BASIC

Daniel Roux

epsitec-system sa
ch. mouette CH-1092 belmont

2ème édition revue et augmentée, 1980

TABLE DES MATIERES

MARQUE: Pour distinguer la lettre "O" du chiffre "0", on a barré le zéro (Ø) dans les cas où la confusion était possible. Dans cette notice, Ø signifie donc zéro et non pas "ensemble vide".

BASIC

Daniel Roux

1. INTRODUCTION

Le BASIC est un langage de programmation très simple, accessible à tout le monde.

Un programme écrit en BASIC comporte un certain nombre de lignes numérotées. Chaque ligne correspond à une instruction. Par exemple, multiplier un nombre par 2, ou bien afficher un texte sur l'écran, etc.

EXEMPLE:

Voici un programme qui calcule le carré d'un nombre:

```
10 INPUT "TAPER UN NOMBRE",X
20 PRINT " LE CARRE VAUT",X*X
```

Au moment de l'exécution de ce petit programme, la première instruction (INPUT "TAPER UN NOMBRE",X) affiche sur l'écran le texte entre guillemets, suivi d'un ?. Le point d'interrogation indique que le système attend que l'on tape un nombre (suivi d'un retour de chariot) sur le clavier. Ce nombre est alors mémorisé par la machine dans une cellule mémoire à laquelle est associé le nom X.

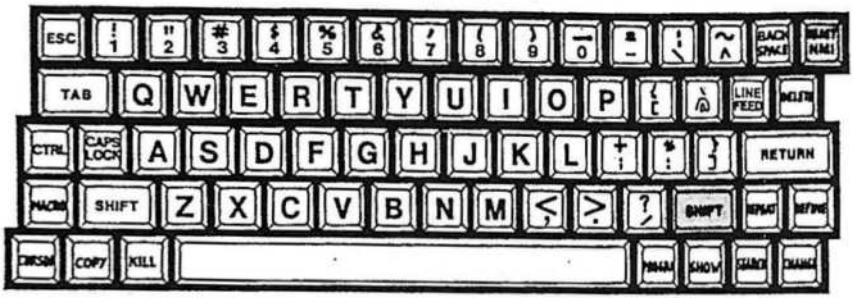
La deuxième instruction (PRINT "LE CARRE VAUT",X*X) affiche sur l'écran le texte entre guillemets suivi du produit X*X, ce qui est bien le carré du nombre X.

Les deux lignes ont été numérotées arbitrairement 10 et 20, mais n'importe quels nombres dans l'ordre croissant auraient fait l'affaire.

Généralement les lignes sont numérotées de dix en dix pour permettre après-coup l'insertion des lignes oubliées.

1.1 COMMENT METTRE EN MARCHE LE SMAKY

1. Enclencher le SMAKY
2. Appuyer sur [SHIFT], puis, en maintenant cette touche pressée, appuyer sur [RESET NMI]



3. Vérifier que la touche [CAPS LOCK] est enfoncée

4. Si le SMAKY est équipé d'un module LOADROM, taper la séquence suivante sur le clavier:

(Barre d'espacement)

On voit apparaître sur l'écran: CHECKSUM: 177 643 (pour la version de mars 79).
Taper ensuite sur n'importe quelle touche autre que [RESET NMI] pour démarrer le BASIC.

Si le SMAKY est équipé d'un floppy disque:

- Insérer une disquette contenant le BASIC
- Faire un RESET:

- Taper la date, par exemple 5.10.80:

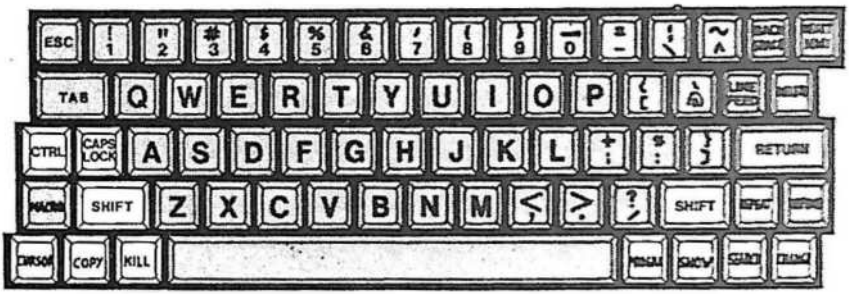
- Après environ 10 secondes, lorsque le système s'est initialisé, taper:

- Le SMAKY est alors prêt à travailler en BASIC.

1.2 COMMENT UTILISER LE SMAKY

Le SMAKY a un clavier de 69 touches réparties en deux catégories:

1) Les touches à action immédiate



2) Les touches qui n'ont une action que si elles sont associées à une touche de la catégorie 1.

La deuxième instruction (PRINT "LE CARRE VAUT",X*X) affiche sur l'écran le texte entre guillemets suivi du produit X*X, ce qui est bien le carré du nombre X.

Les deux lignes ont été numérotées arbitrairement 10 et 20, mais n'importe quels nombres dans l'ordre croissant auraient fait l'affaire.

Généralement les lignes sont numérotées de dix en dix pour permettre après-coup l'insertion des lignes oubliées.

1.1 COMMENT METTRE EN MARCHE LE SMAKY

1. Enclencher le SMAKY
2. Appuyer sur [SHIFT], puis, en maintenant cette touche pressée, appuyer sur [RESET NMI]

3. Vérifier que la touche [CAPS LOCK] est enfoncée

4. Si le SMAKY est équipé d'un module LOADROM, taper la séquence suivante sur le clavier:

(Barre d'espacement)

On voit apparaître sur l'écran: CHECKSUM: 177 643 (pour la version de mars 79).
Taper ensuite sur n'importe quelle touche autre que [RESET NMI] pour démarrer le BASIC.

Si le SMAKY est équipé d'un floppy disque:

- Insérer une disquette contenant le BASIC
- Faire un RESET:

- Taper la date, par exemple 5.10.80:

- Après environ 10 secondes, lorsque le système s'est initialisé, taper:

- Le SMAKY est alors prêt à travailler en BASIC.

1.2 COMMENT UTILISER LE SMAKY

Le SMAKY a un clavier de 69 touches réparties en deux catégories:

1) Les touches à action immédiate

2) Les touches qui n'ont une action que si elles sont associées à une touche de la catégorie 1.

La touche SHIFT doit être maintenue pressée pour introduire un caractère spécial, noté sur la moitié supérieure de certaines touches.

Pour le BASIC, la touche CAPS LOCK doit être enfoncée.

Chaque ligne d'un programme BASIC doit être tapée sur le clavier. Lorsque la ligne est terminée, il faut appuyer sur RETURN.

Les possibilités de correction sont les suivantes:

DELETE — permet d'effacer le dernier caractère entré

CURSOR D — permet de reculer le pointeur dans la ligne.
Il est alors possible par exemple, de rajouter une lettre ou un mot oublié.

CURSOR D veut dire qu'il faut d'abord appuyer sur CURSOR, puis, en maintenant cette touche pressée, qu'il faut appuyer sur D.

CURSOR F — permet d'avancer le pointeur dans la ligne

CURSOR R — positionne le pointeur au début de la ligne

CURSOR C — positionne le pointeur à la fin de la ligne

BACK SPACE — permet d'effacer le caractère sous le pointeur

Par exemple:

```
100 PRINT "LA VALEUR EST",M
```

DELETE — efface le "A" de "VAULEUR"

BACK SPACE — efface le "U" de "VAULEUR"

CURSOR KILL D — permet d'avancer le pointeur en détruisant (comme BACK SPACE)

CURSOR KILL D veut dire qu'il faut appuyer sur KILL et CURSOR dans n'importe quel ordre, puis, en maintenant ces deux touches pressées, qu'il faut appuyer sur D.

CURSOR KILL F — permet de reculer le pointeur en détruisant (comme DELETE)

CURSOR KILL R — détruit depuis le début de la ligne jusqu'au pointeur

CURSOR KILL C — détruit depuis le pointeur jusqu'à la fin de la ligne

Il est toujours possible de taper les trois premières lettres d'un nom d'ordre au lieu du tout.

Par exemple:

PRI est équivalent à PRINT
RAN est équivalent à RANDOMIZE
DIR est équivalent à DIRECTORY
etc.

Il est possible de mettre plusieurs ordres sur une seule ligne en les séparant par un ":" (SHIFT ;).

Essayer maintenant de taper le petit programme vu au paragraphe 1.

Lorsque c'est fini, il faut taper encore RUN♪ pour faire exécuter le programme. Si tout est OK, l'écran montre:

```
TAPER UN NOMBRE ?
```

Il vous faut alors taper un nombre, suivi d'un RETURN

L'écran montre alors, dans le cas où le nombre choisi est 12:

```
TAPER UN NOMBRE?12
LE CARRE VAUT    144
HELLO
>
```

Le caractère > signifie que l'exécution du programme est terminée et que le SMAKY est de nouveau prêt à recevoir vos ordres.

1.3 LES DEUX MODES DE FONCTIONNEMENT DU BASIC:

Le BASIC permet deux modes de fonctionnement.

1) Le BASIC est utilisé pour éditer, puis pour exécuter des programmes (c'est ce que nous avons fait dans l'exemple précédent).
2) Le BASIC permet l'exécution immédiate des instructions, ce qui correspond à l'utilisation comme calculatrice.

Pour qu'une instruction soit exécutée immédiatement, il suffit qu'elle ne soit pas précédée d'un numéro de ligne.

A titre d'exercice, nous allons essayer le programme vu au paragraphe 1 en mode direct (mode 2). Il suffit de taper:

```
INPUT "TAPER UN NOMBRE",X♪
```

L'écran montre:

```
TAPER UN NOMBRE?
```

Il vous faut alors entrer un nombre quelconque suivi d'un RETURN par ex. 12

L'écran montre alors:

```
>
```

ce qui indique que le SMAKY est prêt à exécuter un autre ordre.

Il faut alors taper la deuxième ligne:

```
PRINT "LE CARRE VAUT", X*X↵
```

L'écran montre, par exemple

```
LE CARRE VAUT 144
>
```

Le SMAKY attend un nouvel ordre.

Il est important de remarquer qu'en mode direct, les ordres ne sont pas mémorisés. En effet, si vous voulez calculer le carré d'un nombre (en mode direct), il vous faut tout retaper! Par contre, les valeurs des variables précédemment calculées (par exemple X) sont conservées.

Il peut paraître étonnant de vouloir utiliser le SMAKY de cette façon, mais cela permet de comprendre le fonctionnement d'un instruction. C'est pourquoi nous utiliserons beaucoup ce mode d'exécution immédiate dans cette notice.

2. ORDRES DIRECTS

Nous allons maintenant étudier chaque instruction en détail.
Pour commencer, nous étudierons les ordres qui ne peuvent être exécutés qu'en mode direct.

2.1 RUN, XEQ

RUN ♪ permet d'exécuter un programme depuis la première ligne.

10 RUN ♪ est évidemment incorrect.
L'ordre RUN ne peut être effectué qu'en mode direct.

Par exemple:

```
10 RUN ♪
SYNTAX ERROR IN LINE 10
READY
>
```

L'ordre RUN met à zéro toutes les variables avec de commencer l'exécution du programme. Nous verrons plus loin ce qu'est une variable.

L'ordre XEQ est identique à RUN, sauf que les variables ne sont pas mises à zéro.

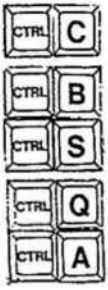
La touche 

permet d'interrompre l'exécution d'un programme.
Pour indiquer quelle ligne du programme était exécutée au moment de l'interruption, le message suivant apparaît sur l'écran:

```
STOP
STOP AT LINE 1225
READY
> 
```

Dans cet exemple, on sait que l'on était à la ligne 1125.

De plus, les touches suivantes ont des actions particulières lorsqu'un programme BASIC est exécuté:



action identique à 

(sauf pour l'ordre ON ESC vu plus loin)

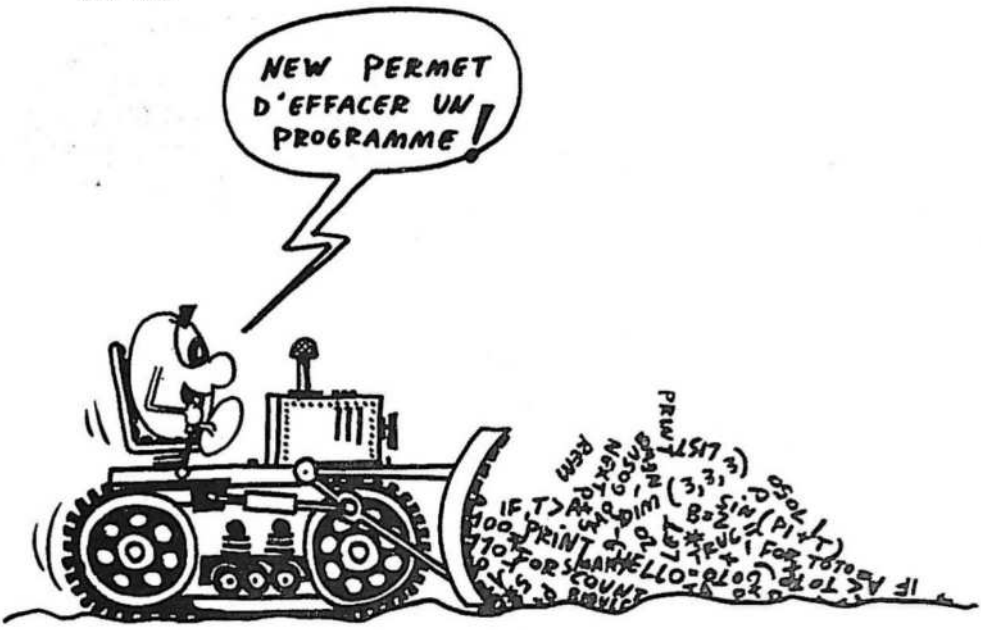
efface l'écran

supprime l'affichage sur l'écran (le programme continue cependant à être exécuté)

autorise l'affichage sur l'écran

une première pression arrête le programme, une deuxième pression continue l'exécution du programme.

2.2 NEW



NEW ♪ efface le programme écrit en BASIC et remet à zéro toutes les variables.

Par mesure de sécurité, le SMAKY répond

ARE YOU SURE ?

Si c'est bien ce que l'on veut faire, il faut répondre [Y] (yes).

Si ce n'est pas ce que l'on veut faire, il suffit de taper sur n'importe quelle touche autre que [Y].

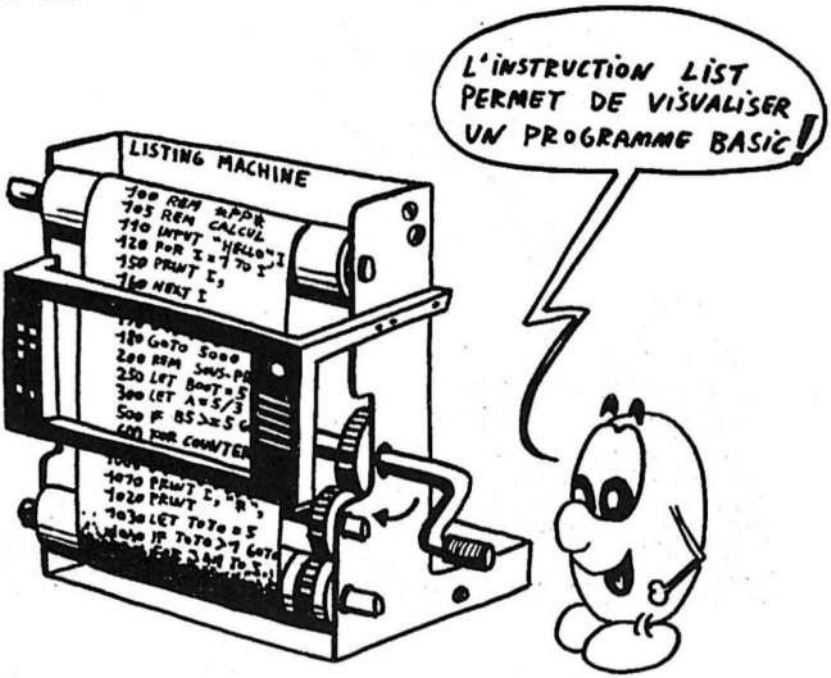
Pour effacer une seule ligne, il suffit de taper son numéro suivi d'un retour de chariot (on crée une ligne vide).

RECOMMANDATION IMPORTANTE:

Utiliser l'instruction NEW avant de taper l'un ou l'autre des petits programmes d'essai que vous trouverez dans cette notice.

NEW *♪ efface tout le programme écrit en BASIC, mais ne détruit pas les variables.

2.3 LIST



LIST↵ visualise sur l'écran tout le programme BASIC.

LIST 150↵ permet de lister seulement la ligne 150.

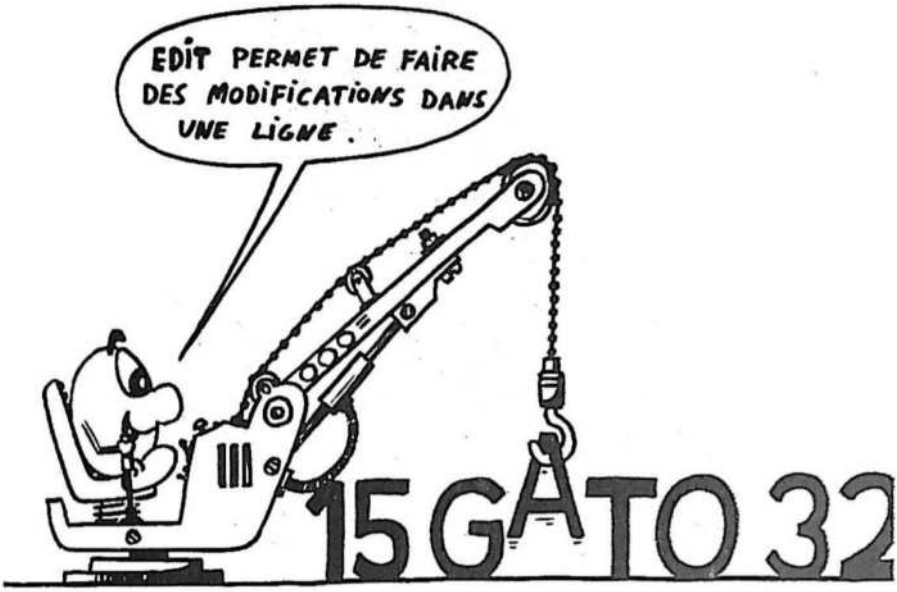
LIST 100,200↵ permet de lister toutes les lignes comprises entre 100 et 200.

LIAST 100,↵ permet de lister depuis la ligne 100 jusqu'à la fin.

LIST ,200↵ permet de lister depuis le début jusqu'à la ligne 200.

ESC permet d'interrompre un listage sur l'écran.

2.4 EDIT



EDIT 100 ♪ permet de modifier la ligne de programme BASIC numéro 100.
On voit apparaître sur l'écran la ligne 100, qu'il est alors possible de modifier à l'aide des touches DELETE, BACK SPACE, CURSOR et KILL (voir p. 4).
Lorsque les modifications sont terminées, il suffit de taper RETURN et la nouvelle ligne est mémorisée à la place de l'ancienne.

2.5 HELP

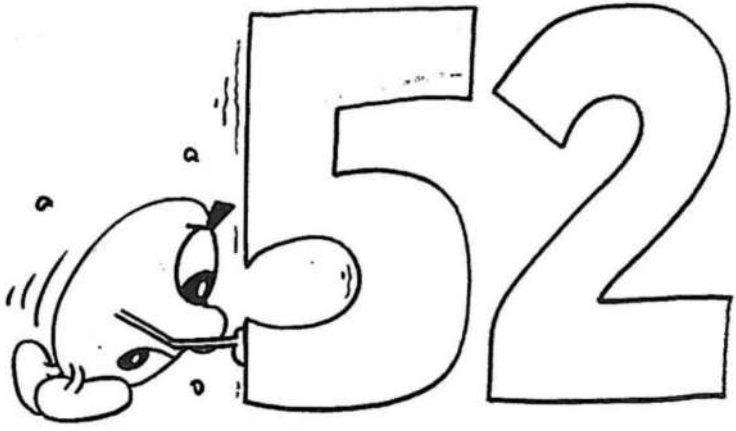
HELP♪ affiche sur l'écran un petit texte qui résume tous les ordres du BASIC.

Le texte est affiché par tranches de 20 lignes.

Pour afficher la suite, taper RETURN

Pour retourner au BASIC, taper ESC

2.6 REN, AUTO



`REN↵` Renumérote toutes les lignes à partir de 100 avec un incrément de 10. La première ligne aura le numéro 100, la deuxième le numéro 110, la troisième 120, etc.

`REN start↵` Renumérote toutes les lignes à partir de "start" avec un incrément de 10.

`REN start,step↵` Renumérote toutes les lignes à partir de "start" avec un incrément de "step".

Par exemple:

Soit le programme:

```
15 LET I=Ø
17 PRINT I
20 LET I=I+1
23 IF I < 10 THEN 17
34 END
```

Après la commande `REN 200↵`, il devient:

```
200 LET I=Ø
210 PRINT I
220 LET I=I+1
230 IF I < 10 THEN 210
240 END
```

AUTO ↵ Facilite l'édition d'un programme BASIC.
Il n'est plus nécessaire de taper les numéros des lignes.
Le SMAKY propose automatiquement le numéro 100 pour la première, 110 pour la deuxième, etc.

Lorsque l'édition est terminée, il faut appuyer sur [ESC] .

AUTO start ↵ Comme ci-dessus, à partir du numéro "start"

AUTO start, step ↵ Le SMAKY propose les numéros de ligne "start", puis "start"+"step", etc.

Par exemple:

L'utilisateur a tapé:

```
AUTO 50,20↵
```

L'écran montre:

```
>AUTO 50,20
READY
>50
```

La première ligne du programme peut être tapée. Après le [RETURN] on voit, dans cet exemple:

```
>AUTO 50,20
READY
>50 FOR I=1 TO 10
>70
```

3 NOMBRES

Voici quelques exemples de nombres, avec la façon de les écrire pour le BASIC.

Nombre	Façon d'écrire pour le BASIC
2	2
-15	-15
-1,003	-1.003
12.10^3	12E3
$105,03 \cdot 10^{-2}$	105.03E-2

Les nombres doivent être compris entre 5E-20 et 5E18.

3.1 OPERATIONS ARITHMETIQUES

Les opérations arithmétiques possibles sont:

+	addition	2+3 vaut 5
-	soustraction	5-3 vaut 2
*	multiplication	3*2 vaut 6
/	division	6/2 vaut 3
↑	élévation à une puissance	2↑3 vaut 8
SQR( )	racine carrée	SQR(9) vaut 3
SIN( )	sinus	SIN(0) vaut 0
COS( )	cosinus	COS(3.14159) vaut -1
TAN( )	tangente	TAN(3.14159/4) vaut 1
ATN( )	arc-tangente	ATN(1) vaut 3.14159/4
LN( )	logarithme népérien	LN(2.71827) vaut 1
LOG( )	logarithme décimal	LOG(100) vaut 2
EXP( )	anti-log népérien	EXP(1) vaut 2.71827
ABS( )	valeur absolue	ABS(-3) vaut 3 ABS(3) vaut 3
ARND( )	valeur arrondie	ARND(5.7) vaut 6 ARND(-5.7) vaut -6 ARND(5.4) vaut 5 ARND(-5.4) vaut -5
INT( )	valeur entière	INT(5.7) vaut 5 INT(-5.7) vaut -5 INT(5.4) vaut 5 INT(-5.4) vaut -5
SGN( )	signe	SGN(0) vaut 0 SGN(5.23) vaut 1 SGN(-3E5) vaut -1
RND( )	valeur aléatoire	RND(100) vaut par exemple 55.1352 (valeur comprise entre 0 et 100)
RANDOMIZE	"coup de sac" du générateur aléatoire	
FRE(Ø)	place restante en mémoire	
PI	constante π	PI=3.14159

3.2 EXPRESSIONS

Voici quelques exemples d'expressions:

```
2+5     T*(A-X)     P2/SIN(A3-B3)
```

T,A,X,P2,A3 et B3 sont des variables. Chaque variable correspond à une valeur. Nous verrons plus tard comment assigner une valeur à une variable. Une variable peut être une lettre ou une lettre suivie d'un chiffre.

```
A AØ A1 ... A9 B BØ B1... B9 .....    Z ZØ Z1... Z9
```

Il est possible d'utiliser la constante π (PI) à l'intérieur d'une expression.

Par exemple, si l'on tape

```
PRINT PI↵
```

on voit sur l'écran:

```
>PRINT PI
 3.14159
>
```

Exemples d'expressions:

```
SIN(PI/2)     A9+PI-1     1+(PI*2)
```

4. ORDRES EXECUTABLES

Les ordres qui vont maintenant être expliqués peuvent être utilisés soit en mode direct, soit à l'intérieur d'un programme.

4.1 PRINT

L'ordre PRINT permet d'afficher sur l'écran des expressions ou des textes. Par exemple, si vous faites:

```
PRINT "SALUT"↵
```

vous voyez sur l'écran:

```
>PRINT "SALUT"
 SALUT
>
```

Ou bien:

```
PRINT 12.3+1↵
```

et sur l'écran:

```
>PRINT 12.3+1
 13.3
>
```

EXERCICE:

Essayer d'afficher sur l'écran le résultat de quelques expressions arithmétiques:

```
PRINT 2+0.3↵
PRINT 2*(15+2E3)↵
PRINT SIN(3.14159)↵
PRINT RND(10)↵
...
```

REMARQUE:

L'ordre PRINT peut être abrégé par ? .

L'espace entre ? et l'expression est optionnel.

Par exemple, ? 1+1↵ est équivalent à PRINT 1+1↵ .

Pour afficher une chaîne de caractères, il faut la mettre entre guillemets. Ne pas confondre:

```
PRINT "S"          et   PRINT S
```

En effet, `PRINT "S"` affiche le caractère S
tandis que `PRINT S` affiche la valeur de la variable S

Il est possible d'imprimer plusieurs nombres ou des nombres et des textes sur une même ligne. Pour cela, il faut les séparer par des virgules ou des points-virgules.

```
PRINT 1+1, 15-3
PRINT "LA VARIABLE S VAUT:",S
```

La virgule affiche des nombres ou des textes avec une tabulation de 8, tandis que le point-virgule appond simplement les nombres ou les textes.

Essayer les exemples suivants:

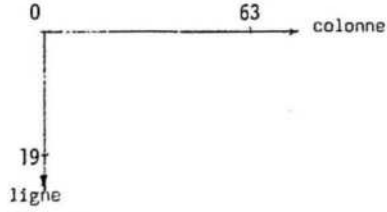
```
PRINT "SALUT","JE SUIS UN SMAKY"↵
PRINT "SALUT";"JE SUIS UN SMAKY"↵
PRINT 1+2, 3+4↵
PRINT 1+2;3+4↵
```

Un nombre commence et finit par un blanc.

La fonction TAB(n) avance le pointeur dans la ligne de n positions. n doit être entier et plus grand ou égal à 1. Essayer:

```
PRINT TAB(1);"*"↵
PRINT TAB(10);"*"↵
PRINT TAB(50);"*"↵
PRINT "*";TAB(5);"*";TAB(5); "*"↵
```

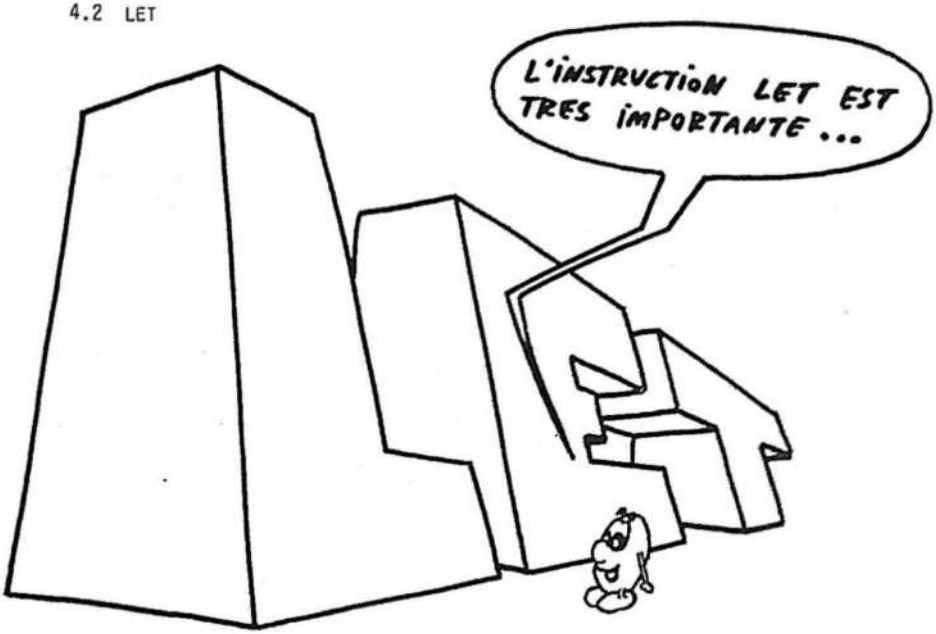
La fonction @(ligne,colonne) positionne le pointeur à la ligne "ligne" et à la colonne "colonne". "ligne" doit être compris entre 0 et 19 et "colonne" entre 0 et 63.



Essayer les exemples suivants:

```
PRINT @(0,0);"*"↵
PRINT @(10,32);"MILIEU DE L'ECRAN"↵
```

4.2 LET



L'ordre LET permet d'assigner une valeur à une variable. La syntaxe est la suivante:

```
LET variable=expression↵
```

Essayer de faire:

```
LET E=2+2↵
PRINT E↵
```

Par exemple, pour ajouter 1 à la variable I, il suffit de faire:

```
LET I=I+1↵
```

Lors de l'édition d'une ligne, le mot "LET" peut être omis.

`LET A=2` est équivalent à `A=2`.

Essayons maintenant de résoudre une équation du deuxième degré du type

$$ax^2 + bx + c = 0.$$

On sait que $X_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

donc le programme sera:

```
40 LET X1=(-B+(SQR(B↑2-4*A*C)))/(2*A)
50 LET X2=(-B-(SQR(B↑2-4*A*C)))/(2*A)
60 PRINT "X1="; X1, "X2="; X2
```

Essayons de résoudre $-3X^2+4X+5 = 0$
Il faut donc faire:

```
10 LET A=-3↵
20 LET B=4↵
30 LET C=5↵
```

Avant de démarrer le programme, on peut faire LIST↵ pour vérifier si c'est OK. Si c'est OK, il faut faire RUN↵ pour démarrer. Si tout a fonctionné correctement, l'écran doit montrer:

```
X1=-.786291    X2=2.11961
```

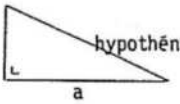
EXERCICE:

Essayer de résoudre d'autres équations en changeant les valeurs de A, B et C. Pour ce faire, il faut simplement retaper les lignes 10, 20 et 30.

Si le nombre dont il faut extraire la racine carrée est négatif ou si A vaut zéro (donc, division pas zéro), le message

```
ARITHMETIC ERROR IN LINE        apparaît.
```

Nous allons maintenant faire un programme qui calcule l'hypothénuse d'un triangle rectangle.



$\text{hypothénuse}^2 = a^2 + b^2$

$\text{hypothénuse} = \sqrt{a^2 + b^2}$

```
30 LET H=SQR(A↑2+B↑2)↵
40 PRINT "L'HYPOTHENUSE VAUT";H↵
```

Si, par exemple, a=3 et b=4

```
10 LET A=3↵
20 LET B=4↵

RUN↵
```

Et sur l'écran:

```
L'HYPOTHENUSE VAUT 5
READY
>
```

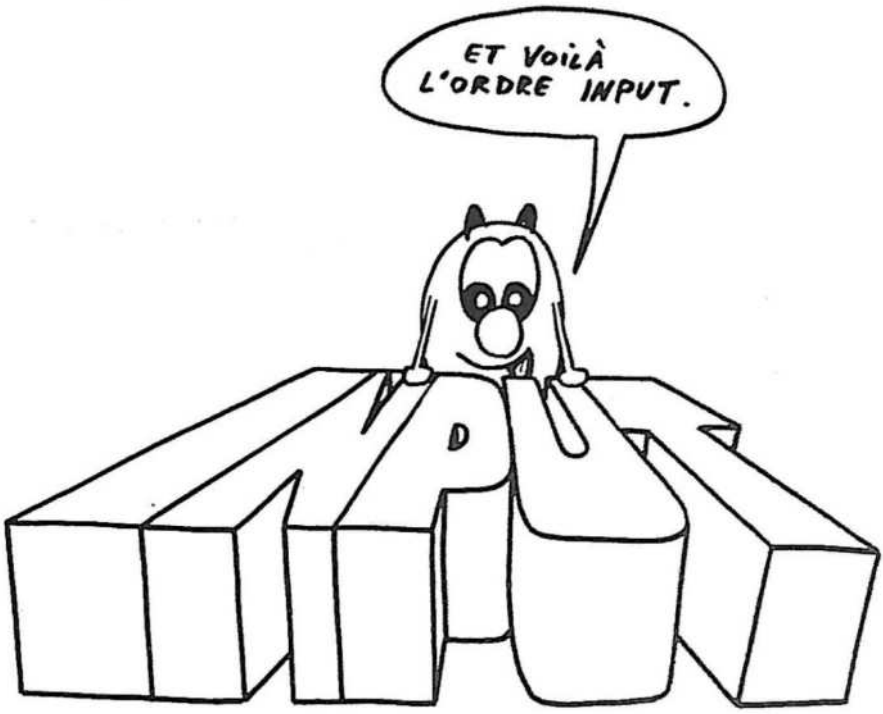
REMARQUE:

Il est possible de réunir les lignes 30 et 40 en une seule:

```
30 PRINT "L'HYPOTHENUSE VAUT"; SQR(A↑2+B↑2)↵
```

et il n'est plus nécessaire d'utiliser la variable H.

4.3 · INPUT



L'ordre INPUT permet, pendant l'exécution d'un programme, d'assigner une valeur tapée sur le clavier à une variable. Le programme qui calcule l'hypothénuse d'un triangle deviendra:

```
10 INPUT A,B↵
20 PRINT "L'HYPOTHENUSE VAUT"; SQR(A↑2+B↑2)↵
RUN↵
```

Un point d'interrogation apparaît sur l'écran, montrant que le programme est arrêté (à la ligne 10) et attend que l'utilisateur tape un nombre suivi d'un

RETURN

Faites par exemple:

```
3↵
```

Un autre point d'interrogation apparaît. Le programme attend un nombre pour la variable B.

```
4↵
```

Et sur l'écran:

```
L'HYPOTHENUSE VAUT 5
READY
>
```

Lorsque l'ordre INPUT A,B,C,D est rencontré, il est possible de faire:

```
1↵
2↵
3↵
4↵
```

ou bien

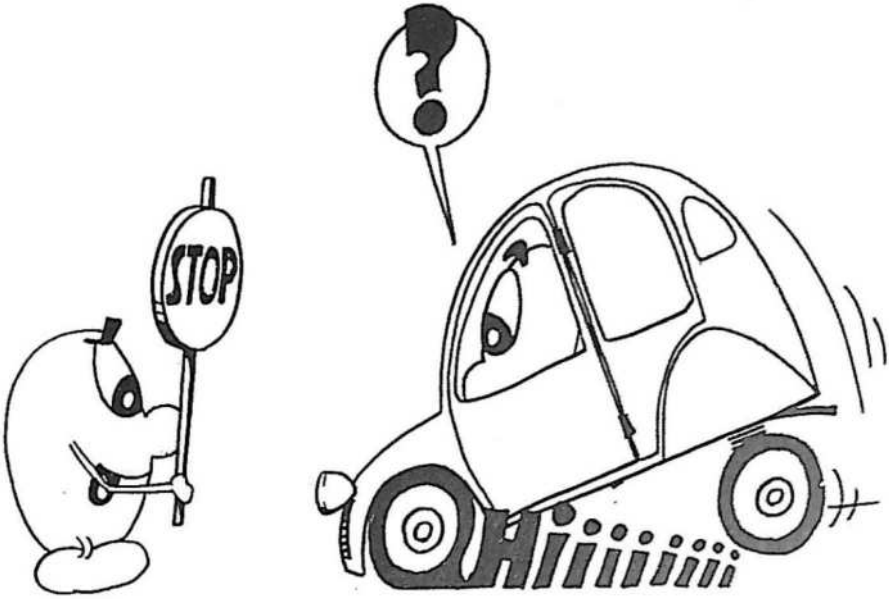
```
1,2,3,4↵
```

Il est souvent pratique d'afficher un texte avant l'entrée d'un nombre. L'ordre INPUT permet de le faire sans utiliser PRINT.

Notre programme devient alors:

```
10 INPUT "PREMIER COTE", A↵
15 INPUT "DEUXIEME COTE", B↵
20 PRINT "L'HYPOTHENUSE VAUT";SQR(A↑2+B↑2)↵
```

4.4 END, STOP, CONTINUE



Les ordres END et STOP ne sont utiles qu'à l'intérieur d'un programme. Ils permettent d'arrêter l'exécution d'un programme lorsqu'il arrive à la ligne correspondante.

Par contre, l'ordre CONTINUE ne peut être utilisé qu'en mode direct. Il permet de continuer l'exécution d'un programme après qu'il ait été arrêté par un STOP ou un [ESC] .

Exemple:

```
10 PRINT 10↵
20 STOP↵
30 PRINT 30↵
40 END↵
```

Si l'on donne l'ordre RUN, le programme démarre à la ligne 10, puis s'arrête à la ligne 20. On peut lire sur l'écran:

```
STOP AT LINE 20
```

On peut alors taper CONTINUE et le programme repart à la ligne 30 pour s'arrêter définitivement à la ligne 40.

Il faudra donc utiliser l'instruction STOP pour mettre au point un programme, tandis que END sera simplement mis à la fin normale d'un programme. Lorsque la ligne qui porte le plus grand numéro est la dernière à effectuer, END n'est pas nécessaire.

Par exemple:

```
10 INPUT "TAPER UN NOMBRE",N
20 PRINT "LE DOUBLE VAUT",N*2
30 END
```

pas nécessaire car la ligne 20 est la dernière du programme.

4.5 GOTO

Normalement un programme est exécuté séquentiellement, c'est-à-dire dans l'ordre croissant des numéros de lignes.

GOTO permet de modifier cet ordre.

Par exemple:

```
10 INPUT "PREMIER COTE",A♪
15 INPUT "DEUXIEME COTE",B♪
20 PRINT "L'HYPOTHENUSE VAUT":SQR(A*A+B*B)♪
30 GOTO 10♪
```

Lorsque le programme arrive à la ligne 30, il saute à la ligne 10, et ainsi tout recommence.

Notre programme comporte maintenant une boucle fermée dont il est impossible de sortir, sauf en pressant sur [ESC] qui est l'arrêt d'urgence.

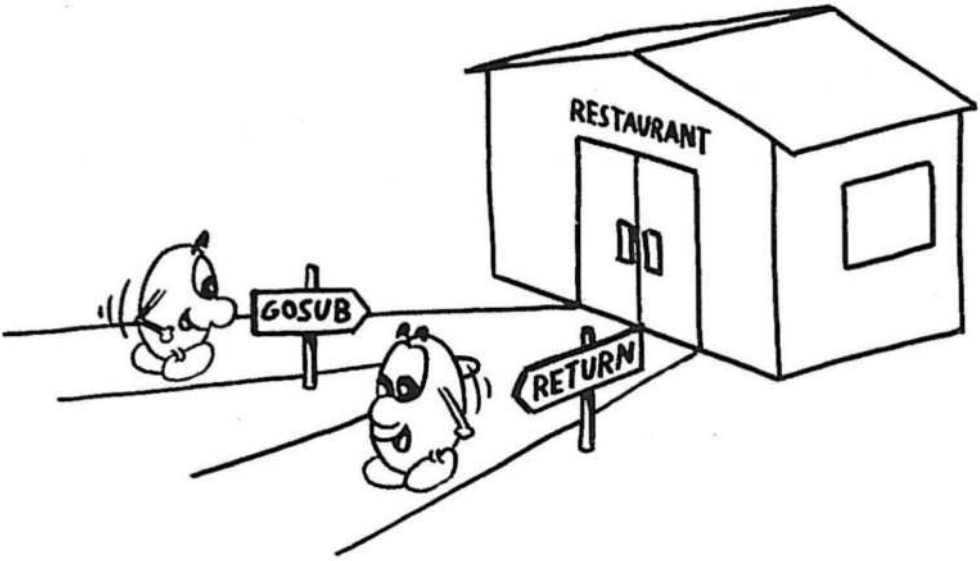
Si l'écran montre alors:

```
STOP
STOP AT LINE 10
READY
>
```

cela indique que le programme était alors à la ligne 10.

L'ordre GOTO peut aussi être utilisé en mode direct pour commencer un programme à n'importe quelle ligne. Les variables ne sont alors pas effacées.

4.6 GOSUB, RETURN



Il arrive souvent, dans les longs programmes qu'une séquence d'instructions soit répétée plusieurs fois.

Pour raccourcir le programme, il est possible de mettre cette séquence dans un sous-programme (qui est alors terminé par un RETURN) et d'appeler toute la séquence par un GOSUB.

Exemple d'une partie de programme sans GOSUB, ni RETURN.

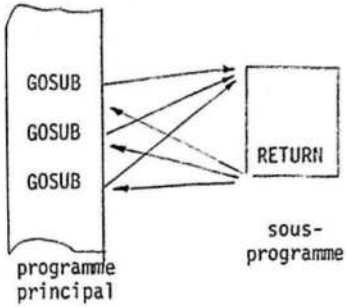
```
100 LET I=B1
110 PRINT "LE CARRE VAUT",I*I
120 PRINT "LE CUBE VAUT",I*I*I
130 PRINT "LE DOUBLE VAUT",I+I
140 LET I=B2
150 PRINT "LE CARRE VAUT",I.I
160 PRINT "LE CUBE VAUT", I*I*I
170 PRINT "LE DOUBLE VAUT",I+I
180 END
```

La même partie de programme, en mettant la séquence avec les trois PRINT dans un sous-programme:

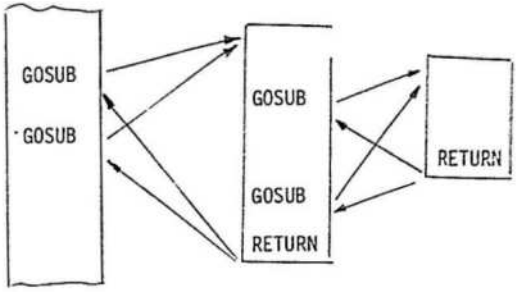
```
100 LET I=B1
110 GOSUB 1000
140 LET I=B2
150 GOSUB 1000
180 END
1000 PRINT "LE CARRE VAUT",I*I
1010 PRINT "LE CUBE VAUT",I*I*I
1020 PRINT "LE DOUBLE VAUT",I+I
1030 RETURN
```

} sous-programme

On peut schématiser cela de la façon suivante:



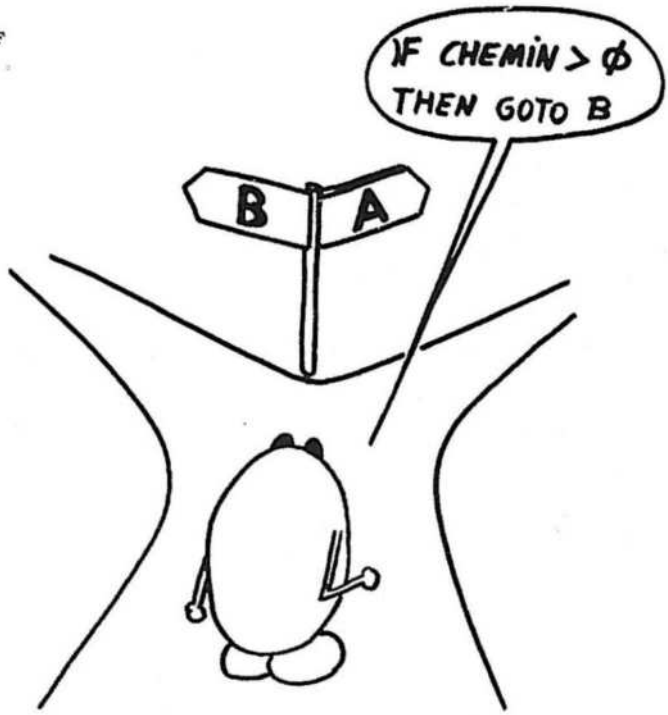
Un sous-programme peut lui-même appeler un autre sous-programme, et ainsi de suite.



On remarque tout de suite le gain de place dans le programme.

L'ordre GOSUB peut aussi être utilisé en mode direct pour exécuter une partie de programme terminée par un RETURN.

4.7 IF



IF permet de faire des débranchements conditionnels, c'est-à-dire d'effectuer un saut si une condition est remplie.

Par exemple:

```
 10 INPUT "TAPER UN NOMBRE COMPRIS ENTRE 0 ET 10",N
 20 IF N<0 THEN GOTO 100
 30 IF N>10 THEN GOTO 100
 40 PRINT "C'EST JUSTE!"
 50 END

100 PRINT "C'EST FAUX!"
110 GOTO 10
```

La syntaxe générale est la suivante:

IF *expression* TEST *expression* THEN *instruction*

Les tests possibles sont:

<	plus petit que
>	plus grand que
=	égal à
<=	inférieur ou égal à
>=	supérieur ou égal à
<>	différent de

Voici quelques exemples corrects:

```
IF C<=M+5 THEN PRINT "*"
IF 2*(M-5)<>0 THEN LET M=M+1
IF T/(5+C2)=2-(B3/11) THEN GOTO 11520.
```

EXERCICE:

Essayer le programme suivant qui calcule une division ou une racine carrée en interdisant les erreurs:

```
 10 PRINT "VOULEZ-VOUS FAIRE UNE DIVISION ?"
 20 GOSUB 500
 30 IF R=1 THEN GOTO 100
 40 PRINT "VOULEZ-VOUS FAIRE UNE RACINE CARREE ?"
 50 GOSUB 500
 60 IF R=1 THEN GOTO 200
 70 PRINT "DESOLE, JE NE SAIS RIEN FAIRE D'AUTRE"
 80 END

100 INPUT "QUEL EST LE DIVIDENDE", D1
110 INPUT "QUEL EST LE DIVISEUR", D2
120 IF D2=0 THEN GOTO 300
130 LET Q=D1/D2
140 PRINT "LE QUOTIENT VAUT", Q
150 END

200 INPUT "TAPEZ UN NOMBRE", N
210 IF N<0 THEN GOTO 300
220 PRINT "LA RACINE CARREE VAUT", SQR(N)
230 END

300 PRINT "DESOLE,MAIS C'EST IMPOSSIBLE :"
310 END

500 INPUT "OUI, TAPEZ 1  NON, TAPEZ 0", R
510 IF R>1 THEN GOTO 500
520 IF R<0 THEN GOTO 500
530 LET R=INT(R)
540 RETURN
```

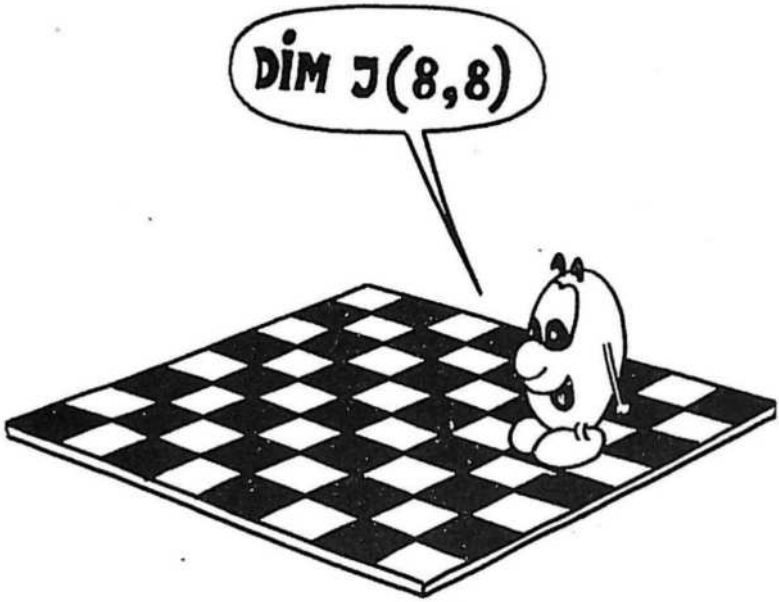
IF permet aussi de faire des débranchements
- si une condition ou une autre condition est remplie
- si une condition et une autre condition sont remplies
- si une condition n'est pas remplie

grâce aux opérateurs suivants:

```
OR   ou
AND  et
NOT  non
```

Exemples de conditions vraies:

```
IF 5>3 OR 6>=7
IF 2<=3 AND 5>4
IF NOT (2>5)
```



DIM permet de réserver de la place pour une variable indicée.

Par exemple, DIM T(5) réserve de la place pour la variable T qui aura le structure suivante:

0	1	2	3	4	5

T(4)

Il est donc possible de faire: LET T(3)=12

DIM A(5,3) réserve de la place pour la variable A qui aura la structure suivante:

0,0	1,0	2,0	3,0	4,0	5,0
0,1	1,1	2,1	3,1	4,1	5,1
0,2	1,2	2,2	3,2	4,2	5,2
0,3	1,3	2,3	3,3	4,3	5,3

A(4,2)

Attention! la place réservée en mémoire est très vite importante.
En cas de dépassement de capacité, le message

OVERFLOW ERROR IN LINE

Si après avoir fait DIM A(3) vous faites LET A(4)=0
le message ARRAY ERROR IN LINE apparaît.

Il est également possible de réserver de la place pour des strings avec l'ordre DIM. Nous verrons comment faire au chapitre 4.16.

4.9 FOR, NEXT, STEP

Les instructions FOR, NEXT et STEP permettent la répétition d'une partie de programme un certain nombre de fois.

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

est équivalent à

```
10 LET I=1
20 PRINT I
30 LET I=I+1
40 IF I<=10 THEN GOTO 20
```

et

```
10 FOR I=1 TO 100 STEP 10
20 PRINT I
30 NEXT I
```

est équivalent à

```
10 LET I=1
20 PRINT I
30 LET I=I+10
40 IF<=100 THEN GOTO 20
```

La syntaxe est la suivante:

FACULTATIF

```
FOR variable=expression TO expression STEP expression
Partie de programme
NEXT variable
```

Si vous faites:

```
10 FOR C=1 TO 10
20 PRINT C
30 NEXT D
```

vous verrez sur l'écran:

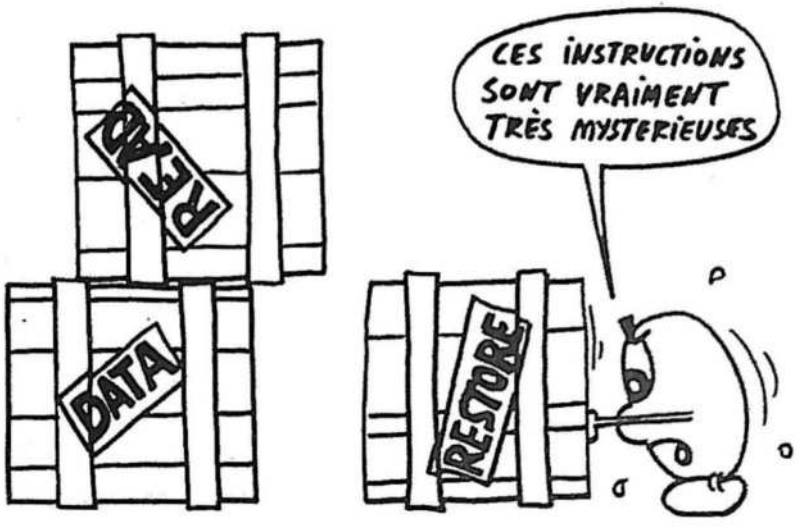
```
NEXT ERROR IN LINE 30
READY
>
```

Il est possible d'imbriquer plusieurs FOR-NEXT.

Par exemple:

```
10 FOR I=1 TO 10
20 FOR J=1 TO 10
30 PRINT I,J
40 NEXT J
50 NEXT I
```

4.10 DATA, READ et RESTORE



L'instruction DATA permet de déclarer une série de valeurs.

L'instruction READ permet de les assigner à des variables.

Par exemple:

```
10 DATA 12,50,100,1,2.3↵
20 READ V1,V2,V3,V4,V5↵
30 PRINT V1;V2;V3;V4;V5↵
RUN↵
```

Vous verrez sur l'écran:

```
 12  50  100  1  2.3
```

RESTORE remet le pointeur de lecture sur la première valeur.

```
10 DATA 1,2,3↵
20 READ A,B↵
30 PRINT A;B↵
40 RESTORE↵
50 READ A,B,C↵
60 PRINT A;B;C↵
RUN↵
```

Sur l'écran:

```
 1  2
 1  2  3
```

Si vous faites

```
10 DATA 45,46↵
20 READ A1,A2,A3↵
RUN↵
```

Vous voyez sur l'écran

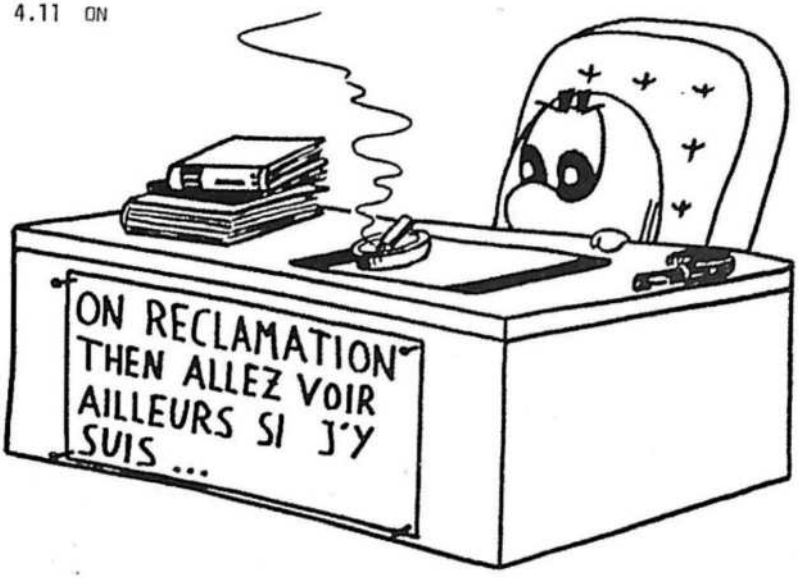
```
DATA ERROR IN LINE 20
READY
>
```

Ces trois ordres sont très pratiques pour pouvoir assigner des valeurs de départ au début d'un programme.

Par exemple:

```
10 DATA 2.71828, 2.71828*2↵
20 READ E,E2↵
```

4.11 ON



`ON A1+B GOTO 105,200,60` permet de sauter à la ligne 105 si A1+B vaut 1, à la ligne 200 si A1+B vaut 2, à la ligne 60 si A1+B vaut 3 et à la ligne suivante si A1+B est différent de 1, 2 ou 3.

Il est possible de mettre autant de numéros que l'on veut après le GOTO.

L'exemple donné ci-dessus est donc équivalent à:

```
IF  A1+B=1  THEN 105
IF  A1+B=2  THEN 200
IF  A1+B=3  THEN  60
```

`ON ESC THEN 1305` permet de sauter à la ligne 1305 lorsque l'on p... sur la touche [ESC].

[CTRL] [C] garde la fonction d'arrêt du ... et n'est donc pas dévié en 1305.

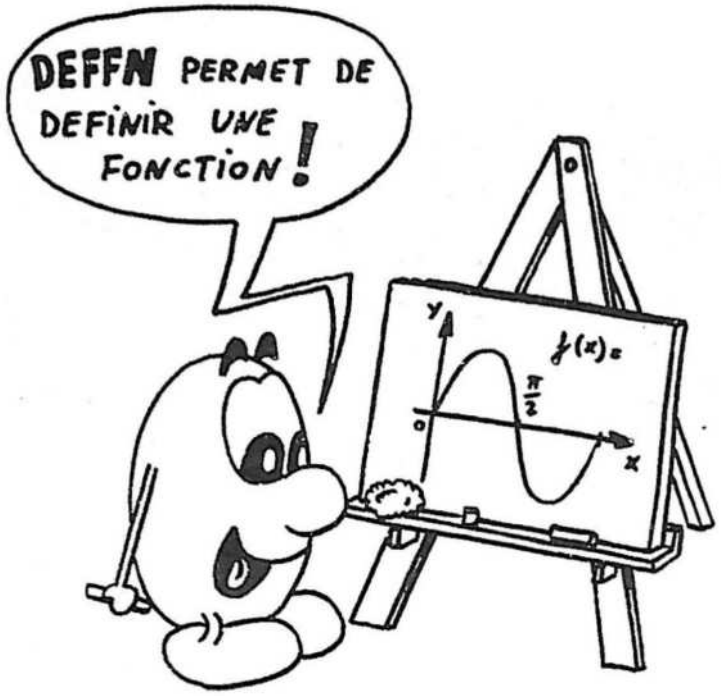
Pour pouvoir utiliser à nouveau la touche [ESC] normalement, il faut utiliser l'instruction

`ON ESC THEN STOP`

`ON ERR THEN 30` permet de sauter à la ligne 30 lorsqu'une erreur quelco... se produit.
Pour qu'une erreur soit à nouveau affichée sur l'écran, il faut utiliser l'instruction

`ON ERR THEN STOP`

4.12 DEFFN, FN



DEFFN permet de définir une fonction à une variable.

Par exemple, pour définir la fonction $f(x)=2x+5$, il faut faire `DEFFN F(X)=2*X+5`.

Le nom de la fonction est "F", mais il est possible de lui donner n'importe quel nom respectant les règles de syntaxe des variables (A,B3,F5,...).

Y=FNF(3) assigne à la variable Y la valeur de la fonction pour X=3.

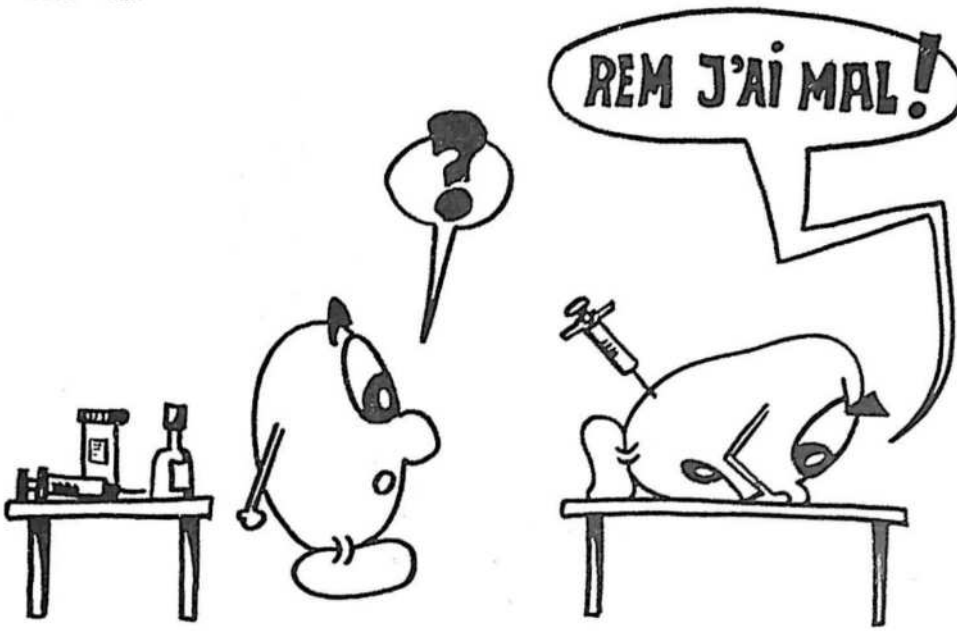
Exemple:

Calculons quelques points du graphe de la fonction $x \mapsto f(x)=x^2+5x-3$

Pour trouver quelques points compris entre 0 et 3, il suffit de faire:

```
10 DEFFN A(X)=X↑2+5*X-3↵
20 FOR I=1 TO 3 STEP 0.2↵
30 PRINT "X=";I,"Y=";FNA(I)↵
40 NEXT I↵
```

4.13 REM



L'instruction REM permet de mettre des commentaires dans un programme. Lors de l'exécution, le texte qui suit l'instruction REM est simplement ignoré.

Par exemple:

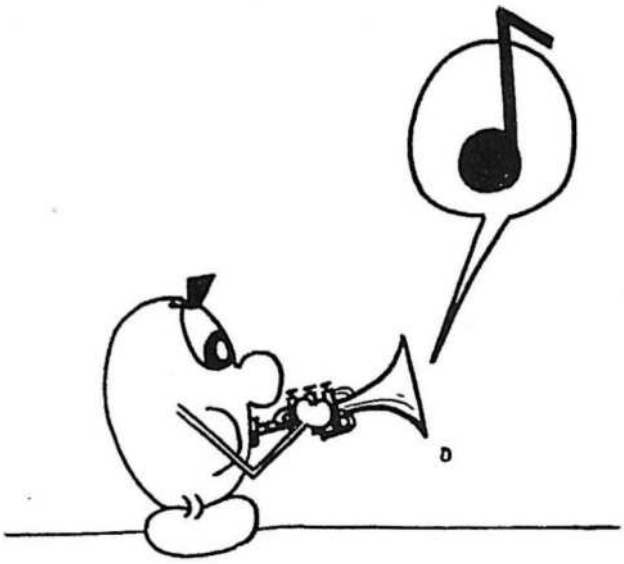
```
 10 REM  **********************
 20 REM  *                    *
 30 REM  * VOICI UN PROGRAMME *
 40 REM  *       FACILE       *
 50 REM  *                    *
 60 REM  **********************
 70 REM
 80 INPUT "TAPER UN NOMBRE",N
 90 PRINT "LE DOUBLE VAUT",N*2
100 END
```

Il est recommandé de mettre des commentaires avant chaque module de programme et avant chaque sous-programme. On précisera la fonction du module, les variables d'entrée et les variables de sortie.

```
100 REM CALCUL DE LA RACINE CARREE D'UN
110 REM NOMBRE AVEC TEST POUR LES
120 REM RACINES NEGATIVES
130 REM
140 REM ENTREE-  A VALEUR DE LA RACINE A CALCULER
150 REM SORTIE-  A VALEUR DE LA RACINE CALCULEE
160 REM
170 IF A<0 THEN GOTO 190
180 A=SQR(A)
190 RETURN
```

! est équivalent à REM.

4.14 BEEP

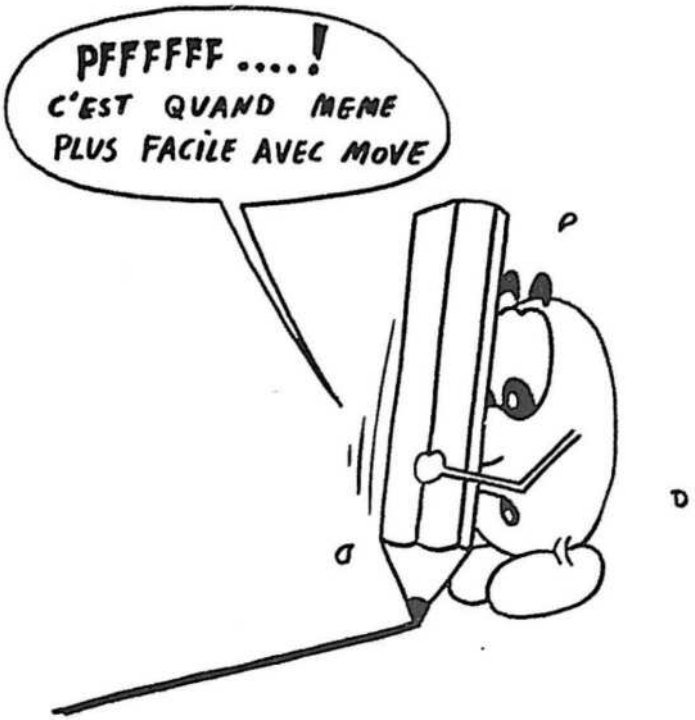


Fait résonner le haut-parleur avec une délicieuse mélodie servant à attirer l'attention de l'opérateur distrait.

Voici un programme qui fait 10 fois "bip", avec un silence après chaque "bip":

```
10 FOR I=1 TO 10
20 BEEP
30 FOR J=1 TO 60
40 REM JE REPRENDS MON SOUFFLE
50 NEXT J
60 NEXT I
```

4.15 MOVE, SCREEN, PAGE

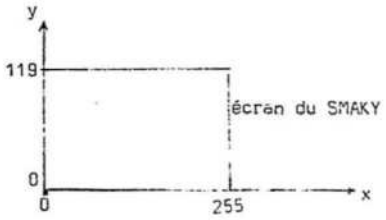


L'ordre MOVE permet de dessiner des traits sur l'écran du SMAKY.
La syntaxe est la suivante:

MOVE (X,Y,m)

X et Y déterminent les coordonnées du point d'arrivée de la droite, les coordonnées du point de départ étant celles du point d'arrivée du précédent trait dessiné.

Les coordonnées possibles sont:



m détermine le mode du trait. Les modes possibles sont:

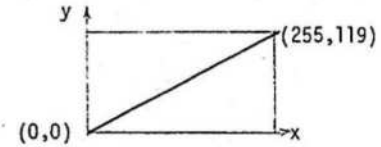
0	efface	4	- - - - -
1	-————	5	-.-.-.-.-
2	————	6	++++++++++
3	.....	7	saut

Pour tracer une diagonale à travers l'écran, il faut faire:

```
10 MOVE (0,0,7)↵
20 MOVE (255,119,1)↵
```

La première instruction trace un trait invisible (mode 7) depuis un point inconnu jusqu'en (0,0).

La deuxième instruction trace un trait mince continu (mode 1) depuis la coordonnée précédente (0,0) jusqu'en (255,119).



Le trait que l'on vient de tracer est superposé avec le texte situé sur l'écran. Pour effacer le texte sur l'écran, il faut faire:

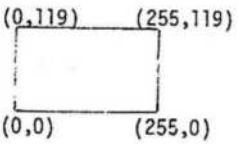
```
PAGE
```

Pour effacer tous les traits dessinés avec MOVE, il faut faire

```
SCREEN.
```

Voici un programme qui trace un cadre autour de l'écran:

```
10 SCREEN↵
20 MOVE (0,0,7)↵
30 MOVE (255,0,1)↵
40 MOVE (255,119,1)↵
50 MOVE (0,119,1)↵
60 MOVE (0,0,1)↵
```



Nous allons maintenant essayer un programme qui fait des traits aléatoirement:

```
10 PAGE
20 SCREEN
30 MOVE (RND(255),RND(119),RND(7))
40 GOTO 30
```

Il faut presser sur [ESC] pour arrêter.

Exercice:

Faire un programme qui trace 6 droites parallèles, une dans chaque mode.

4.16 STRINGS

Un string est une chaîne de caractères.
Nous avons déjà vu comment imprimer des chaînes de caractères avec l'ordre

Par exemple, pour imprimer le string ABC, il faut faire:

```
PRINT "ABC"
```

Il est possible d'avoir des variables qui ne correspondent pas à des nombres, mais à des strings. Pour cela, le nom de la variable doit être suivi d'un "$"
(SHIFT $ 4).

A titre d'exemple, pour assigner la chaîne de caractères TOTO à la variable B1$, il faut faire:

```
LET B1$="TOTO".
```

Toutes les opérations suivantes peuvent être effectuées indifféremment sur des variables nombres ou strings:

```
+   <=   >=   <>   >   <   =
```

Exemples:

```
A$="BON"↵
B$="JOUR"↵
C$=A$+B$↵
PRINT C$↵
```

Et sur l'écran:

```
BONJOUR
>
```

Si, à la fin d'un jeu, on désire demander à l'utilisateur de taper OUI ou NON pour savoir s'il désire recommencer une partie, on peut faire:

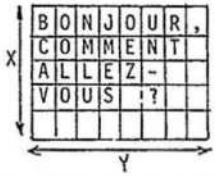
```
100 REM PROGRAMME QUELCONQUE DE JEU
    ...

500 INPUT "VOULEZ-VOUS RECOMMENCER (OUI OU NON)",R$
510 IF R$="NON" THEN END
520 IF R$="OUI" THEN 100
530 PRINT "REPONDEZ CORRECTEMENT"
540 GOTO 500
```

Exemples de conditions vraies:

"ABC"="ABC"
"ABC"<"XYZ"
"ABC">"ABA"

Il est possible de faire des tableaux de strings.

DIM A$(X,Y) réserve de la place pour X strings ayant la longueur Y.



L'ordre INPUT A$ met dans la variable string A$ ce que l'utilisateur frappe sur le clavier en supprimant les espaces.

Pour conserver les espaces, il faut faire: INPUT LINE A$

A$=CHR$(X) crée un string d'un caractère égal au code ASCII décimal de X.
Par exemple, PRINT CHR$(65) donne A.

Voici un tableau donnant le caractère vu sur l'écran en fonction de son code ASCII décimal.

0		20		40	(	60		80	P	100	d	120	x		
1	┌	21	√	41	)	61	=	81	Q	101	e	121	y		
2	efface l'écran	22	┘	42	*	62		82	R	102	f	122	z		
3	┘	23	┤	43	+	63	?	83	S	103	g	123	½		
4	mode normal	24	▒	44	,	64		84	T	104	h	124	¼		
5	mode inversé	25	┼	45	-	65	A	85	U	105	i	125	¾		
6	√	26	▒	46	.	66	B	86	V	106	j	126	÷		
7	coup de buzzer	27	▒	47	/	67	C	87	W	107	k	127	delete		
8	↑	28	▒	48	Ø	68	D	88	X	108	l				
9	tabulateur	29	▒	49	1	69	E	89	Y	109	m				
10		30	▒	50	2	70	F	90	Z	110	n				
11	▒	31	▒	51	3	71	G	91	←	111	o				
12	▒	32	espace	52	4	72	H	92	½	112	p				
13	retour de chariot	33	!	53	5	73	I	93	→	113	q				
14	▒	34	"	54	6	74	J	94	↑	114	r				
15	▒	35	#	55	7	75	K	95	↵	115	s				
16	▒	36	$	56	8	76	L	96	_	116	t				
17	▒	37	%	57	9	77	M	97	a	117	u				
18	▒	38	&	58	:	78	N	98	b	118	v				
19	▒	39	'	59	;	79	O	99	c	119	w				

EXEMPLES:

Affichons sur l'écran le mot "HELLO" en inverse vidéo.

```
PRINT CHR$(5);"HELLO";CHR$(4)
```

`X=ASCII(A$)` ↵ donne dans X la valeur décimale ASCII du premier caractère du string A$.

`A$=NUM$(X)` ↵ crée un string représentant le nombre X.

Par exemple `A$=NUM$(0.002)` puis `PRINT A$` donne `1.99999E-3`

`A=VAL(A$)` crée un nombre égal à celui contenu en ASCII dans A$.

Par exemple: `PRINT VAL("12")` donne 12

`A$=SPACE$(X)` ↵ crée un string contenant X espace (code ASCII décimal 32).

`A$=STRING$(X,Y)` ↵ crée un string de longueur X et contenant des caractères valant le code ASCII Y.

Par exemple, `PRINT STRING$(5,65)` donne AAAAA

`X=LEN(A$)` ↵ donne dans X la longueur d'un string.

Par exemple, `PRINT LEN("ABC")` donne 3.

EXTRACTION DE PARTIES DE STRINGS

`B$=LEFT$(A$,X)` ↵ donne dans B$ un string qui contient X caractères de A$ depuis la gauche.

Par exemple `PRINT LEFT$("ABCD",2)` donne AB.

`B$= RIGHT$(A$,X)` ↵ donne dans B$ un string qui contient X caractères de A$ depuis la droite.

Par exemple `PRINT RIGHT$("ABCD",2)` donne CD

`B$=MID$(A$,X,Y)` ↵ donne dans B$ un string qui contient Y caractères depuis X caractères à gauche de A$.

Par exemple `PRINT MID$("ABCDEF",2,3)` donne BCD.

`X=INSTR(Y,A$,B$)` ↵ compare B$ et A$ à partir du Yème caractère et donne dans X la position de B$ par rapport à A$. Si X vaut zéro, cela signifie que la comparaison était fausse.

Exemples:

```
INSTR(1,"ABCD","CD")  donne 3
INSTR(1,"ABCD","BCX") donne 0
INSTR(3,"ABCD","BC")  donne 0.
```

CHANGE A TO B$ ↵ convertit le vecteur A en un string B$. Le premier élément de A (A(Ø)) détermine la longueur du string.

Exemple:
```
DIM A(2)
A(Ø)=2
A(1)=71
A(2)=72
CHANGE A TO B$
PRINT B$          donne    GH
```

CHANGE B$ TO A ↵ convertit le string B$ en un vecteur A.

Exemple:
```
S$="GH"
CHANGE S$ TO V
FOR I=Ø TO 2
PRINT V(I);
NEXT I            donne  2 71 72
```

Exemple: un programme "renversant"

```
100 PRINT "DONNER UN TEXTE"
110 INPUT T$
120 LET L=LEN(T$)
130 FOR I=L TO 1 STEP -1
140 PRINT MID$(T$,I,1);
150 NEXT I
160 PRINT
170 END
```

```
RUN ↵
DONNER UN TEXTE
?BONJOUR, COMMENT ALLEZ-VOUS ?↵
RUOJNOB
READY
>
```

REMARQUES:

. la virgule étant considérée dans l'ordre INPUT comme séparateur de variable, la suite du texte est ignorée.

Pour remédier à cet inconvénient, on donnera l'ordre

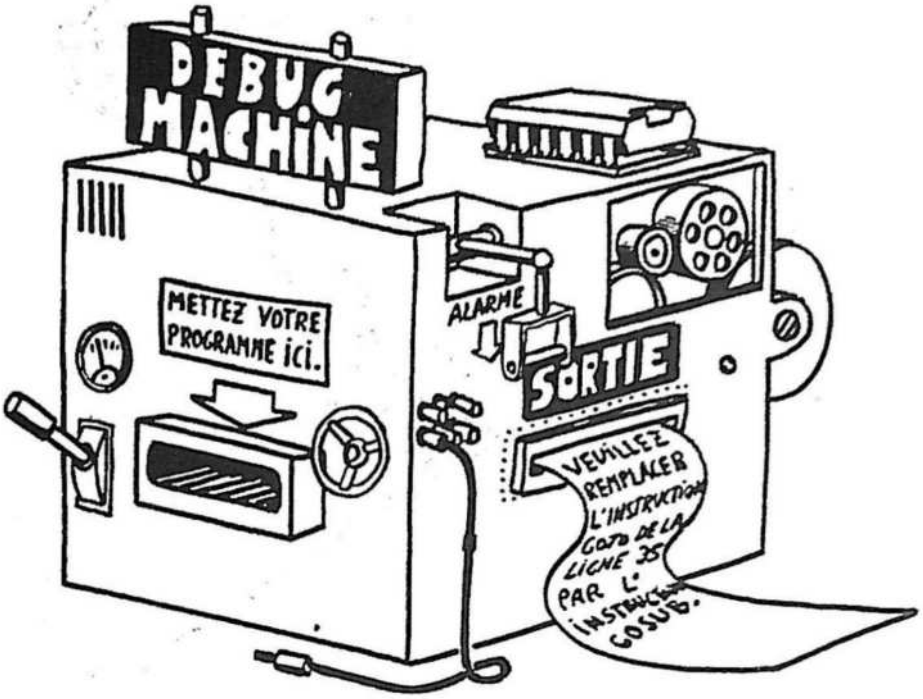
```
110 INPUT LINE T$
```

Après modification, on exécute de nouveau

```
RUN ↵
DONNER UN TEXTE
?BONJOUR, COMMENT ALLEZ-VOUS ?↵
?SUOV-ZELLA TNEMMOC ,RUOJONB
```

. avec l'ordre INPUT LINE les blancs sont significatifs.

4.17 DEBUG



Le debug permet une recherche rapide et facile des fautes à l'intérieur d'un programme. Nous avons déjà vu les ordres GOTO et GOSUB en mode direct, STOP et CONT qui permettent un test d'une partie de programme. Il peut également être utile d'exécuter un programme BASIC ligne à ligne (pas-à-pas).

STEP ON ↵ En mode direct ou à l'intérieur d'un programme, enclenche le mode pas à pas.

Chaque ligne est affichée en inverse vidéo sur l'écran. Pour passer à la ligne suivante, il faut appuyer sur une touche quelconque du clavier.

Pour arrêter, il faut faire [ESC].

STEP ↵ Identique à STEP ON

STEP OFF ↵ Supprime le mode pas-à-pas.

Il existe aussi un autre mode, qui affiche en inverse vidéo sur l'écran les lignes du programme BASIC exécuté, mais qui n'attend pas une touche du clavier pour passer à la ligne suivante.

TRACE ON ↵ En mode direct ou à l'intérieur d'un programme, enclenche le mode de visualisation continue

TRACE ↵ Identique à TRACE ON

TRACE OFF ↵ Déclenche le mode de visualisation

4.18 ORDRES DIVERS

POKE m,expression	Stocke à l'adresse m (décimale) les 8 bits les plus significatifs de l'expression
variable=PEEK(m)	Attribue à la variable les 8 bits trouvés à l'adresse m décimale.
OUT $,expression	Ecrit la valeur "expression" sur le périphérique $
variable=INP($)	Lit le périphérique $ dans la variable.

ATTENTION! ces ordres sont très dangereux car ils permettent de détruire le programme BASIC.

CALL m — appelle une sous-routine en langage assembleur à l'adresse m décimale.

BYE ↵ — retour au système du SMAKY.

Pour redémarrer le BASIC depuis le système (après un BYE ou un 

par exemple), il faut taper:

BASIC ↵

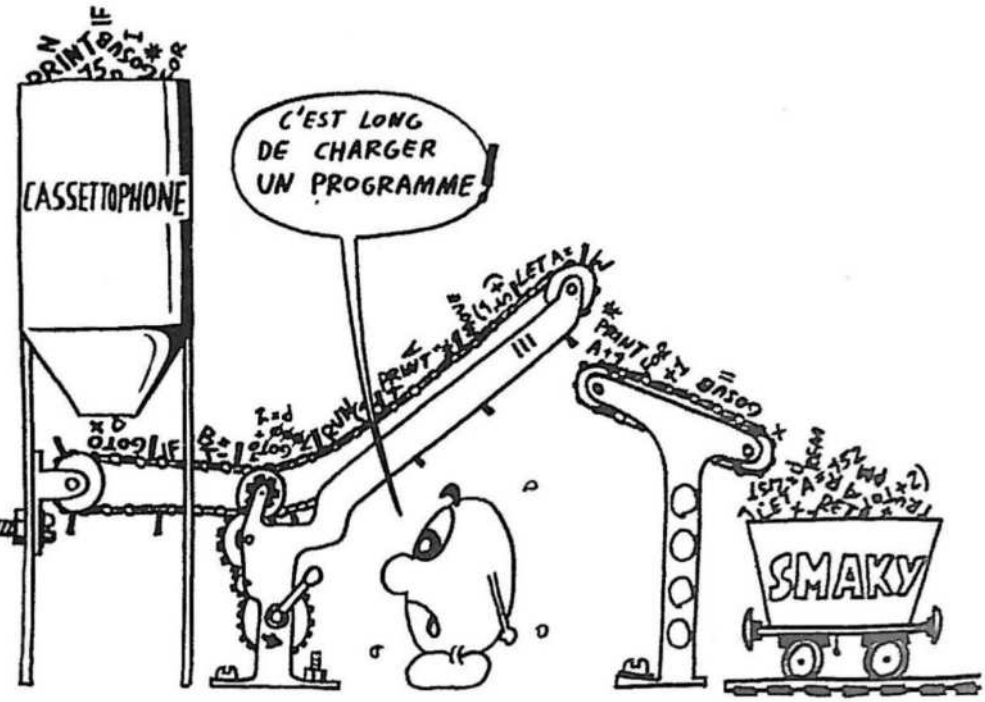
LIBRARY ↵ — Liste sur l'écran tous les fichiers ayant une extension .BS

DIRECTORY ↵ — Liste sur l'écran tous les fichiers.

Ces deux ordres ne peuvent évidemment être exécutés que sur un SMAKY ayant une mémoire de masse.

4.19 ORDRES D'ENTREE/SORTIE

Ces ordres permettent de sauver des programmes BASIC sur cassette ou sur une mémoire de masse.



Les programmes sont sauvés en format ASCII, c'est-à-dire qu'ils peuvent être réutilisés pour d'autres BASICs ou avec des éditeurs (SMILE, par exemple).

COPY $K7	sauve le programme sur un cassettophone
COPY TOTO	sauve le programme sur la mémoire de masse avec le nom TOTO.BS
ENTER $K7	charge le programme depuis un cassettophone
ENTER TOTO	charge le programme TOTO.BS depuis la mémoire de masse.

$K7 ne peut être utilisé que si le SMAKY est équipé d'un magnétophone, tandis que "nom_de_fichier" (par exemple TOTO) ne peut être utilisé que si le SMAKY a une mémoire de masse (floppy).

4.20 ORDRES DE GESTION DE FICHIER

Les ordres de gestion de fichiers n'existent que sur les SMAKY disposant d'une mémoire de masse (floppy disque). Ces ordres permettent de lire ou d'écrire des variables ou des strings à l'intérieur de fichiers sur la mémoire de masse.

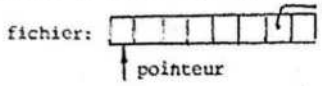
On distingue deux types d'accès aux fichiers possibles:

1) L'accès séquentiel

Pour pouvoir lire un fichier de ce type, il faut l'ouvrir en mode lecture séquentielle avec l'ordre OPEN FILE(canal,3),fichier↵

Le numéro de canal est un chiffre compris entre Ø et 7.
Cela permet d'ouvrir plusieurs fichiers simultanément en assignant des numéros de canaux différents.
"fichier" doit correspondre aux règles de syntaxe des strings.
Par exemple: "TOTO.BS" ou "RESULTAT" ou F$.

L'ordre INPUT FILE(canal),variable permet de lire dans "variable" l'élément du fichier qui est sous le pointeur.



Lorsque l'ordre est terminé, le pointeur est sous l'élément suivant. Donc, lors de chaque INPUT FILE, on lit l'élément suivant dans le fichier.

Lorsque le travail sur un fichier est terminé, il faut fermer le fichier (ce qui libère le numéro de canal correspondant) avec l'ordre:
CLOSE FILE(canal)↵

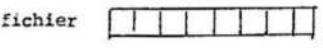
Exemple: essayons de lire, puis d'afficher sur l'écran la première ligne du fichier BASIC.HP.

```
10 OPEN FILE(1,3),"BASIC.HP"↵
20 INPUT FILE(1),LINE A$↵
30 PRINT A$ ↵
40 CLOSE FILE(1)↵
50 END↵
```

Le "LINE" de la ligne 20 permet de lire aussi les espaces (voir page 43).

Après chaque INPUT FILE, on peut tester si cet INPUT est valable ou s'il a été fait en dehors du fichier à l'aide de l'ordre EOF(canal).

EOF signifie "end of file", en français: fin du fichier.



Si vous êtes dans ce cas-là et que vous faites un INPUT FILE, l'élément lu n'est pas valide.

Exemple: lecture d'un fichier BASIC quelconque et affichage sur l'écran:

```
10 INPUT "QUEL FICHIER BASIC VOULEZ-VOUS VOIR ",P$
20 F$=F$+".BS"
30 OPEN FILE(1,3),F$
40 INPUT FILE(1),LINE A$
50 IF EOF(1) THEN 80
60 PRINT A$
70 GOTO 40
80 CLOSE FILE(1)
90 END
```

A la ligne 30, on choisit arbitrairement le canal 1.
Après l'ordre INPUT FILE (ligne 40), on teste la fin du fichier (EOF).

Pour pouvoir écrire un fichier séquentiel, il faut l'ouvrir en mode écriture séquentielle avec l'ordre OPEN FILE(canal,1),fichier

Pour cela, il faut que "fichier" n'existe pas (puisqu'on va le créer!)

S'il existe déjà, on peut le détruire avec l'ordre: DELETE fichier

Par exemple: DELETE "TOTO.BS" ou bien DELETE BS

Vous pouvez détruire plusieurs fichiers à la fois en faisant:

DELETE fichier,fichier...

Exemple: pour détruire n'importe quel fichier, il faut faire:

```
10 INPUT "QUEL FICHIER VOULEZ-VOUS DETRUIRE",F$
20 DELETE F$
30 END
```

Exemple:

Supposons que nous disposons d'une imprimante connectée sur SPP.
Cette imprimante réagit de la façon suivante:

- carriage return (CR): revient au début de la ligne
- line feed (LF): avance le papier jusqu'à la ligne suivante.

Nous allons écrire un petit programme permettant d'imprimer n'importe quel fichier.

```
100 OPEN  FILE (0,1),"$PP"
110 INPUT 'Quel fichier voulez-vous imprimer ',F$
120 OPEN  FILE (1,3),F$
130 INPUT  FILE (1), LINE L$
140 IF EOF(1) THEN  GOTO 180
150 PRINT  FILE (0),L$
160 PRINT  FILE (0),CHR$(10)
170 GOTO 130
180 CLOSE  FILE (0)
190 CLOSE  FILE (1)
200 END
```

En 130, nous lisons une ligne du fichier sur le disque dans la variable string L$.

En 140 nous testons si la ligne qui vient d'être lue est valable ou si elle est en dehors du fichier.

En 150 nous écrivons la ligne sur $PP.

En 160 nous rajoutons un LF pour faire avancer le papier. Le code LF vaut 10 décimal.

Et nous recommençons jusqu'à ce que le test de la ligne 140 soit vrai!

Exercice:

Modifier ce programme pour qu'il transforme les minuscules en majuscules.

Exemple:

Nous allons maintenant écrire un programme qui crée un fichier contenant les 10 premiers nombres entiers, suivis de leurs carrés. Le fichier s'appellera "SUITE.DA". Nous allons ensuite relire ce fichier.

```
100 REM Petit programme utile pour la comprehension des
110 REM ordres de manipulation de fichiers.
120 REM
130 REM *** Destruction du fichier
140 REM
150 ON  ERROR  THEN 180
160 DELETE "SUITE.DA"
170 PRINT "Le fichier SUITE.DA a ete detruit"
180 ON  ERROR  THEN  STOP
190 REM
200 REM *** Ecriture du fichier
210 REM
220 OPEN  FILE (1,1),"SUITE.DA"
230 PRINT "Le fichier SUITE.DA est ouvert en ecriture"
240 FOR I=1 TO 10
250 PRINT I,I^2
260 PRINT  FILE (1),I
270 PRINT  FILE (1),I^2
280 NEXT I
290 CLOSE  FILE (1)
300 PRINT "Le fichier SUITE.DA est ferme"
310 REM
320 REM *** Lecture du fichier
330 REM
340 OPEN  FILE (1,3),"SUITE.DA"
350 PRINT "Le fichier SUITE.DA est ouvert en lecture"
360 INPUT  FILE (1),A
370 IF EOF(1) THEN  GOTO 410
380 INPUT  FILE (1),B
390 PRINT A,B
400 GOTO 360
410 CLOSE  FILE (1)
420 PRINT "Le fichier SUITE.DA est ferme"
430 REM
440 REM *** Destruction du fichier
450 REM
460 DELETE "SUITE.DA"
470 PRINT "Le fichier SUITE.DA est detruit"
480 END
```

Les lignes 120 à 180 servent à détruire "SUITE.DA" seulement s'il existe.

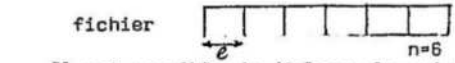
S'il n'existe pas, au lieu de s'arrêter sur la ligne 160, on va directement à la ligne 180 sans afficher le message de la ligne 170.
(S'il n'y avait pas le ligne 150, l'exécution du programme serait stoppée dans le cas où le fichier "SUITE.DA" n'existe pas),

2) L'accès aléatoire

L'accès aléatoire permet de lire ou d'écrire n'importe quel élément du fichier, l'élément étant caractérisé par sa position.

Pour créer un fichier à accès aléatoire, il faut faire:
OPEN FILE(canal,Ø),fichier,ℓ,n↵

"ℓ" est la longueur d'un élément du fichier.
"n" est le nombre d'éléments dans le fichier



Il est possible de déplacer le pointeur n'importe où grâce à l'ordre:
SPOS(canal,rang)↵

Une fois le SPOS fait, on peut écrire un élément comme en mode séquentiel.

Pour modifier un fichier à accès aléatoire, il faut utiliser l'ordre:
OPEN FILE(canal,2),fichier↵

Il est dès lors possible de faire des SPOS puis des INPUT FILE ou PRINT FILE.

Exemple: nous allons créer un fichier à accès aléatoire s'appelant "TOTO" contenant 8 éléments valant:

TOTO:

3	6	9	12	15	18	21	24

Ensuite, nous allons relire ce fichier depuis la fin.

```
100 PRINT 'ECRITURE.....'
110 OPEN  FILE (1,0),"TOTO",5,8
120 FOR I=1 TO 8
130 PRINT I*3,
140 SPOS (1,I)
150 PRINT  FILE (1),I*3
160 NEXT I
170 CLOSE  FILE (1)
180 PRINT
190 PRINT 'LECTURE......'
200 OPEN  FILE (1,2),"TOTO"
210 FOR I=1 TO 8
220 SPOS (1,9-I)
230 INPUT  FILE (1),A
240 PRINT A,
250 NEXT I
260 CLOSE  FILE (1)
270 DELETE "TOTO"
280 END
```

Les lignes 100 à 170 servent à créer le fichier.
Les lignes 180 à 260 servent à lire le fichier.

REMARQUE IMPORTANTE:

Les modes séquentiel et aléatoire ne peuvent pas être mélangés! Par exemple, il n'est pas possible de créer séquentiellement un fichier, puis de le lire aléatoirement.

Résumé:

OPEN FILE(canal,mode),fichier[,ℓ,n]↵

permet d'ouvrir un fichier.

Si mode = 1 fichier ouvert en écriture séquentielle
Si mode = 3 fichier ouvert en lecture séquentielle

Si mode = Ø création en mode aléatoire
Si mode = 2 modification en mode aléatoire

INPUT FILE(canal),variable,variable...↵

lit depuis le fichier correspondant à "canal" des variables ou des strings.

PRINT FILE(canal),variable,variable,...↵

écrit sur le fichier correspondant à "canal" des variables ou des strings

CLOSE FILE(canal)↵

ferme le fichier correspondant à "canal".

CLOSE↵ ferme tous les fichiers ouverts

EOF(canal)↵

permet de tester la fin de fichier sur un "canal"

DELETE fichier,fichier... ↵

permet de détruire un ou plusieurs fichiers

5. EXEMPLES DE PROGRAMMES

1) CALCUL DES DIVISEURS D'UN NOMBRE

Les diviseurs de 12 sont 12, 6, 4, 3, 2, 1.

```
100 PAGE
110 PRINT "CE PROGRAMME CALCULE LES DIVISEURS D'UN NOMBRE N"
120 PRINT
130 INPUT "DONNER LE NOMBRE ";N
140 PRINT
150 LET I=1
160 LET S=S+I
170 LET D=INT(N/S)
180 IF D*S=N THEN 210
190 IF S>N/2 THEN 240
200 GOTO 160
210 PRINT D;
220 IF D=1 THEN 250
230 GOTO 160
240 PRINT ;" 1"
250 END
```

Nous dirons qu'il existe un diviseur D à condition que
INT(N/S)*S = N avec D=INT(N/S).

Il suffit d'exécuter ce test en faisant varier S de 1 jusqu'à N/Z.

Dans notre exemple, nous avons:

N	S	D	INT(N/S)*S	Diviseurs
12	1	12	12	12
12	2	6	12	6
12	3	4	12	4
12	4	3	12	3
12	5	2	10	-
12	6	2	12	2

Le cas où D=1 étant valable pour tous les cas, il est affiché directement.

2) CALCUL DU PGCD

LE PGCD est le plus grand commun diviseur.
Par exemple, le PGCD de 21 et 18 est 3.

```
100 PAGE
110 PRINT "CE PROGRAMME CALCULE LE PGCD DE 2 NOMBRES A ET B (A>B)"
120 PRINT
130 INPUT "DONNER LE GRAND NOMBRE ";A
140 INPUT "DONNER LE PETIT NOMBRE ";B
150 LET E=A
160 LET F=B
170 IF A=B THEN 220
180 LET D=A-B
190 IF D<B THEN 240
200 LET A=D
210 GOTO 170
220 PRINT "PGCD DE";E;"ET";F;"=";A
230 GOTO 270
240 LET A=B
250 LET B=D
260 GOTO 170
270 END
```

Evolution du programme:

A	B	D	Lignes
0	0	0	
21	0	0	130
21	18	0	140
21	18	3	180
18	18	3	240
18	3	3	250
18	3	15	180
15	3	15	200
15	3	12	180
12	3	12	200
12	3	9	180
9	3	9	200
9	3	6	180
6	3	6	200
6	3	3	180
3	3	3	200
			220

impression de A qui vaut 3.

3) CALCUL DU PPCM

Le PPCM est le "plus petit commun multiple".
Par exemple, le PPCM de 18 et 12 est 36.

```
100 PAGE
110 PRINT "CE PROGRAMME CALCULE LE PPCM DE 2 NOMBRES A ET B (A>B)"
120 PRINT
130 INPUT "DONNER LE GRAND NOMBRE ";A
140 INPUT "DONNER LE PETIT NOMBRE ";B
150 LET S=0
160 LET S=S+1
170 LET A1=A*S
180 LET D=INT(A1/B)
190 IF A1=D*B THEN 210
200 GOTO 160
210 PRINT
220 PRINT "LE PPCM DE";A;"ET";B;"=";A1
230 END
```

Exercice:

Ecrire le tableau d'évolution du programme pour les variables S, A1 et D comme dans l'exemple précédent.

4) NOMBRES PREMIERS

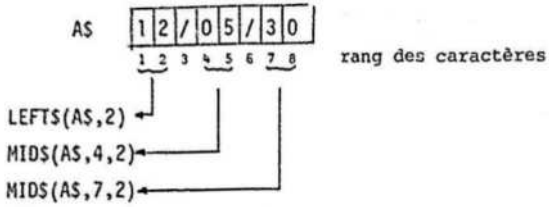
Les nombres premiers sont les nombres qui n'ont pas d'autre diviseur entier qu'eux-mêmes et l'unité.

Par exemple, 4 n'est pas premier, car il est divisible par 2.
7 est premier car il est divisible seulement par 1 et par 7.

```
100 PAGE
110 PRINT "CE PROGRAMME DETERMINE LES NOMBRES PREMIERS PLUS PETITS QUE P"
120 BEEP : PRINT
130 INPUT "DONNER LA LIMITE P= ";Q
140 PRINT
150 LET N=0
160 LET I=1
170 LET N=N+I
180 IF N>Q THEN 280
190 LET S=1
200 LET I=1
210 LET S=S+I
220 IF S*S>N THEN 250
230 IF N=(INT(N/S))*S THEN 170
240 GOTO 210
250 IF N=1 THEN 170
260 PRINT N;
270 GOTO 170
280 PRINT : PRINT : BEEP : END
```

5) HORLOGE

La mise à l'heure (lignes 110 à 140) montre un exemple de manipulation de strings. On extrait d'un string contenant, par exemple, la chaîne de caractères 12/05/30 les nombres 12, 5 et 30.



Avec cette méthode, il est possible de taper 12/05 . Les secondes vaudront 0 par défaut.

Par contre, il n'est pas possible de taper 12/5/30 .

Essayer de modifier le programme pour qu'il admette ce cas.

```
100 SCREEN : PAGE
110 INPUT "Quel est l'heure actuelle (hh/mm/ss) ";A$
120 LET H1=VAL(LEFT$(A$,2))
130 LET M1=VAL(MID$(A$,4,2))
140 LET S1=VAL(MID$(A$,7,2))
150 MOVE (74,70,7): MOVE (166,70,1): MOVE (166,50,1)
160 MOVE (74,50,1): MOVE (74,70,1)
170 PRINT §(6,16)"  HORLOGE DIGITALE SMAKY 6"
180 FOR H=H1 TO 23: FOR M=M1 TO 59: FOR S=S1 TO 59
190 PRINT §(9,16)"      HEURES  MIN   SEC"
200 PRINT §(10,22);H: PRINT §(10,29);M: PRINT §(10,35);S
210 FOR N=1 TO 199: NEXT N
220 BEEP
230 NEXT S
240 LET S1=0
250 BEEP : BEEP
260 NEXT M
270 LET M1=0
280 BEEP : BEEP : BEEP
290 NEXT H
300 END
```

6) POEME

Ce programme génère des phrases aléatoirement, en choisissant un sujet, un verbe et un complément parmi 5 possibilités.

Les 5 sujets sont:	Les 5 verbes sont:	Les 5 compléments sont:
. le baladin	. s'éloigne	. le long des jardins
. l'automne malade	. meurt	. quand la neige tombe
. la rose	. fleurit	. dans ma mémoire
. une silhouette grise	. s'en va	. dans le brouillard
. l'heure	. passe	. si lentement

Les trois parties de phrase sont mémorisées dans 3 tableaux, S$, P$ et C$, qui ont été prévu pour pouvoir recevoir jusqu'à 20 possibilités.

```
100 PAGE
110 PRINT "CE PROGRAMME GENERE 100 PHRASES ALEATOIRES"
120 PRINT "OU...DE L'APOLLINAIRE AU KILOMETRE"
130 PRINT
140 DIM S$(20,20),P$(20,20),C$(20,20)
150 DATA "LE BALADIN","S'ELOIGNE","LE LONG DES JARDINS"
160 DATA "L'AUTOMNE MALADE","MEURT","QUAND LA NEIGE TOMBE"
170 DATA "LA ROSE","FLEURIT","DANS MA MEMOIRE"
180 DATA "UNE SILHOUETTE GRISE","S'EN VA","DANS LE BROUILLARD"
190 DATA "L'HEURE","PASSE","SI LENTEMENT"
200 FOR I=1 TO 5
210 READ S$(I,20),P$(I,20),C$(I,20)
220 NEXT I
230 LET S1=1: LET P1=1: LET C1=1
240 FOR J=1 TO 100
250 GOSUB 400
260 IF N=S1 THEN  GOTO 250
270 LET S1=N
280 PRINT S$(N,20);' ';
290 GOSUB 400
300 IF N=P1 THEN  GOTO 290
310 LET P1=N
320 PRINT P$(N,20);' ';
330 GOSUB 400
340 IF N=C1 THEN  GOTO 330
350 LET C1=N
360 PRINT C$(N,20)
370 FOR K=1 TO 1000: NEXT K
380 NEXT J
390 END
400 RANDOMIZE
410 LET N=INT(RND(6))
420 IF N=0 GOTO 410
430 RETURN
```

7) RESOLUTION D'UNE EQUATION DU 2eme DEGRE

Les solutions d'une équation de la forme $ax^2 + bx + c = 0$

sont: $x_{1,2} = \frac{\sqrt{-b \pm b^2 - 4ac}}{2a}$

Les solutions sont réelles si $b^2 - 4ac$ (appelé discriminant) est plus grand que 0.

Les solutions sont confondues si le discriminant est nul.

Les solutions sont imaginaires si le discriminant est plus petit que zéro.

La discriminant est calculé à la ligne 170.

```
100 PAGE
110 PRINT "CE PROGRAMME RESOUT UNE EQUATION DE LA FORME : AX2 + BX + C =
120 BEEP : PRINT
130 INPUT "VALEUR DE A= ";A
140 INPUT "VALEUR DE B= ";B
150 INPUT "VALEUR DE C= ";C
160 PRINT
170 LET D=B*B-4*A*C
180 IF A<>0 THEN 280
190 IF B=0 THEN 230
200 LET R=-C/B
210 BEEP : PRINT "PREMIER DEGRE : X =";R
220 GOTO 430
230 IF C=0 THEN 260
240 BEEP : PRINT "PREMIER DEGRE : EQUATION IMPOSSIBLE"
250 GOTO 430
260 BEEP : PRINT "PREMIER DEGRE : EQUATION INDETERMINEE"
270 GOTO 430
280 IF D<0 THEN 380
290 IF D=0 THEN 360
300 LET X1=(-B+SQR(D))/(2*A)
310 LET X2=(-B-SQR(D))/(2*A)
320 BEEP : PRINT "DEUXIEME DEGRE : RACINES REELLES :"
330 PRINT "X1=";X1
340 PRINT "X2=";X2
350 GOTO 430
360 BEEP : PRINT "DEUXIEME DEGRE : RACINE DOUBLE : X1=X2=";-B/(2*A)
370 GOTO 430
380 BEEP : PRINT "DEUXIEME DEGRE : RACINES COMPLEXES :"
390 PRINT "X1 : PARTIE REELLE      = ";-B/(2*A)
400 PRINT "      PARTIE IMAGINAIRE = ";-SQR(-D)
410 PRINT "X2 : PARTIE REELLE      = ";-B/(2*A)
420 PRINT "      PARTIE IMAGINAIRE = ";SQR(-D)
430 END
```

8) RESOLUTION D'UN SYSTEME DE 2 EQUATIONS DU PREMIER DEGRE A 2 INCONNUES

Les 2 équations sont de la forme:

$$a_1 x + b_1 y + c_1 = 0$$
$$a_2 x + b_2 y + c_2 = 0$$

et les solutions sont:

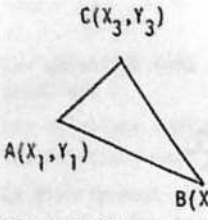
$$x = \frac{b_1 c_2 - b_2 c_1}{a_1 b_2 + a_2 b_1} \qquad \text{et} \qquad y = \frac{a_2 c_1 - a_1 c_2}{a_1 b_2 + a_2 b_1}$$

Il faut faire attention au cas où $a_1 b_2 + a_2 b_1$ est nul !

```
100 PAGE
110 PRINT
120 PRINT "RESOLUTION D'UN SYSTEME DE 2 EQUATIONS"
130 PRINT "DU PREMIER DEGRE A 2 INCONNUES"
140 PRINT
150 PRINT "FORME DU SYSTEME :"
160 PRINT TAB(18)"A1X + B1Y + C1 = 0"
170 PRINT TAB(18)"A2X + B2Y + C2 = 0"
180 BEEP : PRINT
190 PRINT "PREMIERE EQUATION :"
200 INPUT "VALEUR DE A1= ";A1
210 INPUT "VALEUR DE B1= ";B1
220 INPUT "VALEUR DE C1= ";C1
230 BEEP : PRINT
240 PRINT "DEUXIEME EQUATION :"
250 INPUT "VALEUR DE A2= ";A2
260 INPUT "VALEUR DE B2= ";B2
270 INPUT "VALEUR DE C2= ";C2
280 PAGE
290 PRINT §(0,0);"EQUATION 1 :"
300 PRINT A1;"*X + (";B1;")*Y + (";C1;") = 0"
310 PRINT "EQUATION 2:"
320 PRINT A2;"*X + (";B2;")*Y + (";C2;") = 0"
330 LET D=(A1*B2)-(A2*B1)
340 IF D<>0 THEN 440
350 IF B1=0 THEN 380
360 IF C1/B1=C2/B2 THEN 410
370 IF C1/A1=C2/A2 THEN 410
380 BEEP : PRINT
390 PRINT "PAS DE SOLUTIONS"
400 STOP
410 BEEP : PRINT
420 PRINT "INFINITE DE SOLUTIONS"
430 STOP
440 BEEP : PRINT
450 LET N1=(A2*C1)-(A1*C2)
460 LET N2=(B1*C2)-(B2*C1)
470 PRINT "SOLUTION :"
480 PRINT "X=";N2/D
490 PRINT "Y=";N1/D
500 END
```

9) TRIANGLE

Ce programme dessine un triangle connaissant les coordonnées de ses 3 sommets. De plus, il calcule les longueurs des 3 côtés, ainsi que la surface.



$$\ell_{AB} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

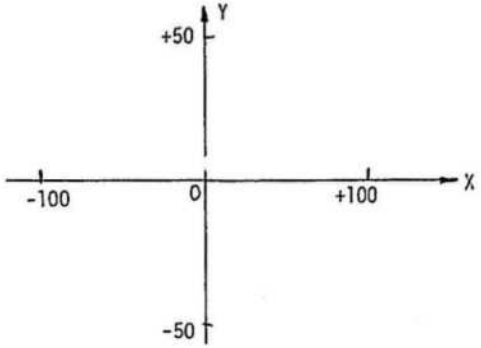
$$\ell_{BC} = \sqrt{(X_3 - X_2)^2 + (Y_3 - Y_2)^2}$$

$$\ell_{CA} = \sqrt{(X_1 - X_3)^2 + (Y_1 - Y_3)^2}$$

```
100 SCREEN : PAGE
110 PRINT "CE PROGRAMME TRACE UN TRIANGLE CONNAISSANT LES COORDONNEES"
120 PRINT "DE SES 3 SOMMETS ET CALCULE LA LONGUEUR DE SES 3 COTES"
130 PRINT "AINSI QUE SA SURFACE"
140 PRINT
150 PRINT "L'AXE DES X EST GRADUE ENTRE -127 ET +128"
160 PRINT "ET CELUI DES Y ENTRE -87 ET +89"
170 PRINT "POUR CERTAINES VALEURS DES COORDONNEES,LE TRACE DU TRIANGLE"
180 PRINT "PEUT DONC SE FAIRE HORS DE L'ECRAN"
190 BEEP : PRINT
200 PRINT "DONNER LES COORDONNEES DU POINT A"
210 INPUT "VALEUR DE X1= ";X1
220 INPUT "VALEUR DE Y1= ";Y1
230 PRINT "DONNER LES COORDONNEES DU POINT B"
240 INPUT "VALEUR DE X2= ";X2
250 INPUT "VALEUR DE Y2= ";Y2
260 PRINT "DONNER LES COORDONNEES DU POINT C"
270 INPUT "VALEUR DE X3= ";X3
280 INPUT "VALEUR DE Y3= ";Y3
290 PRINT : PAGE
300 PRINT "COORDONNEES :"
310 PRINT "A=(";X1;","Y1;")     B=(";X2;",";Y2;")          C=(";X3;",";Y3;")
320 LET D=SQR((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))
330 LET D1=SQR((X3-X2)*(X3-X2)+(Y3-Y2)*(Y3-Y2))
340 LET D2=SQR((X1-X3)*(X1-X3)+(Y1-Y3)*(Y1-Y3))
350 LET P=(D+D1+D2)/2
360 LET S=SQR(P*(P-D)*(P-D1)*(P-D2))
370 BEEP : PRINT "LONGUEURS :"
380 PRINT "AB=";D
390 PRINT "BC=";D1
400 PRINT "CA=";D2
410 PRINT : PRINT "SURFACE =";S
420 MOVE (127,119,7): MOVE (127,0,1): MOVE (255,59,7): MOVE (1,59,1)
430 MOVE (127,59,1)
440 MOVE (27,57,7): MOVE (27,61,1): MOVE (77,57,7): MOVE (77,61,1)
450 MOVE (177,57,7): MOVE (177,61,1): MOVE (227,57,7): MOVE (227,61,1)
460 MOVE (124,92,7): MOVE (130,92,1)
470 MOVE (124,25,7): MOVE (130,25,1)
480 PRINT §(15,33);"-50": PRINT §(4,33);"+50"
490 PRINT §(11,4);
500 PRINT "-100          -50                    +50          +100"
510 PRINT §(0,33);"Y": PRINT §(11,63);"X"
520 MOVE ((X+X1+127),((Y+Y1)/1.5)+59,7)
530 MOVE ((X+X2+127),((Y+Y2)/1.5)+59,3)
540 MOVE ((X+X3+127),((Y+Y3)/1.5)+59,3)
550 MOVE ((X+X1+127),((Y+Y1)/1.5)+59,3)
560 PRINT §(17,0);
570 END
```

10) DESSIN D'UNE FONCTION

Ce programme trace le graphique d'une fonction du premier ou du second degré dans le système d'axes suivant:




```
100 SCREEN : PAGE
110 PRINT
120 PRINT "CE PROGRAMME TRACE LE GRAPHIQUE D'UNE FONCTION "
130 PRINT "DU PREMIER OU DU SECOND DEGRE"
140 PRINT "FORME GENERALE DE LA FONCTION : Y = AX2 + BX + C"
150 PRINT "POUR TRAITER LE CAS PARTICULIER DE LA FONCTION X = A"
160 PRINT "FAIRE A=B=C=0"
170 PRINT
180 PRINT "L'AXE DES X EST GRADUE ENTRE -127 ET +128"
190 PRINT "ET CELUI DES Y ENTRE -87 ET +89"
200 PRINT "LE TRACE DE LA COURBE PEUT DONC SE FAIRE HORS DE L'ECRAN"
210 PRINT
220 BEEP : PRINT "FONCTION DE LA FORME : Y = AX2 + BX + C"
230 INPUT "VALEUR DE A= ";A
240 INPUT "VALEUR DE B= ";B
250 INPUT "VALEUR DE C= ";C
260 PAGE
270 IF ABS(A)+ABS(B)+ABS(C)=0 THEN 700
280 PRINT "FONCTION :"
290 PRINT "Y = (";A;")*X2 +(";B;")*X + (";C;")"
300 MOVE (127,119,7): MOVE (127,0,1): MOVE (255,59,7): MOVE (1,59,1)
310 MOVE (127,59,1)
320 MOVE (27,57,7): MOVE (27,61,1): MOVE (77,57,7): MOVE (77,61,1)
330 MOVE (177,57,7): MOVE (177,61,1): MOVE (227,57,7): MOVE (227,61,1)
340 MOVE (124,92,7): MOVE (130,92,1)
350 MOVE (124,25,7): MOVE (130,25,1)
360 PRINT §(15,33);"-50"
370 PRINT §(4,33);"+50"
380 PRINT §(11,4);
390 PRINT "-100          -50                    +50          +100"
400 PRINT §(0,33);"Y"
410 PRINT §(11,63);"X"
420 IF ABS(A)+ABS(B)+ABS(C)=0 THEN 740
430 FOR X=-127 TO 128
440 LET Y=((A*X*X)+(B*X)+C)/1.5
450 IF Y>58 THEN 490
460 IF Y+59<1 THEN 490
470 MOVE (X+127,Y+59,7)
480 MOVE (X+127,Y+59,3)
490 NEXT X
500 BEEP : PRINT §(12,0);
```

```
510 INPUT "AUTRE FONCTION ";A$
520 IF A$="OUI" THEN 570
530 IF A$="NON" THEN 660
540 PRINT : BEEP
550 PRINT "REPONDEZ CORRECTEMENT !"
560 GOTO 500
570 BEEP : PRINT §(13,0);
580 INPUT "VOULEZ-VOUS CONSERVER L'ANCIEN TRACE ";B$
590 IF B$="OUI" THEN 640
600 IF B$="NON" THEN 680
610 PRINT : BEEP
620 PRINT "REPONDEZ CORRECTEMENT"
630 GOTO 570
640 GOTO 220
650 PRINT
660 BEEP : SCREEN : PAGE
670 STOP
680 SCREEN : PAGE : GOTO 220
690 STOP
700 BEEP : PRINT "FONCTION DE LA FORME X=A"
710 INPUT "VALEUR DE A= ";X
720 PRINT "FONCTION : X=";X
730 GOTO 300
740 FOR Y=0 TO 119
750 MOVE (X+127,Y,7)
760 MOVE (X+127,Y,3)
770 NEXT Y
780 GOTO 500
790 END
```

Ces 10 exemples ont été écrits par Monsieur Yvan Dutoit, maître à Pully, que nous remercions très sincèrement de nous avoir permis de reproduire ces programmes.

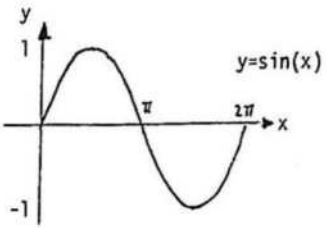
11) SINUSOIDE

Essayons de faire un programme qui dessine une sinusoïde sur l'écran.

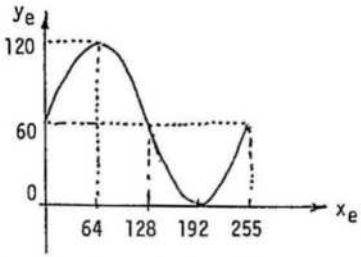
On sait que:

$$y = \sin(x) .$$

Pour tracer une période de la sinusoïde, il faut faire varier x de 0 à 2π. y prendra alors diverses valeurs comprises entre -1 et 1.



Pour pouvoir tracer la sinusoïde sur l'écran du SMAKY, il faut faire des changements de coordonnées. On a alors:



Pour trouver l'équation de cette fonction, commençons par écrire la table de correspondance suivante:

y	1	0	-1
y_e	120	60	0

D'où $y_e = 60 \cdot y + 60.$

L'équation devient: $y_e = 60 \cdot \sin(x) + 60.$

Le même raisonnement pour x donne successivement:

x	0	π	2π
x_e	0	128	256

d'où $x_e = \frac{2\pi x}{256}.$

L'équation s'écrit alors: $y_e = 60 \cdot \sin(\frac{2\pi x_e}{256}) + 60$

Il suffit maintenant d'écrire un programme qui fait varier x_e de 0 jusqu'à 255 et de calculer y_e pour chaque valeur de x_e.

```
FOR XE=1 TO 255
LET YE=60*SIN((2*3.14159*X1)/256)+60
MOVE (X1,Y1,1)
NEXT X1
```

Il faut encore effacer l'écran et initialiser le premier point avant de commencer.

Le programme devient:

```
10 PAGE
20 SCREEN
30 MOVE (0,60,7)
40 FOR X1=0 TO 255
50 LET Y1=60*SIN((2*3.14159*X1)/256)+60
60 MOVE (X1,Y1,1)
70 NEXT X1
```

Vous constatez que le dessin se fait lentement.

Pour accélérer, il est possible de faire varier X1 de 0 à 255 avec un pas plus grand que 1. Plus le pas est grand, plus le dessin se fera rapidement, mais plus il sera imprécis (rien n'est parfait!)

dessin rapide
approximatif

Notre programme devient:

```
10 PAGE
20 SCREEN
30 MOVE (0,60,7)
35 INPUT "INCREMENT:",I
40 FOR X1=0 TO 255 STEP I
50 LET Y1=60*SIN((2*3.14159*X1)/256)+60
60 MOVE (XE,Y1,1)
70 NEXT X1,
```

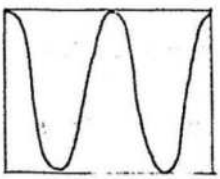
Remarque: les lignes 50 et 60 peuvent être rassemblées sur une seule ligne qui s'écrit:

```
MOVE (X1,60*SIN((2*3.14159*X1)/256)+60,1)
```

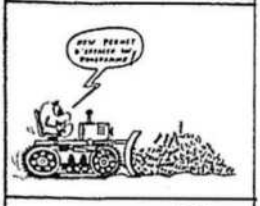
La variable Y1 n'est alors plus utilisée.

Exercice: Faire un programme qui dessine une sinusoïde avec deux périodes sur l'écran.

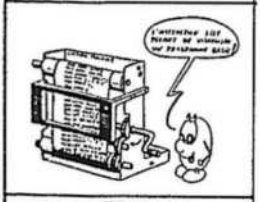
Le dessin qui devrait apparaître est:



RUN

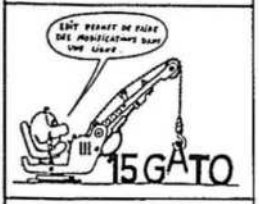


NEW



LIST

LIST 100
LIST 100,200
...



EDIT 200

HELP

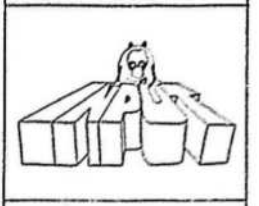
REN

AUTO 100,50

PRINT TOTO,2+A
PRINT "SALUT"
PRINT ∂(5,5);"*"

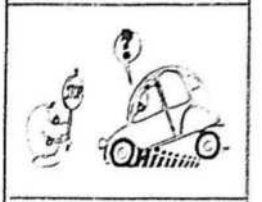
LET A=1

LET T=B/2



INPUT A,B

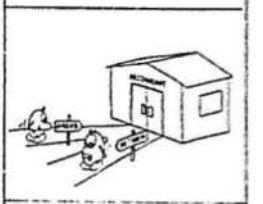
INPUT "TAPEZ UN
NOMBRE",X



STOP

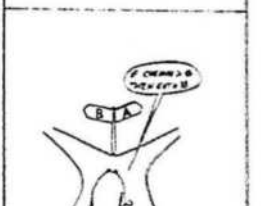
END

GOTO 10



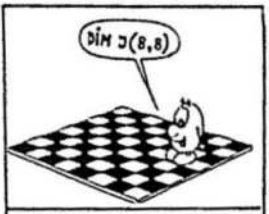
GOSUB 50

RETURN



IF A<B
THEN GOTO 5

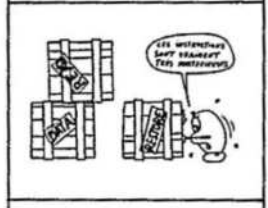
<	<=
>	>=
=	<>



DIM A1(2)

FOR I=1 TO 10
STEP 2

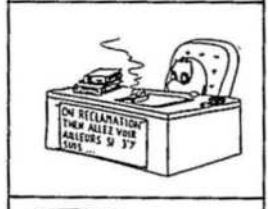
NEXT I



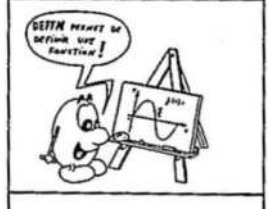
DATA 1,2,3

READ A,B,C

RESTORE

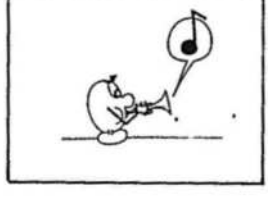


ON A+B GOTO 100
ON ESC THEN 130
ON ERR THEN STOP

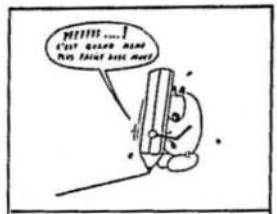


DEFFN A(X)=X+2

REM ***REMARQUE



BEEP

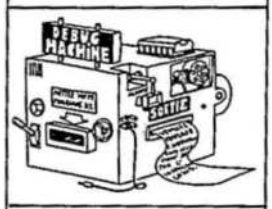


MOVE (X,Y,1)

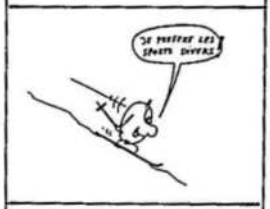
MOVE (10,20,M)

A$="BON"

B$="JOUR"



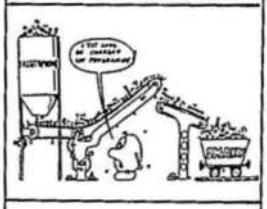
STEP ON
STEP
STEP OFF
TRACE
...



POKE 16384,65
M = PEEK(16384)

OUT 3,1
A = INP(4)

CALL 123



COPY $K7
COPY TOTO

ENTER $K7
ENTER TOTO

+	LN( )
-	LOG( )
*	EXP( )
/	ABS( )
↑	ARND( )
SQR( )	INT( )
SIN( )	SGN( )
COS( )	RND( )
TAN( )	RANDOMIZE
ATN( )	FRE(Ø)