COURS HP 49G

QU'EST CE QUE LA PILE ?

C'est comme une pile d'assiettes où l'on empile et on dépile des variables, des programmes,... ou des objets de manière générale (sachant qu'un objet peut être n'importe quoi). La pile dispose de différents niveaux commençant par le niveau 1 et va en augmentant jusqu'à saturation de la mémoire. Le niveau 1 est toujours le niveau le plus haut ou le plus prioritaire. Quand on empile un nouvel objet sur la pile, celui-ci est placé systématiquement sur le niveau 1 de la pile et les autres objets sont décalés d'un niveau : c'est à dire que l'objet qui se trouvait au niveau 1 sera déplacé au niveau 2, celui du niveau 2 au niveau 3, etc. Pour exemple, si l'on tape 5 7 ENTER, l'entier 57 sera déposé sur le niveau 1 de la pile. Maintenant, s'il l'on tape 2 3 ENTER, l'entier 57 sera déplacé au niveau 2 pour laisser place à l'entier 23 qui se retrouvera au niveau 1 de la pile.

Maintenant, pour réaliser une opération sur les objets qui se trouvent sur la pile, il suffit d'appeler les opérateur. Si l'on veut additionner les deux entiers qui se trouvent sur la pile (ici, 57 et 23), on appuie simplement sur la touche + pour avoir 80 sur le niveau 1 de la pile. Des commandes permettent de faire des choses très utiles sur les objets qui se trouvent sur la pile :

CLEAR : efface toute la pile,
DEPTH : donne le nombre de niveaux de la pile occupés,
DUP : duplique l'objet du niveau 1 de la pile,
DUPDUP : duplique deux fois l'objet du niveau 1 de la pile,
DUP2 : duplique les deux premiers objets de la pile,
DUPN : duplique les N premiers objets de la pile,
DROP : efface le premier objet de la pile,
DROP2 : efface les deux premiers objets de la pile,
DROPN : efface les N premiers objets de la pile,
NDUPN : duplique N fois l'objet du niveau 1 de la pile,
NIP : efface l'objet du niveau 2 de la pile,
OVER : duplique l'objet du niveau 2 de la pile,
PICK : copie l'objet du niveau N de la pile,
PICK3 : copie l'objet du niveau 3 de la pile,
ROLL : déplace l'objet du niveau N sur le niveau 1 de la pile,
ROLLD : déplace l'objet du niveau 1 sur le niveau N de la pile,
ROT : déplace l'objet du niveau 3 sur le niveau 1 de la pile,
SWAP : intervertit la position des deux premiers objets de la pile,
UNROT : déplace l'objet du niveau 1 sur le niveau 3 de la pile.

COMMENT CREER UN PROGRAMME ?

Appuyer sur la touche ‹‹›› et saisir le corps du programme entre les délimiteurs. Une fois le programme terminé, appuyer sur ENTER pour le poser sur le niveau 1 de la pile. Il faut ensuite lui donner un nom : pour cela, on appuie sur la touche ' et on entre le nom après avoir sélectionner le mode ALPHA. Ensuite un simple appui sur STO▶ permet de stocker le programme.

COMMENT LANCER UN PROGRAMME ?

Appuyer sur la touche VAR et appuyer ensuite sur la touche de menu correspondant au programme : F1 à F6. Si le programme ne se trouve pas sous l'une de ces six touches, il suffit d'appuyer sur la touche NXT pour avoir les six propositions suivantes.

COMMENT MODIFIER UN PROGRAMME ?

Rappeler le nom du programme à modifier sur le niveau 1 de la pile et appuyer sur TOOL et F1. Pour le re-stocker, on rappelle à nouveau le nom qu'on lui a déjà attribué et on le stocke comme c'est indiqué dans le second paragraphe.

ENTREE DE VARIABLES

• Pour entrer une variable globale (variable qui pourra être utilisée par tous les programmes), on procède comme pour le stockage d'un programme : on pose la variable sur le niveau 1 de la pile et on la stocke en lui affectant un nom.

• Pour entrer une variable locale, on procède comme ceci :

« *valeur1 valeur2 ...* **¨** *variable1 variable2 ...* **«** *(corps du programme)* **» »**

Comme exemple très simple, stockons respectivement 57 et 23 dans V1 et V2 pour les soustraire, ce qui donne :

« 57 23 ¨ V1 V2 « V1 V2 + » »

Ici, 57 sera posé au niveau 2 de la pile et 23 au niveau 1, et après exécution du programme le résultat doit être 34. Il est possible de poser préalablement les entiers sur la pile et lancer le programme après, et le programme devient :

« ¨ V1 V2 « V1 V2 + » »

LES TESTS

• SI (condition) ALORS (on exécute une suite de commandes) FIN
Ce qui correspond à :

IF *(condition)* **THEN** *(suite de commandes)* **END**

Autre méthode :

(condition) **«** *(suite de commandes)* **» IFT**

• SI (condition) ALORS (on exécute une suite de commandes) SINON (on exécute une autre suite de commandes) FIN
Ce qui correspond à :

IF *(condition)* **THEN** *(suite de commandes)* **ELSE** *(autre suite de commandes)* **END**

Autre méthode :

(condition) **«** *(suite de commandes)* **» «** *(autre suite de commandes)* **» IFTE**

Les tests possibles dans la condition sont : **== ‹ < ‰ > Š SAME NOT AND OR XOR**.
D'autres tests sont aussi disponibles dans le manuel de l'utilisateur.

[phpamart@nordnet.fr]

LES BOUCLES FINIES

• POUR(borne_inférieure,borne_supérieure) (on exécute une suite de commandes) FIN
Ce qui correspond à :
(borne inférieure) (borne supérieure) **START** *(on exécute une suite de commandes)* **NEXT**

• POUR(borne_inférieure,borne_supérieure,pas_du_compteur) (on exécute une suite de commandes) FIN
Ce qui correspond à :
(borne inférieure) (borne supérieure) **START** *(on exécute une suite de commandes) (pas du compteur)* **STEP**

• POUR(nom_de_compteur,borne_inférieure,borne_supérieure) (on exécute une suite de commandes) FIN
Ce qui correspond à :
(borne inférieure) (borne supérieure) **FOR** *(nom_de_compteur) (on exécute une suite de commandes)* **NEXT**

• POUR(nom_de_compteur,borne_inférieure,borne_supérieure,pas_du_compteur) (on exécute une suite de commandes) FIN
Ce qui correspond à :
(borne inférieure) (borne supérieure) **FOR** *(nom de compteur) (on exécute une suite de commandes) (pas du compteur)* **STEP**

LES BOUCLES INFINIES

• TANT QUE (condition) (on exécute une suite de commandes) FIN
Ce qui correspond à :
WHILE *(condition)* **REPEAT** *(on exécute une suite de commandes)* **END**

• FAIRE (on exécute une suite de commandes) TANT QUE (condition) FIN
Ce qui correspond à :
DO *(on exécute une suite de commandes)* **UNTIL** *(condition)* **END**

La structure DO permet d'exécuter la suite de commandes avant le test de la condition, c'est à dire que le corps de boucle est exécuté au moins une fois, contrairement à la commande WHILE qui teste la condition avant exécution du contenu de la boucle.

UN PREMIER PROGRAMME !

Réalisons un programme très simple qui calcule le signe d'un nombre réel qui est déjà posé sur le niveau 1 de la pile. Pour faire cela, il existe plusieurs méthodes. Commençons par la première qui semble la plus intuitive :

```
« ¨ nombre
  « IF nombre 0 > THEN "Positif" END
    IF nombre 0 == THEN "Nul" END
    IF nombre 0 < THEN "Négatif" END » »
```

Utilisons maintenant un peu la pile pour faire la même opération (afin de ne pas utiliser de variable) :

```
« IF DUP 0 > THEN "Positif"
  ELSE
    IF DUP 0 < THEN "Négatif"
    ELSE "Nul" END
  END SWAP DROP »
```

Utilisons aussi les tests contractés de type IFTE :

```
« DUP 0 > « "Positif" DROP » « 0 < "Négatif" "Nul" IFTE » IFTE »
```

QUELQUES COMMANDES TRES UTILES

ABS : valeur absolue, module d'un complexe,
ACOS : arc cosinus,
AND : opérateur ET logique,
ARG : argument d'un complexe,
ASIN : arc sinus,
ATAN : arc tangente,
BEEP : son de fréquence F en Hz et de durée D en secondes,
BIN : base binaire,
BLANK : crée un graphique vierge,
BOX : trace un rectangle,
BYTES : donne la taille et le CRC d'un objet,
B¨R : convertit un entier binaire en réel,
CEIL : arrondi un réel à l'entier supérieur,
CHR : donne un caractère en fonction de son code ASCII,
CLLCD : efface l'écran,
COMB : calcul les combinaisons,
CONJ : conjugué d'un complexe,
COS : cosinus,
CRDIR : crée un répertoire,
C¨R : convertit un complexe en deux réels,
DATE : donne la date,
DDAYS : écart entre deux dates,
DEC : base décimale,
DEG : mode angulaire en degrés,
DISP : affiche l'objet sur la ligne N (1 à 7),
ERASE : efface l'environnement graphique,
EVAL : évalue un objet,
EXP : exponentielle,
EXPAND : développe une fonction,
FACT : factorielle,
FACTOR : factorisation d'équation,
FLOOR : arrondi un réel à l'entier inférieur,
FP : partie fractionnaire d'un réel,
FREEZE : gèle certaines zones d'écran (ex. : **7 FREEZE**),
GET : extrait un élément d'une liste,
HALT : interrompt un programme (qui reprend avec **CONT**),
HEAD : 1ER élément d'une liste, 1ER caractère d'une chaîne,
HEX : base hexadécimale,
HOME : retour à la racine (répertoire HOME),
IM : partie imaginaire d'un complexe,
IP : partie entière d'un réel,
ISPRIME? : un entier est-il premier ???,
KEY : donne le code d'une touche,
KILL : arrête un programme,
LASTARG : récupère le dernier objet avant exécution d'une fonction ou une commande,

Philippe PAMART – Cours HP 49G © 2000
[phpamart@nordnet.fr]

LINE : trace une ligne,
LIST¨ : défait une liste,
¨LIST : construit une liste,
LN : logarithme népérien,
LOG : logarithme décimal,
MAX : maximum de deux nombres,
MEM : donne la mémoire libre restante,
MIN : minimum de deux nombres,
MOD : reste d'une division,
NEG : change le signe d'un nombre,
NOT : opérateur NON logique,
NUM : donne la valeur ASCII d'un caractère,
¨NUM : numérise une équation,
OCT : base octale,
OFF : éteint la machine,
OR : opérateur OU logique,
PATH : donne le chemin d'accès au répertoire,
PGDIR : efface un répertoire,
PIXOFF : éteint un pixel,
PIXON : allume un pixel,
POS : donne la position d'un objet dans une liste,
PURGE : efface une variable,
PUT : place un élément dans une liste,
¨Q : convertit un réel en fraction,
RAD : mode angulaire en radians,
RAND : nombre aléatoire compris en 0 inclus et 1 exclus,
RCL : rappelle le contenu d'une variable,
RE : partie réelle d'un nombre,
REVLIST : inverse l'ordre des éléments d'une liste,
RND : fonction arrondi,
R¨B : convertit un réel en entier binaire,
R¨C : convertit deux réels en complexe,
SIN : sinus,
SIZE : taille d'un objet,
SORT : tri d'une liste par ordre croissant,
STO : stocke un objet,
STR¨ : convertit une chaîne en objet,
¨STR : convertit un objet en chaîne,
SUB : extrait un sous-ensemble de liste ou de chaîne,
TAN : tangente,
TIME : heure,
TYPE : type d'un objet,
UPDIR : remonte sur le répertoire parent,
VARS : donne la liste des variables du répertoire courant,
WAIT : attente en secondes,
XOR : opérateur OU exclusif.

QUELQUES EXEMPLES COMMENTES…

Afficher un texte : **TEXTE**

```
« CLLCD "HP49G" 4 DISP 7 FREEZE »
```

On efface l'écran avec CLLCD, on pose ensuite la chaîne de caractères "HP49G" sur le niveau 1 de la pile, on l'affiche sur la 4ème ligne de l'écran et on gèle la totalité de l'écran.

Addition de 3 nombres : **ADD3**

Pour entrer une variable au sein d'un programme, il existe deux possibilités : soit en utilisant les variables globales, soit en utilisant les variables locales, ces dernières étant les plus utilisées (voir plus haut pour les structures).

1ère solution : on utilise la pile,

```
« ¨ A B C
  « A B C + + » »
```

2ème solution : on utilise les expressions,

```
« ¨ A B C
  « 'A+B+C' ¨NUM » »
```

Ici, on suppose que 3 nombres sont déjà posés sur la pile, comme on peut le voir dans la figure 1. La figure 2 correspond à l'état de la pile après exécution du programme.

fig. 1

Le nombre 25 ira dans la variable locale A, 30 dans la variable locale B, et 45 dans la variable locale C. C'est à dire que l'ordre des objets posés sur la pile correspond à l'ordre des variables à l'intérieur d'un programme.

fig. 2

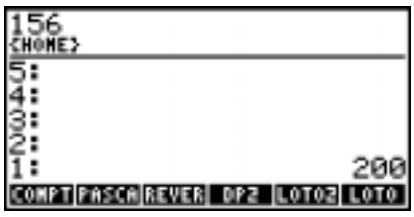
Affiche l'état d'un compteur : **COMPTEUR**

Imaginons que l'on veuille un compteur qui tourne jusque 200, alors il faut préalablement poser cet entier sur la pile :

Le programme est assez simple :

```
« 1 SWAP FOR I I 1 DISP NEXT »
```

Tout d'abord, on inverse les niveaux 1 et 2 pour pouvoir faire la boucle de 1 à 200 dans le bon sens, grâce à la fonction **SWAP**. Comme nom de compteur, on choisit I. Pour afficher le compteur sur la première ligne de l'écran, on utilise la commande **1 DISP**. Et on boucle grâce à **NEXT**.Voici à l'instant t ce que ça peut donner :



Parité d'un nombre entier : **PARITE**

```
« IF 2 MOD THEN "Impair" ELSE "Pair"
END »

« 2 MOD "Impair" "Pair" IFTE »
```

Divisible par 2 : **DP2**

```
« IF 2 MOD NOT THEN
  "Divisible par 2" END »
```

Maximum de 2 nombres sans utiliser MAX : **MAXI**

```
« ¨ A B
  « IF A B > THEN A ELSE B END » »

« IF DUP2 < THEN SWAP END DROP »
```

Multiplication par l'addition : **MULTI**

```
« ¨ A B « 0 1 A START B + NEXT » »
```

Puissance par la multiplication : **PUISS**

```
« ¨ A B « 1 DUP B START A * NEXT » »
```

Puissance par l'addition : **PUISSADD**

```
« ¨ A B
  « 1 DUP B START
      0 1 A START
        OVER + NEXT
    SWAP DROP NEXT » »
```

Calcul d'un racine carrée : **RACINE**

```
« ¨ V I
  « 1 DUP I START
    V OVER / + .5 * NEXT » »
```

Méthode : $U_0=1$
et $U_{n+1}=1/2*(U_n+a/U_n)$

Calcul de Pi par Monte-Carlo : **MONTECAR**

```
« ¨ N
  « 0 1 N START
  IF RAND SQ RAND SQ + 1 <
  THEN 1 + END
  NEXT N / 4 * » »
```

Méthode : on calcule aléatoirement un certain nombre N de couple X et Y dans un carré de 1x1, s'ils appartiennent au cercle C ($X^2+Y^2<1$) alors Pi=4*C/N.

Les bons numéros du LOTO : **LOTO**

```
« {} 1 6 FOR I
  49 RAND * FLOOR 1 +
  IF DUP2 POS NOT
  THEN +
  ELSE DROP I 1 - 'I' STO
  END NEXT SORT »
```

Autre méthode : **LOTO2**

```
« {} DO
  49 RAND * FLOOR 1 +
  DUP2 POS NOT
  { + } { DROP } IFTE
  UNTIL DUP SIZE 6 ==
  END SORT »
```

Inverse des éléments d'une liste : **REVERSE**

```
« DUP SIZE ¨ L T
  « {} T 1 FOR I
    L I GET +
    -1 STEP » »
```

Le triangle de Pascal : **PASCAL**

```
« 0 SWAP FOR I {}
  0 I FOR J I J COMB +
  NEXT NEXT »
```

Le tri à bulles d'une liste : **TRIBULLE**

```
« DUP SIZE ¨ L T
  « 1 T 1 - START
    L HEAD
    2 T FOR I L I GET MIN
    LASTARG MAX NEXT
    T ¨LIST 'L' STO
    NEXT L » »
```

Affichage de la table ASCII : **ASCII**

```
« CLLCD 0 255 FOR I
"Caractère n°" I + " : " +
I CHR + 1 DISP .25 WAIT NEXT »
```

Inversion d'une chaîne de caractères : **INVCHC**

```
« DUP SIZE ¨ C S
  « "" S 1 FOR I
    C I I SUB +
    -1 STEP » »
```

autre version : **INVCHC2**

```
« "" OVER SIZE 1 FOR I
  OVER I I SUB +
  -1 STEP
  SWAP DROP »
```

Code de César : **CESAR**

```
« DUP SIZE ¨ C S
  « "" 1 S FOR I
    C I I SUB NUM 128 + 256 MOD
   CHR + NEXT ROT » »
```

Inversion majuscules/minuscules : **MAJMIN**

```
« DUP SIZE ¨ C S
  « "" 1 S FOR I
    C I I SUB NUM
    IF DUPDUP 65 Š SWAP 90 ‰ AND
    THEN 32 +
    ELSE
    IF DUPDUP 97 Š SWAP 122 ‰ AND
    THEN 32 - END
    END CHR + NEXT »
```

Calcul d'un nombre premier : **PREMIER**

```
« DUP ƒ CEIL ¨ N D
  « WHILE N D / FP 0 ‹
    REPEAT D 1 - 'D' STO END
    IF D 1. SAME
    THEN "Premier"
    ELSE "Non premier"
    END » »
```

Décomposition en facteurs premiers : **FACTEUR**

```
« 2 ¨ a b
  « { } WHILE a 1 > REPEAT
        IF a b MOD 0 ==
        THEN b + a b / 'a' STO
        ELSE b 1 + 'b' STO
        END END » »
```

Décomposition en facteurs premiers : **FACTEUR2**
(autre méthode : deux boucles WHILE)

```
« 2 ¨ a b
  « { } WHILE a 1 > REPEAT
        WHILE a b MOD 0 ==
        REPEAT b + a b / 'a' STO END
        b 1 + 'b' STO
        END » »
```


Philippe PAMART – Cours HP 49G © 2000
[phpamart@nordnet.fr]

Le système de Martin : **MARTIN**

```
« RAD ERASE { # 0h # 0h } PVIEW
3.14 0 0 1 2500 START
DUP2 R¨C C¨PX PIXON OVER SIN -
3 PICK ROT - NEXT »
```

Méthode : X(0)=Y(0)=0 et a=3.14
X(n+1)=Y(n)-sin(X(n))
Y(n+1)=a-X(n)
Xmin=-20, Xmax=20, Ymin=-20, Ymax=20

Le système Hopalong : **HOPA**

```
« ERASE { # 0h # 0h } PVIEW
.4 1 0 0 0
1 2500 START
DUP2 R¨C C¨PX PIXON OVER DUP SIGN
IF DUP 0 SAME THEN NOT END
SWAP 6 PICK * 5 PICK - ABS
ƒ * - 5 PICK ROT - NEXT »
```

Méthode : c=X(0)=Y(0)=0, b=1 et a=0.4
X(n+1)=Y(n)-signe(X(n))*√(abs(b*X(n)-c))
Y(n+1)=a-X(n)
Xmin=-0.6, Xmax=1.15, Ymin=-0.77, Ymax=1

Le système Kam Torus : **KAMTORUS**

```
« RAD ERASE { # 0h # 0h } PVIEW
0 1.5 FOR I I 3 / DUP
1 150 START
DUP2 R¨C C¨PX PIXON SWAP DUP SQ ROT -
DUP2 1.3 SIN * SWAP 1.3 COS * +
ROT 1.3 SIN * ROT 1.3 COS * -
NEXT DROP2 .1 STEP »
```

Méthode : a=1.3 radians, 0≤orbite≤1.5 avec un pas de 0.1,
150 points par orbite, pour chaque orbite : X(0)=Y(0)=orbite/3
X(n+1)=X(n)*cos(a)+(X(n)²-Y(n))*sin(a)
Y(n+1)=X(n)*sin(a)-(X(n)²-Y(n))*cos(a)
Xmin=-0.57, Xmax=0.62, Ymin=-0.61, Ymax=0.61

Triangle de Sierpinski : **SIERPINSKI**

```
« ERASE { # 0h # 0h } PVIEW
RAND RAND 1 3000 FOR K RAND ¨ N
  « IF N 1. 3 / ‰
    THEN .5 * SWAP .5 * SWAP
    END
    IF N 1. 3 / > N 2. 3 / ‰ AND
    THEN 1 + .5 * SWAP .5 + .5 * SWAP
    END
    IF N 2. 3 / >
    THEN .5 * SWAP 1 + .5 * SWAP
    END
    DUP2 R¨C PIXON
    »
NEXT DROP2 »
```

Xmin=0, Xmax=1, Ymin=0, Ymax=1

Les fractions égyptiennes : **EGYPT**

```
« 0 SWAP
WHILE DUP FXND NIP 1 >
REPEAT DUP INV FXND IDIV2 SIGN +
INV ROT OVER + UNROT - EVAL
END DROP »
```

Méthode : a/n = a/(a*p + r) = 1/(p + 1) + (a – r)/((p+1)*n)

Exemples :
3/4 = 1/2 + 1/4
5/17 = 1/4 + 1/23 + 1/1564

Recherche un nombre (jeu) : **CHERCHE**

```
« IF "NIVEAU"
{ { "1 : 1 à 100" 2 }
  { "2 : 1 à 1000" 3 }
  { "3 : 1 à 10000" 4 } } 1 CHOOSE
  THEN ALOG RAND * FLOOR 1 + 0 ¨ N C
  « DO C 1 + 'C' STO
    "Entrer un entier" "" INPUT
    STR¨ I¨R
    "Coup n°" C + "↵" +
    IF OVER N < THEN "Trop petit"
    ELSE IF OVER N > THEN
         "Trop grand"
          ELSE "Gagné : " N +
          END
    END + MSGBOX
    UNTIL N == END »
  END »
```