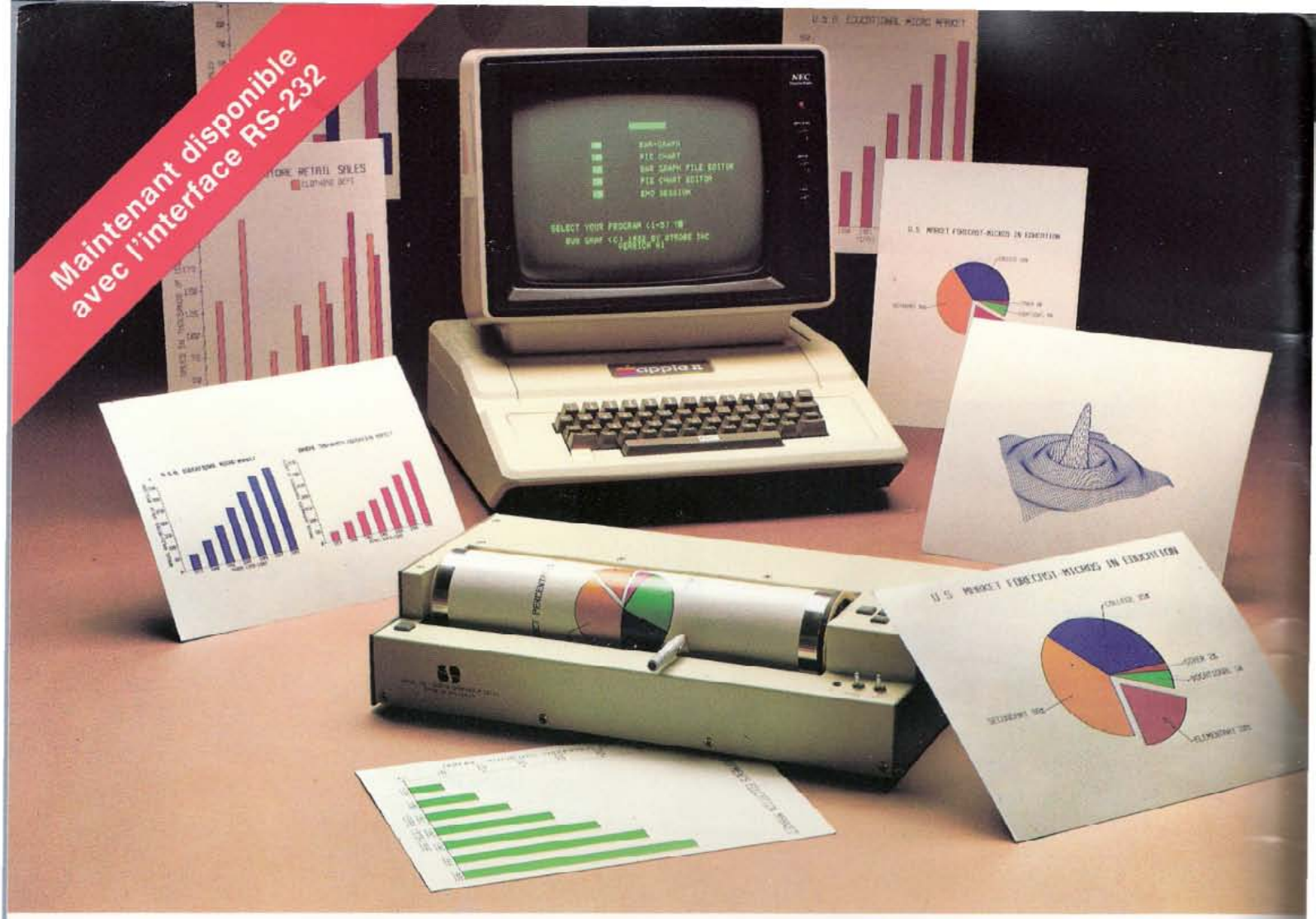
POM'S

NUMERO 6
DECEMBRE 1982
35 F

ISSN: 0294-6068



Mieux qu'un long rapport montrez vos courbes

C'est un fait, une courbe ou un diagramme donnent instantanément les informations essentielles dont vous avez besoin, sans dépenser de longues heures à dépouiller vos «listings».

Maintenant, la table traçante **STROBE M 100** est disponible avec de nombreux logiciels, utilisable directement sur votre microcalculateur à un prix compétitif.

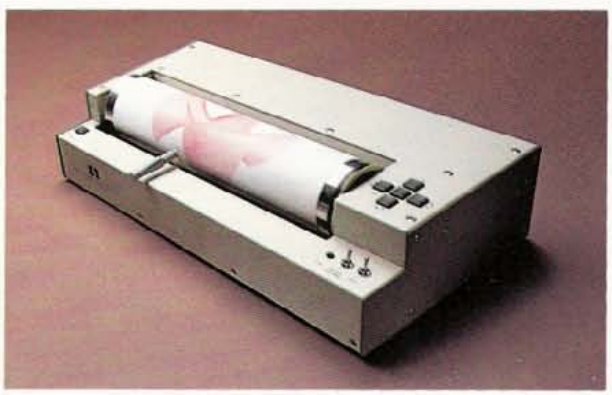
Les logiciels fournis par STROBE permettent le tracé et la modification des courbes sans connaissance approfondie de l'informatique, en utilisant toute la puissance du **langage conversationnel.**

Les informations peuvent être représentées sur papier format standard et sur transparant utilisable en rétroprojection, sous forme :
— d'histogramme,
— de courbes,
— de courbes isométriques.

La définition est de 200 points par cm.

La table traçante STROBE est interfaçable aux microcalculateurs les plus diffusés :

— Apple II, III ™ **— CBM/PET ™**
— OSBORNE, ™ **— XEROX 820 ™**



Strobe Inc.
28, rue de Belle Vue
BOX 7, 1050 Bruxelles,
Belgique
Tél.: (32) (2) 649-5663

Pour tous renseignements et démonstrations

jod électronique
9, rue Noblet,
92500 Rueil-Malmaison
Tél.: 749.70.44.

LA TABLE TRAÇANTE STROBE
Voir c'est croire

TM : marques déposées

pom's n°6

Sommaire

Éditions MEV
49 rue Lamartine
78000 Versailles

Directeur de la publication Hervé Thiriez.
Imprimerie Sim, 75011 Paris.
Imprimé en France.
Dépôt légal : 4e trimestre 1982.

Annonceurs

B.F.I.	4e couv.
B.I.P.	42
C.X. MULTIGESTION	18
ANDRE F. FINOT	61
JCR	3e couv.
LA NACELLE	58
LOGMA	57
MICROGES	64
MICRO INFORMATIQUE SERVICE	4
L'ORDINATEUR INDIVIDUEL	54
P.S.I.	19, 25, 63
STROBE	2e couv.

dis monsieur, apprends-moi à gérer un fichier.

apple II apple III

Carte MEM/DOS 6502

LE SYSTEME D'EXPLOITATION DU 6502 - MONOPOSTE/MULTIPOSTE

UNE EXTRÊME SIMPLICITÉ DE PROGRAMMATION.

- La division de la longueur des programmes par 20.
- La possibilité réelle de dessiner ses masques de saisie ou d'impression.
- Une indépendance totale de la périphérie choisie par rapport au système.
- L'intégralité du système contenu sur une carte mémoire de 20 K.
- Une gestion de mémoire de 140 K à 120 mégas.
- Des utilitaires déterminants
  - un générateur de programmes de gestion de fichiers permettant même le séquentiel indexé multiclé
  - un générateur d'écrans.
- CALL FN, une nouvelle commande basic, très puissante, intégrée au système permettant l'appel des sous-programmes par noms avec passage de paramètres et variables locales.
- Une version multiposte assurant la mise en commun totale des ressources sans conflit et l'autonomie des postes intelligents disposant de leur propre unité centrale.
- Des programmes compatibles APPLE II et APPLE III automatiquement transférables sur COMMODORE 8096.
- Et pour demain, des logiciels développés aujourd'hui directement compatibles avec le réseau local memnet.

S.E.P. Marketing

micro informatique service

3, rue Meyerbeer - 06000 NICE - Télex 461 916 F

DISTRIBUTEURS AGREES

D.S.A. INFORMATIQUE
5, bd Dubouchage
06000 NICE
Tél. (93) 85.15.96

MICRO ALPHA SOFT
11, impasse du Lacquet
25200 MONTBELIARD
Tél. (81) 97.16.46

S E E M I
61, rue Ch. Rivière - B.P. 0701
44401 REZE CEDEX
Tél. (40) 75.52.80

MICROMEGAS
22, rue des 3 Pierres
69007 LYON
Tél. (7) 861.19.52

G-B
C.I.C.C.
Grove house
the bordage
St Peter Port
GUERNSEY
(0481) 20155

BENELUX
MEGAVOLT S.A.
Rue de Bleurmont
32 B
B 4920 EMBOURG

Liste de nos revendeurs page 218

Editorial

Le numéro 6 de Pom's représente un grand changement : tout d'abord, nous avons modifié la couverture et la présentation, comme vous avez pu le remarquer immédiatement. Cela nous permet de vous donner, dans le même format, l'équivalent de quinze pages de plus ; en effet, on peut mettre 50 % de texte en plus dans une page composée. Nous avons aussi gagné une quinzaine de pages en mettant sur deux colonnes les listes en assembleur des programmes HELLO et TORTUE&, avec des caractères compressés. Dites-nous si vous préférez avoir ainsi plus de matière, ou si vous trouvez que la lisibilité en souffre trop. Enfin, le tirage de Pom's passe à 9 000 exemplaires.

Nous avons tenu compte de la remarque de plusieurs lecteurs au Sicob, se plaignant du niveau trop élevé des articles : un effort particulier a été réalisé quant à la réalisation d'articles pédagogiques. Remarquons en outre que, même si un article présentant un programme en assembleur peut paraître trop compliqué à lire, cela ne vous empêche pas d'utiliser ce programme à partir de la disquette (le mode d'emploi est fourni sur le programme MENU de chaque disquette).

Signalons enfin que le Recueil N° 1 de Pom's sera disponible dès le 27 décembre ; ce recueil regroupe, en près de 200 pages, la plupart des articles des quatre premiers numéros de Pom's ; l'ensemble des programmes est fourni sur trois disquettes. Cet album coûte 120 F seul, et 270 F avec ses 3 disquettes d'accompagnement, port compris. Nous sommes maintenant en rupture de stock des numéros 1 à 3 : seuls les numéros 4, 5 et 6 peuvent donc être commandés séparément (ou avec leurs disquettes).

Rappelons aussi qu'il est toujours possible de commander des disquettes séparément : 50 F par disquette d'accompagnement de Pom's, 75 F pour la disquette d'accompagnement du livre « Visicalc sur Apple» (spécifier dans ce dernier cas s'il s'agit de DOS 3.2, DOS 3.3 ou Apple III).

Enfin pour ceux d'entre vous qui souhaitent mettre de la couleur dans leur vie, nous proposons aussi le T-shirt avec la pomme en six couleurs (tailles 36-38, 38-40 et 40-42) à 50 F.

Passons maintenant au programme de ce numéro. Nous vous offrons une course de tortues, entre celle de Michel Crimont en Pascal, et celle de Jacques Duma en assembleur. Olivier Herz vous apporte un nouveau chef-d'œuvre avec l'analyseur de syntaxe, qui vous permettra de tester tout un programme BASIC, même dans ses corridors peu fréquentés. Thierry le Tallec et Jacques Tran-Van vous donnent enfin un programme de HELLO très performant que vous pourrez mettre sur toutes vos disquettes.

Gérard Michel vous offre un programme d'INPUT généralisé, avec un sous-programme en assembleur commenté de façon très détaillée. C'est particulièrement intéressant comme initiation à l'assembleur. Remarquons aussi, au niveau pédagogique, une nouvelle contribution de Guy Mathieu, relative à l'ergonomie des programmes.

Nous avons un banc d'essai, réalisé par André Babeanu, de « The Last One» et de C.O.R.P., des programmes qui écrivent vos programmes pour vous. Enfin, divers petits programmes se préparent à enrichir votre programmathèque et vos longues soirées d'hiver.

Hervé Thiriez

Ont collaboré à ce numéro : André Babeanu — Michel Crimont — Jacques Duma — Philippe Fabert — Olivier Herz — Régis Lardennois — Thierry Le Tallec — Guy Mathieu — Gérard Michel — Hervé Thiriez — Jacques Tran-Van.
Rédacteur : Olivier Herz — *Dessins :* Laurent Bidot et Bertrand Delestrée.

Directeur de la publication — rédacteur en chef : Hervé Thiriez — *Siège social et abonnements :* Éditions MEV — 49, rue Lamartine — 78000 Versailles — *Rédaction :* 59, bd de Glatigny — 78000 Versailles — Tél. (3) 918.13.07 — *Régie publicitaire :* Force 7 — 41, rue de la Grange-aux-Belles — 75483 Paris Cedex 10 — Tél. (1) 238.66.10 — *Diffusion auprès des boutiques informatiques et librairies :* Éditions du PSI — 41 51, rue Jacquard — B.P. 86 — 77400 Lagny-sur-Marne — Tél. (6) 007.59.31. *Composition-Impression :* SIM — 52, rue Servan — 75011 Paris — Tél. (1) 357.51.20.

Transfert d'Applesoft vers EXEC

Jacques Duma

Ce petit programme transforme tout ou partie d'un programme Applesoft en fichier de type TEXT dont l'EXECution charge en mémoire la partie de programme en question. De nombreux programmes ont été publiés à ce sujet, mais la particularité de celui ci tient à sa concision et à son automatisme complet.

Le programme doit être situé sur une disquette non protégée à l'écriture, car il crée (et efface à la fin) deux fichiers EXEC qui s'appellent l'un l'autre : K*P#T1 et K*P#T2. De plus, le drive par défaut doit être celui où est situé CAPTURE (il ne faut par exemple pas utiliser une séquence telle que LOAD CAPTURE, D1 suivi de CATALOGD2 et RUN).

Quand on le lance, CAPTURE demande le nom du programme Applesoft à capturer, le numéro du drive où il se situe et l'identification du bloc de lignes à capturer : il demande ensuite le nom du fichier EXEC à créer et le numéro du drive où il faudra le mémoriser. Il ajoute le préfixe [X.] à ce nom ; si l'utilisateur a appuyé sur RETURN sans donner de nom, le nom sera [X.] suivi du nom du programme Applesoft.

Tout le reste du travail s'effectue de façon entièrement automatique. Le lecteur intéressé par l'analyse de la façon dont tout cela se fait pourra faire un MONCIO avant de lancer CAPTURE, ou bien tout simplement étudier la conception de ce programme.

```
10  GOSUB 1000:T$ = "CAPTURE D'UN PROGRA
    MME  APPLESOFT": GOSUB 2000:T$ =
    "--------------------------------
    -": GOSUB 2000: PRINT
15 CD$ = ",D" +  STR$ ( PEEK (43624))
20 T$ = "CREATION D'UN FICHIER  EXEC": G
    OSUB 2000:T$ = "----------------
    ----------": GOSUB 2000: PRINT :
    PRINT : PRINT : PRINT
30 T$ = "NOM DU PROGRAMME A CAPTURER": G
    OSUB 2000: PRINT : INPUT N$: PRIN
    T : IF N$ = "" THEN  TEXT : HOME
    : PRINT "AU REVOIR.": END
32  INPUT "SUR QUEL DRIVE #";ND$:ND =  V
    AL (ND$): IF ND <  > 1 AND ND <
    > 2 THEN  PRINT  CHR$ (7);: GOTO
    32
33  INPUT "LIGNES (DEBUT-FIN) ? ";LI$: I
    F LI$ = "" THEN LI$ = "1-63999":
    GOTO 40
34 LI =  VAL (LI$): IF LI < 1 OR LI > 63
    999 OR LI <  >  INT (LI) GOTO 39
35  IF LI$ =  STR$ (LI) OR LI$ =  STR$ (
    LI) + "-" GOTO 40
36  IF  MID$ (LI$, LEN ( STR$ (LI)) + 1,
    1) <  > "-" GOTO 39
37 LJ =  VAL ( MID$ (LI$, LEN ( STR$ (LI
    )) + 2)): IF LJ < LI OR LJ > 6399
    9 OR LJ <  >  INT (LJ) GOTO 39
38  GOTO 40
39  PRINT  CHR$ (7);: GOTO 33
40  PRINT :T$ = "NOM DU FICHIER EXEC": G
    OSUB 2000: PRINT : INPUT X$: PRIN
    T : IF X$ = "" THEN X$ = "X." + N
    $: GOTO 45
42 X$ = "X." + X$
45  INPUT "SUR QUEL DRIVE #";XD$:XD =  V
    AL (XD$): IF XD$ = "" THEN XD = N
    D:XD$ =  STR$ (XD): GOTO 50
46  IF XD <  > 1 AND XD <  > 2 THEN  PRI
    NT  CHR$ (7);: GOTO 45
50  GOSUB 1000: VTAB 9:T$ = N$: GOSUB 20
    00: PRINT :T$ = "LIGNES " + LI$:
    GOSUB 2000: PRINT :T$ = "DRIVE #"
     + ND$ + " CONVERTI SUR DRIVE #"
    + XD$ + " EN": GOSUB 2000: PRINT
    :T$ = X$: GOSUB 2000
60  PRINT : PRINT : PRINT : PRINT : INPU
    T "EST-CE CORRECT ? ";R$: IF R$ <
     > "OUI" GOTO 10
80  GOSUB 1000: VTAB 11: FLASH :T$ = " U
    N PEU DE PATIENCE S.V.P. ": GOSUB
     2000: NORMAL
100 D$ =  CHR$ (4):G$ =  CHR$ (34):OP$ =
     "OPEN":WR$ = "WRITE":DL$ = "DELE
    TE":CL$ = "CLOSE":EX$ = "EXEC":K1
    $ = "K*P#T1":K2$ = "K*P#T2":CH$ =
     "PRINT" + G$ +  CHR$ (4) + G$ +
    ";"
110 ZERO$ = "O" + CH$ + G$ + OP$ + X$ +
    ",D" + XD$ + G$ + ":" + CH$ + G$
    + WR$ + X$ + G$ + ":POKE33,33:LIS
    T" + LI$ + ":" + CH$ + G$ + CL$ +
     G$ + ":" + CH$ + G$ + EX$ + K2$
    + CD$ + G$ + ":END"
200  PRINT D$OP$K1$CD$: PRINT D$WR$K1$:
     PRINT "LOAD"N$",D"ND$: PRINT ZERO
    $: PRINT "RUN": PRINT D$CL$
300  PRINT D$OP$K2$CD$: PRINT D$WR$K2$:
     PRINT "LOADCAPTURE": PRINT "RUN90
    0": PRINT D$CL$
500  PRINT D$EX$K1$CD$: END
900  PRINT : PRINT  CHR$ (4)"DELETEK*P#T
    1": PRINT  CHR$ (4)"DELETEK*P#T2"
    : GOTO 10
999  END
1000  TEXT : HOME : COLOR= 10: VLIN 0,46
     AT 0: VLIN 0,46 AT 39: HLIN 0,39
     AT 0: HLIN 0,39 AT 46: VTAB 5: H
    TAB 5: POKE 33,36: POKE 32,2: POK
    E 34,2: POKE 35,22: HOME : RETURN

2000  HTAB 20 -  LEN (T$) / 2: PRINT T$:
     RETURN
```

Logiciel graphique en Pascal

Michel Crimont

Généralités

Dans le précédent numéro de Pom's, nous avons vu comment créer des dessins sur une matrice 15*15 en graphique H.R. et conserver ceux-ci dans un fichier disque. Aujourd'hui, nous allons créer un langage de programmation « graphique interactif », langage permettant de créer, lister des programmes graphiques et bien entendu les exécuter et les conserver.

Dans ce présent article, nous appellerons *texte* la chaîne de caractères représentant le programme, *dessin* les dessins H.R. 15*15 et *planche* la page graphique complète.

Chaque texte représente une figure graphique plus ou moins complexe et porte un nom donné au moment de sa rédaction et nécessaire ensuite pour lister ou exécuter ce texte. Le texte lui-même est une chaîne de caractères (80 caractères maximum) ; un fichier permet de ranger 15 chaînes de ce type, donc 15 programmes différents qui pourront s'appeler les uns les autres ; par ailleurs, il sera toujours possible, lors de l'exécution d'un dessin, de charger un nouveau fichier de 15 textes.

Syntaxe du programme

Elle est la suivante :
[instruction], [instruction], [instruction].

Chaque instruction est séparée de la suivante par une virgule, le texte devant se terminer par un point. Noter que lors de la rédaction du texte, un retour chariot peut être inclus après une virgule pour faciliter l'écriture et la lecture en 40 colonnes. Les instructions comportent une lettre ou un signe décrivant la fonction à réaliser, éventuellement un signe + ou —, puis un chiffre de 0 à 255 ou une chaîne de caractères.

Les instructions sont :

D+ : crayon à blanc (nécessaire pour dessiner).
D— : pas de crayon (déplacement sans écriture).
A+n : avance le crayon de n.
A—n : recule le crayon de n.
T+n : tourne de n degrés dans le sens inverse des aiguilles d'une montre.
T—n : tourne de n degrés dans le sens des aiguilles d'une montre.
Px/y : positionne le crayon en x,y — le / s'inscrit seul lors de l'édition.
Z : éteint le crayon (équivalent à D—) ; positionne le crayon au centre de l'écran et le tourne vers la droite.
C+n : place le caractère n de SYSTEM.CHARSET sur l'écran, le coin inférieur gauche du caractère à la position du crayon. Le signe + inscrit un caractère normal, le — en inversé.
S+(chaîne) : place la chaîne de caractères sur l'écran à la position du crayon. Même signification du + ou du — que pour C.
>n« nom » : prend le dessin numéro n du fichier de dessin « nom » et le place sur l'écran à la position du crayon.
:<nomfigure> : prend dans le fichier texte en cours le dessin « nomfigure » et l'exécute.
Rn ([inst], [inst].) : répète n fois les fonctions entre parenthèses ; l'arrêt de répétition est obtenu en tapant un point, les parenthèses s'inscrivent seules lors de l'édition. A noter qu'une répétition peut contenir une autre répétition.
In : est un incrément, n un facteur multiplicatif n'agissant que sur les déplacements et non sur les angles ; par exemple I2,A+15,T+60 avance le crayon de (2×15)=30 et le tourne de 60 degré. Le facteur In est automatiquement incrémenté de 1 à chaque passage dans une boucle : exemple la figure nommée CARRE.
D+, R4 (A+30, T+90.)
et la figure MULTIPLE
D—, P10/10, I2, R3 (:CARRE.)

le curseur étant passé en 10×10, facteur de répétition à 2, donc le premier carré fera 60×60, le suivant 90×90 et le troisième 120×120.

Chaque figure, une fois définie, est conservée dans un fichier sur le disque.

Le programme

Il utilise les facilités d'analyse de syntaxe et de récursivité du PASCAL.

Ce programme pourra avantageusement être associé en un seul bloc avec le programme de graphisme de Pom's 5, si bien que l'on aura en même temps toutes les possibilités sous la main.

Les procédures PRENINFO, PRENCAR, PRENCHAINE, etc., ont déjà été définies et seront avantageusement incluses dans la librairie, ce qui libérera de la place pour le texte en cas de développement ultérieur.

Les fonctions ECRIMAGE et LIRIMAGE sont dues à un de nos lecteurs, M. Devernay ; elles permettent de transférer la page graphique sur le disque et vice versa sans occuper de buffer supplémentaire grâce aux procédures BLOCKREAD et BLOCKWRITE.

La procédure OPENLIST ouvre la liste de 15 chaînes programme et la laisse en mémoire ; éventuellement, si la liste n'existe pas, cette dernière est créée.

La procédure GETINF renvoie le numéro de fichier correspondant au nom d'un texte. Le fichier ne comportant que 15 textes programme au maximum, aucun tri n'est fait à ce niveau et la recherche est purement séquentielle.

La procédure GETINS : renvoie une chaîne de texte programme complète et utilise AJPM (ajoute + ou —) qui ne permet de rentrer au clavier que + ou —, AJBIT qui ne garde que les valeurs de 0 à 255 et AJCHAÎNE qui prend une chaîne de caractères.

La chaîne de texte en cours est toujours pointée sur le caractère à entrer par la variable INDEX déclarée globale afin de ne pas être influencée par les appels récursifs ; la procédure GETINS elle même gère la syntaxe de chaque instruction et évite toute erreur de frappe au clavier.

Le premier caractère de l'instruction est filtré entre tous les caractères possibles et la commande est placée dans la chaîne à la position de INDEX.

La syntaxe est ensuite guidée par l'instruction CASE. Pour A, T et C, obligation d'entrer + ou — suivi d'un chiffre entre 0 et 255.
Pour D, donner + ou —.
Pour I, un bit.
Pour S, un signe puis une chaîne de caractères (ici limitée à 20 caractères).
Pour P, deux bits successifs.
Pour >, un bit puis le nom du fichier disque.
Pour : le nom de la figure.
Enfin, pour R, un bit puis appel récursif à GETINS.

L'instruction étant totalement chargée, obligation d'entrer une virgule pour

l'instruction suivante ou un point si l'on a terminé ; la suite d'instruction dans une boucle R doit elle-même se terminer par un point.

Le système gère lui-même certains terminateurs, par exemple après la frappe de Px le signe / apparaît en séparateur ; après le Rn apparaît automatiquement une parenthèse gauche, et après la frappe du point la parenthèse droite correspondante. La frappe des lettres de commande et des signes ne nécessitent pas la frappe du RETURN. Par contre, il faudra taper ce dernier pour valider les chaînes de caractères ou les entiers.

A noter que, lorsqu'un RETURN suit une virgule, il est gardé dans le fichier, ce qui permet une écriture et une lecture plus agréables des textes à l'écran.

La procédure EDITE initialise le nom d'un texte à rentrer dans le fichier, cherche une place vide dans le fichier, initialise une chaîne de 80 espaces, l'envoie à GETINS pour prendre le texte du programme puis enregistre la chaîne de caractère et son nom dans le fichier.

Les procédures LIRE et LITINSTRUC sont calquées sur EDITE et GETINS : elles sont utilisées pour lire la chaîne texte tout en respectant la syntaxe d'écriture. La procédure CATALOG permet de lister les textes du fichier LISTE en cours et éventuellement d'effacer un texte caduc ou que l'on veut réécrire. La procédure EXECUTION place la variable IND à 1, début de la chaîne de caractères et exécute le texte PJET grâce à la procédure REALISE : cette procédure, à partir de la syntaxe du langage, effectue les différentes fonctions prévues sur la page graphique.

A noter :

— la récursion introduite par R qui appelle REALISE avec la chaîne d'instruction situées jusqu'à la dernière parenthèse fermante, ce qui permet à un Répète d'être à l'intérieur d'un autre Répète ;

— la récursion introduite par : « nom » qui rappelle EXECUTION pour le texte « nom ». On remarquera que pour A le déplacement est MOVE (x ⁑ INC). INC étant l'incrément, ce dernier ne pourra être modifié que par une nouvelle valeur de I. Si l'on désire qu'à la sortie des boucles INC retrouve automatiquement son ancienne valeur, il faudra déclarer INC dans EXECUTION et introduire INC:=1 dans le corps de la procédure avant l'appel de REALISE.

Le programme principal place le menu à l'écran et permet de choisir les différentes options. Certaines fonctions agissent au niveau des textes (1,2,3,4) et ne sont actives que si la ligne ——TEXTES⇒ "...§ nomme un fichier. Pour créer ou charger un fichier, il faudra donc appeler la fonction 1 et nommer le fichier ; ce fichier sera lu ou créé et son nom apparaîtra entre les crochets. A partir de ce moment, 2 permettra d'éditer une ligne supplémentaire de texte, 3 de lire un texte et 4 de lire le catalogue des textes de ce fichier.

La deuxième partie, intitulée ——GRAPHIQUES ——, permet :
5 : effacement de la page graphique.
6 : visualisation de la page graphique, le RETURN ramène au menu.
7 : sauve la page graphique sur le disque.
8 : lecture d'une page graphique du disque.
9 : exécute un texte.
0 : sortie du programme.

Modifications éventuelles

1. Tous les fichiers : textes, dessins, planches sont sur le disque du lecteur 2 ; si l'on préfère travailler sur le lecteur 1 il suffit de remplacer dans les déclarations VOL='#5:' par VOL='#4:'.

2. Si l'on désire des programmes plus longs, on pourra remplacer LONGPRO=80 par LONGPRO=X (X < 255).

3. Si l'on désire un fichier plus long, il suffit de changer la constante de NBLISTE.

Extensions envisageables

1. Ajouter un véritable éditeur de ligne avec corrections.

2. Ajouter d'autres fonctions ; pour ce faire : définir le caractère de la fonction et l'introduire à PCAR, écrire les lignes correspondantes dans les CASE de GETIND, LITINSTRUC et REALISE.

3. Écrire des procédures PASCAL spécifiques pour certains dessins difficiles à réaliser par ce programme, par exemple : arcs de cercle, polygones réguliers, etc., et placer les lignes correspondantes au menu.

4. Écrire en instruction immédiate sur la page graphique en acceptant des caractères à la place du RETURN. On pourra utiliser la même syntaxe, un déplacement point par point, ou les deux.

5. Utilisation de couleurs pour le fond ou les dessins.

6. Définition de fenêtres.

7. Rotation, agrandissement de dessins, déplacement de dessins dans la planche.

8. ... A vos claviers...

Attention !

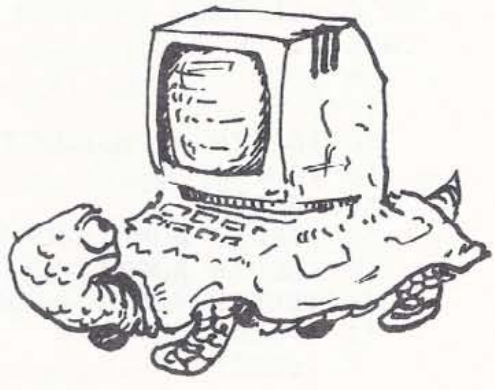
Afin de ne pas surcharger le programme, les contrôles d'erreurs ont été limités ; en particulier, il n'y a pas de contrôle de sortie de la page graphique, celui-ci peut être ajouté à la fonction A et à Y pour P. Lors de la rédaction de la chaîne texte, le contrôle de longueur est un peu frustre et intervient au caractère 77 avec retentissement de bips sonores ; ceci suffit sur les instructions courtes, par contre le nom d'une figure peut être coupé si l'on n'y prend garde.

```
PROGRAM TORTUE;

USES TURTLEGRAPHIC;

CONST NBLISTE       =15;
      TAILLE        =14;
      VOL           ='#5:';
      LONGPROG      =80;
```

```
TYPE  CHOIDECA      =SET OF CHAR;
      COMMANDE      =RECORD
                       NOM      :STRING[10];
                       INSTRUC  :STRING[LONGPROG]
                     END;
      MEMOIRECRAN   =PACKED ARRAY[0..8191] OF 0..255;
      FORME         =PACKED ARRAY[0..TAILLE,0..TAILLE] OF BOOLEAN;

VAR   HOME,BS,EFL,EFB,SON,INV,NORM,CR:CHAR;
      CA            :CHAR;
      COM           :COMMANDE;
      LISTE         :FILE OF ARRAY[1..NBLISTE] OF COMMANDE;
      INDEX,I,INC   :INTEGER;
      NCOM,NIMA     :STRING;
      PROJET        :STRING;
      ECRAN         :RECORD CASE BOOLEAN OF
                       FALSE :(ADRESSE:INTEGER);
                       TRUE  :(POINTEUR:^MEMOIRECRAN)
                     END;
      PHOTO         :FILE;
      ALBUM         :FILE OF FORME;

(*!*)
(* PROCEDURES GENERALES HABITUELLES *)
(*#*)
PROCEDURE PRINFO;
BEGIN
  HOME:=CHR(12);     EFL:=CHR(29);     BS  :=CHR(8);    CR :=CHR(13);
  SON :=CHR(7);      INV:=CHR(18);     NORM:=CHR(20);   EFB:=CHR(11)
END;

PROCEDURE PRENRETURN;
VAR SORT  :CHAR;
BEGIN
  REPEAT
    READ(KEYBOARD,SORT)
  UNTIL EOLN(KEYBOARD)
END;

FUNCTION PRENCAR(BONSET:CHOIDECA):CHAR;
VAR CH   :CHAR;
    BON  :BOOLEAN;
BEGIN
  REPEAT
    READ(KEYBOARD,CH);
    IF EOLN(KEYBOARD) THEN CH:=CR;
    BON:=CH IN BONSET;
    IF NOT BON THEN WRITE(SON)
      ELSE IF CH IN [' '..'^'] THEN WRITE(CH)
  UNTIL BON;
  PRENCAR:=CH
END;

FUNCTION OUI:BOOLEAN;
BEGIN
  OUI:=PRENCAR['O','N'] IN ['O']
END;
```

```
PROCEDURE MESSAGE(Y:INTEGER;S:STRING);
BEGIN
  GOTOXY(0,Y);WRITE(S,EFL)
END;

PROCEDURE ALARME(S:STRING);
BEGIN
  GOTOXY(0,1);WRITE(SON,EFB);
  MESSAGE(10,S);
  MESSAGE(12,'FAITES <RETURN> ');
  PRENRETURN
END;

PROCEDURE PRENCHAINE(LONGMAX:INTEGER;BONSET:CHOIDECA;VAR S:STRING);
VAR S1    :STRING[1];
    CONT  :STRING;
BEGIN
  S1:=' ';
  CONT:='';
  REPEAT
    IF LENGTH(CONT)=0 THEN S1[1]:=PRENCAR(BONSET+[CR])
      ELSE IF LENGTH(CONT)=LONGMAX THEN S1[1]:=PRENCAR([CR,BS])
             ELSE S1[1]:=PRENCAR(BONSET+[CR,BS]);
    IF S1[1] IN BONSET THEN CONT:=CONCAT(CONT,S1)
      ELSE IF S1[1]=BS THEN
        BEGIN
          WRITE(BS,' ',BS);
          DELETE(CONT,LENGTH(CONT),1)
        END;
  UNTIL S1[1]=CR;
  IF LENGTH(CONT)<>0 THEN S:=CONT
    ELSE WRITE(S)
END;

PROCEDURE ENTIER(LONGMAX:INTEGER;VAR S:INTEGER);
VAR S1    :STRING[1];
    I     :INTEGER;
    CONT  :STRING;
    OKSET :CHOIDECA;
BEGIN
  OKSET:=['0'..'9'];S1:=' ';CONT:='';
  REPEAT
    IF LENGTH(CONT)=0 THEN S1[1]:=PRENCAR(OKSET+[CR])
      ELSE IF LENGTH(CONT)=LONGMAX THEN S1[1]:=PRENCAR([CR,BS])
        ELSE S1[1]:=PRENCAR(OKSET+[CR,BS]);
    IF S1[1] IN OKSET THEN CONT:=CONCAT(CONT,S1)
      ELSE IF S1[1]=BS THEN
        BEGIN
          WRITE(BS,' ',BS);
          DELETE(CONT,LENGTH(CONT),1)
        END;
  UNTIL S1[1]=CR;
  IF LENGTH(CONT)<>0 THEN
   BEGIN
     S:=0;
     FOR I:=1 TO LENGTH(CONT) DO
     BEGIN
       S:=S*10;S:=S+(ORD(CONT[I])-ORD('0'))
     END;
```

```
   END ELSE WRITE(S)
END;
(*!*)
(* LECTURE, ECRITURE DE LA PAGE ECRAN SUR DISQUE *)
(*$*)
FUNCTION LIRIMAGE(IMAGE:STRING):BOOLEAN;
BEGIN
  (*$I-*)
  RESET(PHOTO,IMAGE);
  IF IORESULT<>0 THEN LIRIMAGE:=FALSE
    ELSE BEGIN
      LIRIMAGE:=BLOCKREAD(PHOTO,ECRAN.POINTEUR^,16)=16;
      CLOSE(PHOTO)
    END
  (*$I+*)
END;

FUNCTION ECRIMAGE(IMAGE:STRING):BOOLEAN;
BEGIN
  (*$I-*)
  REWRITE(PHOTO,IMAGE);
  IF IORESULT<>0 THEN ECRIMAGE:=FALSE
    ELSE BEGIN
      ECRIMAGE:=BLOCKWRITE(PHOTO,ECRAN.POINTEUR^,16)=16;
      CLOSE(PHOTO,LOCK)
    END
  (*$I+*)
END;

FUNCTION PRENIMA(VAR IMAGE:STRING):BOOLEAN;
BEGIN
  MESSAGE(22,'NOM DE LA PLANCHE :');
  PRENCHAINE(10,['A'..'Z'],IMAGE);
  IF IMAGE='' THEN PRENIMA:=FALSE
    ELSE BEGIN
      PRENIMA:=TRUE;
      IF POS(':',IMAGE)=0 THEN IMAGE:=CONCAT(VOL,IMAGE);
      IF POS('.',IMAGE)=0 THEN IMAGE:=CONCAT(IMAGE,'.IMA')
    END
END;
(*!*)
(* PROCEDURES DE GESTION DES FICHIERS TEXTES *)
(*$*)
PROCEDURE OPENLISTE(ST:STRING);
VAR I     :INTEGER;
    L     :COMMANDE;
BEGIN
  (*$I-*)
  CLOSE(LISTE);
  RESET(LISTE,ST);
  IF IORESULT<>0 THEN
  BEGIN
    CLOSE(LISTE);
    REWRITE(LISTE,ST);SEEK(LISTE,0);
    L.NOM:='';L.INSTRUC:='';
    FOR I:=1 TO NBLISTE DO LISTE^[I]:=L;
    PUT(LISTE);
    CLOSE(LISTE,LOCK);
    RESET(LISTE,ST)
```



```
  END;
  (*$I+*)
END;

FUNCTION GETIND(NM:STRING):INTEGER;
VAR I:INTEGER;
BEGIN
  I:=1;GETIND:=0;
  WHILE (NM<>LISTE^[I].NOM) AND (I<NBLISTE) DO I:=I+1;
  IF NM=LISTE^[I].NOM THEN GETIND:=I
END;

PROCEDURE GETINS(VAR INS:STRING);
VAR C1    :CHAR;

  PROCEDURE AJPM;
  BEGIN
    INS[INDEX]:=PRENCAR(['+','-']);INDEX:=INDEX+1
  END;

  PROCEDURE AJBIT;
  VAR BIT   :INTEGER;
  BEGIN
    REPEAT
      BIT:=0;
      ENTIER(3,BIT);
      IF BIT>255 THEN WRITE(BS,BS,BS,'   ',BS,BS,BS);
    UNTIL BIT<255;
    INS[INDEX]:=CHR(BIT);INDEX:=INDEX+1
  END;

  PROCEDURE AJCHAINE(NB:INTEGER);
  VAR S     :STRING;
  BEGIN
    S:='';
    PRENCHAINE(NB,['A'..'Z'],S);
    IF INDEX+LENGTH(S)<LONGPROG-3 THEN
    BEGIN
      MOVELEFT(S[1],INS[INDEX],LENGTH(S));
      INDEX:=INDEX+LENGTH(S)
    END;
  END;

BEGIN
  REPEAT
    IF INDEX>75 THEN WRITE(SON,SON,SON);
    C1:=PRENCAR['A','T','D','I','Z','S','P','C','R','>',':',CR]);
    IF C1=CR THEN WRITELN ELSE BEGIN INS[INDEX]:=C1;INDEX:=INDEX+1 END;
    CASE C1 OF
      'A',
      'T',
      'C':BEGIN AJPM;AJBIT END;
      'D':AJPM;
      'I':AJBIT;
      'S':BEGIN AJPM;AJCHAINE(20) END;
      'P':BEGIN AJBIT;WRITE('/');AJBIT END;
      '>':BEGIN AJBIT;AJCHAINE(10) END;
      ':':AJCHAINE(10);
      'R':BEGIN
```

```
            AJBIT;WRITE('(');INS[INDEX]:='(';INDEX:=INDEX+1;
            GETINS(INS);
            WRITE(')');INS[INDEX]:=')';INDEX:=INDEX+1
          END;
    END;
    IF C1<>CR THEN C1:=PRENCAR(['',',','.']);
    INS[INDEX]:=C1;INDEX:=INDEX+1;
  UNTIL (C1='.') OR (INDEX>LONGPROG-3);
  IF INDEX>LONGPROG-3 THEN INS[INDEX-1]:='.';
END;

PROCEDURE EDITE;
VAR  NOMFI    :STRING[10];
     INDICE,I:INTEGER;
     L        :STRING[80];
BEGIN
  PAGE(OUTPUT);
  I:=1;
  REPEAT
    IF LISTE^[I].NOM='' THEN INDICE:=I;
    I:=I+1;
  UNTIL (INDICE<>0) OR (I=NBLISTE+1);
  IF INDICE=0 THEN BEGIN ALARME('FICHIER PLEIN...');EXIT(EDITE) END;
  MESSAGE(0,'EDITION:FIGURE ');WRITE(INDICE,' DE ',NCOM);
  MESSAGE(7,'NOM DE LA FIGURE:');
  PRENCHAINE(10,['A'..'Z'],LISTE^[INDICE].NOM);
  INDEX:=1;
  (*$R-*)
  L[0]:=CHR(80);
  FILLCHAR(L[1],80,' ');
  (*$R+*)
  GOTOXY(0,10);
  GETINS(L);
  LISTE^[INDICE].INSTRUC:=L;
  SEEK(LISTE,0);PUT(LISTE);CLOSE(LISTE,LOCK);RESET(LISTE,NCOM)
END;

PROCEDURE LITINSTRUC(S:STRING);
VAR C1 :CHAR;
BEGIN
  REPEAT
    C1:=S[INDEX];WRITE(C1);INDEX:=INDEX+1;
    CASE C1 OF
      'A','T',
      'C'     :BEGIN
                 WRITE(S[INDEX]);INDEX:=INDEX+1;
                 WRITE(ORD(S[INDEX]));INDEX:=INDEX+1
               END;
      'D'     :BEGIN WRITE(S[INDEX]);INDEX:=INDEX+1 END;
      'I'     :BEGIN WRITE(ORD(S[INDEX]));INDEX:=INDEX+1 END;
      'P'     :BEGIN
                 WRITE(ORD(S[INDEX]));INDEX:=INDEX+1;
                 WRITE('/',ORD(S[INDEX]));INDEX:=INDEX+1
               END;
      'S',':':REPEAT
                 C1:=S[INDEX];WRITE(C1);INDEX:=INDEX+1
               UNTIL (S[INDEX] IN ['',',','.']);
      '>'     :BEGIN
                 WRITE(ORD(S[INDEX]));INDEX:=INDEX+1;
```

```
                REPEAT
                  C1:=S[INDEX];WRITE(C1);INDEX:=INDEX+1
                UNTIL (S[INDEX] IN ['',',','.']);
              END;
      'R'     :BEGIN
                WRITE(ORD(S[INDEX]));INDEX:=INDEX+1;
                WRITE(S[INDEX]);INDEX:=INDEX+1;
                LITINSTRUC(S);
                WRITE(S[INDEX]);INDEX:=INDEX+1
              END;
    END
  UNTIL C1='.';
END;

PROCEDURE LIRE;
VAR  INDICE:INTEGER;
     NM     :STRING;
BEGIN
  PAGE(OUTPUT);MESSAGE(0,'LECTURE:');WRITE(NCOM);
  MESSAGE(7,'NOM DE LA FIGURE:');
  PRENCHAINE(10,['A'..'Z'],NM);
  INDICE:=GETIND(NM);
  GOTOXY(0,10);INDEX:=1;
  IF INDICE<>0 THEN LITINSTRUC(LISTE^[INDICE].INSTRUC)
    ELSE MESSAGE(10,'N''EXISTE PAS');
  MESSAGE(23,'TAPEZ LE <RETURN> ');PRENRETURN;
END;

PROCEDURE CATALOG;
VAR I:INTEGER;
    C:CHAR;
BEGIN
  PAGE(OUTPUT);MESSAGE(0,'CATALOGUE:');WRITE(NCOM);
  GOTOXY(0,4);
  FOR I:=1 TO NBLISTE DO WRITELN(I:2,' -',LISTE^[I].NOM);
  MESSAGE(23,'TAPEZ E(FFACE OU  <RETURN> ');
  C:=PRENCAR(['E',CR]);
  IF C='E' THEN
  BEGIN
    MESSAGE(23,'NUMERO: ');I:=0;ENTIER(2,I);
    IF (I<=NBLISTE) THEN
    BEGIN
      LISTE^[I].NOM:='';LISTE^[I].INSTRUC:=''
    END
  END
END;

PROCEDURE PRENLISTE;
BEGIN
  PAGE(OUTPUT);
  MESSAGE(5,'NOM DU FICHIER:');
  PRENCHAINE(10,['A'..'Z'],NCOM);
  IF NCOM='' THEN EXIT(PRENLISTE);
  IF POS(':',NCOM)=0 THEN NCOM:=CONCAT(VOL,NCOM);
  IF POS('.',NCOM)=0 THEN NCOM:=CONCAT(NCOM,'.COM');
  OPENLISTE(NCOM);
END;
```

```
(* EXECUTION D'UN TEXTE SUR LA PAGE GRAPHIQUE *)
(*$*)
PROCEDURE EXECUTION(PJET:STRING);
VAR INDICE,IND :INTEGER;

  PROCEDURE REALISE(S:STRING);
  VAR C1,SI    :CHAR;
      IM       :STRING;
      I,REF,
      LIM,CT,N:INTEGER;
  BEGIN
    REPEAT
      C1:=S[IND];IND:=IND+1;
      CASE C1 OF
        'A':CASE S[IND] OF
              '+':BEGIN
                    IND:=IND+1;
                    MOVE(ORD(S[IND])*INC);IND:=IND+1
                  END;
              '-':BEGIN
                    IND:=IND+1;
                    TURN(180);
                    MOVE(ORD(S[IND])*INC);IND:=IND+1;
                    TURN(-180)
                  END
            END;
        'T':BEGIN
              SI:=S[IND];IND:=IND+1;
              IF SI='+' THEN TURN(ORD(S[IND])) ELSE TURN(-ORD(S[IND]));
              IND:=IND+1
            END;
        'I':BEGIN INC:=ORD(S[IND]);IND:=IND+1 END;
        'C':BEGIN
              SI:=S[IND];IND:=IND+1;
              IF SI='+' THEN CHARTYPE(10) ELSE CHARTYPE(5);
              WCHAR(S[IND]);IND:=IND+1
            END;
```

.../...

Trucs et astuces

Quelquefois, votre disquette se centre mal et vous obtenez une fatidique I/0 ERROR. Beaucoup d'entre elles pourront être évitées si, lors du premier accès à une disquette, vous ne fermez la porte du lecteur qu'une fois la lampe témoin allumée.

En effet, la disquette se centre mieux dans ce cas-là.

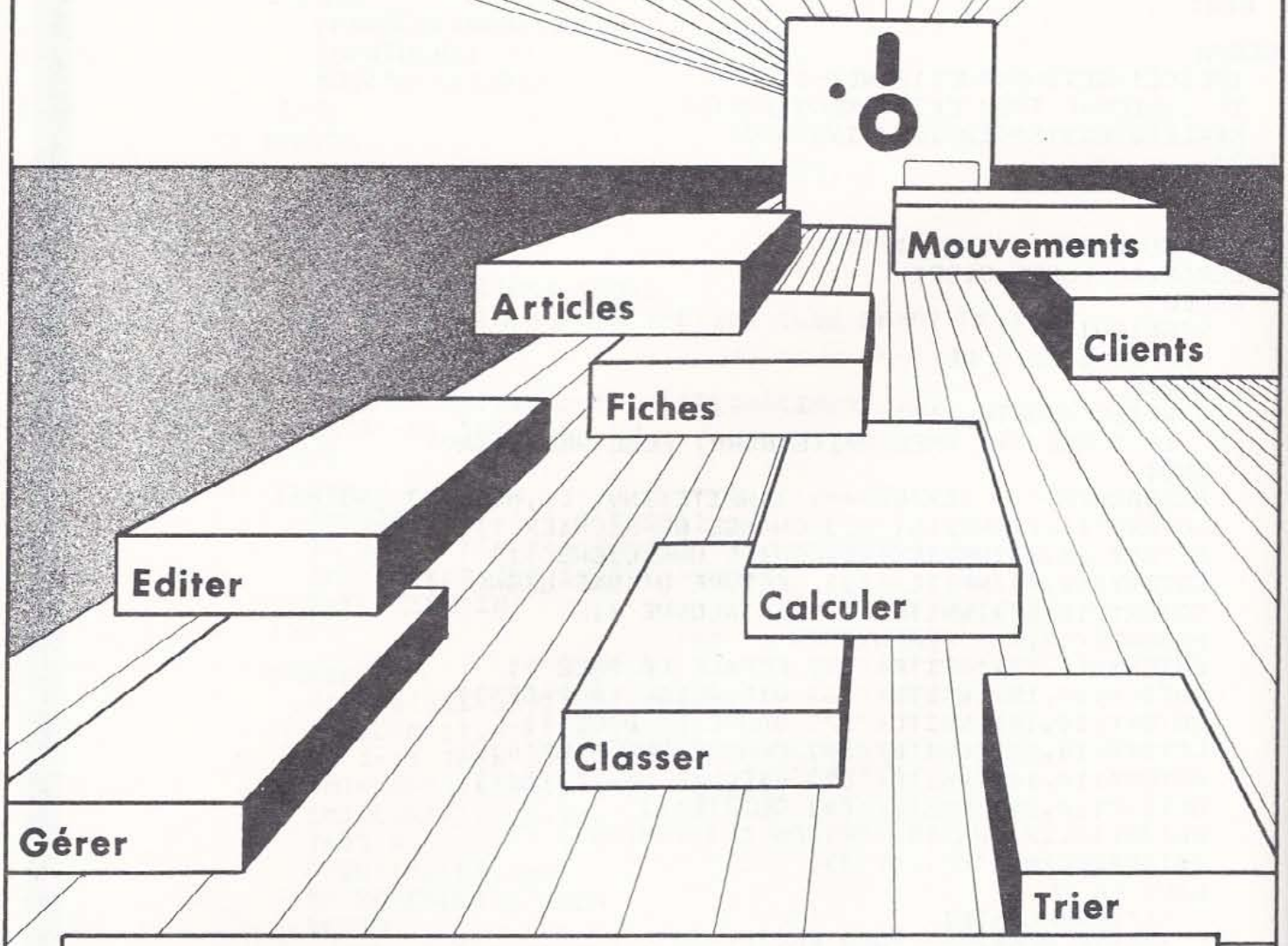
Enfin, si vous voulez éviter les ennuis, n'oubliez quand même pas que la disquette est un support fragile, et que la tête de lecture est appelée à travailler *très près* de la surface.

Un de nos lecteurs a ainsi pu constater que, quand de la confiture tombe sur une disquette, ce n'est bon (pas la confiture, bien sûr) ni pour la disquette (on s'y attendait) ni pour le lecteur (ça, c'était moins évident). Il a fallu déconfiturer le lecteur en atelier.

```
'D':BEGIN
      SI:=S[IND];IND:=IND+1;
      IF SI='+' THEN PENCOLOR(WHITE) ELSE PENCOLOR(NONE);
    END;
'Z':BEGIN
      PENCOLOR(NONE);
      TURNTO(0);
      MOVETO(140,96)
    END;
'P':BEGIN
      MOVETO(ORD(S[IND]),ORD(S[IND+1]));
      IND:=IND+2
    END;
'S':BEGIN
      SI:=S[IND];IND:=IND+1;
      IF SI='+' THEN CHARTYPE(10) ELSE CHARTYPE(5);
      C1:=S[IND];
      REPEAT
        WCHAR(C1);IND:=IND+1;C1:=S[IND]
      UNTIL C1 IN[',','.']
    END;
':':BEGIN
      I:=SCAN(11,=',',S[IND]);
      IF I=11 THEN I:=SCAN(11,='.',S[IND]);
      IM:=COPY(S,IND,I);IND:=IND+I;
      EXECUTION(IM)
    END;
'>':BEGIN
      N:=ORD(S[IND]);IND:=IND+1;
      I:=SCAN(11,=',',S[IND]);
      IF I=11 THEN I:=SCAN(11,='.',S[IND]);
      IM:=COPY(S,IND,I);IND:=IND+I;
      IM:=CONCAT(VOL,IM,'.PIC');
      (*$I-*)
      RESET(ALBUM,IM);
      IF IORESULT=0 THEN
      BEGIN
        SEEK(ALBUM,N);GET(ALBUM);
        DRAWBLOCK(ALBUM^,2,0,0,15,15,TURTLEX,TURTLEY,10)
      END;
      CLOSE(ALBUM)
      (*$I+*);
    END;
'R':BEGIN
      REP:=ORD(S[IND]);IND:=IND+2;
      I:=IND;
      REPEAT
        IF S[I]=')' THEN LIM:=I;
        I:=I+1
      UNTIL (S[I]=' ') OR (I=80);
      IM:=COPY(S,IND,LIM-IND);CT:=IND;
      FOR I:=1 TO REP DO
        BEGIN
          IND:=1;
          IF INC>1 THEN INC:=INC+1;
          REALISE(IM)
        END;
```

```
              IND:=LIM
            END;
      END
    UNTIL C1='.'
  END;

BEGIN
  INDICE:=GETIND(PJET);IND:=1;
  IF INDICE=0 THEN EXIT(EXECUTION);;
  REALISE(LISTE^[INDICE].INSTRUC)
END;

BEGIN
  PRINFO;NCOM:='';NIMA:='';
  ECRAN.ADRESSE:=8132;
  REPEAT
    PAGE(OUTPUT);
    FOR I:=1 TO 6 DO
    BEGIN
      WRITE('TORTUE ');
      IF I MOD 2=0 THEN WRITE(NORM) ELSE WRITE(INV)
    END;
    MESSAGE(5,'-- TEXTES ==>');WRITE(INV,'[',NCOM,']',NORM);
    GOTOXY(10,7);WRITE('[1] CHANGE DE FICHIER');
    GOTOXY(10,8);WRITE('[2] EDITE UNE LIGNE');
    GOTOXY(10,9);WRITE('[3] LECTURE D''UNE LIGNE');
    GOTOXY(10,10);WRITE('[4] CATALOGUE');
    MESSAGE(12,'-- GRAPHIQUES --');
    GOTOXY(10,14);WRITE('[5] EFFACE LA PAGE');
    GOTOXY(10,15);WRITE('[6] VISUALISE LA PAGE');
    GOTOXY(10,16);WRITE('[7] SAUVE LA PAGE');
    GOTOXY(10,17);WRITE('[8] CHARGE UNE PAGE');
    GOTOXY(10,18);WRITE('[9] EXECUTE UN TEXTE');
    GOTOXY(10,20);WRITE('[0] QUITTE');
    GOTOXY(3,22);WRITE('OPTION [ ]',BS,BS);
    CA:=PRENCAR['0'..'9'];
    CASE CA OF
      '1':PRENLISTE;
      '2':IF NCOM<>'' THEN EDITE;
      '3':IF NCOM<>'' THEN LIRE;
      '4':IF NCOM<>'' THEN CATALOG;
      '5':BEGIN INITTURTLE;TEXTMODE END;
      '6':BEGIN GRAFMODE;PRENRETURN;TEXTMODE END;
      '7':IF PRENIMA(NIMA) THEN
            IF NOT ECRIMAGE(NIMA) THEN ALARME ('PAS D''ENREGISTREMENT');
      '8':IF PRENIMA(NIMA) THEN
            IF NOT LIRIMAGE(NIMA) THEN ALARME('PAS DE LECTURE');
      '9':BEGIN
            MESSAGE(22,'NOM DU TEXTE :');
            PRENCHAINE(10,['A'..'Z'],PROJET);
            IF PROJET<>'' THEN
            BEGIN
               MESSAGE(22,'DESSIN ');WRITE(INV,PROJET,NORM,' EN COURS');
               INC:=1;EXECUTION(PROJET)
            END
          END;
    END
  UNTIL CA='0'
END.
```

CX Multigestion



LA GESTION DE FICHIER TOUS AZIMUTS

Un seul programme, un très grand nombre d'applications

Avec ce logiciel, votre APPLE disposera de pouvoirs exceptionnels; vous vous en servirez pour suivre vos clients, pour établir vos prévisions, pour mettre à jour vos tarifs, pour gérer vos commandes, vos articles en stocks, pour calculer la paie de vos employés ou simplement pour tenir votre collection de timbres ou de livres.

Une très grande facilité de mise en œuvre

Nul besoin de connaître l'informatique, nul besoin d'utiliser un vocabulaire de spécialiste; pas de codifications inutiles; les instructions sont simples, naturelles et en français.

Définissez vous-même votre modèle de fichier

Chaque rubrique devient automatiquement un critère de recherche et de classement; vous pourrez à tout moment redéfinir votre modèle, ajouter ou retrancher des rubriques sans avoir à réécrire les données.

Retrouver vos dossiers de multiples façons

Vous pouvez les rechercher à partir de n'importe quelle rubrique, retrouver tous ceux répondant à certaines caractéristiques selon une combinaison de critères (jusqu'à 12 simultanément) tels que "égal", "plus grand que", "plus petit que", "différent", "compris entre".

Faites toutes sortes de calculs et profitez de votre imprimante

Vous pourrez présenter les informations de votre choix et les résultats de vos calculs sous forme d'étiquettes ou de rapports (jusqu'à 13 colonnes), dans n'importe quel ordre alphabétique, numérique, chronologique.

Avec CX Multigestion, vous travaillerez très vite et vous irez très loin

Comparez avec d'autres programmes: vous retrouvez n'importe quel dossier, vous dressez un état combinant classement et sélections en quelques secondes; en outre, CX Multigestion n'est que le premier programme d'une série de logiciels de gestion tous compatibles entre eux.

Demandez à votre boutique informatique une démonstration de CX Multigestion et vous serez vraiment émerveillé; si elle ne l'a pas en stock, demandez lui de nous appeler au (1) 538.98.87 ou écrivez nous à: CONTROLE X - Tour Maine-Montparnasse, 33, av. du Maine - 75755 Paris Cedex 15

Apple est une marque déposée de Apple Computer Inc.

Notions de base : routine d'INPUT généralisé

Gérard Michel

Il y a fort à parier que beaucoup des programmes que vous écrivez, ou envisagez d'écrire, comportent, à un moment quelconque, une phase d'entrée de données au clavier. Certes, une simple suite d'instructions INPUT peut réaliser ce traitement, mais ce n'est ni la solution la plus agréable à l'œil (présentation de l'écran au moment de la saisie), ni la plus sûre à l'usage (contrôle des données entrées).

Même si vous êtes le seul utilisateur de vos programmes, il peut vous être utile de disposer d'une routine « d'INPUT généralisé » qui assure une gestion correcte de l'écran et vérifie que les données entrées correspondent bien à ce qu'elles devraient être.

Principes généraux

Pour faciliter l'analyse de la routine présentée ici, précisons les objectifs que nous voulons atteindre.

1. L'écran doit indiquer clairement les informations à fournir et assigner à chacune d'elles une zone de rentrée individualisée.

2. Il doit être possible de se déplacer sur l'écran, vers le haut comme vers le bas, pour modifier les différentes données avant toute validation.

3. Le contrôle des données doit s'effectuer caractère par caractère au moment de la frappe (pas de lettres dans une information numérique, pas de caractères de contrôle ou susceptibles de perturber les traitements ultérieurs...). De plus, la longueur des informations ne doit pas dépasser le maximum compatible avec les autres traitements.

4. Après validation, les données doivent pouvoir être modifiées ou consultées facilement.

Notre routine repose en conséquence sur les principes suivants (se reporter au programme BASIC pour identifier les variables) :

1. Chaque information est définie par un libellé (LI$), la position du premier caractère à rentrer (ZV% et ZH%), et la longueur maximale de la réponse (ZL%).

2. La touche RETURN permet de terminer l'entrée d'une information et de passer à la suivante. La touche ESC permet de remonter aux données précédentes ou de revenir à la phase de traitement précédente.

3. On distingue trois types de données : alphanumérique (type 1), numérique (type 2) et date (type 3). Chaque information est caractérisée par son type (TY%).

4. Des variables annexes permettent de réaliser l'affichage des données pour

AGRAPH

A VOS MARQUES, PRET, COMPTEZ.

Un ensemble de programmes portant sur la comptabilité, dédié aux petites entreprises, professions libérales, artisans, commerçants, bref aux amateurs d'informatique soucieux de la gestion de leurs affaires professionnelles ou privées.

C'est un outil complet et pratique à utiliser qui, après un rappel sur la comptabilité générale donne des exemples concrets : édition des livres-journal, grands livres, balances, bilans, calculs des ratios ainsi qu'un programme spécial intéressant l'adaptation et la personnalisation du Plan Comptable.

160 pages - 102,00 FF/785,00 FB

P.S.I.

P.S.I. DIFFUSION
41-51, rue Jacquard
BP 86 - 77400 Lagny-s/Marne
FRANCE
Téléphone (6) 007.59.31
P.S.I. BENELUX
5, avenue de la Ferme Rose
1180 Bruxelles
BELGIQUE
Téléphone (2) 345.08.50

En Espagne
P.S.I. Iberica
Ferraz 11
Madrid 8
Tél. : 247.30.00

au Canada
SCE Inc
3449 rue Saint-Denis
Montréal Québec H2X3L1
Tél (514) 843.76.63

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

DESIGNATION	NOMBRE	PRIX
(par avion : ajouter 8 FF (75 FB) par livre)	TOTAL	

NOM _____ PRENOM _____

rue _____ N° _____

Code post. |_|_|_|_|_| Ville _____

P.CA.12

modification ou consultation (ME, X, B%).

Analyse et utilisation de la routine

Le contrôle des données entrées, caractère par caractère, constitue le cœur de la routine. Il est assuré par un petit programme en assembleur, ce qui permet de :

— faire l'économie de nombreux tests en BASIC sur les codes ASCII des caractères, tout en accélérant le traitement ;

— bénéficier facilement de la lecture « active » par la flèche de droite et de garantir que les données stockées dans les variables sont identiques à celles qui apparaissent sur l'écran ;

— illustrer par un exemple simple l'utilisation de l'assembleur et son interaction avec BASIC, ce qui répond aux vœux de nombreux lecteurs concernés par cette rubrique « Notions de Base ».

Opérons donc à cœur ouvert :

Dans les mémoires $28 et $29 est stockée l'adresse de la première colonne de la ligne courante de l'écran (celle où se trouve le curseur) dans la page TEXT (zone de mémoire où sont rangés les caractères qui composent l'écran en mode TEXT). L'adresse $28 est ici repérée par le symbole ADR.

En $24 est stockée la position horizontale du curseur (0 à 39), symbolisée par H.

En $9, nous stockerons le nombre de caractères entrés (LC).

$6 (E) servira de « drapeau » au BASIC pour une éventuelle remontée dans l'écran (touche ESC).

En $7 (LO), nous rangerons la longueur maximale autorisée pour la réponse.

En $8 (TY), stockage du type de l'information.

$311 (ZY) est l'adresse de début de la zone où l'on stockera les caractères entrés au clavier. Cette zone se trouve dans une « région » libre de l'Apple ($300 à $3CF). On aurait pu stocker à partir de $300, ou après la routine elle-même, par exemple.

$FD0C est l'adresse de début du sous-programme du moniteur qui fait clignoter le curseur et attend qu'une touche soit frappée. Après exécution, le code-écran du caractère entré se trouve dans l'accumulateur (vous pouvez vous reporter aux pages 44-45 de Pom's numéro 4 ou à la page 7 du « Manuel de Référence Apple II » pour avoir la liste des codes-écran des différents caractères.

$FDED est l'adresse de début du sous-programme du moniteur qui affiche à l'écran le caractère dont le code-écran se trouve dans l'accumulateur.

Lignes 12 à 16 : nettoyage de la zone de stockage ($80 est le code-écran du caractère nul). En BASIC, ces instructions s'écriraient :

```
12 A$=""
13 X=LO
14 ZY$(X)=A$
15 X=X-1
16 IF X =0 THEN 14
```

Lignes 17 et 18 : E=0.

Ligne 19 : saisie d'un caractère. Rappelons que le résultat se trouve dans l'accumulateur, d'où l'emploi de l'instruction CMP dans les tests sur la valeur du caractère.

Lignes 20 et 21 : si c'est un RETURN, on saute à la fin de routine (FIN1).

Lignes 22 et 23 : X sert à compter le nombre de caractères entrés (à partir de 0). Si X = LO, on saute à SUIT5.

Lignes 24 et 25 : si le caractère est une « flèche de gauche », on saute à RETOUR.

Lignes 26 et 27 : on a entré un caractère de trop. La routine-moniteur dont l'adresse de début est $FBDD émet un « beep » et l'on retourne ensuite à la saisie de caractère. Ainsi, une fois la

Liste en assembleur Lisa 1.5

```
 1            ORG  $91E6
 2            OBJ  $800
 3  ADR       EPZ  $28
 4  H         EPZ  $24
 5  LC        EPZ  $9
 6  E         EPZ  $6
 7  LO        EPZ  $7
 8  TY        EPZ  $8
 9  ZY        EQU  $311
10  GET       EQU  $FD0C
11  OUT       EQU  $FDED
12            LDA  #$80
13            LDX  LO
14  DEP0      STA  ZY,X
15            DEX
16            BPL  DEP0
17            LDX  #0
18            STX  E
19  DEP1      JSR  GET
20            CMP  #$8D
21            BEQ  FIN1
22            CPX  LO
23            BCC  SUIT5
24            CMP  #$88
25            BEQ  RETOUR
26            JSR  $FBDD
27            JMP  DEP1
28  SUIT5     CMP  #$9B
29            BNE  SUIT1
30            CPX  #0
31            BNE  SUIT1
32            LDA  #9
33            STA  E
34            RTS
35  SUIT1     CMP  #$AC
36            BNE  SUIT2
37            LDA  #$AE
38            JMP  SUIT8
39  SUIT2     CMP  #$AE
40            BEQ  SUIT8
41            CMP  #$88
42            BNE  SUIT3
43            CPX  #0
44            BEQ  SUIT3
45  RETOUR    DEX
46            DEC  H
47            JMP  DEP1
48  SUIT3     CMP  #$95
49            BNE  SUIT4
50            LDA  $25
51            JSR  $FBC1
52            LDY  H
53            LDA  (ADR),Y
54            JMP  SUIT8
55  SUIT4     CMP  #$A0
56            BCC  DEP1
57            CMP  #$DB
58            BCS  DEP1
59            LDY  TY
60            CPY  #1
61            BEQ  SUIT6
62            CMP  #$AF
63            BCS  SUIT7
64            CMP  #$AD
65            BNE  DEP1
66  SUIT7     CMP  #$BA
67            BCS  DEP1
68  SUIT6     CPX  #0
69            BNE  SUIT8
70            CMP  #$A2
71            BEQ  DEP1
72  SUIT8     JSR  OUT
73            STA  ZY,X
74            INX
75            JMP  DEP1
76  FIN1      STX  LC
77            RTS
```

longueur maximale atteinte, seuls RETURN et la flèche de gauche sont acceptés.

Lignes 28 et 29 : si ce n'est pas ESC, on saute à SUIT1.

Lignes 30 et 31 : si ESC n'est pas le premier caractère saisi dans la zone, on saute à SUIT1 (cet ESC sera ensuite refusé).

Lignes 32 à 34 : on positionne le « drapeau de remontée » (E=9) et on sort de la routine.

Lignes 35 à 38 : si ce n'est pas une virgule, on saute à SUIT2. Sinon, on la remplace par un point ($AE) et on saute à SUIT8. En effet, la virgule peut poser des problèmes si les informations sont ensuite stockées puis relues par INPUT sur disquette (EXTRA IGNORED).

De plus, pour les informations numériques, il faut traduire la « virgule française » en « point anglo-saxon »...

Lignes 39 et 40 : si c'est un point, on saute à SUIT8.

Lignes 41 et 42 : si ce n'est pas une « flèche de gauche », on saute à SUIT3. Ce test n'est pas redondant avec celui de la ligne 24, puisque nous sommes maintenant dans le cas où la longueur maximale n'est pas atteinte.

Lignes 43 et 44 : si la flèche de gauche est le premier caractère entré dans la zone (X=0), on saute à SUIT3 (elle sera ensuite refusée par la routine).

Lignes 45 à 47 : si la flèche n'est pas le premier caractère, on diminue de 1 le nombre de caractères entrés (DEX), on recule le curseur (DEC H) et on retourne à la saisie de caractère (JMP DEP1).

Lignes 48 et 49 : si ce n'est pas une « flèche de droite », on saute à SUIT4.

Lignes 50 à 54 : si par contre c'est une flèche de droite, on place la position verticale du curseur (stockée en $25 par le système) dans l'accumulateur. La routine qui débute en $FBC1 va utiliser cette valeur pour calculer l'adresse de la première colonne de la ligne courante qu'elle range ensuite en $28 et $29.

On transfère alors la position horizontale du curseur (H) dans Y et on va lire dans la page TEXT le caractère qui se trouve à la position H dans la ligne V dont le stockage dans cette page commence à l'adresse contenue dans ADR, c'est-à-dire le caractère sur lequel « passe » la flèche à l'écran. On saute ensuite à SUIT8 (affichage).

Lignes 55 et 56 : si le caractère est inférieur à « SPACE » (caractère de contrôle), on le refuse.

Lignes 57 et 58 : si le caractère est supérieur à Z, on le refuse.

Ligne 59 : la suite de l'analyse dépend alors du type de donnée, que l'on charge dans le registre Y.

Lignes 60 et 61 : si c'est une donnée alphanumérique, on passe à SUIT6.

Lignes 62 et 63 : si le caractère est supérieur ou égal à "/", on passe à SUIT7.

Lignes 64 et 65 : étant donc inférieur à "/", on le refuse s'il est différent de "_".

Lignes 66 et 67 : s'il est supérieur ou égal à ":", on le refuse.

Lignes 68 à 71 : on refuse les guillemets s'ils sont en premier caractère (problèmes ultérieurs possibles avec les ordres PRINT).

Ligne 72 : le caractère étant maintenant accepté, on l'affiche à l'écran.

Ligne 73 : on le stocke dans la zone réservée à cet usage, dans la « case » numéro X.

Lignes 74 et 75 : on passe au caractère suivant.

Lignes 76 et 77 : on stocke le nombre de caractères entrés ; fin de routine pour retour au BASIC.

L'interaction entre ce programme en langage-machine et le programme BASIC est assurée dans le sous-programme BASIC des lignes 149 à 169. Celui ci est appelé pour chacune des informations à fournir. Nous n'en commenterons que les points qui intéressent directement notre sujet.

Ligne 149 : ME=2 ou X=1 signalent que l'on veut un affichage du contenu de la variable. B%=8 assure une sortie de la routine directement après cet affichage.

Ligne 150 : on transmet (instructions POKE) à la routine-machine les paramètres de l'information à entrer, puis on lance son exécution (CALL 37350). A noter que l'on place à l'adresse 36 ($24) la valeur H-1 et non H, car BASIC fait ses HTAB de 1 à 40, alors que le moniteur calcule de 0 à 39.

Ligne 154 : nous sommes déjà de retour au BASIC ! Les résultats de la saisie de caractères se trouvent aux différentes adresses définies dans les lignes 4 à 9 de la routine-machine. L'instruction PEEK nous permet de les récupérer. E=9 signale que la touche de fonction ESC a été utilisée et que l'opérateur désire « remonter » sur l'écran.

Ligne 156 : LC=0 si seul RETURN a été entré. En conséquence, on quittera le sous-programme en laissant le contenu de la variable inchangé.

Ligne 158 : BASIC récupère maintenant les différents caractères stockés à partir de l'adresse 785 ($311) et reconstitue la donnée dans son ensem-

Récapitulation du code binaire

```
]CALL-151

*91E6.926F

91E6- A9 80
91E8- A6 07 9D 11 03 CA 10 FA
91F0- A2 00 86 06 20 0C FD C9
91F8- 8D F0 72 E4 07 90 0A C9
9200- 88 F0 28 20 DD FB 4C F4
9208- 91 C9 9B D0 09 E0 00 D0
9210- 05 A9 09 85 06 60 C9 AC
9218- D0 05 A9 AE 4C 63 92 C9
9220- AE F0 40 C9 88 D0 0A E0
9228- 00 F0 06 CA C6 24 4C F4
9230- 91 C9 95 D0 0C A5 25 20
9238- C1 FB A4 24 B1 28 4C 63
9240- 92 C9 A0 90 AF C9 DB B0
9248- AB A4 08 C0 01 F0 0C C9
9250- AF B0 04 C9 AD D0 9D C9
9258- BA B0 99 E0 00 D0 04 C9
9260- A2 F0 91 20 ED FD 9D 11
9268- 03 E8 4C F4 91 86 09 60
```

ble (ZZ$). A noter que l'on enlève 128 aux codes lus en mémoire pour retrouver les codes ASCII standards du BASIC (ceux des fonctions ASC et CHR$ - Cf. pages 138 et 139 du Manuel de Référence Applesoft).

Lignes 163 à 167 : on vérifie que la date correspond bien au format voulu, à savoir JJ/MM/AA.

Le reste du programme BASIC constitue un exemple d'utilisation de l'ensemble « routine-assembleur » plus « routine BASIC ». Nous vous laissons le soin d'en conduire vous-même l'analyse et d'en déduire la façon d'intégrer cet « INPUT généralisé » dans vos propres programmes.

En guise de conclusion, nous lançons un appel au peuple ! Cette rubrique « Notions de Base » est destinée aux lecteurs qui partent à la découverte de leur Apple, et bien souvent de l'informatique elle-même. Afin que nous puissions l'orienter dans un sens favorable au plus grand nombre, n'hésitez pas à nous écrire pour nous signaler les problèmes que vous voudriez voir traités.

Selon la nature de votre courrier, nous répondrons par des articles du même type que celui que vous venez de lire, ou par une série de courtes explications sur certains points très précis. Au plaisir de vous lire...

Liste du programme BASIC

```
]LIST

1  HIMEM: 37340
5 D$ =  CHR$ (4): PRINT D$"BLOAD
    INPUT.OBJ": GOTO 400
70  VTAB 21: HTAB 1: INVERSE : PRINT
    ZM$;: NORMAL : INPUT " ? ";Z
    $: VTAB 21: CALL  - 868:Z$ =
     LEFT$ (Z$,1): IF Z$ = "O" OR
    Z$ = "N" THEN  RETURN
71  GOTO 70
80  VTAB V: HTAB 1: CALL  - 868
81  INVERSE : PRINT Z$;: NORMAL :
     RETURN
90  TEXT : HOME : HTAB 20 -  LEN
    (Z$) / 2: INVERSE : PRINT Z$
    : NORMAL : RETURN
149  VTAB V: HTAB H: PRINT  LEFT$
    (PO$,LO)"  ";: VTAB V: HTAB
    H: IF ME = 2 OR X = 1 THEN  PRINT
    ZY$(LL): IF B% = 8 THEN 168
150  VTAB V: POKE 8,TY: POKE 36,H
     - 1: POKE 7,LO: CALL 37350
154 LC =  PEEK (9):E =  PEEK (6):
     IF E = 9 THEN  RETURN
156 ZZ$ = "": CALL  - 868: IF LC =
    0 THEN 168
158  FOR ZZ = 1 TO LC:Z =  PEEK (
    784 + ZZ) - 128:ZZ$ = ZZ$ +
     CHR$ (Z): NEXT
162  IF TY < 3 THEN  RETURN
163  IF  MID$ (ZZ$,2,1) = "/" THEN
    77$ = "0" + ZZ$
164  IF  MID$ (ZZ$,5,1) = "/" THEN
    ZZ$ =  LEFT$ (ZZ$,3) + "0" +
     RIGHT$ (ZZ$,4)
165  IF  LEN (ZZ$) <  > 8 THEN Z$
     = ME$(3): GOSUB 190: GOTO 1
    49
166 Z4 =  VAL ( MID$ (ZZ$,4,2)): IF
     VAL ( LEFT$ (ZZ$,2)) > 31 OR
    Z4 > 12 OR Z4 *  VAL ( RIGHT$
    (ZZ$,2)) <  = 0 THEN Z$ = ME
    $(3): GOSUB 190: GOTO 149
167  RETURN
168  IF ME = 2 OR X = 1 THEN  VTAB
    V: HTAB H:ZZ$ = ZY$(LL): PRINT
    ZZ$ SPC( LO -  LEN (ZZ$))
169  RETURN
190  VTAB Z1: HTAB 1: CALL  - 868
    : INVERSE : FOR Z = 1 TO 150
    :Z1 =  PEEK ( - 16336): NEXT
    : PRINT Z$;: NORMAL : FOR Z =
    1 TO 2500: NEXT : HTAB 1: CALL
     - 868: RETURN
400  DATA  DONNEE ALPHA 1,DONNEE
    ALPHA 2,DONNEE NUM 1,DONNEE
    NUM 2,DATE
410  DATA  1,4,17,20,1,6,17,10,2,
    8,15,6,2,10,15,3,3,12,7,8
420  DATA  ENREGISTREMENT CONFIRM
    E,ENTREE,APPUYER SUR 'RETURN
    ' APRES LECTURE,MODIFICATION
    ,DATE INCORRECTE,CONSULTATIO
    N
430  FOR I = 1 TO 5: READ LI$(I):
     NEXT : FOR I = 1 TO 5: READ
    TY%(I),ZV%(I),ZH%(I),ZL%(I):
     NEXT : FOR I = 1 TO 3: READ
    ME$(I),I$(I): NEXT
440 PO$ = ".....
    ,,,,,,,,,,,,,"
500 C = 1: FOR I = 1 TO 5:ZY$(I) =
    "": NEXT
510 Z$ = I$(C): GOSUB 90:B% = 8 *
    (C > 1): FOR I = 1 TO 5: VTAB
    ZV%(I): HTAB 1: PRINT LI$(I)
    ": ": NEXT :X = 1 * (B% = 8)
    :ME = 0:D% = 1:V = 23:Z$ = "
    'ESC' RENVOIT AU NIVEAU SUPE
    RIEUR": GOSUB 80
520  FOR LL = D% TO 5:TY = TY%(LL
    ):V = ZV%(LL):H = ZH%(LL):LO
     = ZL%(LL)
530 E = 0: GOSUB 149: IF E = 9 AND
    LL = 1 THEN LL = 5: NEXT : GOTO
    600
540  IF E = 9 AND ME = 0 THEN L% =
    LL
550  IF E = 9 THEN LL = LL - 2:ME
     = 2: GOTO 570
560 ZY$(LL) = ZZ$:ME = 2 * (LL <
    L% - 1)
570  NEXT :L% = 0: IF C = 3 THEN
    Z$ = ME$(2):V = 21: GOSUB 80
    : GET Z$: GOTO 500
580 ZM$ = ME$(1): GOSUB 70: IF Z$
     = "N" THEN X = 1:B% = 0:D% =
    5: GOTO 520
590 C = C + 1: GOTO 510
600 L% = 0: IF C > 1 THEN C = C -
    1: GOTO 510
610 Z$ = "TRAITEMENT PRECEDENT": GOSUB
     90: FOR Z = 1 TO 2000: NEXT
    : GOTO 500
```

Ergonomie des programmes

Guy Mathieu

Principes généraux

Entre deux « bons » programmes, c'est-à-dire écrits sans erreurs, donnant les résultats attendus, avec des temps de travail corrects, etc., l'un peut être vraiment bon, et admis comme tel par ses utilisateurs, et l'autre rejeté.

La différence tient souvent dans ce qui peut apparaître comme mineur : la présentation des écrans et la facilité d'utilisation des claviers (ce dernier point étant étroitement lié au matériel utilisé). Nous utiliserons le vocable « ergonomie des programmes » pour désigner tout ce qui, dans un programme, est fait pour faciliter la relation entre l'utilisateur et l'impersonnel « système ».

La première chose qu'il convient de noter est la suivante : hormis quelques personnes très averties, ayant déjà eu maintes occasions de travailler en utilisant un dialogue écran-clavier, le futur utilisateur ne saura pas expliquer ce qu'il veut, ni choisir en connaissance de cause entre des propositions qu'on lui fera, même étayées de fac-similés sur le papier : car l'écran « bouge », le champ de vision n'est pas le même, les contrastes diffèrent, les jeux de caractères sont différents, toute la dynamique du dialogue s'efface derrière quelques schémas statiques.

Une seule solution : simuler le futur outil de dialogue, quasiment à l'identique si cette simulation se fait sur le matériel qui servira plus tard de support à l'application réelle ; au plus près si, par exemple, on simule sur micro-ordinateur une application qui utilisera dans la réalité un gros système multi terminaux.

Mais il faut bien se rendre compte qu'on ne pourra avancer autrement que pas à pas ; la simulation du dialogue complet ne sera donc possible que quand l'étude sera suffisamment avancée, mais pas trop, afin que peu d'éléments soient figés d'avance. Il peut donc être intéressant de réaliser une première approche grâce à une « pédagogie du dialogue écran-clavier », appuyée sur un programme de simulation permettant de mieux montrer aux futurs utilisateurs ce qu'ils peuvent attendre d'un tel dialogue.

Présentation du programme

C'est un programme ayant cet objectif que nous vous proposons. Il fait « défiler » des genres d'écrans variés, et permet de tester des commandes de divers types (il est bien sûr conçu pour Apple II).

Le premier écran (instructions 5000-5040) demande l'entrée de deux données alphanumériques. Il émet une seule exigence : au moins 1 caractère. Il avertit d'une frappe RETURN trop hâtive par un message clignotant (un traitement des erreurs permettrait en outre d'afficher un message si le fichier demandé ne se trouvait pas sur le disque : notre programme complet comprend cette option).

Le deuxième écran (instructions 5050-5095) propose 3 choix (4 avec l'arrêt) et demande une réponse numérique. Toute réponse numérique hors fourchette amène un message clignotant (il aurait également été possible de saisir la réponse en alphanumérique, pour éviter le message système « REENTER » en cas d'entrée d'une réponse non numérique.

Quel que soit le choix effectué (sauf arrêt), l'écran suivant (instructions 5200-5500) propose la liste des produits (bien entendu, dans une application réelle, il faudrait procéder en « cascade » à partir de groupes de produits). La réponse se fait en déplaçant le curseur sur l'écran. Sur l'Apple II, il n'existe pas de flèches de déplacement ↑ et ↓, on utilise donc les flèches ← et → pour monter et descendre.

RETURN provoque la recherche du produit voulu. La frappe de RETURN sur la première ligne vierge permet de sortir de l'option. C'est une règle générale : ne pas construire d'écran dont il soit impossible de s'évader.

Arrêtons-nous un instant sur cet usage de la touche RETURN : elle est normalement utilisée sur l'Apple II pour marquer la bonne conclusion d'une action, une entrée de données par exemple. Il s'y attache ainsi, dès qu'on a acquis quelque habitude de l'Apple II, un sens positif, d'approbation, d'acquiescement. C'est ici le cas quand, après avoir déplacé le curseur, on se trouve en bonne position : OUI, c'est bien ce qu'on voulait.

A l'opposé du clavier, la touche ESC peut être programmée pour jouer un rôle négatif, de refus, d'erreur. Le choix entre l'alternance RETURN et ESC passe très vite dans les mœurs. Ces touches sont utilisées pour confirmer ou infirmer la suppression d'un article ou le souhait de recherche ou de création d'un article (sous-programme 500-550).

Regardons maintenant l'écran de saisie des données relatives à un produit (instructions de 2000 à 2350 et sous-programme 300-309). Là encore, RETURN trouve son sens d'acquiescement : on l'utilise pour conserver la donnée affichée sur l'écran s'il y en a une.

Des contrôles sont réalisés en outre sur la longueur des zones et sur la vraisemblance des données numériques.

La suppression d'un article se fait d'une façon un peu artificielle par CTRL-A : une double touche pour éviter toute fausse manœuvre. De plus, une confirmation est demandée. Il est particulièrement important pour la tranquillité de l'utilisateur qu'une manœuvre risquant de détruire tout ou partie d'un fichier ne puisse pas être effectuée par simple effleurement d'une touche.

A noter que l'entrée d'un nouvel article se fait en deux stades : d'abord en choisissant sur l'écran des produits une ligne vierge (y positionner le curseur, puis RETURN ensuite, en entrant les données, y compris le libellé).

N.B. Dans l'exemple proposé, nous n'avons pas programmé l'option « traitement ».

Aspects ergonomiques généraux

D'une façon plus générale, quelles sont sur l'Apple II les possibilités de jouer sur l'ergonomie des programmes lors du dialogue utilisateur-système ?

1 - Le clavier

Nous avons déjà vu un exemple de gestion du clavier avec l'utilisation des

trois touches : RETURN, ESC et barre d'espacement.

Une lacune importante de l'Apple II est l'inexistance des flèches de déplacement vertical du curseur. Quand le besoin se fait sentir de déplacer le curseur dans les 4 dimensions, il existe deux possibilités :

— ou bien désigner des touches alphabétiques pour les 4 mouvements (ex. : les touches I, J, K, M utilisées en programmation après ESC). C'est très artificiel ;

— ou bien n'utiliser que les deux touches de déplacement horizontal et en inverser la signification à l'aide d'une touche jouant le rôle de bascule (grâce à la basse d'espacement, dans le cas du programme Visicalc).

Dans les éditions de type latin (de gauche à droite et du haut vers le bas), on a alors :

• une flèche d'avance : →, bas, droite

• une flèche de recul : ←, haut, gauche.

Attention à utiliser à bon escient les deux types d'entrées, en mode GET ou par INPUT (avec le RETURN) : le GET a un aspect définitif et doit être suivi de contrôles, alors que la possibilité de retour arrière avant un RETURN a souvent un aspect sécurisant. En revanche, le GET permet d'utiliser toute touche avec une signification particulière à l'application considérée, en particulier d'ailleurs d'autoriser l'utilisation comme en mode INPUT des flèches pour la correction.

2 - L'écran

D'abord, n'y point trop écrire ; sauf pour les libellés qui reviennent souvent, et qu'on connait vite par cœur ; l'écran est alors un rappel. Éviter des textes trop longs : si possible, n'écrire qu'une ligne sur deux. Une page entière de consignes sera généralement plus utile sur une feuille d'instructions.

Ensuite, on peut utiliser toutes les possibilités de gestion des caractères : flash et inverse, titres encadrés, ralentissement de la vitesse d'inscription... Si on travaille en graphique, utilisation rationnelle de la couleur et des graphiques.

Ne pas en abuser : on remarque au premier coup d'œil un message clignotant isolé. On ne remarque plus rien si on voit trop souvent des flashes, des textes inversés et autres « fantaisies ».

3 - Le son

Vous disposez d'une part de la « cloche », d'autre part de la possibilité de programmer des sons plus complexes, voire de la musique.

A utiliser à bon escient. Ne pas oublier que pour l'utilisateur quelque peu habitué à l'Apple II, la cloche simple signifie le plus souvent qu'une erreur vient d'être remarquée par le programme : ne pas utiliser le même son pour marquer une approbation !

Penser aussi à la « pollution sonore » : un bruit attirant l'attention de temps en temps, sur une erreur par exemple, peut être utile. Une trop grande abondance de messages sonores devient vite irritante, et donc inefficace.

Aspects ergonomiques spécifiques

Les observations qui précèdent sont valables pour tous les programmes. Chaque programme particulier peut posséder ou non des vertus ergonomiques liées au contexte dans lequel opère l'utilisateur. Nous n'entreprendrons pas d'en faire un recensement exhaustif, mais quelques points méritent d'être cités, par exemple :

— si les entrées écran émanent d'un document, il est nécessaire qu'elles soient effectuées dans l'ordre normal de lecture du document ;

— si elles résultent de plusieurs documents, épuiser, sauf impossibilité absolue, les informations provenant d'un document avant de passer au suivant ;

— même chose si certaines informations apportées par l'écran doivent être recopiées, par exemple pour compléter un imprimé ;

— pour les graphiques, calculer les échelles de façon à mettre en relief les phénomènes intéressants à observer, tout en évitant des difficultés d'interprétation des dites échelles ;

— s'il faut utiliser des symboles, des formulations inhabituelles, assurez-vous qu'ils seront compris sans ambiguités : tout le monde ne sait pas ce qu'est une échelle logarithmique, ou la représentation d'un nombre en notation scientifique ;

— il en est de même du vocabulaire utilisé et des abréviations : ce qui est sans équivoque pour le concepteur du programme ne l'est pas nécessairement pour l'utilisateur ;

— quand un temps d'attente se produit (calcul, lecture de fichier), l'utilisateur doit être prévenu afin qu'il ne s'impatiente pas, ou ne croie pas que l'ordinateur est en panne. Trois cas peuvent se présenter :

• on connaît le temps d'attente (par un nombre d'articles à lire en fichier, ou de boucles de calcul à exécuter) : afficher une approximation de ce temps — cf. instruction 5035 du programme ;

• le temps d'attente se décompose en phases successives : afficher entre chaque phase un bref message donnant l'état d'avancement — éventuellement, en profiter pour glisser un message rappelant à l'opérateur ce qu'il devra faire ensuite ;

• un temps relativement long, mais non calculable (c'est en outre le cas du garbage collection, pour qui n'utilise pas l'approche présentée dans Pom's 2), va se dérouler sans phases intermédiaires : placer un message expliquant ce qui se passe, et fournissant si possible le délai maximum au-delà duquel l'utilisateur sera en droit de penser à une défaillance du système.

Création d'un fichier

Pour créer un nouveau fichier, faire RUN 6000.

```
]LIST
10 D$ =  CHR$ (4): DIM LI$(20): GOSUB 15
       000:ZERO$ = "
                                ":UN$ = "
            ": GOTO 5000
100  REM
*** ECRITURE DU FICHIER PRODUITS ***
110  PRINT D$"OPENPRODUITS";NOM$;",L80":
       PRINT D$"WRITEPRODUITS";NOM$;",R
     ";CP: PRINT LI$
150  FOR I = 1 TO 8: PRINT CA(I): NEXT :
       PRINT D$"CLOSEPRODUITS";NUM$: RE
     TURN
200  REM
*** LECTURE DU FICHIER PRODUITS ***
210  PRINT D$"OPENPRODUITS";NOM$;",L80":
       PRINT D$"READPRODUITS";NOM$;",R"
     ;CP: INPUT LI$
235  IF LI$ = "" THEN 280
240  FOR I = 1 TO 8: INPUT CA(I): NEXT
280  PRINT D$"CLOSEPRODUITS";NOM$: RETUR
     N
```

```
300  HOME : PRINT "ETABLISSEMENT : ";ET$
       : VTAB 3: PRINT "FICHE PRODUIT ":
       VTAB 5: PRINT "DESIGNATION ('CTR
       L A' POUR SUPPRIMER)": VTAB 7: PR
       INT " ";LI$
305  FOR I = 1 TO 8
306  VTAB I + 8: PRINT CG$(I);"  ";: IF
       CA(I) < > 0 THEN  PRINT CA(I)
307  IF CA(I) = 0 THEN  PRINT
308  NEXT
309  RETURN
500  REM
*** GESTION RETURN ET ESCAPE ***
505  POKE  - 16368,0
510  PRINT "(RETURN/ESCAPE) ";
520  GET Z$:Z =  ASC (Z$)
525  IF Z < > 13 AND Z < > 27 THEN  PR
       INT "";: GOTO 520
550  RETURN
600  REM
*** FICHIER NOMS ***
610  PRINT D$"OPENPRODUITS";NOM$;",L80"
620  FOR CP = 1 TO 20: PRINT D$"READPROD
       UITS";NOM$;",R";CP: INPUT LI$(CP)
       : NEXT
650  PRINT D$"CLOSEPRODUITS";NOM$: RETUR
       N
2000  REM
*** MISE A JOUR DU FICHIER ***
2005  FOR I = 1 TO 8:CA(I) = 0: NEXT : G
       OSUB 200
2020  GOSUB 300
2030  VTAB 7: HTAB 1: INPUT A$
2035  IF A$ < > CHR$ (1) THEN 2040
2036  VTAB 20: PRINT "CONFIRMEZ LA SUPPR
       ESSION ";: GOSUB 500
2037  IF Z = 13 THEN LI$ = "": FOR I = 1
       TO 8:CA(I) = 0: NEXT : GOTO 2170
2038  IF Z = 27 THEN  RETURN
2039  PRINT " ": GOTO 2036
2040  IF A$ = "" THEN  VTAB 7: PRINT " "
       ;LI$: GOTO 2080
2050 LI$ = A$: IF  LEN (LI$) < 6 OR  LEN
       (LI$) > 39 THEN  GOSUB 13000: VT
       AB 8: PRINT ZERO$;: VTAB 7: PRINT
       ZERO$;: GOTO 2030
2070  VTAB 23: PRINT ZERO$;
2080  FOR I = 1 TO 8
2090  VTAB I + 8: HTAB 31: INPUT A$
2100  IF A$ = "" THEN  VTAB I + 8: HTAB
       32: PRINT CA(I): GOTO 2160
2120  IF  LEN (A$) = 0 OR  LEN (A$) > LG
       (I) OR  VAL (A$) > MG(I) THEN  GO
       SUB 13000: VTAB I + 8: HTAB 32: P
       RINT UN$: GOTO 2090
2130  GOSUB 13100: IF ER = 1 THEN ER = 0
       : VTAB I + 8: HTAB 32: PRINT UN$:
       GOTO 2090
2140  VTAB 23: PRINT ZERO$;:CA(I) =  VAL
       (A$)
2160  NEXT
2170  GOSUB 300: VTAB 20: HTAB 1: PRINT
       "O.K. POUR ENREGISTREMENT ?": GOS
       UB 500
2210  IF Z = 13 THEN 2240
2220  VTAB 22: HTAB 1: PRINT "RETURN POU
```

.../...

édi tests

LA BIBLIO-TECHNIQUE DES PROFESSIONNELS

MISE EN ŒUVRE DU BUS IEEE 488. Utilisation et réalisation d'appareils. Par Gérard Bastide et Jean-René Vellas. Plus de mille appareils sont équipés en IEEE 488. Ce livre décrit comment mettre en œuvre toutes les possibilités du BUS IEEE, il comprend la description et les syntaxes sur des calculateurs différents de toutes les commandes unillgnes ou multilignes, universelles ou adressées et la réponse à toutes sortes de questions : comment connaître au premier coup d'œil les capacités d'un périphérique ? Deux appareils peuvent-ils communiquer sans requérir l'intervention ou même la présente du calculateur ?...

LES SYSTÈMES A MICROPROCESSEURS. Par Daniel-Jean David. Ce livre est une initiation aux conditions techniques de la révolution micro-informatique. Les différents circuits intégrés : microprocesseurs, mémoires, boîtiers d'entrées-sorties sont décrits ainsi que la façon de les assembler pour former un système. Les phases du traitement d'une application et du développement d'un système à microprocesseur sont décrites, notamment du point de vue du logiciel (programmation en assembleur) et des choix à effectuer.

Chaque volume : 128 pages - 82,00 FF

E/2.LC

P.S.I. DIFFUSION

P.S.I. DIFFUSION
41-51, rue Jacquard
BP 86 - 77400 Lagny-s/Marne
FRANCE
Téléphone (6) 007.59.31
P.S.I. BENELUX
5, avenue de la Ferme Rose
1180 Bruxelles
BELGIQUE
Téléphone (2) 345.08.50
au Canada :
SCE Inc.
3440 rue Saint Denis
Montréal Québec H2X3L1
Tél. : (514) 843.76.63

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

DESIGNATION	NOMBRE	PRIX
(par avion : ajoutor 8 FF (75 FB) par livre)	TOTAL	

NOM _____ PRENOM _____

rue _____ N° _____

Code post. _____ Ville _____

P.E.12

```
       R CONSERVER ": PRINT "SINON RETAP
       EZ LA LIGNE": GOTO 2020
2240   PRINT : GOSUB 100: HOME : PRINT "A
       UTRE ARTICLE ?": GOSUB 500:LI$(CP
       ) = LI$
2350   IF Z = 13 THEN 5200
2400   RETURN
4000   REM
*** CONSULTATION ***
4010   HOME : GOSUB 200: GOSUB 300
4020   VTAB 24: PRINT "BARRE D'ESPACEMENT
        POUR LA SUITE";: GOSUB 12000
4070   FOR I = 1 TO 8:CA(I) = 0: NEXT : H
       OME : PRINT "AUTRE ARTICLE ?": GO
       SUB 500
4120   IF Z = 13 THEN 5200
4130   RETURN
5000   REM MENU
5005   HOME : VTAB 1: HTAB 7: PRINT "GEST
       ION DU FICHIER-PRODUITS"
5010   VTAB 6: INPUT "NOM DE L'ETABLISSEM
       ENT ?  ";ET$
5015   IF ET$ = "" THEN  VTAB 9: FLASH :
       PRINT "AU MOINS 1 CARACTERE S.V.P
       . ": NORMAL : GOTO 5010
5019   VTAB 9: PRINT ZERO$
5020   VTAB 9: INPUT "NOM DU FICHIER-PROD
       UITS ? ";NOM$
5025   IF NOM$ = "" THEN  VTAB 11: FLASH
       : PRINT "AU MOINS 1 CARACTERE S.V
       .P. ": NORMAL : GOTO 5020
5029   VTAB 11: PRINT ZERO$
5035   VTAB 14: PRINT "PATIENCE POUR QUEL
       QUES SECONDES"
5040   GOSUB 600
5050   HOME : PRINT "VOULEZ-VOUS ?": PRIN
       T
5060   PRINT "1 - METTRE A JOUR LE FICHIE
       R": PRINT "2 - CONSULTER LE FICHI
       ER": PRINT "3 - EFFECTUER UN TRAI
       TEMENT": PRINT : PRINT "    (0 PO
       UR ARRETER)"
5085   VTAB 13: INPUT "    VOTRE CHOIX ?
       ";SU: IF SU = 0 THEN  HOME : PRIN
       T "AU REVOIR": END
5095   IF SU < 0 OR SU > 3 THEN  GOSUB 13
       000: VTAB 18: HTAB 18: PRINT UN$:
        GOTO 5085
5200   HOME : PRINT "LISTE DES PRODUITS":
        VTAB 3: FOR I = 1 TO 20: PRINT L
       I$(I): NEXT :HT = 5:LM = 20: GOSU
       B 30000
5260   PRINT : IF CP = 0 THEN 5050
5500   ON SU GOSUB 2000,4000,10000: GOTO
       5050
6000   REM
*** CREATION DU FICHIER ***
6005 D$ =  CHR$ (4): TEXT : HOME : VTAB
       1: HTAB 7: PRINT "CREATION D'UN F
       ICHIER":ZERO$ = "
                                "
6010   VTAB 6: INPUT "NOM DU FICHIER-PROD
       UITS ? ";NOM$
6015   IF NOM$ = "" THEN  VTAB 9: FLASH :
        PRINT "AU MOINS 1 CARACTERE S.V.
       P. ": NORMAL : GOTO 6010
6019   VTAB 9: PRINT ZERO$
6025   VTAB 12: PRINT "PATIENCE POUR QUEL
       QUES SECONDES"
6030   PRINT D$"OPENPRODUITS";NOM$;",L80"
       : PRINT D$"DELETEPRODUITS";NOM$:
       PRINT D$"OPENPRODUITS";NOM$;",L80
       "
6040   FOR CP = 1 TO 20: PRINT D$"WRITEPR
       ODUITS";NOM$;",R";CP: PRINT "": N
       EXT : PRINT D$"CLOSEPRODUITS";NOM
       $: END
10000  REM
*** PROGRAMME DE TRAITEMENT ***
10010  HOME : PRINT "TRAITEMENT NON PROG
       RAMME"
10020  VTAB 10: PRINT "BARRE D'ESPACEMEN
       T POUR LA SUITE";: GOSUB 12000: R
       ETURN
12000  REM  *** ATTENTE ***
12010  POKE  - 16368,0: PRINT " ";
12020 Z =  PEEK ( - 16384): IF Z <  = 12
       8 THEN 12020
12025  POKE  - 16368,0: RETURN
13000  REM  *** REFUS ***
13010  VTAB 23: FLASH : PRINT " ENTREE R
       EFUSEE ": NORMAL : RETURN
13100  REM *** CARACT. NUMER. ***
13110  FOR JJ = 1 TO  LEN (A$):AA =  ASC
        ( MID$ (A$,JJ,1))
13120  IF (AA > 47 AND AA < 58) OR AA =
       46 THEN  NEXT : RETURN
13130  VTAB 23: FLASH : PRINT " CHIFFRES
        SEULEMENT S.V.P. "
13140 ER = 1:JJ =  LEN (A$): NORMAL : RE
       TURN
15000  REM *** LIBELLES CARACT. ***
15010  DATA  CODE ARTICLE      (6 CHIFFRE
       S)
15011  DATA  CODE FOURNISSEUR (4 CHIFFRE
       S)
15012  DATA  CODE CASIER       (4 CHIFFRE
       S)
15013  DATA  CODE REAPPRO.      (1 CHIFFR
       E)
15014  DATA  POIDS DE LA PIECE          (K
       G)
15015  DATA  PRIX UNITAIRE          (FR.CM
       .)
15016  DATA  STOCK MINI             (NOMBR
       E)
15017  DATA  QUANT. A COMMANDER     (NOMBR
       E)
15019  FOR I = 1 TO 8: READ CG$(I): NEXT

15020  DATA 6,4,4,1,7,7,3,3
15025  FOR I = 1 TO 8: READ LG(I): NEXT
15030  DATA 999999,9999,2999,9,100,1000,
       200,500
15035  FOR I = 1 TO 8: READ MG(I): NEXT
15200  RETURN
30000  REM  *** POSITION DU CURSEUR ***
30001  VTAB 24: PRINT "MONTER ((-) , DES
       CENDRE (-)) , RETURN";
30002 CP = 0: VTAB CP + 2
30005  HTAB HT: GET Z$:Z =  ASC (Z$)
30007  IF Z = 13 THEN  RETURN
30010  IF Z = 21 THEN CP = CP + 1
30015  IF Z = 8 THEN CP = CP - 1
30017  IF Z <  > 13 AND Z <  > 21 AND Z
       <  > 8 THEN  PRINT "";: GOTO 300
       05
30020  IF CP <  = 0 THEN CP = 0
30025  IF CP >  = LM THEN CP = LM
30030  VTAB CP + 2: GOTO 30005
```

Un programme de HELLO complet

Thierry Le Tallec
Jacques Tran-Van

Le programme le plus fréquemment utilisé est sans conteste le programme HELLO, ou BONJOUR, qui nous sert à « booter » les disquettes. De nombreux programmes ont déjà été proposés à cet effet, chacun ayant ses avantages propres. Nous avons voulu réunir dans un programme unique les possibilités qui nous paraissaient les plus intéressantes, à savoir :

— affichage du volume, des secteurs libres et de 40 titres ;

— lancement des programmes par enfoncement d'une seule touche ;

— chargement jusqu'à cinq fois plus rapide ;

— test de la carte langage et chargement de l'INTEGER en 5 secondes ;

— changement de drive avec les flèches gauche et droite ;

— delete par CTRL-D suivi du code du programme ;

— lock/unlock par CTRL-L, suivi du code... ;

— load/bload avec affichage des adresses par ESC, suivi... ;

— catalogue normal avec la touche "/" ;

— revectorisation du RESET dans le moniteur par la touche "*" ;

— taille du programme réduite à 6 secteurs.

Mémorisation sous forme de programme Applesoft

Afin d'obtenir un programme à la fois court et rapide, il a fallu le réaliser en assembleur ; nous avons utilisé pour cela le BIG MAC. Pour être « bootable », un programme doit normalement être écrit en Applesoft ; nous avons donc converti notre programme assembleur B.HELLO en programme assembleur grâce à la procédure suivante :

NEW vide la mémoire,
1 CALL 2061 : appel du programme assembleur,
BLOAD B.HELLO : charge le programme en mémoire,
CALL —151 :passe en assembleur,
*AF : FE OC :redéfinit l'adresse de fin de programme de façon à englober le programme assembleur,
CTRL-C :retour à l'Applesoft,
SAVE HELLO :sauvegarde du nouveau programme de boot.

Sur la disquette Pom's d'accompagnement, nous avons simplement reproduit le programme HELLO ainsi obtenu. La technique que nous venons d'illustrer permet d'incorporer une routine en assembleur à un programme Applesoft qui l'utilise. Cela permet de copier ensuite le programme avec un simple LOAD suivi de SAVE, ce qui est plus simple que de devoir faire appel à FID pour copier le fichier binaire.

Pour fonctionner, HELLO requiert 48K de mémoire (ou plus), le DOS 3.3. et l'Applesoft en ROM.

Vitesse de chargement

Une remarque relative à la vitesse de chargement : depuis la parution du livre BENEATH APPLE DOS, on sait qu'en fait, c'est le DOS, et non l'Apple, qui est lent. Il lit un secteur dans son buffer en $9600, le recopie à son adresse normale et, quand il veut lire le secteur suivant, le disque a déjà tourné, obligeant le DOS à patienter un tour.

En chargeant un secteur directement à son adresse sans passer par un buffer, il n'est plus nécessaire d'attendre, c'est tout. Les ZDOS 2.2., DOS 3.4, HYPERDOS, &BLOAD et autres TURBODOS reposent simplement sur ce principe que nous avons déjà utilisé dans l'article « Chargez vite vos fichiers binaires » paru dans le Pom's 4.
Le chargement de l'INTEGER dans la carte langage se fait de la même manière après les tests :

— Y a-t-il une carte langage ?
— Si oui, contient-elle déjà l'INTEGER ?
— Sinon, y a-t-il INTBASIC sur la disquette ?

Structure du programme

Le programme comporte une première partie, de $80D à $C03, qui gère les commande DELETE, LOCK, ..., la lecture du catalogue, etc.

La seconde partie, placée en $9AA6 (premier buffer du DOS), assure le chargement du programme et son exécution. Cet endroit a été choisi pour que le programme chargé ne puisse écraser ce module vital.

Remarques

Dans la présentation du catalogue, les symboles suivants sont utilisés :
"*" pour un fichier binaire,
"]" pour de l'Applesoft,
">" pour de l'Integer,
"." pour un fichier TEXT ou EXEC.

Le type de chaque programme est présenté par un symbole en inverse si celui-ci n'est pas verouillé.

```
 1 LOWCASE  =    1 ;( =0 si pas de minuscule)
 2 *
 3 *
 4          ORG  $80D     ;(call 2061)
 5 *
 6 *********************************
 7 *                               *
 8 *       MENU ULTRA-RAPIDE       *
 9 *       AVEC LOCK-UNLOCK        *
10 *          ET DELETE            *
11 *                               *
12 *       Thierry Le Tallec       *
13 *              et               *
14 *       Jacques Tran-Van        *
15 *                               *
16 *********************************
17 *
18 *
19 *    /   15 Juin 1982
20 *
21 *
22 *
```

```
FILENUM  EQU  $00      ;rang du pgm
FILEPTR  EQU  $01      ;pointeur de pgm
TSLPTR   EQU  $01      ;pointeur de TSL
LOCKFLG  EQU  $03      ;drapeau de pgm 'LOCKED'
DELFLAG  EQU  $04      ;drapeau de 'DELETE'
START    EQU  $04      ;pointeur de d‚but de pgm
YREG     EQU  $06      ;sauvegarde de Y
RUNLOAD  EQU  $07      ;drapeau 'RUN/LOAD'
TYPE     EQU  $0C      ;type du pgm
POINTER1 EQU  $0E      ;pointeurs de d‚placement
POINTER2 EQU  $10      ;de l'INTEGER
FILEH    EQU  $10      ;poids fort de l'adresse du catal.
INDEX    EQU  $19      ;index dans le bandeau
WRITERR  EQU  $1E      ;drapeau de protection-‚criture
WNDLFT   EQU  $20      ;marge gauche de l'‚cran
WNDWDTH  EQU  $21      ;largeur de la fen^tre ‚cran
WNDTOP   EQU  $22      ;haut de la fen^tre ‚cran
CH       EQU  $24      ;position horizontale du curseur
TRAP     EQU  $48      ;voir Call Apple janvier 82
LOMEM    EQU  $69      ;d‚but de la zone variable simple
EOPRGM   EQU  $AF      ;pointeur de fin de pgm
PGMSTRT  EQU  $CA      ;d‚but de program INTEGER
ERRFLG   EQU  $D8      ;drapeau de ON ERR
YSAV     EQU  $FB      ;sauvegarde de Y
WARMSTRT EQU  $3D0     ;d‚part @ chaud du DOS
MANAGER  EQU  $3D6     ;entree du file manager
LOCIOB   EQU  $3E3     ;localise l'IOB
SOFTEV   EQU  $3F2     ;vecteur de RESET
PWREDUP  EQU  $3F4
SYMBOL   EQU  $7DD     ;adresse ‚cran du symbole-commande
DIRCTORY EQU  $1000    ;adresse de d‚but du catalogue
VTOC     EQU  DIRCTORY+$F38 ;pointeur de d‚but de VTOC
BEGPGM   EQU  $2FFC    ;adresse de chargement de l'INTEGER
BUFFER   EQU  $9600    ;tampon du 1er secteur du pgm
TSL      EQU  $9700    ;tampon de la TSL du pgm
FINALE   EQU  $9AA6    ;1er buffer donn‚es du DOS
WARMDOS  EQU  $9DBF    ;d‚part @ chaud du DOS
INT      EQU  $A59E    ;routine de passage @ l'INTEGER
FLIPCRD  EQU  $A5B2
DOSEXEC  EQU  $A5C6    ;routine 'EXEC'
PRNTERR  EQU  $A702    ;affiche les messages d'erreur
CLRCOLD  EQU  $A75B
CLNDOS   EQU  $A7D4    ;construction des buffers du DOS
DOSHOOK  EQU  $A851    ;reconnecte le DOS
DFLTDRV  EQU  $AA68    ;# disque par d‚faut
NAME     EQU  $AA75    ;tampon de nom du programme
PARMLIST EQU  $B5BB    ;parametres du file manager
RWTS     EQU  $B7B5    ;entr‚e de la RWTS
IOB      EQU  $B7E8    ;param}tres de la RWTS
DRIVE    EQU  IOB+2    ;# disque
VOLUME   EQU  IOB+3    ;volume attendu
TRACK    EQU  IOB+4    ;piste
SECTOR   EQU  IOB+5    ;secteur
BUFADR   EQU  IOB+8    ;pointeur du tampon de donn‚es
CMD      EQU  IOB+12   ;commande de la RWTS
PREVOL   EQU  IOB+14   ;volume trouv‚
CARDON   EQU  $C083    ;autorise la lecture de la RAM
CARDOFF  EQU  $C081    ;autorise la lecture de la ROM
LANGUAG  EQU  $D000    ;d‚but de la carte langage
RUN      EQU  $D566    ;ex‚cution d'un pgm APPLESOFT
BASCTYP  EQU  $E000    ;type du langage
LINPRT   EQU  $ED24    ;affiche les 2 octets de X,A
INTRUN   EQU  $EFEC    ;ex‚cution d'un pgm INTEGER
INTCOLD  EQU  $F000    ;initialisation INTEGER
PRNTAX   EQU  $F941    ;affiche les registres A,X
INIT     EQU  $FB2F    ;text
TABV     EQU  $FB5B    ;positionnement vertical du curseur
BELL     EQU  $FBDD    ;bip
UP       EQU  $FC1A    ;monte le curseur
HOME     EQU  $FC58    ;efface l'‚cran
CLREOL   EQU  $FC9C    ;efface la fin de la ligne
RDKEY    EQU  $FD0C    ;entr‚e d'un caract}re
CROUT    EQU  $FD8E    ;envoie un retour chariot
PRYX2    EQU  $FD96    ;va @ la ligne, affiche X,Y,'-'
COUT     EQU  $FDED    ;affiche le caract}re dans A
SETINV   EQU  $FE80    ;affichage en inverse
SETNORM  EQU  $FE84    ;affichage en mode normal
SETKBD   EQU  $FE89    ;IN#0
SETVID   EQU  $FE93    ;PR#0
ERR      EQU  $FF2D    ;affiche 'ERR'
MONITOR  EQU  $FF69    ;entr‚e du moniteur
*
BEGIN    JSR  INIT     ;plein ‚cran
         JSR  HOME     ;efface l'‚cran
         JSR  SETKBD   ;IN#0
         JSR  SETVID   ;PR#0
         LDA  $3D5     ;quelle configuration ?
         CMP  #$9D     ;48K de ram ?
         BEQ  GOODDOS
         JSR  ERR      ;bip
         JMP  WARMSTRT ;rend la main
GOODDOS  JSR  DOSHOOK
         LDA  #$D3     ;efface PLE
         STA  $9D00
         LDA  #$9C
         STA  $9D01
         LDX  #$00     ;d‚place le pgm en FINALE
         STX  INDEX
MOVEPGM  LDA  LETSGO,X
         STA  FINALE,X
         INX
         BNE  MOVEPGM
         LDY  #<END    ;revectorise le reset
         LDA  #>END    ;pour une sortie
         JSR  SETRST   ;plus propre.
         LDX  #2       ;cr‚e une fen^tre
         STX  WNDTOP
         JSR  HOME
         DEX           ;commande lecture pour la RWTS
         JSR  CATALOG  ;lit le catalogue
*
         LDY  #$8C     ;calcul de l'espace libre disque
NXTDESC  LDA  VTOC-1,Y  ;descripteur de piste
TESTSCT  BEQ  NXTRACK  ;si 0, piste toute utilisee
         ASL           ;si C = 0, le secteur est utilise
         BCC  TESTSCT  ;sinon il est libre
         PHP           ;sauve Z
         INC  COUNTER  ;incr‚mente le compteur
```

```
         BNE  OTHRSCT  ;de secteurs libres
         INC  COUNTER+1
OTHRSCT  PLP           ;recupere Z
         BNE  TESTSCT+2 ;secteur suivant
NXTRACK  DEY           ;piste suivante
         BNE  NXTDESC  ;fin de la VTOC ?
*
TESTINT  BIT  CARDON   ;lecture carte langage
         LDA  BASCTYP  ;type du basic
         LDY  BASCTYP+1
         BIT  CARDOFF  ;{criture carte langage
         CMP  #$4C     ;y-a-t-il une carte langage?
         BNE  CARDIN   ;oui
         CPY  #$28
         BEQ  RELAI    ;pas de carte
CARDIN   CMP  #$20     ;INTEGER d{j@ charg{ ?
         BNE  SEARCH   ;non
         CPY  #0
         BNE  SEARCH   ;non
RELAI    JMP  FRSTFIL  ;affiche le catalogue
*
NEXTPAGE INC  FILEPTR+1 ;secteur suivant du catalogue
*
SEARCH   LDY  #$EB     ;initialise le pointeur de nom
         STY  YSAV     ;et le sauve
NEXTPGM  LDA  YSAV     ;pr{c{dent pointeur de nom
         CLC
         ADC  #35
         TAY           ;nouveau pointeur
         STA  YSAV
         CMP  #$03     ;fin du secteur ?
         BEQ  NEXTPAGE ;secteur suivant
         LDX  #LOADING-INTBASIC ;longueur de 'INTBASIC'
COMPAR   LDA  (FILEPTR),Y ;charge une lettre du nom
         BEQ  NOFILE   ;si 0, fin du catalogue
         CMP  INTBASIC-1,X ;comparaison avec 'INTBASIC'
         BNE  NEXTPGM  ;si <> alors titre suivant,
         INY           ;sinon lettre suivante
         DEX
         BNE  COMPAR   ;fin de la comparaison ?
*
         LDY  YSAV     ;r{cup}re le pointeur de nom
         DEY
         DEY
         LDA  (FILEPTR),Y
         STA  SECTOR   ;range le # de secteur de la TSL
         DEY
         LDA  (FILEPTR),Y
         BMI  NEXTPGM  ;si effac{ on le saute
         STA  TRACK    ;range Le # de piste de la TSL
FINDINT  LDA  LOADING,X ;affiche 'CHARGEMENT DE L'INTEGER'
         STA  $5AF,X
         INX
         CPX  #LOCTRK-LOADING
         BNE  FINDINT
         JSR  INITPTR
         JSR  RDSECT   ;lit la TSL en $9700
         LDY  #<BEGPGM
         LDA  #>BEGPGM
         JSR  LOCBUF   ;charge INTBASIC en $2FFC
         JSR  BOOTPGM  ;revient avec Y=0
         STY  POINTER1
         STY  POINTER2 ;initialise les pointeurs
         LDX  #$30     ;d{place $30 pages
         STX  POINTER1+1 ;POINTER1 <- $3000
         LDA  #>LANGUAG
         STA  POINTER2+1 ;POINTER2 <- $D000
MOVEINT  LDA  (POINTER1),Y
         STA  (POINTER2),Y
         INY
         BNE  MOVEINT  ;d{place l'INTEGER en $D000
         INC  POINTER1+1
         INC  POINTER2+1
         DEX
         BNE  MOVEINT  ;page suivante
*
         JSR  HOME     ;efface l'{cran
*
NOFILE   LDA  #>DIRCTORY
         STA  FILEPTR+1
         BNE  FRSTFIL  ;saute toujours
*
NXTFILE  LDA  YREG
         CLC
         ADC  #35
         BCC  ONEMORE
         INC  FILEPTR+1
FRSTFIL  LDA  #$0B     ;entr{e du 1er fichier
ONEMORE  TAY
         STY  YREG
         LDA  FILENUM
         CMP  #40      ;pas plus de 40 titres
         BCC  LESS40
         JMP  ENDCAT
LESS40   CMP  #20      ;20 titres par fen^tre
         BNE  LESS20
         STA  WNDLFT
         STA  WNDWDTH
         JSR  HOME
         LDY  YREG
         LDA  FILENUM
LESS20   ASL
         ASL
         TAX           ;X=FILENUM*4
         LDA  (FILEPTR),Y ;# de piste
         BMI  NXTFILE  ;saute si effac{
         BEQ  ENDCAT   ;fin si zero
         STA  BUFFER,X
         INC  FILENUM  ;un fichier de plus
         INY
         LDA  FILEPTR+1
         ASL
         ASL           ;sauve le pointeur de nom
         ASL
         ASL
         ORA  (FILEPTR),Y ;mixe avec le # secteur
         STA  BUFFER+1,X
         INY
```

```
         TYA
         STA  BUFFER+3,X ;sauve l'index
         LDA  #$FF
         STA  LOCKFLG  ;abaisse le drapeau de 'LOCK'
         LDA  (FILEPTR),Y ;type du fichier
         BMI  LOCKED
         LSR  LOCKFLG
         LSR  LOCKFLG  ;lève le drapeau de 'LOCK'
LOCKED   AND  #$7F     ;annule le bit de poids fort
         STA  BUFFER+2,X
         LDX  #$07
         ASL
CHKTYPE  ASL
         BCS  PTYPE
         DEX
         BNE  CHKTYPE
PTYPE    LDA  TABLE,X
         AND  LOCKFLG  ;en inverse si non vérouillé
         JSR  PRINT    ;affiche le caractère et '-'
         LDA  FILENUM
         CMP  #27
         BCS  PLUS26
         ADC  #44
PLUS26   ADC  #148
         JSR  PRINT    ;affiche le caractère et '-'
         LDX  #15
PNAME    INY
         LDA  (FILEPTR),Y ;affiche le nom
         CMP  #"       " ;cherche les caractères
         BCS  NORMDSP  ;de controle,
         CMP  #$80
         BCS  PNAME    ;et les saute.
NORMDSP  JSR  COUT
         DEX
         BNE  PNAME
         JSR  CROUT
         JMP  NXTFILE  ;titre suivant
*
PRINT    JSR  COUT
         LDA  #"-"
         JMP  COUT
*
ENDCAT   JSR  INIT     ;plein écran
         LDY  #15
         STY  CH
         LDA  #0
         STA  RUNLOAD  ;drapeau = "RUN"
         JSR  TABV     ;curseur en haut de l'écran
PVOLUME  LDA  TITLE-1,Y
         STA  $3FF,Y
         DEY
         BNE  PVOLUME  ;affiche le volume
         TYA
         LDX  PREVOL
         JSR  LINPRT
         JSR  CLREOL
         LDA  #21
         STA  CH
         LDX  COUNTER
         LDA  COUNTER+1
         JSR  LINPRT
         LDY  #15
PFRESCT  LDA  TITRE,Y
         STA  $418,Y
         DEY
         BPL  PFRESCT
         JSR  INIT     ;Vtab 22
         LDA  #13
         STA  CH       ;Htab 14
         JSR  CLREOL
ENDCAT1  LDX  #$6A     ;('*' clignotant)
         LDA  SETRST-3 ;si reset <> $9DBF
         CMP  #WARMDOS ;affiche '*'
         BNE  ENDCAT2
         LDX  #$60     ;(curseur clignotant)
ENDCAT2  STX  SYMBOL
         LDY  #13
PCHOIX   LDA  MESSAGE-1,Y
         STA  $7CF,Y   ;affiche le message
         DEY
         BNE  PCHOIX
RAZFLAG  LSR  DELFLAG  ;raz du drapeau
         LSR  LOCKFLG
GET      JSR  RDKEY    ;attend le choix
         CMP  #"       "
         BNE  GET1
         JMP  END      ;fin si 'espace'
GET1     CMP  #"*"
         BNE  GET2
         LDA  #<MONITOR ;reset = $FF69
         LDY  #>MONITOR
CHNGRST  STA  SETRST-3
         STY  SETRST-1
         BNE  ENDCAT   ;saute toujours
GET2     CMP  #"]"
         BNE  GET3
         LDA  #<WARMDOS ;reset = $9DBF
         LDY  #>WARMDOS
         BNE  CHNGRST  ;change le vecteur de reset
GET3     CMP  #$95     ;"->" = disque 2
         BNE  GET4
         LDA  #2
SETDRIV  STA  DRIVE
         STA  DFLTDRV
         JMP  BEGIN    ;relit le catalogue
GET4     CMP  #$88     ;"<-" = disque 1
         BNE  GET5
         LDA  #1
         BNE  SETDRIV
GET5     CMP  #$9B     ;esc ?
         BNE  GET6
         STA  RUNLOAD  ;lève le drapeau de 'LOAD'
         LDA  #$3F     ;('?' inverse )
         STA  SYMBOL
         BNE  RAZFLAG
GET6     CMP  #$8C     ;ctrl-L = lock/unlock
         BNE  GET7
         STA  LOCKFLG
```

```
DISPLAY  AND  #$7F
         STA  SYMBOL
         BNE  GET
GET7     CMP  #$84      ;ctrl-D = delete
         BNE  GET8
         STA  DELFLAG
         LSR  LOCKFLG   ;annule le drapeau 'LOCK'
         BPL  DISPLAY   ;affiche le nouveau symbole
GET8     CMP  #"/"      ;compare avec '/'
         BEQ  DSPCAT    ;affiche le catalogue normal
GET9     CMP  #"0"
         BCC  GET       ;pas < "0"
         CMP  #"["
         BCS  GET       ;pas > "Z"
         CMP  #"A"
         BCC  AFTERZ
         SBC  #44
```


```
AFTERZ   SBC  #149     ;calcule le rang
         CMP  FILENUM
         BCS  GET      ;il n'y en a pas tant!
         ASL
         ASL           ;multiplie par 4
         TAX           ;devient l'index
         LDY  BUFFER,X ;# de piste
         LDA  BUFFER+1,X ;secteur de la TSL
         PHA
         AND  #$0F
         JSR  LOCTRK   ;range piste et secteur dans l'IOB
         PLA           ;récupère le pointeur de nom
         LSR
         LSR
         LSR
         LSR
         ORA  #FILEH
         STA  FILEPTR+1
         LDA  BUFFER+2,X ;type du pgm
         STA  TYPE
         LDA  BUFFER+3,X
         STA  FILEPTR
         LDY  #30      ;affiche le nom choisi
SELECT   LDA  (FILEPTR),Y
         STA  SYMBOL-1,Y ;@ partir du curseur
         STA  NAME-1,Y ;et l'écrit dans le tampon du DOS
         DEY
         BNE  SELECT
         BIT  LOCKFLG  ;run ou lock ?
         BPL  CHECK
         JSR  PROTECT  ;disque protégé en écriture ?
         LDA  (FILEPTR),Y
         EOR  #$80     ;bascule le bit de LOCK
         STA  (FILEPTR),Y
         LDX  #2
         JSR  CATALOG  ;réécrit le catalogue
DEBUT    JMP  BEGIN
*
DSPCAT   JSR  HOME
         LDX  #6       ;code de 'catalog'
         JSR  SETPARM  ;appel du file manager
         JSR  RDKEY    ;attend une entrée clavier
         BMI  DEBUT    ;saute toujours
*
PROTECT  BIT  WRITERR
         BPL  NOERROR  ;retour
         LDX  #4       ;code de 'WRITE PROTECTED DISK'
         BNE  ERROR
*
CHECK    BIT  DELFLAG  ;run ou delete ?
         BPL  NODELETE
         JSR  PROTECT  ;disque protégé en écriture ?
         LDA  (FILEPTR),Y
         BPL  OKDEL
         LDX  #10      ;locked -> ne pas effacer
         BNE  ERROR
OKDEL    LDA  #<NAME   ;l'efface
         STA  PARMLIST+8
         LDA  #>NAME
         STA  PARMLIST+9
         LDX  #5       ;commande delete
         JSR  SETPARM
         JMP  BEGIN
*
NODELETE LDA  TYPE
         BNE  NOTEXT
         JMP  DOSEXEC  ;exec
*
NOTEXT   CMP  #4       ;pgm BINAIRE ?
         BEQ  RDTSL    ;oui -> le charger
         BCC  APLSOFT
BADTYPE  LDX  #13      ;code de 'FILE TYPE MISMATCH'
         BNE  ERROR
APLSOFT  CMP  #2       ;pgm APPLESOFT ?
         BEQ  RDTSL
         LDA  #$20
```

```
        JSR  FLIPCRD  ;essaie de passer en INTEGER
        BEQ  RDTSL    ;si reussi, charge le pgm
        LDX  #$01     ;erreur 'LANGUAGE NOT AVALAIBLE'
*
ERROR   CPX  #4       ;'WRITE PROTECTED DISK' ?
        PHP           ;sauve le resultat
        JSR  BELL     ;bip !
        JSR  SETINV   ;affiche le message d'erreur
        JSR  PRNTERR  ;en inverse
        JSR  SETNORM
        JSR  CLREOL
        JSR  RDKEY    ;attend une entr‚e clavier
        PLP           ;est-ce 'WRITE PROTECTED'
        BEQ  DEBUT    ;si oui, jump BEGIN
        JMP  ENDCAT   ;sinon affiche le menu
*
RDSECT  JSR  LOCIOB   ;localise l'IOB
        JSR  RWTS     ;appelle la RWTS
        BCC  NOERROR  ;si C=0 pas d'erreur
        LDX  #8       ;code de 'I/O ERROR'
        BNE  ERROR
NOERROR RTS
*
RDTSL   JSR  INITPTR
        JSR  RDSECT
        LDA  TSL+1    ;une autre TSL ?
        STA  TSL+256  ;si pas de suite,,stop
        TAY
        BEQ  FRSTSCT  ;non -> saute
        LDA  TSL+2
        JSR  LOCTRK
        INC  BUFADR+1 ;page suivante
        JSR  RDSECT
        LDY  #0
MOVETSL LDA  TSL+268,Y ;rend les TSL contigues
        STA  TSL+256,Y
        INY
        BNE  MOVETSL
*
FRSTSCT LDY  TSL+12   ;lit le 1er secteur en $9600
        LDA  TSL+13
        JSR  LOCTRK   ;range piste et secteur dans l'IOB
        LDY  #<BUFFER
        LDA  #>BUFFER
        JSR  LOCBUF   ;positionne le buffer de RWTS
        JSR  RDSECT
        JMP  FINALE   ;continue
*
CATALOG STX  CMD      ;commande RWTS(1=lect, 2=‚crit,)
        STX  WRITERR
        LDY  #<DIRCTORY
        STY  VOLUME
        STY  FILENUM
        STY  FILEPTR  ;pointeur de pgm = $1000
        LDA  #>DIRCTORY
        STA  FILEPTR+1
        JSR  LOCBUF   ;d‚but du buffer = $1000
        LDA  #$0F
        LDY  #$11
        JSR  LOCTRK   ;range piste et secteur dans l'IOB
NXTSECT JSR  RDSECT
        INC  BUFADR+1
        DEC  SECTOR   ;# de secteur = -1 ?
        BPL  NXTSECT  ;non -> encore un secteur
        LDA  $C0ED    ;teste si protection en ‚criture,
        LDA  $C0EE    ;dans le slot 6 !
        BPL  NOSCTCH  ;non
        STA  WRITERR  ;l…ve le drapeau de protection
NOSCTCH RTS           ;retour
*
SETPARM STX  PARMLIST
        LDA  #6
        STA  PARMLIST+6
        LDA  DFLTDRV
        STA  PARMLIST+5
        JMP  MANAGER  ;appel du file manager
*
TABLE   ASC  ",>]I*SR]X"
TITLE   INV  "--POM'S--"
*
        DO   LOWCASE
*
        ASC  "  Vol,"
TITRE   ASC  " secteurs libres"
*
        ELSE
        ASC  "  VOL,"
TITRE   ASC  " SECTEURS LIBRES"
*
        FIN
*
MESSAGE INV  "VOTRE CHOIX"
        ASC  " :"
*
INTBASIC ASC "   CISABTNI"
*
LOADING ASC  "(CHARGEMENT DE L'INTEGER)"
*
LOCTRK  STY  TRACK    ;range la piste dans l'IOB
        STA  SECTOR
        RTS
*
LETSGO  LST  ON
        ORG  FINALE
*
        LDA  TYPE     ;r‚cup…re le type du pgm
        LSR           ;INTEGER ?
        BCS  INTROUT
        LSR           ;APPLESOFT ?
        BCS  APSOFT
*
BINARY  LDA  BUFFER   ;adresse de d‚but
        STA  START
        SBC  #3       ;determine l'adr, du buffer(C=0)
        TAY
        LDA  BUFFER+1
        STA  START+1
        SBC  #0
```

```
        JSR  LOCBUF   ;positionne le buffer de RWTS
        JSR  BOOTPGM  ;charge le pgm
        JSR  CLEANUP  ;nettoie les buffers du DOS
        BPL  BRUN
        LDX  START
        LDY  START+1
        JSR  PRYX2
        LDX  BUFFER+2
        LDA  BUFFER+3
        JSR  PRNTAX   ;affiche la longueur du pgm
LOAD    JMP  WARMDOS  ;sortie par reset
BRUN    JMP  (START)  ;exécute le pgm
*
APSOFT  ADC  BUFFER   ;calcul de LOMEM
        STA  EOPRGM   ;LOMEM = fin du pgm
        STA  LOMEM    ; =$801 + longueur du pgm
        LDA  BUFFER+1
        ADC  #$08
        STA  EOPRGM+1
        STA  LOMEM+1
        LDA  #$FF     ;on le charge en $7FF
        TAY
        LDA  #$07     ; ($801 - $02)
        JSR  LOCBUF   ;positionne le pointeur de buffer
        JSR  BOOTPGM  ;charge le pgm
        STA  ERRFLG
        STA  $800     ; ($800) = 0
        JSR  CLEANUP
        BMI  LOAD
        JMP  RUN
*
INTROUT LDA  #$60     ;poke 'RTS' dans le  DOS cold start
        STA  $9DE7
        JSR  INT      ;passe en INTEGER.
        LDA  #$6C     ;restaure le code
        STA  $9DE7    ;dans le DOS cold start.
        JSR  INTCOLD  ;initialise le LOMEM
        JSR  CLNDOS   ;revient avec A=0
        SEC
        SBC  BUFFER
        STA  PGMSTRT
        LDA  #$96     ;HIMEM = $9600
        SBC  BUFFER+1
        STA  PGMSTRT+1
        SEC
        LDA  PGMSTRT
        SBC  #2
        TAY
        LDA  PGMSTRT+1
        SBC  #0
        JSR  LOCBUF
        JSR  BOOTPGM  ;charge le pgm
        JSR  SETWARM
        BMI  LOAD
        JMP  INTRUN   ;exécute le pgm
*
BOOTPGM LDY  #$0C
BOOTPG2 LDA  (TSLPTR),Y ;# de piste
        BEQ  ENDBOOT  ;si piste = 0 fin du chargement
        STA  TRACK
        INY
        LDA  (TSLPTR),Y ;# de secteur
        STA  SECTOR
        STY  YREG     ;sauve l'index
        LDY  #<IOB
        LDA  #>IOB
        JSR  RWTS     ;secteur suivant du pgm
        BCS  END      ;sortie si erreur
        LDY  YREG     ;restaure l'index
        INY
        BNE  BOOTNXT
        INC  TSLPTR+1 ;lit la TSL suivante
BOOTNXT INC  BUFADR+1 ;on charge 256 octets plus haut
        BNE  BOOTPG2  ;saute toujours
ENDBOOT JMP  CLRCOLD  ;warm start & retour
*
COUNTER DA   00
*
INITPTR LDY  #<TSL    ;buffer de TSL en $9700
        STY  TSLPTR
        LDA  #>TSL
        STA  TSLPTR+1
LOCBUF  STY  BUFADR   ;adresse du buffer de RWTS
        STA  BUFADR+1
        RTS
*
CLEANUP JSR  CLNDOS   ;reconstruit les buffers du DOS
*
SETWARM STA  TRAP     ;(A=0)
        LDA  #22
        JSR  HOME+2
        JSR  UP
        JSR  CROUT
        LDY  #<WARMDOS ;reset = $9DBF
        LDA  #>WARMDOS
SETRST  STY  SOFTEV   ;vecteur de RESET
        STA  SOFTEV+1
        EOR  #$A5
        STA  PWREDUP
        BIT  RUNLOAD
        RTS
*
END     JSR  CLEANUP
        JMP  WARMDOS
*
        LST  OFF
```

Trucs et astuces

Essayez le petit programme suivant :

```
1 PRINT "COMMENT VAS TU ?"
5 VTAB PEEK(37) :PRINT
"...YO..DE..POELE":POKE
2049,066
```

Pour l'apprécier, faire [LIST], puis [RUN] et [LIST], séparés bien entendu par les traditionnels RETURNs.

Récapitulation du programme HELLO

```
]CALL-151

*8000.8395

8000- 20 2F FB 20 58 FC 20 89
8008- FE 20 93 FE AD D5 03 C9
8010- 9D F0 06 20 2D FF 4C D0
8018- 03 20 51 A8 A9 D3 8D 00
8020- 9D A9 9C 8D 01 9D A2 00
8028- 86 19 BD 04 0C 9D A6 9A
8030- E8 D0 F7 A0 9A A9 9B 20
8038- 8C 9B A2 02 86 22 20 58
8040- FC CA 20 62 0B A0 8C B9
8048- 37 1F F0 0F 0A 90 FB 08
8050- EE 67 9B D0 03 EE 68 9B
8058- 28 D0 F1 88 D0 E9 2C 83
8060- C0 AD 00 E0 AC 01 E0 2C
8068- 81 C0 C9 4C D0 04 C0 28
8070- F0 08 C9 20 D0 09 C0 00
8078- D0 05 4C 03 09 E6 02 A0
8080- EB 84 FB A5 FB 18 69 23
8088- A8 85 FB C9 03 F0 EE A2
8090- 0B B1 01 F0 52 DD D8 0B
8098- D0 E9 C8 CA D0 F3 A4 FB
80A0- 88 88 B1 01 8D ED B7 88
80A8- B1 01 30 D7 8D EC B7 BD
80B0- E4 0B 9D AF 05 E8 E0 19
80B8- D0 F5 20 69 9B 20 19 0B
80C0- A0 FC A9 2F 20 71 9B 20
80C8- 3E 9B 84 0E 84 10 A2 30
80D0- 86 0F A9 D0 85 11 B1 0E
80D8- 91 10 C8 D0 F9 E6 0F E6
80E0- 11 CA D0 F2 20 58 FC A9
80E8- 10 85 02 D0 09 A5 06 18
80F0- 69 23 90 04 E6 02 A9 0B
80F8- A8 84 06 A5 00 C9 28 90
8100- 03 4C 8F 09 C9 14 D0 0B
8108- 85 20 85 21 20 58 FC A4
8110- 06 A5 00 0A 0A AA B1 01
8118- 30 D3 F0 66 9D 00 96 E6
8120- 00 C8 A5 02 0A 0A 0A 0A
8128- 11 01 9D 01 96 C8 98 9D
8130- 03 96 A9 FF 85 03 B1 01
8138- 30 04 46 03 46 03 29 7F
8140- 9D 02 96 A2 07 0A 0A B0
8148- 03 CA D0 FA BD A5 0B 25
8150- 03 20 87 09 A5 00 C9 1B
8158- B0 02 69 2C 69 94 20 87
8160- 09 A2 0F C8 B1 01 C9 A0
8168- B0 04 C9 80 B0 F5 20 ED
8170- FD CA D0 EF 20 8E FD 4C
8178- FA 08 20 ED FD A9 AD 4C
8180- ED FD 20 2F FB A0 0F 84
8188- 24 A9 00 85 07 20 5B FB
8190- B9 AC 0B 99 FF 03 88 D0
8198- F7 98 AE F6 B7 20 24 ED
81A0- 20 9C FC A9 15 85 24 AE
81A8- 67 9B AD 68 9B 20 24 ED
81B0- A0 0F B9 BC 0B 99 18 04
81B8- 88 10 F7 20 2F FB A9 0D
81C0- 85 24 20 9C FC A2 6A AD
81C8- 89 9B C9 BF D0 02 A2 60
81D0- 8E DD 07 A0 0D B9 CB 0B
81D8- 99 CF 07 88 D0 F7 46 04
81E0- 46 03 20 0C FD C9 A0 D0
81E8- 03 4C 9A 9B C9 AA D0 0C
81F0- A9 69 A0 FF 8D 89 9B 8C
81F8- 8B 9B D0 86 C9 DD D0 06
8200- A9 BF A0 9D D0 EE C9 95
8208- D0 0B A9 02 8D EA B7 8D
8210- 68 AA 4C 0D 08 C9 88 D0
8218- 04 A9 01 D0 EF C9 9B D0
8220- 09 85 07 A9 3F 8D DD 07
8228- D0 B4 C9 8C D0 09 85 03
8230- 29 7F 8D DD 07 D0 AB C9
8238- 84 D0 06 85 04 46 03 10
8240- EF C9 AF F0 58 C9 B0 90
8248- 99 C9 DB B0 95 C9 C1 90
8250- 02 E9 2C E9 95 C5 00 B0
8258- 89 0A 0A AA BC 00 96 BD
8260- 01 96 48 29 0F 20 FD 0B
8268- 68 4A 4A 4A 4A 09 10 85
8270- 02 BD 02 96 85 0C BD 03
8278- 96 85 01 A0 1E B1 01 99
8280- DC 07 99 74 AA 88 D0 F5
8288- 24 03 10 26 20 B7 0A B1
8290- 01 49 80 91 01 A2 02 20
8298- 62 0B 4C 0D 08 20 58 FC
82A0- A2 06 20 94 0B 20 0C FD
82A8- 30 F0 24 1E 10 6A A2 04
82B0- D0 3F 24 04 10 1D 20 B7
82B8- 0A B1 01 10 04 A2 0A D0
82C0- 30 A9 75 8D C3 B5 A9 AA
82C8- 8D C4 B5 A2 05 20 94 0B
82D0- 4C 0D 08 A5 0C D0 03 4C
82D8- C6 A5 C9 04 F0 3B 90 04
82E0- A2 0D D0 0D C9 02 F0 31
82E8- A9 20 20 B2 A5 F0 2A A2
82F0- 01 E0 04 08 20 DD FB 20
82F8- 80 FE 20 02 A7 20 84 FE
8300- 20 9C FC 20 0C FD 28 F0
8308- 91 4C 8F 09 20 E3 03 20
8310- B5 B7 90 04 A2 08 D0 D9
8318- 60 20 69 9B 20 19 0B AD
8320- 01 97 8D 00 98 A8 F0 17
8328- AD 02 97 20 FD 0B EE F1
8330- B7 20 19 0B A0 00 B9 0C
8338- 98 99 00 98 C8 D0 F7 AC
8340- 0C 97 AD 0D 97 20 FD 0B
8348- A0 00 A9 96 20 71 9B 20
8350- 19 0B 4C A6 9A 8E F4 B7
8358- 86 1E A0 00 8C EB B7 84
8360- 00 84 01 A9 10 05 02 20
8368- 71 9B A9 0F A0 11 20 FD
8370- 0B 20 19 0B EE F1 B7 CE
8378- ED B7 10 F5 AD ED C0 AD
8380- EE C0 10 02 85 1E 60 8E
8388- BB B5 A9 06 8D C1 B5 AD
8390- 68 AA 8D C0 B5 4C
*3D0G
```

Un analyseur de syntaxe

Olivier Herz

Je venais de terminer de taper ce magnifique programme d'Othello publié par une de mes revues préférées et j'avais fait plusieurs parties plus ou moins brillantes de ma part quand, au beau milieu d'une partie passionnante, la sentence tomba comme un couperet de guillotine : SYNTAX ERROR IN LINE 980. Horreur ! ! ! Je listai la ligne 980 : il manquait un deux points entre un HOME et un GOTO, ce qui m'avait échappé au cours des nombreuses relectures du programme.

Cette histoire a déjà dû se produire, de façon plus ou moins semblable, chez nombre d'entre vous. Aussi m'écriai-je alors : plus jamais ! Je m'attelai à l'ouvrage et pondis le programme d'analyse syntaxique (SNTX) ci-joint.

Ce programme répond à trois objectifs : c'est tout d'abord un utilitaire qui détecte les SYNTAX ERROR et les TYPE MISMATCH ERROR avant l'exécution d'un programme en Applesoft ; secundo, il est didactique quant à la forme : c'est un programme que j'ai voulu très structuré afin de bien illustrer la logique de la programmation et que j'ai truffé de remarques (presque une instruction REM par ligne) ; tertio, il est didactique quant au fond : son étude aide à comprendre comment fonctionne l'analyse syntaxique qui est une des trois activités fondamentales d'un interpréteur (avec la gestion des variables et celle des piles — les piles de l'Applesoft permettent de gérer les GOSUB et les FOR...NEXT).

Comment utiliser ce programme

a) Programme sur cassette

Il faut taper les instructions suivantes, le programme à étudier étant déjà chargé en mémoire :

POKE 1912, PEEK (103) : POKE 1913, PEEK (104)
POKE 1914, PEEK (175) : POKE 1915, PEEK (176)

Cela permet de cacher les pointeurs de début et de fin de programme (adresses 103 et 104, 175 et 176 respectivement) dans les mémoires 1912 à 1915 (qui semblent faire partie de la page texte, mais qui en réalité n'en font pas partie — voir le manuel de référence de l'APPLE II).

POKE 103, PEEK (175) : POKE 104, PEEK (176)
POKE PEEK (103) + PEEK (104)*256—1,0

Cela permet de charger le programme d'analyse syntaxique SNTX à la fin du programme à étudier.

LOAD charge SNTX
DEL 1,0 met en place les pointeurs de SNTX
RUN lance SNTX.

b) Programme sur disquette

L'opération ci-dessus est réalisée automatiquement par un fichier EXEC nommé SYNTAXE, fabriqué par le programme en Applesoft FAIT SYNTAXE ci-joint. La procédure à suivre est la suivante :

RUN FAIT SYNTAXE crée le fichier EXEC
LOAD *Nomprog* — *Nomprog* est le nom du programme à analyser
EXEC SYNTAXE change les pointeurs et lance SNTX.

Fonctionnement du programme SNTX

Fonctionnement général

Le programme commence par demander le numéro de la première ligne à étudier. Puis, ligne après ligne, il indique si chaque instruction est correcte ou non. SNTX s'arrête automatiquement après l'analyse de la dernière ligne du programme étudié ; il demande alors à l'utilisateur s'il veut retourner au programme initial : si non, il laisse le programme SNTX en place ; si oui, il rétablit les pointeurs du programme initial.

On peut si on le désire interrompre le programme SNTX par CTRL-C ou RESET ; pour retourner à ce moment-là au programme initial, il suffit de faire GOTO 12000, à la suite de quoi SNTX pose la question du retour.

Messages d'erreur et indications

0 CORRECT (instruction sans erreur)
1 INSTRUCTION NON RECONNUE
2 (signe ou « token » de l'Applesoft) ATTENDU
3 (variable ou fin d'instruction) ATTENDUE
4 NUMERO DE LIGNE > 63999
5 NOMBRE INCORRECT (nombre réel dans le programme)
6 EXPRESSION INCORRECTE
7 CONFUSION DE TYPE (type mismatch error)
8 INSTRUCTION AMPERSAND
9 INSTRUCTION CALL
10 FONCTION USR
11 INSTRUCTION REM
12 INSTRUCTION DATA
13 GET NUMERIQUE

Les erreurs 1 à 7, fatales à l'exécution, sont indiquées par un beep sonore.

Les messages 8, 9 et 10 indiquent des erreurs potentielles : il s'agit d'instructions qui appellent le langage machine et peuvent prendre en main la syntaxe du programme (par exemple & X HPLOT ou CALL 768,A) ; aussi, dans ce cas, SNTX émet un message d'erreur et ignore la fin de l'instruction (attention : il se peut qu'exceptionnellement &, CALL ou USR nécessitent pour leur SYNTAXE le symbole deux-points, par exemple CALL 768:A ; dans ce cas, SNTX peut indiquer une erreur à tort).

Les messages 11 et 12 indiquent une instruction un peu spéciale : la syntaxe de DATA est déterminée, uniquement à l'exécution, par une instruction READ et un REM ignore la fin de la ligne (ne pas faire REM BLABLA : PRINT A$). Alors SNTX émet le message d'erreur et ignore la fin de l'instruction pour un DATA, la fin de la ligne pour un REM.

Le message 13 indique une erreur un peu spéciale : GET A est par exemple licite en Applesoft mais si, en réponse à cette instruction, l'utilisateur tape une touche autre qu'un numéro de 0 à 9, on obtient une SYNTAX ERROR. Le GET numérique est donc à proscrire au profit de GET A$:A = VAL(A$), par exemple.

Il faut signaler que les erreurs indiquées ne sont pas toujours celles faites par l'utilisateur : par exemple A = 1B... engendrera le message FIN D'INSTRUCTION ATTENDUE, que l'utilisateur ait voulu écrire A = 1:B = 2 ou

A=1+B; de même, GET(A$(B$)) sera qualifié non pas de CONFUSION DE TYPE, mais par un GET NUMERIQUE (c'est en quelque sorte ici un « bug » de SNTX, mais il était trop compliqué de faire autrement).

Notons que SNTX relève des erreurs que l'Applesoft ne voit pas. Par exemple, GOTO 123A$ passe en Applesoft, ce qui est dû au fait que, celui-ci étant interprété, il ne regarde pas derrière 123.

Une ligne du genre 10 PRINT « BLABLA sera notée EXPRESSION INCORRECTE par SNTX alors que l'Applesoft tolère l'oubli du « quote » en fin de ligne.

L'Applesoft supporte IF A$ THEN... ou encore ON I GOTO sans numéro de ligne, ce qui ne sert à rien et est dénoncé par SNTX.

De même pour STORE A$, instruction permise par l'Applesoft alors qu'elle ne stocke sur cassette que les adresses et les longueurs des chaînes du tableau A$ et non les chaînes elles-mêmes.

Enfin, l'Applesoft ne vérifie la syntaxe de l'expression située après DEF FN F(X)= que si la fonction FN est appelée au cours du programme, par exemple PRINT FN F(2*I), alors que SNTX analyse la syntaxe de tout ce qui suit le DEF FN.

En revanche, SNTX ne signale pas les autres erreurs d'exécution, comme les OUT OF MEMORY s'il y a par exemple trop de IF THEN, FOR NEXT ou GOSUB emboîtées ; les OVERFLOW, même pour les constantes réelles dans le programme (SNTX aurait pu le faire, mais c'était trop fatiguant !) ; ou encore l'existence de plus de 88 dimensions dans un tableau (ne soyons pas ridicules !) ; enfin, tout « trafic » fait directement sur les octets du programme et qui change la « tokenisation » peut dérouter complètement SNTX.

Analyse du programme

Commençons par indiquer, à l'usage des puristes, que ce programme n'est optimisé ni en place mémoire, ni en vitesse d'exécution. En effet, pour aider la compréhension du programme, on a préféré le structurer au maximum, même si cela paraît parfois peu astucieux (sous-programmes appelés une seule fois par exemple). Nous n'indiquerons ici que les grandes lignes du programme, les nombreuses REM qu'il possède expliquant les détails.

La ligne de programme Applesoft

Indiquons en préambule comment est faite une ligne de programme Applesoft. Les deux premiers octets représentent l'adresse de la ligne suivante (octet bas, puis haut comme toujours) : si cette adresse est nulle, c'est qu'on a atteint la fin du programme.

Les deux octets suivants indiquent le numéro de ligne (de 0 à 63999 en principe — octet bas puis octet haut bien entendu).

Enfin, la ligne est représentée de manière « tokenisée » : les octets valant de 1 à 127 indiquent un caractère ASCII et ceux valant de 128 à 234 indiquent un code (« token ») de l'Applesoft (voir Pom's 4 pages 44 et 45), la fin de ligne étant indiquée par le code 0.

Le programme

Il commence par un saut à la ligne 20000 pour initialiser le tableau des erreurs, demander le numéro de la première ligne à étudier et la chercher, en retrouvant l'adresse du début du programme à étudier dans les mémoires 1912 et 1913.

Puis il saute en 10000 pour débuter l'analyse : il indique le numéro de la ligne et celui de l'instruction dans la ligne puis, en fonction du « token » de l'instruction, il saute vers une des lignes 3000 à 3500 (étude des « tokens ») avant d'écrire le message d'erreur (ligne 100 à 150) et de passer à la prochaine instruction (ligne 10020).

Pour l'étude des « tokens », on utilise des petites routines situées avant la ligne 1000.

Le cœur de l'étude syntaxique est constitué par l'analyse des expressions : il s'agit des routines débutant en 600 (nom de variable), 1000 (expression numérique) et 2000 (expression chaîne), qui s'appellent les unes les autres : et oui ! le BASIC est récursif, le problème étant qu'il ne possède pas de variables locales, difficulté contournée par l'utilisation de tableaux (VAR, A1 et A2) et d'une pile (I, I1 et I2) notant l'enfoncement dans la récursion (lignes 670, 1330 et 1620).

Notons l'importance de la variable ERR dans ces routines : dès qu'une erreur est trouvée, ERR n'est plus nul et l'on sort immédiatement des sous-programmes récursifs par les multiples issues de secours IF ERR THEN RETURN.

Après le retour au niveau général du programme (lignes 3000 et suivantes), l'analyse est arrêtée en cas d'erreur par le sous-programme de la ligne 20.

Notons qu'en fin d'analyse, le programme saute à la ligne 12000 et, le cas échéant, retourne au programme initial en restaurant ses pointeurs qui étaient cachés dans les mémoires 1912 à 1915.

Les variables

PTR : pointeur d'adresses qui indique la progression dans le programme étudié.
A : contient la valeur de l'octet indiqué par le pointeur (ligne 10).
ERR : numéro de l'erreur trouvée.
ES$(ERR) : nom de cette erreur.
FINI : variable indiquant la fin d'une instruction (ligne 30 : A=0 ou A=58 donne FINI=1).
TKN : indique la présence d'un (ou de plusieurs) « token » ou caractère à l'adresse pointée par PTR.
LN : numéro de la ligne étudiée.
NL : numéro de la prochaine ligne.
IST : numéro de l'instruction dans la ligne (ligne 10000 et suivantes).
BEG : numéro de la première ligne demandée par l'utilisateur (ligne 20040).
A$: sert pour les INPUT et les GET.
B : est un code et B$ un nom de « token » ou de caractère pour les messages d'erreurs (lignes 100, 400 et 900).
US : indique la présence de la fonction USR (ligne 1310).
PNT : indique la présence d'un point dans un réel (ligne 800).
NBR : valeur d'un numéro de ligne.
LNG : longueur d'un numéro de ligne (ligne 500).
INF, SUP, EGA et COM sont utilisés pour la détection des opérateurs de comparaison.
VAR et VAR(I) : notent le type d'une variable (lignes 600 à 790).
ADR, A1(I1) et A2(I2) : représentent un stockage provisoire de PTR (lignes 1330, 1620 et 3730).

Améliorations du programme

Je pense avoir pris en compte, à défaut de la totalité, au moins 99 % des cas de figures qui peuvent se présenter dans un programme Applesoft. J'espère que SNTX les traite tous correctement : en effet, le programme SNTX a beau avoir été testé sur de nombreux programmes Applesoft, il

est loin d'avoir eu affaire à tous ces cas de figure. Aussi, j'espère qu'il ne reste pratiquement plus de « bugs ».

En ce qui concerne les limitations de la vitesse d'exécution et de la taille mémoire dont nous avons parlé ci-dessus, il est conseillé d'utiliser une version sans REM du programme SNTX (obtenue avec le CRUNCHER de DAKIN ou AOPTIMIZER). Sans les REM, on a la possibilité d'analyser la syntaxe de programmes Applesoft ayant jusqu'à 110 secteurs sur disquette (ou faisant moins de 28K environ, ce qui est quand même considérable — un programme trop gros aboutit au message PROGRAM TOO LARGE au moment du chargement de SNTX). La version sur cassette tolère des programmes de 40K au plus.

Enfin, nous fournissons sur la disquette de Pom's le résultat de la compilation de SNTX par le compilateur TASC : il s'agit du fichier binaire SNTX COMPILE. Son utilisation est un peu différente de celle de SNTX : il suffit d'avoir le programme à étudier en mémoire, puis de faire BLOAD SNTX COMPILE et CALL 29869 (attention : 29869 n'est pas l'adresse de début du fichier SNTX COMPILE). La suite des opérations est la même que pour SNTX, à la différence près que SNTX COMPILE efface le programme étudié quand il a fini (c'est dû à TASC).

On peut ainsi étudier la syntaxe de programmes dont l'adresse de fin, indiquée par PRINT PEEK(175) + PEEK(176)*256, est inférieure à 25856 (donc des programmes ayant jusqu'à 95 secteurs ou faisant jusqu'à 23K environ). Notons que SNTX COMPILE n'utilise pas d'adresse supérieure à 36820, donc peut être utilisé tant que la HIMEM reste supérieure à ce nombre (en particulier avec PLE qui abaisse la HIMEM à 36864).

```
4  REM
**********************
*   PROGRAMME SNTX:   *
* ANALYSE SYNTAXIQUE  *
* D'UN PROGRAMME EN   *
* BASIC APPLESOFT     *
**********************
5  GOTO 20000: REM INITIALISATION
8  REM
**********************
* ROUTINES COURANTES  *
**********************
9  REM ---INCREMENTE LE POINTEUR      ET
       LIT LE PROGRAMME---
10 A =  PEEK (PTR):PTR = PTR + 1: RETURN

19  REM ---TESTE SI ERREUR---
20  IF ERR THEN  POP : GOTO 100
25  RETURN
29  REM ---TESTE FIN DE LIGNE---
30 FINI = 0:A =  PEEK (PTR): IF A = 0 OR
       A = 58 THEN FINI = 1
35  RETURN
39  REM ---TESTE SI CHIFFRE---
40 TKN = 0:A =  PEEK (PTR): IF A )  = 48
       AND A (  = 57 THEN  GOSUB 10:TKN
       = 1
45  RETURN
49  REM ---TESTE SI LETTRE---
50 TKN = 0:A =  PEEK (PTR): IF A )  = 65
       AND A (  = 90 THEN  GOSUB 10:TKN
       = 1
55  RETURN
59  REM ---TESTE (-) OU (+)---
60 TKN = 0:A =  PEEK (PTR): IF A = 200 O
       R A = 201 THEN  GOSUB 10:TKN = 1
65  RETURN
69  REM ---TESTE SI OPERATEUR---
70 TKN = 0:A =  PEEK (PTR): IF A )  = 20
       0 AND A (  = 204 THEN  GOSUB 10:T
       KN = 1: REM + - * / ET ^
72  RETURN
75 TKN = 0:A =  PEEK (PTR): IF A = 205 O
       R A = 206 THEN  GOSUB 10:TKN = 1:
        REM AND ET OR
77  RETURN
79  REM ---TESTE (-) OU (,)---
80 TKN = 0:A =  PEEK (PTR): IF A = 44 OR
       A = 201 THEN  GOSUB 10:TKN = 1
82  RETURN

89  REM ---TESTE GOTO-GOSUB---
90  GOSUB 10: IF A (  ) 171 AND A (  ) 1
       76 THEN ERR = 2:B$ = "GOTO OU GOS
       UB": POP : GOTO 100
95  RETURN
99  REM ---SAUT->FIN D'INSTR.---
100  PRINT " ";: IF ERR = 2 OR ERR = 3 T
       HEN  PRINT B$;" ";
110  IF ERR ( 8 AND ERR (  ) 0 THEN  PRI
       NT  CHR$ (7);: REM ERREURS FATALE
       S -> BIP
120  PRINT ER$(ERR): REM MESSAGE D'ERREU
       R
130 PTR = PTR - 1: REM PARFOIS NECESSAIR
       E
140  GOSUB 30: GOSUB 10: IF  NOT FINI TH
       EN 140: REM FIN D'INSTRUCTION?
150  GOTO 10020
199  REM ---TEST COMPARATEURS---
200 SUP = 0:INF = 0:EGA = 0
210  GOSUB 10: IF A = 207 THEN SUP = SUP
       + 1: IF SUP ) 1 THEN ERR = 6: RE
       TURN : REM )
220  IF A = 208 THEN EGA = EGA + 1: IF E
       GA ) 1 THEN ERR = 6: RETURN : REM
       =
230  IF A = 209 THEN INF = INF + 1: IF I
       NF ) 1 THEN ERR = 6: RETURN : REM
       (
240  IF A )  = 207 AND A (  = 209 THEN 2
       10: REM ) = OU (
250 PTR = PTR - 1:COM = SUP + EGA + INF:
       RETURN
299  REM ---ATTEND FIN D'INS.---
300  GOSUB 30: IF  NOT FINI THEN ERR = 3
       :B$ = "FIN D'INSTRUCTION"
310  GOTO 100
399  REM ---ATTEND LE TOKEN B---
400 B = 41:B$ = ")": GOTO 450
405 B = 44:B$ = ",": GOTO 450
410 B = 59:B$ = ";": GOTO 450
415 B = 40:B$ = "(": GOTO 450
420 B = 193:B$ = "TO": GOTO 450
425 B = 194:B$ = "FN": GOTO 450
430 B = 196:B$ = "THEN": GOTO 450
435 B = 197:B$ = "AT": GOTO 450
440 B = 208:B$ = "=": GOTO 450
450  GOSUB 10: IF A (  ) B THEN ERR = 2
460  RETURN
499  REM ---LIT NR. DE LIGNE---
```

```
500 NBR = 0:LNG = 0
510  GOSUB 40: IF  NOT TKN THEN 550: REM
       CHIFFRE
520 LNG = LNG + 1: IF LNG ) 5 THEN ERR =
       4: RETURN : REM TROP LONG
530 NBR = NBR * 10 + A - 48: IF NBR ) 63
       999 THEN ERR = 4: RETURN : REM NU
       MERO TROP GRAND
540  GOTO 510
550  IF LNG = 0 THEN ERR = 2:B$ = "NUMER
       O DE LIGNE": REM PAS DE NUMERO
560  RETURN
599  REM ---LIT NOM DE VAR.---
600  GOSUB 700: GOTO 660: REM PREND VARI
       ABLE REELLE
610  GOSUB 710: GOTO 660: REM PREND VARI
       ABLE NUMERIQUE
620  GOSUB 720: GOTO 660: REM PREND VARI
       ABLE CHAINE
650  GOSUB 750: REM VARIABLE QUELCONQUE
660 VAR(I) = VAR: IF ERR THEN  RETURN
665  GOSUB 900: IF  NOT TKN THEN  RETURN
       : REM (
670 I = I + 1:VAR(I) = 0: GOSUB 1000:I =
       I - 1: IF ERR THEN  RETURN : REM
       EXPRESSION DE L'INDICE
680  GOSUB 905: IF TKN THEN 670: REM  V
       IRGULE
690  GOTO 400: REM )
699  REM ---LIT UN NOM DE TABLEAU OU DE
       VARIABLE SS INDEX---
700  GOSUB 750:ERR = (VAR ( ) 0) * 7: R
       ETURN : REM PREND VARIABLE REELLE
710  GOSUB 750:ERR = (VAR = 2) * 7: RETU
       RN : REM PREND VARIABLE NUMERIQUE
720  GOSUB 750:ERR = (VAR ( ) 2) * 7: R
       ETURN : REM PREND VARIABLE CHAINE
750  GOSUB 50: IF  NOT TKN THEN ERR = 3:
       B$ = "VARIABLE": RETURN : REM PRE
       MIERE LETTRE
760  GOSUB 40: IF TKN THEN 760: REM CHIF
       FRE
770  GOSUB 50: IF TKN THEN 760: REM LETT
       RE
780 VAR = (A = 37) + 2 * (A = 36): IF VA
       R ( ) 0 THEN  GOSUB 10: REM VAR=
       0 SI VARIABLE REELLE, 1 SI ENTIER
       E ET 2 SI CHAINE
790  RETURN
799  REM ---LIT UN NOMBRE---
800 PNT = 0
805 PNT = PNT + 1: IF PNT ) 2 THEN ERR =
       5: RETURN : REM 1 SEUL POINT DEC
       IMAL PAR NOMBRE!
810  GOSUB 920: IF TKN THEN 850: REM EXP
       OSANT
820  GOSUB 915: IF TKN THEN 805: REM POI
       NT
830  GOSUB 40: IF TKN THEN 810: REM CHIF
       FRE
840  RETURN
850 LNG = 0: GOSUB 60: REM SIGNE
860  GOSUB 40: IF  NOT TKN THEN  RETURN
       : REM CHIFFRE
870 LNG = LNG + 1: IF LNG (  = 2 THEN 86
       0: REM EXPOSANT D'AU PLUS 2 CHIFF
       RES
880 ERR = 5: RETURN
899  REM ---TOKEN FACULTATIF---
900 B = 40: GOTO 980: REM (
905 B = 44: GOTO 980: REM ,
910 B = 34: GOTO 980: REM "
915 B = 46: GOTO 980: REM POINT
920 B = 69: GOTO 980: REM E
925 B = 171: GOTO 980: REM GOTO
930 B = 199: GOTO 980: REM STEP
935 B = 193: GOTO 980: REM TO
940 B = 197: GOTO 980: REM AT
945 B = 200: GOTO 980: REM +
950 B = 198: GOTO 980: REM NOT
980 TKN = 0:A =  PEEK (PTR): IF A = B TH
       EN  GOSUB 10:TKN = 1
990  RETURN
999  REM
**********************
*EXPRESS. NUMERIQUE *
**********************
1000  GOSUB 60: IF TKN THEN 1000: REM SI
       GNE
1005  GOSUB 950: IF TKN THEN 1000: REM N
       OT
1010  GOSUB 10: IF A = 40 THEN 1040: REM
       (
1015  IF A ( ) 194 THEN 1100
1019  REM --- FN ---
1020  GOSUB 700: IF ERR THEN  RETURN : R
       EM FONCTION
1030  GOSUB 415: IF ERR THEN  RETURN : R
       EM (
1040  GOSUB 1000: IF ERR THEN  RETURN :
       REM ARGUMENT
1050  GOSUB 400: IF ERR THEN  RETURN : R
       EM )
1060  GOTO 1900
1100  IF A ( ) 215 THEN 1200
1109  REM --- SCRN( ---
1110  GOSUB 1000: IF ERR THEN  RETURN :
       REM EXPRESSION 1
1120  GOSUB 405: IF ERR THEN  RETURN : R
       EM VIRGULE
1130  GOTO 1040: REM EXPRESSION 2 ET )
1135  IF ERR THEN  RETURN
1200  IF A ( ) 227 AND A ( ) 229 AND A
       ( ) 230 THEN 1300
1209  REM --- LEN, VAL ET ASC ---
1210  GOSUB 415: IF ERR THEN  RETURN : R
       EM (
1220  GOSUB 2000: IF ERR THEN  RETURN :
       REM EXPRESSION CHAINE
1230  GOTO 1050: REM )
1300  IF A ( ) 213 AND A ( ) 214 AND A
       ( ) 217 THEN 1400
1309  REM --- USR, FRE ET POS ---
```

```
1310 US = 0: IF A = 213 THEN US = 1
1320  GOSUB 415: IF ERR THEN  RETURN : R
      EM (
1330 A2(I2) = PTR:I2 = I2 + 1: GOSUB 100
      0:I2 = I2 - 1: IF ERR = 0 THEN 13
      50: REM ARGUMENT NUMERIQUE
1335  IF ERR ( ) 7 THEN  RETURN
1340 PTR = A2(I2):ERR = 0: GOSUB 2000: I
      F ERR ( ) 0 THEN  RETURN : REM A
      RGUMENT CHAINE
1350  GOSUB 400: IF ERR THEN  RETURN : R
      EM )
1360  IF US THEN ERR = 10: RETURN : REM
      ERREUR FORCEE
1370  GOTO 1900
1400  IF A ( 210 OR A ) 226 THEN 1500
1409  REM --- SGN, INT, ABS, PDL,     SQR
      , RND, LOG, EXP, COS,     SIN, TAN
      , ATN, PEEK ---
1410  GOTO 1030: REM (ARGUMENT)
1500  IF A ( ) 46 AND (A ( 48 OR A ) 57
      ) THEN 1600
1509  REM --- NOMBRE ---
1510 PTR = PTR - 1: GOSUB 800: IF ERR TH
      EN  RETURN
1520  GOTO 1900
1600 PTR = PTR - 1: IF A = 34 OR A = 228
       OR (A )  = 231 AND A (  = 234) T
      HEN 1700: REM DEBUT D'EXPRESSION
      CHAINE
1610  IF A ( 65 OR A ) 90 THEN ERR = 6:
      RETURN
1619  REM --- VARIABLE ---
1620 A1(I1) = PTR:I1 = I1 + 1: GOSUB 650
      :I1 = I1 - 1: IF ERR THEN  RETURN

1630  IF VAR(I) = 2 THEN PTR = A1(I1): G
      OTO 1700: REM DEBUT D'EXPRESSION
      CHAINE
1640  GOTO 1900
1699  REM --- COMPAR.CHAINES ---
1700  GOSUB 2000: IF ERR THEN  RETURN :
      REM EXPRESSION CHAINE
1710  GOSUB 200: IF COM = 0 THEN ERR = 7
      : RETURN : REM ) ( OU =
1720  GOSUB 2000: IF ERR THEN  RETURN :
      REM EXPRESSION CHAINE
1899  REM --- OPERATEURS ---
1900  GOSUB 200: IF COM THEN 1000: REM (
       ) OU =
1910  GOSUB 70: IF TKN THEN 1000: REM +
      - * / OU ^
1920  GOSUB 75: IF TKN THEN 1000: REM AN
      D OU OR
1930  RETURN
1999  REM
**********************
* EXPRESSION CHAINE *
**********************
2000  GOSUB 945: IF TKN THEN 2000: REM +
2010  GOSUB 10: IF A = 46 OR (A ) = 48
      AND A (  = 57) OR A = 194 OR A =
      201 OR (A )  = 210 AND A (  = 230
       AND A (  ) 228) THEN ERR = 7: RE
      TURN : REM TYPE CHAINE?
2020  IF A (  ) 228 AND A (  ) 231 THEN
      2100
2029  REM --- STR$ ET CHR$ ---
2030  GOSUB 415: IF ERR THEN  RETURN : R
      EM (
2040  GOSUB 1000: IF ERR THEN  RETURN :
      REM ARGUMENT
2050  GOSUB 400: IF ERR THEN  RETURN : R
      EM )
2060  GOTO 2900
2100  IF A (  ) 232 AND A (  ) 233 THEN
      2200
2109  REM --- LEFT$ ET RIGHT$ ---
2110  GOSUB 415: IF ERR THEN  RETURN : R
      EM (
2120  GOSUB 2000: IF ERR THEN  RETURN :
      REM EXPRESSION CHAINE
2130  GOSUB 405: IF ERR THEN  RETURN : R
      EM VIRGULE
2140  GOTO 2040: REM ARGUMENT ET )
2200  IF A (  ) 234 THEN 2300
2209  REM --- MID$ ---
2210  GOSUB 415: IF ERR THEN  RETURN : R
      EM (
2220  GOSUB 2000: IF ERR THEN  RETURN :
      REM EXPRESSION CHAINE
2230  GOSUB 405: IF ERR THEN  RETURN : R
      EM VIRGULE
2240  GOSUB 1000: IF ERR THEN  RETURN :
      REM EXPRESSION 1
2250  GOSUB 905: IF TKN THEN 2040: REM A
      RGUMENT ET ) SI VIRGULE
2260  GOTO 2050: REM )
2300  IF A ( 65 OR A ) 90 THEN 2400
2309  REM --- VARIABLE ---
2310 PTR = PTR - 1: GOSUB 620: IF ERR TH
      EN  RETURN
2320  GOTO 2900
2400  IF A (  ) 34 THEN 2500
2409  REM --- CHAINE ---
2410  GOSUB 10: IF A = 0 THEN ERR = 6: R
      ETURN
2420  IF A (  ) 34 THEN 2410: REM FIN DE
       LA CHAINE?
2430  GOTO 2900
2500  IF A (  ) 40 THEN ERR = 6: RETURN
2509  REM --- ( ---
2510  GOSUB 2000: IF ERR THEN  RETURN :
      REM  EXPRESSION CHAINE
2520  GOTO 2050: REM )
2899  REM --- OPERATEURS ---
2900  GOSUB 945: IF TKN THEN 2000: REM +
2910  GOSUB 70: IF TKN THEN ERR = 7: RET
      URN : REM - * / ET ^
2920  RETURN
2999  REM
**********************
* ETUDE DES TOKENS  *
**********************
3099  REM --- FOR..TO ---
3100  GOSUB 700: GOSUB 20: REM VARIABLE
      REELLE SANS INDEX
3110  GOSUB 440: GOSUB 20: REM =
3120  GOSUB 1000: GOSUB 20: REM EXPRESSI
      ON 1
3130  GOSUB 420: GOSUB 20: REM TO
3140  GOSUB 1000: GOSUB 20: REM EXPRESSI
      ON 2
3150  GOSUB 930: IF TKN THEN 3800: REM S
      TEP
3160  GOTO 300
3199  REM --- NEXT ---
3200  GOSUB 30: IF FINI THEN 100: REM PA
      S DE NOM DE VARIABLE
3210  GOSUB 700: GOSUB 20: REM VARIABLE
```

```
       REELLE SANS INDEX
3220   GOSUB 905: IF TKN THEN 3210: REM V
       IRGULE
3230   GOTO 300
3299   REM --- DATA ---
3300 ERR = 12: GOTO 100: REM ERREUR FORC
       EE
3399   REM --- INPUT ---
3400   GOSUB 910: IF  NOT TKN THEN 3450:
       REM "
3410   GOSUB 10: IF A = 0 THEN ERR = 3:B$
       = "VARIABLE": GOTO 100
3420   IF A ( ) 34 THEN 3410: REM FIN DE
       LA CHAINE?
3430   GOSUB 410: GOSUB 20: REM POINT-VIR
       GULE
3449   REM --- READ ---
3450   GOSUB 650: GOSUB 20: REM VARIABLE
3460   GOSUB 905: IF TKN THEN 3450: REM V
       IRGULE
3470   GOTO 300
3499   REM --- DEL ---
3500   GOSUB 500: GOSUB 20: REM LIGNE 1
3510   GOSUB 405: GOSUB 20: REM VIRGULE
3520   GOTO 4450: REM LIGNE 2
3599   REM --- DIM ---
3600   GOSUB 750: GOSUB 20: REM TABLEAU
3610   GOSUB 415: GOSUB 20: REM (
3620   GOSUB 1000: GOSUB 20: REM DIMENSIO
       N
3630   GOSUB 905: IF TKN THEN 3620: REM V
       IRGULE
3640   GOSUB 400: GOSUB 20: REM )
3650   GOSUB 905: IF TKN THEN 3600: REM V
       IRGULE
3660   GOTO 300
3699   REM --- PRINT ---
3700   GOSUB 30: IF FINI THEN 100: REM FI
       N D'INSTRUCTION
3710   GOSUB 10: IF A = 44 OR A = 59 THEN
        3700: REM VIRGULE OU POINT-VIRGU
       LE
3720   IF A = 192 OR A = 195 THEN  GOSUB
       1000: GOSUB 20: GOSUB 400: GOSUB
       20: GOTO 3700: REM SPC( ET TAB(
3730 PTR = PTR - 1:ADR = PTR: GOSUB 1000
       : IF ERR = 7 THEN 3750: REM EXPRE
       SSION NUMERIQUE
3740   GOSUB 20: GOTO 3700
3750 PTR = ADR:ERR = 0: GOSUB 2000: GOSU
       B 20: GOTO 3700: REM EXPRESSION C
       HAINE
3799   REM --- SPEED=, HCOLOR=, IN#,  COLOR
       =, HIMEM:, LOMEM:, VTAB,  HTAB, SCAL
       E=, ROT=, PR# ---
3800   GOSUB 1000: GOSUB 20: GOTO 300: RE
       M EXPRESSION UNIQUE
3899   REM --- CALL ---
3900   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON
3910 ERR = 9: GOTO 100: REM ERREUR FORCE
       E
3999   REM --- PLOT ET POKE ---
4000   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON 1
4010   GOSUB 405: GOSUB 20: REM VIRGULE
4020   GOTO 3800: REM EXPRESSION 2
4099   REM --- HLIN ET VLIN ---
4100   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON 1
4110   GOSUB 405: GOSUB 20: REM VIRGULE
4120   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON 2
4130   GOSUB 435: GOSUB 20: REM AT
4140   GOTO 3800: REM EXPRESSION 3
4199   REM --- DRAW ET XDRAW ---
4200   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON 1
4210   GOSUB 940: IF  NOT TKN THEN 300: R
       EM AT
4220   GOTO 4000: REM EXPRESSIONS 2 ET 3
4299   REM --- HPLOT ---
4300   GOSUB 935: REM TO
4310   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON 1
4320   GOSUB 405: GOSUB 20: REM VIRGULE
4330   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON 2
4340   GOSUB 935: IF TKN THEN 4310: REM T
       O
4350   GOTO 300
4399   REM --- ONERR ---
4400   GOSUB 90: REM GOTO OU GOSUB
4449   REM --- GOTO ET GOSUB ---
4450   GOSUB 500: GOSUB 20: REM LIGNE
4460   GOTO 300
4499   REM --- STORE ET RECALL---
4500   GOSUB 750: GOSUB 20: REM TABLEAU
4510   IF VAR = 2 THEN ERR = 7: GOTO 100:
        REM TABLEAU NUMERIQUE SEULEMENT
4520   GOTO 300
4599   REM --- LET ---
4600   GOSUB 650: GOSUB 20: REM VARIABLE
4610   GOSUB 440: GOSUB 20: REM =
4620   IF VAR(0) ( 2 THEN  GOSUB 1000: GO
       SUB 20: GOTO 300: REM  EXPRESSION
        NUMERIQUE
4630   IF VAR(0) = 2 THEN  GOSUB 2000: GO
       SUB 20: GOTO 300: REM EXPRESSION
       CHAINE
4699   REM --- RUN ---
4700   GOSUB 30: IF FINI THEN 100: REM PA
       S DE LIGNE
4710   GOTO 4450: REM LIGNE
4799   REM --- IF..THEN ---
4800   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON
4810   GOSUB 925: IF TKN THEN 4450: REM G
       OTO
4820   GOSUB 430: GOSUB 20: REM THEN
4830   GOSUB 40: IF TKN THEN PTR = PTR -
       1: GOTO 4450: REM NUMERO DE LIGNE
4840   GOTO 10015: REM SUITE
4899   REM --- AMPERSAND ---
4900 ERR = 8: GOTO 100: REM ERREUR FORCE
       E
4999   REM --- REM ---
5000 ERR = 11: PRINT " ";ER$(ERR):PTR =
       NL: GOTO 10000: REM ERREUR FORCEE

5099   REM --- ON..GOTO,
         ON..GOSUB ---
5100   GOSUB 1000: GOSUB 20: REM EXPRESSI
       ON
5110   GOSUB 90: REM GOTO OU GOSUB
5120   GOSUB 500: GOSUB 20: REM LIGNE
5130   GOSUB 905: IF TKN THEN 5120: REM V
       IRGULE
5140   GOTO 300
5199   REM --- WAIT ---
```


```
5200  GOSUB 1000: GOSUB 20: REM EXPRESSI
      ON 1
5210  GOSUB 405: GOSUB 20: REM VIRGULE
5220  GOSUB 1000: GOSUB 20: REM EXPRESSI
      ON 2
5230  GOSUB 905: IF TKN THEN 3800: REM E
      XPRESSION 3 SI VIRGULE
5240  GOTO 300
5299  REM --- DEF ---
5300  GOSUB 425: GOSUB 20: REM FN
5310  GOSUB 700: GOSUB 20: REM VARIABLE
      REELLE SANS INDEX
5320  GOSUB 415: GOSUB 20: REM (
5330  GOSUB 700: GOSUB 20: REM VARIABLE
      REELLE SANS INDEX
5340  GOSUB 400: GOSUB 20: REM )
5350  GOSUB 440: GOSUB 20: REM -
5360  GOSUB 1000: GOSUB 20: REM EXPRESSI
      ON
5370  GOTO 300
5399  REM --- LIST ---
5400  GOSUB 30: IF FINI THEN 100: REM PA
      S DE NUMERO DE LIGNE
5410  GOSUB 80: IF TKN THEN 5450: REM VI
      RGULE OU TIRET
5420  GOSUB 500: GOSUB 20: REM LIGNE 1
5430  GOSUB 80: IF  NOT TKN THEN 300: RE
      M VIRGULE OU TIRET
5450  GOSUB 30: IF FINI THEN 100: REM PA
      S DE LIGNE 2
5460  GOTO 4450: REM LIGNE 2
5499  REM --- GET ---
5500  GOSUB 620:ERR = 6 * (ERR = 7) + ER
      R: GOSUB 20: REM VARIABLE NUMERIQ
      UE CONSEILLEE (SI ERR=7 ALORS ERR
      =13)
5510  GOSUB 905: IF TKN THEN 5500: REM
      VIRGULE
5520  GOTO 300
9999  REM
**********************
* DEBUT D'UNE LIGNE *
**********************
10000 IST = 0: GOSUB 10:NL = A: GOSUB 10
      :NL = A * 256 + NL: IF NL = 0 THE
      N 12000
10003  GOSUB 10:LN = A: GOSUB 10:LN = A
       * 256 + LN
10006  INVERSE : PRINT " LIGNE ";LN;" ":
       NORMAL : REM LIGNE
10010 IST = IST + 1: INVERSE : PRINT IST
      ;: NORMAL : REM INSTRUCTION
10015  GOSUB 10: REM PREMIER OCTET DE L'
       INSTRUCTION
10020  IF A = 58 THEN 10010: REM FIN D'I
       NSTRUCTION
10030  IF A = 0 THEN 10000: REM FIN DE L
       IGNE
10040  IF (A )  = 65 AND A (  = 90) THEN
        A = 170:PTR = PTR - 1: REM LET I
       MPLICITE
10050 ERR = 0: IF A ( 128 OR A )  = 192
       THEN ERR = 1: GOTO 100
10060 I = 0:I1 = 0:I2 = 0: REM POINTEURS
       DES PILES DE VAR, A1 ET A2
10999  REM ---EN VOITURE SIMONE!
11000  ON A - 127 GOTO 300,3100,3200,330
       0,3400,3500,3600,3450,300,300,380
       0,3800,3900,4000,4100,4100
11010  ON A - 143 GOTO 300,300,3800,4300
       ,4200,4200,3800,300,3800,3800,300
       ,300,300,300,300,300
11020  ON A - 159 GOTO 3800,300,3800,380
       0,3800,4400,300,4500,4500,3800,46
       00,4450,4700,4800,300,4900
11030  ON A - 175 GOTO 4450,300,5000,300
       ,5100,5200,300,300,5300,4000,3700
       ,300,5400,300,5500,300
11999  REM
**********************
* FIN DE L'ANALYSE  *
**********************
12000  PRINT : PRINT "VOULEZ-VOUS REVENI
       R AU PROGRAMME INITIAL?";
12010  GET A$: IF A$ (  ) "O" AND A$ (
       ) "N" THEN 12010
12020  PRINT A$: IF A$ = "O" THEN  POKE
       103, PEEK (1912): POKE 104, PEEK
       (1913): POKE 175, PEEK (1914): PO
       KE 176, PEEK (1915): REM RESTAURE
        LES POINTEURS
12030  END
19999  REM
**********************
*   INITIALISATION  *
**********************
20000  TEXT : HOME : VTAB 8: HTAB 5
20010  PRINT "PROGRAMME D'ANALYSE SYNTAX
       IQUE": PRINT : PRINT
20020  HTAB 12: PRINT "PAR OLIVIER HERZ"
       : PRINT
20030  HTAB 9: PRINT "(C) POM'S OCTOBRE
       1982": PRINT : PRINT : PRINT
20040  PRINT : PRINT "LIGNE DE DEPART ";
       : POKE  - 16368,0: INPUT A$:BEG =
        VAL (A$)
20050  DIM A1(10),A2(10),VAR(10)
20099  REM  ---INITIALISE LE          TAB
       LEAU DES ERREURS---
20100  DIM ER$(13): FOR ERR = 0 TO 13: R
       EAD ER$(ERR): NEXT
20110  DATA CORRECT, INSTRUCTION NON RECO
       NNUE, ATTENDU, ATTENDUE, NUMERO DE L
       IGNE ) 63999, NOMBRE INCORRECT, EXP
       RESSION INCORRECTE
20120  DATA CONFUSION DE TYPE, INSTRUCTIO
```

B.I.P. *Information*

POUR VOTRE APPLE II

A - CARTES MEMOIRES LEGEND U.S.A.

en promotion : Carte 128 K - FF. 5025 HT
Carte 64 K - FF. 3050 HT

Mémoire supplémentaire et Vitesse accélérée.

Ces Cartes Interfaces comportent :

— La carte Mémoire Interface.

— Le Manuel d'Utilisation. (Français en option).

— Les programmes suivants ouverts et d'utilisation facile :

- V.C. Plus, Offre 82 K ou 136 K mémoires (80 colonnes en option) Utilisables pour Visicalc.
- Disque émulateur, simule Drive d'accès rapide DOS (CP/M ou Pascal en option.)
- Sélecteur de Diapositives, qui donne instantanément 16 images par Carte 128 K
- Mémory Master, qui libère 10 K de l'Apple ; Flipps DOS 3.3. et 3.2.
- Mémory Test, pour tester les mémoires.
- Hires Demo, et autres.

Les Cartes d'extension Mémoires LEGEND sont fiables et Garanties 1 an.

B - LOGICIEL GRAPHIQUE ARTISTIQUE

en promotion : F.F. 650 HT

CEEMAC langage graphique et un album. « ORGUE DE FEU » ou « SPARKEE » à votre choix.

Pour le plaisir de vos yeux, des images mouvementées et interactives, jamais exactement les mêmes. Suite d'images « SCORES » créées par d'autres (Album) ou par vous mêmes (CEEMAC).

B.I.P.
22, rue Joseph Dijon - 75018 PARIS
Tél. 255 44 63


```
      N AMPERSAND, INSTRUCTION CALL, FONC
      TION USR, INSTRUCTION REM, INSTRUCT
      ION DATA, GET NUMERIQUE
20199 REM ---CHERCHE LA                LIG
      NE DE DEPART---
20200 PTR =  PEEK (1912) +  PEEK (1913)
      * 256: REM DEBUT (STOCKE) DU PROG
      RAMME A ETUDIER
20210 GOSUB 10:NL = A: GOSUB 10:NL = A
      * 256 + NL: IF NL = 0 THEN 12000:
       REM DEBUT DE LIGNE
20220 GOSUB 10:LN = A: GOSUB 10:LN = A
      * 256 + LN: IF LN )  = BEG THEN 1
      0006: REM BON NUMERO DE LIGNE
20230 PTR = NL: GOTO 20210: REM SAUT->FI
      N DE LA LIGNE
```

```
9  REM ************************
       * FABRIQUE LE FICHIER *
       * EXEC QUI CHARGE SNTX*
       * AU-DESSUS DU PRO-    *
       * GRAMME A ETUDIER     *
       ************************
10 D$ =  CHR$ (13) +  CHR$ (4):N$
     = "SYNTAXE": REM NOM DU FIC
     HIER EXEC
19  REM   ECRITURE DU FICHIER
20  PRINT D$"OPEN"N$D$"DELETE"N$D
    $"OPEN"N$D$"WRITE"N$: REM FO
    RME COMPACTEE DE L'OUVERTURE
     DU FICHIER
29  REM
103,104: DEBUT PROGRAMME
175,176: FIN PROGRAMME
30  PRINT "POKE 1912,PEEK(103):PO
    KE 1913,PEEK(104)"
40  PRINT "POKE 1914,PEEK(175):PO
    KE 1915,PEEK(176)": REM STOC
    KAGE DES POINTEURS DU PROGRA
    MME A ETUDIER
50  PRINT "POKE 103,PEEK(175):POK
    E 104,PEEK(176)"
60  PRINT "POKE PEEK(103)+PEEK(10
    4)*256-1,0": REM DEBUT DE SN
    TX=FIN DU PROGRAMME INITIAL
70  PRINT "LOAD SNTX": REM ON CHA
    RGE SNTX
80  PRINT "DEL 1,0": REM ON MET L
    ES AUTRES POINTEURS EN PLACE

90  PRINT "RUN";: REM ON LANCE SN
    TX
100  PRINT D$"CLOSE"N$D$"LOCK"N$:
     REM FORME COMPACTEE DE LA F
    ERMETURE DU FICHIER
65535  REM ********************
           *  OCTOBRE  1982  *
           *(C) OLIVIER HERZ *
           *     POUR POM'S  *
           ********************
```

Tortue Ampersand

Jacques Duma

La tortue est une petite bête qui se déplace sur l'écran avec une plume pour faire des dessins. Elle obéit à deux ordres de base : AVANCE et TOURNE. Pour pouvoir la programmer et profiter du BASIC Applesoft, de nouvelles instructions ont été créées grâce à l'ampersand (&). Ainsi, sans perdre les instructions HPLOT ou DRAW, on peut, après avoir fait BRUN TORTUE&, multiplier les possibilités graphiques de l'Apple.

Mode d'emploi de la tortue

Dans les définitions ci-dessous, les paramètres des différentes instructions sont des variables ou des expressions arithmétiques en BASIC. Dans les exemples, on utilise les variables A,X,Y,N qui peuvent être remplacées à loisir par des constantes ou des expressions. Tous les calculs sont effectués en nombres réels, mais les coordonnées de la tortue et son angle d'orientation (exprimé en degrés) sont convertis après arrondi en nombres entiers positifs (X<280, Y<192 et A<360) pour tous les déplacements.

&I : initialisation

L'instruction [&IN] initialise la tortue. Selon la valeur de la variable N, on passe en mode HGR [&I1], HGR2 [&I2] ou HGR pleine page [&I3]. La tortue est placée au centre, tournée vers la droite [&P140,96,0], plume levée [&L]. Cette instruction n'est pas tout-à-fait identique à HGR car elle efface l'écran AVANT de l'afficher, ce qui évite de montrer les restes du dernier affichage graphique. Il est indispensable d'initialiser la tortue ou au moins de la centrer avant de l'utiliser au début, sinon la position de départ est indéterminée.

&E : écrire sur une page

[&E1] ou [&E2] indique sur quelle page va s'effectuer la prochaine écriture, qu'elle soit visible ou non... Ne pas confondre avec l'instruction [&MN] qui affiche la page N sur l'écran. Il est donc facile de dessiner sur la page qui n'est pas actuellement visible.

&M : montrer la page HGR

[&MN]affiche sans l'effacer la page de haute résolution graphique numéro N. Si N=0, on repasse en mode texte sans modifier ni la position du curseur, ni l'affichage du texte, ni la taille de la fenêtre. Si N=1, l'affichage se fait en mode mixte page 1, c'est-à-dire avec les quatre lignes de texte en bas de la page. Si N=3, l'affichage de la page 1 se fait en pleine page. Si N=2, affichage pleine page de la page 2.

&R : rideau

[&RC] recouvre l'écran avec la couleur C. Le numéro de la couleur est compris entre 0 et 8. Les nombres de 0 à 7 sont les couleurs usuelles. Le nombre 8 représente la couleur inverse (l'écran est passé en négatif). La couleur d'affichage en cours pour la tortue n'est pas modifiée. L'instruction [&R] travaille sur la page active (celle indiquée par [&E]).

&C : centrer la tortue

[&C]place la tortue au centre de l'écran et la tourne vers la droite. L'instruction [&C] est équivalente à [&P140,96,0]. La ligne jusqu'au centre de l'écran n'est tracée que si la plume est baissée.

&P : placer la tortue

[&PX,Y,A] place la tortue au point de coordonnées (X,Y) avec un angle A par rapport à l'horizontale. [&P0,0,0] place la tortue en haut à gauche de l'écran et [&P279,191,0] la place en bas à droite. Les trois paramètres peuvent être modifiés indépendamment. Il suffit que les deux virgules soient présentes dans l'instruction. Tout champ vide correspond à un paramètre non modifié. [&PX,,], [&P,Y,], [&P,,A] ou [&PX,Y,] sont par conséquent licites. Pour A=0, la tortue est tournée vers la droite, pour A=90 vers le haut, pour A=180 vers la gauche et pour A=270 vers le bas. Ainsi, A=270 est équivalent à A=−90.

&L : lever la plume

Si la plume est levée, la tortue se déplace sur l'écran sans laisser de trace.

&B : baisser la plume

Après [&BC], la tortue trace son chemin sur l'écran dans la couleur C. Il faut remarquer que [&L] n'est pas identique à [&B0] : si la plume est levée, l'écran n'est pas modifié ; mais si la plume est baissée en noir, la tortue efface l'écran sur son passage.

&A : avancer

[&AD] avance la tortue d'une distance D dans la direction vers laquelle elle est tournée. Une distance négative correspond à une tortue qui recule.

&T : tourner la tortue

[&TA] tourne la tortue d'un angle A, exprimé en degrés, par rapport à sa position précédente. La valeur de A sera positive pour tourner dans le sens inverse des aiguilles d'une montre (sens trigonométrique positif). [&T180] fait faire demi-tour à la tortue.

&S : simuler l'avance

Permet de calculer l'avance de la tortue sans la déplacer, ainsi les erreurs de débordement peuvent être contrôlées, et le déplacement peut être dissocié du calcul des coordonnées (voir [&O] et [&V]).

On ne peut pas simuler plusieurs avances successives : une seule est permise à la fois.

&O : où est la tortue ?

Renvoie la position et l'orientation de la tortue dans trois variables réelles quelconques du programme. Après exécution de [&OX,Y,A], les variables X et Y contiennent les coordonnées de la tortue et A l'angle qu'elle fait avec l'horizontale.

Attention ! Dans cette instruction, X, Y et A doivent être **exclusivement** des variables réelles simples.

Les contenus de X, Y et A seront modifiés à l'exécution de l'instruction. Les valeurs retournées dans X, Y et A représentent la position fictive de la tortue après exécution du déplacement simulé par l'instruction [&S].

Si l'instruction [&O] est exécutée après un déplacement réel de la tortue (instructions &A &P 8C &I et &V), on obtient dans X, Y et A la position réelle de la tortue à cet instant.

Il faut remarquer que l'instruction [&S] n'influence que les coordonnées de la tortue, l'angle retourné par l'instruction [&O] est donc toujours l'angle effectif de la tortue.

L'instruction [&O] permet de contrôler si la tortue est encore sur l'écran après [&S], mais peut aussi servir à mémoriser la position réelle de la tortue pour la replacer en un point particulier avec [&P] après un petit détour.

&V : valider l'avance

Après avoir simulé l'avance de la tortue avec [&S] et vérifié qu'elle est toujours sur l'écran avec [&O], on peut valider le déplacement par [&V]. Cette instruction ne fait rien si aucun mouvement n'a été simulé précédemment. Par exemple, [&A120] fait la même chose que [&S120;V] ou [&S120:&V].

Instructions composées

On peut, comme nous venons de le voir à la fin du paragraphe précédent, composer plusieurs instructions derrière un même symbole ampersand. Il suffit de les séparer par un point-virgule.

Par exemple [&I:&R8:&A123] peut s'écrire plus simplement [&I;R8;A123].

On peut ainsi tracer un carré grâce à l'instruction [&I1;B3;A25;T90;A25;T90;A25;T90;A25;L].

Remarques générales

On peut utiliser les instructions graphiques habituelles en même temps que la tortue. La tortue n'est pas influencée par l'utilisation du HPLOT, car tous ses mouvements sont calculés entre sa position initiale TORTUE.X-Y (valeur entière) et TT.X-Y (valeur réelle) et sa destination représentée par BUT.X-Y et BT.X-Y.

Par contre, si l'on fait HPLOT TO X,Y après avoir utilisé la tortue, on trace une ligne entre la position de la tortue et le point X,Y. Si on fait ensuite [&AD] la tortue avance alors de D à partir de sa position précédente et non à partir du point X,Y.

De même, la couleur de la tortue est mémorisée dans TORCOLOR. Cette couleur a une influence sur celle des lignes tracées par HPLOT, mais HCOLOR n'a pas d'effet sur la couleur de la tortue. Ainsi, HCOLOR=2:&B3:HPLOT10,10 va marquer un point blanc (de couleur 3).

Conclusion : pour éviter toute confusion, utilisez [&P] de préférence à HPLOT. En outre, cela transformera votre tortue en lièvre...

Chargement du programme

Pour utiliser la tortue, il suffit de faire [BRUN TORTUE&]. Le programme est chargé en mémoire entre les adresses $9150 (37200) et $95AC (38316).

Il est donc situé juste sous le DOS. Les pointeurs de l'ampersand sont alors initialisés à l'adresse de début de programme :

Début du programme en $916D (37229)
Pointeurs :
$3F5 (1013) = $4C (76 JMP)
$3F6 (1014) = $6D (109)
$3F7 (1015) = $91 (145)

Un CLEAR est alors généré et le programme donne la main à l'Applesoft par un saut à $D43C. On peut utiliser les nouvelles instructions aussi bien en mode immédiat qu'en mode différé.

Si l'on veut changer TORTUE& par programme, il ne faut pas utiliser BRUN car le programme serait interrompu et le contrôle redonné au niveau de commande du BASIC. Il faut alors, au début du programme utilisateur, faire un BLOAD et initialiser les pointeurs de la manière suivante :

```
10 HIMEM: 37200
20 PRINT CHR$(4)«BLOAD TORTUE&»
30 POKE 1013,76 : POKE 1014,109 :POKE 1015,145
40 REM suite du programme.
```

A vous de jouer...

```
1          ORG $9150
2          OBJ $800
3          NLS
4  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
5  x                              x
6  x     TORTUE  AMPERSAND        x
7  x                              x
8  x        PAR  J.DUMA           x
9  x                              x
10 x          22/5/82             x
11 x                              x
12 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
13 ;
14 STXERR   EQU $DEC9
15 ILLERR   EQU $E199
16 MIMERR   EQU $DD76
17 BASIC    EQU $D43C
18 CLEAR    EQU $D66C
19 ;
20 CARACOK  EQU $DEC0          ;? (A) OU STXERR
21 VIRGULE  EQU $DEBE          ;? , OU STXERR
22 ;
23 ENTIER1  EQU $E6F8          ;EXPRESS ^ ACU1
24 ENTIER2  EQU $E105          ;EXPRESS ^ ACU0-1
25 EXPRESS  EQU $DD67
26 REEL.INT EQU $E10C          ;CONVERSION REEL^INT
27 INT.REEL EQU $E2F2          ;CONVERSION INT^REEL
28 SFP.MFP  EQU $EB53
29 AY.MFP   EQU $EAF9
30 AY.SFP   EQU $E9E3
31 MFP.XY   EQU $EB2B
32 ;
33 MULT.AY  EQU $E97F          ;OPERATIONS
34 MULT     EQU $E982
35 PLUS     EQU $E7C1
36 MOINS    EQU $E7AA
37 COSINUS  EQU $EFEA          ;FONCTIONS
38 SINUS    EQU $EFF1
39 PLUS.5   EQU $E7A0
40 ;
41 VAR      EQU $DFE3          ;^ A,Y
42 CHARGE   EQU $EB2B          ;ACU ^ VARIABLE
43 ;
44 BKGND0   EQU $F3F2          ;EFFACE ECRAN
```

```
 45 BKGND    EQU $F3F4          ;COLOR^ ECRAN
 46 HCOLOR   EQU $F6F0          ;COULEUR=(X)
 47 HPLOT    EQU $F457
 48 HLINE    EQU $F53A
 49 ;
 50 MASCOLOR EQU $F6F6
 51 ;
 52 ACU0     EPZ $A0
 53 ACU1     EPZ $A1
 54 PAGHGR   EPZ $E6            ;PAGE TRAVAIL
 55 PLUME    EPZ $6             ;L=0 B=1
 56 TORTUE.X EPZ $7
 57 TORTUE.Y EPZ $9
 58 BUT.X    EPZ $F9
 59 BUT.Y    EPZ $FB
 60 TORCOLOR EPZ $FC
 61 ANGLE    EPZ $FD
 62 ;
 63 AMPER    EQU $3F5
 64 HIMEM    EPZ $73
 65 ;
 66 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 67 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 68 ;
 69 DEPART:
 70         LDA #DEBUT          ;INITIALISE
 71         STA AMPER+1         ;POINTEURS &
 72         LDA /DEBUT
 73         STA AMPER+2
 74         LDA #$4C
 75         STA AMPER
 76 ;
 77         LDA #DEPART         ;HIMEM
 78         STA HIMEM
 79         LDA /DEPART
 80         STA HIMEM+1
 81 ;
 82         JSR CLEAR
 83         JMP BASIC
 84 ;
 85 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 86 ;
 87 DEBUT:
 88         BNE TORTUE
 89         RTS                 ;SI & RETOUR
 90 ;
 91 TORTUE   CMP #'P'
 92         BNE T0
 93         JMP PLACER
 94 T0:
 95         CMP #'A'
 96         BNE T1
 97         JMP AVANCER
 98 T1:
 99         CMP #'T'
100         BNE T2
101         JMP TOURNER
102 T2:
103         CMP #'S'
104         BNE T3
105         JMP SIMULER
106 T3:
107         CMP #'V'
108         BNE T4
109         JMP VALIDER
110 T4:
111         CMP #'O'
112         BNE T5
113         JMP OU
114 T5:
115         CMP #'B'
116         BNE T6
117         JMP BAISSER
118 T6:
119         CMP #'L'
120         BNE T7
121         JMP LEVER
122 T7:
123         CMP #'C'
124         BNE T8
125         JMP CENTRER
126 T8:
127         CMP #'E'
128         BNE T9
129         JMP ECRIRE
130 T9:
131         CMP #'M'
132         BNE T10
133         JMP MONTRER
134 T10:
135         CMP #'R'
136         BNE T11
137         JMP RIDEAU
138 T11:
139         CMP #'I'
140         BNE T.ERR
141         JMP INIT
142 ;
143 T.ERR    JMP STXERR
144 ;
145 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
146 x PLACE LA TORTUE: &P X,Y  x
147 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
148 ;
149 PLACER:
150         JSR $B1
151         CMP #','
152         BEQ P.LIT_Y
153         JSR ENTIER2
154         LDX ACU0
155         CPX #$1
156         BEQ P.256           ;X>256
157         BCS P.ERR           ;X>512 ERREUR
158 P.DANSX  STX BUT.X+1
159         LDX ACU1
160         STX BUT.X
161         JSR TRANSX
162 P.LIT_Y  JSR VIRGULE
```

```
163           CMP #',
164           BEQ P.LIT_A
165           JSR ENTIER2
166           LDX ACU0
167           BNE P.ERR          ;Y>256 ERREUR
168           LDX ACU1
169           CPX #!192
170           BCS P.ERR          ;Y>191 ERREUR
171           STX BUT.Y
172           JSR TRANSY
173 P.LIT_A   JSR VIRGULE
174           BEQ P.FIN
175           CMP #';'
176           BEQ P.FIN
177           JSR EXPRESS
178           JSR REEL.INT
179           LDA ACU0
180           STA ANGLE+1
181           LDA ACU1
182           STA ANGLE
183           JSR CADRANGL
184           JSR TRANSA
185 P.FIN     LDA PLUME
186           BNE P.TRACE
187           JSR TOBUT
188           JMP SORTIE
189 P.TRACE   JSR PLOTOBUT
190           JMP SORTIE
191 P.256     LDX ACU1
192           CPX #!24
193           BCS P.ERR          ;X>279 ERREUR
194           LDX ACU0
195           JMP P.DANSX
196 P.ERR     JMP ILLERR
197 ;
198 CADRANGL:
199 C.MOINS   LDA ANGLE+1
200           BMI C.ADD
201           SEC
202           LDA ANGLE
203           SBC #!104          ;A=A-360
204           STA ANGLE
205           LDA ANGLE+1
206           SBC #!1
207           STA ANGLE+1
208           JMP C.MOINS
209 C.PLUS    LDA ANGLE+1
210           BMI C.ADD
211           RTS
212 C.ADD     CLC
213           LDA ANGLE
214           ADC #!104          ;A=A+360
215           STA ANGLE
216           LDA ANGLE+1
217           ADC #!1
218           STA ANGLE+1
219           JMP C.PLUS
220 ;
221 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
222 x AVANCE LA TORTUE:   &A D  x
223 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
224 ;
225 AVANCER:
226           JSR SIMULE
227           JSR VALIDE
228           JMP SORTIE
229 ;
230 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
231 x TOURNE LA TORTUE:   &T A  x
232 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
233 ;
234 TOURNER:
235           JSR $B1
236           BEQ TT.ERR
237           CMP #';'
238           BEQ TT.ERR
239           JSR EXPRESS
240           JSR REEL.INT
241           CLC
242           LDA ANGLE
243           ADC ACU1
244           STA ANGLE
245           LDA ANGLE+1
246           ADC ACU0
247           STA ANGLE+1
248           JSR CADRANGL
249           JSR TRANSA
250           JMP SORTIE
251 TT.ERR    JMP STXERR
252 ;
253 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
254 x SIMULE L'AVANCE:    &S D  x
255 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
256 ;
257 SIMULER:
258           JSR SIMULE
259           JMP SORTIE
260 ;
261 PI_180    HEX 7B0EFA350F
262 ;
263 TF1       HEX 0000000000
264 TF2       HEX 0000000000
265 ;
266 TT.X      HEX 0000000000
267 TT.Y      HEX 0000000000
268 BT.X      HEX 0000000000
269 BT.Y      HEX 0000000000
270 AGL       HEX 0000000000
271 ;
272 SIMULE:
273           LDA #AGL
274           LDY /AGL
275           JSR AY.MFP
276           LDA #PI_180
277           LDY /PI_180
278           JSR MULT.AY
279           JSR MFP.TF1        ;A*PI/180 DANS TF1
280 ;
```

```
281           JSR $B1
282           JSR EXPRESS
283           JSR MFP.TF2          ;D DANS TF2
284           JSR TF1.MFP
285           JSR COSINUS
286           JSR TF2.SFP
287           JSR MULT
288           JSR TF1.SFP
289           JSR MFP.TF1          ;D*COS(A*PI/180) DANS TF1
290 ;
291           JSR SFP.MFP
292           JSR SINUS
293           JSR TF2.SFP
294           JSR MULT
295           JSR MFP.TF2          ;D*SIN(A*PI/180) DANS TF2
296 ;
297           JSR TF2.MFP
298           LDA #TT.Y
299           LDY /TT.Y
300           JSR AY.SFP
301           JSR MOINS            ;Y=Y-D*SIN(A)
302           LDX #BT.Y            ;->BT.Y
303           LDY /BT.Y
304           JSR MFP.XY
305           JSR PLUS.5           ;ARRONDI-> BUT.Y
306           JSR REEL.INT
307           LDA ACU0
308           BEQ S.OK
309           LDA #!200            ;Y>191
310           STA ACU1
311 S.OK      LDA ACU1
312           STA BUT.Y
313 ;
314           LDA #TT.X
315           LDY /TT.X
316           JSR AY.MFP
317           JSR TF1.SFP
318           JSR PLUS             ;X=X+D*COS(A)
319           LDX #BT.X            ;-> BT.X
320           LDY /BT.X
321           JSR MFP.XY
322           JSR PLUS.5           ;ARRONDI->BUT.X
323           JSR REEL.INT
324           LDA ACU0
325           STA BUT.X+1
326           LDA ACU1
327           STA BUT.X
328           RTS
329 ;
330 MFP.TF1   LDX #TF1
331           LDY /TF1
332           JMP MFP.XY
333 ;
334 MFP.TF2   LDX #TF2
335           LDY /TF2 .
336           JMP MFP.XY
337 ;
338 TF1.MFP   LDA #TF1
339           LDY /TF1
340           JMP AY.MFP
341 ;
342 TF2.MFP   LDA #TF2
343           LDY /TF2
344           JMP AY.MFP
345 ;
346 TF1.SFP   LDA #TF1
347           LDY /TF1
348           JMP AY.SFP
349 ;
350 TF2.SFP   LDA #TF2
351           LDY /TF2
352           JMP AY.SFP
353 ;
354 ********************************
355 * VALIDE L'AVANCE:     &V    *
356 ********************************
357 ;
358 VALIDER:
359           JSR $B1
360           JSR VALIDE
361           JMP SORTIE
362 ;
363 VALIDE:
364           LDA BUT.X+1
365           CMP #$1
366           BEQ V.256
367           BCS V.ERR
368 V.Y       LDA BUT.Y
369           CMP #!192
370           BCS V.ERR
371           LDA PLUME
372           BNE V.TRACE
373           JMP TOBUT
374 V.TRACE   JMP PLOTOBUT
375 V.256     LDA BUT.X
376           CMP #!24
377           BCC V.Y
378 V.ERR     JMP ILLERR
379 ;
380 ********************************
381 * OU EST LA TORTUE: &OX,Y,A *
382 ********************************
383 ;
384 OU:
385           JSR $B1
386           LDA BUT.X+1
387           LDY BUT.X
388           JSR INT.REEL
389           JSR VARIABLE
390           TAX
391           JSR CHARGE           ;X TORTUE
392           LDA #$00
393           LDY BUT.Y
394           JSR INT.REEL
395           JSR VIRGULE
396           JSR VARIABLE
397           TAX
398           JSR CHARGE           ;Y TORTUE
```

```
399           LDA ANGLE+1
400           LDY ANGLE
401           JSR INT.REEL
402           JSR VIRGULE
403           JSR VARIABLE
404           TAX
405           JSR CHARGE          ;ANGLE
406           LDA #$00            ;ACCEPTE TABLEAUX
407           STA $14
408           JMP SORTIE
409 ;
410 VARIABLE LDA #$80             ;REFUSE TABLEAUX
411           STA $14
412           JSR VAR
413           BIT $11
414           BMI V.MRR
415           BIT $12
416           BMI V.MRR
417           RTS
418 V.MRR     JMP MIMERR
419 ;
420 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
421 x LEVER OU BAISSER: $L  $BC x
422 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
423 ;
424 LEVER:
425           JSR $B1
426 L.LEVER   LDA #$00
427           STA PLUME
428           JMP SORTIE
429 ;
430 BAISSER:
431           JSR $B1
432           JSR ENTIER1
433           LDX ACU1
434           CPX #8
435           BCC B.PLUME
436           JMP ILLERR
437 B.PLUME    STX TORCOLOR
438           JSR HCOLOR
439           JSR PLOTO
440           LDA #$1
441           STA PLUME
442           JMP SORTIE
443 ;
444 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
445 x CENTRER LA TORTUE:    &C  x
446 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
447 ;
448 CENTRER:
449           JSR $B1
450           JSR MILIEU
451           LDA PLUME
452           BNE C.TRACER
453           JSR TOBUT
454           JMP SORTIE
455 C.TRACER JSR PLOTOBUT
456           JMP SORTIE
457 ;
458 MILIEU:
459           LDA #!0             ;CENTRE ECRAN
460           STA ANGLE
461           STA ANGLE+1
462           STA BUT.X+1
463           LDA #!140
464           STA BUT.X
465           LDA #!96
466           STA BUT.Y
467 ;
468           JSR TRANSA
469           JSR TRANSY
470 TRANSX    LDA BUT.X+1         ;BUTX -. TT.X
471           LDY BUT.X
472           JSR INT.REEL
473           LDX #BT.X
474           LDY /BT.X
475           JSR MFP.XY
476           RTS
477 ;
478 TRANSY    LDA #$00            ;BUTY -> TT.Y
479           LDY BUT.Y
480           JSR INT.REEL
481           LDX #BT.Y
482           LDY /BT.Y
483           JSR MFP.XY
484           RTS
485 ;
486 TRANSA    LDA ANGLE+1         ;ANGLE->AGL
487           LDY ANGLE
488           JSR INT.REEL
489           LDX #AGL
490           LDY /AGL
491           JSR MFP.XY
492           RTS
493 ;
494 TOBUT:
495           LDX #$04            ;TORTUEXY=BUTXY
496 ^2        LDA BT.X,X
497           STA TT.X,X
498           LDA BT.Y,X
499           STA TT.Y,X
500           DEX
501           BPL <2
502           LDX BUT.X
503           STX TORTUE.X
504           LDY BUT.X+1
505           STY TORTUE.X+1
506           LDA BUT.Y
507           STA TORTUE.Y
508           RTS
509 ;
510 PLOTO:
511           LDX TORCOLOR        ;TORTUE EN TORTUEXY
512           JSR HCOLOR
513           LDX TORTUE.X
514           LDY TORTUE.X+1
515           LDA TORTUE.Y
516           JMP HPLOT
```

```
517 ;
518 PLOTOBUT:
519         JSR PLOTO           ;LIGNE TORTUE-BUT
520         JSR TOBUT
521         LDX BUT.X+1
522         TAY
523         LDA BUT.X
524         JMP HLINE
525 ;
526 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
527 x ECRIRE SUR LA PAGE: &E1/2 x
528 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
529 ;
530 ECRIRE:
531         JSR $B1
532         JSR ENTIER1
533         LDA ACU1
534         CMP #$1
535         BNE E.PAGE2
536         LDA #$20
537         STA PAGHGR
538         JMP SORTIE
539 E.PAGE2 CMP #$2
540         BNE E.ERR
541         LDA #$40
542         STA PAGHGR
543         JMP SORTIE
544 E.ERR   JMP ILLERR
545 ;
546 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
547 x MONTRER LA PAGE HGR  &M N x
548 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
549 ;
550 MONTRER:
551         JSR $B1
552         JSR ENTIER1
553         LDA ACU1
554         BNE M.123
555         BIT $C054           ;N=0
556         BIT $C051           ;PAGE TEXTE
557         JMP SORTIE
558 M.123   CMP #$1
559         BNE M.23
560         BIT $C053           ;N=1
561         BIT $C054           ;HGR1 MIXTE
562         BIT $C057
563         BIT $C050
564         JMP SORTIE
565 M.23    CMP #$2
566         BNE M.3
567         BIT $C052           ;N=2
568         BIT $C055           ;HGR2 TOTAL
569         BIT $C057
570         BIT $C050
571         JMP SORTIE
572 M.3     CMP #$3
573         BNE M.ERR
574         BIT $C052           ;N=3
575         BIT $C054           ;HGR1 TOTAL
576         BIT $C057
577         BIT $C050
578         JMP SORTIE
579 M.ERR   JMP ILLERR
580 ;
581 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
582 x RIDEAU: &R C  INVERSE:C=0 x
583 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
584 ;
585 RIDEAU:
586         JSR $B1
587         JSR ENTIER1
588         LDA ACU1
589         CMP #$8
590         BCC R.COLOR
591         BEQ R.NEGAT
592         JMP ILLERR
593 ;
594 R.COLOR  TAX
595         LDA MASCOLOR,X
596         JSR BKGND
597         JMP SORTIE
598 ;
599 R.NEGAT  LDA PAGHGR
600         STA $1B
601         LDY #$0
602         STY $1A
603 ^1      LDA ($1A),Y
604         EOR #$FF
605         STA ($1A),Y
606         INY
607         BNE <1
608         INC $1B
609         LDA $1B
610         AND #$1F
611         BNE <1
612         JMP SORTIE
613 ;
614 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
615 x INITIALISATION: &I 1 ^ 3  x
616 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
617 ;
618 INIT:
619         JSR MILIEU
620         JSR TOBUT
621         JSR $B1
622         JSR ENTIER1
623         LDA ACU1
624         CMP #$1
625         BNE I.23
626         LDA #$20            ;PAGE 1 MIXTE
627         STA PAGHGR
628         JSR BKGND0
629         BIT $C053
630         BIT $C054
631 I.SORTIE BIT $C057
632         BIT $C050
633         JMP L.LEVER
634 ;
```

```
635 I.23    CMP #$2
636         BNE I.3
637         LDA #$40            ;PAGE 2 TOTALE
638         STA PAGHGR
639         JSR BKGND0
640         BIT $C052
641         BIT $C055
642         JMP I.SORTIE
643 ;
644 I.3     CMP #$3
645         BNE I.ERR
646         LDA #$20            ;PAGE 1 TOTALE
647         STA PAGHGR
648         JSR BKGND0
649         BIT $C052
650         BIT $C054
651         JMP I.SORTIE
652 I.ERR   JMP ILLERR
653 ;
654 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
655 x SUITE DE L'INSTRUCTION &  x
656 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
657 ;
658 SORTIE:
659         JSR $B7
660         BNE S.SUITE
661         RTS                 ;INSTRUCTION TERMINEE
662 S.SUITE LDA #';'
663         JSR CARACOK
664         JMP DEBUT
665 ;
666 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
667 ;
668         END
```

TORTUE & : récapitulation

```
*9150.95AD

9150- A9 6D 8D F6 03 A9 91 8D
9158- F7 03 A9 4C 8D F5 03 A9
9160- 50 85 73 A9 91 85 74 20
9168- 6C D6 4C 3C D4 D0 01 60
9170- C9 50 D0 03 4C CE 91 C9
9178- 41 D0 03 4C 66 92 C9 54
9180- D0 03 4C 6F 92 C9 53 D0
9188- 03 4C 97 92 C9 56 D0 03
9190- 4C 73 93 C9 4F D0 03 4C
9198- 9D 93 C9 42 D0 03 4C F4
91A0- 93 C9 4C D0 03 4C EA 93
91A8- C9 43 D0 03 4C 12 94 C9
91B0- 45 D0 03 4C A5 94 C9 4D
91B8- D0 03 4C C6 94 C9 52 D0
91C0- 03 4C 15 95 C9 49 D0 03
91C8- 4C 4C 95 4C C9 DE 20 B1
91D0- 00 C9 2C F0 14 20 05 E1
91D8- A6 A0 E0 01 F0 51 B0 5A
91E0- 86 FA A6 A1 86 F9 20 3E
91E8- 94 20 BE DE C9 2C F0 12
91F0- 20 05 E1 A6 A0 D0 43 A6
91F8- A1 E0 C0 B0 3D 86 FB 20
9200- 4D 94 20 BE DE F0 18 C9
9208- 3B F0 14 20 67 DD 20 0C
9210- E1 A5 A0 85 FE A5 A1 85
9218- FD 20 3D 92 20 5C 94 A5
9220- 06 D0 06 20 6B 94 4C 9F
9228- 95 20 97 94 4C 9F 95 A6
9230- A1 E0 18 B0 05 A6 A0 4C
9238- E0 91 4C 99 E1 A5 FE 30
9240- 15 38 A5 FD E9 68 85 FD
9248- A5 FE E9 01 85 FE 4C 3D
9250- 92 A5 FE 30 01 60 18 A5
9258- FD 69 68 85 FD A5 FE 69
9260- 01 85 FE 4C 51 92 20 C5
9268- 92 20 7C 93 4C 9F 95 20
9270- B1 00 F0 20 C9 3B F0 1C
9278- 20 67 DD 20 0C E1 18 A5
9280- FD 65 A1 85 FD A5 FE 65
9288- A0 85 FE 20 3D 92 20 5C
9290- 94 4C 9F 95 4C C9 DE 20
9298- C5 92 4C 9F 95 7B 0E FA
92A0- 35 0F 00 00 00 00 00 00
92A8- 00 00 00 00 00 00 00 00
92B0- 00 00 00 00 00 00 00 00
92B8- 00 00 00 00 00 00 00 00
92C0- 00 00 00 00 00 A9 C0 A0
92C8- 92 20 F9 EA A9 9D A0 92
92D0- 20 7F E9 20 49 93 20 B1
92D8- 00 20 67 DD 20 50 93 20
92E0- 57 93 20 EA EF 20 6C 93
92E8- 20 82 E9 20 65 93 20 49
92F0- 93 20 53 EB 20 F1 EF 20
92F8- 6C 93 20 82 E9 20 50 93
9300- 20 5E 93 A9 B1 A0 92 20
9308- E3 E9 20 AA E7 A2 BB A0
9310- 92 20 2B EB 20 A0 E7 20
9318- 0C E1 A5 A0 F0 04 A9 C8
9320- 85 A1 A5 A1 85 FB A9 AC
9328- A0 92 20 F9 EA 20 65 93
9330- 20 C1 E7 A2 B6 A0 92 20
9338- 2B EB 20 A0 E7 20 0C E1
9340- A5 A0 85 FA A5 A1 85 F9
9348- 60 A2 A2 A0 92 4C 2B EB
9350- A2 A7 A0 92 4C 2B EB A9
9358- A2 A0 92 4C F9 EA A9 A7
9360- A0 92 4C F9 EA A9 A2 A0
9368- 92 4C E3 E9 A9 A7 A0 92
9370- 4C E3 E9 20 B1 00 20 7C
9378- 93 4C 9F 95 A5 FA C9 01
9380- F0 12 B0 16 A5 FB C9 C0
9388- B0 10 A5 06 D0 03 4C 6B
9390- 94 4C 97 94 A5 F9 C9 18
9398- 90 EA 4C 99 E1 20 B1 00
93A0- A5 FA AA F9 20 F2 E2 20
93A8- D7 93 AA 20 2B EB A9 00
93B0- A4 FB 20 F2 E2 20 BE DE
93B8- 20 D7 93 AA 20 2B EB A5
93C0- FE A4 FD 20 F2 E2 20 BE
93C8- DE 20 D7 93 AA 20 2B EB
93D0- A9 00 85 14 4C 9F 95 A9
93D8- 80 85 14 20 E3 DF 24 11
93E0- 30 05 24 12 30 01 60 4C
93E8- 76 DD 20 B1 00 A9 00 85
93F0- 06 4C 9F 95 20 B1 00 20
93F8- FB E6 A6 A1 E0 08 90 03
9400- 4C 99 E1 86 FC 20 F0 F6
9408- 20 89 94 A9 01 85 06 4C
9410- 9F 95 20 B1 00 20 28 94
9418- A5 06 D0 06 20 6B 94 4C
9420- 9F 95 20 97 94 4C 9F 95
9428- A9 00 85 FD 85 FE 85 FA
9430- A9 8C 85 F9 A9 60 85 FB
9438- 20 5C 94 20 4D 94 A5 FA
9440- A4 F9 20 F2 E2 A2 B6 A0
9448- 92 20 2B EB 60 A9 00 A4
9450- FB 20 F2 E2 A2 BB A0 92
9458- 20 2B EB 60 A5 FE A4 FD
9460- 20 F2 E2 A2 C0 A0 92 20
9468- 2B EB 60 A2 04 BD B6 92
9470- 9D AC 92 BD BB 92 9D B1
9478- 92 CA 10 F1 A6 F9 86 07
9480- A4 FA 84 08 A5 FB 85 09
9488- 60 A6 FC 20 F0 F6 A6 07
9490- A4 08 A5 09 4C 57 F4 20
9498- 89 94 20 6B 94 A6 FA A8
94A0- A5 F9 4C 3A F5 20 B1 00
94A8- 20 F8 E6 A5 A1 C9 01 D0
94B0- 07 A9 20 85 E6 4C 9F 95
94B8- C9 02 D0 07 A9 A0 85 E6
94C0- 4C 9F 95 4C 99 E1 20 B1
94C8- 00 20 F8 E6 A5 A1 D0 09
94D0- 2C 54 C0 2C 51 C0 4C 9F
94D8- 95 C9 01 D0 0F 2C 53 C0
94E0- 2C 54 C0 2C 57 C0 2C 50
94E8- C0 4C 9F 95 C9 02 D0 0F
94F0- 2C 52 C0 2C 55 C0 2C 57
94F8- C0 2C 50 C0 4C 9F 95 C9
9500- 03 D0 0F 2C 52 C0 2C 54
9508- C0 2C 57 C0 2C 50 C0 4C
9510- 9F 95 4C 99 E1 20 B1 00
9518- 20 F8 E6 A5 A1 C9 08 90
9520- 05 F0 0D 4C 99 E1 AA BD
9528- F6 F6 20 F4 F3 4C 9F 95
9530- A5 E6 85 1B A0 00 84 1A
9538- B1 1A 49 FF 91 1A C8 D0
9540- F7 E6 1B A5 1B 29 1F D0
9548- EF 4C 9F 95 20 28 94 20
9550- 6B 94 20 B1 00 20 F8 E6
9558- A5 A1 C9 01 D0 16 A9 20
9560- 85 E6 20 F2 F3 2C 53 C0
9568- 2C 54 C0 2C 57 C0 2C 50
9570- C0 4C ED 93 C9 02 D0 10
9578- A9 40 85 E6 20 F2 F3 2C
9580- 52 C0 2C 55 C0 4C 6B 95
9588- C9 03 D0 10 A9 20 85 E6
9590- 20 F2 F3 2C 52 C0 2C 54
9598- C0 4C 6B 95 4C 99 E1 20
95A0- B7 00 D0 01 60 A9 3B 20
95A8- C0 DE 4C 6D 91 00
```

Le loto, c'est facile ...

Philippe Fabert

Ce programme est destiné à réjouir les amateurs de loto et d'assembleur, et à enrichir les premiers... Tout d'abord, ils sont sûrs d'obtenir des tirages qui ne se répèteront jamais. En effet, la fonction RND a le défaut de donner la même série de nombres chaque fois que l'on allume l'ordinateur. L'astuce que j'utilise ici permet de pallier ce défaut. Le sous programme gérant un nombre aléatoire peut être facilement adapté à d'autres programmes.

Les variables K, K1, AI et AJ sont utilisées dans les programmes en BASIC et en assembleur. Pour éviter toute confusion, il faut savoir que ces variables représentent des nombres dans le programme en BASIC et des adresses dans la routine en assembleur.

Voici les fonctions en assembleur de ces variables :

A1 : adresse de départ d'une série de numéros.

AI : au début du programme, contient l'adresse de départ d'une série
au cours du programme, l'adresse courante du numéro calculé
à la fin, l'adresse de départ de la prochaine série.

AJ : contient successivement les adresses des numéros d'une série pour la vérification et le classement.

N.D.L.R.

A chaque appel de nombre aléatoire, on regarde un octet d'entrés/sortie ($C051, page graphique), puis on fait autant d'appels à RND que la valeur de cet octet.

Ce RND est particulièrement intéressant dans les programmes dans lesquels on lui fait appel avant que le clavier ne soit utilisé.


```
]CALL-151

*4000.40ED

4000- A5 18 85 06 A5 19 85 07
4008- A5 EB 85 FB AD 51 C0 85
4010- F9 20 AE EF C6 F9 D0 F9
4018- A5 EC 85 FC A5 06 85 1A
4020- A5 07 85 1B A5 08 A4 09
4028- 20 7F E9 20 23 EC A5 FE
4030- A4 FF 20 BE E7 A6 18 A4
4038- 19 20 2B EB C6 FC A5 FC
4040- C5 FB F0 21 A5 18 A4 19
4048- 20 F9 EA A5 1A A4 1B 20
4050- A7 E7 20 82 EB F0 BA 18
4058- A5 1A 69 05 90 02 E6 1B
4060- 85 1A 4C 3C 40 18 A5 18
4068- 69 05 90 02 E6 19 85 18
4070- C6 FB D0 98 A9 00 85 FD
4078- A5 EB 38 E9 01 85 FB 18
4080- A5 07 85 19 85 1B A5 06
4088- 85 18 69 05 90 02 E6 1B
4090- 85 1A A5 1A A4 1B 20 F9
4098- EA A5 18 A4 19 20 A7 E7
40A0- 20 82 EB 10 03 4C BD 40
40A8- A0 04 B1 1A 91 18 88 10
40B0- F9 20 53 EB A6 1A A4 1B
40B8- 20 2B EB E6 FD C6 FB F0
40C0- 14 18 A5 1B 85 19 A5 1A
40C8- 85 18 69 05 90 02 E6 1B
40D0- 85 1A 4C 92 40 A5 FD C9
40D8- 00 D0 99 C6 FA F0 0E 18
40E0- A5 18 69 0A 90 02 E6 19
40E8- 85 18 4C 00 40 60
```

LOTO : programme BASIC

```
]LIST

1  PRINT  CHR$ (4)"BLOADLOTO.OBJO"
5  TEXT : HOME : HTAB 13: INVERSE : PRIN
       T "JOUONS AU LOTO": NORMAL : VTAB
        6: INPUT "COMBIEN DE NOMBRES (AU
        MOINS 6) ? ";CH: POKE 235,CH:CL
       = CH + 1: POKE 236,CL: REM
         235 (OU $EB EN HEXA) CONTIENT L
       A VALEUR CH
10  VTAB 11: INPUT "NOMBRE DE SERIES (AU
        MOINS UNE) ? ";S: POKE 250,S: RE
       M          MET LE NOMBRE DE SERIES
        A TIRER EN 250 (OU $FA EN HEXA)
15 U = S * CH: DIM A(U): REM      TABLEAU
        CONTENANT LES NUMEROS
20 K = 49: POKE 8, PEEK (131): POKE 9, P
       EEK (132): REM       LES MEMOIRES
        EN PAGE ZERO CONTIENNENT RESPECT
       IVEMENT L'ADRESSE BASSE ET HAUTE
       DE 49
25 K1 = 1: POKE 254, PEEK (131): POKE 25
       5, PEEK (132): REM     DE MEME POU
       R 1, 254=$FE ET 255=$FF EN HEXA
30 X = A(1): POKE 24, PEEK (131): POKE 2
       5, PEEK (132): REM     DONNE L'ADR
       ESSE DE DEPART DU 1 NUMERO DE LA
       1 SERIE
35  CALL 16384: REM       APPELLE LE PROG
       RAMME EN ASSEMBLEUR EN 16384 OU $
       4000 EN HEXA
49  REM  AFFICHAGE
50  HOME : FOR I = 1 TO U: PRINT  SPC( 3
        -  LEN ( STR$ (A(I))));A(I);
55  IF I / CH =  INT (I / CH) THEN  PRIN
       T : REM  SAUTE UNE LIGNE
60  NEXT
```

LOTO en assembleur DOS TOOL KIT

```
*************************************
*         *
* LOTO      *
*         *
* JUIN 82       *
*         *
* PAR PH.FABERT       *
*         *
*************************************
;
;TABLE DES LABELS
;
SUB     EQU  $E7A7
ALEA    EQU  $C051
RND     EQU  $EFAE
MOVMF   EQU  $EB2B
MOVFM   EQU  $EAF9
MOVFA   EQU  $EB53
MULP    EQU  $E97F
INT     EQU  $EC23
SIGN    EQU  $EB82
ADD     EQU  $E7BE
A1      EQU  $6
K       EQU  $8
AI      EQU  $18
AJ      EQU  $1A
CHIFFRE EQU  $EB
CHIF1   EQU  $EC
CPT     EQU  $F9
BUL     EQU  $FA
C       EQU  $FB
C1      EQU  $FC
FLAG    EQU  $FD
K1      EQU  $FE
;
;DEBUT DU PROGRAMME
;
        ORG  $4000
PART1   LDA  AI
        STA  A1
        LDA  AI+1
        STA  A1+1
        LDA  CHIFFRE;NOMBRE DE CHIFFRES A CHAQUE SERIE
        STA  C          ;COMPTEUR DE LA SERIE
;
;BOUCLE GERANT UN
;NOMBRE ALEATOIRE
;
PART2   LDA  ALEA       ;NOMBRE ALEATOIRE
        STA  CPT        ;ENTRE 0 ET 255
PART3   JSR  RND        ;FONCTION RND DANS FAC
        DEC  CPT
        BNE  PART3
        LDA  CHIF1
        STA  C1         ;COMPTEUR PROVISOIRE
        LDA  A1
        STA  AJ
        LDA  A1+1
        STA  AJ+1
;
;CALCUL DU NOMBRE
;
        LDA  K          ;K ET K+1=ADRESSE DE 49
        LDY  K+1        ;49 -> ARG PAR MULP
        JSR  MULP       ;FAC=FAC*ARG
        JSR  INT        ;REND ENTIER FAC
        LDA  K1         ;K1 ET K1+1=ADRESSE DE 1
        LDY  K1+1       ;1 -> ARG PAR ADD
```

```
        JSR  ADD        ;ELIMINE 0
        LDX  AI
        LDY  AI+1
        JSR  MOVMF      ;FAC -> AI
PART5   DEC  C1
        LDA  C1
        CMP  C          ;SI C=C1 EVITE VERIFICATION
        BEQ  PART7
;
;LE NOMBRE EXISTE-T-IL DEJA ?
;
        LDA  AI
        LDY  AI+1
        JSR  MOVFM
        LDA  AJ
        LDY  AJ+1
        JSR  SUB        ;FAC=ARG-FAC
        JSR  SIGN       ;SIGNE DANS REGISTRE A
        BEQ  PART3      ;RESULTAT NUL, ON RECOMMENCE
;
;CALCUL L'ADRESSE DE AJ
;
        CLC
        LDA  AJ
        ADC  #$5
        BCC  PART6
        INC  AJ+1
PART6   STA  AJ
        JMP  PART5
;
;CALCUL DE L'ADRESSE DE AI
;
PART7   CLC
        LDA  AI
        ADC  #$5
        BCC  PART8
        INC  AI+1
PART8   STA  AI
        DEC  C
        BNE  PART2
;
;CLASSEMENT DES NUMEROS
;
TRI     LDA  #$0
        STA  FLAG       ;DRAPEAU A ZERO
        LDA  CHIFFRE
        SEC
        SBC  #$01
        STA  C
;
;CALCUL LES ADRESSES
;
        CLC
        LDA  A1+1
        STA  AI+1
        STA  AJ+1
        LDA  A1
        STA  AI
        ADC  #$5
        BCC  TRI1
        INC  AJ+1
TRI1    STA  AJ
COMP    LDA  AJ         ;ADRESSE DU 2 NUMERO
        LDY  AJ+1       ;DANS AJ
        JSR  MOVFM      ;AJ -> FAC
        LDA  AI         ;ADRESSE DU 1 NUMERO
        LDY  AI+1       ;DANS AI
        JSR  SUB        ;FAC=ARG-FAC
        JSR  SIGN
        BPL  ECH        ;RESULTAT POSITIF -> ECHANGE
        JMP  ADR
;
;EFFECTUE L'ECHANGE
;
ECH     LDY  #$4        ;NOMBRE D'OCTETS POUR FORMAT FLOTTANT
ECH1    LDA  (AJ),Y
        STA  (AI),Y     ;AJ -> AI
        DEY
        BPL  ECH1       ;Y<>0, ECHANGE CONTINUE
        JSR  MOVFA      ;ARG -> FAC
        LDX  AJ
        LDY  AJ+1
        JSR  MOVMF      ;FAC -> AJ
        INC  FLAG       ;ECHANGE EFFECTUE -> DRAPEAU A 1
;
;CALCUL L'ADRESSE
;
ADR     DEC  C
        BEQ  FLG        ;COMPTEUR A ZERO
;
;CALCUL LES ADESSES
;
        CLC
        LDA  AJ+1
        STA  AI+1
        LDA  AJ
        STA  AI
        ADC  #$5
        BCC  ADR1
        INC  AJ+1
ADR1    STA  AJ
        JMP  COMP
FLG     LDA  FLAG
        CMP  #$0        ;ECHANGE ?
        BNE  TRI        ;OUI, RETOUR A TRI
;
;LES NUMEROS SONT TIRES
;
        DEC  BUL        ; RESTE-IL ENCORE DES SERIES?
        BEQ  FIN        ; NON, TERMINE
        CLC
        LDA  AI
        ADC  #$0A
        BCC  ADR2
        INC  AI+1
ADR2    STA  AI
        JMP  PART1
FIN     RTS
```

pour mieux choisir votre ordinateur et pour mieux l'utiliser:

lisez

L'ORDINATEUR INDIVIDUEL

Vous y trouverez :
l'actualité et les tendances de l'informatique individuelle • bancs d'essais des principaux matériels • des panoramas et des tests comparatifs • le point des grandes manifestations internationales • des articles d'initiation • des synthèses • des programmes • des interviews "exemplaires" • des conseils • des idées • des astuces

L'ORDINATEUR INDIVIDUEL, chez votre marchand de journaux

Tableaux de taille déclarée en Pascal

Régis Lardennois

Le programme ci-après montre comment on peut créer des tableaux de taille déclarée à l'exécution d'un programme en Pascal. Ce tableau est créé en variable dynamique.

Le programme de démonstration qui suit permet en outre l'affichage des valeurs des pointeurs utilisés.

Remarque : le compilateur Pascal effectue les tests de débordement d'indice de tableau, non par rapport à la longueur réelle du tableau, mais par rapport à la longueur déclarée dans le type (5000 dans notre cas). La longueur réelle ne peut par conséquent pas dépasser la longueur maximale déclarée.

```
PROGRAM TABLEAUXVARIABLES;

CONST MAXTAB  = 5000;
      MINIMUM = 140 ;{ minimum de memoire necessaire pour le deroulement    }
                     { normal du programme en mots de 16 bits sur APPLE II  }
                     { environ 80 en APPLE II et 140 en APPLE ///           }

TYPE
   ARTICLE = RECORD NUMERO:INTEGER;        { variable occupant 10 octets      }
                    NOM   :STRING[ 9];     { c'est a dire  5 mots de 16 bits  }
              END;
   TABLEAU = ARRAY[0..MAXTAB] OF ARTICLE;

   PTINT   = RECORD                        { type variant permettant d'utiliser}
             CASE BOOLEAN OF               { les variables au choix comme etant}
                   TRUE : (P:^INTEGER); { du type 'pointeur designant un    }
                   FALSE: (I:INTEGER ); { nombre entier' ou du type 'nombre }
             END;                          { entier' (ayant alors la valeur du }
                                           { pointeur c'est a dire une adresse }
                                           { memoire }
                                           { l'utilisation de varables de type }
                                           { variant autorise l'affichage      }
                                           { direct des valeurs des pointeurs  }

   PTART   = RECORD                        { idem sauf que le type pointeur    }
             CASE BOOLEAN OF               { designe non un entier mais une    }
                   TRUE : (P:^ARTICLE); { variable de type article          }
                   FALSE: (I:INTEGER );
              END;

   PTTAB   = RECORD                        { le type pointeur designe cette    }
             CASE BOOLEAN OF               { fois-ci une variable de type      }
                   TRUE : (P:^TABLEAU); { tableau d'articles                }
                   FALSE: (I:INTEGER );
             END;

VAR ART     : ARTICLE;

    PART    : PTART  ;

    TAB1,
    TAB2    : PTTAB  ;

    DEPART,           { definit la position de la pile au debut du programme}

    SOMMET  : PTINT  ;{ definit la position instantanee du haut de la pile  }
    CL      : CHAR   ;
    MAXART1,          { nombre maximal d'articles du tableau 1  }
    MAXART2 : INTEGER;{ nombre maximal d'articles du tableau 2  }
```

```
    J       : INTEGER;
    MEMAV   : REAL;


PROCEDURE MENU;
    BEGIN PAGE(OUTPUT);   { APPLE II  }
          WRITE(CHR(28)); { APPLE /// }
          WRITELN('PROGRAMME DE DEMONSTRATION DE CREATION ',
                  'DE TABLEAUX DE DIMENSION VARIABLE');
          WRITELN;
          MEMAV:=2.0 * (MEMAVAIL-MINIMUM);
          WRITELN('memoire disponible ',MEMAVAIL,' mots de 16 bits  soit ',
          TRUNC ( MEMAV/(SIZEOF(ARTICLE)) ),' articles ');
          WRITELN;
          WRITELN('dimension tableau 1 = ',(MAXART1+1),
                '  dimension tableau 2 = ',(MAXART2+1));
          WRITELN;
          WRITELN('valeurs pointeurs en octets');
          WRITELN('DEPART PILE= ',DEPART.I:5,'   SOMMET PILE= ',SOMMET.I:5,
                    '  TABLEAU 1=',TAB1.I:5,'  TABLEAU 2=', TAB2.I:5);
          WRITELN;
          WRITELN('1 DESTRUCTION TABLEAUX');
          WRITELN('2 CREATION    TABLEAU 1');
          WRITELN('3 INITIALISAT TABLEAU 1');
          WRITELN('4 VISUALISAT  TABLEAU 1');
          WRITELN('5 CREATION    TABLEAU 2');
          WRITELN('6 INITIALISAT TABLEAU 2');
          WRITELN('7 VISUALISAT  TABLEAU 2');
          WRITELN('esc QUITTER');
          READ(CL);
          WRITELN;
    END;

  PROCEDURE DESTRUCTION;      { destruction des variables dynamiques crees }
                              { depuis l'instruction MARK(DEPART.P)        }

    BEGIN RELEASE(DEPART.P);
          SOMMET.P:=DEPART.P;
          PART.I:=DEPART.I;TAB1.I:=DEPART.I;TAB2.I:=DEPART.I;
          MAXART1:=0;MAXART2:=0;

    END;

  PROCEDURE CREAT1;
    BEGIN WRITE('nombre d''articles max ? ');READLN(MAXART1);
          MAXART1:=MAXART1-1;                  { parce que l'indice du tableau }
                                               { commence a 0                  }

          IF MAXART1<0 THEN MAXART1:=0;
          MARK(TAB1.P);
          FOR J:=0 TO MAXART1 DO
                          NEW(PART.P);{ on cree MAXART1 fois une variable }
                                      { de type ARTICLE ce qui cree la    }
                                      { place memoire pour le tableau     }

          MARK(SOMMET.P);
    END;

  PROCEDURE CREAT2;
    BEGIN WRITE('nombre d''articles max ? ');
          READLN(MAXART2);
          MAXART2:=MAXART2-1;                  { parce que l'indice du tableau }
                                               { commence a 0                  }
          IF MAXART2<0 THEN MAXART2:=0;
          MARK(TAB2.P);
          FOR J:=0 TO MAXART2 DO
```

```
                        NEW(PART,P);{ on cree MAXART2 fois une variable }
                                    { de type ARTICLE ce qui cree la    }
                                    { place memoire pour le tableau     }
        MARK(SOMMET,P);
   END;

PROCEDURE INIT1;
   BEGIN FOR J:=0 TO MAXART1 DO
                        BEGIN TAB1.P^[J].NUMERO:=J;
                              TAB1.P^[J].NOM   :='TABLEAU 1';
                        END;
   END;

PROCEDURE INIT2;
   BEGIN FOR J:=0 TO MAXART2 DO
                        BEGIN TAB2.P^[J].NUMERO:=J;
                              TAB2.P^[J].NOM   :='TABLEAU 2';
                        END;
   END;

PROCEDURE VISU1;
   BEGIN FOR J:=0 TO MAXART1 DO
            WITH TAB1.P^[J] DO WRITELN(NUMERO:5,' ',NOM);
        READ(KEYBOARD,CL);
   END;


PROCEDURE VISU2;
   BEGIN FOR J:=0 TO MAXART2 DO
```

.../...

logma

Une informatique de gestion adaptée aux besoins des gestionnaires et réalisée par des gestionnaires.

ÉTUDIE
- **opportunité d'utilisation de l'outil micro-informatique**
- **intégration entre informatique traditionnelle et personnelle**
- **politique de la communication dans l'entreprise**

FORME
- **formation à l'utilisation de la micro-informatique**

RÉALISE
- **réalisation de programmes à la demande**

LIVRE
- **livraison de systèmes clés en main, avec des progiciels de GESTION DE STOCK, PAYE, COMPTABILITE.**

Nous sommes gestionnaires avant d'être informaticiens. L'informatique doit s'adapter à l'homme, et non l'inverse.
L'outil micro-informatique répond particulièrement bien à ce souci de qualité et d'efficacité du travail, dans des conditions conviviales.
Nombreuses références en informatique traditionnelle - divers matériels - et en informatique individuelle - principalement Apple - auprès des PME et des groupes industriels.

logma s.a. Centre La Châtaigneraie - 29, avenue de Versailles - 78170 La-Celle-St-Cloud - Tél. : (3) 918.13.07

```
                WITH TAB2,P^[J] DO WRITELN(NUMERO:5,' ',NOM);
            READ(KEYBOARD,CL);
    END;


BEGIN
      MAXART1:=0;MAXART2:=0;
      MARK(DEPART.P);          { pour pouvoir liberer toute la memoire a la }
                               { fin de l'execution du programme             }
      DESTRUCTION;
      REPEAT MENU;
            CASE CL OF '1':DESTRUCTION;
                       '2':CREAT1;
                       '3':INIT1;
                       '4':VISU1;
                       '5':CREAT2;
                       '6':INIT2;
                       '7':VISU2;
            END;
      UNTIL CL=CHR(27);
      DESTRUCTION;          { sinon stack overflow  }
END.
```

Courrier des clubs

Club Observatoire Orion

J'anime un cercle d'astronomie basé sur la correspondance avec l'étranger : nous avons plus de 160 contacts à travers le monde, dont 70 % aux États-Unis. En cinq mois, nous avons réuni plus de 400 listings qu'il nous faudra encoder.

Nous cherchons à rassembler tous les listings que les amateurs ont écrit, se rattachant à l'astronomie et à la technique radio amateur, tant théoriques que pour les nuits d'observation et d'écoute.

Nous avons déjà quelques 400 Ko tournant sur Apple II (DOS 3.3). Pouvez-vous nous aider, échanger avec nous ?

D'un autre côté, nous terminons un ouvrage consacré à l'astronomie d'amateur, en plus de 650 pages et 450 clichés. Un chapitre est consacré à la micro-informatique. Vos listings peuvent encore s'incorporer à l'ouvrage, et rester sous votre copyright.

Thierry Lombry - Club Observatoire Orion, Tienne aux Pierres 94, B. 5150 Wepion, Belgique.

Ma Pomme

Les réunions régulières du club ont été déplacées au troisième mercredi de chaque mois.

Pour tous renseignements, contacter Jean-François Duvivier, au (1) 558.05.78, le soir (éventuellement tard).

Ma Pomme - 6, rue Paul Saunière, 75016 Paris.

LIBRAIRIE INFORMATIQUE

LA NACELLE

ÉLECTRONIQUE • AUTOMATISME • MICROPROCESSEUR

TOUS OUVRAGES ET ABONNEMENTS

FRANÇAIS ET ETRANGERS

Distributeur exclusif pour la France des manuels techniques du Réseau Calvados

Tous les ouvrages français ou étrangers signalés dans cette revue peuvent être obtenus ou commandés à La Nacelle

2, rue Campagne-Première 75014 PARIS - Tél. 322 56 46

Métro Raspail - Parking à la hauteur du 120 bd du Montparnasse

ouvert tous les jours lundi compris, sans interruption de 9 h 30 à 18 h 50, samedi fermeture à 17 h 50.

The Last One à l'essai

André Babeanu

The Last One est un programme qui se donne pour devise : « Je programme pour vous ! ». Au premier abord, ça a l'air vrai, mais ça ne l'est que dans les limites d'un certain nombre de restrictions.

Au lecteur déjà initié à la programmation, le premier contact avec le manuel d'application (en anglais) donne une impression très favorable. Les explications sont claires et précises. Même le non-initié y trouvera son compte, car les termes risquant de poser problème sont expliqués de façon claire.

L'initiation au programme se fait à l'aide d'un exemple, introduit pas à pas. On ne vous explique pas toujours immédiatement les instructions présentées, mais, au bout d'une heure de travail, on a un programme de saisie de données pour fichier qui tourne. Une surprise cependant : il y a de légères différences entre ce que l'on voit à l'écran et la description qu'en fait le manuel d'application. Cela tient au fait que ce dernier est unique pour toutes les versions de The Last One. Un petit livret, fourni en annexe, spécifie les caractéristiques particulières de la version Apple II. Tout rentre alors dans l'ordre.

En fait, l'apprentissage est très rapide, du moins pour ceux qui ont déjà de bonnes notions de programmation. Quant au parfait néophyte... il ferait mieux de s'abstenir. En effet, pour utiliser The Last One valablement, il faut savoir faire l'analyse du programme que l'on désire obtenir et en dresser l'organigramme. A partir de là, et c'est cela le grand atout de ce progiciel, le programme BASIC est généré sans la moindre faute de syntaxe. En d'autres termes, vous définissez les étapes à suivre et vous obtenez un programme qui les exécute. C'est déjà non négligeable pour tous ceux qui sont fâchés avec leurs virgules, deux-points et autres séparateurs ou identificateurs...

Mais, pour en revenir à nos restrictions, dès que l'on sort du tronc commun des BASICs pour exploiter à fond les possibilités de l'Apple (T.L.O. le permet), il faut connaître celles-ci et les écrire telles qu'elles seront dans le programme final. C'est le cas pour les fonctions graphiques, formules, appels à des routines...

De fait, The Last One est un programme qui s'adresse réellement à des gens qui possèdent déjà des connaissances en BASIC.

Mais alors, qu'apporte-t-il ? Avant tout, la certitude, si on ne s'est pas trompé en définissant la logique du programme, que celui-ci marchera du premier coup. Ensuite, pour tout ce qui est traitement de fichiers, description d'écrans d'entrée-sortie de données et branchements... une programmation grandement facilitée et accélérée, car des fonctions spécifiques y sont consacrées. Enfin, mais il faut introduire là des réserves, un gain de temps au niveau de la mise au point des programmes.

Pour comprendre les raisons de ce gain de temps et de nos réserves, regardons comment The Last One fonctionne. D'abord, grâce au menu principal, on établit un organigramme (flowchart) dans lequel on définit dans leur succession la nature des opérations (choix de menu, opérations sur fichiers, calculs...) sans s'occuper des opérandes. Cette phase est rapide et facile à mettre en œuvre. Les possibilités d'édition (création et suppression de lignes) sont très bonnes, malgré un scrolling un peu lent.

Le flowchart est la seule partie du projet de programme que l'on puisse conserver pour modification ultérieure, hormis le listing final en BASIC.

La phase suivante consiste à « coder le programme », c'est-à-dire définir les endroits où s'effectueront les branchements, la description des écrans, les formules pour les calculs... C'est une étape un peu fastidieuse car ceci est fait ligne par ligne de l'organigramme : les temps d'attente entre les demandes de renseignements sont assez longs ; on finit par connaître trop bien le message « Working ! Please wait ».

C'est à cet endroit que se situent nos principales réserves. Si vous n'avez fait aucune erreur (de logique — oubli d'un test ou d'un branchement — car les autres ne sont pas admises par T.L.O.), c'est parfait.

Mais supposons que vous ayez oublié une ligne dans l'organigramme, un test du type « Êtes-vous d'accord ? » par exemple. Il vous faudra alors modifier celui-ci en conséquence, ce qui est facile, mais il faudra ensuite recommencer toute la phase de codage. Ce qui revient quasiment à doubler le temps de mise au point du programme, car c'est cette phase qui prend l'essentiel du temps.

C'est là le défaut majeur de The Last One. S'il ne faut pas recommencer plus d'une fois, le gain de temps au niveau de la mise au point du programme reste cependant appréciable.

Mais, dira-t-on, ne peut-on directement faire les corrections sur le listing engendré ? Techniquement, c'est parfaitement possible. Mais en pratique quasiment irréalisable...

Premièrement, il reste alors un programme BASIC à analyser pour trouver les endroits à modifier, ce qu'on voulait justement éviter en utilisant T.L.O.

Deuxièmement, comme on le constatera sur l'extrait de listing présenté ci-dessous, la complexité du programme engendré a de quoi décourager même le programmeur expérimenté.

En effet, d'une part il n'est pas du tout documenté (les seules REMs sont constituées par l'organigramme inséré en début de listing), d'autre part The Last One génère ses propres sous-programmes, qu'il n'est pas du tout évident de décortiquer. Parmi ceux-ci, il y en a d'ailleurs de très intéressants : la routine d'INPUT, celle de traitement des erreurs (automatiquement intégrée dans tous les programmes).

Notons aussi que, lorsqu'une description d'écran se retrouve identique à elle-même en plusieurs points du programme, The Last One ne sait pas le reconnaître : on la retrouve par conséquent en plusieurs exemplaires dans le listing. Pour cette raison comme pour d'autres, les listings sont relativement

longs. A titre d'exemple, le programme de l'article « Notions de base sur les fichiers » du Pom's 4 prenait 13 secteurs sur une disquette. Réécrit avec The Last One, il en prend 31. Les temps d'exécution s'en ressentent forcément (peu, il est vrai, pour cette taille de programme, mais cela peut devenir gênant).

Il a fallu moins de trois heures pour rendre le programme opérationnel avec The Last One, malgré la nécessité de recoder une fois à cause de l'oubli de positionnement d'un pointeur. Ce dernier aspect fait de The Last One un programme assez attrayant pour la grande majorité des programmeurs « moyens ». Le débutant sera un peu frustré de ne pas pouvoir en tirer le maximum sans apprendre le BASIC - T.L.O. n'est pas un outil pédagogique. Quant au programmeur chevronné, il lui sera plus rapide et efficace de programmer lui-même. Aux autres, je laisse le soin de juger si l'aide, il est vrai non-négligeable, que peut leur apporter ce progiciel vaut son prix de 2 500 à 3 000 francs.

THE LAST ONE : extrait de programme

```
380  PRINT  MID$ (K$ + "." + K$,1,3 * S)
       ;:S = S -  NOT  NOT S:A$ =  MID$
       (A$,1,S): GOTO 370
390  PRINT  MID$ (S$,1,ML - S): RETURN
400 DC = 0:MC = 0:D = 0:S = 0:FD = DL =
       99:A$ = "": PRINT  MID$ (U$,1,ML)
       ;: IF DL AND DL - 99 THEN  PRINT
       "." MID$ (U$,1,DL) MID$ (B$,1,DL
       + 1 -  NOT ML);
410  PRINT  CHR$ (8);
420  GET K$:K =  ASC (K$ + B$): ON K = 8
       GOTO 480: ON K = 13 AND  LEN (A$
       ) GOTO 500: ON (K ( 45 OR K ) 57
       OR K = 47) GOTO 420: IF K = 46 TH
       EN  ON D OR  NOT DL GOTO 420:D =
       1: PRINT  MID$ (B$,1,ML +  NOT ML
        -  NOT MC);:A$ =  MID$ ("-",2 -
       S) +  STR$ ( ABS ( VAL (A$))):MC
       =  LEN (A$) - 1: IF DL ( 99 AND M
       L THEN  PRINT  RIGHT$ (S$ + A$,ML
       );
```

C.O.R.P. à l'essai

André Babeanu

Le progiciel C.O.R.P. se compose de deux disquettes. Au sens strict du terme, le nom de « générateur de programmes » qu'il se donne est justifié dans la mesure où ce qu'on obtient en fin de compte est bien un programme Applesoft. Seulement, on ne peut pas créer avec cet outil n'importe quel type de programme. Les auteurs d'ailleurs le précisent dans la documentation : C.O.R.P. est un générateur de programmes pour la création et l'exploitation des bases de données. Voyons donc ce qu'il en est exactement.

Premier contact avec le produit : surprise agréable, le manuel d'application est en français, les explications sont claires et simples. Il est cependant primordial de tout lire avec attention et de ne pas essayer le programme avant d'avoir lu le mode d'emploi dans son intégralité. Et pour cause : la protection contre les copies pirates repose sur un petit boîtier qu'on enfiche dans le connecteur de jeux, faute de quoi, paraît-il, la disquette s'autodétruit (je n'ai pas essayé et ne vous conseille pas de le faire). Une fois cela fait, le travail sérieux peut commencer.

L'utilisation est d'une facilité étonnante. Prenons par exemple le cas d'un programme de gestion de fichier. Il suffit de définir votre écran pour les entrées/sorties de données, un peu comme en création de masque avec MEM/DOS (ex-M/DOS). Pour chaque ligne non vide, l'utilisateur définit les étiquettes de ses variables (nom, adresse,...), leur nature (alphanumérique, numérique,...) et le nombre maximum d'enregistrements admis. C'est tout ! C.O.R.P. se débrouille ensuite tout seul comme un grand et, cinq minutes plus tard, a mis en place un système complet de gestion de fichier : création, modification, recherche (rapide et lente) ou suppression d'un enregistrement. C'est remarquable...

Qui plus est, il est possible de définir les champs des variables caractère par caractère ; ce qui permet d'avoir une variable du type : « CODE ARTICLE AA# #A », où A représente un caractère alphanumérique et # un chiffre.

Quant à la différence de vitesse dans la recherche, elle vient de ce que, pour la variable définie comme étant la clé, C.O.R.P. intègre au programme généré un système de recherche à table de référence (se reporter à l'article « Notions de base sur les fichiers » du Pom's 4 pour de plus amples explications).

Un fait est à signaler : le programme crée une table de référence pour le maximum d'enregistrements possibles ou autorisés dès le début. De ce fait, même s'il n'y a aucun enregistrement dans le fichier, si on n'a pas fixé de limite de taille (fichier sur toute la disquette), la table de référence prend 25 secteurs.

Récapitulons ! Il m'a fallu 15 minutes (et je suis large) pour créer le masque d'écran, puis encore cinq minutes de travail pour C.O.R.P., et nous avons un programme Applesoft complet, qui marche (dans tous les cas) et permet, grâce à son système de recherche, de retrouver rapidement les articles par une clé, ou lentement tous ceux correspondant à une combinaison de critères en nombre non limité. Remarquable !

Le seul problème (mais en est-ce un réellement ?) est qu'on ne peut pas en obtenir moins... Le programme ainsi créé prend 45 secteurs (contre 13 pour le même programme à la main). Il est vrai qu'il est plus performant par ses possibilités de sélection multicritères.

Jetons un coup d'œil sur le listing engendré. Il est long, bien sûr... mais il est très clair. Il n'y a presque jamais plus d'une instruction par ligne ; il est facilement lisible et en conséquence aisément modifiable (je n'ai pas eu à le faire cependant). On trouvera ci-dessous un exemple.

S'il n'avait que cela, C.O.R.P. serait seulement un bon produit comme on en trouve encore assez souvent. Mais il y a plus... Tout d'abord, il est possible de cross-référencer les fichiers (i.e. faire des renvois). Mais surtout, il y a des possibilités de création d'autres types de programmes :

— programmes d'impression d'états en tous genres ;
— programmes d'édition de formulaires pré-établis avec recherche des données dans le fichier. Très pratique par exemple pour établir des factures ;
— programmes de reclassement de fichiers selon un champ quelconque en ordre croissant ou décroissant ;
— et surtout programmes de mise à jour de tout ou partie d'un fichier selon des formules données. On peut, à l'aide de ceux-ci, appliquer un pourcentage de hausse sur les prix de tous les articles du fichier, ou seulement pour certains articles sélectionnés selon des critères à définir ;
— enfin, la possibilité de reconstituer des fichiers partiellement détruits par une fausse manœuvre.

Comme on le constate, C.O.R.P. est une panoplie très complète de gestion de base de données. En fait, ce n'est même rien de plus. La seule chose qui le distingue d'une base de données du type CX Multigestion ou DB Master, c'est que vous pouvez définir vos écrans comme vous l'entendez et que vous obtenez un programme Applesoft complet et autonome dont les fichiers sont lisibles par tout autre programme du genre. C'est déjà beaucoup, diront certains.

En effet, ça l'est, surtout si l'on prend en considération la facilité d'emploi de C.O.R.P. (on peut réellement s'en servir sans rien connaître en informatique), la vitesse de mise au point des programmes et surtout leurs performances de rapidité et fiabilité.

De plus, le listing étant facile à lire, on peut facilement modifier le programme, par exemple pour traduire en français les commandes (en anglais dans le programme généré).

En conclusion, pour quiconque travaille souvent avec des fichiers, bases de données et autres bêtes du mêmes genre, C.O.R.P. est un excellent outil. Nous aimons beaucoup.

C.O.R.P. coûte 2 990 F H.T. pour le système complet, 2 100 F H.T. en version de base.

CORP : extrait de programme

```
2090  INPUT "";YU$
2100  IF  MID$ (YU$,1,1) = "S" THEN 3260
      O
2110  IF  MID$ (YU$,1,1) = "C" THEN 2130
2120  GOTO 2070
2130  IF KC$ = "*" THEN  GOSUB 300
2135  IF KC$ <  > "*" THEN  GOSUB 20
2140  IF IM = 0 THEN 902
2150 F7 = 1
2160  GOTO 1230
2170 REC$ = "*"
2180  FOR ITEC = 1 TO 4
2190 REC$ = REC$ + FIELD$(ITEC)
2200  NEXT ITEC
```


E.A.O.

Enseignement Assisté par Ordinateur pour Ordinateurs
MOPPE D'ANPICO - SILEX - ITT 2020 - APPLE II - APPLE II +

COURS EN FRANCAIS

de BASIC
. APPLESOFT - APPLE II (+ Carte APPLESOFT)
- APPLE II +
- APPLE III
- MOPPE D'ANPICO (+ APPLESOFT en ROM)
. PALSOFT - ITT 2020 (+ Carte PALSOFT)
. SILROM - SILEX (de Léanord)

(Sur DISQUETTES 5 1/4 Pouces - DOS = DOS 3.3 - Mémoire = 48 K.)

1/ COURS 1 (BASIC) - En Français
. TRES PROGRESSIF - Ne nécessite aucune connaissance préalable en informatique.
. Pour débutants et non débutants. TOUT Y EST EXPLIQUÉ.
. 20 Leçons - Environ 10 à 12 Heures de cours.
. 80 Exercices commentés, expliqués, résolus, exécutés.
. 140 Questions notées sur 20, par groupes. - Réponses aux questions.
. GRAPHISME BASSE et HAUTE RESOLUTION.
. Défilement automatique du Cours avec arrêts et reprises possibles en cours de leçon.
. De nombreux exercices peuvent être réexécutés autant de fois que vous le souhaitez. Ainsi, vous pouvez obtenir les mêmes résultats ou des résultats différents en faisant varier les données d'entrée.

A LA FIN DE CE COURS, VOUS SAUREZ PROGRAMMER.

2/ COURS 2 (BASIC +) - En Français
. Philosophie générale identique à celle du COURS 1 (BASIC).
. 25 Leçons - 12 Heures de cours minimum.
. 120 Exercices commentés, expliqués, résolus, exécutés.
. 160 Questions notées sur 20, par groupes - Réponses aux questions.

3/ CONTRATS-LOCATION du COURS BASIC pour:
. Etablissements d'Enseignement.
. Etablissements de Formation payante.
. Centre de Recherches, Laboratoires, Centre d'Essais,....
. Comité d'Entreprises.

PRIX (T.V.A. comprise)
. COURS 2 (BASIC +) 375 FF.
Prix donnés à titre indicatif, pouvant être modifiés sans préavis.

ANDRE P. PINOT
8 Allée BUFFON
91000 EVRY-COURCOURONNES

R E V E N D E U R S,
C O N T A C T E Z - N O U S.

Erratum

Hervé Thiriez

Il me faut faire amende honorable au sujet du dernier numéro de Pom's : ayant dû le terminer au mois d'août, en même temps que mon livre sur Visicalc, j'ai laissé passer certaines erreurs dans la rédaction et le contrôle des articles. Je ne dispose pas sur mon Olympia des caractères < , > et # , et il faut en effet les créer manuellement. Voici les corrections, dans l'ordre de lecture de Pom's, auxquelles nous avons ajouté des compléments d'information.

1. Le clavier magique

Page 4, colonne 2, ligne 4, il faut lire : #<DEPART >, <INCREMENT >.

2. La programmation facilitée

Page 23, colonne 2, lignes 6 et 7, lire : X < 128 diminue la durée, X > 128 augmente.

15 lignes plus bas, lire « affichant PRINT FN ARR(X) ».

Afin de formater un écran entier avant de placer les INPUT et de créer une commande de formatage pur, il suffit de quelques instructions en plus et de créer une commande de formatage pur. On supprime « INPUT A$ » et le paramètre longueur devient obligatoire, car le « 1 » servira à identifier la commande. Il faut pour cela :

— Modifier les lignes suivantes :

```
 72 LDY #0
 73 STY LONGUEUR
 74STY $6
123 BEQ MASQUE
```

— Insérer les lignes suivantes :

```
 54 CMP#'1'
 55 BEQ INPUT

 75 PHA
 76 CMP #'1'
77 BEQ INPTL

115 DEC $BF

122 INC $BF

137 MASQUE PLA
138        CMP '1'
139        BEQ GTRTS
```

3. HAIFA

Tout d'abord, il faut intervertir les pages 41 et 42. Nous espérons que la plupart des lecteurs s'en étaient aperçus...

Page 35, col. 2, § 3, ligne 5, lire « # $A5 ».

Paragraphe suivant, lignes 4 et 5, lire « fait un bip, déconnecte les routines spéciales d'entrée/sortie, puis la reprogrammation de HAIFA intervient — le RESET agit comme CTRL-C : il fait".

Page 36, la pile. Lire PUSH > et PULL > (pas de signe "=").

Page 36, à propos de & GOTO, & GOSUB, & RESTORE et & NEW : comme on peut utiliser des expressions numériques à la place des nombres, il vaut mieux ne plus utiliser le RENUMBER !

Page 37, le « INPUT anything ». Une propriété supplémentaire a été omise : & INPUT « [chaîne] » ; [nom de variable], [exp. num.]
& INPUT [nom de variable], [exp. num.].

Ces commandes agissent comme & INPUT, mais ajoutent derrière le curseur un nombre de points correspondant à l'expression numérique (de 1 à 255). Ce nombre de points indique la longueur maximale de la chaîne autorisée : en cas de dépassement, la chaîne sera tronquée en conséquence.

Page 42, bas de colonne 1 : le type 9 a pour effet « T⇐⇒E ».

Page 43, col. 1, §6, lignes 6 et 7 : ...« < » pour la gauche, « > » pour la droite...

Page 43, tracer des points et des droites : le HPLOT, le X HPLOT et le N HPLOT peuvent être prolongés par chacune des continuations [ex.1], [ex.2] ou TO [ex.1], [ex.2] ou ON [ex.3].

Page 45, col. 2, §4, lire PR# et IN# au lieu de PR= et IN=.

Page 46, §2 et page 47, ligne 1, lire CTRL-SHIFT-N au lieu de CTRL-%.

Page 46, dernier §, lire CTRL-SHIFT-P au lieu de CTRL-à.
Adresse des routines : GETLIN ($FD6A), RDKEY($FDOC) et COUT1($FDFO).

Page 48, col. 2, §4, ligne 1 : « Cette routine court-circuite l'entrée de » ; ligne 9 : lire PRINT CHR$(4)« IN# 0 » : « IN » et non « PR ».

Page 51, col, 2, §2 : ligne 3 : « ...tables après les fichiers » ;, ligne 9 : « du code (HAIFA.CODE en 48K ou HAIFA. CODE 1 en 64K), il ne ».

Remarques sur la disquette Pom's 5.

Le programme source est constitué de deux fichiers LISA 2.5 version 64K enchaînés par ICL car l'ensemble était trop gros pour tenir en mémoire en un seul fichier. L'assemblage de ces fichiers sauve automatiquement les deux fichiers source. Là encore, il fallait deux fichiers distincts car LISA ne laisse que 4K pour le fichier généré.

L'assemblage des deux versions de HAIFA (48K et 64K) a été réalisé avec LISA en version 64K. Pour pouvoir lister les fichiers avec LISA 48K, il a fallu découper HAIFA.TEXT 1 et HAIFA. TEXT 2 en deux parties chacun (suffixes A à D). Ces quatre fichiers s'enchaînent avec ICL, mais ne sont pas assemblables en 48K car la table des symboles est saturée.

Celle-ci a en effet une longueur de $800 en LISA 48K et de $1500 en LISA 64K. HAIFA est donc beaucoup trop gros pour être assemblé en 48K !

4. Conversion Pascal/ BASIC/Pascal

Nous adressons nos excuses aux lecteurs ayant essuyé des difficultés lors de l'utilisation des programmes de conversion Pascal/BASIC/Pascal, fournis avec les disquettes Pom's 2 à 4, et commentés dans le numéro 3.

Pour ceux d'entre vous qui reçoivent les disquettes, utilisez le programme de transfert de BASIC vers Pascal de la disquette Pom's 5, dans laquelle le bug avait été corrigé, ou celui de la disquette 6, bien entendu.

Ce bug avait été malheureusement absent de nos essais, dans lesquels nous avions remis les programmes sur une disquete Pascal vierge. La conséquence du bug est qu'une partie du programme Pascal risque d'être absente en fin de course.

Le remède est le suivant, à appliquer aux programmes présentés par Gilles Mauffrey dans le Pom's 3 :

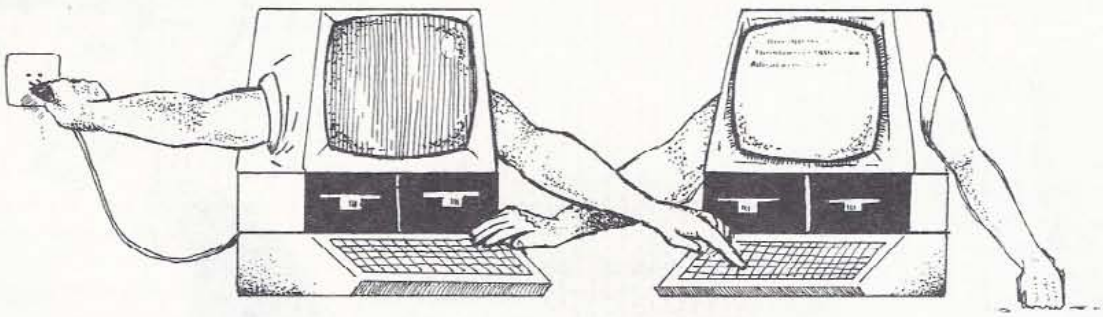
1. Dans PASCBAS.OBJO, ligne 288, lire CMP # $10.

Donc, dans le code objet, faire AE3 : 10.

2. Dans BASPASC.OBJO (sur disquette uniquement), ligne 449, lire CMP # $10.

Donc, dans le code objet, faire C29 : 10.

3. Avant toute utilisation, nettoyer la mémoire avec :
2000 : 0
2001<2000 . 8000 M.



"C.O.R.P." et "The Last One"

AGRAPH

VISICALC, VISICALC...

Avec le modèle Visicalc, vous pouvez créer sur votre P.S.I. (Petit Système Individuel) un tableau comportant des titres, des valeurs et des formules. Dès que vous changez une des valeurs numériques, tout le tableau se met à jour en fonction des formules. Cet ouvrage vous guide pas à pas dans l'utilisation de Visicalc, à l'aide d'exemples progressifs. De nombreux cas d'application sont présentés dans la seconde partie, au rythme d'un chapitre par cas : échéancier de remboursement, feuille d'impôt, gestion de copropriété, paye, facturation, tableau de bord... Chaque exemple sert à introduire de nouvelles instructions ou astuces d'utilisation.

176 pages - 82,00 FF/630,00 FB

P.S.I.

P.S.I. DIFFUSION
41-51, rue Jacquard
BP 86 - 77400 Lagny-s/Marne
FRANCE
Téléphone (6) 007.59.31
P.S.I. BENELUX
5, avenue de la Ferme Rose
1180 Bruxelles
BELGIQUE
Téléphone (2) 345.08.50

En Espagne
P.S.I. Iberica
Ferraz 11
Madrid 8
Tél. : 247.30.00

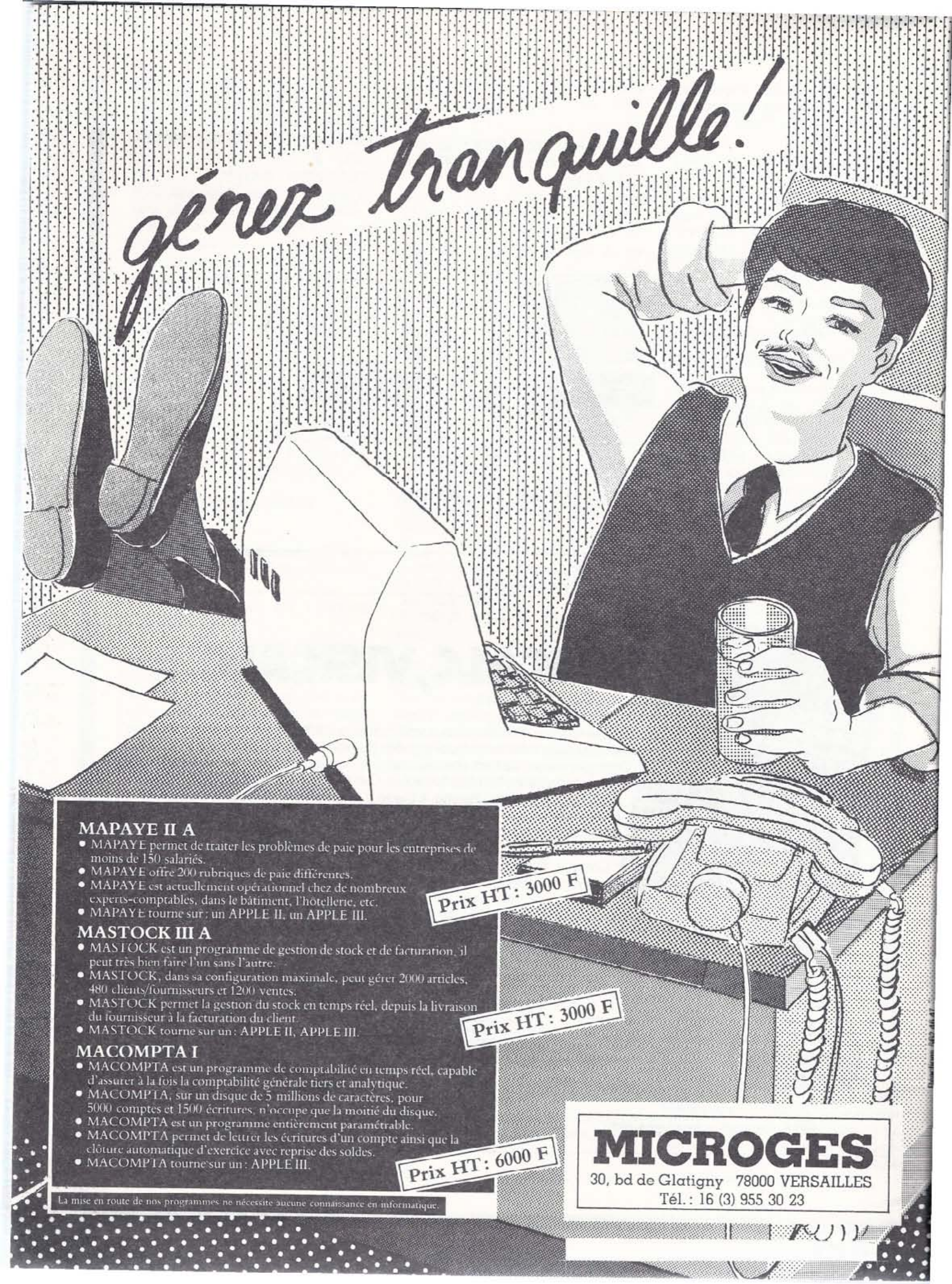
au Canada
SCE Inc
3449 rue Saint-Denis
Montréal Québec H2X3L1
Tél : (514) 843.76.63

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

DESIGNATION	NOMBRE	PRIX
(par avion : ajouter 8 FF (75 FB) par livre)	TOTAL	

NOM _____ PRENOM _____
rue _____ N° _____
Code post [| | | | |] Ville _____

P.VZ.12



MAPAYE II A

- MAPAYE permet de traiter les problèmes de paie pour les entreprises de moins de 150 salariés.
- MAPAYE offre 200 rubriques de paie différentes.
- MAPAYE est actuellement opérationnel chez de nombreux experts-comptables, dans le bâtiment, l'hôtellerie, etc.
- MAPAYE tourne sur : un APPLE II, un APPLE III.

Prix HT : 3000 F

MASTOCK III A

- MASTOCK est un programme de gestion de stock et de facturation. Il peut très bien faire l'un sans l'autre.
- MASTOCK, dans sa configuration maximale, peut gérer 2000 articles, 480 clients/fournisseurs et 1200 ventes.
- MASTOCK permet la gestion du stock en temps réel, depuis la livraison du fournisseur à la facturation du client.
- MASTOCK tourne sur un : APPLE II, APPLE III.

Prix HT : 3000 F

MACOMPTA I

- MACOMPTA est un programme de comptabilité en temps réel, capable d'assurer à la fois la comptabilité générale tiers et analytique.
- MACOMPTA, sur un disque de 5 millions de caractères, pour 5000 comptes et 1500 écritures, n'occupe que la moitié du disque.
- MACOMPTA est un programme entièrement paramétrable.
- MACOMPTA permet de lettrer les écritures d'un compte ainsi que la clôture automatique d'exercice avec reprise des soldes.
- MACOMPTA tourne sur un : APPLE III.

Prix HT : 6000 F

La mise en route de nos programmes ne nécessite aucune connaissance en informatique.

MICROGES

30, bd de Glatigny 78000 VERSAILLES

Tél. : 16 (3) 955 30 23

Micro-informations

Logiciel

Micro Informatique Service a rebaptisé, pour des raisons de propriété commerciale, le M/DOS (Pom's 4) en MEM/DOS.

Vous pouvez maintenant constituer votre réseau local avec MEMNET, à base d'Apple II et Apple III en cohabitation. Maximum de 127 postes (Pom's vous offre une réduction pour 127 abonnements !), avec une distance de 500 mètres maximum entre chaque poste.

Félicitations à M.I.S., qui vient d'obtenir le prix « Système » de la Pomme d'Or et de créer sa filiale aux USA. Il faudrait que le renvoi de la balle aux Américains se fasse un peu plus souvent.

M.I.S. 3, rue Meyerbeer.
06000 Nice. Tél. : (93) 87.74.67.

La société C.D. Soft commercialise un logiciel de gestion de cabinet dentaire. Ecrit par un chirurgien dentiste et baptisé AGATHA, ce programme assure la gestion d'un fichier de 200 patients en cours de soins (plans de traitement, soins réalisés...), l'édition des feuilles de S.S., des devis de prothèses, des ordonnances..., la gestion des recettes, dépenses et des impayés.

Coût du logiciel : 11 600 FF TTC (prix indicatif).

C.D. Soft. Dr Pierre Gaussen.
59 bis, rue de la Biche
30000 Nîmes. Tél. : (66) 28.09.46.

La société MEDEE propose un logiciel d'analyse statistique et de sélection de données (profils, moyennes, écart-type, droite et coefficient de corrélation, analyse à partir de variables obtenues par calcul sur des variables de base, recherche multicritère...).

MEDEE. Logiciel ALED.
9, rue du Professeur Florence.
69003 Lyon. Tél. : (7) 854.31.95.

Autre système d'analyse de données scientifiques ou marketing, P.C.S.S., distribué par Alpha-Systèmes. Structure paramétrable des questionnaires traités, tests paramétriques (variance, corrélation, régression linéaire...), tests non paramétriques (Wilcoxon, Friedman...), plans factoriels, analyses multidimensionnelles (composantes principales, régression multiple...), vous sont offerts à des prix variant de 7 000 à 29 500 FF HT selon les options.

Alpha-Systèmes. 51, rue Thiers.
38000 Grenoble. Tél. : (76) 47.80.67.

Microgrès distribue à présent un progiciel de comptabilité sur Apple III avec Profile (25 000 comptes et 25 000 écritures), entièrement paramétrable. Prix : 6 000 FF HT.

Microgès. 30, bd de Glatigny.
78000 Versailles. Tél. : (3) 955.30.23.

Matériel

ZH Computer propose une large gamme de périphériques pour Apple II :

- Lecteur de cartes magnétiques (carte bleue...).
- Lecteur et encodeur de cartes magnétiques.
- Lecture et impression de «code barre».

- Imprimante de ticket de caisse et tiroir-caisse connecté sur Apple.
- Carte de digitalisation d'image et caméra de surveillance.
- Système de saisie optique de formulaires, questionnaires...

ZH Computer. 34, rue Vivienne.
75002 Paris. Tél. : (1) 233.72.07.

Jod Electronique commercialise depuis peu une imprimante-table traçante permettant le tracé des graphiques en couleurs sur documents opaques ou transparents. Prix de l'imprimante (sur la base du dollar à 7 FF) : 8 800 FF HT ; logiciel de 665 FF HT à 1 377 FF HT par programme.

Jod Electronique. 9, rue Noblet.
92500 Rueil Malmaison.
Tél. : (1) 749.70.44.

Alpha-Systèmes (cf. ci-dessus) présente également des cartes d'extension RAM 32, 64 et 128 K, livrées avec des logiciels permettant de reloger Applesoft Integer et/ou DOS dans la carte, de lire ou d'écrire routines, tableaux ou programmes dans la carte, ou de gérer une ou plusieurs de ces cartes par POS ou CP/M.

Un autre logiciel permet en outre d'utiliser ces cartes avec Visicalc. Prix de 2 500 à 6 500 FF HT.

La même société propose par ailleurs une carte d'interface «bufferisée» permettant la gestion autonome d'imprimante (travaux simultanés de l'imprimante et de l'Apple), avec prise en charge de la mise en page et copie d'écran graphique (prix : 2 500-3 000 FF HT).

La société B.I.P. a baissé le prix de ses cartes : prix public de 5 025 FF HT pour la carte 128 K DF (voir Pom's 5) avec son logiciel, y compris le Disk Emulator et l'extension Visicalc. La carte RAM 64 KC est maintenant proposée à 3 050 FF HT.

B.I.P. 25, rue du Mont Cenis.
75018 France. Tél. : (1) 264.02.32.

Général Automation diffuse des unités de disques souples 8" de capacité unitaire de 10 méga-octets, utilisant des cartouches souples amovibles interchangeables. Prix unitaire : 6 000 $.

General Automation.
Les Mercuriales.
40, rue J.-Jaurès. 93176 Bagnolet.

Publications

Les Éditions Eyrolles (61, bd St-Germain, 75240 Parix Cedex 05) présentent trois nouveaux ouvrages :

— Votre Gestion Avec BASIC, par Guy Ladevie (138 pages - 65 F). Présentation d'exemples d'utilisation du micro-ordinateur (comptabilité personnelle ou d'entreprise, état bancaire, gestion de fichiers...) et de méthodes nécessaires pour une bonne programmation.

— CP/M et sa famille, par P. Dax (144 pages - 65 F). Description et analyse de CP/M, de ses extensions et de son environnement logiciel, et des conséquences de l'avènement des processeurs 16 bits.

— L'Assembleur facile du Z 80, par O. Lepage (120 pages - 60 F).

Courrier des lecteurs

Enfin une revue en français pour l'Apple II. Après beaucoup de temps passé à désassembler et comprendre l'Applesoft, aidé par quelques articles de revues américaines. Quel plaisir enfin...

Je vous fais parvenir un petit exemple de mon «travail», qui peut-être intéressera vos lecteurs. Il s'agit d'une tortue (moins puissante que celle du Pascal) utilisable à partir de l'Applesoft avec « l'ampersand ».

J'ai passé de nombreuses nuits sur l'Applesoft et ai trouvé un ver dans la pomme qui ne semble pas connu (ni grave).

Tout numéro de ligne supérieur à 63 999 provoque une SYNTAX ERROR, sauf les numéros compris entre 437 760 et 440 319 qui plantent l'Applesoft ??? Le saviez-vous ?

Pour terminer, une question (je connais moins bien le DOS que l'Applesoft). La séquence BLOAD «nom de programme» et CALL «début de programme» ne fait pas la même chose qu'un BRUN «nom de programme». Il semble qu'après un BRUN, en fin de programme, le RTS ne rende pas la main dans certains cas. L'avez-vous remarqué ?

Jacques Duma
26-28, rue de Clichy, 75009 Paris

Comme vous pouvez le voir dans ce numéro, nous ne nous sommes pas privés de mettre vos connaissances à contribution.

En ce qui concerne les numéros de lignes entre 437760 et 440319, ce que vous avez remarqué est la conséquence d'un bug Applesoft (du moins, c'est ce que nous croyons après analyse). Il s'agit de la procédure LINGET ($DA0C) ; il faudrait remplacer [DA1E : B0 D4] par [DA1E : B0 D6]. Vous pouvez vous amuser à analyser la raison de cette modification.

Bien entendu, la modification ne peut être effectuée que sur une version d'Applesoft accessible, donc en RAM.

Quant à votre dernière question, le RTS ne rend pas la main chaque fois que le programme en assembleur a modifié les entrées/sorties écran ou clavier. C'est expliqué dans la documentation de HAIFA, qui provoque typiquement ce problème.

Vous avez publié dans Pom's qu'il existe des patches pour Applewriter, en particulier pour le mettre en 80 colonnes. Pouvez-vous me dire ce qu'il faut faire et me donner des tuyaux à propos d'autres transformations de ce programme ?
Je vous donne celle que j'ai trouvée pour obtenir, avec Applewriter I et la ROM minuscules, des minuscules à l'écran. Dans le TEDITOR, il faut entrer :
0AE6 : 20 69 18
1869 : 48 C9 E0 90 02 29 BF C9 C0 90 02 09 20 C9 20 B0 02 09 C0 91 28 C8 68 60
C'est le recodage des caractères qui se fait uniquement pour l'envoi à la vidéo. Le codage Applewriter est conservé en mémoire, et reste correct pour l'envoi à l'imprimante.
Ce qui m'intéresse également, c'est la possibilité d'insérer dans le texte des commandes particulières à l'imprimante, caractères de contrôle ou commandes plus complexes. Mon IDS justifie à droite et à gauche en mode proportionnel avec la commande [CHR$(27);"J,100,600,$"]. Je travaille là-dessus ; avez-vous des tuyaux à ce sujet ?
Dans un autre ordre d'idée, connaissez-vous des programmes de traitement de texte pour Apple qui utilisent la carte 80 colonnes SUP'R'TERM et acceptent ses commandes, et sachent gérer les accents circonflexes ?

Comme base de données, j'utilise General Manager. Non seulement j'en suis satisfait, mais je trouve que ce programme pratiquement inconnu soutient très bien la comparaison avec d'autres, notamment le DB Master. D'autant qu'il coûte seulement 995 francs.

Guido Benvenuto Bettiol.
2, rue des Remparts,
30800 Saint-Gilles.

Merci pour le passage en minuscules, voilà un patch très sympathique. Le seul traitement de texte qui marche à notre connaissance avec la carte SUP'R'TERM est l'Applewriter II, version 2.0. Mais nous ne connaissons pas un patch lui permettant de bien imprimer les accents circonflexes.
Applewriter II permet l'insertion de caractères de contrôle ; renseignez-vous sur les limites de ce logiciel.

Nous sommes désolés, mais nous n'avons pas de réponse aux autres questions. Nous faisons appel aux autres lecteurs de Pom's. Envoyez les réponses, s'il vous plaît !

Je tiens à vous féliciter de votre travail pour votre (notre) revue Pom's. Bien que celle-ci m'apparaisse quelque peu avancée pour le débutant que je suis, je m'y suis abonné dès son apparition.

C'est à vos connaissances et à votre compréhension que je m'adresse aujourd'hui plus particulièrement. Quelques questions restent pour moi sans réponses.

Pouvez-vous m'indiquer les modifications à apporter au programme Applewriter pour que celui-ci accepte la carte 80 colonnes Videoterm ?

Comment peut-on afficher le CATALOG des fichiers réalisés avec Visicalc en restant en mode de travail Visicalc ?

Peut-on modifier le programme Visicalc pour qu'il marche avec la carte Videoterm ?

Comment se fait-il, lorsque j'essaye de créer un fichier Visicalc sur une disquette formatée en DOS avec [/PD], que j'obtienne le message « I/O ERROR » ?

M. Philippe Guérin.
39-41, rue Saint-Fargeau, Bât A,
75020 Paris.

A notre connaissance, seule la carte SUP'R'TERM peut fonctionner avec Applewriter. Et, pour tout arranger, seule Videoterm cohabite avec Visicalc... Contacter Alpha-systèmes (voir la rubrique Micro-informations) par exemple pour cette cohabitation.

Il n'y a pas moyen d'obtenir le CATALOG à l'intérieur de Visicalc. On peut seulement faire défiler les noms de fichiers avec l'instruction [/SL] et la flèche de droite (une flèche par fichier).

Si vous êtes sur le drive 1 et que vous désirez voir les noms de fichiers sur le drive 2, vous y parvenez en entrant [/SL,D2], suivi des mêmes flèches de droite.

Il n'y a aucun problème pour sauvegarder des fichiers Visicalc sur une disquette formatée en DOS 3.3 avec l'instruction habituelle [/SSnom]. Il n'y a rien de spécial à faire. De toute façon [/PD] ou [/SS] marche de la même façon sur une disquette initialisée DOS 3.3 ou Visicalc.

Le message « I/O ERROR » survient par contre quand vous essayez, par distraction, d'effectuer une sauvegarde sur la disquette du programme Visicalc.

JCR, UN NOM. QUATRE NOUVEAUTES TROIS OUVERTURES. UNE TRANSFORMATION.

Produits	Prix JCR TTC Service Compris	Produits	Prix JCR TTC Service Compris	Produits	Prix JCR TTC Service Compris
★★PROMOTION 1 APPLE II 48K 1 DISK. 3.3. + contrôleur 1 MONITEUR VIDEO 12" 1 BOITE DE 10 DISQUETTES		CAVERN… MUSI… C…	180 F 120 F 120 F 240 F	CE 155 8K R… CE 150 …TK7) M… …N.INTERF.) …RIMANTE …T. FLOPPIES …BLE FLOPPY)	1150 F 1870 F 1680 F 6800 F 1020 F 9700 F
★★DISK II + CONTROLEUR ★★DISK II TABLETTE GRAPHIQUE ★★IM…					830 F 7450 F 1200 F 6700 F 9700 F 540 F
ACC… ★★EXT. ME… CARTE IN… CARTE LAN… SUPER CARTE… CARTE Z80					1700 F 7450 F 1720 F 6700 F
LOGICIELS D… VISICALC 16 SECTO… VISIPLOT VISITREND/VISIPLOT VISITERM DESKTOP PLAN II VISIFILE APPLE LOGO STANDARD					3800 F 4590 F 465 F 3060 F 5250 F
APPLE II CARTE SUPERTERM F… CLAVIER NUMERIC… JOYSTICK II POIGNEES DE…					2200 F 2290 F 3800 F 3800 F
★★PROM… 1 APP… 1 MO… 1 SYSTE… 1 BUSINESS… AFFAIRE EXCE…					12800 F 13560 F
SILENTYPE /// DISQUE /// 5" MONITEUR 12" /// ★★PROFILE 5MO					670 F 590 F 1200 F 690 F 620 F 590 F 555 F
VISICALC /// MAIL LIST MANAGER PASCAL APPLE WRITER /// Fr ACCESS ///				★MX 80 F… ★MX 82 FT TYPE… MX 100 TYPE III	5400 F 6000 F 8200 F
VICTOR VICTOR 16K IMP CONTROLEUR A MAIN				**IMPRIMANTES CENT…** ★★CENTRONICS 739	…0 F
				PANASONIC HHC + BASIC + ALIM	5750 F
				DISQUETTES ★★5" SF/SD MEMOREX	210 F
JEUX & PROGRAMM… BIORYTHME CONTRATAC REGATES CHATBYRINTHE ENCERCLEMENT COW-BOYS GLOUTON GOOFY GOLF MICRO-CHESS ROI D'ORDINATRIE	…F 120 F 120 F 120 F 120 F 120 F 120 F 180 F 180 F	SHAR… SHARP MZ… QUANTITE LIMITE…		**MINI-CASSETTES** MAGNETO K7 PC APPLE	380 F
		SHARP ★★PC 1211 CE 121 CE 122	950 F 130 F 900 F	**CASIO** ★★FX 702 P FA 2 INT K7 FP 10 IMPRIMANTE	1150 F 230 F 470 F
		★★PC 1500 CE 151 4K RAM	2300 F 515 F	**NOUVEAUX** SIRIUS ★★VICTOR NOUVEAU CLAVIER	35400 F 3000 F

Aujourd'hui, l'informatique est au centre de l'actualité : affaires, industries, commerce, agriculture et même particuliers sont concernés.

Aujourd'hui, JCR est au centre de l'informatique, sélectionnant les meilleurs matériels, étudiant pour vous les meilleures solutions, regroupant les meilleurs professionnels et vous offrant les meilleures garanties.

Aujourd'hui, JCR est au centre des performances : choix, prix, conseil, JCR c'est l'informatique service compris.

OUVERTURE DE J.C.R. A MARSEILLE

Marseille, 30 novembre 1982, 59, rue du Docteur Escat, un nouveau centre informatique JCR est né. Un centre qui offre à sa région les mêmes avantages que JCR Paris : tous les matériels logiciels, tous les services, l'assistance, le conseil et la maintenance sont sur place. JCR est prêt à vous aider plus rapidement et plus efficacement. JCR Marseille : l'informatique service compris avec vue sur la mer.

OUVERTURE D'UN CENTRE DE FORMATION

… utilisation des logiciels spé… programmation, etc. Ces cours réalisés par des universitaires de haut niveau, vous aide… les possibilités de votre matériel.

OUVERTURE D'UNE SURFACE PROFESSIONNELLE

Rapidité, sécurité, rentabilité, votre entreprise a tout à gagner d'une informatisation sérieuse.
Aussi, ne prenez pas de décision avant d'avoir visité le nouvel espace professionnel JCR :
100 m² de matériels et de logiciels conçus pour vous, plus le service JCR.
Matériel : les meilleurs systèmes des plus grandes marques (Apple, Sharp, Sirius, Thomson, Commodore...
Logiciel : une gamme complète en bureautique, traitement de textes, gestion, paye, facturation, stock, comptabilité, etc.
Service : plus qu'un vendeur, JCR est le véritable partenaire informatique de votre entreprise : démonstration des matériels, conseil, livraison, installation, service après-vente et service maintenance sur mesure.
Assistance : JCR met à votre service une équipe de 14 spécialistes passionnés prêts à répondre à vos questions.

LA SURFACE "INFORMATIQUE POUR TOUS" S'AGRANDIT

★★ ARTICLES EN PROMOTION EXCEPTIONNELLE. D'autres articles en promotion. Nous consulter avant tout achat.

En raison des fluctuations monétaires ces prix sont susceptibles d'être modifiés sans préavis. Nous consulter pour confirmation.

JCR **Boutique**

58, rue Notre-Dame de Lorette
75009 PARIS - Tél. 282.19.80
- Télex : 290350 F

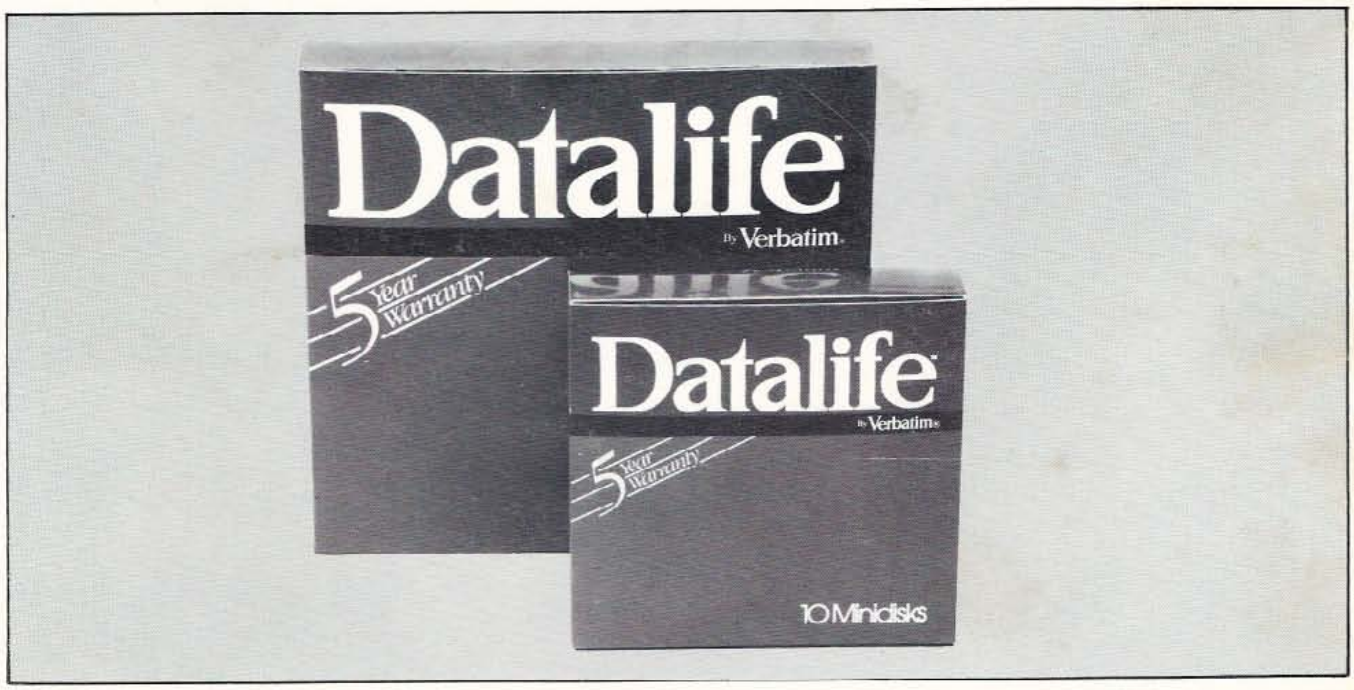
59, rue du Docteur Escat
13006 MARSEILLE
Tél. (91) 37.62.33

Vente par correspondance - Catalogue gratuit sur demande - Crédit 4-36 mois - Leasing 36-48 mois.
Horaires d'ouverture du magasin du mardi au samedi : 10 h - 12 h 45, 14 h - 19 h.
Détaxé à l'exportation.

Datalife

BY Verbatim

DISQUETTES ET MINI DISQUETTES TOUTES CONFIGURATIONS



- Certification unitaire 100 % sans erreur.
- Durée de vie : 30 millions de révolutions (standard de l'Industrie 3,5 millions de révolutions).
- Anneau de renforcement en standard sur le 5 1/4 ".
- 5 1/4 " en 48 et 96 TPI, simple et double face.

Importateur exclusif : BFI ELECTRONIQUE - 9 RUE YVART - 75015 PARIS.
Tél. 533-01-37.