LIGNES

n°7

LE JOURNAL DES AMATEURS DE PROGRAMMATION

ISSN 0761-9936

MARS 1985

AU PROGRAMME

- Basic a vingt ans, une histoire méconnue
- 40 machines ont passé nos dix tests
- Logo, Pascal et... la boîte à malices

GROS PLAN SUR QUATRE LOGICIELS

- Animation graphique sur TO7 et MO5
- Edi-Logo pour Apple II
- D-Basic pour Dai
- Moniteur 1.0 pour Oric-1 et Atmos

A L'ESSAI

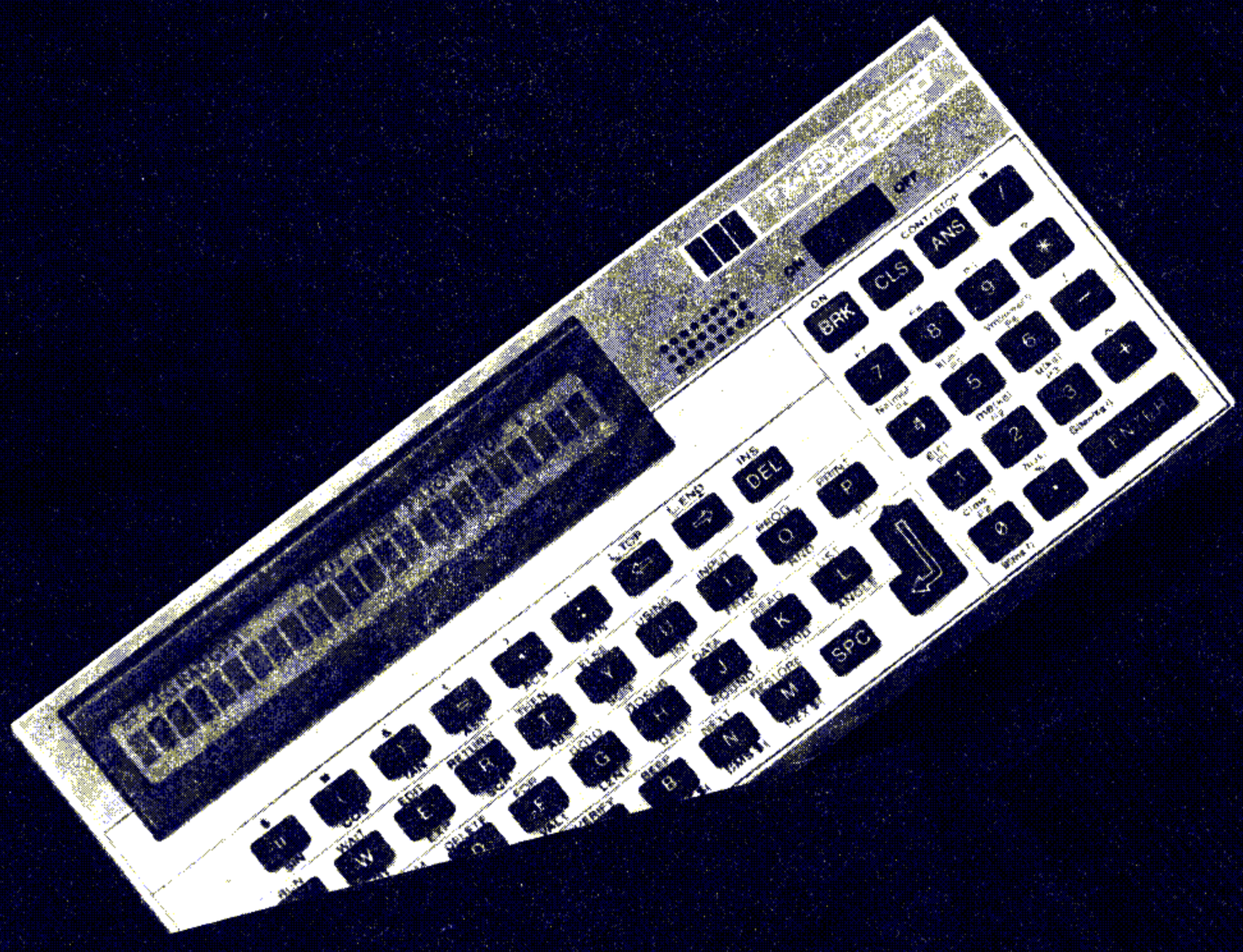
- Deux modestes
- Exelbasic de l'EXL 100
- Super Basic du QL

M 2712-7-20 F

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

20 F

CASIO



LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 7 - MARS 1985

SOMMAIRE

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

Ont collaboré à ce numéro : *Olivier Arbey, Michel Arditti, Pierre Barnouin, François J. Bayard, Robin Bois, Christian Boyer, Nicole Bréaud-Pouliguen, Marc Brochard, Roger Brousmiche, Laura Campagnet, Thierry Chamoret, Sylvain Coron, Raymonde Coudert, Robert Daguesse, Jacques Deconchat, Vincent Di Sanzo, Yves Drillet, Augustin Garcia, Paul Gardan, Florence Gautier-Louette, Pierre Ladislas Gedo, Alain Goubault de Brugière, Max Hagenburger, Jean-Christophe Krust, Jacques Labidurie, Jean-Pierre Lalevée, Xavier de La Tullaye, Alain Lavenir, Thierry Lévy-Abégnoli, Alain Mariatte, Adrien Meaudre, Pierrick Moigneau, Yvon Pérès, Yul Pham Duy, André Reeb-Gruber, Edouard Rencker, Pierre Ricard, Denis Sebbag, Jean Thiberge, André Warusfel.*

Illustrations : *Philippe Burel, Chimulus, Frapar, Bernard Helme, Renée Koch, Fabien Lacaf, Sylvain Lemaire, Alain Mangin, Pasty, Alain Prigent, Nicolas Spinga, Eric Théocharidès.*

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

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

Directeur de la publication : Jean-Luc Verhoye

LIST est une publication du groupe tests

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

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

TURBO PASCAL 625 F HT

avec manuel en français

TURBO PASCAL EST UN SYSTÈME COMPLET

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

DE NOMBREUSES EXTENSIONS

vous permettent d'utiliser à fond votre ordinateur.

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

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

LA COMPILATION SE FAIT EN MÉMOIRE

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

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

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

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

L'ÉDITEUR INTÉGRÉ EST CONFIGURABLE.

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

REVUES...

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

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

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

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

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

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

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

ENVOYEZ-MOI DE SUITE :

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

DISQUES ☐ 3 1/2" ☐ 51/4 ☐ 8"

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

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

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

NOM

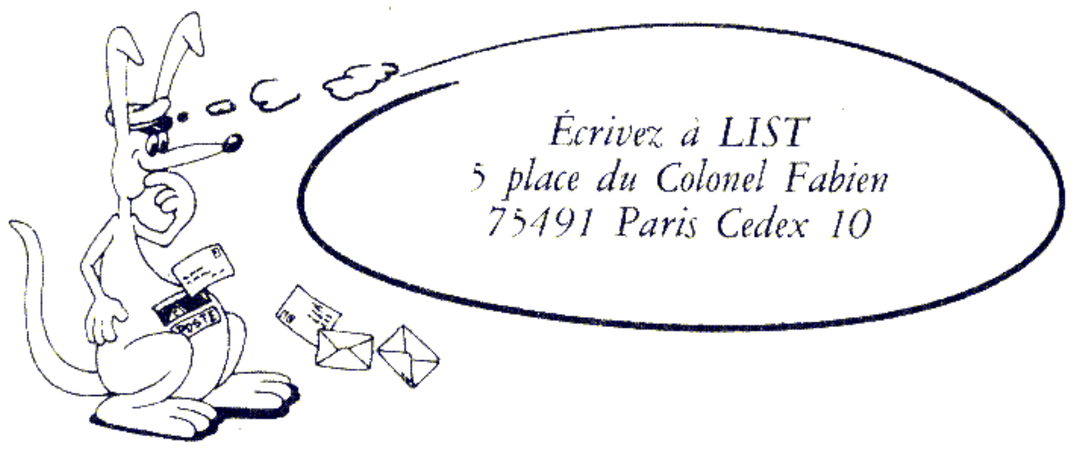
ADRESSE

TÉL.

SIGNATURE :

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

A VOS CLAVIERS



SAUVER SES ŒUVRES

JE viens d'acquérir un Amstrad, moniteur couleur, et toute la famille se met — avec quelques difficultés — au Basic.

Nous venons notamment d'entrer le programme Amstrad-Graphe proposé dans votre numéro 6. Formidable. Les enfants se disputent le clavier. Nous l'avons également sauvegardé en changeant le mode 1 en mode 0, et nous réalisons des dessins en neuf couleurs.

Malheureusement, nous avons un problème pour la réutilisation d'un dessin « archivé ». La procédure que nous employons est la suivante :

- à la fin du dessin, deux fois ESC pour avoir READY ;
- SAVE « nom du dessin », B,&C000,&4000

Lorsque nous voulons réutiliser le dessin, nous frappons LOAD « nom du dessin » et, à l'écran, le dessin apparaît mais d'une part, il n'a plus les couleurs d'origine et d'autre part, les messages BREAK, READY, SAVE,... s'inscrivent aussi. Il n'y a plus moyen alors d'apporter une modification à ce dessin, ni d'envisager son utilisation avec un texte, par exemple.

Le guide livré avec l'Amstrad ne nous a apporté aucune réponse. Nous attendons vos « lumières » avec impatience.

Claudie MORVAN
91 Massy

La solution consiste à programmer la sauvegarde ou le chargement du dessin, c'est-à-dire à introduire les instructions SAVE et LOAD dans des lignes de programmes. Ainsi, en allongeant le programme initial avec les lignes suivantes :

```
385 IF C$="S" THEN SAVE "DES",B,&C000,&4000
386 IF C$="Z" THEN LOAD "DES"
```

votre dessin sera sauvegardé, ou chargé selon votre choix, et pourra être utilisé à nouveau.

INDEX DES ANNONCEURS

LA GUERRE DES TROIS

L'ARTICLE intitulé « La guerre des trois » et publié dans *LIST* 6 page 20, m'a beaucoup plu : c'est une bonne idée de nous présenter des systèmes curieux ou des réalisations bizarres.

Au début de l'article vous écrivez que les nombres de −4 à +4 sont : − −, −0, −1, 0, 1, 1−, 10, 11. Ne serait-ce pas plutôt : − −, −0, −1, −, 0, 1, 1−, 10, 11 ?

Dans ce cas, il est facile d'écrire une valeur négative à partir de la colonne de la table de conversion décimal-ternaire en remplaçant les − par des 1 et les 1 par des −. Par exemple, les décimaux de 1 à 5 sont, en ternaire : 1, 1−, 10, 11, 1− −. Les nombres négatifs de −1 à −5 deviennent donc : −, −1, −0, − −, −11.

Est-ce juste ? Enfin, j'aurais préféré Trit à Tit...

Alain Scerri
94 Arcueil

A tous les stades de la fabrication de notre journal, nous passons beaucoup de temps à chasser les erreurs et les fautes. Et malgré cela, il nous en échappe toujours. Là, un petit tiret (ô combien important) s'est en effet soustrait à nos yeux vigilants. Vous l'avez rattrapé, merci.

Quant à votre idée de conversion décimal-ternaire pour les nombres négatifs, elle est non seulement rapide, mais encore exacte. En effet, le tiret (−) correspond à −1, opposé de 1. Le programme de conversion n'est donc nécessaire que pour les valeurs positives, les valeurs négatives en étant déduites grâce au procédé que vous avez remarqué.

BESOIN D'ESPACE

UNE petite question à propos de mon Canon X-07 : devant chaque instruction LPRINT, je suis obligé de laisser un espace supplémentaire, sinon il se produit une erreur de syntaxe.

Est-ce normal, docteur ? Est-ce moi qui ai mal lu le mode d'emploi ? Est-ce une anomalie de mon Canon ?

Rémi DUCROT
51 Reims

Bizarre, cet espace dont a besoin votre instruction LPRINT. Cela provient peut-être de la série de votre machine. En tout cas, sur les X-07 que nous avons examinés, il n'en va pas de même : LPRINT se comporte comme les autres instructions.

A VOS CLAVIERS

UN DRÔLE D'ACCENT

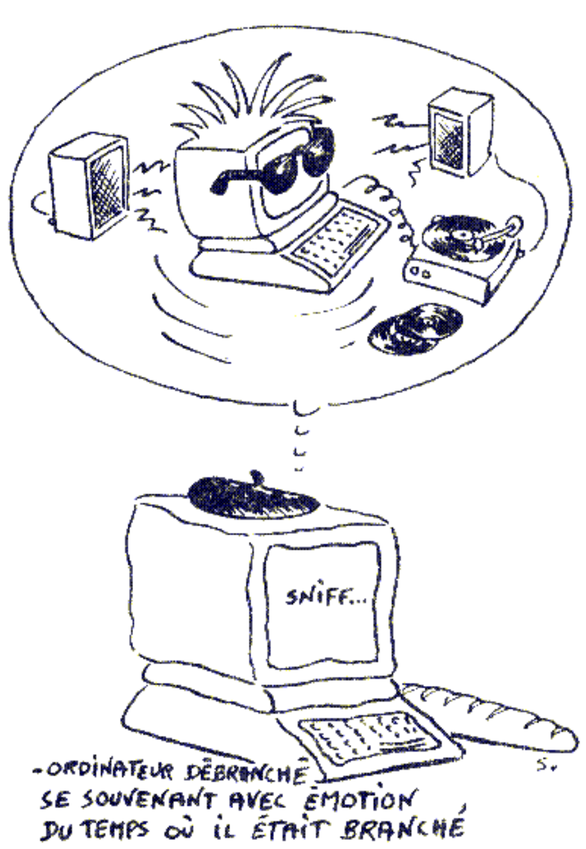
ETANT très, très novices dans la programmation et le maniement de l'Amstrad, nous n'avons pu, mon fils et moi, trouver le signe « accent circonflexe » (^) se trouvant aux lignes 470, 490 et 550 d'un programme publié dans *LIST* 6 page 62 (Amstrad-graphe). Nous enrageons de ne pouvoir faire tourner ce programme au demeurant très intéressant.

Auriez-vous la gentillesse de nous indiquer comment obtenir ce petit signe qui nous arrête.

Pierre di Costanzo
78 Mantes la Jolie

Le petit accent circonflexe qui apparaît dans les listes de programme, placé entre des nombres ou des variables, représente le signe mathématique de l'élévation à la puissance.

Sur le papier, c'est un accent circonflexe. Sur le clavier de l'Amstrad, le signe à taper est la flêche vers le haut (↑) mais attention, celle qui se trouve sous le signe de la livre sterling (£). Ainsi, ce qui s'écrit mathématiquement 3^9 s'écrit 3↑9 sur l'Amstrad et devient 3^9 sur la liste imprimée.



DIVERTISSEMENT EN LOGIQUE

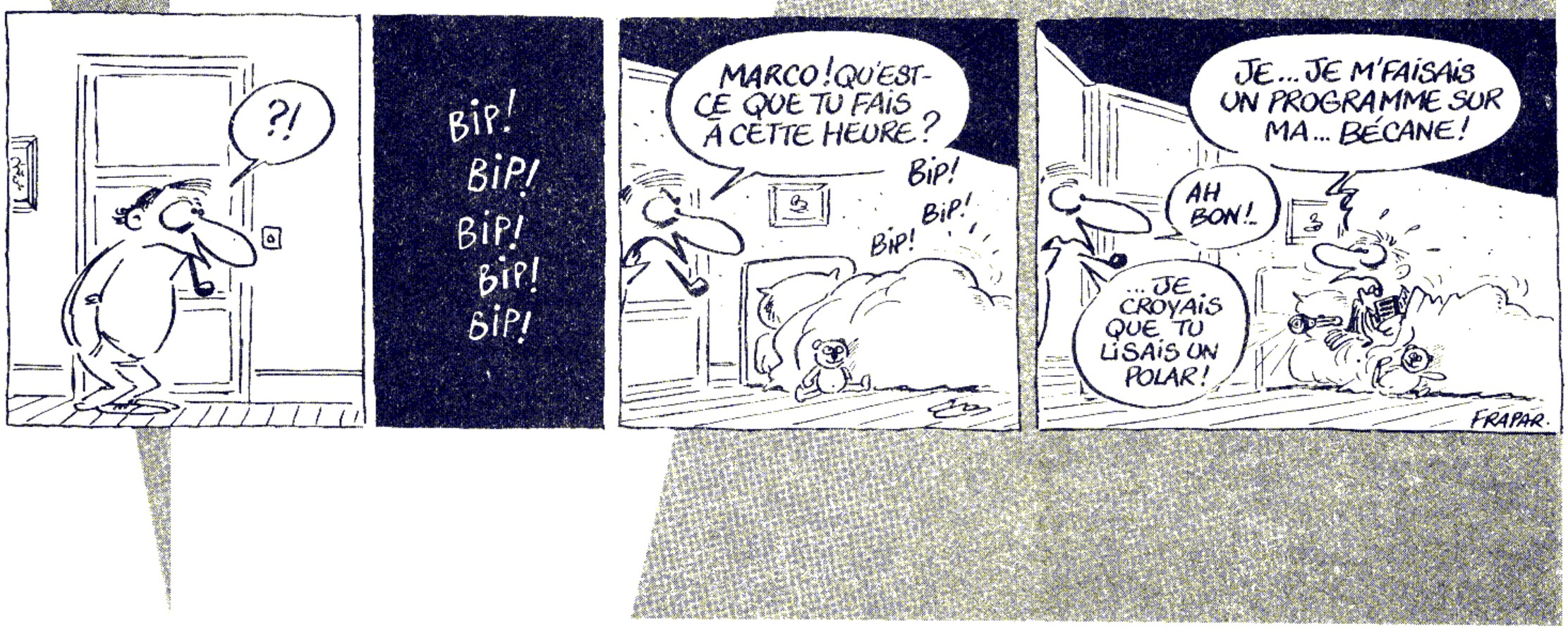
VOICI un petit divertissement qui fait appel à tout votre sens logique et fureteur. Il s'agit de résoudre un problème qui met en jeu cinq machines et cinq périphériques. Les machines sont : un Commodore 64, un IBM 3, un Macintosh, un Léanord Sil'z, un Thomson TO7. Les périphériques sont : un disque, une cassette, un clavier, un télétype et un écran graphique. (Ne pas tenir compte de la comptabilité avec les machines).

A midi, à l'heure du café, cinq personnages discutent : un analyste, un chercheur, un enseignant, un ingénieur, un programmeur. Celui qui a des moustaches est chauve. Voici le document qu'ils commentent :

Si l'histoire que je conte est un peu embrouillée,
Sachez que cinq machines sont ici concernées.
Le programmeur, dont l'IBM est la voisine,
Préfère utiliser la première machine.
Le châtain sur cassette, le chercheur au clavier,
Ont pour l'enseignant roux des notes à calculer.
Au milieu, avec Forth, on fait de la musique.
Même si un voisin peut tout stocker sur disque,
Le nouveau TO7, si tu as des moustaches,
Est pour toi la machine à simplifier les tâches.
Devant le Macintosh, l'ingénieur prend sa pipe.
Un voisin, dit le brun, est sur le télétype.
Tandis que l'analyste en PL/1 programme,
Avec humour le blond pour le Cobol se pâme.
Le Commodore, placé à droite du Sil'z
Un programme en Basic exécute sans aide.

Moralité
Pour qui l'écran graphique présente-t-il un texte ?
Qui utilise Logo dans un pareil contexte ?

Un lecteur



LA GAZETTE DE LIST

UNE CASSETTE

Images
pour Oric Atmos
Distribué par Infogrames
Prix : environ 160 FF

IMAGES est une cassette de création graphique conçue pour le micro-ordinateur Oric Atmos. Un emballage attrayant, avec une séduisante reproduction informatisée de la Joconde (elle sourit bien, merci). On se voit déjà dans la peau de Léonard. La notice a déjà un peu refroidi notre enthousiasme, en annonçant dans les **nouvelles** fonctions disponibles CIRCLE et FILL. A y bien regarder, il me semble que j'avais déjà tout cela sur mon Oric. En fait, l'intérêt du logiciel est, je cite, « qu'il permet d'exploiter facilement les possibilités de l'Oric, sans maîtriser les techniques de programmation ». On comprend mieux. De plus, les images produites seront compactées (gain de 40 % environ) et pourront être reprises dans un programme Basic ou LM. Voilà qui ne manque pas de devenir intéressant.

La technique utilisée pour la création de l'image est assez banale : on retrouve le principe du télécran (un point est mobile sur l'écran, avec ou sans trace). Une portion de l'écran est affectée à des informations pratiques, notamment un zoom sur l'environnement (les 8 points autour) du point mobile. C'est pratique. Une option *écriture* permet d'afficher du texte. *Duplication* permet de reproduire une portion d'image saisie dans un rectangle. *Pattern* offre la possibilité de modifier bit à bit l'octet utilisé pour des tracés de droites (DRAW) ou encore de cercles (CIRCLE).

Position renseigne sur la position et l'état du curseur, et *Fill* permet de remplir une zone de couleur. Le procédé retenu semble convenir pour réaliser des dessins géométriques, mais il est peu adapté aux dessins artistiques. Il n'est, par ailleurs, pas possible de faire une copie de l'écran. Le logiciel sera intéressant pour des applications bien précises, grâce notamment aux possibilités de compactage et à la facilité d'utilisation des images obtenues dans un programme Basic.

JD ■

Canari

MICROLOGUE qui titre « L'informatique décontractée » a annoncé une nouvelle version de son générateur de programmes *Canari*.

Baptisé tout simplement Canari II, le programme peut être implanté sur tout ordinateur supportant MS-DOS, Prologue ou CP/M 86. Six aménagements nouveaux ont été développés, parmi lesquels une nouvelle présentation des écrans (permettant notamment l'inversion vidéo), une touche « magique » afin d'obtenir l'environnement complet (variables et fichiers) du point du programme dans lequel on se trouve, et la possibilité d'introduire des mots de passe quel que soit le programme.

Le tout pour 1 500 FF ht.

Micrologue : Maison de l'informatique
18 rue Trémolet
42700 FIRMINY
Tél. (77) 89 11 11 ■

Pour mieux voyager, l'Apple IIc maigrit

DEPUIS le 1er février, **Apple** propose aux possesseurs d'Apple IIc un écran plat à cristaux liquides (poids : 1,1 kg).

Commercialisé au prix de 7 000 FF ttc, le nouvel écran peut afficher des textes « plein écran » sur 80 caractères par ligne et 24 lignes par écran, ainsi que des graphiques d'une résolution de 560 points horizontaux sur 192 points verticaux.

Facile à installer, l'écran à cristaux liquides se branche directement sur le panneau arrière de l'Apple IIc et ne nécessite aucune alimentation électrique extérieure. Le nouveau moniteur mesure 13,8 cm de large sur 29 cm de long et 4 cm d'épaisseur. ■

Commodore change de cap

APRÈS une sorte de valse-hésitation, **Commodore** vient enfin de dévoiler sa stratégie pour 1985. Une précision toutefois, rien n'est encore très officiel et il faudra pour l'instant se contenter des rares déclarations de Procep pour éclairer nos lanternes. Résultat : ni le *Commodore 16* ni le nouveau *plus 4* ne seraient en définitive commercialisés en Europe. En revanche, l'accent serait mis... sur le *Commodore 64* (vous avez bien lu : le Commodore 64), ainsi que sur le « petit » dernier : le *Commodore 128*. Disposant de 128 Ko (comme son nom l'indique), le nouveau Commodore sera compatible avec les logiciels développés pour le 64. De la même manière, le 128 dispose de 16 couleurs, 8 lutins et un générateur de sons à trois voies.

Seules innovations, la mémoire est extensible à 512 Ko, le clavier de 92 touches a été entièrement remodelé, et le lecteur de disquette adéquat (toujours non intégré) devrait être beaucoup plus rapide que l'ancien modèle (2 000 cps contre 320 pour son ancêtre). Dernière précision, à côté du 8 500 de base qui équipe le Commodore 128, un co-processeur Z 80 devrait permettre d'utiliser la plupart des programmes CP/M. Le prix de ce 64 « musclé » : environ 4 000 FF.

Mais chat échaudé craint l'eau froide, et l'on restera prudent concernant la date de commercialisation « réelle » du 128. D'ici que l'on apprenne qu'il est réservé à d'autres pays que la France... ■

Programmer dans un fauteuil

GARDER ses aises, c'est ce que propose la société **Micro Laser** grâce à une rallonge Péritel de trois mètres. Dite universelle, la *Périlonge* permettra à la fois d'utiliser son ordinateur dans un fauteuil et de ranger son magnétoscope à une distance raisonnable de la télévision. La Périlonge coûte 300 FF et est entièrement française.

Micro Laser
23 rue du Languedoc
31000 Toulouse
Tél. (61) 55 19 77 ■

Concours de logiciels

VOUS résidez en Vendée, Charente-Maritime ou dans les Deux-Sèvres. Vous pouvez donc participer au concours de logiciels grand public organisé par le *Crédit Mutuel* des villes de La Roche-sur-Yon, Rochefort-sur-Mer et Saintes.

Les candidats pourront soumettre au jury deux types de programmes : logiciels de jeux ou logiciels domestiques et programmes d'apprentissage.

Des prix de 10 000 F environ récompenseront les vainqueurs. Clôture du concours le 15 avril. Pour tous renseignements :

Caisse Fédérale du Crédit Mutuel Océan
34 rue Léandre-Merlet
85001 La Roche Sur Yon
Tél. (51) 05 44 44 ■

LA GAZETTE DE LIST

Atari drague les créateurs de logiciels européens

FAISANT suite à l'annonce fracassante des ordinateurs *ST*, alias « Jackintosh » (des Macintosh à la sauce Atari), Sigmund Hartman, Président de la division logiciel d'**Atari**, a déclaré lors de son passage éclair à Paris qu'il *fera largement appel à des sociétés de logiciels européennes*. But de l'opération : fournir en masse des logiciels pour la série des ST dont le « clou » est certainement le 130, une sorte de Macintosh avec la couleur, annoncé à 399 dollars, soit moins de 4 000 FF.

Français, Anglais, Allemands ont donc été sollicités pour développer à la vitesse grand V les logiciels de demain. Il est vrai que sans eux, Macintosh peut dormir tranquille même si Sigmund Hartman a précisé qu'Atari comptait vendre quelque deux millions de ST en 1985.

De la vitesse de la réponse des éditeurs et développeurs de logiciels européens dépendra donc une partie du succès des nouveaux Atari. Avis aux fûtés qui ont des idées. ■

L'EAO à domicile

NE désespérez plus, vous pouvez désormais concevoir des programmes d'enseignement sur IBM/PC. La société **Alpha Education** vient d'annoncer la disponibilité d'un système de création de cours et d'enseignement assisté par ordinateur *EAO Tencore* conçu pour les ordinateurs IBM/PC.

Langage de programmation puissant, Tencore permet au professeur d'imaginer sa propre méthode pédagogique. La création de cours EAO par Tencore nécessite un IBM/PC (ou compatible) disposant de 192 Ko de mémoire, de deux lecteurs de disquette et d'une carte graphique couleur.

Alpha-Education
195 rue de Bercy
Tour Gamma A
75012 Paris
Tél. (1) 347 67 13 ou
(1) 347 67 14 ■

UN PETIT TOUR CHEZ LE LIBRAIRE

Mon Commodore 64
Manuel de l'utilisateur
John Heilborn
et Ran Talbott
Traduit par Jacques Richard
Editions Mc Graw Hill
Paris, 1984
Broché, 428 pages
Prix : 150 FF

Programme interne du Commodore 1541
Milton Bathurst
Editions Data Cap
Distribué par PSI
Lagny, 1984
Broché, 250 pages
Prix : 130 FF

La conduite du MO 5
Jean-Yves Astier
et Olivier Kauf
Editions Eyrolles
Paris, 1985
Broché, 138 pages
Prix : 85 FF

Apprivoiser TO 7, TO 7/70 et MO 5
Pour apprendre à converser en Basic
Bernard Dupuy
et Bernard Violet
Editions Foucher
Paris, 1984
Broché, 176 pages
Prix : 59,50 FF

Faites vos jeux sur PB-700
Jean-Marc Nasr
et François Manchon
Editions Eyrolles
Paris, 1985
Broché, 108 pages
Prix : 90 FF

Les écrans du ZX Spectrum
Livre 1 : Premiers pas en programmation
Livre 2 : Pour en savoir plus en programmation
Ian Graham
Editions Hachette
Paris, 1984
Brochés, 64 pages
Prix : 90 FF chaque livre

Basic MSX
1 - Méthodes pratiques
Jacques Boisgontier
Editions PSI
Lagny, 1985
Broché, 216 pages
Prix : 120 FF

Trucs et astuces du Commodore 64
Angerhausen
L. Englisch
Gerits
Editions Data Becker
Distribué par les Editions Radio et Micro Application
Paris, 1985
Broché, 222 pages
Prix : 149 FF

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

MSX, Initiation au Basic
Rodnay Zaks
Editions Sybex
Paris, 1985
Broché, 238 pages
Prix : 98 FF

Le livre du MSX
Daniel Martin
Editions BCM
Distribué par PSI
Lagny, 1984
Broché, 204 pages
Prix : 110 FF

ProDos sur Apple IIe et IIc
Francis Verscheure
Editions PSI
Lagny, 1985
Broché, 100 pages
Prix : 85 FF

Les écrans du Commodore 64
Premiers pas en programmation
Livre 1
Phil Cornes
Editions Hachette Informatique
Paris, 1984
Broché, 64 pages
Prix : 90 FF

Le Basic sur le bout des doigts avec Commodore 64
Herbert Peckham
Traduit par Léon Collet
Editions Mc Graw Hill
Paris, 1984
Reliure spirale, 378 pages
Prix : 135 FF

Le Logo
Boris Allan
Editions Belin
Collection Modulo
Paris, 1984
Broché, 116 pages
Prix : 95 FF

Microprocesseurs 8086-8088 Architecture et programmation coprocesseur de calcul 8087
Jean-Michel Trio
Editions Eyrolles
Paris, 1984
Broché, 232 pages
Prix : 130 FF

Manuel de l'assembleur 6809 du MO 5
Michel Weissgerber
Editions Cédic/Nathan
Totek International
Paris, 1984
Broché, 192 pages
Prix : 145 FF

Carte Minitel pour Canon X-07

DANS *LIST* 6 (page 16), nous signalions l'existence d'une carte Minitel pour Canon X-07 intitulée Electronic Cable Interface (ECI). Nous imputions ce sigle à la société qui la commercialise. Cette société s'appelle en fait : Electronic Service Computer (ESC). Elle se trouve 67 Cours Lieutaud, 13006 Marseille et, pour tout renseignement, on peut la joindre au (91) 42 99 42.

Jouer avec son micro-ordinateur
Jeux éducatifs - simulations
Jeux d'aventure - Wargames
Olivier Chazoule
Editions Marabout
Informatique
Paris, 1984
Broché, 190 pages
Format poche
Prix : 24 FF

Méthode générale d'analyse d'une application informatique
Tome 2
Xavier Castellani
Editions Masson
Paris, 1984
Broché, 244 pages
Prix : 156 FF

250 questions sur la micro-informatique
Ilya Virgatchik
Editions Marabout
Paris, 1985
Broché, 224 pages
Format poche
Prix : 24 FF ■

Nouvelles cartes pour Canon X-07

INFO-SYSTÈMES met fin au principal reproche que l'on peut faire aux ordinateurs de poche. Mémoire trop faible ? C'est fini, tout au moins pour le Canon X-07.

Info-Systèmes propose effectivement deux nouvelles cartes mémoire CMOS pour augmenter notablement la capacité du X-07. La première carte offrira 40 Ko de mémoire vive supplémentaire (durée de vie d'environ dix ans grâce à sa pile au mercure), la seconde permettra de figer 32 Ko de mémoire en Eprom.

Les deux cartes sont vendues indifféremment 850 FF ttc. D'autres cartes de moindre capacité sont également disponibles : une mémoire vive de 8 Ko à 380 FF et une Eprom (également 8 Ko) à 150 FF.

Info-Systèmes
ZA La Combe
01510 Virieu le Grand
Tél. (79) 87 83 72 ■

Pour les professionnels de l'image

L'ECOLE Nationale des Arts Décoratifs organise des stages d'informatique graphique destinés aux professionnels de l'image. Les participants pourront ainsi se familiariser avec les différents systèmes d'infographie utilisés dans les domaines de la communication, de l'architecture, de la conception d'objets et de formes... ■

TROIS CASSETTES

Des logiciels pour votre PB-700
Edité par Logi'stick
Distribué par DDI
Calc
Graphe
Fichiers
Prix de chaque cassette : 140 FF environ

LOGI'STICK propose trois logiciels aux possesseurs de Casio PB-700 : *Calc*, *Graphe* et *Fichiers*. Comme leurs noms l'indiquent, ils sont directement issus des tableurs et autres bases de données qui sont maintenant monnaie courante sur les ordinateurs à vocation professionnelle.

Les trois programmes se présentent sous la forme d'une cassette au format standard, accompagnée d'une notice simple, mais claire et suffisante. Leur utilisation nécessite une mémoire un peu musclée : il faut au moins un OR-4 (extension de mémoire 4 Ko) pour *Calc* et *Graphe*, et deux pour *Fichiers*. Pour ceux qui en ont la possibilité, les trois extensions possibles sont recommandées ; *Calc* peut

alors fournir un tableau de 26 colonnes sur 40 lignes.

Calc et *Graphe*, bien qu'utilisables séparément, sont prévus pour être combinés. On peut ainsi tracer avec la FA-10 toutes sortes de graphiques en couleurs à partir des données fournies par *Calc :* histogrammes, camemberts ou graphiques en ligne. A noter que l'histogramme peut être à trois dimensions, grâce à un joli dessin en perspective effectué par la table traçante.

Fichiers fonctionne comme une base de données, chaque « fiche » pouvant comporter jusqu'à douze rubriques.

Pour chacun de ces programmes, l'utilisateur peut sauvegarder sur cassette l'application qu'il a définie ; nul doute que ces trois logiciels seront une aide intéressante pour ceux qui apprécient le côté fonctionnel du PB-700.

PM ■

UN LIVRE

La face cachée du TO7-TO7/70
Jean-Baptiste Touchard
Éditions Cédic-Nathan
Paris, 1985
Broché, 160 pages
Prix : 89 FF

DEMANDEZ à l'intérieur ce que vous ne voyez pas à l'extérieur. Tel semble être le propos de Jean-Baptiste Touchard dans cet ouvrage. Il nous fait en effet voyager dans les méandres du TO7, en des endroits que les manuels officiels ne nous laissaient même pas soupçonner.

Après un aperçu des différentes zones de mémoire, un rappel des notions de numération et de codification nous permet de mieux comprendre les nombreux exemples de "dump" émaillant les passages les plus délicats du livre. L'explication du codage d'un programme Basic et de ses variables, sans être propre au Thomson, est claire ; elle servira sûrement aux « bidouilleurs » ou, plus simplement, aux curieux.

Plus intéressantes pour les initiés, sont présentées les normes Vidéotex, les commandes sonores des téléviseurs ou la commande du circuit de l'interface-jeux (PIA). On trouvera en annexe les schémas des connecteurs et une présentation condensée du processeur 6809.

Bref, un livre qui peut aider beaucoup d'utilisateurs intermittents à devenir des passionnés.

JL ■

Des programmes pour Macintosh

INSATISFAIT des programmes pour Macintosh ou de leur nombre ? Trop fidèle à CP/M pour flirter avec un autre système ? **IQ Software** vient d'apporter une solution aux problèmes de compatibilité. Pour 395 dollars en effet (quelque 3 800 FF), cette société américaine propose un CP/M pour Macintosh en version 128 Ko. L'ensemble comprend aussi un compilateur C et le Macro-Assembleur de **Digital Research.** Le CP/M Macintosh n'est pour l'instant disponible qu'aux Etats-Unis. ■

Forth et Assembleur pour HP-71 B

SI, par impossible, vous vous sentez déjà limité par les possibilités de votre HP-71 B, le module Forth/Assembleur arrive à point pour vous apporter une bouffée d'oxygène... Le Basic vous paraît-il un langage trop figé ?

Voici le langage Forth, redéfinissable et modulable à volonté par l'utilisateur. Quant à l'Assembleur, il vous permettra d'écrire vos propres fonctions Basic, et de gagner du temps. Ame d'écrivain ? L'éditeur de textes devrait déjà vous faciliter l'ouvrage.

Le Forth choisi par HP correspond au standard Forth 83, autrement dit la version la plus récente (la précédente étant le Forth 79). Par-dessus le marché, le constructeur nous octroie quelques petits « plus », comme une pile supplémentaire en virgule flottante de quatre registres X, Y, Z, T et LAST X, ce qui rappellera quelque chose aux utilisateurs de la HP-41 C. Il est également possible d'appeler des mots Forth à partir du Basic ou des fonctions Basic à partir du Forth... L'Assembleur permet enfin l'écriture des fameux fichiers LEX (extension de langage) et BIN (binaires) annoncés par le manuel du 71 B. Un fichier LEX peut servir, par exemple, à traduire les messages d'erreurs en français.

Il ne faut cependant pas se leurrer, car l'assemblage est impossible sans l'aide des IDS (Internal Design Specification), quelques milliers de pages de documentation fournies par HP sur les différents aspects du 71, depuis la liste des 64 Ko du système d'exploitation jusqu'aux plans complets de l'engin.

L'Assembleur autorise aussi l'écriture de nouvelles primitives Forth si vous vous sentez à l'étroit avec les quelque 280 mots du vocabulaire de base.

L'éditeur de textes sert essentiellement à la génération des fichiers sources destinés à l'assemblage ou à la compilation et peut, éventuellement, être employé comme un traitement de texte rudimentaire.

D'autres fonctions sont présentes, comme KEYBOARD IS qui permet avec le module HP-IL, de choisir un autre clavier pour le HP-71. On trouve aussi des fonctions de manipulation de fichiers (REPLACE, DELETE#, INSERT#, SEARCH..) que l'on aurait bien aimé avoir sur le 71 en version de base.

Un manuel de 154 pages, en anglais, est joint à ce module. Attention : il ne s'agit pas d'un manuel d'initation au Forth (ni à l'Assembleur) dont les bases sont supposées acquises. Le tout devrait être disponible à un prix d'environ 1 800 FF ttc.

OA ■

CASSETTES ET DISQUETTES

Masterpaint
Utilitaire d'aide à la création graphique
Cassette pour Oric 1 et Atmos
Edité par Ère Informatique
Prix : 250 FF

Hadès
Assembleur, désassembleur, sourceur, débuggeur, moniteur
Cassette pour Oric 1 et Atmos
Edité par Ère Informatique
Prix : 250 FF

Logo Koala
Permet de construire les procédures Logo en dessinant directement sur la tablette Koala Pad
Disquette pour Apple II ou C.64
Edité par BIP
Prix : 455 FF

Mocking Board : la voix d'Apple

POUR ceux qui racontent leur ordinateur en déclarant « Il ne lui manque que la parole » ou qui en ont assez d'un tête-à-tête passionnant mais désespérément muet avec leur Apple, *Mocking Board* est une solution. Conçue par la société Bip, Mocking Board, destiné à l'Apple IIc, fait office à la fois de synthétiseur de musique et de voix.

Le système se connecte simplement, comprend deux haut-parleurs incorporés (pour la stéréophonie, rien que ça) et 6 canaux répartis sur 3 voies.

Mocking Board est livré avec un logiciel de démonstration et une disquette de développement. Doté d'un vocabulaire potentiel illimité (le systèine est fondé sur les phonèmes), Mocking Board permet aussi bien la synthèse du français, de l'anglais que de l'allemand.

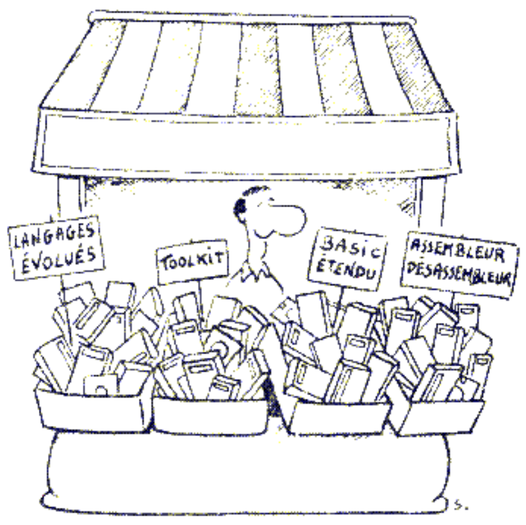
Une précision, Mocking Board peut être utilisé immédiatement avec n'importe quel jeu. Le tout pour 3 000 FF ttc. ■

Amlettres
Traitement de texte
Cassette pour Amstrad CPC 464
Edité par Amsoft
Prix : 149 FF

Easy Calc
Tableur
Cassette pour Canon X-07 (+ extension de mémoire)
Compatible Péritel
Edité par Powersoft
Prix : 130 FF

Assembleur/Désassembleur
Cassette pour Canon X-07
Version 16 Ko
Compatible Péritel
Edité par Logi'stick
Distribué par DDI
Prix : 150 FF

Guide du programmeur
Idées, méthodes et routines en langage-machine pour créer ses propres applications sur Koala Pad
Disquette pour toutes les versions Koala Pad
Edité par BIP
Prix : 455 FF

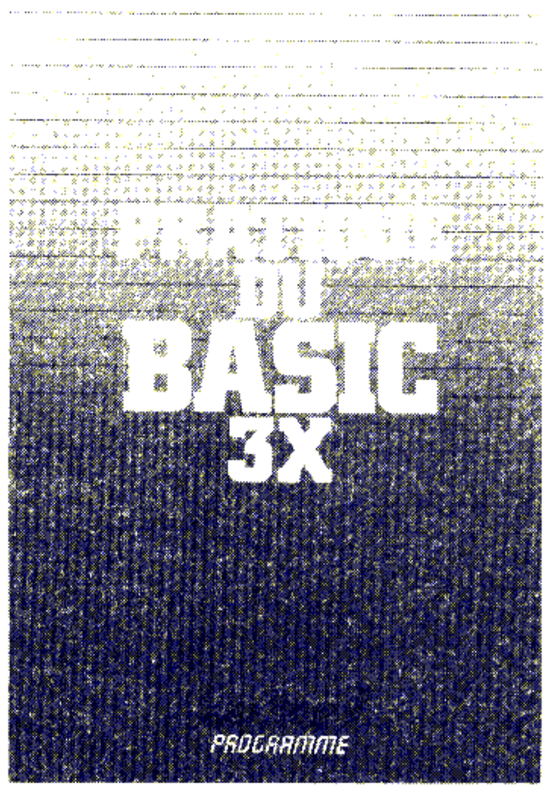


Data Save
Sept fonctions supplémentaires
Cassette pour Oric 1
Edité par ARG Informatique
Prix : 120 FF

D.A.O.
Dessin haute résolution
Cassette pour Oric 1 et Atmos
Edité par ARG Informatique
Prix : 100 FF

Caractor
Redéfinition de caractères
Cassette pour Oric 1 et Atmos
Edité par ARG Informatique
Prix : 100 FF ■

Hector HRX est polyglotte



DANS son numéro 6, *LIST* a testé le Basic de l'Hector 2HR+. Pour les possesseurs du modèle HRX, je voudrais signaler une version Basic 3X spécifique à ce modèle et qui peut leur apporter beaucoup de satisfactions. Ce Basic est disponible sous forme de cartouche, de disquette ou de cassette au même prix que celui du modèle 2HR. Attention, la dernière version porte le n° 5.

L'Hector HRX dispose du langage Forth résident et l'introduction du Basic ne supprime pas la possibilité de l'utiliser. Bien au contraire, une instruction Basic (écrite "FORTH") permet de créer à l'intérieur d'un programme Basic des lignes en langage Forth. Le passage des paramètres d'un langage à l'autre est facilité par les instructions DPEEK et DPOKE (en Forth, l'utilisation courante de la pile se fait par deux octets à la fois) ainsi que VARPTR qui retourne l'adresse du descriptif d'une variable Basic.

L'appareil disposant d'une haute résolution graphique, que reste-t-il comme mémoire réellement disponible avec ces deux langages résidents ? Réponse : 32 Ko, car la mémoire vidéo se situe en dehors de l'espace normal de 64 Ko ; elle reste cependant facilement accessible grâce à VPEEP et VPOKE (V! et V en Forth).

Ainsi, un programme Basic ▸▸

HRX peut contenir des lignes de Forth, des lignes en langage-machine (grâce à l'instruction USR) et, bien sûr, du Basic.

Plus étonnant encore, le Basic est bilingue, et si vous avez une allergie profonde pour la langue de Shakespeare, frappez FRENCH et tous les mots-clés pourront être entrés en bon français (FOR devient POUR, PRINT devient ECRIS, etc.). Un programme écrit en français peut être listé ou poursuivi en anglais et vice-versa grâce aux mots ANGLAIS et FRENCH. Dans les deux langues, tous les mots-clés peuvent s'abréger par un point. Frappez "J." lors de l'entrée d'une ligne et le mot "JUSQU'A" apparaîtra à l'écran au prochain listage.

Le Basic 3X permet de nombreuses possibilités pour la sauvegarde sur cassette :
- fonction VERIFY ;
- PRINT # et INPUT # pour une ou plusieurs variables alphanumériques ou numériques ;
- LOADC et SAVEC pour une zone mémoire quelconque avec rechargement possible à une adresse différente ;
- sauvegarde d'un écran ;
- sauvegarde de la totalité des variables d'un programme ;
- PROTECT pour protéger les programmes et déclencher un "auto-run" ;
- et toutes les possibilités du 2HR telles MERGE, APPEND, etc., mais sans passer au préalable par EDIT.

Dans un autre domaine, les amateurs de jeux pourront utiliser les nouvelles instructions sonores LASER, BELL, SIREN et SHOT. Pour le graphisme, tout ce qui a été dit dans le test du 2HR reste valable, mais nous disposons de nombreuses possibilités supplémentaires.

Les textes peuvent s'afficher avec deux alphabets : STANDARD avec minuscules accentuées, possibilité du mode calque ou mode papier, vidéo normale ou inverse, et SPECIAL qui ne comporte que les majuscules mais offre la possibilité de multiplier leurs dimensions en hauteur et en largeur avec l'ordre SCALE.

Pour les dessins, deux modes sont également disponibles : LITTLE et BIG. L'énorme avantage de BIG est qu'il permet de travailler aussi bien pour les tracés que pour les peintures avec une encre multicolore, ce qui permet par exemple à BOX de produire sur l'écran un rectangle de 255 façons différentes suivant l'argument d'INK.

A l'aide de FROM et de TOWARDS, on trace une succession de lignes en ne précisant à chaque fois que le point d'arrivée. Autres instructions graphiques :
- BRIGHT passe une couleur en demi-luminosité ;
- RUB gomme ;
- CLS(n) efface l'écran de la couleur n ;
- FLASH(m,n) fait clignoter la couleur m, n fois ;
- VOLUME permet de donner aux dessins certains effets de volume (élimination des lignes cachées).

En ce qui concerne les instructions de chaînes, signalons quelques originalités par rapport au 2HR :
- INKEY$ permet de saisir au vol la frappe d'un caractère en précisant le temps de scrutation ;
- FREE donne soit le nombre d'octets disponibles, soit l'espace disponible réservé aux chaînes ;
- ASC accepte un deuxième argument et retourne alors le code ASCII du énième caractère d'une chaîne ;
- RECTIFY permet de modifier une variable.

Signalons enfin aux amateurs qu'ils disposent de trois ports d'entrée-sortie entièrement réinitialisables, de huit entrées "contacts", et de deux entrées analogiques.

Le Basic 3X est vendu environ 350 FF sur cassette et 850 FF en cartouche. Il existe également une version disquette (380 FF seule) qui est donnée à l'achat d'un lecteur de disquette.

MB ■

DU CÔTÉ DES CLUBS

Dans le département de la Charente

LE club Micromut, rattaché à la Fédération Nationale *Microtel*, reste ouvert toute la semaine, du lundi au dimanche, à partir de 9 heures et propose, le mercredi de 18 à 20 heures, des cours d'initiation tous niveaux sur Goupil 2.

Pour assister à ces cours, gratuits pour les adhérents, une participation forfaitaire de 150 francs vous sera demandée. Si vous souhaitez adhérer au club, vous devez être âgé de 15 ans au minimum ; la cotisation est annuelle et coûte 290 francs.

Pour tous renseignements, vous pouvez contacter Alain Jally au (45) 84 16 55 ou Joël Chardonne au (45) 84 18 18.

Club Informatique Micromut
28 allées de Blossac - BP 35
16500 Confolens

Dans le Pas de Calais

LA Maison du Temps Libre de Douvrin abrite l'Association Douvrinoise de Micro-informatique qui elle-même, moyennant une cotisation de 100 francs par an, accueille ses adhérents tous les mardis et vendredis soirs à partir de 18 heures.

Différents secteurs d'activités sont prévus : initiation et perfectionnement sur ZX 81, TO 7 et Silz 16, gestion et traitement de texte sur Silz 16 (avec l'aide d'un informaticien professionnel).

Association Douvrinoise de Micro-informatique
Maison du Temps Libre
2 rue Séraphin Cordier
62138 Douvrin
Tél : (21) 79 80 11

En Seine Maritime

A Gainneville (près du Havre, sur la RN 15), la section informatique du Centre de Loisirs propose des cours d'initiation ouverts à tous (enfants à partir de dix ans et adultes). Ces cours ont lieu une semaine sur deux, à la bibliothèque de Gainneville : le mercredi de 17 h 30 à 19 h 30 pour les nouveaux venus et le jeudi de 18 à 20 heures pour les « anciens ».

Un samedi sur deux, de 10 à 12 heures et toujours à la bibliothèque, se tient une permanence où les membres du club échangent trucs et astuces. S'ils possèdent déjà un ordinateur, ils peuvent l'apporter ; sinon, le club met à leur disposition un Dragon 32.

La cotisation pour l'année scolaire 1984/85 (septembre à juin) s'élevait à 230 francs.

Pour en savoir plus, vous pouvez vous renseigner auprès de Gilbert Rosay au (35) 20 47 11.

Centre de Loisirs de Gainneville
76700 Gainneville

Dans l'Essonne

LE club Micro-informatique d'Etampes, qui fonctionne au Centre Social sous l'égide de l'Association « Expression » depuis septembre 1984, étend son activité. Moyennant un « droit d'inscription » de 50 francs auquel s'ajoute une cotisation trimestrielle de 100 francs, vous pourrez participer, quel que soit votre niveau, aux cours d'informatique dispensés sur Amstrad CPC 464 et Texas TI 99/4A.

Des cours d'initiation ont lieu chaque mardi, de 18 à 19 heures pour les enfants, et de 19 à 20 heures pour les adultes.

Deux permanences sont également assurées, le jeudi à partir

Faites-vous connaître...

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

de 20 h 30 et le vendredi à partir de 18 h ; elles permettront à tous d'échanger expérience et idées et de consulter les livres et revues du club.
Club de Micro-informatique
Centre Social
Association « Expression »
Place Saint Gilles
91150 Etampes
Tél. : (6) 494 59 39,
(6) 494 78 29, (6) 494 96 42.

En Seine et Marne

LE Microtel Club de Melun serait heureux d'accueillir de nouveaux adhérents chaque jeudi soir de 18 à 22 heures. Au programme, initiation à l'informatique, pratique des langages de programmation (Basic, Pascal, Forth, Logo), fabrication de cartes (interface minitel par exemple) et de robots, montage d'ordinateur en kit.
Microtel Club Melun
6 rue Paul Valéry
77000 Melun
(6) 068 67 83

De particulier à particulier

SI vous avez du matériel à vendre, vous pouvez contacter la Maison de Quartier de Courcouronnes qui organise le dimanche 10 mars prochain, de 10 à 18 heures, sa deuxième journée du micro-ordinateur d'occasion. La réservation d'un stand vous coûtera 50 francs, et vous pourrez vendre ou acheter du matériel directement de particulier à particulier. L'entrée visiteur est de 5 francs et seulement 10 francs par famille.

Vous trouverez sur place un buffet et une animation (jeux, librairie informatique...).
Maison de Quartier de Courcouronnes
Le Mail de Thorigny
91000 Courcouronnes
Tél. : (6) 077 03 95 ■

UN PETIT TOUR CHEZ LE LIBRAIRE

Pratique des Apple
Volume 1
Basic Applesoft
Henri Lilen
Editions Radio
Paris, 1984
Broché, 190 pages
Prix : 105 FF

Pratique des Apple
Volume 2
Au-delà du Basic avec l'Assembleur
Alain Andrieux
et Gérard Creuzet
Editions Radio
Paris, 1984
Broché, 198 pages
Prix : 120 FF

Guide du VG 5000 Philips
Benoît Amsler
et Christophe Bardon
Editions Edimicro
Paris, 1984
Broché, 176 pages
Prix : 88 FF

Oric Atmos, vos programmes
Michel Bussac et
Gil Espèche
Editions Cédic/Nathan
Collection Micromonde
Paris, 1984
Broché, 128 pages
Prix : 35 FF

Tout savoir sur Lynx
Bruno Vanryb et
Roger Politis
Editions Eyrolles
Paris, 1984
Broché, 164 pages
Prix : 95 FF

La solution RS-232
Joe Campbell
Editions Sybex
Paris, 1984
Broché, 200 pages
Prix : 148 FF

Initiation aux bases de données
Nigel Freestone
Editions Belin
Collection Modulo
Paris, 1984
Broché, 118 pages
Prix : 95 FF

La conduite des Atari XL
600 XL/800 XL/1245 XLD
Patrick Oros
Editions Eyrolles
Paris, 1984
Broché, 240 pages
Prix : 98 FF

La micro en 100 questions
Bruno de Latour
Editions Cedic/Nathan
Collection Micromonde
Paris, 1984
Broché, 160 pages
Prix : 39 FF

Tout sur le MO5
Jean-François Bieber,
Alain Perbost et
Gilles Renucci
Editions Edimicro
Paris, 1984
Broché, 256 pages
Prix : 98 FF

Initiation pratique à l'informatique
Jean-Luc Charron,
Elian Fanouillet
et Monique Yelloz-Danhiez
Editions Nathan-Technique
Paris, 1984
Tome 1 - **Information et programmation**
Broché, 128 pages
Prix : 58 FF
Tome 2 - **Fichier et analyse**
Broché, 176 pages
Prix : 69 FF

Encyclopédie de la micro-informatique
Peter Rodwell
Editions Hachette
Informatique
Paris, 1984
Relié, 206 pages
Prix : 149 FF ■

LE PASSÉ MÉCONNU DU BASIC

CURIEUSEMENT, alors que les documents abondent sur la façon dont les langages de programmation sont nés et parvenus à maturité, on ne trouve guère de matériaux pour retracer la trajectoire de celui qui est actuellement le plus répandu sur les micro-ordinateurs : le Basic. Et pourtant, sa petite histoire existe, nous l'avons retrouvée.

On ne voit jamais Popeye sans épinards ni de chiens sans puces ; pas de micro non plus sans Basic. Faudrait-il croire que le Basic a été inventé pour les micros ? Que ceux-là n'ont pu exister que parce qu'il leur offrait sa simplicité ? On pourrait s'y tromper. Pourtant, l'histoire est formelle : il n'y a aucun rapport effectif entre ces deux inventions, dont la conjugaison a connu le succès prodigieux que l'on sait.

En ce qui concerne les premiers micro-ordinateurs, ils sont nés de la commercialisation de microprocesseurs dont les inventeurs ne cherchaient certainement pas, au départ, une utilisation en langage évolué. Les premiers appareils disponibles, américains notamment, étaient de simples cartes (parfois avec un clavier hexadécimal !) qu'il fallait programmer directement en Assembleur. L'idée de recourir à des langages du genre de Basic n'aurait certainement pas paru naturelle à ces pionniers.

Inversement, John Kemeny et Thomas Kurtz auraient de toute évidence haussé les épaules si on leur avait prédit, en 1965, que leur travail pédagogique prendrait, quelque quinze ans plus tard, une telle importance et qu'il contribuerait de cette façon à l'explosion de la micro-informatique !

Classiquement, l'histoire du Basic se résume à quelques lignes : « Basic est né, vers 1965, pour des besoins universitaires au Dartmouth College, dans le cadre du *time sharing* ; il a connu la gloire avec la fulgurante percée d'Apple et de ses concurrents vers la fin de la décennie suivante ». Il semble aujourd'hui très difficile de se procurer un exemplaire du premier manuel consacré à ce langage.

Un travail universitaire

La biographie de John Kemeny, au moins, est maintenant assez connue, en partie grâce à un incident indépendant de notre sujet : la catastrophe nucléaire de Three Mile Island, en 1979, survenue en plein cœur d'une centrale importante, proche de la ville d'Harrisburg en Pennsylvanie. A la suite de cette catastrophe, John Kemeny fut désigné comme président de la commission d'enquête. Pendant sept mois, il s'est attaqué au redoutable problème de savoir si cet accident ne devait pas sonner le glas d'une industrie d'un poids économique et politique si essentiel à notre société dévoreuse d'énergie. On sait que ses conclusions furent assez optimistes : espérons qu'il avait de bonnes raisons scientifiques pour cela !

Né en Hongrie en 1926, John Kemeny débarque aux États-Unis en 1940, entre comme étudiant à l'Institute for

Advanced Study (IAS) de Princeton où, après avoir servi comme assistant d'Albert Einstein pendant un an, il obtient en 1949 un Ph.D. en mathématiques. Son premier travail scientifique sérieux n'avait pas non plus été très banal : il fit partie de l'équipe de Los Alamos (déjà le nucléaire) qui, en 1945, résolvait quasiment « à la main » les équations différentielles nécessaires à la prédiction du déroulement de l'explosion de la bombe, avec comme seule aide des machines de bureau de type « Marchant » et des tabulatrices IBM à cartes perforées du nom bizarre de « collators » sous la direction de Stanley Phillips Frankel, Nicholas Constantine Metropolis et, surtout, son compatriote John (Janos) von Neumann, émigré dès 1930, également universitaire hors pair et gloire de Princeton.

Un article important publié dans le Scientific American en 1955 (*Man Viewed as a Machine*), partiellement inspiré justement de conférences de von Neumann à l'IAS, révéla son nom comme celui d'un vulgarisateur de talent. Il était déjà professeur au Dartmouth College depuis 1953. En 1957, il publiait en collaboration *Introduction to Finite Mathematics* suivi de *Finite Mathematics with Business Applications* (1962), bien connus en France car traduits par Dunod dès 1964, deux des premiers livres à populariser les fameuses « mathématiques modernes », ici tournées vers les activités humaines et la gestion. En 1965, l'ouvrage consacré par David Bergamini aux mathématiques dans la collection Life publiait — rare honneur — la photographie de Kemeny entouré de jeunes étudiants, expliquant qu'il parcourait inlassablement les États-Unis afin de persuader les meilleurs scientifiques de ne pas se détourner de cette science pour devenir ingénieurs ! On voit que l'homme est plutôt peu commun, mais l'histoire de l'informatique est assez riche en ce genre de canards...

Dartmouth est le nom d'un vieux collège du célèbre groupe d'universités de la côte Est. Il occupe toute une petite ville : Hanover, à 250 miles de New York vers Montréal, à la frontière entre le New-Hampshire et le Vermont (un souvenir personnel : introduit auprès de Kemeny durant l'été 1967 par l'intermédiaire d'un ami commun et parcourant à sa recherche pour la première fois un campus américain, j'ai dû faire deux fois le tour de l'agglomération en guettant le collège avant de comprendre qu'il était à la fois partout et nulle part ! Même le boulanger était presque de la maison...). C'était là que Georges Stibitz, des Bell Telephone Laboratories (ceux qui inventeront le transistor), avait installé le 11 septembre 1940 pour le congrès de l'American Mathematical Society, des télétypes reliés à l'un des précurseurs à relais (mais non programmable) de l'ordinateur, le *complex number calculator*, qu'il avait construit en avril 1939.

Cette anecdote est, d'une certaine façon, directement liée à la naissance de Basic car la machine de Stibitz préfigurait l'utilisation d'un matériel informatique à partir d'un terminal, exactement comme dans le concept de temps partagé *(time sharing)*. Ce concept commençait à se répandre lorsque John Kemeny décida, en 1964, poussé par John McCarthy, père de Lisp, de créer le *Dartmouth Time Sharing System* (DTSS). On sait que ce procédé, alors tout à fait révolutionnaire (il avait été inventé au MIT en 1959), consistait à faire travailler ensemble — apparemment — plusieurs utilisateurs sur la même unité centrale, chacun pouvant imaginer être le seul à accéder aux ressources de l'ordinateur par l'intermédiaire de sa machine à écrire (il n'y avait pas encore d'écran à l'époque).

Deux ans de mise au point

John Kemeny s'était adjoint Thomas Kurtz, lui aussi mathématicien Ph.D. de Princeton en 1956 et enseignant à Dartmouth depuis. Leur travail essentiel, avec le recul du temps, n'est pourtant pas le DTSS mais plutôt, bien entendu, une réalisation a priori annexe, le *Beginner's All-purpose Symbolic Instruction Code* (en d'autres termes, le Basic) qu'ils mirent au point entre 1963 et 1965. Fixer une date plus précise semble difficile, même si les auteurs se réfèrent (d'après un article de Computer Language de juillet 1984 et une interview dans PC World de novembre 1984) à... 4 heures de l'après-midi du 1er mai 1964, quand, pour la première fois, un programme Basic tourna sans faute sur un General Electric 225 piloté par télétype, créant en même temps ce langage et le système de temps partagé de Dartmouth.

Il serait faux de croire que ces deux professeurs cherchaient ainsi à rendre service à des entreprises bloquées par les délais prohibitifs que prenaient les allers et retours entre le service informatique et les machines inabordables. Leur but n'était nullement économique, mais essentiellement pédagogique. En fait, le recours à l'ordinateur sous forme de libre-service, sans passage obligé par des spécialistes en blouse blanche, était tout simplement destiné à aider leur enseignement mathématique (toujours dans le but de motiver le plus grand nombre possible d'étudiants aux maths).

Basic est, en effet, le premier langage

LE PASSÉ MÉCONNU DU BASIC

simple et vraiment interactif (le merveilleux INPUT...). A l'époque, seul Fortran existait réellement ; c'était le langage d'IBM par excellence. Algol n'avait pas pris la place que les universitaires auraient voulu lui donner ; Lisp était trop spécialisé et Cobol, bien entendu en pleine activité déjà, ne pouvait servir qu'à des activités trop éloignées des besoins des étudiants. Mais Fortran était, surtout à l'époque, d'un maniement assez pénible ; il fallait absolument disposer d'instructions assez rustiques, assimilables en quelques heures ; une trop longue initiation ne pouvait être envisagée. Il était donc nécessaire d'extraire de Fortran un sous-ensemble assez efficace pour les besoins limités que visait le Dartmouth College et, surtout, assez simple comme le prouve le choix même des termes (beginner = débutant, all-purpose = tous usages) et, évidemment, le jeu de mots qu'est l'acronyme lui-même (basic = de base).

On a fait des progrès depuis

C'est donc une erreur que de dire, comme on le lit parfois, que Basic était destiné à faciliter l'apprentissage de Fortran ou de l'informatique ; il ne s'agissait, en fait, que de rendre presque immédiat le recours aux possibilités d'un ordinateur pour rendre plus attrayant le métier de mathématicien aux yeux d'étudiants plus intéressés par le dollar ou le fer à souder que par les décimales de pi et autres cycloïdes raccourcies.

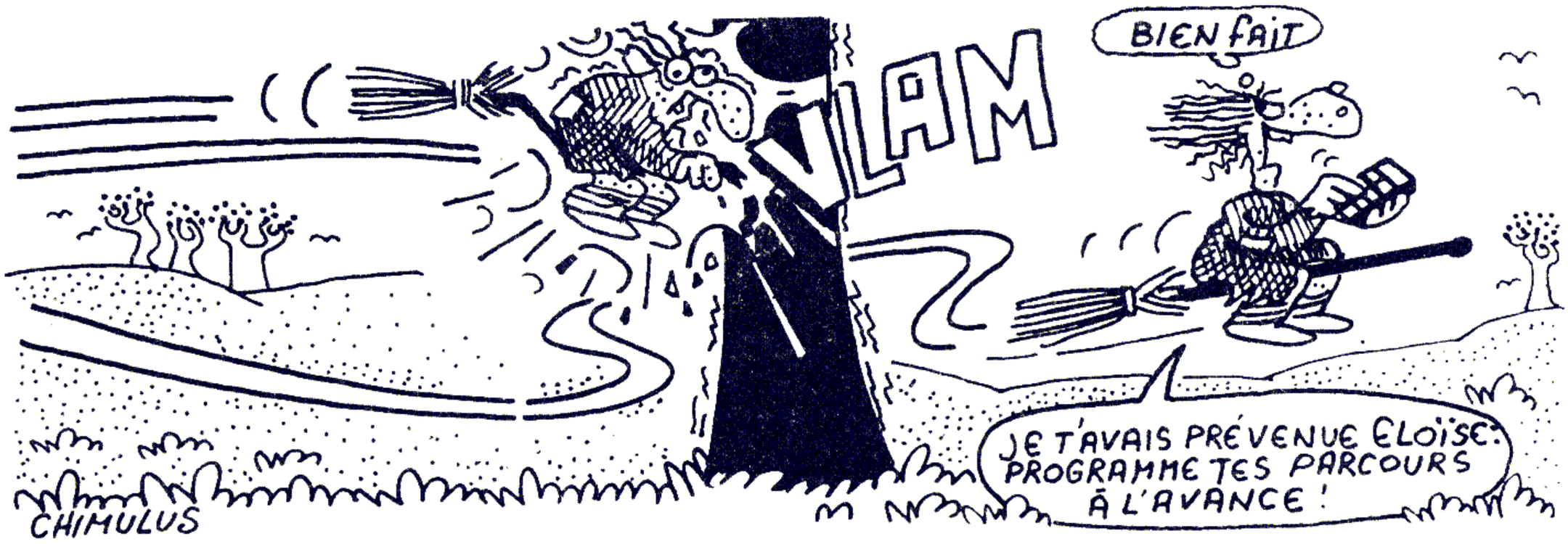
On dit que John Kemeny regrette parfois que son projet initial n'ait pas réellement réussi à bouleverser suffisamment l'enseignement des mathématiques — quoique tout ne soit pas dit, loin de là, en ce domaine. Ce qui est certain, et éclaire mieux les raisons de la longue guerre Basic/LSE en France, c'est que cette toute première version de Dartmouth était, sans faire injure à John Kemeny, bien médiocre, surtout pour des yeux de 1985.

Jacques Hebenstreit a raison de rappeler que le IF-THEN manquait alors de ELSE, qu'il n'y avait pas de WHILE ni de REPEAT, ni de procédure, et qu'il exista très vite de trop nombreuses versions incompatibles. La description du tout premier Basic implanté sur leur machine, vers 1974 ou 1975, par la société française MBC de Georges Cottin et Georges Bouhot, créateurs d'Alcyane, est éloquente. La liste des instructions en est bien pauvre : STOP INSERT DELETE DELETE[n] SCRATCH CONTINUE LIST (trois types différents) FIXED[n] DIM (une dimension, huit octets par élément seulement) INPUT LET DISP PRINT TAB GOTO GOSUB RETURN IF-THEN.

On peut le constater : rien pour les fichiers. Il faudra attendre d'autres versions plus satisfaisantes pour que l'on puisse commencer à les traiter et à utiliser des réels (FLOAT[n]). Surtout pas de FOR...NEXT (qui était pourtant, sauf erreur, dans le premier Basic de 1964). Bien entendu des problèmes de taille mémoire peuvent expliquer cette austérité. Rien non plus sur les ordres graphiques (DRAW...) qui ne virent le jour que vers 1970 (les machines de l'époque étaient évidemment incapables de gérer de telles instructions, pour lesquelles un écran est quasiment indispensable), ni sur le maniement des chaînes de caractères, point fort par exemple du LSE destiné en particulier aux applications non mathématiques. Ces ajouts trop tardifs sont, sans aucun doute, la cause des difficultés insurmontables que rencontra la tentative de normalisation de l'ANSI, plus ou moins avortée, de 1979, qui ne pouvait qu'aboutir à un noyau beaucoup trop restreint — et que certains qualifient, à tort, de Basic « standard ».

Les évolutions ultérieures de ce langage et son avenir méritent encore bien des développements. En particulier, avec la révolution que Bill Gates et Paul Allen provoqueront en écrivant le premier logiciel de Microsoft pour l'Altaïr, ou avec le merveilleux *True Basic* que Kemeny et Kurtz viennent de nous offrir. L'histoire n'est sans doute pas finie et Dartmouth est encore un endroit où, à chaque instant, il se passe quelque chose !

André WARUSFEL



FAMILLE COMMODORE

LES CONTRÔLES ONT DU CARACTÈRE

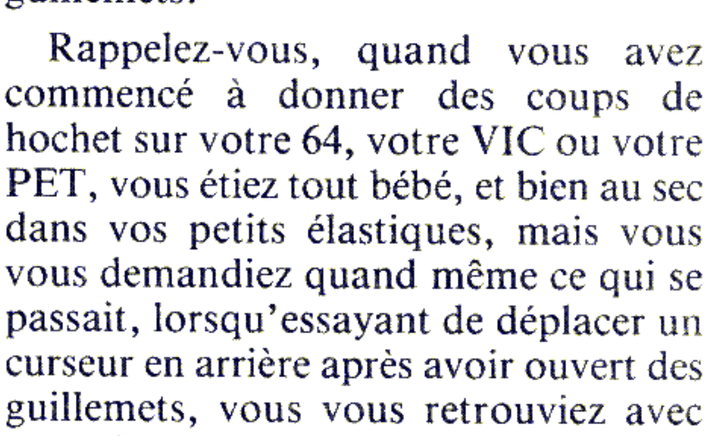
LES Commodore 4000 grand écran ou 8000, Vic 20 ou C.64 font souvent appel à des caractères de commande. Certains sont connus, d'autres moins, et le bon usage des guillemets permet de les rendre visibles, mais aussi de les utiliser plus commodément.

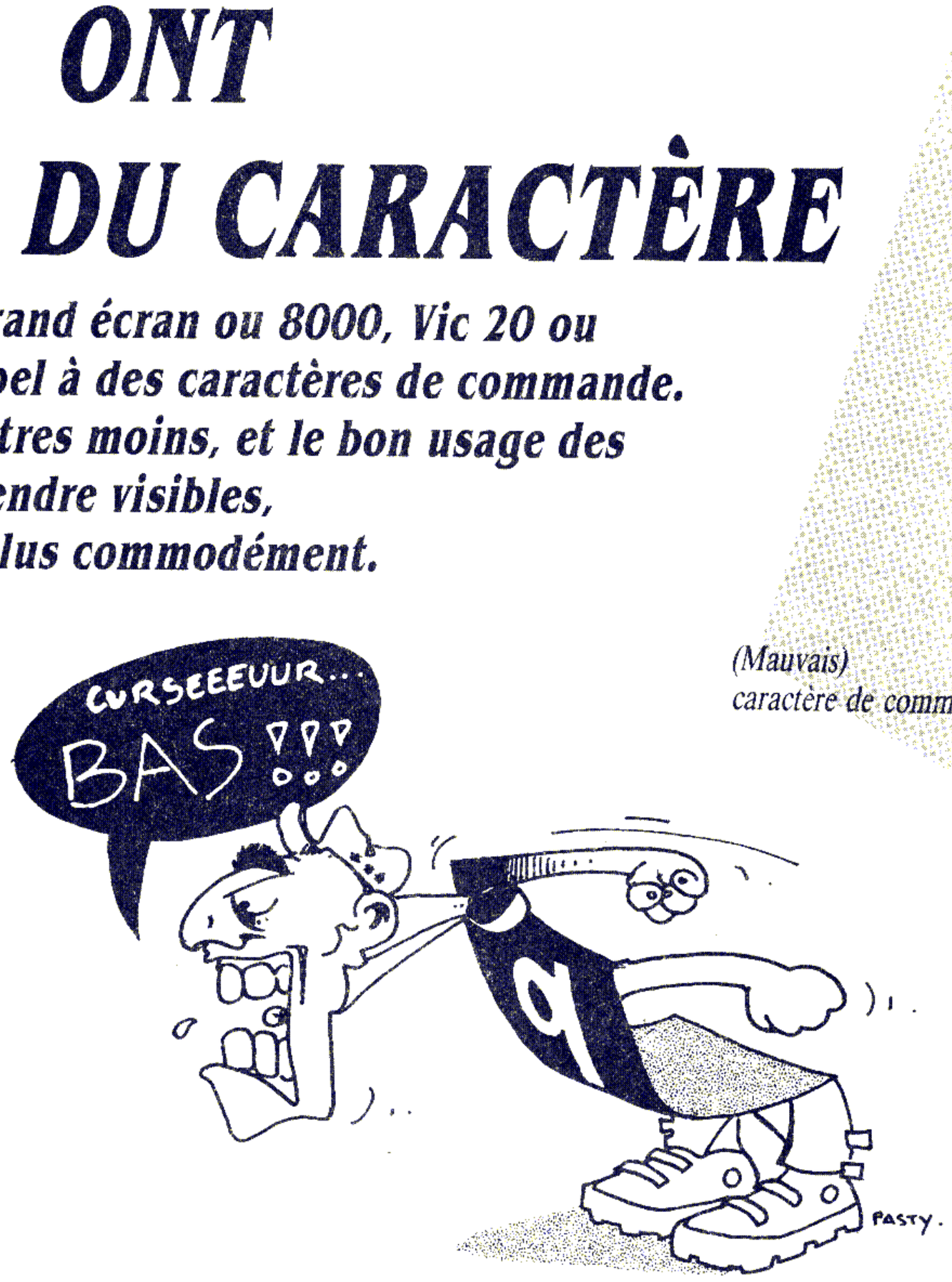
Connaissez-vous un esprit plus secret, une âme plus cryptique, un surmoi plus dissimulateur que celui de votre ordinateur ? Si on laisse de côté ceux qui se servent d'un ordinateur comme on prend le métro, sans jamais programmer et qui par conséquent ne sont pas des lecteurs de *LIST*, le branché-info moyen doit passer 97,32 % de son temps de clavier à essayer de découvrir ce que sa bécane lui cache.

Ah, le don de dévoiler le voilé, les lunettes à voir l'invisible... Mais c'est qu'elles existent, ces lunettes, et chez Commodore, on appelle ça des guillemets.

Rappelez-vous, quand vous avez commencé à donner des coups de hochet sur votre 64, votre VIC ou votre PET, vous étiez tout bébé, et bien au sec dans vos petits élastiques, mais vous vous demandiez quand même ce qui se passait, lorsqu'essayant de déplacer un curseur en arrière après avoir ouvert des guillemets, vous vous retrouviez avec une tripotée de hiéroglyphes biscornus qui vous plongeaient dans une rogne Conan-le-Barbaresque mâtinée d'un soupçon d'étonnement socratique.

C'était hier. Maintenant, les caractères de contrôle les plus fréquents n'ont guère de secret pour vous. Pourtant,





(Mauvais) caractère de commande

n'en reste-t-il pas d'autres, plus discrets, moins voyants, mais à la chair plus fine et au parfum plus délicat ? C'est eux que nous allons découvrir aujourd'hui.

Mode texte et mode graphique

Lorsque vous mettez sous tension un CBM 8000, il est en mode minuscules/majuscules ; lorsque vous mettez sous tension un 4000 grand écran, il est en mode majuscules/graphiques. Vous voulez changer, que faites-vous ? POKE 59468,12 pour passer en majuscules/graphiques et POKE 59468,14 pour passer en minuscules/majuscules ? C'est une solution, mais ce n'est pas la seule.

Essayez donc, à la mise sous tension, d'afficher un bout de texte, et, par exemple un gros pavé inversé :

```
100 PRINT"[CLR]COUCOU MAMAN !"
110 FOR I=1 TO 10
120 PRINT"[RVS][10 ESPACES][OFF]"
130 NEXT I
```

Certes, le texte est en majuscules sur 4000 GE et en minuscules sur 8000, mais il y a une autre différence : là où le 4000 affiche un pavé carré, le 8000 affiche une manière de maillot à rayures ; les lignes ne sont pas jointives. C'est qu'indépendamment des questions de majuscules, existent un mode dit "texte" et un mode dit "graphique". Le premier ajoute des espaces entre les lignes pour plus de lisibilité, l'autre les resserre pour permettre des verticales continues. Or le passage de l'un à l'au-

LES CONTRÔLES ONT DU CARACTÈRE

tre ne se fait pas par POKE, mais avec un PRINT et un caractère de commande. Le 8000, à la mise sous tension, est en mode TEXTE. Faites PRINT CHR$(142) : vous verrez votre écran se ratatiner, votre message personnel s'afficher en majuscules, et votre pavé perdre ses rayures pour devenir un impressionnant monolithe. Le 4000 GE, à la mise sous tension, est en mode GRAPHIQUE. Faites PRINT CHR$(14) : vous verrez votre écran éclater (le changement de mode n'attend pas la synchro image, c'est parfois spectaculaire), votre pavé se rayer, et votre salutation s'afficher en minuscules.

Naturellement, si vous aimez les textes tassés ou les graphiques lâches, rien ne vous empêche de faire suivre votre PRINT CHR$(142) d'un POKE 59468,14 ou votre PRINT CHR$(14) d'un POKE 59468,12.

Cela dit, soyez franc : lorsque vous lisez dans une liste PRINT CHR$(147), frappez-vous au clavier CHR$(147) ? Non, bien sûr. Vous ouvrez des guillemets, vous frappez SHIFT CLR, un cœur en vidéo inversée apparaît et l'écran ne s'effacera qu'à l'exécution. C'est donc que le CHR$(147), lorsqu'il est précédé de guillemets, prend la forme d'un caractère visible exécutable plus tard, comme les caractères de déplacement du curseur.

Preuve :

PRINT CHR$(147)

L'écran s'efface.

PRINT CHR$(34);CHR$(147)

L'écran ne s'efface pas, mais le cœur à l'envers apparaît derrière les guillemets.

Hé bé alors, nyapluquà continuer :

PRINT CHR$(14)

L'écran passe en mode TEXTE (minuscules et lignes écartées).

PRINT CHR$(34);CHR$(14)

Le caractère quatorzième apparaît enfin à nos yeux émerveillés. Il est aussi sobre que puissant et se présente sous la forme d'un "n" en vidéo inversée. Continuons.

PRINT CHR$(34);CHR$(142)

Voilà le mode GRAPHIQUE "caractérisé" à son tour : comme on pouvait s'y attendre puisque la différence entre 142 et 14 est de 128, c'est un "N" majuscule en vidéo inversée.

Alors, de là à avoir envie d'incorporer ces caractères dans une chaîne, comme un CLR... Seulement, s'il y a une touche CLR, il n'y a pas de touche TEXTE/GRAPHIQUE. Et si l'on essaie d'écrire en vidéo inversée derrière des guillemets ouverts, on a un "r" inversé, et des caractères normaux ensuite. Rusons donc.

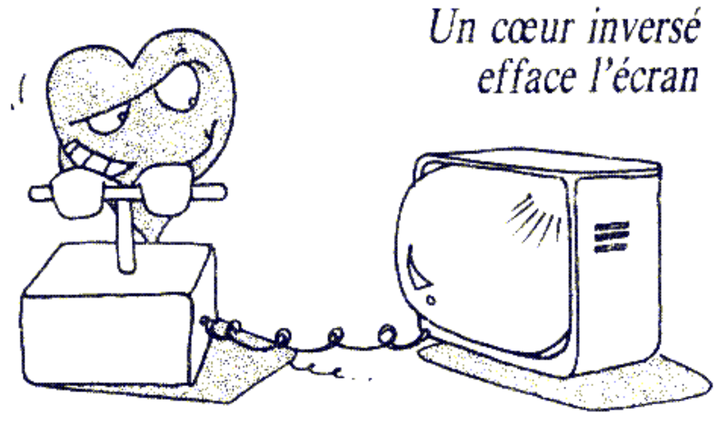
Sur ma ligne 100, après le cœur du CLR, j'insère un espace avec la touche SHIFT INST. Mais je ne suis guère avancé, car un espace inséré est en mode guillemets. Je vais donc frapper un espace, pour faire croire à la machine qu'on n'est plus en mode QUOTE. Mais ce n'est là qu'une feinte ! Un petit coup de curseur gauche et me voici sur le blanc, tandis que l'ordinateur ne se doute de rien. Alors, rapide tel le Cheyenne aux souples mocassins, je passe en vidéo inversée, mais cette fois, en mode direct : aucun caractère n'apparaît à l'écran. Et quand je frappe la touche du "n", il est bel et bien inversé. Un coup d'index preste et discret sur la touche RETURN, ma ligne est validée, et le caractère sera lu à l'exécution comme un CHR$(14).

Frappons en mode direct PRINT CHR$(142) pour rendre l'expérience plus concluante, puis RUN. Ce petit bout de "n" inversé de rien du tout a suffi pour faire repasser l'écran en mode TEXTE. Quand on cherche à gagner de la place en mémoire, c'est radical : si on profite d'un PRINT déjà existant, ça coûte un octet contre dix pour POKE 59468,14.

Comment faire grelotter un CBM

Le vénérable code ASCII a prévu de faire tintinnabuler les vieux télex, et les CBM 8000 et 4000 GE ont suivi le mouvement. Il suffit d'un PRINT CHR$(7) pour les faire sonner un coup, et du même en shifté, CHR$(135) pour les faire sonner deux coups. Là encore, notre légitime curiosité doublée des moyens techniques de l'assouvir, nous fait chercher à "voir" ce qui s'entend : PRINT CHR$(34);CHR$(7):PRINT CHR$(34);CHR$(135).

Cette fois, ce sont des "g" inversés,



Un cœur inversé efface l'écran

le minuscule et le majuscule, qui nous montrent à quoi ressemble la sonnette. Pour les reproduire à l'intérieur d'une chaîne entre guillemets, même système :

```
100 OPEN 15,8,15,"I0"
110 OPEN 82,8,2,"ZXCFWR,S,R":
    GOSUB 900
120 END
900 INPUT # 15,E1,E2$,E3,E4:IF E1
    < > 0 THEN RETURN
910 PRINT"[RVS]ERREUR NO.";
    E1;":";E2$
920 PRINT"PISTE";E3;"SEC
    TEUR";E4
930 CLOSE 15:END
```

Ramener le curseur sur [RVS], frapper la séquence SHIFT INST, ESPACE, CURSEUR GAUCHE, RVS, SHIFT G. Un "g" inversé apparaît. Exécutez le programme avec les portes des drives ouvertes si vous avez un fichier "zxcfwr" ! Le message d'erreur sera précédé d'une sonnerie à vous tirer de votre lecture.

Deux caractères bien commodes présents sur le 8000 et le 4000 GE, et dont on peut déplorer qu'ils aient été lâchés sur VIC et C.64, sont le 16 et le 150.

Le CHR$(16) a la vertu d'effacer tout le reste d'une ligne, après l'endroit où se trouve le curseur. Le CHR$(150), lui, efface toute la ligne avant l'endroit où se trouve le curseur. Pour voir à quoi ils ressemblent, PRINT CHR$(34); CHR$(16) et PRINT CHR$(34);CHR$(150) nous montrent respectivement un "p" et un "V" shifté, l'un et l'autre inversés. Le premier surtout est bien commode :

```
100 PRINT
110 INPUT"DRIVE";DN$
120 IF DN$ < > "0" AND DN$
    < > "1" THEN 110
```

Ce machin-là affichera la question autant de fois qu'il le faudra pour obtenir une des deux réponses exigées, au risque de ficher en l'air le reste de l'écran dans l'hypothèse d'un utilisateur

mal embouché ou obtus. Il serait alors opportun de prévoir. Ramenant le curseur sur le D de DRIVE, on insère deux espaces. Espace inséré = mode guillemets, disions-nous ? Profitons de l'aubaine pour remplir le premier d'un curseur haut. Le second sera déguillemettisé (?) avec un espace, puis on y fera la séquence CURSEUR GAUCHE, RVS, P, suivie, naturellement, d'un RETURN.

L'effet produit sera peu spectaculaire au premier coup, puisque le CHR$(16) n'effacera qu'une ligne vide. Mais si on répond "3" à la question, le curseur-haut reviendra à la ligne qu'il vient de quitter, et l'effacera avant d'afficher derechef la question et de lire la réponse. L'utilisateur réfractaire peut écrire "TROIS VIRGULE SEPT" en toutes lettres si ça lui chante, la question sera toujours reposée au même endroit et sur une ligne propre.

A noter que même chez les grands, on peut se tromper : vous avez remarqué que tous ces caractères vont par paire, l'un shifté, l'autre pas. Le shifté a donc un code ASCII supérieur de 128 à celui de l'autre. Sauf pour CHR$(150) et CHR$(16). Correspondant au CHR$(150), on aurait attendu CHR$(150 – 128 = 22), et on a CHR$(16). Mais $16, en hexadécimal, c'est 22 en décimal. De là à soupçonner en pro de chez Commodore d'avoir eu un moment d'inattention...

Si vous voulez passer à tab...

Attaché que je suis à ma vieille bête, et résolument opposé à la conception mercantile de l'ordinateur jetable dès qu'il est réputé démodé, je vous livrerai encore un des secrets du 4000 GE, le FAT-40 comme disent les anglo-saxons, avant de donner quelques tuyaux propres au VIC et au C.64. C'est que le FAT-40, ce n'est pas un gros 4000, c'est un petit 8000. Entre autres fioritures, il comporte des taquets de tabulation. Mais ça, Procep ne me l'a jamais signalé, que je sache. Essayez donc ceci :

```
100 PRINT"[42 ESPACES]"
110 PRINT"COL.1COL.2COL.3
    COL.4"
```

Maintenant, revenez à la ligne 100, déplacez le curseur de 12 espaces, et frappez la séquence RVS SHIFT I. Avancez de 6 espaces et frappez la même séquence, puis de 15 espaces et encore la même séquence. Sur la ligne 110, placez le curseur sur le deuxième C, et faites la séquence INST, ESPACE, CURSEUR GAUCHE, RVS, I, OFF. Cette séquence sera répétée pour le troisième et le quatrième C.

Que s'est-il passé ? La ligne 100 a posé en mémoire des taquets de tabulation, avec un "I" shifté inversé, correspondant au CHR$(137). Le "i" non shifté inversé de la ligne 110, CHR$(9), fait afficher les éléments qui les suivent au tab précédemment établi. Faites GOTO 110 autant de fois que vous voudrez, les colonnes seront séparées. Mais si vous faites RUN ou GOTO 100, un deuxième passage sur les taquets les annule et l'effet de tabulation disparaît.

Place aux jeunes

Que les explorateurs de VIC et autres C.64 ne m'en veuillent pas de les avoir fait attendre un peu, et s'ils ont sauté ce qui précède, qu'ils veuillent bien le parcourir quand même. Ils en retiendront la leçon suivante : chez Commodore, un certain nombre d'opérations importantes sont effectuées grâce à des caractères de commande ; ces caractères peuvent naturellement être désignés par un CHR$, mais ils peuvent être incorporés dans une chaîne de caractères faisant l'objet d'un PRINT. Au listage, ils se présentent alors comme des caractères inversés. Pour les caractères shiftés, comme sur les CBM, il faut se livrer à la gymnastique précédemment décrite afin de les obtenir à l'écran. Ce sera surtout utile pour le CHR$(142) (majuscules/graphiques). On peut aussi chercher à reproduire les caractères de couleur shiftés, pour mieux connaître son clavier.

Mais sur le VIC et le C.64, il existe une manière commode d'obtenir ceux qui ne sont pas shiftés : c'est la touche de contrôle (CTRL).

On pourra reproduire les mouvements de curseur avec d'autres touches que celles qui sont prévues pour. Par exemple : CTRL Q sort un curseur bas, CTRL T détruit le caractère précédent, CTRL S amène le curseur en haut et à gauche de l'écran, CTRL *point-virgule* reproduit un curseur droite, CTRL M, un retour-chariot. Les couleurs aussi utiliseront le même principe, puisque CTRL E donne le blanc, CTRL *sterling*, le rouge, CTRL *flèche haut*, le vert et CTRL =, le bleu.

Sur Vic 20 et C.64, pas de lignes non-jointives, mais c'est tout de même par CHR$(14) et CHR$(142) que l'on passe le plus commodément par programme de majuscules en minuscules et réciproquement. Or, ici, la "manip" est encore plus simple, puisqu'elle peut se faire dans la foulée. Il suffit de frapper CTRL N pour obtenir le "n" minuscule inversé.

Je n'ai donc pas trouvé de combinaison simple pour produire entre guillemets un caractère shifté inversé. Mais vous, qui êtes intelligents, vous avez peut-être la solution. Ou d'autres astuces. Pensez à nous les communiquer.

François J. BAYARD

Caractères de contrôle

Sur CBM 4000 GE/8000

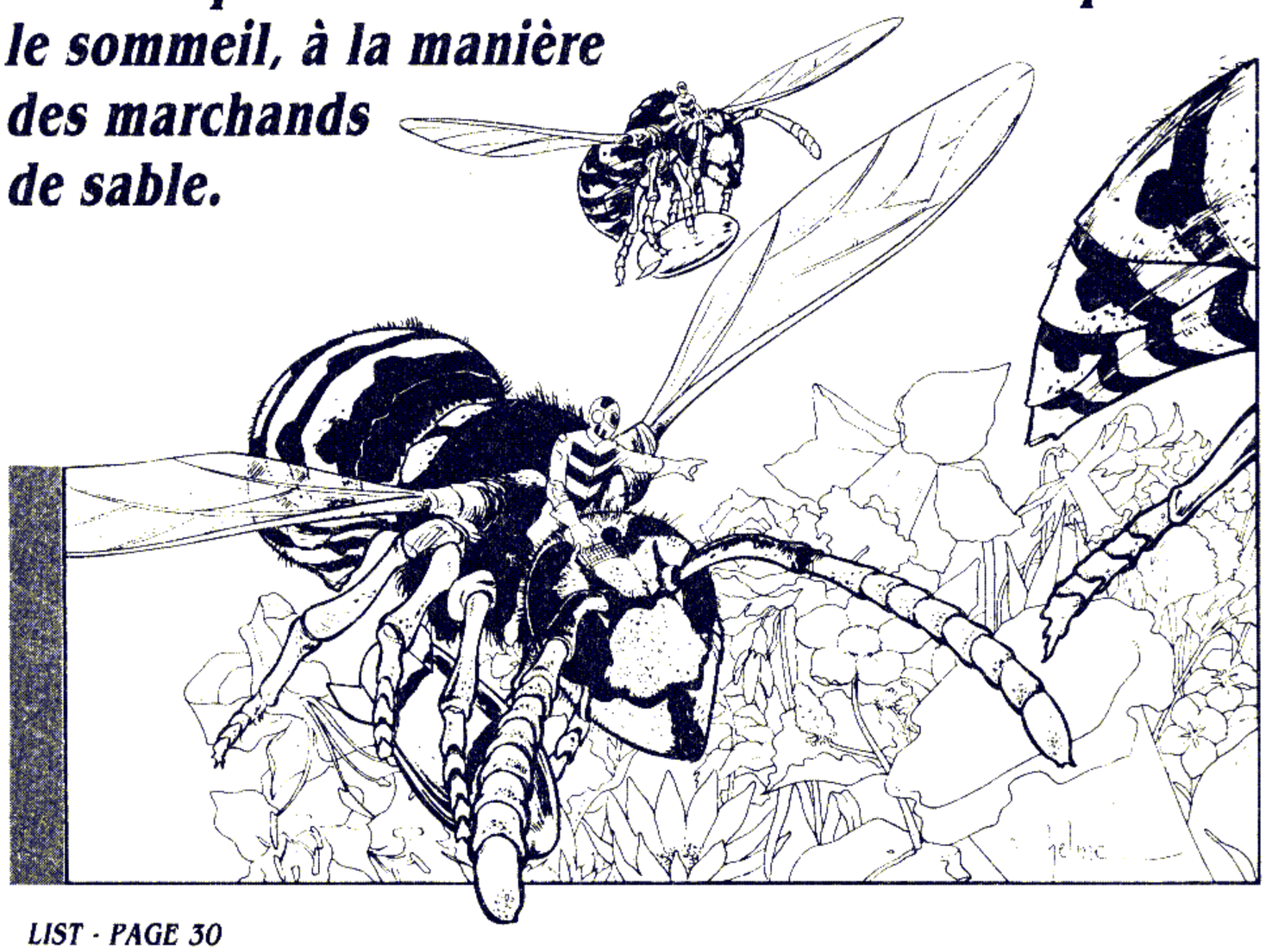
CHR$(14)		RVS N	Mode texte
CHR$(142)	SHIFT	RVS N	Mode graphique
CHR$(16)		RVS P	Efface la ligne après le curseur
CHR$(150)	SHIFT	RVS V	Efface la ligne avant le curseur
CHR$(9)		RVS I	Pose un taquet de tabulation
CHR$(137)	SHIFT	RVS I	Envoie au prochain tab
CHR$(7)		RVS G	Sonnerie simple
CHR$(135)	SHIFT	RVS G	Sonnerie double

Sur Vic 20 et C.64

CHR$(14)		CTRL N	Mode texte
CHR$(142)	SHIFT	RVS N	Mode graphique
CHR$(17)		CTRL Q	Curseur bas
CHR$(18)		CTRL R	Inversion vidéo
CHR$(19)		CTRL S	Home
CHR$(20)		CTRL T	Delete
CHR$(29)		CTRL +	Curseur droit
CHR$(5)		CTRL E	Blanc
CHR$(28)		CTRL £	Rouge
CHR$(30)		CTRL ↑	Vert
CHR$(31)		CTRL =	Bleu

LES MARCHANDS DE SABLE

Il ne faut pas toujours beaucoup d'octets pour concevoir un jeu amusant. Le programme Basic présenté ici en est une preuve. Dirigées depuis le clavier du PB-700, des guêpes survolent un champ de fleurs. Leur tâche consiste à répandre le sommeil, à la manière des marchands de sable.



Si les possibilités graphiques du PB-700 méritent d'être soulignées, il faudra tout de même faire preuve d'imagination pour voir des guêpes et des fleurs sur l'écran.

Les premières ont pour mission d'endormir les secondes. Elles y parviendront en lançant un petit jet de poudre, commandé depuis le clavier. En touchant la fleur, ce jet l'endort définitivement. Une simple pression sur une touche — autre que BREAK, CAPS, STOP ou SHIFT — suffit pour lâcher de la poudre. Mais un nouveau jet ne peut avoir lieu que si la poudre du précédent a touché le sol.

Dès qu'une fleur est touchée, la guêpe a rempli sa mission et le score s'affiche. Si elle rase le champ de fleurs, elle repart pour une nouvelle mission après avoir touché le sol. L'exercice est alors plus périlleux que le précédent, jusqu'à ce que disparition s'ensuive.

La guêpe qui disparaît à droite de l'écran reviendra à gauche au tour suivant. Et, sous l'effet du sable, de nouvelles fleurs plongeront dans un lourd sommeil.

André REEB-GRUBER

LIST - PAGE 30

Marchands de sable
Programme pour PB-700
Auteur André Reeb-Gruber

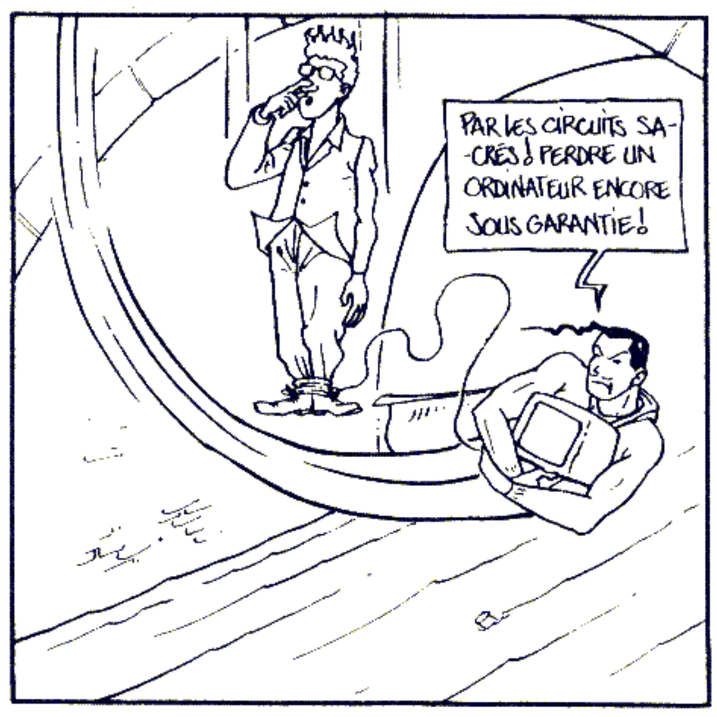

```
*----------      BLITZ      ----------*
5 CLEAR :CLS :M=8
10 DRAW(0,31)-(159,31):FOR I= 17 TO 137
STEP 6:Y=INT(RND*M)*2+2
15 FOR J=0 TO 4:DRAW(I+J,31)-(I+J,31-Y):
NEXT J:NEXT I:H=1:I=1
20 I=I+6:IF I>151 THEN I=7:H=H+2
25 IF H>28 THEN M=M+1:GOTO 10
30 DRAW(I-2,H-1)-(I-2,H)-(I+2,H)
35 IF POINT(I+4,H)=1 THEN 80
40 IF C<>0 THEN 65
45 IF INKEY$<>"" THEN D=5
50 GOTO 55+D
55 DRAWC(I-2,H-1)-(I-2,H)-(I+2,H):GOTO 2
0
60 C=1:N=H:V=I
65 S=S+1:N=N+2:DRAW(V,N+2):DRAWC(V-2,N)-
(V+2,N):DRAWC(V-2,N+1)-(V+2,N+1):BEEP
70 IF N=29 THEN C=0:D=0:BEEP 1
75 GOTO 55
80 FOR Z=1 TO 100:NEXT Z:BEEP 1:BEEP 1:B
EEP 1:PRINT "PERDU:score ->";S
85 FOR I=1 TO 300:NEXT I
90 IF INKEY$="" THEN 90 ELSE 5
```

Liste des variables

- **C** drapeau de « verrouillage » du jet de sable
- **D** drapeau de commande du lancer de jet
- **H** hauteur de la guêpe
- **I** compteur de boucle (lignes 10 et 15) et position horizontale de la guêpe
- **J** compteur de boucle
- **M** niveau de difficulté
- **N** hauteur du jet de sable
- **S** score
- **V** position horizontale du sable
- **Y** hauteur de la fleur
- **Z** compteur de boucle pour la temporisation.

Le programme ligne par ligne

Ligne 5 : initialisation des mémoires, de l'écran et du niveau de difficulté.
Lignes 10 et 15 : tracé du sol et des fleurs, initialisation des coordonnées de la guêpe.
Ligne 20 : calcul des nouvelles coordonnées de la guêpe ; si elle est arrivée à la limite de l'écran, elle réapparaîtra au tour suivant avec une altitude plus faible.
Ligne 25 : si la guêpe a fini de raser le sol, le niveau de difficulté augmente d'une unité et l'on repart pour endormir de nouvelles fleurs.
Ligne 30 : tracé de la guêpe.
Ligne 35 : si la guêpe touche une fleur, envoi en ligne 80, c'est perdu.
Ligne 40 : test sur le jet de sable, grâce à un drapeau (C) ; si C est à zéro, il n'y a pas de jet de sable en cours ; si C est à 1, le sable est en train de tomber.
Ligne 45 : test de la commande de tir grâce au drapeau D ; si D vaut zéro, la commande de tir du jet n'est pas activée ; si D vaut 5, la commande est activée.
Ligne 50 : branchement à une adresse dépendant de la valeur de D.
Ligne 55 : effacement de la guêpe et retour à la ligne 20.
Ligne 60 : mise à 1 du drapeau C, initialisation des coordonnées de la guêpe qui permettent de repérer le déplacement du jet de sable.
Ligne 65 : incrémentation du score : tracé du sable qui endort éventuellement une fleur en émettant un « bip ».
Ligne 70 : si le sable a complètement endormi une fleur, les drapeaux C et D sont mis à zéro ; un nouveau « bip » retentit.
Lignes 80 : temporisation, signal sonore, affichage du résultat.



CONSERVEZ VOS MONSTRES

APRÈS avoir fait apparaître de petits monstres à l'écran (LIST 5) et les avoir animés (LIST 6), voyons ce mois-ci comment les dessiner sur imprimante et les sauvegarder sur disquette ou sur cassette.

Le programme d'écran graphique pour Oric-1 et Atmos et ses extensions–gestion et animation des dessins (1) –étaient écrits de façon structurée. Il est donc facile d'y ajouter des fonctions supplémentaires, comme l'impression, le chargement ou la sauvegarde sur disquette ou sur cassette de nos dessins. Pour greffer une nouvelle fonction sur le programme, nous devrons, outre la fonction elle-même, introduire sa reconnaissance en début de commande, et son mnémonique dans le bandeau de saisie en fin de commande. Pour l'impression, par exemple, cela nous donne :

```
92   IF CIMPR THEN GOSUB 2500
291  CIMPR = (A = 9 AND NOT
     SEUL)
2840 PRINT "...Ani Lis ICS" ;
2500 PRINT "Impression...
(...)
2595 RETURN
```

Nos idées directrices seront :

- créer un petit système d'exploitation des dessins entre la mémoire et la disquette de l'Oric, en particulier pour mélanger les dessins ;
- n'imprimer que le dessin, et non pas l'écran en entier : il faut tenir compte de la lenteur de l'impression en Basic.

Quelques points particuliers méritent explication. Ainsi, on peut se demander comment utiliser la touche FUNCT du clavier de l'Atmos pour reconnaître, par exemple, une de nos fonctions supplémentaires. Il faut savoir qu'il s'agit d'une touche de fonction, comme Shift et Contrôle, c'est-à-dire que l'appui sur cette touche modifie temporairement l'octet 521 de la mémoire mais ne produit aucun caractère. Cet octet doit donc être lu en même temps qu'un caractère. Il vaut 165 ou 56 selon qu'il y a ou non appui sur FUNCT. D'où GET A$: T0UCHE = PEEK (521) ; attention, il faut écrire "T-zéro-UCHE" pour éviter le mot réservé "TO".

Une impression ligne à ligne

La séquence Ctrl-I permet l'impression d'un dessin sur une imprimante Seikosha GP80 ou GP100. On envoie des colonnes de 6 points de large pour imprimer des lignes de 6 (ou 7) points de haut. Le mode graphique commence toujours avec le code 8 ; l'impression de chaque ligne est commandée par le code 13. Chaque caractère occupe 7 points graphiques sur l'imprimante contre 6 seulement sur l'écran. La solution consiste ici à donner la valeur du point 6 au point 7, le point 8, quant à lui, restant toujours à 1. Les codes de ces trois derniers points sont respectivement 32, 64 et 128.

L'octet 598 (voir ligne 2510) indique le nombre de colonnes de l'imprimante que l'on fait coïncider avec le nombre de lignes graphiques de l'écran : 200 donc. Pour l'Oric-1, il faudra mettre 255 dans l'octet 49. Dans tous les cas, il ne sera imprimé que le dessin sans cadre, et non pas les 48000 points de l'écran.

Avec Funct-C et Funct-S, nous abordons maintenant le chargement et la sauvegarde sur disquette. La gestion des dessins en mémoire centrale demande un adressage variable pour le chargement depuis le lecteur de disque. Les dessins chargés peuvent être ajoutés à l'ensemble de ceux qui sont déjà présents dans la mémoire, et donc mélangés dans une animation. On ne gardera dans un même fichier que ceux qui demandent à être animés ensemble. Cela explique la ligne :

```
63998 ! LOAD FIC$, A#+ + + + :
      RETURN
```

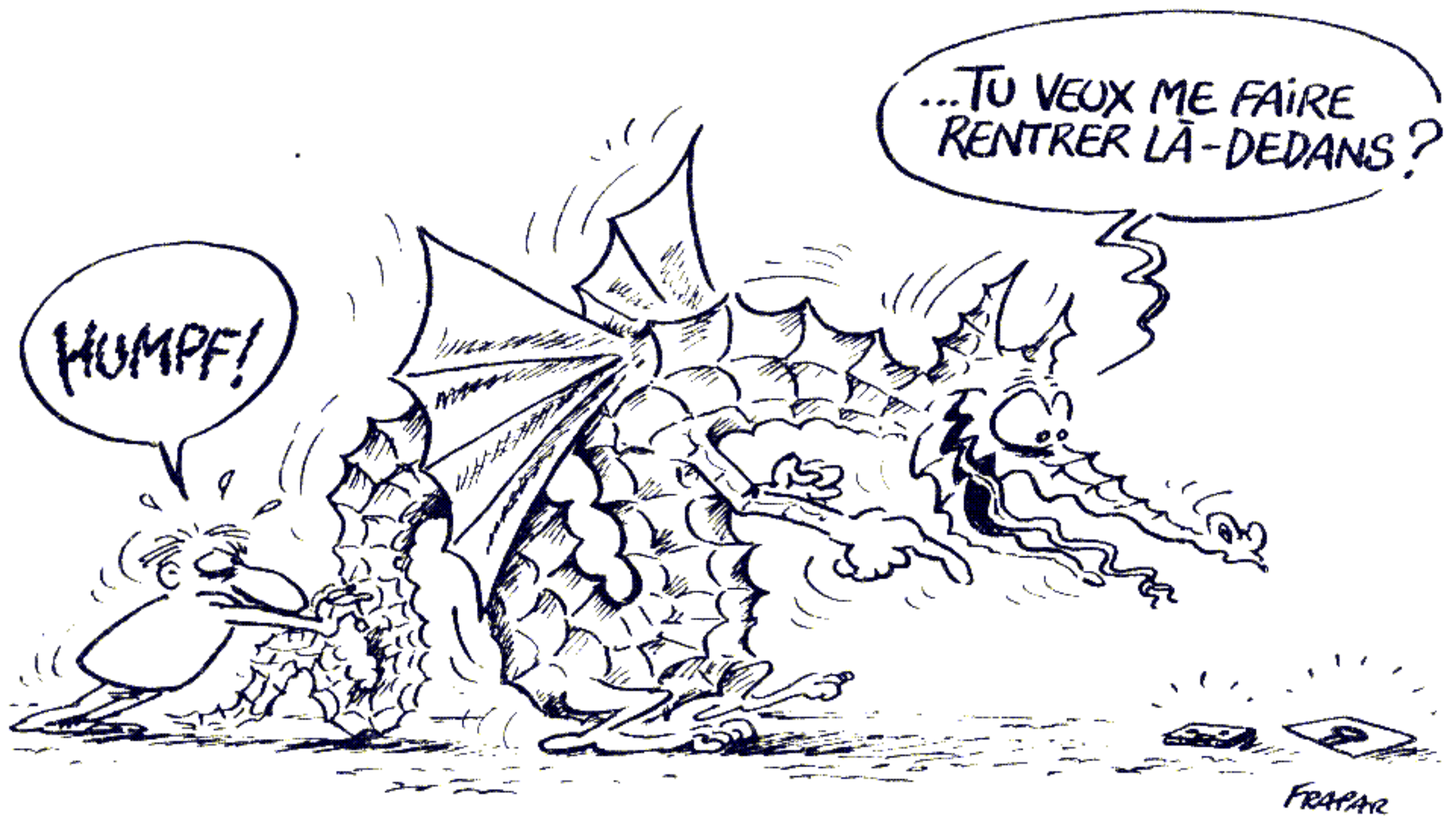
De la même façon, il ne faut sauvegarder que la partie occupée par les dessins, d'où :

```
63999 !SAVE FIC$, A#5001,
      E#+ + + + : RETURN
```

Dans ces deux dernières lignes, les signes "+" qui correspondent à l'adresse de chargement seront automatiquement remplacés par l'adresse exacte des dessins. Le principe retenu ici consiste à placer les ordres à modifier (LOAD et SAVE) très près des limites du programme, limites que l'on connaît grâce aux pointeurs 154 (début) et 156 (fin du programme Basic). Comme on le voit, nous avons choisi la fin, le numéro de ligne le plus élevé en Basic de l'Oric étant 63999. On compte donc les octets en commençant par la fin :

---

(1) Voir LIST *nos 5 et 6.*



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


```
0 REM ** ecran graphique **
1 REM ----------------------
5 GOSUB 100 'repete les commandes
10 REPEAT :GOSUB 20 :UNTIL NOT COMMAN
15 GOSUB 2900 'fin =======
20 GOSUB 200  'commande :
...
92 IF CIMPR THEN GOSUB 2500
94 IF FCARG THEN GOSUB 2600
96 IF FSAUV THEN GOSUB 2700
99 GOSUB 2800 :RETURN :REM =====

100 REM initialisations
...
155 PRINT"ICS:ctrl-Impres funct-Char
funct-Sauv";
...
200 REM debut une commande
...
288 :SEUL=(TOUCHE=56)
289 MOUV=(MOUV AND SEUL) :CERC=(A$="C
" AND SEUL) :SUPP=(A$="S" AND SEUL)
291 CIMPR=(A=9 AND NOT SEUL)
292 :FUNCT=(TOUCHE=165)
293 FCARG=(A$="C" AND FUNCT)
294 FSAUV=(A$="S" AND FUNCT)
...
2500 PRINT"Impression de la forme";
2505 E=0 :GOSUB CADR
2510 POKE 598,200
2515 LPRINT CHR$(8)
2520 PNT=40960+Y0*40+INT(X9/6)
2525 LARG=INT(X9/6)-INT(X0/6)+1
2530 FOR COL=PNT TO PNT-LARG STEP -1
2535 FOR LIG=COL TO COL+LY*40 STEP 40
2540 CODE=PEEK(LIG)
2545 IF CODE=)128 THEN CODE=CODE-128
2550 IF CODE=)64 THEN CODE=CODE-64
2555 IF CODE=)32 THEN CODE=(CODE+192)
ELSE CODE=(CODE+128)
2560 LPRINT CHR$(CODE);
2565 IF KEY$=CHR$(27) THEN 2585
2570 NEXT LIG
2575 LPRINT CHR$(13)
2580 NEXT COL
2585 POKE 49,40
2590 E=1 :GOSUB CADR
2595 RETURN :REM -----
2600 INPUT "Charge les formes ";FIC$
2610 IF FIC$="" THEN RETURN
2620 FIC$=RIGHT$("  "+FIC$+".DES",7)
2630 DES$="z" :GOSUB 3230 :REM limite
2660 POKE 1277,1
2670 :GOSUB 63000 'modif load
2680 POKE 1277,0
2690 IF PEEK(1279)=1 THEN PRINT"incon
nu";:PING:GET A$
2695 RETURN :REM -----

2700 INPUT "Sauve les formes ";FIC$
2710 IF FIC$="" THEN RETURN
2720 FIC$=RIGHT$("  "+FIC$+".DES",7)
2730 DES$="z" :GOSUB 3230 :REM limite
2760 POKE 1277,1
2770 :GOSUB 63100 'modif save
2780 POKE 1277,0
2790 IF PEEK(1279)=9 THEN PRINT"exist
e deja";:PING:GET A$
2795 RETURN :REM -----

2800 REM fin une commande
...
2840 PRINT"Txt 'EPYCo For Gar Pla Sup
Ani Lis ICS";
2870 GET A$ :TOUCHE=PEEK(521)
...
63000 REM load: adresse debut
63010 A=DEEK(156)-42
63020 FOR I=2 TO 5 :Z=ASC(MID$(HEX$(A
DRS),I,1)) :POKE A+I,Z :NEXT
63030 :GOTO 63998 'load

63100 REM save: adresse fin
63110 A=DEEK(156)-11
63120 FOR I=2 TO 5 :Z=ASC(MID$(HEX$(A
DRS+2),I,1)) :POKE A+I,Z :NEXT
63130 :GOTO 63999 'save

63998 !LOAD FIC$,A#++++:RETURN
63999 !SAVE FIC$,A#5001,E#++++:RETURN
```

A = DEEK (156) — 11 et A = DEEK (156) — 42 ; puis l'on fait poker par le programme les 4 caractères de l'adresse (en hexadécimal) à l'endroit voulu, autrement dit à la place des "+ + + +".

Normalement, il vaut mieux s'interdire de telles acrobaties en programmation, car il est difficile de contrôler le processus ou même de modifier le programme par la suite. Ici, nous n'avions pas le choix.

Disquette ou cassette ?

Le système d'exploitation des disquettes utilise l'octet 1277 pour autoriser (valeur 1) ou non (valeur 0) la détection d'erreur sans arrêt du programme, et il place le numéro de l'erreur dans l'octet 1279. On peut ainsi gérer soi-même les erreurs : fichier inconnu ou déjà existant (lignes 2690 et 2790).

En l'absence de lecteur de disquette, on stockera les dessins sur une cassette. Les modifications à apporter au programme sont mineures. Sur Atmos, on ne peut changer l'adresse de chargement puisque l'ordre CLOAD n'admet pas d'adresse, on écrira donc :

```
(...)
2660 PRINT "Cassette prete O"
     H$;:GET O$:PRINT O$;: IF O$
     < > "O"THEN RETURN
2670 CLOAD FIC$
2695 RETURN : REM---
```

et il sera malheureusement impossible de mélanger deux ensembles de dessins.

Pour utiliser l'ordre CSAVE, on écrira :

```
2760 PRINT "Cassette prete O" H$;:
     GET O$: PRINT O$;: IF
     O$ < > "O" THEN RETURN
2770 : GOSUB 63100 'modif save
2795 RETURN :REM-----
63100 REM save: adresse fin
63110 A = DEEK(156) – 11
63120 FOR I = 2 TO 5 :Z = ASC(MID$
     (HEX$(ADRS),I,1)) :POKE
     A + I,Z :NEXT
63999 CSAVE FIC$,A#5001, E#
     + + + + :RETURN
```

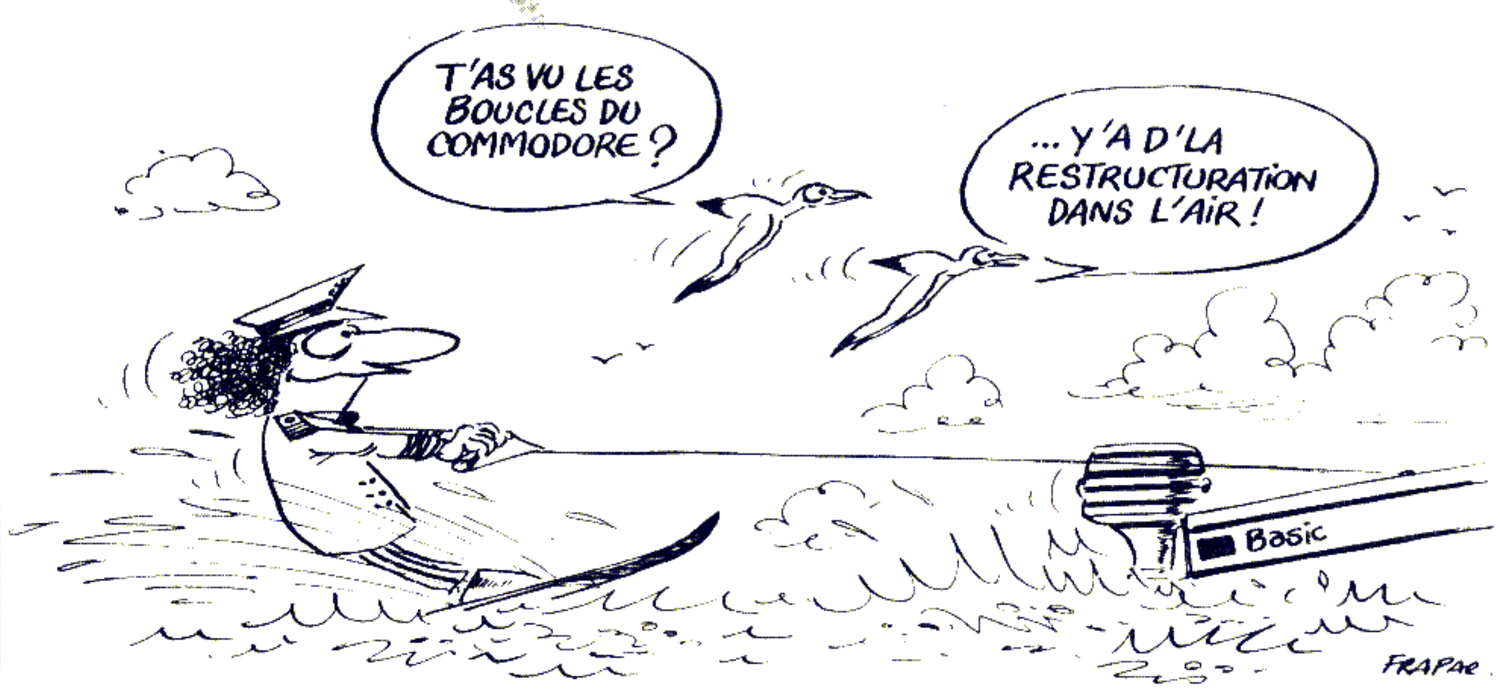
Sur l'Oric-I, on peut faire comme sur l'Atmos, mais le programme est interrompu après chaque chargement. On doit rectifier le pointeur 156, qui est altéré par l'exécution du CLOAD, en y plaçant la valeur du pointeur 158 : DOKE 156, DEEK (158) puis RUN, et le fichier est chargé.

Max HAGENBURGER

LES BOUCLES STRUCTURÉES SE FONT ATTENDRE

POUSSÉS par une saine curiosité, nous avons cherché à connaître les performances en vitesse du nouveau Commodore 16, liées à l'usage des structures de boucles qu'offre son Basic élaboré. Vous l'allez voir, tout ne va pas pour le mieux au pays de la structuration. Quelques conclusions s'imposent.

A la décharge de notre valeureux cobaye, il faut dire que nous avons utilisé ses structures de boucle là où elles n'ont pas lieu d'être employées, c'est-à-dire là où elles sont les plus inefficaces, donc les plus lentes. Le centre de nos recherches fut un simple programme de comptage, c'est dire si nous avons fait preuve d'originalité !



Les boucles du Basic 3.5

Le Basic 3.5 équipe les nouveaux Commodore 16 et +4. C'est un Basic complet qui est au Basic 2.0 des Vic et Commodore ce qu'est la Sixtine au regard de l'église du village. Il offre en particulier des structures de boucles qui éviteront aux programmeurs du dimanche de se perdre dans des GOTOs à rallonges.

DO...LOOP est le maître-mot des structures. Il peut s'agrémenter, selon les besoins, de UNTIL ou de WHILE ; tandis que le bon vieux FOR...NEXT reste vaillant au poste. Pour les amateurs de sorties de boucles en catastrophe, EXIT apporte un secours bienveillant et salutaire.

Nous disions donc qu'un programme de compteur avait servi de base à toute cette affaire. Par simple mauvais esprit, nous avons chronométré le désastre que constitue l'utilisation d'une structure « manuelle ». Résultat, 101.26 secondes pour ce programme :

```
10 I=I+1
20 IF I<10000 GOTO 10
```

Pas brillant donc (1). Mais rappelons que ce type de compteur devait être utilisé à chaque fois qu'une boucle FOR...NEXT pouvait être interrompue par un GOTO branchant à l'extérieur...

Dans la foulée, nous obtenions 18.8 secondes pour le programme suivant, qui utilise la structure la plus adéquate en un tel cas :

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

Durée même réductible selon les variantes du programme : NEXT I, ou NEXT.

Une fois connus ces deux extrêmes, il était facile de voir ce qu'allaient donner, par comparaison, les diverses versions de la chose.

```
10 DO
20 I=I+1
30 LOOP WHILE I<10000
```

...exige 96.6 secondes. C'est long.

```
10 DO WHILE I < 10000
20 I=I+1
30 LOOP
```

(1) A titre de comparaison, le même programme s'exécute en 103 secondes sur C.64 et 33 secondes sur Amstrad.

...prend 96.42 secondes. Ce n'est guère mieux.

```
10 DO
20 I=I+1
30 LOOP UNTIL I=10000
```

...demande 95.26 secondes. Hum !

```
10 DO UNTIL I=10000
20 I=I+1
30 LOOP
```

...consomme 94.9 secondes. Aïe !

```
10 DO UNTIL I>9999
20 I=I+1
30 LOOP
```

...est le meilleur, avec 91.67 secondes.

Bref, aucune de ces boucles ne vient à bout de ses 10000 itérations en un temps comparable à celui d'un FOR....NEXT. Heureusement, toutes s'exécutent en moins de temps qu'une boucle manuelle, mais il s'en faut de peu.

Dans un premier temps, nous pouvons penser que cette relative lenteur provient du Basic 3.5 lui-même, qui peut-être n'est pas optimisé à ce niveau. Dans un second temps, il est possible de considérer que le 7501 (nouvel avatar du 6502, et cerveau du C.16) n'est pas tout à fait à la hauteur de la tâche qui lui est demandée. Pensez, un cœur qui ne bat qu'à 1 MHz environ !

Mais, dans un troisième temps, il est sage de faire avec ce qu'on a. Le Basic 3.5 reste un très bon Basic et le plaisir d'écrire de façon structurée n'a pas de prix.

Notons donc que l'usage du UNTIL est légèrement plus rapide que le WHILE. Observons que DO UNTIL est meilleur que LOOP UNTIL, ce qui est vrai également du WHILE. Et enfin, tirons parti du fait que les inégalités sont traitées plus vite que les égalités, comme l'indiquent les deux derniers tests.

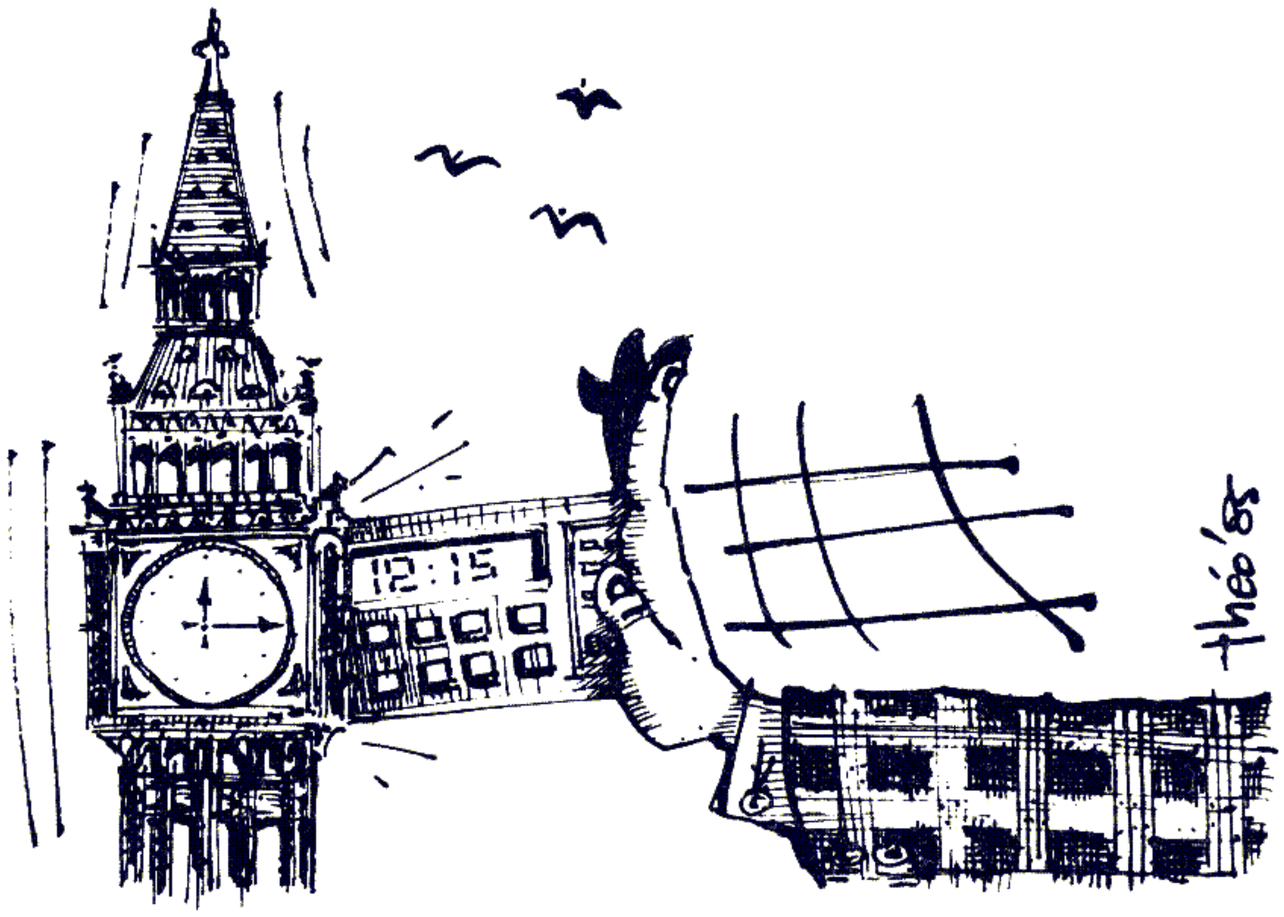
Qui nous donnera une idée des temps d'exécution de ces mêmes structures sur une autre machine qui en dispose ?

Robin BOIS

LA FONCTION TIME DU PC-1500

UNE PENDULETTE DE POCHE

LE Sharp PC-1500 est un ordinateur réellement portatif puisqu'il tient aisément dans la poche d'un manteau. La tentation est grande de l'employer comme agenda électronique, calendrier et (pourquoi pas ?) réveille-matin.



Certaines applications nécessitent une connaissance correcte des nombreux problèmes liés à la gestion du temps et donc à l'utilisation de la fonction spécialisée TIME. Son étude très minutieuse — allant au-delà des modestes déclarations du manuel — permet une utilisation optimale de sa puissance.

La mémoire vive du PC-1500 est de type CMOS. Entre autres cela implique que des informations particulières y sont conservées — voire mises à jour — même lorsque la machine est « éteinte ». Ceci permet au quartz de l'ordinateur d'entretenir en permanence une horloge dite *temps réel*. En fait, c'est plus un horodateur car l'heure et la date du jour y sont gérées.

Grâce au langage Basic, on accède à ce dateur-horloge par la fonction TIME dédiée à la gestion du temps. On l'initialise en y introduisant les informations suivantes : le mois (MM), le jour (JJ), l'heure (HH), les minutes (mm) et les secondes (ss). Une telle mise à l'heure s'effectue simplement sous la forme d'un nombre décimal comme on affecte

LA FONCTION TIME DU PC-1500

UNE PENDULETTE DE POCHE

une valeur à n'importe quelle variable : TIME = MMJJHH.mmss

Ainsi définie, la variable spéciale TIME est constamment entretenue par l'ordinateur. On en consulte le contenu le plus naturellement possible par : PRINT TIME ou encore A = TIME : PRINT A

La documentation de l'ordinateur est bien obscure quant au comportement détaillé de cette gestion du temps. Une erreur de type 23 signale un réglage incorrect de TIME mais cela peut être le résultat de causes très différentes ! Par ailleurs, lorsqu'on souhaite traiter la valeur fournie par TIME pour en extraire, par exemple, une date, il est commode de procéder à partir d'une chaîne de caractères. Or TIME ne fournit pas un nombre d'un format régulier : les zéros non significatifs sont ignorés à gauche comme à droite. Par exemple, un mois de mars, au 23e jour, à 15 h 34 mn et 20 s retournerait : 32315,342.

Carte de la variable TIME

Chiffres 1 et 2	mois de 1 à 12. Si le mois 0 est admis, une valeur supérieure à 12 provoque une erreur.
Chiffres 3 et 4	quantième du jour de 1 à 31. Au-delà de 31 et jusqu'à 99 (ainsi que pour 0) aucune erreur n'est détectée. On s'aperçoit que le PC-1500 connaît le nombre de jours de chaque mois.
Chiffres 5 et 6	heure de 0 à 23. Toute valeur supérieure à 23 est forcée à 0 sans message d'erreur. Dès minuit, le jour et, éventuellement, le mois sont incrémentés (+1).
Chiffres 7 et 8	minutes de 0 à 59. Les valeurs de 60 à 89 sont traduites en 0 à 9, tandis que pour 90 à 99 le premier chiffre est 1 et le second reste inchangé.
Chiffres 9 et 10	secondes de 0 à 59. Toute valeur à partir de 60 incrémente le compteur des minutes.

Un algorithme simple

C'est ainsi que, connaissant un peu mieux la variable TIME, nous allons écrire une fonction produisant, à partir de sa valeur, une chaîne de caractères en contenant toujours dix :
— il y a, au plus, six chiffres avant le point décimal,
— il en reste, au plus, quatre à gauche du même point décimal.

L'algorithme qui se dégage est très simple. Il faut séparer la partie entière du nombre (MMJJHH) avec STR$ INT TIME car les parenthèses sont superflues. A l'inverse, la partie décimale (mmss) sera obtenue par STR$ (TIME-INT TIME).

Puis, on ajoutera à gauche du premier résultat, ainsi qu'à droite du second, le nombre de caractères « 0 » nécessaire pour les compléter respectivement à six et à quatre chiffres. Ainsi, en employant une chaîne intermédiaire de cinq chiffres « 00000 » (ce qui garantit le complément dans tous les cas), on écrira C$ + STR$ INT TIME pour la partie entière, et STR$ (TIME - INT TIME) + C$ pour la partie décimale.

Enfin, dernier stade des manipulations, il faut tronquer ces chaînes de caractères, maintenant trop longues, ce qui donne :
RIGHT$(C$ + STR$ INT TIME ,6) + MID$(STR$ (TIME -INT TIME) + C$,3,4)

Horodateur
Programme pour PC-1500
Auteur Yul Pham Duy


```
10: A$="84":R=0:K$
    ="":C$="00000"
    :WAIT 0
20: "HORDAT"DATA
    RIGHT$ (C$+
    STR$ INT TIME
    ,6)+MID$ (STR$
    (TIME -INT
    TIME )+C$,3,4)
30: DATA MID$ (T$,
    3,2)+"/"+LEFT$
    (T$,2)+"/"+A$
40: DATA MID$ (T$,
    5,2)+"H"+MID$
    (T$,7,2)+" "+
    RIGHT$ (T$,2)
50: RESTORE "HORDA
    T":READ T$,D$,
    H$:PRINT D$;"
      ";H$
60: IF R$=RIGHT$ (
    T$,6)LET R=1
70: BEEP R,200,255
    :K$=INKEY$
80: IF K$="R"AND R
    =0INPUT "HEURE
     D/ALARME: ";R
    $
90: IF K$<>""LET R
    =0:K$=""
100: GOTO 50
```

L'heure en poche

En partant du principe de découpage détaillé dans l'article, quelques lignes de Basic suffisent à programmer une horloge doublée d'une alarme sonore. Le programme en fonctionnement affiche la date et l'heure. En pressant la touche R (comme Réveil) on introduira l'heure de l'alarme. Cette dernière sera interrompue sur simple pression d'une touche du clavier.

Le programme emploie une possibilité du Basic ignorée par le manuel du constructeur (1) : la conservation de fonctions à calculer dans les lignes de DATA. Ainsi, à la manière d'un sous-programme implicite, les DATA constituent un moyen commode d'évaluation d'expression avec un appel précisé d'un simple RESTORE "nom" et sans se préoccuper du retour de la procédure (automatique avec READ).

(1) Voir l'ouvrage Basic, système et langage-machine sur PC-1500, *de Jean-Christophe Krust, Ed. PSI 1984.*

Yul PHAM DUY

LE BASIC DE L'EXL 100

Il va bientôt falloir plusieurs mains pour compter les ordinateurs français sur ses doigts. L'EXL 100, un des derniers venus, apporte une bonne dose de nouveauté. Et son Basic, s'il n'est pas très standard, n'en présente pas moins des caractéristiques intéressantes et originales.

Liaison par infrarouges : le clavier prend des libertés

La conception de l'EXL 100, par la CGCT (Compagnie Générale de Constructions Téléphoniques), a été pensée avec un grand sens pratique : le nombre de fils est réduit au minimum. Il en subsiste précisément trois, celui de l'alimentation (incluse dans le boîtier), le cordon Péritel et le raccordement au lecteur de cassettes (sans télécommande, malheureusement).

La conversation entre le clavier ou les poignées de jeu et l'unité centrale (bloc indépendant) se fait par clins d'œil invisibles. Entendez : le dialogue est assuré par liaisons infrarouges. Il n'est donc plus nécessaire de rester attaché à l'ordinateur, on peut travailler — ou jouer — confortablement installé sur un canapé.

Pour les jeux, des cartouches enfichables sont disponibles. Certaines utilisent très judicieusement le synthétiseur sonore inclus dans l'unité centrale.

La bosse des maths

Le Basic se présente, lui aussi, sous la forme d'une cartouche de mémoire morte. L'absence de langage résident garantit la souplesse de l'interpréteur. Et on peut espérer que d'autres langages se présenteront sous cette forme, ou même que l'actuelle version du Basic sera corrigée pour combler certaines lacunes.

Dès l'ouverture de la notice, les origines du Basic se révèlent connues : une profusion de CALL nous rappelle le langage d'un certain CC-40 de Texas Instruments. Il s'agissait d'un ordinateur de "cartable", disparu peu après sa naissance (vers juin 1983), et qui avait pourtant l'aspect sérieux.

La version de l'EXL 100 laisse un peu à désirer dans les domaines graphique et sonore. Mais, pour le reste, c'est du grand art : des instructions riches et puissantes et une force de calcul que n'ont pas toujours les « gros ».

Tout commence très fort avec les calculs directs au clavier. Pas besoin de l'habituel PRINT ou d'un point d'interrogation pour afficher les résultats d'une quelconque opération. On les écrit directement, comme sur une cal-

LE BASIC DE L'EXL 100

culatrice ou un ordinateur de poche. La touche SHIFT opère en bascule et un indicateur apparaît en haut de l'écran, sur une ligne d'état. L'appel de contrôle, de fonction et de mode angulaire (RAD, DEG ou GRA) ont, eux aussi, leurs indicateurs (encore des airs d'ordinateur de poche). La touche de fonction sert à introduire rapidement les mots clés du Basic qui sont inscrits sur un cache-clavier.

Les calculs se font sur 14 chiffres, 10 seulement étant affichés. Pour vérifier ou établir de telles informations, j'utilise un truc : l'exécution d'un test d'égalité sur une division qui ne tombe pas juste. Sur l'EXL 100, par exemple, j'ai frappé au clavier :

PRINT 2/3=0.66666666666667

L'instruction PRINT est facultative ici. Le résultat apparaît : −1, soit vrai. Il indique que cette égalité, dont le deuxième terme est formé de 14 chiffres, est vérifiée alors qu'avec moins de chiffres significatifs, elle ne l'était pas. Et pour connaître le nombre de décimales affichées, il suffit de taper PRINT 2/3. Ici, le résultat est 0.6666666667. Soit 10 chiffres significatifs. C'est donc une bonne précision dans les calculs décimaux. En notation scientifique, ça ne se passe pas mal non plus : les calculs couvrent la gamme de 1E-128 à 9E+127. Beaucoup d'autres ordinateurs sont encore limités à 1E99, quand ce n'est pas 1E38.

Dans les programmes, les nombres sont destinés à être manipulés grâce à des variables. L'EXL 100 accorde une grande souplesse à leur nom de baptême. Chacun peut s'étendre jusqu'à 15 caractères alphanumériques. On ne sera donc pas limité dans les identificateurs et les noms de variables pourront être vraiment parlants. Dans un même programme, le nombre d'identificateurs est limité à 95.

Les variables alphanumériques atteignent leur trop-plein à 75 caractères. Pour en afficher davantage, il faut concaténer en utilisant le signe "&", au lieu de l'habituel ";". Mais on ne peut pas dépasser trois juxtapositions si les variables sont remplies au maximum.

Il n'y a pas que la concaténation qui se pratique de manière originale. Le « saucissonnage » des chaînes de caractères a lieu grâce à SEG$. Il remplace à lui seul les trois trancheurs classiques : LEFT$, MID$ et RIGHT$. Ce qui revient à dire qu'il agit exactement comme MID$, les deux autres n'ayant pas d'équivalent.

On peut créer ses caractères

Les chaînes de caractères sont aussi traitées par RPT$ pour la répétition et POS pour la recherche de « maillons » dans une chaîne. La conversion d'alphanumérique en numérique est possible par VAL. Elle est complétée par NUMERIC qui vérifie la validité numérique d'une chaîne.

Les caractères accentués de la langue française existent sur le clavier, mais ils ne sont pas reconnus par le Basic qui ne sait les utiliser qu'à partir de leur code (non standard) avec CHR$. Par ailleurs, CALL CHAR donne la liberté de créer des caractères quelconques par codage hexadécimal d'une matrice de 8 sur 10 points.

Avant de rentrer plus avant dans l'examen des instructions, il faut préciser que les lignes d'un programme sont numérotées en-dessous de 32766 (codage 15 bits), et qu'un espace est obligatoire après le numéro. C'est cet espace qui distingue le numéro de ligne d'un programme du premier terme d'une opération.

Après chaque instruction, un autre espace est obligatoire, sauf en cas de parenthèse. Ce dernier point est plus gênant car la plupart des Basic s'en dispensent. Une ligne ne peut dépasser 80 caractères. Mais il est vrai qu'avec des lignes plus longues, les programmes sont peu lisibles.

Les fonctions mathématiques se montrent à la hauteur de la précision numérique. Un jeu complet de fonctions trigonométriques s'attaque aux angles, dans les trois modes disponibles (degrés, radians, grades). C'est plutôt rare.

Les logarithmes népériens n'ont pas éclipsé les logarithmes décimaux, si souvent oubliés dans les Basic. Les subtilités du générateur de nombres aléatoires feront le délice des inventeurs de jeux : la fonction RND s'accompagne de RANDOMIZE qui initialise une séquence vraiment aléatoire et qui, si elle est suivie d'un paramètre de 1 à 65535, déclenche la production de séquences reproductibles. Enfin, la fonction INTRND(n) délivre des nombres aléatoires entiers compris entre 1 et n, évitant ainsi de passer par les formules d'arrangement à partir d'un nombre fractionnaire.

Les instructions d'introduction de données à partir du clavier foisonnent. Par exemple, KEY$ et CALL KEY surveillent l'appui des touches. Avec la seconde, on remplit deux variables, l'une contenant le code ASCII de la touche enfoncée, l'autre l'indicateur d'état qui précise si la même touche a été

Liste des mots clés du Basic de l'EXL 100

ABS
ACCEPT
ACS
AND
ASC
ASN
ATN
ATTACH
BREAK
CALL
CHAR*
CHR$
CLEAN UP*
CLOSE
CLS
COLOR*
CONTINUE
COS
DATA
DEG
DELETE
DIM
END
ERR*
EXEC*
EXP
FOR...STEP...NEXT
GOSUB
GOTO
GRAD
HRON*
HROFF*
IF...THEN...ELSE
IMAGE
INPUT
INT
INTRND
KEY1*
KEY2*
KEY$
LEN
LET
LINE*
LINPUT
LIST
LN
LOAD
LOCATE
LOG
NEW
NOT
NUMBER
NUMERIC
ON BREAK
ON ERROR
ON...GOSUB
ON...GOTO
ON WARNING
OPEN
OR
PAUSE
PEEK*
PI
PLOT*
POKE*
POS
PRINT
PRINT TAB
PRINT USING
RAD
RANDOMIZE
READ
RELEASE
REM
RENUMBER
RESTORE
RETURN
RND
RPT$
RUN
SAVE
SEG$
SGN
SIN
SPEECH*
SQR
STOP
STR$
SUB
SUBEND
SUBEXIT
TAN
UNBREAK
VAL
VERSION*
XOR

* Ces mots doivent être précédés par CALL (par exemple : CALL CHAR, CALL POKE,...)

actionnée plusieurs fois. L'instruction ACCEPT se comporte comme un INPUT intelligent. Elle est suivie de précisions de contrainte : BEEP (un signal sonore d'attente), VALIDATE (une restriction du type de caractères acceptables — alphabétiques, majuscules, numériques, alphanumériques, etc.) ou NULL (définissant la valeur introduite si seul un retour chariot est enfoncé).

Le PRINT est bien loti, lui aussi. Il gère les formats de sortie avec USING, complété par IMAGE qui prédéfinit un format sur une ligne de programme en y incluant éventuellement du texte. PAUSE suspend l'affichage pendant une durée déterminée en évitant le recours à des boucles de temporisation et LOCATE précise les coordonnées d'apparition à l'écran.

Fiche technique de l'EXL 100

Constructeur : *Compagnie Générale de Constructions Téléphoniques*
Distributeur : *Exelvision*
Prix public : *2 690 FF (comprend l'unité centrale, le clavier, la cartouche de Basic et les cordons)*
Processeur : *TMS 7020*
Mémoire vive disponible : *32 Koctets*
Mémoire morte : *8 Koctets*
Langage : *Basic (cartouche de mémoire morte de 32 Ko)*
Variables numériques : *de 1E-128 à 9E+127*
Précision : *calculs sur 14 chiffres, 10 chiffres affichés*
Nombre de mots clés du Basic : *97*

Pas de mode « trace »

Dans le jeu d'instructions, on trouve aussi ELSE, complémentaire de IF...THEN, ON...GOTO et ON... GOSUB. Un original ON BREAK décide de la marche à suivre en cas d'interruption de l'exécution, et BREAK, seul, se comporte comme une instruction à part entière en définissant des points d'arrêt que UNBREAK supprime. En l'absence de mode Trace, cela aide à la mise au point. On trouve encore un ON ERROR et même un ON WARNING pour traiter les erreurs moins graves.

De nombreuses instructions sont gérées comme des procédures par le Basic et sont appelées par CALL. On y trouve la maigre panoplie servant à réaliser des graphismes : PLOT qui opère point par point, ou LINE qui trace une ligne droite. Il n'est donc pas simple de dessiner de beaux ronds ou de remplir des figures. Et pas un seul petit lutin ne se promène à l'écran ! Le Basic de l'EXL 100 n'a pas assez évolué dans ce domaine par rapport à celui du CC-40.

Le domaine des sons est encore plus désertique : aucune possibilité de création musicale. L'exploitation du synthétiseur vocal incorporé s'apparente à un exploit. Le CALL SPEECH précède des files de caractères hexadécimaux dont le rôle n'est pas précisé par la notice. Il faut explorer à tâtons, après avoir corrigé l'erreur du seul exemple de programme livré par le manuel.

D'autres CALL autorisent le discours en langage-machine (CALL PEEK, POKE, EXEC). Mutisme de la notice ici encore. Elle recommande simplement de se référer à un bon ouvrage sur la programmation du microprocesseur TMS 7020 et c'est plutôt difficile (voire impossible aujourd'hui) à trouver.

Comme pour se faire pardonner ces sérieuses lacunes, CALL peut appeler des procédures définies par l'utilisateur. Celles-ci sont créées à la suite de SUB et terminées par SUBEND et SUBEXIT. Elles gèrent des variables locales indépendantes de celles exploitées par le programme principal, et acceptent le passage de paramètres.

Cela donne accès à des possibilités qu'ignore habituellement le Basic, comme la récursivité. Et le programmeur pourra s'en servir pour réaliser des instructions qui manquent à l'« Exelbasic », et se définir son propre langage, un peu comme avec Forth. Des accessoires de l'EXL 100 seront bien utiles dans ce domaine : les cartouches de mémoire vive auto-alimentées. Elles stockent des programmes en offrant une sauvegarde et une lecture quasi instantanées. Facile donc de se réserver une cartouche pour y inscrire des procédures complétant le Basic. Un langage à la carte, en quelque sorte.

Le Basic de l'EXL 100 présente donc les caractéristiques intéressantes qui étaient déjà disponibles sur le CC-40 et se mariaient parfaitement avec ce type d'ordinateur (une seule ligne d'affichage à cristaux liquides). Les ajouts propres à l'EXL 100, en particulier le graphisme et les sons, s'avèrent décevants. Consolation et lueur d'espoir : le Basic est en cartouche et donc, facilement améliorable. Sa structure et le foisonnement de CALL s'y prêtent parfaitement.

Xavier de LA TULLAYE

METTONS NOS TITRES EN VALEUR

RIEN n'est plus tristounet qu'un écran tout gris et immobile. Même avec des capacités graphiques telles que celles d'un ZX 81, on peut donner de la vie aux pages de titre de ses programmes. Voici quelques idées pour commencer. A vous de les perfectionner et de les adapter à vos machines.

Certains ordinateurs ont une instruction Basic « FLASH » pour créer un message clignotant. On retrouve également sur Minitel un code de ce genre. Le Basic du ZX n'est pas aussi luxueux, mais quelques lignes y pourvoient aisément.

```
9000 REM CINEGRAMME 1
9010 LET A$=" N IMPORTE
     QUOI "
9020 LET B$="                "
9030 FOR I=1 TO 10
9040 PRINT AT 10, INT ((32-LEN
     A$)/2) ;A$
9050 PRINT AT 10,INT ((32-LEN
     B$)/2);B$
9060 NEXT I
```

L'astuce (élémentaire) consiste ici à écrire simplement le message et une chaîne de blancs de même longueur à la même place, grâce à l'instruction PRINT AT. Ici, la ligne 9010 a été choisie arbitrairement pour l'affichage et l'expression INT ((32 – LEN A$)/2) permet le centrage automatique du message quelle qu'en soit la longueur.

Un effet plus vibrant encore consiste à faire alterner un message en vidéo normale et ce même message en vidéo inversée. On le rencontre parfois dans certaines *pubs* télévisées. Le programme est simple :

```
9000 REM CINEGRAMME 2
9010 LET A$=" N IMPORTE
     QUOI "
9020 LET B$=" N IMPORTE
     QUOI " (en vidéo inversée)
9030 FOR I=1 TO 10
9040 PRINT AT 10,INT ((32-LEN
     A$)/2);A$
9050 PRINT AT 10, INT ((32-LEN
     B$)/2);B$
9060 NEXT I
```

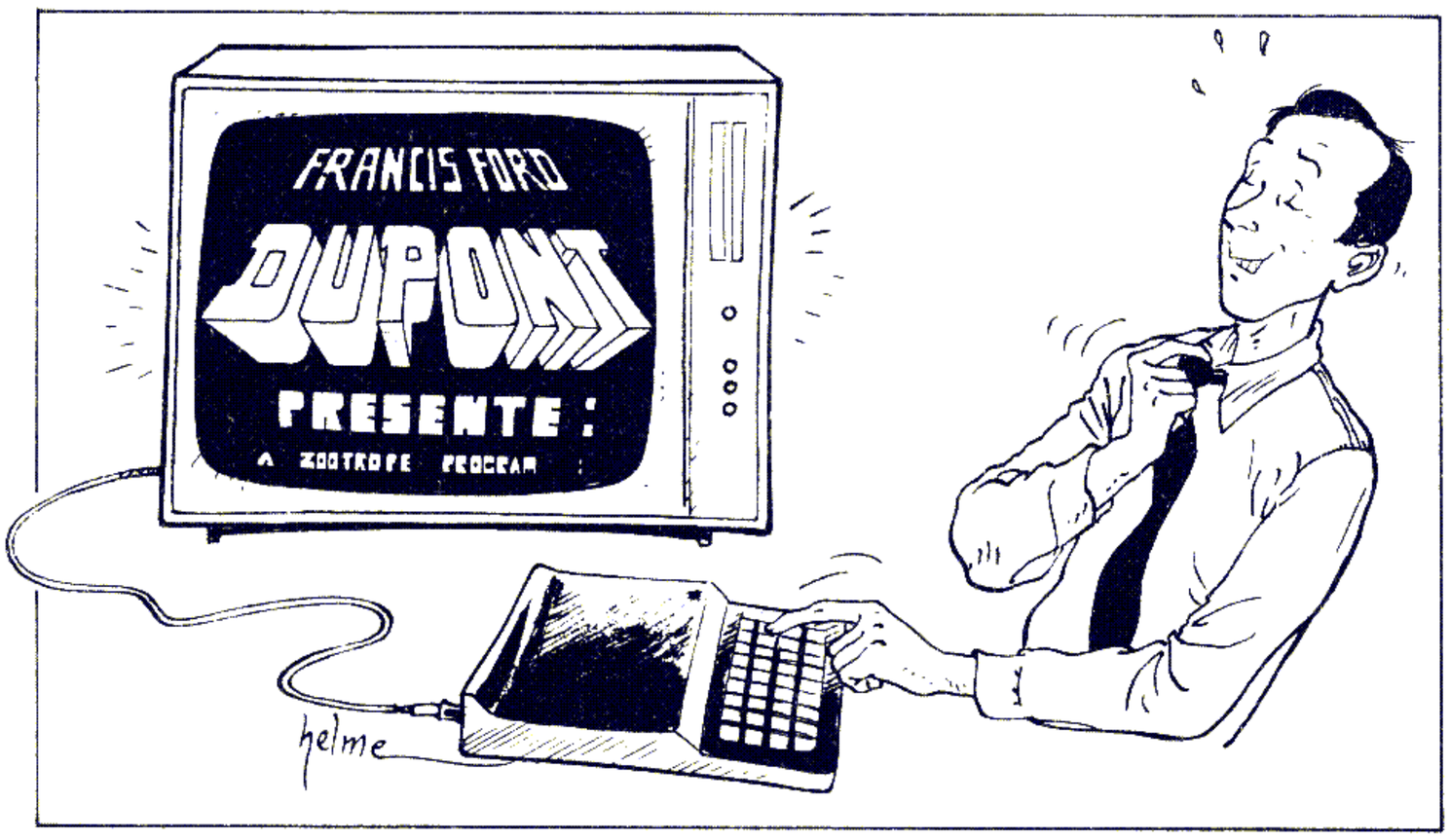
Suivant le vieux principe de la boule de neige (DUBO, DUBON, DUBONNET, ça existe encore ?), on peut facilement faire apparaître un message lettre par lettre, en triangle :

```
9000 REM CINEGRAMME 3
9010 LET A$=" N IMPORTE
     QUOI "
9020 FOR I=1 TO LEN A$
9030 PRINT A$(TO I)
9040 NEXT I
```

Et sens dessus dessous ?

Si l'on préfère les carrés magiques, on peut meubler l'écran d'un beau pavé de titre, lisible dans tous les sens :

```
9000 REM CINEGRAMME 4
9010 LET A$=" N IMPORTE
     QUOI "
9020 LET L=LEN A$
9030 LET X=INT ((32-L)/2)
9040 LET Y=INT ((22-L)/2)
9050 PRINT AT Y,0 ;
9060 LET A$=A$+" "+A$
9070 FOR I=1 TO L+2
9080 PRINT TAB X; A$(I TO I+L-1)
9090 NEXT I
```



Ce n'est pas le célèbre et mystérieux SATOR
AREPO
TENET
OPERA
ROTAS(1)
mais on fait ce qu'on peut !

On peut préférer ménager un habile suspense en faisant entrer les lettres une par une sur l'écran, en commençant par la droite :

```
9000 REM CINEGRAMME 5
9010 LET A$=" N IMPORTE
     QUOI "
9020 FOR I=LEN A$ TO 1 STEP -1
9030 FOR J=1 TO 31-LEN A$+1
9040 IF A$(I) < > " " THEN PRINT
     AT 10,J-1;" ";AT 10,J ;A$(I)
9050 NEXT J
9060 NEXT I
```

Vos convictions sont-elles d'une autre latéralité ? Entrons par la gauche :

```
9000 REM CINEGRAMME 6
9010 LET A$="N IMPORTE
     QUOI "
9020 FOR I=1 TO LEN A$
9030 FOR J=30 TO I STEP -1
9040 IF A$ (I) < > " " THEN PRINT
     AT 10,J+1 ; " " ;AT 10,J;A$(I)
9050 NEXT J
9060 NEXT I
```

Mais, inspiré par la comète de Halley, vous pouvez souhaiter laisser sur l'écran de fulgurantes (hum !) traînées. Facile :

```
9000 REM CINEGRAMME 7
9010 LET A$="N IMPORTE
     QUOI"
9020 LET A$=A$+ " "
9030 FOR I=1 TO LEN A$
9040 FOR J=30 TO I STEP -1
9050 PRINT AT 10,J ; A$(I)
9060 NEXT J
9070 NEXT I
```

(1) Cette phrase latine au sens sibyllin peut se lire aussi bien de droite à gauche que de gauche à droite, de bas en haut ou de haut en bas.

Laissons tomber les lettres

L'horizontalité vous lasse-t-elle ? Passons au vertical et regardons choir les lettres.

```
9000 REM CINEGRAMME 8
9010 LET A$="N IMPORTE
     QUOI"
9020 LET T=INT ((32-LEN A$)
     /2)—1
9030 FOR I=1 TO LEN A$
9040 FOR J=1 TO 10
9050 IF A$(I)< > " " THEN PRINT
     AT J-1,T+I ; " " ; AT J,
     T+I ;A$(I)
9060 NEXT J
9070 NEXT I
```

Jusqu'ici, une fois les lettres mises en places, il ne se passe plus rien. Si l'on veut qu'il se passe toujours quelque chose sur son écran, pourquoi ne pas recourir au bon vieux chenillard des familles...

```
9000 REM CINEGRAMME 9
9010 LET A$="N IMPORTE
     QUOI"
9020 LET A$=A$+ " *** "
9030 LET L=LEN A$
9040 LET A$=A$+A$
9050 FOR I=1 TO L
9060 FOR I=1 TO L
9070 PRINT AT 10,(32-L)/2;A$ (I
     TO I+L-1)
9080 NEXT I
9090 NEXT J
```

Avec ce dernier exemple, on s'aperçoit que pour un PRINT AT, le ZX 81 ne prend que la partie entière du nombre ; l'expression PRINT AT 10,INT ((32-L)/2); A$ (I TO I+L-1), si elle est syntaxiquement plus précise, ralentit un brin le mouvement. A vous de choisir.

Mais rien ne vous oblige à réserver aux pages de titre vos efforts d'animation. Comme le ZX 81 s'arrête sur un message d'erreur dès que la mémoire d'écran est pleine, au lieu de procéder à un déroulement automatique, il est souvent utile d'avoir dans un petit coin un sous-programme de tourne-page qui demande d'appuyer sur NEWLINE (ou sur ENTER pour les pionniers : les premiers ZX distribués en France portaient en effet ENTER) avant d'effacer l'écran. Alors, adaptons le chenillard à l'envoi d'un message de ce genre en bas à droite de l'écran.

```
100  GOSUB 9000
120  PRINT "SUITE..."
130  STOP
9000 REM CINEGRAMME 10
9010 LET A$="FRAPPEZ
     NEWLINE"
9020 LET A$=A$ + " *** "
9030 LET L=LEN A$
9040 LET A$=A$+A$
9050 FOR I=1 TO L
9060 PRINT AT 21,31-L;A$(I TO
     I+L-1)
9070 IF INKEY$=CHR$ 118 THEN
     GOTO 9100
9080 NEXT I
9090 GOTO 9050
9100 CLS
9110 RETURN
```

Si l'on veut utiliser ce sous-programme pour un autre message, rien de plus simple : on précise avant de l'appeler LET A$="AUTRE MESSAGE" et on l'appelle par un GOSUB 9020.

Voilà. Le champ est vaste, et nul doute que vous aurez d'autres idées d'animation pour compléter votre bibliothèque de sous-programmes. Si, de surcroît, vous possédez un TOOLKIT ou un utilitaire de ce genre permettant les APPEND, vous pourrez vous livrer aux joies de la programmation modulaire.

Mais, au fait, vous, les as de la programmation en langage-machine, vous ne pourriez pas offrir aux lecteurs de *LIST* un équivalent ô combien plus rapide ?

François J. BAYARD

EDI-LOGO
POUR APPLE II, IIc et IIe

EDI-LOGO est l'interpréteur Logo conçu au M.I.T. en 1981, et adapté en français par Édiciel (Matra et Hachette). Le principal public visé est l'enfant qui va s'initier à la programmation par le biais des graphiques et de la musique. Mais les possibilités d'extension du système (grâce à la programmation en Assembleur et à la création de 'sprites' pouvant remplacer la tortue) donnent à Édi-Logo un intérêt professionnel.

Une fois l'ordinateur Apple amorcé avec la disquette *Langage* et après quelques dizaines de secondes d'attente, l'écran se couvre du texte précisant l'origine du produit :

```
      ***     EDI-LOGO     ***
VERSION MIT POUR APPLE II 64K
ECRIT PAR S. HAIN, P. SOBALVARRO
ET L. KLOTZ SOUS LA DIRECTION DE
H. ABELSON.
ADAPTATION FRANCAISE: C. BERDONNEAU
COPYRIGHT (C) 1981, 1983 MIT
COPYRIGHT (C) 1983:
EDICIEL MATRA ET HACHETTE
BIENVENUE A EDI-LOGO
?
```

La présentation du logiciel

Le caractère de sollicitation est le point d'interrogation préfixant le curseur clignotant. Sans attendre, tapons : BONOUR. Aïe, une erreur de frappe ! Comment corriger avant d'appuyer sur Return ? Essayons :

- la touche flèche-à-gauche fait bien reculer le curseur ;
- tout caractère tapé est inséré juste avant le curseur (les personnes ayant tapé des programmes en Basic ou en Apple Logo seront un peu gênées : les modalités de correction sont ici différentes) ;
- la touche **'esc'** efface le caractère situé à gauche du curseur ;
- une fois l'insertion ou la suppression faite, il ne reste plus qu'à valider la ligne par Return.

Une pression sur Return donc, et l'affichage indique : BONJOUR.

Les messages renvoyés par Edi-Logo sont clairs et nombreux (52) avec un tutoiement inattendu, ainsi "S'IL TE PLAIT, EFFACE QUELQUE CHOSE" (quand l'espace de travail est saturé), ou encore "TU ESSAIES DE DIVISER PAR ZERO".

Si l'on tape un message quelconque sous la sollicitation de ?, Edi-Logo répond que le premier mot du message n'a pas été défini. Il a cherché une procédure portant ce nom sans succès. La définition d'une procédure est introduite par POUR, ce dernier conduisant en mode éditeur-écran :

- une kyrielle de commandes d'édition sont disponibles pour modifier le texte affiché dans ce mode éditeur pleine page (esc, →, ←, CTRL – A,B,C,D, E,F,K,N,O,P) ;
- un bandeau en dernière ligne précise les modalités de sortie de ce mode.

```
POUR BONJOUR
AFFICHE "BONJOUR
FIN
'CTRL-C'
```

La procédure BONJOUR est alors définie. L'écriture des commandes et des procédures est donc très simple. Un problème délicat se pose pourtant quand on veut taper les caractères crochet ouvrant et crochet fermant. Sur un clavier d'Apple II, le crochet ouvrant est obtenu par SHIFT N et le fermant par SHIFT M. Sur le clavier d'un IIe, ces caractères sont présents à condition que le commutateur de clavier soit en position QWERTY. Pour le IIc, en AZERTY, le crochet ouvrant a pour équivalent le degré ° et le crochet fermant l'alinéa §. Le degré est accessible directement en mode majuscule sur la ligne des chiffres, l'alinéa nécessite deux doigts, SHIFT-6. En Apple IIc, on peut aussi commuter en clavier QWERTY à condition de redessiner sur chaque touche le caractère correspondant...

Il faut aussi en IIe et IIc s'assurer que les lettres sont tapées en majuscules : l'affichage est trompeur puisqu'il transforme les lettres tapées en minuscules en majuscules ; l'interpréteur, lui, intercepte les minuscules et ne les comprend pas.

Ces détails d'écriture étant connus, il n'y a plus qu'à se laisser entraîner dans le royaume des listes ou des mots. A propos, quelle est la taille de l'espace-mémoire disponible ?

En Logo, l'atome d'information s'appelle un *nœud,* il contient généralement deux pointeurs et nécessite par conséquent quatre octets.

```
?AFFICHE .ESPACE
2281
```

Cette réponse donnant le nombre de nœuds libres a été obtenue juste après le démarrage : c'est donc la taille maximum. Ce n'est pas le Pérou ! Il est clair qu'Édi-Logo ne sait pas prendre en compte la mémoire auxiliaire de 64 Ko d'un Apple IIe ou IIc. Une bonne connaissance du système Apple et de l'occupation de l'Édi-Logo en mémoire devrait permettre aux spécialistes d'agrandir leur espace de travail. Comme base de recherche, nous savons que l'interpréteur Logo, implanté à partir de $4000, occupe 22944 octets. Nous savons aussi que l'espace des nœuds est situé entre $D000 et $FFF7 de la mémoire vive commutée (on retrouve à peu près 5 × 2280 octets ; à chaque nœud de 4 octets est attribué un autre octet pour le type de nœud). Mais arrêtons là notre voyage à l'intérieur des mémoires pour passer plus de temps à l'extérieur, c'est-à-dire à la découverte des fichiers disponibles sur la disquette dont le titre est *Utilitaires*.

Cette disquette est recopiable puisqu'elle fonctionne sous le DOS 3.3. Il faut signaler qu'Édi-Logo dispose d'une procédure permettant d'utiliser la plupart des commandes du DOS. Elle s'appelle justement DOS. Par exemple ?DOS °CATALOG, D2§ permet d'afficher la liste des fichiers de la disquette présente dans le lecteur D2.

En lisant le catalogue de la disquette *Utilitaires,* on se réjouit de pouvoir utiliser des procédures prêtes à l'emploi et probablement spectaculaires. Le premier fichier, *Animal,* contient toutes les procédures permettant à l'ordinateur de deviner le nom d'un animal grâce aux réponses oui/non que l'interlocuteur apporte à des questions qu'il a lui-même données d'avance. Les connaissances zoologiques de l'ordinateur s'accroissent au fur et à mesure de ses déficiences puisque, si l'utilisateur ne trouve pas le bon animal, il doit fournir une question permettant de distinguer un animal déjà connu de celui nouvellement acquis. Cet ensemble de procédures, bien que déjà connu par ailleurs, constitue un très bon exemple de programmation avancée en Logo, tout en étant un excellent jeu éducatif.

En suivant le catalogue, on trouve les quatre fichiers *Assembleur, Amodes, Opcodes, Adresses* qui permettent d'écrire des sous-programmes en Assembleur (avec les facilités du mode éditeur). Il s'agit, bien entendu, de produire du langage-machine du 6502 à partir des noms symboliques des codes d'opérations, des modes d'adressage, d'étiquettes et d'adresses symboliques. Pour faciliter cette programmation (en Assembleur) d'extension du langage, les différents points d'ancrage dans l'interpréteur, les adresses des variables ou des pointeurs sont disponibles sous forme de noms de variables Logo déjà enregistrées dans le fichier *Adresses*. Une fois le sous-programme assemblé et implanté dans une zone adéquate, l'appel depuis Logo se fait par :• APPEL :ADR :ARG. Le premier argument correspond au point d'entrée dans le sous-programme et le second argument correspond à un paramètre à fournir au S/P. Le résultat est récupéré à une adresse fixe, connue, de la page zéro.

Quand un Apple prend la parole

Les applications concernent principalement l'utilisation de périphériques non standard. Des exemples sont fournis pour piloter un lecteur de cassette ou contrôler toute carte d'interface. D'ailleurs, le plus bel exemple est la carte *Porte-Parole* vendue par *Édiciel* qui se connecte dans l'Apple II ou IIe et qui est un synthétiseur vocal. Le programme PICOLO.PARLE de la disquette *Utilitaires* s'en sert pour donner des explications de vive voix sur le maniement de la tortue.

Pour programmer en Assembleur des sous-programmes d'entrée/sortie par exemple, les explications du manuel de référence sont plutôt réduites. Il s'agira donc d'un travail de fourmi, même si le programme de démonstration musicale fait une bonne impression. Dans ce domaine, les applications sont l'objet des fichiers *Musique, Music.Bin, Music.Src.* En tapant RAMENE "MUSIQUE puis FRERE, vous aurez à reconstituer la musique d'une comptine célèbre, à partir de quatre motifs ou blocs mélodiques dénommés FA, FE, FI et FO, la constituant. Cette approche de la synthèse musicale sans solfège est originale.

Sinon, la procédure JOUE :N1 :D1, où N1 est une liste de notes (nombres) et D1 la liste des durées de chacune des notes de N1, vous permet de composer une mélodie. Quant à manipuler cette mélodie avec des répétitions ou des changements de rythmes et de tonalités, cela s'avère aussi simple que de faire tracer un polygone par la tortue. Le manuel « Premiers pas en Édi-Logo » suggère d'aller très loin dans ce domaine en inventant des procédures récursives en musique.

Pas de Logo sans tortue

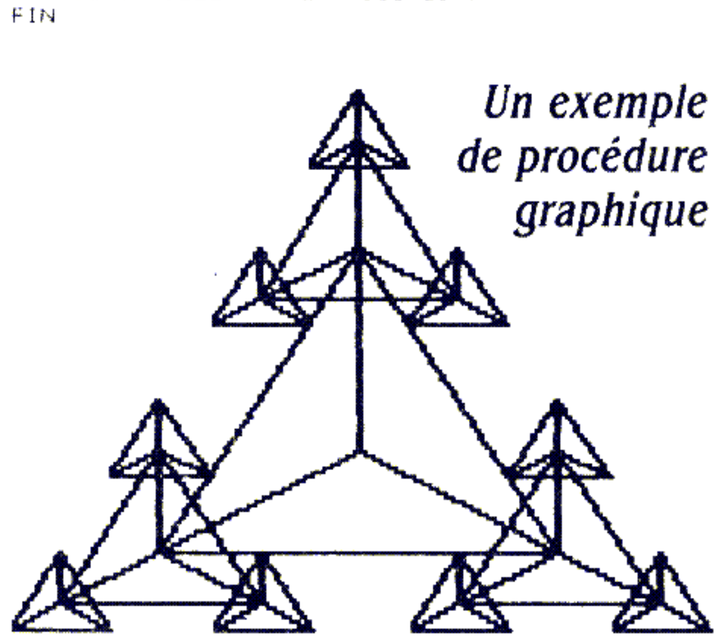
Et la tortue ? Elle apparaît avec l'injonction DESSINE. Les possibilités graphiques sont celles (largement diffusées) de maniement d'une tortue dont le cap est variable de 0° à 360° et qui dispose d'un terrain de 280 par 240 pas. Si elle sort de l'écran, c'est pour revenir aussitôt sur le côté opposé. Les variations d'échelles ou de couleurs ne posent aucune difficulté.

On trouvera ci-dessous une procédure graphique récursive donnée en exemple dans la disquette *Utilitaires.* Pour des raisons de précision du dessin, nous avons dû nous limiter au niveau 3 de profondeur, mais rien n'est impossible dans le domaine de la récursivité : le niveau 4 dessinera 3 figures de niveau 3 aux 3 sommets du triangle de base, soit 27 petits triangles, le niveau 5 en dessinera 81, le niveau 6, 243, etc.

La récursivité due à l'appel de TET à l'intérieur même de la procédure TET

```
POUR TET :TAILLE :PROFONDEUR
TESTE :PROFONDEUR > 0
SIVRAI REPETE 3 [MORCEAU :TAILLE TET
   :TAILLE *. .5 :PROFONDEUR - 1 DR 150
   AV :TAILLE DR 150 MORCEAU :TAILLE DR
   180]
SIFAUX STOP
FIN

POUR MORCEAU :TAILLE
AVANCE :TAILLE / ( 2 * COS 30 )
FIN
```



Un exemple de procédure graphique

sous la forme TET : TAILLE * .5 :PROFONDEUR – 1, est non-terminale ; dans ce cas, la mémoire va faire défaut au-delà d'une certaine profondeur : laquelle ?

La sauvegarde des dessins sur disquettes, c'est-à-dire des 8 Ko situés entre les adresses $2000 et $3FFF, est obtenue par la primitive GARDE.DESSIN avec comme paramètre un nom attribué au dessin.

Il existe également une primitive pour afficher directement un dessin enregistré sur disquette : c'est RAMENE.DESSIN suivie du nom du dessin tel qu'il a été baptisé au moment de la sauvegarde. Et, pour compléter le tout, la primitive DELETE.DESSIN permet de supprimer un dessin de la disquette.

Le logiciel en quelques lignes
Nom : Édi-Logo
Ordinateurs : Apple II 64 Ko, Apple IIe et IIc
Forme : disquette
Édité par : Édiciel (Matra et Hachette)
Prix public : 1 480 FF
Principale orientation : interpréteur Logo

La sauvegarde des procédures par la primitive GARDE assortie d'un nom de fichier, fait une copie sur disquette de l'espace de travail occupé, c'est-à-dire de toutes les procédures créées en mémoire vive. On peut éventuellement en effacer quelques-unes avec EFFACE avant la sauvegarde. Ces opérations sont assez restrictives quand on les compare à celles qu'Apple-Logo permet : groupement de procédures et de variables, enterrement de groupes pour les protéger, déterrement.

Dans la disquette *Utilitaires*, il faut signaler la présence d'EDIT.FORME qui est un programme éditeur de formes. Ces formes (ou « shapes » ou « sprites »), une fois créées sous cet éditeur et baptisées, peuvent se déplacer sur l'écran, changer de taille et remplacer éventuellement la tortue, mais elles ne peuvent être orientées que dans quatre directions.

L'utilisation d'Édi-Logo est à conseiller pour découvrir le langage Logo sur l'Apple, et le manuel « Premiers Pas » écrit par Catherine Berdonneau contient beaucoup d'exemples motivants et d'idées de projets. Cela dit, l'exploitation du logiciel à l'aide du manuel de référence est plutôt difficile : il faut analyser les exemples fournis sur la disquette *Utilitaires* qui couvrent les applications essentielles.

Nicole BRÉAUD-POULIGUEN

STORY BOARD

DES DESSINS ANIMÉS POUR LA GAMME THOMSON

LORSQU'ON a la chance de posséder un ordinateur capable de performances graphiques, on rêve souvent de créer de belles images et même, de leur donner vie à la manière des dessins animés. Avec un TO7, un TO7/70 ou un MO5, le rêve devient réalité grâce à un logiciel intitulé Story Board. La mise en scène se met à la portée de tous, une fée les y attend.

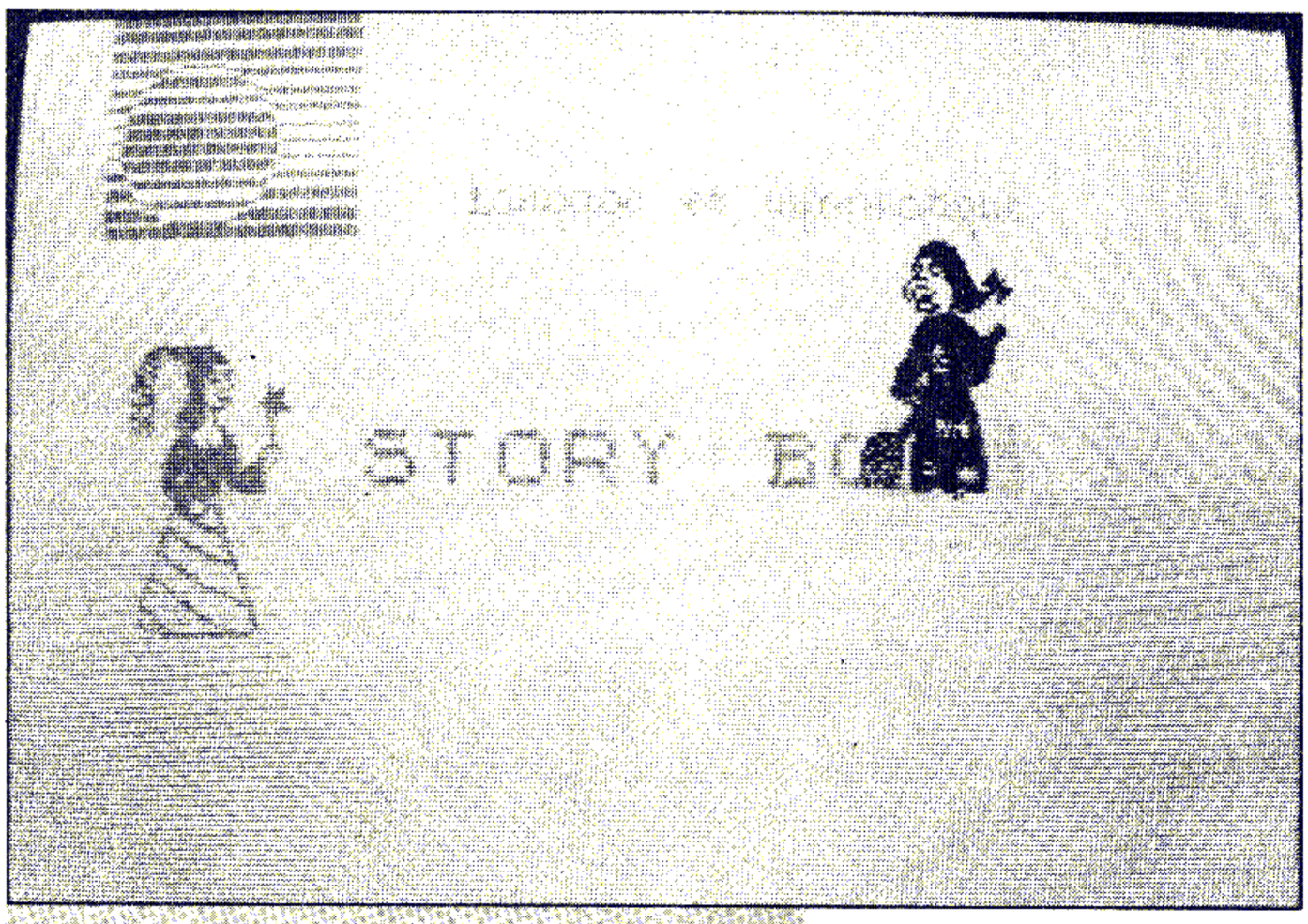
Bien que son nom ne l'indique pas, *Story Board* est un logiciel français. Il est édité par Langage et Informatique, une société toulousaine. Il se présente sous forme de disquette ou de cassette. La première forme est plus facile d'utilisation et beaucoup plus rapide. Mais elle entraîne la présence d'un lecteur de disquette, périphérique coûteux. Pour ceux qui n'ont pas la chance de le posséder, la version sur cassette sera suffisante. Elle nécessitera plus de patience.

Le chargement du logiciel (ici sous forme de disquette) est simple, il s'effectue sans manipulation particulière. Le programme nous présente alors une charmante fée. Elle est aidée dans ses efforts par une sorcière qui traverse l'écran en le nettoyant consciencieusement, avec son véhicule habituel : un balai !

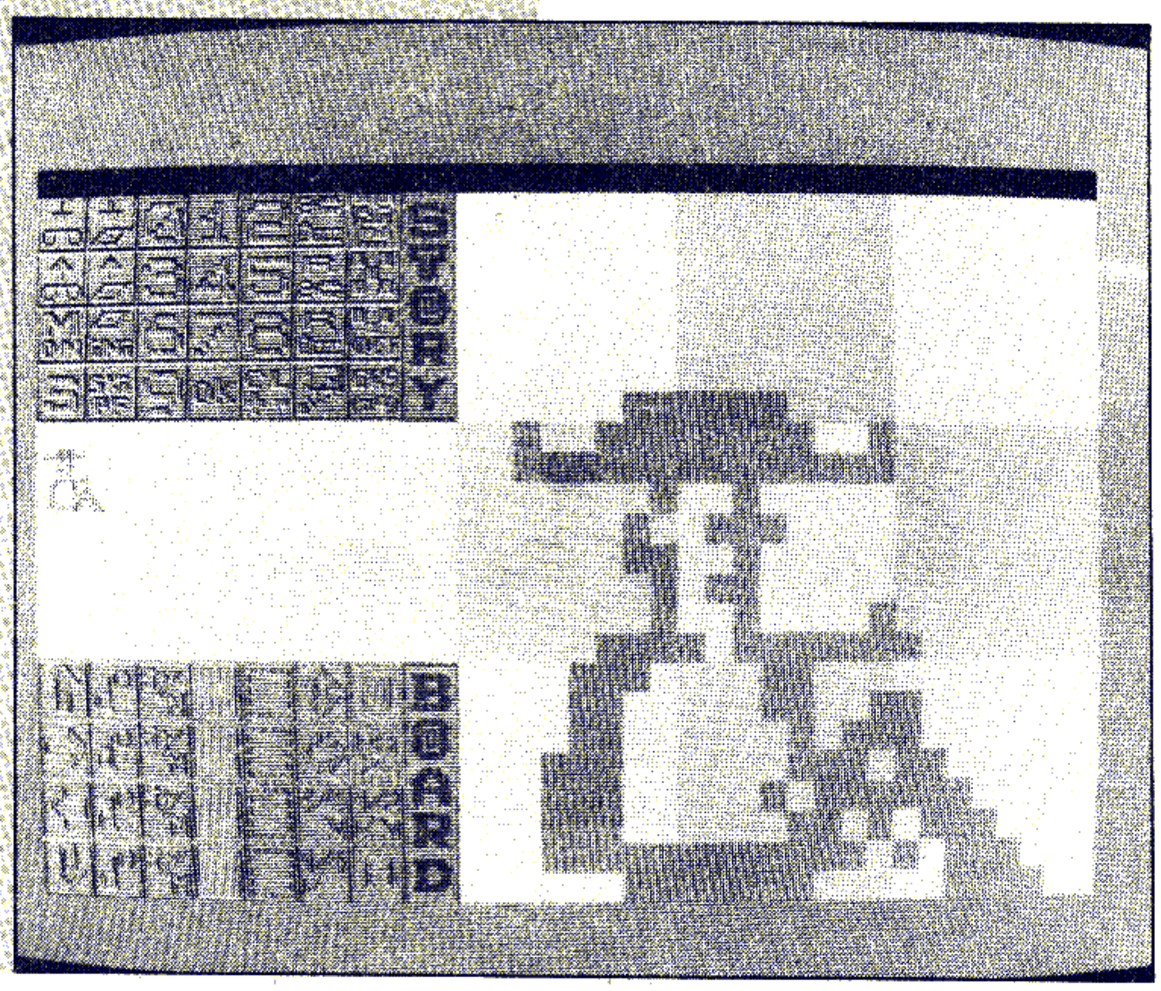
L'exécution de cette scène, sans doute réalisée par *Story Board* lui-même, est assez convaincante et peut séduire les plus incrédules. Mais au prix de quels efforts un tel résultat est-il obtenu ?

Tout se fait avec le crayon optique

Le logiciel est un utilitaire graphique complet. Il propose en premier lieu la création de dessins avec, pour seul outil, le crayon optique. C'est lui qui permet de sélectionner les diverses options proposées par le menu et de réaliser les dessins. Dès qu'il allume un point dans la zone droite de l'écran, ce point est reproduit à gauche de l'écran, à une



Le générique du logiciel



A droite de l'écran, un agrandissement du dessin en cours d'exécution

échelle huit fois plus petite, correspondant à l'aspect réel de l'œuvre en gestation. Le suivi est donc immédiat.

Le menu propose 28 fonctions sélectionnables par le crayon optique, qui permettent de nombreuses fantaisies. Des déplacements, des retournements, des duplications, des rotations sont prévus avec, éventuellement, la protection d'un dessin déjà créé par une fonction dite « rideau ». Ce qui pourra éviter des désagréments. D'autant plus que certaines fonctions, conduites imprudemment, peuvent mener à la perte d'une partie du dessin déjà réalisé.

Les dessins sont conservés sous forme de GR$. Leur taille est donc limitée à une grille horizontale de 16 points sur 8, puisque 128 seulement de ces GR$ sont disponibles. Ce nombre est réduit de façon significative sur TO7 sans extension mémoire : 70 GR$ seulement sont disponibles.

A partir du dessin initial, *Story Board* permet alors un véritable travail de mise en scène. Il propose de mettre au point un montage pouvant comporter un maximum de dix scènes de dix plans chacune. Ici encore, la réduction des possibilités est très significative sur le TO7 de base, puisqu'alors le maximum possible est de deux scènes de quatre plans chacune.

L'animation du dessin exige toutefois un réel effort de la part du créateur. Pour chaque plan, sept paramètres doivent être définis : déplacements horizontal et vertical, début et fin de la portion de dessin affectée, couleurs de l'image et du fond, temporisation. La mise au point d'un montage de la taille maximum exigera donc l'entrée de 700 paramètres successifs !

Là encore, tout se fait avec le crayon optique, ce qui nécessite une certaine habitude d'utilisation avant d'atteindre précisément le but recherché.

Certains outils facilitent le travail. Il est possible de corriger un plan précis, voire toute une scène si elle est mauvaise, La projection du montage scène par scène sur l'écran permet de juger, en direct, les résultats atteints. Avec de la patience, ils peuvent être spectaculaires.

Le menu offre alors la possibilité de conserver l'œuvre sur cassette ou disquette. Le dessin est enregistré sous forme de fichier binaire, les paramètres sous forme de DATA. L'ensemble peut ainsi être récupéré sans difficulté et réemployé à volonté, même à l'intérieur d'autres logiciels. La disquette contient, à cet effet, un programme non protégé qu'il suffira d'adjoindre à n'importe quel autre programme pour lui apporter une touche artistique.

L'inconvénient majeur de *Story Board* réside dans l'utilisation du crayon optique. S'il apporte une grande souplesse dans les manipulations, il oblige malheureusement à rester trop près de l'écran dont la luminosité a dû être poussée suffisamment pour compenser la « myopie » du crayon.

Les séances d'utilisation de ce logiciel sont donc une épreuve pour les yeux. Cet inconvénient disparaîtrait avec une tablette à digitaliser, mais elle n'existe pas sur la gamme Thomson.

Le logiciel en quelques lignes

Nom : *Story Board*
Ordinateurs : *TO7, TO7/70, MO5*
Forme : *cassette ou disquette*
Auteur : *Guy Leblond*
Édité et distribué par : *Langage et Informatique (Toulouse)*
Prix public : *290 FF la cassette, 340 FF la disquette*
Principale orientation : *création de dessins et animation*

Avant de pouvoir jongler avec *Story Board,* il faudra faire de nombreux essais. Les effets souvent spectaculaires seront le résultat d'une bonne dose de patience et de précision. Ce logiciel est donc précieux et propre à développer la créativité, surtout pour un prix de 290 FF ttc dans sa version sur cassette, et de 340 FF ttc dans sa version sur disquette.

Robin BOIS

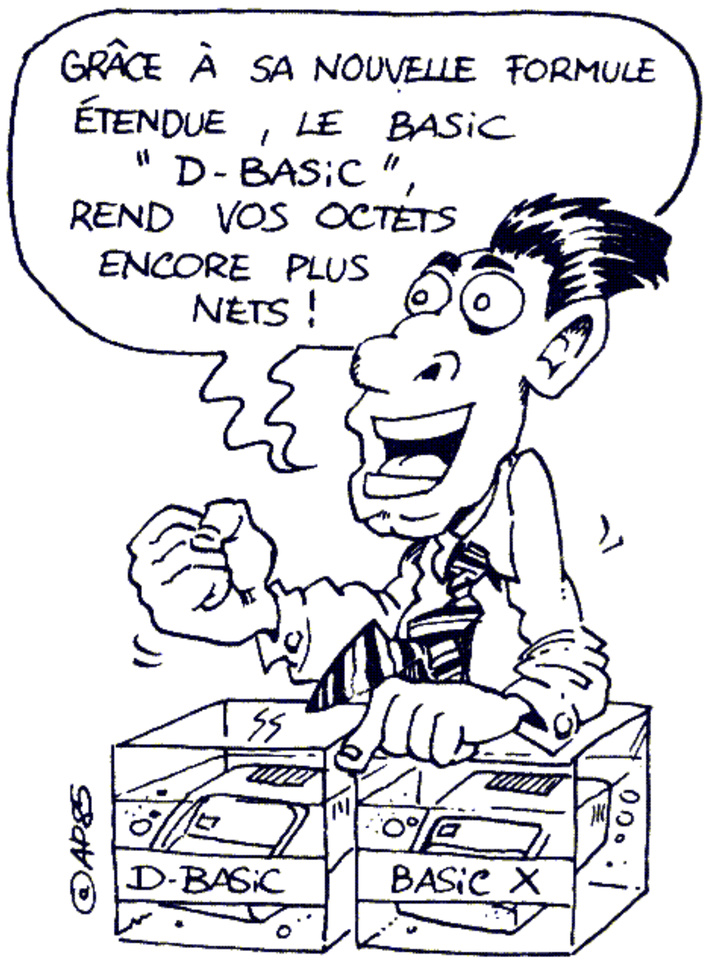
D-BASIC
UN BASIC ÉTENDU
POUR DAI

MALGRÉ tout le soin que le constructeur apporte à la conception matérielle et logicielle de la machine, il manque toujours les « petits riens » qui rendent le Basic plus attrayant et plus performant. D-Basic est un de ces utilitaires qui étendent le langage et le dotent de possibilités dignes des langages dits « structurés ».

D-Basic est disponible sous forme de cassette ou micro-cassette numérique, à charger sous *Utility*. Contrairement à une pratique couramment observée, la protection (sérieuse) n'allonge pas outrageusement le temps de chargement. C'est un bon point. Le lancement est automatique, une fois le chargement achevé, et il est possible de programmer comme à l'accoutumée en Basic, mais avec toute une série de nouveaux ordres qui pallient les manques du langage initial.

Il y a du Pascal dans l'air

La seule différence dans les ordres habituels concerne l'instruction conditionnelle IF. En effet, la structure du Basic Dai rend nécessaire, sous *D-Basic,* d'achever la condition par ENDIF. En contrepartie, sous ce format, la condition peut s'étirer sur plusieurs lignes,



sans restriction. La nouvelle forme est, bien sûr, IF...THEN...ELSE...ENDIF. Les amateurs de programmation structurée apprécieront l'apparition du ELSE. Ils aimeront également trois structures qui autorisent la construction de programmes dignes de Pascal : ce sont les branchements sur labels, les procédures, les fonctions. Plus de GOTO ni de GOSUB renvoyant à des numéros de ligne incompréhensibles. Il suffit de déclarer un label, en début de ligne, sous la forme : 100 "TOTO. Ensuite, les appels se font de façon simple, par exemple :

```
10 ON B GOTO "TOTO, "TITI,
   "TATA
```

Les procédures et fonctions s'écrivent comme en Pascal. Elles sont définies en tête de programme, sous la forme DEF PROC, ou PROCEDURE, ou bien DEF FN, ou FUNCTION, et s'achèvent par END PROC ou END FN. Bien entendu, les variables locales sont autorisées, par la commande LOCAL X,Y,Z. Pour couronner le tout, la récursivité est présente, si bien que pour calculer une factorielle, il suffit de définir une fonction du genre :

```
10 FUNCTION FAC (I)
20 IF I=0 THEN FN=1
30 ELSE FN=I*FAC (I-1): END IF
40 END FN
```

La variable obligatoire FN contient le

résultat du calcul. L'appel se fait de façon classique :

```
100 INPUT "VALEUR";V: PRINT
110 PRINT FAC (V)
120 GOTO 100
```

Les expressions que le programme peut passer aux procédures et fonctions sont d'emploi très souple : il est possible de passer en paramètre une autre fonction, qui n'a pas besoin d'être évaluée. Il est parfaitement valide de passer l'expression SIN (X), si la fonction est libellée, par exemple :

```
200 DEF FN MACHIN (FN Z)
```

Les deux structures de contrôle REPEAT...UNTIL et WHILE...DO... WEND sont présentes, ainsi que la commande RESTORE < line number >, ou mieux RESTORE < label >. Voilà de quoi construire des programmes irréprochables ! D'autres nouvelles facilités, si elles ne sont pas indispensables, se révèlent d'emploi agréable. Par exemple DOKE, qui, vous l'avez peut-être deviné, est un POKE sur deux octets, DEEK (V), qui est l'inverse (PEEK sur 2 octets) ou bien le DIMensionnement de matrices qui, sous *D-Basic*, n'est plus limité à 255 éléments, mais... à 2000 par défaut (plus, si nécessaire, dans les limites de la mémoire vive).

Les erreurs sont traquées

Bien plus intéressants sont les ordres gérant les erreurs : ON ERROR GOTO...RESUME font désormais partie du Basic Dai, ainsi qu'un utile ON BREAK GOTO et BREAK ON / BREAK OFF, permettant à peu de frais de protéger assez efficacement un programme écrit sous *D-Basic*.

Pareille adjonction nécessite de nouveaux messages d'erreur. *D-Basic* porte leur nombre à 55, fonctionnant comme par le passé lors de l'analyse syntaxique de chaque ligne, dès qu'elle est tapée. J'ai beaucoup apprécié le fait que les nouvelles commandes s'écrivent normalement dans le programme, sans abondance de CALLM, comme pour d'autres logiciels prétendant offrir les mêmes services. Une des rares servitudes est l'obligation de compiler le programme avant de le sauver (commande COMPILE), mais l'avantage est l'autodémarrage d'un tel programme au chargement. Cette dernière particularité, combinée avec BREAK OFF rend un logiciel non listable (et, a fortiori, non copiable, du moins simplement).

D'autres commandes (encore !), dont la description sortirait de cette présentation, se révèlent très puissantes lors des conversions d'une valeur numérique en chaîne de caractères, ou bien passent à une procédure la dimension d'une matrice, dimension qui n'est pas forcément connue à l'avance dans le programme. Au fait, vous ai-je dit qu'il est possible de passer tout un tableau à une procédure ou fonction, au moyen d'un ordre unique ?

Des instructions nouvelles pour le Dai
A titre d'exemple, ce programme graphique écrit par l'auteur de *D-Basic*

```
1    REM
2    REM --- TURTLE COMMAND ---
3    REM
10   PROCEDURE PENUP: PENFLAG=0: END PROC
20   PROCEDURE PENDOWN: PENFLAG=1: END PROC
30   DEF PROC FORWARD R: XNEW =X+R*COS (ALPHA): YNEW=Y+SIN(ALPHA)
40   IF PENFLAG=1 THEN DRAW X,Y XNEW,YNEW PENCOL: END IF
50   X=XNEW: Y=YNEW: END PROC
60   DEF PROC BACK R: XNEW=X-R*COS(ALPHA):YNEW=Y-R*SIN(ALPHA)
70   IF PENFLAG=1 THEN DRAW X,Y XNEW,YNEW PENCOL: END IF
80   X=XNEW:Y=YNEW: END PROC
90   PROCEDURE LEFT TETA:ALPHA=ALPHA+TETA*PI/180:END PROC
100  PROCEDURE RIGHT TETA: ALPHA=ALPHA-TETA*PI/180: END PROC
110  PROCEDURE CLEAN: FILL 0,0 XMAX,YMAX 0: END PROC
200  PROCEDURE SQUARE L: PENDOWN: FOR INDEX=1 TO 4
210  FORWARD L:LEFT 90: NEXT: RIGHT 360:PENUP:END PROC
300  DEF PROC PENTA L:PENDOWN: FOR INDEX=1 TO 5
310  FORWARD L:LEFT 72: NEXT: RIGHT 360: PENUP: END PROC
400  REM
500  REM
600  REM --- DEMO ---
700  REM
800  MODE 6: COLORG 0 5 8 14: PENCOL=5
810  X=XMAX/2: Y=YMAX/2
820  FOR I= 50 TO 10 STEP -1
830  PENTA I: FORWARD 5: LEFT 5
840  NEXT
850  END
```

Le logiciel en quelques lignes

Nom : *D-Basic*
Ordinateurs : *Dai PC et Dai T*
Édité par : *Dainamic club Belgique Mottaart 20 3170 Herselt (Belgique)*
ou : Dainamic France, 9 rue Lavoisier, 59140 Dunkerque
Prix public : *2 000 F belges, version cassette audio (350 F français)*
Principales orientations : *nouvelles commandes étendant le Basic du Dai, et permettant la programmation structurée*

On trouvera ci-contre, un exemple, écrit par l'auteur de *D-Basic*, montrant comment réaliser facilement un embryon de langage Logo graphique.

Dernière minute : la version 2.2 est désormais disponible. Elle apporte deux nouvelles commandes $LIST et XREF qui donnent respectivement une liste formatée avec indication des boucles et sous-programmes et une table des références croisées avec, en particulier, la liste des procédures et des fonctions employées, et leurs numéros de lignes.

Alain MARIATTE

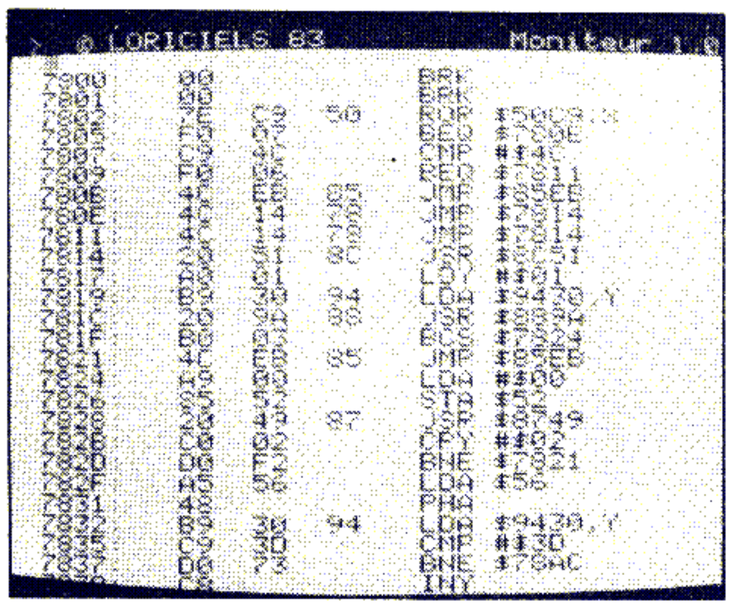
MONITEUR 1.0
UN MONITEUR POUR ORIC-1 ET ATMOS

LA programmation en langage-machine est possible sur Oric-1 ou Atmos, à condition d'avoir un logiciel adéquat. Moniteur 1.0 en est un exemple, il est spécialement tourné vers l'essai et la mise au point de programmes.

Il ne faut pas toujours se fier aux apparences. Bien que *Moniteur 1.0* soit présenté comme un logiciel pour Oric-1 (dans la version que nous avons testée), il se charge parfaitement sur un Atmos.

Une notice de deux feuillets décrit toutes les fonctions du logiciel. C'est assez court. Le programme chargé s'exécute automatiquement et l'on se retrouve sous moniteur. L'éditeur est de type ligne : les commandes tapées et corrigées avec les touches curseur sont validées par la touche RETURN. Tout ce qui se trouve sur la ligne est alors pris en compte.

L'utilisation de cet éditeur, différent de celui qui est disponible sous Basic, est agréable. L'écriture d'octets ou de codes ASCII se fait directement en tapant l'adresse concernée suivie des données à rentrer. Les commandes sont celles de désassemblage, DUMP d'octets et DUMP ASCII.



A l'écran, sous Moniteur 1.0, une routine désassemblée

Un petit assembleur intégré

Le déroulement de l'affichage, entièrement revu, est extrêmement rapide. La commande STEP propose une exécution pas à pas d'une routine en langage-machine. De même que TRACE. Ces deux instructions présentent l'inconvénient de traiter pas à pas les appels de sous-programmes effectués par JSR. Elles peuvent être suivies, de même que la commande d'exécution d'une routine, de paramètres visant à modifier les registres A, X et Y du 6502 avant exécution. Mais avec une exécution directe (commande GOTO), cette option « plante » le programme. De telles « bogues » sont sans doute le fruit d'une programmation rapide. Elles se retrouvent fréquemment dans les logiciels.

La commande MOVE déplace des blocs de mémoire en précisant les adresses de début, de fin et d'arrivée. Elle n'impose aucune restriction quant à la disposition relative des blocs. Mais pour reloger facilement les routines, il manque une commande de déplacement avec calcul des instructions sur trois octets.

La commande de recherche ne porte que sur des caractères ASCII, ce qui est pratiquement inutile. Elle aurait été préférable sur les octets.

Une pression sur CTRL P envoie à l'imprimante tout ce qui se trouve à l'écran. Des listes d'octets, de caractères ou même de mnémoniques, tapées au clavier, sont assemblées immédiatement. Des modifications locales du programme-objet peuvent donc être effectuées.

Le logiciel en quelques lignes

Nom : *Moniteur 1.0*
Ordinateurs : *Oric-1 et Atmos*
Forme : *cassette*
Édité et distribué par : *Loriciels*
Prix public : *140 FF*
Principales orientations : *moniteur, assembleur et mise au point des routines*

Les inconditionnels du Basic pourront trouver, dans ce logiciel, de quoi concevoir des routines en Assembleur. Il leur en coûtera 140 FF ttc.

Denis SEBBAG

POUR PEEKER TOUT DE MÊME

LE langage Pascal ne dispose pas d'instruction effectuant la lecture d'un octet en mémoire centrale. La fonction présentée ici, équivalent de PEEK, a pour objet de combler cette lacune. Elle permet d'aller lire la valeur d'un octet dont on connaît l'adresse en mémoire.

C'est une procédure très courte qui va nous permettre la lecture d'un octet en mémoire centrale : le corps du programme tient en deux lignes si l'on ne compte pas le *begin* ni le *end*.

Dans la déclaration de la fonction, le paramètre ADRESSE est l'adresse de l'octet que l'on veut lire. La valeur retournée par la fonction est celle de l'octet situé à cette adresse.

Deux octets par adresse

L'adresse est transformée sous forme d'un entier qui peut prendre toutes les valeurs utiles. Comme la constante MAXINT, prédéfinie en Pascal, est l'entier le plus grand pouvant être représenté sur un mot-mémoire de la machine, ADRESSE peut varier de −MAXINT − 1 à MAXINT. Cet intervalle englobe toutes les adresses possibles. Toutefois, pour se limiter aux seuls emplacements existants, le paramètre ADRESSE, si la mémoire est installée de façon contiguë, variera généralement entre les valeurs −(N div 2)... et (N div 2) − 1, formules où N représente la capacité mémoire de l'ordinateur exprimée en octets.

Pour parcourir dans l'ordre numérique toutes les mémoires que la machine peut adresser, il faut faire varier ADRESSE de 0 à MAXINT, puis de −MAXINT − 1 à −1. En effet, la valeur MAXINT + 1 est généralement égale à −MAXINT − 1, en raison du débordement que cause l'addition MAXINT + 1 où la retenue vient se mémoriser à la place du signe.

Ainsi, sur un ordinateur à mots de seize bits ADRESSE devra prendre les valeurs de 0 à 32767, puis de −32768 à −1. Ce parcours de tous les emplacements de la mémoire est illustré par le tableau ci-dessous.

Notons par ailleurs qu'un compilateur travaillant (même exemple) sur une machine à mots de seize bits refusera, dans un programme, la constante −32768. Cette dernière devra donc être remplacée par l'expression équivalente −32767 − 1 qui est normalement acceptée.

Toujours pour les machines travail-

Numéro d'ordre	Valeur décimale	Valeur binaire
1	0	00000000 00000000
2	1	00000000 00000001
3	2	00000000 00000010
4	3	00000000 00000011
5	4	00000000 00000100
(...)	(...)	(...) (...)
32766	32765	01111111 11111101
32767	32766	01111111 11111110
32768	32767	01111111 11111111
32769	−32768	10000000 00000000
32770	−32767	10000000 00000001
32771	−32766	10000000 00000010
(...)	(...)	(...) (...)
65533	−4	11111111 11111100
65534	−3	11111111 11111101
65535	−2	11111111 11111110
65536	−1	11111111 11111111

Les 65536 cases de la mémoire diversement représentées.

Positions de la mémoire

UNE PROCÉDURE PASCAL

POUR PEEKER TOUT DE MÊME

lant avec des mots de seize bits, la lecture doit s'effectuer en deux fois puisqu'il faut accéder d'une part à l'octet de poids faible et d'autre part à l'octet de poids fort (voir encadré).

L'octet de poids faible peut être placé en mémoire **avant** l'octet de poids fort (on dit alors que la polarité de l'ordinateur est négative). Dans ce cas, pour la lecture du mot placé à l'adresse 17520, l'octet de poids faible est lu à l'adresse paire 17520 et l'octet de poids fort à l'adresse 17521.

Inversement, sur un ordinateur dit *à polarité positive,* les octets de poids faible sont aux adresses impaires et les octets de poids fort sont aux adresses paires immédiatement inférieures (adresses des octets de poids faible diminuées de 1).

Enfin, il faut savoir que l'accès à certaines adresses de la mémoire peut être interdit par le matériel. Pour connaître ces adresses, il est nécessaire soit de se référer à la notice du constructeur, soit, si elle ne les indique pas, de procéder à un test général.

C'est à dessein que les concepteurs du langage Pascal n'ont pas prévu en standard une fonction de lecture en mémoire équivalant au PEEK du Basic. Ils ont en effet voulu ce langage aussi proche que possible de l'utilisateur. Dans le même état d'esprit, toute instruction proche du fonctionnement réel de la machine, peu lisible, a été bannie. De plus, les lectures en mémoire sont toujours spécifiques à un matériel ou même à un système d'exploitation donné. Leur utilisation ne permet donc pas de réaliser des programmes portables.

Evitons la routine

Cette fonction permet cependant de lire certaines variables-système qui ne sont pas accessibles directement. Elle évite d'écrire une routine en Assembleur pour un simple accès à un octet en mémoire centrale. De toute façon, il faut essayer, dans toute la mesure du possible, d'isoler avec précision ces accès afin qu'ils puissent être facilement modifiés si le programme est transporté sur une autre machine.

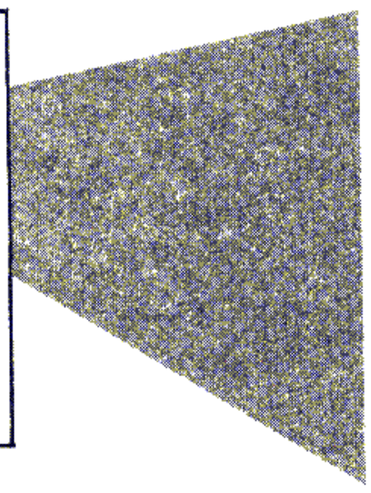
Enfin, une telle fonction est indispensable pour l'écriture d'un programme de vidage de la mémoire dont les résultats (bien que difficiles à exploiter) sont très utiles pour connaître mieux à la fois le matériel et le système d'exploitation utilisé.

Dans notre procédure, la variable MEMOIRE est un enregistrement qui est considéré

- soit comme un entier qui indique une adresse,
- soit comme un pointeur vers un tableau de deux octets.

Les champs ADRESSE et POINTEUR contiennent toujours la même valeur, puisqu'ils partagent la même zone en mémoire centrale. La technique utilisée ici est un moyen détourné d'initialiser un pointeur, ce que ne permettent pas les compilateurs, à l'exception de la valeur prédéfinie NIL.

```
function lecmen (adresse : integer) : integer ;
type     mot = packed array [0..1] of 0..255 ;
var      memoire : record case boolean of
                     true : (adresse : integer) ;
                     false : (valeur : ^ mot)
                   end ;
begin
    memoire.adresse := adresse ;
    lecmen := memoire.valeur^ [0]
end ;
```



Poids faible et poids fort

Pour mémoriser un nombre entier compris entre 0 et 255 (bornes comprises), un octet suffit. Avec deux octets, on peut enregistrer des nombres valant jusqu'à $2^{16}-1$, soit 65535. On distingue alors l'octet de poids faible et l'octet de poids fort.

Pour obtenir la valeur enregistrée sur deux octets, on additionne octet de poids faible + (octet de poids fort × 256). Autrement dit, l'octet de poids fort comprend la partie entière du résultat de la division de la valeur par 256. Et l'octet de poids faible contient le reste de cette division.

Ainsi, pour coder 38637, on aura :

- poids fort = int(38637/256) = 150
- poids faible = 38637 – (256 × 150) = 38637 – 38400 = 237

La première instruction consiste donc à mettre dans le champ ADRESSE la valeur qui a été transmise en paramètre. La seconde instruction reprend cette valeur, mais en la considérant comme un pointeur vers un tableau d'octets situé en mémoire centrale. Comme ce pointeur a été initialisé auparavant à la valeur ADRESSE, la valeur retournée par la fonction sera celle de l'octet dont on a transmis l'adresse.

Voilà comment en deux instructions, et avec une pincée d'astuce, il est possible de se fabriquer, en Pascal, une instruction PEEK.

Thierry CHAMORET

UN PROGRAMME BASIC

DES TEXTES A LA PELLE

UN programme en Basic se met au service de ceux qui doivent écrire des textes. Les résultats prennent des formes différentes, mais ils sont toujours cohérents. Une nouvelle façon de traiter son courrier de vacances, par exemple.

Enfin un programme qui compose du texte ! Attention, il n'invente rien, et sûrement pas le sujet qu'il va traiter. Mais il puise son inspiration dans un réservoir de phrases construites au préalable par l'utilisateur. Ces phrases, "à développement par arborescence", sont stockées dans des lignes de DATA (voir lignes 660 et suivantes du programme). Dans l'exemple présenté ici, elles provoquent la rédaction d'une courte lettre transmettant toujours les mêmes idées, avec des mots différents.

Variations autour d'un thème

Voyons, sur un petit exemple, le principe général. Une phrase "arborescente" se présente sous la forme d'une expression formée de chaînes de caractères, de parenthèses et d'opérateurs (+ et *). Nous choisirons : (Le téléphone + est + (une invention*un instrument de communication) + (extraordinaire* formidable) + .). L'opérateur + représente un "et", il provoque la concaténation de chaînes de caractères. Ainsi *Le téléphone + est* donnera *Le téléphone est*. L'opérateur * représente un "ou aléatoire", une seule des chaînes de caractères qui l'entourent sera choisie. Et donc, dans l'expression *extraordinaire*formidable,* seul un des deux adjectifs sera choisi, de manière aléatoire, par le programme.

Ainsi, l'exemple initial pourra donner les phrases :

- Le téléphone est une invention extraordinaire.
- Le téléphone est un instrument de communication extraordinaire.
- Le téléphone est une invention formidable.
- Le téléphone est un instrument de communication formidable.

Les expressions qui seront traitées par le programme apparaissent dans des lignes de DATA (lignes 660 à 740, ici). L'opérateur "ou aléatoire", inventé pour la circonstance, est susceptible de s'appliquer à plusieurs opérandes différents, mais il n'est pas associatif. Le principe consiste à employer trois piles :

- une pile "opérandes" dans laquelle sont stockées les chaînes de caractères ou opérandes ;
- une pile "opérateurs" dans laquelle sont stockés les opérateurs (* et +) et les parenthèses ouvrantes ;
- une pile "exécution" qui reçoit les opérandes dépilés de la pile opérandes en vue de l'exécution globale d'un opérateur.

Dans chaque ligne de DATA, l'analyse des expressions est effectuée de gauche à droite. Elle suit les règles suivantes :

- si une parenthèse fermante est rencontrée, les chaînes de la pile opérandes sont dépilées, empilées dans la pile exécution, et les opérateurs sont dépilés ;

Comment introduire ses propres textes

L'introduction des phrases à partir de la ligne 660 suit certaines règles imposées par le principe d'évaluation :

- il n'y a pas d'opération prioritaire. Des parenthèses judicieusement placées sont donc nécessaires pour lever toute ambiguïté. Ainsi l'expression (coucou*hello + c'est moi) est incorrecte, le programme ne sachant pas quelle opération traiter en priorité. Il faut écrire, par exemple : ((coucou*hello) + c'est moi) ;
- la "remontée" vers le début de l'expression ne s'effectue que lorsqu'une parenthèse fermante est rencontrée. Il est donc nécessaire de fermer la parenthèse à la fin d'une ligne, mais aussi de lui associer une parenthèse ouvrante qui sera le premier caractère de la ligne de DATA ;
- d'une manière générale, il faut qu'à toute parenthèse ouvrante corresponde une parenthèse fermante et réciproquement. Le nombre de parenthèses ouvrantes doit donc être égal au nombre de parenthèses fermantes.

Si l'une de ces règles n'est pas respectée, le message "erreur de parenthèse" apparaît ou alors, le programme se "plante".

UN PROGRAMME BASIC

DES TEXTES A LA PELLE

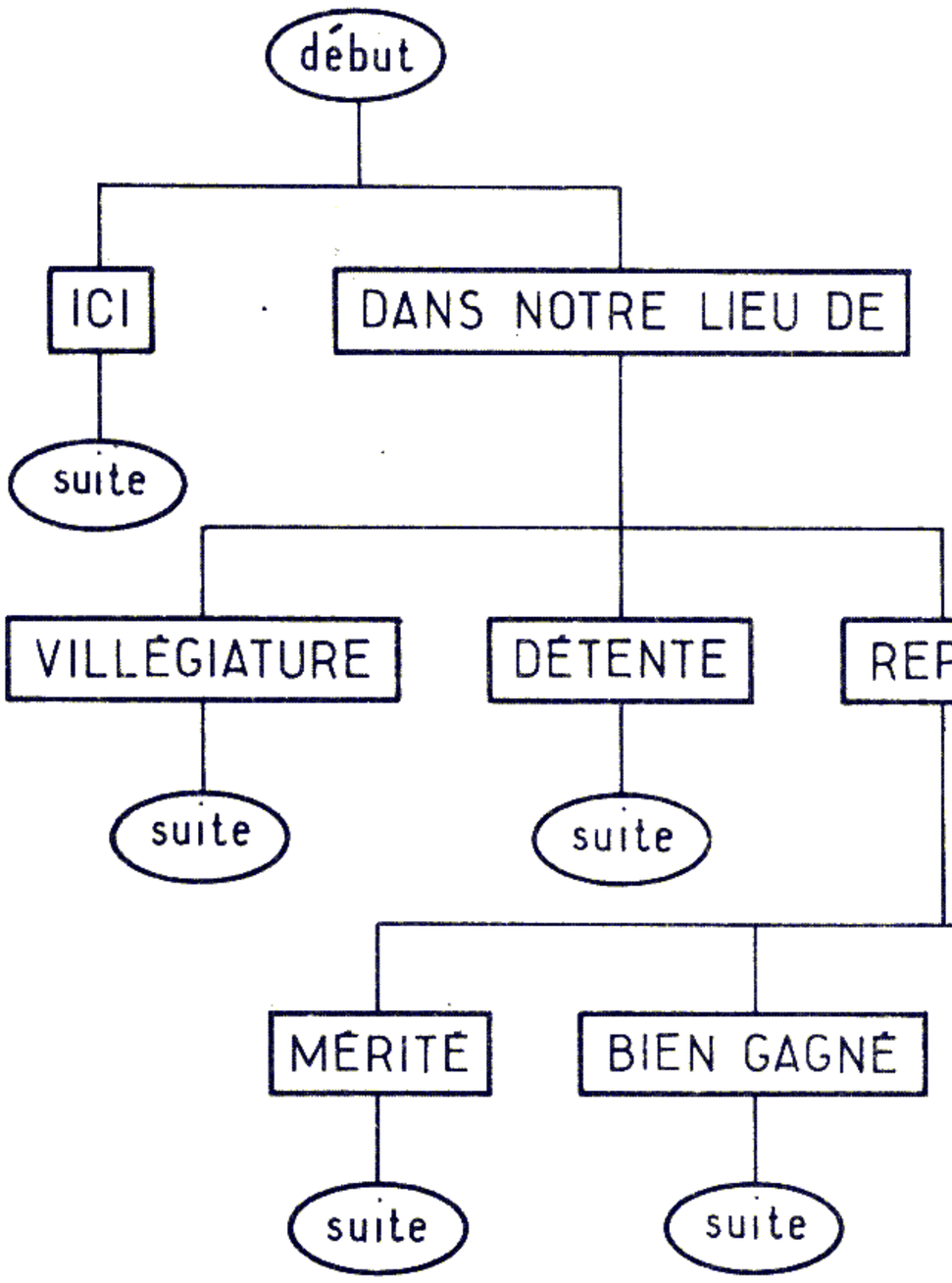
Générateur de textes
Programme en Basic
Auteur Thierry Lévy-Abégnoli
Copyright *LIST* et l'auteur

```
10 REM----INITIALISATION----
20 CLEAR 1000:REM RESERVE 1000 OCTETS POUR LES CHAINES DE CARACT
ERES
30 DEFINT A-Z:REM ACCELERE LES CALCULS (FACULTATIF)
40 DIM PDO$(30),POP$(30),PEX$(20)
50 CLS:REM EFFACE L'ECRAN
60 REM----LECTURE D'UNE "PHRASE POTENTIELLE"----
70 READ CH$
80 IF CH$="FIN" THEN END
90 D=0:O=0:E=0
100 REM----RECHERCHE D'ERREURS DE PARENTHESES----
110 FOR I=1 TO LEN(CH$)
120   IF MID$(CH$,I,1)="(" THEN I1=I1+1:GOTO 140
130   IF MID$(CH$,I,1)=")" THEN I2=I2+1
140 NEXT I
150 IF I1<>I2 THEN PRINT "ERREUR DE PARENTHESE":END
160 REM---------------------------------
170 REM EXTRACTION D'UNE PHRASE ALEATOIRE
180 REM A PARTIR DE LA PHRASE POTENTIELLE
190 REM---------------------------------
200 FOR I=1 TO LEN(CH$)
210 CAR$=MID$(CH$,I,1)
220 IF CAR$=")" THEN 290
230 IF CAR$<>"(" AND CAR$<>"+" AND CAR$<>"*" THEN 270
240 IF MOTS$<>"" THEN PDO$(D)=MOTS$+" ":MOTS$="":D=D+1
250 POP$(O)=CAR$:O=O+1
260 GOTO 510
270 MOTS$=MOTS$+CAR$
280 GOTO 510
290 IF MOTS$<>"" THEN PDO$(D)=MOTS$+" ":MOTS$="":D=D+1
300 GOSUB 600
310 OP$=POP$(O-1):O=O-1
320 GOSUB 600
330 IF POP$(O-1)=OP$ THEN O=O-1:GOTO 320
340 IF POP$(O-1)<>"(" PRINT "ERREUR DE PARENTHESE":END
350 O=O-1
360 IF OP$="*" THEN 480
370 REM---EXECUTION DE L'OPERATION "ET"---
380 REM----------NOTEE + ----------------
390 PDO$(D)=""
400 FOR J= E-1 TO 0 STEP -1
410   PDO$(D)=PDO$(D)+PEX$(J)
420 NEXT J
430 D=D+1
440 E=0:REM VIDAGE DE LA PILE EXECUTION
450 GOTO 510
460 REM--EXECUTION DE L'OPERATION "OU ALEATOIRE"--
470 REM------------------NOTEE * ----------------
480 PDO$(D)=PEX$(RND(E)-1):REM RND(X) TIRE UN NOMBRE "ALEATOIRE"
COMPRIS ENTRE 1 ET X INCLUS
490 D=D+1
500 E=0
510 NEXT I
520 REM----AFFICHAGE DE LA PHRASE ALEATOIRE----
530 PRINT PDO$(0)
540 GOTO 70
550 REM---------------------------------
560 REM DEPILAGE DE LA PILE DE DONNEE ET
570 REM EMPILAGE DE LA DONNEE DANS LA
580 REM PILE EXECUTION
590 REM---------------------------------
600 PEX$(E)=PDO$(D-1):E=E+1:D=D-1
610 RETURN
620 REM*********************************
630 REM RESERVOIR DE PHRASES DEFINIES
640 REM  DE MANIERE ARBORESCENTE
650 REM*********************************
660 DATA "(      +(BONJOUR*SALUT*COUCOU)+!+((ON VA+(ASSEZ BIEN*B
IEN*TRES BIEN))*CA VA*CA ROULE POUR NOUS*(NOUS NOUS PORTONS+(COM
ME DES CHARMES*MERVEILLEUSEMENT)))+.)"
670 DATA "(NOTRE PERIODE DE+(VACANCES*REPOS*DETENTE)+(TOUCHE A S
A FIN*ARRIVE A SON TERME)+,+(DOMMAGE*HELAS*TANT PIS)+!)"
680 DATA "(ON+(ESPERE*SUPPOSE)+(QUE CHEZ VOUS*QU'A LA MAISON*QU'
AU BERCAIL)+(CA+(VA*MARCHE*BAIGNE))+COMME+(VOUS+(VOULEZ*LE SOUHA
ITEZ))+.)
690 DATA "(((ICI*(DANS NOTRE LIEU DE+(VILLEGIATURE*DETENTE*(REPOS
+(MERITE*BIEN GAGNE*TANT ATTENDU)))))+, LE+(CLIMAT*TEMPS)+(EST*R
ESTE*DEMEURE)+(PLUTOT*ASSEZ)+(CLEMENT*AGREABLE)+.)"
700 DATA "((NOUS ALLONS POURTANT REVENIR*C'EST BIENTOT LE RETOUR
*DANS QUELQUES JOURS, NOUS REVOILA)+.)"
710 DATA "(SEUL+(LE BONHEUR*LE PLAISIR)+DE+((VOUS REVOIR+(DANS Q
UELQUES JOURS*BIENTOT))*NOS RETROUVAILLES)+(NOUS CONSOLE*ECLAIRE
 NOTRE COEUR)+.)"
720 DATA "((A TOUT DE SUITE*A BIENTOT*BYE BYE)+,+(BISOUS*BAISERS
)+.)
730 DATA "(      JOJO+TOTO+COCO+MOMO)"
740 DATA"FIN":REM PERMET LA DETECTION DE LA FIN DES DATAs
```

Deux exemples d'exécution

```
       SALUT ! NOUS NOUS PORTONS MERVEILLEUSEMENT .
 NOTRE PERIODE DE VACANCES ARRIVE A SON TERME , TANT PIS! .
 ON SUPPOSE QU'A LA MAISON CA VA COMME VOUS LE SOUHAITEZ .
 ICI , LE TEMPS EST ASSEZ CLEMENT .
 DANS QUELQUES JOURS, NOUS REVOILA .
 SEUL LE PLAISIR DE NOS RETROUVAILLES ECLAIRE NOTRE COEUR .
 A TOUT DE SUITE , BAISERS .
       JOJO TOTO COCO MOMO


       COUCOU ! ON VA BIEN .
 NOTRE PERIODE DE DETENTE TOUCHE A SA FIN , DOMMAGE! .
 ON ESPERE QU'A LA MAISON CA BAIGNE COMME VOUS VOULEZ .
 DANS NOTRE LIEU DE REPOS TANT ATTENDU , LE CLIMAT DEMEURE PLUTOT
  CLEMENT .
 C'EST BIENTOT LE RETOUR .
 SEUL LE PLAISIR DE VOUS REVOIR BIENTOT ECLAIRE NOTRE COEUR .
 BYE BYE , BISOUS .
       JOJO TOTO COCO MOMO
```



Un exemple d'arbre

Chaque expression comprise entre deux opérateurs + de premier niveau (ou avant le premier ou après le dernier d'une ligne) engendre un arbre. L'exemple représenté ici, correspond à l'expression située avant le premier opérateur + de premier niveau de la ligne 690. Il comprend six chemins différents, ici terminés par le mot "suite" qui renvoie la branche à l'arbre suivant (un arbre à une seule branche :,LE). Les autres arbres de cette ligne comprennent respectivement 1, 2, 3, 2 et 2 branches. La ligne 690 peut donc générer, potentiellement, $6 \times 1 \times 2 \times 3 \times 2 \times 2 = 144$ phrases différentes.

construite et se trouve sur la pile opérandes. Il ne reste plus qu'à la cueillir pour l'afficher.

Le hasard est-il poète ?

Pour construire une pile en Basic, un tableau et une variable suffisent : la donnée empilée est mise dans l'élément du tableau ayant la variable pour indice, puis cette variable est incrémentée.

D'un Basic à l'autre

L'adaptation de ce programme à d'autres ordinateurs (l'Amstrad, par exemple) est très facile. Il suffit de savoir que la ligne 20 y est inutile, que THEN est obligatoire à la suite de IF (ligne 340, il faut écrire : IF POP$(O – 1) < > "(" THEN PRINT "ERREUR DE PARENTHESE":END) et que RND(E), ligne 480, doit être remplacé par INT(E* RND + 1). En effet, RND(E) génère, sur le TRS-80 modèle 1, un nombre aléatoire entier compris entre 1 et E. Mais, sur d'autres ordinateurs, RND génère un nombre compris entre 0 et 1 et, ainsi, INT(E* RND + 1) est bien un nombre entier compris entre 1 et E.

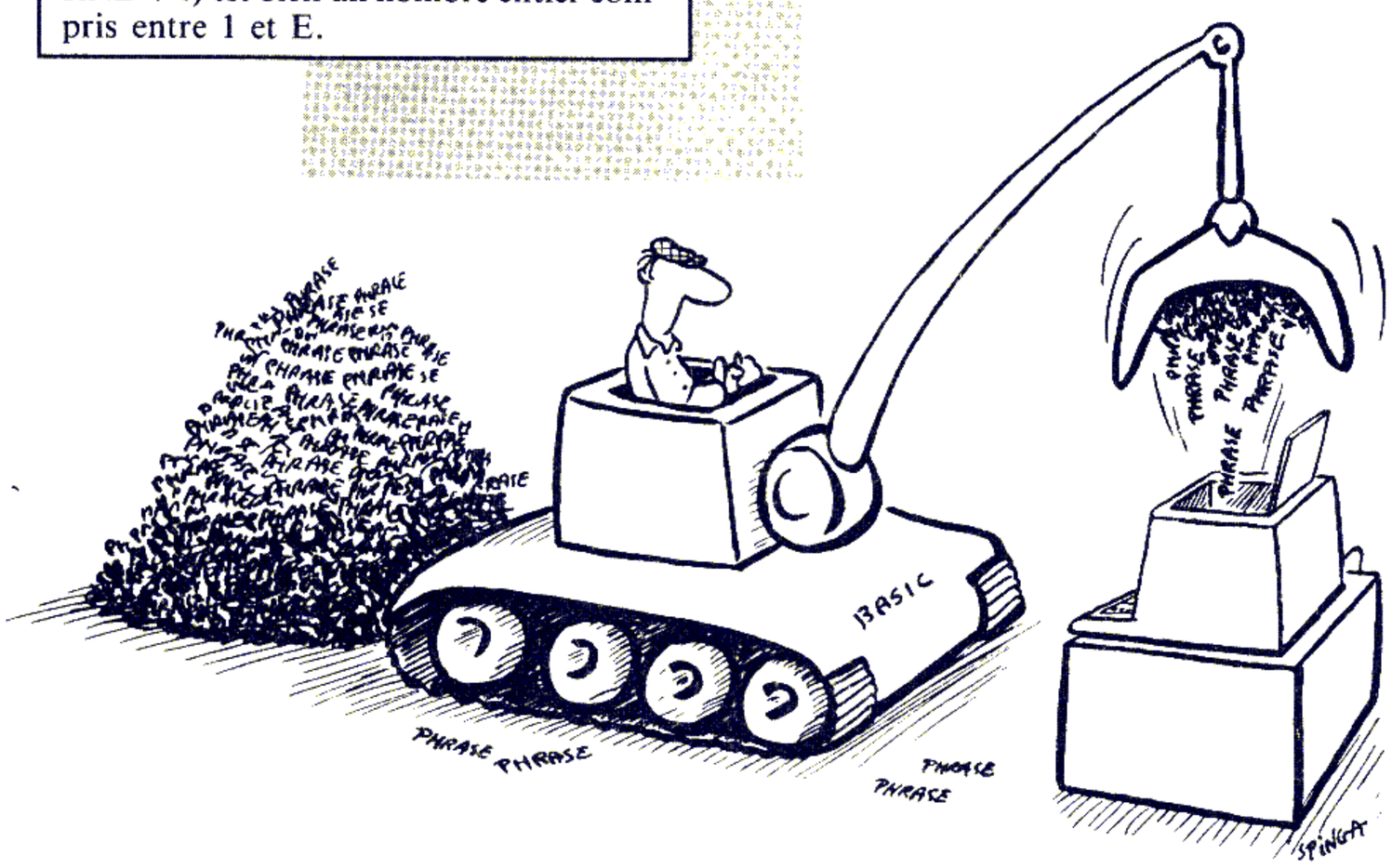
tout cela a lieu jusqu'à ce qu'une parenthèse ouvrante soit rencontrée sur la pile opérateurs. L'opérateur peut alors être appliqué aux opérandes de la pile exécution que l'on vide immédiatement ; on place ensuite le résultat sur le haut de la pile ;

- si un opérateur est rencontré, il sera empilé dans la pile opérateurs ;
- si une chaîne de caractères est rencontrée, elle sera empilée dans la pile opérandes.

En remontant ainsi jusqu'à la première parenthèse ouvrante, la phrase est

Liste des variables

PDO$(i) : pile de données, contient les morceaux de la phrase en construction.
D : pointeur de la pile de données, pointe sur l'élément recevant le prochain empilage, PDO$(O).
POP$(i) : pile d'opérateurs, contient les opérations et les parenthèses en attente.
O : pointeur de la pile opérateurs.
PEX$(i) : pile contenant les données qui attendent une exécution globale de l'opération en haut de la pile opérateurs.
E : pointeur de la pile exécution.



Dans l'exemple donné plus haut, deux niveaux de parenthèses seulement ont été utilisés. Mais il n'y a pas de limite. On peut rapidement créer une "explosion combinatoire" qui aura pour effet de multiplier les résultats. Ainsi, les textes seront variés.

Certains ont peut-être reconnu le principe des listes du langage Lisp. La mise en place de ces textes correspond en quelque sorte à un mini-langage de programmation.

Thierry LÉVY-ABÉGNOLI

LOGO

AVEC DES FLEURS

EN vingt années d'existence, malgré sa simplicité et sa puissance, Logo est toujours resté en retrait par rapport aux langages de programmation classiques tels que Basic ou Pascal. Et pourtant, Logo est maintenant disponible sur la plupart des ordinateurs. Pourquoi ne pas échanger des idées et des procédures sans trahir la philosophie du produit ? Vos idées seront les bienvenues.

Logo a été créé dans le cadre de recherches en intelligence artificielle. On peut, grâce à Logo, étudier les processus de pensée d'une population non formée à l'informatique, car ce langage est simple d'accès. Il n'en est pas pour autant simpliste. Apprendre des mots ne signifie pas comprendre une langue, mais l'acquisition de la maîtrise d'une langue débute par l'apprentissage de mots recouvrant des concepts clairs. On pourrait comparer Logo à un objet très personnel, comme une brosse à dents par exemple. Il ne viendrait à l'idée de personne d'emprunter, d'échanger des brosses à dents — ou des brosses à méninges. Dans cette série d'articles donc, nous allons essayer plutôt d'échanger nos manières de b(r)osser. Nous supposerons de votre part une connaissance des primitives et nous insisterons sur les modes de raisonnement puissants que permet ce langage.

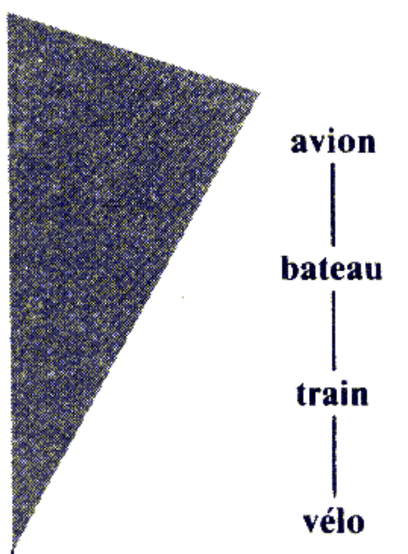
Nous commencerons par nous intéresser aux arbres binaires. Un arbre binaire est une structure dans laquelle tout élément est suivi de deux autres éléments, contrairement à ce qui se passe dans le cas de la liste simple dont chaque article n'a qu'un suivant. L'arbre binaire est un cas particulier des structures arborescentes, elles-mêmes cas particulier des graphes. Dans un arbre, chaque élément, sauf le premier (qui est appelé *racine*) n'a qu'un précédent, alors que dans un graphe, il peut exister plusieurs précédents.

Un arbre à fleurs

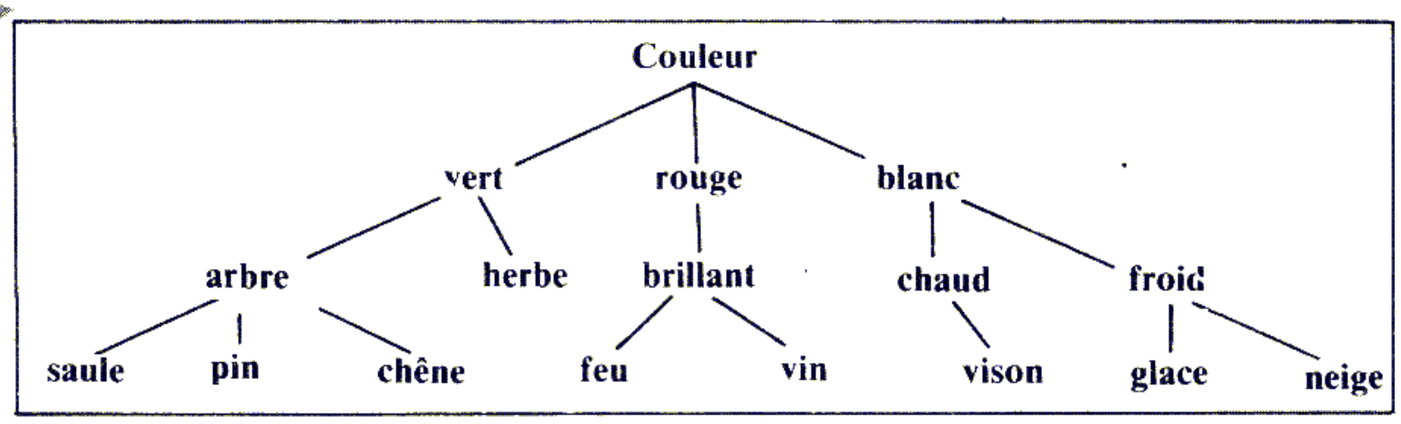
En général, les arbres binaires sont utilisés lorsque la recherche d'un élément peut résulter d'une suite de questions dont la réponse est oui ou non : vaste champ d'applications. Ainsi, dans le cas de la recherche d'une fleur, on aura :

— Est-elle rouge ?
— Non.
— Est-elle bleue ?
— Oui.
— Est-ce une fleur des montagnes ?
— Oui.
— Est-ce un colchique ?
(etc.)

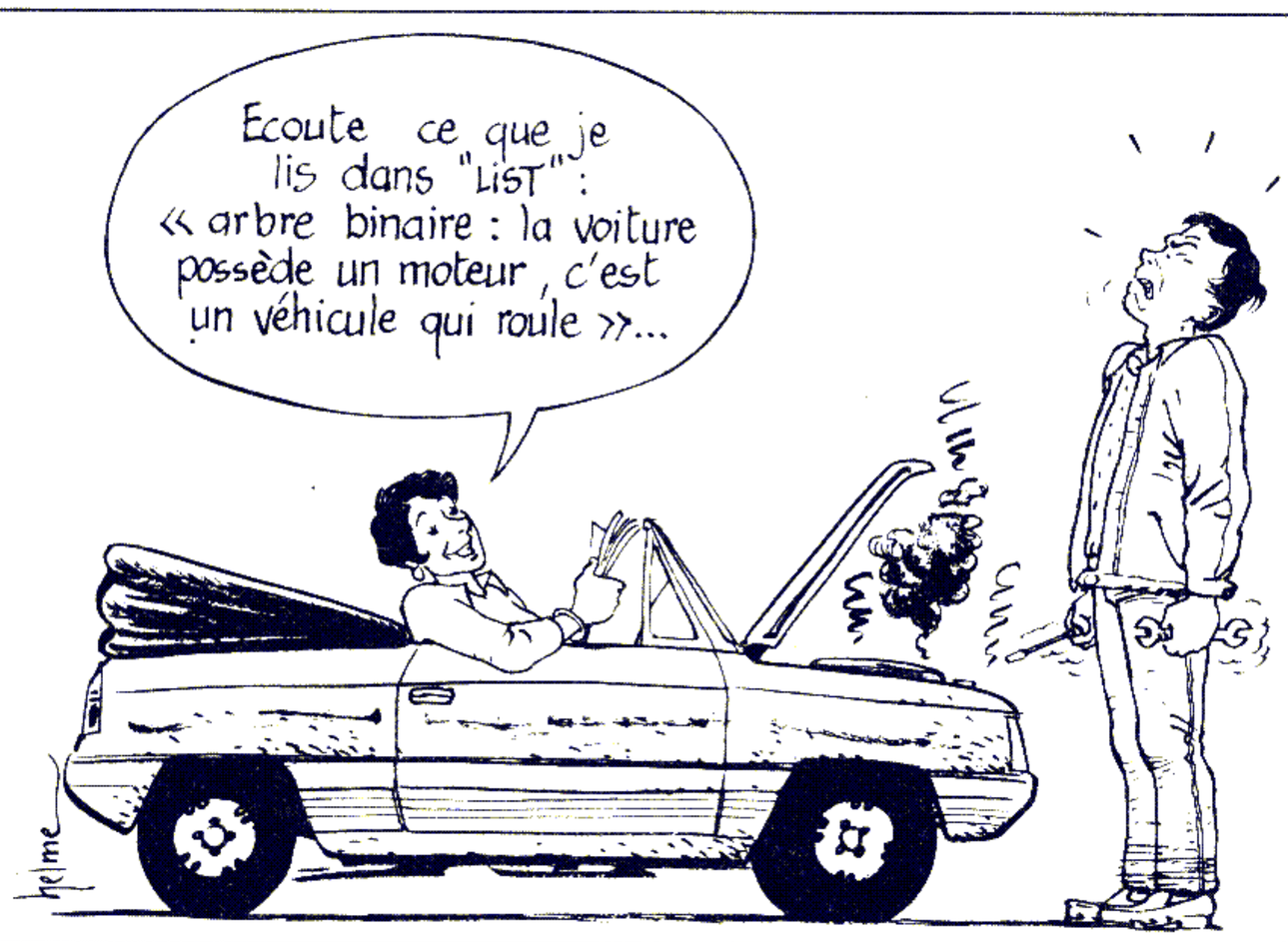
L'arbre binaire correspondant *(Recherche d'un nom de fleur)* est représenté page ci-contre. Que remarque-t-on ? Tout d'abord que chaque branche d'arbre se termine par un nom de fleur ;

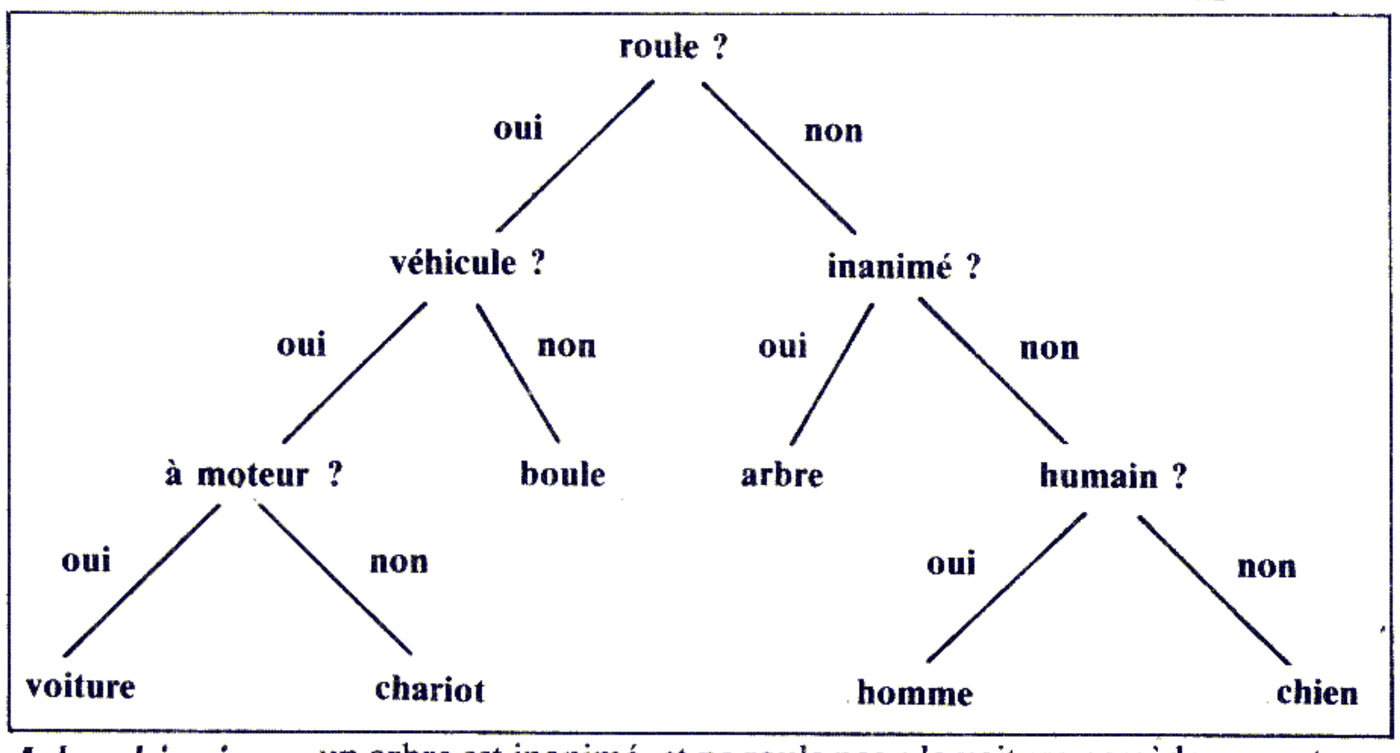


Liste simple :
j'ai pris l'avion, le bateau
puis le train pour finir à vélo.

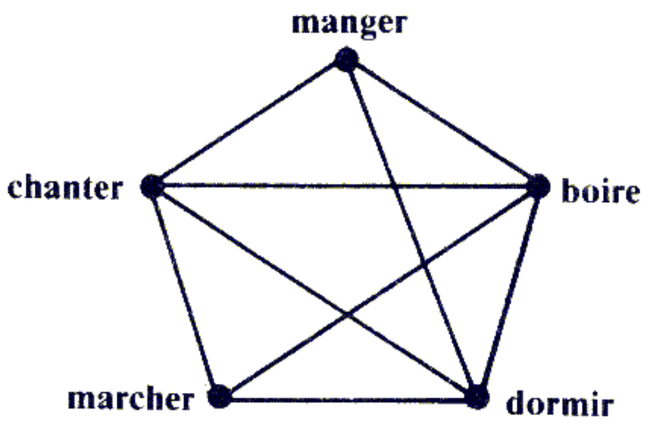


Arborescence : le chêne est un arbre de couleur verte.



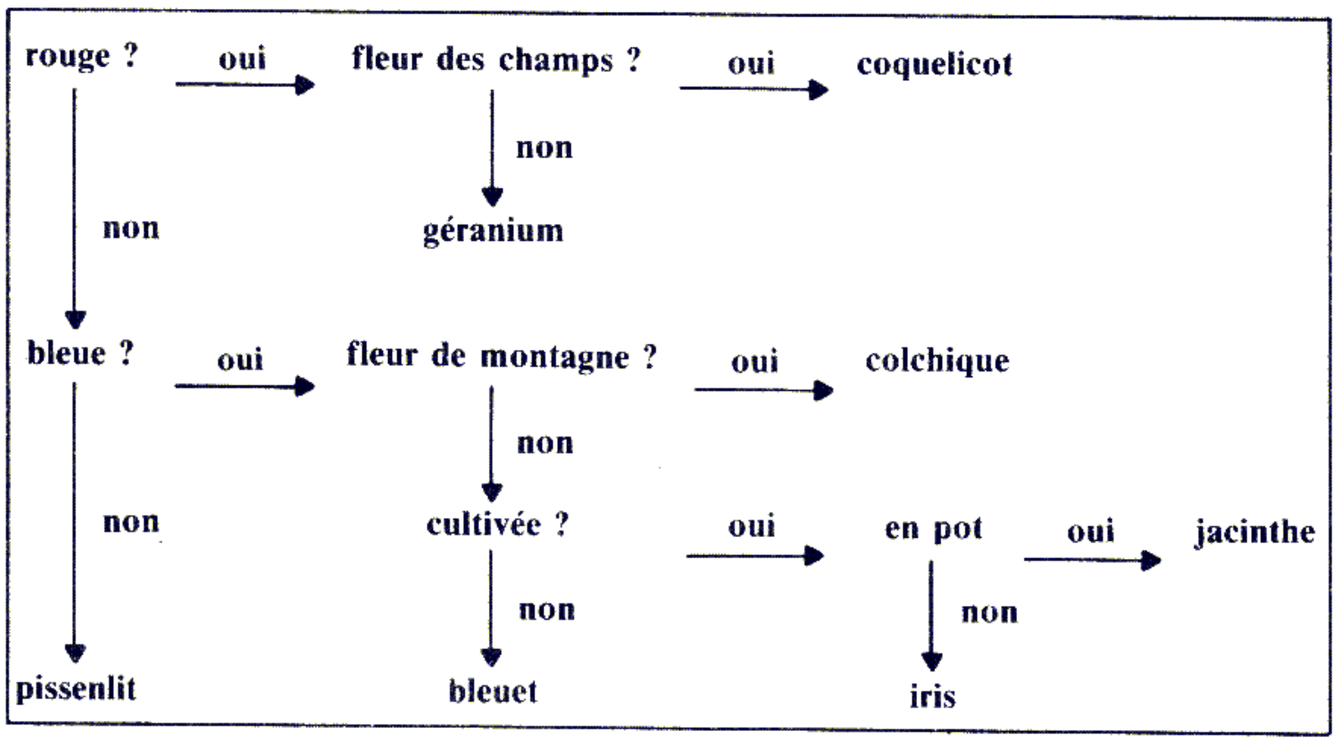


Arbre binaire : un arbre est inanimé, et ne roule pas ; la voiture possède un moteur, c'est un véhicule qui roule.



Graphe :

si je commence par marcher,
je peux faire ensuite
ce que je veux,
sauf manger.



Recherche d'un nom de fleur

l'arbre est et doit toujours être complet. Deuxième remarque : on ne peut ajouter de branche qu'à partir d'une extrémité en remplaçant un nom de fleur par une question qui sera suivie d'une nouvelle réponse en cas de non. Dans notre exemple, l'extrémité *pissenlit* sera remplacée par la question « jaune ? » et par les réponses « pissenlit » et « lys » selon que la réponse sera « oui » ou « non ».

Un arbre sens dessus dessous

Dans la suite de cet article, nous distinguerons les questions, que nous appellerons *nœuds* et les noms de fleurs appelés *feuilles* en langage mathématique. Un arbre binaire est donc constitué d'une racine (premier nœud), de nœuds et de feuilles.

Pour transformer un arbre binaire en une structure de liste, nous considérerons un nœud comme une liste de trois éléments. Le premier élément sera le nœud lui-même, sous forme de liste. Le deuxième élément sera la réponse « oui ». Si cette réponse est un nouveau nœud, l'élément sera une liste, sinon ce sera un mot. Le dernier élément sera la réponse « non ». Ici encore, si la réponse est un nouveau nœud, l'élément sera une liste, sinon un mot. De cette façon, nous pourrons distinguer les nœuds, qui sont des listes, des feuilles qui sont des mots. C'est la méthode la plus simple. L'arbre binaire sera donc une liste des ensembles triples de chaque nœud. Exemple :

```
[...[[cultivée ?][[en pot ?]jacinthe iris]bleuet]]
                   1      2     3
          1                2                3
```

La réponse « non » de « cultivée ? » est « bleuet ». La réponse « oui » est une liste de trois éléments, et donc une nouvelle question. Le tout constitue la réponse « non » à la question « fleur de montagne ? » Cela peut paraître un peu complexe, mais Logo se débrouille très bien dans les listes de listes. De plus, cette structure est très simple :

- la question est toujours le PREMIER élément d'une liste ;
- la réponse OUI en est toujours le second, c'est-à-dire le PREMIER du SAUFPREMIER ;
- la réponse NON est toujours le DERNIER élément.

Un petit peu d'analyse maintenant.

AVEC DES FLEURS

Rappel de la syntaxe LOGO

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

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

En Logo, le problème des arbres binaires se divise en trois parties :

1 - Initialiser l'arbre et lui donner un nom.

2 - Explorer l'arbre jusqu'à ses feuilles.

3 - Ajouter éventuellement une branche par acquisition d'un nouveau nœud et d'une nouvelle feuille.

Nous aborderons aujourd'hui les deux premiers points. Nous étudierons le troisième dans un prochain article, puis nous affinerons les procédures dans le but d'écrire un véritable programme conversationnel pour l'exploitation des arbres binaires.

Pour initialiser l'arbre binaire, nous pouvons le réduire à sa racine et à deux feuilles :

```
POUR INIT.ARBRE
DONNE "ARBRE [[ROUGE?] PIS
SENLIT COQUELICOT]
FIN
```

Nous pourrions même le réduire à une seule feuille, puisque le but du point 3 (ajouter éventuellement une branche) est justement de constituer l'arbre. Cela nous donnerait :

```
POUR ARBRE.MINIMUM
DONNE "ARBRE "PISSENLIT
FIN
```

L'exploration de l'arbre doit se faire au moyen d'aiguillages sur chacun des nœuds, et cela jusqu'à la rencontre d'une feuille. Rien n'est plus simple à obtenir avec une procédure récursive. Le choix de la branche suivante est conditionné par la réponse — oui ou non — de l'utilisateur. La sortie de la procédure intervient lorsque la réponse aiguille sur une feuille, autrement dit sur un mot et non pas sur une nouvelle liste.

Exprimons le problème en français. Pour PARCOURIR un arbre, si l'arbre à parcourir se réduit à une feuille, PROPOSER cette feuille comme solution. Sinon, AFFICHER la question correspondant au premier élément de la liste constituant le reste de l'arbre à parcourir. Si la réponse est alors OUI, PARCOURIR le reste de l'arbre correspondant au deuxième élément de l'arbre initial. Sinon, PARCOURIR le reste de l'arbre correspondant au troisième et dernier élément de l'arbre initial.

Exprimons maintenant le même problème en Logo (Logo-Apple LCSI par exemple) :

```
POUR PARCOURIR :ARBRE
SI MOTP :ARBRE [PROPOSER
:ARBRE STOP]
ECRIS PREMIER :ARBRE
DONNE "REPONSE LISLISTE
SI :REPONSE = [OUI][PARCOURIR
PREMIER SAUFPREMIER :ARBRE]
[PARCOURIR DERNIER :ARBRE]
FIN
```

Proposer une solution

C'est tout. Et cela marche bien à condition que l'utilisateur joue le jeu, et réponde bien par oui ou par non (1). En fait, la variable REPONSE n'est pas indispensable, et l'on aurait pu écrire la dernière ligne comme suit :

```
SI LISLISTE = [OUI] [PARCOURIR
PREMIER SP :ARBRE] [PARCOU
RIR DERNIER :ARBRE]
```

Voyons maintenant ce qui se passe quand il s'agit de proposer une solution, et commençons par exprimer le problème en français. Pour proposer une feuille comme solution :

```
AFFICHER la feuille
AFFICHER la question : LA
REPONSE EST-ELLE CORRECTE ?
```

Si la réponse est OUI, AFFICHER un message triomphal.

Dans le cas contraire, AFFICHER la question : QUELLE EST LA REPONSE ?

Puis AFFICHER une question demandant à l'utilisateur de formuler lui-même une question dont la réponse est OUI pour sa réponse et NON pour celle du programme.

Appeler une procédure qui complètera l'arbre en fonction de son ancienne structure, de la feuille à remplacer et des nouveaux éléments acquis (nouvelle feuille et nouveau nœud).

Exprimons maintenant le même problème en Logo (Apple LCSI) :

```
POUR PROPOSER :FEUILLE
(ECRIS [JE PROPOSE:] :FEUILLE)
ECRIS [EST-CE EXACT ?]
SI LISLISTE = [OUI] [EC [JE SUIS
GENIAL] STOP]
ECRIS [A QUELLE FLEUR
PENSAIS-TU ?]
DONNE "REP.OUI DERNIER
LISLISTE
(ECRIS [QUELLE QUESTION
AURAIS-TU POSEE POUR QUE LA
REPONSE SOIT OUI POUR]
:REP.OUI [ET NON POUR]
:FEUILLE])
DONNE "NOUVEAU.NOEUD
LISLISTE
COMPLETER :ARBRE :REP.OUI
(LISTE :NOUVEAU.NOEUD
:REP.OUI :FEUILLE)
FIN
```

La traduction en Logo est donc quasi immédiate. Rappelons que la réponse OUI doit être un mot — donc DERNIER LISLISTE —, et qu'en Logo LSCI, les parenthèses permettent aux primitives d'accepter un nombre quelconque de paramètres.

La procédure COMPLETER doit être une procédure récursive qui explore l'arbre et le reconstitue en remplaçant la feuille erronée par un nouveau nœud suivi de deux feuilles. Peut-être allez-vous l'écrire seul ? Dans ce cas, j'attends vos propositions. Sinon, à très bientôt pour compléter et affiner le programme.

Robert DAGUESSE

(1) Si l'utilisateur ne joue pas le jeu, il faudra légèrement modifier PARCOURIR (nous verrons comment dans un prochain article).

LES DISQUETTES COMMODORE

TOUT N'EST PAS PERDU

POURSUIVANT plus avant nos investigations dans le domaine mystérieux du système d'exploitation des disquettes du Commodore 64 (en d'autres termes, son DOS, disk operating system), voici un nouvel utilitaire qui permettra de remettre au catalogue un ou plusieurs fichiers détruits accidentellement ou par manque de prudence lors de certaines manipulations.

Détruit... ou escamoté ?

Si les études présentées ici sont destinées spécialement aux utilisateurs du C.64, elles sont presque toujours adaptables sur les autres modèles de la gamme Commodore.

Avant tout, les points essentiels concernant les disquettes doivent être rappelés. La piste 18 y joue le rôle de piste maîtresse.

Située en partie médiane de la zone d'écriture, elle renferme toutes les indications nécessaires au système pour l'utilisation de ce support magnétique.

Elle contient en particulier, commençant sur le secteur numéro 1, toutes les indications qui sont affichées sur l'écran lors d'un LOAD"$",8.

On y trouve donc, inscrits sur les secteurs successifs :

- le type du fichier, sous la forme d'un seul octet ;
- le numéro de la piste, et celui du secteur où débute l'implantation du fichier (2 octets) ;
- le titre, sous sa forme ASCII, constitué de 16 octets dont les derniers sont des CHR$(160) (espaces shiftés) lorsque le titre proprement dit a moins de 16 caractères ;
- le nombre de secteurs (ou BLOCKS !) occupés par ce fichier sur la disquette.

Un certain nombre de ces éléments sont gardés secrets lors de l'affichage du catalogue. Il s'agit en particulier, des deux octets représentant les numéros de piste et de secteur. Quant à l'octet du type de fichier, il décide de l'affichage ou du non-affichage de tous les autres renseignements sur le fichier.

Ainsi, un fichier de type DEL (donc, détruit), n'apparaît jamais au catalogue.

LES DISQUETTES COMMODORE

TOUT N'EST PAS PERDU

Outre le type DEL, quatre autres types de fichiers sont courants : SEQ (séquentiel), PRG (programme), USR (utilisateur) et REL (relatif).

Mais le fait qu'un fichier DEL ne figure plus au catalogue, signifie-t-il sa disparition totale de la disquette ? Autrement dit, tous les octets qui composent ce fichier sont-ils éliminés à tout jamais du support magnétique ?

Heureusement, il n'en est rien. Lors de la destruction volontaire d'un fichier, le SED (Système d'Exploitation des Disquettes) se contente de remplacer l'octet qui représente le type de fichier sur

Récupérateur de fichiers
Programme pour C.64 et lecteur de disquettes
Auteur Jean-Pierre Lalevée

Utilisation du programme

Il sera prudent, comme à chaque fois, de tester le programme sur une disquette sans importance. Une erreur dans la transcription risquant d'être fatale à tous les fichiers qu'elle contient. Ce n'est qu'après vérification minutieuse que vous pourrez sans risque l'appliquer sur une disquette contenant de précieux programmes.

La manipulation de l'utilitaire ne pose pas de problème particulier. Le programme commence par passer en revue la totalité du catalogue de la disquette. Lorsqu'il découvre un fichier détruit, le titre correspondant est affiché à l'écran, et le programme vous interroge pour savoir si vous souhaitez le remettre au catalogue. Si votre réponse est positive, le comptage des blocs commence.

Si le résultat de cette investigation est correct, vous devez indiquer au programme le type du fichier concerné (SEQ, PRG, USR ou REL). Le catalogue est alors modifié, puis une VALIDATION est effectuée, pour remettre à jour la BAM. Ce processus se poursuit jusqu'à épuisement du catalogue de la disquette.
Un affichage du catalogue, suivi d'un chargement du fichier récupéré, vous assurera de la réussite de l'opération...

```
100 REM ******************************
110 REM *  UTILITAIRE DISQUE N.4      *
120 REM * RECUPERATEUR DE FICHIERS    *
130 REM ******************************
140 :
150 PRINT"■","■ RECUPERATEUR DE FICHIERS "
160 PRINT"■PERMET DE REMETTRE AU CATALOGUE LES FICHIERS DETRUITS."
170 :
180 C0$=CHR$(0)
190 DE=0:REM NB DE FICHIERS DEL
200 DIM S(35):REM NB DE SECTEURS/PISTE
210 FOR I=1 TO 17:S(I)=21:NEXT
220 FOR I=18 TO 24:S(I)=19:NEXT
230 FOR I=25 TO 30:S(I)=18:NEXT
240 FOR I=31 TO 35:S(I)=17:NEXT
250 :
260 REM ****************** INITIALISATIONS DISQUE ********************
270 OPEN 15,8,15,"I0":GOSUB 1130
280 OPEN 2,8,2,"#":GOSUB 1130
290 :
300 REM ************************ TITRE ET ID ***************************
310 PRINT#15,"U1"2;0;18;0:GOSUB 1130
320 PRINT#15,"B-P"2;144:GOSUB 1130
330 :
340 FOR I=0 TO 19:GET#2,A$:F$=F$+A$:NEXT I
350 PRINT"■■■ TITRE: ■";LEFT$(F$,16);"■ ID: ■";RIGHT$(F$,2)
360 :
370 REM ************************ PROG PPAL *****************************
380 P=18:S=1
390 PRINT#15,"U1"2;0;P;S:GOSUB 1130
400 PRINT#15,"B-P"2;0:GOSUB 1130
410 :
420 REM +++++++++++++++++++ EMPLACEMENT SUIVANT ++++++++++++++++++++++
430 GET#2,P$:P1=ASC(P$+C0$):REM PISTE
440 GET#2,S$:S1=ASC(S$+C0$):REM SECTEUR
450 :
460 FOR J=0 TO 7:REM IL Y A 8 TITRES MAXI PAR BLOC
470 :
480 REM ++++++++++++++++++++ TYPE DU FICHIER ++++++++++++++++++++++++
490 PRINT#15,"B-P"2;J*32+2
500 GET#2,X$:X=ASC(X$+C0$) AND 15
510 IF X THEN 990:REM SI FICHIER NON DEL
520 :
530 REM ++++++++++++++++++++++ PISTE, SECTEUR ++++++++++++++++++++++++
540 GET#2,N$:PF=ASC(N$+C0$)
550 IF PF=0 THEN 990:REM PLUS DE TITRES ?
560 GET#2,N$:SF=ASC(N$+C0$):REM PF,SF : PISTE & SECT DE DEBUT
570 :
580 REM ++++++++++++++++++++++ TITRE FICHIER ++++++++++++++++++++++++
590 DE=1
600 PRINT"TITRE DU FICHIER :";CHR$(34);
610 FOR I=1 TO 16:GET#2,A$:IF ASC(A$)<>160 THEN PRINT A$;
620 NEXT I:PRINT CHR$(34)
630 INPUT"■ON RECUPERE ";R$
```


```
640 IF R$="N" THEN 990
650 IF R$<>"O" THEN 630
660 :
670 REM ++++++++++++++++++++++ NB DE SECTEURS ++++++++++++++++++++++++
680 PRINT#15,"B-P"2;J*32+30:GOSUB 1130
690 GET#2,L$,H$:L=ASC(L$+C0$):H=ASC(H$+C0$)
700 NS=H*256+L
710 :
720 REM ++++++++++++++++++ VERIFICATION DES BLOCS ++++++++++++++++++++
730 CB=-1:PRINT"TEST DE BLOCS (";NS;")."
740 CB=CB+1
750 IF PF=0 THEN 840:REM FIN DU FICHIER
760 IF PF>35 THEN 840:REM PISTE ILLEGALE
770 IF SF>S(PF) THEN 840:REM SECT ILLEGAL
780 PRINT#15,"U1"2;0;PF;SF:GOSUB 1130
790 PRINT#15,"B-P"2;0:GOSUB 1130
800 GET#2,PF$,SF$
810 PF=ASC(PF$+C0$):SF=ASC(SF$+C0$)
820 GOTO 740
830 :
840 PRINT:IF CB<>NS THEN PRINT"COMPTE DE BLOCS INCORRECT.":GOTO 990
850 PRINT"TOUS BLOCS OK."
860 :
870 REM ++++++++++++++++++ REMISE AU CATALOGUE ++++++++++++++++++++
880 PRINT"1- SEQ":PRINT"2- PRG":PRINT"3- USR":PRINT"4- REL"
890 INPUT"TYPE DU FICHIER ";Y$
900 Y=VAL(Y$):IF Y<1 OR Y>4 THEN 890
910 Y=128+Y
920 PRINT#15,"U1"2;0;P;S:GOSUB 1130
930 PRINT#15,"B-P"2;J*32+2:GOSUB 1130
940 PRINT#2,CHR$(Y);:GOSUB 1130
950 PRINT#15,"B-P"2;0:GOSUB 1130
960 PRINT#15,"U2"2;0;P;S:GOSUB 1130
970 :
980 PRINT"■FICHIER RETABLI."
990 NEXT J
1000 :
1010 IF P1 THEN P=P1:S=S1:GOTO 390:REM SUITE ?
1020 :
1030 REM ++++++++++++++++++++ FIN DE RECHERCHE ++++++++++++++++++++++
1040 PRINT"■RECHERCHE TERMINEE."
1050 IF DE=0 THEN PRINT"■PAS DE FICHIER ■DEL■ DETECTE.":GOTO 1090
1060 PRINT"■VALIDATION DISQUE EN COURS."
1070 PRINT#15,"V0":GOSUB 1130
1080 :
1090 CLOSE 2:CLOSE 15
1100 GOTO 1170
1110 :
1120 REM ********************* S/P ERREURS DISQUE **********************
1130 INPUT#15,E1,E$,E3,E4
1140 IF E1>20 THEN PRINT"■ERREUR DISQUE :":PRINT E1","E$","E3","E4
1150 RETURN
1160 :
1170 END
READY.
```

la piste 18, par un zéro. C'est l'indicateur de type DEL.

Quant au fichier, il reste présent intégralement sur la disquette, en attente d'être remplacé par un nouveau fichier qui viendra ainsi, à la première occasion, détruire sa belle ordonnance.

Un repêchage bloc par bloc

Cette mise à zéro d'un seul octet n'est pas la seule action du SED lors de la destruction d'un fichier : son second travail consiste à indiquer sur la « carte des blocs disponibles » (BAM, Block Availability Map) que les secteurs précédemment occupés par le fichier redeviennent libres et utilisables.

Il faudra donc, pour remettre en fonction un fichier détruit, effectuer les opérations inverses :

- remettre à sa valeur initiale l'octet indiquant le type du fichier (c'est l'utilisateur qui devra fournir lui-même cette indication qui ne figure plus nulle part ailleurs sur la disquette) ;
- remettre en état la BAM pour indiquer au système que les blocs sont à nouveau occupés (cette nouvelle allocation est facile à exécuter avec la commande VALIDATE dont c'est précisément la fonction) ;
- compter les blocs réellement occupés par le fichier à récupérer et comparer cette valeur à celle qui figure sur le catalogue.

Cette dernière action devra être menée afin de garantir l'efficacité de la récupération du fichier. En effet, un fichier détruit risque d'être écrasé à la première occasion par un nouveau fichier enregistré sur la disquette. Dans ce cas, la perte est irrémédiable, les octets du nouveau fichier venant remplacer tout ou partie de l'ancien fichier. L'action proposée est une manière de contrôler ce risque. Elle n'est pas absolue.

Et si les deux valeurs obtenues ne correspondent pas, la récupération peut être considérée comme impossible : adieu fichier ! Dans le cas contraire, le fichier est très probablement retrouvé. Ceci explique qu'il vaudra mieux éviter de trop attendre avant de récupérer un fichier. Le mieux est d'agir dès que l'erreur est constatée !

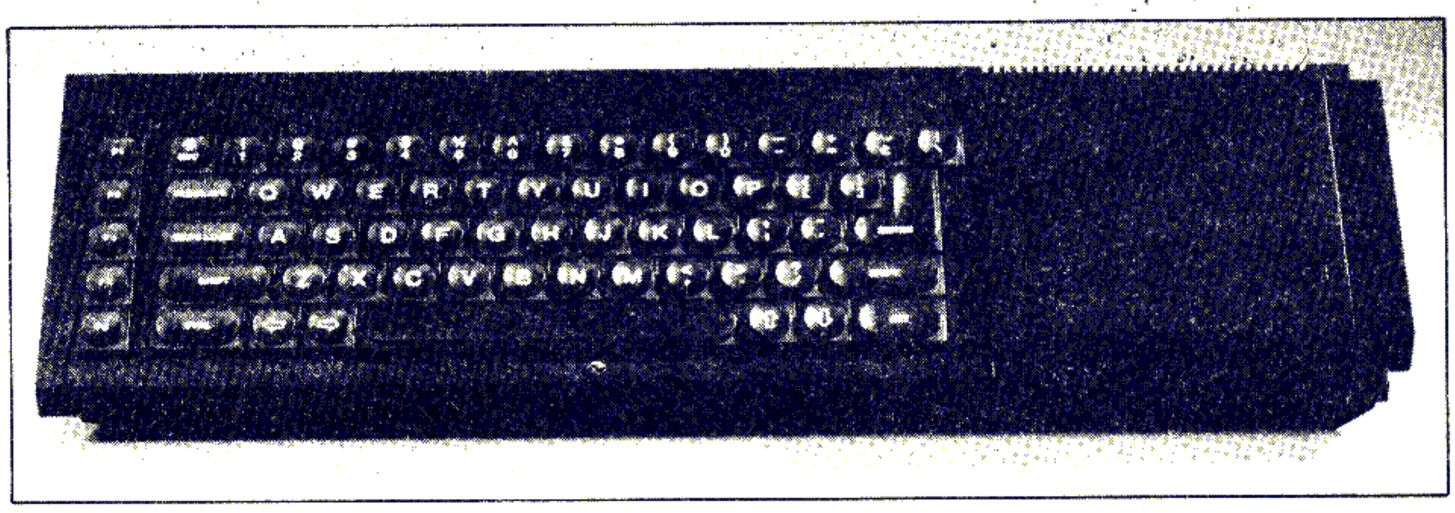
Jean-Pierre LALEVÉE

LE BASIC DU QL

ON sait que Sinclair aime sortir des sentiers battus. Avec le QL, dernier né de sa gamme, il propose un Basic très original sur plusieurs points et qui, en particulier, se prête bien à la programmation structurée.

Après avoir longtemps joué à cache-cache avec ceux qui l'attendaient, le QL fait son apparition sur le marché français. Peut-être sera-t-il disponible en version AZERTY lorsque vous lirez ces lignes, ce qui fera tout de même plusieurs mois de retard par rapport à l'annonce initiale.

Actuellement, le QL est livré avec un transformateur d'alimentation extérieur au boîtier, un câble péritel et un câble UHF. On peut donc le connecter aussi bien à un téléviseur couleur que noir et blanc. Si l'on veut utiliser un moniteur, il faut se procurer le câble correspondant. Contrairement aux apparences, le clavier n'est pas vraiment professionnel : les touches ont une course un peu trop molle sur la fin et trop longue, mais ce défaut n'est pas suffisamment prononcé pour gêner les programmeurs.



Téléviseur ou moniteur ?

Avant d'entrer dans le vif du sujet, c'est-à-dire le Basic du QL, il nous faut d'abord dire un mot sur la façon dont l'ordinateur communique avec ses périphériques. Le QL est relié au monde extérieur par l'intermédiaire de ce que Sinclair appelle des canaux. Ces canaux peuvent déboucher soit sur une fenêtre de l'écran, soit sur un « micro-drive », soit sur une interface RS-232 C, soit sur un réseau comprenant d'autres QL ou Spectrums. Pour entrer en communication avec un périphérique ou une fenêtre, il suffit d'ouvrir un canal et de le lui assigner.

Un design très pur qui joue sur la sobriété

A la mise sous tension, trois canaux sont attribués d'office :

- le canal 0 est relié à la fenêtre de l'entrée des commandes et de l'édition ;
- le canal 1 est attribué à la fenêtre où s'affichent les résultats du programme ;
- le canal 2 correspond à la fenêtre dans laquelle apparaissent les listes de programmes.

Au démarrage, le QL donne le choix entre deux options. Si l'on choisit la première (elle est destinée aux heureux possesseurs d'un moniteur), la définition est de 512 sur 256 points et l'on dispose de 25 lignes de 84 caractères. L'écran est alors divisé en 3 parties occupées par la fenêtre commande et édition, la fenêtre liste et la fenêtre exécution. La deuxième option correspond à l'utilisation d'une TV couleur. La définition est alors de 256 × 256, on a 25 lignes de 42 caractères et la fenêtre des listes est confondue avec la fenêtre exécution.

En fait, la qualité de l'image obtenue sur le téléviseur qui a servi à cet essai (modèle très récent, il est vrai) est excellente. D'ailleurs, même en 84 colonnes, les caractères sont nets ; une seule ombre vient ternir le tableau : les deux lignes du bas, ainsi que les caractères de la première et de la dernière colonnes sont tronqués, mais il est possible de redéfinir les fenêtres d'affichage.

Première constatation, l'éditeur du QL n'est pas vraiment plein écran. Ce n'est pas une tare en soi à condition que la conception se révèle sérieuse, et c'est le cas. Cet éditeur est construit autour de la commande EDIT suivie d'un numéro de ligne. La ligne demandée s'affiche dans la fenêtre commande/édition qui occupe, sauf redéfinition, cinq lignes en bas de l'écran. On corrige ladite ligne en détruisant les caractères situés à droite ou à gauche du curseur, et en en insérant de nouveaux.

La ligne ainsi modifiée ne pourra être validée que lorsqu'elle sera syntaxiquement correcte. Si la personne qui programme ne parvient pas à repérer où se trouve l'erreur, seule une pression simultanée sur la touche d'espace et sur CTRL permet de sortir du mode édition. Après validation, la ligne est instantanément corrigée sur la fenêtre des listes.

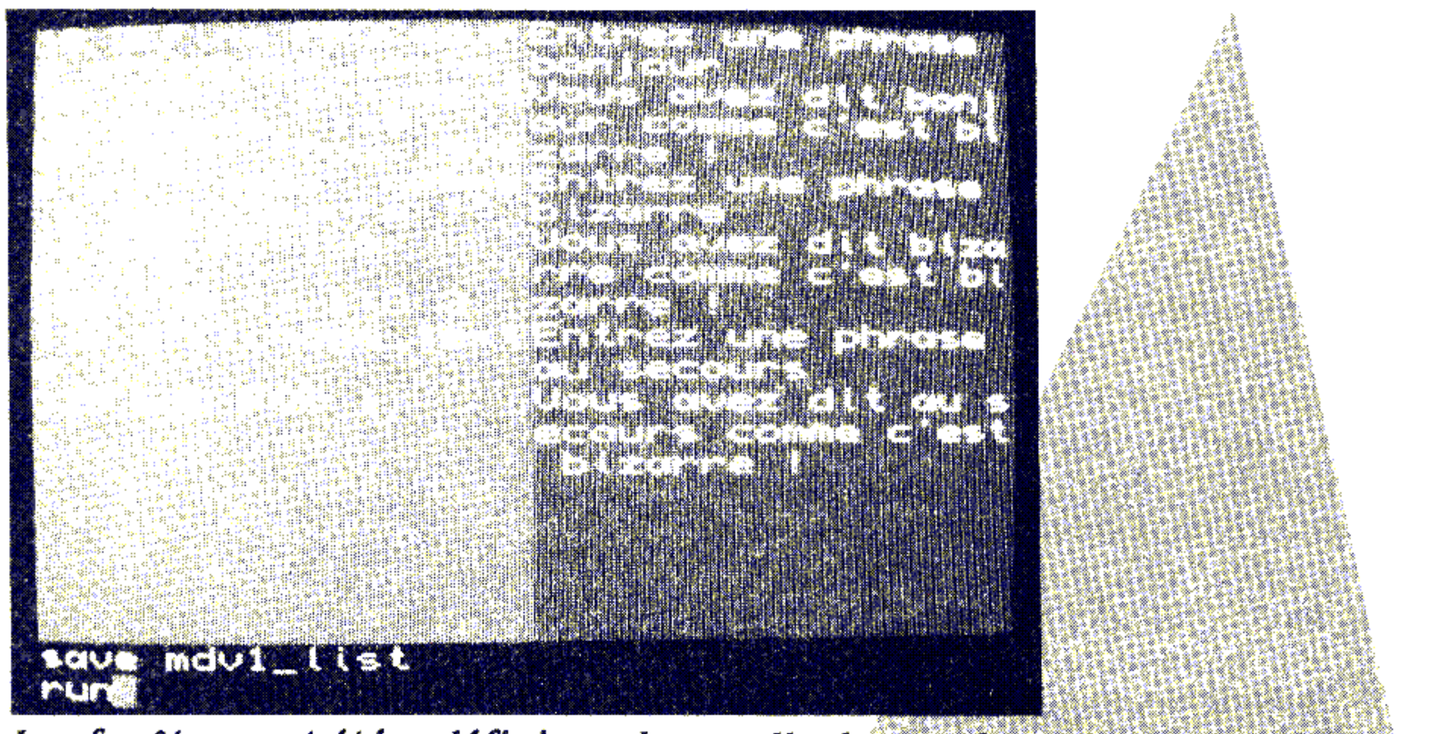
Pour créer une ligne, il suffit évidemment de la taper et, tant qu'elle n'a pas été validée, les différentes commandes

d'édition sont disponibles. En fait, une bonne surprise nous attend : dans ce mode, la frappe sur une des flèches verticales fait apparaître, dans la fenêtre d'édition, la ligne précédant ou suivant la dernière ligne éditée. Il est ainsi possible de monter ou de descendre à volonté dans le programme : on n'est pas si loin de l'éditeur plein écran.

Les autres commandes de l'édition sont AUTO, RENUM (rénumérotation de tout ou partie d'un programme), et DLINE qui détruit le groupe de lignes spécifié : même rôle donc que le DELETE de Microsoft. En réalité, DELETE existe aussi dans le Basic QL, mais il concerne la gestion des « microdrives » où il remplace l'ordre KILL...

Les mots clés peuvent être abrégés, le QL rajoutant seul, en minuscules, les lettres manquantes. Exemple : si l'on frappe DEF FN, il apparaîtra sur la liste DEFine-FuNction.

Le plus souvent, un programme informatique traite des informations qui se présentent sous forme de variables. C'est assez dire si les variables sont importantes. Voyons ce dont est capable le QL dans ce domaine.



Les fenêtres ont été redéfinies : dans celle de gauche, la liste du programme exécuté dans celle de droite.

Malheureusement pas de double précision

Les noms de variables peuvent atteindre jusqu'à 255 caractères de longueur et sont composés de majuscules, de minuscules ou de chiffres. Mises à part les variables alphanumériques, dont le type sera spécifié par le suffixe $, le QL n'admet que deux types de variables numériques :

1 - Les variables réelles dont les valeurs sont comprises entre E + 615 et E – 615, gigantesque intervalle, mais la précision n'est que de 8 chiffres. Quand on sait que le Basic du QL ne permet pas la double précision, il y a de quoi être un peu déçu.

2 - Les variables entières codées classiquement sur deux octets et donc comprises entre –32768 et +32767.

L'ordre DIM permet, comme on s'y attend, de déclarer un tableau dont la dimension est quasi illimitée, mais il y a moins courant : en Basic QL, une variable n'est créée que si le programme rencontre une instruction attribuant une valeur à cette variable. Ainsi, les lignes 10 A = 1 ou 10 INPUT B provoqueront la création de la variable A. En revanche, ON C GOTO 100 provoquera un message d'erreur si C n'a encore reçu aucune valeur. De la même façon, si la variable C n'a pas été créée, l'exécution d'une ligne 10 PRINT C provoquera l'affichage d'une étoile, et non d'un zéro (comme ce serait le cas sur d'autres Basic).

A propos des variables, ajoutons que l'interpréteur est capable d'effectuer des conversions de type lorsqu'une opération ne peut avoir lieu : 10 PRINT 1 + "2" affichera 3. Si cette possibilité paraît intéressante, elle ne va guère dans le sens d'une programmation rigoureuse.

Le Basic QL se démarque nettement de la plupart des Basic type Microsoft. Si vous le voulez bien, jouons un instant au jeu des différences. La ligne qui s'écrit habituellement : 10 IF A = 1 LET X = 1 : Y = 2 : GOTO 1000 : ELSE IF B = 1 THEN 2000 ELSE 3000 pourra s'écrire en Basic QL :

```
 10 IF A=1
 20    LET X=1 : Y=2
 30    GO TO 1000
 40       ELSE
 50          IF B=1
 60             GO TO 2000
 70                ELSE
 80                GO TO 3000
 90          END IF
100 END IF
```

Vous me direz que le QL emploie plus de lignes ! C'est exact, mais avouez que la version en 10 lignes est plus claire et mieux structurée. D'ailleurs, qu'importe la longueur quand on dispose d'une mémoire vive de plus de 70 Ko. Pour la petite histoire, sachez que j'ai dû créer une routine pour mesurer l'espace disponible car, apparemment, aucune fonction de type MEM ou FREE n'existe.

Toujours au chapitre du contrôle des structures, il est possible, comme en Pascal, de créer des procédures. Qu'est-ce qu'une procédure ? C'est un sous-programme très perfectionné auquel on donne un nom et que l'on appelle ensuite par ce nom en lui communiquant éventuellement des paramètres. Reprenons notre jeu des différences. Imaginons que nous voulons créer un sous-programme affichant la moyenne arithmétique de trois nombres. En Basic classique on pourrait écrire :

```
10 PRINT (X1+X2+X3)/3
20 RETURN
```

Si nous voulons utiliser ce sous-programme pour calculer la moyenne des nombres 10, 15 et 20, il faudra faire :

```
100 X1=10 : X2=15 : X3=20
110 GOSUB 10
```

Avouez que ce « GOSUB 10 » n'est pas très explicite ! En Basic QL, on créera la procédure MOYENNE comme ceci :

```
10 DEF PROCedure MOYENNE (A,
   B, C)
20 PRINT (A+B+C)/3
30 END DEFine
```

Jusque-là, peu de différence mais, pour utiliser cette procédure, il suffit de citer son nom, suivi de ses trois paramètres : 100 MOYENNE 10, 15, 20 provoquera l'affichage de 15.

Les instructions GOSUB et RETURN deviennent ainsi inutiles, et si elles sont néanmoins présentes, c'est pour assurer une certaine compatibilité avec la plupart des autres Basic. Dans le même souci de compatibilité, le QL a inclus dans sa panoplie les instructions ON...GOTO et ON...GOSUB faisant l'une comme l'autre double emploi avec la structure SELect...ENDSELect, aussi puissante que le CASE...OF du Pascal.

Ajoutons, pour en terminer avec les procédures, qu'il est possible d'y déclarer des variables dites « locales », c'est-à-dire qui ne vaudront qu'à l'intérieur de la procédure. Cela permet, par exem-

ple, d'employer les traditionnels indices de boucles I, J et K aussi bien dans le programme principal que dans les sous-programmes, et cela sans observer la moindre interaction néfaste.

Une autre structure inspirée de Pascal est présente dans le Basic du QL, c'est REPeat...ENDREPeat qui définit les deux extrémités d'une boucle. La sortie se produit grâce à l'instruction EXIT placée « quelque part » entre REPeat et ENDREPeat. Pourquoi ne pas avoir choisi un véritable REPEAT...UNTIL ?

On s'éloigne du standard

Au chapitre des boucles, d'ailleurs, on constate que les très classiques FOR...NEXT sont là, mais timidement pour deux raisons au moins. Tout d'abord parce que NEXT peut être remplacé par ENDFOR (décidément le QL aime beaucoup les END-quelque-chose). Ensuite parce que cette instruction permet certaines fantaisies. Ainsi, quand on place une séquence d'instructions sur la même ligne que FOR, NEXT n'est pas nécessaire. D'autres possibilités intéressantes sont offertes : une ligne comme "10 FOR I = 0, 1, 2, 5 TO 8 : PRINT I;" provoquera l'affichage de "0 1 2 5 6 7 8".

Pour le traitement des chaînes de caractères aussi, les concepteurs de ce Basic ont conservé un minimum de compatibilité avec le langage standard. Si l'on retrouve bien LEN et INSTR (et encore avec une syntaxe bizarre pour INSTR), ils ont rebaptisé ASC et STRING$ qui deviennent respectivement CODE? et FILL$. Pourquoi, d'autre part, ne pas implémenter les habituels MID$, RIGHT$ et LEFT$? Sans doute parce que l'on peut facilement travailler sur les chaînes avec la seule instruction TO : 10 A$ = "BON JOUR" et 20 PRINT A$ (2 TO 5) affichera ONJO. Mais il y a plus fort :

```
10 A$ = "BONJOUR"
20 B$ = "BLEU"
30 A$ (3 TO 5) = B$ (1 TO 3)
40 PRINT A$
```

affichera BOBLEUR.

Comme nous l'avons évoqué plus haut, le QL offre la possibilité de redéfinir les fenêtres d'affichage ou d'en créer de nouvelles. Pour ouvrir une fenêtre reliée au canal 6, on pourra taper la séquence : open # 6, scr – 128 × 128 a 0 × 0. On aura ainsi défini une fenêtre occupant un huitième de l'écran et dont le coin supérieur gauche est situé au point de coordonnées 0,0 (en haut, à gauche). Si l'on ne spécifie pas la taille de la fenêtre, l'écran est occupé en totalité.

Pour lister le programme en mémoire sur la fenêtre 6, il suffit de faire LIST # 6. Pour faire tourner un programme sur cette même fenêtre, un simple RUN # 6 fait l'affaire. Parmi les autres commandes liées à la gestion de ces fenêtres, citons PAPER et BORDER qui permettent respectivement de redéfinir la couleur d'une fenêtre et de créer un cadre ; AT permet de positionner le curseur; INK fixe la couleur des caractères d'une fenêtre ; OVER offre des possibilités de surimpression. Avec CLS (tout le monde l'a deviné), on efface la fenêtre spécifiée dans la couleur désirée. Enfin, grâce à CSIZE, on peut redéfinir pour chaque fenêtre la taille des caractères affichés. Tout cela est très souple, mais un regret tout de même : certaines de ces instructions ne sont disponibles qu'en mode moniteur.

Le graphisme est un des domaines de prédilection de cette machine. Tout d'abord, la définition graphique est excellente : 512 × 256 en mode moniteur, c'est assez rare. Inutile d'entrer dans le détail de chaque instruction ; il suffit de savoir que presque tout est possible : tracer une ligne, un cercle, un arc de cercle, un ellipse, allumer un point, un pavé de points, mettre en mouvement et diriger des figures. On peut tout aussi bien travailler en coordonnées relatives ou absolues, ou encore redéfinir l'origine. Il est même possible de « piloter » une tortue grâce à un jeu d'instructions complet (comme quoi le Logo fait école).

Dans le domaine des fonctions mathématiques, Sinclair a largement débordé du strict nécessaire qui sévissait souvent sur les machines de table. C'est ainsi que sont présentes toutes les fonctions trigonométriques (y compris cotangente) et leurs inverses, et que les fonctions DEG et RAD permettent les conversions d'angles. Malheureusement, ici non plus, on n'a pas toujours repris les appellations standard : ATN devient ATAN, ACS se mute en ACOS et ainsi de suite... Tant pis pour les vieilles habitudes.

Pour le reste, le Basic du QL met à la disposition du mathématicien la division entière (DIV), le reste de cette division (MOD), les logarithmes décimaux et népériens (LOG10 et LN) et l'extraction de la racine carrée (SQRT). Toutes ces fonctions peuvent être complétées par de nouvelles grâce à la structure DEFine FuNction... ENDEFine, beaucoup plus puissante que le DEFFN des Basic classiques. Elle permet en effet de définir une fonction sur plusieurs lignes, et donc d'introduire des tests, des appels à des sous-programmes, etc.

Liste des mots clés du Basic du QL

ABS
ACOS
ACOT
ADATE
ARC
ARC-R
ASIN
AT
ATAN
AUTO
BAUD
BEEP
BEEPING
BLOCK
BORDER
CALL
CHR$
CIRCLE
CIRCLE-R
CLEAR
CLOSE
CLS
CODE
CONTINUE
COPY
COPY-N
COS
COT
CSIZE
CURSOR
DATA
DATE$
DAY$
DEG
DEFFN
DEFPROC
DELETE
DIM
DIMN
DIR
DIV
DLINE
EDIT
ENDDEF
ENDFOR
ENDIF
ENDREP
ENDSEL
EOF
EXEC
EXEC-W
EXIT
FILL
FILL$
FLASH
FOR...STEP...NEXT
FORMAT
GOSUB
GOTO
IF...THEN...ELSE
INK
INKEY$
INPUT
INSTR
INT
LBYTES
LEN
LET
LINE
LINE-R
LIST
LN
LOAD
LOCAL
LOG10
LRUN
MERGE
MOD
MODE
MOVE
MRUN
NET
NEW
ON...GOTO
ON...GOSUB
OPEN
OPEN-IN
OPEN-NEW
OVER
PAN
PAPER
PAUSE
PEEK
PEEK-W
PEEK-L
PENUP
PENDOWN
PI
POINT
POINT-R
POKE
POKE-W
POKE-L
PRINT
RAD
RANDOMIZE
READ
RECOL
REM
REP
RESPR
RESTORE
RET
RETRY
RUN
SAVE
SBYTES
SCALE
SCROLL
SDATE
SEL
SEXEC
SIN
SQRT
STOP
STRIP
TAN
TURN
TURNTO
UNDER
WIDTH
WINDOW

LIST - PAGE 62

Fiche technique du Sinclair QL

Constructeur : *Sinclair*
Prix public : *environ 6 000 FF*
Processeur : *68008*
Mémoire vive : *128 Ko en version de base dont 32 Ko pour l'écran et 72 Ko environ pour l'utilisateur*
Mémoire morte : *48 Ko pour Basic et QDos*
Langage : *Basic résident (130 mots clés)*
Affichage : *512 x 256 points en 4 couleurs et/ou 25 lignes de 84 caractères. 256 x 256 points en 8 couleurs et/ou 25 lignes de 42 caractères*
Variables : *8 chiffres significatifs pour les réels avec exposant de -615 à +615. 2 octets pour les entiers (-32768 à +32767). Variables alphanumériques de longueur pratiquement illimitée.*

Une précision qui a son importance : une fonction, de même qu'une procédure, peut s'appeler elle-même. Le Basic du QL connaît donc la récursivité sans restriction.

Quand Sinclair innove, les uns crient au miracle, les autres font la moue, mais beaucoup se demandent comment le constructeur britannique fait pour créer des ordinateurs aussi bon marché. Eh bien, il adopte des compromis. Quand on lit pour la première fois sa fiche technique, le QL peut paraître très séduisant. Certains « détails » toutefois font tiquer. Si le clavier est sans contexte meilleur que celui d'un ZX (sensitif) ou d'un Spectrum première manière (touches à la gomme) il reste d'une qualité moyenne. Mais ce sont les fameux « micro-drives » qui ont suscité le plus d'inquiétude.

Un DOS sans suprise

Les commandes de QDos, directement accessibles à partir du Basic, peuvent être considérées comme une extension de celui-ci. Elle sont, dans l'ensemble, classiques : FORMAT, bien entendu, formate une cartouche vierge, et COPY effectue, non moins évidemment, la copie d'un programme d'un drive sur un autre. Cette dernière commande sert également à transmettre un programme vers l'imprimante, la syntaxe étant COPY *périphérique1* TO *périphérique2*.

BAUD permet de spécifier la vitesse de transmission par une des interfaces série (de 75 à 19200 bauds). LOAD, SAVE et DELETE assurent respectivement la sauvegarde, la relecture et la suppression d'un programme (LRUN charge et lance le programme spécifié). DIR affiche l'espace restant libre et la liste des programmes enregistrés sur une cartouche. MERGE considère un fichier comme un programme Basic et le fusionne avec le précédent en détruisant éventuellement les lignes qui se chevauchent ; les erreurs de syntaxe sont signalées (l'origine du fichier en question est en effet quelconque et peut, par exemple provenir d'un autre ordinateur, sous forme ASCII, par l'intermédiaire d'une RS 232 C). Il est par ailleurs possible de créer des fichiers et de sauvegarder des données. A titre d'exemple, prenons l'adaptation au QL des deux derniers tests de rapidité (1) de *LIST*.

Text n° 9 : écriture. On relie le micro-drive n° 1 au canal 7.

```
10 a$ = "LISTEST"
20 OPEN-NEW # 7, mdv1-test
30 FOR i = 1 TO 10000
40 PRINT # 7, a$
50 NEXT i
60 CLOSE # 7
```

Test n° 10 : lecture.

```
10 OPEN-IN # 8, mdv1-test
20 FOR i = 1 TO 10000
30 INPUT # 8, a$
40 NEXT i
50 CLOSE # 8
```

Ces tests donnent d'assez bons résultats : 135 sec. en écriture, et 99 sec. en lecture. Il semble que le système exploite intelligemment la mémoire vive, en évitant les accès trop fréquents à la mémoire de masse. Si nous n'avons, d'autre part, constaté aucun problème de fiabilité durant l'essai, nous n'avons pas infligé aux lecteurs/enregistreurs de cartouches un véritable test d'endurance — qui reste à faire. Enfin, on retiendra qu'une capacité de 100 Ko par unité n'est pas énorme, surtout pour un usage professionnel.

Le QL peut émettre des sons par l'intermédiaire d'un haut-parleur intégré, mais ils sont d'une piètre qualité. En outre un volume faible limite les possibilités de la machine dans le domaine musical. D'ailleurs, si l'instruction BEEP est perfectionnée (elle reçoit jusqu'à huit paramètres), elle est assez difficile à maîtriser, et l'on ne peut contrôler qu'une voie.

Le processeur du QL est souvent présenté comme un 32 bits. Il ne faut rien exagérer. En fait, seuls les registres internes possèdent 32 bits, les opérations ne s'effectuent que sur 16 bits, et les accès-mémoire sur 8 bits. Le QL offre pourtant certaines possibilités jusque-là inconnues sur du matériel grand public : si PEEK, POKE et CALL sont familiers, PEEK-W, POKE-W, PEEK-L et POKE-L permettent la lecture et l'écriture en mémoire sur 16 et 32 bits.

Les instructions EXEC, EXEC-W et SEXEC permettent, en principe du moins, le chargement et l'exécution en parallèle de plusieurs programmes. En ce qui me concerne, malgré de nombreux essais, je ne suis pas parvenu à les mettre en œuvre. Dommage, car le fonctionnement en multitâche d'une machine aussi bon marché serait une innovation bienvenue.

Le QL est doté d'une horloge interne. Cette constatation laconique pourrait être suivie d'une ribambelle d'instructions. Je résumerai en disant que toutes les fonctions généralement associées à ce type de gadget (ce terme n'a rien de péjoratif) sont présentes. Il est même possible d'obtenir le jour de la semaine (par l'intermédiaire d'un chiffre).

Avant de conclure, notons que le Basic du QL, impressionnant par ailleurs, comporte quelques lacunes. Nous avons déjà évoqué l'absence de double précision. Il faut y ajouter l'impossibilité d'exécuter un programme pas à pas (office généralement rempli par les fameux TRON et TROFF). Nous n'avons pas, non plus, réussi à mettre en évidence un jeu d'instructions permettant le traitement des erreurs. L'absence de possibilités de formatage (style PRINT USING) est moins gênante : une petite procédure bien sentie viendra facilement y remédier.

Le Sinclair QL était annoncé à l'origine comme un ordinateur à vocation professionnelle. De ce point de vue, il ne tient pas toutes ses promesses. C'est cependant une machine remarquable pour les passionnés de la programmation. Malgré quelques imperfections, le terme de « SuperBasic » n'est pas usurpé. L'inspiration très Pascalienne de ce Basic le rend moins concis qu'un Basic courant, mais cela présente plutôt un avantage dans la mesure où les programmes sont plus faciles à relire et à mettre au point. Une utilisation intelligente de certaines structures (en particulier des procédures, fonctions, SELect et IF...THEN...ELSE...ENDIF) permettront de passer un cap dans la complexité des problèmes. Revers de la médaille, ce langage s'écarte sur beaucoup de points du standard de fait qu'est le Basic Microsoft. L'adaptation, par l'amateur, de programmes créés sur d'autres machines sera loin d'être aisée.

(1) Pour les résultats des autres tests, voir ici-même, page 83.

Thierry LÉVY-ABÉGNOLI

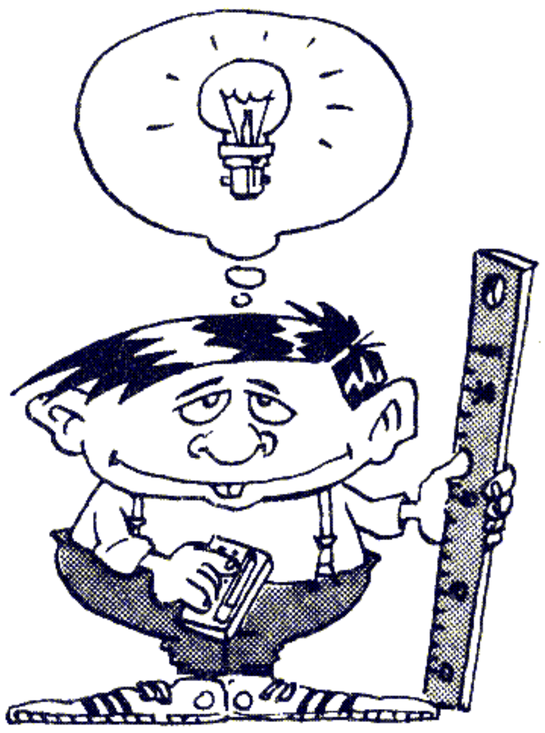
TRIANGLE DE PASCAL

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

Que de plans n'a-t-on pas tirés sur ce fameux tableau de chiffres nommé *Triangle de Pascal*. S'il existait un concours de beauté des tableaux, celui-là en serait le Phénix à n'en point douter.

Sa construction est élémentaire (voir ci-dessous) : à partir du sommet 1, on a posé que chaque nombre d'une nouvelle ligne est égal à la somme de celui qui est situé juste au-dessus de lui et de celui qui se trouve à gauche de ce dernier. L'absence de nombre vaut 0.

Immédiatement, des symétries apparaissent. La première colonne vaut toujours 1 et la seconde est égale au numéro n de la ligne concernée. Ces deux remarques sont aussi valables, respectivement, pour la diagonale principale et celle qui lui est immédiatement inférieure : le tableau est absolument symétrique.



N° de ligne	Tableau de Pascal						
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1
	...						

Simplifier, c'est condenser

En fait, chaque ligne n du tableau correspond aux coefficients du développement de l'expression $(a+b)^n$. Ainsi, la seconde ligne correspond à : N = 2, soit $(a+b)^2 = a^2 + 2ab + b^2$; les coefficients sont 1, 2, 1.

De même, pour la cinquième ligne, on a : N = 5, soit $(a+b)^5 = a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5$; et les coefficients sont 1, 5, 10, 10, 5, 1.

On simplifiera ce calcul de coefficients en condensant l'expression du

Ma solution

Le programme stocke pour un nombre n donné — entier, positif ou nul — tous les coefficients de $(a+b)^n$: les éléments du triangle de Pascal.

Mon programme occupe 24 octets sur 17 pas seulement. Seule la pile opérationnelle est employée dans le calcul. Avec le •END• final, un SIZE 316 peut être réalisé : la limite de n est donc 315 (pour une HP-41 CV ou CX).

Pour cette 315e ligne du tableau, l'élément central (R157 et R158) vaut $2.993653731 \times 10^{93}$ et le temps de calcul total est inférieur à 65 secondes. Sachant la symétrie des coefficients de part et d'autre de l'élément central, le programme range la valeur calculée à la fois dans le registre p et dans (n-p) ce qui divise par 2 le nombre de boucles et évite les erreurs d'arrondi en fin de liste.

L'initialisation de départ est minimale : pour bien faire, il conviendrait d'ajouter un CLRG effaçant les mémoires et ABS et INT pour se prémunir des entrées vicieuses de n négatifs ou non entiers. Mais alors on limiterait n à 314 du fait de l'augmentation de la taille du programme.

L'algorithme employé ne calcule pas directement C_n^p à l'aide de n!/p!(n-p)! car sinon on serait limité à n = 69. En effet 70! provoque un OUT OF RANGE (erreur de dépassement de capacité). La ligne 10 n'est jamais exécutée. En fin de programme, les coefficients sont dans les registres 00 à n. Si la limite matérielle est n = 315, la limite absolue — car OUT OF RANGE — de l'algorithme est n = 336.

Jean THIBERGE

Triangle de Pascal
Programme pour HP-41C
Auteur Jean Thiberge


```
01•LBL "THI
"
02 SIGN
03 LASTX
04 0
05•LBL 01
06 RDN
07 X<>Y
08 STO IND
Y
09 STO IND
T
10 ISG T
11 X<>Y
12 X<>Y
13 ST* Y
14 R↑
15 DSE Y
16 ST/ Z
17 X<=Y?
18 GTO 01
19 END
```

QUI DIT MIEUX ?

COSINUS, sinus, tangente et fonctions inverses, c'est bien. Modes degrés, radians ou grades, c'est pas mal non plus. Mais les fonctions hyperboliques, c'est mieux !

D'autant plus que la HP-41 C ne les possède pas et que cela nous offre un prétexte à une optimisation effrénée. Le tableau ci-dessous exprime ces fonctions hyperboliques à l'aide de fonctions de la HP-41 C ou d'autres fonctions hyperboliques : certaines, donc, devront pouvoir être appelées comme sous-programme de calcul...

Nom	Équation
Cosinus hyp.	$\cosh(x) = (e^x + e^{-x})/2$
Sinus hyp.	$\sinh(x) = (e^x - e^{-x})/2$
Tangente hyp.	$\tanh(x) = \sin(gd(x))$
Argument cos.	$\cosh^{-1}(x) = \text{sech}^{-1}(1/x)$
Argument sin.	$\sinh^{-1}(x) = gd^{-1}(\text{arctg}(x))$
Argument tan.	$\tanh^{-1}(x) = gd^{-1}(\arcsin(x))$
Amplitude hyp.	$gd(x) = 2\text{arctg}(e^x) - \pi/2$
Sécante hyp.	$\text{sech}(x) = 1/(\cosh(x))$
Amplitude inv.	$gd^{-1}(x) = \text{LN}(\tan(\pi/4 + x/2))$
Sécante inv.	$\text{sech}^{-1}(x) = gd^{-1}(\arccos(x))$

où arcsin = $\sin^{-1}$, arccos = $\cos^{-1}$ et arctg = $\tan^{-1}$

Mon programme, classique, occupe 90 pas mais aucun registre de mémoire. J'ai nommé chaque sous-programme (dans l'ordre du tableau) CH, SH, TH, ACH, ASH, ATH, GD, SE, AGD et ASE. L'argument x est dans X avant l'appel et en Last x au retour du calcul. Le résultat se trouve évidemment en X.

Alain GOUBAULT de BRUGIÈRE

développement de $(a+b)^n$ en fonction de n :

$$(a+b)^n = \sum_{p=0}^{p=n} C_n^p * a^p b^{n-p}$$

où $C_n^p = n!/(p!(n-p)!)$

Le coefficient numérique est donné par C_n^p et, pour notre tableau, n correspond au numéro de la ligne et p à celui de la colonne. Ainsi, le tout premier 1 correspond bien à C_0^0 car 0! vaut 1. De même, pour toute la première colonne, p = 0 et donc $C_n^0 = n!/n!$

Le défi lancé par Monti dans *LIST* n° 5 était donc : *sachant le numéro n d'une ligne quelconque du tableau de Pascal, trouver la suite des chiffres qui la compose. Par exemple, n = 5, trouver 1, 5, 10, 10, 5. La solution réside, bien sûr, dans l'examen attentif du développement de $(a+b)^n$. Par commodité, on stocke dans les mémoires R00 à Rnn les nn + 1 chiffres de la ligne à trouver. Faites vite et court.*

Et c'est à Jean Thiberge — grand connaisseur de la HP-41 C — que nous devons la version gagnante de ce programme de calcul des coefficients du tableau de Pascal. Il en décortique lui-même les rouages.

Jean-Christophe KRUST

UN RÉPERTOIRE DES RÉPERTOIRES

SANS doute vous est-il arrivé, à vous aussi, de rechercher un programme ou un fichier soigneusement mis de côté sur une disquette. La question est de savoir sur quelle disquette. Alors on les passe toutes en revue. Quand il est si simple de demander à l'ordinateur où se trouve quoi... Faisons le ménage dans nos disquettes.

Le programme DIR/BAS est un utilitaire de classement. Il délivre sur imprimante des listes triées des fichiers enregistrés sur une, deux, dix, cent disquettes. Le programme inscrit aussi, en regard du nom de chaque fichier, les noms (également triés) de toutes les disquettes contenant ce fichier. Si le fichier TOTO/BAS a été sauvé sur deux de vos disquettes intitulées respectivement JEUX et DIVERS, l'utilitaire de classement vous l'indiquera par écrit.

Le nom d'une disquette est tout simplement celui qu'aux fins d'identification, vous avez écrit avec un crayon-feutre sur son étiquette.

L'imprimante est indispensable

Après avoir entré ce nom et le numéro du lecteur de disque utilisé, le répertoire (DIR simple) est affiché sur l'écran où il est lu par le programme qui range dans un tableau F$ (500) les noms des fichiers suivis du nom de la disquette.

Dès que vous répondez « N » à la demande de lecture d'une autre disquette, le tableau est trié et l'on passe aux éditions sur imprimante. Vous avez alors le choix entre une liste complète (il faut alors répondre "*" à la question "LETTRE ?") ou une liste sélective des

seuls fichiers dont le nom commence par la lettre indiquée.

Ce petit programme, bien utile pour ne pas laisser la pagaille s'installer parmi nos disquettes, n'est malheureusement pas accepté par tous les Systèmes d'Exploitation des Disquettes (SED, ou DOS en anglais). Il tourne sans problème sous NEWDOS 80 Version 2 (modèle 1 ou 3) qui reconnaît la commande basique CMD "DIR, 1" pour afficher à l'écran le répertoire de la disquette se trouvant dans l'unité 1.

Il sera également accepté sous TRSDOS 1.3 (modèle 3) ou son frère cadet, le TRSDOS 2.7 DD (modèle 1, double densité) après que les lignes 40 et 50 aient été modifiées comme suit :

40 DB = 15
50 DR$ = "D :"

La variable DB contient la position du premier caractère utile du DIR à l'affichage, et DR$ est l'orthographe de la commande affichant le répertoire.

Le CMD "O", J, F$ (1) de la ligne 200 assure le tri rapide des J premiers éléments de F$ () à partir de F$ (1). Cette dernière instruction est acceptée par tous les SED déjà cités. Le bon vieux TRSDOS 2.3 n'accepte, quant à lui, aucune de ces instructions... Pour les autres DOS, veuillez consulter la notice.

Roger BROUSMICHE

Un répertoire des répertoires
Programme pour TRS-80 I et III
Auteur Roger Brousmiche


```
1 '-----------------------------------------------------+
2 '  DIR/BAS - (C) 1985, LIST et Roger BROUSMICHE       !
3 '  Configuration :  (imprimante obligatoire)          !
4 '  Modele 3, disque , TRSDOS 1.3 ou NEWDOS80          !
5 '  Modele 1, disque , TRSDOS 2.7DD ou NEWDOS80        !
6 '-----------------------------------------------------+
10 '----------initialisations
20 CLS : CLEAR 10000 : DEFINT A-Z
30 K$= STRING$ (16,32) : DIM F$(500)
40 DB=128      ' DB=15      ! sous TRSDOS 1.3
50 DR$="DIR," ' DR$="D:"   ! ou TRSDOS 2.7DD
60 '----------lecture des repertoires
70 INPUT "
LECTURE DU REPERTOIRE D'UNE DISQUETTE (O/N) ";R$
80 IF LEFT$(R$,1)()"O" THEN 190
90 INPUT "SE TROUVANT SUR L'UNITE DISQUE # ";UD$
100 LINE INPUT "VEUILLEZ ENTRER SON NOM : ";ND$
110 CMD DR$+UD$ : PRINT STRING$ (63,131)
120 FOR I=15360+DB TO 16382 : P=PEEK(I)
130 IF P)127 THEN I=16382 : GOTO 170
140 IF P=32 THEN 170 :ELSE L=0
150 IF P()32 THEN L=L+1 : MID$(K$,L,1)=CHR$(P) : I=I+1 : P=PEEK(
I) : GOTO 150
160 J=J+1 : F$(J)=LEFT$(K$,L)+ STRING$ (13-L,32)+ND$
170 NEXT I : GOTO 70
180 '----------tri
190 PRINT : IF J(1 THEN END
200 CMD "O",J,F$(1)
210 '----------edition imprimante
220 PRINT "EDITION DES FICHIERS SUR L'IMPRIMANTE :
ENTREZ '*' POUR LE LISTING COMPLET,
OU UNE LETTRE DE 'A' A 'Z' POUR UN LISTING SELECTIF,
OU (BREAK) POUR QUITTER LE PROGRAMME."
230 INPUT "LETTRE ";LT$ : LT$=LEFT$(LT$,1)
240 FOR I=1 TO J : IF LT$="*" THEN 270
250 IF LEFT$(F$(I),1)(LT$ THEN 300
260 IF LEFT$(F$(I),1))LT$ THEN I=J : GOTO 300
270 LPRINT F$(I);
280 IF LEFT$(F$(I),12)=LEFT$(F$(I+1),12) THEN I=I+1 : LPRINT MID
$(F$(I),13); : GOTO 280
290 LPRINT
300 NEXT I : GOTO 230
```

TROIS MINIATURES

TOUT programmeur a une manière bien à lui, un coup de patte. La recherche d'une extrême concision est le trait caractéristique d'un style très difficile à pratiquer. Pour beaucoup de gens, c'est même le grand art.

Messieurs les Jurés,

Le gang des Jivaros, auquel l'accusé a donné son nom, est responsable de multiples méfaits tellement abominables que le Parquet a failli demander le huis clos pour juger cette affaire.

Les pièces à conviction que vous avez aujourd'hui sous les yeux concernent deux programmes classiques qui s'étalaient hier encore innocemment sur 70 à 80 lignes chacun.

Leurs restes décharnés, meurtris jusqu'au cœur des algorithmes, sont devant vous, méconnaissables, réduits à rien : 3 lignes pour l'un et 2 pour l'autre. Et même pas structurées, quelle déchéance ! Mais le plus atroce est que ces malheureux mutilés sont encore vivants, ils « tournent même sans problème », comme l'avouent dans leur jargon ces bourreaux du Basic.

— Que dites-vous, Monsieur le Président ? Il y a une troisième pièce à conviction ? Un programme piqué dans *LIST* 5 page 72, et ramené sournoisement de 27 lignes à combien ? Deux seulement ?

— Plus rien n'est donc sacré pour cet énergumène !

— Votre verdict, Messieurs les Jurés, sera sans pitié comme sans illusion. Même condamné au maximum, cet incorrigible récidiviste trouvera toujours un truc pour réduire sa propre peine à trois fois rien.

(...)

— Superbe, le réquisitoire du procureur. Comment diable as-tu fait pour t'en sortir ?

— Oh, tu sais, la force de l'habitude : je me suis fait tout petit.

Dr JIVARO

Ces programmes (ou plutôt ce qu'il en reste), sont écrits pour CanonX-07 et assez facilement transposables sur d'autres matériels.

Les deux premiers peuvent être rapprochés de façon instructive des programmes de Loto et d'Interpolation du manuel de *Jeux et programmes pour Canon X-07*, et le troisième de celui que *LIST* vient de publier sous la même dénomination : « donnée manquante ».

Le programme de Loto tire des boules jusqu'à ce qu'on appuie sur une touche, ou jusqu'à épuisement des 49 boules.

A noter : la manière simple et efficace d'interdire toute répétition d'une même boule, et la réinitialisation aléatoire automatique de la séquence RND.

Avec le deuxième programme, le X-07 demande le nombre de points connus, puis leurs coordonnées. Il calcule ensuite les ordonnées correspondant aux abscisses de votre choix par la méthode du polynôme d'interpolation de Lagrange.

A noter : l'utilisation d'une double boucle (I,J) exige l'élimination des cas I = J, qui comporteraient une division par 0.

Enfin, avec *Donnée manquante*, on tape directement RETURN si le programme demande la valeur de l'inconnue. Les solutions de ce genre paraissent évidentes... dès qu'on les a sous les yeux.

A noter : en changeant les noms de grandeurs et les formules, ce programme peut s'adapter à tout ensemble de trois grandeurs dont chacune peut être calculée à partir des deux autres. Toutefois les grandeurs connues ne peuvent être nulles.

Pierre BARNOUIN

Trois concentrés de programmes
Adaptation pour X-07
Auteur Pierre Barnouin

Loto

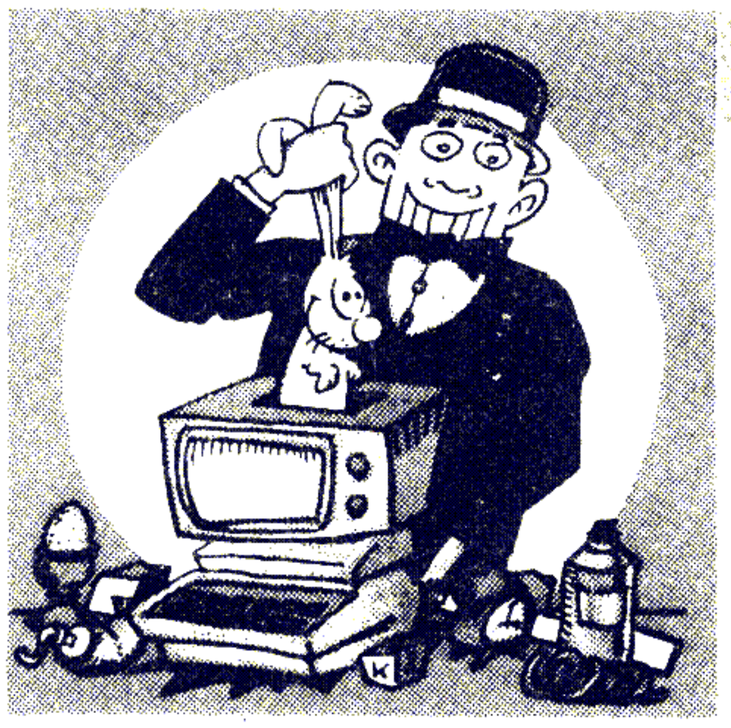
```
1 DIMA(49):FORI=1TO49:A(I)=I:NEXT
2 J=J+1:K=J+(50-J)*RND(J-1):PRINTA(K);:A
(K)=A(J):IFINKEY$=""THEN2
```

Interpolation

```
1 INPUT"Nb pts connus";N:DIMX(N),Y(N):FO
RI=1TON:PRINTI;"x,y":INPUTX(I),Y(I):NEXT
2 INPUT"Pour x=";X:FORI=1TON:Z=Y(I):FORJ
=1TON:IFI-JTHENZ=Z*(X-X(J))/(X(I)-X(J))
3 NEXT:Y=Y+Z:NEXT:PRINT"y=";Y:Y=0:GOTO2
```

Donnée manquante

```
1 INPUT"DISTANCE";D:INPUT"HEURES";T:PRIN
T"VITESSE";:IFD*TTHENPRINTD/T:END
2 INPUTV:IFDTHENPRINT"HEURES";D/V:ELSEPR
INT"DISTANCE";V*T
```



LA BOÎTE A MALICES...

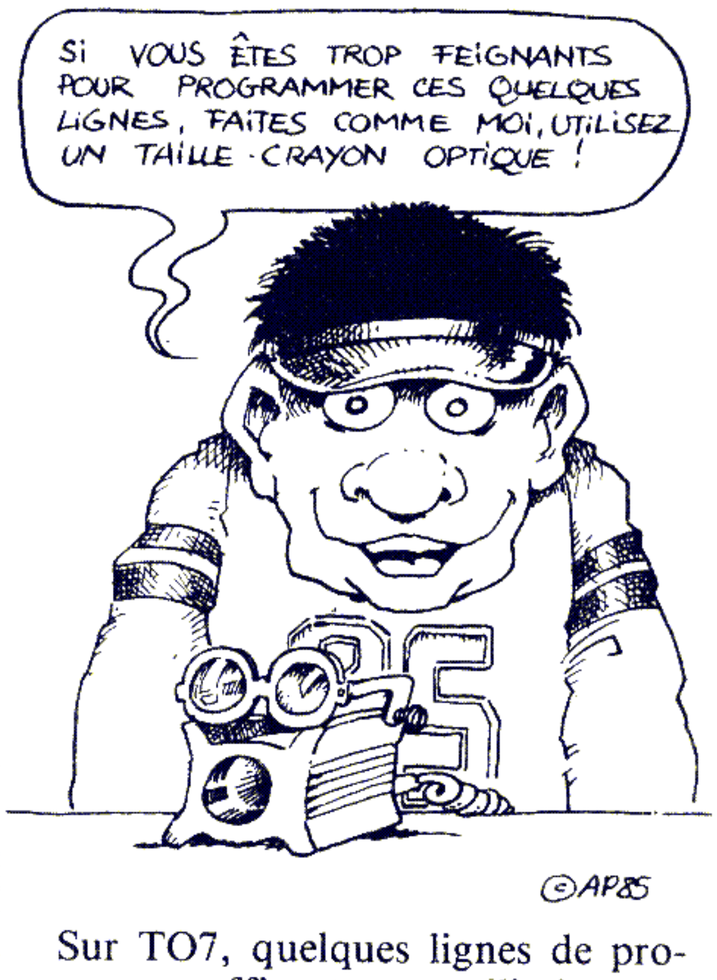
PRENEZ un programme et ôtez-en très soigneusement toutes les astuces, des plus élémentaires aux plus subtiles. Vous êtes certain de n'en avoir laissé passer aucune ? Bien. Que reste-t-il ? Rien, ou peut-être une bogue ou deux (tout le monde peut se tromper). En fait, tout programme n'est qu'une suite 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

AFFÛTEZ VOTRE CRAYON OPTIQUE

Quand on se sert du crayon optique pour désigner une zone de l'écran (dans le cas d'un menu ou d'un signe quelconque à sélectionner), il arrive fréquemment, si la zone est petite ou étroite, que l'indication transmise par le crayon tombe à côté de la zone recherchée.

Cette imprécision est due pour une part au système même, d'autre part au fait que, pour obtenir un fonctionnement correct, on doit tenir le crayon toujours dans le même sens et surtout bien perpendiculairement à l'écran, ce qui n'est pas toujours commode.



Sur TO7, quelques lignes de programme suffisent pour éliminer ce phénomène gênant. Dès que l'on approche le crayon de l'écran, le programme fait apparaître un curseur clignotant indiquant en permanence l'endroit visé par le crayon. Il suffit donc de diriger le curseur sur l'endroit désiré, puis d'appuyer sur le crayon pour actionner le petit interrupteur placé à son extrémité. Après avoir écrit ce sous-programme, en 63000 par exemple, remplacez donc vos « INPUTPEN C,L » par un GOSUB 63000 et vous ne rencontrerez pratiquement plus jamais d'erreur de visée.

Pour changer la couleur du curseur,

on modifiera le POKE de la ligne 63050 en remplaçant le 09 par le chiffre qui convient.

Si l'ordinateur n'est pas un TO7 première manière, mais un TO7-70, on peut améliorer le programme. En effet, la précision du crayon optique est supérieure sur ce type d'appareil : elle permet la détection de chaque point alors que le TO7 en est resté à l'octet.

Pour affûter le crayon optique
Programmes pour TO7 et TO7-70
Auteur Paul Gardan
Copyright *LIST* et l'auteur

Version pour TO7

```
63000 'CURSEUR POUR CRAYON
      OPTIQUE TO7
63001 'AUTEUR Paul GARDAN
63005 INPENC, L
63010 IF C < 0 OR L < 0 THEN 63000
63020 ZZ = 16384 + C/8 + L*40
63030 POKE &HE7C3, PEEK
      (&HE7C3) AND &HFE
63040 ZZ0 = PEEK (ZZ)
63050 POKE ZZ,09
63060 FOR W = 1 TO 60:NEXT
63080 POKE ZZ,ZZ0
63100 IF PTRIG = 0 THEN 63000
      ELSE RETURN
```

Version pour TO7-70

```
63200 'CURSEUR POUR CRAYON
      OPTIQUE TO7-70
63210 'AUTEUR Paul GARDAN
63220 INPEN C1, L1
63230 IF C1 < 0 OR L1 < 0 THEN
      63220
63240 ZZ = 16384 + C1/8 + L1*40
63250 POKE &HE7C3, PEEK
      (&HE7C3) AND &HFE
63260 ZZ0 = PEEK (ZZ)
63270 POKE &HE7C3, PEEK
      (&HE7C3) OR &H01
63280 ZZ1 = PEEK (ZZ)
63290 PSET (C1, L1), 5
63300 FOR W = 1 TO 20:NEXT
63310 POKE &HE7C3, PEEK
      (&HE7C3) OR &H01
63320 POKE ZZ,ZZ1
63330 POKE &HE7C3, PEEK
      (&HE7C3) AND &HFE
63340 POKE ZZ,ZZ0
63350 IF PTRIG = 0 THEN 63220
      ELSE RETURN
```

Sur TO7-70, le curseur obtenu est en magenta. Si l'on préfère une autre teinte, il faut modifier le PSET de la ligne 63290.

Le programme pour TO7-70 fait donc apparaître un point clignotant (à la place du curseur de huit points sur TO7) et il autorise de ce fait un pointage plus rigoureux.

Paul GARDAN

PETITES ÉCONOMIES

A RUBAN, RUBAN ET DEMI...

Si vous utilisez une imprimante à aiguilles et si vous aimez que vos documents soient bien lisibles, écrits noir sur blanc, alors vous n'avez pas manqué de constater que le coût des rubans d'impression n'est pas du tout négligeable.

La prolongation du temps de service de ces rubans, qu'ils soient conditionnés en bobines ou en cartouches, pourrait épargner les tirelires. Quelques astuces simples peuvent vous faire faire bien des économies.

Première possibilité : le ruban n'est souvent utilisé par la tête d'impression que sur une partie de sa largeur, et rarement sur la zone centrale. La piste encrée finit donc par s'épuiser sur la zone utilisée tout en restant quasiment neuve sur la zone inemployée par la tête. Dans ce cas, un simple retournement du ruban doublera sa durée de vie. Si le ruban est en bobine, cette opération ne posera aucun problème. S'il est en cartouche, il suffira pour le retourner de le plier et de lui faire parcourir manuellement dans son boîtier un tour complet jusqu'à réapparition du pli d'inversion.

Autre possibilité : après avoir été retourné, le ruban qui vient ainsi de doubler sa période de service est-il bon pour la réforme ? Sans doute pas, si toutefois vous acceptez de perdre la garantie de votre matériel et de prendre quelques risques pour votre tête d'impression (les encres sont tout de même des produits chimiques). Faites l'achat chez votre papetier favori d'un flacon d'encre à tampon de la couleur requise. Déroulez votre ruban, humectez-le de quelques gouttes de cette encre (attention : ça tache !) puis laissez-le s'en imprégner.

Quand l'opération sera terminée, vous aurez en main un ruban neuf, prêt à de nombreuses semaines de bons et loyaux services. Si le ruban est en cartouche, on devra ouvrir le boîtier avec précaution pour humecter le ruban. Si la cartouche est scellée, on sortira le ruban en tirant sur sa partie visible et l'on procédera comme avec une bobine.

Si votre matériel s'y prête, n'oubliez pas de réutiliser le truc n° 1 (renversement du ruban) lorsque le vieillissement se fera à nouveau sentir. Résultat : durée de vie d'un ruban quadruplée.

Enfin, une autre opération (salissante elle aussi, mais rentable) consiste à utiliser un *dispersant* (tel celui employé dans les photocopieuses à encre liquide) pour réhumecter le ruban. Veillez bien dans ce cas à ne pas trop le mouiller, et vérifiez la compatibilité entre le ruban et le dispersant dont vous disposez.

Cette dernière solution est légèrement moins bonne que la précédente pour diverses raisons. D'abord, elle consiste seulement à répartir à nouveau sur la surface du ruban des pigments colorés qui s'y trouvent encore, sans en ajouter. Ensuite, selon la nature du dispersant utilisé, il est possible que le ruban prenne un aspect gras, avec risque d'encrassement de la tête d'impression. Bien que l'efficacité de cette méthode né soit pas garantie, la durée de vie du ruban a de bonnes chances de s'en trouver augmentée.

Et puis, on fait avec ce que l'on a, n'est-ce pas ?

Robin BOIS

FAUSSE MÉMOIRE D'ÉCRAN

Il était une fois de petits ordinateurs pas chers du tout, simples à utiliser, et qui n'avaient pas beaucoup de mémoire. Et les petits informaticiens amateurs qui voulaient s'en servir pour faire de grands et beaux programmes avaient recours à toutes sortes d'astuces.

En particulier, ils avaient constaté qu'il était possible de se servir de l'écran comme d'une mémoire supplémentaire, puisque celui-ci retenait effectivement les informations qu'il affichait. Et ils bâtirent des superprogrammes, l'un jouant aux dames, l'autre pratiquant les envahisseurs, un autre encore traitant ses textes, à grands coups de PEEK et de POKE dans la partie de la mémoire vive consacrée à l'écran. Puis les informaticiens en herbe ont grandi, les mémoires et les capacités de leurs appareils aussi d'ailleurs, et les gentils fabricants, pour ne pas les priver de leurs chères bonnes vieilles habitudes, leur ont offert des outils commodes pour examiner directement le contenu d'une case, ou l'état d'un point de l'écran. Et chacun de proposer son mot : POINT, SCREEN, SCREEN$, etc. Tout le monde était heureux et faisait de beaux programmes, avec de jolis écrans examinés sous toutes les coutures.

Et puis, et puis... Les temps ont changé. Les prix ont encore baissé, les ingénieurs sont devenus plus malins, et ils n'ont pas voulu que les pauvres petits programmeurs puissent confondre leur mémoire de programme et leur mémoire d'écran, au risque de spectaculaires téléscopages. Ils ont créé des circuits spécialisés, chargés de gérer l'écran à leur façon, avec leur mémoire personnelle. Plus question de PEEK et de POKE dans cette mémoire. Certes, quelques-uns ont gentiment prévu des VPOKE et VPEEK, mais la plupart ont verrouillé l'accès : non, on ne regarde pas, c'est seulement pour faire voir, pas pour voir. Et, non content de cela, par quelque mesquinerie incompréhensible, ils ont même été jusqu'à supprimer le SCREEN de la liste des instructions. D'accord, il est encore possible de savoir quelle est la couleur d'un pixel. Mais plus de connaître la lettre qui est affichée en ligne 12, colonne 15.

Et voilà certains des nouveaux programmeurs en herbe bien malheureux. Oh certes, quand on conçoit le programme soi-même, on peut toujours s'en tirer, par une astuce ou une autre. Mais si, par exemple, on veut adapter un programme qui ne se lasse pas d'utiliser des SCREEN pour examiner les caractères présents sur l'écran, on est perdu.

Une astuce existe. Elle est simple à utiliser, même si elle manque un peu d'élégance. Il s'agit de jouer sur le fait que les appareils actuels ont en général à leur disposition des mémoires suffisamment impressionnantes pour en détourner une toute petite partie qui simulera la mémoire d'écran. Les textes affichés sur l'écran seront introduits lettre par lettre dans cette mémoire, en respectant les contraintes imposées par le programme initial. Si le programme de départ utilise une instruction SCROLL (déroulement de l'écran), une adaptation sera nécessaire : par exemple, modifier l'adresse de départ choisie pour la fausse mémoire d'écran.

Prenons, par exemple, un appareil qui dispose pour l'affichage de 25 lignes de 40 caractères, soit 1000 positions mémoire, avec des adresses allant de 32000 à 32999 pour la mémoire d'écran. On choisira donc un emplacement mémoire disponible de 1000 octets, le même si c'est possible, ou pour prendre un exemple, de 42000 à 42999.

Toutes les instructions de type LOCATE 2,5 : PRINT " xxx " seront doublées par un renvoi dans un sous-programme gérant cette partie de la mémoire. Trois paramètres devront être passés à ce sous-programme : LI pour le numéro de la ligne, CO pour la colonne, et TE$ pour le texte à afficher concrètement, supposons une ligne

```
50 LOCATE 12,15 : PRINT " BON JOUR "
```

On la remplacera par

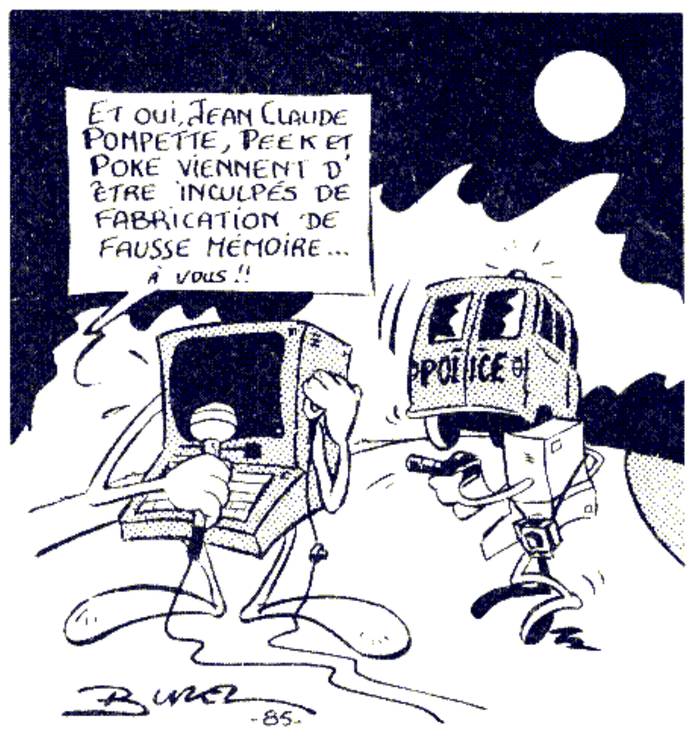
```
50 LOCATE 12,15 : PRINT " BON JOUR " : LI = 12 : CO = 15 : TE$ = " BONJOUR " : GOSUB 9000
```

(si le sous-programme est implanté en 9000).

Bien entendu, LOCATE n'étant pas présente dans tous les Basic, elle sera éventuellement remplacée par l'instruction disponible qui lui correspond.

Le sous-programme en 9000 pourra se présenter de la façon suivante :

```
9000 REM INSCRIPTION EN MEM D'ECRAN
9010 FOR II = 1 TO LEN(TE$)
9020 POKE 42000 + 40 × LI + CO + II - 1, ASC(MID$(TE$,II,1))
9030 NEXT II
9040 RETURN
```



L'intérêt du procédé est de permettre une adaptation sans trop de modifications par rapport au programme initial. Supposons maintenant que le programme initial lise le code ASCII du caractère présent en ligne 12 et en colonne 18 :

```
120 C = SCREEN (12,18)
```

La ligne 120 sera tout simplement remplacée par un appel du sous-programme de lecture, avec passage des paramètres ligne et colonne :

```
120 LI = 12 : CO = 18 : GOSUB 9050
```

et le sous-programme de lecture, débutant en 9050, pourrait être :

```
9050 REM LECTURE D'ECRAN
9060 C = PEEK(42000 + 40 × LI + CO)
9070 RETURN
```

La variable C est restituée en retour de sous-programme.

Un SCROLL sera éventuellement simulé en définissant une variable de début de mémoire d'écran, ME = 42000, et en ajoutant derrière l'instruction SCROLL une instruction du type : ME = ME + 40.

Toutes ces indications devraient faciliter l'adaptation de certains programmes sur des appareils récents.

Jacques DECONCHAT

APPLICATION SYNTHÉTIQUE

Réaliser un programme plus court et, a priori, plus inutile que celui-ci est rigoureusement impossible. Pourtant, si la manipulation proposée n'a guère de sens pratique, en revanche, les questions qu'elle pose quant à la santé mentale de notre bonne vieille HP-41 ne sont pas sans intérêt.

La seule instruction synthétique du programme est STO d. On la créera facilement, directement (codes 145 126) ou grâce au clavier synthétique (*LIST* n°6) dont chacun doit aujourd'hui goûter les avantages.

Le programme, donc, si trois instructions peuvent être ainsi nommées, est reproduit ci-contre. La constante PI est simplement stockée dans le registre d'état « d ». Ce dernier contenant, bit par bit, l'état de chacun des 56 flags (ou drapeaux) de la HP-41. Ainsi, chaque indicateur binaire voit-il son état modifié en fonction de la valeur PI introduite avec STO d.

A l'exécution du programme PI, rien ne semble se passer d'extraordinaire, seul l'indicateur USER s'allume et le format d'affichage change pour passer en ENG 4.

Mais, si l'on presse la touche R/S, se produit à l'affichage un défilé pour le moins surprenant de caractères spéciaux, de fonctions synthétiques ou classiques. Si l'imprimante est connectée, on obtient une liste de ces petits monstres alphabétiques (en minuscules non expansées) reproduite ici à la suite du programme.

Le programme

```
01✱LBL "PI"
02 PI
03 STO d
04 .END.
```

Le catalogue

```
Ø
cD
@@✱a&GB1⊢W>
?
ST
*
OD
−
ßαARCL
DLac×
]Ø
F
IEW
F
D
OPY
ß5CF
W
?
⊢
]Ø
LP
U
HCLRG
DLac×
α⊢]@>
S
W
@BST
MS
MPT
8
X<>
@@✱a&GB1⊢W>
AHHH
✱:Σ
≠αe2
DLac×
MPT
U
HCLRG
αH]@>
CHS
W
?
DLac×
LP
Ø
U
e@@N%
HCLRG
%X<=Ø?
HCLRG
W
R
Ø
Y?
Ø
W
cue0
&
b
H
W
−
ΣCLST
U
e@@N%
]Ø
LP
α
HCLRG
]Ø
−
b
CT
HCLRG
%X<=Ø?
MS
HCLRG
8
HSTO
CF
@@✱a&GB1⊢W>
J
;
DLac×
MPT
]Ø
MPT
TER↑
MPT
]Ø
a≤b
;
α
dα
eGØBEEP
52−$T+NuIAα
uΓ
⊢
X
Aα
≠αe2
```

En fait, il s'agit là d'un catalogue de fonctions : on peut le ralentir en pressant une touche quelconque, le stopper avec R/S, remonter ou descendre la liste avec BST et SST et, enfin, en sortir avec R/S suivi de ←.

Mieux, en interrompant dès le début le défilement du catalogue, on n'en pourra pas moins remonter avec BST avant même ce début et découvrir des « instructions » comme <=0 ? ou encore @ ¯a ⌶ GBIW>... Etonnant, non ?

Et si vous parveniez à trouver tout de même, sinon une application pratique, du moins une explication à ce phénomène, pensez à ceux, nombreux, aux prises avec les affres de la perplexité.

Adrien MEAUDRE

UTILITAIRE TI-57

UNE VRAIE MACHINE DE BASE

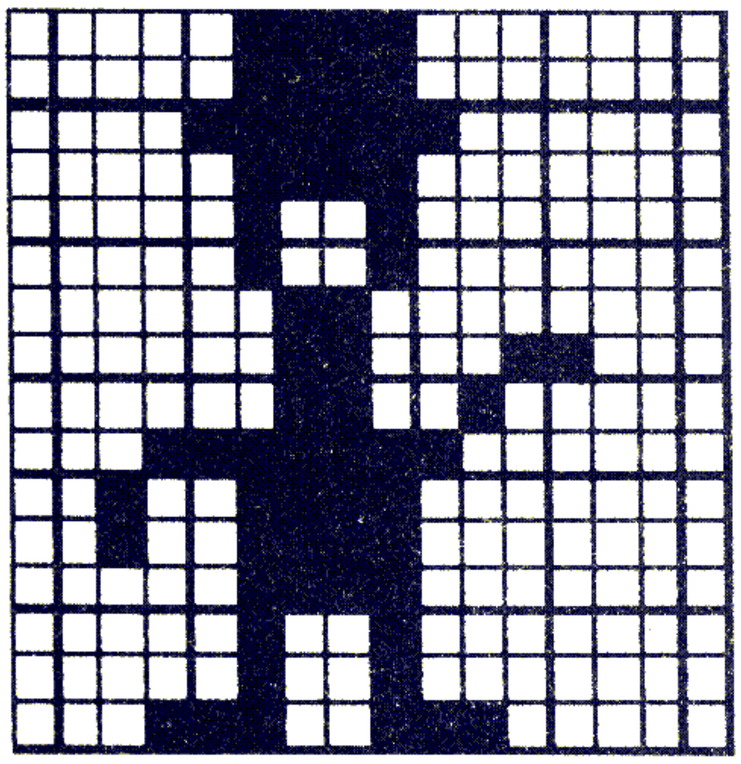
Si vous êtes en possession d'un véritable ordinateur depuis peu de temps, la tentation est sans doute forte de laisser tomber complètement la petite programmable de vos débuts. Résistez ! Non seulement elle peut vous être très utile pour faire des calculs auxiliaires pendant la mise au point de vos programmes, mais vous pourrez aussi lui confier une application vraiment indispensable pour les programmeurs : le changement de base.

Le programme est écrit dans une version volontairement très courte (37 pas, mais on pourrait encore optimiser). Il permettra de passer directement d'une base de départ **a** quelconque à une autre base d'arrivée **b** également quelconque, et ceci sans limitations sérieuses concernant la longueur des nombres introduits. L'adaptation pour la TI-57 version LCD sera très facile à réaliser (modifications minimes).

Pour utiliser le programme.

- entrer la base de départ en M1 ; exemple 16 STO 1
- entrer la base d'arrivée en M2 ; exemple 8 STO 2
- taper RST, puis entrer le premier chiffre du nombre donné en base **a**, en commençant par la droite ; presser sur R/S et introduire le chiffre suivant, etc.

Pour EF78A donné en base 16, on entrera 10 R/S 8 R/S 7 R/S 15 R/S 14 R/S, puis on obtiendra le résultat en base **b** en tapant SBR1. Les chiffres seront affichés en commençant par la droite, et séparés par une pause mais on peut très bien intercaler un R/S entre deux affichages, au pas 31. La fin de l'affichage est signalée par l'apparition de 0 00. Ici, EF78A (base 16) donne 3573606 en base 8.



D'une base à l'autre
Programme pour TI-57
Auteur Jacques Deconchat
Copyright *LIST* et l'auteur

Pas	Code	Instruction
00	19	2nd Ct
01	32 5	STO 5
02	51 2	GTO 2
03	86 3	2nd Lbl 3
04	55	×
05	33 1	RCL 1
06	35	y^x
07	33 7	RCL 7
08	85	=
09	49	2nd Int
10	34 5	SUM 5
11	86 2	2nd Lbl 2
12	01	1
13	34 7	SUM 7
14	15	CLR
15	81	R/S
16	51 3	GTO 3
17	86 1	2nd Lbl 1
18	19	2nd Ct
19	33 5	RCL 5
20	45	:
21	33 2	RCL 2
22	85	=
23	49	2nd Int
24	32 0	STO 0
25	55	×
26	33 2	RCL 2
27	85	=
28	−34 5	Inv SUM 5
29	33 5	RCL 5
30	36	2nd Pause
31	36	2nd Pause
32	33 0	RCL 0
33	32 5	STO 5
34	−66	2nd Inv x=t
35	51 1	GTO 1
36	42	EE

11110000000 780_{16}

111111000000 $FC0_{16}$

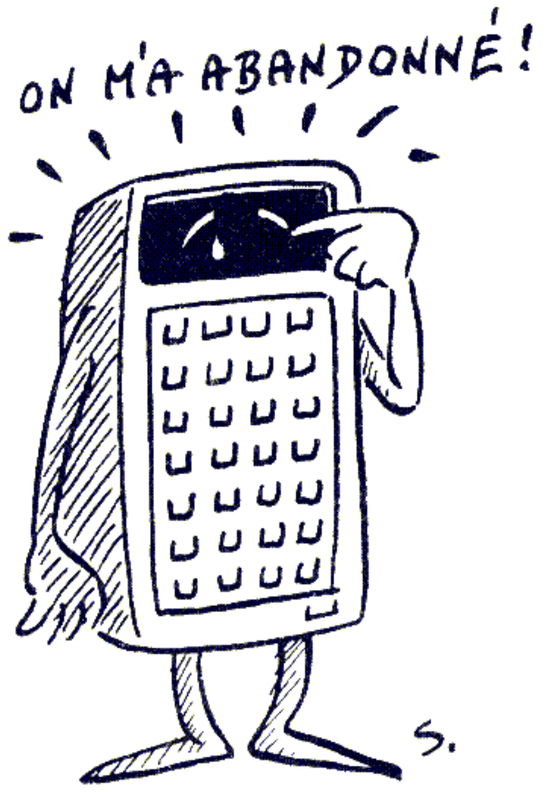
10010000000 480_{16}

etc.

Définition d'un lutin 16×16

Ce programme servira, entre autres, à définir de nouveaux "sprites" sur des pavés 8×8 ou 16×16. C'est un exemple type de passage de base 2 en base 16 avec des nombres trop longs pour l'affichage d'une calculatrice.

Jacques DECONCHAT



DAI

TEXTE EN 16 COULEURS

La page texte (MODE 0) du Dai permet d'écrire en deux jeux de deux couleurs. Habituellement, seul le premier jeu est utilisé. Il correspond aux deux premières valeurs de la commande Basic COLORT C1 C2 C3 C4, où C1 C2 C3 C4 codent les numéros de couleurs désirées. Pour se servir du second jeu (C3 C4), c'est déjà plus compliqué : il faut « POKEr » dans la mémoire d'écran les octets de couleur de chaque lettre, lesquels se situent trois octets plus bas que l'octet codant la lettre. Si l'octet de couleur contient 0, c'est le premier jeu de couleur qui est sélectionné ; s'il contient 255, c'est le second jeu qui entre en service. Pas simple, n'est-ce pas ? De là à dire que passer en 16 couleurs d'écriture devient « chinois », il n'y a qu'un pas... que nous ne franchirons pas. En

Texte en 16 couleurs
Programme pour Dai
Auteur Alain Mariatte


```
10      REM
20      REM
30      REM ************************************************
40      REM *** S/P 16 COULEURS CARACTERES.LIGNES      ****
50      REM *** ADRESSEES PAR LA FN CURSOR(X,Y)       ****
60      REM *** SANS CALCULS COMPLIQUES !             ****
70      REM ************************************************
80      REM
90      REM
100     MODE 0:PRINT CHR$(12):CF=0:COLORT CF 10 CF CF
110     PRINT :PRINT
120     INPUT "LIGNE TEXTE A ECRIRE:";A$:PRINT
130     INPUT "CURSOR (X,Y):";X,Y:PRINT :CURSOR X,Y:GOSUB 1000
140     REM
150     REM MODE 16 COULEURS (de CA a FA selon la taille)
160     POKE CB,#FA
170     REM
180     PRINT A$
190     REM
200     REM ON CHOISIT DES COULEURS  ALEATOIRES
210     FOR I=1 TO LEN(A$)
220     POKE AC-3,(RND(16) SHL 4)+CF
230     AC=AC-2
240     NEXT
250     REM
260     GOTO 100
270     REM
280     REM DETERMINER L'ADRESSE DU CONTROL BYTE & DU CURSEUR
290     REM
300     CB=PEEK(#79)*256+PEEK(#78)
310     AC=PEEK(#73)*256+PEEK(#72)
320     RETURN
*
```

ATTENTION : ce programme rend fous les caméléons !!!

fait, c'est très simple, à condition de faire preuve d'astuce, et la *Boîte à malices* est faite pour cela, n'est-ce pas ?

Pendant que nous y sommes, quitte à tripoter des octets, nous allons ajouter le moyen de *faire varier la taille des lettres*. Cela ne sera guère plus compliqué, car la routine proposée ici va se charger de tout le travail : l'utilisateur n'a qu'à spécifier ce qu'il veut écrire (dans la chaîne A$), où il veut l'écrire (commande CURSOR X,Y). C'est tout ! Pour la taille des lettres, la ligne 160 effectue un POKE qui choisit simultanément (c'est économique !) le mode texte 16 couleurs *et la taille des lettres*. Les valeurs # CA, # DA, # EA, # FA donnent le choix entre quatre tailles (# FA est la plus petite).

Un peu de gaieté dans vos textes

Le mystère réside dans les lignes 280 à 320. Ce sous-programme se charge de déterminer les adresses ennuyeuses à trouver : celle du CONTROL BYTE (CB), et celle de la position courante du curseur (AC). Dès lors, le POKE de la ligne 160 ajuste la ligne d'écriture au mode désiré, et la boucle des lignes 200-240 allume des couleurs aléatoires (une parmi 16) pour chaque lettre de la chaîne A$, imprimée en ligne 180. Comme le programme se charge de tout, vous pourrez même oublier ces explications, et continuer à avoir des textes multicolores tant que vous le voulez. C'est tout de même plus gai !

Alain MARIATTE

PC-1211

UNE OPÉRATION DÉLICATE

Quand on le pousse dans ses derniers retranchements, un ordinateur sait-il toujours bien faire les additions ? Non. Le PC-1211 n'échappe pas à la règle. A propos, avez-vous songé à tester votre matériel sur ce point précis ?

Ce n'est pas tous les jours que l'on a réellement besoin d'exploiter tous les chiffres significatifs du résultat d'un calcul fourni par ordinateur. Mais, en cas de nécessité impérieuse, il importe de savoir jusqu'où on peut aller... assez loin.

Essayons donc de pousser un petit ordinateur, un PC-1211, dans ses derniers retranchements face à cette opération réputée la plus élémentaire : l'addition.

Rappelons que le PC-1211 affiche 10 chiffres, stocke également 10 chiffres, mais – et c'est essentiel – calcule les résultats non stockés, en particulier les résultats intermédiaires, avec 12 chiffres (les deux derniers chiffres sont les chiffres de garde). On est donc en droit d'espérer que le résultat *non stocké* d'une addition soit exprimé avec une précision absolue de 12 chiffres exacts. Voyons si notre espoir est fondé en prenant pour exemple, le cas le plus simple : la somme de deux nombres de même signe (tous deux positifs ou tous deux négatifs).

Nous nous proposons de vérifier :

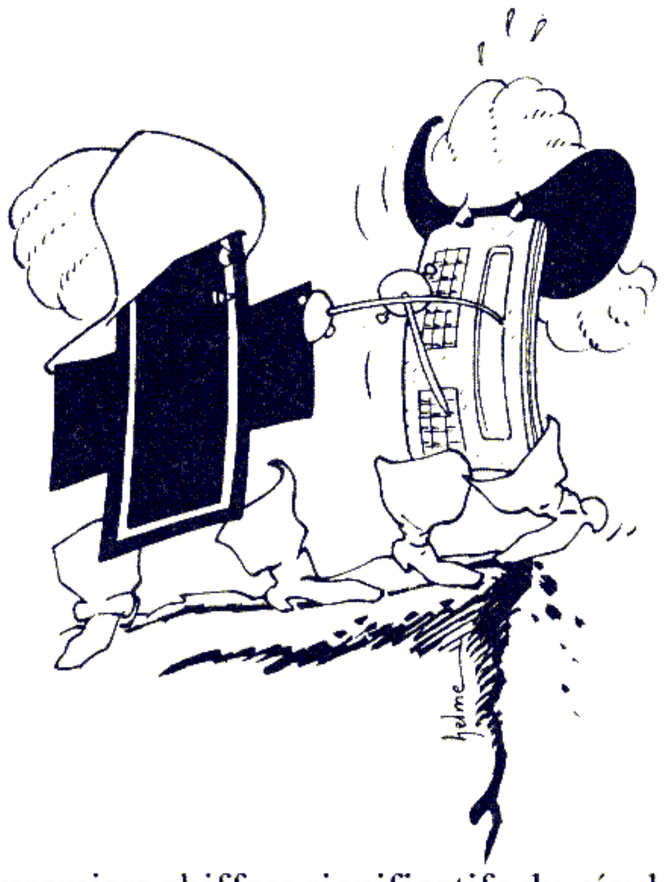
- si le résultat (sous entendu *non stocké*) est exprimé avec 12 chiffres exacts ;
- si l'ordre des deux termes de la somme est indifférent, soit si A + B et B + A donnent le même résultat.

L'analyse d'additions dont les données et le résultat comportent moins de 11 chiffres est sans intérêt : ce résultat est toujours exact ! Avec des données de plus de 13 chiffres, l'erreur est « normale » puisque la machine travaille sur 12 chiffres.

Il reste donc à examiner les additions dont le résultat comporte exactement 12 chiffres. Soit par exemple, la somme des deux nombres Z =

0,3605 et A = 0,000 000 999 957 : Z + A = 0,360 500 999 957.

En mode calcul, on affecte aux mémoires Z et A les valeurs ainsi définies et on vérifie l'exactitude du résultat. Si on fait Z + A ENTER, l'écran indique 0,360 501. On retrouve bien la valeur approchée, avec 10 chiffres significatifs, du résultat recherché. Si l'on veut récupérer les chiffres de garde, il faut soustraire de Z + A, *avant* d'appuyer sur ENTER, les deux premiers chiffres significatifs du résultat. Soit : Z + A − 0,36 ENTER. L'écran indique 0,000 500 999 957 (plus exactement 5.009 999 57 E − 04), et la preuve est ainsi faite que le calcul interne de Z + A a été effectué avec 12 chiffres exacts.



Inversons maintenant l'ordre des termes et introduisons au clavier A + Z − 0,36 ENTER. Surprise, l'écran indique 0,000 500 999 95. On a perdu le dernier chiffre significatif.

Vérifions : (Z + A) − (A + Z) ENTER donne 7 E − 12. C'est la valeur du chiffre perdu. On peut en conclure que le résultat d'une addition est susceptible d'être modifié par l'ordre d'introduction des termes.

Mais le plus surprenant, c'est que si l'on modifie la valeur absolue des termes sans en altérer le rapport, on obtient des résultats différents. Par exemple, si l'on multiplie par 10 les valeurs initiales de Z et de A, soit Z = 3,605 et A = 0,000 009 999 57, les conclusions précédemment établies ne sont plus valables : il n'y a pas de perte du dernier chiffre et (Z + A) − (A + Z) est égal à zéro.

En revanche, en multipliant par 10^7 les valeurs initiales de Z et de A, on retrouve les mêmes conclusions que dans le premier exemple, avec perte du dernier chiffre significatif.

Nous avons voulu tirer au clair ce phénomène et, après de nombreux tests, on peut conclure que :

- quel que soit l'ordre des termes, il n'y a pas de perte du dernier chiffre si l'un des termes est supérieur ou égal à 1 et l'autre inférieur ;
- le dernier chiffre est éventuellement perdu si les deux termes sont supérieurs à 1, ou bien si tous deux sont inférieurs à 1 ;
- si les termes sont introduits dans l'ordre décroissant, il n'y a pas de perte du douzième chiffre.

Cette dernière propriété reste valable même si le résultat de l'addition comporte 13 chiffres ou plus. Par exemple, pour Z = 0,360 5 et A = 0,000 000 999 957 364, la machine donne : Z + A − 0,36 = 0,000 500 999 957 et A + Z − 0,36 = 0,000 500 999 95. Là encore, on gagne un chiffre en introduisant d'abord le plus grand terme.

Dans la mesure où l'on connaît l'ordre de grandeur des variables, on a intérêt, avec un PC-1211, à introduire les termes d'une addition dans l'ordre décroissant. Et dans un programme, on pourra même écrire LET Z = D + A + C + B, au lieu de l'éternel LET Z = A + B + C + D.

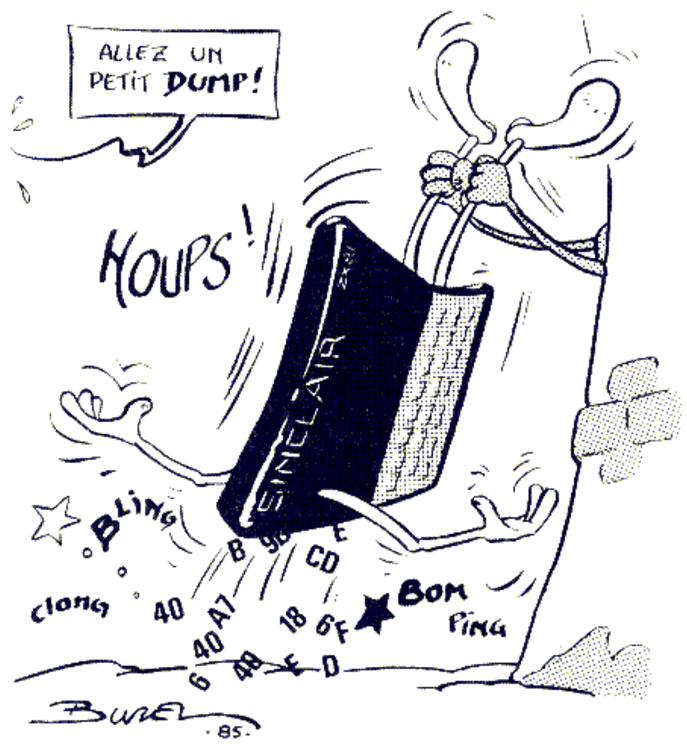
Pierre Ladislas GEDO

ZX 81 + 16 Ko

QU'Y A-T-IL EN MÉMOIRE ?

En anglais, cela s'appelle un « dump », ce qui se traduit par « vidage ». En fait, dans le cas qui nous occupe, il s'agit d'afficher sur l'écran de l'ordinateur le contenu des mémoires, mais sans altérer ce contenu bien sûr.

Nous obtiendrons un affichage de 16 lignes dont chacune est formée par le contenu de 8 adresses consécutives. Une page d'écran représente donc le contenu de 128 adresses en mémoire. Pour que l'utilisateur s'y retrouve facilement, au début de chaque ligne se trouve affichée la représentation hexadécimale de la première adresse.



Ceux et celles qui ont connu

Dump en langage-machine
Utilitaire pour ZX 81 (16 Ko)
Auteur Sylvain Coron
Copyright *LIST* et l'auteur

Chargement de la routine

```
1 REM 12345678901234567890123456789012345678901234567890123456789012345
2 PRINT AT 20, 1 ; " ENTREZ LES CODES HEXA 1 PAR 1 TERMINER PAR UN POINT (.) "
3 FOR F = 16514 TO 17777
4 INPUT H$
5 IF H$ = "." THEN STOP
6 LET A = 16 * (CODE H$(1) − 28) + CODE H$ (2) − 28
7 SCROLL
8 POKE F,A
9 PRINT F; " " ; H$
10 NEXT F
```

La routine en langage-machine

```
16514      : 40 82 06 10        Ld B, 10 H          Initialise le compteur de lignes à 16.
LIGNE      : 40 84 21 7C 40     Ld HL, 40 7C H      Place dans HL l'adresse de début de ligne.
             40 87 18 13        Jr, SUITE
TRAD       : 40 89 3E 00        Ld A, 00 H          ┐ Routine de traduction en caractères Hexa du contenu
             40 8B CD 94 40     Call IMPR           │ de l'adresse traitée et impression en Hexa du code
             40 8E CD 94 40     Call IMPR           │ obtenu.
             40 91 ED 6F        RLD                 │
             40 93 C9           Ret                 ┘
IMPR       : 40 94 ED 6F        RLD                 ┐ Routine de rotation pour impression du code Hexa de
             40 96 C6 1C        Add 1CH             │ chaque demi-octet.
             40 98 D7           Rst 10H             │
             40 99 D6 1C        Sub 1CH             │
             40 9B C9           Ret                 ┘
SUITE      : 40 9C CD 89 40     Call TRAD           ┐ Impression du code Hexa de l'adresse d'un début de ligne
             40 9F 2B           Dec HC              │ (toutes les 8 adresses).
             40 A0 CD 89 40     Call TRAD           ┘
             40 A3 0E 08        LdC, 08 H           Initialise le compteur d'octets à 8 par ligne.
OCT. SUIV. : 40 A5 3E 00        Ld A, 00 H          ┐ Impression d'un blanc séparateur.
             40 A7 D7           Rst 10 H            ┘
             40 A8 ED 5B 7B 40  Ld DE, (407BH)      ┐ Place dans DE l'adresse de l'octet à imprimer et recopie
             40 AC 1A           Ld A, (DE)          │ à l'adresse 407BH le contenu de l'adresse analysée ; qui est
             40 AD 32 7B 40     Ld (40 7BH), A      ┘ l'octet traité.
             40 B0 21 7B 40     Ld HL, 407B H       Fait pointer HL sur l'adresse où est l'octet traité.
             40 B3 CD 89 40     Call TRAD           Appelle l'impression du code de l'octet.
             40 B6 0D           Dec C               ┐ Si on a traité 8 adresses, saut au test de lignes, sinon incré-
             40 B7 28 07        Jrz, TEST LIGN      │ menter l'adresse à traiter et renvoi à l'impression de
             40 B9 13           Inc DE              │ l'octet correspondant.
             40 BA ED 53 7B 40  Ld (40 7BH), DE     │
             40 BE 18 E7        Jr, OCT. SUIV.      ┘
TEST LIGN. : 40 C0 D7           Rst 10H             ┐ Termine le remplissage de la ligne en cours avec 4 blancs.
             40 C1 D7           Rst 10H             │
             40 C2 D7           Rst 10 H            │
             40 C3 D7           Rst 10 H            ┘
             40 C4 05           Dec B               ┐ Si les 16 lignes ont été imprimées, alors retour au Basic,
             40 C5 C8           Ret Z               │ sinon incrémenter l'adresse à traiter et retourner à l'im-
             40 C6 13           Inc DE              │ pression d'une ligne.
             40 C7 ED 53 7B 40  Ld (40 7BH), DE     │
             40 CB 18 B7        Jr LIGNE            ┘
```

Programme Basic d'appel de la routine LM

```
1 REM ←——75 caractères——→
2 PRINT " ADRESSE DE DEPART
  DU DUMP "
3 INPUT A
4 LET B = INT (A/256)
5 POKE 16507, A-256*B
6 POKE 16508, B
7 CLS
8 RAND USR 16514
```

Avec le ZX 81 équipé des 16 Ko de mémoire supplémentaire, le « dump » est immédiat. L'exécution avec le ZX 81 en version de base est un peu plus lente car la machine évalue continuellement la mémoire disponible en fonction des affichages.

Début de l'implantation de la routine

```
 ENTREZ LES CODES HEXA 1 PAR 1
 TERMINER PAR UN POINT (.)
16514  05
16515  10
16516  21
16517  7C
16518  40
16519  18
16520  13
16521  3E
16522  00
16523  CD
16524  94
16525  40
16526  CD
16527  94
16528  40
16529  ED
16530  6F
```

l'épreuve de la traduction Déci-mal/Hexa d'un extrait de mémoire morte, ou l'attente d'un résultat de conversion de ce même extrait par un programme Basic, apprécieront certainement la vitesse.

Pour charger notre programme, on commencera par taper le petit utilitaire de la page précédente, qui effectuera la conversion Hexa/Déci et qui chargera les codes voulus en mémoire. (Cet utilitaire peut d'ailleurs être utilisé pour introduire n'importe quel programme LM en hexadécimal.)

La ligne 1 comporte une instruction REM suivie de 75 caractères à l'emplacement desquels les codes de la routine en langage-machine seront pokés par l'utilitaire. Une fois la routine implantée, on conservera la ligne 1 et l'on tapera les lignes 2 à 8 du programme d'appel.

La routine est transposable sur d'autres machines à condition toutefois de modifier les adresses d'appel (Call IMPR et Call TRAD) dont les octets ont été encadrés dans la liste.

Sylvain CORON

COMMODORE 64

LISTAGE DÉTOURNÉ

Problème : vous aimeriez lister sur votre imprimante un logiciel écrit à l'aide d'un utilitaire d'extension du Basic (comme *The Tool*, par exemple). Or, vous ne possédez pas cet utilitaire... Dans ces conditions, la liste obtenue par la méthode normale produit un résultat désastreux, qui ne correspond en rien au programme réel, puisqu'alors l'ordinateur est incapable de traduire les mots clés spécifiques de l'utilitaire.

Vous pouvez quand même obtenir une liste correcte. Il faudra toutefois (c'est le plus difficile) avoir pris la précaution de sauvegarder le programme sous la forme d'un fichier séquentiel ASCII. Ceci se réalise facilement, lorsque le programme est en mémoire (avec l'utilitaire d'extension en place)

en tapant : OPEN 3,8,3, "titre,S, W":CMD 3:LIST

Puis, quand le curseur réapparaît : PRINT # 3: CLOSE 3

Pour réaliser cette première partie de l'opération, il faut évidemment disposer de l'utilitaire. Puisque vous ne l'avez pas, il faudra demander au fournisseur du programme de vous redonner ce dernier, sous la forme indiquée plus haut.

Ayant maintenant en main le fichier-programme, il suffit d'un programme simple pour en obtenir le listage qui sera rigoureusement semblable au listage du vrai programme :

```
10 OPEN 3,8,3, "TITRE,S,R"
20 OPEN 1,4
30 GET # 3, A$
40 IF ST = 0 THEN PRINT # 1, A$;:
   GOTO 30
50 PRINT # 1
60 CLOSE 1: CLOSE 3
70 END
```

Pour que ce truc marche avec un magnétophone, la sauvegarde doit se faire avec : OPEN 3,1,1, "TITRE": CMD 3: LIST

Et pour la relecture, la ligne 10 devient : 10 OPEN 3,1,0, "TITRE"

Ce truc peut être utilisé pour lister sur un autre CBM les programmes du C.64, ou du Vic 20, ou encore ceux des nouveaux C.16 et PLUS 4.

Robin BOIS

CANON X-07

DELETE ET RENUM

Premier utilitaire : simuler l'ordre DELETE grâce auquel on supprime en une fois plusieurs lignes d'un programme. On lance le programme (RUN " DELETE ") et on indique les numéros de la première et de la dernière ligne à détruire.

Pour renuméroter vos programmes maintenant, je vous propose deux solutions. La première est à la fois simple et rapide. Comme pour DELETE, l'utilitaire doit être inscrit en zone fichier ; on le lance donc au moyen de RUN " RENUM ".

Si l'on veut au contraire le rajouter à la fin du programme à renuméroter, on insérera une ligne 65528 qui empêchera l'auto-renumérotation de l'utilitaire :

65528 IF PEEK (Z+2) + PEEK (Z+3) * 256 = 65525 THEN END

La seconde solution est plus lente à l'exécution mais beaucoup plus pratique. Elle indique en effet sur l'imprimante l'ancien et le nouveau numéro des lignes ainsi que la présence des GOTO, des GOSUB et des THEN, ce qui facilite grandement la renumérotation des branchements.

Yves DRILLET

Delete et Renum
Utilitaires pour le Canon X-07
Auteur Yves Drillet

DELETE

```
5 'DELETE
10 CLS:INPUT"1ere & derniere lignes a ann
uler";C,D
20 Z=1363:INPUT"D'accord (O/N)";E$:IFE$<
>"O"THENEND
30 INIT#1," ",500
40 A=PEEK(Z)+256*PEEK(Z+1):B=PEEK(Z+2)+2
56*PEEK(Z+3):IFA=0THEN70
50 IFB>=CANDB<=DTHENPRINT#1,STR$(B)
60 Z=A:GOTO40
70 INIT#5," ":EXEC&HEE1F:DELETE" ","D":E
ND
```

RENUM version courte

```
65525 'RENUM
65526 Z=1363:INPUT"No 1e LIGNE & INCREME
NT";D,C
65527 A=PEEK(Z)+PEEK(Z+1)*256:IFA=0THENE
ND
65529 POKEZ+2,DMOD256:POKEZ+3,INT(D/256)
:D=D+C:Z=A:GOTO65527
```

RENUM version longue

```
65520 'RENUM
65521 Z=1363:INPUT"No 1e LIGNE & INCREME
NT";D,C:GOSUB65528
65522 Z=Z+1:E=PEEK(Z):IFE=0THENIFPEEK(Z+
2)=0THENENDELSEZ=Z+1:GOSUB65528
65523 IFE=136ORE=140ORE=206THENLPRINT"*"
;
65526 GOTO65522
65528 LPRINT[1,1]"-";PEEK(Z+2)+PEEK(Z+3)
*256;"/";D;
65529 POKEZ+2,DMOD256:POKEZ+3,INT(D/256)
:D=D+C:Z=Z+3:RETURN
```

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...

19

Fonctions logiques

Les fonctions « logiques » sont appliquées à des propositions qui ne peuvent prendre que deux valeurs : vrai ou faux. Et le résultat est soit vrai, soit faux.

Ainsi, le ou exclusif (ox), l'équivalence (equ), l'implication (imp) et la réciprocité (rpc) sont définis par une « table de vérité » (voir ci-dessous).

Cette table doit son nom aux valeurs qui la composent : 0 est la représentation de « faux » et 1 celle de « vrai ».

On peut y lire que A ox B est « vrai » si A ou B l'est, mais pas les deux ; A equ B est vrai s'ils ont la même valeur, etc.

Ces fonctions sont rarement disponibles dans les langages de programmation : essayez donc de trouver quatre expressions logiques pour les remplacer.

Table de vérité

A	B	A ox B	A equ B	A imp B	A rpc B
0	0	0	1	1	1
0	1	1	0	1	0
1	0	1	0	0	1
1	1	0	1	1	1

20

Interdit de sauter

Le petit extrait de programme présenté plus loin provient d'un progiciel de comptabilité. Cette comptabilité permet de travailler sur deux

---

** Les solutions des jeux proposés ici paraîtront dans le prochain numéro de LIST.*

périodes comptables (exercices) en même temps. Il est possible d'ajouter ou de modifier des écritures sur un mois donné tant que celui-ci n'a pas fait l'objet d'une clôture.

Cette application utilise un tableau nommé *Arrete*, de 24 éléments qui correspondent aux mois de 2 années. Chaque élément de ce tableau mémorise 0 si le mois n'a pas été clôturé (ou centralisé, pour les comptables), et une autre valeur selon les types de clôtures qui ont été réalisées.

Les instructions ci-dessous recherchent le premier arrêté non clôturé. Il s'agit du premier élément du tableau *Arrete* qui est égal à 0. A l'étiquette Suite, la valeur de Prear correspond à l'indice du premier élément de *Arrete* qui est égal à 0.

```
POUR I:= 1 JUSQU_A 24
   SI Arrete(I)=0 ALORS
      Prear:=I
      ALLER_A Suite
   FIN_SI
FIN_POUR
Suite...
```

Nous savons que les sauts (ALLER_A) ne sont pas à recommander dans les programmes, puisqu'ils nuisent à leur clarté. D'ailleurs, on dit souvent que la qualité d'un programmeur est inversement proportionnelle au nombre de GOTO utilisés. Pour remédier à cela, nous pouvons écrire ce fragment de programme de la façon suivante :

```
POUR I:= 1 JUSQU_A 24
   SI Arrete(I)=0 ALORS Prear:=I
FIN_POUR
```

Nous pouvons toutefois nous poser les questions suivantes :

1. Ce dernier extrait de programme fournit-il **toujours** le même résultat que le précédent ? Sinon, comment peut-on le corriger simplement ?
2. Comment peut-on encore le récrire pour qu'il soit plus rapide à l'exécution ?

21

Max et les tableaux

Dans certains langages, la fonction MAX admet un tableau comme paramètre. Il est alors possible de rechercher, en une instruction, la plus grande valeur d'un tableau.

Si une telle fonction n'est pas disponible, il est nécessaire d'écrire quelques instructions comme dans le programme Basic ci-dessous. Il travaille sur un tableau T de 10 éléments indicés de 1 à 10. Il est chargé de rechercher et d'afficher la plus grande valeur mémorisée dans le tableau.

```
10 M=0
20 for I=1 to 10
30 if T (I) <=M then goto 50
40 M=T (I)
50 next I
60 print M
```

Trois questions se posent au sujet de ce programme.

1. Comment le rendre plus clair en supprimant le GOTO de la ligne 30 ?

2. Dans quels cas le résultat est-il faux ? C'est-à-dire dans quelles circonstances la valeur affichée ne correspond-elle pas à la plus grande valeur du tableau T ?

3. Comment corriger ce programme pour que le résultat soit toujours juste ?

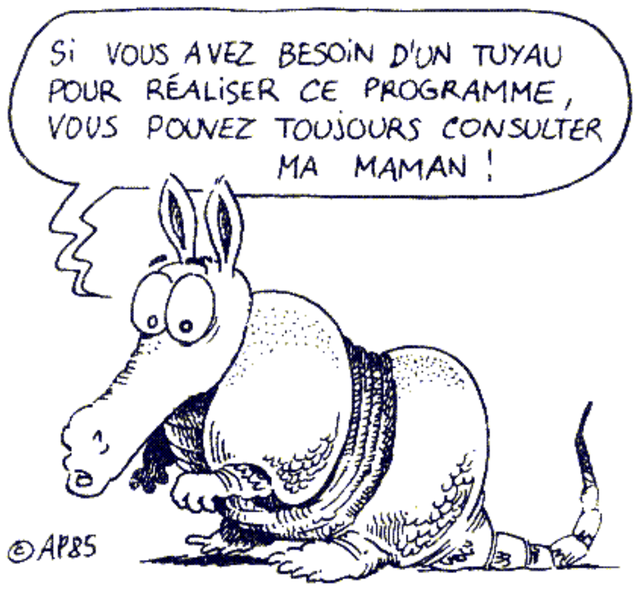
Pourriez-vous répondre à toutes ces questions ?

22

Les clones informatiques

Vous avez certainement déjà entendu parler des clones : ce sont des individus parfaitement semblables. Dans le monde végétal, une bouture de géranium, par exemple, est un clone. Dans le monde animal, les femelles tatous (mammifères d'Amérique tropicale) mettent au monde quatre petits clones.

Dans le monde informatique, tout est à faire : pouvez-vous écrire un programme clone dont l'exécution ne produira rien d'autre que la liste exacte du programme lui-même (des instructions comme PEEK, POKE ou LIST sont interdites).



▶▶▶

LES JEUX ET CASSE-TÊTE INFORMATIQUES

SOLUTIONS DU NUMÉRO PRÉCÉDENT

Les solutions présentées ici répondent aux problèmes posés dans le précédent numéro de LIST. Ce ne sont pas forcément les meilleures !

16

Opérations DIV et MOD

Sans l'aide d'un ordinateur, on trouve les réponses qu'il ferait aux dix opérations proposées avec DIV et MOD :

1. A MOD 1 donne 0
2. A MOD A donne 0
3. A MOD −1 donne 0
4. A MOD −A donne 0
5. A MOD 0 donne un message d'erreur
6. A DIV 1 donne A
7. A DIV A donne 1
8. A DIV −1 donne −1
9. A DIV− A donne − 1
10. A DIV 0 donne un message d'erreur

17

Construire la matrice unité

L'initialisation en moins de 20 secondes d'une matrice unité de 100 lignes sur 100 colonnes peut être effectuée par le programme qui suit :

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

Comme MAXLIG = 100 et MAXCOL = 100 pour cette matrice, le temps d'exécution d'un tel programme est de : 100 fois la ligne 10, (100 × 100) fois la ligne 20, (100 × 100) fois la ligne 30, 100 fois la ligne 50. Soit en microsecondes : (100 × 850) + (10000 × 850) + (10000 × 1130) + (100 × 1130).

Le résultat de ces opérations donne 19 998 000 microsecondes, ce qui est inférieur à 20 secondes.

On le constate, ce programme commence par initialiser toute la matrice à zéro, puis, dans un second temps, il met la valeur un dans les éléments de la diagonale. Des calculs inutiles sont donc effectués, mais le temps d'exécution se trouve réduit par l'élimination du test consistant à détecter les cas particuliers. En effet, il est inutile de tester N^2 fois si L vaut C pour avoir seulement N réponses vraies.

18

La fausse égalité

L'associativité de la multiplication permet d'écrire l'égalité (A*B) *C = A*(B*C). L'ordinateur, lui, semble ignorer cette propriété.

En prenant, par exemple, A = 1.6, B = 1.00008, C = 6.00016, un ordinateur qui effectue les calculs avec six chiffres significatifs renverra les résultats suivants :
(A*B)*C = 1.60012*C = 9.60097 et
A*(B*C) = A*6.00064 = 9.60102 (ces égalités tiennent compte de la troncature des nombres à 6 chiffres lors des calculs).

Après seulement deux multiplications, seuls trois chiffres significatifs sur six peuvent donc être considérés comme sûrs.

En prenant d'autres valeurs, par exemple : A = 1.1, B = 1.00009, C = 8.18113, on obtient des résultats encore moins bons. En effet, (A*B)*C = 1.10009*C = 8.99997 et A*(B*C) = A*8.18186 = 9.00004.

Dans certains domaines, comme la météorologie, où de nombreux calculs doivent être effectués, les ordinateurs utilisés travaillent avec un nombre plus important de chiffres significatifs. Mais des erreurs subsistent.

LES DIX TESTS DE LIST

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.

Ordinateurs de poche et ordinateurs de table sont réunis dans le même tableau : leur Basic est mis à l'épreuve. Sur les quarante testés ici, une dizaine "entrent dans la poche". Ils sont petits, autonomes (ils n'ont pas besoin du secteur, des piles suffisent), ils ont des écrans à cristaux liquides, des mémoires permanentes (même éteints, ils conservent les programmes) et ils se distinguent généralement par le nombre de leurs fonctions scientifiques préprogrammées.

Les ordinateurs de table ont un véritable écran, celui de la télévision ou d'un moniteur, et leur Basic possède souvent des fonctions graphiques. La taille de la mémoire de base est généralement plus importante que celle des ordinateurs de poche.

Les tests ont été rédigés sous une forme "standard", celle qui convient au plus grand nombre. Dans la mesure du possible, ils ont été appliqués sous cette forme, sans tenir compte des caractéristiques propres à chaque Basic. Par exemple, certains auraient supporté une boucle FOR...NEXT sans spécification de variable (la boucle se termine alors par NEXT au lieu de NEXT I) et ainsi, les résultats des tests auraient été meilleurs. Mais cette propriété n'est pas vraie sur tous les Basic.

Face aux dix tests de *LIST*, le Basic le plus lent (parmi les quarante testés) est celui du PC-1211. Il s'agit là du *premier* ordinateur de poche programmable en Basic. Ses qualités, très originales au moment de sa naissance, excusent cette lenteur.

Parmi les records de vitesse, on remarque le Dai (tests 1 et 3), le BBC

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

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

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

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

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

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

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

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

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

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

MESURER LE BASIC

(entre la première et la deuxième place sur tous les tests, sauf les 6, 9 et 10), le QL (tests 5 et 6, c'est-à-dire arithmétique et calcul scientifique), le Lynx (test 4) ; mais aussi, l'Electron, l'Hector ou l'Amstrad.

Pour les tests 9 et 10, l'Atmos semble donner des résultats particulièrement bons. Son système d'exploitation des disquettes (DOS) est très rapide : il est vendu pour 250 000 bauds (bits par seconde), ce qui est en effet très rapide.

Quant à Macintosh, il n'a pas de Basic résident. Celui qui est testé ici est la première version du Basic Microsoft (sous forme de disquette).

On ne dira jamais assez que les comparaisons ne peuvent être entreprises qu'avec prudence. Chaque Basic a ses caractéristiques. L'absence de norme permet à certains de briller là où d'autres sont faibles, et réciproquement.

LIST

Résultats des dix tests

Ordinateurs	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Processeur
Alice 90	12	38	54	135	87	460 9 chiffres	237				6 803
Amstrad CPC 464	11	23	44	73	66	285 9 chiffres	545	1 130 320 × 200	765 cassette	765 cassette	Z80A
Apple IIe	13	43	65	136	93	479 9 chiffres	130	484 280 × 160	404 disquette	710 disquette	6 502
Atari 600XL	22	88	78	271	204	2 040 9 chiffres	225	1 766 320 × 192			6 502C
Atmos	17	130	89	187	123	629 9 chiffres	244	766 240 × 200	141 disquette	77 disquette	6 502
BBC	7	15	32	41	57	405 9 chiffres	114	239 160 × 256	1 944 cassette	1 944 cassette	6 502A
Canon X07	37	79	100	236	383	1 680 14 chiffres	2 420	2 830 120 × 32	43 300 cassette	43 300 cassette	NSC 800
Colour Compurer 2	14	47	69	173	116	598 9 chiffres	168	880 256 × 192	998 cassette	998 cassette	6 809
Commodore 64	15	44	75	155	105	295 9 chiffres	200	731 320 × 200	394 disquette	323 disquette	6 510
Dai 48K	5	30	21	87	87	799 6 chiffres	427	2 518 336 × 256	77 disquette	62 disquette	8 080
Dragon 32	14	49	68	162	112	590 9 chiffres	262	874 256 × 192	1 103 cassette	1 103 cassette	6 809E
Electron	9	20	43	55	64	578 9 chiffres	191	299 160 × 256			6 502A

Les temps sont exprimés en secondes

Résultats des dix tests

Ordinateurs	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Processeur
EXL 100	21	76	80	273	232	3 090 14 chiffres	457	790 320×250			TMS 7 020
FX-702P	180	920	740	1 220	1 250	4 450 12 chiffres	2 550				Casio
Hector HRX	7	54	34	86	57	202 7 chiffres	1 180	423 231×243			Z80
HP-71B	95	206	97	266	262	823 15 chiffres	1 146		420 mémoire	200 mémoire	HP
HX-20	25	71	152	209	179	558 9 chiffres	1 973	1 234 120×32	2 460 cassette	2 400 cassette	6301
Lynx 48K	10	34	38	34	124	814 7 chiffres	4 400	2 520 256×248			Z80
Macintosh (Basic Microsoft)	10	24	33	101	92	778 14 chiffres	890	448 496×294	85 disquette	67 disquette	MC68000
MO 5	16	45	87	135	122	376 6 chiffres	295	1 430 320×200	940 cassette	940 cassette	6809
MPF II	15	46	73	170	98	505 9 chiffres	337	517 280×192			6 502
Oric-1	20	150	104	217	142	1 058 9 chiffres	284	897 240×200			6 502
PB-100	70	480		470	440	1 700 12 chiffres	1 000				Casio
PB-700	115	547	705	1 500	750	2 100 12 chiffres	1 450	2 750 160×32			Casio
PC-1211/1212	2 350	4 450	4 672		4 900	11 300 12 chiffres					Sharp
PC-1251/1255	427	970	870	1 430	1 400	5 615 10 chiffres	1 547				Sharp
PC-1260	75	296	399	970	740	3 180 12 chiffres	1 220				Sharp
PC-1350	69	316	384	940	710	3 180 12 chiffres	1 620	3 500 150×32			Sharp
PC-1500	149	435	298	691	608	2 897 10 chiffres	503				Sharp
QL Sinclair	19	68	66	119	54	145 8 chiffres	1 269	2 293 512×256	135 microdrive	99 microdrive	68 008
QX-10	21	56	85	146	99	305 16 chiffres	499	67 320×200	132 disquette	126 disquette	Z80/8 049
Spectrum et Spectrum +	42	106	115	195	132	1 180 8 chiffres	243	610 256×176			Z80
TI99/4A	28	58	138	2 320	168	3 190 14 chiffres	2 500		1 010 disquette	1 840 disquette	TMS 9 900
TO 7	18	46	100	138	124	417 6 chiffres	222	1 037 320×200			6 809
TRS-80 Modèle 1	27	86	111	213	166	516 6 chiffres	263				Z80
TRS-80 Modèle 4	10	23	46	70	64	244 6 chiffres	339		283 disquette	165 disquette	Z80A
Vic 20	12	37	63	130	88	445 9 chiffres	173		369 disquette	310 disquette	6 502
Yéno MSX	20	41	60	108	185	1 937 16 chiffres	139	1 165 256×192	1 863 cassette	1 863 cassette	Z80
ZX81 (mode fast)	45	100	100	160	125	1 150 9 chiffres					Z80
ZX81 (mode slow)	176	380	400	640	500	4 540 9 chiffres	5 300				Z80

Les temps sont exprimés en secondes

Imprimerie SIMA, rue des Epinettes, ZI de Torcy, 77200 Torcy. Printed in France. Directeur de la publication : Jean-Luc Verhoye.
Dépôt légal février 1985. Diffusion NMPP. Numéro de commission paritaire : 63617