JPC

MAI 1988

NUMERO 54

Le numéro 40 F

A PROPOS DU CLUB

HP28

HP75

HP71

EDITORIAL

Chers Pêcheurs,

J'espère que vous n'avez pas été trop nombreux à mordre à l'hameçon de notre numéro d'avril. La firme suédoise "zoo de Vincennes" n'existe réellement pas... Allez ! Ce n'était pas bien méchant.

Ce mois-ci, il n'y a que du sérieux. Par exemple, l'article de Serge Vaudenay représente un travail colossal. Mais le résultat en vaut la peine ! Un langage de plus sur HP-71, c'est un événement majeur...

La rubrique HP-28 (et HP-27) augmente de volume. Tant mieux ! Continuez à nous envoyer vos articles, vos programmes et vos astuces. Faites de cette petite machine une grande réussite !

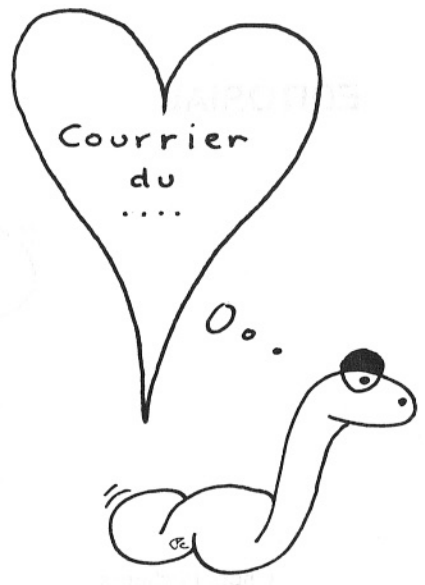
En revanche, vous ne verrez aucun article pour HP-41. N'y a-t-il donc plus de possesseur de HP-41 dans le Club ? Rassurez-moi, écrivez.

En attendant de lire vos contributions dans *JPC*,

Pierre David (37)

Gilles Barret
8 rue du Prieuré
95000 Cergy
Tél : (1) 30 31 94 04 (après 19h)

Vend :
Imprimante HP-82162A (12/83) : 1500 F, lecteur de cassettes HP-82162A (6/83) : 1500 F, interface vidéo 32 colonnes HP-82163A : 400 F, interface Minitel (nécessite un convertisseur HP-82166A) : 300 F, convertisseur HP-82166A : 1200 F, module 32 Ko Ram (CMT) : 1000 F, module Curve Fit pour HP-71 : 500 F, module HP-IL pour HP-71 : 1000 F, imprimante Seikosha SP800 (utilisable avec convertisseur HP-82166A) : 2500 F, moniteur Goldstar : 500 F, le tout en excellent état.



Sébastien Lalande
12 rue de Seine
78920 Croissy sur Seine
Tél : (1) 39 76 27 41

Fait :
Pose de modules 4, 32 ou 64 Ko pour HP-28C ; accélération HP-28C et connexion entre HP-28C et autres ordinateurs.

Vend :
Module 32 Ko Ram : 1000 F.

Scandale à PPC :
Le module 49.5 E.P. était un poisson d'avril !

Alain Villatte
3 rue Emile Masson
56300 Pontivy

Vend :
HP-71 : 2400 F, imprimante ThinkJet : 2100 F, module HP-IL : 500 F, Texte : 400 F, Finance : 400 F, Forth + HP-41 : 500 F (tous ces modules pour HP-71).

HP28

UN TRESOR CACHE DU HP-27S

Le tout nouveau calculateur scientifique de Hewlett-Packard, le HP-27S, présente une caractéristique tout à fait étonnante que j'ai trouvée un peu par hasard : il est capable de visualiser en hexadécimal le contenu de sa mémoire. Le fonctionnement est similaire à celui du debugger du HP-71 ou de la commande SYSEDIT de *JPC Rom* avec toutefois beaucoup moins de facilité : vous ne pouvez que visualiser la mémoire mais pas la modifier.

Pour activer ce mode il faut appuyer simultanément sur [CLR] et sur la troisième touche de menu à partir de la droite. Vous voyez alors apparaître trois barre verticales : une à chaque extrémité et une au centre. Vous relâchez les deux touches enfoncées et vous appuyez sur [÷]. La ligne supérieure de l'afficheur contiendra alors :

0209D:792D7A491F9830055

0209D est l'adresse et 792D7A491F9830055 représente le contenu de la mémoire à partir de cette adresse.

Vous disposez alors des touches suivantes :
- [÷] décrémente l'adresse de 100_{16} quartets,
- [×] incrémente l'adresse de 100_{16} quartets,
- [-] décrémente l'adresse de 1 quartet,
- [+] incrémente l'adresse de 1 quartet,
- [TAN], découverte par Jean-Jacques Dhénin, permet d'imprimer le contenu de la mémoire et incrémente l'adresse de 10_{16} quartets,
- [CLR] et, simultanément, la troisième touche de menu à partir de la gauche provoque un MACHINE RESET qui vous sort de ce mode de visualisation de la mémoire.

A partir de ce moment en jouant sur ces touches, il vous est possible d'examiner tout le contenu de la mémoire Ram ou Rom.

Quelques résultats d'une étude très rapide :

- la Rom fait 64 Ko et se trouve située entre 00000_{16} et 20000_{16}.

- les textes de copyright et de version se trouvent, comme dans le HP-28 (voir *JPC 47* page 14), dans les adresses hautes de la Rom. On a en effet :

```
1FFC2:34F4059525947484   soit  COPYRIGH
1FFD2:4502840502139383   soit  T HP 198
1FFE2:7300F4B4D2237335   soit  7.OK-27S
1FFD2:D2451481BF9564     soit  -E.....
```

Les . représentent des caractères non ASCII. Le premier est 00 et sert de séparateur entre les deux messages, les derniers correspondent vraisemblablement aux sommes de contrôle de la Rom.

- La Ram commence en 50000_{16} et semble se répéter tous les 4000_{16}. Une caractéristique similaire a été observée sur le HP-28S.

Je n'ai pas eu le temps d'explorer davantage : c'est à vous de poursuivre. N'hésitez pas à envoyer vos découvertes au Journal si vous trouvez de nouvelles séquences de touches intéressantes (si seulement on pouvait modifier directement le pointeur d'adresse ou le contenu de la mémoire !) ou si vous faites des trouvailles sur la structure interne de la machine. Concluons par un grand merci à HP pour avoir laissé la porte entrouverte !

Janick Taillandier (246)

CHOCS EN RETOUR

Le programme PCNP de Guillaume Le Stum (*JPC 52* page 5) donne des résultats aberrants comme celui-ci :
$C_n^P(540,254) = 1$ au lieu de 4,76...E160
Aucune parade n'étant prévue en cas d'overflow.

Il est possible de prévenir des résultats illicites en modifiant le programme PANP :

```
«
  63 CF
  OVER SWAP - 1 3 ROLLD
  WHILE
    DUP 2 >
  REPEAT
    1 + ROT OVER * 3 ROLLD
  END
  DROP2
  63 FS?
  « 1000 1 BEEP » IFT
»
```

Guy Toublanc (276)

Dans mon article HP-28GTI (JPC 53), il s'est glissé une erreur. En effet, page 9 en haut à gauche la suite du programme de la page 8 :

```
 WHILE ... DROP »
```

est fausse comme me l'a gentiment signalé Jean-Philippe Amans. En voici une autre version :

```
DROP
DUP SIZE 1 SWAP
FOR
  X DUP X X SUB NUM 256 + R→B →STR
  4 99 SUB 18
  FOR Y DUP Y Y SUB
    IF "1" ==
    THEN
      4000 .001 BEEP
    ELSE
      4000 .0000001 BEEP
    END
  NEXT
  DROP
NEXT
DROP »
```

Sébastien Lalande (442)

LE FACTEUR EST CONSTANT

Les lignes qui suivent s'inspirent d'un article intitulé *Constante arithmétique en RPN*, article paru dans les numéros 13 et 14 de PPC-T et destiné au HP-41.

Ouvrages cités :

- *Algorithms for RPN Calculators* par John Ball. (1978) John Wiley & Sons.

- *Extend your HP-41* par W.A.C. Mier-Jedrzejowicz. (1985).

- *Le calculateur programmable de poche et ses jeux* par Didier Guérin, Pierre Vaschalde, André Warusfel. (1978) Hachette.

- *HP-41CX Manuel d'utilisation* Volumes 1 et 2 (1984).

- *HP-28C Manuel de référence* (octobre 1986).

Les exemples sont empruntés à plusieurs manuels de Hewlett-Packard et d'autres constructeurs. Voici quelques passages des ouvrages précités :

Extend your HP-41, page 49 :

« De nombreux et nouveaux utilisateurs des calculateurs RPN se plaignent de n'y pas trouver un dispositif de *facteur constant*. Ils ne se rendent pas compte que le registre L et LASTX fournissent la même possibilité... Une autre possibilité de faire de l'arithmétique avec constante est d'utiliser la faculté qu'a la pile de descendre à chaque opération et de copier T en Z. Cela implique de placer la constante en X, de presser [ENTER] trois fois, puis de taper successivement les nombres à traiter et de presser l'une des touches [+], [-], [*] ou [/]. Après chaque calcul, il faut presser [CLX] puis le nombre suivant avant de déclencher le calcul avec la constante. Cela paraît plus compliqué que d'utiliser LASTX, mais cela donne la possibilité de soustraire d'une constante ou de diviser une constante, car la constante est en Y pas en X »

HP-28C Manuel de référence, pages 358-359.

Sous la rubrique *Pas de duplication automatique du registre T*, le manuel rappelle la possibilité citée précédemment et indique qu'elle est exclue sur le HP-28C, mais qu'il est facile de créer un programme, et il poursuit, sous la rubrique *Mémoire affectée à la pile opérationelle* :

« Une pile opérationnelle dynamique a l'avantage de vous permettre d'utiliser autant de niveaux que vous en avez besoin, sans craindre de perdre des objets «par le haut de la pile» lorsque vous en saisissez de nouveaux « dans le bas ». Mais cela a aussi l'inconvénient d'immobiliser une quantité importante de mémoire avec de « vieux » objets si vous les laissez dans la pile une fois vos calculs terminés »

Par les routines '?', '??', 'H' et 'Y', le présent programme élimine cet inconvénient et, d'autre part, exploite l'avantage énoncé auparavant, à savoir que le facteur constant est stocké en haut de pile (Top). Pour la simplicité de l'énoncé et par analogie avec le HP-41, le niveau 1 du HP-28C sera souvent nommé X et le niveau supérieur (Top niveau) T.

En outre, l'avantage cité par Mier-Jedrzejowicz de pouvoir soustraire d'une constante et de diviser une constante est mis en valeur au menu (2) par les routines 'R' et 'Q'. Le programme dispose en effet de cinq menus, dont trois menus principaux et deux menus accessoires (routines de service). Les menus principaux proposent huit opérations arithmétiques courantes (voir ci-dessous) ainsi que leur routine

d'initialisation (qui vide la pile de son contenu et place en X le facteur constant), et, à titre d'exemple, six routines d'applications financières ('I%', 'µ', 'Σ', '%Σ', 'N' et 'E').

Les nombreux calculateurs à facteur constant présentent bien l'avantage qu'ils indiquent, mais seulement pour une suite de calculs dont le cours ne doit pas être brisé par un évènement fortuit. Dans ce cas, deux inconvénients se manifestent :

(1) Si le calculateur ne comporte pas une touche [X↔Y] (étant entendu que le facteur constant serait stocké en Y), on ne peut rappeler le facteur constant pour en connaître la valeur. Il est vrai que certains calculateurs affichent le dernier opérateur arithmétique utilisé (+, -, *, /) mais, pour retrouver alors la valeur du facteur constant, il faut respectivement taper: 0 [=], 0 [=] [CHS], 1 [=], 1 [=] [1/X].

(2) Lorsqu'on passe d'un calculateur à un autre, on s'aperçoit que le facteur constant est tantôt - en général - le deuxième paramètre de l'opération diadique (voir Ball) tantôt le premier paramètre. Ici, le facteur constant est toujours entré en première position, puisqu'il est entré dans la routine d'initialisation. Dans leur ouvrage (page 16) Guérin, Vaschalde et Warusfel signalent, au cours de la description du processus, que « le coefficient a été mis automatiquement en facteur constant multiplicatif ». Mais qu'en est-il si l'on veut utiliser ce facteur constant comme facteur constant pour une addition, une soustraction, une division ?

Le présent programme tente de répondre à ces questions sur un HP-28C. L'affichage sur trois lignes permet de montrer en permanence le facteur constant sur la ligne supérieure (Top niveau). Le facteur constant se trouvant en haut de pile, on pourra, à tout moment, s'en servir pour effectuer l'une des autres opérations arithmétiques. Au fur et à mesure, un résultat pourra, soit être traité par les fonctions habituelles du calculateur, soit stocké dans la variable X réservée à cet usage, soit enfin utilisé comme nouveau facteur constant à l'aide de '?' ou '??'.

DESCRIPTION DES MENUS

1) Le maniement du programme est plus aisé si l'on retrouve toujours les mêmes menus dans l'ordre indiqué ci-dessous. Il est préférable, avant d'entrer les programmes, d'entrer les noms des variables (leur libellé ayant été écourté au maximum). Pour faire cela, stocker dans la pile sur trente niveaux une valeur (1, par exemple) en tapant sur la ligne de commande en mode alpha :

```
{ 30 } 1 CON ARRY→ DROP
```

puis [ENTER]. Entrer ensuite les noms des variables de droite à gauche et de bas en haut (du menu 5 au menu 1): X[STO], W[STO], etc... E[STO], N[STO], '%Σ'[STO], etc... ?[STO], X->[STO].

On pourra alors entrer les programmes dont on aura le plus besoin. Ils peuvent être tous présents en même temps dans la mémoire, mais, dans ce cas, le programme 'Σ' (portefeuille) ne pourra stocker que trois valeurs. Si l'on souhaitait en stocker un plus grand nombre, on pourrait, par exemple, vider de leur contenu les variables 'N', 'E' et leurs routines de service 'Z', 'U', 'V', 'W' et 'X'.

2) Pour disposer du maximum de mémoire, il est recommandé d'invalider les opérations de récupération de données [] [COMMAND], [] [UNDO] et [] [LAST].

Menus

```
Initialisations
           |‾‾‾|  |‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾|
1)  X->    | ? |  | A    S    D    M |  'Foncteurs'
           |   |  |----|  |    |      |   arithmé-
2)   I%    | ??|  | u  |  R    Q  | ->X |   tiques
           |‾‾‾|  MarkUp‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾
    |‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾| |‾‾‾‾‾‾‾‾‾| Taux nomi-
3)  |  Σ   | ■ |   C   %Σ | | N     E |  nal  <-->
     ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾   ‾‾‾‾‾‾‾‾‾  taux ef-
   | P o r t e f e u i l l e |            fectif

4)   G     H     I     J     K     L    Routines
                                           de
5)   Y     Z     U     V     W     X    service
```

Explication des symboles

Par analogie avec le calculateur HP-41 : T est le contenu du niveau supérieur, X du niveau 1.

Libellé du menu	Opération effectuée	Sort réservé à X
1er menu:		
X->	"X^ "	l'élever à 'puissance T'
A	"X+ "	lui Ajouter (Add) T
S	"X- "	lui Soustraire T
D	"X/ "	le Diviser par T
M	"X* "	le Multiplier par T
2ème menu:		Sort réservé à T
R	" -X"	Reste de T moins X
Q	" /X"	Quotient de T par X
->X	" ^X"	l'élever à 'puissance X'

Premier (et deuxième) menu

1) Reprise d'un nombre pour un calcul
Manuel HP-41CX volume 1, page 24

Exemple : Calculez

$$\frac{96.704 + 52.394706}{52.394706}$$

Appuyez sur :

```
52.394706 [?] 96.704 [A] [D]
```

Affichage :

```
3:      52.3947
2:   "149.0987/ "
1:       2.8457
```

Exemple : Parmi les étoiles les plus proches de la Terre, Centaure et Sirius se situent respectivement à 4.3 et 8.7 années de lumière. Utilisez *c*, vitesse de la lumière (9.5×10^{12} km par an), pour calculer leur distance par rapport à la Terre, en km.

```
9.5 [EEX] 12[?]  4.3 [M] Terre-Centaure: 4.0850E13
                 8.7 [M] Terre-Sirius:   8.2650E13
```

2) Calcul avec constante
Manuel HP-41CV (1980) page 41

Exemple: La population d'une culture bactériologique croît quotidiennement de 15%. Calculez, pour une culture de départ de 1000, la population en fin de journée pendant cinq jours consécutifs.

```
1.15 [?] 1000 [M]     après 1 jour:     1150.0000
              [M]     après 2 jours:    1322.5000
              [M]     après 3 jours:    1520.8750
              [M]     après 4 jours:    1749.0063
              [M]     après 5 jours:    2011.3572
```

Ces calculs de facteur constant peuvent se dérouler aussi facilement avec les quatre opérations.

Deuxième menu (suite)

'I%'
Cette routine réalise, au choix, deux objectifs :
1) Après avoir saisi en ligne de commande un taux d'intérêt I (en %), l'appui de `[??]` suivi de `[I%]` affiche :
- au niveau 3: le taux I qui vient d'être saisi
- au niveau 2: le taux I2, supérieur à I
- au niveau 1: le taux I1, inférieur à I

Pour une valeur I fixée comme *marge* ou *marque*, sont calculées les valeurs I1 & I2 qui l'encadrent.

Exemple: Quelles sont *marge* et *marque* si I = 20 ?

```
20 [??] [I%]      3:      20.0
                  2:      25.0
                  1:      16.7
```

Si I est considéré comme marge, la marque est 25.0
Si I est considéré comme marque, la marge est 16.7

Avantage: Le résultat trouvé peut être aussitôt saisi comme facteur constant, à utiliser avec la routine `[I%]`, comme ci-dessous (exemple).

2) Pour un taux de pourcentage donné (en %), soit positif soit négatif : à partir de diverses bases, calcul du montant-pourcentage (à ajouter ou à ôter) et du montant total, après imputation de ce pourcentage.

au niveau 3: en facteur constant, le pourcentage
au niveau 2: détail de l'opération
au niveau 1: montant total

Exemple (*Manuel d'utilisation HP-25*, page 62) : Un article coûtant 1500 F supporte une taxe de 6.5%. Quel est le montant de cette taxe et le prix de vente taxe comprise ? On pose les mêmes questions pour un article coûtant 1750 F.

```
6.5 [??] 1500 [I%]      3:          6.50
                        2: "1500.00+97.50"
                        1:       1597.50

         1750 [I%]      3:          6.50
                        2: "1750.00+113.75"
                        1:       1863.75
```

Exemple : La calculatrice financière spécialisée TI-44 propose, dans son prospectus, le problème suivant :
Détermination du prix de vente correct.
Vous souhaitez lancer une campagne de solde en préservant une marge de 20%. Quels seront les prix de vente d'articles achetés à 8.50 F et à 14.50 F ?

```
20 [??]  Si la marge est de  3:      20.00
         20.00%, la marque   2:      25.00
      est le taux supérieur, 1:      16.67
      soit: 25.00%.
[DROP][??] (25.00 en facteur constant)

8.5 [I%]                     3:      25.00
                             2: "8.50+2.13"
                             1:      10.63
```

```
14.5 [I%]                3:          25.00
                         2:    "14.50+3.63"
                         1:          18.13
```

'μ'

Calcul de la *marque* et de la *marge* entre une base et diverses valeurs.

Exemple : Dans le problème précédent, que deviendra votre marge si, au lieu de 10.63 F et 18.13 F, vous fixez le prix de vente respectivement à 9.90 F et à 19.90 F ?

```
8.5 [??] 9.9 [μ]         3:           8.5
                         2:    "8.5-CP-9.9"
                         1:    { 16.5 14.1 }
```

La marge sera le plus petit des deux nombres: 14.1%

```
14.5 [??] 19.9 [μ]       3:          14.5
                         2:   "14.5-CP-19.9"
                         1:    { 37.2 27.1 }
```

La marge sera le plus petit des deux nombres: 27.1%.

3ème menu

'Σ' - Portefeuille
Routine de saisie d'une série d'articles. Calcul du montant des postes, du total et du pourcentage du total (en %) pour chacun des postes.

Avant de lancer le programme, initialiser par appui de `0 [■]`, ce qui affiche 0 (zéro) aux niveaux 1 et 2.

Le programme permet au choix de saisir chacun des articles de deux façons :

1) Si, pour l'article, on connaît le montant du poste, le saisir en ligne de commande et appuyer sur `[Σ]`.

2) Si, pour l'article, on connaît le nombre d'exemplaires N et le prix unitaire P, saisir en ligne de commande N virgule P. P étant un prix, il peut comporter des décimales ; entrer celles-ci, après avoir tapé un point (décimal). Frapper `[Σ]`. Le montant du poste est calculé et s'affiche en X.

Après chaque saisie d'un poste, on peut connaître n'importe lequel des postes en entrant en ligne de commande le numéro # du niveau (où se trouve le poste que l'on veut interroger) et en tapant `[%Σ]`. L'affichage fait apparaître quelques instants (dont la durée est facilement modifiable) le montant du poste et son pourcentage du total.

Si l'on décide alors que la saisie est terminée, taper `0 [Σ]`.

Pour continuer la saisie du portefeuille, entrer les articles suivants de la manière notée ci-dessus. A tout moment, l'on peut corriger (ou supprimer) un poste # en se servant de la fonction "corbeille à papier" affectée à [■].

Lorsque tous les articles sont entrés, appuyer sur `[] [CONT]` ou sur `0 [Σ]`.

Le montant des postes et, pour chacun d'entre eux, le pourcentage du total apparaissent sur l'écran, le niveau 1 étant constitué par le dernier article entré.

Si toutefois, l'on veut attacher à chaque article son numéro d'ordre (niveau 1 : 1er article entré... niveau N : Nième article entré), appuyer sur `[C]`. Si l'on se ravise, appuyer à nouveau sur `[C]`.

Exemple modifié (*HP-27 Manuel d'utilisation*, p.64) : Vous achetez 150 actions Merlin-Gerin à 450 F, ce qui constitue, vous l'avez calculé, un poste de 67500 F, puis vous achetez 52 actions Locatel à 1404 F. Vous vous apercevez alors que vous avez commis une erreur en entrant le cours de l'action Locatel (1500 au lieu de 1404). Rectifiez l'erreur. Calculez alors le pourcentage du total de chacun des postes.

Vous décidez ensuite d'acheter 200 actions Stockvis à 1500 F.
Votre portefeuille ainsi constitué, vous souhaitez connaître, dans l'ordre où vous avez fait vos achats, le montant de chaque poste, le total et le pourcentage du total pour chacun des postes. Vous voulez modifier la présentation et imprimer votre portefeuille dans l'ordre où il apparaît sur un relevé bancaire (1ère valeur entrée: en haut).

```
(1) Initialiser                  2:          0.00
    0 [■]                        1:          0.00

(2) 67500 [Σ]          total     3:      67500.00
       nb. d'articles entrés     2:          1.00
            montant du poste     1:      67500.00

(3) 52,1500 [Σ]        (total    4:     145500.00)
       nb. d'articles entrés     3:          2.00
                  Merlin-Gerin   2:      67500.00
             erreur sur Locatel  1:      78000.00

(4) 1 [■]  Après correction de l'erreur, pile (2).

(5) 52,1404 [Σ]        (total    4:     140508.00)
       nb. d'articles entrés     3:          2.00
                  Merlin-Gerin   2:      67500.00
                       Locatel   1:      73008.00
```

```
(6) 1 [%Σ]          Locatel                51.96%
    2 [%Σ]     Merlin-Gerin                48.04%

(7) 200,1500 [Σ]     (total   4:          440508.00)
                Merlin-Gerin   3:           67500.00
                     Locatel   2:           73008.00
                    Stockvis   1:          300000.00

(8) [] [CONT]        (total   4:        "440508.00")
                Merlin-Gerin   3: "15.3%  67500.00"
                     Locatel   2: "16.6%  73008.00"
                    Stockvis   1:"68.1%  300000.00"

(9) Relevé bancaire: appuyez sur [C]

(10) Pour étudier un autre portefeuille, (1).
```

'%Σ' - Pourcentage en "tant pour cent" du total

En dehors de l'utilisation précitée au sein de la routine 'Σ', la routine '%Σ' rend le même service que la fonction préprogrammée '%T', avec l'avantage de conserver en T le total (facteur constant). Pour chaque valeur entrée et traitée, le résultat précédent est écrasé par le nouveau résultat. Il faut d'abord entrer la base au niveau 1. A la suite, entrer en ligne de commande chacune des valeurs dont on veut connaître le pourcentage du total, puis [%Σ].

Exemple : Par rapport à 340 quel est le "tant pour cent" de 64 ? Et de 86 ?

```
[] [CLEAR]
340 [ENTER] 64 [%Σ]          2:          340.00
                             1:           18.82

            86 [%Σ]          2:          340.00
                             1:           25.29
```

Exemple (*HP-12C Manuel d'utilisation*, page 31) : Le mois dernier, les résultats de votre société se chiffraient à 3.92 millions de francs aux U.S.A., à 2.36 millions de francs en Europe et à 1.67 millions de francs dans le reste du monde. Quel est le « poids » de vos ventes dans ces trois parties du monde ?

```
[] [CLEAR]
3.92 [ENTER] 2.36 [+] 1.67 [+]                7.95
3.92 [%Σ]                U.S.A.              49.31
2.36 [%Σ]                Europe              29.69
1.67 [%Σ]                Reste du monde      21.01
```

```
'N' 'E'  taux Nominal <-> taux Effectif (actuariel)
```

Dans l'une et l'autre routine, il faut entrer le taux donné au niveau 1, puis les diverses valeurs du « nombre de périodes ». Appuyer sur [E] (ou [N]). Le taux continu calculé est affiché en ligne 3, le détail de l'opération au niveau 2, le résultat au niveau 1. Le taux donné (facteur constant) est au Top niveau.

Exemple (*HP-18C Business Consultant*, page 78) :

Conversion d'un taux nominal en taux effectif.

Vous envisagez d'ouvrir un compte d'épargne. Laquelle des trois banques ci-dessous offre le taux d'intérêt le plus avantageux? (présenter les résultats avec trois décimales)

Banque 1 : 6.70% d'intérêt nominal annuel composé par trimestre

Banque 2 : 6.65% d'intérêt nominal annuel composé par mois

Banque 3 : 6.65% d'intérêt nominal annuel composé par jour

```
6.7 [ENTER] 4 [E] taux continu 3:          6.930
                               2:       "6.7N,4"
                               1:          6.870
[] [CLEAR]
6.65 [ENTER] 12 [E]    continu 3:          6.876
                               2:      "6.65N,12"
                               1:          6.856

             365 [E]   continu 3:          6.876
                               2:     "6.65N,365"
                               1:          6.875
```

Le taux le plus favorable est celui de la banque 3 qui offre un taux d'intérêt nominal annuel de 6.65% à composition journalière.

DEMANDEZ LE PROGRAMME !

Premier menu

Le facteur constant est le deuxième paramètre nommé.

X→ (Puissance)
Niveau 1 (X) à la 'puissance facteur constant'

```
« H SWAP ^ "^ " J »
```

? : 'Etablir' comme facteur constant le nombre affiché au niveau 1. la routine vide la pile et établit le facteur constant au Top niveau (1)

```
« → z « CLEAR z » »
```

A : (Addition) Niveau 1 + Facteur constant T

```
« H   + "+ " J »
```

S : (Soustraction) Niveau 1 - Facteur constant T

```
« H SWAP - "- " J »
```

D : (Division) Niveau 1 / Facteur Constant T

```
« H SWAP / "/ " J »
```

M : (Multiplication) Niveau 1 * Facteur constant T

```
« H * "* " J »
```

Deuxième menu

Le facteur constant est le premier paramètre nommé.

I% : (Pourcentage à ajouter à la base)

```
«
  DEPTH
  IF
    2 >=
  THEN
    H % DUP2 +
    ROT ->STR "+" + ROT ->STR +
    (le cas échéant, PR1)
    SWAP
    (le cas échéant, PR1)
  ELSE
    Y DUP
    INV .01 - DUP INV   Marque (MarkUp on Cost)
    SWAP .02 + INV      Marge  (MarkUp on Price)
  END
»
```

?? : (voir la routine '?')

```
« ? »
```

μ : MarkUp on Cost = Marque appelée *Marge appliquée au coût* sur HP-18C. MarkUp on Price = Marge appelée *Marge appliquée au prix* sur HP-18C

```
«
  STD H               Diminuer le nombre d'objets.
  SWAP →STR "-CP-" +  Préparer affichage niveau 2
  SWAP →STR +         de l'opération effectuée:
                          C: MU sur Cost
  1 FIX                   P: MU sur Price
  DEPTH PICK          Placer aux niveaux 1 et 2,
  ROT DUP2            3 et 4, valeurs Price & Cost.
  %CH                 MU sur Cost  = Marque
  SWAP ROT
  %CH ABS             MU sur Price = Marge
  2 →LIST             Affichage au niveau 1.
»
```

R : (Soustraction hp) Reste de (Facteur constant T - Niveau 1)

```
« H - " -" K »
```

Q : (Division hp) Quotient de (Facteur constant T / Niveau 1)

```
« H / " /" K »
```

→X : Facteur constant T 'Puissance Niveau 1 (X)'

```
« H ^ " ^" K »
```

Troisième menu: Portefeuille Taux nominal <-> Taux effectif

```
   (Portefeuille; pourcentage du total - en % -
                  de chaque poste)
                    Entrer un poste sous la forme:
«                   - Nb d'articles, Prix unitaire.
  L  ROLL             ou bien:
  1 +               - Montant du poste
  L ROLLD             ( = Nb * Prix )
  DEPTH DUP PICK -
  2 MOD             Test de détection de la forme
  IF                de l'objet qui a été entré:
    0 <>
  THEN              - soit, multiplier:
    *                 Nb d'articles * Prix unitaire
   END                → montant du poste
  DUP               - soit, garder le montant, et,
                      dans les deux cas, le copier
                      au niveau 2.
  DEPTH ROLL +      Mettre au niveau 1 le total
                    et y ajouter montant du poste;
  DEPTH ROLLD       mettre en haut de pile le
  DUP               nouveau total; dupliquer le
                    montant du poste.
  IF                Test de détection de zéro:
    0 <>            - soit le montant du poste est
  THEN                différent de zéro: l'afficher
    HALT              et arrêter le programme
  ELSE              - soit le montant du poste
    DROP              égale zéro: l'éliminer.
  END
  G                 Lancer la procédure de calcul
»                   du "pourcentage (%) du total".
```

■ : (Initialisation de 'Σ' ; "Corbeille à papier")

```
«
  DUP                  Test:
  IF 0 <>              - si un nombre # a été entré
                         ( numéro du niveau où se
  THEN                   trouve le poste que l'on
                         veut ôter du portefeuille),
    L ROLL               placer au niveau 1 le compteur
                         de nombre d'articles,
    1 -                  le décrémenter d'une unité.
    L ROLLD              Placer au niveau 1 le montant
    ROLL                 du poste à supprimer,
    NEG                  en changer le signe.
    DEPTH ROLL +         Placer au niveau 1 le total;
                         par addition, nouveau total.
    DEPTH ROLLD          Le placer en haut de pile.
  ELSE                   - si 0 a été entré, vider pile
    CLEAR                a) initialisation à 0 du
    0                       montant total
    0                    b) initialisation à 0 du
                            compteur de nb d'articles
  END
  KILL                   Oter de l'affichage le témoin
                         de "programme suspendu".
»
```

C : Transformer, le cas échéant, une pile :

```
«                      'Total du PF, nb d'articles,
  1 DEPTH                valeur n, n-1,..., 2, 1'
  2 -                          en une pile:
  FOR x                'Total du PF, nb d'articles,
    x ROLL               valeur 1, 2,..., n-1, n'.
  NEXT
»
```

% : Pourcentage (en %) du Total

```
«                      Deux cas d'utilisation:
  DEPTH                Au sein du programme ' ':
  IF                   à l'arrêt du prgm, si l'on
    3 >                veut le %  d'une valeur,
  THEN                 entrer le numéro du niveau
    PICK DUP           de cette valeur, presser [%]
    I                  pour affichage momentané du
    CLLCD              poste et de son pourcentage
    →STR " %" +        du total.
    3 DISP 2 DISP      Déclencher le calcul des
    2 WAIT             pourcentages du total et
    KILL               l'impression du portefeuille,
  END                  en entrant 0 [ ].


  Y                  - Fonction traditionnelle de
                       "pourcentage du total":
  I                    a) vider la pile
                       b) entrer le total
                       c) entrer le montant partiel
                       d) presser [% ] -> résultat
»                      e) autre partiel, puis c), d)
```

N : Conversion taux effectif en taux nominal

```
«
  3 FIX
  W "E,"               Préparer l'affichage des
  STD                  données au niveau 2.
  U
  1 DEPTH PICK %
  + LN * EXP           Placer le taux nominal
  Z *                  au niveau 1 et
  1 OVER V             le taux continu, niveau 3.
»
```

E : Conversion taux nominal en taux effectif

```
«
  STD
  W "N,"               Préparer l'affichage des
  U                    au niveau 2.
  DEPTH PICK           Taux nominal au niveau 1.
  % * 1 +
  SWAP ^ Z             Taux effectif
  1
  DEPTH PICK           Placer le taux effectif
  V                    au niveau 1 et
                       le taux continu
»                      au niveau 3.
```

G : Calcul de "% du total" utilisé dans 'Σ'

```
«
  L ROLL
  DROP
  2 L
  FOR x
    DUP I
    1 FIX                Nombre choisi de décimales.
    ->STR "%   " +       Constitution de la chaîne
    SWAP                 donnant, pour chaque poste,
    2 FIX                son "% " et
    ->STR +              son montant.

    L ROLLD              Ranger la chaîne sous le
  NEXT                   total du portefeuille.
  DEPTH ROLL ->STR       De ce total, former une
  DEPTH ROLLD            chaîne, la ranger en T.
  (le cas échéant,       Imprimer les résultats.
               PRSTC)
  KILL
»
```

H : Pour réduire le nombre d'objets dans la pile

```
« Y DUP2 »
```

I : Routine utilisée par '%Σ' et 'G'

```
« DEPTH PICK SWAP %T »
```

J : (J et K: routines utilisées par 'X→', '→X', 'A', 'S', 'D', 'M' et 'R', 'Q')

```
« ROT →STR SWAP + SWAP »
```

K :

```
« ROT →STR + SWAP »
```

L : (routine utilisée par 'Σ', '■', 'G', 'Y')

```
« DEPTH 1 - »
```

Nota : Si l'on souhaite afficher l'opération arithmétique qui vient d'être effectuée, on peut (après avoir libéré de la mémoire, en vidant de leur contenu, par exemple, les variables 'N', 'E' et 'Z', 'U', 'V', 'W') entrer les routines suivantes :

```
'J': « ROT →STR SWAP DEPTH PICK →STR X »
'K': « ROT →STR DEPTH PICK →STR ROT ROT X »
'X': « "=" + + + OVER →STR + SWAP »
```

Y : Pour réduire le nombre d'objets dans la pile

```
«
  DEPTH
  IF
    3 >=
  THEN
    L ROLLD
    DEPTH 2 -
    DROPN
  END
»
```

Z : (routine utilisée par 'N', 'E' et 'V')

```
« 1 - 100 * »
```

U : ('U', 'V' et 'W' sont utilisées par 'N' et 'E')

```
«
  + SWAP →STR +
  3 FIX
  SWAP DUP
  INV 1
»
```

V :

```
« % EXP Z ROT ROT »
```

W :

```
« Y DUP DEPTH PICK →STR »
```

X : Variable libre : on peut y stocker un objet, sans faire se décaler tous les menus ; par exemple, la routine décrite précédemment.

Philippe Heilbronn (233)

LISTE DES ADRESSES DU HP-28S

La liste suivante correspond à celle publiée dans *JPC 47* page 13. Elle donne les adresses des fonctions de la HP-28S précédées du numéro de la fonction.

00	DUP	75DC	01	DUP2	75F6
02	SWAP	7610	03	DROP	762A
04	DROP2	7644	05	ROT	76E5
06	OVER	7678	07	DEPTH	7692
08	DROPN	76B1	09	DUPN	76CB
0A	PICK	76E5	0B	ROLL	76FF
0C	ROLLD	7719	0D	CLEAR	7733
0E	→LIST	7752	0F	R→C	776C
10	RE	77A4	11	IM	77F0
12	SUB	7832	13	LIST→	7888
14	C→R	78C0	15	SIZE	78EE
16	POS	796C	17	→STR	79AE
18	STR→	79C8	19	NUM	79EC
1A	CHR	7A10	1B	TYPE	7A34
1C	→ARRY	7AE4	1D	ARRY→	7B85
1E	RDM	7BD6	1F	CON	7C5E
20	IDN	7D4A	21	TRN	7DE1
22	PUT	7E32	23	PUTI	7FB9
24	GET	814A	25	GETI	8277
26	SAME	87C3	27	AND	87EC
28	OR	886F	29	NOT	88F2
2A	XOR	8952	2B	==	89D5
2C	≠	8A53	2D	<	8AD6
2E	>	8B4A	2F	<=	8BBE
30	>=	8C32	31	=	8CA6
32	NEG	8D24	33	ABS	8D70
34	CONJ	8DBC	35	π	8E08
36	MAXR	8E36	37	MINR	8E64
38	e	8E92	39	i	8EC0
3A	+	8EEE	3B	-	8FD0
3C	*	9076	3D	/	9144
3E	^	91FE	3F	INV	92DB
40	ARG	9327	41	P→R	937D

42	R→R	9469	43	SIGN	94E2	B6	UTPF	ADC2	B7	UTPT	ADE6
44	√	9524	45	SQ	958E	B8	SCLΣ	AE0A	B9	DRWΣ	AE24
46	SIN	9602	47	COS	9644	BA	COLΣ	AE3E	BB	SINV	AE62
48	TAN	9686	49	SINH	96C8	BC	SNEG	AEF9	BD	SCONJ	AF90
4A	COSH	970A	4B	TANH	974C	BE	STO+	B013	BF	STO-	B118
4C	ASIN	978E	4D	ACOS	9802	C0	STO/	B222	C1	STO*	B34F
4E	ATAN	9876	4F	ASINH	98B8	C2	EXGET	B5C6	C3	EXSUB	B5EA
50	ACOSH	98FA	51	ATANH	9969	C4	OBSUB	B622	C5	OBGET	B646
52	EXP	99DD	53	LN	9A1F	C6	FORM	B66A	C7	COLCT	B6A2
54	LOG	9A89	55	ALOG	9AF3	C8	EXPAN	B6DA	C9	ISOL	B712
56	LNP1	9B35	57	EXPM	9B6D	CA	QUAD	B736	CB	SHOW	B75A
58	FACT	9BA5	59	IP	9BDD	CC	TAYLR	B788	CD	∂	B7B6
5A	FP	9C15	5B	FLOOR	9C4D	CE	RCEQ	B84D	CF	STEQ	B871
5C	CEIL	9C85	5D	XPON	9CBD	D0	ROOT	B88B	D1	∫	B8E1
5E	MAX	9CF5	5F	MIN	9D41	D2	ASR	BB53	D3	RL	BB77
60	MOD	9D8D	61	MANT	9DD9	D4	RLB	BB9B	D5	RR	BBBF
62	D→R	9E11	63	R→D	9E49	D6	RRB	BBE3	D7	SL	BC07
64	→HMS	9E77	65	HMS→	9E9B	D8	SLB	BC2B	D9	SR	BC4F
66	HMS+	9EBF	67	HMS-	9EE3	DA	SRB	BC73	DB	R→B	BC97
68	RNRM	9F07	69	CNRM	9F2B	DC	B→R	BCBB	DD	CONVERT	BCDF
6A	DET	9F4F	6B	DOT	9F73	DE	INDEP	8D21	DF	PMIN	8D45
6C	CROSS	9F97	6D	RSD	9FBB	E0	PMAX	BD69	E1	AXES	BD8D
6E	%	9FE9	6F	%T	A035	E2	CENTR	BDB1	E3	RES	BDD5
70	%CH	A081	71	RAND	A0C3	E4	*H	BDF9	E5	*W	BE1D
72	RDZ	A0DD	73	COMB	A101	E6	DRAW	BE41	E7	PIXEL	BE5B
74	PERM	A125	75	LCD→	A149	E8	DRAX	BE7F	E9	PR1	BE99
76	→LCD	A163	77	DGTIZ	A187	EA	PRSTC	BEB3	EB	PRST	BECD
78	MENU	A1A1	79	RCL	A1CF	EC	CR	BEE7	ED	PRUSR	BF01
7A	STO	A220	7B	PURGE	A26C	EE	PRVAR	BF1B	EF	PRMD	BF5D
7C	MEM	A32B	7D	ORDER	A359	F0	PRLCD	BF77	F1	CRDIR	BF91
7E	CLUSR	A396	7F	KILL	A3B0	F2	PATH	BFB5	F3	HOME	BFCF
80	ABORT	A3CA	81	ERRN	A3E4	F4	VARS	BFE9			
82	ERRM	A3FE	83	EVAL	A418						
84	IFTE	A450	85	IFT	A500						
86	SYSEVAL	A54C	87	DISP	A5A6						
88	RND	A5D4	89	BEEP	A616						
8A	→NUM	A63A	8B	LAST	A654						
8C	WAIT	A73A	8D	CLLCD	A75E						
8E	KEY	A778	8F	CLMF	A792						
90	SF	A7B6	91	CF	A7F3						
92	FS?	A830	93	FC?	A86D						
94	DEG	A8AF	95	RAD	A8D3						
96	FIX	A8F7	97	SCI	A92F						
98	ENG	A967	99	STD	A99F						
9A	FS?C	A9B9	9B	FC?C	AA0A						
9C	BIN	AA60	9D	DEC	AA7A						
9E	HEX	AA94	9F	OCT	AAAE						
A0	STWS	AAC8	A1	RCWS	AAEC						
A2	RCLF	AB06	A3	STOF	AB20						
A4	STOΣ	ABBC	A5	CLΣ	ABD6						
A6	RCLΣ	ABF0	A7	Σ+	AC0A						
A8	Σ-	AC38	A9	NΣ	AC52						
AA	CORR	AC6C	AB	COV	AC86						
AC	MAXΣ	ACA0	AD	MEAN	ACBA						
AE	MINΣ	ACD4	AF	SDEV	ACEE						
B0	TOT	AD08	B1	VAR	AD22						
B2	LR	AD3C	B3	PREDV	AD56						
B4	UTPC	AD74	B5	UTPN	AD9E						

Les suivantes sont dans l'ordre mais les numéros recommencent vraisemblablement à 00.

IF	E3E2		THEN(IF)	E3F8	
ELSE	E459		END(IF)	E479	
«	E9D0		»	E9E6	
WHILE	E4D2		REPEAT	E4F2	
END(WHILE)	EA2D		DO	E535	
UNTIL	E555		END(DO)	EA4D	
START	E56B		FOR	E5B3	
NEXT	E60F		STEP	E37E	
IFERR	E77A		THEN(IFERR)	EA90	
HALT	E842				

Sébastien Lalande (442)

GRAND JEU POUR TOUTE HP-28

Et voici enfin pour JPC et uniquement pour lui un jeu fantastique. Le but : très simple, trouver à quoi sert le programme ; ou plutôt, ce qu'il fait. Cette première version n'est pas très compliquée de manière à ce que

tout le monde ait ses chances. Mais il n'en reste pas moins fort amusant pour ceux qui pourront l'apprécier et apprendre à s'en servir.

Les règles : tous les coups sont permis. La meilleure façon de trouver est de le taper, de l'assembler avec ASS et →LEX pour HP-28C (1BB et 1CC) ou ASSEMBLEUR pour HP-28S, de l'exécuter, et enfin d'essayer plusieurs combinaisons.

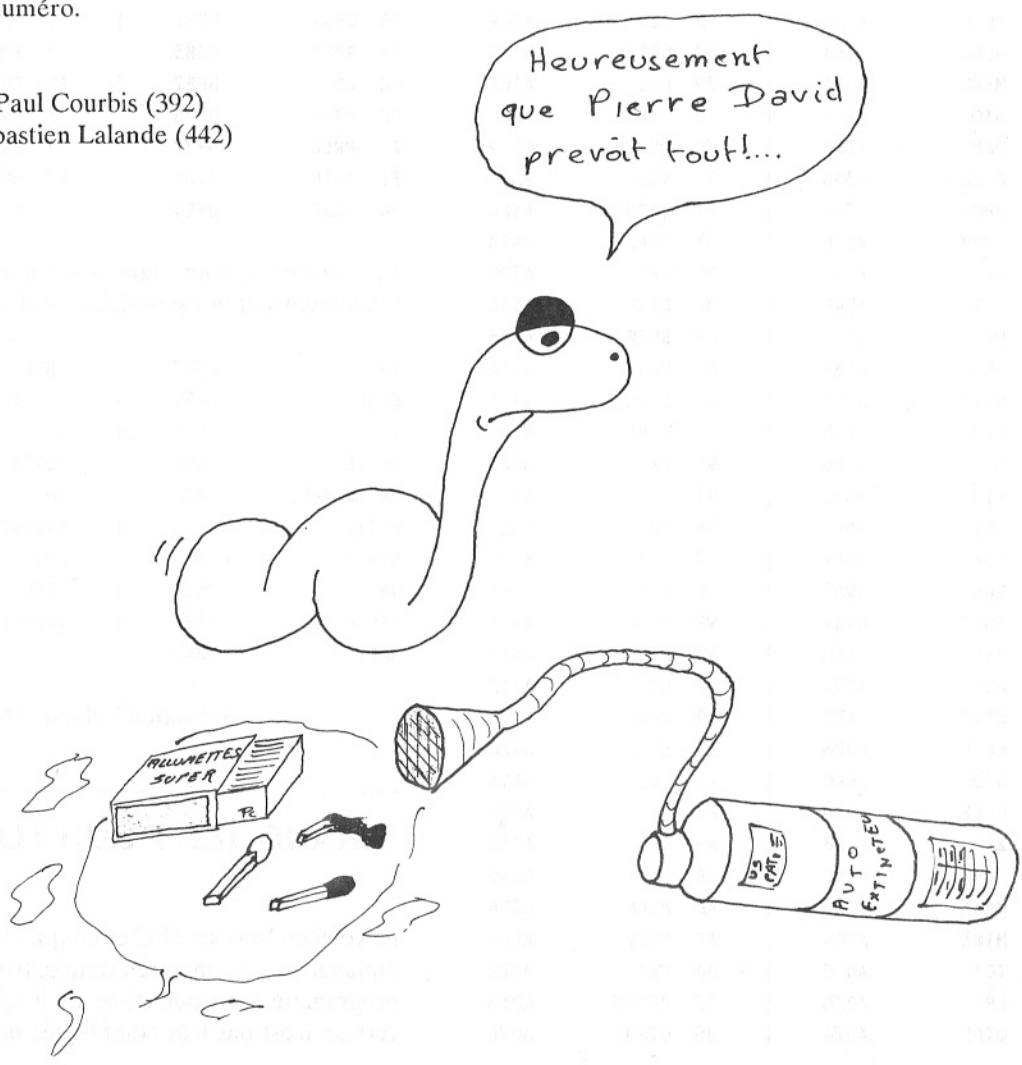
Attention, ce jeu n'est pas un bête Memory Lost et vos programmes ne risquent absolument rien, tant que vous le tapez correctement. Que les mauvaises langues qui parlent toujours avant d'avoir essayé se taisent. Si ce programme a été mis sous forme de jeu c'est uniquement pour vous faire refléchir sur les choses passionnantes qu'il met en oeuvre. Je n'en dirais pas plus long, et...

Bonne chance !

```
"69C2053000808F1331F410F430015
D01F4100C30015D0131142164808C"
```

Envoyez vos réponses et suggestions au club. Résultat dans le prochain numéro.

Paul Courbis (392)
Sébastien Lalande (442)



HP75

J.Y. Hervé

M41LEX

16

M41LEX

M41LEX est un Lex que j'ai développé peu après MADLEX dans un but professionnel : l'utilisation par le HP-75 de données stockées dans des fichiers de type DATA créés par la HP-41. Je pense qu'il devrait également fonctionner avec les fichiers SDATA du HP-71, ce qui devrait constituer un pont supplémentaire entre HP-71 et HP-75.

Ce Lex reprend trois fonctions de MADLEX : `MSEEKR("Nom:dv")`, `MSEEKL("Nom:dv")` et `MREAD$(":dv",R)`. Contrairement à ce qui se passe dans MADLEX, cette dernière fonction ne lit qu'un enregistrement complet de 256 octets à la fois (ne pas oublier de dimensionner la variable en conséquence). Une fonction complémentaire particulière, `REG41$(A$)`, permet de convertir les caractères de la chaîne lue sur le support magnétique dans le format de la HP-75 (NdlR : mot clé initialement publié avec R41LEX).

Voici donc une description des fonctions du Lex, avec une courte démonstration de lecture de fichier DATA de 41 et l'indispensable page de codes hexadécimaux.

MSEEKR

Syntaxe :

```
MSEEKR ("Nom:dv")
```

Cette fonction fournit le numéro *R* de l'enregistrement correspondant à l'adresse physique de début du fichier *Nom* sur le support *:dv*.

Messages d'erreur :

Erreur 62 : `File not found`
Erreur 63 : `Invalid filespec`

MSEEKL

Syntaxe :

```
MSEEKL ("Nom:dv")
```

Cette fonction fournit la longueur *L* en nombre d'enregistrements du fichier *Nom* sur le support de masse *:dv*. Le dernier enregistrement du fichier *Nom* sera donc à l'adresse *R*+*L*-1.

Messages d'erreur :

Erreur 62 : `File not found`
Erreur 63 : `Invalid filespec`

MREAD$

Syntaxe :

```
MREAD$ (":dv",R)
```

Lecture des 256 octets de l'enregistrement *R* sur le support de masse *:dv*. Pour le lecteur de cassettes, la valeur de *R* varie entre 0 et 511 (0 à 2463 pour la disquette double face).

La fonction retourne une chaîne de 256 caractères correspondant aux octets lus sur le support.

Messages d'erreur :

Erreur 63 : `Invalid filespec`
Erreur 64 : `Mass mem error`

REG41$

Syntaxe :

```
REG41$ (A$)
```

Cette fonction convertit les huit premiers caractères de la chaîne *A$* en une chaîne de 0 à 16 caractères correspondant à la valeur numérique ou à la chaîne alphanumérique stockée par la HP-41 dans le registre correspondant du fichier de type DA (DATA).

L'erreur 27 `Invalid subscript` est générée si la chaîne *A$* comporte moins de huit caractères.

Pour un nombre, la chaîne retournée qui comportera de 1 à 16 caractères pourra être interprétée par la fonction `VAL`.

Pour une chaîne alphanumérique, la chaîne retournée comportera de 0 à 6 caractères.

Attention : `REG41$` peut convertir les premiers octets de toute chaîne, mais la conversion qu'elle effectue n'est valable que pour des données stockées sous forme registre, c'est-à-dire par groupe de huit octets dans des fichiers de type DATA41 ou SDATA71.

Et voici les exemples promis :

```
R = MSEEKR ("TEST:M1") ---> R = 30
```

Le fichier TEST du support :M1 débute à l'enregistrement 30.

```
L = MSEEKL ("TEST:M1") ---> L = 3
```

Le fichier TEST comporte trois enregistrements. Il se terminera donc à l'enregistrement $R+L-1 = 32$.

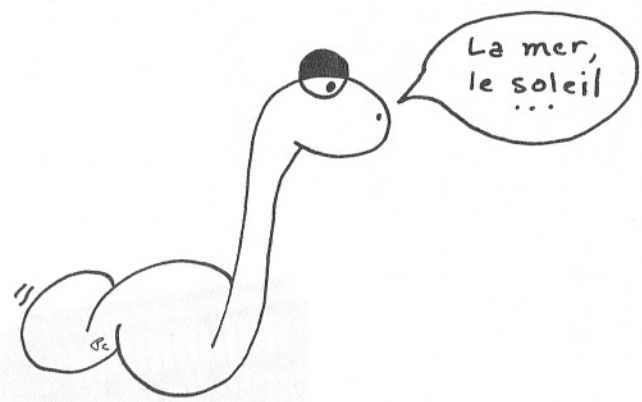
```
D$ = MREAD$ (":M1",30)
```
Lecture du premier enregistrement du fichier TEST.

```
N$ = REG41$ (D$[9]) ---> N$ = "3.14159265"
```
Lecture et traduction du deuxième registre de données du fichier TEST.

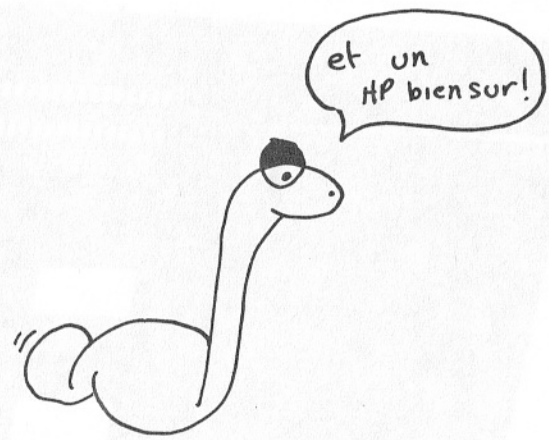
```
C$ = REG41$ (D$[17]) ---> C$ = "PI"
```
Lecture et traduction du troisième registre de données du fichier TEST.

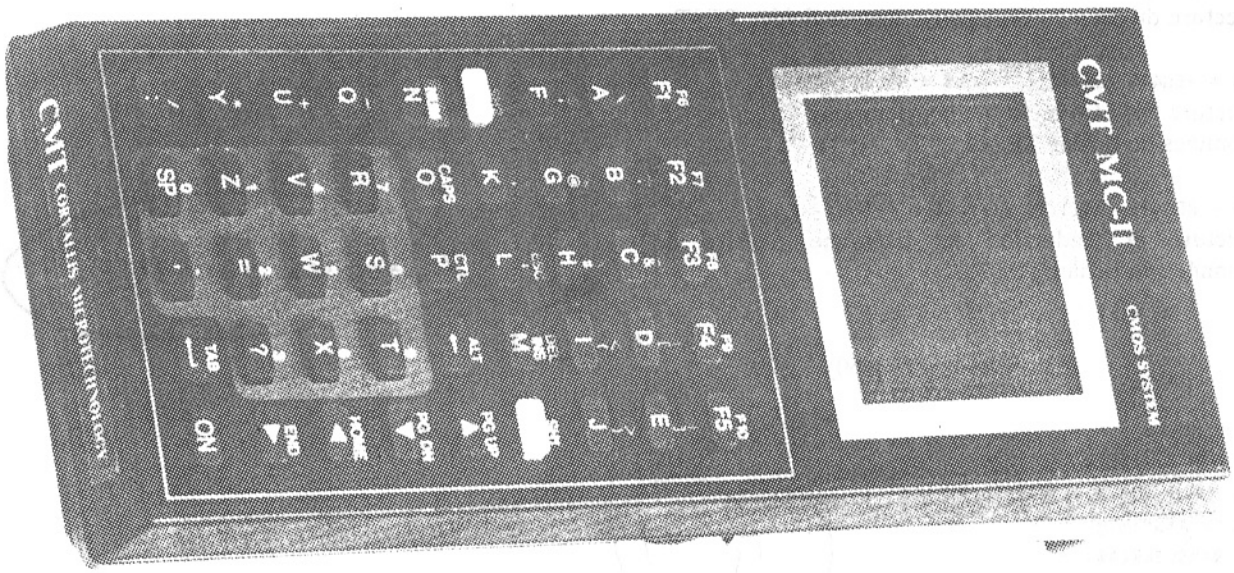
Jean-Yves Hervé (450)

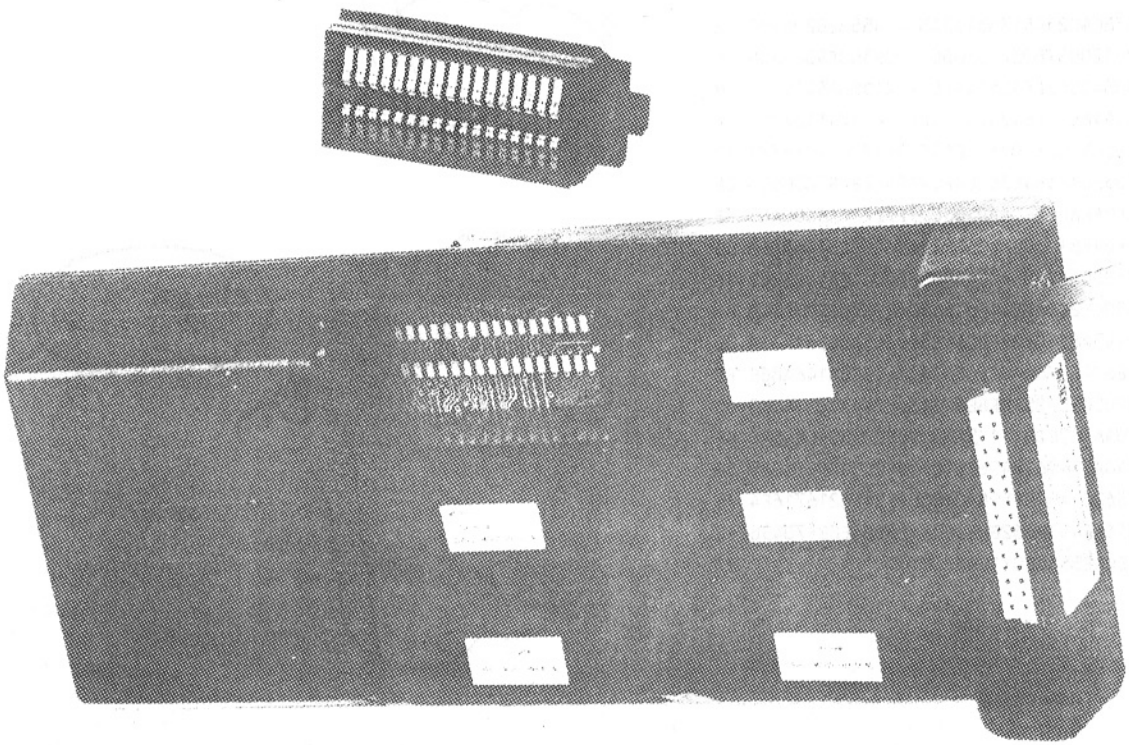
```
10 DIM A$[256]
20 INPUT "DATA41: Fichier? ";A$
30 A$=A$&":M1"
40 R=MSEEKR(A$)
50 L=MSEEKL(A$)+R-1
60 FOR J=R TO L
70 A$=MREAD$(":M1",J)
80 FOR I=1 TO 255 STEP 8
90 PRINT REG41$(A$[I]);" ";
100 NEXT I @ NEXT J @ PRINT
```




```
M41LEX    351 octets

Line ---------------- Code ------------- Check

001 92954D018D4C2A4553A34D34314C455820206B00 FE
002 0A001E00120037003A003D004E00B300F9003A00 1F
003 3A003A003A00FFFF4D5345454BD24D5345454BCC 3B
004 4D52454144A45245473431A4FFFF61179E182D54 A8
005 B1A38214C65F00640AE35EE3F00F182D54B1A382 18
006 14C65F005E0AE364E3CEE4FC9E9842B0B38306E4 CB
007 43A8FFCEDFEACE774AF8306C91F72234A3600AE5 7F
008 CE61460F6AF81C600AE3CE61461468CE6146C36E EE
009 621EB526000AE5F00ACEA44C3FF004CEA44C3942 76
010 B0B3834306E2CEDFEACE6146AD6D9E282E9842B0 BF
011 B38306E443A8FFCEDFEACE8B3EB36584CE61463A 8E
012 68F8CACE61460F6AF8C965B1648492CE6146A86B FB
013 F8BD6EA90001CE25FCF8B826A36E24A3CE6146F3 DC
014 6BF8A8F0AA182ECE111F5AE3C90800F504CEAC4C BF
015 1B601CE10AE599618498C810F704CE8BFC9E6EA9 64
016 060060E36221A099859820A0CE1F24F8216716E4 75
017 66E465E464E463E462E4E360A4F6086E8BF70456 A2
018 89F0F46E0AE556CE071F9E                   B7
019 A243
```




CMT MC-II
CMOS SYSTEM
CMT COMPATIBLE MICROTECHNOLOGY



FORTH

ASSEMBLEUR

BASIC

LE COIN DES LHEX

INITIATION A LISP

Le langage Lisp déroute souvent par la grande prolifération de ses parenthèses et sa rigueur. Ce langage, qualifié de « langage d'un autre type » par Christian Queinnec, est toutefois d'une puissance que l'on soupçonne peu au premier abord. Je vais aujourd'hui essayer de présenter des rudiments de Lisp, tout en décrivant le *Lisp71* que j'ai écrit avec peut-être trop de rigueur.

PRESENTATION

Lisp est un langage qui nous dispense de toutes les tâches habituelles en informatique, qui sont fastidieuses, comme la gestion de la mémoire. Oserais-je dire que cette gestion est automatique ? Je dirai plutôt qu'elle ne se fait pas, mais que cela n'empêche pas les programmes de tourner.

Lorsque l'on rencontre un problème, en Lisp, on peut directement passer à sa résolution, alors qu'un autre langage nous obligerait à déclarer des structures de stockage, etc. Ceci a pour conséquence la puissance de Lisp en intelligence artificielle, mais aussi sa faiblesse dans tous les problèmes de gestion de données.

Une telle puissance, dans l'utilisation de Lisp, s'accompagne d'une rigueur, que je n'ai rencontrée nulle part ailleurs en informatique (même en Forth). Elle repose sur des règles strictes. Lisp est plus qu'un langage informatique, c'est un système formel, au sens mathématique.

LES OBJETS LISP

Lisp ne manipule pas de simples nombres, adresses, ou chaînes de caractères : il manipule des objets. Dans le noyau Lisp, il y a déjà des objets prédéfinis : ce sont les objets primitifs. On peut, à partir de ceux-ci, définir par récurrence d'autres objets. Ainsi, si A_1, A_2, ... A_n sont des objets, l'assemblage "(A_1 A_2 ... A_3)" est un objet.

A partir de cela, on peut distinguer deux types d'objets : les atomes et les listes. La plupart des Lisps généralisent la notion de liste en *s-expressions* (abréviation courante de *expressions symboliques*), ou *paires pointées*, mais j'en parlerai peu ici, ce type d'objet étant absent dans le Lisp71. Ce que j'ai appelé *objet primitif* est un exemple d'atome, et lorsque A_1 ... A_n sont des atomes ou des listes (et non des s-expressions), "(A_1 ... A_n)" est une liste.

Une liste est associée à un *car* et un *cdr*. Intuitivement, une liste, au sens de Lisp, est une liste, au sens familier, d'objets qui sont des atomes ou des listes. Le car représente alors le premier objet de la liste, et le cdr représente la liste obtenue en enlevant ce premier objet. Par exemple, si A_1 ... A_n sont des atomes ou des listes, le car de la liste (A_1 ... A_n) est A_1, et son cdr est (A_2 ... A_n).

Par convention, une *liste vide* n'est pas une liste, mais un atome. On le représente par "()", ou par son nom : "NIL". Ceci montre combien la description intuitive des listes est mal adaptée à celle des listes de Lisp. Je donnerai une description mieux adaptée plus loin.

Parmi les atomes, on distingue les identificateurs et les nombres. Dans le Lisp71, les nombres sont des entiers relatifs de 16 bits représentés dans la base courante du mode Forth. On a un autre type d'atome : le message d'erreur, bien utile pour l'évaluation.

Les identificateurs peuvent représenter n'importe quoi. Dans le Lisp71, un identificateur est un nom (une chaîne ASCII), une définition (un objet), et un type (un objet également). Il y a cependant une exception : les identificateurs primitifs sont des mots Forth.

STRUCTURE DES OBJETS DE LISP71

Un objet est représenté par un mot de 20 bits, dont les quatre bits de poids fort décrivent le type d'objet. Sauf pour les entiers et les erreurs, ce mot représente un type et une adresse *adr*.

Pour un identificateur primitif, *adr* est le CFA d'un mot Forth.

Pour un identificateur non primitif, *adr* pointe ce que j'appelle une *donnée moyenne*, qui contient deux mots (représentant le type et la définition de l'atome), et un chaîne ASCII.

Pour une s-expression, *adr* pointe une *donnée courte*, qui contient deux mots : le car et le cdr. Pour une s-expression, le car et le cdr sont des objets quelconques, alors que pour un liste, le car est un atome ou une liste, et le cdr est NIL ou une liste. Une s-expression de car A et de cdr B se représente par "(A.B)".

Ainsi, si A et B sont des atomes ou des listes, on a :

(A.NIL) = (A)

AH ! VOUS ECRIVEZ

Vous vous sentez en verve, mais vous ne savez pas sous quelle forme "l'équipe de rédaction" souhaite recevoir votre prose. C'est ici que se trouvent les réponses à vos questions.

Dans la mesure du possible, vous devez nous envoyer vos écrits sur support magnétique (carte, cassette ou disquette). Soyez sans crainte, nous vous retournerons vos biens après copie.

Si vous ne pouvez pas utiliser de support magnétique, ou ne pouvez vous rendre aux réunions, alors et alors seulement faites le sur papier.

Que ce soit sur une feuille de papier, ou sur support magnétique, ne dépassez pas 50 caractères par ligne.

Pour nous épargner du travail, insérez dans votre texte les commandes de formattage suivantes (et non les commandes du formatteur HP) :

"^" centre un titre, par exemple :
^TITRE

"\" (CHR$(92)) marque le début et la fin d'un paragraphe. Par exemple :

\Début de paragraphe exprimant le contenu de vos idées qui, même si vous en doutez, intéressera certains des membres du Club. Surtout si vous vous sentez débutant. Les articles pour débutants écrits par des débutants sont ceux qui manquent le plus. Fin de paragraphe.\

N'oubliez pas de mettre les accents. Utilisez le jeu de caractères Roman8. Les possesseurs de HP71 utiliseront les redéfinitions de touches ci-dessous, ainsi que le fichier CHARLEX listé dans le coin des Lhex.

Jean-Jacques Dhénin (177)

```
DEF KEY 'fW', CHR$(197);      (é)
DEF KEY 'fE', CHR$(193);      (ê)
DEF KEY 'fR', CHR$(201);      (è)
DEF KEY 'fY', CHR$(203);      (ù)
DEF KEY 'fU', CHR$(195);      (û)
DEF KEY 'fI', CHR$(209);      (î)
DEF KEY 'fO', CHR$(194);      (ô)
DEF KEY 'f/', CHR$(92);       (\)
DEF KEY 'fA', CHR$(192);      (â)
DEF KEY 'fS', CHR$(200);      (à)
DEF KEY 'fD', CHR$(205);      (ë)
DEF KEY 'fJ', CHR$(207);      (ü)
DEF KEY 'fK', CHR$(221);      (ï)
DEF KEY 'f*', CHR$(124);      (|)
DEF KEY 'fC', CHR$(181);      (ç)
```

PPC PARIS SE REUNIT
UNE FOIS PAR MOIS

Comme vous le savez peut être déjà, PPC Paris se réunit une fois par mois, en plein coeur de Paris. Amenez votre matériel, votre bonne volonté et vos idées ! Plus vous en apporterez, et plus vous en trouverez chez vos collègues de PPC.

Ces réunions se déroulent de manière très libre, aucun ordre du jour, discussion ou autre n'étant imposé. Un membre du bureau est toujours présent. Ainsi, si vous désirez remettre votre article tout frais au Journal, si vous avez des suggestions à faire, si vous voulez vous procurer des anciens numéros de JPC, ce sera en principe toujours possible.

Si donc cela vous intéresse, n'hésitez plus un seul instant, venez nous rejoindre tous les premiers samedis de chaque mois (sauf en période de vacances scolaires) au :

Centre de Jeunesse et de Loisirs Jean Verdier
11 rue de Lancry
75010 Paris

et en montant au deuxième étage, vous entendrez des éclats de rire et des discussions passionnées vers la salle 215. Attention, toutefois, de venir entre 16 et 19h.

Pour l'accès en métro, trois possibilités s'offrent à vous :
- Métro Strasbourg Saint Denis :
Sortie porte St Martin / Bd St Denis, coté pairs
- Métro République :
Sortie Bd St Martin, coté pairs
- Métro Jacques Bonsergent :
Sortie Bd Magenta, coté impairs.

Ah, j'oubliais ! JPC est (souvent) distribué en avant première lors de ces réunions... A bon entendeur, salut !

Les dates des prochaines réunions sont :
Samedi 7 mai 1988
Samedi 4 juin 1988

Pierre David (37)

NOUS EN AVONS

La coopérative du Club vous propose :

- de **lecteurs de cartes** magnétiques pour HP-71, neufs, dans leur boîte d'origine, avec 5 cartes magnétiques, pour 500 F (port compris),
- des **anciens numéros** de JPC, au prix de 40 F + 7,40 F de frais d'affranchissement,
- d'une **année complète** de numéros de JPC (février à janvier) pour 300 F (offre spéciale) port compris,
- des **I.D.S.** du module Forth / Assembleur (listing interne commenté par HP) pour 250 F (port compris),
- des **VASM** pour HP-41 (listings des Roms internes commenté par HP) pour 300 F (port compris),
- de **manuels de service** du HP-41 au prix de 75 F (port compris),
- de **manuels de service** du HP-75 au prix de 75 F (port compris).

En outre, le module **JPC Rom** pour HP-71 est disponible. Vous nous adressez votre Eprom CMT (de préférence 64 Ko), et nous la programmons suivant une des options ci-dessous :
- JPC Rom + Manuel, pour 600 F,
- JPC Rom + Manuel + vos propres programmes, pour 800 F.

Si vous souhaitez des renseignements complémentaires, n'hésitez pas à nous contacter.

VOUS EN VOULEZ

Nom :
Prénom :
No de membre :
Adresse :

Commande :

	Qté		Prix Unitaire		Prix Total
lecteur de cartes pour HP-71		x	500	FF	
anciens numéros de JPC		x	47,40	FF	
année complète de JPC		x	300	FF	
I.D.S. du module Forth		x	250	FF	
V.A.S.M.		x	300	FF	
Manuel de service pour HP-41		x	75	FF	
Manuel de service pour HP-75		x	75	FF	
JPC Rom + Manuel		x	600	FF	
JPC Rom + Manuel + vos propres programmes		x	800	FF	
			Total		FF

Préciser éventuellement les
numéros de JPC commandés :

PPC PARIS

Association régie par la loi de 1901, enregistrée
à Paris le 2 décembre 1982 sous le numéro 82/3240

BULLETIN D'ADHESION

Nom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
Prénom |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_| Date de naissance |_|_|/|_|_|/|_|_|
Adresse |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
Code Postal |_|_|_|_|_| Ville |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
Téléphone Domicile |_|_|_|_|_|_|_|_|_|_|_|_|_|_| Bureau |_|_|_|_|_|_|_|_|_|_|_|_|_|_|

Profession ____________________
Intérêts ____________________
____
Matériel HP en votre possession ____________________
____
Autre matériel informatique ____________________
____

Comment avez-vous connu PPC Paris ? ____________________
____

Que recherchez-vous au sein de PPC Paris ? ____________________
____
____

La Loi No 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés, garantit à toute personne justifiant de son identité un droit d'accès et de rectification auprès des services ou organismes chargés de mettre en oeuvre des traitements informatiques comportant des informations nominatives le concernant.

Je souhaite adhérer au club PPC Paris conformément aux statuts de l'Association. Au mieux de ma connaissance, je déclare avoir le droit de fournir tous les programmes et informations que je vous enverrai (sans enfreindre des obligations de secret à l'égard d'autres personnes ou organismes) pour publication dans le Journal de liaison, sans obligations ni responsabilité d'aucune sorte (en cas d'utilisation frauduleuse) de la part des dirigeants de PPC-Paris.

Date |_|_|/|_|_|/19|_|_|
Signature, précédée de la mention "Lu et Approuvé" ____________________

Le montant de la cotisation s'élève à 350.00 F pour un an.
Etudiants: 300.00 F (justificatif indispensable)

Paiement effectué le |_|_|/|_|_|/19|_|_| à l'ordre de PPC Paris,
par [ ] chèque bancaire, [ ] chèque postal ou [ ] mandat lettre

Veuillez envoyer toute correspondance à :
PPC Paris, BP 604, 75028 Paris Cedex 01, France

(A.(B.NIL)) = (A B)

((A.NIL).(B.NIL)) = ((A) B)

EVALUATION

Je vais énoncer les règles fondamentales d'évaluation. J'espère ne pas me tromper, ceci étant bien loin d'être évident.

Le résultat de l'évaluation d'un atome est sa définition. S'il n'en a pas (cas des atomes primitifs), le résultat est l'atome lui-même. Pour évaluer une s-expression (A.B), on *évalue A sur la liste d'arguments B* :

- si A est un entier ou une erreur, et B n'est pas NIL, le résultat est un message d'erreur, mais si B est NIL, le résultat est A ;

- si A est un identificateur primitif, le mot Forth est exécuté sur les arguments B, et retourne le résultat de l'évaluation ;

- si A n'est pas un atome primitif, on évalue (AA.BB). Si A est une liste AA est son car, et BB son cdr. Si A est un atome, AA est son type, et BB sa définition. Si, après cette évaluation, B n'est pas NIL, le résultat sera une erreur, sinon, ce sera le résultat de l'évaluation de (AA.BB).

Prenons un exemple. CDR est un identificateur primitif. Pour évaluer (CDR (A1 A2 ... An)) qui sous la forme de s-expression, est (CDR ((A1 A2 ... An).NIL)), donc A=CDR et B=((A1 A2 ... An).NIL), et CDR étant un identificateur primitif, le mot Forth CDRp est exécuté sur ((A1 ... An).NIL).

L'EDITEUR LISP

L'éditeur demande une chaîne de caractères devant représenter une liste ou un atome, avec les conventions usuelles : les listes commencent par "(", comportent une liste (au sens familier) d'objets séparés par des espaces, et finissent par ")" ; et les atomes sont représentés par leur nom.

Cette chaîne étant entrée, et validée, l'éditeur évalue la chaîne, et retourne le résultat. C'est tout !

Pour se lancer dans les exemples, il est temps de parler de la primitive QUOTE : QUOTE demande un argument B, et retourne B ! A quoi peut bien donc servir un programme aussi bête ? Voici un exemple : supposons que nous voulions déterminer le car de la liste "(+ 1 2 3)". Si l'on tapait (CAR (+ 1 2 3)), la primitive CAR évaluerait l'argument (+ 1 2 3), qui vaut 6, comme nous le verrons, et calculerait le car de l'atome 6, qui se solderait par une erreur, puisque 6 n'est pas une s-expression. Si l'on tape (CAR (QUOTE (+ 1 2 3))), CAR évalue (QUOTE (+ 1 2 3)) qui donne (+ 1 2 3), et calcule son car, qui est "+" : (CAR (QUOTE (+ 1 2 3))) donne +.

Notons tout de suite une convention d'écriture : QUOTE revenant souvent, au lieu d'écrire (QUOTE objet), on écrit 'objet. Ainsi, (CAR '(+ 1 2 3)) donne +. Autre exemple: (CDR '(+ 1 2 3)) donne (1 2 3). Nous pouvons maintenant décrire les primitives.

PRIMITIVES

CAR et CDR opèrent sur un seul argument. Ces primitives évaluent cet argument et retournent son car ou son cdr.

CONS opère sur deux arguments, qui sont évalués. CONS retourne une liste formée à partir de ces arguments. Par exemple, (CONS '(1 2 3) '(4 5 6)) donne ((1 2 3) 4 5 6). (CONS 1 NIL) donne (1). (CONS (CAR (CDR '(1 2 3))) (CDR '(1 2 3))) donne (2 2 3). Note : CONSp ne marche pas si l'on enchaîne des CONS (bug à corriger).

QUOTE opère sur un seul argument qu'il n'évalue pas, et qu'il retourne.

EVAL opère sur un argument qu'il évalue deux fois et qu'il retourne.

+, -, *, et / opèrent sur au moins un entier et retournent le résultat de l'opération. Par exemple, (+ 1 2 3) donne 6. (* (+ 2 3) 4 (- 10 2)) donne 160. (/ 5 2 0) donne "/zero" (le résultat de l'évaluation est un message d'erreur). Si l'un des arguments est un message d'erreur, le résultat est ce message. Si l'un des arguments n'est pas représentable par un entier, le résultat est un message d'erreur.

PLUSP, MINUSP, ZEROP sont des prédicats opérant sur un entier. Ils testent si cet entier est positif, négatif, ou nul. Comme tous les prédicats, leur résultat est NIL pour faux, et T pour vrai.

<, <, =, et /= sont des prédicats opérant sur une liste d'entiers. Ils testent si la liste est croissante, décroissante, si les entiers sont tous égaux, ou tous différents.

ATOM et NUMBERP sont des prédicats opérant sur un argument. Ils testent si cet argument est un atome, ou si c'est un entier.

Nous allons voir DEFVAR, DEFUN, LAMBDA, SETQ, PSETQ.

DECLARATION DE VARIABLES

Lorsque l'on fait référence à un nouvel atome, dans une liste à évaluer, un atome portant ce nom, de type et de définition NIL est créé. A partir de ce moment, cet atome existe et a une adresse. On peut le laisser sans définition ni type, si ce n'est pas nécessaire, comme pour évaluer (CAR '(TOTO TATA TITI)) qui ne demande pas d'évaluer ni TOTO, ni TATA, ni TITI.

Pour créer une variable, il faut donner à cet atome le type GLOBAL. C'est le rôle de la primitive DEFVAR.

Par exemple (DEFVAR TOTO) donne le type global à TOTO (qui n'est pas évalué). Si TOTO a déjà un type, autre que NIL, une erreur se produit.

Il faut bien noter que ce n'est pas DEFVAR qui crée l'atome TOTO, puisque celui-ci existe déjà lorsque Lisp se rend compte qu'il va falloir exécuter DEFVAR, et qu'il a le type NIL.

DEFVAR permet également de donner une définition en même temps que le type GLOBAL. Exemple : (DEFVAR UN 1) définit la variable UN avec la valeur initiale 1. Pour modifier le contenu des variables, on se sert de SETQ et PSETQ.

(SETQ var_1 val_1 var_2 val_2... var_n val_n) donne séquentiellement la valeur val_i (évaluée) à la variable var_i (non évaluée).

(PSETQ var_1 val_1 var_2 val_2... var_n val_n) donne simultanément la valeur val_i (évaluée) à la variable var_i (non évaluée).

Exemple : les variables A, B et C étant déclarées,

(SETQ A (+ 1) B (+ 1 1) C (+ 1 1 1)) donne 3, et a mis 1 dans A, 2 dans B et 3 dans C.

A donne 1
B donne 2
C donne 3

(SETQ A B B A) donne 2, et a mis 2 (B) dans A, puis 2 (A) dans B :

A donne 2
B donne 2

(SETQ A 1) remet 1 dans A.

(PSETQ A B B A) donne 1, et a mis 2 (B) dans A, et 1 (A) dans B :

A donne 2
B donne 1

DECLARATION DE FONCTIONS

Pour créer une fonction, il faut donner le type FUNCTION à un atome, et on est obligé de le définir en même temps par une syntaxe particulière :

(DEFUN nom (arguments) corps)

Exemple : pour définir la fonction +1 qui opère sur un entier et lui ajoute 1, il faut écrire :

(DEFUN +1 (N) (+ N 1)) qui donne +1.

Alors,

(+1 5) donne 6
(+1 1 2) donne "Parameter Mismatch"

+1 se sert d'un atome de type LOCAL de nom N. Dans ce Lisp71, la notion d'environnement local est absente, et un seul atome peut porter le nom N. Cependant, les variables locales ont une structure particulière car l'exécution de +1 avec un argument (5 par exemple) va "empiler" 5 quelque part, et l'évaluation de N donnera 5. +1 évalué, la valeur 5 est dépilée. C'est la primitive LAMBDA qui effectue ces opérations (si vous évaluez l'atome +1, la machine retourne sa définition (LAMBDA (N) (+ N 1))).

En fait, le type de N est un atome sans nom, qui lui, porte le type LOCAL. La définition de cet atome est une liste des valeurs de N dans les différents environnements, et la définition de N est sa valeur courante. ((LAMBDA (N) ...) 5) ajoute 5 à la liste des valeurs de N, et donne la valeur 5 à N, et (+ N 1) évalué, 5 est retiré de la liste, et N reprend sa valeur initiale, l'exécution de ((LAMBDA (N) (+ N 1)) 5) retourne 6.

Je pense que les évaluateurs qui vont regarder si le car du car de la liste à évaluer est LAMBDA ne sont pas de bons évaluateurs, car il n'y a pas de raison pour privilégier une primitive LAMBDA devant une autre, et mettre une telle "rustine" dans un évaluateur. C'est l'évaluateur l'élément le plus intéressant de Lisp. Il doit être aussi simple que possible. Le mien est déjà trop compliqué. On doit pouvoir simplifier Lisp71, ce qui l'accélérera par la même occasion.

Comme le message d'entrée en Lisp l'indique, ce que je vous présente est une version 1.1 : je l'ai écrite du début à la fin en effectuant quelques corrections. Une révision devrait le rendre plus présentable.

Serge Vaudenay (124)

LISP EN FORTH

J'ai découvert Lisp depuis peu, et je me suis soudain trouvé en manque d'un tel langage sur le matériel que j'utilise. J'ai donc profité des vacances de fin d'année pour faire une première version du *sv Lisp*.

Ce Lisp est écrit en Forth, et est donc assez lent. Cependant, les mots Forth devraient progressivement se transformer en primitives. Il serait bon d'entreprendre un travail collectif sur se sujet, car je suis persuadé qu'un langage tel le Lisp, c'est à dire très axiomatique, et qui fait appel à la logique, aux mathématiques, mais également à des disciplines fréquemment utilisées en intelligence artificielle telles la linguistique, la psychologie... soulève beaucoup plus de problèmes que la simple programmation sur un ordinateur.

Le Lisp que je vous propose est une vision personnelle du Lisp. L'un des grands principes du Lisp étant la transparence des mécanismes internes, cette vision personnelle ne devrait pas se voir à l'utilisation. J'ai essayé de respecter les conventions de *Common Lisp* exposées par Sheila Hughes dans son *Computer Handbook*, mais j'ai dû introduire une notion nouvelle en Lisp : celle de type...

Je ne m'étendrai pas sur ce sujet aujourd'hui, je vais simplement décrire l'installation du Lisp.

Le fichier source, pour des raisons techniques, est scindé en quatre parties, dont les trois premières font environ 4 Ko : LISP1, LISP2, LISP3, et LISP4. Avant de les charger avec LOADF, réservez au moins 9 Ko pour le fichier FORTHRAM à l'aide de GROW. Si une erreur se produit au cours du chargement de LISP3 par exemple, pour reprendre le chargement, faites :
`FORGET LISP3 " LISP3" LOADF`
cela évite d'avoir à recharger LISP1 et LISP2.

Le programme chargé, il est déconseillé de modifier le fichier FORTHRAM tant que le fichier travail du Lisp sera présent. Pour lancer le Lisp, écrivez LISP tout simplement en mode Forth. Si ce n'est déjà fait, LISP créera un fichier LISPRAM de type 2 de 1 Ko que vous ne pourrez qu'effacer.

Entrée dans le Lisp, votre machine affiche un message de présentation. Vous pouvez alors évaluer des objets Lisp. Il est déconseillé de sortir brutalement du mode Lisp ; en général, cela ne peut se produire, mais si à la suite d'un bug (affichage d'un message d'erreur en Forth, ou d'un message non standard destiné au débugage), ou à la suite d'une extinction de la machine, vous vous retrouviez en Forth, revenez en Lisp à l'aide de GO. Pour sortir du Lisp, utilisez la primitive BYE (tapez (BYE)).

Les primitives Lisp sont en majuscule. Voici leur liste : (NIL BYE + - * / ATOM EQ NOT AND OR PLUSP MINUSP ZEROP NUMBERP < > = /= QUOTE EVAL CAR CDR CONS DEFVAR SETQ PSETQ LAMBDA DEFUN). Les objets Lisp traités par ce programme sont uniquement les listes et les atomes. Parmi ces atomes, on distingue les atomes *primitifs* et ceux définis par l'utilisateur. Parmi les atomes primitifs, on retrouve les primitives énoncées ci-dessus, ainsi que les entiers (écrits dans la base courante du Forth) de 20 bits, et les messages standards.

Ainsi, la liste (/ 1 0) a pour valeur le message standard /Zero.

Les atomes non primitifs sont automatiquement créés. Ainsi, si dans une liste telle (QUOTE (CHOUCROUTE RAVIOLI)), vous faites référence à des atomes inconnus, ils sont créés et codés... Le type NIL leur est assigné, si bien que leur évaluation ne fait rien de spécial. Dans une liste comme (DEFVAR UN 1), UN est d'abord créé, et l'évaluation de la liste attribue le type *variable globale* à UN, qui prend la valeur 1. Il existe également les types *variable locale*, et *fonction*, utilisés par LAMBDA et DEFUN.

Ce sera tout pour aujourd'hui. J'attend vos remarques sur cette version du Lisp.

Serge Vaudenay (124)

Fichier LISP1 :

```
DECIMAL .( ..lisp1..)

: LISP1 ;
8 STRING FIC-TRAVAIL
0 CONSTANT DD
0 CONSTANT FD
VARIABLE CDO
VARIABLE MDO
VARIABLE LDO

: INIT
  FIC-TRAVAIL FINDF
  DUP 0= ABORT" pas trouve"
  DUP 52 + [ ' DD 5+ ] LITERAL !
```

```
  DUP 32 + DUP @ + 1- [ ' FD 5+ ] LITERAL !
  37 + DUP @ CD0 !
  5+ DUP @ MD0 ! 5+ @ LD0 !
;

: FIN
  FIC-TRAVAIL FINDF 37 +
  CD0 @ OVER !
  5+ MD0 @ OVER !
  5+ LD0 @ SWAP !
;

: LISPT
  FIC-TRAVAIL S!
  FIC-TRAVAIL FINDF
  0=
  IF
    FIC-TRAVAIL 2000 CREATEF
    DUP 0= ABORT" peut pas"
    DUP 16 + 1 SWAP +!
    37 + 0 OVER ! 5+ 0
    OVER ! 5+ 0 OVER ! 5+ 0 SWAP N!
  THEN
  INIT
;

: TAILLE
  DUP N@ DUP
  CASE
     8 OF 2DROP 0 11 ENDOF
     7 OF 2DROP 1 11 ENDOF
     0 OF 1- FD ROT - ENDOF
    15 OF 2DROP 0 11 ENDOF 6 <
     0 OF 2 MOD SWAP 1+ @ ENDOF
    DROP 1 SWAP ROT
  ENDCASE
;
: CH-FIN
  CD0 @ MD0 @ LD0 @ MAX MAX DD +
  BEGIN
    DUP TAILLE ROT +
    SWAP 1+
    0=
  UNTIL
;
: CH-PLACE
  BEGIN
    DUP N@ 4 PICK <>
    OVER TAILLE SWAP 1+
    OVER 6 PICK <
    4 ROLL OR OVER AND
  WHILE
    DROP +
  REPEAT
  OVER 5 PICK < ABORT" plus de place"
  2/ *
;
: CR-D
  DUP >R >R ROT 1+ OVER N!
  2DUP 1+ ! 2DUP + R> 4 ROLL - R>
  IF
    DUP 5 >
    IF
      6 3 PICK N! SWAP 1+ !
    ELSE
      DUP
      IF
        SWAP N!
      ELSE 2DROP
      THEN
    THEN
  ELSE
    DROP 0 SWAP N!
  THEN
;

: CR-MD
  9 SWAP MD0 @ DD + CH-PLACE CR-D DUP DD - MD0 !
;

: CR-LD
  11 SWAP LD0 @ DD + CH-PLACE CR-D DUP DD - LD0 !
;
: CR-CD
  7 11 CD0 @ DD + CH-PLACE CR-D DUP DD - CD0 !
;

: -ADR
  65536 /MOD
  CASE
    0 OF DD + ENDOF
    1 OF DD + ENDOF
    6 OF 195325 + ENDOF
    7 OF 195325 + ENDOF
    1 ABORT" pas d'adresse"
  ENDCASE
;

0 CONSTANT 'AFFICHE
0 CONSTANT mNIL

: ADR.
  DUP N@ DUP
  15 =
  IF DROP 8 THEN
  CASE
     8 OF
        40 EMIT
        BEGIN
          1+ DUP @
          'AFFICHE EXECUTE
          5+ @ DUP mNIL <>
        WHILE
          -ADR SPACE
        REPEAT
        DROP 41 EMIT
```

```
      ENDOF
    10 OF 16 + COUNT TYPE ENDOF
    1 ABORT" non affichable"
  ENDCASE
;

: AFFICHE
  65536 /MOD
  CASE
    0 OF DD + ADR. ENDOF
    1 OF DD + ADR. ENDOF
    4 OF
        DUP 32768 AND
        30 * + DUP ABS 0
        <# #S ROT SIGN #>
        TYPE
      ENDOF
    5 OF " MSG$(FORTHI)" BASIC$ TYPE ENDOF
    6 OF 195325 + ADR. ENDOF
    7 OF
        195325 + 2- -1
        TRAVERSE COUNT 31 AND
        1- TYPE
      ENDOF
    ABORT" mot inconnu"
  ENDCASE
;

' AFFICHE ' 'AFFICHE 5+ !

: CR-ATOME
  DUP 2* 18 + CR-MD mNIL OVER 6 + !
  mNIL OVER 11 + !
  DUP 2SWAP ROT 16 + 2DUP C! 2+ SMOVE DD -
;

29 CONSTANT NPRIM
CREATE PRIMITIVES NPRIM 5 * NALLOT

: PRIM
  ' 2- -1 TRAVERSE PRIMITIVES ROT 5 * + !
;
: CH-ATOME
  1 NPRIM 5 * PRIMITIVES + PRIMITIVES
  DO
    DROP 2DUP
    I @
    COUNT 31 AND 1- S=
    IF
      I @ 1 TRAVERSE 263429 + 0
      LEAVE
    THEN
    1 5
  +LOOP
  IF
    DD 0
    BEGIN
      DROP DUP N@ 10 = OVER DUP TAILLE ROT + SWAP 1+ 0= ;
    IF
      2DROP DD 1 - DUP
    ELSE
      ROT ROT
      IF
        DUP 16 + COUNT 6 PICK 6 PICK S=
      ELSE
        0
      THEN
    THEN
  UNTIL DD - SWAP DROP
THEN
>R 2DROP R>
;

: CHR-ATOME
  2DUP CH-ATOME DUP 1+
  IF
    ROT ROT 2DROP
  ELSE
    DROP CR-ATOME
  THEN
;

: PARENTHESE
  DROP DUP 0
  BEGIN
    OVER C@
    CASE
      40 OF 1+ ENDOF
      41 OF 1- ENDOF
    ENDCASE
    SWAP 2+ SWAP DUP 0=
  UNTIL
  DROP OVER - 2/
;

: PREMIER
  OVER C@
  CASE
    40 OF PARENTHESE ENDOF
    BL OF DROP 0 ENDOF
    DROP OVER DUP >R >R 2* + R>
    BEGIN
      2+ DUP C@
      CASE
        BL OF 1 ENDOF
        40 OF 1 ENDOF
        41 OF 1 ENDOF
        OVER 4 PICK = SWAP
      ENDCASE
    UNTIL
    R> SWAP OVER - 2/ ROT
  ENDCASE
```

```
: BLANCS
  BEGIN
    OVER C@ BL = OVER AND
  WHILE
    1- SWAP 2+ SWAP
  REPEAT
  BEGIN
    2DUP 1- 2* + C@ BL = OVER AND
  WHILE
    1-
  REPEAT
;

: NOMBRE
  OVER C@ 45 =
  IF
    DUP 1-
    IF
      1- SWAP 2+ SWAP -1
    ELSE
      1
    THEN
  ELSE
    1
  THEN
  ROT ROT -1 SWAP 0 DUP >R
  DO
    OVER I 2* + C@ DUP 47 > OVER 58 < AND SWAP OVER
    IF 7 + THEN
    DUP 64 > OVER 91 < AND SWAP 55 -
    BASE @ J OVER *
    3 PICK + R> R> R>
    DROP 3 PICK >R >R
    >R -65536 AND 0= >R < >R OR R> R> AND AND AND
  LOOP
  SWAP DROP SWAP R> * SWAP
;
```

Fichier LISP2 :

```
.( ..lisp2..)

: LISP2 ;

: ID-ATOME
  2DUP NOMBRE
  IF
    65535 AND 262144 + ROT ROT 2DROP
  ELSE
    DROP CHR-ATOME
  THEN
;

: CR-LISTE
  [ SMUDGE ]
  OVER C@ 40 <>
  IF
    ID-ATOME
  ELSE
    2- SWAP 2+ SWAP BLANCS
    DUP
    IF
      2DUP PREMIER 2DUP >R >R SWAP DROP DUP >R -
      SWAP R> 2* + SWAP R> R> CR-LISTE ROT ROT 2+
      SWAP 2- 40 OVER C! SWAP 2DUP 2* + 41
      SWAP 2- C! CR-LISTE CR-CD SWAP OVER 6 + !
      SWAP OVER 1+ ! DD -
    ELSE
      2DROP mNIL
    THEN
  THEN
  [ SMUDGE ]
;

: EFFACE
  DUP @ 3 PICK DD - MIN SWAP !
  DUP N@ 1- SWAP N!
;

: EF-LISTE
  [ SMUDGE ]
  65536 /MOD 0= OVER DD + N@ 8 = AND
  IF
    DD + DUP CD0 EFFACE 1+ DUP @
    SWAP 5+ @
    EF-LISTE EF-LISTE
  ELSE
    DROP
  THEN
  [ SMUDGE ]
;

: ENTITE
  SWAP mNIL <>
  IF DROP 327755 THEN
;

0 CONSTANT 'EVALUE

: SEC?
  65536 / 1 =
;

: SEC
  DUP 65536 / 0=
  IF 65536 + THEN
;

: CDAR
  DUP SEC? SWAP -ADR DUP 6 + @ 3 PICK
  IF SEC THEN
  SWAP 1+ @ ROT
  IF SEC THEN
;
```

```
: ADR
  DUP N@ DUP 15 =
  IF DROP 8 THEN
  CASE
    8 OF DROP CDAR ENDOF
   10 OF 6 + @ ENDOF
    1 ABORT" non evaluable"
  ENDCASE
  'EVALUE EXECUTE ENTITE
;


: EVALUE
  DUP 65536 /MOD
  CASE
    0 OF DD + ADR ENDOF
    1 OF DD + ADR ENDOF
    4 OF DROP ENTITE ENDOF
    5 OF DROP ENTITE ENDOF
    6 OF 195325 + ADR ENDOF
    7 OF SWAP DROP 195325 + EXECUTE ENDOF
    ABORT" mot inconnu"
  ENDCASE
;


' EVALUE ' 'EVALUE 5+ !


: ATOM
  DUP 65536 / DUP 7 = SWAP 2/ 2 = OR
  IF
    DROP -1
  ELSE
    -ADR N@ 10 =
  THEN
;

: ADEF
  DUP 65536 /
  CASE
    0 OF -ADR 11 + @ ENDOF
    1 OF -ADR 11 + @ ENDOF
    6 OF -ADR 11 + @ ENDOF
  ENDCASE
;

: EXE
  mNIL OVER DUP ATOM 0=
  IF
    CDAR EVALUE
  ELSE
    ADEF
  THEN
  ROT EF-LISTE SWAP DROP
;

1 STRING """ " '" """ S!
```

```
: QUOTAGE
  BEGIN
    """ 2OVER POS DUP
  WHILE
    >R 2DUP R> DUP >R DUP >R 2* ROT +
    SWAP R> - 2DUP DUP
    IF PREMIER THEN
    2SWAP 3 PICK - SWAP 3 PICK 2* +
    SWAP 2SWAP 6 ROLL 6 ROLL R> 1- LEFT$
    " FORTH$&'(QUOTE '&FORTH$&')'&FORTH$" BASIC$
  REPEAT
  DROP
;


: PAR
  0 OVER 0
  DO
    3 PICK I 2* + C@
    CASE
      40 OF 1+ ENDOF
      41 OF 1- ENDOF
    ENDCASE
    DUP 0<
    IF LEAVE THEN
  LOOP
;

: FERME
  PAR DUP 0>
  IF
    0
    DO
      " )" S<&
    LOOP
    0
  THEN
;

100 STRING LTIB

: ENTRE
  LTIB DROP EXPECT96 SPAN @ DUP
  IF
    LTIB DROP 2- C! LTIB FERME
    IF
      327760 -1
    ELSE
      QUOTAGE BLANCS 2DUP 2DUP PREMIER
      ROT <> >R <> R> OR
      IF
        DROP 327719 -1
      ELSE
        0
      THEN
    THEN
  ELSE
    mNIL -1
```

```
  THEN
;

: EDITE
  BEGIN
    ENTRE
    IF
      ." )" AFFICHE DROP
    ELSE
      CR-LISTE EXE ." )" DUP AFFICHE EF-LISTE
    THEN
    CR 0
  UNTIL
;

: BYEp
  0 ONERR ! FIN DROP 2DROP ." ..." QUIT
; 1 PRIM BYEp

: GO
  ." sv LISP v1.1" CR EDITE
;

: LISP
  " LISPRAM" LISPT GO
;

: LIST-
  0
  BEGIN
    OVER mNIL <>
  WHILE
    1+ SWAP CDAR SWAP ROT
  REPEAT
  SWAP DROP
;

: -LIST
  mNIL SWAP 0
  DO
    CR-CD SWAP OVER 6 + !
    SWAP OVER 1+ ! DD -
  LOOP
;

: NDROP
  DUP 1+ 0
  DO
    DROP
  LOOP
;

: AEXE
  0
  BEGIN
    OVER mNIL <>
  WHILE
    1+ SWAP CDAR EXE SWAP ROT
  REPEAT
  SWAP DROP
;

: ARG
  OVER <>
  IF NDROP R> DROP 327716 THEN
;

: ERR
  DUP 0
  DO
    I 2+ PICK 65536 / 5 =
    IF
      I 2+ PICK >R NDROP R> R> R> R>
      DROP >R >R LEAVE
    THEN
  LOOP
;

: OARG
  mNIL <>
  IF R> DROP 327716 THEN
;

: NUM
  65536 / 4 =
;

: LIST
  DUP ATOM 0= SWAP mNIL = OR
;

: NUMS
  DUP 0
  DO
    DUP 1+ ROLL 65536 /MOD 4 <>
    IF
      SWAP NDROP R> R> R> DROP >R >R 327711 LEAVE
    THEN
    DUP 32768 AND 30 * + SWAP
  LOOP
;

: NILp
  OARG mNIL
;

' NILp 263427 + ' mNIL 5+ ! 0 PRIM NILp
HERE 160 C, 20 , mNIL , 0 , 1 C, 84 C,
197892 + CONSTANT mT

: NOARG
  DUP 0=
  IF R> 2DROP 327716 THEN
;
```

```
: +p
  AEXE NOARG ERR NUMS 0 SWAP 0
  DO
    +
  LOOP
  65535 AND 262144 +
; 2 PRIM +p

: -p
  AEXE NOARG ERR NUMS 0 OVER 2+ ROLL ROT 0
  DO
    SWAP -
  LOOP
  65535 AND 262144 +
; 3 PRIM -p

: *p
  AEXE NOARG ERR NUMS 1 SWAP 0
  DO
    *
  LOOP
  65535 AND 262144 +
; 4 PRIM *p
```

Fichier LISP3 :

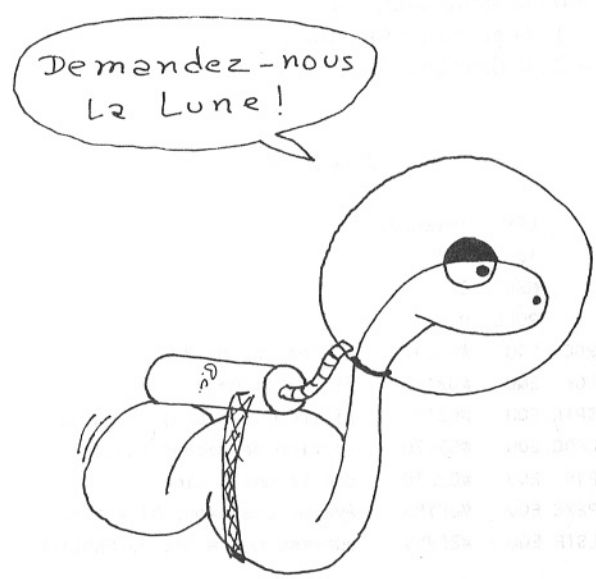
```
.( ..lisp3..)

: LISP3 ;

: /p
  AEXE NOARG ERR NUMS 1 OVER 2+ ROLL 1 4 ROLL
  DO
    SWAP DUP
    IF /
    ELSE DROP I NDROP 327688 LEAVE
    THEN
    -1
  +LOOP
  DUP 327688 <>
  IF
    65535 AND 262144 +
  THEN
; 5 PRIM /p

: PREDICAT
  IF mT
  ELSE mNIL
  THEN
;

: ATOMp
  LIST- 1 ARG DROP EXE ATOM PREDICAT
; 6 PRIM ATOMp

: EQp
  AEXE NOARG OVER -1 ROT 0
  DO
    >R OVER = R> AND
  LOOP
  PREDICAT SWAP DROP
; 7 PRIM EQp


: NOTp
  LIST- 1 ARG DROP EXE mNIL = PREDICAT
; 8 PRIM NOTp


: ANDp
  AEXE NOARG -1 SWAP 0
  DO
    SWAP mNIL = 0= AND
  LOOP
  PREDICAT
; 9 PRIM ANDp

: ORp
  AEXE NOARG 0 SWAP 0
  DO
    SWAP mNIL = 0= OR
  LOOP
  PREDICAT
; 10 PRIM ORp

: PLUSPp
  AEXE 1 ARG ERR NUMS DROP 0> PREDICAT
; 11 PRIM PLUSPp

: MINUSPp
  AEXE 1 ARG ERR NUMS DROP 0< PREDICAT
; 12 PRIM MINUSPp

: ZEROPp
  AEXE 1 ARG ERR NUMS DROP 0= PREDICAT
; 13 PRIM ZEROPp

: NUMBERPp
  LIST- 1 ARG DROP EXE NUM PREDICAT
; 14 PRIM NUMBERPp

: <p
  AEXE NOARG ERR NUMS 65536 -1 ROT 0
  DO
    >R OVER > R> AND
  LOOP
  SWAP DROP PREDICAT
; 15 PRIM <p

: >p
  AEXE NOARG ERR NUMS -65536 -1 ROT 0
  DO
    >R OVER < R> AND
  LOOP
  SWAP DROP PREDICAT
; 16 PRIM >p
```

```
: =p
  AEXE NOARG ERR NUMS OVER -1 ROT 0
  DO
    >R OVER = R> AND
  LOOP
  SWAP DROP PREDICAT
; 17 PRIM =p

: /=p
  AEXE NOARG ERR NUMS -1 OVER 0
  DO
    OVER 1- DUP
    IF
      0
      DO
        I 4 + PICK 4 PICK <> AND
      LOOP
    ELSE DROP
    THEN
    ROT DROP >R 1- R>
  LOOP
  SWAP DROP PREDICAT
; 18 PRIM /=p

: QUOTEp
  DUP LIST- 1+ 2 ARG DROP DUP SEC 3 PICK -ADR 1+ !
  SWAP SEC?
  IF SEC THEN
; 19 PRIM QUOTEp

: EVALp
  LIST- 1 ARG DROP EXE EXE
; 20 PRIM EVALp

: CARp
  AEXE 1 ARG ERR DROP DUP ATOM
  IF mNIL =
    IF mNIL ELSE 327711 THEN
  ELSE
    DUP -ADR 1+ DUP @ DUP SEC ROT !
    OVER EF-LISTE SWAP SEC? IF SEC THEN
  THEN
; 21 PRIM CARp

: CDRp
  AEXE 1 ARG ERR DROP DUP ATOM
  IF mNIL =
    IF mNIL ELSE 327711 THEN
  ELSE
    DUP -ADR 6 + DUP @ DUP SEC ROT ! OVER
    EF-LISTE SWAP SEC? IF SEC THEN
  THEN
; 22 PRIM CDRp

: CONSp
  AEXE 2 ARG ERR DROP CR-CD SWAP OVER 6 + ! SWAP
  OVER 1+ ! DD -
; 23 PRIM CONSp

: GLOBALt
  -ADR 11 + @
; ' GLOBALt 263427 + CONSTANT mGLOBAL

: SETVAR
  DUP 131072 /
  IF
    2DROP 2DROP 327740
  ELSE
    -ADR DUP N@ 10 <>
    IF
      2DROP 2DROP 327691
    ELSE
      DUP 6 + @ mNIL <> 5 PICK =
      IF
        2DROP 2DROP 327763
      ELSE
        ROT 4 ROLL
        IF OVER 6 + ! ELSE DROP THEN
        11 + SWAP SEC DUP ROT !
      THEN
    THEN
  THEN
;

: DEFVARp
  LIST- NOARG DUP 1 =
  IF DROP mNIL 2 THEN
  2 ARG DROP EXE -1 SWAP ROT mGLOBAL ROT ROT SETVAR
; 24 PRIM DEFVARp

: SETQp
  LIST- NOARG DUP 2 MOD
  IF
    NDROP 327720
  ELSE
    0 OVER 2/ 0
    DO
      DROP 2- 0 OVER 4 + ROLL DUP 4 PICK
      5+ ROLL EXE SWAP SETVAR
    LOOP
    SWAP DROP
  THEN
; 25 PRIM SETQp

: PSETQp
  LIST- NOARG DUP 2 MOD
  IF
    NDROP 327720
  ELSE
    DUP 2/ 0
    DO
      DUP 1+ ROLL SWAP DUP 1+ ROLL EXE SWAP
    LOOP
    0 OVER 2/ 0
    DO
      DROP 2- 0 OVER 4 + ROLL DUP
      4 PICK 5+ ROLL SWAP SETVAR
```

```
    LOOP
    SWAP DROP
  THEN
; 26 PRIM PSETQp

: LOCALt
  DROP -ADR 11 + @ mNIL SWAP
; ' LOCALt 263427 + CONSTANT mLOCAL

: ASN
  SWAP -ADR 6 + DUP @ 3 PICK ROT 5+ ! -ADR 11 +
  DUP @ CR-CD SWAP OVER 6 + ! ROT OVER 1+ ! DD -
  SEC SWAP !
;

: LOCAL
  DUP ATOM OVER 131072 / 0= AND
  IF
    -ADR 6 + DUP @ DUP mNIL <>
    IF
      DUP ATOM SWAP -ADR 6 + @ mLOCAL =
      AND SWAP DROP
    ELSE
      DROP NULL$ CR-ATOME mLOCAL
      OVER -ADR 6 + ! SWAP ! -1
    THEN
  ELSE DROP 0
  THEN
;

: NASN
  DUP
  IF
    1 SWAP
    DO
      I 2* ROLL I 1+ ROLL OVER LOCAL 0=
      IF I 2* 1+ NDROP 327740 LEAVE THEN
      EXE ASN -1
    +LOOP
  ELSE DROP
  THEN
;

: POP
  -ADR 6 + DUP @ -ADR 11 + DUP @ -ADR DUP CDO
  EFFACE 6 + @ DUP ROT ! DUP mNIL <>
  IF -ADR 1+ @ THEN
  SWAP 5+ !
;

: NPOP
  DUP
  IF
    0
    DO
      DUP LOCAL
      IF POP ELSE DROP THEN
    LOOP
  ELSE DROP
  THEN
;

: LAMBDAp
  LIST- 2 ARG ERR DROP OVER DUP ATOM
  IF
    2DROP 2DROP mNIL 327711
  ELSE
    LIST- DUP 4 + ROLL LIST- DUP 2+ ROLL 2DUP <>
    IF
      + 2+ NDROP 327716
    ELSE
      DROP NASN EXE SWAP LIST- NPOP
    THEN
    mNIL SWAP
  THEN
; 27 PRIM LAMBDAp

: FUNCTIONt
  -ADR 11 + @ EVALUE mNIL SWAP
; ' FUNCTIONt 263427 + CONSTANT mFUNCTION
```

Fichier LISP4 :

```
.( ..lisp4..)

: LISP4 ;

: DEFUNp
  DUP LIST- 1+ 4 ARG ERR DROP 2DROP CR-CD ROT
  -ADR 6 + DUP @ mNIL ROT ! OVER 6 + !
  [ ' LAMBDAp 263427 + ] LITERAL
  OVER 1+ ! DD - -1 mFUNCTION ROT 4 PICK SETVAR
  DUP 65536 / 5 =
  IF SWAP THEN
  DROP
; 28 PRIM DEFUNp
```



QUELQUES FONCTIONS

Et voici pour terminer la série, ce Lex qui contient quelques fonctions utiles. En voici la liste :

Ordre ELSTR

syntaxe : ELSTR [*chaîne*]

Cet ordre prend la chaîne de trois caractères fournie en paramètre, ou la chaîne CR/LF/LF par défaut, et la met à la place de la chaîne définie par ENDLINE.

Fonction EL

syntaxe : EL (*n*), 0 ≤ n ≤ 2

Cette fonction force la longueur de la chaîne définie par ENDLINE et retourne le nombre de caractères avant modification.

Fonction INT$

syntaxe : INT$ (*n*)

Cette fonction est analogue à STR$, mais force le format d'affichage à STD au préalable. Ceci permet d'avoir des nombres toujours bien formattés dans les chaînes de caractères.

Fonction MDIM

syntaxe : MDIM(*X*,*n*) ou MDIM(*X$*,*n*)
Où *X* et *X$* est une matrice, un vecteur ou un tableau alphanumérique, et *n* est un entier compris entre 0 et 2.

Cette fonction renvoie, si :
n = 0 : la limite inférieure
n = 1 : la première dimension
n = 2 : la deuxième dimension

Jack Elhay

```
        LEX   'MISCLEX'
        ID    #5C
        MSG   0
        POLL  0
STRGCK  EQU   #036BA    chaîne valide ?
EOLCK   EQU   #02A7E    fin de ligne ?
RESPTR  EQU   #03172    si trop loin dans la parse
DROPDC  EQU   #05470    routine de décompilation
POP1S   EQU   #0BD38    dépile une chaîne
EXPEXC  EQU   #0F186    évalue une expr. D1 = M.S.
EOLSTR  EQU   #2F95B    adresse de la chaîne ENDLINE
NXTSTM  EQU   #08A48    sortie des ordres
RNDAHX  EQU   #136CB    dépile en hexa dans A(A)
RDATTY  EQU   #17CC6    Data type ! beeeep
ARGERR  EQU   #0BF19    Invalid Arg ! beeeep
EOLLEN  EQU   #2F95A    adresse longueur de ENDLINE
FNRTN1  EQU   #0F216    retour pour les fonctions
A-MULT  EQU   #1B349    multiplie 2 nb hexa
HDFLT   EQU   #1B31B    hex -> flottant
EXPR    EQU   #0F23C    retour pour les fonctions
STR$SB  EQU   #18149    version interne de STR$
DSPFLG  EQU   #2F6DC    display format
MEMBER  EQU   #1B098    octet A(B) dans un ensemble ?
POP1R   EQU   #0E8FD
RJUST   EQU   #12AE2
FLOAT   EQU   #1B322

        ENTRY ENTER
        CHAR  #D        ordre
        ENTRY LABEL1
        CHAR  #F        fonction
        ENTRY LABEL
        CHAR  #F        fonction
        ENTRY entry
        CHAR  #F        fonction

        KEY   'ELSTR'
        TOKEN 8
* syntaxe : ELSTR [ CHR$(N1)&CHR$(N2)&CHR$(N3) ]
        KEY   'EL'
        TOKEN 9
* syntaxe : EL(n), 0<=n<=2
        KEY   'INT$'
        TOKEN 10
* syntaxe : voir STR$
        KEY   'MDIM'
        TOKEN 11
* syntaxe : MDIM(X,n) ou MDIM(X$,n)
* X, X$ = matrice, vecteur ou tableau alphanumérique
* 0<=n<=2
        ENDTXT

ELSTRp  GOSBVL EOLCK    parse ELSTR : paramètre ?
        GOC    resptr   non
        GOSBVL RESPTR   oui: pointeur avant la chaîne
        GOSBVL STRGCK   vérifier chaîne ok
resptr  GOVLNG RESPTR
ELSTRd  GOVLNG DROPDC   routine de décompilation

        REL(5) ELSTRd
        REL(5) ELSTRp
ENTER   A=DAT0 B        Y-a-t'il un  paramètre ?
        LCHEX  F0       Cette partie est tirée des
        ?A<C   B        IDS III (ENDLINE)
        GOYES  EL1      Si oui, brancher en EL1

        D1=(5) EOLSTR   Non: alors utiliser la chaîne
        LCHEX  0A0A0D   par défaut
        P=     5
```

```
        DAT1=C WP       stocker la chaîne dans EOLSTR
        GONC   nxtstm   B.E.T.

EL1     GOSBVL EXPEXC   évaluer l'expression, et
        GOSBVL POP1S    dépiler la chaîne
        C=0    W
        LC(1)  6
        ?A=C   A        6 quartets ?
        GOYES  OK
        GOVLNG ARGERR   non : beeeep !
OK      ADOEX           sauve DO
        DO=(5) EOLSTR   DO pointe sur EOLSTR
        P=     5
        C=DAT1 WP       copie la Math Stack dans C
        DAT0=C WP       copie C dans EOLSTR
        ADOEX           restaure DO
nxtstm GOVLNG NXTSTM    sortie

* fonction EL
* un paramètre numérique obligatoire
        NIBHEX 811
LABEL1 CDOEX
        R3=C            sauver DO en R3
        DO=(5) EOLLEN   DO pointe sur EOLLEN
        A=0    W        efface A
        A=DAT0 1
        ?A=0   W        0 renvoie 0
        GOYES  LABEL2
        A=A-1  A        convertit 4 en 1, 6 en 2
        A=A-1  A
        ASRB            divise par 2
LABEL2 R2=A             sauve l'ancien en R2
        GOSBVL RNDAHX
        GONC   Argerr   pas d'argument négatif
        P=     0
        LC(1)  #2
        ?A>C   P        pas d'argument >2
        GOYES  Argerr
        LC(1)  #5
        C=C-A  A        convertit l'argument en un nb
        GOSBVL A-MULT   de quartets
        DAT0=A 1        stocke résultat dans EOLLEN
        C=R3
        CDOEX           restaure DO
        A=R2            prépare l'ancienne valeur
        GOSBVL HDFLT    pour la sortie
        ACEX   W
        GOVLNG FNRTN1   renvoie l'ancienne valeur
Argerr GOVLNG ARGERR
* fonction INT$
* un paramètre numérique
        NIBHEX 811
LABEL   CDOEX           sauver DO
        R3=C            dans R3
        DO=(5) DSPFLG   adresse du quartet à sauver
        C=0    W        et à changer
        C=DAT0 S        C(S) = ce quartet
        R4=C            sauver en R4
        C=0    W
        P=     0
        LC(1)  #8       Cette valeur va forcer STD
        DAT0=C 1        dans DSPFLG
        GOSBVL POP1R
        GOSBVL RJUST
        GOSBVL FLOAT
        DAT1=A W
        A=0    W
        GOSBVL STR$SB   conversion
        C=R4            restaurer le quartet originel
        DAT0=C S
        C=0    W
        C=R3            restaurer DO
        CDOEX
        GOVLNG EXPR     sortie, chaîne déjà sur M.S.
* MDIM
* 1er : chaîne ou tableau
* 2eme : numérique
        NIBHEX 8E22
entry   GOSBVL RNDAHX   dépiler le numérique en hexa
        GONC   argerr
        D1=D1+ 16       sauter le numérique
        R1=A            le sauver dans R1
        A=DAT1 B        type de l'objet sur la M.S.
        P=     0
        LCHEX  #1A1B1C1D1E les types des vecteurs
        P=     9
        GOSBVL MEMBER   est-ce un de ceux-là ?
        GONC   start1   si oui, brancher
        LCHEX  #2A2B2C2D2E1F matrices, F1 = tab$
        P=     11
        GOSBVL MEMBER   est-ce un de ceux-là ?
        GONC   start2   si oui, brancher
        GOVLNG RDATTY   si non : Data Type !
start1 ST=1    4        flag 4 = 1 si vecteur
        GOTO   start
argerr GOVLNG ARGERR
start2 ST=0    4        flag 4 = 0 sinon
start   A=R1            A = paramètre numérique
        C=0    W
        ?A=C   A        était-ce 0 ?
        GOYES  base     limite inférieure
        C=C+1  A
        ?A=C   A        était-ce 1?
        GOYES  dim1     nombre de lignes
        C=C+1  A
        ?A>C   A        argument > 2 ?
        GOYES  argerr   Invalid Arg ! sinon : dim2
dim2    ?ST=1  4        est-ce un vecteur ?
        GOYES  vector
        GOSUB  saveD1   sans commentaire...
        D1=D1+ 3        q. 3-6 = dim 2 (ou maxlen)
        A=DAT1 4        dans A
        GOTO   out      et retour à l'utilisateur
base    GOSUB  saveD1
        D1=D1+ 2        q. 2 = limite inférieure
        A=DAT1 1        dans A
```

```
out     C=R0            retour à l'utilisateur
        CD1EX
out1    GOSBVL HDFLT    D1 était bon pour "vecteur"
        D1=D1- 16       préparer D1 pour la sortie
        ACEX   W
        GOVLNG FNRTN1   renvoyer le résultat
saveD1  CD1EX           routine pour sauver D1
        R0=C            dans R0
        CD1EX
        RTN
dim1    GOSUB  saveD1
        D1=D1+ 7        q. 7-10 =  nb de lignes
        A=DAT1 4        dans A
        GOTO   out
vector  A=0    W
        GOTO   out1     0 pour dim2 des vecteurs
        END
```

COMPILEZ VOS CONSTANTES

A l'image du programme de musique de Serge Vaudenay, (voir JPC 22, page 46) voici un programme qui crée des fichiers Lex. Ne vous emballez pas, il ne s'agit pas de construire des routines évoluées, mais des programmes simples dont la réalisation manuelle est fastidieuse : les fonctions constantes numériques.

On introduit les noms des fonctions, puis leur valeur, les caractéristiques propres du programme : nom du Lex, ID, numéro du plus bas token, puis le programme fait le reste. La dernière étape à réaliser consiste à assembler le fichier Texte créé.

Dans ce programme, la partie la plus intéressante est la recherche de l'ordre dans lequel il faut mettre les mots. En effet, il faut que les mots soient classés par ordre alphabétique, mais avec toutefois une exception : par exemple si on a les mots AA et AAS, il faudra placer AAS avant AA pour que AAS soit reconnu.

Dernière recommandation : les noms de fonctions doivent avoir au minimum 2 lettres et au maximum 8 lettres.

Amusez-vous bien...

Lionel Guillou (326)

CALENDROMANIE

Tout d'abord je tiens à vous remercier pour avoir bien voulu publier mes deux programmes de calcul de dates et de durées dans *JPC 50*.

Le calendrier Julien

Pour que *JPC* soit complet, il me paraît indiqué de communiquer mes cogitations à nos amis à propos du calendrier Julien en usage du jeudi 4 janvier 45 avant Jésus Christ (c'est à dire l'an 709 de Rome) au jeudi 4 octobre 1582 où la réforme de Jules César a fait place à celle de Grégoire XIII, le lendemain devenant le vendredi 15 octobre 1582.

Le calendrier Julien offre moins d'intérêt que l'actuel. Cependant, pour les amoureux de l'Histoire...

Voici quelques dates de contrôle :

- 15 janvier 69 : Marcus Silvius Othon couronné Imperator. C'était un dimanche.
- 1 juillet 987 : couronnement de Hugues Capet. C'était un vendredi.
- 14 octobre 1066 : bataille de Hastings. C'était un samedi.
- 25 août 1270 : mort de Saint Louis devant Tunis. C'était un lundi.
- 26 août 1346 : bataille de Crécy. C'était un samedi.
- 25 octobre 1415 : bataille d'Azincourt. C'était un vendredi.
- 30 mai 1431 : mort de Jeanne d'Arc. C'était un mercredi.
- 12 octobre 1492 : découverte de l'Amérique par Christophe Colomb. C'était un vendredi.
- 13 septembre 1515 : bataille de Marignan. C'était un jeudi.
- 24 août 1572 : massacre de la saint Barthélémy. C'était un dimanche.

Les jours de Pâques

Je complète ma « calendromanie » par un programme donnant les jours de Pâques par la méthode de Gauss. En référence, outre l'année en cours, on peut noter :

- 30 mars 1282 : Lundi de Pâques dit *Vêpres Siciliennes*,
- 5 avril 1722 : découverte de l'Ile de Pâques,
- 16 avril 1797 : Pâques Véronaises.

En espérant toute votre indulgence, je vous adresse toutes mes félicitations pour la très haute qualité de *JPC* où mes petits programmes peuvent faire piètre mine.

Jean Maille (392)

Programme "BUILD" (création de fichier Lex de constantes numériques)

```
 10 STD @ DELAY .5 @ DESTROY ALL @ LC OFF
 20 DISP "      Lex builder"
 30 INPUT "Fichier destination:";A$
 40 CREATE TEXT A$
 50 INPUT "Nom du lex:";N$ @ N$=N$[1,8]
 60 INPUT "Id:";I$
 70 INPUT "Plus bas token:";T
 80 INPUT "Nbre de fonctions:";N
 90 DIM B$(N),V(N)
100 FOR I=1 TO N @ DISP "Nom Fonction ";I;":"; @ INPUT B$(I) @ B$(I)=B$(I)[1,8]
110     DISP "Valeur Fonction";I;":"; @ INPUT V(I) @ NEXT I
  - Construction fichier lex
120 ASSIGN #1 TO A$
130 PRINT #1;"       LEX     '"&N$&"'"
140 PRINT #1;"       ID      #"&I$
150 PRINT #1;"       MSG     0"
160 PRINT #1;"       POLL    0"
170 PRINT #1;"FNRTN1 EQU     #0F216"
  - Tri des fonctions
180 DIM C$(N),W(N)
190 C$(1)=B$(1) @ W(1)=V(1)
200 FOR I=2 TO N @ J=1

210     'BOUCLE': IF C$(J)=B$(I)[1,LEN(C$(J))] THEN GOSUB 'INTER' @ GOTO 'NEXT'
220     IF B$(I)<C$(J) THEN GOSUB 'INTER' @ GOTO 'NEXT'
230     J=J+1
240     IF J#I THEN 'BOUCLE'
250     C$(I)=B$(I) @ W(I)=V(I)

260     'NEXT': NEXT I
270 FOR I=1 TO N
280     PRINT #1;"       ENTRY   F"&STR$(I)
290     PRINT #1;"       CHAR    #F" @ NEXT I
300 FOR I=1 TO N
310     PRINT #1;"       KEY     '"&C$(I)&"'"
320     PRINT #1;"       TOKEN   "&STR$(T+I-1)
330 NEXT I
340 PRINT #1;"      ENDTXT"
350 FOR I=1 TO N @ DESTROY Z$
360     PRINT #1;"       NIBHEX 00"
370     PRINT #1;"F"&STR$(I)
380     PRINT #1;"       P=      0"
  -     Nbr:smmmmmmmmmmmeee
        s=signe:0 si nbre>0 / 9 si nbre<0
        m=12 chiffres de mantisse
        E=exponent(n),n=nbre
        eee=exposant;si n>1 e=E / si n<1 e=1000+e
390     IF W(I)<0 THEN Z$[1,1]="9" ELSE Z$[1,1]="0"
400     W(I)=ABS(W(I))
410     E=EXPONENT(W(I))
420     FIX 12 @ IF E<0 THEN Z2$=STR$(W(I)*10^(12-E)) @ Z$[2,13]=Z2$[1,1]&Z2$[3,13] @ E1=1000+E
430     IF E>=0 THEN Z2$=STR$(W(I)*10^(12-E)) @ Z$[2,13]=Z2$[1,1]&Z2$[3,13] @ E1=E
  -     Place ou non les 0 d'entête pour l'exposant
440     STD @ IF LEN(STR$(E1))=3 THEN Z$[14,16]=STR$(E1)
450     IF LEN(STR$(E1))=2 THEN Z$[14,16]="0"&STR$(E1)
```

```
460     IF LEN(STR$(E1))=1 THEN Z$[14,16]="00"&STR$(E1)
470     PRINT #1;"       LCHEX  "&Z$
480     PRINT #1;"       GOVLNG FNRTN1" @ NEXT I
490 PRINT #1;"       END"
500 END

510 'INTER':
  - Intercaler un élément.
    Décalage
520 FOR K=N TO J+1 STEP -1
530     C$(K)=C$(K-1) @ W(K)=W(K-1)
540 NEXT K
  - copie l'elt.
550 C$(J)=B$(I) @ W(J)=V(I)
560 RETURN
```

****

Programme "JULIEN" (calendrier Julien (1 janvier 45 avant J.C. au 4 octobre 1582)

```
 10 DESTROY ALL
 20 DISP "Calendrier Julien (1/1/45 av J.C <= DATE <= 4/10/1582 ap J.C)"
 30 DISP @ DISP
 40 INPUT "Jour:";J
 50 INPUT "Mois:";M
 60 INPUT "Anneé:";Y
 70 IF Y<-45 OR Y>1582 THEN 60
 80 DISP @ DISP
 90 S=INT(Y/100)
100 A=Y-100*S
110 D3=MOD(J-1,7)
120 IF M=2 OR M=3 OR M=11 THEN D2=0
130 IF M=6 THEN D2=1
140 IF M=9 OR M=12 THEN D2=2
150 IF M=4 OR M=7 THEN D2=3
160 IF M=1 OR M=10 THEN D2=4
170 IF M=5 THEN D2=5
180 IF M=8 THEN D2=6
190 D1=MOD(A+INT(A/4),7)
200 D0=MOD(6*S,7)
210 D=MOD(1+D0+D1+D2+D3,7)
220 D=D+1
230 DISP "--------"
240 ON D GOTO 250,260,270,280,290,300,310
250 DISP "Dimanche" @ GOTO 320
260 DISP "Lundi" @ GOTO 320
270 DISP "Mardi" @ GOTO 320
280 DISP "Mercredi" @ GOTO 320
290 DISP "Jeudi" @ GOTO 320
300 DISP "Vendredi" @ GOTO 320
310 DISP "Samedi"
320 DISP "--------"
```

```
**************************************************************************************************

Programme "PAQUES" (calcul des dates de Pâques)

  - Calcul des dates de Pâques
    Jean Maille
 10 DESTROY ALL
 20 DISP "Calcul des dates de Pâques par la méthode de Gauss pour une période donnée"
  - Elles sont comprises du 22 Mars au 25 Avril inclus
 40 DISP
 50 INPUT "Année de départ:";Y0
 60 INPUT "Année d'arrivée:";Y1
 70 FOR Y=Y0 TO Y1
 80     S=INT(Y/100) @ A=Y-100*S
 90     IF Y>1582 THEN 120
100     A0=15 @ B0=6
110     GOTO 140
120     A0=15+S-INT(S/4)-INT(S/3)
130     B0=MOD(4+S-INT(S/4),7)
140     R1=MOD(Y,19)
150     R2=MOD(Y,4)
160     R3=MOD(Y,7)
170     R4=MOD(19*R1+A0,30)
180     R5=MOD(2*R2+4*R3+6*R4+B0,7)
190     D=22+R4+R5
200     IF D<=31 THEN DISP "En ";Y;"  le ";D;"Mars" @ GOTO 280
210     D=R4+R5-9
220     IF R5#6 THEN 270
230     IF R4#28 THEN 250
240     IF R1>=11 THEN 260 ELSE 270
250     IF R4#29 THEN 270
260     DISP "En ";Y;"  le ";D-7;"Avril" @ GOTO 280
270     DISP "En ";Y;"  le ";D;"Avril" @ GOTO 280
280 NEXT Y
290 END
```

LE COIN DES LHEX

Comme de coutume, cette rubrique contient la liste des codes hexadécimaux des fichiers Lex parus ce mois-ci.

Rappelons ce qu'est un fichier Lex : c'est un programme pour le HP-71, en assembleur, qui apporte de nouvelles fonctions. Celles-ci sont utilisables directement, ou dans des programmes Basic.

Pour bénéficier de ces nouvelles fonctions, vous n'avez pas besoin de programmer vous-même en assembleur, ni de posséder un module Forth/Assembleur.

Il suffit de recopier le petit programme basic "MAKELEX" ci-dessous, de le lancer et de recopier les codes du fichier Lex désiré. Quand vous avez fini, les nouvelles fonctions sont accessibles, après avoir éteint et rallumé votre HP-71.

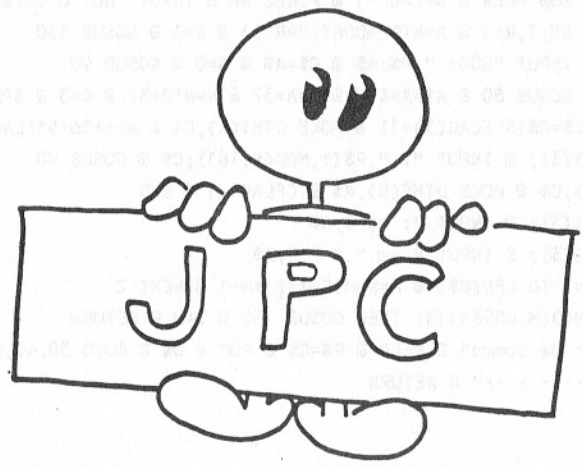
Si l'erreur "Erreur de somme" apparaît, vérifiez la ligne que vous avez introduite.

Vous trouverz donc le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !"), ainsi que le Lex de programmation structurée de ce mois-ci.

```
CHARLEX

MISCLEX   ELSTR     XWORD     92008          EL        XFN       92009
          INT$      XFN       92010          MDIM      XFN       92011
```

```
 10 CALL MLEX @ SUB MLEX @ SFLAG -1 @ PURGE AH @ INPUT "Nb. d'octets: ";N @ LC OFF
 20 CREATE DATA AH,1,N-4 @ A=HTD(ADDR$("AH")) @ B=A @ GOSUB 130
 30 Q=1 @ X=0 @ INPUT "000: ",P$;A$ @ C$=A$ @ S=0 @ GOSUB 90
 40 Q=2 @ X=1 @ GOSUB 80 @ A$=A$&C$ @ A=A+37 @ N=N*2+37 @ Q=3 @ SFLAG 5 @ FOR X=2 TO N DIV 16-1
 50 GOSUB 80 @ C$=C$[5*FLAG(5)+1] @ POKE DTH$(A),C$ @ A=A+16-5*FLAG(5,0) @ NEXT X @ Q=4
 60 DISP DTH$(X)[3]; @ INPUT ": ",P$[1,MOD(N,16)];C$ @ GOSUB 90
 70 POKE DTH$(A),C$ @ POKE DTH$(B),A$ @ CFLAG -1 @ END
 80 DISP DTH$(X)[3]; @ INPUT ": ",P$;C$
 90 DISP DTH$(X)[3]; @ INPUT " sm ","---";D$
100 M=S @ FOR Z=1 TO LEN(C$) @ M=NUM(C$[Z])+M+1 @ NEXT Z
110 IF D$=DTH$(MOD(M,4096))[3] THEN GOSUB 130 @ S=M @ RETURN
120 DISP "Erreur de somme" @ BEEP @ P$=C$ @ POP @ ON Q GOTO 30,40,50,60
130 P$="----------------" @ RETURN
```

```
CHARLEX        624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E000000000000 68D
002: 5E4001EFF0000000 9FD
003: FE00000008000001F D57
004: F31BF961400032BF 0EA
005: 38F14A11DB10AD23 484
006: 07D532BFB8FD7911 837
007: 11AD754D7A101743 BBA
008: 11014D1CB15D0000 F25
009: 71450375FF864834 2A2
00A: 5655581008355654 5F9
00B: 5810070507701724 93F
00C: 7700775070077517 C92
00D: 2077040708364545 FE0
00E: 4A30000A49724000 333
00F: 0808094A2C180814 69C
010: A464242008355455 9F6
011: 581000054C714000 D3C
012: 0C3142404C700832 098
013: 41414A70002078A0 3F0
014: 2F30000000000000 71B
015: 0000000000000000 A2B
016: 0000000000000000 D3B
017: 0000000000000000 04B
018: 0000000000000000 35B
019: 0000000000000000 66B
01A: 0000000000000000 97B
01B: 0000000000000000 C8B
01C: 0000000000000000 F9B
01D: 0000000000000000 2AB
01E: 0000000000000000 5BB
01F: 0000000000000000 8CB
020: 0000000000000000 BDB
021: 000000000000080C F06
022: 1A28080008080A2C 270
023: 180008040E340800 5B9
024: 08001E3018000000 8F3
025: 0000000000000000 C03
026: 0000000000000000 F13
027: 0000000000000000 223
028: 0201000000010200 539
029: 0000000201020000 84E
02A: 0001000100000002 B62
02B: 0102010000000000 E76
02C: 0000000000000000 186
02D: 045E755142400101 4D2
02E: 0101010000000000 7E5
02F: 0000000000000000 AF5
030: 0000070507000000 E18
031: 0000000083444C4 156
032: 44400D7901112D70 4B6
033: 050D750509700000 800
034: 0D70000000384540 B43
035: 4020014E322E3140 E97
036: 084E794142400000 1E7
037: 00000000002E4559 525
038: 3200000000000000 83A
039: 0000000000000026 B52
03A: 5556587008365556 EB1
03B: 5810083645464830 202
03C: 0832414248700024 543
03D: 5655587008345655 8A0
03E: 5810083446454830 BEF
03F: 0C3042414C700024 F44
040: 5556587008355654 2A1
041: 5810083546444830 5F0
042: 0C3142404C700025 946
043: 5455587008355455 CA0
044: 5810083544454830 FEE
045: 0C3140414C700875 350
046: 14141870000A4972 6A1
047: 40000E3159454E30 A01
048: 0C7A0F7949400024 D79
049: 5554587000084A71 0D5
04A: 40000C523A262D10 436
04B: 0424587458400875 78D
04C: 1415187000094A70 ADD
04D: 4000083544454830 E21
04E: 0C3140414C300C74 189
04F: 5655545000054C71 4E0
050: 40000             5D9

MISCLEX        335 octets

     0123456789ABCDEF sm

000: D4943534C4548502 373
001: 802E000000000000 6A2
002: 2A200C580B000000 9F9
003: F230000000000000 D24
004: 0E7000DD00AD000F 0B3
005: 410C3100FF104910 41A
006: 0F954C4354525803 78C
007: 54C490794E44542A B12
008: 07D44494D4B01FF8 EB5
009: FE7A204018F27130 23A
00A: 8FAB6308D271308D 5D9
00B: 074509FFFF5DFFF1 9B6
00C: 4A310F9E2A11FB59 D5E
00D: F235D0A0A0251551 0D7
00E: 5938F681F08F83DB 48A
00F: 0AF23068A2908D91 816
010: FB01321BB59F2251 BA7
011: 57115411328D84A8 F16
012: 081113610B1BA59F 295
013: 2AF015A097890CCC 63F
014: C81C1028FBC63156 9D9
015: 320302986C2305E2 D3E
016: 8F943B1158011B13 0B5
017: 61128FB13B1AFE8D 46E
018: 612F08D91FB08111 7F6
019: 3610B1BCD6F2AF21 BA4
01A: 56410CAF22030815 F13
01B: C08FDF8E08F2EA21 2E4
01C: 8F223B11517AF08F 67F
01D: 9418111C1544AF21 9F3
01E: 1B1368DC32F08E22 D8B
01F: 8FBC6315D417F101 125
020: 14B2039E1D1C1B1A 4BD
021: 1298F890B15323BF 850
022: 1E2D2C2B2A22B8F8 C04
023: 90B15718D6CC7185 F9A
024: 46D008D91FB08441 327
025: 11AF28A242E68A2D 6CD
026: 4E68B6FD87405703 A68
027: 017215B36E007120 DC2
028: 17115B01181378FB 138
029: 13B11CFAFE8D612F 500
02A: 01371081370171FF 864
02B: 17615B36FCFAF06E C24
02C: CFF               CF6
```



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 Baudier et Janick Taillandier, grâce à un système comprenant un HP71B, un lecteur de disquettes HP9114A, un HP9807A, deux HP9154 et une imprimante LaserJet.

Directeur de la publication : Pierre David
Numéro ISSN : 0762 - 381X

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

Imprimé par Copy-Express, 42 86 91 94.

ENGLISH SUMMARY

JPC 54 - MAY 1988

We hope you have not believed in the new HP-41 module describes last month. It was the April issue ! You have probably missed a lot of French jokes in the article...

The HP-28 column is taking more and more weight in *JPC* : it includes 5 articles this month. The first one, by Janick Taillandier on page 4 describes a very interesting behavior of the new HP-27S. If you press [CLR] and together the third menu key from the right, then release both keys and press [←] : you get a memory dump of the HP-27S. The following keys [*], [/], [+] and [-] are active and allow you to scroll through the memory. The same behavior has been noticed on the HP-17B.

Next, Philippe Heilbronn presents you a set of interesting utilities around one common subject : the use of constant factors in RPL.

After the entry point lists for the HP-28C (version 1BB), Sébastien Lalande gives us the list for the HP-28S.

The final article of this large column is a game by Paul Courbis and Sébastien Lalande. The goal : discover the purpose of the program.

The HP-75 column includes a new Lex file from Jean-Yves Hervé which allows you to exchange data with an HP-41 via a mass storage. Among the keywords, you find pointer positioning, data reading and format conversion.

The HP-71 column includes a very important article from Serge Vaudenay. It is a Lisp interpreter written in Forth. There is a very detailed introductory article about Lisp mechanisms and its implementation on the HP-71. It is not a full featured Lisp interpreter, but it includes all fundamental mechanisms of the language and so is very valuable to learn the language. It is a major event : a new language on the HP-71 !

Our Australian friend Jack Elhay (hello Jack !) presents us one of his realizations in assembly language. They are very interesting to study if you begin to write in assembly language. In any case, they will help you if you wish to control your printer.

The next article is a constant compiler from Lionel Guillou. It creates a Lex using a list of symbols and their numeric values. This increases the legibility of your Basic programs.

The last article by Jean Maille is a variation on the theme of date computations.

Until next month,

Happy Programming and JPC reading !