JPC

JUIN 1992

NUMERO 77

Le numéro 40 F

A PROPOS DU CLUB

HP-28

HP-48

HP-95

EDITORIAL

Chers Amis,

Le mois dernier, nous vous avions annoncé, au sujet de la sortie du nouvel HP-95, qu'EduCALC organisait une opération de mise à jour de ces machines afin de les doter de 1 Mo. Cette information était erronée. En fait c'est directement HP USA qui organise cette opération. Ce qui nous ammène à poser une seule petite question : Pourquoi HP France refuse de faire la même chose ?

Autre chose, PPC Paris a été créé en décembre 1982. Ce qui veut dire que, si nos comptes sont bons, le club va bientôt accuser le vénérable âge de dix ans. Comme nous ne refusons jamais de faire une petite fête, nous pensons organiser pour l'occasion une réunion exceptionnelle, étalée sur un ou deux jours. Nous n'en avons pas encore déterminé la forme, mais cela pourrait être soit un concours de programmes, comme les tournois d'Othello organisés il y a quelques années par *L'Ordinateur Individuel*, soit une conférence internationnale, d'un modèle identique à ce qui a été fait dans le passé par certains clubs étrangers, où différents intervenants présenteraient le fruit de leur travail, ainsi que des présentations de nouveaux matériels. Vous pouvez bien sûr nous faire savoir quelle est la formule qui vous intéresserait le plus, ainsi que par exemple le sujet ou le jeu sur lequel serait basé le concours. Enfin, il est très important que nous sachions très vite quel serait le nombre de personnes intéressées ainsi que la date qui leur serait le plus adéquate (en décembre 92 ou Janvier 93), afin que nous puissions déterminer la date précise et surtout une salle suffisamment grande pour accueillir une assemblée, qui nous l'espérons, sera très nombreuse.

En parlant de fêtes, il faut noter aussi celle qu'organise le club anglais, HPCC, les 19 et 20 Septembre. Nous étudions la possibilité de nous y rendre, afin de vous en faire un compte rendu détaillé dans le journal.

En ce qui concerne ce journal, vous pouvez constater qu'il est encore plus fin que le précédent. Cela est surtout dû à une section HP-48 plutôt succinte, ce qui est étonnant lorsque l'on connait le nombre d'adhérents possédant cette machine. Les deux derniers numéros du journal ont été faits avec les quelques articles que nous avions en avance, nous n'avons absolument plus rien pour le mois prochain. Nous sommes pourtant sûrs que vous faites des choses géniales, dont beaucoup de membres voudraient bien avoir connaissance. Dans le même ordre d'idée, vous avez pû remarquer que les programmes MS-DOS parus dans les derniers numéros avaient été écrits en C. Il est bien évident que nous acceptons tous les articles basés sur d'autres langages (Basic, Pascal, Assembleur...). Nous avons maintenant prouvé que nous pouvions relancer le club, maintenant c'est à vous de nous aider à le faire vivre.

Pour finir, une petite information encore au conditionnel : Hewlett-Packard annoncerait bientôt la sortie du premier disque dur de 1,3 pouces, développé spécialement pour les prochains ordinateurs de poche.

Le Bureau

MAGASINS AGREES PPC

De nombreux adhérents ignorent encore que leur adhésion leur permet d'obtenir des réductions sur du matériel Hewlett-Packard dans deux magasins de la région parisienne. Pour en bénéficier, il vous suffit de vous présenter, avec votre carte de membre, aux adresses suivantes :

Maubert Electronic
49, Bd Saint Germain
75005 PARIS

Compta France
3, route de la Reine
92100 BOULOGNE BILLANCOURT
(Fermé le Samedi)

Pour les non-adhérents, sachez qu'ils distribuent aussi les derniers numéros de *JPC*.

Si des adhérents provinciaux connaissent des magasins suceptibles d'être intéressés de faire la même chose dans leur région, il peuvent nous envoyer leur adresse afin que nous puissions les contacter.

Jacques Belin (123)

COURRIER DU COEUR

DUTOUR Jean-Pierre
Tel. pers. : (1) 47 89 36 78.
Tel. prof. : (1) 30 40 02 00.

Vend :

HP-71B, Module mathématique, RAM 32K, JPC Rom + Manuel, Lecteur de cartes + cartes, Imprimante ThinkJet, Lecteur de cassettes rapides HP. Le tout pour 3000F. Matériel disponible lors de la réunion du 13 Juin.

HERLICQ B.
Tel : (1) 46 40 15 33.
Vend :

HP-48SX 3 mois d'utilisation, Extension 32 Ko RAM, Kit d'interface PC, Le tout pour 2200F.

BARNAUD Frédéric
Tel. pers. : (16) 50 95 36 73 (France)
Tel. prof. : (19 41) 22 710 12 23 (Suisse)

Vend :

HP-75C avec module de connexion HP-IB et cartes magnétiques, HP-71B avec modules Math, Forth/Assembleur et HP-IL, en parfait état avec manuels d'utilisation.

HP-28

CALCUL FORMEL SUR HP-28S ET HP 48SX

- "Voyage au Centre de la HP-28C/S"
  Paul Courbis & Sébastien Lalande.

- "HP-28 Insights. Principles and Programming of the HP-28C/S"
  William C. Wickes.
  Larken Publications

William Wickes, page 125 de son ouvrage, écrit: *La HP-28 est unique parmi les calculateurs, par sa faculté d'appliquer des opérations mathématiques à des quantités 'symboliques' - objets pour lesquels aucune valeur numérique n'a été assignée* et, page 137, dans le chapitre Evaluation des Constantes Symboliques, il écrit: *Le mode d'évaluation symbolique et le mode d'évaluation numérique modifient la manière dont les fonctions intégrées dans la HP-28 évaluent les arguments symboliques.*

Les cinq constantes symboliques `11`, `e`, `i`, `MAXR` et `MINR` *se comportent comme des fonctions à argument zéro - et comme des fonctions sensibles au mode d'évaluation. Lorsque le flag 36 est Set (allumé), l'évaluation de l'une de ces constantes renvoie un résultat symbolique, qui se trouve être la constante elle-même non modifiée. Lorsque le flag 36 est Clear (éteint), l'évaluation de la constante symbolique remplace celle-ci par sa valeur numérique.*

A l'aide du flag 35, il est possible de choisir une forme restreinte du mode d'évaluation numérique qui affecte uniquement ces constantes. Lorsque le mode d'évaluation symbolique est actif (flag 36 Set), le fait d'éteindre le flag 35 (35 CF) aboutit à ce que les constantes symboliques soient évaluées numériquement, sans que cela affecte l'évaluation des autres fonctions. Cela permet, par exemple, de remplacer des constantes symboliques par des valeurs numériques dans des expressions qui contiennent des variables formelles (Undefined Names).

La nouvelle HP-48SX, dans son menu MODES `[ ][MODES]`, comporte en première page une touche `SYM` qui n'existe pas sur la HP-28, et dont la présence facilite grandement les opérations de changement de mode. Elle se présente sous les deux aspects `{SYM}` ou `{SYM▪}` (avec un carré plein dans l'étiquette signifiant que l'opération est active, c'est-à-dire que la machine se trouve en mode d'évaluation symbolique). Ce 'toggle' remplace avantageusement l'appui de trois touches dans le menu `[ ][MODES]` (ou dans le menu PRG TEST) :

```
3 [+/-] {SF}     (ou {CF})
```

sur la HP-28S, dans le menu `PRG TEST` :

```
36 {SF}          (ou {CF}).
```

On peut regretter que la touche `{BEEP}` de la première page du menu MODES `[ ][MODES]` ne se trouve pas à la sixième place à droite (vide actuellement) de la quatrième page du même menu (page accessible aisément à partir de la page 1, par la touche `[PREV]`). Cela aurait libéré un emplacement où pourrait se trouver une touche de changement du mode d'évaluation des constantes symboliques (flag -2), touche que l'on pourrait nommer `K` et qui s'afficherait sous l'aspect de `{K}` ou de `{K▪}`, lorsque le mode d'évaluation symbolique serait actif.

En transposant cette idée sur la HP-28, on peut procéder ainsi :

- En menu HOME, entrer la variable `'u'` qui contiendrait le programme suivant inspiré du `BIP/BIP▪`, page 264 du livre de Courbis et de Lalande :

```
« -> x y z
   « RCLF 31 CF
      IFERR x RCL THEN
         y
      ELSE
         DROP x
      END
      SWAP STOF DUP DUP RCL VARS DUP
      4 ROLL POS 1 - 1 SWAP SUB ROT
      PURGE z DUP
      IF FC? THEN
        SF y
      ELSE
        CF x
      END
      ROT OVER STO ORDER
   »
»
'u' STO
```

- Créer un répertoire: `'MODE' CRDIR` dans lequel se trouverait deux variables `'sym'` et `'k'`, apparaissant au menu sous l'aspect `{SYM}` ou `{SYM▪}`, `{K}` ou `{K▪}` suivant que le mode symbolique est actif ou non :

```
« 'sym' 'sym▪' 36 u » 'sym' [STO]
« 'k' 'k▪' 35 u » 'k' STO
```

On pourrait ajouter le programme de Courbis & Lalande :

```
{ 'beep=' 'beep' 51 u } 'beep' STO
```

Exemple: Après avoir invalidé LASTX, entrer l'exemple suivant pris dans le livre de Wickes, page 137, à la suite du passage cité ci-dessus :

- Choisir le mode d'affichage STD.
- 'X' PURGE
- Entrer l'expression '2*II*X' au niveau 1, la stocker dans 'Q'
- Par appuis successifs des touches, choisir les étiquettes du symbolisme activé, et pour les fonctions, et pour les constantes :
  {SYM=} et {K=}
- Appuyer sur {Q} pour placer l'expression sur le niveau 1,
  puis [][->NUM]. Message d'erreur :
  'Undefined Name'
- Supprimer le message.
- Au lieu de {K=}, faire apparaître {K}.
- Appuyer sur {Q} pour placer l'expression sur le niveau 1,
  puis [EVAL] '6.28318530718*X'
- 'Q' PURGE

II a été ainsi évalué à sa valeur numérique, pendant que, le mode symbolique restant toujours actif, l'opération * renvoyait comme auparavant un produit symbolique.

Note: On peut regretter que, sur la HP-28S, dans la deuxième page du menu MODE, la commande {PRMD} -qui se trouve déjà au menu PRINT- n'ait pas été remplacée par {SYM}.

Philippe HEILBRONN (233)

REVERSE LINE

Ce programme pour HP-28S 2BB permet d'effectuer une inversion vidéo à l'écran de la ligne de son choix.

Listing commenté :

```
Code asm  Mnémoniques       Commentaires

76C20     CON(5) 02C67
A63C0     CON(5) 0C36A      Met à jour le STACKSIZE
58A81     CON(5) 18A85      Fige l'écran.
69C20     CON(5) 02C96      Début objet routine LM
      début :
6E000     CON(5) fin-début  Longueur du programme
                             en quartets.
8F18050   GOSBVL SAV.REG    Sauvegarde registres
                             D0, D1, B et D.
AF2       C=0    W          Efface Registre C.
1BF510C   D0=C015F          Mise à zéro du
                             CMD NUMBER pour ne
144       DAT0=C A          pas faire référence
                             à une instruction
                             particulière lors de
                             l'envoi éventuel d'un
                             message d'erreur.
1B9F00C   D0=C00F9          D0=Adresse de la
                             STACKSIZE.
142       A=DAT0 A          Vérification du nombre
                             d'arguments.
30A       LC(1)  A
8B9       ?C=<A  A
D0        GOYES  s1         (+13d = 13 quartets en
                             décimal)
3410200   LC(5)  00201      Code message
                             "Too Few Arguments".
63A0      GOTO   error      (+163d)
      s1 :
8F8B050   GOSBVL LOAD.REG   Recharge les registres
                             précédement saugardés.
143       A=DAT1 A          A=Adresse de l'objet
                             sur la pile.
130       D0=A              Vérification du type
                             de l'objet.
142       A=DAT0 A          A=Entête de l'objet.
3407A20   LC(5)  02A70      Type #Binaire.
8A2       ?A=C   A
D0        GOYES  s2         (+13d)
3420200   LC(5)  00202      Code message
                             "Bad Argument Type".
6C70      GOTO   error      (+124d)
      s2 :
169       D0=D0+10          Recherche l'info utile.
142       A=DAT0 A          A=N° de la ligne.
3440000   LC(5)  00004      Vérification validité
                             du numéro.
8B2       ?C>A   A
D0        GOYES  s3         (+13d)
3430200   LC(5)  00203      Code message
                             "Bad Argument Value".
6F50      GOTO   error      (+95d)
      s3 :
34A8000   LC(5)  0008A      OK, Initialisations.
D8        B=A    A
C5        B=B+B  A
DA        A=C    A          A=Nombre d'itérations.
      loop :
CC        A=A-1  A          Tête de boucle.
3444000   LC(5)  00044
8BE       ?C=<A  A          Test de la valeur du
                             compteur pour
                             déterminer la valeur
```

```
                                de l'offset a ajouter
                                car la mémoire
                                d'écran est divisée
                                en deux zones.
D0          GOYES  s4           (+13d)
34048FF     LC(5)  FF840        Adresse début de la
                                première partie de
                                l'écran.
6A00        GOTO   s5           (+10d)
      s4 :
348D9FF     LC(5)  FF9D8        Adresse début de la
                                deuxième partie de
                                l'écran.
      s5 :
D7          D=C    A            Calcul de l'adresse de
                                la colonne courante.
D6          C=A    A
C6          C=C+C  A
C6          C=C+C  A
C6          C=C+C  A
C9          C=C+B  A
CB          C=C+D  A            C=Adresse calculée.
134         D0=C
14E         C=DAT0 B            Lecture des deux
                                quartets à inverser.
FE          C=-C-1 A            Inversion des deux
                                quartets.
14C         DAT0=C A            Ecriture des deux
                                quartets modifiés.
8AC         ?A?0   A            A t'on atteint la fin
                                de la boucle ?
4C          GOYES  loop         (-60d)
8F8B050     GOSBVL LOAD.REG     Fin du programme.
142         A=DAT0 A
164         D0=D0+5
808C        PC=(A)
      error :
DA          A=C    A            Sous programme de
8F8B050     GOSBVL LOAD.REG     gestion des erreurs.
8DA6930     GOVLNG ERROR
      fin :
A2670       CON(5) 0762A
09F20       CON(5) 02F90

'→RL' STO
```

Avertissement :

Il convient d'être très prudent lors de la saisie de la chaîne à assembler, car une erreur de frappe risquerait de provoquer un "Memory Lost" de votre machine.

Chaine à assembler :

```
"76C20A63C058A8169C206E0008F18050AF21BF510C1441B9
F00C14230A8BED0341020063A08F8B0501431301423407A20
8A2D034202006C7016914234400008B2D034302006F5034A8
000D8C5DACC34440008BED034048FF6A00348D9FFD7D6C6C6
C6C9CB13414EFE14C8AC4C8F8B050142164808CDA8F8B0508
DA6930A267009F20"
```

Exemples d'utilisation :

```
#0h →RL     Inverse la première ligne écran.
#2h →RL     Inverse la troisième ligne écran.
```

Tous les cas d'erreur sont traités par l'envoi d'un message système standard.

Bonne Programmation à Tous !

Jean-Louis ATTENOUX (83)

ALTER-EGO

Tout d'abord, je tiens à signaler que le programme que je vous propose s'adresse aux possesseurs de HP-28S version 2BB exclusivement. Ce, car il est fait utilisation de routines en ROM dont les adresses diffèrent suivant les versions.

Comme son nom l'indique RENAME sert à redonner un nom, mais redonner un nom à quoi ? Tout simplement à changer le nom de n'importe quelle variable ou programme utilisateur qui se trouve sur le menu USER que vous êtes en train d'utiliser (c'est à dire que si vous voulez changer le nom d'un programme qui est dans un sous-menu, vous devez vous placer dans ce sous-menu, autrement la machine vous dira qu'elle ne trouve pas le programme en question). Pour se faire voici la marche à suivre :

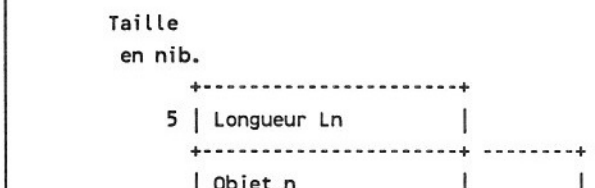
- Placer au niveau 2 l'ancien nom sous sa forme de global name habituel (ex : 'titi').
- Placer au niveau 1 de la pile le nouveau nom de votre variable USER sous forme de string (ex : "toto"), ce qui implique que vous pouvez non seulement changer un nom qui ne vous convient pas, mais en plus le changer pour un global name que la machine n'accepte normalement pas (ex : 'to=to').

ATTENTION : le nouveau nom doit être de la même longueur que l'ancien !!!

- Executer le programme RENAME.

Il est à noter que le programme prend en compte toutes les erreurs que vous pourriez faire et qui en empêcheraient le bon déroulement (ex: pas d'objet dans la pile, le nom n'existe pas, etc.). Cela grâce à l'utilisation de routines situées en ROM, leur utilisation n'étant pas explicitée ici puisque ce n'est pas le sujet du présent article. Le programme RENAME ne commence donc qu'à partir des lignes : 69C20 CC000 8F18050..., les lignes les précédant consistent en une routine qui vérifie si le nombre d'objets dans la pile est suffisant (deux en l'occurence) et en une seconde un peu plus complexe à utiliser qui vérifie si leur nature est correcte (niveau 2 un string, niveau 1 un global name).

Il ne vous reste plus qu'à introduire RENAME dans votre HP et vous voici pourvu d'une nouvelle fonction.

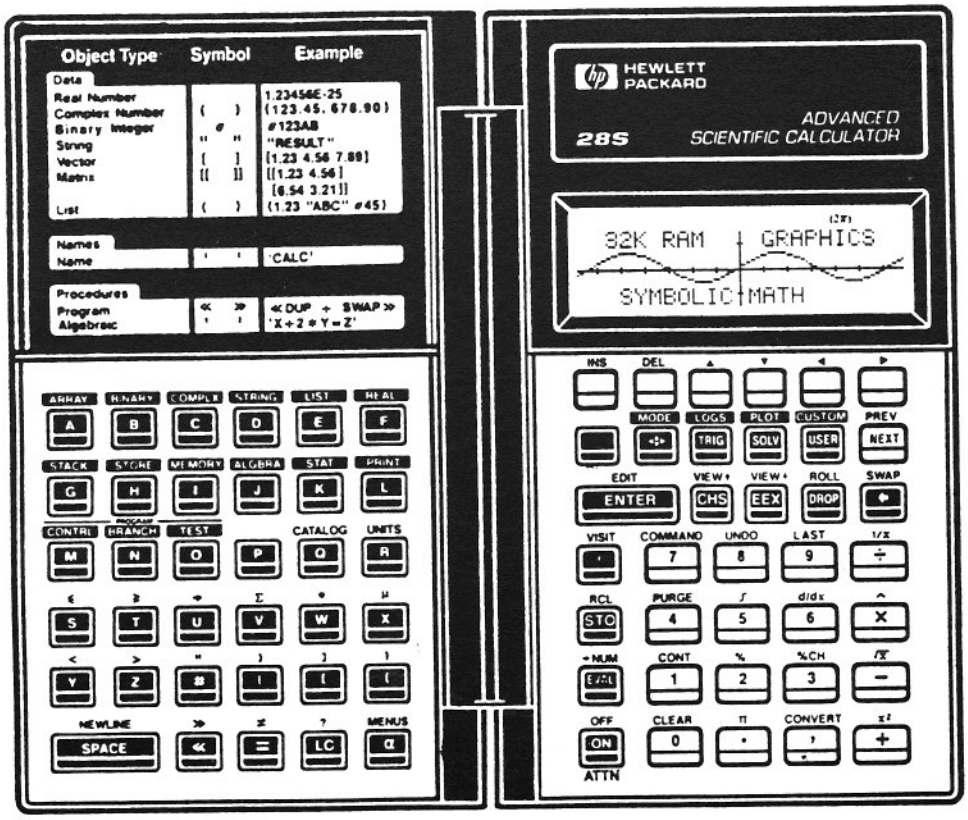


Structure d'un objet USER en RAM.

```
76C20     PROLOGUE PROGRAMME
69C20     PROLOGUE PGM ASSEMBLEUR
E1000     LONGUEUR PGM ASSEMBLEUR
132       ADOEX        Vérification du nombre
1BF510C   DO=C015F     d'objets dans la pile
D2        C=0      A
144       DAT0=C   A
130       D0=A
8D6D3C0   GOTLNG 0C3D6
76C20     PROLOGUE PROGRAMME
5A7C0                  Vérifie si la nature du 1er
                       objet dans la pile correspond
7C170                  à celle demandée.
5E020     SWAP
09F20     FIN DE STRUCTURE PROGRAMME
76C20     PROLOGUE PROGRAMME
5A7C0                  Verifie si la nature du 2eme
                       objet dans la pile correspond
5E170                  à celle demandée.
5E020     SWAP
09F20     FIN DE STRUCTURE PROGRAMME
69C20     PROLOGUE PGM ASSEMBLEUR
                       Debut de RENAME
CC000     LONGUEUR DU PGM
8F18050   GOSLNG SAVE-REG
174       D1=D1+5      objet 2 dans la pile
143       A=DAT1   A   adr du global name
132       ADOEX        D0=A
164       D0=D0+5      adr longueur du global name
AA2       C=0      XS
14E       C=DAT0   B   C=long. du global name
1C4       D1=D1-5      objet 1 dans la pile
143       A=DAT1   A   adr du string
132       ADOex        D0=A
100       R0=A         R0= adr long du global name
164       D0=D0+5      adr long du string
14A       A=DAT0   B   A= long du string + 5
132       ADOex
184       D0=D0-5
132       ADOex        A= long du string
A36       C=C+C    X   C= long global name en quartet
962       ?A=C     B   est ce la meme longueur ?
91        GOYES SUITE
8F8B050   GOSLNG LOAD-REG
3410500   LCHEX 00501  "Invalid Dimension"
DA        A=C      A
8DA6930   GOVLNG ERROR
    SUITE:
1B2800C   D0=C0082     adr menu en cours
142       A=DAT0   A
101       R1=A
8F19B50   GOSLNG 05B91  routine qui cherche l'adr du
591       GONC FIND     gl.name et la met dans D0,
8F8B050   GOSLNG LOAD-REG  sinon set carry
3440200   LCHEX  00204  erreur car le global name
DA        A=C      A    n'est pas dans le menu
8DA6930   GOVLNG ERROR  en cours.
    FIND:
8F54260   GOSLNG  06245 met  dans D0 adr du 1er
D2        C=0      A    longueur objet
14E       C=DAT0   B
161       D0=D0+2       adr du global name en RAM
1F4600C   D1=C0064
143       A=DAT1   A
131       D1=A          récupere le nouveau nom a
143       A=DAT1   A    transferer.
131       D1=A
179       D1=D1+10
    TRANS:
14B       A=DAT1   B    transfère le nouveau nom
148       DAT0=A   B    en RAM
161       D0=D0+2
171       D1=D1-2
CE        C=C-1    A
8AE       ?C=0     A
FE        GO TRANS
8F8B050   GOSLNG LOAD-REG
8DF1120   DROP2         Fin du programme
09F20     FIN DE STRUCTURE PROGRAMME

76C20 69C20 E1000 1321B F510C D2144 1308D 6D3C0
76C20 5A7C0 7C170 5E020 09F20 76C20 5A7C0 5E170
5E020 09F20 69C20 CC000 8F180 50174 14313 2164A
```

```
A214E 1C414 31321 00164 14A13 21841 32A36 96291
8F8B0 50341 0500D A8DA6 9301B 2800C 14210 18F19
B5059 18F8B 05034 40200 DA8DA 69308 F5426 0D214
E1611 F4600 C1431 31143 13117 914B1 48161 171CE
8AEFE 8F8B0 508DF 11200 9F20
```

Ludovic VALOIS



HP-48

MATRICES TRIDIAGONALES

Il est fréquent d'inverser en analyse numérique des matrices tridiagonales . Celles-ci sont des matrices creuses telles que tous leurs coefficients soient nuls sauf ceux de la diagonale principale et des deux diagonales contigues.

Le programme proposé résoud un système linéaire en supposant que la matrice est tridiagonale à diagonale dominante en appliquant la méthode de Cholesky. Il nécessite seulement la connaissance des trois diagonales et évite d'entrer une matrice dont beaucoup d'éléments sont nuls.

Le programme

```
« { n } 0 CON DUP → gam u
  « 'b(1)' →NUM → bet
    « 'r(1)/bet' →NUM 'u(1)' STO 2 n
       FOR j 'c(j-1)/bet' →NUM 'gam(j)'
         STO 'b(j)-a(j)*gam(j)' →NUM 'bet' STO
         '(r(j)-a(j)*u(j-1))/bet' →NUM 'u(j)'
         STO
       NEXT n 1 - 1
       FOR j 'u(j)-gam(j+1)*u(j+1)'
         →NUM 'u(j)' STO -1
       STEP u
    »
  »
»
'TRI' STO
```

Mode d'emploi

1) Entrer la diagonale inférieure.

Par exemple :

```
[ 0 -6 -2 8 ] 'a' STO
```

2) Entrer la diagonale principale.

Par exemple :

```
[ 1 4 1 3 ] 'b' STO
```

3) Entrer la diagonale supérieure.

Par exemple :

```
[ 2 3 9 0 ] 'c' STO
```

4) Entrer le vecteur.

Par exemple :

```
[ -2 53 8 27 ] 'v' STO
```

5) Entrer la dimension de la matrice.

Par exemple :

```
4 'n' STO
```

6 Lancer le calcul:

```
TRI
```

6 Lire le résultat dans la pile:

```
[ -6 2 3 1 ]
```

La matrice tridiagonale M considérée est:

```
[[  1  2  0  0 ]
 [ -6  4  3  0 ]
 [  0 -2  1  9 ]
 [  0  0  8  3 ]]
```

D'où la construction des vecteurs a, b et c . En vous souhaitant une bonne résolution de systèmes tridiagonaux.

Heureuse Programmation à Tous !

Asdin AOUFI (562)

MATRICES PENTADIAGONALES

En analyse numérique il n'est pas rare que l'on ait à manipuler des matrices pentadiagonales - par exemple discrétisation du Laplacien en coordonnées cylindriques -. Celles-ci sont des matrices creuses telles que tous les coefficients d'une ligne donnée soient nuls sauf cinq d'entre eux. En outre elles font apparaitre cinq diagonales et sont de dimension n^2 où n est la dimension de la matrice élémentaire.

Le programme que je vous propose permet de construire une matrice pentadiagonale. Il est rédigé en suivant la définition mathématique de ces matrices sans rechercher une optimisation particulière aussi bien au niveau de la rapidité que de l'utilisation de la pile. Nul doute que vous pourrez facilement l'améliorer.

Le programme

```
« → a b c d e n
  « n SQ IDN c * → A
    « 1 n SQ
      FOR i
        IF n i <
          THEN a 'A(i,i-n)' STO
        END
        IF 2 i ≤
          THEN b 'A(i,i-1)' STO
        END
        IF i n SQ <
          THEN d 'A(i,i+1)' STO
        END
        IF i n SQ n - ≤
          THEN e 'A(i,i+n)' STO
        END
      NEXT A
    »
  »
»
'PENTA' STO
```

Mode d'emploi

1) Entrer les cinq coefficients souhaités:

Par exemple :

```
1 2 3 4 5
```

2) Entrer la dimension de la matrice désirée:

Par exemple :

```
3
```

3) Lancer le calcul:

```
PENTA
```

4) Lire le résultat dans la pile:

```
[[ 3 4 0 5 0 0 0 0 0 ]
 [ 2 3 4 0 5 0 0 0 0 ]
 [ 0 2 3 4 0 5 0 0 0 ]
 [ 1 0 2 3 4 0 5 0 0 ]
 [ 0 1 0 2 3 4 0 5 0 ]
 [ 0 0 1 0 2 3 4 0 5 ]
 [ 0 0 0 1 0 2 3 4 0 ]
 [ 0 0 0 0 1 0 2 3 4 ]
 [ 0 0 0 0 0 1 0 2 3 ]]
```

Comme vous pouvez le constater la taille de ces matrices croit très vite, il peut être judicieux de désactiver les possibilités de sauvegarde du menu MODES.

Cet outil peut facilement s'intégrer dans des applications plus complexes.

Bonne Programmation à Tous !

Asdin AOUFI (562)

LES CARTES DU HP-48SX

Les cartes d'application permettent de doper la HP-48SX en la transformant en un petit système expert sur un domaine pointu comme jadis la mythique HP-41 avec ses modules .

Commençons ce tour d'horizon thématique par:

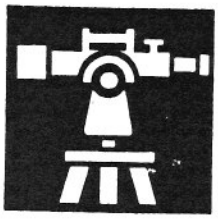
Math : E.Z. Math, Algebra, Arithmetic d' *E.Z. Software* $110, Ces trois cartes de niveau décroissant du lycée au collège permettent des calculs élémentaires. La première offre une bibliothèque de fonctions et une analyse graphique, les suivantes un cours d'algèbre et d'arithmétique. **Math. App. Pac** de *Sparcom* 90$, Cette carte est d'un niveau plus élévé avec les résolutions d'équations algébriques (degré 3 et 4) , la trigonomètrie hyperbolique. Une table des dérivées et intégrales ainsi que les transformées de Fourier, Laplace et Z complètent le tout.

Chimie : Gen. Chem. App. Pac de *Sparcom* 90$, Contient 150 équations : cinétique, acide/Base etc. Courbes de titrage acido-basique et un tableau périodique très complet. Possibilité d'écrire des équations de réactions et de les équilibrer automatiquement. **Chem. Ref. Lib.** de *Sparcom* 90$, Contient 3000 entrées sur 11 domaines tels que: éléments, solides, thermodynamique, complexes, électrochimie, ...

Physique : Phys. Pac de *Sparcom* 90$, Possède 250 équations des divers domaines de la physique ainsi qu'un formulaire des fonctions de la physique mathématique telles que Bessel, Erreur. Elle permet une analyse des champs vectoriels et scalaires et dispose de tables d'intégrales et la résolution d'équations polynomiales.

Calcul de structures : Struct. Analys. Card de *TDS* 110$, A l'aide d'un menu très performant permet tous les calculs de structures, poutres.. pour l'étudiant ou l'ingénieur.

Topographie : Surv. Soft. Rom Card de *SMI* 92$, Permet tous les calculs de base en topographie et de stocker jusqu'à 12000 points. **Survey Card** de *TDS* 430$, Distribué en France par Le Pont Equipement à 4750FF HT. Le must permet tous les calculs du géomètre , sauvegarder jusqu'à 12000 points et d'être connecté à la plupart des tachéomètres du marché tels que ZEISS, NIKON, PENTAX, ...



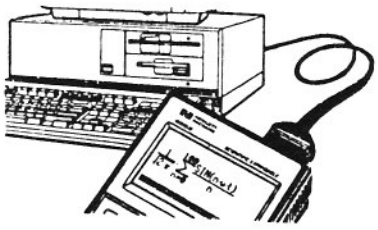
Asdin AOUFI(562)

REGLES D'ENVOI
DES PROGRAMMES HP-48

Pour la rédaction de vos articles suivez les conseils de la page *ah! vous écrivez.*

Nous souhaiterions recevoir les programmes transférés sur disquettes MS-DOS dans les deux modes :
- binary
- ASCII avec translate code : 3

Les fichiers MS-DOS contenant les programmes en binaires auront l'extension .BIN et ceux contenant les programmes en ASCII auront l'extension .ASC.



Les programmes composés de plusieurs variables devront être dans un répertoire à transférer sur disquette et dans les deux modes.

Si vous insérez dans le corps de l'article le texte des programmes veuillez faire précéder ce texte de ‹rpl et le faire suivre de ‹endrpl.

Cette façon de procéder nous évitera beaucoup de travail car il est possible de traiter automatiquement la forme ASCII fournie par une HP-48 afin de coder les caractères spéciaux et mathématiques avant transfert dans l'IPC.

Guy TOUBLANC (276)

HP-95

LES CARTES DU HP-95

Comme ce qui a été fait dans les pages précédentes pour le HP-48, cet article a pour but de vous présenter, de façon succinte, les diférentes cartes disponibles sur le HP-95. Dès que nous aurons la possibilité de les essayer, nous vous présenterons ces cartes de façon plus complète dans les prochains numéros.

DERIVE de *Soft Warehouse* :

Cette carte permet de résoudre toutes sortes de calculs mathématiques avec une précision de plusieurs milliers de décimales, tous calculs sur les complexes, conversions d'unités, simplifications de formules, developpements et factorisations de polynômes, mise en dénominateur commun d'expressions, résolutions algébrique d'equations ou d'inéquations, résolution de systèmes linéaires d'équations ou d'inéquations... Tout ceci peut être affiché sur l'écran en 2D ou en 3D. Prix : environ 2000F HT.

NOGETEL EMULATEUR de *NOGEMA informatique* :

Cette carte contient un logiciel émulateur permettant de vous connecter sur le réseau vidéotexte, pour visualiser, enregistrer et imprimer des pages Minitel. Le HP-95 doit être connecté, par la liaison série, soit à un modem V23, soit à un Minitel par un cable péri-informatique. Le prix de la carte avec ce type de cable est d'environ 1250F HT, 1750F HT avec un modem V23.

GTS de *Globalink* :

Ces cartes contiennent des logiciels de traduction de fichiers textes. Différents modèles existent : Anglais-Français, Français-Anglais, Anglais-Allemand, Anglais-Espagnol... Prix aux environs de 2300F HT.

Medical Software de *ComputerBooks* :

Ensemble de quatre cartes très utiles pour les milieux médicaux. La première calcule tous les dosages et donne les éventuelles contre-indications de plus de 1700 médicaments (américains). Prix : 369$. La seconde carte inclue 26 sous-programmes d'analyses médicales. Prix : 279$. La toisième carte contient un programme de gestion de fiches concernant les patients, avec leur historique médical, résultats d'analyses... Prix : 374$. La dernière carte contient un logiciel donnant toutes les interactions entre différents médicaments ou composés. Prix : 374$.

Dictionary/Thésaurus de *Hewlett-Packard* :

Cette carte contient un dictionnaire de mots et de synonymes (en Anglais), accessible par l'appui d'une simple touche à partir du MEMO du HP-95. Prix : 130$.

Jacques Belin (123)

VOYAGE AU CENTRE DE LA MEMOIRE

Présentation

Il existe, le plus souvent dans le domaine public, de nombreux utilitaires permettant de visualiser le contenu de la mémoire du PC. Celui que je vous propose aujourd'hui n'offre rien de révolutionnaire, si ce n'est qu'il a été conçu pour l'affichage réduit du HP-95.

Vous pourrez donc examiner n'importe quelle zone des programmes, des données, du BIOS ou de la mémoire vidéo, en fait tout ce qui se trouve dans la zone d'un Méga-octet accessible par le 8086. Cette visualisation est dynamique, c'est à dire que pour les zones modifiées en permanence, par exemple le compteur du timer, l'affichage se fait en temps réél, sans que l'on aie à appuyer sur une touche.

Il est bien évidement possible de se positionner sur n'importe quelle adresse, que ce soit par diverses incrémentations de l'adresse courante, ou par entrée directe de cette adresse. Il est également possible de rechercher une chaine de caractères dans l'ensemble de la mémoire.

Mode d'emploi

La syntaxe est la suivante :

```
VMEM95 [segment][:offset]
```

où `segment` et `offset` sont des options permettant de positionner l'affichage à une adresse donnée au lancement du programme. Elles sont comprises entre 0000 et FFFF.

Une fois l'affichage présent, on peut naviguer dans la mémoire à l'aide des touches UP et DOWN pour déplacer l'affichage d'une ligne vers le bas ou vers le haut. Les touches PgUp et PgDn permettent de passer à la page précedente ou la page suivante. Enfin, la touche HOME ramène l'affichage au début de la mémoire, alors que END le place en fin de la mémoire.

Il est possible d'incrémenter l'adresse courante, quartet par quartet, à l'aide des huit premières touches de fonctions. Les quatre premières sont réservées au segment, les suivantes à l'offset. Un appui simultané de ces touches avec Shift permet de décrémenter l'adresse de la valeur correspondante.

La touche F9 permet de sélectionner une adresse, sous la forme *segment:offset*. Il est possible de n'entrer qu'une seule de ces valeurs, dans ce cas l'autre valeur n'est pas modifiée. Si seulement l'offset est spécifié, il doit être précédé du caractère ':'.

La touche F10 permet d'entrer une chaîne de caractères à rechercher dans la mémoire. Cette recherche se fait indépendament des majuscules ou de minuscules. Ensuite, un appui sur Shift+F10 recherchera les occurences suivantes de la chaine. Il est à noter que la chaîne sera toujours trouvée au moins une fois, qui correspond à l'emplacement de la chaîne de référence stockée dans une variable du programme.

La touche ESC permet de sortir du programme.

Notes sur le développement

Le programme a été développé en TURBO C, sans rechercher à conserver un quelconque compatibilité avec un autre compilateur, Microsoft ou autre.

Afin simuler un adressage linéaire et contourner les problèmes dûs à la segmentation du microprocesseur Intel, un type de variable FULLPTR a été créé, décrivant une adresse avec ses deux constituants, segment et offset. Ceci permet de manipuler cette adresse de façon plus simple, notamment lors du passage des paramêtres vers les fonctions. Pour additionner ou soustraire une valeur à une adresse sans avoir à se préoccuper d'un dépassement d'offset (qui ne met pas automatiquement à jour la valeur du segment), deux fonctions ont été créées : incptr et decptr. Leur fonctionnement est très simple : si le résultat d'un calcul sur l'offset dépasse FFFFh pour une addition ou 0000h pour une soustraction, la valeur du segment est ajustée de 1000h, en plus ou en moins suivant l'opération est effectuée. La même méthode est utilisée dans la routine d'affichage et dans la recherche de chaîne particulièrement, car il est tout à fait possible que la chaine recherchée soit placée sur deux segments consécutifs, au vu du référentiel segment:offset déterminé par le programme.

De plus, afin d'accroitre la vitesse (d'un rapport de 1 à 20 environ), les routines d'affichage et de recherches de chaines on été partiellement écrites en assembleur. Ayant encore peu d'expérience en assembleur 80X86, je pense qu'il est possible de gagner encore un peu de vitesse en optimisant certains transferts vers la mémoire. D'autre part, pour des raisons de lisibilité, je n'ai pas converti en assembleur certaines routines, comme incptr ou decptr. Cela aurait donné un code beaucoup plus efficace, mais ces fonctions n'étant utilisées qu'une fois après chaque pression de touche, la vitesse d'exécution importe peu.

Quelques adresses intéressantes.

Pour finir, je vous offre un petit tour dans quelques zones de la mémoire particulièrement interressantes :

Zones communes à tous les IBM-PC :

0040:001E : Buffer du clavier. D'une longueur de 32 octets, cette zone contient les Scan Codes des 16 dernières touches tapées au clavier. Vous pouvez remarquer que toutes les touches, y compris celles spécifiques au HP-95 sont prises en compte dans ce buffer. Pour plus de détails se reporter à l'article consacré à ce sujet dans ce journal.

0040:006C : Compteur de l'heure. D'une longueur de 4 octets, cette zone incrémentée 18.2 fois par secondes par l'interruption 1A, contient l'heure courante depuis minuit. En 0040:0070, un octet passe à 1 pour indiquer éventuellement un changement de jour. L'affichage dynamique du programme permet de visualiser son évolution. De même, toute zone étant mise à jour par une interruption (clavier, liaison série) peut être surveillée.

B000:0000 ou B800:0000 suivant le type de carte vidéo installée : Mémoire écran en mode texte. D'une longueur de 4000 octets, cette zone contient les caractères affichés, sous la forme du code ASCII du caractère suivi de son attribut de couleur.Pour le HP-95 cette zone commence en B000:0000.

F000:0000 : cette zone contient le BIOS, d'une taille de 64k. Si vous vous placez à la fin de cette zone, qui correspond aussi à la fin de la mémoire accessible par le 8086 (en appuyant sur la touche END), vous pouvez voir en F000:FFF5 une date codée sous forme ASCII qui correspond à la date de fabrication du BIOS. Sur la machine que j'ai testée, c'est le 2 Avril 1991.

Zones spécifiques au HP-95 :

1000:0000 : D'une longueur de 4000 octets, cette zone est une copie exacte de la mémoire vidéo, adressée en B000:0000. Je n'ai pas fait de tests précis à ce sujet, mais il pourrait s'agir de l'adresse réelle de la mémoire écran, placée ainsi pour être à l'intérieur de la RAA de 512 Ko. Un système de pagination permettrait l'adressage vers le segment B000, afin de conserver la compatibilité avec tous les programmes MS-DOS adressant directement la mémoire vidéo.

4080:0000 pour une machine configurée avec la taille de la mémoire principale par défaut (258ko, sur un modèle à 512 Ko) : Début du disque C, se terminant quelque soit le paramêtrage en 7000:FFFF. Vous pouvez donc y trouver les classiques FAT, entrées du directory (Racine en 40E0:0000 et contenus des fichiers.

A000:0000 : Zones de mémoire paginée permettant d'accéder à la totalité de la ROM du HP-95 (Lotus, Editeur, Calculatrice, certaines fonctions DOS...). Cette méthode, identique à ce qui se fait pour l'EMS sur les PC, est la seule méthode permettant d'accéder à une ROM faisant à elle seule la capacité d'adressage du 8086. La selection de ces pages se fait à l'aide d'une interruption. Nous pouvons donc voir dans ces "fenêtres" des extraits des applications livrées avec la machine. Si vous lancez VMEM95 après avoir exécuté différentes application Rom, vous pourrez constater que le contenu de ces pages est modifié. D'autres pages sont disponibles après la mémoire vidéo, à partir de C000:0000. Malheureusement, je n'ai pas eu le temps de déterminer quelle était l'interruption permettant la sélection de cas pages (peut-être tout simplement celle gérant l'EMS sur IBM-PC ?), et donc la structure interne de cette ROM.

Enfin, vous devez savoir que le nom de code du HP-95 pendant son développement était "JAGUAR". Or, si l'on recherche cette chaine de caractères dans la mémoire, on a la relative surprise (car ce n'est pas une habitude chez HP) de retrouver ce code à différents endroits de la mémoire, par exemple pour les identificateurs de certains drivers.

Ce ne sont que quelques exemples de ce que nous pouvons examiner avec le programme. Il y a beaucoup d'autres choses à voir dans le HP-95, mais je vous laisse le soin d'en faire vous même l'exploration. Si vous possédez aussi un IBM-PC, je vous conseille de faire la même chose sur les deux machines, afin de comparer ensuite vos observations.

Pour finir, je voudrais indiquer que ce programme, les observations en résultant, et en général tous les articles consacrés à cette machine dans ces deux derniers numéro de JPC, n'ont pu se faire que grâce à Eric Gengoux, qui m'a aimablement prêté sa machine pendant près d'un mois. Qu'il en soit remercié.

Jacques Belin (123)

AYEZ UNE TOUCHE AVEC LE CLAVIER DU HP-95

Comme sur tout compatible IBM, un programme tournant sur un HP-95 doit, le cas échéant, lire l'état du clavier, et savoir quelles touches sont appuyées.

Les fonctions de bas niveau d'accès au clavier sont fournies par le BIOS à l'aide de l'interruption 16. Pour lire un caractère, on utilisera la fonction 0.

En retour, l'ordinateur doit pouvoir transmettre au programme une information standard, indépendante du type du clavier (84, 102 touches ou claviers des portables). La fonction 0 de l'interruption 16 renvoie un code standard, sur deux octets, sous la forme ccss, qui admet les deux cas suivants :
- Si ss est égal à zéro, cc contient le code ASCII de la touche correspondante.
- Si ss est différent de zéro, ss contient le *Scan Code* d'une touche spéciale, constant quelque soit le clavier utilisé. cc contient alors le code géographique de la touche.

Sur le HP-95, on peut s'attendre à ce que le fonctionnement soit le même que sur un IBM classique, ce qui est confirmé par l'exécution de n'importe quel programme MS-DOS. Mais qu'en est-il des touches spécifiques de cette machine (123, FILLER...) ?

Un test rapide de la fonction BIOS standard montre qu'un appui sur ces touches n'est pas détecté.

Pour résoudre ce problème, il est nécessaire de comprendre un peu mieux ce qui se passe, à un niveau plus bas que l'interruption 16 :

Lorsque l'on presse une touche, le circuit intégré présent dans le clavier provoque une interruption matérielle, qui détourne l'ordinateur de son occupation actuelle. Le BIOS prend alors le relais et stocke le code de la touche (sous la forme ccss dans un buffer, avant de retourner à son travail précédent.

L'interruption 16 se contentera, plus tard et à la demande immédiate du programme, de lire le contenu de ce buffer. Pouvant contenir le code de 16 touches en attente il est présent à une adresse fixe sur l'IBM PC, entre les adresses 0040:001E et 0040:003D. Ce buffer est circulaire, c'est à dire que quand une touche à été placée dans la dernière case du buffer, la touche suivante sera placée dans la première. Afin de s'y retrouver, le BIOS s'aide de deux pointeurs, placés aux addresses 0040:001A et 0040:001C. Le premier pointeur contient l'offset (par rapport au segment 0040) de la case du buffer contenant le code de la prochaine touche à lire par l'interruption 16. Le deuxième pointeur contient l'adresse de la dernière touche ayant été placée dans le buffer. Si les deux pointeurs sont égaux, il n'y a aucune touche en attente.

Or, si on utilise certains utilitaires permettant de visualiser le contenu de la mémoire, on peut s'apercevoir que le buffer du clavier est mis à jour de façon normale.

Si le code d'une touche spéciale est stocké, mais que l'interruption 16 ne le renvoie pas, c'est que le BIOS, jouant le rôle de filtre dans ce cas précis, ne renvoie pas ce code, et fait comme si il n'y avait aucune touche dans le buffer, en mettant en prime les pointeurs à jour.

Il est donc tentant de créer une routine attendant une modification de ce buffer, puis renvoyant le code de la première touche en attente. Cela fonctionne parfaitement, à une notable exception près.

En effet, le HP-95 a la possibilité de s'éteindre automatiquement au bout d'un certain temps d'inactivité, lorsqu'il est en attente d'une pression de touche précisement. Or, cette possibilité n'est active que lorsque le programme passe par les fonctions du BIOS dédiées au clavier. Une routine incluant une boucle qui ne contient que le test du buffer ne s'arrêtera que si on appuie sur un touche et la machine ne s'arrêtera pas automatiquement.

Une recherche empirique sur l'interruption 16 m'a permis de trouver une fonction résolvant ce problème. Le fonction numéro 17, ne fait (apparemment) rien et retourne immédiatement au programme, mais assure l'extinction de la machine en cas de non-utilisation de celle-ci,

Voici un listing de cette fonction, écrit en langage C, sur un compilateur Borland :

```
unsigned int GetKey(void)
  {
   int far *next_key = MK_FP(0x40, 0x1a);
   int far *last_key = MK_FP(0x40, 0x1c);
   int far *key_ptr;
   unsigned keycode;
   union REGS Register;

   while(*next_key == *last_key)
     {
      bioskey(17);      /* fonction spéciale HP-95 */
     }

   key_ptr=MK_FP(0x40, FP_OFF(*next_key));
   keycode = *key_ptr;

   if(bioskey(1))
      keycode=bioskey(0);

   return( (int)((keycode & 0xff) ?
          keycode & 0xff : (keycode>>8) | 256));
  }
```

Cette fonction retourne un entier. Si cette valeur est inférieure à 256, c'est le code ASCII d'un caractère normal ou accentué. Si cette valeur est supérieure à 256, il s'agit d'une touche spéciale (touche de fonction, flèches...).

Il est à noter que j'ai utilisé des moyens relativement empiriques, et à la limite de la compatibilité pour développer cette routine. Il est cependant fort possible que Hewlett Packard ait prévu une autre fonctions permettant d'obtenir un résultat identique, mais je ne l'ai pas trouvée.

Un petit programme utilisant cette fonction peut donc afficher les diférents codes du clavier. En voici la synthèse. Les touches sont classées en suivant l'ordre du clavier, ligne par ligne. Les codes suivis d'un astérisque (*) ne sont pas disponibles sur IBM PC. Les codes `Sft`, `Alt` ou `Ctl` indiquent que les action de ces touches sont les mêmes que pour les touches Shift, Alt ou Control.

Touche	Shft	Alt	Ctl	CHAR	S+A	S+C	A+C	S+CH
ESC	27	(1)	/	27	27*	Sft	Sft	Sft	Sft*
->|	9	271	421*	404*	9*	Alt	Ctl	Alt	Sft*
F1	315	340	360	350	475*	Alt	Ctl	Alt	500*
F2	316	341	361	351	476*	Alt	Ctl	Alt	501*
F3	317	342	362	352	477*	Alt	Ctl	Alt	502*
F4	318	343	363	353	478*	Alt	Ctl	Alt	503*
F5	319	344	364	354	479*	Alt	Ctl	Alt	504*
F6	320	345	365	355	480*	Alt	Ctl	Alt	505*
F7	321	346	366	356	481*	Alt	Ctl	Alt	506*
F8	322	347	367	357	482*	Alt	Ctl	Alt	507*
F9	323	348	368	358	483*	Alt	Ctl	Alt	508*
F10	324	349	369	359	484*	Alt	Ctl	Alt	509*

Touche	Shift	Alt	Ctl	CHAR	S+A	S+C	A+C	S+CH
UP	328	329*	(2)	397*	K	(2)	388	/	UP
ON	(3)	(3)	(3)	418*	/	(3)	(4)	(3)	(3)
FILER	424*	420*	427*	430*	239	423*	/	Alt	/
DATAC	428*	'`'	431*	434*	249	/	/	Alt	/
APPT	432*	'~'	435*	438*	/	/	/	Alt	/
PHONE	436*	'!'	439*	442*	/	376*	/	Alt	/
MEMO	440*	'#'	443*	446*	/	378*	/	Alt	/
LOTUS	444*	'$'	447*	450*	/	379*	/	Alt	/
CALC	448*	'&'	451*	454*	/	382*	/	Alt	/
(	'('	'|'	384*	/	'|'	/	28	Alt	Chr
)	')'	'\'	385*	/	')'	/	28	Alt	Chr
<-	8	K	270	(5)	K	Alt	(5)	/	K
DEL	339	338	/	403*	K	/	402*	(6)	Sft
LEFT	331	327	(2)	371	K	(2)	375	/	Sft
DOWN	336	337	(2)	401*	K	(2)	374	/	Sft
RIGHT	333	335	(2)	372	K	(2)	373	/	Sft
Q	'q'	'Q'	272	17	/	Alt	Ctl	Alt	Chr
W	'w'	'W'	273	23	/	Alt	Ctl	Alt	Chr
E	'e'	'E'	274	5	(7)	Alt	Ctl	Alt	(7)
R	'r'	'R'	275	18	(7)	Alt	Ctl	Alt	(7)
T	't'	'T'	276	20	(7)	Alt	Ctl	Alt	(7)
Y	'y'	'Y'	277	25	(7)	Alt	Ctl	Alt	(7)
U	'u'	'U'	278	21	(7)	Alt	Ctl	Alt	(7)
I	'i'	'I'	279	9	(7)	Alt	Ctl	Alt	(7)
O	'o'	'O'	280	15	/	Alt	Ctl	Alt	/
P	'p'	'P'	281	16	/	Alt	Ctl	Alt	/
7	'7'	'['	/	/	/	/	ESC	/	'['
8	'8'	']'	<-	/	/	/	ESC	<-	']'
9	'7'	'{'	->|	/	/	/	ESC	->|	'{'
/	'/'	'}'	/	/	/	/	29	/	'}'
A	'a'	'A'	286	1	/	Alt	Ctl	Alt	/
S	's'	'S'	287	19	/	Alt	Ctl	Alt	Chr
D	'd'	'D'	288	4	/	Alt	Ctl	Alt	/
F	'f'	'F'	289	6	/	Alt	Ctl	Alt	Chr
G	'g'	'G'	290	7	/	Alt	Ctl	Alt	Chr
H	'h'	'H'	291	8	/	Alt	Ctl	Alt	Chr
J	'j'	'J'	292	10	/	Alt	Ctl	Alt	Chr
K	'k'	'K'	293	11	/	Alt	Ctl	Alt	Chr
L	'l'	'L'	294	12	/	Alt	Ctl	Alt	Chr
4	'4'	';'	/	/	'4'	/	/	/	';'
5	'5'	':'	/	/	'5'	/	/	<-	':'
6	'6'	'''	/	30	'6'	/	/	->|	'''
*	'*'	/	311	406*	/	/	/	Alt	'"'
CTRL	/	/	/	/	/	/	/	/	/
Z	'z'	'Z'	300	26	/	Alt	Ctl	Alt	Chr
X	'x'	'X'	301	24	/	Alt	Ctl	Alt	Chr
C	'c'	'C'	302	3	/	Alt	Ctl	Alt	Chr
V	'v'	'V'	303	22	/	Alt	Ctl	Alt	Chr
B	'b'	'B'	304	2	/	Alt	Ctl	Alt	Chr
N	'n'	'N'	305	14	/	Alt	Ctl	Alt	Chr
M	'm'	'M'	306	13	/	Alt	Ctl	Alt	Chr
ENTER	13	K	294	10	/	Alt	Ctl	Alt	K
1	'1'	'<'	/	/	/	307	/	/	/
2	'2'	'>'	/	259	/	308	/	<-	/
3	'3'	'?'	/	/	/	/	/	->|	/
-	'-'	'^'	/	398*	'-'	381	30	Alt	'^'

Touche	Shift	Alt	Ctl	CHAR	S+A	S+C	A+C	S+CH
SHIFT	/	/	/	/	/	/	/	/	/
ALT	/	/	/	/	/	/	/	/	/
CHAR	/	/	/	/	/	/	/	/	/
Space	' '	K	K	K	K	K	K	K	K
,	','	K	307	/	/	Alt	/	Alt	/
@	'@'	K	377	259	K	Alt	Ctl	Alt	K
MENU	456*	457*	459*	458*	/	Alt	Ctl	Alt	Sft
SHIFT	/	/	/	/	/	/	/	/	/
0	'0'	/	/	/	K	/	/	/	/
.	'.'	/	308	/	K	/	(8)	Alt	/
=	'='	'_'	387	/	K	386	31	Alt	Sft
+	'+'	'%'	/	400*	/	380	/	/	Sft

Notes :

(1) : Impression d'écran.

(2) : Pas de Code, mais déplacement de la fenêtre de l'écran.

(3) : Extinction de la machine, mais code A000h

(4) : Cold Start

(5) : Ctrl-Break

(6) : Warm Start

(7) : Attente de caractère pour accentuation.

(8) : Ctrl+C (arrêt programme)

K : Identique à une pression simple de la touche.

Vous pouvez noter que toutes les combinaisons de touches ne sont pas possibles, alors que des combinaisons à peu près équivalentes le sont.

Enfin, ces résultats on été obtenus sur un HP-95 équipé d'un clavier US. Il est donc normal que certaines touches renvoient un résultat sensiblement différent sur la version francaise de la machine. Si, en testant d'autres versions du HP-95, je trouve des différences notables par rapport à ces résultats, je vous le ferais savoir.

Jacques Belin (123)

```
/*****************************************************************************/
/*****************************************************************************/
/**                          VMEM95.C                                       **/
/**              Visualisation dynamique de la mémoire                      **/
/**                                                                         **/
/**   Création : 20/09/91                          Auteur : Jacques Belin   **/
/**   Dernière modification : 16/05/92                                      **/
/**    Ajout de msg_box()                                                   **/
/**                                                                         **/
/**  Note : Développé et testé uniquement sur Turbo C++ 1.01                **/
/**         TASM nécessaire (pour l'assemblage en ligne)                    **/
/*****************************************************************************/
/*****************************************************************************/
#pragma inline                          /*  Autorisation assemblage en ligne */

#include <stdio.h>
#include <dos.h>
#include <io.h>
#include <conio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <bios.h>
#include <ctype.h>


/*--------- Codes de quelques touches tels que GETKEY() les renvoie ---------*/
#define ESC     27                                  /* Code de touche Escape */
#define F1      315                                     /* Code de touche F1 */
#define F2      316                                     /* Code de touche F2 */
#define F3      317                                     /* Code de touche F3 */
#define F4      318                                     /* Code de touche F4 */
#define F5      319                                     /* Code de touche F5 */
#define F6      320                                     /* Code de touche F6 */
#define F7      321                                     /* Code de touche F7 */
#define F8      322                                     /* Code de touche F8 */
#define F9      323                                     /* Code de touche F9 */
#define F10     324                                    /* Code de touche F10 */
#define HOME    327                                   /* Code de touche Home */
#define UP      328                                  /* Codes de curseur haut */
#define PgUp    329                                 /* Code de touche page up */
#define END     335                                     /* Code de touche Fin */
#define DOWN    336                                   /* Codes de curseur bas */
#define PgDn    337                                /* Code de touche page down */
#define SF1     340                                 /* Code de touche Shift+F1 */
#define SF2     341                                 /* Code de touche Shift+F2 */
#define SF3     342                                 /* Code de touche Shift+F3 */
#define SF4     343                                 /* Code de touche Shift+F4 */
#define SF5     344                                 /* Code de touche Shift+F5 */
#define SF6     345                                 /* Code de touche Shift+F6 */
#define SF7     346                                 /* Code de touche Shift+F7 */
#define SF8     347                                 /* Code de touche Shift+F8 */
#define SF9     348                                 /* Code de touche Shift+F9 */
#define SF10    349                                /* Code de touche Shift+F10 */

#define NB_LINES  13                               /* nombre de lignes de dump */
#define NB_BYTES   8                              /* nombre d'octets par ligne */


#define CRT_START ((unsigned far *) MK_FP(0x40, 0x4E))     /* ptrs Ram Video */
#define ADDR_6845 ((unsigned far *) MK_FP(0x40, 0x63))


struct velb             /* structure dans mem vidéo d'un caractère affiché */
        {
         unsigned char caract;                                 /* code ASCII */
         unsigned char attr;                              /* attribut de couleur */
        };
typedef struct velb far * VP;

typedef struct                                  /* redéfinition de pointeur long */
        {
         unsigned int seg;
         unsigned int off;
        } FULLPTR;

                                              /* prototypes des fonctions utilisées */
unsigned int GetKey(void);
int  chk_addr(char *string, FULLPTR *addr);
void incptr(FULLPTR *currptr, unsigned decalage);
void decptr(FULLPTR *currptr, unsigned decalage);
int  search(char *string, FULLPTR *addr);
void disp_help(void);
void msg_box(char *message);


FULLPTR last_fnd;                  /* addresse de la derniere chaine trouvée */
int auto_SF10=0;           /* flag d'autorisation de l'appui sur "Shift F10" */

                                                    /* Texte de la 1ere ligne */
char *titre     = "VMEM95              Address = 0000:0000";
                                                   /* texte des lignes de menu */
char *helpline0 = "   s+100    s+1    o+100    o+1    find";
char *helpline  = "s+1000  s+10  o+1000    o+10   addr    ";



/*****************************************************************************/
/*                            Module Principal                               */
/*****************************************************************************/
void main(int argc, char *argv[])
  {
   int fin=0;                                        /* flag de fin du programme */
   FULLPTR curr_addr={0, 0};        /* pointeur sur le premier octet affiché */
   char s_newaddr[20];       /* tampon alpha d'entrée d'une nouvelle adresse */
   int sens;     /* sens (incrémentation/décrémentation) d'action des touches */
   VP d_vptr;                                          /* adresse RAM Video */
   VP dl_vptr;                 /* pointeurs sur adresse vidéo 1ere ligne dump */
   unsigned seg_vptr, off_vptr;      /* décomposition pointeur adresse vidéo */
   int key;                                              /* code de touche */
   int ret;                                    /* code de retour de fonction */
   int errflag=0;                                         /* flag d'erreur */


   if(argc > 2)                          /* si 2 parametre (ou plus) -> erreur */
     errflag=1;

   if((!errflag) && (argc == 2))                            /* si 1 parametre */
      errflag=chk_addr(argv[1], &curr_addr);          /* controle du format */

   if(errflag == 1)                                             /* si erreur */
     {
      fprintf(stderr, "usage : vmem95 [segment][:offset]\n");
      exit(-1);
     }
              /* acquisition adresse RAM Vidéo (pour tous types de cartes) */
   d_vptr= (VP) MK_FP((*ADDR_6845 == 0x3B4) ? 0xB000 : 0xB800, 0);
                                    /* calcul adresse vidéo de la 2eme ligne */
   dl_vptr = (VP) ((unsigned char far *) d_vptr + *CRT_START) + 80;
   seg_vptr=FP_SEG((void far *) dl_vptr);    /* séparation en segment/offset */
   off_vptr=FP_OFF((void far *) dl_vptr);
```

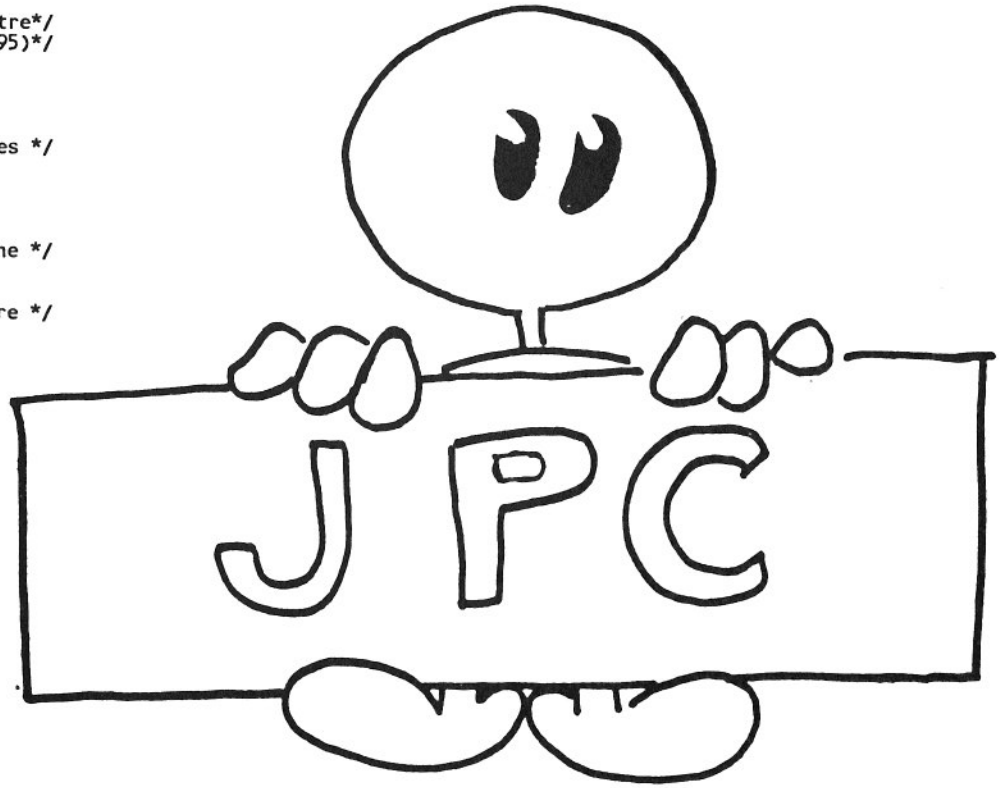
```
window(1, 1, 40, 16);   /* aff. 1ere ligne en gris clair (noir sur le 95) */
textbackground(LIGHTGRAY);
clrscr();

window(1, 2, 40, 14);         /* aff. zone Dump en noir (blanc sur le 95) */
textbackground(BLACK);
textcolor(LIGHTGRAY);
clrscr();

window(1, 1, 40, 16);
textbackground(LIGHTGRAY);  /* sel. couleur texte menu (blanc/noir sur 95)*/
textcolor(BLACK);

gotoxy(1, 1);
cputs(titre);                                /* affichage 1ere ligne écran */
gotoxy(1, 15);
cputs(helpline0);                             /* affichage 1ere ligne menu */
disp_help();                              /* affichage de la 2eme ligne d'aide */

do
  {
   gotoxy(31, 1);                        /* affichage adresse 1er octet affiché */
   cprintf("%04X:%04X",  curr_addr.seg, curr_addr.off);

   do                                                      /* affichage du dump */
     {
       asm     push    es           /* sauvegarde  des registres de segment */
       asm     push    di
       asm     push    ds
       asm     push    si

       asm     mov     ax,curr_addr.seg    /* initialisation des pointeurs */
       asm     mov     bx,curr_addr.off
       asm     mov     cx,seg_vptr
       asm     mov     dx,off_vptr

       asm     mov     es,ax                   /* segment:offset dans es:di */
       asm     mov     di,bx
       asm     mov     ds,cx                                /* vptr dans ds:si */
       asm     mov     si,dx

       asm     mov     dh,NB_LINES
           next_line :
       asm     push    es                        /* sauvegarde segment:offset */
       asm     push    di
                                            /* affichage adresse sur 5 digits */
       asm     mov     bx,di                  /* calcul adresse sur 5 digits */
       asm     mov     cl,4
       asm     shr     bx,cl
       asm     mov     ax,es
       asm     add     bx,ax
       asm     mov     ch,bh              /* affichage des 2 premiers nibbles */
       asm     call    dsp_hexbyte
       asm     mov     ch,bl              /* affichage des 2 nibbles suivants */
       asm     call    dsp_hexbyte
       asm     mov     bx,di              /* affichage nibble de poids faible */
       asm     mov     ch,bl
       asm     call    dsp_digit
       asm     add     si,4                          /* on passe deux espaces */

       asm     mov     dl,NB_BYTES
           next_hex :                /* affichage des octets sous forme hexa */
       asm     mov     ch,es:[di]                        /* lecture de l'octet */
       asm     call    dsp_hexbyte                               /* affichage */
       asm     inc     si                                /* on passe un espace */
       asm     inc     si
       asm     inc     di                                 /* incrémentation offset */
       asm     jnz     no_inc_seg0
       asm     mov     ax,es                      /* si offset = 0, maj segment */
       asm     add     ax,1000h
       asm     mov     es,ax
           no_inc_seg0 :
       asm     dec     dl
       asm     jnz     next_hex /* boucle tant que dernier caract non aff. */

       asm     inc     si                                 /* on passe un espace */
       asm     inc     si

       asm     pop di                /* réinit segment:offset en début de ligne */
       asm     pop es
       asm     mov     dl,NB_BYTES
           next_char :                 /* affichage des octets sous forme ASCII */
       asm     mov     ch,es:[di]                        /* lecture de l'octet */
       asm     mov     ds:[si],ch                 /* écriture en mémoire vidéo */
       asm     inc     si                  /* on passe a l'adresse vidéo suivante */
       asm     inc     si
       asm     inc     di                                 /* incrémentation offset */
       asm     jnz     no_inc_seg1
       asm     mov     ax,es                      /* si offset = 0, maj segment */
       asm     add     ax,1000h
       asm     mov     es,ax
           no_inc_seg1 :
       asm     dec     dl
       asm     jnz     next_char  /*boucle tant que dernier caract non aff.*/

       asm     add     si,80                    /* on passe à la ligne suivante */

       asm     dec     dh
       asm     jnz     next_line  /*boucle tant que derniere ligne pas aff.*/

       asm     pop     si                  /* récup valeurs init de es:di et ds:si */
       asm     pop     ds
       asm     pop     di
       asm     pop     es

       asm     jmp     SHORT  end_display  /* on saute les sous programmes */

                                                          /* SOUS PROGRAMMES */
       asm dsp_hexbyte :                  /* affichage d'un octet sous forme hexa */
       asm     mov     ah,ch                     /* entrée : CH = code à afficher */
       asm     mov     cl,4                      /* décalage nibble de poids fort */
       asm     shr     ch,cl
       asm     call    dsp_digit                  /* affichage nibble poids fort */
       asm     mov     ch,ah                        /* récupération de l'octet */
       asm     call    dsp_digit              /* pour affichage octet poids faible */
       asm     ret

       asm dsp_digit :                              /* affichage d'un digit en hexa */
                                                  /* entrée : CH = nibble à afficher */
       asm     and     ch,0Fh                    /* on ne garde que le 1er nibble */
       asm     cmp     ch,9                                /* si ch superieur à 9 */
       asm     ja      hex_digit                                   /* c'est un hexa */
       asm     add     ch,'0'                             /* conversion en '0'..'9' */
       asm     jmp     SHORT display
           hex_digit :
       asm     add     ch,('A'-10)                        /* conversion en 'A'..'F' */
           display :
       asm     mov     ds:[si],ch                    /* ecriture en mémoire vidéo */
       asm     inc     si                         /* décalage ptr sur mémoire vidéo */
       asm     inc     si
       asm     ret
```

```
      end_display :
      ;
   }
while(!kbhit());                    /* tant qu'aucune touche n'est pressée */
                    /* (c'est ce qui permet l'affichage en temps réel) */

key=GetKey();                       /* acquisition du code de la touche */

sens = 1;
switch(key)                           /* suivant le code de la touche */
  {
                                   /* décalage d'une ligne vers le haut */
   case DOWN  : incptr(&curr_addr, NB_BYTES);
                break;

                                   /* décalage d'une ligne vers le bas */
   case UP    : decptr(&curr_addr, NB_BYTES);
                break;
                                                      /* page suivante */
   case PgDn  : incptr(&curr_addr, NB_BYTES*NB_LINES);
                break;
                                                    /* page précédente */
   case PgUp  : decptr(&curr_addr, NB_BYTES*NB_LINES);
                break;

                                      /* affichage début de la mémoire */
   case HOME  : curr_addr.seg = 0;
                curr_addr.off = 0;
                break;

                                       /* affichage fin de la mémoire */
   case END   : curr_addr.seg = 0;
                curr_addr.off = 0;
                decptr(&curr_addr, NB_BYTES*NB_LINES);
                break;

               /* incrémentations/décrémentations des offsets/segments */
   case SF1   : sens = -1;
   case F1    : curr_addr.seg += 0x1000*sens;
                break;

   case SF2   : sens = -1;
   case F2    : curr_addr.seg += 0x0100*sens;
                break;

   case SF3   : sens = -1;
   case F3    : curr_addr.seg += 0x0010*sens;
                break;

   case SF4   : sens = -1;
   case F4    : curr_addr.seg += 0x0001*sens;
                break;

   case SF5   : sens = -1;
   case F5    : curr_addr.off += 0x1000*sens;
                break;

   case SF6   : sens = -1;
   case F6    : curr_addr.off += 0x0100*sens;
                break;

   case SF7   : sens = -1;
   case F7    : curr_addr.off += 0x0010*sens;
                break;

   case SF8   : sens = -1;
   case F8    : curr_addr.off += 0x0001*sens;
                break;

                                      /* sélection d'une nouvelle adresse */
   case F9    : auto_SF10=0;          /* Shift F10 n'est plus autorisée */
                gotoxy(14, 1);
                cputs("   New Address =           ");
                gotoxy(31, 1);
                s_newaddr[0]=10;       /* nombre de caractères maxi+1 */
                cgets(s_newaddr);        /* lecture chaine au clavier */
                                    /* controle du format de la chaine */
                ret=chk_addr(s_newaddr+2, &curr_addr);
                if(ret != 0)                              /* si erreur */
                  {
                   msg_box("Invalid Address Format");
                   gotoxy(1, 16);
                   cputs(helpline);       /* réaffichage ligne d'aide */
                  }
                gotoxy(1, 1);               /* réaffichage 1ere ligne */
                cputs(titre);
                break;

   case F10   : gotoxy(14, 1);
                cputs("String Search :            ");
                s_newaddr[0]=11;       /* nombre de caractères maxi+1 */
                gotoxy(30, 1);
                cgets(s_newaddr);                /* acquisition chaine */
                last_fnd.seg = last_fnd.off = 0;      /* départ à 0:0 */
                search(s_newaddr+2, &curr_addr);            /* cherche */
                gotoxy(1, 1);                  /* réaffichage du titre */
                cputs(titre);
                break;

   case SF10  : if(auto_SF10)                  /* si Shift F10 autorisé */
                  {                           /* on relance la recherche */
                   search(s_newaddr+2, &curr_addr);
                  }
                break;

   case ESC   : fin=1;                              /* fin du programme */
                break;
      }
  }
 while(!fin);

 textbackground(BLACK);                          /* effacement de l'écran */
 textcolor(LIGHTGRAY);
 clrscr();
 printf("VMEM95 Version 2.00\n");
}


/*****************************************************************************/
/* GETKEY: lecture d'un caractère au clavier                                 */
/* Entrée  : Aucune                                                          */
/* Sortie  : Code de touche reçue        < 256 : touche normale              */
/*                                      >= 256 : touche étendue              */
/*****************************************************************************/
unsigned int GetKey(void)
  {
   union REGS Register;  /* Variable de registres pour appel d'interruption */


   do
     {
```

```
        Register.h.ah = 2;       /* Numéro de fonction pour Lire état clavier */
        int86(0x16, &Register, &Register);    /* Appeler int. clavier du BIOS */
       }
     while(!bioskey(1));                /* Répéter jusqu'à ce que touche prête */
     Register.h.ah = 0;                    /* N° de fonction pour Lire touche */
     int86(0x16, &Register, &Register);  /* Appeler interruption clavier BIOS */
     return((Register.h.al) ? Register.h.al : Register.h.ah | 256);
    }



/*****************************************************************************/
/* CHK_ADDR : Vérification d'une chaine au format "segment:offset"           */
/*            Formats admis : "segment" ou ":offset" ou "segment:offset"     */
/*            avec "segment" et "offset" compris entre 0000 et FFFF          */
/* Entrées :  string = chaine a tester                                       */
/* Sorties :  addr = adresse correspondant au contenu de  la chaine. Si le   */
/*                 segment ou l'offset n'est pas spécifié, la variable       */
/*                 correspondante n'est pas modifiée                         */
/* valeur retournée : 1 si erreur de format, 0 sinon.                        */
/*****************************************************************************/
int chk_addr(char *string, FULLPTR *addr)
   {
    unsigned int sg;                /* offset et segment convertis en entier */
    unsigned int of;
    char s_seg[10];                                  /* chaines temporaires */
    char s_off[10];
    int dp_found = 0;               /* flag indiquant que le ':' est trouvé */
    int pres_seg=0;                       /* flag de présence d'un segment */
    int pres_off=0;                        /* flag de présence d'un offset */
    int i1, i2;                                 /* indices sur les chaines */
    int errflag = 0;                                        /* flag d'erreur */


    strupr(string);                         /* conversion chaine en majuscules */

    memset(s_seg, 0, 10);                   /* initialisation de la chaine à 0 */
    memset(s_off, 0, 10);

    i2=0;
    for(i1=0; i1<strlen(string); i1++)  /* pour tous les caract. de "string" */
      {
       if(string[i1]==':')                                    /* si ':' trouvé */
         {
          dp_found = 1;
          i2=0;
         }
       else                                                        /* sinon */
         {
          if(dp_found)                                  /* si ':' a été trouvé */
            {
             pres_off=1;   /* le caract. dans "string" appartient à l'offset */
             s_off[i2++]=string[i1];
            }
          else                                                     /* sinon */
            {
             pres_seg=1;   /* le caract. dans "string" appartient au segment */
             s_seg[i2++]=string[i1];
            }
         }
      }
                     /* si les chaines ont une longueur sup. à 4 caractère */
    if((strlen(s_seg) > 4) || (strlen(s_off) > 4))
       errflag = 1;                                              /* erreur */

    if(!errflag)                              /* si pas d'erreur précédement */
      {
       if(pres_seg)                                /* si présence d'un segment */
          if(sscanf(s_seg, "%x", &sg) != 1)          /* test si format Hexa */
             errflag = 1;
       if(pres_off)                                 /* si présence d'un offset */
          if(sscanf(s_off, "%x", &of) != 1)          /* test si format Hexa */
             errflag = 1;
      }

     if(!errflag)                              /* si pas d'erreur précédement */
      {
       if(pres_seg)                            /* stockage du nouveau segment */
          addr->seg=sg;
       if(pres_off)                              /* stockage du nouvel offset */
          addr->off=of;
      }

     return(errflag);                                /* retour du flag d'erreur */
    }



/*****************************************************************************/
/* INCPTR : Incrémentation d'un pointeur, tenant compte des limites de       */
/*          segment.                                                         */
/* Entrées :  addr = adresse du pointeur                                     */
/*            décalage = décalage à appliquer                                */
/* valeur retournée : aucune                                                 */
/*****************************************************************************/
void incptr(FULLPTR *addr, unsigned decalage)
   {
    int old_offset;                                        /* ancien offset */


    old_offset = addr->off;                 /* sauvegarde de l'ancien offset */
    addr->off += decalage;                                        /* décalage */
    if(addr->off < old_offset)           /* si dépassement de limite d'offset */
       addr->seg += 0x1000;                          /* mise à jour du segment */
   }



/*****************************************************************************/
/* DECPTR : Décrémentation d'un pointeur, tenant compte des limites de       */
/*          segment.                                                         */
/* Entrées :  addr = adresse du pointeur                                     */
/*            décalage = décalage à appliquer                                */
/* valeur retournée : aucune                                                 */
/*****************************************************************************/
void decptr(FULLPTR *addr, unsigned decalage)
   {
    int old_offset;                                        /* ancien offset */


    old_offset = addr->off;                 /* sauvegarde de l'ancien offset */
    addr->off -= decalage;                                        /* décalage */
    if(addr->off > old_offset)           /* si dépassement de limite d'offset */
       addr->seg -= 0x1000;                          /* mise à jour du segment */
   }



/*****************************************************************************/
/* SEARCH : Recherche d'une chaine en mémoire.                               */
/* Entrées :  string = chaine a trouver                                      */
/* Sorties :  addr = adresse du premier caractère de la chaine, si trouvée.  */
```

```
/*                    sinon, pas de changement                              */
/* met à jour : last_fnd = adresse de la derniere chaine trouvée, qui sera   */
/*                         le nouveau départ si Shift F10                    */
/* valeur retournée : 1 si chaine trouvée,   0 sinon.                        */
/*****************************************************************************/
int search(char *string, FULLPTR *addr)
   {
    int  found;                  /* flag indiquant que la chaine est trouvée */
    unsigned seg_str, off_str;              /* segment et offset de "string" */


    strupr(string);                      /* conversion "string" en majuscules */

    seg_str = FP_SEG((char far *) string); /* decomposition adr. de "string" */
    off_str = FP_OFF((char far *) string);

    gotoxy(1, 16);                                  /* affichage du message */
    cputs(" Searching...                          ");


    asm     push    es              /* sauvegarde  des registres de segment */
    asm     push    di
    asm     push    ds
    asm     push    si

    asm     mov     ax,last_fnd.seg          /* initialisation des pointeurs */
    asm     mov     bx,last_fnd.off
    asm     mov     cx,seg_str
    asm     mov     dx,off_str

    asm     mov     es,ax                              /* 'sch' dans es:di */
    asm     mov     di,bx
    asm     mov     ds,cx                             /* string dans ds:si */
    asm     mov     si,dx

    asm     inc     di                          /* incrémentation adresse 'sch' */
    asm     jnz     no_inc_seg0
    asm     mov     ax,es                          /* si offset=0, incr segment */
    asm     add     ax,1000h
    asm     mov     es,ax
        no_inc_seg0 :
    asm     mov     ch,ds:[si]                 /* lecture du premier caractère */
    asm     mov     dx,0                          /* initialisation found à 0 */

        loopgen :                               /* boucle sur toute la mémoire */
    asm     mov     bl,es:[di]                      /* lecture *'sch' dans bl */

    asm     cmp     bl,'a'                                /* conversion en upr */
    asm     jb      suite0
    asm     cmp     bl,'z'
    asm     ja      suite0
    asm     sub     bl,20h
        suite0 :
    asm     cmp     bl,ch           /* comparaison 1er octet des deux chaines */
    asm     jne     next_addr        /* si non égaux, on reprend la recherche */
    asm     push    es              /* sauvegarde adresse courante sch et str */
    asm     push    di
    asm     push    ds
    asm     push    si
        loop_cmp :                             /* comparaison des deux chaines */
    asm     inc     si                       /* incrémentation adresse str */
    asm     jnz     no_inc_seg1
    asm     mov     ax,ds                          /* si offset=0, incr segment */
    asm     add     ax,1000h
    asm     mov     ds,ax
        no_inc_seg1 :
    asm     mov     bh,ds:[si]                 /* lecture caract de str dans bh */
    asm     cmp     bh,0          /* si fin de str, les 2 chaines sont egales */
    asm     je      founded
    asm     inc     di                          /* incrémentation adresse 'sch' */
    asm     jnz     no_inc_seg2
    asm     mov     ax,es                          /* si offset=0, incr segment */
    asm     add     ax,1000h
    asm     mov     es,ax
        no_inc_seg2 :
    asm     mov     bl,es:[di]               /* lecture caract de 'sch' dans bl */
    asm     cmp     bl,'a'                                /* conversion en upr */
    asm     jb      suite1
    asm     cmp     bl,'z'
    asm     ja      suite1
    asm     sub     bl,20h
        suite1 :
    asm     cmp     bh,bl                  /* comparaison des deux caractères */
    asm     jne     not_founded     /* si non egaux, on continue la recherche */
    asm     jmp     SHORT loop_cmp      /*si égaux, on regarde caract suivants*/

        founded :                                          /* si chaine trouvée */
    asm     mov     dx,1                              /* armement flag found */
        not_founded :
    asm     pop     si          /* récupération adresses str et sch courantes */
    asm     pop     ds
    asm     pop     di
    asm     pop     es
        next_addr :                     /* poursuite de la recherche dans 'sch' */
    asm     cmp     dx,1                                /* fin si chaine trouvée */
    asm     je      fin
    asm     inc     di                          /* incrémentation adresse 'sch' */
    asm     jnz     loopgen                  /* suite recherche si offset != 0 */
    asm     mov     ax,es                          /* si offset=0, incr segment */
    asm     add     ax,1000h
    asm     mov     es,ax
    asm     cmp     ax,0000           /* es:di != 0000:0000, suite recherche */
    asm     jne     SHORT loopgen
        fin :
    asm     mov     ax,es          /* récupération valeurs seg_sch et off_sch */
    asm     mov     bx,di
    asm     pop     si       /* récupération anciennes valeurs es:di et ds:si */
    asm     pop     ds
    asm     pop     di
    asm     pop     es
    asm     mov     last_fnd.seg,ax       /* initialisation last_fnd = 'sch' */
    asm     mov     last_fnd.off,bx
    asm     mov     found,dx                   /* récupération valeur de found */

    if(found)                                              /* si chaine trouvée */
      {
       auto_SF10=1;                        /* autorisation de l'accès à Shift F10 */
       addr->seg = last_fnd.seg;              /* stockage de l'adresse trouvées */
       addr->off = last_fnd.off;
       gotoxy(1, 16);                                   /* affichage du message */
       cputs(" Shift-F10 to continue the Search     ");
      }
    else                                                  /* si chaine non trouvée */
      {
       auto_SF10 = 0;              /* la touche Shift F10 n'est plus accessible */
       msg_box("    String not found");                    /* affichage du message */
       disp_help();                          /* affichage derniere ligne du menu */
      }
    return(found);                                  /* retour 1 si chaine trouvée */
}
```

```
/*****************************************************************************/
/* DISP_HELP : affichage par défaut de la derniere ligne de menu             */
/* Entrées :  aucune                                                         */
/* valeur retournée : aucune                                                 */
/*****************************************************************************/
void disp_help(void)
   {
    gotoxy(1, 16);
    cputs(helpline);
   }



/*****************************************************************************/
/* MSG_BOX : Affichage d'une fenêtre de message, avec attente de pression    */
/*           de touche.                                                      */
/* Entrées : message = chaine contenant le texte du message (22 caract. maxi)*/
/* valeur retournée : aucune                                                 */
/*****************************************************************************/
void msg_box(char *message)
   {
    char under_box[2*24*2];


    gettext(9, 7, 32, 8, under_box); /* sauvegarde de la zone sous la fenêtre*/
    window(9, 7, 32, 8);            /* visualisation fenêtre, noir/blanc (95)*/
    textbackground(LIGHTGRAY);
    textcolor(BLACK);
    clrscr();

    gotoxy(2, 1);                              /* affichage lignes de textes */
    cputs(message);
    gotoxy(7, 2);
    cputs("Press any Key");

    GetKey();                                /* attente de pression de touche */

    window(1, 1, 40, 16);
    puttext(9, 7, 32, 8, under_box);    /* restauration zone sous la fenêtre */
   }
```



Le Journal JPC est le bulletin de liaison entre les membres de l'Association "PPC Paris", régie par la loi de 1901. Le Club est éditeur de JPC, et son siège social est au 56, rue Jean-Jacques Rousseau, 75001 Paris.

La maquette de ce numéro a été préparée et réalisée par Jacques Belin et Asdin Aoufi.

Les dessins sont de Jean-Jacques Dhénin et Paul Courbis.

Directeur de la publication : Jacques Belin
Numéro ISSN : 0762 - 381X

Veuillez adresser toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01.