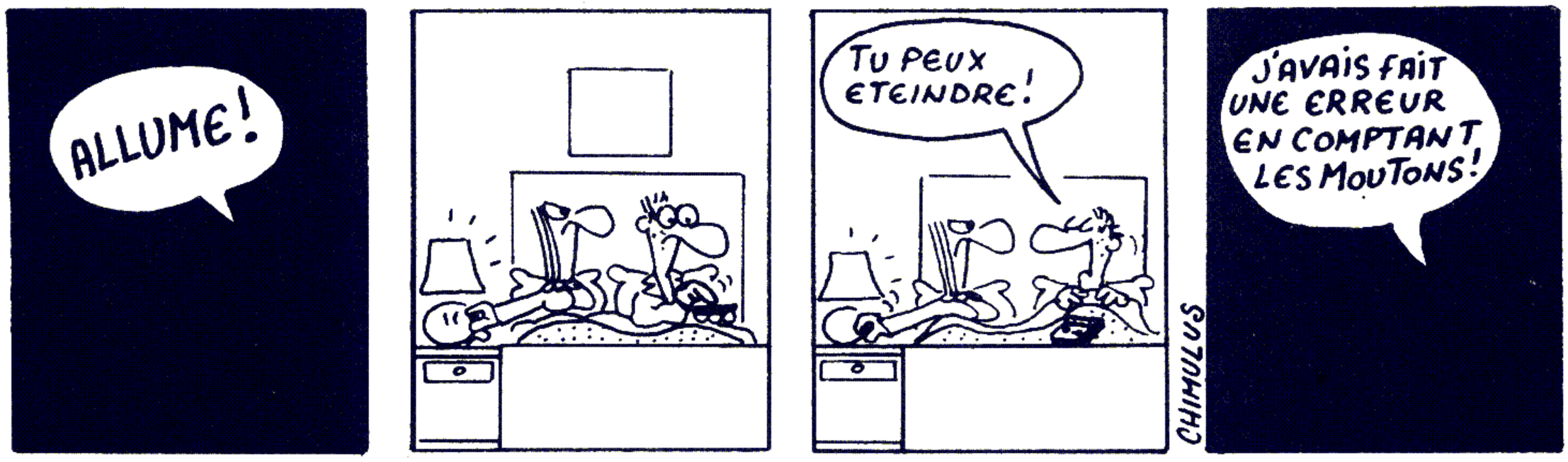
Le langage C se montre fort là où pèchent les autres langages. La rédaction de logiciels volumineux exige la compilation, et C est capable d'approcher le processeur, donc d'obtenir les mêmes performances que le langage-machine, avec beaucoup moins d'efforts qu'en Assembleur ; il est surtout possible d'utiliser un programme tel quel si on change de machine (en théorie seulement, si on veut constater la disparité dans les réservations mémoires des variables pour chaque compilateur).

Facilement transposable

Aujourd'hui C existe sur des systèmes dits « familiaux » (Dragon 64), ce qui est séduisant ; il ne faut pas perdre de vue qu'un langage dépend tout d'abord du système (plus de dix compilateurs C pour l'IMB PC) et de la configuration du matériel (un recours intensif au disque souple est très pénalisant).

Il est évident que la transcription d'un algorithme est plus sûre et plus efficace en langage structuré. En outre, il est possible de reprendre ultérieurement le programme très facilement (mise à jour, évolution, etc.). Le langage C offre par rapport aux autres langages structurés l'avantage, non négligeable, d'éviter pratiquement tout recours aux routines en Assembleur.

Claude NOWAKOWSKI



DES SECTEURS EN PISTE

LE Système d'Exploitation des Disquettes (SED) de Commodore permet de faire beaucoup de choses a priori impossibles au prix d'un peu de méthode et de réflexion. Bien que plus spécialement destinées aux usagers du C.64, les applications présentées ici seront utilisables sans grandes modifications sur les machines plus "anciennes" : les gammes 3000 et 4000 avec lecteurs doubles ou monodisques, ainsi que les VIC 20 équipés du lecteur 1540 ou 1541.

L'organisation des disquettes Commodore paraîtra transparente à ceux qui sont familiarisés avec l'utilisation des fichiers à accès direct. Ils trouveront là un terrain connu. Pour les autres, il est préférable de progresser lentement. On commencera donc par une observation générale des enregistrements disque.

Chaque face d'une disquette se trouve découpée, lors de son formatage, en une série de pistes concentriques, chacune étant elle-même découpée en secteurs successifs. Sur chaque secteur, la tête d'enregistrement est susceptible d'inscrire 256 octets, formant un *bloc*. Le tableau 1 indique le nombre de secteurs disponibles sur chaque piste : la piste 1 est la plus extérieure, donc la plus longue, tandis que la piste 35 — la plus proche du centre — est la plus courte. C'est pour cette raison que le nombre de secteurs par piste va en diminuant de la piste 1 à la piste 35. Quant aux pistes 18 à 24, elles possèdent un secteur de plus lorsqu'elles sont issues d'un lecteur ancien modèle. Ceci explique certains problèmes de compatibilité...

Commençons, en toute logique, par l'étude de cette piste centrale. Centrale à plus d'un titre puisqu'elle est non seulement placée au milieu de la disquette (sur la piste 18), mais aussi parce qu'elle est un pivot du fonctionnement du système. C'est elle en effet qui contient la liste des fichiers enregistrés, et diverses informations fondamentales sur ces fichiers.

Toutefois, remettons à plus tard l'étude du premier secteur de cette piste.

Tableau 1
Pistes et secteurs selon les types de lecteurs utilisés

Type de lecteur	Pistes	Secteurs
2031 - 4040 1540 - 1541	1 à 17 18 à 24 25 à 30 31 à 35	0 à 20 0 à 18 0 à 17 0 à 16
2040 - 3040	1 à 17 18 à 24 25 à 30 31 à 35	0 à 20 0 à 19 0 à 17 0 à 16

SECTEURS EN PISTE

Penchons-nous plutôt sur le second secteur, qui porte... le numéro 1 ! Ce secteur contient le premier bloc de 256 octets renfermant les titres des fichiers présents sur la disquette. Nous avons l'habitude de prendre connaissance de son contenu en composant l'instruction : LOAD"$",8.

Qui est quoi qui fait quoi

Le tableau 2 donne un exemple type de ce qu'il est possible d'obtenir dans un tel cas. En tout état de cause, si le système est capable de fournir ces informations, c'est qu'il les lit quelque part... tout simplement sur la piste 18. Faisons donc une comparaison avec le tableau suivant (tableau 3).

Ce tableau constitue une image des 256 octets présents sur la piste 18, secteur 1 du disque. Les valeurs sont hexadécimales. Les nombres de gauche sont décimaux, et représentent le numéro de l'octet qui suit immédiatement le ' >'. Les caractères de droite sont la traduction ASCII des 16 octets de chaque ligne.

Les deux premiers octets du bloc (12 et 04) représentent respectivement le numéro de piste et de secteur du bloc suivant ; bloc qui contient évidemment la suite du catalogue (piste 18, secteur 4 en décimal).

L'octet suivant (octet 2) a pour valeur 82 : il indique que le premier fichier est un PRG. Mais d'autres valeurs sont possibles. Elles sont présentées dans le tableau 4. Ainsi, l'octet 34 (troisième ligne) indique la présence d'un fichier SEQ, l'octet 66 (cinquième ligne) indique un fichier REL, tandis que l'octet 98 (septième ligne) indique un fichier qui a été détruit. Il n'apparaît donc pas au catalogue.

Les quatrième et cinquième octets (octets 3 et 4) représentent, comme les deux premiers octets (octets 0 et 1), un numéro de piste, et un numéro de secteur. Il s'agit de l'emplacement physique où commence le fichier. Ici, les valeurs 11 et 00 indiquent un début sur la piste 17, secteur 0. Cette information n'apparaît pas au catalogue.

Les 16 octets suivants (octets 5 à 20) contiennent le titre du fichier, sous forme ASCII. Si le titre a moins de 16 caractères, il est complété par une suite d'espaces "shiftés" (code 160, ou A0 en hexa).

Les octets 21 à 29 sont toujours à 0, sauf dans le cas d'un fichier REL. Certains sont remplacés alors par un pointeur piste/secteur, et un indicateur de longueur des enregistrements. Il en est ainsi des octets 85 à 87.

Les octets 30 et 31 indiquent le nombre de blocs occupés par le fichier sur la disquette, sous la forme poids faible-poids fort. Le programme FICH PROG occupe donc 00 15 blocs, soit 21 blocs en décimal. C'est bien ce qu'indique le catalogue.

Les octets 32 à 256 répondent aux mêmes schémas. Un maximum de huit fichiers peuvent figurer dans un bloc-disque ; les titres suivants sont inscrits un peu plus loin sur la piste 18. Il reste que bien des choses, inscrites sur le support magnétique, sont encore cachées à l'utilisateur...

L'instruction LOAD"$",8 a pour effet de charger en mémoire le contenu de la piste 18, provoquant du même coup la destruction du programme présent en mémoire lors du chargement.

Type	Valeur
DEL	00
SEQ	81
PRG	82
USR	83
REL	84

Tableau 4
Type de fichier associé à la valeur de l'octet

Le catalogue de la disquette

Connaissant maintenant la structure des blocs de cette piste, nous allons pouvoir utiliser les commandes du SED pour afficher le catalogue à l'écran, sans perdre le programme en mémoire. Quelques lignes nécessaires sont fournies par le programme 1. On peut les introduire mais en respectant scrupuleusement la numérotation.

Tableau 2
Le catalogue classique

```
0 "DISQUETTE TEST"          01 2A
21    "FICH PROG"           PRG
4     "FICH SEQ"            SEQ
29    "FICH REL"            REL
8     "PROG 5"              PRG
6     "PROG 6"              PRG
12    "PROG 7"              PRG
12    "PROG 8"              PRG
12    "PROG 9"              PRG
577 BLOCKS FREE.
```

Tableau 3
Le catalogue tel qu'il est sur la disquette

```
PISTE 18 SECTEUR 1
  0 > 12 04 82 11 00 46 49 43 48 20 50 52 4F 47 A0 A0  : .....FICH PROG..
 16 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 15 00  : ................
 32 > 00 00 81 13 00 46 49 43 48 20 53 45 51 A0 A0 A0  : .....FICH SEQ...
 48 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 04 00  : ................
 64 > 00 00 84 13 02 46 49 43 48 20 52 45 4C A0 A0 A0  : .....FICH REL...
 80 > A0 A0 A0 A0 A0 17 0A 64 00 00 00 00 00 00 1D 00  : .......$........
 96 > 00 00 00 13 03 50 52 4F 47 20 44 45 54 52 55 49  : .....PROG DETRUI
112 > 54 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 08 00  : T...............
128 > 00 00 82 10 00 50 52 4F 47 20 35 A0 A0 A0 A0 A0  : .....PROG 5.....
144 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 08 00  : ................
160 > 00 00 82 10 01 50 52 4F 47 20 36 A0 A0 A0 A0 A0  : .....PROG 6.....
176 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 06 00  : ................
192 > 00 00 82 10 05 50 52 4F 47 20 37 A0 A0 A0 A0 A0  : .....PROG 7.....
208 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0C 00  : ................
224 > 00 00 82 14 04 50 52 4F 47 20 38 A0 A0 A0 A0 A0  : .....PROG 8.....
240 > A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0C 00  : ................
```

Programme 1
Lecture du catalogue

```
160 REM LECTURE DU CATALOGUE
170 Z$=CHR$(.):P=18:S=1
180 OPEN 15,8,15,"I0"
190 OPEN 2,8,2,"#"
200 :
210 PRINT#15,"U1";2;0;P;S
220 PRINT#15,"B-P";2;0
230 GET#2,B$,H$
240 P=ASC(B$+Z$)
250 S=ASC(H$+Z$)
260 FOR X=0 TO 7
270 PRINT#15,"B-P";2;X*32+2
280 GET#2,C$
290 IF ASC(C$+Z$)=0 THEN 380
300 PRINT#15,"B-P";2;X*32+5
310 NF$=""
320 FOR Q=1 TO 16
330 GET#2,C$
340 IF ASC(C$+Z$)<>160 THEN NF$=NF$+C$
350 NEXT Q
360 REM
370 PRINT"■  ";NF$;SPC(18-LEN(NF$));
380 NEXT X
390 IF P THEN 210
400 :
410 CLOSE 2
420 CLOSE 15
430 END
```

Programme 2
Catalogue auto-chargeur

```
100 REM ********************************
110 REM *  UTILITAIRE DISQUE NO.1   *
120 REM * CATALOGUE AUTO-CHARGEUR. *
130 REM ********************************
140 PRINT"□":POKE 53280,0:POKE 53281,2
150 W=-1:S0$="■■ >■":S1$="■■■  "
160 DIM F$(46)
170 Z$=CHR$(.):P=18:S=1
180 OPEN 15,8,15,"I0"
190 OPEN 2,8,2,"#"
200 :
210 PRINT#15,"U1";2;0;P;S
220 PRINT#15,"B-P";2;0
230 GET#2,B$,H$
240 P=ASC(B$+Z$)
250 S=ASC(H$+Z$)
260 FOR X=0 TO 7
270 PRINT#15,"B-P";2;X*32+2
280 GET#2,C$
290 IF ASC(C$+Z$)=0 THEN 380
300 PRINT#15,"B-P";2;X*32+5
310 NF$=""
320 FOR Q=1 TO 16
330 GET#2,C$
340 IF ASC(C$+Z$)<>160 THEN NF$=NF$+C$
350 NEXT Q
360 W=W+1:F$(W)=NF$
370 PRINT"■  ";NF$;SPC(18-LEN(NF$));
380 NEXT X
390 IF P THEN 210
400 :
410 CLOSE 2
420 CLOSE 15
430 :
440 REM CHOIX DU PROG
450 PRINT"■■"S0$;
460 NP=0
470 R=PEEK(197)
480 IF R=4 THEN 560
490 IF R=3 THEN 520
500 IF R=1 THEN 610
510 GOTO 470
520 REM DESCENTE
530 NP=NP+1:IF NP>W THEN NP=W:GOTO 470
540 PRINT S1$;SPC(18);S0$;
550 GOTO 590
560 REM MONTEE
570 NP=NP-1:IF NP<0 THEN 450
580 PRINT S1$;"□";SPC(18);S0$;
590 FOR Q=0 TO 60:NEXT Q:GOTO 470
600 :
610 REM CHARGEMENT DU PROGRAMME
620 PRINT"□LOAD"CHR$(34);F$(NP);CHR$(34);",8,1"
630 PRINT"■■■■RUN"
640 POKE 631,19
650 FOR Q=632 TO 636:POKE Q,13:NEXT
660 POKE 198,6
670 END
```

Le programme 1 ligne par ligne

ligne 170 : la piste à lire porte le numéro 18, le secteur a le numéro 1 ;

ligne 180 : on ouvre le canal de commande sous le numéro 15, en initialisant la disquette ;

ligne 190 : le canal de données est ouvert sous le numéro 2, avec réservation d'une mémoire-tampon du lecteur. Les 256 octets lus y seront placés, pour être utilisés par la suite dans le programme ;

ligne 210 : on envoie une commande de lecture sur le canal 2, lecteur 0, piste 18, secteur 1 ;

ligne 220 : on indique que l'acquisition des octets qui sont maintenant présents dans le tampon, doit se faire à partir du premier octet (octet 0) ;

lignes 230 à 250 : les deux premiers octets qui sont le numéro de piste et le numéro de secteur sont récupérés. Ils seront à lire ensuite ;

ligne 260 : huit titres au maximum sont à lire dans chaque bloc ;

lignes 270 à 290 : on vérifie pour chaque titre lu l'octet qui indique le type du fichier. S'il s'agit d'un 0, le fichier est détruit et on passe au titre suivant ;

lignes 300 à 370 : si l'octet n'est pas à 0, on récupère le nom du fichier qui sera affiché à l'écran ;

ligne 390 : si la valeur de P (numéro de piste suivant) n'est pas 0, le catalogue est incomplet, et on recommence le processus pour lire le bloc suivant (secteur S) ;

lignes 410 à 430 : si P est à 0, tout est terminé, et les deux canaux ouverts sont soigneusement refermés.

Ce programme 1 peut être inclus sans difficulté comme sous-programme dans un programme plus complet. Une application immédiate est proposée par le programme 2 (auto-chargeur).

Le programme 1 pourra être placé sur toutes vos disquettes sous un titre quelconque ; par exemple "MENU". Il vous permettra d'afficher sur l'écran le catalogue de la disquette, et par la simple manipulation des touches F1, F7 et RETURN, de sélectionner et de lancer *dans la foulée* le programme de votre choix.

Il y a tout de même une limitation due à la taille de l'écran, qui impose de ne pas dépasser 46 titres sur la disquette. Mais ce cas est assez rare.

Il reste à voir (c'est pour une prochaine fois) comment obtenir un catalogue complet, comportant, en plus des informations habituelles, tout ce que l'on peut savoir sur les fichiers présents sur la disquette : emplacement physique sur le disque, adresses d'implantation en mémoire, fichiers détruits, etc.

Jean-Pierre LALEVÉE

LIST A TESTÉ

LE BASIC DU BBC

Le clavier du BBC ; BBC pour British Broadcasting Corporation

FABRIQUÉ par Acorn, le BBC nous vient d'Outre-Manche. Il est muni d'un Basic étonnant : à la fois riche et rapide. Il est importé par Sterco International et coûte environ 6000 FF ttc.

A la mise sous tension du BBC, un bip se fait entendre, et s'affiche alors « BBC Computer 32K ». Nous sommes donc devant un modèle B, le modèle A ne faisant que 16 Koctets. Un PRINT HIMEM-LOMEM nous précise qu'en fait, seulement 28158 octets sont disponibles.

Avec la présence au clavier des quatre curseurs, on peut penser avoir un éditeur plein écran. Il n'en est rien, et la touche COPY vient heureusement à notre secours. Elle est d'un emploi un peu déconcertant au début, mais s'avère fort commode à l'usage. L'appui sur l'une des quatre touches fléchées affiche un pavé blanc à l'endroit où était le curseur clignotant, et le nouveau curseur peut être déplacé à volonté : la touche COPY copiera alors au niveau du pavé blanc le caractère pointé par le curseur. Grâce à l'*autorepeat* des touches, on peut recopier rapidement des lignes déjà introduites (et encore présentes à l'écran) et les modifier commodément.

Une sensation désagréable gêne au début : la première ligne de l'écran est tronquée sur le moniteur, et n'est donc guère visible. Tous les essais de réglage du moniteur sont inopérants, et seule une plongée dans la documentation nous apprend que *TV 255 descend l'écran d'une ligne.

Un appui sur BREAK est nécessaire après *TV 255 pour valider le changement d'écran. La vingt-cinquième et dernière ligne descend elle aussi comme les 24 autres, mais reste visible.

Pour arrêter un programme en cours, il faut appuyer sur ESCAPE. Un simple BREAK arrête lui aussi le programme, mais il est plus violent : il réinitialise la machine. Après lui, un LIST reste inopérant, le programme ayant été apparemment effacé. Toutefois, l'ordre OLD permet de le récupérer.

L'entrée et la mise au point de programmes sont facilitées par la présence de AUTO, DELETE, et RENUMBER. Avec TRACE ON et TRACE OFF, il est possible de suivre l'exécution des programmes. De plus, TRACE 130 indique que seules les lignes inférieures à 130 seront en mode Trace.

Les numéros de ligne peuvent aller de 0 à 32767, et non pas jusqu'à 65535 comme sur beaucoup de machines. La plupart des instructions Basic peuvent être abrégées : ainsi P. pour PRINT, L. pour LIST, REN. pour RENUMBER, etc. mais ce Basic ne comprend pas les ordres écrits en minuscules.

Souple avec les variables

Bien sûr, 1 5 RETURN efface la ligne 15, mais 1 5 SPACE RETURN donne une ligne contenant un espace : cela permet de séparer commodément les parties du programme, sans avoir à utiliser de REM.

Le déroulement d'une liste à l'écran ne semble pas être arrêté par l'appui sur

une touche. Toutefois CTRL N met le BBC en mode page : dès qu'un écran est rempli, les diodes *Caps Lock* et *Shift Lock* s'affichent simultanément, et le déroulement s'arrête. L'appui sur Shift le libère, et la liste continue jusqu'à la page suivante. CTRL O remet le BBC en mode de défilement normal.

Le BBC est particulièrement souple avec les variables : leurs noms ne sont pas limités en longueur. Les variables alphanumériques peuvent contenir jusqu'à 255 caractères. Les entiers sont stockés sur 4 octets, et peuvent donc être compris entre $-2\,147\,483\,648$ et $2\,147\,483\,647$: c'est bien mieux que les $-32\,768$ à $32\,767$ auxquels nous étions habitués. Les variables A%, B%, ..., Z% sont en permanence allouées en mémoire vive, et les ordres RUN et NEW ne les effacent pas.

Une curiosité : après avoir fait N% = 2 147 483 647, P.N% affiche 2.14748365E9. En notation exponentielle, et en perdant un chiffre significatif ?! Mais non, il existe une variable spéciale nommée @ % qui permet de préciser le format de PRINT : le nombre de chiffres significatifs, la notation exponentielle ou non, le nombre de chiffres après la virgule. Grâce à un @ % = &A0B (& pour l'hexadécimal), P. N% affichera 2 147 483 647.

Les réels sont stockés sur cinq octets, et peuvent traiter des nombres entre $1{,}7\times10^{38}$ et $2{,}9\times10^{-39}$ (de même pour les nombres négatifs).

Les dimensionnements ne sont pas limités. On peut donc trouver des DIM A(3,3,3,3,3), par exemple.

L'exécution de la ligne 10 INPUT N : PRINT N ne provoquera pas d'erreur après l'introduction de COUCOU dans N, mais affichera « 0 ». Quant à PRINT *var,* où *var* est une variable qui n'a jamais été initialisée, il donnera « No such variable » au lieu du 0 traditionnel.

Les instructions GET ou GET$ attendent l'appui d'une touche, la première retournant le code ASCII. Avec INKEY(t) ou INKEY$(t), l'appui d'une touche est attendu seulement pendant t centièmes de seconde.

L'introduction interne des données est possible grâce aux classiques READ, DATA et RESTORE.

La présence de FOR... NEXT et de IF... THEN... ELSE est normale. Celle de REPEAT... UNTIL l'est déjà moins. Mais surtout, c'est la définition des fonctions et des procédures qui est ici vraiment extraordinaire.

Commençons par les fonctions. Par exemple, celles de la forme DEF FNMOY (A,B) = (A + B)/2 sont déjà connues. Un P.FNMOY (2,3) affiche 2.5. Ici, en plus, la définition peut s'étaler sur plusieurs lignes. Et des variables peuvent y être déclarées locales. Voici par exemple ce que cela peut donner :

```
100 DEF FN PUISS(A,N)
110 LOCAL I,J
120 J=1
130 FOR I=1 TO N:J = A*J:NEXT I
140 =J
```

Tout appel de FNPUISS (A,N) élèvera A à la puissance N. Inutile alors de faire attention à l'emploi de I et J dans le reste du programme, ces variables ont été déclarées locales dans la fonction.

Des conversions d'angle aisées

La récursivité est possible, et la fonction factorielle se programme facilement :

```
100 DEF FN FACT(N)
110 IF N=1 THEN=N ELSE=
    N*FNFACT (N-1)
```

Quant aux procédures, leur définition commence par DEF PROC, et doit se terminer par ENDPROC. Là aussi variables locales et récursivité sont possibles.

Les GOTO, GOSUB, ON... GOTO, ON... GOSUB ont la particularité de pouvoir employer des numéros de ligne calculés. Ainsi GOTO 100 + 10*L est équivalent à ON L GOTO 110, 120, 130,... A employer toutefois avec précaution, un RENUMBER devenant catastrophique.

Les opérations arithmétiques sont au nombre de sept : +, −, *, /, ↑, DIV, MOD. DIV donne le résultat d'une division entière, alors que MOD en donne le reste. Sur BBC, la fonction TIME fournit le temps en centièmes de seconde et l'affichage en heures-minutes-secondes peut, par exemple, s'obtenir par :

```
10 S=(TIME DIV 100) MOD 60
20 M=(TIME DIV 6000) MOD 60
30 H=(TIME DIV 360 000) MOD 24
40 PRINT H ; « H » ; M ; « M » ;
   S ; « S »
```

Les chaînes alphanumériques sont aisément manipulées par LEFT$, MID$, RIGHT$. De plus INSTR recherche la position d'une chaîne à l'intérieur d'une autre. La répétition des chaînes est possible, mais les arguments sont inversés par rapport à d'autres Basic : il faudra faire A$ = STRING$ (2,« COU »), par exemple.

Liste des mots du Basic du BBC

ABS	NEXT
ACS	NOT
ADVAL	OFF
AND	OLD
ASC	ON
ASN	OPENIN
ATN	OPENOUT
AUTO	OPT
BGET	OR
BPUT	PAGE
CALL	PI
CHAIN	PLOT
CHR$	POINT
CLEAR	POS
CLG	PRINT
CLOSE	PROC
CLS	PTR
COLOUR	RAD
COS	READ
COUNT	REM
DATA	RENUMBER
DEF	REPEAT
DEG	REPORT
DELETE	RESTORE
DIM	RETURN
DIV	RIGHT$
DRAW	RND
ELSE	RUN
END	SAVE
ENDPROC	SGN
ENVELOPE	SIN
EOR	SOUND
EOF	SPC
ERL	SQR
ERR	STEP
ERROR	STOP
EVAL	STR$
EXP	STRING$
EXT	TAB
FALSE	TAN
FN	THEN
FOR	TIME
GCOL	TO
GET	TRACE
GET$	TRUE
GOSUB	UNTIL
GOTO	USR
HIMEM	VAL
IF	VDU
INKEY	VPOS
INKEY$	WIDTH
INPUT	*CAT
INSTR	*DISC
INT	*DISK
LEFT$	*EXEC
LEN	*FX
LET	*KEY
LINE	*LOAD
LIST	*MOTOR
LN	*NET
LOAD	*OPT
LOCAL	*ROM
LOG	*RUN
LOMEM	*SAVE
MID$	*SPOOL
MOD	*TAPE
MODE	*TELESOFT
MOVE	*TERMINAL
NEW	*TV

LIST A TESTÉ

LE BASIC DU BBC

Le BBC possède un évaluateur de calcul. A$ = « 2+3*SQR(16) » suivi de PRINT EVAL(A$) affichera 14. Vraiment commode ! Les fonctions logiques sont là avec NOT, AND, OR, ainsi que le *ou exclusif* noté curieusement EOR (au lieu du XOR habituel). Pour la trigonométrie, tout y est, même ASN (arc sinus) et ACS (arc cosinus) en plus de ATN. Les conversions sont extrêmement facilitées : SIN(RAD(90)) donnera 1, alors que DEG(PI) donnera 180.

La gestion des erreurs est elle aussi fort complète avec ON ERROR GOTO, ERR, ERL, et même REPORT qui affiche le message explicite correspondant à la dernière erreur commise.

Il est possible de demander un affichage graphique couleur par l'instruction MODE. Les trois principaux modes sur le modèle B sont 640×256 points avec 2 couleurs, 320×256 points avec 4 couleurs, et 160×256 points avec 16 couleurs. Les caractères sont toujours sur 32 lignes, mais s'étalent suivant 80, 40, ou 20 colonnes. Le gros inconvénient de ces trois modes est de demander 20 Koctets de mémoire vive, ce qui ne laisse plus que 8 Koctets à l'utilisateur. Il y a cependant des modes intermédiaires, comme par exemple 160×256 avec 4 couleurs qui occupe 10 Koctets de mémoire vive.

En fait, même dans le mode 16 couleurs, 8 sont réellement utilisables. Les 8 autres ne sont que des clignotements entre une couleur et sa complémentaire. Par exemple COLOUR 11 fera afficher alternativement jaune et bleu.

Quel que soit le mode choisi, l'écran est toujours *virtuellement* composé de 1280×1024 points, l'origine étant en bas à gauche. Ce sera fortement apprécié par celui qui programme : il n'aura pas à recalculer la position des tracés s'il veut changer de mode.

Le principal ordre d'affichage graphique est PLOT qui trace points, lignes, triangles, en continu ou en pointillé. Et l'état d'un point est demandé par POINT(x,y).

Pour l'affichage des caractères, le PRINT TAB(x) est, bien sûr, connu, mais le PRINT TAB(x,y) l'est moins et s'avère fort logique.

Fiche technique du BBC

Constructeur : *Acorn (Grande-Bretagne)*
Distributeur : *Sterco International*
Prix : *environ 6 000 FF ttc*
Mémoire vive : *32 Koctets (modèle B)*
Mémoire morte : *32 Koctets*
Langages : *Basic, Assembleur, Forth (disponible en logiciel)*
Affichage graphique : *160, 320 ou 640 sur 256 points selon le mode couleur ; écran virtuel de 1280 sur 1024 points*
Variables : *jusqu'à 255 caractères pour les variables alphanumériques ; de −2147483648 à 2147483647 pour les entiers ; de $1,7\times10^{38}$ à $2,9\times10^{-39}$ pour les réels*
Nombre de mots du Basic : *138*

L'étrange fonction COUNT

Sur les autres machines les PRINT CHR$($x_1$) ; CHR$(x_2) ;... sont pénibles et fastidieux à rentrer pour celui qui programme. Sur le BBC, cet ordre existe aussi mais il peut être avantageusement remplacé par VDU x_1, x_2. Il suffisait d'y penser ! Ainsi VDU 24 suivi de quatre doubles octets définit une fenêtre graphique ; de même, VDU 28 définit une fenêtre de texte. Par la suite, les PLOT n'afficheront que dans la première fenêtre, et les PRINT dans la seconde. Vraiment très commode pour séparer texte et image sur un même écran.

A noter, la présence de l'étrange fonction COUNT qui fournit le nombre de caractères affichés depuis le dernier RETURN.

Le BBC dispose d'un haut-parleur intégré branché sur un circuit intégré à 3 canaux, plus un canal de bruit. Cela peut être simplement commandé par l'ordre SOUND suivi de 4 paramètres : numéro de canal, amplitude, hauteur, et durée. Et avec l'ordre ENVELOPE suivi de 14 paramètres (pour définir complètement le son à produire), le BBC devient un véritable synthétiseur.

Les instructions PEEK et POKE ne sont pas présentes, mais sont remplacées par un point d'interrogation. Un N = PEEK(A) s'écrit N = ?A, alors que POKE A,N devient ?A = N. Pour agir sur 4 octets à la fois (un entier Basic est stocké ainsi), ce n'est plus « ? » mais « ! ».

La conversion en hexadécimal se fait par « ~ », et une liste hexa de la mémoire s'obtient par FOR A=0 TO &FFFF:PRINT~A,~?A:NEXT A.

Les variables HIMEM, LOMEM, TOP et PAGE fournissent les différents pointeurs de stockage du programme Basic et de la mémoire vive disponible. Ils peuvent être (sauf TOP) réinitialisés au gré de l'utilisateur. De l'espace mémoire peut-être réservé pour le langage-machine par un emploi spécial de l'ordre DIM.

Mais la particularité intéressante du BBC, que l'on retrouve sur les autres Acorn, est de pouvoir intégrer de l'Assembleur au milieu d'un programme Basic. Il suffit que le programme-source 6502 soit précédé d'un « [» pour que l'interpréteur Basic sache que ce qui suit est de l'Assembleur. Logiquement, un «] » doit clore le source. Les références symboliques sont possibles, et la sortie ou non des erreurs d'assemblage et du listing du code assemblé peut être commandé par OPT.

L'appel d'un programme en langage-machine peut se faire par CALL ou USR suivant le choix de passage des paramètres.

On a aussi accès directement à certains ordres de l'OS (Operating System) : l'ordre est alors précédé d'un astérisque « * ». Nous avons déjà vu *TV. Par *KEY, il est possible de définir les touches de fonction (touches rouges du clavier). Par *LOAD, *SAVE, *RUN, on peut gérer des fichiers langage-machine. Par *FX, on règle les débits de l'interface RS-423, l'autorepeat des touches, etc.

En conclusion, le Basic du BBC est l'un des Basic les plus complets et les plus rapides (1). Il est fort original, peut-être même trop : il s'éloigne souvent des « standards ». L'obligatoire rançon d'un Basic différent.

Christian BOYER

(1) Voir les dix tests de LIST appliqués au BBC, pages 59 et 60.

PROGRAMME PERFECTIONNÉ

UN TIC-TAC-TOE IMBATTABLE

DANS le numéro 2 de LIST, un programme de tic-tac-toe était présenté. Il était autodidacte : au début, il jouait à un niveau de rare nullité, puis il allait en se perfectionnant plus ou moins rapidement suivant le niveau du jeu. Cette fois-ci, le problème est pris à l'envers : le programme conçu joue tout de suite à la perfection !

Pour qu'un programme rende un ordinateur imbattable au tic-tac-toe, on peut imaginer une méthode qui consiste à explorer toutes les combinaisons de coups possibles.

Livrons-nous à un petit calcul. Au début d'une partie, les neuf cases de l'échiquier sont vides (les règles du jeu sont rappelées en encadré page suivante).

Le premier joueur a le choix entre ces neuf cases. Quand il a joué, son adversaire ne peut plus opter qu'entre huit cases. Restent alors sept cases entre lesquelles le premier joueur doit choisir son coup. Et ainsi de suite.

Il y a donc a priori $9\times8\times7\times6\times5\times4\times3\times2\times1$ (soit 9!, prononcer "factorielle de neuf") déroulements de parties possibles, soit encore 362 880. Il ne s'agit là que de parties mais chacune est composée de coups. Le calcul montre que le nombre de coups à simuler serait en fait de : $(9!/8!)+(9!/7!)+\ldots+(9!/1!)=623\ 529$.

Un seul coup gagnant suffit

Plusieurs remarques s'imposent :

- certaines parties sont terminées avant que le jeu ne soit "plein" ; il est alors inutile de continuer l'exploration, ce qui réduit le nombre de coups à simuler ;
- si l'on explore l'arbre des coups possibles grâce à l'algorithme minimax alpha-bêta, il sera possible d'élaguer certaines branches sans aucune perte d'information (élagage alpha-bêta) ;
- si l'on explore chaque groupe de coups dans un ordre cohérent — centre, coins, puis bords (1) — on atteint plus rapidement ceux qui donnent la victoire à un niveau donné. Lorsqu'on trouve un coup gagnant parmi un groupe de coups, il est inutile d'explorer les autres. On pourrait au mieux trouver un autre coup gagnant. Un seul suffit. De même, l'élagage alpha-bêta (voir bibliographie page suivante) aura lieu, en général, plus rapidement que si les coups étaient essayés dans un ordre quelconque.

En tenant compte de ces remarques,

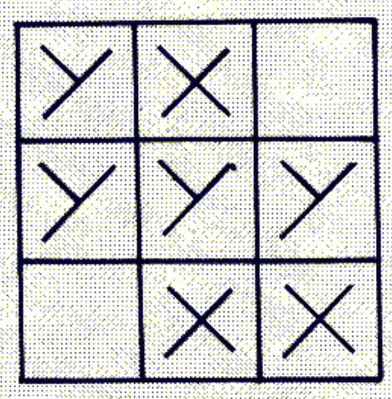
(1) A ce jeu, les meilleurs coups sont plus souvent au centre que dans les coins, et plus souvent dans les coins que sur les bords.

PROGRAMME PERFECTIONNÉ

UN TIC-TAC-TOE IMBATTABLE

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.



le programme gagne énormément de temps. Ainsi quand l'ordinateur commence, il trouve son premier coup en étudiant "seulement" 16 533 positions différentes au lieu des 623 529 théoriques.

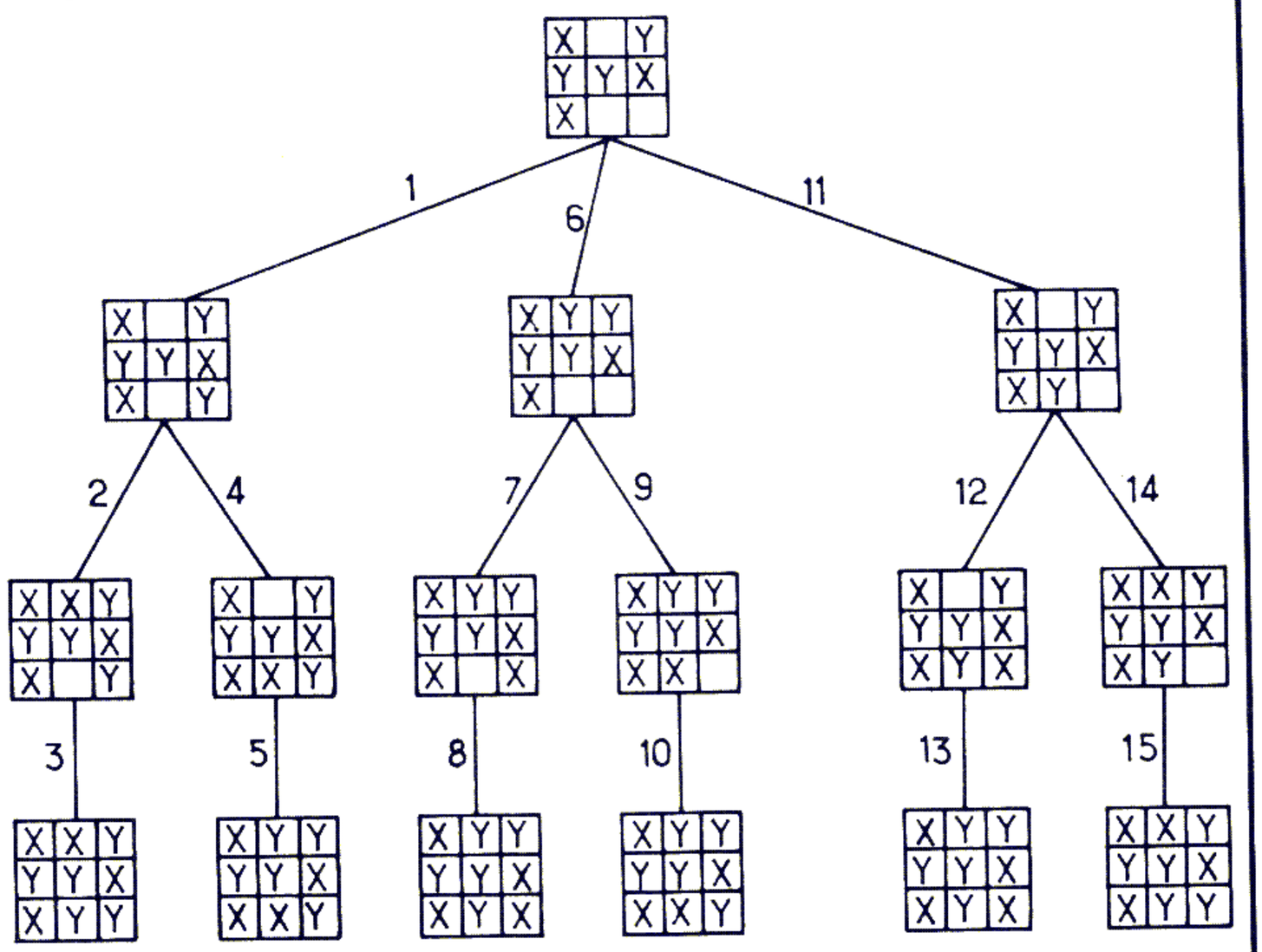
Un seul but : la victoire

Si vous commencez la partie, l'ordinateur se trouve en face d'un jeu qui ne contient plus que 8 cases vides et il n'a donc plus que 4 200 positions à étudier. C'est encore trop long. Et puis, lorsque l'on a découvert la perfection, il est inutile de la redécouvrir. C'est pourquoi, pour son premier coup (celui qui demande le plus de calculs), l'ordinateur vous propose de choisir pour lui entre un premier coup "de mémoire", c'est-à-dire instantané, ou un premier coup étudié parmi les 16 533.

Une autre option intéressante vous est proposée : la possibilité de voir s'afficher chaque position étudiée par la machine. Bien sûr, cela augmentera le temps de la recherche. Et à chaque fois que l'ordinateur annoncera son coup, il donnera le nombre de positions qu'il a étudiées pour arriver à ce résultat.

L'ARBRE EXPLORÉ

Un exemple d'exploration d'un arbre par le programme minimax est donné ci-dessous. La position de départ est fournie par la première grille, les chiffres correspondent à l'ordre d'exploration des coups.



Le coup 13 mène à la victoire, mais seulement si l'adversaire joue le coup 12 au niveau supérieur : ce serait une erreur de sa part. La position initiale ne vaut donc déjà pas mieux que le match nul.

Bibliographie

Une bonne description de l'algorithme alpha-bêta se trouve dans les livres suivants : "La programmation des jeux de réflexion" de Louis Jardonnet et "Techniques de programmation des jeux" de David Lévy (tous deux édités par PSI).

Jouer parfaitement ne signifie pas seulement trouver une combinaison de coups qui mène à la victoire ; cela signifie aussi trouver celle qui y mène *le plus rapidement*. Ce programme obtient un tel résultat en mettant une note d'autant plus positive à une position gagnante qu'elle intervient à un niveau peu profond dans l'exploration de l'arbre.

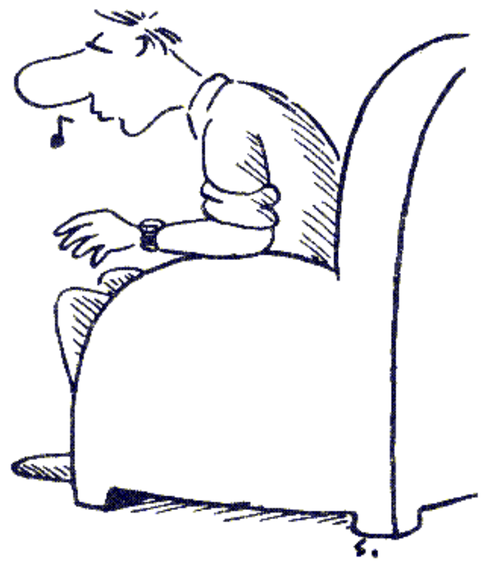
Thierry LÉVY-ABÉGNOLI

Meilleur coup au centre

Le jeu du tic-tac-toe pourrait être qualifié de jeu des "matchs nuls". Il est en effet possible pour chacun des deux joueurs d'arriver, au pire, au match nul. Un programme qui fonctionne selon l'algorithme alpha-bêta suppose que l'adversaire ne commet pas d'erreur.

Au début, tous les coups valent donc potentiellement le match nul. L'ordinateur ne choisit le centre que parce qu'il l'étudie avant les autres.

Par la suite, il pourra y avoir des coups qui mèneront, à terme, à la victoire. Une telle éventualité ne peut avoir lieu que si le joueur — humain — commet une erreur. Et si le cas se présente, l'ordinateur la détectera toujours.



Tic-tac-toe à la perfection
Programme en Basic Microsoft
Auteur Thierry Lévy-Abégnoli

Liste des variables

C(i)	représente l'état du jeu
S(i)	représente la valeur potentielle du groupe de coups en cours d'étude au niveau i
CX(i)	désigne le camp qui joue au niveau i
I(i)	pointe sur le coup étudié au niveau i
O(I(i))	case pointée par I(i)
P	niveau étudié
N	nombre de positions étudiées
M	meilleur coup
C R	nombre de cases pleines
CM(i)	contient la case du coup simulé au niveau i

```
7 REM------------------
8 REM  INITIALISATION
9 REM------------------
10 CLS:REM EFFACE L'ECRAN
20 DEFINT A-Z
30 DIM C(12),S(12),CX(12),I(12),A$(2)
35 V=0:I=0:P=0:REM CETTE LIGNE ACCELERE LE PROGRAMME
 EN DECLARANT EN PREMIER LES VARIABLES LES PLUS UTIL
ISEES
40 DATA 5,1,3,9,7,2,6,8,4
50 FOR I=1 TO 9:READ O(I):NEXT I
60 VRAI=-1:FAUX=1
70 FOR I=0 TO 9 STEP 2:MACH(I)=FAUX:MACH(I+1)=VRAI:N
EXT I
80 A$(0)=" X":A$(2)=" Y"
90 DATA-1,1,-1,1,-1,1,-1,1,-1,1,-1,1,-1,1,-1
95 FOR I=0 TO 12:READ CX(I):NEXT I
99 REM----CHOIX DU JOUEUR
100 INPUT "VOULEZ-VOUS VOIR APPARAITRE A L'ECRAN LES
 POSITIONS QUE J'ETUDIE (O OU N)";D$:INPUT "VOULEZ-V
OUS COMMENCEZ (O OU N)";A$
105 GOSUB 5000
110 IF A$="O" THEN 145
115 IF A$="N" THEN 125
120 GOTO 100
125 INPUT "VOULEZ-VOUS GAGNER UNE HEURE ENVIRON EN F
AISANT JOUER LE PREMIER COUP SUR BIBLIOTHEQUE D'OUVE
RTURE (O OU N)";B$
130 IF B$="O" THEN M=5:GOTO 260
135 IF B$="N" THEN 250
140 GOTO 125
145 INPUT "VOULEZ-VOUS GAGNER 15 MINUTES ENVIRON EN
FAISANT JOUER LE PREMIER COUP SUR BIBLIOTHEQUE D'OUV
ERTURE (O OU N)";C$
150 IF C$="O" THEN 180
155 IF C$="N" THEN 180
160 GOTO 145
177 REM-----------------------
178 REM JOUEUR CHOISIT SON COUP
179 REM-----------------------
180 INPUT "OU JOUEZ-VOUS";I
185 IF I>9 OR I<1 THEN 180
190 IF C(I)<>0 PRINT"CASE PLEINE":GOTO 180
200 P=12:CR=CR+1:V=CX(P):C(I)=V
210 GOSUB 1010:GOSUB 5000:IF S(12)<>0 THEN 3000
215 IF CR=9 THEN 3200
217 IF CR<>1 THEN 250
220 IF C$="N" THEN 250
230 IF C(5)=0 LET M=5:C$="":GOTO 260
240 M=1:C$="":GOTO 260
247 REM-------------------------
248 REM MACHINE CHOISIT SON COUP
249 REM-------------------------
250 GOSUB 300
260 P=11:CR=CR+1:V=CX(P):C(M)=V:I=M
265 PRINT"JE JOUE EN";M:PRINT"POUR DETERMINER CE COU
P J'AI ETUDIE";N;" POSITION(S) DIFFERENTE(S)":N=0
270 GOSUB 1010:GOSUB 5000:IF S(11)<>0 THEN 3100
275 IF CR=9 THEN 3200
280 GOTO 180
290 GOTO 180
292 REM--------------------------------
293 REM   S/P D'EXPLORATION DE L'ARBRE
294 REM    DES COMBINAISONS DE COUPS
295 REM SELON LES PRINCIPES DU MINIMAX
299 REM--------------------------------
300 P=1
310 S(P-1)=CX(P-1)*20
320 I(P)=0
330 I(P)=I(P)+1
340 IF I(P)=10 THEN 430
350 IF C(O(I(P)))<>0 THEN 330
360 GOSUB 1000
370 IF ABS(S(P))>0 THEN 400
380 IF CR=9 THEN 450
390 P=P+1:GOTO 310
400 GOSUB 2000
410 IF P=1 THEN M=O(I(1)):RETURN
420 S(P-1)=S(P)
430 IF P=1 RETURN
440 P=P-1
450 GOSUB 2000
460 IF MACH(P)=VRAI THEN 500
470 IF S(P)<=S(P-2) LET P=P-1:GOSUB 2000:GOTO 330
480 IF S(P)>=S(P-1) THEN 330
490 GOTO 520
500 IF S(P)<=S(P-1) THEN 330
510 IF P=1 LET M=O(I(1))
520 S(P-1)=S(P):GOTO 330
994 REM--------------------------------
995 REM S/P COUP SIMULE EN CASE C(O(I(P)))
996 REM A LA PROFONDEUR DE RECHERCHE P PAR
997 REM LE JO(P) AVEC TEST DE VICTOIRE
999 REM--------------------------------
1000 CR=CR+1:V=CX(P):I=O(I(P)):C(I)=V:CM(P)=I
1005 N=N+1:IF D$="O" PRINT"J'ETUDIE LA POSITION SUIV
ANTE:":GOSUB 5000
1010 ON I GOTO 1100,1200,1300,1400,1500,1600,1700,18
00,1900
1099 REM------COUP JOUE EN CASE 1
1100 IF C(2)=V IF C(3)=V THEN 1950
1110 IF C(4)=V IF C(7)=V THEN 1950
1120 IF C(5)=V IF C(9)=V THEN 1950
1130 S(P)=0:RETURN
1199 REM------COUP JOUE EN CASE 2
1200 IF C(5)=V IF C(8)=V THEN 1950
1210 IF C(1)=V IF C(3)=V THEN 1950
1220 S(P)=0:RETURN
1299 REM------COUP JOUE EN CASE 3
1300 IF C(1)=V IF C(2)=V THEN 1950
1310 IF C(5)=V IF C(7)=V THEN 1950
1320 IF C(6)=V IF C(9)=V THEN 1950
1330 S(P)=0:RETURN
1399 REM------COUP JOUE EN CASE 4
1400 IF C(1)=V IF C(7)=V THEN 1950
1410 IF C(5)=V IF C(6)=V THEN 1950
1420 S(P)=0:RETURN
1499 REM------COUP JOUE EN CASE 5
1500 IF C(1)=V IF C(9)=V THEN 1950
1510 IF C(2)=V IF C(8)=V THEN 1950
1520 IF C(3)=V IF C(7)=V THEN 1950
1530 IF C(4)=V IF C(6)=V THEN 1950
1540 S(P)=0:RETURN
1599 REM------COUP JOUE EN CASE 6
1600 IF C(4)=V IF C(5)=V THEN 1950
1610 IF C(3)=V IF C(9)=V THEN 1950
1620 S(P)=0:RETURN
1699 REM------COUP JOUE EN CASE 7
1700 IF C(1)=V IF C(4)=V THEN 1950
1710 IF C(5)=V IF C(3)=V THEN 1950
1720 IF C(8)=V IF C(9)=V THEN 1950
1730 S(P)=0:RETURN
1799 REM------COUP JOUE EN CASE 8
1800 IF C(2)=V IF C(5)=V THEN 1950
1810 IF C(7)=V IF C(9)=V THEN 1950
1820 S(P)=0:RETURN
1899 REM------COUP JOUE EN CASE 9
1900 IF C(1)=V IF C(5)=V THEN 1950
1910 IF C(7)=V IF C(8)=V THEN 1950
1920 IF C(3)=V IF C(6)=V THEN 1950
1930 S(P)=0:RETURN
1944 REM--------------------------------
1945 REM EVALUATION D'UNE POSITION GAGNANTE
1946 REM POUR UN DES CAMPS, NOTE D'AUTANT PLUS
1947 REM ELEVEE (EN VALEUR ABSOLUE) QUE LA
1948 REM VICTOIRE EST PROCHE
1949 REM--------------------------------
1950 S(P)=(14-P)*CX(P)
1960 RETURN
1995 REM--------------------------------
1996 REM   S/P DE DEMODIFICATION DU JEU
1997 REM LORS DE LA REMONTEE DANS L'ARBRE:
1998 REM TRAVAIL INVERSE DU S/P COUP JOUE
1999 REM--------------------------------
2000 CR=CR-1:C(CM(P))=0
2010 RETURN
2990 REM--------------
2991 REM FIN DE PARTIE
2992 REM--------------
2993 REM----LE JOUEUR A GAGNE
2994 REM    (CE MODULE EST INUTILE CAR
2995 REM    CE PROGRAMME EST IMBATTABLE!)
3000 PRINT"VOUS AVEZ GAGNE":GOTO 4000
3099 REM----LA MACHINE A GAGNE
3100 PRINT "J'AI GAGNE":GOTO 4000
3199 REM----MATCH NUL
3200 PRINT  "MATCH NUL"
4000 PRINT "POUR REFAIRE UNE PARTIE APPUYEZ SUR <ENT
ER>"
4010 INPUT A$:RUN
4995 REM-----------------
4996 REM AFFICHAGE DU JEU
4997 REM-----------------
5000 PRINT"-------------"
5010 FOR I4=0 TO 6 STEP 3
5020 FOR I5=0 TO 2
5030 IF C(I4+I5+1)=0 LET V$=STR$(I4+I5+1):GOTO 5050
5040 V$=A$(C(I4+I5+1)+1)
5050 PRINT"!";V$;" " ;
5060 NEXT I5
5070 PRINT"!"
5080 NEXT I4
5090 PRINT"-------------"
5100 RETURN
```

Inutile d'essayer, le programme ne passera jamais par les lignes 2993 à 3000 !

MCODER II
COMPILATEUR POUR SPECTRUM

MCODER II est un compilateur Basic pour le Spectrum dans sa version 48 K. Comme il est dit dans la notice, c'est « un outil sophistiqué et versatile dont il ne faut cependant pas attendre de miracles ». Néanmoins une bonne maîtrise de ce compilateur permet d'obtenir une vitesse d'exécution remarquable pour certains programmes.

Distribué sur cassette, *Mcoder II* est accompagné d'une légère notice (accordéon fait de quatre petits feuillets) glissée dans le boîtier de la cassette. La notice dont nous disposions était écrite en anglais. *Mcoder II* est chargé sous forme de Code d'une longueur d'environ 5 Koctets, implanté à partir de l'adresse 59990. Le compilateur ne pourra donc être chargé que sur un Spectrum disposant de 48 Koctets de mémoire vive.

Que fait ce compilateur ? Il transcrit en langage-machine un programme Basic pour que, lors de l'exécution, il n'y ait plus besoin d'interpréter chaque ligne de programme. Cette interprétation est faite une fois pour toutes lors de la compilation et l'emplacement des variables est alloué pendant cette même phase.

Pour ce qui est des nombres, *Mcoder* ne traite que des entiers signés codés sur deux octets (de −32768 à 32767). Cette limitation rendra difficile la comparaison des vitesses d'exécution d'un programme Basic Spectrum (les nombres sont codés en virgule flottante sur cinq octets) avec le programme équivalent compilé. Par ailleurs les programmes nécessitant des variables ou nombres autres qu'entiers ne pourront être compilés sans donner des erreurs à l'exécution.

D'autres restrictions du compilateur obligent malheureusement à réadapter des programmes Basic déjà écrits.

Ainsi le STEP ne peut pas être inclus dans une boucle FOR... NEXT. Le pas est toujours implicitement de 1. Pour introduire un pas négatif, il faut alors procéder à une gymnastique telle que la réalisent les lignes 105 et 110 du programme 1 de test. De plus, à la différence du Basic Spectrum, les lignes à l'intérieur d'une boucle sont exécutées au moins une fois.

Des temps d'exécution records

Les tableaux numériques ne peuvent comporter qu'une seule dimension dans un programme destiné à la compilation. Les tableaux de chaînes de caractères ne sont pas reconnus.

Donc, soit pour compiler vos programmes Basic préexistants, soit pour écrire des programmes pouvant être compilés d'emblée, vous aurez à bien connaître les possibilités et les limites de *Mcoder*. Dans certaines applications

Programme 1
Tri par échange d'une suite de nombres

```
  1 REM #1
  5 INPUT "Combien de nombres ";s
 10 CLS
 15 POKE 23692,255
 20 DIM t(s)
 30 LET nn=s
 40 LET n1=nn+1
 50 FOR n=1 TO nn: LET t(n)=n1-n: NEXT n
 60 PRINT "Suite initiale"
 70 GO SUB 500
 80 FOR i=2 TO nn
 90 LET j=nn
100 IF t(j-1)>t(j) THEN  LET x=t(j-1): LET t(j-1)=t(j): LET t(j)=x
105 LET j=j-1
110 IF j >= i THEN  GO TO 100
120 NEXT i
130 PRINT "Apres tri par echange"
140 GO SUB 500
150 GO TO 1000
500 FOR n=1 TO nn: PRINT t(n);" ";: NEXT n: PRINT
510 RETURN
1000 PRINT "Fin."
```

Tri de 100 nombres en *Basic* : 3 mn 30 sec
avec *Mcoder* : 0 mn 03 sec

Tri de 500 nombres avec *Mcoder* : 1 mn 07 sec

Programme 2
Inversion des 45000 pixels de l'écran

```
 1 REM #1
10 CLS
20 FOR x=1 TO 22
30 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ123456"
40 NEXT x
50 FOR y=0 TO 175
60 FOR x=0 TO 255
70 PLOT  OVER  POINT (x,y);x,y
80 NEXT x
90 NEXT y
```

Temps d'exécution en *Basic* : 9 mn 45 sec
avec *Mcoder* : 1 mn 08 sec

Programme 3
Tracé de cercles

```
 5 CLS
10 LET n=1
20 CIRCLE 127,87,n
30 LET n=n+3
40 IF n<87 THEN  GO TO 20
```

Temps d'exécution en *Basic* : 22 sec
avec *Mcoder* : 21 sec

Programme 4
Tracé de droites

```
10 CLS
30 FOR x=1 TO 175
40 DRAW x,0
50 DRAW 0,x
60 DRAW -x,0
70 DRAW 0,-x
80 NEXT x
```

Temps d'exécution en *Basic* : 15 sec
avec *Mcoder* : 13 sec

Programme 5
Impression de nombres

```
10 CLS
20 FOR n=0 TO 1000: POKE 23692,255: PRINT n;" ";: NEXT n
```

Temps d'exécution en *Basic* : 24 sec
avec *Mcoder* : 8 sec

vous pourrez apprécier la puissance du compilateur.

Pour illustrer les performances de *Mcoder*, les temps d'exécution de huit programmes ont été chronométrés (voir ci-contre et page suivante), dans leur version originale et dans leur version compilée. En Basic compilé, on ne peut travailler qu'avec des entiers. Il a donc fallu choisir les tests en conséquence.

Des tests bien adaptés

Les huit programmes utilisés ici ne prétendent pas évaluer toutes les performances du compilateur. Ils servent plutôt à mettre en évidence les applications où le programme compilé est beaucoup plus rapidement exécuté que le programme interprété.

Les temps d'exécution ne sont qu'indicatifs. Le facteur de gain n'a pas été calculé lorsque l'incertitude sur une mesure était trop grande.

Le premier programme semble beaucoup plus rapide dans sa version compilée, mais il faut encore rappeler que *Mcoder* travaille sur des nombres codés sur deux octets au lieu de cinq. Néanmoins le tri de 500 nombres avec *Mco-*

MCODER II POUR SPECTRUM

Programme 6
Impression de caractères

```
 1 FOR y=1 TO 10
10 CLS
20 FOR n=1 TO 88
30 FOR x=0 TO 7
40 PRINT  PAPER x;" ";
50 NEXT x
60 NEXT n
70 NEXT y
```

Temps d'exécution en *Basic* : 1 mn 10 sec
avec *Mcoder* : 0 mn 08 sec

Programme 7
Calculs

```
 1 FOR x=1 TO 10
 5 LET a=0
10 FOR n=1 TO 250
20 LET a=a+n
30 NEXT n
35 NEXT x
40 PRINT a
```

Temps d'exécution en *Basic* : 20 sec
avec *Mcoder* : < 1 sec

```
10 FOR x=33 TO 127
20 CLS
30 PRINT  AT 21,0; CHR$ x
40 FOR n=0 TO 7
50 FOR z=0 TO 7
55 FOR f=0 TO 1
60 IF  POINT (z,7-n)=1 THEN  PRINT  AT 4+n*2+f,8+z*2; CHR$ 143; CHR$ 143
65 NEXT f
70 NEXT z
80 NEXT n
90 NEXT x
```

Programme 8
Graphiques

Temps d'exécution en *Basic* : 4 mn 22 sec
avec *Mcoder* : 0 mn 22 sec

der est encore trois fois plus rapide que le tri de 100 nombres en Basic interprété.

Les deuxième et huitième programmes utilisent les fonctions PLOT et POINT. Les programmes compilés montrent là leur très grande supériorité et l'intérêt du compilateur pour de telles applications.

Performant malgré ses limites

L'impression de nombres et de caractères est testée par les programmes 5 et 6. La remarque concernant les entiers s'applique encore au programme 5.

On note cependant que le tracé de cercles ou de droites n'est pas beaucoup plus rapide avec la version compilée, car ces fonctions sont intrinsèquement lentes (programmes 3 et 4).

Lors de la compilation, les lignes Basic sont listées une à une, le programme objet est placé au-dessus de RAMTOP, au niveau que vous avez donné. L'adresse d'appel du programme compilé est affichée. Si une commande Basic n'est pas reconnue, la compilation s'arrête et la ligne concernée est précisée.

Une fonction TRACE est disponible. L'option 1 crée un programme compilé qui pourra être arrêté par la touche BREAK lors de l'exécution. Le programme-objet créé avec l'option 0 ne teste pas la touche BREAK. L'exécution de ce programme est alors plus rapide (mais la différence est souvent insignifiante). Attention aux boucles infinies dans ce cas !

Le programme-objet, pour être sauvegardé sur cassette, doit l'être avec les 5 Koctets du compilateur, car celui-ci contient des routines nécessaires à l'exécution. Ceci est un peu lourd, surtout si le programme-objet est court.

Malgré toutes ses limites, *Mcoder II* augmente les performances de nombreuses applications. Il coûte environ 120 FF ttc.

Le logiciel en quelques lignes

Nom : *Mcoder II*
Ordinateur : *Spectrum 48 K*
Forme : *cassette*
Edité par : *Personal Software Services*
Distribué par : *Innelec*
Prix : *120 FF ttc*
Principale orientation : *compilation*
Occupation mémoire : *5 Koctets*

Tableau des résultats des tests

Programme	1 Tri	2 Pixels	3 Cercle	4 Droite	5 Nombre	6 Alpha	7 Calcul	8 Graph.
Mcoder	3	68	21	13	8	8	< 1	22
Basic	210	585	22	15	24	70	20	262
Rapport entre les temps		8	1	1	3	9		12

Les temps sont exprimés en secondes.

Benoît THONNART

CARACTOR

CRÉATION GRAPHIQUE POUR TO7

EDITÉ sous forme de cartouche, le logiciel Caractor offre à l'utilisateur d'un TO7 ou d'un TO7/70 la possibilité de créer des dessins à l'écran et de les sauvegarder. Même les débutants y ont accès : Caractor est facile d'emploi. Un seul obstacle, son prix : 750 FF ttc.

C'est dans la collection Création de TO TEK qu'est éditée la cartouche de logiciel *Caractor*. Ce logiciel est en effet une aide à la création graphique, même sophistiquée. Mais *Caractor* permet aussi de sauvegarder les œuvres ainsi créées en les transformant, selon la demande, en programmes en Basic ou en Assembleur. Et là, il devient un outil de programmation.

Après avoir inséré la cartouche dans le TO7 (ou le TO7/70) et sélectionné l'option Caractor, une page d'écran apparaît. Elle comprend un premier menu, le menu principal, comportant des « touches » de commande. Pour accéder à l'une ou à l'autre, il suffit de la pointer à l'aide du crayon optique. A noter une agréable nouveauté, l'approche du crayon (sans toucher) sur l'une de ces touches fait apparaître en écho un carré bleu foncé clignotant. Ceci permet d'éviter la relative imprécision de ce crayon (dûe bien sûr à la fatigue des longues nuits de programmation !).

Le menu principal comprend quatre options : DESSIN, TABLEAU, CONSULTATION, FICHIER.

L'option DESSIN est celle qui permet une véritable création graphique. Elle affiche à l'écran une grille de 32 × 24 cases dans laquelle il est possible de dessiner point par point, d'effacer, de grossir une zone, de la déplacer... (photo 1).

La grille ne correspond qu'à une zone de l'espace disponible pour le dessin. Elle est de douze caractères graphiques (chaque caractère étant une matrice de huit cases sur huit), alors que l'espace total dans lequel on peut dessiner en contient 96 (8 sur 12). Cet espace est représenté à droite de la grille, pour visualiser le dessin créé au fur et à mesure.

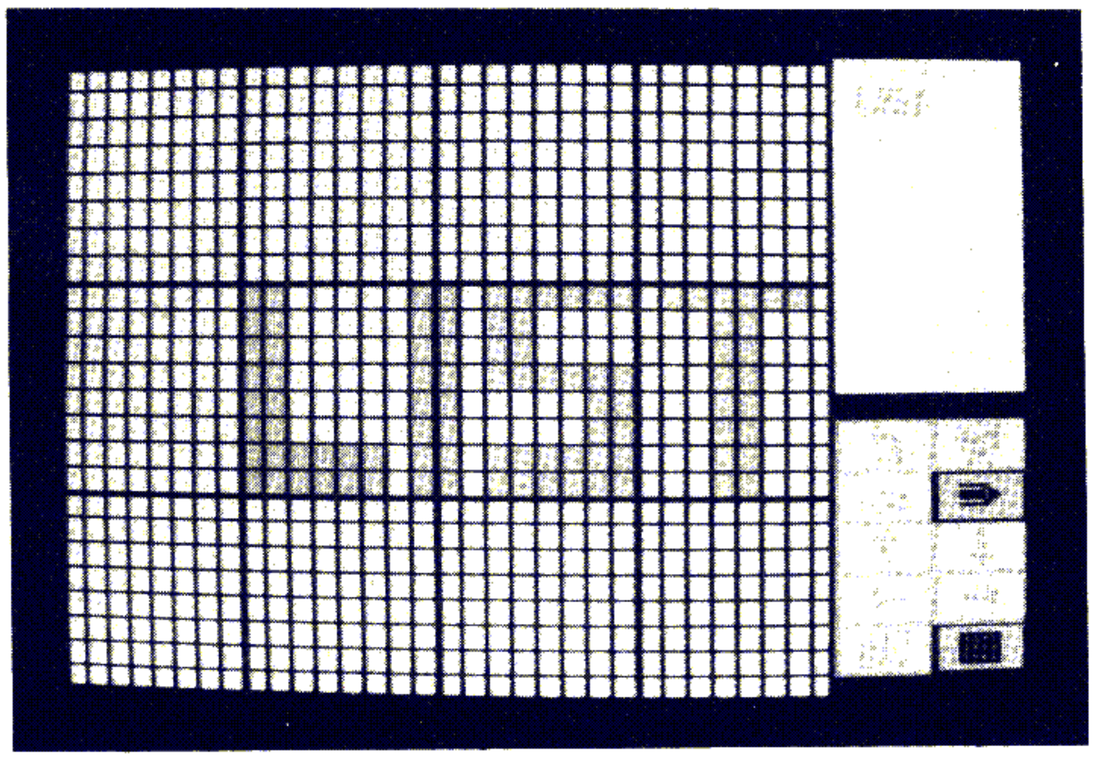


Photo 1
L'option DESSIN offre une grille pour dessiner, l'espace total disponible, et un menu

Des crayons et des couleurs

Enfin, un menu propose les différentes fonctions graphiques :

- un crayon pour dessiner point par point ;
- une gomme pour effacer ;
- une loupe (fonction ZONE) pour définir une zone rectangulaire de la grille ; il est alors possible de se déplacer dans le dessin grâce à quatre touches de direction ;
- une grille pour revenir à la définition de départ.

Ce menu propose aussi deux commandes de changement d'option : l'une revient au menu principal et l'autre accède à l'option TABLEAU.

Cette option TABLEAU propose une palette de couleurs et permet « d'af-

CARACTOR POUR TO7

ner » le dessin. Sur une zone délimitée, il est possible d'assembler (rapprocher les caractères pour former un motif), de désassembler, de donner un nom à la zone assemblée, de colorer le fond ou le motif (grâce à l'une des couleurs de la palette).

Sur un seul caractère graphique, le choix de TABLEAU permet un travail plus précis encore : suppression sans perte (le caractère est récupérable), copie à un autre endroit, rotation d'un quart de tour, superposition, etc.

Si le dessin est terminé, il faut retourner au menu principal pour entrer dans le mode CONSULTATION ou dans le mode FICHIER.

La CONSULTATION offre effectivement une consultation à l'écran ou sur l'imprimante des différents motifs assemblés et l'examen de tous les caractères graphiques qui les composent.

On peut alors consulter soit un élément du tableau en le pointant avec le stylo, soit l'ensemble des éléments (un élément étant un motif assemblé ou un caractère graphique). On peut aussi avoir sa représentation ou son codage Basic, décimal et hexadécimal (photo 2).

Quant à l'option FICHIER, elle met en liaison l'unité centrale du TO7 avec ses périphériques de stockage, lecteur de cassette ou lecteur de disquette.

Elle propose ainsi de formater une disquette, de charger ou de sauver un dessin. La sauvegarde prend quatre formes :

- le codage interne, C01, seul mode qui permette de recharger le dessin ;
- l'option CARactère devrait générer un fichier qui peut être réutilisé dans les autres outils de création graphique (on a essayé avec PICTOR, mais ça n'a pas marché !) ;
- l'option BASic, tout simplement formidable ! Elle génère automatiquement un sous-programme en Basic (photo 3) qui pourra être utilisé pour toute autre application ;
- l'option Assembleur, ASM, comme l'option BAS génère un sous-programme en Assembleur.

Cher mais... étonnant

En résumé, la cartouche *Caractor* représente une aide inestimable à la conception de « programmes graphiques » sur TO7 ou TO7/70. On peut regretter que le manuel d'utilisation ne soit pas aussi efficace : il est un peu fouillis. On y trouve bien tous les renseignements nécessaires, mais s'agissant d'un logiciel qui, sans être complexe, est néanmoins difficile d'accès, on aurait souhaité plus de pédagogie.

Un dernier point d'importance pour les utilisateurs du TO7 ou du TO7/70, le prix de cette cartouche : 750 FF ttc. C'est un des logiciels les plus chers existant pour ces ordinateurs, mais aussi un des plus étonnants. Encore quelques longues nuits, en couleurs celles-là, à passer devant la machine.

Jacques LABIDURIE

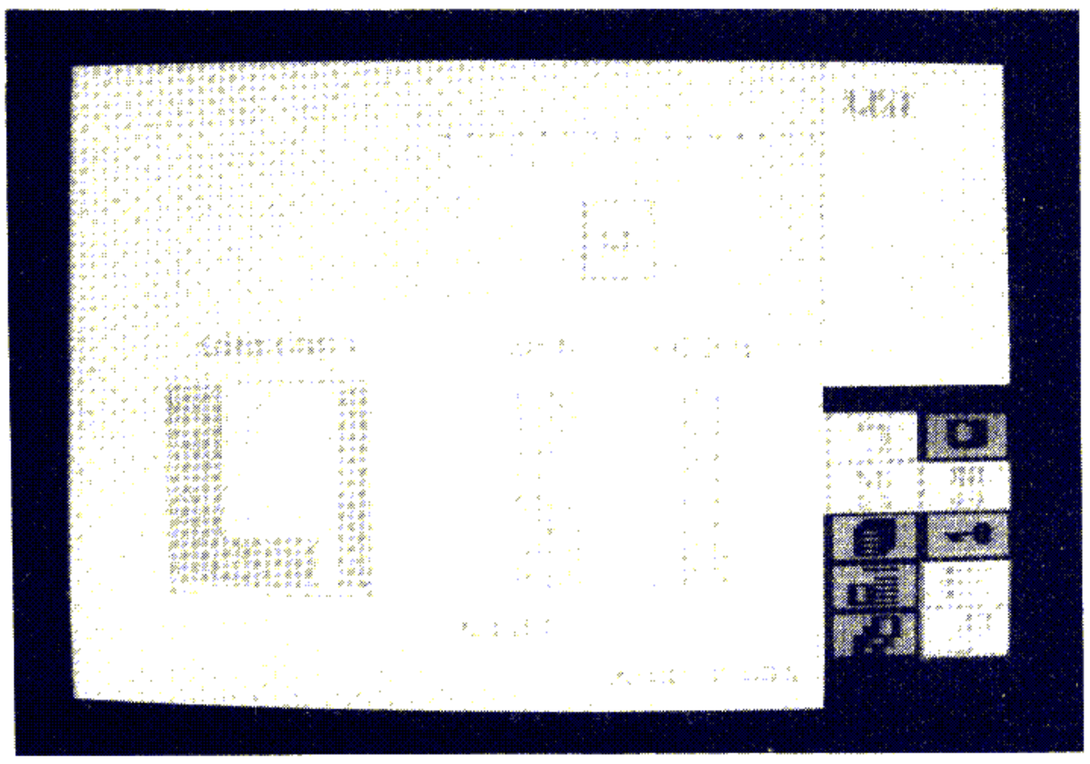


Photo 2
Le codage du caractère de coordonnées X=1, Y=01 de la photo 1

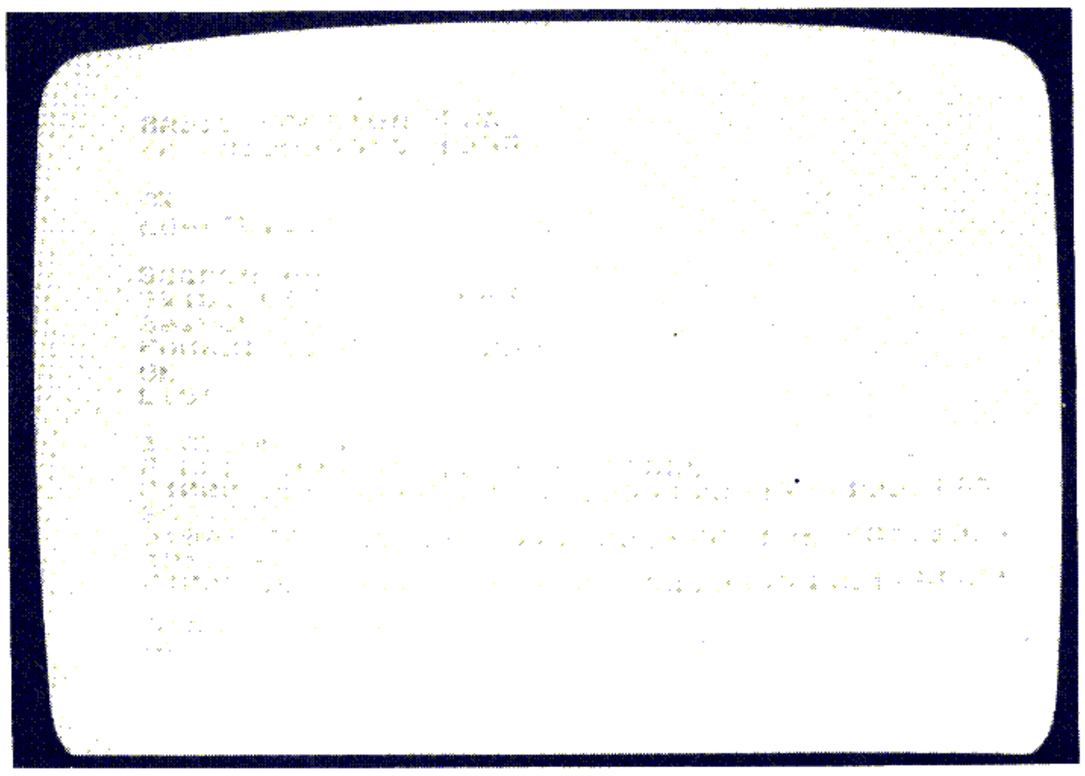


Photo 3
Le sous-programme Basic généré par Caractor

Le logiciel en quelques lignes

Nom : Caractor
Ordinateur : TO7 et TO7/70
Forme : cartouche
Edité par : TO TEK
Distribué par : les revendeurs THOMSON
Prix public : 750 FF ttc
Principale orientation : création graphique
Autres orientations : génération de sous-programmes graphiques en Basic ou en Assembleur

ASSEMBLER
MACRO-ASSEMBLEUR POUR COMMODORE 64

LE macro-assembleur Assembler est édité sous label officiel Commodore. Ce logiciel se présente sous la forme d'une pochette mince renfermant les deux éléments fondamentaux que sont la disquette contenant les programmes utiles, et la documentation. Son utilisation impose au moins deux choses : la possession d'une unité de disquettes et de bonnes connaissances du langage Assembleur.

Le macro-assembleur *Assembler* pour Commodore 64 est un logiciel « sérieux ». Pour mieux l'aborder, il faut commencer par jeter un coup d'œil approfondi sur la documentation. Elle est constituée d'un livret de 70 pages environ, au format 15×21 cm, auquel s'ajoute un feuillet glissé en première page, constituant un correctif à quelques indications incomplètes ou erronées du livret.

L'objet de notre essai étant en version anglaise (non sous-titrée), personne ne s'étonnera du fait que la documentation soit intégralement rédigée en anglais. Vous serez en droit d'exiger de votre revendeur la livraison d'une version « bien de chez nous », qui rendra votre apprentissage plus aisé (cette version s'appelle *Assembleur 64* et elle est distribuée par Procep).

La documentation paraît assez touffue, en effet... et la préface ne manque pas d'indiquer dès ses premières lignes que (je cite) : « le logiciel et sa documentation sont destinés aux utilisateurs expérimentés, familiarisés au langage d'assemblage de la série 6500, et à la manipulation du Commodore 64 ». Nous voici donc prévenus...

Le manuel de référence est divisé en quatre chapitres principaux.

Le premier, sous le titre d'*Introduction*, rappelle le fonctionnement des assembleurs en général, et les conventions d'écriture et d'utilisation pour *Assembler* en particulier.

Le second chapitre constitue le mode d'emploi des programmes permettant la création et l'édition de programmes-sources.

Le suivant décrit l'art d'obtenir un programme-objet à partir d'un programme-source (assemblage), de le tester, de le corriger, etc.

Enfin, le dernier chapitre rassemble sur 26 pages divers éléments utiles : une carte mémoire du système, une autre carte du registre d'entrées/sorties, la liste des mnémoniques 6500, la liste des messages d'erreurs d'*Assembler*, etc. Gageons que ce dernier chapitre sera le plus utile quand la phase d'apprentissage sera dépassée.

Le logiciel en quelques lignes

Nom : *Assembler en version anglaise, Assembleur 64 en version française*
Ordinateur : *Commodore 64*
Forme : *disquette*
Edité par : *Commodore*
Distribué par : *Procep pour la version française*
Prix public : *version anglaise, 225 FF ttc ; version française, 350 FF ttc*
Principale orientation : *macro-assemblage*
Autres orientations : *moniteur, chargeur, éditeur, table de références croisées*

Le catalogue du logiciel

Ces différents chapitres sont relativement bien documentés, avec des exemples fournis chaque fois que c'est nécessaire. L'ensemble est donc pratique et tout à fait utilisable. Il manque toutefois un exemple de programme-source simple, qui serait utile aux débutants ; et, plus prosaïquement, un rappel du titre du chapitre en haut de page. Le manuel n'est visiblement pas destiné aux débutants : ces derniers devront se documenter sérieusement avec d'autres ouvrages plus pédagogiques.

Quant au logiciel, il semble assez

ASSEMBLER POUR C.64

complet. Voyons un peu. Le catalogue de la disquette (figure 1) nous indique la présence de dix programmes ou chargeurs de programmes. Le premier est bien connu de tous les utilisateurs de disquettes puisqu'il s'agit du programme *Dos Wedge*, qui ajoute au C.64 quelques commandes disques fort utiles et, surtout, simplifie l'usage des commandes habituelles.

Editor 64 est le programme éditeur qui sert à la mise au point du source, en ajoutant des commandes indispensa-

Figure 1
Les programmes de la disquette

```
1000 ; TEST ASSEMBLER 64 POUR LIST
1010 FOND=$D021;ADRESSE FOND ECRAN
1020 CDRE=$D020;ADRESSE CADRE ECRAN
1030 ;
1040 .OPT LIST,ERR,GEN
1050 *=$8000     ;IMPLANTATION PROGRAMME =32768
1060 ;
1070 JSR $E544   ;ROUTINE CLR/HOME
1080 LDA #$01
1090 STA FOND    ;FOND BLANC
1100 LDA #$00
1110 STA CDRE    ;CADRE NOIR
1120 ;
1130 LDY #176    ;176 FOIS
1140 BOUCL LDX #$04    ;5 LETTRES
1150 PRINT LDA MESSG,X
1160 JSR $FFD2   ;ROUTINE PRINT
1170 DEX
1180 BPL PRINT
1190 TYA         ;CHANGE COULEURS TEXTE
1200 AND #%00001111
1210 STA 646     ;#646=ADR.COUL.TEXTE
1220 DEY
1230 BNE BOUCL   ;ET CAETERA
1240 RTS
1250 ;
1260 MESSG .BYTE '*TSIL'
1270 .END
READY.
```

Figure 2
Un exemple de programme-source, à l'usage des novices

bles (de celles qui — soit dit en passant — manquent cruellement sur le C.64 de base) comme : AUTO, CHANGE, DELETE, FIND, NUMBER... et d'autres spécifiques à l'assemblage. Une fois créé, le source est rangé sur disquette sous la forme d'un fichier séquentiel.

Assembler 64 traduit le fichier-source en fichier-objet contenant les données nécessaires à la génération du programme en langage-machine (assemblage).

Loloader et *Hiloader* sont deux programmes chargeurs semblables qui peuvent s'implanter à deux endroits différents de la mémoire. Leur fonction est de traduire le fichier-objet créé par *Assembler* en une écriture réelle en mémoire, à l'adresse spécifiée.

Les deux *Monitor* sont des moniteurs langage-machine indispensables à la recherche et à la correction des erreurs commises dans le programme. Ils contiennent un assembleur immédiat (qui ne traite pas les étiquettes et les symboles), un désassembleur, et de nombreuses commandes d'interrogation mémoire, remplissage et déplacement de zones, commandes de périphériques, etc. Tout le nécessaire pour un travail de débogage (!) efficace.

Du sérieux avec souplesse

Crossref est utilisé pour obtenir une table des références croisées du programme. Son usage pourra se révéler utile.

Quant à *Boot All*, il permet de charger et lancer dans la foulée les programmes *Dos*, *Loader*, et *Editor*. Il simplifie donc les manipulations nécessaires.

A l'usage, *Assembler* se montre performant à bien des égards. La création du programme-source sous le contrôle d'*Editor* est facile et rapide (figure 2). Les opérandes numériques peuvent être écrits en décimal, hexadécimal, binaire, et même octal. La syntaxe employée est standard. L'utilisation d'étiquettes (six caractères maximum) et une dizaine de directives d'assemblage (réservation d'octets en mémoire, options de listage...) apportent toute la souplesse indispensable. Il est même possible à l'utilisateur de définir des macro-instructions. Voilà qui est sérieux !

Le programme-source étant écrit et sauvegardé en fichier, l'assembleur peut alors exécuter sa tâche. Celle-ci se déroule en deux passes successives, rapidement. La liste du texte source, com-

plétée des codes correspondants, défile à l'écran. La sélection du mode HARD COPY (impression) rendra des services à ceux qui possèdent une imprimante. Les autres devront avoir les réflexes prompts pour réagir à temps aux éventuels messages d'erreurs. La demande de création de la table des références croisées est possible, et sera exploitée par la suite grâce à *Crossref*.

Une erreur : tout à refaire

Si tout s'est déroulé sans qu'apparaisse un message d'erreur fatal, le chargeur intervient ensuite pour implanter les octets en mémoire à l'emplacement souhaité.

Pour parachever l'œuvre, l'utilisation de *Monitor* est utile, au moins pour vérifier que l'implantation est réalisée correctement.

C'est alors que le plus délicat reste à faire : tester le bon fonctionnement du programme ! Si vous êtes fort, ou si vous avez de la chance, votre programme marche à merveille. Ouf !... Sinon, dans le meilleur des cas, *Monitor* vous aidera à vous tirer d'affaire. Mais si une grosse erreur a été commise, vous aurez tout à recommencer, en rechargeant successivement les éditeur, assembleur, chargeur et moniteur.

Même dans le cas où une erreur apparaît lors de l'assemblage, tout est à refaire. Et c'est bien là que le bât blesse. La mise au point d'un programme, si vous êtes débutant, vous prendra beaucoup, beaucoup de temps... Chargement, lancement, modification, destruction (du fichier précédent), sauvegarde, chargement, essai, etc. On n'en finit plus. Et si en plus vous avez eu la mauvaise idée de ranger votre fichier-source sur une autre disquette que la disquette *Assembler*, que de manipulations et de fausses manœuvres en perspective !

Ce défaut, la lourdeur d'emploi, est dû au fait que les programmes résident sur disquette, tous séparés, et doivent être appelés successivement. Leur taille interdit peut-être de les inscrire en mémoire morte dans une cartouche à enficher à l'arrière du Commodore pour les rendre disponibles immédiatement et sans effort. C'est en tout cas le prix à payer pour utiliser cet assembleur performant.

Pour les initiés de préférence

Pour conclure, *Assembler* est un logiciel intéressant, aux possibilités étendues, réservé de préférence aux programmeurs avertis. Son emploi n'est pas vraiment pratique et impose trop souvent des manipulations fastidieuses. La documentation est suffisante — pour les mordus — mais trop peu pédagogique. Les nouveaux venus au langage Assembleur devront donc se documenter sérieusement avant de se lancer à corps perdu dans l'utilisation de ce logiciel. Et pour le prix de 225 FF ttc, le jeu en vaut certainement la chandelle.

Robin BOIS

Figure 3
Le programme de la figure 2 assemblé

```
LINE# LOC    CODE          LINE

00001  0000                ; TEST ASSEMBLER 64 POUR LIST
00002  0000                FOND=$D021;ADRESSE FOND ECRAN
00003  0000                CORE=$D020;ADRESSE CADRE ECRAN
00004  0000                ;
00006  0000                *=$8000                    ;IMPLANTATION PROGRAMME =32768
00007  8000                ;
00008  8000  20 44 E5      JSR     $E544              ;ROUTINE CLR/HOME
00009  8003  A9 01         LDA     #$01
00010  8005  8D 21 D0      STA     FOND               ;FOND BLANC
00011  8008  A9 00         LDA     #$00
00012  800A  8D 20 D0      STA     CORE               ;CADRE NOIR
00013  800D                ;
00014  800D  A0 B0         LDY     #176               ;176 FOIS
00015  800F  A2 04         BOUCL   LDX #$04           ;5 LETTRES
00016  8011  BD 24 80      PRINT   LDA MESSG,X
00017  8014  20 D2 FF      JSR     $FFD2              ;ROUTINE PRINT
00018  8017  CA            DEX
00019  8018  10 F7         BPL     PRINT
00020  801A  98            TYA                        ;CHANGE COULEURS TEXTE
00021  801B  29 0F         AND     #%00001111
00022  801D  8D 86 02      STA     646                ;#646=ADR.COUL.TEXTE
00023  8020  88            DEY
00024  8021  D0 EC         BNE     BOUCL              ;ET CAETERA
00025  8023  60            RTS
00026  8024                ;
00027  8024  2A 54         MESSG   .BYTE '*TSIL'
00027  8026  2A 54 53
00028  8029                .END


ERRORS = 00000


SYMBOL TABLE

SYMBOL VALUE
 BOUCL    800F    CORE    D020    FOND    D021    MESSG    8024
 PRINT    8011

END OF ASSEMBLY
```

ON NE LE DIRA JAMAIS ASSEZ VIVENT LES PARAMÈTRES !

Les constantes, Marius, c'est comme les allumettes :
ça ne sert qu'une fois.
(Attribué, peut-être à tort, à Marcel Pagnol)

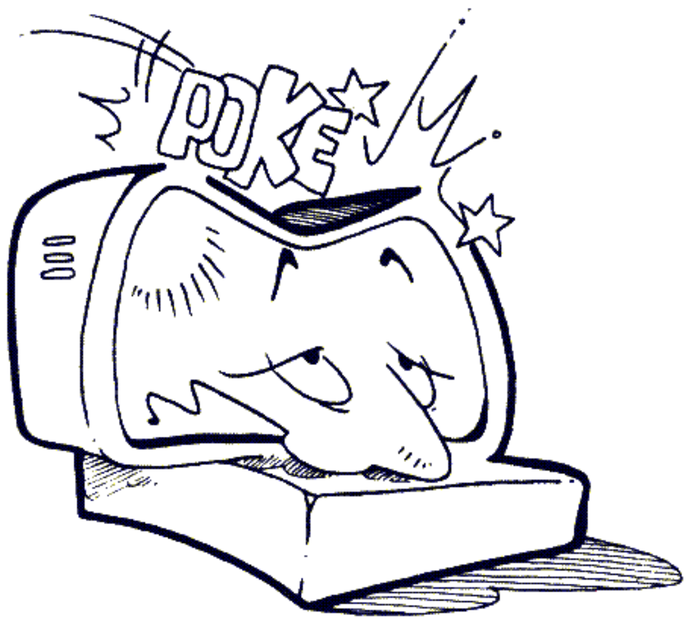
UN nouveau chapitre consacré aux paramètres où l'auteur, économe de la sueur de ses phalanges, vous invite encore à préférer les variables aux constantes.

Dans le premier numéro de *LIST*, j'ai entonné la louange des variables sur plusieurs tons, ce qui explique en partie la météo correspondante, tout en criant haro sur les constantes. Mais la chanson n'est pas finie, car les douces et souriantes variables rendent service dans bien des domaines, et je le prouve.

Pour ce qui est des graphismes, il est parfois intéressant de procéder à grands coups de POKE dans la mémoire d'écran. Sur PET/CBM, on peut imaginer le programme suivant :

```
100 FOR I=1 TO 38
110 POKE 32768+I,64:POKE 33728
    +I,64
120 NEXT I
130 FOR I=32808 TO 33688 STEP
    40
140 POKE I,93:POKE I+39,93
150 NEXT I
160 POKE 32768,112:POKE
    32807,110
170 POKE 33728,109:POKE
    33737,125
```

Résultat prévisible, on trace un cadre sur l'écran, et on n'a pas touché au curseur, qui est toujours là où on l'a laissé. Quelques explications si vous voulez : 32768, c'est le début de la mémoire d'écran. Sachant que l'écran est de 25 lignes sur 40 colonnes, la première boucle trace un trait horizontal (code-écran du caractère : 64) de la ligne 1, colonne 2 à la ligne 1, colonne 39 (POKE 32768+I,64) et de la ligne 25, colonne 2 à la ligne 25, colonne 39. La deuxième boucle trace un trait vertical (code-écran : 93) de la ligne 2, colonne 1 à la ligne 24, colonne 1 et de la ligne 2, colonne 40 à la ligne 24, colonne 40. Il ne manque plus que les quatre coins (l.1 col.1, l.1 col.40, l.25 col.1 et l.25 col.40 ; l. *est mis ici pour ligne et* col. *pour colonne).* C'est ce que font les lignes 160 et 170, avec les codes-écran correspondant aux quatre caractères de coins possibles.

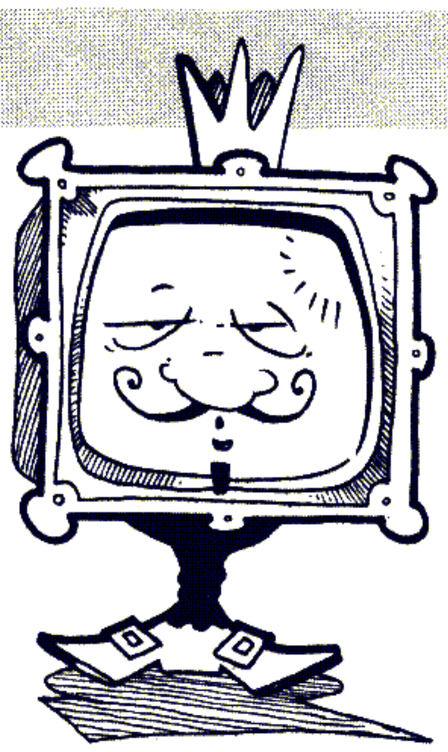


« ...A grands coups de POKE dans la mémoire d'écran. »

Mon programme tourne, on ne s'en lasse pas, et je veux en faire profiter Georges. Or Georges, lui, a un VIC. Il peut lire mes programmes sur cassette sans difficulté, mais il a 23 lignes de 22 colonnes et, pour ce qui est du départ de la mémoire-écran, ce n'est sûrement pas la même adresse que sur les PET/CBM. Il va falloir aviser.

Quant à Ginette, elle s'est offert un Commodore 64 que lui a rapporté son beau-frère qui connaît quelqu'un dont

« ...Les douces et souriantes variables. »



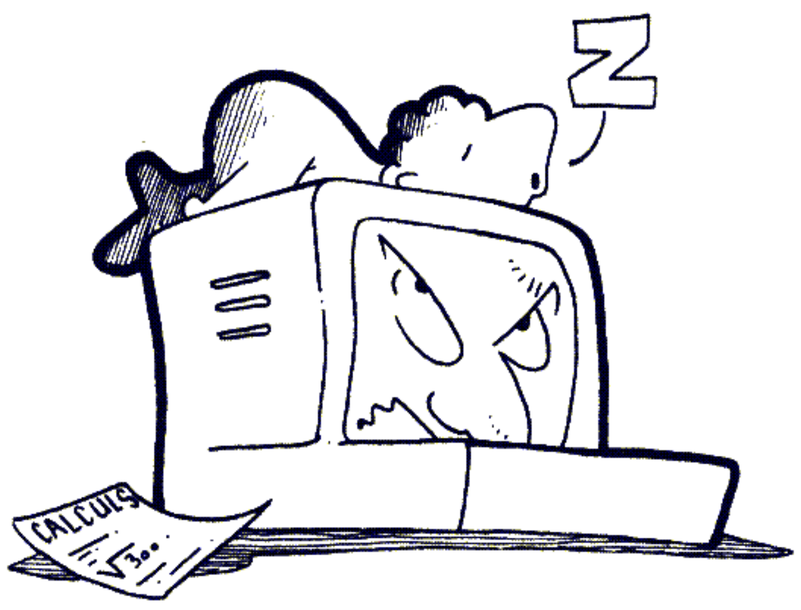
« ...On trace un cadre à l'écran. »

la tante a un ami qui travaille en Angleterre. Eh bien, pour elle, à tous les coups, les adresses seront différentes. Or donc, si je veux donner mon programme à Georges ou à Ginette, ils ne s'en serviront pas, pour la simple raison qu'ils auront la flemme de refaire tous les calculs.

Mais je me suis laissé dire qu'un ordinateur, accessoirement, ça savait aussi effectuer quelques menues opérations arithmétiques. C'est là que la ruse paramétrée va intervenir, et vous allez voir qu'elle a du pain sur la planche, la ruse paramétrée. Alors, au charbon, l'ordinateur ! Je lui offre les données et il fait le travail.

Les données pour mon bon vieux PET sont :
DE, départ de la mémoire-écran (DE = 32768) ; LE, largeur de l'écran (40 colonnes) et NL, nombre de lignes (ici, 25). On y va !

```
100 DE = 32768:LE = 40:NL = 25
110 FOR I = 1 TO LE - 2
120 POKE DE + I,64:POKE DE + LE
    *(NL - 1) + I,64
130 NEXT I
140 FOR I = LE TO LE*(NL - 2)
    STEP LE
150 POKE DE + I,93:POKE DE + I +
    LE - 1,93
160 NEXT I
170 POKE DE,112:POKE DE + LE - 1,
    110
180 POKE DE + LE *(NL - 1),109:
    POKE DE + LE*NL - 1,125
```



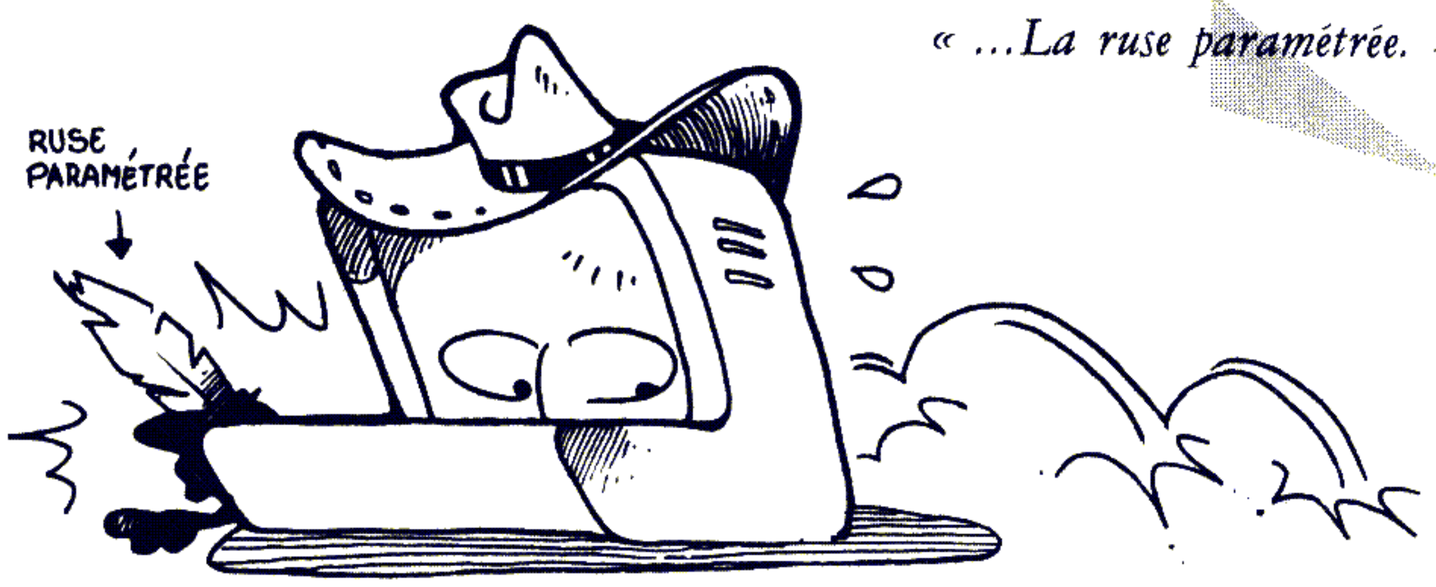
« ...Ils auront la flemme de refaire tous les calculs. »

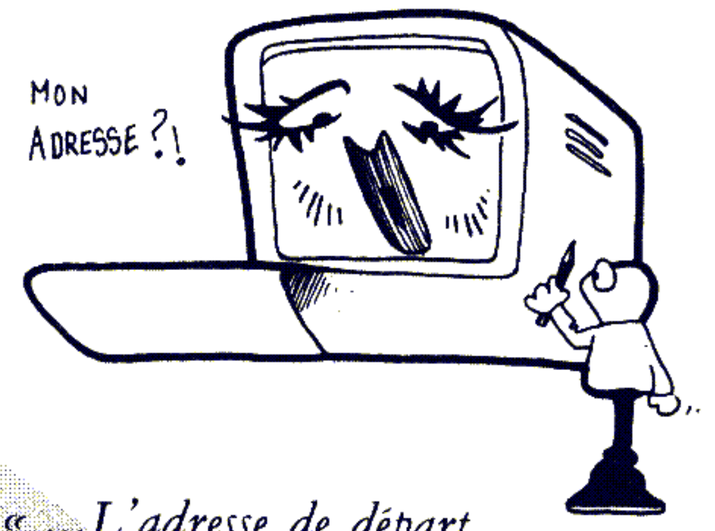
Un petit tour d'adresse

Mon programme à moi comporte trois paramètres : l'adresse de départ de la mémoire d'écran, la largeur d'écran et le nombre de lignes.

Comme moi, Ginette a 25 lignes de 40 caractères, mais Georges, sur son VIC, a 23 lignes de 22 caractères. Pour l'adresse de départ de la mémoire d'écran, chez Ginette, c'est 1024 et tout est dit. Mais chez Georges, c'est une autre paire de manches ! Sur le VIC, à la question "Quelle est l'adresse de départ de la mémoire d'écran ?", la réponse exacte est : "Ça dépend." Ça dépend si Georges laisse son VIC tout seul, avec ses 3,5 Koctets de mémoire vive,

« ...La ruse paramétrée. »





« ...L'adresse de départ de la mémoire d'écran ? »

ou s'il le leste de 3 Ko de mieux, ou s'il le gonfle jusqu'à 8, voire 16 Ko.

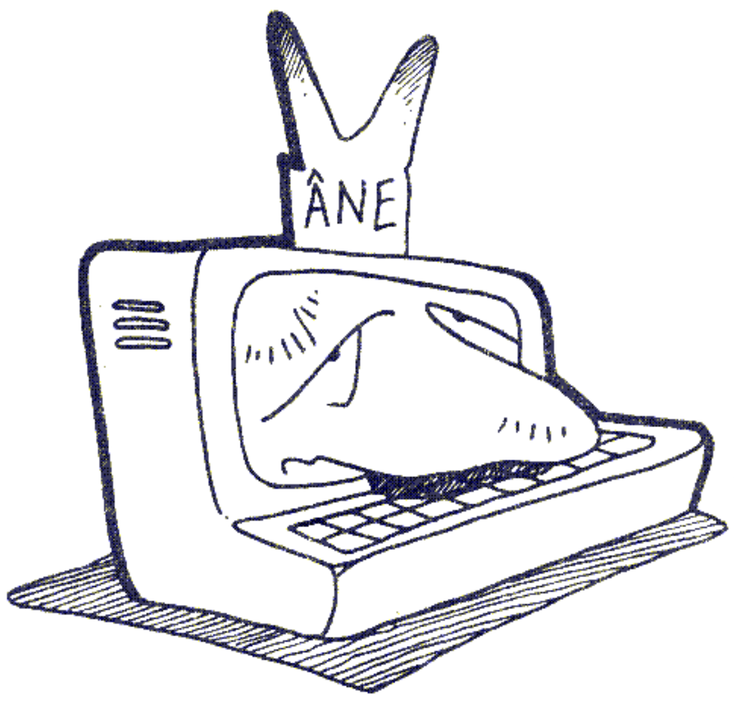
Rien n'est simple, mon pauvre Georges, mais pour simplifier, réduisons le problème à deux cas :
1. George a branché une extension 8 ou 16 Ko. Dans ce K — oh pardon ! — dans ce cas, la mémoire d'écran commence en 4096.
2. Georges n'a rien, ou seulement 3 Ko d'extension, et la mémoire d'écran commence en 7680.

Je pourrais ajouter, histoire d'affoler Georges, que sans le savoir, il a aussi le Basic baladeur, mais c'est une autre histoire, et je garde pour une autre fois le plaisir de lui annoncer la nouvelle et de le voir rouler des yeux en CHR$(209). Car ce n'est pas tout. Georges et Ginette ont la couleur. Ils l'ont voulue, ils l'ont eue, c'est bien fait pour eux : ils en subissent les conséquences. Ils doivent, en plus, mettre par POKE une valeur donnée dans une mémoire-couleur qui contient autant de cases que la mémoire d'écran, faute de quoi, le cadre apparaîtrait en blanc sur blanc, ou bleu sur bleu, bref, n'apparaîtrait pas ! Cette mémoire commence en 55296 sur le C.64, et comme Georges cherche décidément la complication, sa mémoire couleur se balade aussi selon l'extension de mémoire vive (MEV) branchée sur son VIC : s'il a 8 ou 16 Ko de MEV supplémentaire, c'est 37888 ; sinon, c'est 38400. Il sera simple d'ajouter quelle couleur ils choisissent (0 pour noir, 1 pour blanc, 2 pour rouge, etc.) et de doubler les POKEs en mémoire d'écran de POKEs en mémoire-couleur, encore faut-il préciser où elle démarre.

Tout cela paraît bien décourageant, et on pourrait désespérer de jamais s'y retrouver si une pensée ne venait nous consoler : l'ordinateur s'y retrouve, lui, hein ? Et vous et moi qui sommes futés, nous n'allons pas

LES PARAMÈTRES

« ...NOUS N'ALLONS PAS NOUS LAISSER IMPRESSIONNER PAR UNE BÉCANE AMORPHE. »



nous laisser impressionner par une bécane amorphe, un tigre de silicium, que dis-je, un sac à puces ! Alors, approchez-vous, je vais vous dire : l'ordinateur a un truc pour s'y retrouver lui-même. Mais oui, il truque ! Tout est truqué : l'ordinateur carotte ! Il a de petites notes écrites dans les coins, qu'il consulte à la dérobée en faisant semblant de chercher un buvard, et son truc, ça s'appelle les *pointeurs*.

Un pointeur, c'est souvent une paire d'adresses qui se suivent en mémoire, disons A1 et A2, et qui contiennent en tout et pour tout une autre adresse, que nous appellerons AA. La formule rituelle pour lire un pointeur est : AA=PEEK(A1)+256*PEEK(A2). Pour peu que l'on soit certain que A1 contienne 0, ce qui est souvent le cas, il suffit de faire : AA=256*PEEK(A2).

Quant au pointeur de la mémoire-couleur, selon la position du point sur l'écran, il peut être augmenté de 1 sur VIC, ou de 3 sur le C.64. Pour être certain de lire la valeur du début de la mémoire-couleur, le plus sûr est d'inscrire : AA=256*2*INT(PEEK(AA)/2).

Or, le VIC et le C.64, malgré toutes leurs différences, possèdent en commun un certain nombre de pointeurs. Et parmi eux, devinez quoi ? Un pointeur qui contient l'adresse de départ de la mémoire d'écran (en 648) et un autre qui contient l'adresse de départ de la mémoire-couleur (en 244).

Alors, les choses s'éclairent soudain grâce aux deux formules magiques que je vous livre : départ de la mémoire d'écran en 256*PEEK(648) et départ de la mémoire-couleur en 256*4*INT(PEEK(244)/4).

PEEK(648) peut contenir 30 ou 16 sur le VIC, il contient toujours 4 sur le C.64 (essayez un peu de diviser par 256 les adresses que je vous donnais tout à l'heure, et vous comprendrez pourquoi).

Tiens, mais au fait, c'est magnifique ça. Le programme sera capable de savoir tout seul s'il tourne sur VIC ou sur C.64 et il pourra s'y adapter au-to-ma-ti-que-ment :

```
100 DE=256*PEEK(648):LE=40:
    NL=25:IF PEEK(648) >
    4 THEN LE=22:NL=23
105 MC=256*4*INT(PEEK(244)/
    4):C=0
```

Il ne reste plus qu'à doubler le programme en ajoutant les POKEs en mémoire-couleur. S'il fallait refaire tous les calculs, ce serait décourageant. Là, on va mettre le curseur sur le dernier chiffre des lignes 120, 150, 170 et 180, et changer le 0 en 5, puis DE en MC, puis le code du caractère en C, code-couleur. Cela donne :

```
110 FOR I=1 TO LE
120 POKE DE+I,64:POKE DE+LE
    *(NL-1)+I,64
125 POKE MC+I,C:POKE MC+LE
    *(NL-1)+I,C
130 NEXT I
140 FOR I=LE TO LE*(NL-2)
    STEP LE
150 POKE DE+I,93:POKE DE+I+
    LE-1,93
155 POKE MC+I,C :POKE MC+
    I+LE-1,C
160 NEXT I
170 POKE DE,112:POKE DE+
    LE-1, 110
175 POKE MC,C:POKE MC+
    LE-1,C
180 POKE DE+LE*(NL-1),109:
    POKE DE+LE*NL-1,125
185 POKE MC+LE*(NL-1),C:
    POKE MC+LE*NL-1,C
```

Un plaisir ! Mais c'est vrai qu'on a trop souvent tendance à faire le travail à la place de sa machine. C'est nocif, ça... Reste le troisième couplet M'ssieurs-Dames, couplet sentimental : les souvenirs d'enfance.

François J. BAYARD

Adresses de début des mémoires Basic, écran et couleur, sur Commodore 64 et sur VIC en fonction de l'extension de mémoire branchée

	PEEK (44)	Début de mémoire Basic	PEEK (648)	Début de mémoire-écran	PEEK (244)	Début de mémoire-couleur
VIC 3,5 Ko	16 $10	256*16=4096 $1000	30 $1E	256*30=7680 $1E00	150 ou 151 $96 ou $97	256*2*INT((150 ou 151)/2)=38400 $9600
VIC + 3 Ko	4 $04	256*4=1024 $0400	30 $1E	256*30=7680 $1E00	150 ou 151 $96 ou $97	256*2*INT((150 ou 151)/2)=38400 $9600
VIC + 8 Ko ou 16 Ko	18 $12	256*18=4608 $1200	16 $10	256*16=4096 $1000	148 ou 149 $94 ou $95	256*2*INT((148 ou 149)/2)=37888 $9400
C.64	8 $08	256*8=2048 $0800	4 $04	256*4=1024 $0400	216 à 219 $D8 à $DB	256*4*INT((216 à 219)/4)=55296 $O800

Formules indépendantes de l'appareil (VIC ou C.64) et de la taille-mémoire
Début de la mémoire Basic : 256*PEEK(44)
Début de la mémoire-écran : 256*PEEK(648)
Début de la mémoire-couleur : 256*4*INT(PEEK(244)/4)

LES PILES EN FORTH

DEUX piles se partagent les tâches en Forth : une pile de données et une pile de retour. Elles sont à l'origine de la rapidité et d'une occupation mémoire réduite de ce langage.

Tous les ordinateurs qui disposent d'un Forth travaillent avec un minimum de deux piles. Ce sont des zones de la mémoire que le processeur va utiliser pour y accumuler les données, en vue de leur utilisation ultérieure.

La première est la *pile de données,* encore appelée *pile de paramètres* ou plus simplement *la pile* (on l'a déjà rencontrée dans le numéro 2 de *LIST*). La seconde est la *pile de retour,* interlocuteur discret mais habituel et nécessaire de la pile de données.

Une seule pile beaucoup d'astuces

Prenons un exemple. Soixante assiettes — des jaunes, des vertes et des blanches — sont empilées dans le désordre. On voudrait les regrouper par couleur dans une seule pile. Une solution permet un tel tri : décomposer la pile en trois petites piles, chacune accueillant une couleur, puis les réunir de telle sorte que les assiettes soient regroupées par couleur.

Dans cet exemple, le tri aura été effectué à l'aide de trois piles de retour. Forth, lui, n'en donne qu'une, mais avec beaucoup d'astuces pour en tirer le maximum.

Dans certains Forth particulièrement complets, il peut exister d'autres piles : la pile image de Hector HRX permet des acrobaties graphiques, les piles de chaîne (sur PDP 11, VAX, etc.) sont utilisées par les astronomes professionnels, etc.

Observons concrètement ce qui se passe. L'entrée de XX RETURN renvoie le message ?N'EXISTE PAS OK, ou tout autre message d'erreur, car le mot "XX" n'existe pas en tant que mot Forth. Le résultat de cette introduction est de vider la pile ! Entrons maintenant : 1 2 3 4 5 6 7 8 9 10. Et étudions alors les opérateurs de pile.

Tout d'abord, un coup de "sondeur" donné par .S RETURN affichera le contenu de la pile sans le modifier : 10 9 8 7 6 5 4 3 2 1 OK précédé de ADR. EN PREMIER et suivi d'une flèche (→).

Le rôle de chaque opérateur

ADR. EN PREMIER est l'adresse où entrera la prochaine valeur empilée. Elle reste constante tant qu'il y a dix valeurs sur la pile. Elle diminue de deux points dès que l'on y ajoute des données.

Chaque donnée entrée est mémorisée sur 16 bits. En Forth, ces seize bits constituent une *cellule.* Donc cette adresse est celle de la prochaine cellule libre, et comme la pile part de valeurs élevées dans la mémoire et descend vers des valeurs basses, il est normal qu'à chaque entrée, l'adresse de la nouvelle cellule libre ait diminué de deux points.

Pour se familiariser avec cette notion, il ne reste qu'à entrer, sonder, réentrer et resonder. Pour faire l'inverse, c'est-à-dire rechercher la valeur contenue à une adresse, il suffit d'utiliser le point d'interrogation (?), dont la définition en Forth est la suivante :

```
: ?@ . ;
```

Pour en revenir aux opérateurs, il faut à nouveau vider la pile par XX RETURN et introduire les dix valeurs (sans oublier les espaces) : 1 2 3 4 5 6 7 8 9 10.

Chaque opérateur doit être défini clairement et pourra être testé au fur et à mesure.

DROP (n--) : c'est l'opérateur le plus radical ! Il fait sauter, il efface, il supprime la donnée qui figurait en sommet de pile. C'est ce que matérialise la notation conventionnelle (n--) qui donne l'état du sommet de pile avant DROP, le "n" désignant une valeur quelconque, les deux tirets (--) représentant l'entrée du mot considéré (ici DROP) et l'espace qui précède la seconde parenthèse marquant que le "n" a disparu. Ainsi, quand on n'a plus besoin d'une valeur sur la pile, on la DROPpe !

SWAP (a,b--b,a) : cet opérateur intervertit les deux valeurs qui se succèdent en sommet de pile. Si on l'entre à la console en le faisant suivre de .S, on obtient : → xxxxx 9 10 8 7 6 5 4 3 2 1 OK (les xxxxx représentent l'adresse qui varie selon les ordinateurs).

DUP (n--n,n) : c'est un opérateur de duplication du sommet de pile. Il sera très utile lors des tests, les opérateurs de test "consommant" le sommet de pile. Opérons un DUP et sondons par .S : → xxxxx 9 9 10 8 7 6 5 4 3 2 1 OK.

ROT (a,b,c--b,c,a) : provoque une permutation circulaire des trois valeurs du sommet de pile, ce dernier étant toujours à droite dans la notation conventionnelle. Cet opérateur est assez astucieux. Testons ROT et voyons le résultat (par .S) : → xxxxx 10 9 9 8 7 6 5 4 3 2 1 OK.

OVER (a,b--a,b,a) : OVER copie le sous-sommet en sommet de pile, selon le même principe que DUP, mais en s'exerçant sur la deuxième valeur de la pile au lieu de la première. Un test de OVER donne : → xxxxx 9 10 9 9 8 7 6 5 4 3 2 1 OK. C'est parfait.

Mais que faire pour remettre la pile dans l'ordre primitif sans réintroduire les données ? Les opérateurs dont on dispose nous le permettent très facilement : DROP SWAP DROP RETURN. On vérifie par .S : → xxxxx 10 9 8 7 6 5 4 3 2 1 OK.

Et pour opérer plus profondément dans la pile ? Par exemple, pour faire remonter le 5 en sommet de pile par permutation circulaire ? Ce sera possible avec une nouvelle classe d'opérateurs : ceux qui demandent une donnée sur la pile.

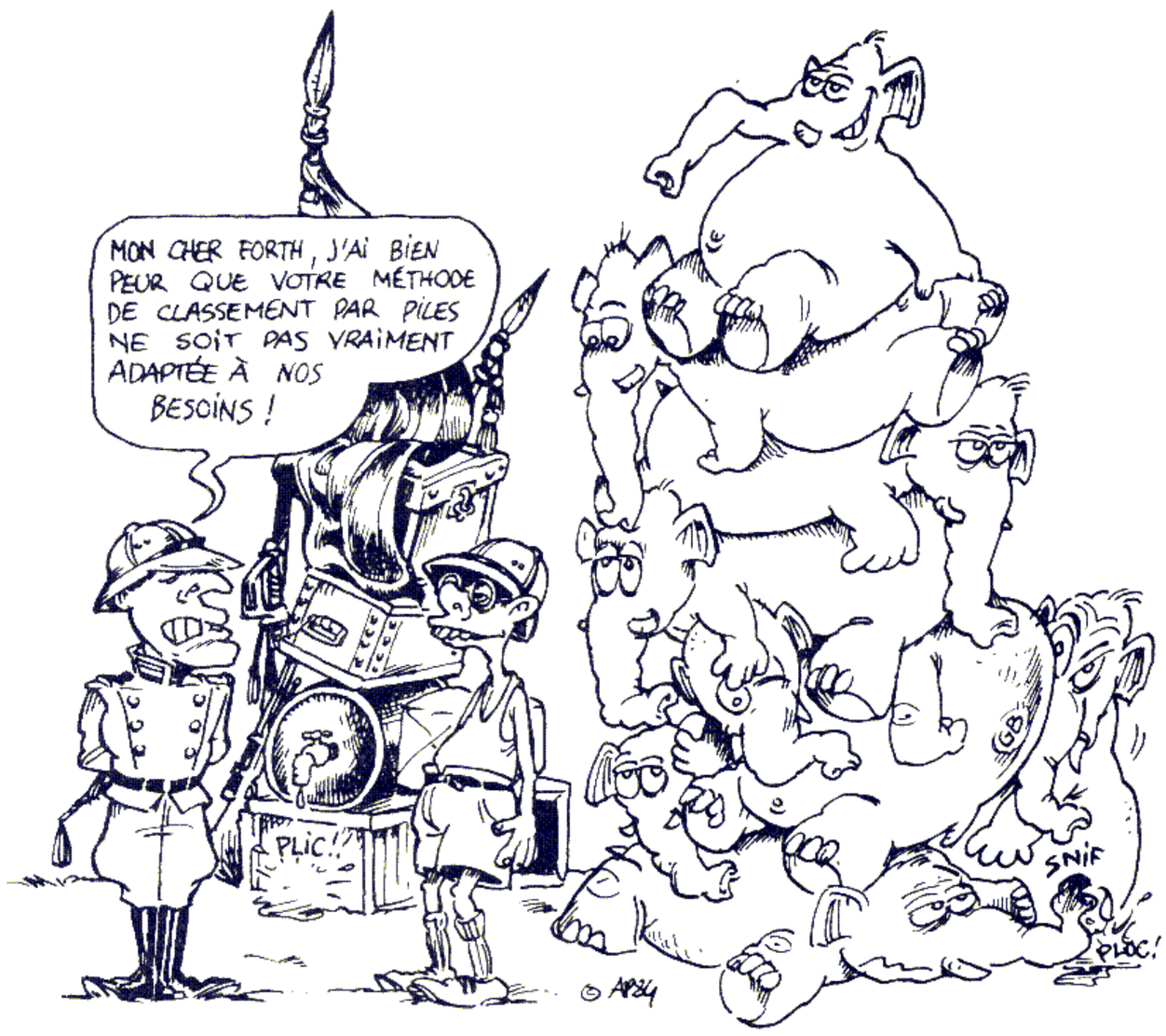
n ROLL (n--) : la notation conventionnelle indique par un espace avant la seconde parenthèse, que ROLL consomme le sommet de la pile pour exécuter sa tâche. Il pratique la permutation circulaire du nième terme de la pile, avant introduction de n. Certains Forth ne disposent pas de ROLL. Ils pourront le créer de la manière suivante :

```
: XROLL DUP 2 <
      IF DROP ." <2 ROLL IMPOSSIBLE" CR
      ELSE DUP 0
           DO SWAP R> R>
             ROT >R >R >R
           LOOP 1
           DO R> R> R> R> SWAP >R
             ROT ROT >R >R
           LOOP R>
      THEN ;
```

Le système de décalage de ce programme donne une grande clarté au texte et en facilite la compréhension. Les têtes de ligne sont occupées par les mots de structuration et de bouclage, avec une correspondance verticale des mots qui travaillent ensemble (DO avec LOOP, IF avec ELSE et THEN). Quelques explications sont encore nécessaires avant que ce programme ne soit parfaitement clair.

n PICK (n--) : est le dernier opérateur de cette série. Son action est analogue à celle de OVER, mais ici c'est le nième élément de la pile que PICK va recopier au sommet. Ce mot n'existe pas dans tous les Forth. Il peut être créé lui aussi, le langage Forth étant conçu de telle sorte qu'un mot qui n'appartient pas à son vocabulaire peut presque toujours être construit. Même s'il faut chercher un peu.

La liste des opérateurs doit être complétée par celle des opérateurs arithmétiques : + (a,b--S), S représentant la somme et − (a,b--D), D représentant la différence. Il faut noter que cette différence se fait dans le sens a − b, c'est-à-dire que c'est la dernière valeur empilée qui est soustraite à la précédente. L'opérateur de la multiplication est * (a,b--P) où P représente le produit.



Quant à la division, elle n'est pas simple : c'est une combinaison d'additions, de multiplications et de soustractions.

Tous ces opérateurs permettent déjà de créer de nouvelles fonctions. Par exemple :

- un calculateur automatique de carrés
  : CARRE DUP * . ;
- un "cubeur" infaillible
  : CUBE DUP DUP * * . ;

Rien n'empêche de concevoir ces fonctions différemment. En effet, les points (.) provoquent l'affichage du résultat, mais aussi sa perte, en le sortant de la pile. Une nouvelle définition de CARRE, sans le point d'affichage, permet au résultat d'être utilisé par la suite. Ainsi, une nouvelle définition de CARRE et de CUBE peut être (après avoir fait FORGET CARRE pour nettoyer le dictionnaire) :

```
: CARRE DUP * ;
: CUBE DUP CARRE * ;
```

Ainsi CUBE utilise la fonction CARRE.

En intermède aux jeux de pile, il est bon de se pencher sur la structure de prise de décision la plus fréquemment employée. Une décision, c'est un choix consécutif à un examen de la situation. Par exemple :
Peuvent se produire A et B
Si c'est A, je ferai PLIC
Si c'est B, je ferai PLOC
Et après cela, dans tous les cas, je ferai GATOUM !

Après une prise de décision aussi importante, dont beaucoup d'événements dépendront à la face du monde, on peut s'étendre sur une couche moel-

Les mots de base

CLS	nettoie l'écran
CR	envoie le curseur au début de la ligne suivante par un retour-chariot (Carriage Return)
FORGET	oublie le mot qui suit FORGET dans le flot d'entrée, ainsi que tous les mots créés après lui qui figurent dans le dictionnaire
VLIST	liste tous les mots du dictionnaire en commençant par les derniers créés. La limite entre les mots créés (mots de l'utilisateur) et les mots existants déjà (mots du système ou primitives) est marquée par le mot FORTH. Les primitives ne sont pas effacées par FORGET
[:]	introduit la création d'un mot nouveau dont le nom suit ce signe
[;]	termine la définition du mot créé. Elle est alors compilée dans le dictionnaire
[."]	signale à la machine le début d'une chaîne de caractères qui se terminera par ["]. L'exécution du mot affichera alors la chaîne placée entre guillemets
[(]	commence un commentaire qui se terminera par [)] (ne nécessitant pas d'espace devant lui). Comme [."], il doit être suivi d'un espace
DO...LOOP	exécute une boucle
.S	affiche le contenu de la pile sans le modifier.

Un mot FORTH est toujours terminé par un espace qui est son délimiteur.

leuse et profiter du repos du guerrier !

En Forth, ces décisions sujettes à condition se présentent avec la structure : IF...ELSE...THEN (si c'est vrai...si c'est faux...et puis).

Si le résultat du test est 1, soit vrai (pour IF), le traitement ira à PLIC, puis à GATOUM (pour THEN). Si le résultat est à 0, soit faux (pour THEN), il ira à PLOC avant d'aller à GATOUM (pour THEN).

Toute valeur non nulle peut remplacer le 1, mais le 0 pour faux est impératif.

```
: PLIC CR ." INDIVISIBLE PAR 3 !" CR ;          (traitement si vrai)
: PLOC CR ." DIVISIBLE PAR 3 !" CR ;            (traitement si faux)
: TEST 3 MOD ;                                  (MOD divise et donne
                                                seulement le reste éven-
                                                tuel)

: GATOUM ." ES-TU CONTENT, MAÎTRE ?" CR ;
: PAR3 TEST IF PLIC                             (s'il y a un reste, VRAI
                                                devant IF envoie à
                                                PLIC)
        ELSE PLOC                               (s'il n'y a pas de reste,
                                                FAUX devant IF envoie
                                                à PLOC)
        THEN GATOUM QUIT ;                      (QUIT supprime le OK
                                                mais pas le curseur)
```

Es-tu content, Maître ?

Il existe toute une série de tests. Ils ont tous la caractéristique de renvoyer sur la pile un drapeau ou flag, ou encore booléen, c'est-à-dire soit un 0 soit un 1 ayant toujours le même sens : 1 pour vrai, 0 pour faux.

Ceci à l'intention de IF qui consommera le booléen pour orienter le processeur vers ce qui le suit dans le cas du 1, ou pour pousser le processeur vers ce qui suit ELSE si le booléen était nul.

Vous avez compris ?
Oui, vous passez à la suite.
Non, vous relisez !

Essayons ce qui suit :

Il ne reste plus qu'à entrer, par exemple, 4324 PAR3 RETURN, pour voir s'afficher :
INDIVISIBLE PAR 3 !
ES-TU CONTENT, MAÎTRE ?

Mais en entrant 4326 PAR3 RETURN, le résultat devient :
DIVISIBLE PAR 3 !
ES-TU CONTENT, MAÎTRE ?

On peut éprouver une certaine fierté à se voir gratifier ainsi par l'ordinateur, mais aussi la grande satisfaction d'avoir bien compris que MOD ne place un reste sur la pile que si la division par 3 n'aboutit pas à un quotient entier. Et si IF trouve le reste, il affiche INDIVISIBLE, alors que s'il trouve un 0, il renvoie à ELSE qui affiche DIVISIBLE.

C'est donc le reste qui remplit le rôle de booléen. Mais ce ne sera pas toujours le cas et une batterie de tests sera, là encore, bien utile.

0<(n--B) : le booléen (B) qui remplace n en sommet de pile est à 1 si n est négatif, et à 0 sinon.

0>(n--B) : le booléen est à 1 si n est positif, et à 0 si n est négatif ou nul.

0 =(n--B) : le booléen est à 1 si n est nul, et à 0 sinon.

=(a,b--B) : le booléen est à 1 si a=b, à 0 sinon.

>(a,b--B) : le booléen est à 1 si a>b, à 0 sinon.

<(a,b--B) : le booléen est à 1 si a<b, à 0 sinon.

Il faut remarquer que 0 > signifie "plus grand que 0" à l'inverse de la notation traditionnelle : ceci est dû à la notation polonaise inverse.

Tous ces outils permettent de passer à la pile de retour.

Les opérateurs de cette pile ne doivent pas être confondus avec les tests, bien qu'ils utilisent le même type de notation. Le signe > a ici valeur de flèche. Ces opérateurs doivent être maniés avec une grande prudence, et toujours par paire de valeurs inversées. De plus, ils ne peuvent être utilisés qu'entre le signe deux-points (:) et le signe point-virgule (;).

Cette dernière remarque évitera de voir apparaître un tas de manifestations bizarres qui conduisent à un RESET, tragique puisqu'il vide la mémoire vive de tout ce qui y avait été introduit !

>R (n--) : envoie le sommet de pile au sommet de pile de retour.

R >(--n) : ramène le sommet de pile de retour en sommet de pile.

R (--n) : recopie en sommet de pile le sommet de pile de retour, en sauvegardant ce dernier.

I (--n) : joue exactement le même rôle que R, son utilisation se situant exclusivement dans les boucles DO... LOOP.

J (--n) : recopie en sommet de pile le sous-sommet de pile de retour, avec la même limite aux boucles DO...LOOP que I.

La pile de retour ayant été détaillée, chacun peut démonter le programme XROLL présenté plus haut. C'est, on peut l'apprécier maintenant, un bon exemple du rôle discret mais indispensable que joue la pile de retour...

Bernard LAMBEY

Début du démontage

Introduction de la pile témoin 1 2 3 4 5 6 7 8 9 10 RETURN
OK

	Pile	Retour	Commentaires
:	—	—	Début compilation
4	--- 8 9 10 4	—	4 au sommet. C'est "n"
XROLL	--- 8 9 10 4	—	NOM du nouveau mot
DUP	--- 8 9 10 4 4	—	Duplique le sommet
2	-- 8 9 10 4 4 2	—	Introduction du deuxième terme en vue de la comparaison
<	- 7 8 9 10 4 0	—	booléen 0 car 4<2 est faux

Conventionnellement, le sommet de pile est à droite. Ici c'est l'inverse, mais la flèche à l'écran rappelle où est le sommet.

UN PROGRAMME ANTI-PIRATES

LES ordinateurs de poche ont vocation, du fait de leur taille et de la permanence de leur mémoire, d'accompagner leurs propriétaires en des lieux très divers, de la salle de classe au bureau. Or, certains programmes peuvent être confidentiels et jusqu'ici aucune protection n'apportait une sécurité vraiment sérieuse.

Un très court programme, rédigé en langage-machine, empêche de lister ou d'exécuter un programme principal en Basic que l'on désire cacher. Le principe est simple : comme s'il s'agissait d'un message secret, on va le coder.

L'utilitaire de codage est court (33 octets), concis et très rapide. Il fait appel à deux données que le propriétaire de l'ordinateur est seul à connaître : l'adresse d'implantation de la routine en langage-machine (relogeable, elle peut être dissimulée n'importe où en mémoire) et, bien sûr, la clé du codage.

Ce codage conduit à ajouter la valeur 1 à tous les octets du (ou des) programme(s) Basic.Chaque fois que la routine de codage est appelée, la valeur 1 est de nouveau ajoutée aux codes du Basic. L'effet est immédiat : les programmes sont devenus illisibles et inexécutables. Vous pouvez ainsi les coder et les recoder un nombre de fois connu de vous seul.

La clé du codage-décodage

Pour décoder, la même routine est employée mais vous devez en indiquer la clé. Si cette clé correspond, tous les octets sont alors décrémentés d'une unité. Il convient de décoder autant de fois qu'on aura codé pour récupérer le programme Basic.

Le programme reproduit figure 1 réalise la programmation de la routine de codage-décodage. Lancé par RUN ou

Figure 1
Programme d'introduction et de contrôle de la routine en langage-machine

```
10:REM CHARGEUR
20:DATA &A5, &78, &
   65, &18, &A5, &78
   , &66, &1A, &94, &
   A7, &78, &67, &89
   , &06, &14, &A7, &
   78, &68
30:DATA &8B, &0C, &
   15, &4E, CLE, &89
   , &03, &DF, &8E, &
   01, &DD, &51, &9E
   , &18, &9A
40:RESTORE :INPUT
   "CLE";CLE, "ADR
   ";ADR:FOR I=0
   TO 32:READ X:
   POKE ADR+I, X:
   NEXT I
50:REM CONTROLE
60:A=0:FOR I=ADR
   TO ADR+32:A=A+
   PEEK I:NEXT I:
   IF A<>3309+CLE
   PRINT "ERREUR
   CODES":END
70:PRINT "PRGM =
   OK":END
```

Figure 2
Programme de codage-décodage

Étiquettes	Mnémoniques	Commentaires

ADR :	LD A, (&7865)
	LD YH, A
	LD A, (&7866)
	LD YL, A
		Range dans Y
		l'adresse de
		début des programmes
		Basic.

CONT :	LD A, YH
	CP A, (&7867)
	JR NZ +06
	LD A, YL
	CP A, (&7868)
	JR Z +&0C
		Compare avec l'adresse de fin.
		Teste YH.
		Va en TRA si ≠.
		
		Teste YL.
		Va en FIN si =.

TRA :	LD A, (Y)
	CP XL, clé
	JR NZ +03
	DEC A
	JR +01
	INC A
	
	LDI (Y), A
	
	JR −&18
		A = Code Basic.
		Compare XL avec la clé.
		Va à INC A si ≠.
		Ote 1 au code (décodage).
		Va à LDI (Y), A.
		Ajoute 1 au code (codage).
		
		Remet le code en Basic et ajoute 1
		à l'adresse Y.
		Va en CONT.

FIN :	RTN	Fin, rend la main au Basic.

Note : après l'appel par CALL ADR, Z le registre X du microprocesseur contient la valeur de Z.

par GOTO 10, il pose deux questions : CLE et ADR. Il s'agit de la clé du codage-décodage (ne l'oubliez pas) et de l'adresse en mémoire du premier octet où la routine sera programmée. Introduisez, par exemple, 60 pour la CLE (tout nombre compris entre 0 et 255 convient) et &78C0 pour ADR (adresse de la variable A$).

Aucune marge d'erreur possible

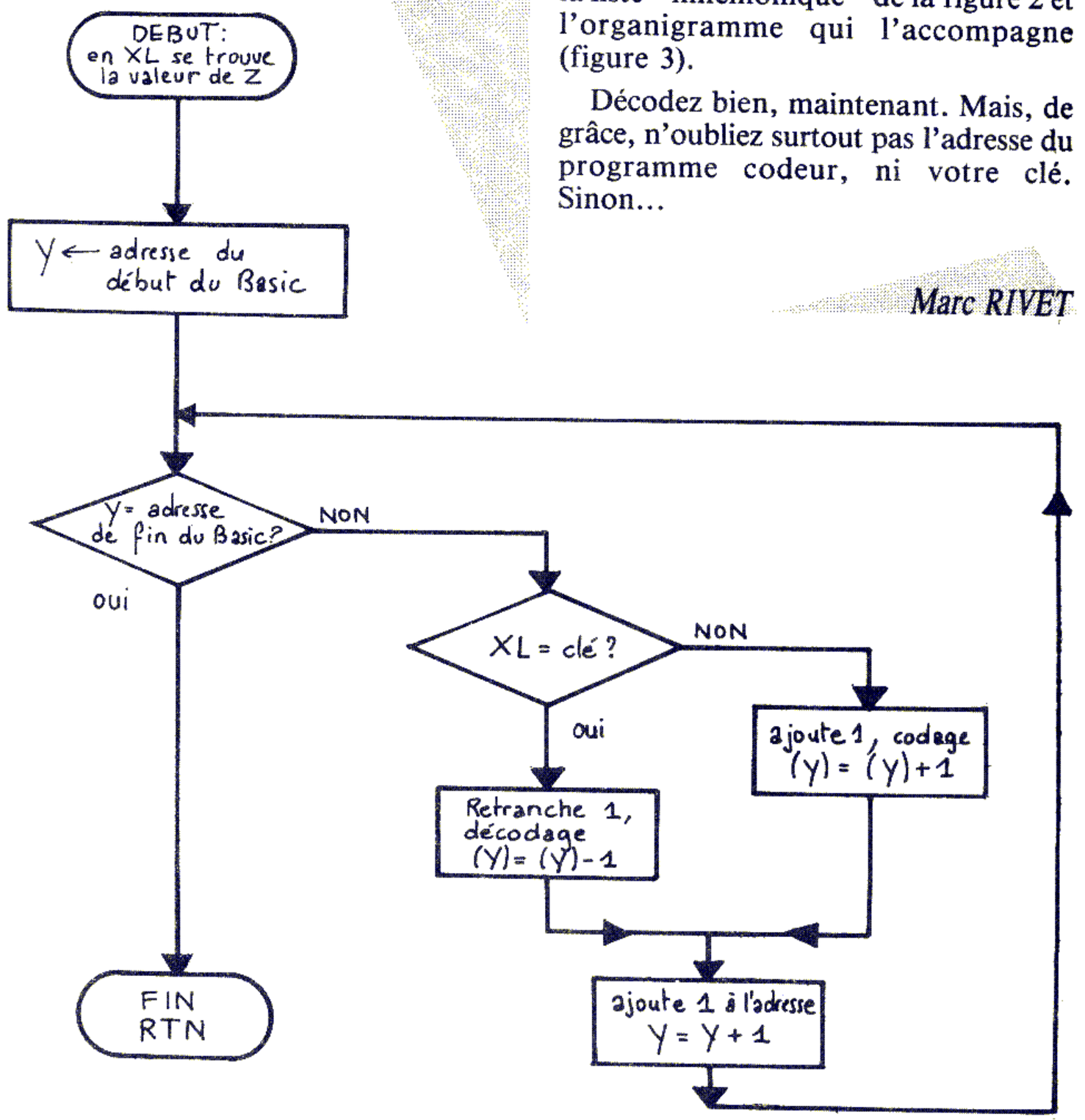
Alors le chargement s'effectue. Si tout se passe bien, c'est-à-dire si vous n'avez commis aucune erreur de frappe, en particulier dans les codes des lignes 20 et 30, le message PRGM = OK doit s'afficher. En revanche, si un code est mal tapé (&A4 au lieu de &A5, par exemple) alors le message ERREUR CODES apparaît. Il est impératif de ne faire aucune erreur de programmation. S'il y en a, revoyez les lignes de DATA.

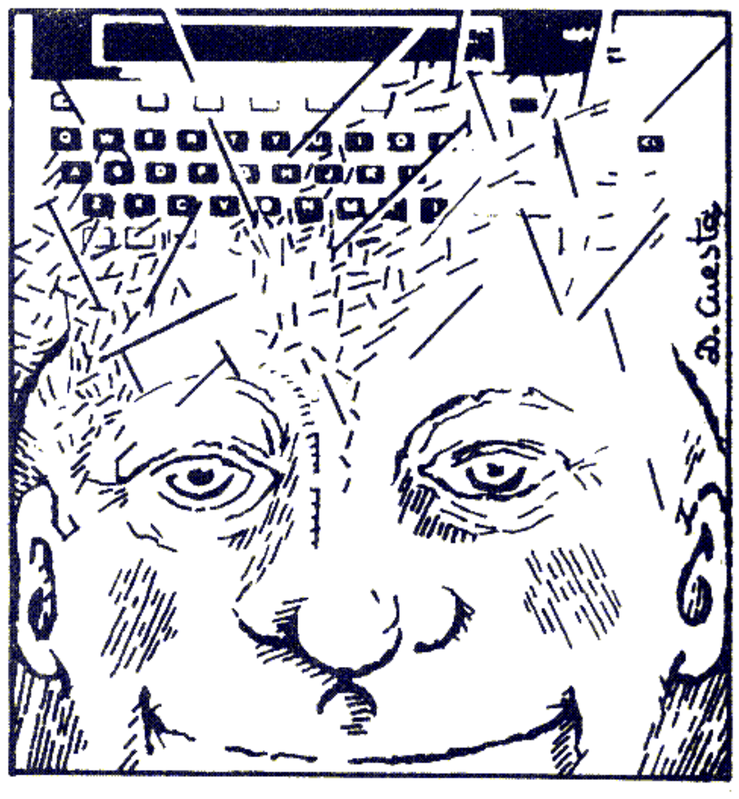
Vous aurez noté, à l'examen de la ligne 30 que la CLE est elle-même programmée dans la routine. Vous pourrez donc la modifier ultérieurement en tapant : POKE ADR + 22, *nouvelle clé*.

Pour coder les lignes de Basic, dès lors il suffit de taper CALL ADR. Dans notre exemple, CALL &78C0. Vous pouvez consulter ce qui reste du Basic !

Essayez donc de l'utiliser... Pour

Figure 3
Organigramme





décoder, tapez CALL ADR, CLE. Dans cet exemple, faire Z = 60 puis CALL &78C0,Z pour récupérer le Basic reconstitué. Si vous aviez codé 10 fois le programme (10 fois CALL &78C0), il aurait fallu le décoder autant de fois (Z = 60 puis 10 fois CALL &78C0,Z).

Et le pirate indélicat n'a même pas la possibilité de rechercher la clé en essayant tous les nombres de 0 à 255 car chaque essai infructueux code un peu plus le programme.

Les amateurs de langage-machine trouveront un grand intérêt à éplucher la liste "mnémonique" de la figure 2 et l'organigramme qui l'accompagne (figure 3).

Décodez bien, maintenant. Mais, de grâce, n'oubliez surtout pas l'adresse du programme codeur, ni votre clé. Sinon...

Marc RIVET

TGV, UN TRI A GRANDE VITESSE

LE problème des tris est un des thèmes poignants de la littérature informatique. Il n'est pas des plus simples. La méthode de Shell-Metzner est une formule facile à coder, et efficace pour les petites ou moyennes listes. La même idée de base, mais bien plus raffinée, se retrouve dans le tri Quicksort de Hoare. Il s'agit de déplacer plus rapidement et en moins d'étapes les éléments d'une liste qui sont les plus éloignés de leur position finale.

Si le principe du tri *Quicksort* est facile à saisir, sa réalisation pose bon nombre d'embûches. Surtout dans un langage comme le Basic. Celui-ci semble avoir été expressément conçu pour nous empêcher d'utiliser des formules récursives, c'est-à-dire qui se font appel à elles-mêmes afin de simplifier un problème jusqu'à ce que la solution devienne évidente (ce qui s'applique admirablement au raisonnement tenu par l'auteur du présent algorithme).

Des listes plus courtes

Hoare s'est dit que si on pouvait prendre une liste, choisir un élément au milieu, et déplacer à sa gauche toutes les valeurs qui sont plus petites et à sa droite toutes celles qui sont plus grandes, on aurait deux listes plus courtes à trier, ce qui est beaucoup plus aisé. Poursuivant le même raisonnement, on continue à réduire chaque liste obtenue par la même méthode, et ainsi de suite, jusqu'à n'avoir que des listes d'un ou deux éléments, pour lesquelles la solution est simpliste.

Comme chacune de ces listes ne peut contenir, en raison du principe de base, que des éléments plus petits que ceux de sa voisine de droite, le problème est par le fait même résolu. Ce genre d'algorithme se prête d'ailleurs à des traitements bien plus élégants dans des langages qui admettent la récursion, comme Pascal ou Logo.

Il se pose cependant quelques petites difficultés : que se passe-t-il si la liste est irrégulière, c'est-à-dire si on n'a pas un nombre égal d'éléments à droite et à gauche de l'élément central ? Et comment, avec cette multitude de petites listes, fait-on pour savoir où on en est rendu ?

Simulation d'une pile

La réponse au premier problème est de rendre mobile la partition centrale qui divise la liste. Si on a trop de valeurs à gauche, on la déplace vers la droite, et vice versa... mais en prenant bien garde de ne pas laisser glisser dans l'autre bloc des valeurs qui dépassent la limite ! Ce qui n'est pas évident, il s'en faut.

Pour garder en mémoire le point où

Tri de Hoare
Programme en pseudo-langage
Auteur Yves Leclerc
Copyright *LIST* et l'auteur

```
Pour une liste ELEMENT (T) de T éléments :
Mettre T sur la pile
Initialiser à 0 la cloison C
Répéter jusqu'à ce que la pile soit vide :
    Mettre la limite basse B = C
    Mettre la limite haute H = valeur sur la pile
    Si l'espace ainsi défini a moins de 3 valeurs :
      S'il y a 2 valeurs, les mettre en ordre
      Mettre C = sommet de la pile + 1
      Enlever le sommet de la pile
    Sinon :
      Choisir un élément central de comparaison M
      Répéter jusqu'à ce que B et H se rencontrent :
        Avancer B jusqu'à ce qu'elle rencontre H ou que
          ELEMENT (B) soit plus grand que ELEMENT (M)
        Reculer H jusqu'à ce qu'elle rencontre B ou que
          ELEMENT (H) soit plus petit que ELEMENT (M)
        Echanger ELEMENT (B) avec ELEMENT (H) s'ils
          sont différents
      Si B est égale ou supérieure à M, la reculer
      Si H est différente de M, échanger ELEMENT (B)
        avec ELEMENT (M)
    Placer B sur la pile
Fin.
```

on en est, en l'absence d'un mécanisme de récursion (qui fait le gros de ce travail pour nous), on crée (ou on simule, en Basic) une « pile opérationnelle » dans laquelle on entrepose des pointeurs vers les limites des parties de la liste sur lesquelles on travaille. Une fois chaque étape terminée, on peut reprendre ces pointeurs sur la pile et s'en servir pour définir les nouvelles limites.

Voici deux versions de l'algorithme de Hoare, l'une en « pseudo-langage » qui permet de mieux en comprendre le mécanisme, l'autre en Basic. Le pseudo-code se comprend tout seul, sans besoin d'autres explications que celles données par la liste (voir le programme *Tri de Hoare* en pseudo-langage).

Une bonne vitesse

Le programme en Basic est dans un Basic des plus standards à une exception près : la fonction SWAP du Basic de TSC (Goupil), sur lequel ce code a été testé, permet d'échanger directement les contenus de deux variables. Si vous ne l'avez pas, vous remplacez par la formule habituelle, soit U = E(x) : E(x) = E(y) : E(y) = U.

Le programme effectuera un tri à une bonne vitesse (de deux à cinq fois plus vite que le tri de Shell-Metzner en moyenne, pour des listes assez longues, de 200 éléments et plus), mais il y a place pour beaucoup d'optimisation : partout où il y avait doute, la vitesse a été sacrifiée à la clarté. Rien ne vous empêche de faire le contraire.

Yves LECLERC

Tri de Hoare
Programme en Basic
Auteur Yves Leclerc
Copyright *LIST* et l'auteur

```
900 REM SOUS-PROGRAMME DE TRI QUICKSORT
910 REM
920 REM TRI D'UNE LISTE E() DE T ELEMENTS
930 REM NUMEROTES DE 0 A T-1
940 REM P() EST UNE PILE OPERATIONNELLE
950 REM SIMULEE, DONT LE NIVEAU EST N.
960 REM C EST LA CLOISON, B ET H LES LIMITES
970 REM HAUTE  ET BASSE,  ET M LA VALEUR MOYENNE.
980 REM SWAP EST UNE FONCTION D'ECHANGE DE 2 VALEURS.
990 REM
1000 N=0:P(N)=T:C=0
1010  B=C-1:H=P(N)
1020  IF H<C+3 GOTO 1130
1030    M=INT((B+H)/2)
1040      B=B+1:IF B=H GOTO 1090
1050      IF E(B)<=E(M) GOTO 1040
1060      H=H-1:IF B=H GOTO 1090
1070      IF E(H)>=E(M) GOTO 1060
1080      SWAP E(B),E(H):GOTO 1040
1090    IF B>=M THEN B=B-1
1100    IF H<>M THEN SWAP E(B),E(M)
1110    N=N+1:P(N)=B
1120  GOTO 1160
1130    IF H=C+2 AND E(C)>E(C+1) THEN SWAP E(C),E(C+1)
1140    C=P(N)+1
1150    N=N-1
1160 IF N>=0 GOTO 1010
1170 RETURN
```

LA FONCTION MODULO N'EST PLUS CE QU'ELLE ÉTAIT

LA HP-41 C possède une fonction arithmétique souvent méconnue car on ne la trouve que rarement employée en informatique (et vice-versa) : la fonction modulo, dite MOD, qui calcule le reste de la division entière d'un nombre Y par un nombre X.

En fait, ce calcul précis de reste d'une division peut s'avérer fort utile là où on ne s'y attend pas ; l'essentiel étant d'y songer ! L'opération modulo (Y ENTER X MOD s'écrit aussi y modulo x) est très simple à concevoir mais elle est une porte ouverte sur un monde merveilleux.

On teste la divisibilité d'un nombre Y par un autre nombre X en examinant le résultat de l'opération MOD. S'il est entier, Y est divisible par X, sinon il ne l'est pas (évidemment). Avec MOD cela s'écrit : Y ENTER X MOD. Si le résultat est 0, alors Y est divisible par X ; Y modulo X vaut 0.

Dans le même ordre d'idées élémentaires, si l'on désire tester la parité (pair ou impair) d'un nombre Y, on programmera : Y ENTER 2 MOD. Si Y est pair le résultat est nul, sinon il vaut 1. Zéro et un ? Il est des amateurs éclairés dont l'esprit fait « tilt » (bien programmé...) dès qu'on parle de zéro et un. C'est ici avec raison.

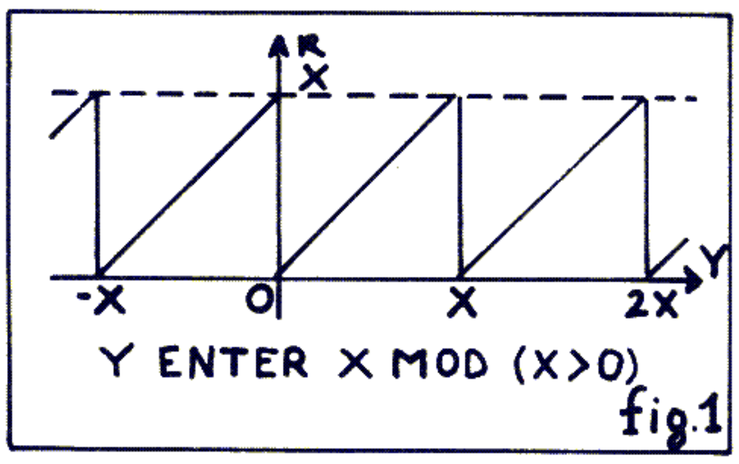
Pour reproduire sur HP-41 C la fonction d'algèbre booléenne XOR (*ou exclusif*) qui, en considérant deux chiffres 0 ou 1, retourne 1 s'ils sont différents et 0 sinon (0 et 0 donne 0, 1 et 1 donne 0, 1 et 0 donne 1, 0 et 1 donne 1) on utilise la séquence de fonctions : + 2 MOD.

Mais la logique *binaire* n'est pas la seule accessible avec MOD. Des niveaux plus complexes peuvent être atteints : 3 MOD introduirait la logique « ternaire »...

Appliquée à d'autres nombres que les entiers naturels, la fonction MOD ouvre des perspectives. Mais il devient nécessaire de disposer d'une définition sans ambiguïté de la fonction MOD.

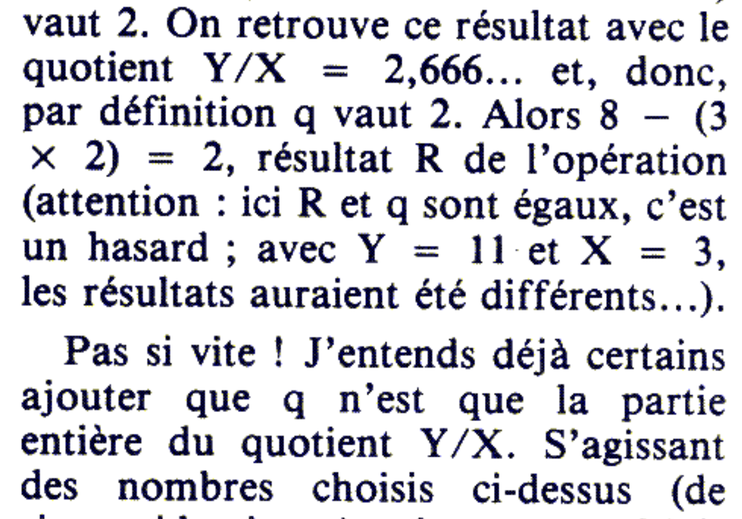
Hé... pas si vite !

Soient Y et X deux nombres, le résultat R de la fonction MOD (Y modulo X) correspond à : R = Y − Xq, où q se définit comme *le plus grand nombre entier inférieur ou égal au quotient Y/X.*

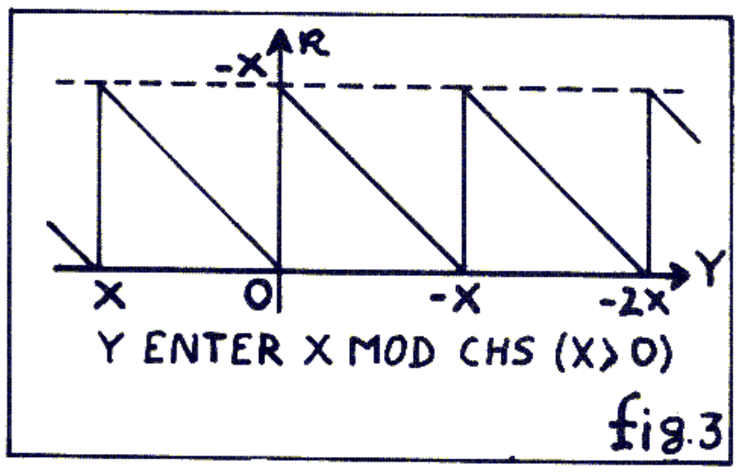


Ainsi, pour Y = 8 et X = 3, Y modulo X (soit Y ENTER X MOD) vaut 2. On retrouve ce résultat avec le quotient Y/X = 2,666... et, donc, par définition q vaut 2. Alors 8 − (3 × 2) = 2, résultat R de l'opération (attention : ici R et q sont égaux, c'est un hasard ; avec Y = 11 et X = 3, les résultats auraient été différents...).

Pas si vite ! J'entends déjà certains ajouter que q n'est que la partie entière du quotient Y/X. S'agissant des nombres choisis ci-dessus (de signes identiques), c'est exact. Mais avec −8 modulo 3 et 8 modulo −3 ? 8 CHS ENTER 3 MOD retourne bien 1. Si −8/3 donne −2,66... la valeur de q est −3 selon notre définition, *le plus grand nombre entier inférieur ou égal à Y/X.* Ce n'est pas exactement la même chose que −2,66... ENTER FRC −.



Graphiquement, l'opération modulo, R = Y modulo X, correspond à une courbe en dents de scie (figure 1) pour X positif. Les figures suivantes (figures 2 et 3) représentent cette fonction, respectivement, pour X négatif et X positif (mais au résultat inversé par CHS).



Ces courbes montrent le chemin à l'électronicien qui souhaiterait simuler, par exemple, le fonctionnement d'un circuit électrique : signal en dents de scie de période X.

Les graphes sont continus, donc valables aussi pour des valeurs Y et X non entières.

Avec Y non entier et X toujours égal à 1 le résultat sera la différence entre Y et le *plus grand nombre entier inférieur ou égal* à Y :
2,768 ENTER 1 MOD donne 0,768
2,768 CHS ENTER 1 MOD donne 0,232.

Et c'est ainsi qu'on arrive à la partie entière d'un nombre, vraie au sens mathématique de la définition : *le plus grand nombre entier inférieur ou égal.* Et cette définition n'est pas celle de la classique fonction INT, voyez les nombres négatifs. La séquence sera : RCLX 1 MOD −.

Guère éloignée, la suite : RCLX 1 CHS MOD − retournera *le plus petit nombre entier supérieur ou égal.*

Et si l'on ne voulait conserver d'un nombre positif que ses décimales à partir de la nième ? C'est n CHS 10^x MOD qu'il convient de programmer. Inversement, rajouter RCL X en tête et − en fin, tronquera le nombre à sa nième décimale.

A titre d'illustration d'emplois, parfois peu orthodoxes, de MOD, voici quelques problèmes, résolus, dignes de figurer dans la rubrique « Misez p'tit optimisez ! ».

Éliminer le premier chiffre d'un nombre quelconque. Par exemple, si X = 3527,32 on veut obtenir 527,32. Solution en 10 octets seulement qui emploie deux fois la fonction MOD : RCLX LOG RCLX 1 MOD − 10 ↑ X MOD.

D'autres problèmes

Générer un nombre pseudo-aléatoire compris entre 0 et 5 *mais avec une probabilité double de tirer les numéros 0, 1 et 2* (un dé pipé en quelque sorte). Il suffit d'employer un générateur non truqué de nombres pseudo-aléatoires (voir le livret d'applications de la HP-41 C) qui tire de 0 à 8 (et non pas de 0 à 5). Le résultat est soumis à l'opération 6 MOD ce qui donne le tableau suivant :

Nombre tiré	0	1	2	3	4	5	6	7	8
6 MOD	0	1	2	3	4	5	0	1	2

Les chiffres 0, 1 et 2 apparaissent deux fois dans les possibilités de tirage après 6 MOD et ont donc une probabilité double (soit 2/9) d'être tirés par rapport à 3, 4 ou 5.

Mieux : un dé « super-pipé ». Le chiffre 0 doit avoir une chance de sortir trois fois plus grande que les chiffres 3, 4 ou 5.

Nombre tiré	0	1	2	3	4	5	6	7	8	9
7 MOD	0	1	2	3	4	5	6	0	1	2
6 MOD	0	1	2	3	4	5	0	0	1	2

Le tableau ci-dessus donne le traitement subi par un nombre pseudo-aléatoire compris entre 0 et 9, soit 7 MOD 6 MOD.

La fonction MOD a bien des tours dans son sac. Elle offre souvent une solution élégante dans des situations où l'on songe plutôt à une longue suite d'instructions classiques. Si vous avez mésestimé MOD jusqu'ici, il serait peut-être temps de réviser votre jugement.

Robert PULLUARD

COBOL TOUJOURS JEUNE

LE Cobol n'est pas mort. La disparition de ce très vieux langage était pourtant prévue par certains pour... le début des années soixante-dix ! Il a aujourd'hui vingt-cinq ans et reste le langage le plus utilisé au monde.

C'est dans le cadre d'une réunion très officielle du *Cobol Codasyl* que l'on s'est réjoui cette année de l'excellente santé du client, dont on fêtait le vingt-cinquième anniversaire.

Si le mot « Codasyl » n'évoque rien pour vous — peut-être une sorte de médicament ? — vous êtes pardonné, bien qu'il concerne un passage passionnant de l'histoire de l'informatique. Reprenons le long chemin qui mène de la programmation des monstres Mark I, Eniac et autres SSEC jusqu'à nos jours.

Il fallait un nouveau langage

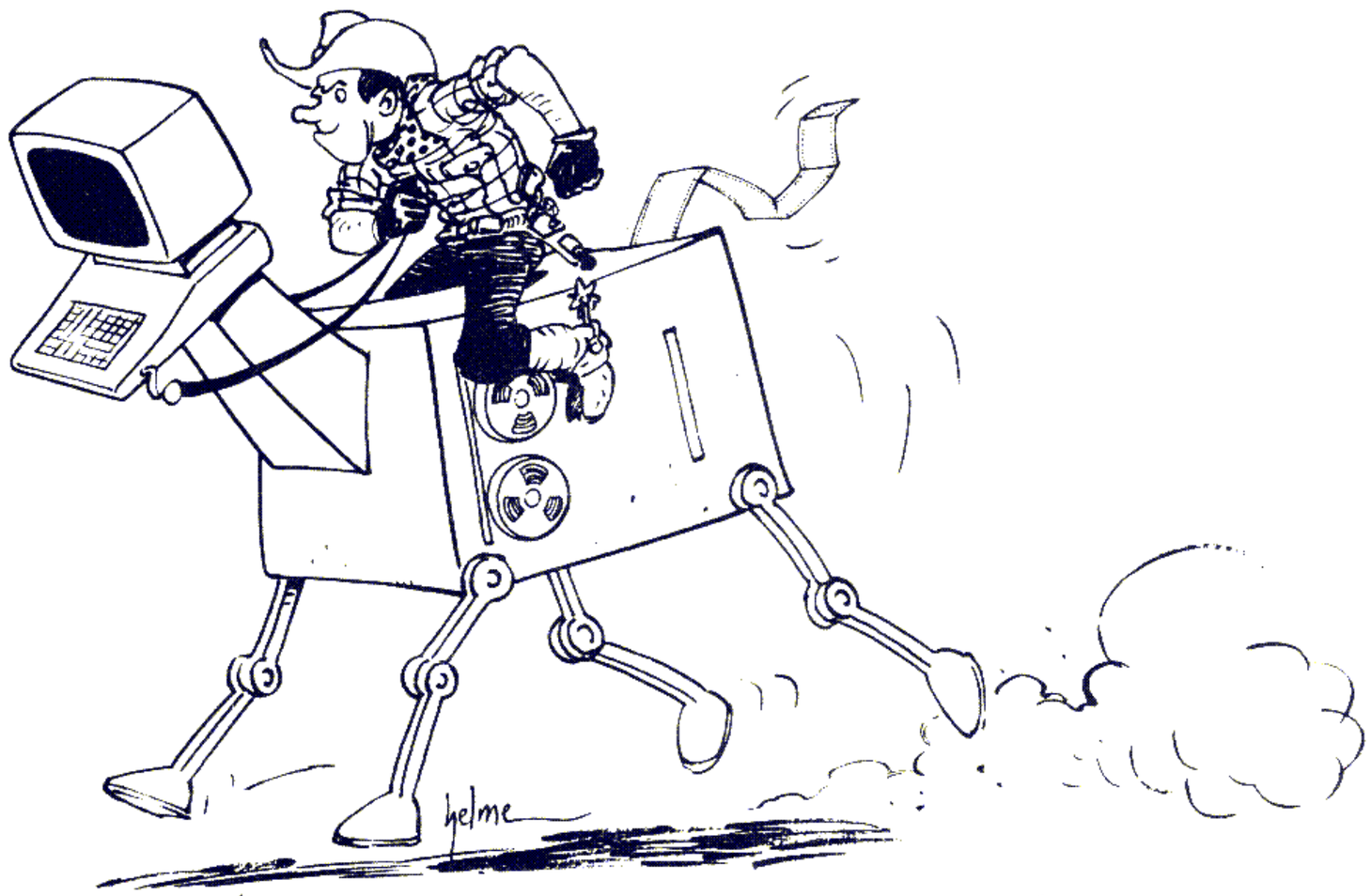
Au début était le langage-machine, vite relayé par l'Assembleur et le très célèbre Fortran de John Warner Backus (10 novembre 1954). Avec cette **invention, les langages évolués étaient nés qui, d'une certaine manière, allaient permettre à des non-informaticiens de travailler sur ordinateur : sautait là un verrou formidable.**

Mais Fortran était marqué par la première vocation de l'informatique : les calculs mathématiques. Très vite, elle se découvrait une autre zone de développement, qui allait devenir gigantesque : la gestion. Les exemples classiques sont bien connus : réservation de billets d'avion, programmes de paye, établissement des feuilles d'impôts — déjà Blaise Pascal avait fabriqué sa machine pour aider son père, haut fonctionnaire du fisc rouennais, à calculer les tailles en Haute Normandie !

Pour toutes ces applications, Fortran était plus que malcommode (il le restera jusqu'en 1977 pour l'essentiel). Il fallait donc innover avec la mise au point d'un autre langage, conformément au stéréotype de l'époque qui voyait informatique scientifique (beaucoup de calculs, peu de données) et informatique de gestion (beaucoup de données, peu de calculs) se partager à jamais deux champs d'activité aux modalités bien disjointes.

Contrairement à une idée reçue, ce n'est pas Cobol qui fut le premier à être conçu dans ce but. Il eut plusieurs prédécesseurs. Ainsi, le Flowmatic fut mis au point en 1958 chez Univac (Remington Rand, devenue aujourd'hui Sperry Rand), compagnie consacrée aux ordinateurs du même nom développés par John William Mauchly (1907-1980) et John Presper Eckert (né en 1919). Ces derniers avaient conçu et construit l'Eniac en 1946 à la Moore School of Electrical Engineering de l'Université de Pennsylvanie. Deux autres langages existaient encore : Aimaco (AIr MAterial COmmand) mis au point pour l'Armée de l'Air, et Comtran (COMmercial TRANslator) conçu par IBM.

Les concepteurs de Flowmatic et d'Aimaco avaient reçu séparément



Cobol partant à la conquête de nouveaux territoires (allégorie)

l'aide d'une même personne qui avait l'art d'être présente à peu près partout où il se passait quelque chose d'intéressant.

L'incroyable Grace M. Hopper

Peut-être certains ont-ils deviné : il s'agit bien de l'incroyable Grace Murray Hopper, née le 9 décembre 1906 à New York, programmeur des Mark I, II et III — elle y calcula notamment une table de sinus —, inventeur du mot « bug » à Harvard en août 1945, théoricienne de la notion de compilateur dès 1951, co-mère de l'Assembleur et de Mathmatic, professeur de mathématiques et d'engineering à la Moore School, et, depuis la mise à la retraite de l'Amiral Hyman Rickover, le plus ancien officier de marine américain en activité... C'est pourtant Cobol qui donnera à Grace Hopper sinon ses galons les plus prestigieux (car ses performances intellectuelles des années quarante et cinquante sont incomparables) du moins sa plus grande notoriété, puisqu'elle fut l'âme du fameux comité *Codasyl* dont nous allons voir le rôle essentiel dans l'histoire des langages.

A la suite du précédent créé par des universitaires européens qui s'étaient déjà réunis à Zurich au printemps 56 pour étudier la possibilité de créer un néo-Fortran (qui deviendra Algol), le Centre de Calcul de l'Université de Pennsylvanie fut à l'origine, le 8 avril 1959, de la décision de définir un langage de gestion universel. Le DOD (Department Of Defense), promoteur d'Aimaco, aida à concrétiser ce vœu par un séminaire tenu au Pentagone les 28 et 29 mai 1959 d'où sortit, le 4 juin, le célèbre *Codasyl* (COmmittee on DAta SYstem anaLysis) réunissant gouvernement des Etats-Unis, constructeurs et utilisateurs. Ces dates (1) ne peuvent que susciter un sentiment d'admiration devant une administration capable de réagir en moins de deux mois pour une tâche qui aurait pu sembler de routine... Efficacité semble bien être un maître mot de la langue américaine !

Le nom du langage vient de l'un des sous-comités de *Codasyl* (il est tiré de COmmon Business Oriented Language). Ce groupe était dit « à court terme » ; il sut en effet aller très vite puisque le premier projet était prêt dès septembre. C'était surtout Flowmatic qui avait été utilisé pour la définition de Cobol. Un autre sous-comité, à plus longue vue, voulait consacrer davantage de temps à établir un cahier des charges plus précis. Le Pentagone prit alors une décision incroyable : il annonça que les commandes gouvernementales ne seraient désormais passées qu'aux constructeurs ayant des compilateurs Cobol. Parallèlement, ce dernier langage était déclaré, en quelque sorte, d'utilité publique et ne pouvait être considéré comme la propriété exclusive d'aucune compagnie (premier pas vers la non-brevetabilité du logiciel).

Cette histoire a été souvent racontée ces dernières années, en particulier parce qu'elle s'est répétée au moins une fois. Le fameux Department Of Defense prit en effet une seconde initiative analogue lorsqu'il mit au concours vers 1976 la conception du langage ADA - challenge gagné en 1979 par un groupe d'ingénieurs français de chez Bull dont Jean Ichbiah.

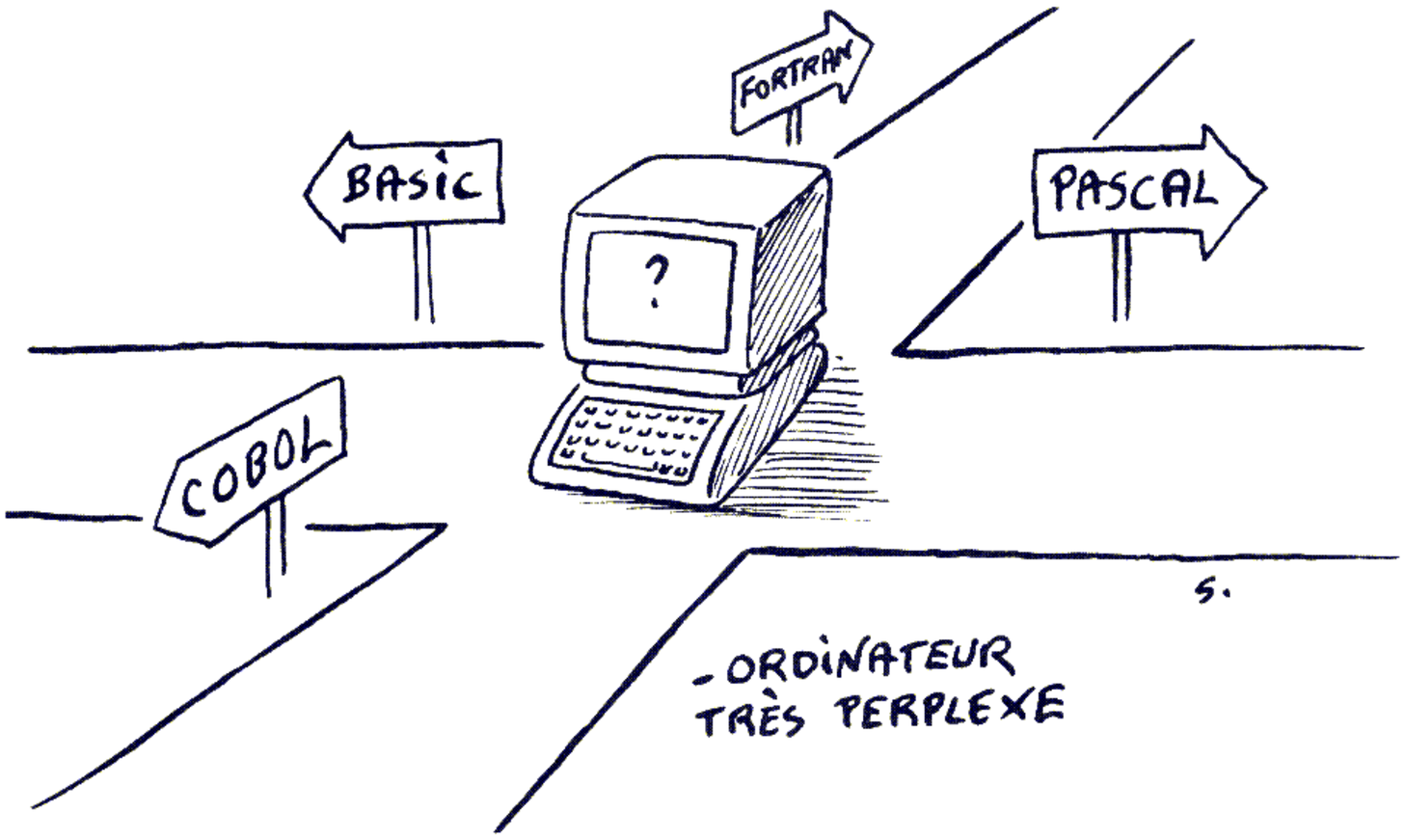
S'unir devant l'adversaire

On comprend bien les raisons vitales que présente, pour un ministère de la Guerre, l'unification des langages employés par tous ceux qui dépendent de lui : en cas de conflit, si la marine, l'air, l'armée de terre et les fournisseurs en logistique « babélisaient » à qui mieux mieux, leurs ordinateurs bafouillants rendraient à l'ennemi bien plus de services que toutes les cinquièmes colonnes du monde ! Voilà qui explique les démarches « dirigistes » du Pentagone dans l'histoire de l'informatique et justifie son coup de force, même si l'on peut regretter effectivement les grandes lourdeurs du langage finalement adopté, peut-être un peu trop rapidement.

Bien entendu, les choses ne sont pas tout à fait aussi simples : la première codification complète sérieuse n'eut lieu qu'entre 1962 et 1963, et plusieurs mises à jour (dont une en 1968) se produisirent jusqu'à la publication de la norme ANSI de 1974 — comme pour Fortran revu trois ans plus tard — pour tenir compte de l'évolution considérable des matériels, y compris des nouvelles possibilités graphiques inconnues au moment de la conception de ces deux ancêtres. Mais, d'après la dernière réunion de *Codasyl* à Londres cet été, 90 % du Cobol originel est encore en fonc-

(1) Elles sont tirées, ainsi qu'un grand nombre d'informations de cet article, d'un livre de René Moreau, Ainsi naquit l'informatique, *paru chez Dunod en 1982.*

COBOL TOUJOURS JEUNE



tion. Par ailleurs, ne serait-ce que parce qu'il est toujours enseigné aux Etats-Unis dans les écoles professionnelles — par lesquelles passent la plupart des programmeurs —, il reste de loin le langage professionnel le plus utilisé en dépit de tous ses défauts.

A l'attention des anglophones

Peut-être faut-il dire maintenant quelques mots sur les particularités de Cobol. Au contraire de Fortran, un habitué d'ordinateur individuel ne se sent pas a priori en terrain connu (quoiqu'on ignore souvent qu'il en existe des versions, parfois excellentes, sur des matériels très courants comme TRS-80 modèle I, II, III ou IV). La raison principale en est que ce langage très verbeux évite le plus possible les symbolisations de type mathématique. Au lieu d'écrire X = Y pour affecter à la variable X la valeur contenue dans Y, Cobol emploie la périphrase MOVE Y TO X (les créateurs craignaient par exemple les réactions des utilisateurs, généralement non scientifiques, devant le « scandaleux » : X = X + 1). Un produit aussi simple que X = Y*Z se traduit par le pénible COMPUTE X = Y*Z voire, surtout dans les versions plus anciennes, par l'horrible MULTIPLY Y BY Z GIVING X.

On a compris le but de ces manœuvres : le Cobol se veut presque transparent, c'est-à-dire lisible par un (anglophone) quasi ignorant de toute formulation mathématique. Il est vrai que la lecture d'une liste où le signe universel < est remplacé par les mots LESS THAN est peut-être plus claire pour qui a gardé mauvais souvenir de ses années de collège... Mais on ne saurait dire que l'on y gagne en place ; Cobol est responsable de la disparition de forêts, transformées en tonnes de papier, qu'il phagocyte abusivement. La clarté se paie par un certain alourdissement qui se retourne sans doute contre la volonté des auteurs.

Le grand intérêt de Cobol réside, à nos yeux exigeants d'aujourd'hui, dans sa grande structuration. Qu'il suffise de dire qu'une liste en Cobol, numérotée comme en Basic, est constituée nécessairement de quatre parties soumises chacune à une syntaxe très rigoureuse (gare à celui qui oublie un point final où il faut, et en met de façon intempestive !). Pour obtenir un texte acceptable, plusieurs compilations sont nécessaires (surtout compte tenu de la longueur des instructions), soit à cause des erreurs d'écriture, soit parce qu'il y a eu faute dans l'analyse des nombreuses ouvertures et fermetures de fichiers séquentiels ou indexés, par les OPEN, CLOSE, READ, WRITE...

Pour survoler Cobol

Le lecteur intéressé pourra par exemple se reporter à l'excellent ouvrage d'initiation « Langages de programmation » de Stéphane Berche et Claude Lhermitte (PSI, 1982) où les huit pages qui sont consacrées au Cobol donnent un bon aperçu de ses modalités. Ou encore à un article assez ancien (mai 1980) du numéro 17 de notre confrère l'Ordinateur Individuel, avec un exemple malheureusement non détaillé, faute de place, de programme financier.

Enfin le livre « Cobol : perfectionnement et pratique » de Michel Koutchouk (Masson, 1981), expose avec grande précision une chaîne de traitement comptable assez complète formée de huit programmes et deux sous-programmes dont la lecture, abondamment commentée, éclaire l'essentiel.

Indépendance avant tout

Ces « divisions » (*identification, environment, data* et *procedure*) sont elles-mêmes décomposées en « sections » dont l'ordre est immuable. Par exemple, l'*environment divison* comporte nécessairement la *configuration section* détaillant l'ordinateur sur lequel a été écrit le programme et celui sur lequel on doit l'utiliser, donc des instructions du type :

SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.

Ce souci correspond, bien entendu, au désir de portabilité du langage, dont nous avons vu qu'il se devait de rester (presque) totalement indépendant du matériel utilisé. Dans des applications de gestion, il est également normal de trouver, pour chaque variable, des précisions sur les formats ; ainsi rencontre-t-on des expressions de la forme

PICTURE 9(6)V9(2)

(ou 999999V99) qui signifient que le réel considéré doit avoir 6 chiffres — éventuellement nuls — avant la virgule, suivis de deux décimales. Le reste est classique : sous-programmes, tests, boucles rendent finalement Cobol assez aisé à apprendre... passé le premier effroi devant sa légèreté pachydermique !

André WARUSFEL

MESURER LE BASIC

POUR évaluer en partie les performances d'un ordinateur donné, et plus précisément les qualités de son Basic, nous avons retenu (provisoirement) dix tests. Ils permettent de mesurer la vitesse avec laquelle la machine exécute ses appels de sous-programmes et diverses instructions de traitement de chaînes de caractères, de calculs arithmétiques, d'opérations sur les tableaux de variables, etc.

La batterie des tests présentée ici n'est évidemment pas parfaite. Elle permet de se faire une première idée de son ordinateur. Mais certaines adaptations peuvent être nécessaires.

Ainsi pour l'Epson QX-10, les tests font apparaître une différence selon la forme de l'instruction NEXT. Avec NEXT suivi du nom de la variable (NEXT I, par exemple), les boucles se révèlent plus longues qu'avec NEXT tout seul : le test 1 passe de 21 à 15 secondes, le test 2 de 56 à 50 secondes, le test 3 de 85 à 78 secondes, etc. De plus, les fonctions scientifiques travaillent sur seize chiffres significatifs, alors que les nombres sont sur 6,5 chiffres significatifs.

Quant au BBC, il appelle certaines adaptations pour les tests 8, 9 et 10. Ainsi, le test 8 doit être modifié comme suit :

```
5   MODE 0
10  FOR I=1 TO 10000
15  MOVE 0,1023:DRAW 1279,0
20  NEXT I
30  END
```

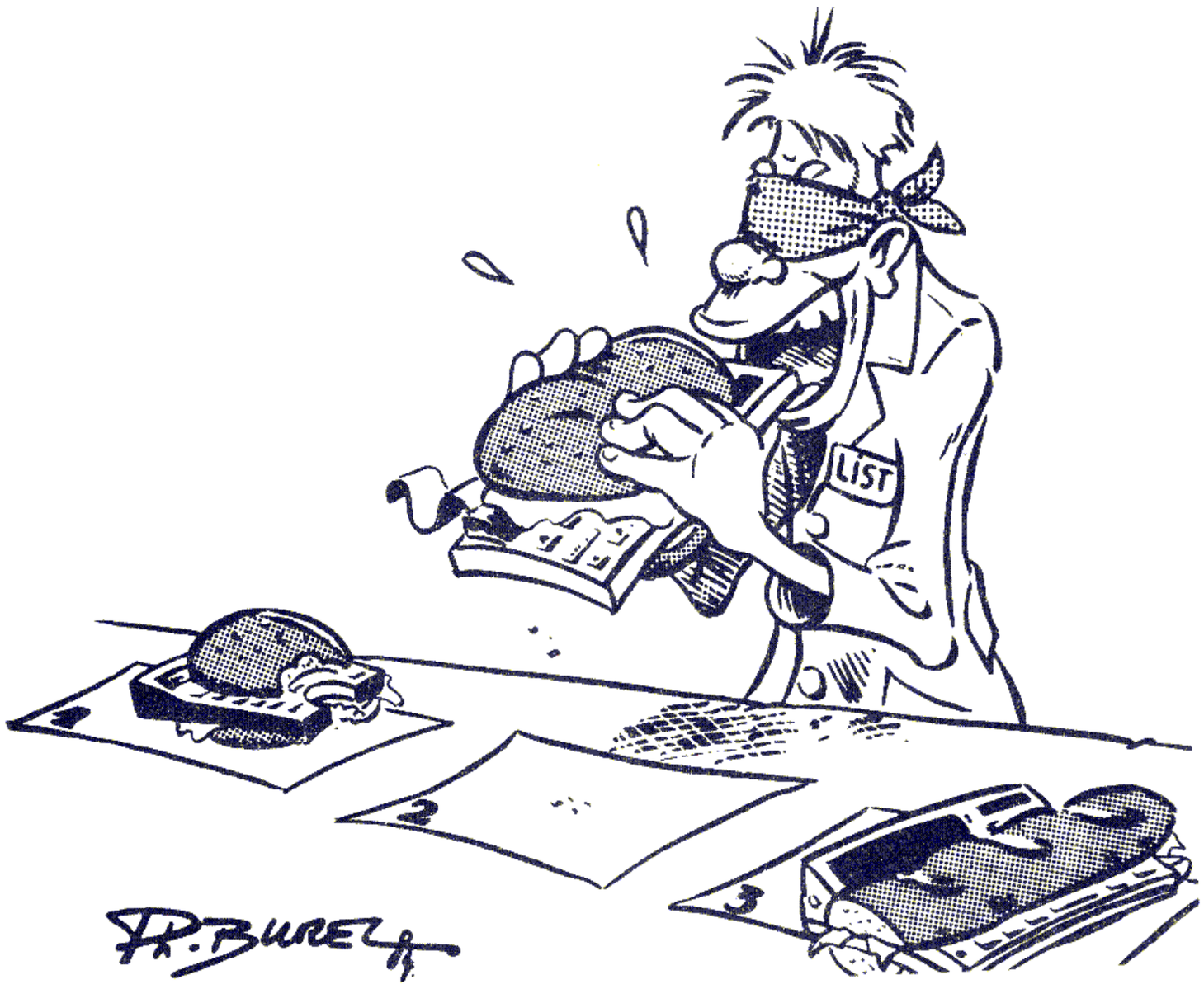
Les tests 9 et 10 subissent les mêmes transformations. Ainsi, le test 9 devient :

```
5   A$="LISTEST"
6   X=OPENOUT "FICHIER"
10  FOR I=1 TO 10000
15  PRINT # X,A$
20  NEXT I
25  CLOSE # X
30  END
```

Avec son 6502 à 2 MHz, le Basic du BBC est l'un des plus rapides du marché actuel. Seules les routines scientifiques ne semblent pas optimisées.

Les dix tests ont été appliqués à deux ordinateurs de conception différente. Cela peut donner une idée de leurs capacités respectives. Il ne s'agit pas, ici, de les comparer.

LIST



LES DIX TESTS DE LIST

MESURER LE BASIC

***Test 1** - Boucle vide*

```
10  FOR I = 1 TO 10000
20  NEXT I
30  END
```

***Test 2** - Sous-programmes*

```
10  FOR I = 1 TO 10000
15  GOSUB 100
20  NEXT I
30  END
100 GOSUB 110
110 RETURN
```

***Test 3** - Matrice*

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

***Test 4** - Opérations sur les chaînes de caractères*

```
 5  A$ = « LISTEST »
10  FOR I = 1 TO 10000
15  B$ = LEFT$(A$,2)
      + MID$(A$,3,3)
      + RIGHT$(A$,2)
20  NEXT I
30  END
```

***Test 5** - Arithmétique*

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

***Test 6** - Calcul scientifique*

```
10  FOR I = 1 TO 10000
15  J = SIN (LOG(I))
20  NEXT I
30  END
```

***Test 7** - Affichage*

```
10  FOR I = 1 TO 10000
15  PRINT CHR$(11) ; « LIS
    TEST » ; I
20  NEXT I
30  END
```

***Test 8** - Tracé d'une ligne graphique*

```
10  FOR I = 1 TO 10000
15  LINE (0,0) – (319,199)
20  NEXT I
30  END
```

***Test 9** - Ecriture de fichiers*

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

***Test 10** - Lecture de fichiers*

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

Les dix tests appliqués au QX-10

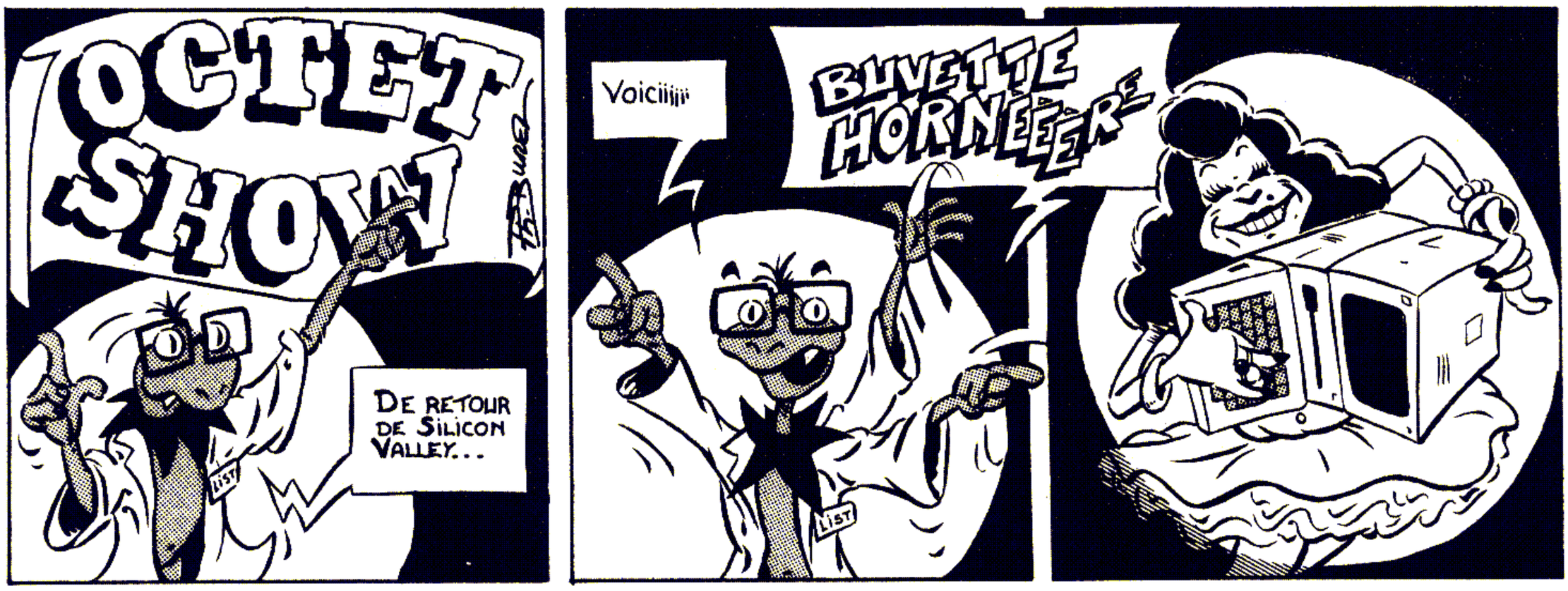
Tests	Résultats	
1 - Boucle vide	21	
2 - Sous-programmes	56	
3 - Matrice	85	
4 - Chaînes de caractères	146	
5 - Arithmétique	99	
6 - Calcul scientifique	305 (16 chiffres)	
7 - Affichage	499	
8 - Tracé graphique	67 (320 × 200)	67 (640 × 400)
9 - Ecriture fichier	132 (disquette)	
10 - Lecture fichier	126 (disquette)	

Les temps sont exprimés en secondes.
(Ces résultats nous ont été envoyés par Jean-Marie Donat.)

Les dix tests appliqués au BBC

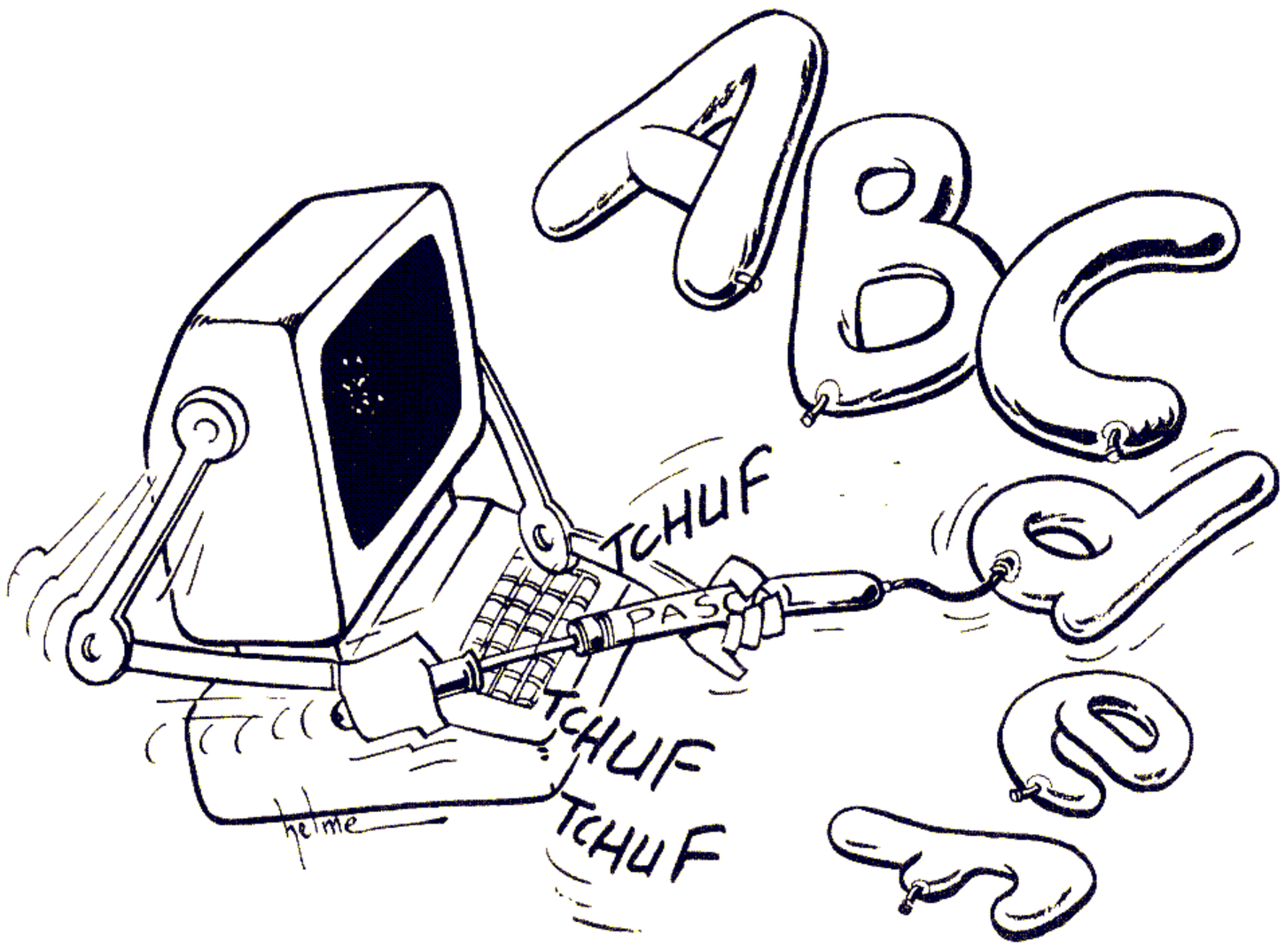
Tests	Résultats	
1 - Boucle vide	7	
2 - Sous-programmes	15	
3 - Matrice	32	
4 - Chaînes de caractères	41	
5 - Arithmétique	57	
6 - Calcul scientifique	405 (9 chiffres)	
7 - Affichage	114	
8 - Tracé graphique	239 (160 × 256)	468 (640 × 256)
9 - Ecriture fichier	1944 (cassette)	
10 - Lecture fichier	1944 (cassette)	

Les temps sont exprimés en secondes.



MINUSCULE EN MAJUSCULE

UNE procédure Pascal permet de convertir une chaîne comportant éventuellement des minuscules en une chaîne identique, mais ne possédant que des majuscules. Voilà qui donnera un peu de souplesse aux applications informatiques.



■ La conversion des minuscules en majuscules passé, en Pascal, par une procédure. Celle-ci est déclarée : *procedure majuscules (var chaine : string) ;*

L'unique paramètre de ce sous-programme, CHAINE, est transmis par référence. Au moment de l'activation de cette routine, il doit contenir une chaîne de caractères. Cette variable est retournée, et contient au retour la chaîne originale, dont les lettres alphabétiques ont été converties en majuscules.

Cette procédure transforme deux types de caractères. La première catégorie concerne les lettres minuscules courantes. Celles-ci sont tout simplement transformées en majuscules.

Cédilles et accents divers

Le second type de caractères qui est affecté par ce sous-programme est celui des lettres propres aux claviers français AZERTY. En effet, ces claviers génèrent les lettres suivantes : *a* accent grave, *c* cédille, *e* accent aigu, *e* accent grave, et *u* accent grave.

Ces différents caractères sont convertis en la lettre majuscule qui leur est associée.

La conversion de chaînes de caractè-

UNE PROCÉDURE PASCAL

MINUSCULE EN MAJUSCULE

res est très utile, dès que l'on désire donner un peu de souplesse aux programmes informatiques. Considérons en effet une application comme la gestion du personnel d'une entreprise. Si cet ensemble de programmes est suffisamment convivial, l'accès aux individus s'effectue, non pas à partir d'un matricule, mais plutôt à l'aide d'un nom abrégé.

Celui-ci sera généralement une chaîne de huit ou dix caractères extraite du nom complet de la personne. Ce peut être, par exemple, les dix premiers caractères, ou les dix premières consonnes du nom. Le nom abrégé représente donc une clef d'accès.

Si l'on observe le comportement des utilisateurs d'un tel système, on constate immédiatement que ces noms abrégés sont souvent saisis d'une façon incorrecte. Telle clef est entrée en minuscules, telle autre est indiquée avec une initiale majuscule, les autres lettres étant minuscules, une troisième étant saisie sans accent alors que le nom complet en comporte un.

Il est bien difficile avec tout cela de retrouver un individu dans le fichier du personnel. Et que dire des noms abrégés qui ont été saisis, au moment de la création du fichier, en minuscules, et qui sont recherchés en majuscules.

La procédure permet de résoudre ce genre de problème, puisque toutes les lettres minuscules sont converties en majuscules, et puisque les accents disparaissent. Dans ce cas précis, un appel à cette routine sera donc effectué, une première fois lors de l'enregistrement des noms abrégés dans le fichier, et une seconde fois lorsque l'utilisateur du système effectuera la saisie d'une telle clef.

La programmation de la routine est assez aisée, à condition de posséder les équivalents des lettres spécifiques à la France.

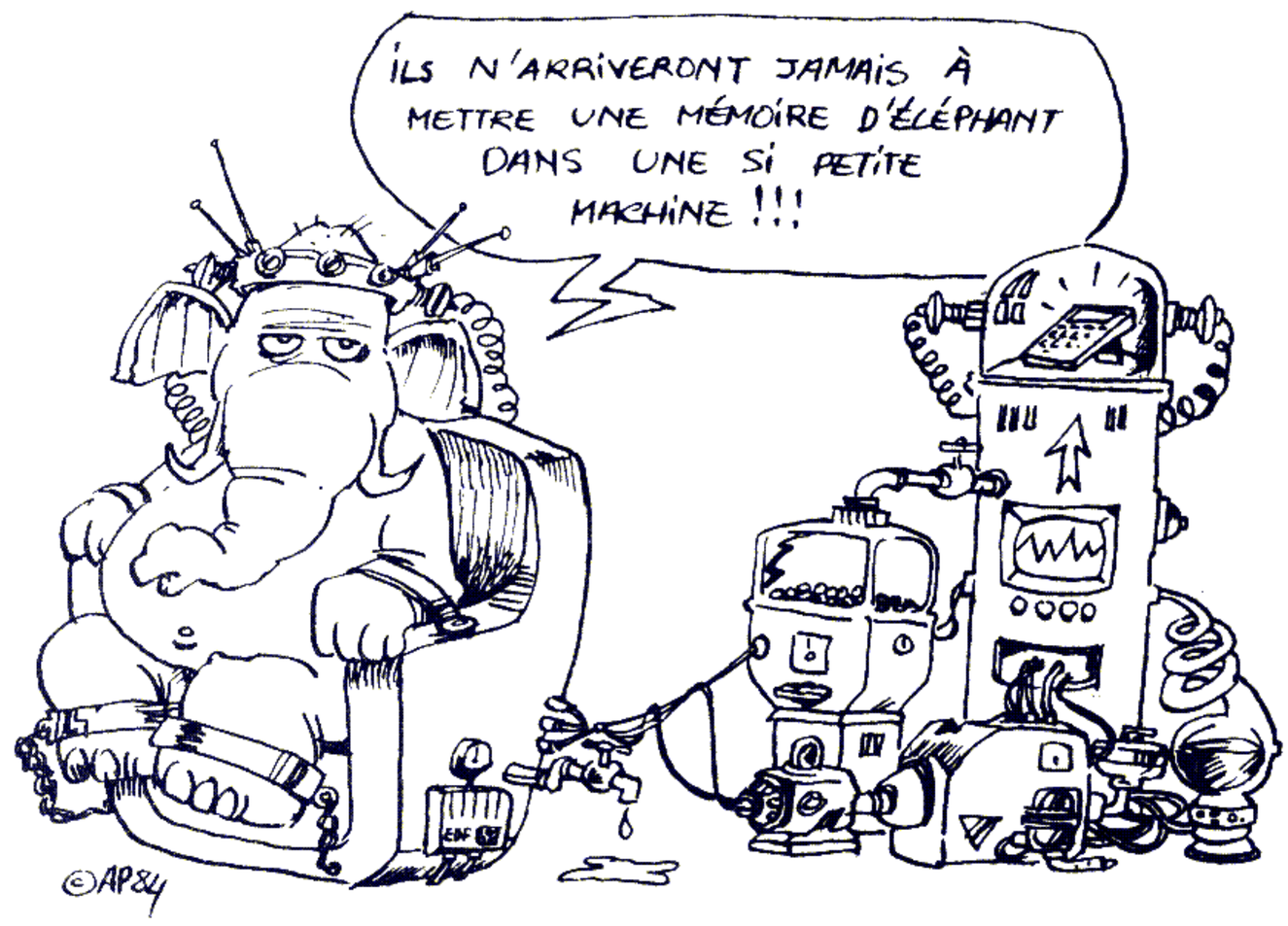
La procédure effectue une boucle sur la totalité des caractères de la chaîne, passée en paramètre. Ceux-ci sont analysés un par un pour savoir s'ils doivent être traités.

Dans un premier temps, le sous-programme recherche si le caractère est une lettre appartenant à l'ensemble des minuscules. Si tel est le cas, il est transformé en majuscule en retranchant à son code ASCII celui de la lettre 'a', et en lui ajoutant celui de la lettre 'A'.

Si le symbole en cours de traitement n'appartient pas à l'ensemble des minuscules, il est recherché dans celui des lettres accentuées ou cédillées. S'il y est présent, un aiguillage, réalisé par la construction CASE, permet la conversion en majuscule.

Enfin, lorsque la lettre a été traitée, elle est mémorisée dans la chaîne, en remplacement de celle qui y était auparavant.

Et c'est ainsi que les minuscules deviennent toutes des majuscules.



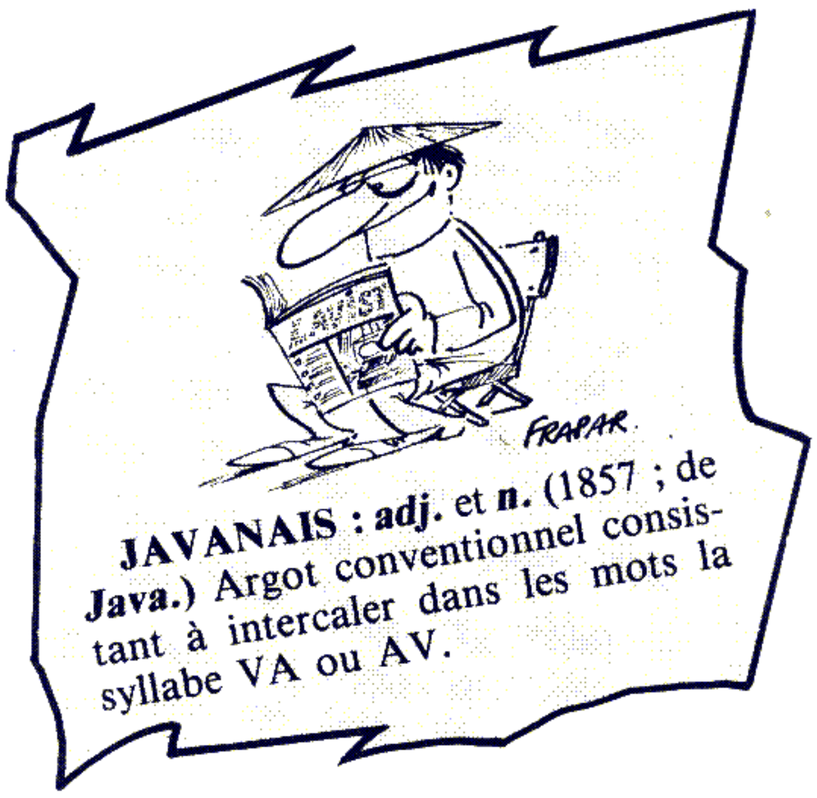
Conversion majuscule
Programme en Pascal
Auteur Thierry Chamoret


```
procedure majuscules (var chaine : string) ;
var i    : integer ;
    car : char ;
begin
    for i : =1 to length (chaine) do
        begin
          car : =chaine[i] ;
          if car in ['a'..'z']
            then
              car : =chr(ord(chaine[i])-ord('a')+ord('A'))
            else
              if car in ['à','ç','é','è','ù']
                then
                  case car of
                    'à'     : car : = 'A' ;
                    'ç'     : car : = 'C' ;
                    'é','è' : car : = 'E' ;
                    'ù'     : car : = 'U'
                  end ;
          chaine[i] : = car
        end
end ;
```

Thierry CHAMORET

JAVANAIS EN BAVASAVIC

APRÈS le Basic, le Pascal, le Forth et bien d'autres, le Javanais va s'introduire dans votre ordinateur.
De deux manières au choix, l'une instantanée, primesautière et approximative, l'autre plus lente, plus réfléchie, mais aussi, peut-être, infiniment plus efficace.



JAVANAIS : adj. et **n.** (1857 ; de **Java.**) Argot conventionnel consistant à intercaler dans les mots la syllabe VA ou AV.

Le Javanais, rappelons-le à ceux qui auraient travaillé pendant leurs récrés à l'école primaire (!), est cette langue secrète qui consiste à ajouter la syllabe AV avant chaque voyelle : bec se dit bAVec ; bonjour, bAVonjAVour, et, par conséquent, javanais, jAVavAVanAVais.

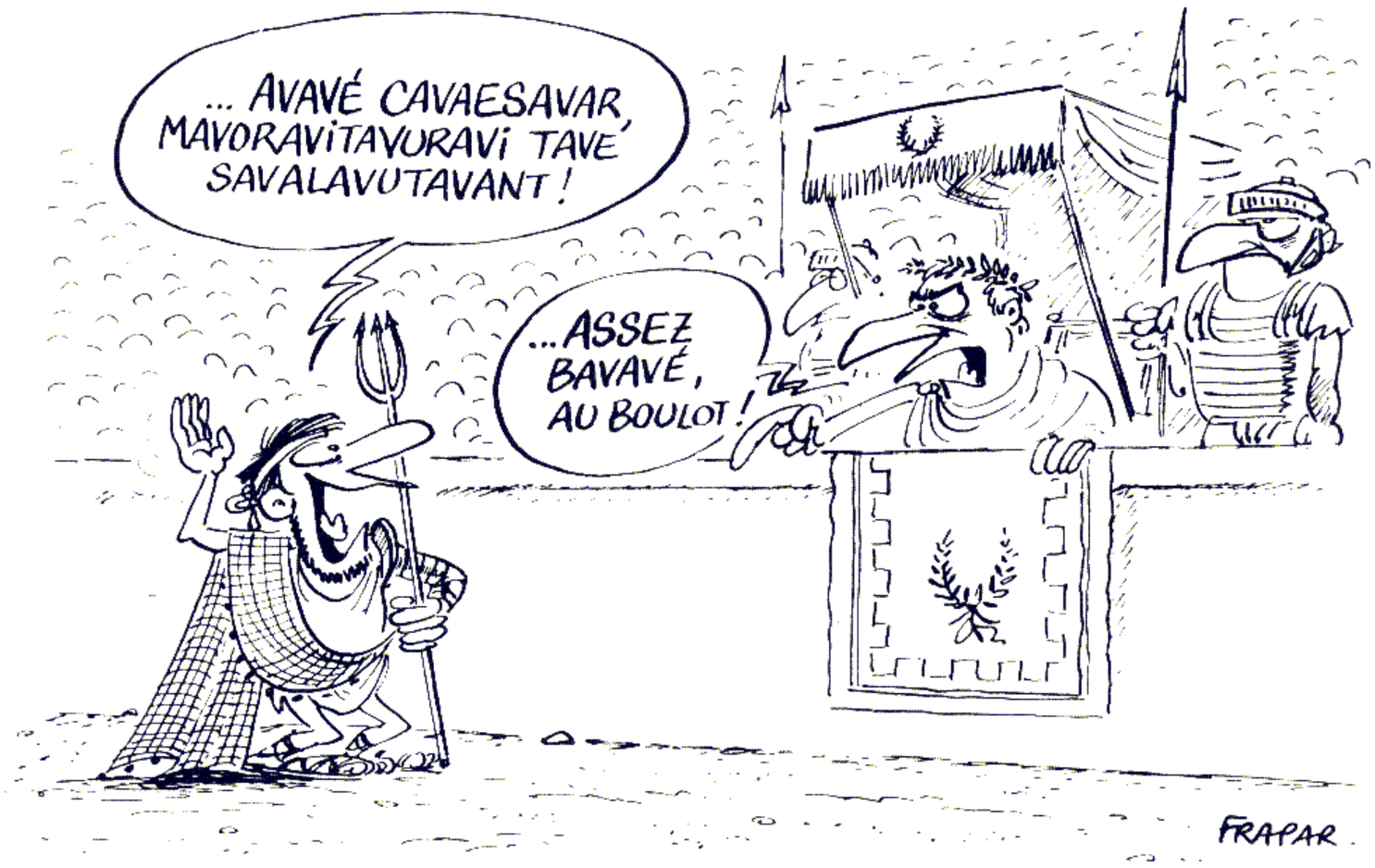
Pour l'ordinateur, traduire comporte trois opérations : la saisie du texte français, le traitement du français et la sortie du texte javanais.

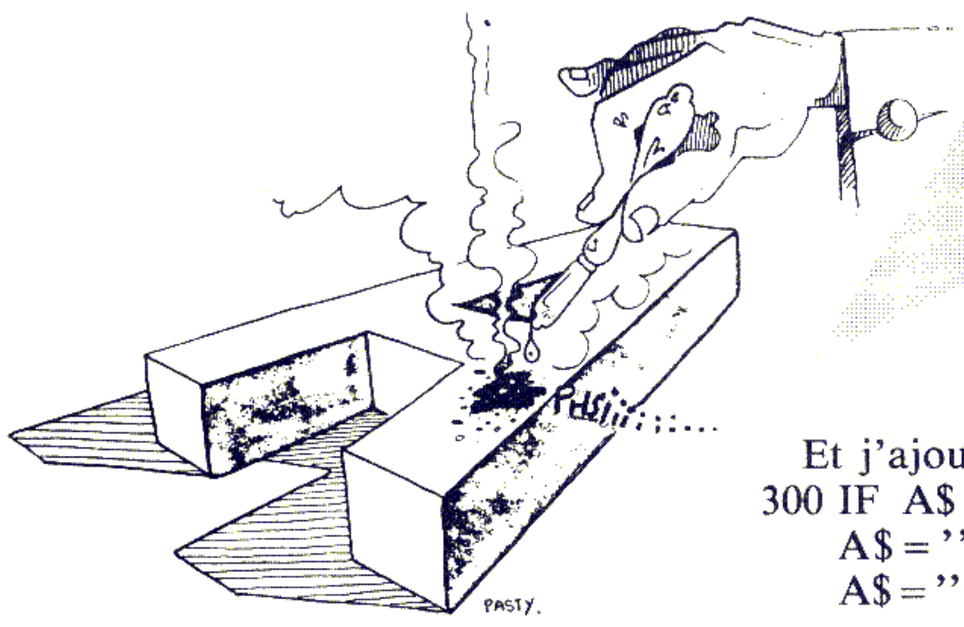
Texte français... texte français... tout de suite les grands mots ! En Basic, un texte, c'est tout simplement une chaîne de caractères ; notre *saisie* se réduira donc pour le moment à attendre le texte français caractère par caractère. Ce qui va donner, si l'on veut traduire « en temps réel », comme on dit, GET R$ (sur certains Basics, ce sera R$ = INKEY$), avec une boucle d'attente quand rien ne vient. Notre *sortie*, pour le moment, se contentera de reproduire le français, puisque c'est au *traitement* d'ajouter là où il faut les AV caractéristiques. Pour le moment toujours, il peut se réduire à PRINT R$ suivi d'un point-virgule. Le *traitement* mérite réflexion.

Si la traduction consiste à ajouter AV devant chaque voyelle, cela veut dire qu'il faut tester chaque caractère.

Pour tester si l'on est en présence d'une voyelle ou d'une consonne, on partira du principe philosophique bien connu selon lequel tout ce qui n'est point voyelle est consonne. Et comme, dans notre belle langue française, le « Y » fonctionne rarement comme une vraie voyelle, nous le laisserons choir et nous bornerons à ajouter AV devant les autres voyelles. Retroussons donc nos manches et frappons gaiement :

```
100 GET R$:IF R$ = '' '' THEN 100
110 IF R$ = ''A'' OR R$ = ''E'' OR
    R$ = ''I'' OR R$ = ''O'' OR
    R$ = ''U'' THEN GOSUB 300
```





« ...On va tester les voyelles... »

```
200 PRINT R$;
210 GOTO 100
300 REM
400 PRINT ''AV'';
410 RETURN
```

Il vaut mieux numéroter large quand on commence ! Ne pas oublier que certains Basics ne comprennent pas le GET R$ de la ligne 100. Il faudra le remplacer par R$ = INKEY$, par exemple. Et après RUN, en frappant BEC, on obtient BAVEC. Essayez aussi avec SALUT, JAVA ou CHARLEMAGNE FILS DE PEPIN LE BREF. Tout se passe bien...

Maintenant, tapez BONJOUR. Le résultat ne se fait pas attendre : BAVONJAVOAVUR.

La voyelle double : un premier os

Tiens ! Il y a un os. Nous venons de choir sur la plaie de la programmation, la peste de l'analyse, que dis-je, le phylloxéra de l'algorithme : le *cas particulier* !

On peut se dire que ce n'est que le début, car un cas particulier peut en cacher un autre... Ici, c'est une simple voyelle double. Bonbonbon. Je reste cmale... lacme... calme, je vais vider un ou deux cendriers, je range une feuille de papier qui dépasse d'un millimètre et produit un effet d'inacceptable désordre sur ma table de travail si méticuleusement rangée (vous savez ce que c'est) et je vais m'armer d'une barre à mines d'horloger pour attaquer le problème avec délicatesse et circonspection.

Si les groupes de voyelles gênent, on va tester les groupes de voyelles, oh là là, on ne va pas en faire une histoire. On ne mettra pas de AV si la lettre qui précède est une voyelle et puis c'est tout ! Voyons, je garde en mémoire la lettre précédente par une « siouxe » ligne :

```
120 A$ = R$
```

Et j'ajoute finement :

```
300 IF A$ = ''A'' OR A$ = ''E'' OR
    A$ = ''I'' OR A$ = ''O'' OR
    A$ = ''U'' THEN RETURN
```

Toc ! Testons : BONJOUR donne... BAVONJAVOUR.

Ça colle. Essayez encore avec VIVE LE GRAND BABU ! (1), ou avec DES FIGUES, DES BANANES, DES NOIX.

Et là, des jurons énergiques retentissent : ils sont le contrepoint naturel et nécessaire des soirées de programmation.

C'est « FIGUE » qui ne marche pas. En Javanais correct, « figue » se dit « favigue », et non « FAVIGAVUE », enfin quoi... Encore un cas particulier. Et je parie que la même chose existe avec les mots en « QUE ». Si le découragement croit qu'il va nous abattre...

Il va de soi que le reste du travail va être consacré aux cas particuliers, alors j'ajoute, courtoisement, mais tout en n'en pensant pas moins, une ligne de ce style :

```
310 IF R$ = ''U'' AND (A$ = ''G'' OR
    A$ = ''Q'') THEN RETURN
```

Voilà voilà. On re-teste : DES FIGUES devient DAVES FAVIGUES. Essayons encore avec ENFIN UN PROGRAMME..., puis avec QUI AGIT BIEN. Dernier résultat : QUI AVAGAVIT BAVIEN. « AVAGAVIT » ? « GAVIT » ? « GEAVIT » ! Oui, alors celui-là, on aurait pu y penser avant. C'est le « G » qui se prononce « J » devant « I » ou « E ». Non non, je ne me fâche pas, je remédie, c'est tout, je remédie...

```
320 IF A$ = ''G'' AND (R$ = ''E'' OR
    R$ = ''I'') THEN PRINT ''E'';
```

Naturellement, il va rester bien d'autres cas particuliers. Pour les découvrir, frappons successivement les Mémoires de Saint-Simon, les pages jaunes de l'annuaire, le dictionnaire de l'Académie ou la réédition de 1972 de « Signé Furax » !

---

(1) A moins que vous ne soyez encore en couche-culotte, je serais bien surpris que vous ignorassiez le chef-d'œuvre de Pierre Dac et Francis Blanche. Si tel était le cas, écoutez ou lisez ; vous y apprendrez l'hymne du grand BABU :
Des figues des bananes des noix,
Des noix des bananes des figues...
A vous de découvrir la suite.

Cela dit, ne nous leurrons pas, les difficultés qui nous attendent sont telles qu'il nous sera impossible de les résoudre dans une traduction simultanée, car le programme devra tenir compte également de la lettre qui suit la lettre considérée : il va falloir nous résoudre à abandonner le « temps réel ».

En lieu et place de la traduction simultanée, la traduction séquentielle peut permettre au programme d'explorer au mieux les cas de figures plus subtils. Ce que je vous propose ici, c'est un embryon de programme, qui ne demande qu'à être complété par vos améliorations et vos astuces. Vous n'avez pas été sans remarquer, par exemple, que « QUI » devrait devenir « QUAVI ». Comment concilier « QUI », « AMERIQUE » et « PIQURE » ? « FIGUE » et « FIGURE » ? Voir plus loin...

Préparer la chaîne

Les trois parties essentielles du programme seront les mêmes, mais elles seront traitées différemment, puisqu'au lieu du GET, nous allons utiliser INPUT, et la chaîne de caractères français — car maintenant, il ne s'agit plus de caractères isolés mais de chaînes — sera baptisée A$.

Malheureusement, INPUT ne se prête pas toujours de bonne grâce à la saisie de phrases entières, puisque les virgules et les deux-points sont interprétés de manière fort peu littéraire par les machines qui ne connaissent pas le LINEINPUT.

Une solution peut consister à forcer des guillemets en tête de la réponse, mais on peut préférer un sous-programme par GET qui supprime les mouvements de curseur.

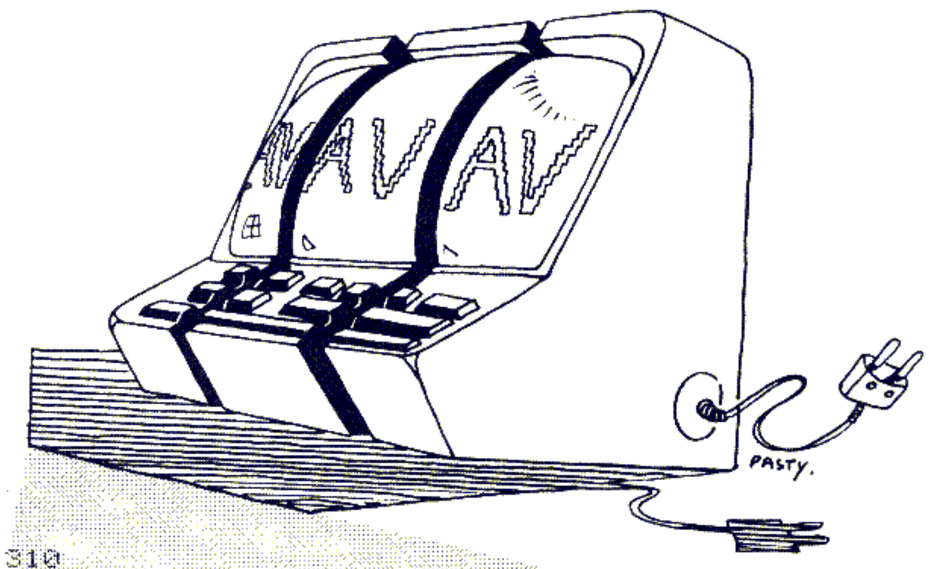
La prudence conseille peut-être de préparer une chaîne pour le futur texte javanais. Evidemment, au début du programme, cette chaîne sera vide encore ; ça donne quelque chose comme : B$ = '' ''. Notre *sortie* de texte peut se réduire encore à PRINT B$.

Si la traduction consiste à ajouter AV devant chaque voyelle, cela veut dire d'une part qu'il va falloir prendre un par un chaque caractère de la chaîne A$, et d'autre part qu'il va falloir le tester.

Pour prendre un par un chaque caractère d'une chaîne, tout Basic un peu prévoyant possède dans son atelier la fonction MID$. Comment ? Vous ne connaissez pas encore MID$? Mais per-

Javanais
Programme en Basic
Auteur François J. Bayard


```
200 GOTO 700
210 REM
220 REM                    ===============
230 REM                    SOUS-PROGRAMMES
240 REM                    ===============
250 REM
260 REM ------------------ INPUT AMELIORE
270 REM
300 A$="":P$=CHR$(166):LA=0:PRINT P$;
310 GET R$:IF R$="" THEN 310
320 R=ASC(R$):IF R$="" AND R=13 THEN 4000
330 IF R=13 THEN PRINT CHR$(157);CHR$(32):RETURN
340 IF (R>16 AND R<20) OR (R>144 AND R<149) OR R=29 OR R=157 THEN 310
350 IF R=20 AND LA=0 THEN 310
360 PRINT CHR$(157);R$;P$;:A$=A$+R$:LA=LA+1
370 IF R=20 THEN LA=LA-2:A$=LEFT$(A$,LA)
380 IF LA>253 THEN PRINT"■ ":PRINT"TROP LONG !":GOTO 300
390 GOTO 310
400 REM
410 REM ------------------- VOYELLE ?
420 REM
500 V=0:IF C$="A" OR C$="E" OR C$="I" OR C$="O" OR C$="U" THEN V=-1
510 RETURN
520 REM
530 REM ------------------- FORMATAGE
540 REM
600 IF LEN(B$)<LE THEN PRINT#1,B$:RETURN
610 FOR I=LE-1 TO 1 STEP-1:IF MID$(B$,I,1)=" " THEN L=I:I=1:GOTO 630
620 NEXT I:L=LE-1
630 PRINT#1,LEFT$(B$,L-1)
640 B$=MID$(B$,L+1):GOTO 600
650 REM
660 REM                    ===============
670 REM                    INITIALISATIONS
680 REM                    ===============
690 REM
700 PRINT CHR$(142):REM MAJUSCULES
710 REM SUR CBM 3000 OU 4000 PETIT ECRAN, POKE 59468,12
720 LE=40:REM SUR VIC 20 LE=22
730 PRINT"■■JAVANAIS■■"
740 PRINT"■SORTIE SUR ■E■CRAN OU SUR ■I■MPRIMANTE ?"
750 PRINT,,,"■(E/I)"
760 GET R$:IF R$<>"I" AND R$<>"E" THEN 760
770 NP=3:IF R$="I" THEN NP=4:LE=80
780 OPEN 1,NP:PRINT CHR$(147)
790 REM
800 REM                    ================
810 REM                    BOUCLE PRINCIPALE
820 REM                    ================
830 REM
900 GOSUB 300:REM INPUT A$ AMELIORE
910 REM
920 REM ------------------- TRADUCTION
930 REM
940 REM PREPARATION DES CHAINES
950 REM
1000 A$=CHR$(32)+A$+CHR$(32):B$=""
1010 REM
1020 REM EXAMEN DE A$
1030 REM
1040 FOR I=2 TO LEN(A$)-1
1050 REM
1060 REM C1$=LETTRE EN COURS   :V1=-1 SI VOYELLE, 0 SI CONSONNE
1070 REM C0$=LETTRE PRECEDENTE:V0=-1 SI VOYELLE, 0 SI CONSONNE
1080 REM C2$=LETTRE PRECEDENTE:V2=-1 SI VOYELLE, 0 SI CONSONNE
1090 REM
1100 C0$=MID$(A$,I-1,1):C$=C0$:GOSUB 500:V0=V
1110 C1$=MID$(A$,I,1)  :C$=C1$:GOSUB 500:V1=V
1120 C2$=MID$(A$,I+1,1):C$=C2$:GOSUB 500:V2=V
1200 IF V1 THEN GOSUB 3000
2000 B$=B$+C1$
2010 NEXT I
2020 PRINT
2030 GOSUB 600
2040 PRINT
2050 GOTO 900
2900 REM
2910 REM CONDITIONS POUR AJOUTER "AV"
2920 REM
3000 IF V0 THEN RETURN
3010 IF C1$="U" AND (C0$="G" OR C0$="Q") THEN RETURN
3400 IF C0$="G" AND (C1$="E" OR C1$="I") THEN B$=B$+"E"
3500 B$=B$+"AV"
3510 RETURN
3950 REM
3960 REM                          =====
3970 REM                          F I N
3980 REM                          =====
3990 REM
4000 CLOSE 1
4010 END
READY.
```



Ne découpez pas votre ordinateur en tranches, mais le programme.

mettez que je vous présente : si A$ = ''ABCDEFGHI'', alors MID$(A$,3,2) prend le troisième caractère, compte jusqu'à deux, et retourne donc CD. MID$(A$,2,5) prend le deuxième caractère, compte jusqu'à cinq et sort BCDEF. Dans certains Basics, on peut même se dispenser du troisième argument, et l'ordinateur comprendra « tout le reste » ! MID$(A$,4) retourne DEFGHI.

Une ficelle : le drapeau

Les possesseurs de ZX 81 savent se passer de MID$. Pour obtenir CD, ils font A$ (3 TO 4), pour BCDEF, A$(2 TO 5), et pour DEFGHI, A$(4 TO).

Si donc je veux prendre un par un les caractères d'une chaîne A$, rien de tel qu'une bonne vieille boucle du genre :

```
100 FOR I=1 TO LEN(A$)
110 C$=MID$(A$,I,1)
120 NEXT I
```

En « zédixois », on remplacera bien sûr la ligne 110 par 110 LET C$=A$(I), et si vous voulez vérifier ce que j'avance, ajoutez quand même 115 PRINT C$, sinon, le spectacle sera mince.

Pour le test « voyelle ou consonne », ce qui m'intéresse, c'est de garder en mémoire le résultat. Dans ma boîte à ficelles, j'en ai une qui s'appelle le drapeau (en anglais FLAG). Il s'agit d'une variable contenant un nombre qui ne

Exercice

Quelques phrases et expressions usuelles que vous pouvez donner à traduire au programme :

Anticonstitutionnellement. Nitchevo. Va vite voir arriver Tatave, le laveur batave. Vive la vie, Zarie. Petit pot de beurre, quand te dépetitpotdebeurreras-tu ? Je me dépetitpotdebeurrerai quand tous les petits pots de beurre se seront dépetitpotdebeurrés. My taylor is rich.

représente pas une quantité, mais le fait qu'une idée, une proposition, une égalité soit vraie ou fausse. Si la proposition est fausse, la variable vaut 0, si elle est vraie, la variable vaut −1 ou 1 selon les machines (−1 chez Commodore, 1 sur ZX 81). Mieux encore, au lieu de dire IF V = −1, il suffira de dire IF V tout court, l'ordinateur comprendra. Et au lieu de IF V = 0, IF NOT V marche aussi.

Le test donnera donc : V = 0:IF C$ = ''A'' OR C$ = ''E'' OR C$ = ''I'' OR C$ = ''O'' OR C$ = ''U'' THEN V = −1. En « zédixois », on ne manquera pas d'ajouter les LET réglementaires.

Le cœur du programme pourrait se présenter ainsi :

```
100 B$ = '' ''
110 INPUT A$:REM OU
    EQUIVALENT
120 FOR I = 1 TO LEN(A$)
130 C$ = MID$(A$,I,1):GOSUB 500
200 IF V THEN GOSUB 600
210 B$ = B$ + C$
220 NEXT I
230 PRINT B$
240 GOTO 100
500 V = 0:IF C$ = ''A'' OR C$ = ''E''
    OR C$ = ''I'' OR C$ = ''O'' OR
    C$ = ''U'' THEN V = −1
510 RETURN
600 B$ = B$ + ''AV''
610 RETURN
```

Des blancs très utiles

Seulement, pour pouvoir examiner le caractère précédent, il faut étoffer un peu tout ça : faire de C$ et de V des variables temporaires et créer, par exemple, un C1$ qui sera le caractère en cours, avec un drapeau V1 à −1 si c'est une voyelle, et un C0$ qui sera le caractère précédent, avec un drapeau V0. Pendant qu'on y est, on peut même ajouter, à toutes fins utiles, un C2$ et un V2 pour le caractère suivant.

```
100 B$ = '' ''
110 INPUT A$:REM OU
    EQUIVALENT
120 FOR I = 1 TO LEN(A$)
130 C1$ = MID$(A$,I,1)
    :C$ = C1$:GOSUB 500:V1 = V
140 C0$ = MID$(A$,I − 1,1):C$ = C0$:
    GOSUB 500:V0 = V
150 C2$ = MID$(A$,I + 1,1):C$ = C2$:
    GOSUB 500:V2 = V
200 IF V1 THEN GOSUB 600
210 B$ = B$ + C1$
220 NEXT I
230 PRINT B$
240 GOTO 100
500 V = 0:IF C$ = ''A'' OR C$ = ''E''
    OR C$ = ''I'' OR C$ = ''O'' OR
    C$ = ''U'' THEN V = −1
510 RETURN
600 IF V0 THEN RETURN
610 IF C1$ = ''U'' AND (C0$ = ''G''
    OR C0$ = ''Q'') THEN RETURN
620 IF C0$ = ''G'' AND (C1$ = ''E''
    OR C1$ = ''I'') THEN
    B$ = B$ + ''E''
630 B$ = B$ + ''AV''
640 RETURN
```

On fait un RUN, et bing ! on se plante : ILLEGAL QUANTITY ERROR IN 140. Ah oui, c'est vrai, au premier tour de boucle, I vaut 1, I − 1 vaut 0 et il n'y a pas de 0e caractère à la chaîne A$.

La solution ? mais c'est tout simple voyons ! Et pourtant, j'ai mis un moment à la trouver ! On va ajouter un blanc en tête de A$, ça ne gênera personne, et puis aussi un en queue. On modifiera les limites de boucle en conséquence. Hop ! exécution :

```
115 A$ = '' '' + A$ + '' ''
120 FOR I = 2 TO LEN(A$) − 1
```

Et on refait RUN. Ça a l'air d'aller cette fois. On va tout de même ajouter un peu de sauce, avec une routine de formatage des sorties, le choix de sortie sur écran ou sur imprimante (en paramétrant le numéro de périphérique). Ce qui donne le programme Basic de Javanais (page précédente), preuve que le Javanais peut trouver son épanouissement dans le Basic !

François J. BAYARD

LAVA CAVIGAVALAVE AVET
LAVA FAVOURMAVI

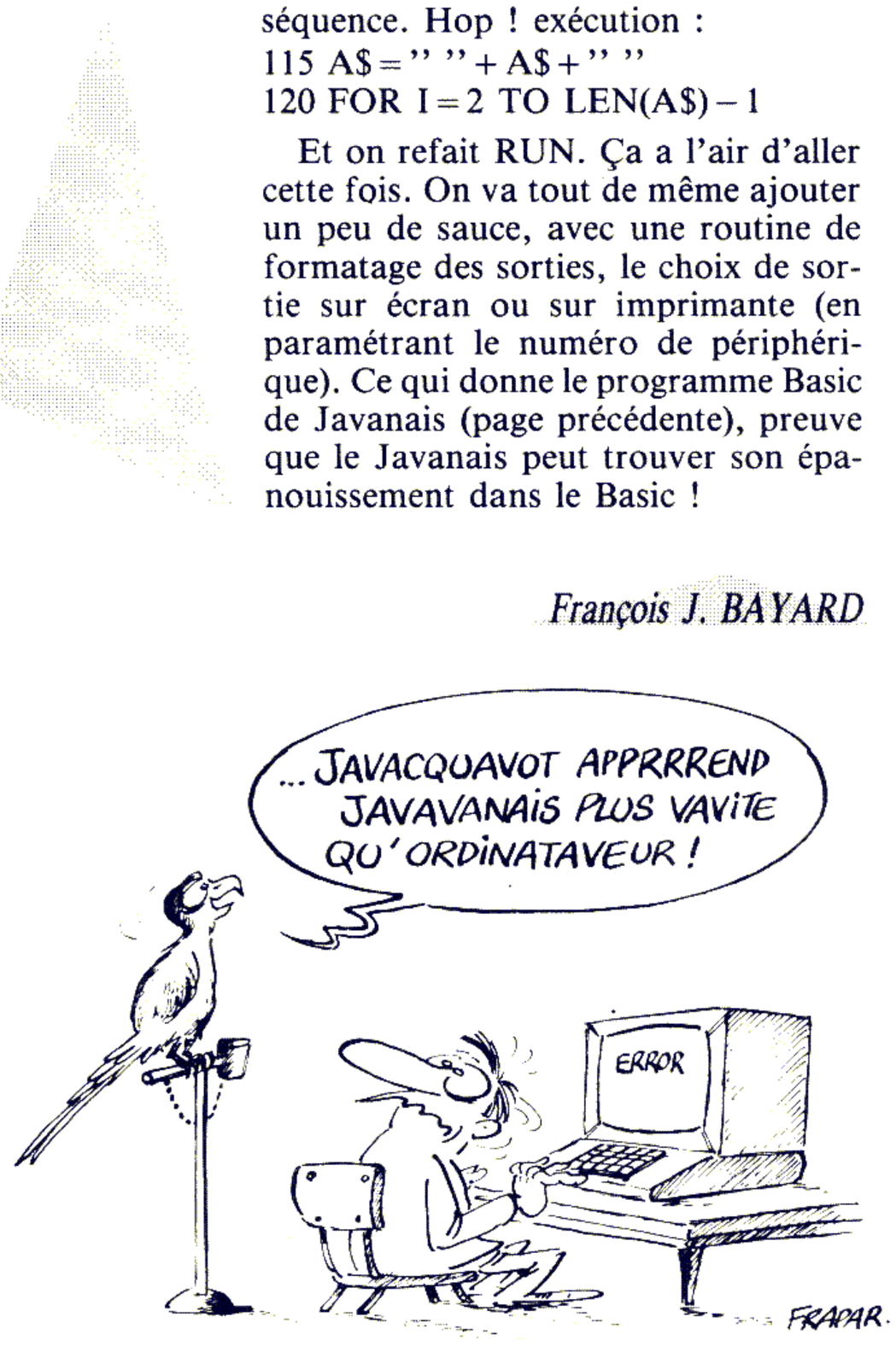
Lava çavigavalave, avayavant chavantavé tavout l'avétavé
Save travouvava favort davépavourvavue
Quand lava bavisave favut vavenavue :
Pavas avun saveul pavetavit mavorçaveau
Dave mavouchave avou dave vavermavissaveau.
Avellave avallava cravier favamavinave
Chavez lava favourmavi sava vavoisavinave,
Lava praviant dave lavui pravêtaver
Quelque gravain pavour savubsavistaver
Javusqu'avà lava savaisavon navouvavellave.
« Jave vavous pavaieravai, lavui davit-avellave,
Avavavant l'avaoût, favoi d'avanavimaval,
Avintavéravêt avet pravinçavipaval. »
Lava favourmavi n'avest pavas pravêtaveusave :
C'avest lavà savon mavoindrave davéfavaut.
« Que favaisaviez-vavous avau tavemps chavaud ?
davit-avellave avà çavettave avempravuntaveusave.
— Navuit avet javour avà tavout vavenavant
Jave chavantavais, nave vavous davéplavaisave.
— Vavous chavantaviez ? J'aven savuis favort avaisave :
Aveh bavien ! davansavez mavaintavenavant.

Lava Favontavainave

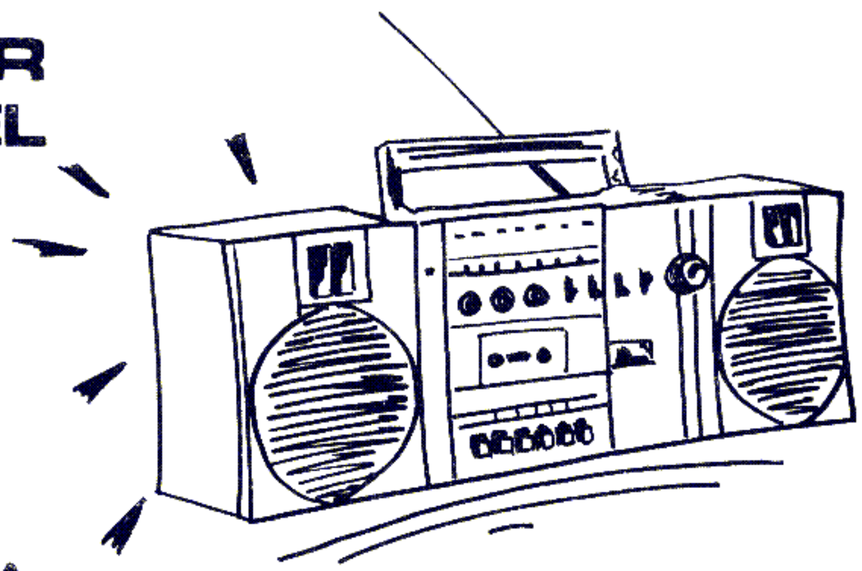
Dans cet exemple d'exécution, seuls les accents, cédilles et majuscules ont été retravaillés (intéressante amélioration que de faire faire ça par le programme). Mais des problèmes plus graves subsistent :

- pour les « QU » car la ligne 3010 supprime les AV devant le « U », mais ne les rétablit pas après. Il faudrait « QUAVAND » au lieu de « QUAND » (vers 4) « QUAVELQUAVE » au lieu de « QUELQUE » (vers 10), « QUAVE » au lieu de « QUE » (vers 17) ;
- pour les diérèses. « CRI-ER » fait deux syllabes. Il faudrait donc « CRAVIAVER ». « PRI-ANT » aussi devrait donner « PRAVIAVANT ».

Bon courage !



L'ORDINATEUR INDIVIDUEL



GILDA presse

Branchez-vous sur le

PETIT ORDINATEUR ILLUSTRÉ

depuis le 10 septembre ce **magazine radio** de 15 minutes
est diffusé **chaque semaine** sur les stations suivantes(*)

Alençon : AFM, 89.4 MHz
Tél. : 26.23.99

Alès : Filasoi
88.6 MHz - Jeudi 19 h.

Amiens : RCC, 101 MHz
Tél. : 92.08.08

Angers : Angers 101,
101 MHz. Tél. : 68.44.44

Angoulême : Radio Marguerite,
99.9 MHz. Tél. : 92.39.39

Bayonne : Radio Adour Navarre,
90.7 MHz. Tél. : 25.50.10

Belfort : Radio Soleil, 88.1 MHz
Samedi 12 h. 05

Besançon : RVF, 98.1 MHz
Tél. : 83.24.22

Bordeaux : Radio 100, 94.3 MHz
Tél. : 52.51.57

Bourges : Radio Recto-Verso,
98 MHz. Tél. : 21.18.18

Brest : FM 101
Samedi 19 h. 30

Brive : Radio Brive Licorne,
95 MHz. Mardi 18 h. 30

Caen : FM 96,8
Dimanche 10 h.

Cannes : Fréquence Sud
97,7 MHz. Samedi 19 h.

Carcassonne : Radio 11,
94.1 MHz. Mercredi 8 h. 45

Castres : Radio Tarn Sud,
97.5 MHz. Vendredi 21 h.

Chambéry : Fréquence Horizon,
100 MHz. Tél. : 69.76.71

Chartres : Radio Loisirs n° 1,
97,3 MHz. Tél. : 21.51.85

Clermont-Ferrand : MU,
96.2 MHz. Tél. : 36.80.30

Dax : ACQS 95, 95.1 MHz
Tél. : 57.81.61

Dijon : Radio 2000, 90.7 MHz
Lundi 21 h. 15

Gap : RTM, 90 MHz
Mardi 13 h.

Guéret : REM, 100.1 MHz
Tél. : 52.79.03

Haute-Loire-Ardèche : Radio RCL,
103 MHz. Tél. : 26.26.00

Lannion : Pays de Trégor,
91.6 MHz. Samedi 9 h. 30

Laval : Perrine
101.3 MHz. Mardi 19 h. 30

La Rochelle : Radio La Rochelle,
92 MHz. Tél. : 41.66.00

Le Havre : EVA
103,5 MHz. Tél. : 89.27.02

Le Mans : FM 104, 104 MHz
Tél. : 82.44.22

Lille : Radio Contact, 93.4 MHz
Tél. : 24.34.34

Limoges : HPS, 102.7 MHz
Tél. : 37.77.54

Lyon : Ciel FM, 96.9 MHz
Tél. : 842.58.55

Marseille : Fréquence Marseille
94,7 MHz. Mardi 20 h. 30

Metz : Radio L, 93.3 MHz
Tél. : 733.21.22

Montpellier : Radio Alligator,
94.5 MHz. Tél. : 92.00.44

Nancy : Rockin'chair, 95.8 MHz
Tél. : 355.16.74

Nantes : Atlantic FM, 96.8 MHz
Mercredi 19 h. 30

Narbonne : Radio Corail,
93.6 MHz. Samedi 9 h. 30

Orléans : Orléans FM, 93.6 MHz
Dimanche 9 h. 15

Paris : Gilda, 103.5 MHz
Dimanche 10 h.

Poitiers : RPO, 90 MHz
Tél. : 58.59.55

Rennes : RBS, 89.1 MHz
Vendredi 21 h.

Rouen : Arlequin, 103 MHz
Dimanche 11 h. 15

St-Etienne : Radio Loire Service,
Tél. : 34.12.23

Salon-de-Provence : Radio
Centuries, 99.7 MHz. Tél. : 42.18.00

Sens : Radio Horizons,
91.2 MHz. Tél. : 95.15.31

Strasbourg : Nuée Bleue,
89.5 MHz. Mardi 18 h. 15

Tarbes : Pirène
98 MHz. Samedi 19 H. 30

Toulon : Radio Mistral,
104 MHz. Tél. 75.03.34

Tours : Méga-Tours, 103 MHz
Tél. : 61.22.88

Troyes : Discone Radio,
92 MHz. Jeudi 18 h. 45

poi petit ordinateur illustré

(*) Pour obtenir l'horaire de diffusion exact du "Petit Ordinateur Illustré", veuillez contacter la radio émettant dans votre région.

UN MINI-MONITEUR POUR PC-1251

LORS de la mise au point des programmes en langage-machine, il est souvent utile de pouvoir contrôler l'état des registres. Et même d'en connaître leurs valeurs. Un ordinateur petit comme le PC-1251 doit se faire aider par un programme en Basic. Ainsi, il affiche la valeur de tous les registres internes.

Pour permettre au PC-1251 de contrôler ses registres internes, commencez par entrer le *Mini-moniteur* ci-contre, programme en Basic, puis faites DEF S pour le stocker en mémoire vive.

Supposons maintenant que vous vouliez tester la routine suivante :

```
03 08  LIB 08
02 00  LIA 00
43     DECA
```

Et n'oubliez pas le JP C110 qui branche au moniteur.

Ajoutez alors le programme à tester (page suivante), toujours en Basic. Faites RUN pour exécuter ce programme en langage-machine. Un simple DEF M donnera alors l'état des registres en affichant en particulier ce que vous attendiez : A = 255.B = 8.NZ C

Vous pourrez améliorer encore le programme en faisant afficher par exemple les registres en hexadécimal et non en décimal.

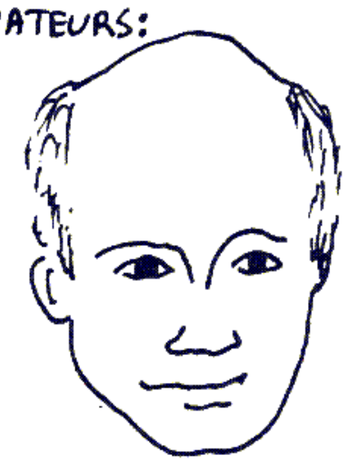
Le seul registre que ne donne pas notre moniteur est DP : nous n'avons trouvé aucun moyen de le lire. Et vous ?

Bernard BOUVIER

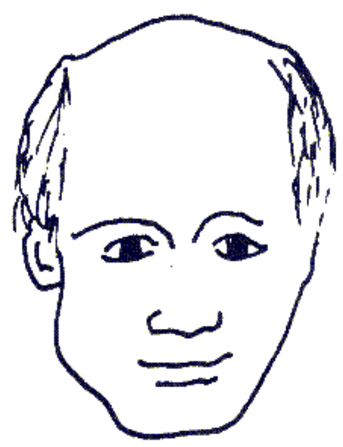
Mini-moniteur
Programme pour PC-1251
Auteur Bernard Bouvier


```
800:"M"A=&C100
805:Z$=" Z": IF PEEK (A+
    1)=0 LET Z$=" NZ"
806:C$=" C": IF PEEK (A+
    2)=0 LET C$=" NC"
810:PRINT "A="; PEEK A;"
    B="; PEEK (A+8);Z$;C
    $
820:PRINT "P="; PEEK (A+
    3);"Q="; PEEK (A+4);
    "R="; PEEK (A+5)
830:PRINT "I="; PEEK (A+
    6);"J="; PEEK (A+7)
840:L= PEEK (A+9):H=
    PEEK (A+10)
850:PRINT "XL=";L;"XH=";
    H;"X=";256*H+L
860:L= PEEK (A+11):H=
    PEEK (A+12)
870:PRINT "YL=";L;"YH=";
    H;"Y=";256*H+L
880:PRINT "K="; PEEK (A+
    13);"L="; PEEK (A+14
    )
890:END
900:"S"A=&C100: RESTORE
    910
905:FOR I=&10 TO &40:
    READ J: POKE (A+I),J
    : NEXT I
906:END
910:DATA &10,&C1,&00,&52
    ,&11,&01,&02,&00,&28
    ,&03,&02,&01,&52,&11
    ,&02,&02
920:DATA &00,&2A,&03,&02
    ,&01,&52,&11,&03,&20
    ,&52,&11,&04,&21,&52
    ,&11,&05
930:DATA &22,&52,&11,&06
    ,&80,&53,&11,&07,&81
    ,&53,&11,&08,&83,&00
    ,&06,&19
940:DATA &37
```

C'EST ÉTONNANT LE PRIX DES ORDINATEURS:



ÇA BAISSE TOUS LES JOURS.



SURTOUT CEUX QUE LE CONSTRUCTEUR A ABANDONNÉS!

Comment ça marche ?

La plupart des registres sont stockés en mémoire vive interne (non accessible par PEEK) de la façon suivante :

00 I
01 J
02 Accu A
03 B
04 XL
05 XH
06 YL
07 YH
08 K
09 L

Le travail du mini-moniteur est de les lire, et de les recopier en mémoire vive accessible aux adresses suivantes :

C100 Accu A
C101 Flag Z
C102 Flag C
C103 P
C104 Q
C105 R
C106 I
C107 J
C108 B
C109 XL
C10A XH
C10B YL
C10C YH
C10D K
C10E L

Par exemple, pour J cela peut se faire par :

```
LIDP C107   ; DP←C107
LP 01       ; P←01
MVDM        ; (DP)←(P)
```

Pour les registres tels que Q qui ne sont pas dans la mémoire vive interne, cela se fait en passant par l'Accu A :

```
LIDP C104   ; DP←C104
LDQ         ; A←Q
STD         ; (DP)←A
```

Ce que ça donne :

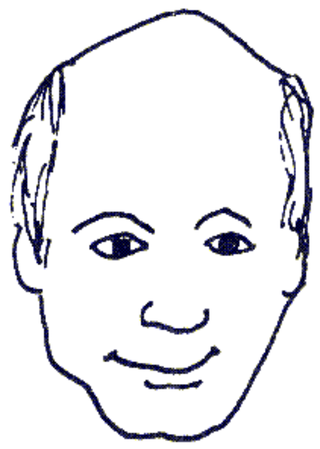
C110	10	C1	00	LIDP	C100	Accu
	52			STD		
	11	01		LIDL	01	Flag Z
	02	00		LIA	00	
C118	28	03		JRNZP		
	02	01		LIA	01	
	52			STD		
	11	02		LIDL	02	Flag C
	02	00		LIA	00	
C121	2A	03		JRNCP		
	02	01		LIA	01	
	52			STD		
	11	03		LIDL	03	P
C128	20			LDP		
	52			STD		
	11	04		LIDL	04	Q
	21			LDQ		
	52			STD		
	11	05		LIDL	05	R
C130	22			LDR		
	52			STD		
	11	06		LIDL	06	I
	80			LP	00	
	53			MVDM		
	11	07		LIDL	07	J
C138	81			LP	01	
	53			MVDM		
	11	08		LIDL	08	B, XL, XH, YL, YH, K, L
	83			LP	03	
	00	06		LII	06	
	19			EXWD		
C140	37			RTN		Fini !

Un programme à tester

```
 5:REM *** PGM A TESTER
10:POKE &C000,&03,&08,&
   02,&00,&43
15:REM *** JP C110
20:POKE &C005,&79,&C1,&
   10
25:REM *** EXECUTION
30:CALL &C000
40:END
```

ILS SONT CARRÉMENT BRADÉS.

LE MARCHAND DU COIN OFFRE UN ORDINATEUR GRATUIT



A TOUT ACHETEUR DE L'INTERFACE DISQUETTE!

DÉTECTEUR DE NOMBRES PREMIERS

La suite de Perrin est une suite numérique qui semble mettre en valeur les nombres premiers. Mais aussi de rares nombres composés. Ces derniers seront éliminés par deux autres suites.

En 1899, un mathématicien français nommé Perrin observait le comportement étonnant d'une suite numérique définie par :

- ses trois premiers termes, u(0) = 3, u(1) = 0 et u(2) = 2 ;
- son terme général, u(n+3) = u(n) + u(n+1) où n est un entier positif.

Le comportement de cette suite, pour n petit, apparaît dans le tableau 1. On constate curieusement que n divise u(n) uniquement lorsque n est un nombre premier. Cela est vrai pour n petit. Mais est-ce toujours vrai ? A l'époque, Perrin ne trouva aucun contre-exemple.

Récemment, deux Américains, Adams et Shanks se penchèrent à nouveau sur cette curiosité oubliée. Ils confirmèrent que la propriété est vraie jusqu'à 271 440. Mais pour n = 271 441, n divise u(n) alors que n n'est pas premier : 271 441 est égal à 521^2.

Les deux Américains eurent alors l'idée d'observer la suite pour des n négatifs. En effet, la récurrence précédente se retourne parfaitement : u(−n) = u(−n+3) − u(−n+1) où −n est inférieur à 2. Les premiers termes de cette suite apparaissent dans le tableau 2. Et là, on remarque que n divise u(−n) + 1 lorsque n est premier. Les contre-exemples sont plus nombreux qu'avec la première suite : 7^2, 7×11, 11^2, etc.

Mais si l'on associe ce test à celui de Perrin, la réponse à la question : « est-ce que u(n) **et** u(−n) + 1 sont divisibles par n ? » est un excellent test de primalité.

Si la réponse est *non,* alors n est

Tableau 1
Les premiers termes de la suite de Perrin

n	0	1	2*	3*	4	5*	6	7*	8	9	10	11*	12	13*
u(n)	3	0	2	3	2	5	5	7	10	12	17	22	29	39

L'astérisque (*) indique que n est premier. On observe alors que n divise u(n).

Tableau 2
Les premiers termes de la suite de Perrin pour $-n \leqslant 2$

n	−2	−1	0	1	2*	3*	4	5*	6	7*	8	9	10	11*	12	13*
u(−n)	2	0	3	−1	1	2	−3	4	−2	−1	5	−7	6	−1	−6	12

L'astérisque (*) indique que n est premier. On observe alors que n divise u(−n) + 1.

composé. Si c'est *oui,* alors n est très probablement premier.

C'est ainsi que 271 441 se révèle cette fois composé grâce à la deuxième condition. Existe-t-il encore des contre-exemples ? La réponse est oui, mais ce sont des nombres très grands. On ne connaît pas le premier, mais le plus petit actuellement connu est : $C_1 = 27\,664\,033 = 3\,037 \times 9\,109$.

Six valeurs pour une signature

Il est facile de faire un programme calculant u(n), mais il faut remarquer que les valeurs croissent très vite avec n. Pour les matheux, signalons qu'une approximation de u(n) est donnée par α^n, où $\alpha = 1{,}324718$ est la racine réelle de l'équation $x^3 = 1 + x$. Ainsi u(271 441) est un nombre ne possédant pas moins de 33 150 chiffres !

Il n'est donc pas question de calculer la valeur exacte de u(n) mais de reformuler le test. Nous utilisons alors les congruences. Rappelons que *a congru à b modulo c* (noté a≡b, mod c) signifie que b est le reste de la division entière de a par c, soit a = b + kc avec b<c et k entier positif. Par exemple, a divisible par c se note a≡0, mod c. D'où le test reformulé : n est "premier" si et seulement si $u(n) \equiv u(1)$, mod n et $u(-n) \equiv u(-1)$, mod n.

Appelons alors *signature* de n les six valeurs, modulo n : $u(-n-1)$, $u(-n)$, $u(-n+1)$, $u(n-1)$, $u(n)$ et $u(n+1)$. Le calcul d'une signature peut être fait simplement par le programme n° 1, et un nombre premier aura une signature de la forme : a, u(−1), b, c, u(1), d où a, b, c et d suivent aussi des règles.

Ainsi, on peut montrer que la signature d'un nombre premier appartient toujours à l'un des trois types T1, T2 ou T3 (voir tableaux 3 et 4).

On remarque assez vite que T1 est un type plus courant que T2, et que T2 est lui-même plus courant que T3. Asymptotiquement, on montre que les nombres premiers se répartissent entre les trois types dans les proportions de 1/2 pour T1, 1/3 pour T2, et 1/6 pour T3.

Grâce aux congruences, les dépassements de capacité sont évités, mais les temps de calcul restent trop élevés pour des n supérieurs à 10 000.

Heureusement, une propriété que nous appellerons « règle de doublement » et que nous ne démontrerons pas ici, vient à notre secours : $u(2n) = (u(n))^2 - 2\,u(-n)$. Elle est également vraie pour −n : $u(-2n) = (u(-n))^2 - 2\,u(n)$.

Cette règle permet de calculer rapi-

```
10 DIM S(6)
20 INPUT N:M=N
30 RESTORE 40:FOR I=1 TO 6:READ S(I):NEXT I
40 DATA 1,-1,3,3,0,2
50 FOR I=2 TO N
60 J=S(4)+S(5):IF J>=M THEN J=J-M
70 K=S(3)-S(1):IF K<0 THEN K=K+M
80 S(4)=S(5):S(5)=S(6):S(6)=J
90 S(3)=S(2):S(2)=S(1):S(1)=K
100 NEXT I
110 FOR I=1 TO 6:PRINT S(I);:NEXT I:PRINT
120 GOTO 20
```

Programme n° 1
Calcul de la signature
Auteur Christian Boyer

Il est possible de réaliser un programme de 1 Koctet en Assembleur du Z 80 permettant de calculer les signatures de nombres contenant jusqu'à 610 chiffres. Il serait beaucoup plus rapide puisque des nombres de 25 chiffres y seraient traités en 25 secondes !

Type	Signature
T1	A, u(−1), B, B, u(1), C
T2	u(1), u(−1), D, E, u(1), u(−1)
T3	u(−2), u(−1), u(0), u(0), u(1), u(2)
A, B, C et D, E ne peuvent pas prendre n'importe quelle valeur. Par exemple, on a toujours : • B ≢ u(0) et C ≡ 9B − 3A + 1, mod n pour une signature de type T1 • D ≢ E et E ≡ n − D − 3, mod n pour une signature de type T2	

Tableau 3
Trois types de signatures caractérisent les nombres premiers

n	Signature	Type
5	3, 4, 2, 2, 0, 0	T1
7	5, 6, 5, 5, 0, 3	T1
3	0, 2, 1, 2, 0, 2	T2
13	0, 12, 7, 3, 0, 12	T2
23	1, 22, 3, 3, 0, 2	T3
59	1, 58, 3, 3, 0,2	T3

Tableau 4
Les deux premières signatures de chaque type (n > 2)

DÉTECTEUR DE NOMBRES PREMIERS

dement la signature de n. Le programme n° 2 utilise deux vecteurs principaux, S et D, déclarés par DIM S(6), D(11). Le premier représente la signature, et le second sert à stocker temporairement les doublements : D(1) = u(−2k−2), D(2) = u(−2k−1), D(3) = u(−2k), D(4) = u(−2k+1), D(5) = u(−2k+2), D(6) n'est pas utilisé, D(7) = u(2k−2), D(8) = u(2k−1), D(9) = u(2k), D(10) = u(2k+1), D(11) = u(2k+2).

Supposons la signature de k connue avec S(1), S(2), ..., S(6). La règle de doublement permet de calculer D(1), D(3), D(5), D(7), D(9) et D(11) : D(1) = S²(1) − 2 S(6), D(3) = S²(2) − 2 S(5), etc.

Pour remplir les trous laissés dans D, on utilise simplement la récurrence u(n+3) = u(n) + u(n+1), soit : D(2) = D(5) − D(3), D(4) = D(2) + D(3). De même pour D(8) et D(10).

Maintenant que D est complet, on peut en déduire la signature de 2k : S(1 à 3) = D(2 à 4) et S(4 à 6) = D(8 à 10). Et la signature de 2k + 1 est : S(1 à 3) = D(1 à 3) et S(4 à 6) = D(9 à 11).

Après avoir écrit n en base 2, nous le parcourrons bit par bit de la gauche vers la droite. Puisque le premier bit significatif vaut toujours 1 (évidemment !!), nous initialiserons S avec la signature de k = 1. Du deuxième au dernier bit, nous recalculerons à chaque fois la nouvelle signature de k' = 2k + 1 (si le bit vaut 1), ou de k' = 2k (s'il vaut 0).

Le programme n° 2 se complique par rapport au n° 1, mais permet d'obtenir bien plus rapidement les signatures. Par exemple, avec un Basic de 16 chiffres, les signatures de nombres à 7 chiffres sont calculées en 10 secondes et celles des nombres à 14 chiffres en 2 minutes. Des nombres de cette taille étaient *impossibles* à calculer par le programme n° 1.

Deux autres suites présentent les mêmes propriétés que la suite de Perrin. Nous les appellerons S2 et S3 (S1 désignera la suite de Perrin). S2 est définie par : v(0) = 3, v(1) = 1, v(2) = 1 et v(n+3) = v(n) + v(n+2). Quant à S3, elle est définie par : w(0) = 3, w(1) = 1, w(2) = 3 et w(n+3) = w(n) + w(n+1) + w(n+2).

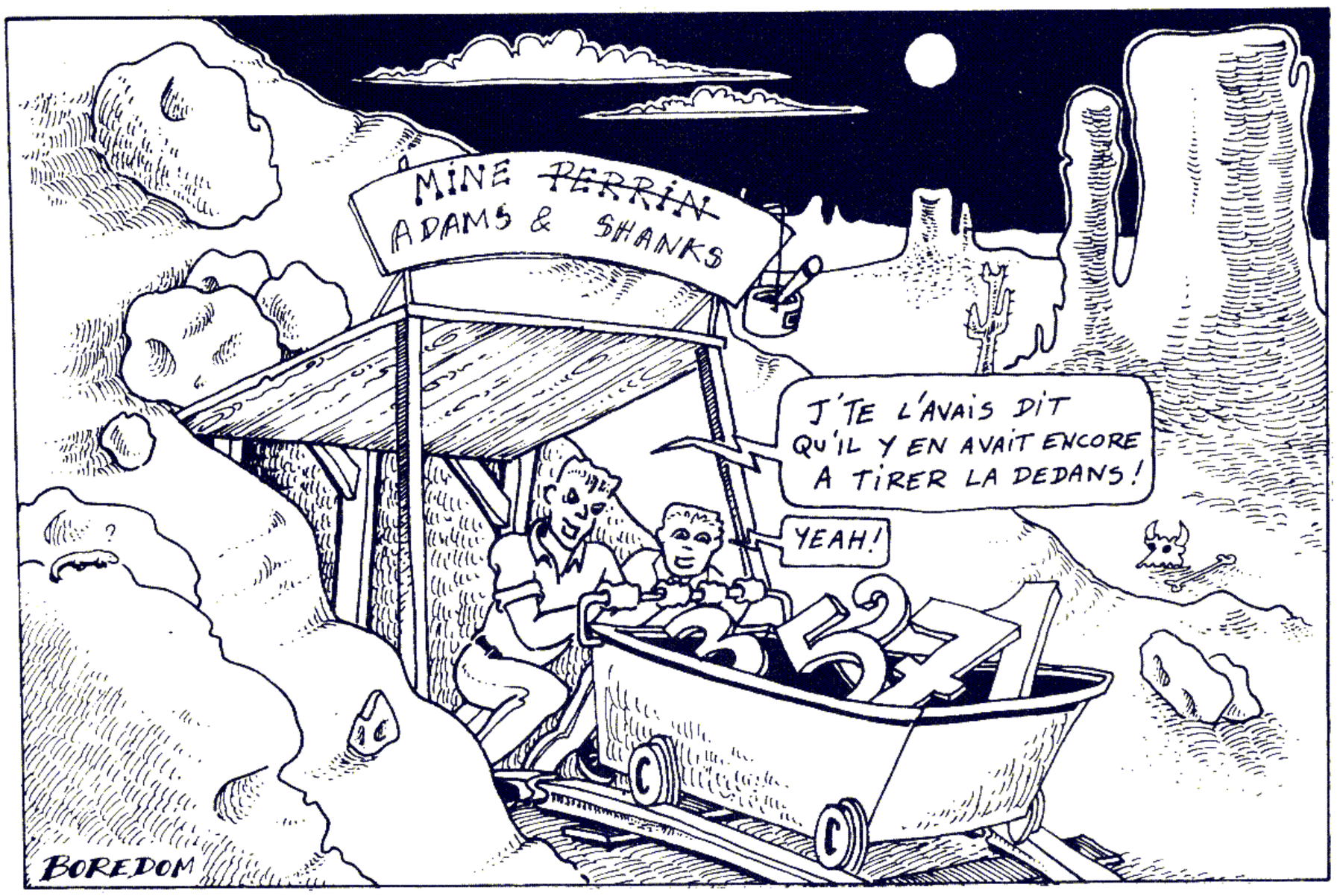
Pour ces deux suites, les signatures des nombres premiers sont de la même forme que pour la première étudiée : a, v(−1), b, c, v(1), d pour S2, avec v(−1) = 0 ; e, w(−1), f, g, w(1), h pour S3, avec w(−1) = −1.

Là aussi, les contre-exemples (1) sont extrêmement rares. Les plus petits connus sont : C_2 = 6 693 621 481 = 607×1 213×9 091 pour S2 et C_3 = 517 567 051 = 16 087×32 173 pour S3.

La précision est indispensable

Le calcul des signatures de ces deux suites est similaire à celui de la première, la règle de doublement étant toujours valable. Seuls les remplissages des trous ne se font pas de la même manière. Le programme n° 2 permet de calculer les signatures des trois suites. Lors du lancement de ce programme, introduisez correctement la précision de votre Basic, c'est-à-dire le nombre de chiffres significatifs utilisables sur votre machine : une surestimation de cette précision provoquerait des résultats faux.

On a vérifié que pour tous les n



(1) Un petit tuyau : les contre-exemples des suites ont été cherchés dans des familles particulières de nombres. Ainsi C_2 n'est autre qu'un nombre de Carmichael, alors que C_1 et C_3 sont respectivement de la forme k(3k − 2) et k(2k − 1).

```
3 DIM A(6),S(6),D(11)
5 PRINT:INPUT" Precision du BASIC  ? ";P
6 P=INT(P/2)-1
7 VP=VAL("1"+STRING$("0",P))
10 REM ==========================================================
15 REM ------ Introduction du nombre
20 PRINT:INPUT" Suite 1, 2, ou 3 ? ";S
21 IF (S<>1)*(S<>2)*(S<>3) THEN 20
25 INPUT" Nombre ? ";N
26 IF N<>INT(N) THEN PRINT" Nombre non entier":GOTO 25
27 IF N<2 THEN PRINT" Nombre inferieur a 2":GOTO 25
28 IF N>VP*VP THEN PRINT" Nombre trop grand pour ce BASIC":GOTO 25
30 M=N:REM M=Modulo
35 IF (S=3)*(M/2=INT(M/2)) THEN PRINT" Pair interdit (S3)":GOTO 25
39 REM ------ Initialisation signatures
40 ON S GOSUB 1000,2000,3000
41 FOR I=1 TO 6
42 READ A(I):IF A(I)<0 THEN A(I)=A(I)+M
43 S(I)=A(I):NEXT I
45 REM ------ Decomposition binaire
48 O=N:N$=""
50 O=O/2:IF O=INT(O) THEN N$="0"+N$:GOTO 50
60 O=INT(O):N$="1"+N$:IF O<>0 THEN 50
65 REM **********************
70 FOR L=2 TO LEN(N$)
75 REM ------ Regle de doublement
78 IF M>10*VP THEN GOSUB 5000:GOTO 120
80 FOR I=1 TO 6:J=2*I-1
90 D(J)=S(I)*S(I)-2*S(7-I)
100 D(J)=D(J)-M*INT(D(J)/M)
110 NEXT I
115 REM ------ Remplissage des trous
120 ON S GOSUB 1100,2100,3100
155 REM ------ Nouvelle signature
160 J=VAL(MID$(N$,L,1))
170 FOR I=1 TO 3:S(I)=D(I+1-J):NEXT I
180 FOR I=4 TO 6:S(I)=D(I+4+J):NEXT I
185 REM ------
190 NEXT L
195 REM **********************
200 PRINT" Signature =";:FOR I=1 TO 6:PRINT S(I);:NEXT I:PRINT
210 IF (S(5)<>A(5))+(S(2)<>A(2)) THEN PRINT" Compose":GOTO 10
220 PRINT" Probablement premier"
230 GOTO 10
990 REM ==========================================================
1000 REM ------ Suite 1
1010 RESTORE 1020:RETURN
1020 DATA 1,-1,3,3,0,2
1099 REM ------
1100 FOR I=1 TO 2:J=6*I-5
1110 D(J+1)=D(J+4)-D(J+2):IF D(J+1)<0 THEN D(J+1)=D(J+1)+M
1120 D(J+3)=D(J)+D(J+1):IF D(J+3)>=M THEN D(J+3)=D(J+3)-M
1130 NEXT I:RETURN
2000 REM ------ Suite 2
2010 RESTORE 2020:RETURN
2020 DATA -2,0,3,3,1,1
2099 REM ------
2100 FOR I=1 TO 2:J=6*I-5
2110 D(J+3)=D(J)+D(J+2):IF D(J+3)>=M THEN D(J+3)=D(J+3)-M
2120 D(J+1)=D(J+4)-D(J+3):IF D(J+1)<0 THEN D(J+1)=D(J+1)+M
2130 NEXT I:RETURN
3000 REM ------ Suite 3
3010 RESTORE 3020:RETURN
3020 DATA -1,-1,3,3,1,3
3099 REM ------
3100 FOR I=1 TO 2:J=6*I-5
3110 K=(D(J)+D(J+4))/2:IF K<>INT(K) THEN K=K+M/2:IF K>=M THEN K=K-M
3115 D(J+3)=K
3120 D(J+1)=D(J+3)-D(J)-D(J+2):D(J+1)=D(J+1)-M*INT(D(J+1)/M)
3130 NEXT I:RETURN
3990 REM ==========================================================
4995 REM ------ Regle de doublement en double precision
5000 FOR I=1 TO 6:J=2*I-1
5010 MA=S(I):MB=S(I):GOSUB 6000:D(J)=MC-2*S(7-I)
5020 D(J)=D(J)-M*INT(D(J)/M)
5030 NEXT I:RETURN
5099 REM ------ MC=MA*MB mod M    ( multipl. double prec. )
6000 A1=INT(MA/VP):A2=MA-VP*A1
6010 B1=INT(MB/VP):B2=MB-VP*B1
6020 MC=A1*B1:MC=MC-M*INT(MC/M)
6030 FOR K=1 TO P:MC=10*MC:MC=MC-M*INT(MC/M):NEXT K
6040 MC=MC+A1*B2+A2*B1:MC=MC-M*INT(MC/M)
6050 FOR K=1 TO P:MC=10*MC:MC=MC-M*INT(MC/M):NEXT K
6060 MC=MC+A2*B2:MC=MC-M*INT(MC/M)
6070 RETURN
9990 REM ==========================================================
9999 END
```

Programme n° 2
Test de primalité Perrin-Adams-Shanks
Auteur Christian Boyer

inférieurs à 360 001 aucun contre-exemple n'existe, ni dans S1, ni dans S2, ni dans S3. Donc, si le nombre à tester est inférieur à cette limite, le résultat de l'une quelconque des trois suites est suffisant, et la réponse est certaine et prouvée.

Au-delà, *on ne connaît aucun nombre résistant à deux suites* (et a fortiori aux trois). Par exemple, alors que C_1 est « probablement premier » dans S1, il est finalement prouvé composé par S2 (ainsi que par S3).

Un excellent test de primalité sera donc de soumettre le nombre aux trois suites. Si les trois tests répondent « probablement premier », vous pourrez considérer que votre nombre est réellement premier. Au premier contre-exemple, avertissez vite la rédaction !

Christian BOYER

Ce programme a été écrit dans un Basic le plus standard possible. Les seules différences éventuelles viennent des lignes 21 et 35 où les "*" sont des AND, alors qu'en ligne 210 le "+" est un OR. Quant à la ligne 7, elle est équivalente à VP = 10^P, mais évite les erreurs d'arrondi.

POUR ÊTRE TOUT A FAIT PRÉCIS...

LES ordinateurs calculent rapidement et fournissent des résultats précis. Cela dit, il leur arrive aussi « d'en rajouter ». A quoi bon dix chiffres significatifs quand cinq suffisent ? Et que dire des décimales superflues qui, de plus, sont fausses ? Il faut les éliminer.

L'excès en tout est un défaut. Trop de décimales dans un résultat, c'est trop. Lorsqu'il s'agit de francs ou de centimes, ou bien encore de tout résultat devant être exprimé avec un nombre de décimales constant, la solution est le plus souvent très facile.

Sur les machines de la famille *TI* (TI-58 ou 59), on dispose de l'instruction Fix. Par exemple, si le résultat doit être exprimé en francs, il suffit de faire Fix 0 (en mode calcul ou en mode programme) et le résultat est arrondi au franc le plus proche.

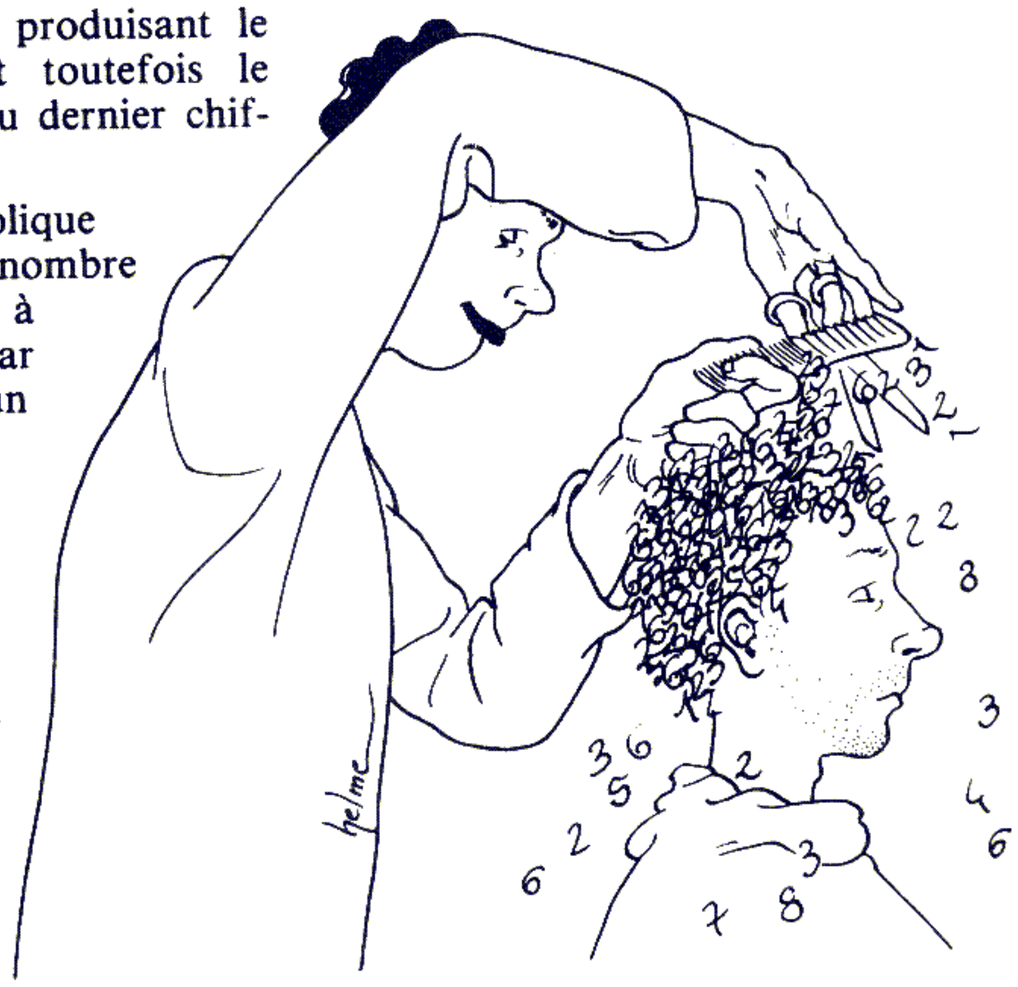
Sur les machines de la famille *Sharp* ou similaires, on dispose de l'instruction USING (uniquement en mode programme), plus lourde à manier que Fix, mais produisant le même effet, affichant toutefois le résultat sans arrondi du dernier chiffre.

La situation se complique quelque peu lorsque le nombre de chiffres significatifs à afficher est variable, par exemple dépendant d'un paramètre quelconque. Là encore, l'instruction Fix se révèle particulièrement maniable.

Ainsi, si vous voulez, sur une machine telle que la TI-58 ou 59, afficher la valeur de π avec un nombre de décimales égal à K, entrez K dans un registre quelconque, par exemple le registre 5. Si K = 4, faites 4 STO 05. Demandez alors π Fix Ind 5 =, et le résultat sera bien celui que l'on attendait : 3, 1416. En fait, la séquence Fix Ind 5 signifie : nombre de décimales contenu dans le registre 5.

Pour pousser le raisonnement un peu plus loin, imaginons que vous vouliez afficher π avec un nombre de décimales égal au chiffre des dizaines de l'âge du capitaine, ou bien à la vitesse du vent exprimée en nœuds. Entrez alors le petit programme suivant :

STO 5 π Fix Ind 5 R/S



Si le capitaine est âgé de 45 ans, vous devez taper 4 RST R/S. La réponse sera 3,1416. Si la vitesse du vent est de 7 nœuds, vous faites 7 RST R/S et vous obtenez 3,1415927.

Où va la virgule décimale ?

Dans le cas des machines dépourvues de l'instruction Fix, il n'existe pas d'équivalent à la gestion variable de la décimalisation que nous venons de décrire. En effet, l'instruction USING utilise des # dont la disposition exprime le formatage. Malheureusement, le nombre de # n'est pas programmable, et si ce nombre est variable, il faut recourir à un procédé que nous allons illustrer par un exemple : le calcul automatique de grandes factorielles.

Le programme calcule les factorielles des nombres inférieurs à 10^9. Il est établi pour PC-1211/PC-1 et comporte deux particularités :

- une précision générale poussée ;
- un affichage ne comportant que des chiffres significatifs exacts.

Le programme est divisé en deux parties :

- lignes 10 à 90, calcul direct des factorielles des nombres de 0 à 23 ;
- lignes 100 à 270, calcul par la formule de Stirling des factorielles des nombres supérieurs à 23 et inférieurs à 10^9.

Dans la deuxième partie, la formule utilisée est une version peu connue de la formule de Stirling, particulièrement adaptée au calcul logarithmique, qui s'écrit : $Z! = Z^Z \sqrt{2\pi Z}\ e^{-TZ}$ avec $T = 1 - 1/(12\ Z^2) + 1/(360\ Z^4) - 1/(1260\ Z^6) + \ldots$

Ecrivons Z en fonction de la mantisse D et de l'exposant C, soit $Z = D \times 10^C$, avec $1 \leqslant D < 10$ et C entier ; on obtient en logarithmes décimaux

(nous vous épargnons les calculs intermédiaires) : $\log Z! = C(Z + 0{,}5) + Z(\log D - \log e) + \log \sqrt{2\pi D} + U \log e$ où $U = 1/(12\,Z) - 1/(360\,Z^3) + 1/(1260\,Z^5)\ldots$ et $e = 2{,}718\ldots = 1/\ln 10$.

Le quatrième terme de U est $-1/(1680\,Z^7)$. Mais il serait illusoire de le retenir car, dans le champ d'application que nous étudions, sa prise en compte n'affecterait que la treizième décimale du résultat, qui sera toujours exprimé avec moins de 13 chiffres significatifs.

On remarquera que la formule utilisée permet de réduire la taille des nombres sur lesquels s'effectuent les calculs : le premier terme de la formule, $C(Z+0{,}5)$, n'intervient pas dans le calcul des décimales de log Z!, puisqu'il est soit entier, soit de la forme « ...,5 ». Par ailleurs, l'algorithme retenu sépare la partie entière de la partie décimale de log Z! (voir, dans l'analyse du programme, les lignes 120 à 150). La combinaison de ces deux astuces permet de gagner au moins une décimale dans l'expression finale du résultat.

Autopsie d'un résultat

Mais l'objet véritable de notre propos concerne les lignes 210 à 270 : l'expression du résultat, débarrassé de toutes les décimales parasites.

Le calcul des factorielles par la formule de Stirling est certes séduisant, mais les résultats obtenus doivent être interprétés avec beaucoup de précautions. En effet, on calcule Z! à partir de log Z!. Si, par exemple, Z = 5 000 000, on a : log Z! = 31 323 381,360 7..., et les chiffres qui suivent le 7 deviennent incertains. Pour calculer Z! en fonction de log Z!, on écrira :

Exposant de Z! = 31 323 381

Mantisse de Z! = antilogarithme de $0{,}3607\ldots = 10^{0{,}3607} = 2{,}294563072$

Puisque 0,3607 n'est connu qu'avec 4 décimales, il est logique de penser que l'on ne peut afficher $10^{0{,}3607}$ qu'avec 4 chiffres significatifs. En réalité, si l'on veut s'entourer de garanties raisonnables quant à l'exactitude du dernier chiffre, il est prudent de n'afficher le résultat qu'avec 3 chiffres significatifs, en l'occurrence 2,29.

Pour obtenir ce résultat, voici comment on procède : on range en

Calcul de grandes factorielles
Programme pour PC-1211
Auteur Pierre Ladislas Gedo


```
10:"L"INPUT " Z
   = ";Z
20:IF (Z>=0)*(Z
   <1E9)*(Z=INT
   Z)=0PRINT "
   VERIFIER Z":
   GOTO 10
30:IF Z>23GOTO
   100
40:A=1
50:FOR B=1TO Z
60:A=A*B
70:NEXT B
80:PRINT " Z! =
   ";A
90:GOTO 10
100:C=INT LOG Z
110:D=Z/10^C
120:E=C*Z+INT (C
   /2)
130:F=C/2-INT (C
   /2)
140:G=INT (Z*(
   LOG D-1/LN 1
   0))
150:H=Z*(LOG D-1
   /LN 10)-G
160:I=.5*LOG (2*
   Π*D)
170:J=(1/12/Z-1/
   360/Z^3+1/12
   60/Z^5)/LN 1
   0
180:K=INT (F+H+I
   +J)
190:L=E+G+K
200:M=F+H+I+J-K
210:N=10^M
220:O=9-INT LOG
   L
230:P=10^O
240:Q=(INT (N*P+
   .5))/P
250:IF Q=10LET Q
   =1:L=L+1
260:PRINT " Z! =
   ";Q;"E ";L
270:GOTO 10
```

Analyse du programme et liste des variables

Lignes	Variables	Commentaires
		Première partie : calcul direct
10	Z	Nombre dont on veut calculer la factorielle
20		Elimination des introductions non valables
30		Renvoi au calcul par la formule de Stirling
40	A	Variable dont la valeur initiale est 1 et la valeur finale Z!
50	B	Compteur de boucles
60		Valeurs successives de A (multiplications successives)
70		Incrémentation de B
80		Affichage de Z!
90		Réinitialisation
		Deuxième partie : calcul par la formule de Stirling
100	C	Exposant de Z
110	D	Mantisse de Z
120	E	Partie entière de $C(Z+0{,}5)$
130	F	Partie décimale de $C(Z+0{,}5)$
140	G	Partie entière de $Z(\log D - \log e)$
150	H	Partie décimale de $Z(\log D - \log e)$
160	I	$= \log \sqrt{2\pi D}$
170	J	$= U \log e$ (correction sur logarithme décimal de la formule de Stirling)
180	K	Retenue, partie entière de la somme des parties décimales de log Z!
190	L	Exposant de Z!
200	M	Logarithme de la mantisse de Z!
210	N	Mantisse brute de Z! (comportant les décimales parasites)
220	O	Nombre de décimales à afficher
230	P	Variable de calcul ($=10^O$)
240	Q	Mantisse nette de Z! à afficher (débarrassée des décimales parasites)
250		Elimination des mantisses à deux chiffres
260		Affichage
270		Réinitialisation

POUR ÊTRE PRÉCIS

mémoire N le résultat brut de la mantisse de Z! (N = 2,294694863) et en mémoire O le nombre de décimales du résultat à afficher, 2 dans notre exemple. La ligne 240 décale la virgule de deux positions vers la droite (229,4694863), ajoute 0,5 au résultat pour obtenir l'arrondi au chiffre le plus proche (229,9694863), en retient la partie entière (229) et redécale la virgule de deux positions vers la gauche pour obtenir et afficher 2,29.

J'attire tout particulièrement votre attention sur le mécanisme de cette ligne 240, qui élimine littéralement toutes les décimales parasites.

L'affichage indique donc :
Z! = 2,29 E 31 323 381

Si l'on avait pris comme exemple Z = 50 000 000 au lieu de 5 000 000, il est clair que la mantisse aurait dû être exprimée avec 2 chiffres significatifs au lieu de 3, l'exposant comportant alors 9 chiffres au lieu de 8. On voit donc qu'en règle générale, si le nombre de chiffres de l'exposant augmente d'une unité, on perd un chiffre significatif sur la mantisse.

C'est le tribut à payer pour l'exploitation logarithmique des fonctions exponentielles, comme la formule de Stirling. Ainsi, la somme des chiffres significatifs de la mantisse et de l'exposant sera toujours égale à 11, ce qui est en accord avec la précision de la machine (12 chiffres significatifs pour les calculs internes, avec incertitude quant au dernier).

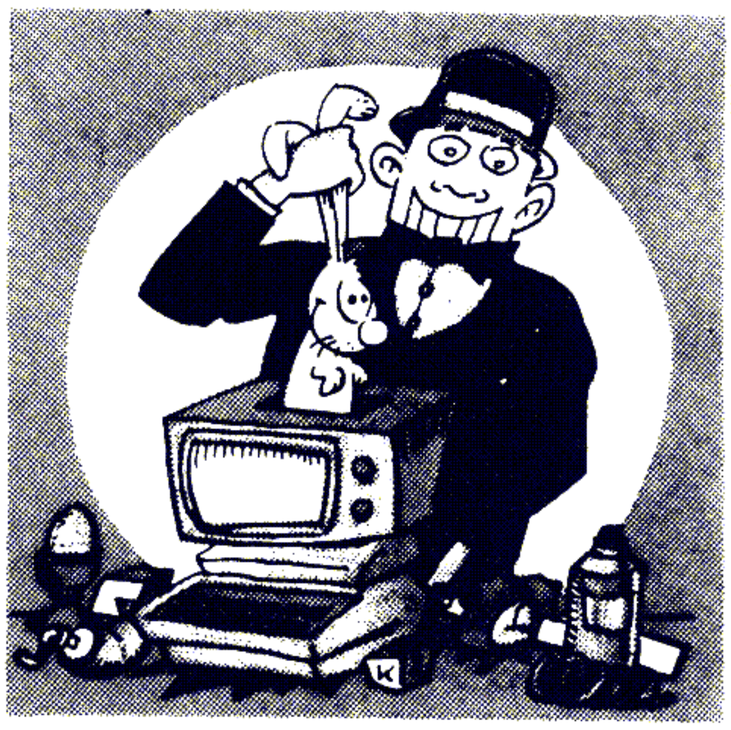
Si vous voulez exprimer les résultats avec un chiffre de plus, remplacez 9 par 10 à la ligne 220, mais ce sera à vos risques et périls, car vous multiplierez par 10 la probabilité de voir le dernier chiffre affiché de la mantisse entaché d'une erreur d'une unité.

Le mode d'emploi du programme

Un mot sur le mode d'emploi du programme (il peut s'utiliser avec ou sans imprimante) : initialiser par SHFT L, introduire la valeur de Z et presser sur ENTER. Réinitialiser par une autre pression sur ENTER.

Le temps d'exécution est de 2 à 10 secondes pour Z compris entre 0 et 23, et de l'ordre de 10 secondes pour Z supérieur à 23.

Pierre Ladislas GEDO



LA BOÎTE A MALICES...

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

LIST

FX-702P

UN NOM POUR CHAQUE LISTE

Rien ne ressemble plus à une liste de programme qu'une autre liste de programme. Surtout lorsque ces listes sont imprimées avec la même imprimante, sur un même papier.

Heureusement, il existe un moyen de les différencier : leur donner un nom. Et avec le FX-702P et son imprimante, la FP-10, c'est facile : le nom peut apparaître sous la forme d'un titre, sans avoir été programmé. Ce résultat s'obtient en protégeant le programme par un mot de passe et, alors seulement, en le listant. Par exemple, un programme d'Othello dont le titre sera OTHELLO, justement, sera protégé par PASS "OTHELLO" (EXE) avant d'être listé par LIST "OTHELLO" (EXE).

Et ainsi, le titre apparaît sur le papier aluminisé, en toutes lettres !

Fabrice ROGISTER

ZX SPECTRUM

CANAUX ET « MICRODRIVES »

Ainsi qu'il est expliqué dans la notice sur l'interface ZX1 et le microdrive, les données du Spectrum peuvent circuler à destination et en provenance de divers éléments d'un système informatique : clavier, écran, imprimante, microdrive, synthétiseur vocal, générateur de sons...

Ces informations passent par l'intermédiaire de canaux. Sur un Spectrum de base, quatre canaux sont préétablis, et les axes de circulation des données, appelés « voies », sont tels que :

• canal « R », voie#0, correspond aux sorties vers la partie inférieure de

l'écran, celle réservée au compte rendu ;

• canal « K », voie#1, correspond aux entrées à partir du clavier et à l'affichage de la saisie vers la partie inférieure de l'écran ;

• canal « S », voie#2, correspond à la sortie vers la partie supérieure de l'écran, celle réservée à l'utilisateur ;

• canal « P », voie#3, correspond à la sortie vers l'imprimante.

Ainsi PRINT#2 revient au même que d'omettre la spécification #2 (cela équivaut à PRINT seul). LPRINT#3 revient au même que LPRINT seul. Mais PRINT#3, malgré un PRINT et non un LPRINT enverra les données vers l'imprimante et non pas sur l'écran.

A l'adresse pointée par la variable système CHANS, se trouvent 20 octets concernant ces quatre canaux préétablis, soit 5 octets par canal :

• les deux premiers donnent l'adresse de la routine en mémoire morte de la sortie sur ce canal ;

• les deux suivants donnent l'adresse de la routine en mémoire morte de l'entrée sur ce canal ;

• le cinquième et dernier octet correspond au code d'une lettre du canal concerné K, S, R ou P.

Ceci peut nous intéresser, car si l'on détourne une sortie vers une autre adresse, on peut alors faire toutes sortes de manipulations sur les données à transmettre sur le canal choisi.

C'est ce que montre le programme suivant, qui remplace l'adresse de la sortie sur le canal S par une autre adresse. A cette adresse, nous avons au préalable implanté une petite routine qui ne fait qu'ajouter 32 aux codes de certains caractères.

```
10 CLEAR 31999
20 LET Z=5+PEEK 23631+256*
   PEEK 23632
30 POKE Z,0 : POKE Z+1, 125
40 FOR N=0 TO 8 : READ A :
   POKE 32000+N, A : NEXT N
50 DATA 254, 32, 56, 2, 203, 239,
   195, 244, 9
60 PRINT "aBcDeFgH"
70 LIST
```

Résultat surprenant, n'est-ce pas ? Si vous avez la petite imprimante *Sinclair,* vous pouvez détourner la sortie en modifiant les lignes :

```
20 LET Z=15+PEEK 23631+256*
   PEEK 23632
```

et

```
70 LLIST
```

Maintenant que les microdrives et l'interface ZX1 sont disponibles en France, votre Spectrum peut s'ouvrir sur d'autres canaux et d'autres voies.

Mais, pour s'ouvrir sur le monde extérieur, le Spectrum a besoin d'autres variables système. En réservant de l'espace pour ces nouvelles variables, l'interface ZX1 effectue quelque chose qui ne convient vraiment pas à certains logiciels déjà écrits pour le Spectrum. Il réalise en fait une translation de toute la zone mémoire, à partir de la zone programme, et cela vers le haut. L'adresse du début de la zone programme n'est donc plus fixe et égale à 23755 : elle est devenue variable. Or, les concepteurs de logiciels ont gardé la vieille habitude, héritée du ZX 81, de nicher le langage-machine dans une REM constituant la première ligne du programme.

Ainsi, un excellent logiciel, PASCAL HP4T HISOFT ne peut tout simplement pas être fourni sous la forme d'un microdrive. Mais pourquoi se priver de son implantation sur celui-ci à partir d'une cassette ? On le chargerait ainsi en moins de 20 secondes, contre 2 mn 30 quand il est lu à partir de la cassette.

La solution existe, et elle est assez simple : il faut qu'après chargement à partir du microdrive, le début de la zone programme soit à nouveau à l'adresse 23755 et donc redescendre de 58 octets (ceux que s'approprie l'interface ZX1).

Voilà comment y parvenir. Entrez le programme suivant :

```
10 CLEAR 31999
20 FOR N=0 TO 19 : READ A :
   POKE 23296+N, A : NEXT N
30 DATA 33, 0, 125, 17, 22, 96, 1,
   113, 82, 237, 176, 33, 240, 92, 17,
   182, 92, 195, 229, 25
40 LOAD*"m" ; 1 ; "pascal 1"
   CODE 32000
50 LOAD*"m" ; 1 ; "pascal 2"
```

et sauvegardez-le sur microdrive en demandant SAVE*"m" ; 1 ; "HP4S" LINE 10 et vérifiez par VERIFY*"m" ; 1 ; "HP4S"

Prenez alors la cassette PASCAL ; faites (en mode commande) CLEAR 31999, puis MERGE"". Quand le programme est en mémoire, arrêtez la cassette et rajoutez une ligne numérotée 14 :

```
14 CLEAR 24576 : RANDOMIZE
   USR 23296 : GOTO 3
```

Chargez ensuite en mémoire le code du compilateur Pascal à partir de l'adresse 32000 en faisant : LOAD""CODE 32000. Cette opération terminée, faites alors en séquence : SAVE*"m" ; 1 ; "pascal 1" CODE 32000, 21105 puis, VERIFY*"m" ; 1 ; "pascal 1" CODE puis, SAVE*"m" ; 1 ; "pascal 2" LINE 14 et, VERIFY*"m" ; 1 ; "pascal 2".

Et voilà, le tour est joué. Vous avez maintenant sur la cartouche du microdrive :

• un programme permettant de charger les autres programmes du Pascal venant de la cartouche, ainsi qu'une petite routine en langage-machine qui rendra silencieuse l'interface ZX1 ;

• un bloc d'octets correspondant au compilateur ;

• un programme enfin (légèrement modifié) correspondant à l'interface Basic/Pascal.

Désormais, en faisant : LOAD* "m" ; 1 ; "HP4S", le Pascal se chargera en un temps record...

Benoît THONNART

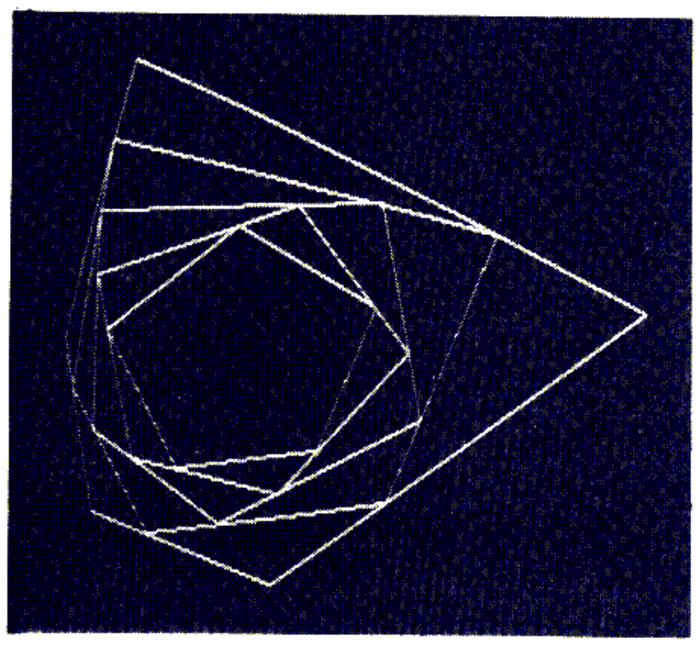
TI-57

TOURNE, TOURNE

Pour lancer la TI-57 dans une ronde sans fin, entrez la suite d'instructions suivante : LRN Exc SST Lbl 1 . R/S RST . LRN RST R/S R/S LRN 1 LRN RST R/S SST 5 LRN RST R/S R/S.
Essayez donc d'interrompre la ronde en pressant sur R/S (mais oui, c'est possible) et quand vous y serez parvenu, listez le programme, par curiosité... et essayez de la relancer. Heureusement, vous avez toujours la possibilité d'éteindre la machine. Et de recommencer !

Philippe MORALES et Christophe TAVERNER

BASIC



UN MOTIF GÉOMÉTRIQUE A LA MODE

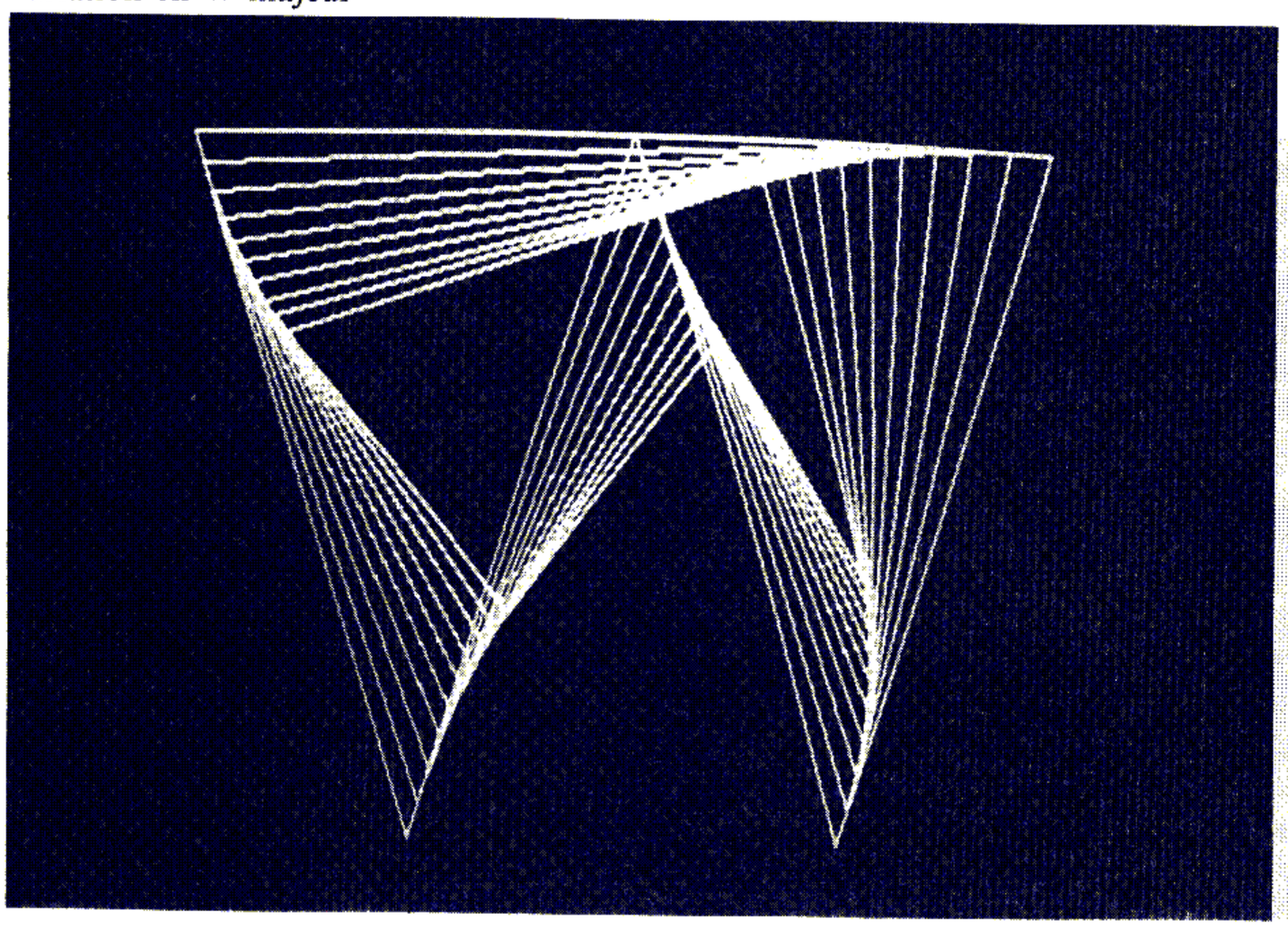
On voit de plus en plus, à la télévision ou sur les écrans d'ordinateurs, des figures formées de polygones superposés légèrement déformés les uns par rapport aux autres, donnant parfois une impression de relief très remarquable. Leur élaboration est une application simple de la notion de barycentre. Le programme ci-dessous permet de choisir le nombre n de sommets des polygones, le nombre total p de polygones à construire, et un « coefficient réducteur » k. On le choisit généralement entre 0 et 1, et voisin de l'un de ces deux nombres (par exemple : 0,1 ou 0,9).

Si le polygone initial est ABCD..., on construit à l'aide du nombre k un polygone A'B'C'D'... en définissant A' comme barycentre de A et de B munis respectivement des masses (1-k) et k, puis B' comme barycentre de B et de C et ainsi de suite (cela signifie simplement que le vecteur $\overrightarrow{AB'}$ est égal au produit du vecteur $\overrightarrow{AB}$ par k). Choisir k négatif ou supérieur à 1 n'a plus d'effet réducteur, mais agrandit au contraire le polygone : il faut s'attendre alors à dépasser éventuellement les limites de l'écran. Le programme est écrit pour TRS 80 en haute résolution ; mais il s'adapte immédiatement, par exemple, à la table traçante CE-150 du PC-1500 ou à tout autre ordinateur ayant des possibilités graphiques.

Motifs géométriques
Programme en Basic
Auteur André Warusfel


```
10 CLS:CLR
20 INPUT "Nombre de sommets ";N
30 INPUT "Nombre de polygones ";P
40 INPUT "Coefficient réducteur ";K
50 DIM X(N): DIM Y(N)
60 FOR I=0 TO N-1
70     PRINT "X(";I+1;")= ";: INPUT X(I)
80     PRINT "Y(";I+1;")= ";: INPUT Y(I)
90 NEXT: SCREEN 0
100 FOR J=1 TO P
110     X(N)=X(0): Y(N)=Y(0)
120     FOR I=0 TO N-1
130         LINE (X(I),Y(I))-(X(I+1),Y(I+1))
140     NEXT I
150     IF J=P GOTO 200
160     FOR I=0 TO N-1
170         X(I)=(1-K)*X(I)+K*X(I+1)
180         Y(I)=(1-K)*Y(I)+K*Y(I+1)
190     NEXT I
200 NEXT J
210 END
```

Variation en W majeur



André WARUSFEL

FX-602P

ELLE DÉFILE ENCORE

La FX-602P ne se laisse pas toujours interrompre dans son travail. Voyez comment elle réagit au lancement d'un tout petit programme : LBL 1 = PAUSE GOTO 1. Elle tourne et ne peut être arrêtée que par HLT. Même des appuis répétés sur AC sont vains. Pendant

qu'elle "travaille", essayez de frapper 5× ou 3+, etc. Puis, essayez 5++ ou 3××, etc. Et si le défilé est trop rapide, vous pourrez ajouter un PAUSE dans le programme.

Ce peut être un moyen d'apprendre ses tables.

Thierry DEMOY

PET

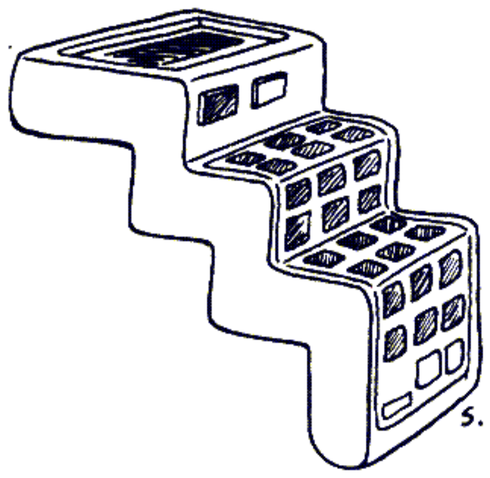
ÇA FAIT DÉSORDRE

Les historiens du PET/CBM savent reconnaître une machine dès la mise sous tension. En effet, le PET 2001, équipé du Basic 1, que ce soit le 8 Ko au clavier de calculatrice ou le modèle « business » à 32 Ko affichait :

*** COMMODORE BASIC ***

La série 3000, équipée du Basic 2, affichait à l'ouverture :

COMMODORE BASIC



-PETIT ORDINATEUR AYANT L'ESPRIT D'ESCALIER

Et puis est venue la glorieuse série 4000, avec le Basic 4.0. Ce qui est advenu du Basic 3, personne ne l'a jamais su. Toujours est-il que les acquéreurs de la nouvelle série, ou ceux qui avaient changé les MEMs voyaient avec orgueil s'afficher un nouveau message :

*** COMMODORE BASIC 4.0 ***

Vous êtes dans ce cas ? Alors frappez donc en mode direct la ligne suivante :

```
FOR I = 54360 TO 54384 : PRINT CHR$(PEEK(I)) ; : NEXT I
```

Dites, vous ne trouvez pas que ça fait désordre d'avoir laissé traîner ça ?

François J. BAYARD

ATARI

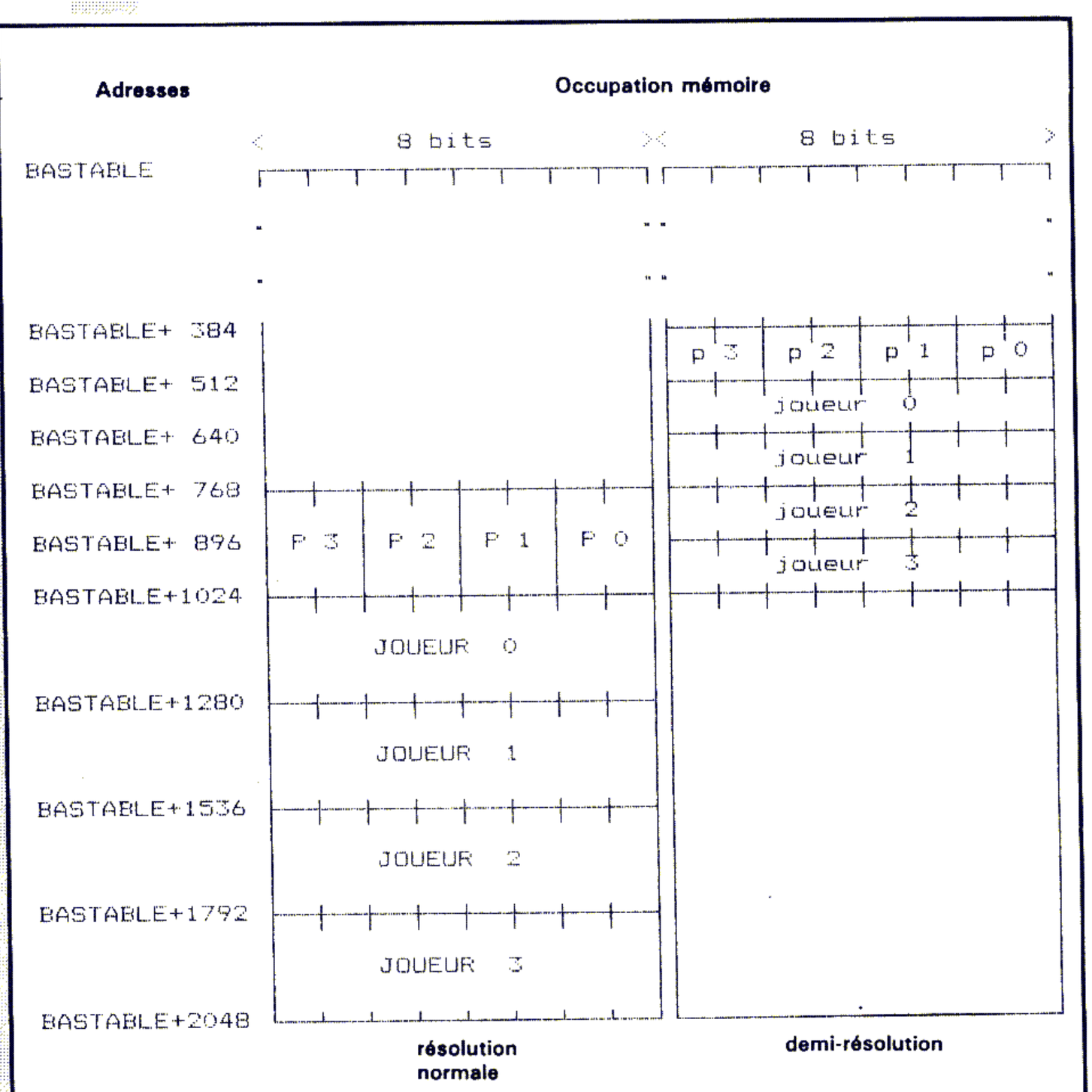
JOUEURS ET PROJECTILES

Les dessins de joueurs et de projectiles sont le leitmotiv des jeux vidéos d'action : des joueurs sur l'écran tiennent leurs adversaires sous le feu de leurs canons ; le circuit spécialisé CTIA est un manipulateur de formes, il peut se charger rapidement de les représenter dans leur mobilité. Le MPG (Missile Player Graphics) est capable d'animer simultanément quatre formes dites « joueurs » que nous désignerons de J0 à J3 ainsi que quatre formes projectiles, de P0 à P3.

Etudions la génération de formes avec l'exemple du programme (page suivante) ; après RUN, l'écran passe en mode graphique 2, réservé au texte ; pourtant un avion de couleur rouge occupe le centre de l'image. Le MPG est indépendant du mode graphique en cours, il superpose les formes sur l'écran prédéfini par l'ordre GRAPHICS.

En examinant le contenu de la mémoire à partir de l'adresse BASTABLE +1024, sur une hauteur de 256 octets (la boucle en 240 initialise cette zone à 0, puis de 300 à 340 y introduit les DATAs), on constate que la représentation binaire de la mémoire est analogue au dessin observé à l'écran : les 1 correspondent aux points allumés et

Table d'occupation mémoire du Missile Player Graphics



Avant toute utilisation :

```
POKE 53277,3:AUTORISATION DU MPG
POKE 559,RESOLUTION(62 = normal;46 = demi-résolution)
POKE 54279,NUMERO DE PAGE MEMOIRE DE DEBUT DE TABLE
```

les 0 aux points éteints. Ainsi le MPG agit-il en projetant une tranche de mémoire à l'écran.

Nous pouvons maintenant modifier quelques paramètres.

Le paramètre ABCISSE:'POKE 53248,ABCISSE'. En tapant 120 ABCISSE = 100, on pousse l'avion vers la gauche. Ajoutons les lignes :

1040 V = −1
1050 ABCISSE = ABCISSE + V : IF ABCISSE < 0 THEN ABCISSE = 225
1060 GOTO 1030

L'avion s'anime d'un mouvement régulier de droite à gauche ; le test en

1050 empêche la tranche en mouvement de quitter l'écran.

Le paramètre COULEUR :'POKE 704,COULEUR'. Nous retrouvons la palette habituelle des teintes de 0 à 15 ; bien qu'une forme soit invisible sur fond de même couleur, il est possible d'établir des priorités.

Le paramètre TAILLE:'POKE 53256,TAILLE'. La tranche superposée à l'écran peut avoir différentes largeurs, le système se chargeant de doubler ou de quadrupler l'image sans la modifier.

Le MPG peut animer trois autres formes selon les mêmes principes, les adresses de la table et des paramètres étant seules modifiées.

Récapitulons les étapes nécessaires à

Génération de formes
Programme pour Atari
Auteur Alain Lavenir


```
10 GRAPHICS 2
100 REM INITIALISATION DES PARAMETRES DU M.P.G.
110 HAUT = 90
120 ABCISSE  = 120
130 COULEUR = 3
140 TAILLE = 0
200 REM RECHERCHE D'UNE ZONE MEMOIRE POUR LOGER LA TABLE
210 HM = PEEK(742) : REM PAGE DU HAUT DE LA MEMOIRE
UTILISATEUR
220 M = (8 * INT (HM/8)) - 8 : REM ON PREND LE MULTIPLE
DE 8 JUSTE EN DESSOUS ET ON RESERVE 8 PAGES
230 BASTABLE = M * 256
240 FOR J = 0 TO 2048 : POKE BASTABLE + J ,0 : NEXT J
300 REM ENTREE DE LA FORME
310 FOR L = 0 TO 6
320 READ FORME
330 POKE BASTABLE + 1024 + HAUT + L , FORME
340 NEXT L
400 REM INITIALISATION DU M.P.G.
410 POKE 54279,M : REM NUMERO DE PÂGE MEMOIRE DE DEBUT
DE TABLE
420 POKE 53277,3 : REM VALIDATION DU M.P.G.
430 POKE 559,62 : REM DEFINITION NORMALE
1000 REM UTILISATION DU M.P.G.
1010 POKE 53256 , TAILLE
1020 POKE 704 , COULEUR
1030 POKE 53248 , ABCISSE
10000 DATA 6 , 24 , 97 , 255 , 97 , 24 , 6
```

l'utilisation du Missile Player Graphics :

- dessiner des formes dans une grille de 8 colonnes sur 256 lignes, en grisant les cases (voir schéma *dessin des formes*) ; établir la liste des DATAs en commençant par la ligne la plus haute, pour chaque joueur ;
- choisir une zone mémoire pour loger la table des formes ; le MPG utilise 8 pages mémoire de 256 octets à partir d'une valeur qui doit être un multiple de 8. Il est prudent de se placer sous le haut de la mémoire utilisateur et d'initialiser cette zone avec des 0 (soit BASTABLE l'adresse du bas de cette table) ;
- entrer la table des formes (par 256 octets) ;
- valider le MPG en lui fournissant l'adresse de départ de la table en 54279, 3 en 53277, et 62 en 559 qui est un code permettant au MPG d'analyser les formes plus finement ;
- manipuler les formes en modifiant les paramètres.

Le cas des quatre projectiles est peu différent : leurs tranches de projection larges de 2 bits sur 256 sont adjacentes ; on construit la table de DATAs de la même manière, en traitant les quatre missiles simultanément ; la couleur d'un projectile est celle du joueur qui porte son numéro ; les tailles sont fixées globalement par la mémoire 53260 (0 = simple, 1 = double, 3 = quadruple). La position du p-ième projectile est donnée par PEEK 53252 + p (c'est la loi de l'Ouest).

Enfin, pour agir sur les formes, il faut faire :

POKE 704 + J,COULEUR JOUEUR J
POKE 53248 + J,ABCISSE JOUEUR J
POKE 53252 + P,ABCISSE PROJECTILE P
POKE 53256 + J,TAILLE JOUEUR J(0 ou 1 ou 3)
POKE 53260 ,TAILLE DE TOUS LES MISSILES (0 ou 1 ou 3)

Le dessin des formes

		128	64	32	16	8	4	2	1
60 =	32+16+8+4			■	■	■	■		
90 =	64+16+8+2		■		■	■		■	
255 =	128+64+32+16+8+4+2+1	■	■	■	■	■	■	■	■
255 =	128+64+32+16+8+4+2+1	■	■	■	■	■	■	■	■
66 =	64+2		■					■	
66 =	64+2		■					■	

Alain LAVENIR

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...

7

Minimum et maximum

En Basic et en Fortran, les fonctions MIN(A,B) et MAX(A,B) retournent respectivement la plus petite et la plus grande des deux valeurs représentées par A et B. En Pascal, ces fonctions ne sont pas disponibles en standard. Leur programmation ne pose bien entendu aucun problème. On peut par exemple écrire :

```
fonction MIN (A, B : entier) : entier ;

debut
     si A<B alors MIN:=A sinon MIN:=B
fin ;

fonction MAX (A, B : entier) : entier ;

debut
     si A<B alors MAX:=B sinon MAX:=A
fin ;
```

Ces deux fonctions, que vous n'aurez aucun mal à comprendre, font une comparaison entre A et B (si A<B). Le problème est d'éliminer ce test. Il s'agit donc de trouver deux expressions arithmétiques dépendant de A et de B, la première ayant pour résultat le minimum entre A et B, et la seconde le maximum entre A et B. Vous pouvez utiliser toutes les opérations ou fonctions disponibles sur votre ordinateur, à l'exclusion bien entendu de MIN et de MAX, ainsi que des opérateurs de relation (<, >, =, etc.).

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

SOLUTIONS DU NUMÉRO PRÉCÉDENT

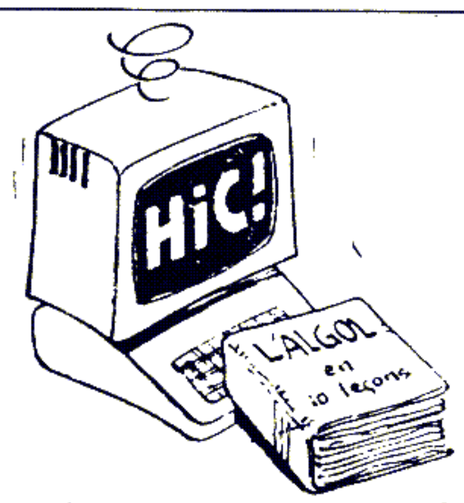
Les solutions présentées ici répondent aux problèmes posés dans le précédent numéro de LIST. Ce ne sont pas forçément les meilleures !

8

L'affectation, cette incomprise

Chaque langage informatique a sa particularité. Si nous prenons l'instruction la plus simple qui soit, l'affectation de la valeur d'une variable à une autre, il existe presque autant de possibilités que de langages. Ainsi :

en Pascal ou en Algol, on écrit	:	A:=B
en Basic	:	LET A=B
en Cobol	:	MOVE B TO A
en Fortran	:	A=B
en APL ou en LSE	:	A←B
et même, dans d'autres langages	:	B→A



Voilà qui ne simplifie pas tellement la compréhension des programmes, surtout si, comme dans les deux derniers cas, les notations sont aussi divergentes. Mais au fait, savez-vous précisément ce qu'est une affectation ? Ainsi, dans une instruction aussi simple (et inutile) que A:=A, le premier A ne signifie pas la même chose que le second. Alors, que représente chacun de ces deux A ?

9

Sigles et abréviations

La liste ci-dessous est constituée de huit sigles ou abréviations rencontrés en informatique. Pouvez-vous indiquer, pour chacun d'entre eux, la signification de l'abréviation utilisée ?

- BIT (zéro ou un)
- CP/M (système d'exploitation)
- ASCII (codage des caractères)
- BS (caractère de contrôle)
- SOS (circuits intégrés)
- CRT (écrans d'ordinateurs)
- ALGOL (langage de programmation)
- SSII (sociétés de logiciel)

➤➤ 4

Court et structuré

Le programme proposé affiche les puissances successives de 2. Il peut être récrit beaucoup plus élégamment de la façon suivante :

```
10 for X=0 to 17
20 print X,2^X
30 next X
40 end
```

C'est la boucle *for...next* qui rend le programme plus court et plus structuré.

➤➤ 5

Pair ou impair

Le résultat de l'expression $(1-(-1)^i)/2$ est 0 si i est pair et 1 si i est impair. L'expression arithmétique plus simple qui permet d'obtenir le même résultat est celle-ci : i−2*ENT (i/2) où ENT(x) est la fonction qui retourne la partie entière de x.

➤➤ 6

L'inventaire incomplet

L'article disparu a par exemple fait l'objet d'une commande de 1 unité pendant un mois. La valeur de Commande était donc égale à 1. Au cours de ce même mois, un seul acheteur s'est présenté pour cet article, mais, au lieu de l'acheter, il a demandé que celui-ci soit repris et échangé contre un autre. La valeur de Vendu a donc été mise à −1, puisqu'il s'agissait d'une reprise. Le résultat de l'opération Commande + Livre + Vendu est donc égal à 0, puisque les deux valeurs s'annulent mutuellement. Ainsi, l'article n'a pas été imprimé dans la liste de contrôle.

Le programme d'édition peut être corrigé de deux manières différentes. La première consiste non pas à tester le total des entrées ou sorties de stock, mais à vérifier individuellement ces valeurs :

```
REPETER
    Lire_article_suivant
    Si Commande <>0 OU Livre <>0 OU Vendu <>0 ALORS
        Imprimer_article
    FIN_SI
JUSQU_A Fin_de_fichier
```

L'autre méthode consiste à ne pas tenir compte des valeurs qui peuvent éventuellement s'annuler, en ne retenant que les valeurs absolues. Il est alors possible d'écrire le programme d'édition ainsi :

```
REPETER
    Lire_article_suivant
    SI ABS(Commande) + ABS(Livre) + ABS(Vendu) <>0 ALORS
        Imprimer_article
    FIN_SI
JUSQU_A Fin_de_fichier
```

■

Chez Duriez : 15 micros portatifs + 9 domestiques

Super branchés (pas d'angoisse !)

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

Avez-vous vu les **300 prix**

Charter© Duriez ?

valables jusqu'au 20 octobre

A U 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.

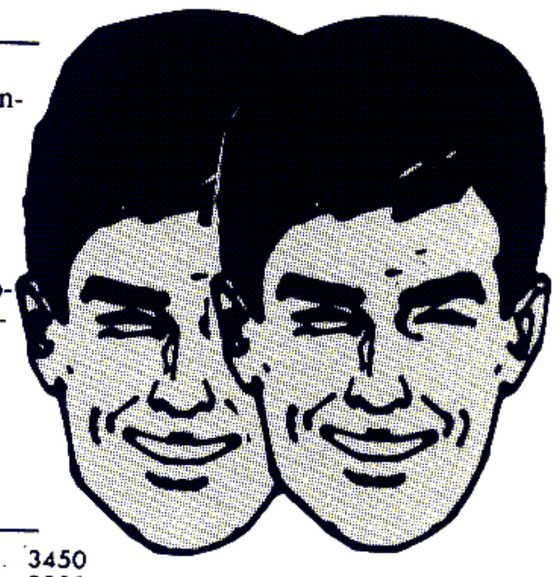
Ce banc d'essai est gratuit en magasin, ou envoyé par poste contre 3 timbres à 2,10 Frs.

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.



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
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

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
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

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

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 41 CV	**2190**
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

Média Conseil

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.

☐ 1 Catalogue général Duriez (Calculatrices, Machines à écrire, Répondeurs, Photocopieurs, Classeurs, Dicteurs, Papeterie, etc...) contre 3 timbres à 2F10.

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

☐ 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.

Si changement de prix, je serai avisé avant expédition.

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

.....

.....

.....

Date et Signature .....

.....

LI *octobre 84*

LE LIVRE QU'IL VOUS FAUT EST DANS CE CATALOGUE GRATUIT

PSI au SICOB
du 19 au 28 septembre
CNIT
SICOB Boutique.
Stand n° 30

Plus de 170 titres disponibles pour :

- apprendre à programmer sur Commodore 64, Sinclair, MO 5, TO 7, Apple, Alice, Oric...
- découvrir de nouveaux langages Basic, C, Fortran, Logo, Assembleur, Pascal...
- acquérir les techniques et méthodes de programmation
- mieux utiliser votre ordinateur : des applications pratiques, professionnelles ou pédagogiques et les disquettes d'accompagnement sur Apple II.

CHEZ VOTRE LIBRAIRE OU EN BOUTIQUE SPÉCIALISÉE

DEMANDEZ LE CATALOGUE GRATUIT Bon à découper, à retourner à :

En France	En Belgique	En Suisse	Au Canada	Au Maroc
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 1** **1701 Fribourg - SUISSE** **Tél. : (037) 23.18.28** **(CCP 17.56 84)**	SCE Inc. 65, avenue Hillside Montreal (Westmount) Québec H3Z1W1 - CANADA Tél. : (514) 935.13.14	SMER DIFFUSION 3, rue Ghazza Rabat MAROC Tél. : (7) 237.25

☐ Je possède un ordinateur du type _____ ☐ Je ne possède pas d'ordinateur

☐ Je désire recevoir régulièrement le catalogue P.S.I.; voici les thèmes que je souhaite voir abordés dans les livres P.S.I. _____

_____

NOM _____ Prénom _____

Profession _____

Rue _____ N° _____

Code postal _____ Ville _____

LC 10

AGRAPH