BASIC

ADDITION +

Objet de l'opérateur

Opérer la somme de deux éléments.

Formulation

Variable = élément + élément

Exemple de programme

```
10 REM *** ADDITION (+) ***
20 X=12*2
30 Y=24/2
40 Z=X+Y
50 PRINT Z
60 END
```

Résultat de l'exécution

Ligne 20 : Affectation du résultat de l'opération 12*2 à la variable X
Ligne 30 : Affectation du résultat de l'opération 24/2 à la variable Y
Ligne 40 : Somme des valeurs représentées par les éléments X et Y et affectation de cette somme à la variable Z
Ligne 50 : Affichage de la valeur de Z, soit 36

A noter que...

- Certains systèmes emploient ce même opérateur pour assembler deux chaînes. Exemple :
  10 A$ = "ABCDEF" : B$ = "GHIJKL" : C$ = A$ + B$: PRINT C$
  donnera la chaîne : ABCDEFGHIJKL
- Certains systèmes utilisent ce même opérateur à la place de l'opérateur de relation logique **OR**, à l'intérieur de l'instruction conditionnelle **IF**... **THEN**... Exemple :
  10 IF (X = 1) + (Y = 2) THEN 1000

AFFECTATION =

Objet de l'opérateur

Separer les deux eléments d'une instruction, l'élément se trouvant à droite de l'opérateur étant toujours affecté à l'élément se trouvant à gauche de l'opérateur.

Formulation

Elément = Elément

Exemple de programme

```
10 REM *** AFFECTATION (=) ***
20 A=5+3
30 B=6+4
40 A=B
50 PRINT A
60 END
```

Résultat de l'exécution

Ligne 20 : Le résultat de l'addition 5 + 3, soit 8, est affecté à la variable A

Ligne 30 : Le résultat de l'addition 6 + 4, soit 10, est affecté à la variable B

Ligne 40 : L'élément B, à droite de l'opérateur, est affecté à l'élément A, à gauche de l'opérateur

Ligne 50 : Affichage de la valeur de A, soit 10

A noter que...

- Il est important de savoir faire la distinction entre la notion traditionnelle d'égalité de l'opérateur = (10 = 6 + 4) et sa fonction d'affectation ou, autrement dit, de transfert (A = B). En effet, à la ligne 40, A n'est pas égal à B puisque A vaut 8 et B, 10
- = est également utilisé comme opérateur de relation logique à l'intérieur de l'instruction conditionnelle **IF... THEN...** Exemple :
  10 IF A = B THEN 1000
  20 IF A$ = "ABCD" THEN 2000

AND

De l'anglais **AND**, et.

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, exige que *deux* ou *plusieurs conditions* soient remplies pour l'exécution de l'instruction qui suit **THEN**.

Formulation

IF condition **AND** condition **THEN** instruction

Exemple de programme

```
10 REM *** AND ***
20 A=10 :B=50 :C=100
30 IF A<B AND C>B THEN 50
40 PRINT"FAUX" :GOTO 100
50 PRINT"LES 2 CONDITIONS SONT REMPLIES"
60 A$="A" :B$="B" :C$="C"
70 IF A$<B$ AND C$>B$ THEN 90
80 PRINT"FAUX" :GOTO 100
90 PRINT"LES 2 CONDITIONS SONT REMPLIES"
```

Résultat de l'exécution

Ligne 50 : Les 2 conditions sont remplies (A, 10, est plus petit que B, 50 et C, 100 est plus grand que B)

Ligne 90 : Les 2 conditions sont remplies (A$, A est différent de B$, B et C$, C est différent de B$)

A noter que...

- Si l'une des deux conditions, précédant ou suivant **AND**, n'était pas remplie, le branchement n'aurait pas été exécuté et l'ordinateur aurait affiché *faux*
- Certains systèmes affichent —1 lorsque les conditions de **AND** sont remplies et 0 lorsqu'elles ne le sont pas

ASC

De l'anglais **A**merican **N**ational **S**tandard **C**ode for Information Interchange.

Objet de la fonction

Donne le code décimal ASCII correspondant à l'argument.

Formulation

ASC (variable de chaîne)

Exemple de programme

```
10 REM *** ASC ***
20 PRINT ASC("A")
30 C$="BCDE"
40 PRINT ASC(C$)
50 FOR B=2 TO 4
60 PRINT ASC(MID$(C$,B,1))
70 NEXT B
80 D$="ABCDEF"
90 PRINT ASC(RIGHT$(D$,1))
100 END
```

Résultat de l'exécution

Ligne 20 : 65 (code ASCII de la lettre A)
Ligne 40 : 66 (code de B, première lettre de la chaîne C$)
Ligne 60 : 67 pour C, 68 pour D et 69 pour E qui sont, respectivement, les deuxième, troisième et quatrième lettres de la chaîne C$
Ligne 90 : 70 pour F qui est la première lettre prise à l'extrême droite de la chaîne D$

A noter que...

- L'argument doit être inclus entre parenthèses
- **ASC** ne fournit le code que du premier caractère d'une chaîne
- Des fonctions de chaîne telles que **MID$** et **RIGHT$** ont été utilisées comme argument
- Si l'argument est une chaîne, elle doit être incluse entre guillemets à l'intérieur des parenthèses ("ABCDEF")

AUTO

De l'anglais **AUTOMATIC**, automatique.

Objet de la commande

Numéroter automatiquement les lignes d'instruction.

Exemples de formulation

1- **AUTO**
2- **AUTO** 10, 5
3- **AUTO** 10, 10
3- **AUTO** 10, 100
5- **AUTO** 456, 3

Résultat de l'exécution

1. Numérotation automatique de 10 en 10	10, 20, 30 etc.
2. Numérotation à partir de 10 par pas de 5	10, 15, 20 etc.
3. Numérotation à partir de 10 par pas de 10	10, 20, 30 etc.
4. Numérotation à partir de 10 par pas de 100	10, 110, 210 etc.
5. Numérotation à partir de 456 par pas de 3	456, 459, 462 etc.

A noter que...

- Dans certains systèmes, **AUTO**, sans argument donnera automatiquement le numéro 100 à la première ligne et 10 à l'incrément
- Pour arrêter la fonction de la commande, il suffira d'appuyer sur la touche **BREAK** ou **RETURN**, suivant le système utilisé
- Si la commande fournit un numéro de ligne déjà existant dans le programme, ce numéro sera suivi d'une astérisque pour prévenir l'effacement d'une instruction par inadvertance
- Certains systèmes gardent en mémoire le dernier argument entré après la commande lors d'une précédente numérotation
- Pour certains systèmes, ce sont les touches **CTRL** et **C**, pressées simultanément, qui arrêtent l'exécution de la commande **AUTO**

BREAK

De l'anglais **BREAK**, arrêt.

Objet de l'instruction

Arrêter l'exécution d'un programme au numéro de ligne indiqué.

Formulation

BREAK numéro de ligne

Exemple de programme

```
10 REM ♦♦♦ BREAK ♦♦♦
20 BREAK 30,50-70
30 PRINT "ARRET A LA LIGNE 30"
40 PRINT "PAS D'ARRET A LA LIGNE 40"
50 PRINT "ARRET A CHAQUE LIGNE DE 50 A 70"
60 PRINT "ARRET"
70 PRINT "ARRET"
80 PRINT "SUITE DU PROGRAMME ARRET"
90 END
```

Résultat de l'exécution

Ligne 20 : L'intruction ordonne un arrêt de l'exécution du programme à la ligne 30 et un arrêt à chaque ligne depuis la ligne 50 jusqu'à la ligne 70

Ligne 30 : Affiche la chaîne ARRET A LA LIGNE 30

Ligne 40 : La chaîne PAS D'ARRET A LA LIGNE 40 ne sera affichée que si, après l'arrêt à la ligne 30, on a tapé le mot **CO** suivi de la touche **ENTER**

A noter que...

- Après chaque arrêt provoqué par l'instruction **BREAK**, il faut taper au clavier la commande directe **CO** suivie de la touche **ENTER**, pour relancer l'exécution du programme

CHAIN

De l'anglais **CHAIN**, enchaîner.

Objet de l'instruction

Enchaîner l'exécution de deux programmes différents enregistrés séparément sur un périphérique (disques ou cassettes).

Formulation

CHAIN nom du programme, numéro de ligne

Exemple de programme

```
10 REM *** CHAIN ***
20 PRINT "PROGRAMME APPELE"
30 FOR B=1 TO 5: PRINT B+5: NEXT B
40 CHAIN APPELANT,120
50 END
100 PRINT "PROGRAMME APPELANT"
110 CHAIN APPELE
120 PRINT "TERMINE"
130 END
```

Résultat de l'exécution

Ligne 100 du programme APPELANT : Affiche la chaîne PROGRAMME APPELANT

Ligne 110 du programme APPELANT : Instruction d'enchaînement au programme APPELE

Ligne 20 du programme APPELE : Affiche la chaîne PROGRAMME APPELE

Ligne 30 du programme APPELE : Affiche le résultat du calcul de la boucle B

Ligne 40 du programme APPELE : Instruction d'enchaînement à la ligne 120 du programme APPELANT

Ligne 120 du programme APPELANT : Affiche la chaîne TERMINE

A noter que...

- Les deux programmes avaient été préalablement enregistrés sous les noms respectifs de APPELE et APPELANT

CHR$

De l'anglais **CHARACTER**, caractère.

Objet de la fonction

Représenter le caractère correspondant à un code exprimé en nombre décimal.

Formulation

CHR$ (nombre)

Exemple de programme

```
10 REM *** CHR$ ***
20 FOR B=65 TO 68
30 PRINT CHR$(B)
40 NEXT B
50 END
60 PRINT CHR$(65+101)
70 END
```

Résultat de l'exécution

Ligne 30 : A (caractère dont le code décimal est 65)
B pour B = 66
C pour B = 67
D pour B = 68
Ligne 60 : % (caractère dont le code décimal est 101)

A noter que...

- Le code utilisé est celui de l'American National Standard Code for Information Interchange
- Dans certains systèmes, le code ASCII comporte jusqu'à 225 caractères
- Le code doit être inclus entre parenthèses
- Le code peut être représenté par un nombre, une variable ou une expression (65 + X ! 2)
- Certains systèmes emploient le mot équivalent de **CHARS**

DATA

De l'anglais **DATA**, donnée.

Objet de l'instruction

Contenir une liste de données destinées à être lues par l'instruction **READ**

Formulation

DATA liste de données

Exemple de programme

```
10 REM *** DATA ***
20 FOR D=1 TO 3
30 READ X
40 PRINT X
50 NEXT D
60 DATA 10,20,30
70 FOR D=1 TO 3
80 READ X$
90 PRINT X$
100 NEXT D
110 DATA AA,BB,CC
```

Résultat de l'exécution

Ligne 40 : 10 (première donnée lue par **READ** X et affectée à X)
20 (deuxième donnée lue par **READ** X et affectée à X)
30 (troisième donnée lue par **READ** X et affectée à X)

Ligne 90 : AA (première chaîne lue par **READ** X$ et affectée à X$)
BB (deuxième chaîne lue par **READ** X$ et affectée à X$)
CC (troisième chaîne lue par **READ** X$ et affectée à X$)

A noter que...

- De nombreux systèmes permettent de mettre dans la liste des **DATA** une combinaison de valeurs numériques et de chaînes (10, AA, 20, BB, CC)
- Certains systèmes permettent l'emploi d'abréviations telles que **D.** ou **DAT** au lieu de **DATA**

DEF FN

De l'anglais **DEFINE**, définir, et **FUNCTION**, fonction.

Objet de l'instruction

Définir une fonction pour son utilisation ultérieure.

Formulation

DEF FN variable (variable) = expression

Exemple de programme

```
10 REM *** DEF FN ***
20 DEF FNA(X)=INT(X*100+.5)/100
30 DEF FNB(X)=INT(X*1000+.5)/1000
40 M=6.123456
50 N=6.123456
60 PRINT M;FNA(M)
70 PRINT N;FNB(N)
80 END
```

Résultat de l'exécution

Ligne 20 : Définition de la fonction A pour un arrondi à 2 décimales
Ligne 30 : Définition de la fonction B pour un arrondi à 3 décimales
Ligne 40 : La valeur affectée à la variable M comporte 6 décimales
Ligne 50 : La valeur affectée à la variable N comporte 6 décimales
Ligne 60 : 6.123456 6.12 (2 décimales conformément à la définition de la fonction FNA)
Ligne 70 : 6.123456 6.123(3 décimales conformément à la définition de la fonction FNB)

A noter que...

- Une variable fictive, soit X en l'occurrence, a été utilisée comme argument des deux fonctions : FNA(X) et FNB(X), cette même variable étant reprise dans les expressions situées à droite du signe d'affectation = INT (X * 100 + .5)/100
- L'emploi de cette variable fictive est indispensable pour la définition de la fonction

DELETE

De l'anglais **DELETE**, supprimer.

Objet de la commande

Supprimer une ou plusieurs lignes d'instruction.

Exemples de formulation

1. **DELETE** 10
2. **DELETE** 10—50
3. **DELETE** —50
4. **DELETE** 50—

Résultat de l'exécution

1. Supprime la ligne numéro 10
2. Supprime toutes les lignes numérotées de 10 à 50 y compris
3. Supprime toutes les lignes depuis le début du programme jusqu'à la ligne 50 y compris
4. Supprime la ligne numéro 50 et toutes celles qui suivent

A noter que...

- Pour supprimer la totalité d'un programme, on utilise les commandes **NEW** ou **SCRATCH** suivant le système utilisé
- Pour supprimer une ligne à la fois, on peut également taper le numéro de la ligne et appuyer sur la touche **ENTER** ou **RETURN**
- Certains systèmes permettent de formuler en une seule instruction la suppression de plusieurs lignes non consécutives. **DELETE** 70,120-150 permettra, par exemple, de supprimer simultanément la ligne 70 et les lignes numérotées de 120 à 150 inclusivement
- Certains systèmes permettent l'emploi de l'abréviation **DEL** pour **DELETE**

DIESE

Objet du caractère

Définir une ou plusieurs variables comme étant du type double précision.

Formulation

Variable #

Exemple de programme

```
10 REM *** DIESE (#) ***
20 A=4
30 B=3
40 C=A/B
50 PRINT C
60 C#=A/B
70 PRINT C#
80 END
```

Résultat de l'exécution

Ligne 50 : 1.333333 (résultat de l'opération en simple précision)
Ligne 70 : 1.333333373069763 (résultat de la même opération en double précision)

A noter que...

- En double précision, le résultat contient 17 chiffres en mémoire dont seuls 16 chiffres significatifs sont affichés
- Le nombre des chiffres significatifs varie suivant les systèmes. Pour le résultat de la ligne 40, seuls 6 chiffres significatifs ont été affichés
- L'utilisation inconsidérée des variables du type double précision réduit considérablement la capacité mémoire disponible

DIFFÉRENT DE < >

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, permet d'effectuer une comparaison entre deux valeurs numériques ou deux expressions arithmétiques logiques.

Formulation

IF variable **< >** variable **THEN** instruction

Exemple de programme

```
10 REM *** DIFFERENT DE < > ***
20 A=10 : B=100
30 IF B<>A THEN 50
40 PRINT"FAUX" : GOTO 100
50 PRINT"B<>A"
60 A$="A" : B$="Z"
70 IF B$<>A$ THEN 90
80 PRINT"FAUX" : GOTO 100
90 PRINT"B$<>A$"
100 END
```

Résultat de l'exécution

Ligne 50 : B < > A
Ligne 90 : B$ < > A$

A noter que...

- Pour comparer deux lettres ou deux chaînes, l'ordinateur se réfère au Code ASCII de ces lettres ou de ces chaînes. Le code de A étant 65 et celui de Z, 90, il est donc logique de dire que B$ est différent de A$
- Pour une comparaison entre deux chaînes, ABCD et AZCD, la comparaison se fait, à partir de la gauche, lettre par lettre. A est égal à A, B est différent de Z (B = 66 et Z = 90), il est donc logique de dire que la chaîne ABCD est différente de AZCD
- Certains systèmes emploient les symboles **><**, ≠ ou l'abréviation **NE** au lieu de **< >**

DIM

De l'anglais **DIMENSION**, dimension.

Objet de l'instruction

Fixer la "dimension" de la mémoire à réserver pour une liste ou un tableau.

Formulation

DIM variable (indices)

Exemple de programme

```
10 REM *** DIM ***
20 DIM A(4,3)
30 FOR I=1 TO 4
40 FOR J=1 TO 3
50 READ A(I,J)
60 NEXT I
70 NEXT I
80 DATA 1,3,1,4,2,5,1,4,2,3,2,5
90 END
```

Résultat de l'exécution

Ligne 20 : Réservation de 12 cases-mémoire pour les 12 éléments du tableau A
Ligne 30 : Boucle pour les 4 rangées du tableau
Ligne 40 : Boucle pour les 3 colonnes du même tableau
Ligne 50 : Lit les 12 données de la ligne **DATA** pour les affecter aux 12 éléments du tableau
Lignes 60/70 : Retour aux boucles respectives
Ligne 80 : Les 12 données

A noter que...

- Tous les indices commencent à 0. L'instruction **DIM** A (4) dimensionne une liste dont les éléments ont pour indices 0, 1, 2, 3 et 4.
- Certains systèmes permettent des tableaux de plus de deux dimensions **DIM** A (5, 6, 7)

DIVISION /

Objet de l'opérateur

Opérer la division d'un élément par un autre élément.

Formulation

Variable = élément / élément

Exemple de programme

```
10 REM *** DIVISION /
20 X=12*2
30 Y=24/2
40 Z=X/Y
50 PRINT Z
60 END
```

Résultat de l'exécution

Ligne 20 : Affectation du résultat de l'opération 12 * 2 à la variable X

Ligne 30 : Affectation du résultat de l'opération 24/2 (division) à la variable Y

Ligne 40 : Division de X par Y et affectation du résultat à la variable Z

Ligne 50 : Affichage de la valeur représentée par Z, soit 2

A noter que...

- Le choix du signe / comme operateur de la division permet d'éviter toute confusion avec le signe traditionnel (:) qui, lui, est un séparateur d'instructions, lorsque celles-ci sont écrites sur une seule et même ligne. Voir à **DEUX POINTS**

DOLLAR $

Le caractère **$**, placé après une variable, définit celle-ci comme étant du type alphanumérique.

Formulations

A$, A1$, AA$, ALPHA$

Exemple de programme

```
10 REM *** DOLLAR ($) ***
20 A$="LE SIGNE $"
30 A1$=" DEFINIT UNE VARIABLE"
40 AA$=" COMME ETANT DU TYPE"
50 ALPHA$=" ALPHANUMERIQUE"
60 PRINT A$ PRINT A1$
70 PRINT AA$ PRINT ALPHA$
80 PRINT A$+A1$+AA$+ALPHA$
90 PRINT LEN(A$+A1$+AA$+ALPHA$)
```

Résultat de l'exécution

Ligne 60 : LE SIGNE **$**
DEFINIT UNE VARIABLE
Ligne 70 : COMME ETANT DU TYPE
ALPHANUMERIQUE
Ligne 80 : LE SIGNE **$** DEFINIT UNE VARIABLE COMME ETANT DU TYPE ALPHANUMERIQUE
Ligne 90 : 66 (longueur totale des quatre chaînes)

A noter que...

- Seuls les deux premiers caractères de la variable ALPHA$ sont pris en compte par l'ordinateur pour l'exécution
- L'utilisation d'un mot clé comme variable (exemple : **CHR$**) est interdite
- La longueur des chaînes admise varie suivant les systèmes. Certains acceptent des chaînes d'une longueur maximum de 255 caractères
- Certains systèmes exigent une instruction de dimensionnement préalable — exemple : DIM A$ (75)

ÉGAL A =

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, permet d'effectuer une comparaison entre deux valeurs numériques ou deux expressions arithmétiques logiques.

Formulation

IF variable = variable **THEN** instruction

Exemple de programme

```
10 REM ♦♦♦ EGAL A =  ♦♦♦
20 A=10  B=10
30 IF B=A THEN 50
40 PRINT"A<>B"  GOTO 100
50 PRINT"B=A"
60 A$="A"  B$="A"
70 IF B$=A$ THEN 90
80 PRINT"B$<>A$"  GOTO 100
90 PRINT"B$=A$"
100 END
```

Résultat de l'exécution

Ligne 50 : B = A
Ligne 90 : B$ = A$

A noter que...

- Pour comparer deux lettres ou deux chaînes, l'ordinateur se réfère au Code ASCII de ces lettres ou de ces chaînes. Le code de A étant 65 et les deux variables B$ et A$ contenant la même lettre, il est donc logique de dire que B$ = A$
- Pour une comparaison entre deux chaînes, ABCD et AGCD, la comparaison se fait, à partir de la gauche, lettre par lettre. A est égal à A, B est plus petit que G (B = 66 et G = 71), donc, la chaîne ABCD n'est pas égale à la chaîne AGCD
- Certains systèmes emploient l'abréviation **EQ** au lieu de =

ELSE

De l'anglais **ELSE**, sinon.

Objet de l'instruction

Indiquer l'instruction qui devra être exécutée dans le cas où la condition spécifiée par **IF**...**THEN**...n'est pas satisfaite.

Formulation

IF condition **THEN** instruction **ELSE** instruction

Exemple de programme

```
10 REM *** ELSE ***
20 A=1  B=0
30 IF A=B THEN PRINT "OUI" ELSE PRINT "NON"
40 IF B<A THEN PRINT "OUI" ELSE PRINT "NON"
50 FOR X=1 TO 5
60 IF X<5 THEN 70 ELSE 90
70 PRINT X;
80 NEXT X
90 END
```

Résultat de l'exécution

Ligne 30 : Affiche NON puisque A n'est pas égal à B
Ligne 40 : Affiche NON puisque B n'est pas plus grand que A
Ligne 70 : Affiche 1 2 3 4, la condition ayant été satisfaite pour X = 1, X = 2, X = 3, X = 4. Lorsque X a pris la valeur 5, l'exécution est passée à la ligne 90 comme le précisait l'instruction suivant **ELSE**

A noter que...

- **ELSE** permet d'écrire sur une seule et même ligne les deux options possibles suivant que la condition a été satisfaite ou non
- Si l'on ne dispose que de l'instruction **IF**... **THEN**..., la ligne 40 pourrait s'écrire en 2 lignes :
  40 IF B<A THEN PRINT "OUI"
  45 PRINT "NON"

END

De l'anglais **END**, fin.

Objet de l'instruction

Indiquer la fin d'un programme ou d'une partie de programme

Formulation

END

Exemple de programme

```
10 REM *** END ***
20 GOSUB 100
30 IF A=X*2 THEN 130
40 END
100 INPUT X
110 A=14
120 RETURN
130 PRINT"CONDITION SATISFAITE"
140 END
```

Résultat de l'exécution

Ligne 20 : Transfert à la sous-routine qui commence à la ligne 100
Ligne 100 : Entrée de la valeur à affecter à X
Ligne 110 : Affectation de la valeur 14 à la variable A
Ligne 120 : Retour à la ligne 30 (qui suit celle de **GOSUB**)
Ligne 30 : Instruction conditionnelle. Si X = 7, la condition est satisfaite, l'exécution se branche à la ligne 130 (affichage de CONDITION SATISFAITE) puis à la ligne 140 qui met fin au programme
Si la valeur de X est différente de 7, l'exécution se branche à la ligne 40 qui met fin au programme

A noter que...

- Sur de nombreux systèmes, on peut omettre l'instruction **END** à la "fin logique" du programme, par exemple à la ligne 140 du programme ci-dessus

FOR

De l'anglais **FOR**, pour.

Objet de l'instruction

Affecter à la variable d'une boucle répétitive les valeurs successives comprises entre les valeurs initiale et finale indiquées de part et d'autre du mot **TO**

Formulation

FOR variable = valeur initiale **TO** valeur finale

Exemple de programme

```
10 REM *** FOR ***
20 FOR B=1 TO 4
30 PRINT B;
40 NEXT B
50 END
```

Résultat de l'exécution

Ligne 30 : 1 2 3 4

A noter que...

- B a pris successivement les valeurs 1, 2, 3, et 4, c'est-à-dire de 1 (valeur initiale) à (**TO**) 4 (valeur finale) de la boucle B
- Les instructions comprises entre **FOR** et **NEXT** sont exécutées un nombre de fois égal au nombre suivant **TO**
- La valeur initiale de la boucle est incrémentée de 1 si aucune autre indication n'a été précisée après le mot **STEP** (pas)
- Les valeurs initiale et finale ainsi que l'incrément peuvent être représentées par un nombre, une variable ou une expression
- Dans certains systèmes, la variable suivant le mot **NEXT** est facultative mais, dans certains cas, elle peut être indispensable
- Plusieurs boucles peuvent être écrites dans un même programme mais elles doivent obligatoirement s'emboîter l'une dans l'autre
- Certains systèmes permettent l'emploi de l'abréviation F. pour **FOR**

GET

De l'anglais **GET**, chercher.

Objet de l'instruction

Chercher un caractère numérique ou alphanumérique tapé au clavier.

Formulation

GET variable

Exemple de programme

```
10 REM *** GET ***
20 GET A$:IF A$<>"A"THEN 20
30 PRINT "VOUS AVEZ TAPE "A""
40 GET B :IF B <> 1 THEN 40
50 PRINT "VOUS AVEZ TAPE "1""
60 END
```

Résultat de l'exécution

Ligne 20 : Instruction ordonnant de chercher un caractère alphanumérique. Tant que l'on n'aura pas tapé la lettre A, l'exécution sera figée. Si l'on tape la lettre A, la condition sera satisfaite et l'exécution se branchera à la ligne 30

Ligne 40 : Instruction ordonnant de chercher une valeur numérique. Si l'on tape 1, la condition sera satisfaite et l'exécution se branchera à la ligne 50

A noter que...

- Cette instruction est fréquemment employée dans les programmes de jeux. Elle remplace avantageusement l'instruction **INPUT** pour ne pas perturber l'image de l'écran pendant un jeu
- Elle est toujours suivie de l'instruction conditionnelle **IF...THEN**, les deux instructions étant formulées sur une même ligne

GOSUB

De l'anglais **GO**, aller à, et **SUB**, sous.

Objet de l'instruction

Interrompre l'exécution séquentielle du programme et transférer l'ordinateur à une section dont le numéro de ligne est indiqué après **GOSUB**

Formulation

GOSUB numéro de ligne

Exemple de programme

```
10 REM ♦♦♦ GOSUB... ♦♦♦
20 GOSUB 50
30 PRINT"RETOUR A LA LIGNE 30"
40 END
50 PRINT"SUIS PASSE A LA LIGNE 50"
60 RETURN
```

Résultat de l'exécution

Ligne 50 : SUIS PASSE A LA LIGNE 50
Ligne 30 : RETOUR A LA LIGNE 30

A noter que...

- Le transfert s'est fait de la ligne 20 à la ligne 50
- L'instruction **RETURN** est obligatoirement placée à la fin de la sous-routine pour la reprise de l'exécution séquentielle du programme
- Le transfert de retour **RETURN** s'est fait de la ligne 60 à la ligne 30, ligne qui suit immédiatement celle de l'instruction **GOSUB** (ligne 20)
- Certains systèmes permettent d'écrire **GOSUB** en deux mots **GO SUB**
  L'orthographe **GOSUB** est rétablie automatiquement, et dans les listings et pour l'exécution du programme
- Certains systèmes emploient l'abréviation **GOS.** au lieu de **GOSUB**

GOTO

De l'anglais **GO TO**, aller à.

Objet de l'instruction

Transférer l'exécution au numéro de la ligne indiqué.

Formulation

GOTO numéro de ligne

Exemple de programme

```
10 REM *** GOTO ***
20 GOTO 40
30 END
40 PRINT "JE PASSE A LA LIGNE 40"
50 GOTO 70
60 END
70 PRINT "JE PASSE A LA LIGNE 70"
80 END
```

Résultat de l'exécution

Ligne 40 : Affiche la chaîne SUIS PASSE A LA LIGNE 40
Ligne 70 : Affiche la chaîne SUIS PASSE A LA LIGNE 70

A noter que...

- Les transferts se sont faits de la ligne 20 à la ligne 40 et de la ligne 50 à la ligne 70
- Les lignes 30 et 60 ont été "sautées"
- Certains systèmes permettent d'écrire **GOTO** en deux mots séparés **GO TO**. L'orthographe **GOTO** est rétablie automatiquement, et dans les listings et pour l'exécution
- Certains systèmes permettent l'emploi d'abréviations telles que **G.** ou **GOT** au lieu de **GOTO**

GUILLEMETS " "

Objet de l'opérateur

Délimiter une chaîne alphanumérique.

Formulations

```
1 REM ♦♦♦ GUILLEMETS (" ") ♦♦♦
2 READ D$,D1$,D2$
3 PRINT "A"
4 PRINT ASC("A")
5 PRINT "10"
6 INPUT "ENTREE",A
7 I$="INFORMATIQUE" PRINT I$
8 INPUT E$ PRINT E$
9 PRINT D$,D1$,D2$
10 DATA "DONNEE",DONNEE," DONNEE"
```

Observations

Ligne 3 : A est une chaîne puisqu'elle est placée entre guillemets.

Ligne 4 : L'argument de la fonction **ASC** étant un caractère de chaîne, celui-ci est donc écrit entre guillemets.

Ligne 5 : Le nombre 10 est imprimé en tant que chaîne, donc il doit être inclus entre guillemets.

Ligne 6 : Lorsque l'instruction **INPUT** fait également fonction de **PRINT** la chaîne doit être incluse entre guillemets et séparée de la variable numérique par un point-virgule ;

Ligne 7 : Une chaîne affectée à une variable alphanumérique doit être incluse entre guillemets.

Ligne 8 : Par contre, une chaîne entrée par l'instruction **INPUT** n'a pas à être incluse entre guillemets.

Ligne 10 : Pour certains systèmes, les chaînes énumérées dans une ligne **DATA** peuvent ne pas être incluses entre guillemets. Par contre, lorsque ces chaînes comportent des espaces, des caractères graphiques etc., elles doivent obligatoirement être incluses entre guillemets.

IF

De l'anglais **IF**, si.

Objet de l'instruction

Employé avec **THEN**, sert à tester une condition. Lorsque la condition de l'opérateur de relation suivant **IF** est satisfaite, l'ordinateur exécute l'instruction indiquée après **THEN**, sinon, il passe à la ligne suivant celle de l'instruction **IF...THEN**.

Formulation

IF condition **THEN** instruction

Exemple de programme

```
10 REM ♦♦♦ IF ♦♦♦
20 INPUT X
30 IF X=3 THEN X=3333: PRINT X
40 IF X=6 THEN PRINT "D'ACCORD"
50 IF X=0 THEN 70
60 GOTO 20
70 END
```

Résultat de l'exécution

Ligne 30 : 3333 (si à la ligne 20, 3 a été affecté à X)
Ligne 40 : D'ACCORD (si à la ligne 20, 6 a été affecté à X)
Ligne 50 : READY (si a la ligne 20, 0 a été affecté à X)

A noter que...

- A la ligne 30, l'instruction après **THEN** indique l'affectation d'une nouvelle valeur (3333) à la variable **X**
- A la ligne 40, l'instruction après **THEN** demandait l'impression d'une chaîne
- A la ligne 50, l'instruction après **THEN** donnait le numéro de la ligne à laquelle devait continuer l'exécution du programme
- Certains systèmes permettent d'omettre le mot **THEN** dans certaines instructions

IF...GOSUB...

De l'anglais **IF**, si, **GO**, aller à, et **SUB**, sous.

Objet de l'instruction

Transférer l'exécution à une sous-routine dont le numéro de ligne est spécifié après **GOSUB**

Formulation

IF condition **GOSUB** numéro de ligne

Exemple de programme

```
10 REM ♦♦♦ IF...GOSUB... ♦♦♦
20 Y=12
30 IF Y=12 GOSUB 60
40 PRINT "RETOUR A LA LIGNE 40"
50 END
60 PRINT "SUIS PASSE A LA LIGNE 60"
70 RETURN
80 END
```

Résultat de l'exécution

Ligne 60 : Affiche la chaîne SUIS PASSE A LA LIGNE 60
Ligne 40 : Affiche la chaîne RETOUR A LA LIGNE 40

A noter que...

- Le tranfert s'est fait de la ligne 30 à la ligne 60, la condition de la ligne 30 étant satisfaite (Y = 12)
- L'instruction **RETURN** est obligatoirement placée à la fin de la sous-routine pour la reprise de l'exécution séquentielle du programme
- Le transfert de retour (**RETURN**) s'est fait de la ligne 70 à la ligne 40, ligne qui suit immédiatement celle de l'instruction **GOSUB** (ligne 30)
- L'instruction conditionnelle **IF... GOSUB...** n'est pas acceptée par certains systèmes
- Certains systèmes permettent l'emploi d'une abréviation telle que **IF... GOS** au lieu de **IF... GOSUB**

IF...GOTO...

De l'anglais **IF**, si, et **GO TO**, aller à.

Objet de l'instruction

Tester une condition. Lorsque la condition de l'opérateur de relation est satisfaite, l'ordinateur se branche au numéro de la ligne précisée après **GOTO**

Formulation

IF condition **GOTO** numéro de ligne

Exemple de programme

```
10 REM *** IF...GOTO... ***
20 Y=12
30 IF Y<20 GOTO 50
40 END
50 PRINT"Y<20, SUIS PASSE A LA LIGNE
60 IF Y>10 GOTO 80
70 END
80 PRINT"Y>10, SUIS PASSE A LA LIGNE
90 END
```

Résultat de l'exécution

Ligne 50 : Y < 20, SUIS PASSE A LA LIGNE 50
Ligne 80 : Y > 10, SUIS PASSE A LA LIGNE 80

A noter que...

- Les transferts d'exécution se sont faits de la ligne 30 à la ligne 50 et de la ligne 60 à la ligne 80
- Sur certains systèmes, on peut employer le mot **THEN** à la place de **GOTO** pour un transfert d'exécution
- Certains systèmes emploient les abréviations **IF...G.** ou **IF... GOT** au lieu de **IF... GOTO**

IF...THEN...

De l'anglais **IF**, si, et **THEN**, alors.

Objet de l'instruction

Tester une condition. Lorsque la condition de l'opérateur de relation est satisfaite, l'ordinateur exécute l'instruction indiquée après **THEN**, sinon, il passe à la ligne suivant celle de l'instruction **IF... THEN**

Formulation

IF condition **THEN** instruction

Exemple de programme

```
10 REM *** IF...THEN... ***
20 Y=12
30 FOR X=1 TO 6
40 IF X=6 THEN PRINT X
50 IF X=5 THEN END
60 IF X=4 THEN 90
70 IF X=3 THEN Y=1234 : PRINT Y
80 NEXT X
90 PRINT"X=4, SUIS PASSE A LA LIGNE 90"
100 END
```

Résultat de l'exécution

Ligne 70 : 1234 (la valeur de Y a été changée et imprimée lorsque X a pris la valeur 3)

Ligne 90 : X = 4, SUIS PASSE A LA LIGNE 90 (cette ligne a été imprimée lorsque X a pris la valeur 4)

A noter que...

- La boucle **FOR** n'a pas été entièrement exécutée et X = 6 n'a pas été imprimé car le programme s'est arrêté (ligne 50) lorsque X a pris la valeur 5
- Dans les systèmes qui permettent plus d'une instruction par ligne : si la condition n'est pas satisfaite, l'exécution passe à la ligne suivante et *non* à l'instruction qui suit sur la même ligne que **IF...**
- Certains systèmes emploient les abréviations **IF...T.** ou **IF...THE**

INFÉRIEUR OU ÉGAL A <=

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, permet d'effectuer une comparaison entre deux valeurs numériques ou deux expressions arithmétiques logiques.

Formulation

IF variable < = variable **THEN** instruction

Exemple de programme

```
10 REM *** INFERIEUR OU EGAL A (<=) ***
20 A=10 :B=100 :C=10
30 IF A<=B THEN 50
40 PRINT"FAUX" :GOTO 90
50 PRINT"A<B "
60 IF A<=C THEN 80
70 PRINT"FAUX" :GOTO 90
80 PRINT"A=C"
90 END
```

Résultat de l'exécution

Ligne 50 : A < B (10, en effet, est plus petit que 100).
Ligne 80 : A = C (10, en effet, est égal à 10)

A noter que...

- L'opérateur < = est également employé pour comparer des chaînes. Pour cette comparaison, l'ordinateur se réfère au Code ASCII. Le code de A étant 65 et celui de B, 66, la lettre A est évaluée comme étant plus petite que B
- La comparaison entre deux chaînes de longueur inégale, ABCDEF et ABG, se fait, à partir de la gauche, lettre par lettre. A est égal à A, B est égal à B, mais C étant plus petit que G (C = 67, G = 71), la chaîne ABCDEF est évaluée comme étant plus petite que ABG, bien que comptant 3 lettres de plus
- Certains systèmes emploient les symboles = <, ≤ ou l'abréviation **LE** au lieu de < =

INPUT

De l'anglais **INPUT**, entrer.

Objet de l'instruction

Permet d'entrer des données par l'entremise du clavier.

Formulation

INPUT variable

Exemple de programme

```
10 REM *** INPUT ***
20 INPUT C
30 INPUT C$
40 INPUT X,Y$,Z
50 INPUT "DONNEES";D
60 PRINT C
70 PRINT C$
80 PRINT X;Y$;Z
90 PRINT D
100 END
```

Résultat de l'exécution

Ligne 60 : 12345 (valeur entrée, par exemple, à la ligne 20)
Ligne 70 : ZONES (chaîne entrée, par exemple, à la ligne 30)
Ligne 80 : 3 B 6 (données entrées, par exemple, à la ligne 40 pour être affectées respectivement aux variables **X**, **Y$** et **Z**

A noter que...

- **INPUT** peut être suivi d'une variable numérique, d'une variable de chaîne ou d'une combinaison des deux (ligne 40)
- A la ligne 50, la formulation de l'instruction **INPUT** englobe celle de **PRINT**, le mot DONNEES entre guillemets précédant la variable D
- Le texte et la variable doivent être séparés par un point-virgule
- Certains systèmes permettent l'emploi de l'abréviation **IN.** au lieu de **INPUT**

LEFT$

De l'anglais **LEFT**, gauche.

Objet de la fonction

Extraire un certain nombre de caractères d'une chaîne alphanumérique en partant de l'extrême-gauche.

Formulation

LEFT$ (variable, nombre)

Exemple de programme

```
10 REM ♦♦♦ LEFT$ ♦♦♦
20 A$="INFORMATIQUE"
30 FOR B= 1 TO 3
40 PRINTLEFT$(A$,B)
50 NEXT B
60 END
```

Résultat de l'exécution

Ligne 40 : I pour B = 1
IN pour B = 2
INF pour B = 3

A noter que...

- La chaîne doit être incluse entre guillemets : "INFORMATIQUE"
- Le nombre de caractères à extraire peut être désigné par un nombre, une variable ou une expression (A$, X ! 2 + 1)
- Une virgule doit séparer la chaîne, du nombre, de la variable ou de l'expression
- Si le nombre, la variable ou le résultat de l'expression comporte des décimales (4.56), l'ordinateur n'en retiendra que le nombre entier (4)
- L'argument doit être inclus entre parenthèses ("INFORMATIQUE",5)
- Les variables alphanumériques sont toujours suivies du signe distinctif **$**
- Certains systèmes utilisent le mot **LEFT** sans le signe distinctif **$**

LEN

De l'anglais **LENGTH**, longueur.

Objet de la fonction

Calculer le nombre de caractères compris dans une chaîne.

Formulation

LEN (variable)

Exemple de programme

```
10 REM *** LEN ***
20 A$="INFORMATIQUE"
30 PRINT LEN(A$)
40 PRINT LEN("INFORMATIQUE")
50 PRINT LEN("I          E")
60 B$="1980"
70 PRINT LEN(A$+B$)
80 END
```

Résultat de l'exécution

Ligne 30 : 12 (La variable A$ contient la chaîne INFORMATIQUE qui est un mot de 12 lettres)
Ligne 40 : 12 (La chaîne INFORMATIQUE comporte 12 lettres)
Ligne 50 : 12 (La chaîne I.....E comporte 12 caractères, 2 lettres et 10 espaces)
Ligne 70 : 16 (Les chaînes A$ et B$ comportent, à elles deux, 16 caractères, 12 lettres dans A$ plus 4 chiffres dans B$)

A noter que...

- La longueur d'une chaîne est égale au nombre de caractères (lettres, chiffres, espaces, signes, caractères graphiques, etc.) qu'elle comporte
- On peut opérer, dans certains systèmes, l'assemblage de plusieurs chaînes en utilisant l'opérateur + (A$ + B$)
- L'argument doit être inclus entre parenthèses. Si l'argument est une chaîne, elle doit être incluse entre guillemets à l'intérieur des parenthèses ("INFORMATIQUE")

LET

De l'anglais **LET**, soit.

Objet de l'instruction

Indiquer une affectation.

Formulation

LET variable = variable

Exemple de programme

```
10 REM *** LET ***
20 LET X=10/2
30 PRINT X
40 Y=20/5
50 PRINT Y
60 END
```

Résultat de l'exécution

Ligne 20 : Affectation à la variable X du résultat de la division 10/2
Ligne 30 : Affichage de la valeur contenue dans X, soit 5
Ligne 40 : Affectation à la variable Y du résultat de la division 20/5
Ligne 50 : Affichage de la valeur contenue dans Y, soit 4

A noter que...

- Pour certains systèmes, **LET** est facultatif comme le démontre la ligne 40
- Certains programmeurs l'utilisent cependant pour repérer les lignes où ils ont changé la valeur affectée à la même variable.
  Exemple :
  10 A = 10 * 2
  100 LET A = 10 * 3

LIST

De l'anglais **LIST**, reproduire une liste.

Objet de la commande

Afficher le programme résident en mémoire.

Formulation

LIST numéro (s) de ligne

Exemple de programme

```
10 REM *** LIST ***
20 C=100 : PRINT C
30 C=150 : PRINT C
40 C=200 : GOTO 60
50 END
60 IF C=200 THEN LIST 10-50
70 END
```

Résultat de l'exécution

Ligne 20 : Affiche la valeur affectée à la variable C, soit 100
Ligne 30 : Affiche la nouvelle valeur affectée à C, soit 150
Ligne 40 : Affectation de la valeur 200 à C et branchement à la ligne 60
Ligne 60 : La condition étant satisfaite (C = 200), affichage des lignes du programme numérotées de 10 à 50

A noter que...

- **LIST** a été employée comme instruction à la ligne 60
- **LIST**, en mode direct, et sans indication de numéro de ligne, permet d'afficher la totalité du programme
- L'affichage d'un programme peut être arrêté en appuyant sur la touche spécifique au système utilisé : **STOP**, **CTRL C**, **CTRL S**, **SHIFT A**, etc.
- Certains systèmes permettent l'emploi d'abréviations telles que **L.**, **LI**, **LIS** au lieu de **LIST**

LOAD

De l'anglais **LOAD**, charger.

Objet de la commande

Rentrer en mémoire un programme préalablement enregistré.

Formulation

LOAD nom de programme

Exemple de programme

```
10 REM *** LOAD ***
20 C=200
30 PRINT C
40 IF C=200 THEN LOAD "PN1"
50 END
```

Résultat de l'exécution

Ligne 30 : Affiche la valeur de C, soit 200
Ligne 40 : La condition étant satisfaite (C = 200), l'instruction ordonne de charger en mémoire le programme intitulé PN1

A noter que...

- **LOAD** a été employée comme instruction à la ligne 40
- En mode direct, **LOAD**, sans aucune autre indication, rentrera en mémoire le premier programme rencontré sur la cassette n° 1
- **LOAD**, suivie du nom d'un programme, rentrera en mémoire le programme spécifié
- Certains systèmes emploient le mot **CLOAD** au lieu de **LOAD**
- Le chargement d'un programme en mémoire efface automatiquement celui qui résidait en mémoire lors du chargement
- La commande **LOAD** s'accompagne de l'emploi de la touche **PLAY** qui déclenche la lecture de la cassette et le chargement du programme

MID$

De l'anglais **MIDDLE**, milieu.

Objet de la fonction

Extraire un certain nombre de caractères d'une chaîne alphanumérique en partant d'une position spécifiée dans l'argument.

Formulation

MID$ (chaîne, nombre, nombre)

Exemple de programme

```
10 REM *** MID$ ***
20 A$="INFORMATIQUE"
30 FOR B=4 TO 6
40 PRINT MID$(A$,3,B)
50 NEXT B
60 END
```

Résultat de l'exécution

FORM 4 lettres (B = 4) à partir de F (troisième lettre de INFORMATIQUE)
FORMA 5 lettres (B = 5) à partir de F (troisième lettre de INFORMATIQUE)
FORMAT 6 lettres (B = 6) à partir de F (troisième lettre de INFORMATIQUE)

A noter que...

- La chaîne doit être incluse entre guillemets : "INFORMATIQUE"
- Le nombre de caractères à extraire, comme la position à partir de laquelle l'extraction doit se faire, peuvent être désignés par un nombre, une variable ou une expression
- Une virgule doit séparer les trois paramètres de l'argument, ce dernier devant être inclus entre parenthèses (A$, 3, 4)
- Si le nombre, la variable ou le résultat de l'expression, utilisés comme paramètres, comporte des décimales, l'ordinateur n'en retiendra que le nombre entier
- Dans le cas où l'on omettrait le paramètre désignant le nombre de caractères à extraire, l'ordinateur affichera automatiquement tous les caractères de la chaîne qui suivent la position indiquée
- Certains systèmes emploient les mots équivalents **EXT$**, **MID** ou **SEG$**

MOD

De l'anglais **MODU**LO, modulo.

Objet de l'instruction

Obtenir le reste d'une division.

Formulation

PRINT nombre **MOD** nombre

Exemple de programme

```
10 REM *** MOD ***
20 PRINT 27 MOD 4
30 PRINT 45 MOD 8
40 END
```

Résultat de l'exécution

Ligne 20 : 3 (Reste de la division de 27 par 4.
6 * 4 = 24 + 3 = 27)
Ligne 30 : 5 (Reste de la division de 45 par 8.
5 * 8 = 40 + 5 = 45)

A noter que...

- Certains systèmes extraient automatiquement le nombre entier du reste et ignorent les décimales
- Dans certains systèmes, la formulation de l'exécution est : **MOD** (X,Y), X représentant le dividende et Y, le diviseur

MULTIPLICATION *

Objet de l'opérateur

Opérer la multiplication d'un élément par un autre élément.

Formulation

Variable = élément * élément

Exemple de programme

```
10 REM *** MULTIPLICATION (*) ***
20 X=12*2
30 Y=X*2
40 Z=X*Y
50 PRINT Z
60 END
```

Résultat de l'exécution

Ligne 20 : Affectation à la variable X du résultat de la multiplication 12 * 2

Ligne 30 : Affectation à la variable Y du résultat de la multiplication X * 2

Ligne 40 : Affectation à la variable Z du résultat de la multiplication de X par Y

Ligne 50 : Affichage de la valeur de Z, soit 1152

A noter que...

- Le choix du signe * comme opérateur de la multiplication évite toute confusion avec le signe traditionnel (×) qui lui représente la lettre X
- Certains systèmes utilisent le signe * à la place de l'opérateur de relation logique **AND**, à l'intérieur de l'instruction conditionnelle **IF...THEN...** Exemple :
  10 IF (X = 1) * (Y = 2) THEN 1000

NEW

De l'anglais **NEW**, nouveau.

Objet de la commande

Effacer le programme résident en mémoire.

Formulation

NEW

A noter que...

- Cette commande est généralement tapée avant d'entrer un nouveau programme en mémoire
- Lorsque l'on tape **LOAD** pour charger en mémoire un programme quelconque à partir d'une cassette, la commande **NEW** est sous-entendue : le programme résident en mémoire est effacé et remplacé par le nouveau qui est chargé par l'instruction **LOAD**
- Pour s'assurer que la commande **NEW** a bien été exécutée, on peut, par précaution, taper la commande **LIST** avant d'entrer le nouveau programme
- Certains systèmes permettent l'emploi d'abréviations telles que **N.** ou **NE** au lieu de **NEW** ou du mot équivalent **SCRATCH** et de son abréviation **SCR**
- **NEW** pourrait, à la rigueur, être utilisée comme instruction au cours d'un programme. Exemple :
  10 INPUT X
  20 IF X = 3 THEN NEW
  A l'exécution, si l'on entre la valeur 3 à la ligne 10, la condition de la ligne 20 étant satisfaite, l'instruction sera exécutée. Si l'on tape **LIST**, on constatera que le programme a été effacé

NEXT

De l'anglais **NEXT**, suivant.

Objet de l'instruction

Renvoyer l'exécution du programme à l'instruction **FOR**, qui la précède, et qui est affectée de la même variable qu'elle.

Formulation

NEXT variable

Exemple de programme

```
10 REM *** NEXT ***
20 FOR B=1 TO 4
30 PRINT B;
40 NEXT B
50 END
```

Résultat de l'exécution

Ligne 30 : 1 2 3 4

A noter que...

- B a pris successivement les valeurs 1,2,3 et 4, c'est-à-dire de 1 (valeur initiale) à (**TO**) 4 (valeur finale) de la boucle répétitive B de la ligne 20
- Les instructions comprises entre **FOR** et **NEXT** sont exécutées un nombre de fois égal au nombre suivant **TO**
- La valeur initiale de la boucle est incrémentée de 1 si aucune autre indication n'a été précisée après le mot **STEP** (pas)
- Les valeurs initiale et finale ainsi que l'incrément peuvent être représentés par un nombre, une variable ou une expression
- **NEXT** renvoie l'exécution du programme à l'instruction **FOR** correspondante un nombre de fois égal à la valeur finale de la boucle
- A la fin de la boucle, l'exécution du programme se poursuit à la ligne qui suit l'instruction **NEXT**
- Plusieurs boucles peuvent être utilisées dans un même programme mais toutes doivent s'emboîter l'une dans l'autre
- Certains systèmes permettent l'emploi d'abréviations telles que N. ou NEX

NOT

De l'anglais **NOT**, non.

Objet de l'opérateur

Inverser l'évaluation d'une expression.

Formulation

PRINT NOT (expression)

Exemple de programme

```
10 REM *** NOT ***
20 PRINT (100<50)
30 PRINT NOT (100<50)
40 PRINT (100>50)
50 PRINT NOT (100>50)
60 END
```

Résultat de l'exécution

Une expression logique prend pour valeur —1 lorsqu'elle est vraie et 0 lorsqu'elle est fausse.

Ligne 20 : 0 puisque l'expression (100 plus petit que 50) est fausse
Ligne 30 : —1 donc vraie puisque **NOT** a inversé l'évaluation de l'expression
Ligne 40 : —1 puisque l'expression (100 plus grand que 50) est vraie
Ligne 50 : 0 donc fausse puisque **NOT** a inversé l'évaluation de l'expression

A noter que...

- L'opérateur **NOT** inversant la condition logique de comparaison, on peut dire :
  Une expression est reconnue comme vraie (—1) lorsqu'elle est fausse et que inversement, elle est reconnue comme fausse (0) lorsqu'elle est vraie
- **NOT** peut être utilisé avec **AND** et **OR**
- **NOT** est fréquemment employé avec l'instruction conditionnelle **IF...THEN...**

ON ERROR GOTO

De l'anglais **ON**, sur, **ERROR**, erreur, et **GOTO**, aller à.

Objet de l'instruction

Brancher l'ordinateur, en cas d'erreur, sur une ligne dont le numéro est précisé.

Formulation

ON ERROR GOTO numéro de ligne

Exemple de programme

```
10 REM *** ON ERROR GOTO ***
20 ON ERROR GOTO 60
30 INPUT X
40 P=5/X
50 IF X<>0 THEN PRINT P  GOTO 80
60 PRINT"PAS DE DIVISION PAR ZERO.RECOMMENCEZ"
70 RESUME 30
80 END
```

Exemple de l'exécution

Ligne 20 : Instruction de branchement à la ligne 60 en cas d'erreur
Ligne 30 : Entrée par le clavier d'un nombre à effecter à la variable X
Ligne 50 : Imprimera le résultat de l'opération de la ligne 40 si X est différent de 0. Et branchement à la ligne 80
Ligne 60 : Affiche la chaîne PAS DE DIVISION PAR ZERO. RECOMMENCEZ
Ligne 70 : Instruction ordonnant de reprendre l'exécution à la ligne 30

A noter que...

- Si à la ligne 30, la valeur entrée par **INPUT** était, par exemple, un 0, la division par zéro étant considerée comme une erreur, l'ordinateur se serait branché à la ligne 70 pour imprimer la chaîne puis passant à la ligne 80, aurait repris l'exécution du programme depuis la ligne 30

ON...GOSUB...

De l'anglais **ON**, sur, **GO**, aller à, et **SUB**, sous.

Objet de l'instruction

Transférer l'exécution du programme à l'un des numéros de ligne indiqués après **GOSUB** suivant la valeur de la variable qui suit le mot **ON**

Formulation

ON variable **GOSUB** numéros de ligne

Exemple de programme

```
10 REM *** ON...GOSUB... ***
20 X=1
30 ON X GOSUB 90,110
40 X=2
50 ON X GOSUB 90,110
80 END
90 PRINT"TRANSFERT A LA LIGNE 90"
100 RETURN
110 PRINT"TRANSFERT A LA LIGNE 110"
120 RETURN
```

Résultat de l'exécution

Ligne 90 : TRANSFERT A LA LIGNE 90 (X égalant 1, le transfert s'est fait au premier numéro de ligne indiqué après **GOSUB**

Ligne 110 : TRANSFERT A LA LIGNE 110 (X égalant 2, le transfert s'est fait au deuxième numéro de ligne indiqué après **GOSUB**

A noter que...

- Si X prenait une valeur inférieure à 1 ou supérieur à 2, le transfert se serait fait, par défaut, au premier numéro de ligne
- L'ordinateur ignorera les décimales. Si X = 1.99, le transfert se fera sur l'évaluation de X = 1
- Pour un branchement correct, il faut autant de lignes de transfert que de numéros de ligne après **GOSUB**

ON...GOTO...

De l'anglais **ON**, sur, et **GOTO**, aller à.

Objet de l'instruction

Transférer l'exécution du programme à l'un des numéros de ligne indiqués après **GOTO** suivant la valeur de la variable qui suit le mot **ON**

Formulation

ON variable **GOTO** numéros de ligne

Exemple de programme

```
10 REM *** ON...GOTO... ***
20 X=1
30 ON X GOTO 80,100,120
40 X=2
50 ON X GOTO 80,100,120
60 X=3
70 ON X GOTO 80,100,120
80 PRINT"TRANSFERT A LA LIGNE 80"
90 GOTO 40
100 PRINT"TRANSFERT A LA LIGNE 100"
110 GOTO 60
120 PRINT"TRANSFERT A LA LIGNE 120"
```

Résultat de l'exécution

Ligne 80 : TRANFERT A LA LIGNE 80 (X égalant 1, le tranfert se fait au premier numéro de ligne indiqué après **GOTO**)

Ligne 100 : TRANFERT A LA LIGNE 100 (X égalant 2, le tranfert se fait au deuxième numéro de ligne indiqué après **GOTO**)

Ligne 120 : TRANFERT A LA LIGNE 120 (X égalant 3, le transfert se fait au troisième numéro de ligne indiqué après **GOTO**)

A noter que...

- Si X prenait une valeur inférieure à 1 ou supérieur à 3, le tranfert se serait fait, par défaut, au premier numéro de ligne
- L'ordinateur ignorera les décimales. Si X = 1.76, le transfert se ferait sur l'évaluation de X = 1
- Certains systèmes emploient les abréviations **GOTO...OF**,**ON...G.** ou **ON...GOT**

OR

De l'anglais **OR**, ou.

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, exige que *l'une* de *deux* ou *plusieurs conditions* soit remplie pour l'exécution de l'instruction qui suit **THEN**

Formulation

IF condition **OR** condition **THEN** instruction

Exemple de programme

```
10 REM ♦♦♦ OR ♦♦♦
20 A=10  B=50  C=100
30 IF A<B OR B<C THEN 50
40 PRINT"FAUX" GOTO 100
50 PRINT"L'UNE DES 2 CONDITIONS EST REMPLIE"
60 A$="A"  B$="B"  C$="B"
70 IF A$<B$ OR C$<B$ THEN 90
80 PRINT"FAUX" GOTO 100
90 PRINT"L'UNE DES 2 CONDITIONS EST REMPLIE"
100 END
```

Résultat de l'exécution

Ligne 50 : L'UNE DES 2 CONDITIONS EST REMPLIE (A, 10 est plus petit que B, 50 alors que B, 50 n'est pas plus grand que C, 100)

Ligne 90 : L'UNE DES 2 CONDITIONS EST REMPLIE (A$, A est différent de B$, B alors que C$, B n'est pas différent de B$, B)

A noter que...

- Si les deux conditions, précédant et suivant **OR**, n'avaient pas été remplies, le branchement n'aurait pas été exécuté et l'ordinateur aurait affiché FAUX
- Certains systèmes affichent —1 lorsque l'une de deux ou plusieurs conditions de **OR** est remplie et 0 lorsqu'elle ne l'est pas

PARENTHÈSES ( )

Objet de l'opérateur

Délimiter les arguments des fonctions et les opérations arithmétiques qui seront effectues suivant un ordre de priorité d'exécution.

Formulations

```
1 REM *** PARENTHESES ( ) ***
2 PRINT FRE(0)
3 P=INT(X)
4 DIM D(14,26)
5 PRINT TAB(10);"ABCD"
6 PRINT SPC(7);Y
7 A=ASC("A")
8 C$=CHR$(65)
9 X=(2*3+4)^2+5)*(3^2-4)
```

Observations

Lignes 2 à 8 : arguments de diverses fonctions, placés entre parenthèses

Ligne 9 : Les calculs se feront de gauche à droite :

1) d'abord à l'intérieur des parenthèses, dans l'ordre où l'ordinateur les rencontre par paire
2) et, à l'intérieur de ces parenthèses, dans l'ordre de de priorité des opérateurs arithmétiques

A noter que...

- L'ordre de priorité des opérateurs arithmétiques est la suivante :
  1. Elévation à la puissance ↑
  2. Négation −
  3. Multiplicateur et Division * et /
  4. Addition et soustraction + et −

  Puis les opérateurs de comparaison :

  5. NOT
  6. AND
  7. OR

PLUS GRAND QUE >

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, permet d'effectuer une comparaison entre deux valeurs numériques ou deux expressions arithmétiques logiques.

Formulation

IF variable **>** variable **THEN** instruction

Exemple de programme

```
10 REM *** PLUS GRAND QUE (>) ***
20 A=10  B=50
30 IF B>A THEN 50
40 PRINT"FAUX"  GOTO 100
50 PRINT"B EST > QUE A"
60 A$="A"  B$="B"
70 IF B$>A$ THEN 90
80 PRINT"FAUX"  GOTO 100
90 PRINT"B$ EST > QUE A$"
100 END
```

Résultat de l'exécution

Ligne 50 : B EST > QUE A
Ligne 90 : B$ EST > QUE A$

A noter que...

- Pour comparer deux lettres ou deux chaînes, l'ordinateur se réfère au code ASCII de ces lettres ou de ces chaînes. Le code de A étant 65 et celui de B, 66, il est donc logique de dire que B est plus grand que A
- Pour une comparaison entre deux chaînes, ABCD et AGCD, la comparaison se fait, à partir de la gauche, lettre par lettre. A est égal à A, B est plus petit que G (B = 66 et G = 71), donc, la chaîne AGCD est plus grande que la chaîne ABCD
- Certains systèmes emploient l'abréviation **GT** au lieu de >

PLUS PETIT QUE <

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, permet d'effectuer une comparaison entre deux valeurs numériques ou deux expressions arithmétiques logiques.

Formulation

IF variable **<** variable **THEN** instruction

Exemple de programme

```
10 REM *** PLUS PETIT QUE (<) ***
20 A=10  B=50
30 IF A<B THEN 50
40 PRINT"FAUX"  GOTO 100
50 PRINT"A EST < QUE B"
60 A$="A"  B$="B"
70 IF A$<B$ THEN 90
80 PRINT"FAUX"  GOTO 100
90 PRINT"A$ EST < QUE B$"
100 END
```

Résultat de l'exécution

Ligne 50 : A EST < QUE B
Ligne 90 : A$ EST < QUE B$

A noter que...

- Pour comparer deux lettres ou deux chaînes, l'ordinateur se réfère au Code ASCII de ces lettres ou de ces chaînes. Le code de A étant 65 et celui de B, 66, il est donc logique de dire que A est plus petit que B
- Pour une comparaison entre deux chaînes, ABCD et AGCD, la comparaison se fait, à partir de la gauche, lettre par lettre. A est égal à A, B est plus petit que G (B = 66 et G = 71), donc, la chaîne ABCD est plus petite que la chaîne AGCD
- Certains systèmes emploient l'abréviation **LT** au lieu de **<**

POINT D'EXCLAMATION !

Objet du caractère

Définir une ou plusieurs variables comme étant du type simple précision.

Formulation

Variable !

Exemple de programme

```
10 REM *** POINT D'EXCLAMATION (!) ***
20 A=4
30 B=3
40 C!=A/B
50 PRINT C!
60 C#=A/B
70 PRINT C#
80 END
```

Résultat de l'exécution

Ligne 50 : 1.333333 (résultat en simple précision de l'opération de la ligne 40)

Ligne 70 : 1.333333373069763 (résultat en double précision de l'opération de la ligne 60)

A noter que...

- En simple précision, le résultat contient 7 chiffres en mémoire dont seuls 6 significatifs sont affichés
- Le nombre des chiffres significatifs affiché varie avec les systèmes. Certains affichent 9 chiffres significatifs en simple précision
- L'utilisation inconsidérée de variables du type double précision réduit considérablement la capacité mémoire disponible

POINT-VIRGULE ;

Objet de l'opérateur

Séparer divers éléments numériques ou alphanumériques.

Formulations

```
1 REM ♦♦♦ POINT-VIRGULE ... ♦♦♦
2 PRINT 1;2;3;4
3 PRINT -1;-2;-3;-4
4 PRINT A$;B$;C$;D$
5 PRINT "A";"B";"C";"D"
6 INPUT"ENTREZ UN NOMBRE";A
7 INPUT"ENTREZ UN NOM";A$
8 PRINT X;"KILOS =";Y;"GRAMMES"
```

Observations

Ligne 2 : Impression des chiffres 1,2,3,4, précédés et suivis d'un espace. Le premier espace est réservé automatiquement pour les signes + ou —. Si le chiffre est positif, le signe + n'est pas imprimé

Ligne 3 : Impression des chiffres —1,—2,—3,—4, suivis d'un espace. Un nombre negatif est toujours précédé de son signe.

Ligne 4 : Impression des 4 chaînes A$,B$,C$,D$ liées les unes aux autres

Ligne 5 : Impression des 4 lettres ABCD sans espace entre elles. Dans le cas où l'on voudrait un espace, avant ou après une chaîne quelconque, cet espace devrait être incorporé dans la chaîne et la chaîne placée entre guillemets

Ligne 6 : Un point virgule doit séparer la chaîne ENTREZ UN NOMBRE de la variable A

Ligne 7 : Même observation que pour la ligne 6

Ligne 8 : Sépare les éléments numériques des éléments alphanumériques

A noter que...

- Dans une instruction **PRINT**, certains systèmes n'exigent pas le point-virgule comme operateur de séparation entre une chaîne et une variable

POURCENTAGE %

Objet de l'opérateur

Définir une variable comme ne pouvant contenir que le nombre entier d'une valeur.

Formulation

Variable % = expression

Exemple de programme

```
10 REM *** POURCENTAGE (%) ***
20 A=12.3456
30 A%=A
40 B=A*3
50 B1=A%*3
60 C=INT(A*3)
70 PRINT A
80 PRINT A%
90 PRINT B
100 PRINT B1
110 PRINT C
```

Résultat de l'exécution

Ligne 70 : 12.3456 (valeur affectée à la variable A)
Ligne 80 : 12 (A% ne contient que la partie entière de A)
Ligne 90 : 37.0368 (résultat de l'opération de la ligne 40)
Ligne 100 : 36 (A% ne contenant que 12, 3 fois 12 font bien 36)
Ligne 110 : 37 (nombre entier du résultat de l'opération avec la valeur de A = 12.3456)

A noter que...

- La formulation C = INT(A * 3) aurait pu être simplifiée par C% = A * 3, le résultat, dans les deux cas, étant 37
- Certains systèmes utilisent l'opérateur % avec l'instruction **PRINT USING** pour l'impression d'un nombre de caractères égal au nombre d'espaces contenu entre les deux opérateurs %.....%

PRINT

De l'anglais **PRINT**, imprimer.

Objet de l'instruction

Imprimer des valeurs numériques et/ou des chaînes alphanumériques.

Formulation

PRINT variable et/ou chaîne

Exemple de programme

```
10 REM *** PRINT ***
20 X=100  C$="CHAINE"
30 PRINT X,X
40 PRINT X;X
50 PRINT C$,C$
60 PRINT C$;C$
70 END
```

Résultat de l'exécution

Ligne 30 : 100 100
Ligne 40 : 100 100
Ligne 50 : CHAINE CHAINE
Ligne 60 : CHAINECHAINE

A noter que...

- **PRINT** suivie d'une virgule imprime suivant une tabulation prédéfinie (exécution de la ligne 30)
- **PRINT** suivie d'un point-virgule imprime les nombres en insérant un espace avant et après chaque nombre (ligne 40)
- Pour l'impression d'une chaîne, **PRINT** suivie d'une virgule n'insère pas d'espace avant chaque chaîne (ligne 50)
- Deux chaînes séparées par un point-virgule sont soudées l'une à l'autre
- L'espace qui précède les nombres est prévu pour l'impression éventuelle du signe moins (—)
- Certains systèmes permettent l'emploi d'abréviations telles que **P.** ou **PRI** au lieu de **PRINT**

PRINT USING

De l'anglais **PRINT**, imprimer, et **USING**, utilisant.

Objet de l'instruction

Préciser la disposition graphique d'une impression.

Formulation

PRINT USING chaîne

Exemple de programme

```
10 REM *** PRINT USING ***
20 FOR I=1 TO 5
30 READ A
40 PRINT USING "####.##";A
50 NEXT I
60 DATA 1.23,11.1,1111.2345,.23
70 END
```

Résultat de l'exécution

Ligne 40 :
```
    1.23
   11.10
 1111.23
    0.23
```

A noter que...

- La disposition graphique précisée à la ligne 40 "`####.##`", indiquait le nombre de chiffres de la partie entière, la place du point décimal et le nombre de décimales, ce qui a permis d'aligner tous les nombres lus (**READ** A) dans les données (**DATA**) et de les disposer suivant le dessin de la chaîne "`#### .##`"
- Le premier nombre a été reproduit tel quel. Un 0 a été rajouté au deuxième. Deux décimales ont été supprimées dans le troisième. Un 0 a été rajouté au quatrième, devant le point décimal
- Les divers spécificateurs de zone utilisés dans les chaînes de **PRINT USING** sont : le dièse # , l'astérisque * , le dollar $, le pourcentage % , le point d'exclamation ! , le signe plus + et le signe moins —

PUISSANCE ↑

Objet de l'opérateur

Opérer un calcul exponentiel.

Formulation

Variable = expression ↑ expression

Exemple de programme

```
10 REM *** PUISSANCE (↑) ***
20 A=2↑2
30 B=(A+2)↑3
40 C=(A+B)↑2
50 PRINT A
60 PRINT B
70 PRINT C
80 END
```

Résultat de l'exécution

Ligne 20 : 2 élevé à la puissance 2 et résultat affecté à la variable A
Ligne 30 : Somme de A + 2, soit 6 ; élévation de 6 à la puissance 3 et résultat affecté à la variable B
Ligne 40 : Somme de A + B, soit 220 ; élévation de 220 à la puissance 2 et affectation du résultat à la variable C
Lignes 50 à 70 : Affichage des valeurs de A, B, C qui sont respectivement 4, 216 et 48400

A noter que...

- Certains systèmes utilisent des symboles différents tels que * *, ^ ou ∩
- On peut utiliser cet opérateur pour calculer la racine d'un nombre en incluant entre parenthèses l'inverse de l'exposant. Exemple : 16 ↑ (1/2) équivaut à la racine carrée de 16. Résultat : 4

RANDOMIZE

De l'anglais **RANDOMIZE**, donner au hasard.

Objet de l'instruction

Provoquer une séquence différente à chaque exécution de **RND**

Formulation

RANDOMIZE

Exemple de programme

```
10 REM *** RANDOMIZE ***
20 FOR I=1 TO 4
30 PRINT RND(0)
40 NEXT I
50 END
```

Résultat de l'exécution

	1re séquence	2e séquence
Ligne 30 :	.572612	.205576
	.496633	.416608
	.256576	.478662
	.470038	.216956

A noter que...

- L'instruction **RANDOMIZE** ne prend pas d'argument
- Elle doit se placer avant la fonction **RND**
- Certains systèmes exigent que **RND** soit suivie d'un argument fictif entre parenthèses
- L'argument, s'il est exigé, peut être représenté par un nombre, une variable ou une expression
- Certains systèmes emploient le mot équivalent de **RANDOM** ou une abréviation telle que **RAN**

READ

De l'anglais **READ**, lire.

Objet de l'instruction

Lire les données contenues dans la ligne **DATA** et les affecter, dans l'ordre de leur lecture, aux variables correspondantes.

Formulation

READ variable

Exemple de programme

```
10 REM ♦♦♦ READ ♦♦♦
20 FOR I=1 TO 3
30 READ X
40 PRINT X
50 NEXT I
60 DATA 10,20,30
70 FOR I=1 TO 3
80 READ X$
90 PRINT X$
100 NEXT I
110 DATA AA,BB,CC
```

Résultat de l'exécution

Ligne 40 : 10 (première donnée affectée à X lors de la première lecture)
20 (deuxième donnée affectée à X lors de la deuxième lecture)
30 (troisième donnée affectée à X lors de la troisième lecture)

Ligne 90 : AA (première chaîne affectée à X$ lors de la première lecture)
BB (deuxième chaîne affectée à X$ lors de la deuxième lecture)
CC (troisième chaîne affectée à X$ lors de la troisième lecture)

A noter que...

- De nombreux systèmes permettent que **READ** soit suivi d'une combinaison de variables numériques et de variables de chaînes (X,X$,Y,Z,Z$)
- Certains systèmes permettent l'emploi d'abréviations telles que **REA** ou **REA.** au lieu de **READ**

REM

De l'anglai **REMARK**, commentaire.

Objet de l'instruction

Insérer des commentaires.

Formulation

REM chaîne

Exemple de programme

```
10 REM *** REM ***
20 REM CALCUL DE LA
30 REM CIRCONFERENCE
40 REM D'UN CERCLE(C)
50 REM RAYON (R)
60 REM π=3.14159265
70 INPUT R
80 C=π*R*2 PRINT C
90 END
```

Résultat de l'exécution

Ligne 70 : Entrée de la valeur à affecter à la variable R
Ligne 80 : Calcul de la circonférence d'un cercle et affichage du résultat. Si la valeur entrée était 2, le résultat donnerait 12.5663706

A noter que...

- Les lignes 20 à 60 expliquent l'objet du programme et apportent des précisions utiles : calcul de la circonférence d'un cercle ; variables utilisées : C pour cercle, R pour rayon ; définition de la valeur de Pi
- **REM** est une instruction qui n'est pas exécutée. Il faut éviter de la faire suivre d'une équation quelconque car celle-ci serait ignorée
- Certains systèmes emploient une apostrophe (') à la place de **REM** D'autres emploient le mot entier **REMARK**

RESTORE

De l'anglais **RESTORE**, rétablir.

Objet de l'instruction

Permettre la réutilisation des mêmes données par la même instruction **READ**

Formulation

RESTORE

Exemple de programme

```
10 REM *** RESTORE ***
20 FOR B=1 TO 3
30 RESTORE
40 FOR D=1 TO 3
50 READ X
60 PRINT X;
70 NEXT D
80 PRINT
90 NEXT B
100 DATA 10,20,30
```

Résultat de l'exécution

```
Ligne 60 : 10   20   30
           10   20   30
           10   20   30
```

A noter que...

- Les données de la ligne 100 ont été imprimées trois fois en raison de la boucle B de la ligne 20
- A chaque exécution, l'instruction **RESTORE** de la ligne 30 "repositionnait" le pointeur en début de la ligne **DATA** pour une nouvelle lecture
- Le **PRINT** de la ligne 80 équivaut à un "à la ligne"
- Certains systèmes permettent l'emploi d'abréviations telles que **RES** ou **REST.** au lieu de **RESTORE**

RETURN

De l'anglais **RETURN**, retour.

Objet de l'instruction

Faire reprendre l'exécution du programme à l'instruction qui suit le **GOSUB** correspondant.

Formulation

RETURN

Exemple de programme

```
10 REM *** RETURN ***
20 GOSUB 50
30 PRINT "RETOUR A LA LIGNE 30"
40 END
50 PRINT "SUIS PASSE A LA LIGNE 50"
60 RETURN
70 END
```

Résultat de l'exécution

Ligne 20 : Transfert à la ligne 50
Ligne 50 : Affichage de la chaîne SUIS PASSE A LA LIGNE 50
Ligne 60 : Retour à la ligne 30
Ligne 30 : Affichage de la chaîne RETOUR A LA LIGNE 30
Ligne 40 : Fin du programme

A noter que...

- L'instruction **RETURN** est obligatoirement placée à la fin de la sous-routine appelée par **GOSUB** pour la reprise de l'exécution séquentielle du programme. Un **GOSUB** sans **RETURN** couplé provoque un message d'erreur
- **RETURN** est également une touche de fonction que l'on frappe après la fin de chaque ligne d'instruction entrée au clavier
- Certains systèmes permettent l'emploi d'abréviations telles que **RET** ou **RET.** au lieu de **RETURN**

RIGHT$

De l'anglais **RIGHT**, droite.

Objet de la fonction

Extraire un certain nombre de caracteres d'une chaîne alphanumérique en partant de l'extrême-droite.

Formulation

RIGHT$ (variable, nombre)

Exemple de programme

```
10 REM *** RIGHT$ ***
20 A$="INFORMATIQUE"
30 FOR B= 1 TO 3
40 PRINT RIGHT$(A$,B)
50 NEXT B
60 END
```

Résultat de l'exécution

Ligne 40 : E pour B = 1
UE pour B = 2
QUE pour B = 3

A noter que...

- La chaîne doit être incluse entre guillemets : "INFORMATIQUE"
- Le nombre de caractères à extraire peut être désigné par un nombre, une variable ou une expression (A$, X ↑ 2 + 1)
- Une virgule doit séparer la chaîne du nombre, de la variable ou de l'expression
- Si le nombre, la variable ou le résultat de l'expression comporte des décimales (4,56), l'ordinateur n'en retiendra que le nombre entier (4)
- L'argument doit être inclus entre parenthèses ("INFORMATIQUE",5)
- Les variables alphanumériques sont toujours suivies du signe distinctif **$**
- Certains systèmes utilisent le mot **RIGHT** sans le signe distinctif **$**

RND

De l'anglais **RANDOM**, hasard.

Objet de la fonction

Fournir un nombre au hasard.

Formulation

RND (argument)

Exemple de programme

```
10 REM *** RND ***
20 REM * ENTRE 0 ET 1
30 A=RND(0)
40 REM * ENTRE 1 ET 6
50 B=INT(6*RND(1))+1
60 REM * ENTRE 6 ET 12
70 C=6+INT((12-6)*RND(1))
80 PRINT A
90 PRINT B
100 PRINT C
```

Résultat de l'exécution

Ligne 80 : .580544747 (entre 0 et 1)
Ligne 90 : 4 (entre 1 et 6)
Ligne 100 : 9 (entre 6 et 12)

A noter que...

- Certains systèmes n'exigent pas d'argument après **RND**
- Si l'argument est exigé, il doit être inclus entre parenthèses
- L'argument peut être représenté par un nombre, une variable ou une expression pouvant comporter des nombres, des variables et des opérateurs arithmétiques. Exemple : C = X + INT ((Y — X)* RND(Z))
- Pour obtenir des nombres entiers, on emploie la fonction **INT**
- Certains systèmes permettent l'emploi de l'abréviation équivalente **R.**

RUN

De l'anglais, **RUN**, exécuter.

Objet de la commande

Exécuter le programme résident en mémoire.

Formulation

RUN

Exemple de programme

```
10 REM *** RUN ***
20 C=100  PRINT C
30 END
40 C=200
50 PRINT C
60 IF C=200 THEN RUN 20
70 END
```

Résultat de l'exécution

Si l'on tape **RUN** en mode direct, l'exécution se fera du début du programme :

Ligne 20 : Affichage du nombre 100, valeur affectée à la variable C
Ligne 30 : Fin du programme

Si l'on tape **RUN** et le numéro de ligne 40, l'exécution débutera à la ligne indiquée :

Ligne 40 : Affectation de la valeur 200 à la variable C
Ligne 50 : Affichage de cette valeur, soit 200
Ligne 60 : La condition étant satisfaite (C = 200), l'exécution reprendra à la ligne 20. Les nombres affichés seront successivement : 200 et 100 et le programme s'arrêtera à la ligne 30

A noter que...

- **RUN** a été incluse comme instruction à la ligne 60
- Si après **RUN**, on indique un numéro de ligne non existant, l'ordinateur affiche un message d'erreur
- Certains systèmes permettent l'emploi d'abréviations telles que **R.** ou **RU** au lieu de **RUN**

SAVE

De l'anglais **SAVE**, sauver.

Objet de la commande

Enregistrer sur une cassette le programme résidant en mémoire.

Formulation

SAVE nom du programme

Exemple de programme

```
10 REM *** SAVE ***
20 C=200
30 PRINT C
40 IF C=200 THEN SAVE "PN1"
50 END
```

Résultat de l'exécution

Ligne 30 : Affiche la valeur de C, soit 200
Ligne 40 : La condition étant satisfaite (C = 200), l'instruction ordonne d'enregistrer sur cassette le programme intitulé PN1

A noter que...

- **SAVE** a été employée comme instruction à la ligne 40
- Si aucun nom n'est donné au programme à enregistrer, celui-ci le sera mais sans aucun en-tête pour une référence ultérieure
- Certains systèmes sont dotés de deux cassettes. Sans aucune précision, l'enregistrement se fera sur la cassette n° 1
- Certains systèmes emploient le mot de **CSAVE** au lieu de **SAVE**
- La commande **SAVE** s'accompagne de l'emploi simultané des touches **RECORD** et **PLAY** qui déclenchent l'enregistrement

SOUSTRACTION —

Objet de l'opérateur

Opérer une soustraction entre deux éléments.

Formulation

Variable = expression — expression

Exemple de programme

```
10 REM *** SOUSTRACTION (-) ***
20 X=12-2 :PRINT X
30 Y=X-2 :PRINT Y
40 A=X-Y
50 B=(X↑3)-(X↑2)
60 C=(X*Y)-(X*Y/2)
70 PRINT A
80 PRINT B
90 PRINT C
100 END
```

Résultat de l'exécution

Ligne 20 : Affectation à la variable X du résultat de la soustraction 12—2 et affichage du résultat : 10
Ligne 30 : Affectation à la variable Y du résultat de la soustraction X—2 et affichage du resultat : 8
Ligne 70 : Affichage de la valeur de A : 2
Ligne 80 : Affichage de la valeur de B : 900
Ligne 90 : Affichage de la valeur de C : 40

A noter que...

- Le signe — est également utilisé comme signe de négation dans une opération arithmétique. Exemple :
  40 A = Y—X (en prenant les valeurs de l'exemple ci-dessus)
  50 N = —(Y—X)
  Les résultats seront : A = —2 N = 2
  A la ligne 50, le signe de négation a inversé le signe — à l'intérieur des parenthèses en signe +

SPC

De l'anglais **SPACE, espace.**

Objet de l'instruction

Insérer entre deux impressions le nombre d'espaces indiqué dans l'argument .

Formulation

PRINT SPC (nombre)

Exemple de programme

```
10 REM *** SPC ***
20 FOR X=1 TO 6
30 PRINTSPC(X)"ABCDE"
40 NEXT X
50 PRINT SPC(6);"6 ESPACES"
60 END
```

Résultat de l'exécution

```
ABCDE
 ABCDE
  ABCDE
   ABCDE
    ABCDE
     ABCDE
      6 ESPACES
```

A noter que...

- La différence entre les instructions **TAB** et **SPC** est que le premier tabule à partir du début de la ligne tandis que le second insère des espaces entre les impressions
- Certains systèmes permettent l'emploi de l'abréviation **SPA** ou les mots équivalents de **SPACE** ou **SPACES**

STEP

De l'anglais **STEP**, pas.

Objet de la fonction

Préciser la valeur du pas d'incrémentation dans l'instruction **FOR...NEXT**

Formulation

STEP nombre

Exemple de programme

```
10 REM ♦♦♦ STEP ♦♦♦
20 FOR B=0 TO 10 STEP 2
30 PRINT B;
40 NEXT B
50 PRINT
60 FOR X=10 TO 0 STEP -2
70 PRINT X;
80 NEXT X
90 END
```

Résultat de l'exécution

Ligne 30 : 0 2 4 6 8 10 (Boucle B par pas de 2)
Ligne 70 : 10 8 6 4 2 0 (Boucle X par pas de —2)

A noter que...

- Si la fonction **STEP** est omise dans l'instruction **FOR...NEXT**, le pas est + 1 par défaut
- La valeur du pas peut être positive, négative et comporter des décimales
- La valeur de **STEP** peut être représentée par un nombre, une variable ou une expression
- Une boucle est toujours exécutée au moins une fois. Au lieu de FOR X = 1 TO 1, essayez FOR X = 4 TO 0
- Certains systèmes permettent l'emploi d'abréviations telles que **S.**, **ST** ou **STE** au lieu de **STEP**

STOP

De l'anglais **STOP**, arrêt.

Objet de l'instruction

Arrêter l'exécution du programme à la ligne choisie.

Formulation

STOP

Exemple de programme

```
10 REM *** STOP ***
20 A$="ARRET A LA LIGNE 40"
30 PRINT A$
40 STOP
50 B$="CONTINUATION A LA LIGNE 50"
60 PRINT B$
```

Résultat de l'exécution

Ligne 30 : ARRET A LA LIGNE 40
Ligne 40 : (Arrêt du programme. Sur l'écran est affichée la phrase BREAK IN 40, qui signifie "Interruption à la ligne 40")
Ligne 60 : Cette ligne ne sera exécutée que si l'on tape la commande directe **CONT** après l'arrêt du programme

A noter que...

- Alors que certains systèmes arrêtent l'exécution à la ligne comportant l'instruction **STOP**, d'autres, en lisant cette même instruction, se brancheront directement à la ligne qui comporte le mot **END**
- De nombreux systèmes permettent d'utiliser **STOP** plusieurs fois dans un même programme
- La différence majeure entre les instructions **STOP** et **END**, c'est que la première provoque un arrêt momentané du programme tandis que la seconde indique la fin du programme ou d'un sous-programme
- Certains systèmes permettent l'emploi d'abréviations telles que **S.**, **ST.** ou **STO** au lieu de **STOP**

STR$

De l'anglais **STRING**, chaîne.

Objet de la fonction

Convertir une valeur numérique en une chaîne alphanumérique.

Formulation

STR$ (variable numérique)

Exemple de programme

```
10 REM *** STR$ ***
20 A=123456789
30 A$=STR$(A)
40 PRINT A$
50 PRINT LEFT$(A$,4)
60 PRINT RIGHT$(A$,5)
70 END
```

Résultat de l'exécution

Ligne 40 : 123456789 (chaîne contenue dans la variable A$)
Ligne 50 : 123 (extraction de 4 caractères de la chaîne A$ en partant de l'extrême-gauche, le 1er étant un espace)
Ligne 60 : 56789 (extraction de 5 caractères de la chaîne A$ en partant de l'extrême-droite)

A noter que...

- L'argument doit être inclus entre parenthèses
- La conversion d'une valeur numérique en une chaîne par l'emploi de la fonction **STR$** permet la manipulation de cette chaîne par des fonctions telles que **MID$**, **LEFT$**, **RIGHT$**, **LEN**, etc.
- L'argument peut être un nombre, une variable ou une expression pouvant comporter des nombres, des variables et des opérateurs arithmétiques (((A + B) / B) ↑ 3)

SUPÉRIEUR OU ÉGAL A >=

Objet de l'opérateur

Employé avec l'instruction conditionnelle **IF...THEN...**, permet d'effectuer une comparaison entre deux valeurs numériques ou deux expressions arithmétiques logiques.

Formulation

IF variable **>** = variable **THEN** instruction

Exemple de programme

```
10 REM *** SUPERIEUR OU EGAL A (>=) ***
20 A=10  B=100  C=10
30 IF B>=A THEN 50
40 PRINT"FAUX"  GOTO 90
50 PRINT"B>A"
60 IF A=C THEN 80
70 PRINT"FAUX"  GOTO 90
80 PRINT"A=C"
90 END
```

Résultat de l'exécution

Ligne 50 : B > A (100, en effet, est plus grand que 10)
Ligne 80 : A = C (10, en effet, est égal à 10)

A noter que...

- L'opérateur **>** = est également employé pour comparer des chaînes. Pour cette comparaison, l'ordinateur se réfère au code ASCII. Le code de B étant 66 et celui de A, 65, la lettre B est évaluée comme étant plus grande que A
- La comparaison entre deux chaînes de longueur inégale, ABG et ABCDEF, se fait, à partir de la gauche, lettre par lettre. A est égal à A, B est égal à B, mais G étant plus grand que C (G = 71, C = 67), la chaîne ABG est évaluée comme étant plus grande que ABCDEF, bien que comptant 3 lettres de moins
- Certains systèmes emploient les symboles **=>**, **≥** ou l'abréviation **GE** au lieu de **>=**

SWAP

De l'anglais **SWAP**, échanger.

Objet de l'instruction

Echanger les valeurs de deux variables de même type.

Formulation

SWAP variable, variable

Exemple de programme

```
10 REM *** SWAP ***
20 READ X,Y
30 PRINT X,Y
40 IF Y>X THEN SWAP X,Y
50 PRINT X,Y
60 END
70 DATA 4,7
```

Résultat de l'exécution

Ligne 20 : Lit les données à affecter respectivement aux variables X et Y
Ligne 30 : Affichage des valeurs des deux variables :X = 4 Y = 7
Ligne 40 : La condition étant satisfaite (Y = 7 est plus grand que X = 4), échange des valeurs respectives des variables X et Y
Ligne 50 : Affichage des nouvelles valeurs après échange : X = 7 Y = 4

A noter que...

- La très grande utilité de cette instruction pour les opérations de tri
- L'échange ne peut se faire qu'entre variables de même type

VAL

De l'anglais **VALUE**, valeur

Objet de la fonction

Convertir une chaîne alphanumérique en une valeur numérique.

Formulation

VAL (variable de chaîne)

Exemple de programme

```
10 REM *** VAL ***
20 A$="123456"
30 A=VAL(A$)
40 PRINT A
50 B$="25"
60 B=VAL(B$)
70 PRINT B
80 C=A/B  PRINT C
90 END
```

Résultat de l'exécution

Ligne 40 : 123456 (nombre contenu dans la variable numérique A)
Ligne 70 : 25 (nombre contenu dans la variable numérique B)
Ligne 80 : 4938.24 (résultat de l'opération en ligne 80)

A noter que...

- La fonction **VAL** est l'opposée de la fonction **STR$**
- La conversion d'une chaîne en valeur numérique permet d'effectuer des opérations arithmétiques à l'aide de ces valeurs
- L'argument doit être inclus entre parenthèses. Si l'argument est une chaîne, il doit être inclus entre guillemets à l'intérieur des parenthèses ("123456")
- Si le premier caractère d'une chaîne utilisée comme argument est une lettre ou un caractère graphique, le résultat affiché sera un zéro
- Si le premier caractère de la chaîne est l'opérateur + , un blanc précèdera le nombre (123) ; si c'est l'opérateur — , le nombre sera précédé du signe (—123)

VERIFY

De l'anglais **VERIFY**, vérifier.

Objet de la commande

Vérifier si le programme enregistré est la copie conforme du programme résident en mémoire.

Formulation

VERIFY nom du programme

Exemple de programme

```
10 REM ♦♦♦ VERIFY ♦♦♦
20 C=200
30 PRINT C
40 IF C=200 THEN VERIFY "PN1"
50 END
```

Résultat de l'exécution

Ligne 30 : Affiche la valeur de C, soit 200
Ligne 40 : La condition étant satisfaite (C = 200), l'instruction ordonne de vérifier si le programme enregistré sous le label « PN1 » est la copie conforme du programme résident en mémoire

A noter que...

- **VERIFY** a été employée comme instruction à la ligne 40
- Pendant l'exécution de la commande, le programme enregistré et celui qui est en mémoire sont comparés octet par octet (byte)
- Certains systèmes emploient le mot **CLOAD ?** au lieu de **VERIFY**
- A l'encontre de **SAVE**, **VERIFY** n'efface pas le programme résidant en mémoire. Si l'enregistrement a été mal réalisé, on pourra ainsi recommencer l'opération

WHILE...WEND

De l'anglais **WHILE**, tant que.

Objet de l'instruction

Répéter l'exécution d'une instruction tant que la condition précisée est satisfaite.

Formulation

WHILE variable **WEND**

Exemple de programme

```
10 REM *** WHILE ***
20 S=1
30 WHILE S
40 INPUT Q$
50 IF Q$="FIN" THEN S=0
60 WEND
70 PRINT "PROGRAMME TERMINE"
80 END
```

Résultat de l'exécution

Ligne 20 : Affectation de la valeur 1 à la variable S
Ligne 30 : Instruction de début de boucle suivie de la condition : Tant que S vaudra 1, la boucle se répétera
Ligne 40 : Entrée d'une donnée à affecter à la variable Q$
Ligne 50 : Instruction conditionnelle :
Si la donnée entrée à la ligne 40 est FIN, la valeur de S devient 0
La condition de **WHILE** n'étant plus satisfaite, l'exécution se branche à la ligne 70
Ligne 60 : WEND (**WHILE END**) limite extérieure de la boucle **WHILE**

A noter que...

- A l'instar de **NEXT**, **WEND** retourne au **WHILE** correspondant.

Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE

https://acpc.me/