JPC

JUIN 1987
NUMERO 45

Le numéro 40 F

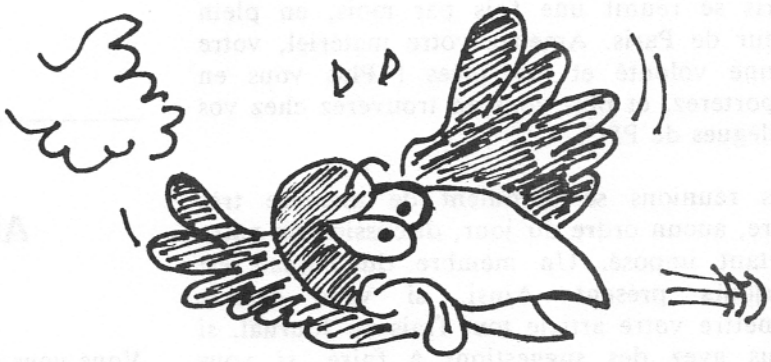
A PROPOS DU CLUB

HP28

HP41

HP71

EDITORIAL



Chers Membres,

Janick Taillandier, Jean-Jacques Dhénin et moi-même sommes heureux de vous présenter ce nouveau numéro de JPC.

Ce numéro couronne une année d'efforts. Cela fait en effet un an que nous travaillons d'arrache pied tous les trois pour que le Club réponde à vos aspirations. Je tiens à remercier ici Janick et Jean-Jacques. Sans eux, le Club ne serait pas ce qu'il est actuellement.

Le meilleur moyen de nous encourager est de nous envoyer vos articles. Vous êtes nombreux à le faire, et je tiens à vous remercier également. A l'heure où beaucoup de Clubs étrangers connaissent quelques difficultés, nous sommes un exemple de vitalité.

Je me retire sur la pointe des pieds, et vous laisse tourner la page...

Pierre David (37)

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 6 juin 1987
Samedi 5 septembre 1987
Samedi 3 octobre 1987
Samedi 21 novembre 1987
Samedi 5 décembre 1987
Samedi 16 janvier 1988
Samedi 20 février 1988
Samedi 5 mars 1988
Samedi 16 avril 1988
Samedi 7 mai 1988
Samedi 4 juin 1988

Pierre David (37)

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 en fin de journal.

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)
```

COURRIER DU COEUR

Rappelons notre nouvelle adresse :

PPC Paris Chapter
B.P. 604
75028 Paris Cedex 01

PPC-Paris
B.P. 604
75028 Paris Cedex 01

Vend :
Interfaces vidéo HP82163B (32 colonnes) neuves (dans leur boîte, avec documentation) : 600 F seulement.

Lecteurs de cartes magnétiques HP82400A pour HP-71 neufs (dans leur boîte, avec documentation et 5 cartes magnétiques) : 500 F seulement.

Des lots de 100 cartes magnétiques pour HP-41, HP-67, HP-97 et même HP-65 : 100 F seulement.

Pierre David
33 Bd St Martin
75003 Paris
Tél : (1) 48 87 68 93

Vend :
Imprimante 80 colonnes HP82905B + papier : 2500 F.

Convertisseur HP82166A : 1000 F.

Alain Goubault de Brugiere
27 avenue de Brimont
78400 Chatou
Tél dom. : (1) 39 52 56 01
Tél bur. : (1) 45 61 99 11

Vend :
HP-71B avec accessoires d'origine et manuels (01/86) + livret *HP-71 utilities* + livre *Le HP-71, c'est facile* + *IDS* Volume I : 3400 F.

Lecteur de cartes pour HP-71 (05/86) + environ 100 cartes avec programmes (Basic ou Forth) et fichiers Lex + 5 étuis rigides + classeur 3 anneaux : 600 F.

Module Translator Pac : 800 F.

ou HP-71 + lecteur : 3700 F, ou HP-71 + module : 4000 F, ou l'ensemble : 4300 F.

Prix négociables, profitez-en !

Bernard Garry
9 rue Louis Georges
92140 Clamart
Tél : (1) 47 36 24 64

Recherche :
Modules XMemory pour HP-41.

Paul Carles
10, Cité Verte
94370 Sucy en Brie
Tél : (1) 45 90 11 17

Recherche un membre du Club pour l'aider, moyennant salaire, à utiliser au mieux de ses possibilités le livret d'applications de la HP-41C intitulé *bibliothèque mathématiques*.

Recherche également un membre du Club pour l'aider à résoudre des exercices (niveau Terminale et Deug A) à l'aide de la machine.

MODULE GRAPHIQUE POUR HP-71B

La société BCMW commercialise un nouveau produit pour le HP-71B. Il s'agit d'un logiciel écrit par Pierre David permettant de faire du graphique sur imprimante ThinkJet.

Ce logiciel est entièrement écrit en assembleur, ce qui signifie :

- une très grande rapidité,

- une facilité d'utilisation sans égale : les mots-clefs sont appelables comme n'importe quelle fonction standard du HP-71B,

- et une très grande compacité du logiciel. Moins de 9 Ko !

Faire du graphique sur ThinkJet nécessitera sans doute de la mémoire additionnelle. A titre indicatif, 32 Ko suffisent pour un tiers de page. La seule limite est la taille mémoire disponible.

Les mots-clefs de ce logiciel sont :

BOX : trace un rectangle.

CSIZE : spécifie la hauteur et la largeur des caractères.

DRAW : trace un segment de droite entre la position courante et le point spécifié.

FRAME : trace un cadre autour de la zone de tracé.

GDUMP : imprime le graphique sur ThinkJet.

GEND : termine une session graphique.

GINIT : initialise une session graphique. Spécifie la dimension du tracé.

IDRAW : trace un vecteur.

IMOVE : déplace le point courant.

LABEL : trace des lettres dont la taille est spécifiée par CSIZE, la direction par LDIR et l'origine par LORG.

LDIR : spécifie l'angle de tracé des lettres par rapport à l'horizontale.

LINETYPE : sélectionne le type et la longueur des motifs tracés par DRAW ou IDRAW.

LORG : spécifie la position des lettres par rapport au point courant.

MOVE : déplace le point courant.

PENDOWN : baisse la plume, c'est à dire trace un point.

PLOTTER IS : sélectionne l'appareil graphique.

TICLEN : définit la longueur des marques de graduation sur les axes.

XAXIS et **YAXIS** : tracent des axes horizontaux et verticaux, avec des marques de graduation optionnelles.

Le logiciel est disponible sur cassette ou disquette au prix de 1950 F H.T. Il est fourni avec un manuel en français ou en anglais.

Demandez-le à votre distributeur habituel, ou renseignez-vous directement à BCMW.

BCMW
2 bis rue Nicolas Houël
75005 Paris
Tél : (1) 43 36 12 05

Janick Taillandier (246)

Post Scriptum : ce n'est pas parce que Pierre est Président du Club qu'il faut passer sous silence ses réalisations, surtout lorsqu'elles sont aussi remarquables !

HP28

D. Dalila

Utilisation de la pile opérationnelle 6

EXEMPLE D'UTLISATION DE LA PILE OPERATIONNELLE

Une des caractéristiques du HP-28C est sa pile opérationnelle. Cette pile, connue des calculateurs HP, ne comporte plus quatre niveaux mais autant que la mémoire le permet.

Cette pile illimitée nécessite un système qui puisse la gérer. Celui-ci est du type Forth : en effet, le Forth possède des instructions adaptées à la gestion de la pile telles que [DUP], [OVER], [DROP] ou encore [SWAP] et autres [ROT]...

Le programme que je vous propose, bien qu'il soit déjà intégré dans le HP-28C, montre une des manipulations possibles de la pile, qui sont l'unique base de ce programme.

Il résoud les équations $ax^2 + bx + c = 0$. Il calcule le discriminant (défini par $d = b^2-4ac$. Si cette quantité est positive ou nulle, il calcule les deux racines ou solutions $-b\pm\sqrt{d}/2a$. Si *d* est nul, il existe une seule racine -b/2a. Ce point est le sommet de la parabole dans ce cas uniquement.

La première opération est d'entrer le programme dans le HP-28C. Il est bien évident que ce programme est introduit à la file, sans les commentaires. Une fois rentré, vous appuyez sur [ENTER].

Les instructions font partie des menus suivants :
- [OVER], [DUP] et [PICK] appartiennent au menu [STACK],
- [START], [NEXT], [IF], [THEN] et [NEXT] appartiennent au menu [BRANCH],
- [HALT], [ABORT et [BEEP] appartiennent au menu [CTRL], et
- [ROLL], [SWAP] et [DROP] sont sur le clavier.

Le programme

```
«                 Début du programme
  OVER            copie niveau 3 en 1
  3 PICK          copie niveau 3 en 1
  5 PICK          copie niveau 5 en 1
  1 3             déplace 3 fois
  START           ... le niveau 6
    6 ROLL        ... au niveau 1
  NEXT
  3 PICK *        copie niveau 3 * niveau 1
  -4 *            -4 * niveau 1
  SWAP            échange niveau 1 et niveau 2
  SQ +            x^2 + niveau 2
  DUP             duplique le niveau 1
  2500 .05        signal sonore à 2500 Hz
  BEEP            ... pendant .05 secondes
  IF 0 <          si d<0
    THEN            alors
      "S=ens vide"    le programme s'arrête
      ABORT           en affichant "S=ens vide"
  END
  HALT            arrête le HP-28C
  DUP             duplique le niveau 1
  SQRT            racine carrée du niveau 1
  -1 *            -1 * niveau 1
  6 ROLL          déplace niveau 6 au niveau 1
  -               ... et niveau 2 - niveau 1
  2 /             niveau 1 / 2
  ROT             déplace le niveau 3 au niveau 1
  /               niveau 2 / niveau 1
  2500 .05        signal sonore à 2500 Hz
  BEEP            ... pendant .05 secondes
  HALT
  DROP            efface le niveau 1
  SQRT
  ROT             déplace le niveau 3 au niveau 1
  -               niveau 2 - niveau 1
  2 /             niveau 1 / 2
  SWAP /          niveau 1 / niveau 2
  2500 .05        signal sonore à 2500 Hz
  BEEP            ... pendant .05 secondes
»                 et fin du programme
```

Utilisation

L'utilisation est très simple. Le programme étant rentré dans la machine, stockez-le à l'aide de l'instruction [STO] : faire 'EUPO [STO]. Ainsi, votre programme est stocké, et vous pouvez le visualiser dans le menu [USER].

Pour l'utiliser, un exemple reste le meilleur moyen :

Nous allons prendre $2x^2 - 4x - 6 = 0$. Nous calculons le discriminant de tête : le résultat est 64. Nous en déduisons les racines : -1 et 3.

Maintenant, nous allons utiliser le programme EUPO.

L'entrée de *a*, *b* et *c* dans la pile se fait en tapant :

```
2 [ENTER]
4 [CHS] [ENTER]
6 [CHS] [ENTER]
[USER] [EUPO]
```

Le discriminant s'affiche : 64 au niveau 1. Dans le coin supérieur gauche s'inscrit le symbole octogonal d'attente. Pour continuer, vous appuyez sur [CONT]. La première racine s'affiche alors : -1. Vous appuyez sur la même touche, et la deuxième racine s'affiche : 3.

Remarques

1- Vous n'êtes pas obligé d'effacer la pile avant exécution. Le programme fonctionne tant que *a*, *b* et *c* sont placés dans les trois premiers niveaux.

2- Il ne faut jamais effacer la pile lors de l'exécution du programme. Si vous l'avez effacée, alors vous appuierez sur [ON] et [^] en même temps. Vous relacherez [ON], puis [^]. Cette séquence de touche n'efface pas vos programmes, mais réinitialise le HP-28C.

3- Les résultats se lisent sur le niveau 1.

4- Cas particuliers :

$9x^2 - 6x + 1 = 0$

Le discriminant est nul. Il existe une seule solution, soit 1/3, ou 0,3333... Sachant que le discriminant est nul et qu'il existe une seule racine, vous devrez traiter sur votre HP-28C les mêmes séquences de touches que si le discriminant était supérieur à 0. Ce cas est traité comme cas général et non comme cas particulier.

$-5x^2 + x - 1 = 0$

Le discriminant est égal à -19, il n'y a donc pas de racine. Le HP-28C s'arrête et affiche au niveau 1 : "S=ens vide". Votre calculateur est alors prêt pour un nouveau traitement.

5- Notez que les beeps peuvent être changés ou supprimés. Le test peut être supprimé, en sachant que si le résultat est négatif, le HP-28C vous donnera des résultats incohérents ou un message d'erreur.

Evolution de la pile

Prenons le premier exemple ($2x^2-4x-6 = 0$).

Actions		Pile opérationnelle niveaux : 1	2	3	4	5	6	7
entrée a, b et c		-6	-4	2				
OVER		-4	-6	-4	2			
3 PICK		-4	-4	-6	-4	2		
5 PICK		2	-4	-4	-6	-4	2	
1 3 START 6 ROLL NEXT		-6	-4	2	2	-4	-4	
3 PICK	d	2	-6	-4	2	2	-4	-4
*	i	-12	-4	2	2	-4	-4	
multiplié par -4	s	48	-4	2	2	-4	-4	
SWAP	c	-4	48	2	2	-4	-4	
SQ	r	16	48	2	2	-4	-4	
+	i	64	2	2	-4	-4		
DUP		64	64	2	2	-4	-4	
test signe d		64	2	2	-4	-4		
DUP	1	64	64	2	2	-4	-4	
racine carrée	r	8	64	2	2	-4	-4	
multiplié par -1	a	-8	64	2	2	-4	-4	
6 ROLL	c	-4	-8	64	2	2	-4	
-	i	-4	64	2	2	-4		
divisé par 2	n	-2	64	2	2	-4		
ROT	e	2	-2	64	2	-4		
/		-1	64	2	-4			
DROP		64	2	-4				
racine carrée	2	8	2	-4				
ROT	r	-4	8	2				
-	a	12	2					
divisé par 2	c	6	2					
SWAP	i	2	6					
/	n	3						

Quelques mots

Quand j'ai voulu faire ce programme, j'ai rapidement constaté que ce n'était pas facile. En effet, manipuler des nombres dans une pile nécessite un crayon et beaucoup de papier ! Il ne faut pas hésiter à dessiner le tableau ci-dessus pour trouver les erreurs et ne pas se perdre tout au long de la construction du programme.

Pour faire un programme de ce type, il faut travailler avec méthode. Si on examine le discriminant $d = b^2-4ac$ et les racines $-b\pm\sqrt{d}/2a$, on remarques que les trois nombres servent plusieurs fois. *a* et *b* interviennent 3 fois dans les calculs, alors que *c* ne sert qu'une seule fois. On en déduit qu'il faut copier deux fois *a* et *b*, contrairement à *c* qui ne doit pas être copié.

On a alors déjà fait 50 % du travail en analysant le problème.

Vous comprenez tout de suite l'utilité de OVER, 3 PICK et 5 PICK du programme. La boucle sert à ramener , *b* et *c* dans les trois premiers niveaux pour qu'ils soient prêts pour le traitement.

Remarquez que la copie de *a* se fait dans le calcul du discriminant, 3 PICK après la boucle. Pourquoi ? Si vous faites la copie de *a* au début du programme, vous aurez deux instructions supplémentaires. Conclusion, vous perdez de la mémoire, ô combien précieuse dans notre HP-28C. Vous me direz « pour deux instructions... », mais nous n'avons que 1700 octets...

Le programme duplique *d* juste avant le premier beep. Cette duplication est dûe exclusivement au test du signe de *d*. Lorsque le programme exécute un test, il efface le niveau 1 pour le tester. Le deuxième DUP est dû au fait qu'il y a deux racines. Par conséquent, il faut deux discriminants.

La multiplication par -1 peut paraître un peu bizarre. En fait, je fais un changement de signe, et normalement on devrait faire avec [CHS]. Eh bien non ! Quand vous faites un programme, et que vous avez besoin de [CHS], vous verrez que le calculateur vous affiche un - au lieu d'un changement de signe CHS. Le HP-28C vous fera une soustraction, c'est logique... seulement vos résultats seront faux (*).

Après la première racine, j'efface le niveau 1 par DROP, car nous n'en avons plus besoin. Ainsi, la pile est prête pour la suite du traitement.

Conclusion

J'espère que ce programme vous aidera à apprécier la manipulation de la pile opérationnelle. Vous verrez que travailler avec la pile est amusant et que l'exécution d'un programme est assez rapide (ce programme est 8 fois plus rapide qu'avec un HP-15C).

Ce programme satisfera, peut être, les élèves de seconde et surement les élèves de première qui utilisent souvent cette méthode de résolution, tout comme moi.

Je vous souhaite une bonne utilisation...

David Dalila

(*) : il suffit d'utiliser la fonction [NEG] du menu [REAL].

HP41

UTILITAIRES POUR MATRICES

Heureux Possesseur du module *Advantage* pour HP-41, je vous propose deux petits programmes venant compléter la panoplie des fonctions matricielles.

MATRICE IDENTITE
TRACE DE MATRICE

Le premier programme remplit une matrice avec l'identité (IDN), ou calcule la trace d'une matrice carrée (TRC).

La matrice identité est la matrice qui possède des 1 sur la diagonale et des 0 ailleurs.

1 0
est la matrice identité d'ordre 2
0 1

La trace d'une matrice carrée est la somme de ses éléments diagonaux. La trace de la matrice identité d'ordre *n* est donc *n*.

Utilisation de IDN :

Placer le nom de la matrice en alpha puis faire XEQ "IDN".

Au bout de quelques secondes, le programme vous rend la main.

Utilisation de TRC :

Placer le nom de la matrice en alpha puis faire XEQ "TRC".

Au bout de quelques secondes, la valeur de la trace est en X.

Note : ces deux utilitaires travaillent sur des matrices carrées et l'erreur DIM ERROR est renvoyée dans le cas contraire.

POINTEURS DE MATRICES

Le deuxième programme permet de décomposer un pointeur (IJXY) ou de le composer (XYIJ).

Utilisation de IJXY :

Placer en X le pointeur (*iii,jjj*), puis faire XEQ "IJXY". L'utilitaire retourne le code ligne (*iii*) en Y, le code colonne (*jjj*) en X.

Utilisation de XYIJ :

Placer en X le code colonne (*jjj*), en Y le code ligne (*iii*). Faire ensuite XEQ"XYIJ", et X contient le pointeur (*iii,jjj*).

Voilà, j'espère que ces quelques petits utilitaires vous satisferont.

Christophe Gottheimer (202)

```
01*LBL "IJXY"
INT  LASTX  FRC  E3  *  RTN

08*LBL "XYIJ"
E3  /  +  END

01*LBL "IDN"
XEQ 02

03*LBL 00
MRIJ  IJXY  X#Y?  CLX  X=Y?  1  MSR+  FC?C 10
GTO 00  RTN

14*LBL "TRC"
XEQ 02

16*LBL 01
MR  +  I+  FC? 10  XEQ "J+"  FC?C 10  GTO 01  RTN

25*LBL 02
DIM?  XEQ "IJXY"  X#Y?  MDET  CLST  MSIJ  END
```

FORTH

ASSEMBLEUR

BASIC

LE COIN DES LHEX

CATALOGUE DES MOTS ET UTILITAIRES

Le but est de lister les mots du dictionnaire utilisateur, en dissociant les mots vocabulaires des autres. Quelques utilitaires facilitant l'accès à Basic ont été ajoutés.

PRINCIPE DES VOCABULAIRES

Dans le dictionnaire, créez un vocabulaire : VOCABULARY TOTO, puis un mot : HOP ;. Lorsque vous tapez TOTO, ce dernier devient vocabulaire de contexte, c'est à dire que seuls les mots appartenant à ce vocabulaire sont reconnus, ainsi que ceux définis avant TOTO. Ici par exemple HOP ne sera pas reconnu. L' apparition d'une erreur redéfinit FORTH comme vocabulaire de contexte et courant.

Un vocabulaire est dit courant lorsque la compilation d'un mot l'ajoute à l'ensemble des mots du vocabulaire. Les vocabulaires de contexte et courant peuvent différer. Pour qu'un vocabulaire de contexte devienne courant, il faut entrer : TOTO DEFINITIONS. Le TOTO spécifie que le vocabulaire est devenu contexte. DEFINITIONS égale la valeur de CURRENT @ à celle de CONTEXT @. Maintenant on peut créer le mot : HOP ;. HOP est une version différente de la première car elle se trouve dans un autre vocabulaire. Pour d'autres informations, voyez le manuel au chapitre vocabulaire.

ET NOUS, ET NOUS...

Lister les mots ne peut se faire qu'en fonction du vocabulaire de contexte. L'algorithme de base est relativement simple. Chaque entrée au dictionnaire contient entre autres un NFA et un LFA . Le LFA (Link Field Address) est le lien qui relie ce mot à celui defini précédemment. Son contenu est l'adresse NFA du mot précédent. NFA (Name Field Address) est le nom du mot agrémenté de quelques petites choses. Le premier octet du NFA est assez spécial puisque seuls les 5 bits de poids faible contiennent la taille du nom (2^5 = 32 lettres maximum). En ce qui concerne le reste du codage, voyez votre livre préféré.

Quant au rangement en mémoire, on a pour un mot donné la séquence LFA / NFA / CFA / PFA, donc pour un même mot LFA = NFA - 5. Bref, ce que nous désirons c'est décrypter le mot dont le NFA est connu, repérer le NFA précédent grâce au LFA présent et recommencer la boucle.

AU COEUR DE L'ARBRE

Le problème est de découvrir le NFA du dernier mot sous un vocabulaire donné (vocabulaire de contexte). Or le mot LATEST équivaut à CURRENT @ @ et ce n'est pas ce qu'on veut. TOTO ou FORTH sont à la fois des entrées au dictionnaire et des vocabulaires. FORTH, en particulier, a son LFA nul car il est le premier mot dans la chaîne. Lorsqu'on précise FORTH DEFINITIONS, on remarque que 9 quartets séparent le CFA de FORTH de la valeur CONTEXT @.

Un codage typique d'un mot vocabulaire est le suivant :

LFA, NFA, CFA = E7160, PFA = 81/A0/ *ct*

ct est le NFA du dernier mot entré dans ce vocabulaire. A cet endroit pointe la variable CONTEXT, et à fortiori la variable CURRENT si le vocabulaire est courant.

LES UTILITAIRES

De façon empirique, il a été possible de calculer le nombre de quartets disponibles dans le dictionnaire, c'est parfois utile.

Une fonction «simple» qui remplace le trop long " EDTEXT XXX,T" BASICX où XXX,T est un paramètre possible et enfin, une fonction qui permet de remplacer " " BASICX. Ces mots ($ et %) se servent de l'interprèteur, TIB, WORD, et autres. S'ils n'avaient pas «tourné» du premier coup, je n'aurais jamais tenté de les mettre au point. En effet, mes connaissances en ce domaine sont minces. Mais ne désespérons pas, peut-être que quelqu'un saura donner l'explication à ce mystère.

MODE D'EMPLOI

CATALOG (--)
Liste les mots du dictionnaire en commençant par les derniers définis. CATALOG ne liste pas au-delà du mot vocabulaire auquel appartient le reste. Autrement dit, CATALOG liste le vocabulaire de contexte, sans prendre en compte ce qui précede le mot vocabulaire.

Les touches flêches ([^], [v]) permettent de se déplacer dans la chaine. [ON] stoppe le processus. Codification :

w. : signifie simple mot.
* : mot protégé par FENCE.
voc.: si le mot représente un vocabulaire.
ct. : si vocabulaire de contexte.
cu. : si vocabulaire courant.

MEM (-- n)
Renvoie *n*, le nombre de quartets disponibles dans le dictionnaire.

Xavier Bille (203)

```
HEX

( décrypte un mot sachant son NFA                )

: PR-WORD ( NFA -- )
  DUP COUNT 1F AND TYPE SPACE
  DUP 5- FENCE @ <
  ( mot protégé ? )
  IF ." * " ELSE 2 SPACES THEN
  1 TRAVERSE 2+ DUP @ E7160 =
  ( est-ce un vocabulaire,  contexte, courant ?  )
  IF
    ." voc: " 9 + DUP
    CONTEXT @ = IF ." ct. " ELSE 4 SPACES THEN
    CURRENT @ = IF ." cu. " ELSE 3 SPACES THEN
  ELSE
    DROP ." w."
  THEN ;

: CATALOG ( -- )
  ( pas d'interruption, sauvegarde de la pile    )
  F 2F441 N!
  S0 >R SP@ SP0 !
  CONTEXT @ @
  ( on remonte la chaine à partir du dernier NFA )
  BEGIN
    DUP PR-WORD KEY
    CASE
      E OF SP! R> SP0 ! 0 2F441 ! -1
        ENDOF CR ( [ON] )
      12 OF CONTEXT @ OVER <
          IF DUP 5- @ THEN 0
        ENDOF ( [^] )
      13 OF DEPTH 1 >
          IF DROP THEN 0
        ENDOF 0 SWAP ( [v] )
    ENDCASE ( autre touche, on ne fait rien    )
  UNTIL ;

( renvoie le nombre de quartets libres           )

: MEM ( -- n )
  SP@ 1C1 - HERE - ;

DECIMAL

( Syntaxe :                                      )
( % 'ligne Basic'                                )
(   par exemple :                                )
(   % BEEP 200,.5 <=> " BEEP 200,.5" BASICX      )
( $ filename,commande                            )
(   $ TOTO,12,T <=> " EDTEXT TOTO,'12,T'" BASICX )
( % et $ se servent de l'interprèteur            )

: % 13 WORD COUNT BASICX ;

: $ 32 WORD COUNT " EDTEXT" 2SWAP S>& BASICX ;
```

L'ASSEMBLEUR DU HP-71

Comme promis, voici donc un article d'initiation à l'assembleur. Il est destiné à ceux d'entre nous qui voudraient bien utiliser ce langage, mais qui ne le font pas encore.

Tous les membres de PPC, en tout cas ceux qui possèdent un HP-71 connaisent la puissance de cette machine. Excellent Basic, très bon module mathématique, et surtout confort d'utilisation inégalé. Mais ceux qui pratiquent l'assembleur, même un peu, vous diront que ce langage apporte encore bien plus de choses qu'on ne peut imaginer.

Mais au fait ? Qu'est-ce que l'assembleur ?

Le langage assembleur est la représentation du langage machine qui est le seul compréhensible par le microprocesseur. Le seul problème est que la machine ne comprend que des zéros ou des uns, et qu'une instuction est une suite de ces deux symboles binaires. Pour l'homme, le langage binaire n'est pas très parlant. Alors, on préfère utiliser le langage asssembleur. Chaque instruction en langage binaire correspond à un mot, dit *mnémonique*, qui est le nom de l'instruction ou une suite de caractères permettant d'identifier l'instruction. Par exemple, le mnémonique de l'instruction qui permet d'ajouter 1 au registre C est : C=C+1 .

Le programme que l'on écrit est donc une suite d'intructions placée dans un fichier texte. Le fichier obtenu après écriture du programme (on l'appelle *fichier source*) doit être traduit pour être compris par le microprocesseur. Cette traduction est réalisée par un programme dit *programme d'assemblage* ou *assembleur*, qu'il ne faut pas confondre avec le nom du langage. Un tel programme est indispensable pour faire de l'assembleur, à moins de faire l'assemblage à la main (bon courage !).

Sur HP-71, il faut utiliser le programme contenu dans le module Forth / Assembleur.

Maintenant, je vais tenter d'expliquer ce qui fait pour moi l'intérêt et la puissance de l'assembleur du HP-71. Tout d'abord, il faut dire que cet assembleur est très particulier, comme tout ce que fait Hewlett-Packard en général. Le processeur du HP-71 a été d'ailleurs développé spécialement. Il a été conçu pour effectuer des calculs mathématiques.

Ensuite, je dirai que la mémoire morte de base du HP-71 apporte énormément à la puissance de l'assembleur. Les programmes que l'on écrit peuvent appeler des routines de cette Rom système. Hewlett-Packard a vraiment réalisé un chef-d'oeuvre en écrivant cette Rom. Les routines qui la composent sont puissantes et pourtant souples et simples à employer.

Enfin, il faut signaler qu'une bonne part de la puissance de l'assembleur est dûe au fait qu'on peut se procurer chez Hewlett-Packard toute la documentation possible et souhaible à propos du HP-71. C'est avec cette documentation, plus connue sous le nom d'*I.D.S.* (pour *Internal Design Specification*), que l'on peut réellement tirer parti de la puissance de notre brave machine.

En fait, il existe une autre raison à la puissance de cet assembleur. Je parle de vous, les membres de PPC. En effet ce sont les routines écrites en assembleur par les membres de PPC qui feront la richesse et la puisance de ce langage. C'est donc à vous de jouer maintenant.

Mais au fait ? Pourquoi faire de l'assembleur ?

Eh oui, c'est vrai ça ! On pourrait très bien ce contenter de la puissance du Basic. Généralement, l'assembleur a plusieurs avantages sur les langages évolués : rapidité et possibilité d'avoir le contrôle total de la machine. En fait le HP-71 est conçu de telle manière qu'on puisse écrire en assembleur de nouvelles fonctions et de nouveaux ordres pour enrichir le vocabulaire du Basic et du Forth.

Voilà pour aujourd'hui. J'espère être parvenu à vous inciter à tenter la grande aventure de l'assembleur, sans pour autant décevoir ceux qui s'attendaient à ce que je rentre plus en détail dans ce langage. Mais ceci sera l'objet d'un prochain article dans les colonnes de votre Journal préféré.

Je vous dis donc à bientôt et j'attends vos avis sur mon article car, sans eux, je croirai que je n'intéresse personne avec ma prose. N'hésitez pas à me critiquer ou à donner votre avis sur la question. Ecrivez ou appelez-moi, ou bien écrivez au Club. N'oubliez pas que je compte sur vous...

Jacques Baudier (192)

Jacques Baudier
4, impasse Daniel René
78800 Houilles
Tél : (1) 39 68 33 93 (Week End)

ENCORE LES MEMOIRES DE MASSE

Voici quelques nouveautés dans le domaine du Lex pour HP-71. Oh ! Cela n'est pas grand chose devant les monstres de si grande qualité que sont FINPUT ou VECTLEX (et bien d'autres), mais je pense très humblement que cela pourra rendre quelques bons services à tous les bricoleurs de mémoire de masse de notre cher Club d'utilisateurs.

Le fichier Lex RWLEX contient la fonction RREC$ et l'ordre WREC permettant respectivement de lire et d'écrire sur les mémoires de masse.

RREC$

Syntaxe :

```
RREC$ ( <No enreg.>, <Spécif.d'appareil> )
```

Cette fonction permet la lecture directe d'un enregistrement sur une mémoire de masse. Les enregistrements sont de longueur fixe (256 octets).

Exemple :

```
10 DIM A$[256]
20 A$=RREC$(14,":TAPE(2)")
```

En ligne 10, on dimensionne la variable A$ à 256 octets. C'est le minimum pour la valeur renvoyée par RREC$ (256 octets).

En ligne 20, on charge l'enregistrement 14 de la deuxième mémoire de masse de la boucle HP-IL dans A$.

Le premier paramètre de la fonction est le numéro de l'enregistrement. Ce paramètre peut-être numérique (décimal) ou alphanumérique (hexadécimal). Dans les deux cas, si l'adresse n'est pas une adresse valide, l'erreur `Invalid Arg` est retournée. Si l'adresse est valide mais qu'elle est trop grande par rapport au nombre d'enregistrements du support, l'erreur `Size of File` est renvoyée.

Les enregistrements sont numérotés de 0 à N.
Pour les cassettes : 0 à 511 ("0" à "1FF")
Pour les disquettes double face : 0 à 2463 ("0" à "99F")
Pour les disquettes simple face : 0 à 1055 ("0" à "41F")

WREC

Syntaxe :

```
WREC <ASCII>, <No.enr.>, <Spéc. appareil>
```

Cet ordre permet l'écriture directe d'un enregistrement sur une mémoire de masse. La chaîne ASCII (premier paramètre) doit impérativement être de longueur égale à 256 octets.

Exemple :

```
WREC RPT$(CHR$(255),256),2, :1
```

On écrit des caractères de code décimal 255 (#FF en hexadécimal) dans l'enregistrement numéro 2 de la mémoire de masse. Cela permet de ré-initialiser très rapidement un support.

Le premier paramètre est la valeur à écrire dans l'enregistrement. Cette valeur doit impérativement être de longueur égale à 256 octets.

Les deuxième et troisième paramètres sont respectivement les mêmes que les deux paramètres de la fonction RREC$.

Voilà, ce n'est pas grand chose mais j'espère que ce Lex vous rendra beaucoup de petits services.

Heureuse Programmation

Michel Martinet (12)

```
      LEX   'RWLEX'
      CON(2) #E1
      CON(2) 94
      CON(2) 95
      CON(5) 0
      NIBHEX F
      REL(4) 1+(TXTBST)
      CON(4) 0
      CON(5) 0
************************************************
* RREC$(<num.enr.>,<spec. appareil>)
************************************************
      CON(3) (TXT001)-(TXTBST)   RREC$
      REL(5) RREC
      CON(1) 15
************************************************
* WREC <chaîne>,<num.enr.>,<spec. appareil>
************************************************
      CON(3) (TXT002)-(TXTBST)   WREC
      REL(5) WREC
      CON(1) 13
TXTBST
TXT001 CON(1) 9
      NIBASC 'RREC$'
      CON(2) 94        token 94
TXT002 CON(1) 7
      NIBASC 'WREC'    token 95
      CON(2) 95
TXTBEN NIBHEX 1FF
ADDRCK EQU    #1C5A5
ARGERR EQU    #0BF19
ATNFLG EQU    #2F442
Attn   EQU    12
BSERR  EQU    #0939A
CHKMAS EQU    #425C
COLLAP EQU    #091FB
COMCK+ EQU    #032AE
CONVUC EQU    #152AA
CSRW5  EQU    #0ED2C
D=AVMS EQU    #1A460
DEVPAR EQU    #1BF0
DRANGE EQU    #1B076
DVCSPp EQU    #7925
EXPEX- EQU    #0F178
EXPEXC EQU    #0F186
EXPPAR EQU    #03FD9
EXPR   EQU    #0F23C
EXPRDC EQU    #05922
FIXDC  EQU    #05493
GETERR EQU    #6791
GETMBX EQU    #3B62
GETPIL EQU    #6E0B
GNXTCR EQU    #03064
HEXASC EQU    #17148
I/OFND EQU    #118BA
ID     EQU    #E1
IVEXPe EQU    #02E35
LEXPIL EQU    #FF
MEMERR EQU    #0944D
MINTK  EQU    94
MTHSTK EQU    #2F599
NXTSTM EQU    #08A48
OUTBY+ EQU    #02CE5
PACKd  EQU    #7B4A
POP1S  EQU    #0BD38
R3=D10 EQU    #03526
R<RSTK EQU    #014DD
RANGE  EQU    #1B07C
READR# EQU    #44FF
REV$   EQU    #1B38E
REVPOP EQU    #0BD31
RNDAHX EQU    #136CB
RSTK<R EQU    #014A8
SNAPBF EQU    #2F7F0
STRGCK EQU    #036BA
STRNGP EQU    #0379D
SYNTXe EQU    #02E2B
WRITE# EQU    #453F
bLEX   EQU    #BFC
eABORT EQU    4
eNODEF EQU    3
ePIL   EQU    2
eTAPE  EQU    1
eXWORD EQU    #23
reclen EQU    512
sSTK   EQU    7
************************************************
* W R E C
************************************************
* Routine d'analyse (parse) de l'ordre WREC
WRECp  GOSBVL STRGCK   1er paramètre : alpha
       GOSBVL COMCK+
       GONC   syntxe
       GOSBVL GNXTCR
       GOSBVL R3=D10
       GOSBVL EXPPAR   2ème paramètre: alpha
        *              ou numérique.
       ?ST=1  0
       GOYES  ivexpe
       GOSBVL COMCK+
       GONC   syntxe
       GOSUB  JUMPER
       CON(5) DVCSPp   3ème paramètre: spécifi-
        *              cateur d'appareil HPIL.
       RTNCC
```

```
ivexpe ST=1   4
       GOVLNG IVEXPe
syntxe GOVLNG SYNTXe

* Routine de décompilation de l'ordre WRECd
WRECd  GOSBVL EXPRDC   1er paramètre
       LCASC  ','
       GOSBVL OUTBY+
       GOSBVL EXPRDC   2ème paramètre
       LCASC  ','
       GOSBVL OUTBY+
       GOSUB  JUMPER
       CON(5) PACKd    3ème paramètre: HPIL
       RTN

ERReur GOTO   erreur

       REL(5) WRECd
       REL(5) WRECp

WREC   GOSBVL EXPEX-   ) Dépose les 2 premiers
       D0=D0+ 2        ) paramètres
       GOSBVL EXPEXC   ) de l'ordre
       D0=D0+ 2        ) WREC sur la M.S.
       CD1EX
       RSTK=C          RSTK = Math Stack
       ST=0   sSTK
       GOSUB  JUMPER
       CON(5) GETPIL   Cette routine permet
        *              d'évaluer le spécificateur
        *              d'appareil.
       GOC    ERReur
       C=RSTK
       D1=C            D1 = Math Stack
       GOSUB  JUMPER
       CON(5) GETMBX   Routine de recherche de
        *              l'adresse de la MAILBOX de
        *              la boucle décrite dans le
        *              spécificateur d'appareil.
        *              Au retour: D0 ^ M.B.
       GOSUB  VERT10   Vérification de la présence
        *              de la mémoire de masse sur
        *              la boucle.
       A=R0            ) Mise en place de D1 sur
       D1=A            ) le deuxième paramètre de
        *              ) l'ordre WREC
       C=D    A        Sauvegarde de l'adresse de
       RSTK=C          la mémoire de masse dans
        *              la pile de retour
       A=DAT1 B        ) l'adresse d'écriture est-
        *              ) elle en HEXA ?
       LC(2)  15       )
       ?A#C   B        )
       GOYES  rndahx   Non => RNDAHX
       GOSBVL ADDRCK   Vérification de la validité
        *              de l'adresse (2e param.)
       C=B    A
       GOC    EXEC00

rndahx GOSBVL RNDAHX
       GONC   ivarg
       D1=D1+ 16
       C=A    A

EXEC00 RSTK=C          ) adresse dans la pile de
        *              ) retour
       GOSBVL REVPOP   Renversement de la chaine
        *              sur la Mathstack.
       C=0    A        ) La longueur de la chaine
       LC(3)  reclen   ) est-elle de 256 octets ?
       ?A#C   A
       GOYES  ivarg    Si non => ERREUR
       C=RSTK          ) Restitution de l'adresse
       A=C    A        ) de l'enregist. dans A(A)
       C=RSTK          ) Mise en place de
        *              ) l'adresse de la mémoire
       D=C    A        ) de masse dans D(X).
        *
       GOSUB  JUMPER       ******************
       CON(5) WRITE#       * E C R I T U R E *
        *                  ******************
       GOC    erreur   Au retour de la
        *              routine, si Cy=1,
        *              nous sommes en présence
        *              d'une erreur.
       GOSBVL COLLAP   Remise en place de MTHSTK
       GOVLNG NXTSTM   BASIC continu
ivarg  GOVLNG ARGERR

************************************************
* La routine VERTAP verifie si le paramètre
* pointé par D1 correspond bien à une mémoire
* de masse d'identificateur appareil 16.
************************************************

Dvnfnd P=      ePIL     Classe d'erreur 32 à 47
       C=0     A        Erreur 32.
       GOC     erreur   B.E.T.

VERTAP GOSUB   JUMPER
       CON(5)  DEVPAR   Routine d'analyse du
        *               spécificateur d' appareil
        *               de la fonction. En sortie
        *               D1 ^ Math Stack - 16 pour
        *               un paramètre numérique.
       GOC     erreur   Si carry: ERREUR
VERT10 ?D=0    X        D[X] contient l'adresse du
        *               périphérique. Si D[X] = 0,
       GOYES   Dvnfnd   le périphérique n'est pas
        *               présent. On renvoie
        *               "Device Not Found"
       CD1EX            ) Sauvegarde de D1 dans R0
        *               ) pour le retour
       R0=C             ) à BASIC (fin).
       GOSUB   JUMPER   Vérification de l'AID du
       CON(5)  CHKMAS   périphérique (16).
       RTNNC            Cy=0 : OK, Cy=1 : ERREUR
```

```
*************************************************
* Traitement des erreurs HPIL. P indique la
* classe d'erreur (mémoire de masse... etc.) et
* C[0] le numéro de l'erreur. Le retour à la
* main loop se fait via BSERR.
*************************************************
erreur ?P=    eTAPE
       GOYES  ERROR1
       ?P=    ePIL
       GOYES  ERROR1
       ?P#    eABORT
       GOYES  ERROR0
       GOSUB  JUMPER
       CON(5) GETMBX
       GOSUB  atnchk
       GOC    ERROR0
       GOSUB  JUMPER
       CON(5) GETERR
       GONC   ERROR-
       ?P#    eABORT
       GOYES  ERROR1
ERROR- P=     eABORT
ERROR0 C=P    0
       P=     eNODEF
ERROR1 C=P    1
       P=     2
       LCHEX  FF
       A=C    A
       P=     0
bserr  GOVLNG BSERR    Envoyez l'erreur !
Erreur GOC    erreur   Ralonge

*************************************************
* R R E C $
*************************************************
       NIBHEX CC22     Alpha ou un numérique pour
        *              les deux paramètres.
RREC   CD0EX           ) Sauvegarde de D0 dans
       RSTK=C          ) la pile de retour
       GOSUB  r<rstk   Sauvegarde de 5 niveaux
       GOSUB  VERTAP   Vérification mém. de masse
       A=R0            R0 = MS-16
       D1=A
       D1=D1+ 16       D1 = M.S.
       C=D    A        ) D[X] = device address
       RSTK=C          ) RSTK = D[X]
       A=DAT1 B        L'adresse est-elle
       LC(2)  15       donnée sous forme de
       ?A#C   B        chaîne ?
       GOYES  rndah2   Non => RNDAHX
       GOSBVL ADDRCK   Vérification validité
       GOC    EXEC10
rndah2 GOSBVL RNDAHX
       GONC   Ivarg
       D1=D1+ 16
       B=A    A
EXEC10 AD1EX
       C=0    A
       LC(3)  (reclen)+16  ( 210H = 528D
        *                  (Soit la longueur de
        *                ( l'enregistrement plus
        *               ( 16 quartets d'en-tête)
       A=A-C  A        A[A] = MS-528
       GOSBVL D=AVMS
       C=A    A        C[A] = AVMEMS
       ?C<D   A        Y a-t-il assez de mem. ?
       GOYES  memerr   Non: Erreur
       R1=C            Sauvegarde de MS-528
       D1=C            D1 = Math Stack - 512
       D1=D1+ 16
       A=B    A        A[A] = # enreg. à lire
       C=RSTK          ) Restitution de
       D=C    A        ) dev. add. dans D[X]
        *
       GOSUB  JUMPER      ****************
       CON(5) READR#      * L E C T U R E *
        *                 ****************
       GOC    Erreur
       A=R1            A[A] = MS-528
       D1=A
       C=0    W        ) construction e l'en-tête
        *               ) de la chaîne avant le
        *               ) retour à BASIC
       LCHEX  2000F    ) C(W)= 000000000002000F
       DAT1=C W        O.K.
       AD1EX
       R1=A
       P=     4        ) Restitution de 5 niveaux
       GOSBVL RSTK<R   ) de pile
       C=RSTK          ) puis de
       D0=C            ) D0  (4 niveaux & PC)
       C=R1            Restitution pointeur
       D1=C            D1 = MS+528
       GOSBVL REV$     On retourne l'ordre des
        *              caractères sur la M.Stack.
       GOVLNG EXPR     Retour à BASIC.
Ivarg  GOTO   ivarg
memerr GOVLNG MEMERR

*************************************************
* ROUTINE R<RSTK
* Cette routine sauvegarde 5 niveaux de pile
*************************************************
r<rstk P=     4
       GOVLNG R<RSTK   I N D I S P E N S A B L E !
        *              Pour l'exécution d'une
        *              fonction faisant appel à
        *              des routines HPIL.
*************************************************
* ROUTINE ATNCHK
* Cette routine a été pompée dans les IDS V, elle
* n'est pas supportée. Elle permet la sortie
* de l'erreur Aborted lorsque l'on arrête l'exécu-
* tion des routines HPIL en appuyant deux fois sur
* la touche ATTN.
*************************************************
```

```
atnchk ?ST=0  Attn
       GOYES  ATNCHc
       RSTK=C
       CDOEX
       DO=(5) ATNFLG
       C=DATO S
       DO=C
       C=RSTK
       ?C=0   S
       GOYES  ATNCHc
       C=C+1  S
       GOC    ATNCHc
       P=     eABORT
       RTNSC
ATNCHc RTNCC
***********************************************
* Routine Jumper:
* Cette routine permet l'accès aux différents
* points d'entrées du module HPIL
***********************************************
JUMPER RSTK=C
       CD1EX
       D1=(5) SNAPBF
       DAT1=C A
       D1=(2) (SNAPBF)+5
       C=RSTK
       DAT1=C W
       D1=(4) (SNAPBF)+21
       DAT1=A W
       D1=(2) (SNAPBF)+37
       C=B    A
       CPEX   5
       P=     6
       C=0    P
       GONC   JUMP05
       C=C-1  P
JUMP05 P=     7
       C=0    P
       C=C-1  P
       DAT1=C 8
       SETHEX
       P=     0
       LC(3)  bLEX
       GOSBVL I/OFND
       GONC   JUMP90
       LC(2)  LEXPIL
       B=C    A
       A=0    A
       A=A+1  A
JUMP10 C=DAT1 6
       ?C=0   B
       GOYES  JUMP90
       ?B#C   B
       GOYES  JUMP20
       CSR    W
       CSR    A
       ?A<C   B
       GOYES  JUMP20
       CSR    A
       CSR    A
       C=C-A  B
       GONC   JUMP30
JUMP20 D1=D1+ 11
       GONC   JUMP10
JUMP90 LC(4)  eXWORD
       GOTO   bserr
JUMP30 D1=D1+ 6
       C=DAT1 A
       B=C    A
       C=RSTK
       D1=C
       D1=D1+ 5
       CD1EX
       RSTK=C
       C=DAT1 A
       C=C+B  A
       RSTK=C
       D1=(5) (SNAPBF)+21
       A=DAT1 W
       D1=D1+ 16
       C=DAT1 8
       B=C    A
       P=     7
       C=C+1  P
       GOC    JUMP50
       SETDEC
JUMP50 P=     6
       ?C#0   P
       GOYES  JUMP60
JUMP60 P=C    5
       D1=(4) (SNAPBF)+5
       C=DAT1 W
       D1=(2) SNAPBF
       RSTK=C
       C=DAT1 A
       D1=C
       C=RSTK
       RTN
       END
```

TROUVEZ VOS CHAINES (ACTE II)

Si vous utilisez le Basic du HP-71B, vous avez certainement pesté plus d'une fois contre cette maudite ligne que vous n'arrivez plus à retrouver dans ce gigantesque programme sur lequel vous travaillez depuis fort longtemps...

En effet, vous ne pouvez vous déplacer dans un programme Basic que par les touches de curseur et par la commande FETCH qui positionne sur un numéro de ligne donné.

Vous vous êtes sûrement dit plus d'une fois qu'il serait bien que FETCH, suivi d'un argument alphanumérique, cherche une chaîne plutôt qu'un label. Le HP-75 ne dispose pas des labels, donc sa commande FETCH cherche une chaîne alphanumérique dans un programme.

Dans JPC 31 (février 1986), Jean-Jacques Moreau avait publié le source de son ordre FIND, destiné à combler cette lacune.

Malheureusement, cet ordre souffrait de quelques défauts, ce qui m'a amené à reprendre ce Lex, à le modifier et vous le présenter. Mais je pense qu'il n'est pas inutile de rappeler l'utilisation de cet ordre.

UTILISATION DE FIND

La syntaxe est très simple :

FIND *chaîne alphanumérique*

Par exemple : FIND "GOTO 10" cherche la chaîne de caractères "GOTO 10" dans le programme Basic courant.

Le domaine de recherche commence à la ligne suivant la ligne courante pour se terminer à la dernière ligne.

Par exemple, si votre programme est :

```
10 BEEP
20 DISP "A"
30 IF A>10 THEN DIM A$[20]
40 DISP "B"
50 GOTO 10
```

Faites FETCH 20 pour vous positionner à la ligne 20. Faites ensuite FIND "DISP". Le HP-71 vous affiche la ligne 40. Le domaine de recherche était situé entre les lignes 30 et 50.

Si vous refaites FIND "DISP", le HP-71 vous répondra cette fois :

```
ERR: Not Found
```

Notez que la recherche ne prend pas en compte les numéros de ligne. FIND "20 DISP" ne trouvera aucune occurence de la chaîne.

LES DIFFERENCES AVEC L'ANCIENNE VERSION

Comme je l'ai dit plus haut, la version originale de Jean-Jacques souffrait de quelques défauts de jeunesse. Voici les principales modifications :

- FIND s'arrêtait parfois sur des lignes où il n'y avait pourtant aucune occurrence de la chaîne recherchée. Par exemple, essayez le programme suivant :

```
1000 ! Test FIND
1010 IF ("A">F$ OR F$>"Z") AND A THEN 1020
1020 F1$="0"
```

Le programme doit être entré exactement comme il est listé, en respectant les numéros de lignes. Exécutez ensuite FIND "F$" deux fois à partir du début du fichier. S'il est normal de s'arrêter sur la ligne 1010, ça l'est beaucoup moins sur la ligne 1020. Ceci résultait d'un mauvais calcul de la longueur de la chaîne à analyser.

- Le Lex d'origine prévoyait d'afficher le messsage "Not Found" en cas d'échec de la recherche. Malheureusement, sa longueur n'est pas un multiple de 16 quartets, ce qui est obligatoire pour le premier message d'un Lex. Ceci fait que ce message ne pouvait être affiché.

- Enfin, lorsque la chaîne cherchée était trouvée, le curseur restait positionné en début de ligne ; il aurait été préférable de le voir sur le début de la chaîne trouvée.

Mon intervention s'est limitée à réaliser ces modifications en touchant très peu à la structure générale du programme. La seule ombre au tableau est qu'il n'est pas possible d'avoir un message d'erreur clair respectant la règle du premier message rappelée plus haut. J'ai décidé d'utiliser le message numéro 232 du HP-71 (" Not Found"). Certes, le caractère espace en début de message n'est pas très idéal, mais le résultat est clair. Dans JPCLEX, le même problème ne se pose pas et on utilise le message numéro 225002 "Not Found".

Pour ce qui est du reste, je vous invite à comparer les deux versions.

Voilà, je pense vous avoir tout dit. Amusez-vous bien, et trouvez vos chaînes...

Janick Taillandier (246)

```
       LEX    'FINDLEX'
************************************************
* FIND <chaine-alphanumerique>
* permet de trouver, dans un fichier Basic la
* premiere occurence de la chaine specifiee
* apres la ligne courante.
* Historique:
*  01/86 J.J. Moreau
*        conception et premiere version
*  12/86 J. Taillandier
*        - debogage
*        - suppression du message d'erreur dans
*          cette version "autonome"
*        - positionement du curseur sur la
*          chaine trouvee
************************************************
        ID      #E1
        MSG     0
        POLL    0
        ENTRY   Find
        CHAR    #7
        KEY     'FIND'
        TOKEN   75
=AVE=D1 EQU   #18BB8
=BF2DSP EQU   #01C0E
=BLDDSP EQU   #01898
=BSERR  EQU   #0939A
=CK"ON" EQU   #076AD
=CSLW5  EQU   #0ED3D
=CSRW5  EQU   #0ED2C
=CURRL  EQU   #2F7E8
=CURSRT EQU   #096C1
=DSPCNO EQU   #09716
=EXPEXC EQU   #0F186
=FINDL  EQU   #0FFE4
=GETPRO EQU   #06BEE
=GETSTC EQU   #07726
=LDCM10 EQU   #04F6F
=LDCOMP EQU   #04F69
=LDCSPC EQU   #2F6C1
=MAIN30 EQU   #0037E
=MFERR  EQU   #09393
=NULLP  EQU   #07999
=NXTLIN EQU   #10031
=NXTSTM EQU   #08A48
=OUTBS  EQU   #2F58F
=REVPOP EQU   #0BD31
=S-R1-2 EQU   #2F88B
=S-R1-3 EQU   #2F890
=STRNGP EQU   #0379D
=eFTYPE EQU   #0003F
=eNFOUN EQU   232
        ENDTXT

 Strngp GOVLNG =STRNGP

        REL(5) Strngp   pas de decompile, FIND
          *             est non programmable

Find    GOSBVL =EXPEXC  depose "search" sur la MS
        GOSBVL =REVPOP  renverse search
        ?A=0   A        LEN(search)=0 ?
        GOYES  NStrng   oui, erreur
        D0=(5) =S-R1-2  S-R1-2 := LEN(search)
        DAT0=A A
        CD1EX           ^ search
        D1=C
        D0=D0+ 5        D0 := ^ S-R1-3
        DAT0=C A        S-R1-3 := ^ search
        GOSBVL =AVE=D1  AVMEME := D1

        GOSBVL =GETPRO  fichier prive ?
        GOC    Mferr    oui, erreur
        GOSBVL =GETSTC  fichier Basic ?
        GONC   Find20   oui
        LC(2)  =eFTYPE  non, erreur
Mferr   GOVLNG =MFERR
Find20  GOSBVL =NULLP   fichier vide ?
        GONC   Find30   non
NStrng  P=     0        erreur 'Not Found'
        LC(4)  =eNFOUN
        GOVLNG =BSERR
Find30  D1=(5) =CURRL   positionnement ..
        C=0    A        .
        C=DAT1 4        .
        GOSBVL =FINDL   .. ligne courante
        GONC   NStrng   non trouvee
        CD1EX           C[A] := ^ numero de ligne
        B=C    A        sauve dans B[A]
        CD1EX           restaure D1
        GOSBVL =NXTLIN  ligne suivante
        ?C<D   A        avant fin de fichier ?
        GOYES  Find15   oui
        A=B    A        non, on commence a la ..
        D1=A            .. ligne courante
Find15  C=D    A        C[A] := ^ fin fichier
        GOSBVL =CSLW5
        CD1EX           C[A] := ^ debut de ligne
        D1=C            D1 := ^ debut de ligne
Findx   R3=C            sauvegarde des pointeurs
          *             de fichier dans R3
        GOSBVL =LDCM10  decompile la ligne

* A ce stade :
*   (OUTBS) ^ ligne decompilee ("objet")
*   B[A] = LEN(object) en octets
*   P indetermine
*
*   S-R1-2  = LEN(search)
*   S-R1-3  = ^ search
*   R3[9:5] = ^ debut de ligne courante
*   R3[4:0] = ^ fin de fichier
Find40  P=     0        par securite
* Comme la recherche porte simplement sur la
* partie utile de la ligne, excluant le numero de
* ligne, il faut actualiser la longueur de la
* ligne
```

```
        C=0    W
        D0=(5) =OUTBS
        A=DAT0 A        A[A] := ^ debut de ligne
        D0=(4) =LDCSPC
        C=DAT0 A        C[A] := ^ espace apres
          *             numero de ligne
        C=C-A  A
        CSRB            longueur en octets
        C=C+1  A        + 1 espace
        A=B    A        A[A] longueur totale
        A=A-C  A        A[A] := longueur sans
          *             en-tete
        B=A    A        B[A] := longueur utile
        A=A+A  A        A[A] := longueur en nibs
        C=DAT0 A        C[A] := ^ objet
        R2=C            R2 := ^ objet
        D1=C            D1 := ^ objet
        D0=(5) =S-R1-3  .
        C=DAT0 A        .
        R1=C            R1 := ^ search
        D0=D0- 5        D0 ^ S-R1-2
        C=DAT0 A        C[A] := LEN(search)

* A ce stade :
*  A[A] = LEN(objet)
*  R2   = ^ objet
*  C[A] = LEN(search)
*  R1   = ^ search
* Le code est inspire de POS et STREQL mais ces
* deux sous-programmes travaillent sur des chaines
* "inversees" (premier caractere en adresse haute)
* Ici le premier caractere est en adresse basse.
* La chaine est divisée en n blocs de 16 quartets
* et un reste.

        A=A-C  A
        GOC    Pos5     LEN(objet)<LEN(search)
        C=C-1  A        LEN(search)-1
        P=C    0        RMD(Len(search$)-1,16)
         *                -> reste
        CSR    A        DIV(Len(search$)-1,16)
        D=C    A          -> n
        GONC   Pos3     B.E.T.

Pos2    C=B    A
        RSTK=C          Scanindex -> Stack;
        CD1EX
        RSTK=C          Scanpt -> Stack;
        CD1EX
        C=D    A
        B=C    A        B[A] := n
        GOSUB  streql   compare search avec la --
          *             suite de objet          |
        C=RSTK                                  |
        D1=C            restaure D1             |
        C=RSTK                                  |
        B=C    A        restaure B[A]           v
        GONC   Pos6     search incluse dans objet

        B=B-1  A        utilise plus loin -------
Pos3    C=R1            ^ search                |
        D0=C            D0 := ^ search          |
        GOC    Pos5     objet est analyse <------
        A=DAT0 B        A[B] := premier caractere
          *             de search
Pos4    D1=D1+ 2        caractere suivant de objet
        C=DAT1 B        C[B] := caractere suivant
        ?A=C   B
        GOYES  Pos2     debut d'egalite
        B=B-1  A        reste-t-il des caracteres
        GONC   Pos4     oui
Pos5    A=R2            sortie
        D1=A            pour avoir 0 plus loin
Pos6    A=R2
        C=0    W
        C=A    A        A[A] := ^ objet
        AD1EX           D1 := ^ premier caractere
        A=A-C  A
* A[A] contient le deplacement (dans objet)
* pour lequel on a trouve search dans objet.
* Si cette valeur est non nulle on va afficher la
* ligne
        P=     0
        C=R3
        D1=C            D1 := ^ debut de ligne
        ?A=0   A        a-t-on trouve search ?
        GOYES  Find60   non
* On affiche la ligne.
* Il faudra positionner le curseur sur le
* premier caractere ou on trouve search
        R3=A            R3 := deplacement
        GOSBVL =LDCOMP  Decompile la ligne pointee
          *             par D1; elle devient
          *             ligne courante
        D0=(5) =OUTBS
        C=DAT0 A
        D0=C            D0 := ^ premier caractere
        D1=(5) =LDCSPC
        A=0    M
        A=DAT1 A        A[A] := ^ espace apres le
          *             numero de ligne
        A=A-C  A
        C=R3            C[A] := deplacement
        A=A+C  A        ajoute a l'offset
        ASRB            converti en octets
        R0=A            sauve dans R0
        C=B    A        2*LEN(ligne)
        RSTK=C          sauvegarde
        GOSUB  Find*
        NIBHEX B1C3     curseur eteint
        NIBASC '>'      '>'
        NIBHEX B1E3     curseur allume
        NIBHEX FF       fin de sequence
Find*   C=RSTK          C[A] := ^ debut sequence
        D1=C
        GOSBVL =BF2DSP  affiche la sequence
        C=RSTK          B[A] := 2*LEN(ligne)
```

```
        B=C    A
        GOSBVL =DSPCNO  affiche buffer de sortie
        C=R0            nombre de curseur a droite
        GOSBVL =CURSRT  execution ...
        GOSBVL =BLDDSP  construit l'affichage
        GOVLNG =MAIN30  retour a Basic
* on n'a rien trouve sur la ligne courante, on
* essaie de passer a la ligne suivante.
 Find60 GOSBVL =CSRW5
        D=C    A        D[A] := ^ fin de fichier
        GOSBVL =NXTLIN  passage ligne suivante
        P=     0
        ?C<D   A        dans le fichier ?
        GOYES  Find70   oui
        GOTO   NStrng    fin de fichier et rien
          *             trouve
* on va passer a la ligne suivante, on remet les
* pointeurs en place et on verifie que
* l'utilisateur n'a pas appuye sur [ON].
 Find70 C=R3            .
        CD1EX           .
        R3=C            restaure R3
        GOSBVL =CK"ON"  verifie [ON] ------------
        C=R3                                    |
        D1=C            D1 := ^ numero de ligne |
        GOC    Findx+   [ON] non demande <------
        GOVLNG =NXTSTM  si demande, fin
 Findx+ GOTO   Findx    recommencons ...
 morest A=DAT0 W
        C=DAT1 W
        ?A#C   W
        RTNYES          # retour Carry Set
        D0=D0+ 16       ^ mot suivant
        D1=D1+ 16
 streql B=B-1  A        termine avec les mots
          *             complets
        GONC   morest   non
 tailst A=DAT0 WP       test sur le reste
        C=DAT1 WP
        ?A#C   WP
        RTNYES
        RTNCC
        END
```

TO KEN OR NOT TO KEN

Le programme Basic qui je vous présente est un éditeur de fichiers Lex.

Je vous déçois tout de suite : il ne s'agit pas d'un désassembleur, ni même d'un éditeur hexadécimal à la manière de la fonction RAMED du module ZENROM du HP-41. Non ! Ce programme vous permettra simplement de connaître l'Id du fichier édité ainsi que le nom et le token des mots clés qu'il renferme, puis de les modifier dans certaines limites.

Lancez le programme par RUN ; si vous voulez l'affichage en vidéo inversée, faites RUN,15 (la première fois seulement). Je reviendrai plus tard sur cette caractéristique.

Le programme vous demande le nom d'un fichier ; la question est posée tant que le fichier nommé n'existe pas. Une entrée vierge sort du programme.

On entre ensuite réellement dans l'éditeur grâce au sous programme TOKEN. Celui-ci commence par vérifier que le fichier choisi est bien un fichier Lex. Si ce n'est pas le cas, vous aurez droit à un message d'erreur approprié. Le programme va ensuite rechercher divers paramètres du fichier Lex :

- l'octet d'identification du fichier ou Id (le numéro d'XROM si vous préférez),

- le plus petit et le plus grand token du fichier (très important, vous le constaterez plus loin),

- un éventuel lien avec un autre fichier Lex si vous les avez chaînés par exemple,

- et enfin l'offset de la "Text Table" et l'adresse elle même de cette table qui contient les noms et tokens des mots clés présents.

Il ne reste plus qu'à boucler sur le nombre de tokens ; en fait, jusqu'à la fin de la "Text Table" signalée par le triquartet 1FF. Entre-temps, on édite pour chaque token l'Id du fichier, le nom du mot clé et son token.

L'attente de l'appui sur une touche permet de piloter le programme :

- n'importe quelle touche sauf [M] ou [Q] fait poursuivre l'édition au token suivant.

- la touche [Q] termine l'édition.

- la touche [M] bifurque vers le sous-programme de modification. A ce moment, le programme vous propose de modifier successivement l'Id du fichier puis le nom du mot clé et enfin le token. La seule restriction concerne le nom du mot clé si vous le modifiez : il doit avoir *exactement* le même nombre de caractères avant et après modification. A chaque fois, le programme vous rappelle le contenu de chacun des paramètres que vous pouvez modifier. Bien sûr, si le fichier est protégé, un message d'erreur vous rappelle à l'ordre.

Après l'affichage de chaque token, le programme calcule les tokens minimum et maximum. En cas de modification de ces derniers, il est impératif de mettre à jour ces valeurs, stockées juste après l'Id (vous vous en souvenez j'espére car je vous en ai parlé un peu plus haut). C'est très important car si HP soigne particulièrement ses systèmes d'exploitation, ceux-ci ont une fâcheuse tendance à l'amnésie. C'est pourquoi, d'ailleurs, lorsque vous terminez l'édition avant le terme, le programme est obligé de la poursuivre en douce (sans affichage) pour être sûr de bien mettre à jour ces deux valeurs. C'est également pour cette raison que je vous recommande de ne pas interrompre le programme.

Lorsque la boucle est terminée, le programme se rappelle si un autre fichier Lex est chaîné à celui-là et recommence éventuellement l'opération.

Si, d'aventure, vous éditiez un fichier sans mot clé (interception de poll par exemple), le programme éditera un message entendu.

En analysant le programme, vous constaterez que tous les accès à la mémoire se font par Forth interposé. A cela, plusieurs raisons qui justifient qu'on perde quelques 2500 octets de mémoire :

- Forth est plus rapide (10 fois plus : on ne vous le dira jamais assez, Basic est facile mais très très lent).

- Forth est plus facile à mettre en oeuvre que PEEK$ et POKE car on peut manipuler indifféremment des nombres entiers ou des chaînes de caractères.

- Forth remet tout à l'endroit : allez donc récupérer une chaîne de caractères avec PEEK$ alors que celle-ci est complètement inversée ; je veux dire quartet par quartet en partant de la fin : dingue, je vous le répète !

- Forth n'a que faire des zones privatisées ou protégées.

Enfin, si vous voulez étendre les possibilités du programme, vous pouvez le faire bien entendu. Il y a certainement bien d'autres choses intéressantes à découvrir dans les fichiers Lex. Mais je crois que le plus judicieux serait de reprendre le programme SCAN de Jean-Pierre Bondu (JPC 39), d'y faire appel au Forth, puis d'y incorporer mon programme ; on obtiendrait ainsi un programme très complet d'édition de fichiers Lex. A suivre...

Pour terminer, quelques mots sur l'affichage en vidéo inversée dont je parlais plus haut : un peu de fantaisie dans un programme n'a jamais nui à quiconque. Je me suis donc amusé à décaler dans la deuxième page de caractères (+128 donc) certains titres non changeant qui sont envoyés à l'affichage ("New Keyword:", "ID:", etc.). Le sous-programme CHARSET appelé en ligne 15 construit alors un CHARSET$ avec les caractères standards en vidéo inversée et le tour est joué : c'est plus rapide que d'inverser tout ou partie de l'affichage en temps réel.

Voici comment procéder pour introduire un texte en video inversée dans un programme :

- initialisez une variable alphanumérique avec la chaîne de caractères normale ; soit P$ cette chaîne
- tapez au clavier CALL INVSTR(P$) ; votre variable contient la même chaîne en vidéo inversée
- il ne vous reste plus qu'à affecter cette chaîne à une touche du clavier (mode USER) pour l'introduire telle quelle dans le programme:
DEF KEY "*code touche*",P$;

Attention : le fichier Lex CHARLEX (caractères accentués) est incompatible avec ce procédé ; il réinitialise CHARSET$.

Bonne édition et maintenant, remettez un peu d'ordre dans la multitude de tokens qui encombrent la mémoire de votre machine préférée.

Alain Goubault (308)

PROGRAMME DE DEBUTANT

On peut être débutant et pourtant navigateur ! C'est pourquoi, puisque JPC réclame des articles de débutants, je vous propose aujourd'hui un petit programme pour les vacances.

Le programme DISTANCE permet, muni d'un sextant, de calculer à quelle distance on se trouve d'un amer dont la hauteur est connue, qu'il se trouve avant ou après l'horizon.

Le programme donne en prime la distance de l'horizon et la distance théorique maximum de visibilité pour l'observateur, ceci évitera d'écarquiller les yeux pour chercher un phare qu'on ne saurait apercevoir avant la fin de la nuit !

Pour obtenir directement la distance de l'horizon et la distance théorique de visibilité d'un amer non encore aperçu, entrer 0 comme hauteur sextant.

Bonne programmation et bonne navigation !

François Duret-Lamouroux (329)

LE POINT SUR LES NUAGES

Les fonctions statistiques du HP-71 sont particulièrement puissantes et souples d'utilisation. Cependant, pour des besoins précis, il peut être souhaitable de les compléter par des programmes. Voici ma contribution :

BUT

Ce programme permet de trouver l'équation d'un nuage de point. Il utilise les fonctions statistiques du HP-71. A partir de l'ensemble de points (x,y), nous pouvons obtenir une des quatre équations suivantes :

droite : $y = a + b\ x$

exponentielle : $y = a * e^{b\ x}$

logarithmique : $y = a + b\ \mathrm{Log}\ x$

puissance : $y = a + b^{x}$

Le programme peut traiter jusqu'à 500 points mais, généralement, une dizaine bien choisie suffit.

UTILISATION

Faites : `RUN NUAGE`

Le programme affiche alors son menu principal :

```
Raz. Ad. Cherc. Previ.
```

[Raz.]

La touche [R] (Remise à zéro) permet d'effacer tous les points existants. Comme le programme utilise l'environnement principal, il est recommandé de faire une remise à zéro lors d'une première utilisation.

[Ad.]

La touche [A] permet d'ajouter de nouveaux points, de supprimer des points existants ou d'isoler des points.

[Cherc.]

La touche [C] permet de rechercher la meilleure corrélation pour les points disponibles et de choisir une des quatre équations.

[Previ.]

La touche [P] permet de faire une prévision (extrapolation) en fonction de l'équation retenue à l'étape précédente (Cherc.).

EXEMPLES D'UTILISATION

Introduction des données : ([Raz.] et [Ad.])

Après avoir appuyé sur [R], le programme affiche :

```
X1: 0
```

Tous les points sont remis à zéro, le programme est prêt à traiter un nouveau cas.

Pour introduire une valeur, il suffit de frapper le nombre voulu ; par exemple : (129,7) soit X1=129 et Y1=7.

Affichage	Touches
X1: 0	[1]
X1? 1	[2]
X1? 12	[9]
X1? 129	[ENDLINE]
Y1: 0	

Automatiquement le programme passe à la valeur suivante: Y1.

Affichage	Touches
Y1: 0	[7]
Y1? 7	[ENDLINE]
X2: 0	

Vous pouvez vous déplacer d'une valeur à l'autre à l'aide des touches [^] et [v] ou directement comme ceci :

Affichage	Touches
X2: 0	[Y]
X2: 0	[1]
Y1	[6]
Y16	[2]
Y162	[ENDLINE]
Y162: 0	

L'introduction des différentes valeurs se fait selon les mêmes procédés que le point 1 (129,7).

Affichage	Touches
Y162: 0	[1]
Y162? 1	[7]
Y162? 17	[ENDLINE]
Y162: 17	

A ce point du programme, nous avons un nuage de deux points, soit (129,7) et (0,17).

Pour sortir de l'éditeur et revenir au menu principal, on utilise la touche [ATTN].

```
Raz. Ad. Cherc. Previ.
```

Recherche de la bonne corrélation

Soit la suite de points suivante :

```
(4,  1024)
(1,     1)
(3,   243)
(7, 16807)
(8, 30000)
```

Après avoir introduit les 5 points ci-dessus et être revenu au menu principal, appuyez sur la touche [C] (Cherc.).

Le programme donne les quatre corrélations suivantes :

Affichage	Touches
Dr. .913260728017	[ENDLINE]
Ex. .953914151914	[ENDLINE]
Lg. .765623658583	[ENDLINE]
Pu. .999966898932	[ENDLINE]
Dr. Ex. Lg. Pu. Me.	

Nous voyons que la meilleure corrélation est obtenue avec une courbe du type Puissance (.999966898932) et que la moins bonne est obtenue avec une courbe du type Log (.765623658583).

Le sous-menu suivant vous permet de choisir le type de courbe que vous désirez :

Droite
Exponentielle
Logarithmique
Puissance
Meilleure des quatre

Affichage	Touches
Dr. Ex. Lg. Pu. Me.	[M]
y = a * x ^ b	[ENDLINE]
corr. = .9999669	[ENDLINE]
a = 1.0147922	[ENDLINE]
b = 4.9751662	[ENDLINE]
Raz. Ad. Cherc. Previ.	

Nous obtenons donc l'équation suivante :
$y = 1.0147922 \times ^{4.9751662}$

Nous sommes presque en présence d'une équation du type : $y = x^5$.

Si l'on regarde bien les 5 points, on remarque que les quatre premiers Y sont bien égaux aux puissances cinquièmes de leurs X respectifs mais que le cinquième Y est différent. En effet, 8^5=32768 et non 30000.

Revenons sous l'éditeur : [A]

Affichage	Touches
X6: 0	[^]
Y5: 30000	[/]
[Y5: 30000]	[^]
[X5: 8]	[ATTN]
Raz. Ad. Cherc. Previ.	

Nous avons isolé le point numéro 5, il ne sera plus pris en compte lors de la prochaine recherche.

Affichage	Touches
	[C]
Dr. .888918507746	[ENDLINE]
Ex. .953046045207	[ENDLINE]
Lg. .713783989995	[ENDLINE]
Pu. 1	[ENDLINE]
Dr. Ex. Lg. Pu. Me.	[M] ou [P]
y = a * x ^ b	[ENDLINE]
corr. = 1	[ENDLINE]
a = 1	[ENDLINE]
b = 5	[ENDLINE]
Raz. Ad. Cherc. Previ.	

Le point 5 peut être supprimé...

	[A]
[X5: 8]	[-]
Suppression 5	
X5: 0	[ON]
Raz. Ad. Cherc. Previ.	[C]
Dr. .888918507746	[ENDLINE]
Ex. .953046045207	[ENDLINE]
Lg. .713783989995	[ENDLINE]
Pu. 1	[ENDLINE]
Dr. Ex. Lg. Pu. Me.	[M] ou [P]
y = a * x ^ b	[ENDLINE]
corr. = 1	[ENDLINE]
a = 1	[ENDLINE]
b = 5	[ENDLINE]
Raz. Ad. Cherc. Previ.	

Prévision

La touche [P] permet d'effectuer une prévision sur l'équation choisie durant la phase de recherche de la meilleure corrélation.

Affichage	Touches
Raz. Ad. Cherc. Previ.	[P]
Prev. X ou Y ?	[Y] (par exemple)
X ?	[7]
X ? 7	[ENDLINE]
Y = 16807	[ENDLINE]
X ?	[-]

```
X ? -                    [3]
X ? -3                   [ENDLINE]
Y = -243                 [ENDLINE]
X ?                      [ENDLINE]
Raz. Ad. Cherc. Previ.   [P]
Y ?                      [1]
Y ? 10                   [0]
Y ? 100                  [0]
Y ? 1000                 [0]
Y ? 10000                [0]
Y ? 100000               [ENDLINE]
X = 10                   [ENDLINE]
Y ?                      [ENDLINE]
Raz. Ad. Cherc. Previ.   [ATTN]
```

Voilà pour ce petit programme qui, j'espère, vous amusera beaucoup.

Je n'ai pas cherché à gagner des octets. Si vous faites mieux en moins de place, n'hésitez pas à envoyer votre oeuvre au Journal.

Michel Martinet (12)

Programme "LEXED" (Editeur de Lex. Necessite FILELEX, KEYWAIT et DESLEX)

```
   - Programme d'édition de fichiers LEX
     A. Goubault de brugière 25 nov 1986
     LEX utilisés: FILELEX, KEYWAIT$ (Sub TOKEN)
     et DISPLEX (sub CHARSET) ainsi que FORTH
  10 GOTO 20
  15 CALL CHARSET
  20 DIM N$[8] @ INPUT 'Name of file: ';N$ @ IF N$='' THEN STOP
  25 IF NOT FILE?(N$) THEN 20 ELSE CALL EDLEX(N$)

================================================================
  30 SUB EDLEX(N$)
  35 SFLAG 1 @ SFLAG 2 @ SFLAG 3
  40 DIM I$[2],R$[8],F$[16],R1$[2]
  45 D0=HTD(ADDR$(N$)) @ D1=D0+16 @ D2=D0+37 @ D3=D0+43 @ D4=D0+49
  50 FORTHX '4N@',D1 @ T=FORTHI
  55 IF T<>HTD('E208') AND T<>HTD('FF') THEN 170
  60 FORTHX 'N@',D1+4 @ P=FORTHI
  65 FORTHX 'DUP C@ SWAP 2+ DUP C@ SWAP 2+ C@',D2 @ L2=FORTHI @ L1=FORTHI @ I$=DTH$(FORTHI)[4,5]
  70 FORTHX '@',D3 @ D=FORTHI @ IF D=0 THEN CFLAG 1
  75 FORTHX 'N@',D3+5 @ IF FORTHI=0 THEN D4=D4+79
  80 FORTHX '4N@',D4 @ M0=FORTHI @ IF M0=0 THEN 180
  85 D5=D4+M0-1 @ M1=255 @ M2=0
  90 FORTHX 'N@',D5 @ S=FORTHI+1
  95 FORTHX '',D5+1,S/2 @ R$=FORTH$ @ F$=R$&'          ' @ R$=F$[1,8]
 100 IF NUM(R$[1,2])=255 THEN 150
 105 FORTHX 'C@',D5+S+1 @ R1=FORTHI @ R1$=DTH$(R1)[4,5]
 110 IF NOT FLAG(2) THEN 140
 115 CFLAG 3 @ CALL DISP(I$,R$,R1$)
 120 ON POS("MQ",KEYWAIT$)+1 GOTO 140,125,135
 125 IF P=0 THEN CALL MODID(I$,D2,R$,S/2,R1$,D5) @ GOTO 140
 130 IF P<>0 THEN DISP 'Secured or Private' @ WAIT 1 @ GOTO 105 ELSE 140
 135 CFLAG 2
 140 D5=D5+S+3 @ R1=HTD(R1$) @ M1=MIN(M1,R1) @ M2=MAX(M2,R1)
 145 GOTO 90
 150 IF M1<>L1 OR M2<>L2 THEN FORTHX 'C! C!',M1,D2+2,M2,D2+4
 155 IF NOT FLAG(1) THEN 175
 160 D2=D3+D @ D3=D2+6 @ D4=D2+12
 165 GOTO 65
 170 BEEP @ DISP 'Not a LEX file' @ GOTO 190
 175 IF L1<>0 AND L2<>0 THEN PUT "#43" @ GOTO 190
 180 IF NOT FLAG(3) THEN 190
 185 BEEP @ DISP 'èáø ';I$;' òÃ  Ÿÿ åÿšíŸ'
 190 CFLAG 1 @ CFLAG 2 @ CFLAG 3 @ END

================================================================
 195 SUB DISP(N$,T$,T1$)
 200 DISP 'èáø'&N$&' ù¼ '&T$&' åŠ'&T1$
 205 END SUB

================================================================
 210 SUB MODID(I$,D2,T$,L,T1$,D5)
 215 INPUT ' öí¼ Èèáø ',I$;I$ @ I=HTD(I$)
 220 FORTHX 'C!',I,D2
 225 INPUT ' öí¼ ùí²¼ÿ đø ',T$[1,L];T$[1,L]
 230 FORTHX 'SWAP CMOVE',T$[1,L],D5+1
```

```
 235 INPUT ' öí¼ åÿŠÍŸ¤ ',T1$;T1$ @ T1=HTD(T1$)
 240 FORTHX 'C!',T1,D5+L*2+1
 245 END SUB

============================================================
 250 SUB CHARSET
 255 DIM A$[128] @ CHARSET '' @ A$=''
 260 FOR N=0 TO 127 @ L=N
 265 IF L=0 OR L=8 OR L=10 OR L=13 OR L=27 THEN L=32
 270 A$=A$&CHR$(L) @ NEXT N
 275 K2=0
 280 FOR I=1 TO 5
 285 K1=K2+1 @ K2=K1+21
 290 DISP A$[K1,K2] @ INVERSE @ CHARSET CHARSET$&GDISP$
 295 NEXT I
 300 DISP A$[111,128] @ GDISP INV$(GDISP$[1,108]) @ CHARSET CHARSET$&GDISP$
 305 END SUB

============================================================
 310 SUB INVSTR(C$)
 315 FOR I=1 TO LEN(C$)
 320 C$[I,I]=CHR$(NUM(C$[I,I])+128)
 325 NEXT I @ END SUB
```

****

Programme "NAV" (Navigation)

```
  10 DESTROY D,H,S,V
  11 DISP "Pour dist.hor.et vis.S=0" @ WAIT .2
  20 INPUT "Elévation de l'oeil:",'?';E0
  30 INPUT "Hauteur objet,mètres=",'?';H
  40 INPUT "Hauteur sextant, minutes=",'?';S
  50 INPUT 'Avant horizon? (O/N): ','O';W$
  55 IF S=0 THEN GOTO 100
  70 IF W$='O' THEN 85 ELSE 80
  80 D=1.19*(SQR((S-.97*SQR(E0*3.2808))^2+3.12*(H-E0))-(S-.97*SQR(E0*3.2808))) @ GOTO 90
  85 D=1.856*H/S @ GOTO 90
  90 DISP 'Dist.observée ';D;'milles' @ WAIT 1
 100 O=2*SQR(E0)
 110 DISP "Distance à l'horizon=";O;"milles" @ WAIT 1
 120 V=2.1*(SQR(E0)+SQR(H))
 130 DISP "Visible à ";V;"milles"
```

****

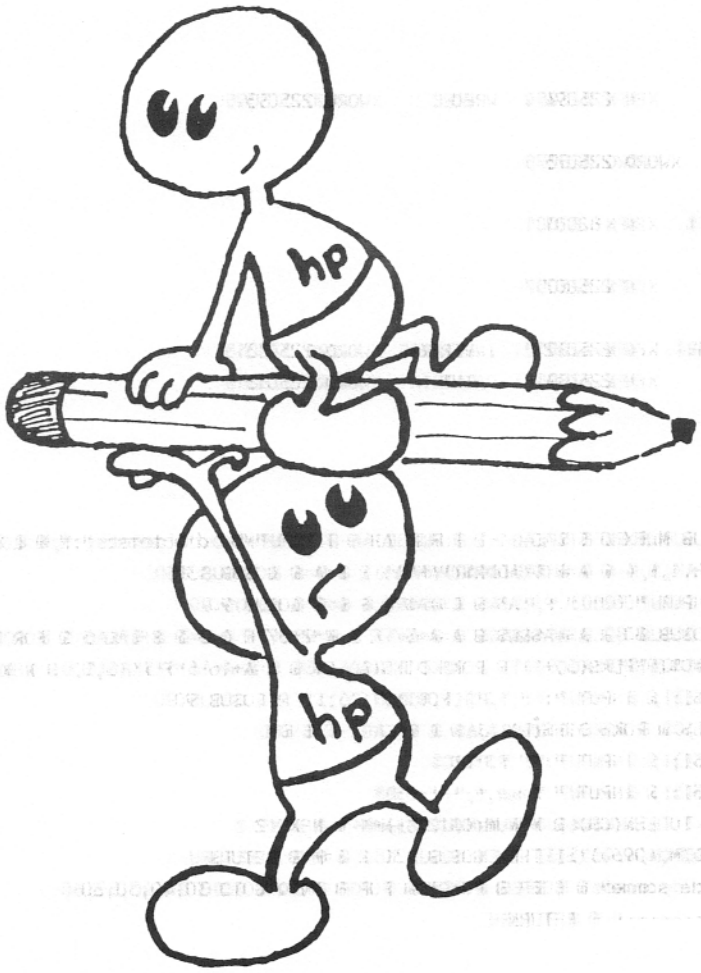
Programme "NUAGEPT" (Programme d'ajustement interactif de courbes)

- PROGRAMME NUAGEPT
  Ce programme a pour but principal de trouver l'équation ou les équations d'une courbe donnée ou d'un nuage de points. On peut généralement être satisfait d'un résultat lorsque l'on obtient un coefficient de corrélation

```
    compris entre 0.90 et 1.
    Ce programme utilise l'environement principal.
    mise en pratique des fonctions statistiques et régressions linéaires.
    Aucune fonction LEX supplémentaire n'y est utilisée.
    Un simple HP-71 sans HP-IL et sans module supplémentaire suffit.
    Il est indispensable d'exécuter une remise à zéro du programme si vous
    n'êtes pas sûr de la dimension et de la valeur de vos variables.
 100 POKE "2F441","1" @ DEFAULT EXTEND @ OPTION ROUND NEAR @ WIDTH 80
 105 LC OFF @ STD @ DIM A$[3],K$[4],X$,Y$
 110 DELAY 0,0 @ DISP "Raz. Ad. Cherc. Previ."
 120 K$=KEY$ @ IF NOT LEN(K$) THEN 120
 130 IF NUM(K$)#35 THEN 140 ELSE IF K$="#43" THEN PUT K$ @ POKE "2F441","0" @ STD @ END
   - Remise à zéro (82 => "R")
 140 IF NUM(K$)=82 THEN GOSUB 1000
   - Addition de nouveaux points (65 => "A")
 150 IF NUM(K$)=65 THEN GOSUB 1020
   - Recherche équation en fonction du nuage
     de points actuel (67 => "C")
 160 IF NUM(K$)=67 THEN DELAY 8 @ GOSUB 2000
   - Interpolation en fonction de l'équation trouvée
     (80 => "P")
 170 IF NUM(K$)=80 THEN DELAY 8 @ GOSUB 3000
 180 GOTO 110
   - Préparation de l'environnement principal pour une nouvelle étude.
1000 OPTION BASE 0 @ REAL I,J,L,M,N,R,U,V,X,Y,Z @ REAL A(L,1) @ INTEGER W9(L)
1010 A(0,0)=0 @ A(0,1)=0 @ W9(0)=0
   - La ligne 1030 est un point de retour obligé pour la redéfinition de
     de la taille des tableaux.
1020 REAL A(L,1) @ CFLAG 5 @ INTEGER W9(L)
1030 IF W9(R) THEN A$="[ ]" ELSE A$="   "
1040 DISP A$[1,2]&CHR$(88+Z)&STR$(R+1)&":";A(R,Z);A$[3]
   - Attention : la fonction KEYDOWN est légèrement boguée lorsqu'elle est
     utilisée avec un paramètre. Si l'expression de K$ est différente de
     celle de la touche pressée lors du test, le système renvoie une erreur.
     Ex : K$="#51" avant test, la touche pressée durant le test est
     "fG" ===> ERREUR ! #XX <> [g[f[X]]]
1050 IF KEYDOWN(K$) THEN 1070
1060 K$=KEY$ @ IF NOT LEN(K$) THEN 1060 ELSE IF LEN(K$)>3 THEN 1060
1070 IF K$="#50" THEN Z=MOD(Z+1,2) @ R=R-(Z*(R>0)) @ GOTO 1020
1080 IF K$="#51" THEN Z=MOD(Z+1,2) @ R=R+NOT Z*(R<499) @ L=L+(R>L) @ GOTO 1020
1090 IF POS(".0123456789",K$) THEN 1150
1100 IF NUM(K$)=88 THEN Z=0 @ GOTO 1210
1110 IF NUM(K$)=89 THEN Z=1 @ GOTO 1210
1120 IF NUM(K$)=45 THEN 1250 ELSE IF K$="#43" THEN RETURN
1130 IF NUM(K$)=47 AND A(R,0)+A(R,1) THEN W9(R)=MOD(W9(R)+1,2) @ M=M-(W9(R)#0)+NOT W9(R)
1140 GOTO 1030
1150 N=A(R,0)+A(R,1)#0
1160 WINDOW LEN(STR$(R+1))+4,22 @ PUT K$ @ INPUT A(R,Z) @ WINDOW 1,22
1170 IF A(R,Z)<0 THEN 1160
1180 IF A(R,0)+A(R,1) AND NOT N THEN M=M+1
1190 IF N AND NOT (A(R,0)+A(R,1)) THEN M=M-1
1200 K$="#51" @ GOTO 1080
1210 K$=KEY$ @ IF NOT LEN(K$) THEN 1210
1220 IF NOT POS("123456789",K$) THEN 1030 ELSE DISP CHR$(88+Z)
1230 PUT K$ @ WINDOW 2,22 @ INPUT "";R @ IF R<1 OR R>500 THEN 1230 ELSE R=R-1
1240 L=L+(R-L)*(R>L) @ WINDOW 1,22 @ GOTO 1020
1250 DISP "Suppression";R+1 @ N=(A(R,0)+A(R,1)#0)-(W9(R)#0)
1260 FOR I=R+1 TO L @ A(I-1,0)=A(I,0) @ A(I-1,1)=A(I,1) @ W9(I-1)=W9(I) @ NEXT I
```

```
1270 M=M-(N#0) @ L=L-(L>R)
1280 IF R=499 THEN A(R,0)=0 @ A(R,1)=0
1290 GOTO 1020
   - Aucune régression ne peut être faite pour un seul point.
2000 IF M<2 THEN DELAY 1 @ BEEP @ DISP "Pas assez de points !" @ RETURN
   - Remise à zéro des tableaux statistiques
2010 STAT S(4) @ CLSTAT @ REAL C(4),P(1)
   - Préparation du tableaau S() pour la régression. Lorsqu'un point
     est nul (X=0 et Y=0) il n'est pas pris en compte.
2020 FOR I=L TO 0 STEP -1
2030 IF NOT (A(I,0)+A(I,1)) OR W9(I) THEN 2080
2040 FOR J=0 TO 1
2050 IF NOT A(I,J) THEN P(J)=EPS ELSE P(J)=A(I,J)
2060 NEXT J
2070 ADD P(0),LN(P(0)),P(1),LN(P(1))
2080 NEXT I @ ON ERROR GOTO 2240
2090 C(1)=ABS(CORR(1,3)) @ DISP "Dr.";C(1)
2100 C(2)=ABS(CORR(1,4)) @ DISP "Ex.";C(2)
2110 C(3)=ABS(CORR(2,3)) @ DISP "Lg.";C(3)
2120 C(4)=ABS(CORR(2,4)) @ DISP "Pu.";C(4) @ OFF ERROR
2130 DISP "Dr. Ex. Lg. Pu. Me."
2140 K$=KEY$ @ IF NOT LEN(K$) THEN 2140
2150 C(0)=POS("DELPM",K$) @ IF NOT C(0) THEN 2140 ELSE K$=""
2160 IF C(0)=5 THEN CALL MCORR(C)
2170 ON C(0) GOSUB 2200,2210,2220,2230 @ SFLAG 5
2180 FIX 7 @ V=VAL(STR$(V)) @ W=VAL(STR$(W)) @ C(C(0))=VAL(STR$(C(C(0)))) @ STD
2190 DISP "corr. =";C(C(0)) @ DISP "a =";V @ DISP "b =";W @ RETURN
   - linéaire
2200 LR 3,1,V,W @ DISP "y = a + b * x" @ RETURN
   - exponentielle
2210 LR 4,1,V,W @ V=EXP(V) @ DISP "y = a * e ^ ( b * x )" @ RETURN
   - logarithmique
2220 LR 3,2,V,W @ DISP "y = a + b * Log ( x )" @ RETURN
   - puissance
2230 LR 4,2,V,W @ V=EXP(V) @ DISP "y = a * x ^ b" @ RETURN
2240 DELAY 1 @ BEEP @ DISP "Pente = 0 !" @ OFF ERROR @ RETURN
   - PREVISION ! Non utilisation de la fonction PREDV du système pour ne pas
     refaire une régression à chaque changement de variable dépendante.
3000 IF NOT FLAG(5) THEN RETURN
3010 DISP "Prev. X ou Y ?"
3020 K$=KEY$ @ IF NOT LEN(K$) THEN 3020
3030 IF K$="#43" THEN RETURN ELSE IF NUM(K$)=89 THEN 3090 ELSE IF NUM(K$)#88 THEN 3020
3040 DISP "Y ? ";
3050 K$=KEY$ @ IF NOT LEN(K$) THEN 3050 ELSE IF K$="#43" THEN PUT "#38" ELSE PUT K$
3060 INPUT "";Y$ @ IF NOT LEN(Y$) THEN RETURN ELSE Y=VAL(Y$)
3070 DISP " X = "; @ ON C(0) GOSUB 4100,4200,4300,4400 @ GOTO 3040
3080 GOTO 3000
3090 DISP "X ? ";
3100 K$=KEY$ @ IF NOT LEN(K$) THEN 3100 ELSE IF K$="#43" THEN PUT "#38" ELSE PUT K$
3110 INPUT "";X$ @ IF NOT LEN(X$) THEN RETURN ELSE X=VAL(X$)
3120 DISP " Y = "; @ ON C(0) GOSUB 4500,4600,4700,4800 @ GOTO 3090
3130 GOTO 3000
   - Droite prévision de X
4100 X=(Y-V)/W @ DISP X @ RETURN
   - Exponentiel prévision de X
4200 X=LN(Y/V)/W @ DISP X @ RETURN
   - Log. prévision de X
4300 X=EXP((Y-V)/W) @ DISP X @ RETURN
```

```
   - Puissance prévision de X
4400 X=(Y/V)^(1/W) @ DISP X @ RETURN
   - Droite prévision de Y
4500 Y=V+W*X @ DISP Y @ RETURN
   - exponentielle prévision de Y
4600 Y=V*EXP(W*X) @ DISP Y @ RETURN
   - log. prévision de Y
4700 Y=V+W*LN(X) @ DISP Y @ RETURN
   - puissance prévision de Y
4800 Y=V*X^W @ DISP Y @ RETURN
   - Ce sous-programme calcule le meilleur coefficient de corrélation
     Les coefficients sont dans C(1 à 4), le no du meilleur revient dans C(0).

================================================================
9000 SUB MCORR(C()) @ C(0)=1 @ FOR I=2 TO 4
9010 IF C(I)>C(C(0)) THEN C(0)=I
9020 NEXT I
```



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.

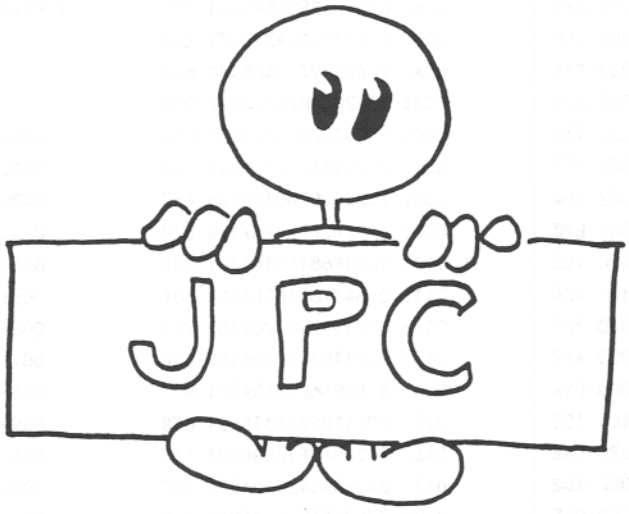
Vous trouverz donc, outre les Lex de la rubrique Assembleur, le Lex CHARLEX nécessaire à la rédaction de votre article (voir "Ah ! Vous écrivez !") et les Lex utilisés par les programmes Basic.

```
CHARLEX

RWLEX     RREC$       XFN 225094    WREC      XWORD 225095

FINDLEX   FIND      XWORD 225075

KEYWAIT   KEYWAIT$    XFN  82001

FILELEX   FILE?       XFN 225007

DESLEX    CONSTRAST   XFN 225012    INVERSE   XWORD 225013
          INV$        XFN 225014    PAINT     XWORD 225015
```

```
 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  ID#E1  624 octets

     0123456789ABCDEF sm

000: 34841425C4548502 35E
001: 802E000221525078 6AD
002: 5E4001E000000000 9F1
003: FE0000000800001F D4B
004: F31BF961400032BF 0DE
005: 38F14A11DB10AD23 478
006: 07D532BFB8FD7911 82B
007: 11AD754D7A101743 BAE
008: 11014D1CB15D0000 F19
009: 71450375FF864834 296
00A: 5655581008355654 5ED
00B: 5810002455565870 93A
00C: 0026555658700836 C8C
00D: 5556581008364545 FE2
00E: 4A30000A49724000 335
00F: 0808094A2C180814 69E
010: A464242008355455 9F8
011: 581000054C714000 D3E
012: 0C3142404C700832 09A
013: 41414A70002078A0 3F2
014: 2F30000000000000 71D
015: 0000000000000000 A2D
016: 0000000000000000 D3D
017: 0000000000000000 04D
018: 0000000000000000 35D
019: 0000000000000000 66D
01A: 0000000000000000 97D
01B: 0000000000000000 C8D
01C: 0000000000000000 F9D
01D: 0000000000000000 2AD
01E: 0000000000000000 5BD
01F: 0000000000000000 8CD
020: 0000000000000000 BDD
021: 000000000000080C F08
022: 1A28080008080A2C 272
023: 180008040E340800 5BB
024: 08001E3018000000 8F5
025: 0000000000000000 C05
026: 0000000000000000 F15
027: 0000000000000000 225
028: 0201000000010200 53B
029: 0000000201020000 850
02A: 0001000100000002 B64
02B: 0102010000000000 E78
02C: 0000000000000000 188
02D: 045E755142400101 4D4
02E: 0101010000000000 7E7
02F: 0000000000000000 AF7
030: 0000070507000000 E1A
031: 000000000834444C4 158
032: 44400D7901112D70 4B8
033: 050D750509700000 802
034: 0D70000000384540 B45
035: 4020014E322E3140 E99
036: 084E794142400000 1E9
037: 000000000002E4559 527
038: 3200000000000000 83C
039: 0000000000000026 B54
03A: 5556587008365556 EB3
03B: 5810083645464830 204
03C: 0832414248700024 545
03D: 5655587008345655 8A2
03E: 5810083446454830 BF1
03F: 0C3042414C700024 F46
040: 5556587008355654 2A3
041: 5810083546444830 5F2
042: 0C3142404C700025 948
043: 5455587008355455 CA2
044: 5810083544454830 FF0
045: 0C3140414C700875 352
046: 14141870000A4972 6A3
047: 40000E3159454E30 A03
048: 0C7A0F7949400024 D7B
049: 5554587000084A71 0D7
04A: 40000C523A262D10 438
04B: 0424587458400875 78F
04C: 1415187000094A70 ADF
04D: 4000083544454830 E23
04E: 0C3140414C300C74 18B
04F: 5655545000054C71 4E2
050: 40000            5DB

RWLEX    ID#E1  494 octets

     0123456789ABCDEF sm

000: 2575C45485020202 356
001: 802E001221525078 6A6
002: 1E3001EE5F500000 A1A
003: F020000000000000 D42
004: 0FD100FD00FA000D 0E2
005: 92525543442E5775 44C
006: 255434F51FF8FAB6 802
007: 308FEA2305B38F46 BA0
008: 0308F625308F9DF3 F35
009: 0870718FEA230571 2B2
00A: 7B62529700385480 62A
00B: 53E208DB2E208F22 9C2
00C: 95031C28F5EC208F D64
00D: 2295031C28F5EC20 0EC
00E: 7B22A4B700160E0D 477
00F: CFFF77FFF8F871F0 870
010: 1618F681F0161137 BDE
011: 0684776F1B0E604C F6F
012: C0713575E126B307 2E8
013: 280110131DB0614B 64C
014: 31F0966E08F5A5C1 9ED
015: D94118FBC6315E31 D8B
016: 7FD6068F13DB0D23 139
017: 20028A64207DA07D 4B9
018: 77A81F35404148FB 84C
019: F1908D84A808D91F BF9
01A: B022D245272610FB F78
01B: 1049193BAE137108 2EF
01C: 7B41C52405008915 657
01D: 3892038845270312 9A8
01E: 6B307CF045170211 D18
01F: 9760570884A02480 07D
020: C02380C12231FFDA 41A
021: 208DA939042BCC22 7B0
022: 1360672B07C7F110 B24
023: 13117FDB0614B31F EB4
024: 0966C08F5A5C1411 23F
025: 8FBC63158717FD81 5EB
026: 33D232012EA8F064 96D
027: A1D68B3D51091351 CF3
028: 7FD407D77380FF44 0A0
029: 0478111131AF234F 411
02A: 0002155713310124 744
02B: 8F8A410071341191 AAB
02C: 358FE83B18DC32F0 E5E
02D: 6ACE8DD4490248DD 224
02E: D41086C62061361B 599
02F: 244F215641340794 8F7
030: AC0B464602402030 C5C
031: 61371F0F7F21451D FE8
032: 5F0715571E508F15 36B
033: 171D51D980F526A8 6FF
034: 2550A0E27A82A0E1 A8C
035: 5D7042032CFB8FAB E43
036: 81154331FFD5D0E4 1DF
037: 15F596A1296561BF 576
038: 6F69E2C0F6F6B625 932
039: 2117A5AD33320063 C9C
03A: 7E175147D5071351 00B
03B: 7413706147C9061F 37C
03C: 508F2153717F15F7 702
03D: D527B06440052690 A6A
03E: E2080D51E5F7F157 E0D
03F: 71D0F06147135070 171
040: 1                1A3

FINDLEX  ID#E1  317 octets

     0123456789ABCDEF sm

000: 6494E444C4548502 377
001: 802E002221525078 6C8
002: F72001EB4B400000 A39
003: F710000000000000 D67
004: 002000776494E444 0BF
005: B41FF8DD97309FFF 4A0
006: F8F681F08F13DB08 859
007: A8841BB88F214013 BE4
008: 71351641448F8BB8 F6A
009: 18FEEB604018F627 30D
00A: 705D031F38D39390 68E
00B: 8F99970511203 38E A0A
00C: 008DA93901F8E7F2 DAF
00D: D215F38F4EFF05CD 183
```

```
00E: 137D51378F130018 4ED
00F: B370D4131DB8FD3D 8A5
010: E013713510B8FF6F C41
011: 4020AF21BF85F214 FD3
012: 21A1C6F146E281EE 37C
013: 6D4EAD8C414610A1 721
014: 351B098F21461091 A8B
015: 84146EA4D4CE80D0 E38
016: F6D7542D90613706 1BE
017: 137DBD5701107135 531
018: 07D5562CD1191344 8AC
019: 5114A17114F9629C C29
01A: CD52F112131112AF FB1
01B: 2D6133EA2011B135 329
01C: 8A8671038F96F401 6B3
01D: BF85F21461341F1C A4E
01E: 6F2AD0143EA11BCA E07
01F: 81C100D9067C00B1 184
020: C3E3B1E3FF071358 531
021: FE0C1007D58F6179 8D5
022: 01188F1C6908F898 C67
023: 108DE73008FC2DE0 00F
024: D78F13001208B360 382
025: 696E11B13710B8FD 720
026: A67011B1354908D8 A9C
027: 4A80639E15271577 E13
028: 9760016F17FCD5AE 1C6
029: 152115719160003  4CF

KEYWAIT  ID#52  55 octets

     0123456789ABCDEF sm

000: B454957514944502 366
001: 802E003221525078 6B8
002: 3700025101000000 9DB
003: F710000000000000 D09
004: 0E1000FFB4549575 094
005: 14944542101FF001 3F4
006: 361081371098F2C6 764
007: 0045111913511813 AA0
008: 48D8ACA18F127006 E3C
009: BDF              F0B

FILELEX  ID#E1  64 octets

     0123456789ABCDEF sm

000: 6494C454C4548502 376
001: 802E003221525078 6C8
002: 580001E707000000 A09
003: F710000000000000 D37
004: 091000F96494C454 0A7
005: F3701FF411136068 422
006: F59B9057097C908D 7C5
007: A9390137068F77F9 B57
008: 0AF0480B44BF4071 EE8
009: 351CF15170713480 263
00A: C32F0            386

DESLEX   ID#E1  281 octets

     0123456789ABCDEF sm

000: 445435C454850202 35A
001: 802E004221525078 6AD
002: 732001EC0F000000 A08
003: F230000000000000 D33
004: 0CE000F310160000 0A0
005: 4200B000FF20DF00 420
006: 0FF34F4E44525143 7AE
007: 545C0D94E4655425 B34
008: 3554D0794E46542E EBC
009: 09051494E445F01F 23B
00A: FF30001400034C50 59E
00B: 001F401E27C20310 901
00C: 61E002E7E1031C41 C7D
00D: E003E70108D84A80 003
00E: 8D303500314BBEC1 38F
00F: 49171CECE8AEDE01 75D
010: 4111371351088F83 AB9
011: DB0D68AA607BCF12 E7E
012: 81358DC32F000136 1F3
013: 1081BEF3E2D015A0 58F
014: 8FB13B11201188D9 918
015: 12F0888331361081 C73
016: 371091358FC8CB01 FFF
017: 0A8F322B11228F32 380
018: 2B112210317F8FC1 6FE
019: DB08F322B1123313 A79
01A: 89E65112231709E6 DEF
01B: 9012267006080034D 143
01C: 50001B442E28B680 4B4
01D: 1A4A1E31D28B6801 844
01E: A401E136C2C21361 BBE
01F: 4A11AAE5AF230196 F5D
020: 9C0A66A6D64FFAE7 334
021: 0E62123AE811310A 6AC
022: AEB969A00E6E6A00 A64
023: BEE0E661481128FB E12
024: 13B1120119137118 15C
025: 8D912F0          2F1
```



PPC PARIS CHAPTER

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 ? ____________________

Je souhaite adhérer au club PPC Paris Chapter 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 Chapter,
par [ ] chèque bancaire, [ ] chèque postal ou [ ] mandat lettre

Eventuellement: je m'abonne à compter du |0|1|/|_|_|/19|_|_|

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

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

La maquette de ce numéro a été préparée et réalisée par Pierre David, Jean-Jacques Dhénin et Janick Taillandier, grâce à un système comprenant un HP71B, deux lecteurs de disquettes HP9114A, une imprimante HP2225B et une imprimante LaserJet.

Numéro ISSN : 0762 - 381X

ENGLISH SUMMARY

JPC 45 - JUNE 1987

The Journal has been produced since JPC 35 by a new team, with Pierre David, Jean-Jacques Dhénin and Janick Taillandier. A lot of work was needed to produce the Journal in time and improve its presentation. We are proud of what has been accomplished. The same team will now try to improve the general activity of the Club. The mail box is a first step to get better answer delays. Please, use only this address :

PPC Paris
B.P. 604
75028 Paris Cedex 01
France

We are happy to announce the availability of a new software product for the HP-71. It is a Lex file which provides graphic functions for the ThinkJet printer or any other printer using "raster" graphics (QuietJet or LaserJet). This is a wonderful realization written by Pierre David. You will find a list of available functions on page 4.

In the HP-28 column, we find a program which explains the use of the infinite RPN stack to solve $ax^2+bx+c=0$. It is important to note that people are now beginning to write to JPC about this new machine : our members are very interested in it.

The HP-41 program provides matrix utilities (matrix trace, identity matrix and pointer conversions) using the Advantage Rom. They ease the keyboard use of the Rom capabilities.

The Forth program provides us with a new, enhanced VLIST type word. It handles vocabularies (CONTEXT and CURRENT). Some utility words are included to improve access to Basic from Forth.

As promised last month by Jacques Baudier, here is the first article in a series introducing assembly language on the HP-71.

Two Lex files for the HP-71 in this issue : the first one gives RREC$ and WREC. These keywords read and write sectors from or to a mass-memory device. This allows a clean and easy access to these devices without being obliged to use low-level HP-IL commands.

FIND was already published in JPC 31 (February 86). This keyword allows you to search for a string in a Basic program as does FETCH on the HP-75. This new, debbuged version moves the cursor to the beginning of the matching string.

Then we have a "Lex editor", written in Basic but using Forth, which allows some actions on a Lex : nice example of the capabilities provided by the Basic / Forth interface.

François Duret-Lamouroux, a beginner programmer, presents us a small navigation program on page 25.

At the end of the Journal is a curve fitting program adaptated to the HP-71 from an old HP-75 program published in PPC Computer Journal. Michel Martinet took great care to program a nice and pleasant user interface. More, this program uses no Lex file.

Until next month,

Happy Programming and JPC reading !