

Martine Trabaud

**initiation
au
langage**

BAWA

 FOUCHER

Martine Traud

initiation

au

langage

BASIC

Sommaire

Préambule	
L'ordinateur, organisation, fonctionnement	3
La programmation. Les langages	5
1 Présentation du BASIC. Règles générales : conventions d'écriture, les variables, les constantes, les opérateurs	7
2 Instructions d'entrée-sortie	13
3 Opérations arithmétiques. Exemples d'écriture d'expressions	19
Instructions d'affectation	20
4 Les branchements	23
Les branchements inconditionnels	25
Répétitions de calcul : lecture directe de données	28
Branchements conditionnels	31
Contrôle de boucle	34
5 Les fonctions	41
Fonctions de calcul	41
Fonctions définies par l'opérateur	45
6 Traitement des chaînes de caractères	51
Instructions de conversion	54
Opérations sur les chaînes de caractères	57
7 Listes et tables	61
8 Séquences d'exécution	69
Fin de programme	69
Commentaires	70
9 Les fichiers. Utilisation des fichiers en BASIC	75
ANNEXE 1 : Mots réservés BASIC	79
ANNEXE 2 : Code ASCII	80

BIBLIOGRAPHIE

LE LANGAGE BASIC, B. Drieux et A.-L. Liju, *Themis*.

INFORMATIQUE APPLIQUÉE A LA COMPTABILITÉ ET A LA GESTION,
R. Reix, *Foucher*.

COMPRENDRE LES MICROPROCESSEURS, D. Queyssac, *Radio*.

préambule

Ce manuel est un outil de travail destiné à des lecteurs ayant déjà des connaissances en informatique.

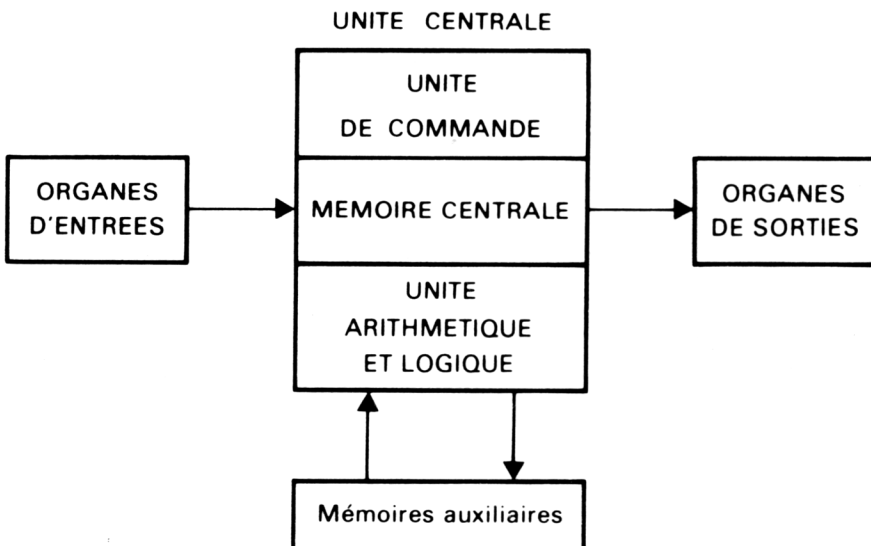
Toutefois, il paraît nécessaire de rappeler quelques notions essentielles.

1. L'ORDINATEUR

1. Définition

Un ordinateur est un ensemble d'organes liés physiquement à un organe central de mémoire : **la mémoire centrale**, et dépendant d'un centre de décision, **le programme**, qui est écrit par le programmeur.

2. Organisation générale d'un ordinateur



- Les organes d'entrée reçoivent des informations enregistrées préalablement sur des supports divers tels que carte perforée, bande magnétique, disque magnétique, ou saisies directement à partir d'un clavier.
- Les organes de sortie permettent à l'ordinateur de fournir des résultats sur des supports externes tels que listing de l'imprimante, bande magnétique, disque magnétique, ou sur des écrans de visualisation.
- La mémoire centrale étant de capacité limitée, on a recours à des mémoires auxiliaires qui permettent de stocker des informations de façon « externe ». Ce sont, en général, des mémoires à support magnétique.
- La mémoire centrale permet d'enregistrer les informations provenant des organes d'entrée (ces informations peuvent être des données ou des ordres), de les conserver, et de restituer les données ou les résultats de calcul, sans aucune altération.
- L'unité arithmétique et logique est composée de circuits dans lesquels est traitée l'information. Ces circuits permettent :
 - les calculs arithmétiques, dont l'addition est l'opération de base ;
 - les calculs logiques, qui sont des comparaisons telles que : égalité, différence, plus petit que, etc...
- L'unité de commande, également composée de circuits, reçoit le programme de la mémoire centrale, instruction par instruction, assure la mise en fonctionnement des organes concernés par le programme, et permet les déplacements de l'information.

3. Fonctionnement d'un ordinateur

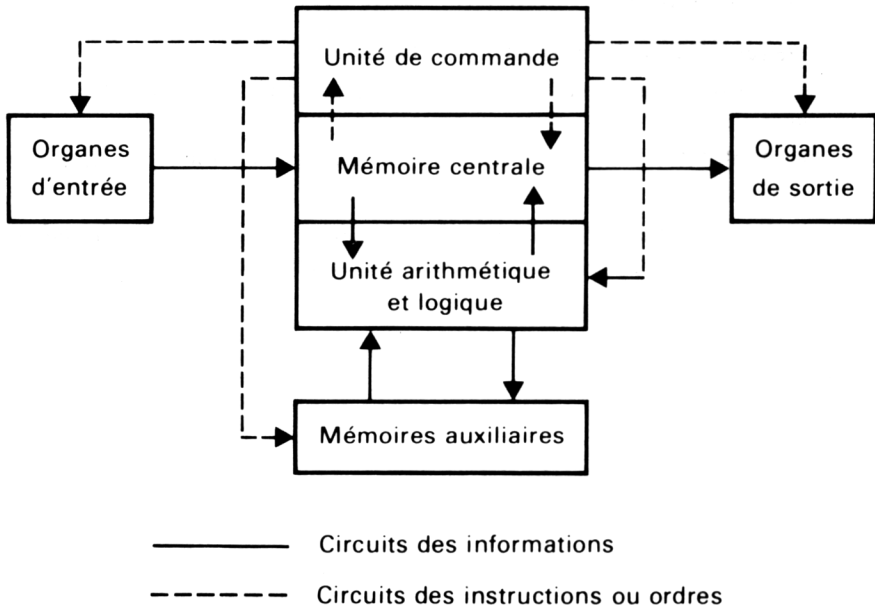
LE PROGRAMME

Un programme est une suite ordonnée d'instructions (ou ordres) fournies à la machine par l'utilisateur afin d'obtenir un résultat déterminé à partir de données connues.

Pour écrire un programme, il faut :

- bien connaître le problème,
- savoir le décomposer logiquement en opérations élémentaires,
- connaître un langage assimilable par l'ordinateur.

SCHÉMA DE PRINCIPE DU FONCTIONNEMENT DE L'ORDINATEUR



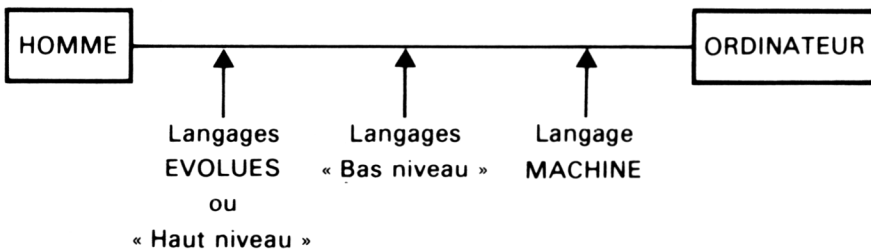
2. LA PROGRAMMATION

La programmation est l'ensemble des tâches qui permettent d'élaborer un programme.

Ce programme doit être enregistré dans la mémoire centrale de l'ordinateur, ce qui suppose son écriture dans un langage assimilable par la machine.

• Les langages

Il existe différents langages de programmation que l'on peut classer comme le présente le schéma suivant :



Le langage machine est principalement le langage BINAIRE.

Parmi les langages « bas niveau », on trouve les langages mnémoniques, les symboliques, et le langage « assembleur ».

Les langages évolués comprennent, entre autres :

- le COBOL, plus particulièrement utilisé pour les applications de gestion ;
- le FORTRAN et l'ALGOL, pour les applications du domaine scientifique ;
- le BASIC, dont le sigle signifie : *code symbolique d'instruction pour toutes applications et pour débutants.*

introduction

1. PRÉSENTATION GÉNÉRALE DU BASIC

Le terme BASIC (ce sigle signifiant Beginner's All purpose Symbolic Instruction Code) désigne un des langages de programmation dits *évolués*.

Il a été conçu aux États-Unis, en 1965, pour traiter des problèmes de calculs, en mode conversationnel, à partir des terminaux.

L'usage du BASIC tend à se développer rapidement sur les micro-ordinateurs, car c'est un langage facile.

En raison de la grande variété de ces micro-ordinateurs, on peut trouver différentes versions du langage BASIC, chaque constructeur ayant fait des adaptations du BASIC en fonction des caractéristiques propres à chaque machine.

Nous nous attacherons à exposer ici le BASIC STANDARD, c'est-à-dire la partie commune aux différentes versions du BASIC.

2. RÈGLES GÉNÉRALES

1. Conventions d'écriture

Afin de simplifier l'utilisation de ce manuel, il faut préciser plusieurs points :

a) Dans les formats d'instructions, les mots en caractères gras doivent être utilisés exactement tels quels. Ce sont des commandes ou des instructions. Ils sont appelés **mots réservés** (*exemple*, page 8 : `n INPUT X`). La liste de ces mots est donnée en annexe.

b) Les rubriques entourées de [] sont en option. Les informations suivies de pointillés (...) peuvent être éventuellement répétées.

c) La ponctuation doit être respectée.

2. Les variables

a) Définition

Comme son nom l'indique, une variable est une expression dont le contenu peut varier.

On a généralement deux types de variables :

- les variables alphanumériques,
- les variables numériques.

b) Nom de variable

Une variable se définit par un nom qui commence toujours par une lettre de l'alphabet (de A à Z). Cette lettre peut être, ou non, suivie de lettres ou de chiffres.

Pour les variables alphanumériques, le nom doit toujours être suivi du symbole dollar \$.

Exemples

Variables alphanumériques : B \$
A5 \$
B3 \$
NOM \$

Variables numériques : B
A5
BA
NOM

REMARQUE. Tout nom de variable doit être différent des *mots réservés BASIC*.

3. Les constantes

a) Constantes numériques

Une constante numérique peut être un nombre positif, négatif ou nul, de 8 chiffres maximum, avec éventuellement un point décimal (.) et un exposant.

Exemples 270
 - 4.256
 2.345 E + 34 où E doit être lu « dix puissance ».

b) Constantes alphanumériques

Une constante alphanumérique peut avoir une longueur maximum de 255 caractères. Elle doit être exprimée entre guillemets (" ").

Exemples " NOM " " NUMÉRO DE CLIENT "

4. Les opérateurs

a) Les opérateurs arithmétiques

On utilise les opérateurs suivants :

- + pour l'addition
- pour la soustraction
- * pour la multiplication
- / pour la division
- ^ pour l'exponentiation

Ces opérateurs peuvent relier entre eux des constantes, des variables indicées ou non, des fonctions mathématiques.

b) Les opérateurs logiques

Ils sont utilisés dans les opérations logiques telles que ET, OU, NON, OU EXCLUSIF.

- AND pour l'opération ET

On écrira par exemple $C = A \text{ AND } B$

La machine considère les valeurs VRAIE ou FAUSSE de A et B exprimées en BINAIRE et opère ensuite selon la table suivante :

A	B	C = A AND B
1	1	1
1	Ø	Ø
Ø	1	Ø
Ø	Ø	Ø

où 1 signifie que l'expression est VRAIE
 Ø FAUSSE

Ainsi, si A est VRAIE ET B est VRAIE, alors C est VRAIE.

- **OR** pour l'opération **OU**

$$C = A \text{ OR } B$$

A	B	C = A OR B
1	1	1
1	Ø	1
Ø	1	1
Ø	Ø	Ø

- **NOT** pour l'opération **NON**

$$A = \text{NOT } B$$

B	A = NOT B
Ø	1
1	Ø

c) *Les opérateurs de relations logiques*

On utilise les symboles suivants :

=	égal	$A = B$
< >	différent	$A < > B$
<	inférieur	$A < B$
>	supérieur	$A > B$
<=, =<	inférieur ou égal	$A < = B$ ou $A = < B$
>=, =>	supérieur ou égal	$A > = B$ ou $A = > B$

d) *Priorité des opérateurs*

Lorsqu'une opération est placée entre parenthèses, elle est effectuée la première. Mais dans une expression mathématique où ne figure aucune parenthèse, les opérations sont effectuées selon des priorités dont l'ordre est le suivant :

1. \wedge exponentiation
2. - négation
3. *, / multiplication, division
4. +, - addition, soustraction
5. <, >, =, etc. relations logiques
6. NOT, AND, OR.

5. Rédaction d'un programme Basic

Un programme BASIC est une succession d'instructions.

Chaque instruction réalise une fonction bien définie.

Une même ligne peut renfermer plusieurs instructions différentes séparées entre elles par le symbole `:`.

Toute ligne du programme doit commencer par un numéro n , qui indique l'ordre dans lequel les instructions doivent être exécutées, quel que soit l'ordre dans lequel elles sont écrites :

n instruction BASIC [`:` instruction BASIC...]

La numérotation de 10 en 10 permet d'insérer de nouvelles instructions.

Exemples `10 Y = X + B`
 `20 Z = Y / 2 : C = Y AND Z`

Les espaces n'ont pas de signification dans l'écriture d'un programme BASIC (sauf pour les impressions), mais ils permettent une plus grande lisibilité.

Lorsqu'on frappe au clavier un programme BASIC, il faut appuyer sur la touche `RETURN` à la fin de chaque ligne.

Pour faire exécuter le programme, c'est-à-dire pour obtenir les résultats, on frappe au clavier le mot `RUN` suivi de `RETURN`.

instructions d'entrée-sortie

1. PRÉSENTATION

Ces instructions permettent de fournir des données à la machine à partir du clavier, et d'afficher les résultats depuis la machine sur l'écran ou sur l'imprimante.

2. ENTRÉE

a) INPUT

Format n INPUT x [, y] [, z]...
x, y, z représentent des variables numériques ou alpha-numériques.

Rôle L'instruction INPUT permet au calculateur d'attendre une information en provenance du clavier.

Les caractères enregistrés au clavier sont affectés aux variables de la liste suivant l'ordre d'entrée.

Pour « donner la main » à l'opérateur qui doit frapper les données au clavier, un point d'interrogation (?) apparaît sur l'écran et la machine attend une donnée.

Exemples

- 2Ø INPUT Q
 RUN
 ? 12 alors Q = 12

- 2Ø INPUT A, B, C \$
 RUN
 ? 3, 5, TOTAL alors A = 3, B = 5, C = " TOTAL "

b) INPUT " Chaîne de caractères "

Format n INPUT " Chaîne de caractères " [; liste de variables].

Rôle C'est le même que celui de l'instruction précédente. La chaîne de caractères placée entre guillemets apparaît sur l'écran avant le point d'interrogation.

Exemple 2Ø INPUT " NOMBRE DE COLIS : " ; Q
 RUN
 NOMBRE DE COLIS : ? 12 alors Q = 12

3. SORTIE

a) PRINT

Format n PRINT x [, y] [, z]... [;]

Rôle L'instruction d'édition PRINT permet d'afficher sur l'écran le contenu des expressions x, y, z.

Le point-virgule facultatif (;) à la fin de l'instruction permet d'éviter le retour à la ligne pour l'édition suivante.

De même que pour l'instruction INPUT, les expressions " chaînes de caractères " doivent être écrites entre guillemets (").

Exemples • 1Ø INPUT Q
 2Ø PRINT " NOMBRE DE COLIS =" ; Q
 RUN
 ? 15
 NOMBRE DE COLIS = 15

EXEMPLE 1

```
1Ø INPUT A,B
2Ø INPUT B$
3Ø PRINT
4Ø PRINT TAB(1Ø)"PROGRAMME NUMERO 1"
5Ø PRINT
6Ø PRINT TAB(8)A;TAB(12)B$;TAB(23)B
7Ø PRINT TAB(8)"-----"
RUN
? 25, 1980
? SEPTEMBRE
```

PROGRAMME NUMERO 1

25 SEPTEMBRE 1980

```
RUN
? 3, 1979
? JUIN
```

PROGRAMME NUMERO 1

3 JUIN 1979

EXEMPLE 2

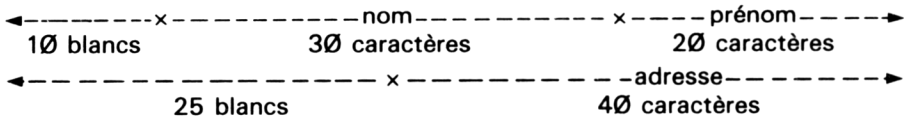
```
1Ø PRINT "LOSANGES"
2Ø PRINT
3Ø PRINT TAB(15)" +";TAB(2Ø)" +"
4Ø PRINT TAB(14)" +";TAB(16)" +";TAB(19)" + + +"
5Ø PRINT TAB(13)" +";TAB(17)" +";TAB(18)" + + + + +"
6Ø PRINT TAB(14)" +";TAB(16)" +";TAB(19)" + + + +"
7Ø PRINT TAB(15)" +";TAB(2Ø)" +"
```

```
RUN
LOSANGES
```

```
      +      +
     + +   + + +
    +   + + + + +
     + +   + + +
      +      +
```

EXERCICES

1. Écrire le petit programme BASIC permettant d'introduire de façon explicite les nom, prénom et adresse d'un individu, et de les écrire ensuite selon le modèle suivant :



2. Imaginer l'instruction BASIC permettant d'écrire la ligne d'en-tête suivante :

```
!   DÉSIGNATION   !   CODE   !   QUANTITÉ   !   PRIX MOYEN   !  
-----
```


opérations arithmétiques

1. Des expressions arithmétiques permettent d'écrire les calculs à effectuer sur des variables ou des constantes.
2. Un opérateur symbolise une opération entre deux opérandes.

Exemples $X + Y$ ou $Z * 3$ ou encore $T \wedge 2$

Chaque opération s'effectue sur les valeurs numériques des opérandes.

3. Dans une expression arithmétique, les différentes opérations sont effectuées en respectant les priorités des opérateurs.

Exemples Dans l'expression $X + 3 * Y \wedge 2$, la première opération effectuée est $Y \wedge 2$. Supposons que le résultat de cette opération soit A .
Il reste à effectuer $X + 3 * A$.
La machine fait d'abord $3 * A$. Supposons que $3 * A = B$. Le dernier calcul $X + B$ est enfin réalisé.

2. EXEMPLES D'ÉCRITURE D'EXPRESSIONS

- $X + \frac{Y Z}{4}$ s'écrit en BASIC $X + Y / 4 * Z$
ou encore $X + Y * Z / 4$
- $X + \frac{Y}{4 Z}$ s'écrit en BASIC $X + Y / 4 / Z$

Mais cette dernière expression heurte les habitudes mathématiques. Afin de pouvoir considérer $4 Z$ comme une opérande, il suffit de mettre entre parenthèses l'opération $4 * Z$ et l'expression devient alors $X + Y / (4 * Z)$.

- $\frac{X(Y + \frac{Z}{4}) - T}{5 + X}$ s'écrit en BASIC $(X * (Y + Z / 4) - T) / (5 + X)$.

REMARQUES

1. Une expression mathématique doit contenir autant de parenthèses « ouvrantes » que de parenthèses « fermantes ».
2. Une expression mathématique ne doit jamais contenir deux opérateurs consécutifs.
3. Une expression mathématique ne doit jamais commencer par un opérateur, sauf pour obtenir la valeur opposée de l'opérande ou de l'expression mathématique. Dans ce cas, on peut commencer par l'opérateur - (symbole de la négation). *Exemple* : $-(X + 3 * Y)$.

3. INSTRUCTION D'AFFECTATION

- LET

Format n LET X = Y
 X nom d'une variable numérique ou alphanumérique
 Y expression numérique ou alphanumérique

Rôle Cette instruction affecte l'expression Y à la variable X.

Après l'exécution de cette instruction, la valeur de la variable X est celle de l'expression Y, jusqu'à la rencontre d'une nouvelle instruction d'affectation.

REMARQUE. Le mot LET est facultatif et par conséquent
 $1\emptyset$ LET X = 4 est équivalent à $1\emptyset$ X = 4

Exemples $1\emptyset$ INPUT X
 $2\emptyset$ LET A = X + 1
 $3\emptyset$ INPUT B \$
 $4\emptyset$ C \$ = B \$
 $5\emptyset$ PRINT A, C \$
 RUN
 ? 15
 ? VARIABLE
 16 VARIABLE

EXEMPLE 3

```
10 PRINT "EXEMPLE RECAPITULATIF"
20 INPUT A,B,C,D
30 X=A-B
40 PRINT "A =" ; A ; TAB (10) "B =" ; B ; TAB (20) "C =" ; C ; TAB (30) "D =" ; D
50 PRINT "A-B =" ; X
60 PRINT "C+D =" ; C+D
70 Y=(A+B * C)/D
80 PRINT "Y=(A+B * C)/D =" ; Y
90 Z=A ^ 3
100 T=A-C+B * D/5
110 PRINT "A ^ 3 =" ; Z
120 PRINT "T =" ; T
```

EXEMPLE RECAPITULATIF

```
? 3, 2, .5, 10
A= 3      B= 2      C= .5      D= 10
A-B = 1
C+D = 10.5
Y=(A+B * C)/D= .4
A ^ 3=27
T= 6.5
```

EXERCICES

1. Écrire le programme BASIC qui permet de calculer

$$Z = \frac{(3 A + 5 B) - A^4 (5 B - A)}{4 C - D}$$

après avoir introduit ensemble A, B, C et D.

2. Calculer et imprimer la circonférence et la surface d'un cercle dont on connaît le rayon.

les branchements

1. PRÉSENTATION

Jusqu'à présent nous n'avons vu que des calculs séquentiels qui se présentent comme une succession d'opérations élémentaires.

a) Mais dans la plupart des problèmes que doit traiter un ordinateur, les calculs se répètent plusieurs fois, par exemple : le calcul d'une ligne de facture, la paie d'un employé, etc.

b) D'autre part, dans la résolution de certains problèmes, l'ordinateur aura à exécuter certaines opérations à l'exception d'autres en fonction d'une condition. Par exemple, lors de la résolution de l'équation du second degré $ax^2 + bx + c = 0$, trois cas sont possibles ; le programme doit les prévoir tous les trois, mais l'ordinateur ne doit exécuter que l'un des trois en fonction du signe de $\Delta = b^2 - 4ac$. Il faut donc que le programme « se branche » sur l'une des trois séquences possibles.

Dans le cas a), on aura des branchements systématiques ou **inconditionnels**.

Dans le cas b), le branchement est **conditionnel**.

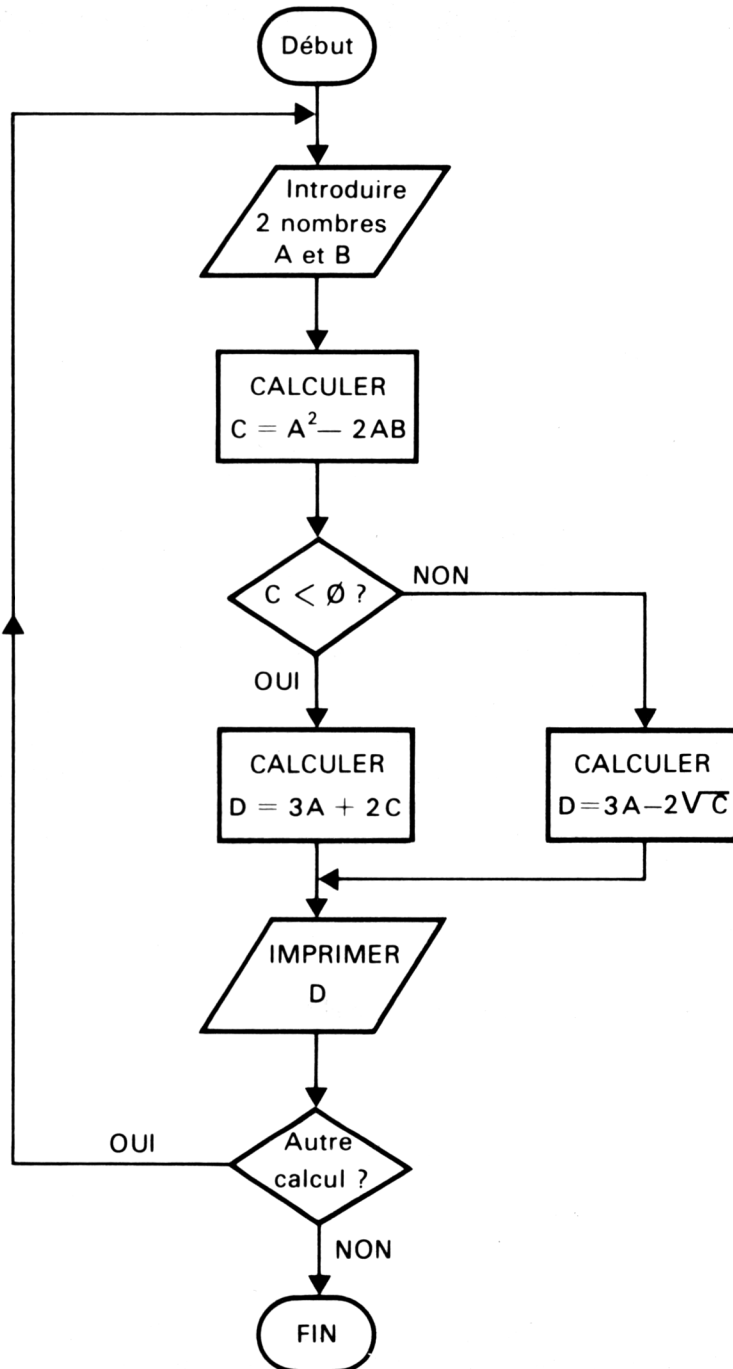
Exemple

- Soit 2 nombres A et B.
Calculer $C = A^2 - 2AB$.

Si $C \geq 0$, calculer $D = 3A - 2\sqrt{C}$ et imprimer D.

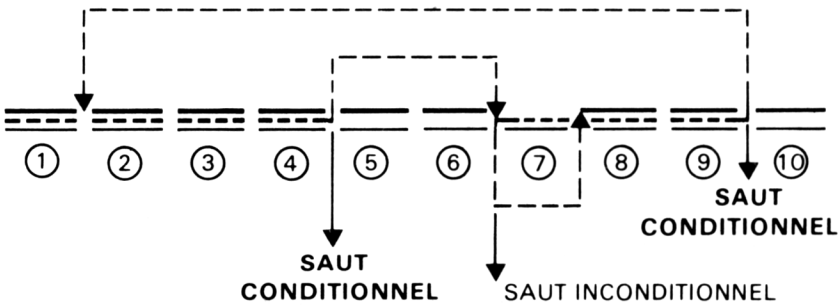
Si $C < 0$, calculer $D = 3A + 2C$ et imprimer D.

• Organigramme



• **Opérations**

- ① DÉBUT
- ② INTRODUIRE A et B
- ③ CALCULER $C = A^2 - 2 A B$
- ④ SI $C < 0$ ALLER en ⑤ SINON ALLER en ⑦
- ⑤ CALCULER $D = 3 A + 2 C$
- ⑥ ALLER en ⑧
- ⑦ CALCULER $D = 3 A - 2 \sqrt{C}$
- ⑧ AFFICHER D
- ⑨ Si autre calcul ALLER en ② SINON ALLER en ⑩
- ⑩ FIN



2. LES BRANCHEMENTS INCONDITIONNELS

a) GOTO

Format n GO TO numéro de ligne.

Rôle Permet le branchement systématique du programme à la ligne dont le numéro est précisé derrière GOTO.

Exemple 30 GOTO 100
 ⋮
 100 PRINT " FIN DE PROGRAMME "

Dès que la ligne 30 est exécutée, le programme saute à l'instruction de la ligne 100 et ignore les intructions intermédiaires.

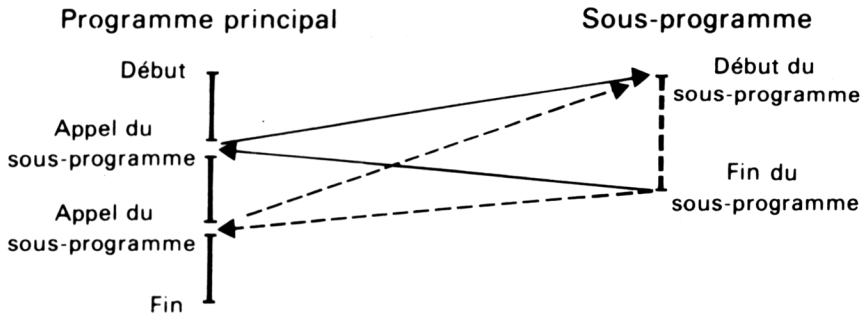
REMARQUE. Le numéro de la ligne suivant GOTO peut être inférieur à n , numéro de ligne de l'instruction GOTO.

```
30 INPUT X, Y
40 PRINT X
50 PRINT Y
60 GOTO 30
RUN
? 4, 10
4
10
? 5, 0
5
0
?
```

b) GOSUB

Notion de sous-programme

Un sous-programme est une suite d'instructions utilisée plusieurs fois non consécutives au cours de l'exécution d'un programme.



Le programme principal se branche à un sous-programme par l'intermédiaire de l'instruction GOSUB.

Format n GOSUB numéro de ligne.

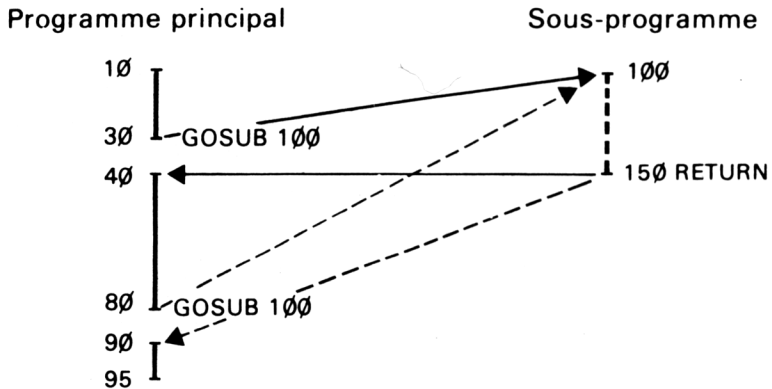
Rôle Permet au programme principal de se brancher au sous-programme dont la première instruction est à la ligne précisée après GOSUB.

Mais le sous-programme doit pouvoir se brancher à son tour au programme principal. Pour cela, on utilise l'instruction BASIC : RETURN.

c) RETURN

Format *n* RETURN.

Rôle Permet, après l'exécution d'un sous-programme, de revenir au programme principal, à la première ligne suivant l'instruction GOSUB d'origine.



Exemple

```

10 INPUT A
20 PRINT " A = "; A
30 GOSUB 110
40 C = A + B / 2
50 PRINT " C = "; C

110 INPUT X
120 B = (2 * X ^ 2) / 4
130 PRINT " B = "; B
140 RETURN

RUN
? 10
A = 10
? 4
B = 8
C = 14

```

3. RÉPÉTITIONS DE CALCUL : LECTURE DIRECTE DE DONNÉES

Un programme n'est vraiment intéressant que lorsqu'il permet d'effectuer les calculs pour plusieurs séries de valeurs.

Exemples

```
1Ø INPUT A, B
2Ø C = (A + B) ^ 2
3Ø PRINT " A = "; A, " B = "; B, " C = "; C
RUN
? 2, 3
A = 2      B = 3      C = 25
```

A ce niveau-là, si nous voulons refaire le calcul pour des valeurs différentes de A et B, deux solutions sont possibles :

- soit revenir à l'instruction 1Ø par un ordre GOTO, mais la machine s'arrêtera toujours sur le ? de l'ordre INPUT ;
- soit de fournir à la machine une série de données dès l'écriture du programme. Cela se fait par deux instructions BASIC :
 - une description des données : DATA,
 - une lecture de ces données : READ.

Dans ce cas, l'instruction INPUT n'existe plus.

a)

DATA

Format n DATA liste de constantes.

Rôle Permet de ranger dans la mémoire des constantes numériques ou alphanumériques.

Les différentes données de la liste doivent être séparées par des virgules ; exemple : 5Ø DATA 1, 2, - 3, 5

REMARQUES

- Une ligne d'instruction DATA peut être placée n'importe où dans un programme.
- Il est possible d'écrire un nombre quelconque d'instructions DATA dans un même programme.
- Les instructions DATA permettent de ranger toutes les données en mémoire.

Exemple

Dans un programme, on trouve trois lignes DATA :

```
25 DATA 3, - 2
60 DATA 10, - 4.5, 182
75 DATA 3, 8, 1
```

En mémoire, la zone données est constituée comme suit :

3	- 2	10	- 4.5	182	3	8	1
---	-----	----	-------	-----	---	---	---

L'ordinateur affecte ensuite ces données aux variables du programme par l'instruction de lecture des données READ.

b)

READ

Format *n* READ liste de variables.

Rôle Permet d'accéder aux constantes rangées en mémoire par l'ordre DATA.

Exemples

- 10 READ X
20 PRINT X
30 GOTO 10
40 DATA 4, 2, 1, 3
RUN
4
2
1
3
- 10 READ X, Y
20 PRINT X, Y
30 GOTO 10
40 DATA 4, 2, 1, 3, 6, 5
RUN
4 2
1 3
6 5
- 10 READ X, Y
20 PRINT X, Y
30 GOTO 10
70 DATA 4, 2, 1, 3
80 DATA 6, -7, 4.5, 9

```

90 DATA 10, 185, -100, 2
RUN
  4          2
  1          3
  6         -7
  4.5        9
  10         185
- 100        2

```

REMARQUE. La machine s'arrête lorsqu'elle a lu toutes les valeurs des constantes fournies par les ordres DATA. Si un autre ordre READ est rencontré dans le programme, la machine imprimera un message d'erreur. Cependant, il est possible de faire reprendre la lecture au début de la zone des données.

c) RESTORE

Format *n* RESTORE

Rôle Permet de réutiliser, par un ordre READ, les constantes de DATA en commençant par la première.

Exemples

- 10 READ A, B, C
 20 PRINT A, B, C
 30 RESTORE
 40 READ X
 50 PRINT X : GOTO 40
 60 DATA 4, - 3, 10
 RUN
 4 - 3 10
 4
 - 3
 10
- 10 READ X, Y
 20 PRINT X, Y
 30 RESTORE
 40 READ Z
 50 PRINT Z
 60 DATA 4, - 3, 10, 2.5
 RUN
 4 - 3
 4

EXEMPLE 4

```
1Ø PRINT "CIRCONFÉRENCE ET SURFACE D'UN CERCLE"  
2Ø READ R  
3Ø C = 2 * 3.14 * R  
4Ø S = 3.14 * R * R  
5Ø PRINT "R = " ; R, "C = " ; C, "S = " ; S  
6Ø GOTO 2Ø  
7Ø DATA 1, 3, 4, 7, 2, 8, 12  
8Ø END
```

RUN

CIRCONFÉRENCE ET SURFACE D'UN CERCLE

R = 1	C = 6.28	S = 3.14
R = 3	C = 18.84	S = 28.26
R = 4	C = 25.12	S = 50.24
R = 7	C = 43.96	S = 153.86
R = 2	C = 12.56	S = 12.56
R = 8	C = 50.24	S = 200.96
R = 12	C = 75.36	S = 452.16

4. BRANCHEMENTS CONDITIONNELS

a)

IF... THEN...

Format *n* IF expression THEN *p*
p représente un numéro de ligne, mais il peut être remplacé par une instruction BASIC ;
exemple : 1Ø IF A = B THEN PRINT " A = B " .

Rôle Branchement conditionnel permettant de se placer en un point ou un autre d'un programme en fonction d'une expression.

– Si l'expression est VRAIE, l'opération *p* est exécutée ;

– Si l'expression est FAUSSE, l'opération de l'instruction suivant immédiatement l'instruction *n*, est exécutée.

REMARQUE. Dans certaines versions de BASIC, on trouve le format suivant de IF.

n IF expression THEN p ELSE m

Si l'expression est FAUSSE, l'opération m est exécutée.

Exemples

- 10 INPUT A
20 IF A = 5 THEN 50
30 C = A - 5
40 PRINT " C = "; C,
50 PRINT " A = "; A
60 GOTO 10
RUN
? 5
A = 5
? 6
C = 1 A = 6
?
- 10 INPUT A
20 IF A <> 0 THEN 40
30 PRINT " A = ZÉRO " : GOTO 10
40 B = 1 / A
50 PRINT " B = "; B
RUN
? 2
B = .5
RUN
? 0
A = ZÉRO
?
- 10 INPUT A, B, C
20 IF A = B THEN IF B = C THEN PRINT
 " A = C " ELSE PRINT " A < > C " ELSE
 PRINT " A = "; A, " B = "; B
30 GOTO 10
RUN
? 2, 2, 2
A = C
? 1, 1, 5
A < > C
? 1, 2, 1
A = 1 B = 2

b) ON... GOTO...

Format n ON expression GOTO liste de numéros de ligne.

Rôle Cette instruction permet des branchements multiples en fonction de l'expression suivant ON.

La machine calcule donc cette valeur. Soit m cette valeur (arrondie au nombre entier le plus proche). Puis le programme exécute l'instruction dont le numéro est le $m^{\text{ième}}$ de la liste.

Exemple

```

1Ø INPUT X
2Ø ON 2 * X + 1 GOTO 3Ø, 5Ø, 7Ø
3Ø PRINT " NO 1 : X = Ø "
4Ø GOTO 1Ø
5Ø PRINT " NO 2 : X = 1/2 "
6Ø GOTO 1Ø
7Ø PRINT " NO 3 : X = 1 "
8Ø GOTO 1Ø
RUN
? 1
NO 3 : X = 1      (2 X + 1 = 3 ⇒ branchement à 7Ø)
? Ø
NO 1 : X = Ø     (2 X + 1 = 1 ⇒ branchement à 3Ø)
? .5
NO 2 : X = 1/2   (2 X + 1 = 2 ⇒ branchement à 5Ø)
?

```

c) ON... GOSUB...

Format n ON expression GOSUB liste de numéros de ligne.

Rôle Identique à celui de l'instruction ON... GOTO, mais, ici, les numéros de ligne sont ceux des premières de différents sous-programmes.

Exemple

```

1Ø INPUT X
2Ø ON X GOSUB 1ØØ, 2ØØ, 3ØØ
3Ø GOTO 1Ø

```

```

100 PRINT " SOUS-PROGRAMME-1 "
110 Y = 2 * X + 1
120 PRINT " Y = "; Y
130 RETURN

200 PRINT " SOUS-PROGRAMME-2 "
210 Y = X / 2 + 1
220 PRINT " Y = "; Y
230 RETURN

300 PRINT " SOUS-PROGRAMME-3 "
310 PRINT " Y = "; X
320 RETURN
RUN
? 1
SOUS-PROGRAMME-1
Y = 3
? 3
SOUS-PROGRAMME-3
Y = 3
?

```

5. CONTRÔLE DE BOUCLE

Pour contrôler le nombre de fois où une boucle a été effectuée, on compare la valeur d'un compteur à un nombre N connu.

Exemple

```

10 INPUT N
20 I = 1
30 READ X
40 Y = X ^ X
50 PRINT X, Y
60 I = I + 1
70 IF I = < N THEN 30
80 PRINT " FIN "
90 DATA 1, 3, 2
RUN
? 2
1      1
3      9
FIN

```

Il est possible d'écrire ce programme de façon plus claire grâce à l'instruction « de boucle » FOR.

- FOR... NEXT

Format

n FOR X = Y TO Z [STEP V]

⋮

m NEXT X

où : X est la variable itérative

Y la valeur initiale de la variable

Z la valeur limite de la variable

V le pas d'itération (V = 1 implicite).

Rôle

Permet d'exécuter une boucle de programme plusieurs fois.

Au départ X = Y, le programme exécute toutes les lignes jusqu'à l'instruction NEXT.

A ce niveau-là, X prend la valeur X + V et la machine compare X à Z :

- si $V > 0$ ET si $X \leq Z$
ou
- si $V < 0$ ET si $X \geq Z$ le programme se branche à la première instruction suivant FOR.
- si non, il se branche à la première instruction suivant NEXT.

Exemples

- 10 FOR I = 1 TO 5
20 READ X
30 PRINT " X = "; X
40 NEXT I
50 PRINT " FIN "
60 DATA 1, 3, 2, 5, 10, 9, 7, 6, 8, 4
RUN
X = 1
X = 3
X = 2
X = 5
X = 10
FIN
- 10 FOR I = 3 TO 1 STEP - 1
20 INPUT A
30 PRINT " A = "; A
40 NEXT I
50 PRINT " FIN "

```

RUN
? 4
A = 4
? 10
A = 10
? 4.5
A = 4.5
FIN

```

• **BOUCLES IMBRIQUÉES**

```

10 FOR I = 1 TO 3
20 FOR K = 1 TO 2
30 PRINT "I = "; I
40 PRINT "K = "; K
50 NEXT K
60 NEXT I
RUN
I = 1      K = 1
I = 1      K = 2
I = 2      K = 1
I = 2      K = 2
I = 3      K = 1
I = 3      K = 2

```

REMARQUES

- Une boucle de niveau inférieur doit être entièrement imbriquée dans une autre de niveau supérieur.

Le schéma est le suivant :

```

---10 FOR I...
  | 20 FOR K...
  | 70 NEXT K
---80 NEXT I

```

On ne doit jamais trouver :

```

---10 FOR I...
  | 20 FOR K...
  | 60 NEXT I
  | 70 NEXT K

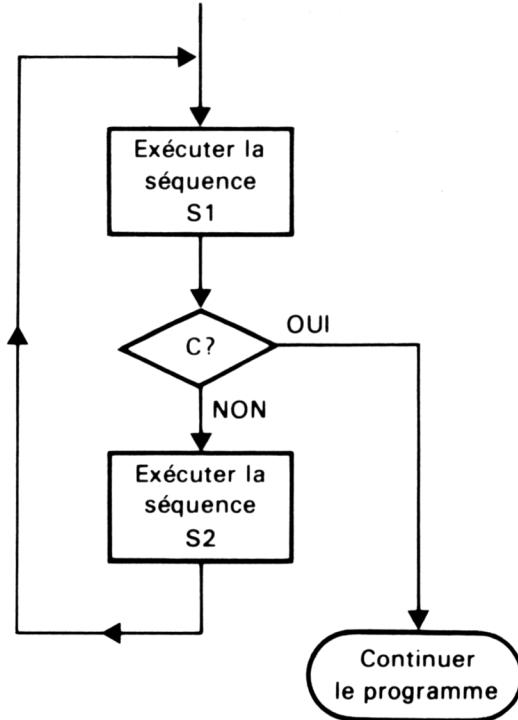
```

- On ne peut pas, au cours d'un programme, se rebrancher à une instruction se trouvant à l'intérieur d'une boucle FOR. C'est une des raisons pour lesquelles on a souvent intérêt à éviter l'utilisation de l'instruction FOR et à lui préférer une des trois possibilités suivantes :

1. Boucle ITÉRATION

L'organigramme logique se présente ainsi :

C ? signifie : la condition C est-elle satisfaite ?



Exemple

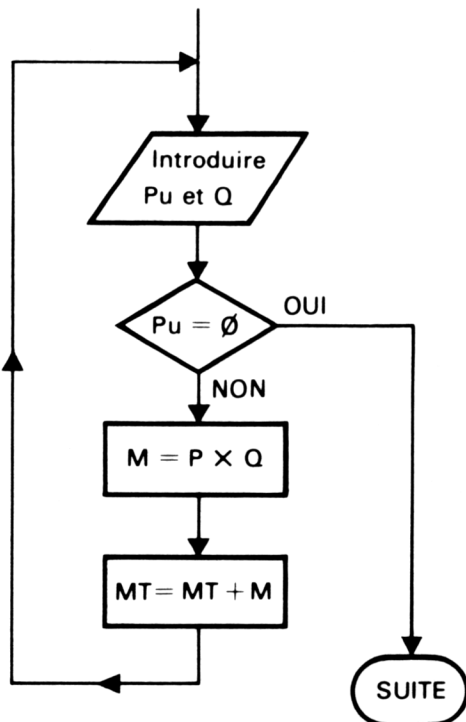
Considérons une facturation. On introduit le prix unitaire P_u et la quantité Q .

On calcule $M = P_u \times Q$
et le cumul $MT = \Sigma M$.

On introduit $P_u = 0$ pour signaler la dernière ligne-détail.

```

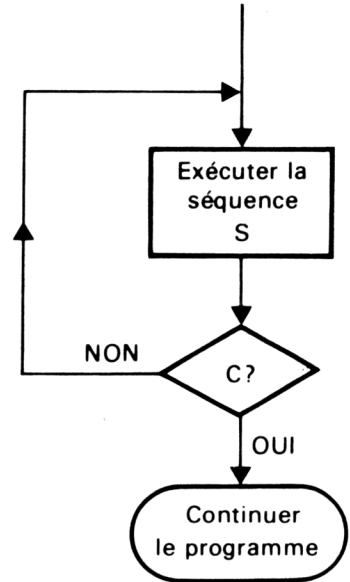
30 INPUT PU, Q
40 IF PU = 0 THEN 80
50 M = PU * Q
60 MT = MT + M
70 GOTO 30
80 ...
  
```



2. Boucle RÉPÉTER jusqu'à...

Le test se trouve à la fin et l'organigramme est le suivant :

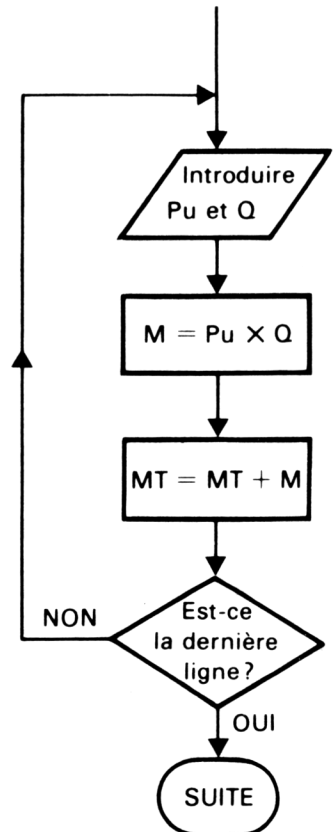
RÉPÉTER la séquence S JUSQU'À satisfaction de la condition C.



Exemple

Même exemple, sauf au niveau de la dernière ligne, pour laquelle la machine pose la question plus nettement :

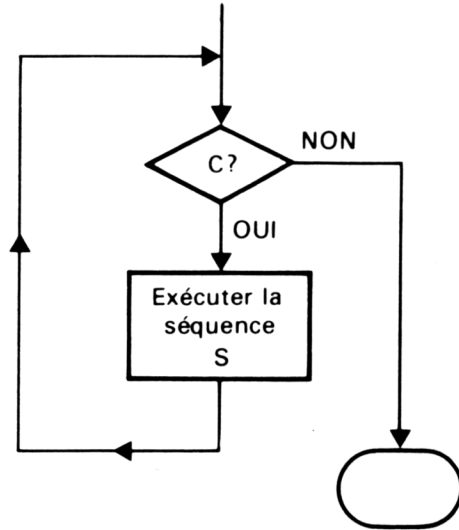
```
40 INPUT PU, Q
50 M = PU * Q
60 MT = MT + M
70 PRINT " EST-CE LA DERNIÈRE LIGNE ? "
80 INPUT X $
90 IF X $ = " OUI " THEN 110
100 GOTO 40
110 ...
```



3. Boucle TANT QUE

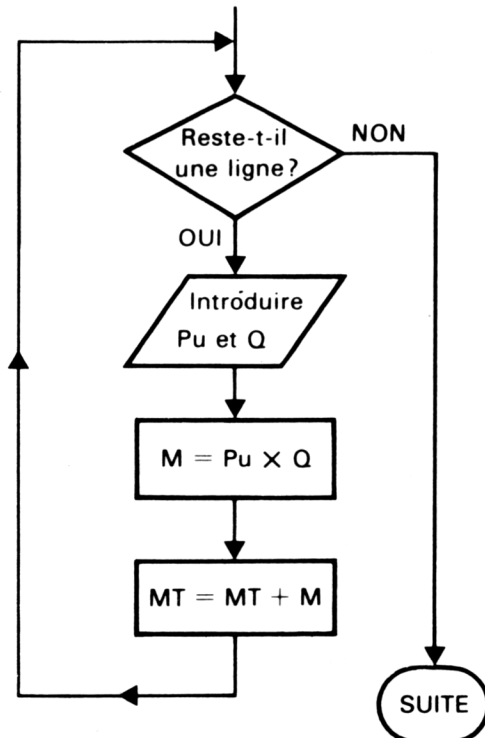
Ici, le test est au début :

TANT QUE la condition C n'est pas satisfaite, exécuter la séquence S.



Exemple Conservons l'exemple de la facturation.

```
40 INPUT " AVEZ-VOUS UNE LIGNE A CALCULER ? ", X $  
50 IF X $ = " NON " THEN 100  
60 INPUT PU, Q  
70 M = PU * Q  
80 MT = MT + M  
90 GOTO 40  
100 ...
```



EXEMPLE 5

```
J10 PRINT "TABLE DES SALAIRES"  
J20 PRINT "HEBDO", "MENSUEL", "ANNUEL"  
J30 FOR M=1000 TO 10000 STEP 500  
J40 A=12*M  
J50 H=A/52  
J60 PRINT H, M, A  
J70 NEXT M  
J80 END
```

JRUN

TABLE DES SALAIRES

HEBDO	MENSUEL	ANNUEL
230.769231	1000	12000
346.153846	1500	18000
461.538462	2000	24000
576.923077	2500	30000
692.307693	3000	36000
807.692308	3500	42000
923.076923	4000	48000
1038.46154	4500	54000
1153.84615	5000	60000
1269.23077	5500	66000
1384.61538	6000	72000
1500	6500	78000
1615.38462	7000	84000
1730.76923	7500	90000
1846.15385	8000	96000
1961.53846	8500	102000
2076.92308	9000	108000
2192.30769	9500	114000
2307.69231	10000	120000

EXERCICES

1. Calculer le P.G.C.D. de deux nombres.
2. Calculer la somme des n premiers nombres impairs.
Application numérique : $n = 50$, $n = 150$.
3. Écrire le programme permettant de calculer le total brut d'une facture, c'est-à-dire la somme des montants $P_u \times Q$.
Les nombres de lignes de chaque facture sont introduits par une instruction DATA (10 factures).
4. Reprendre l'exercice précédent et calculer une remise :
 - de 5 % pour un total brut ≥ 350 F,
 - de 10 % pour un total brut $\geq 1\ 000$ F.

les fonctions

1. PRÉSENTATION

Dans une expression mathématique, les opérandes peuvent être des constantes ou des variables. *Exemple* : $X = 2 * (Y + Z)$

Dans certains cas, les opérandes peuvent être les résultats de calculs particuliers tels que **racine carrée**, **logarithme**, etc.

BASIC possède un certain nombre de fonctions **standard** qui fournissent directement ces résultats particuliers et qui peuvent donc constituer un opérande d'une opération.

2. FONCTIONS DE CALCUL

1. Calculs algébriques

a) ABS

Format **ABS (X)**

Rôle Fournit la valeur absolue de l'argument X.

- Si $X \geq 0$ $ABS(X) = X$
- Si $X < 0$ $ABS(X) = -X$

Exemple 1Ø INPUT A
 2Ø PRINT " ABS (A) = " ; ABS (A)
 RUN
 ? - 4
 ABS (A) = 4

b) SGN

Format **SGN (X)**

Rôle Permet de déterminer le signe d'une variable ; en effet :

$\text{SGN}(X) = 1$ si X est positif

$\text{SGN}(X) = \emptyset$ si X est nul

$\text{SGN}(X) = -1$ si X est négatif

Exemple

```
1Ø INPUT I
2Ø A = SGN (I)
3Ø IF A = -1 THEN PRINT " I EST NÉGATIF " : GOTO 1Ø
4Ø IF A = Ø THEN PRINT " I EST NUL " : GOTO 1Ø
5Ø PRINT " I EST POSITIF " : GOTO 1Ø

RUN
? 5. 315
I EST POSITIF
? Ø
I EST NUL
? - 3.14
I EST NÉGATIF
?
```

c) SQR contraction de l'expression « square root »
(racine carrée)

Format **SQR (X)**

Rôle Pour tout X positif ou nul,
donne la racine carrée de X.

Exemple

```
1Ø INPUT A
2Ø PRINT SQR (A)
3Ø GOTO 1Ø

RUN
? 2
1.41421
? 4
2
?
```

2. Calculs trigonométriques

a) COS

Format COS (X)

Rôle Fournit le cosinus de l'angle X, pour tout X exprimé en **radians**.

Exemple 1Ø INPUT X
 2Ø Y = COS (X)
 3Ø PRINT " COS (X) = "; Y
 RUN
 ? 3.14159
 COS (X) = 1

b) SIN

Format SIN (X)

Rôle Fournit le sinus de l'angle X, pour tout X exprimé en **radians**.

Exemple 1Ø INPUT X
 2Ø Z = SIN (X)
 3Ø PRINT " SIN (X) = "; Z
 RUN
 ? 3.14159
 SIN (X) = Ø

REMARQUE. Selon les constructeurs, on peut disposer d'autres fonctions trigonométriques telles que TANGENTE, ARCTANGENTE..., etc...

3. Calculs logarithmique et exponentiel

a) LOG

Format **LOG (X)**

Rôle Donne le logarithme népérien (c'est-à-dire à base e) de X pour tout X positif

Exemple 1Ø INPUT X
 2Ø PRINT LOG (X)
 RUN
 ? 2
 Ø. 693147

REMARQUE. Pour calculer le logarithme d'un nombre dans une autre base, on utilisera la formule suivante :

$$\log_b (X) = \log_e (X) / \log_e (b)$$

Exemple 1Ø INPUT X
 2Ø Y = LOG (X) / LOG (1Ø)
 3Ø PRINT " LOGARITHME DÉCIMAL DE X = "; Y

b) EXP

Format **EXP (X)**

Rôle Fournit la valeur exponentielle de X en base e .

Exemple 1Ø INPUT X
 2Ø PRINT EXP (LOG (X))
 RUN
 ? 3
 3 car X = EXP (LOG (X))

4. Calculs de conversion

- `INT` (abréviation de *integer* qui signifie entier).

Format INT (X)

Rôle Donne une représentation entière de X.

C'est la valeur du plus grand nombre entier inférieur ou égal à X.

Exemple

```
1Ø INPUT X
2Ø PRINT " PARTIE ENTIÈRE DE X = "; INT (X)
3Ø GOTO 1Ø

RUN
? 3.85
PARTIE ENTIÈRE DE X = 3
? - 2.94
PARTIE ENTIÈRE DE X = - 3
? 256 357 . 9
PARTIE ENTIÈRE DE X = 256 357
```

3. FONCTIONS DÉFINIES PAR L'OPÉRATEUR

1. Présentation

La liste des fonctions **standard** est limitée à des fonctions classiques. Dans un programme, on peut avoir à calculer plusieurs fois la valeur d'une même expression pour des valeurs différentes des variables.

Par exemple, il faudra calculer pour différentes valeurs de X et Y, les valeurs de l'expression :

$$(X * Y) / \text{SQR}(X \wedge Y)$$

Afin d'éviter d'écrire cette expression à chaque fois, on crée une nouvelle fonction que l'on pourra appeler comme une fonction **standard**.

2. Définition de la fonction

Elle se fait grâce à l'instruction DEF.

- DEF

Format n DEF nom de la fonction (liste de variables) = expression. Le nom de la fonction doit obligatoirement commencer par FN.

Rôle Permet au programmeur d'utiliser de nouvelles fonctions n'appartenant pas au BASIC.

Exemple 10 DEF FNA (X, Y) = (X * Y) / SQR (X ^ Y)

3. Utilisation de la fonction

Elle s'utilise comme une fonction standard.

Exemples • 10 DEF FNA (X, Y) = (X * Y) / SQR (X ^ Y)
 • 50 A = 4 + FNA (3,5)
 • 10 DEF FNAR (X, Y) = (X * Y) / SQR (X ^ Y)
 • 50 INPUT " MESURES = "; A, B
 • 90 V = 3 * FNAR (A, B)

EXERCICES

1. Convertir un nombre décimal en binaire.
Écrire le programme BASIC correspondant.
2. Convertir en décimal un nombre donné en binaire.
Écrire le programme BASIC correspondant.
3. Un nombre donné est-il premier ? Écrire le programme BASIC.

EXEMPLE 6

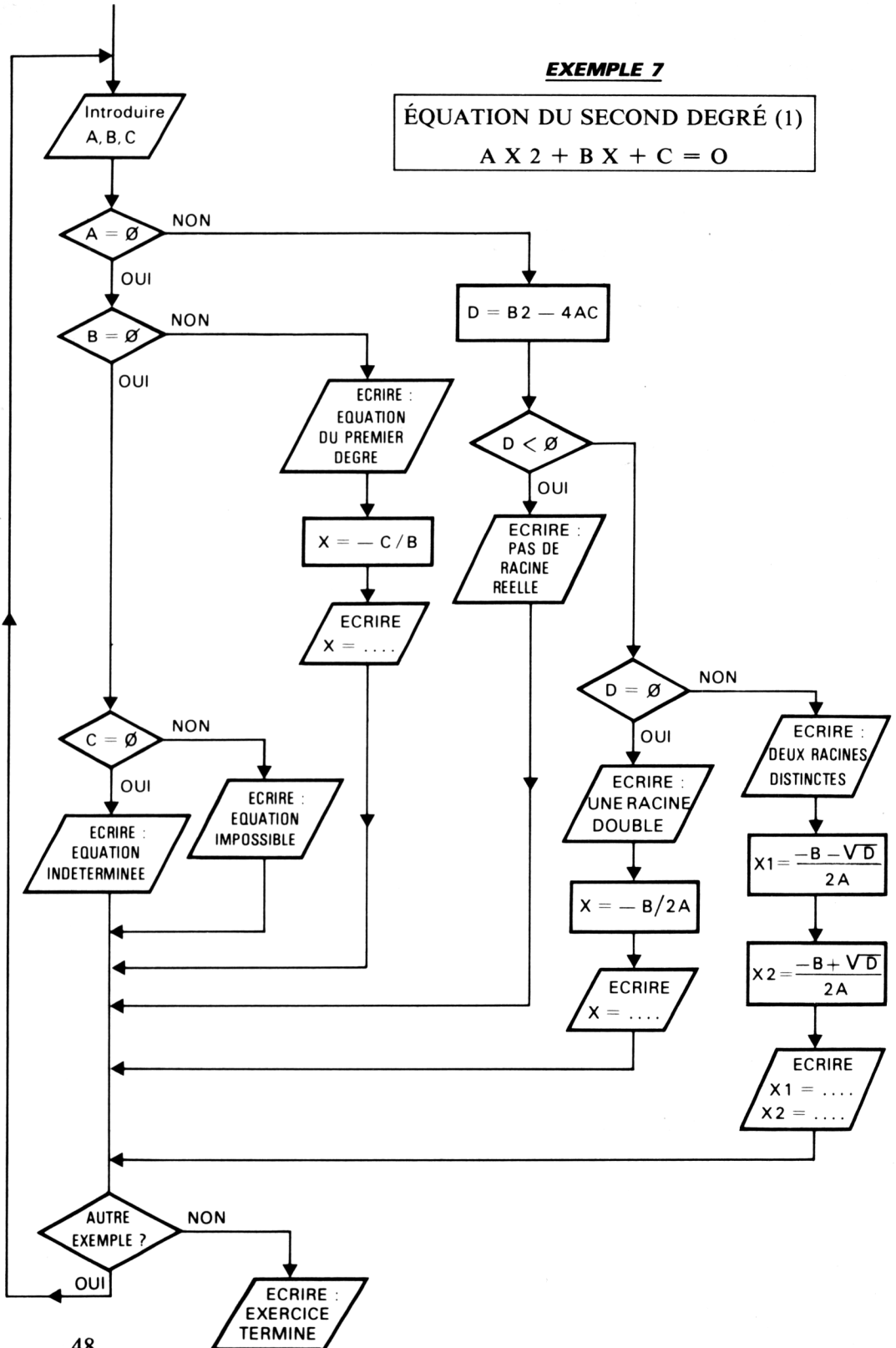
```
2 PRINT "LISTE DES NOMBRES PREMIERS"  
4 PRINT "ENTRE 2 ET 60"  
5 PRINT  
10 FOR N = 2 TO 100  
20 T = INT (SQR (N) + 2)  
30 FOR I = 2 TO T  
40 IF INT (N / I) = N / I THEN 55  
45 NEXT I  
50 PRINT N  
55 NEXT N
```

```
RUN  
LISTE DES NOMBRES PREMIERS  
ENTRE 2 ET 60
```

```
5  
7  
11  
19  
23  
29  
31  
37  
41  
43  
47  
53  
59
```

EXEMPLE 7

ÉQUATION DU SECOND DEGRÉ (1)
 $A X^2 + B X + C = 0$



ÉQUATION DU SECOND DEGRÉ (2)

```
110 PRINT "EQUATION DU SECOND DEGRE"  
115 PRINT "AX2+BX+C=0"  
117 PRINT  
120 INPUT A,B,C  
130 IF A<> 0 THEN 110  
140 IF B=0 THEN 80  
150 PRINT "EQUATION DU PREMIER DEGRE"  
160 PRINT "LA RACINE EST EGALE A : - C/B SOIT X = "; - C/B  
170 GOTO 210  
180 IF C=0 THEN 100  
190 PRINT "EQUATION IMPOSSIBLE": GOTO 210  
100 PRINT "EQUATION INDETERMINEE" : GOTO 210  
110 D=B*B-4 * A * C  
1120 IF D < 0 THEN PRINT "PAS DE RACINE REELLE" : GOTO 210  
1130 IF D=0 THEN 190  
1140 PRINT "DEUX RACINES DISTINCTES"  
1150 X1=(-B-SQR(D))/(2* A)  
1160 X2=(-B+SQR(D))/(2* A)  
1170 PRINT "X1 = "; X1, "X2 = "; X2  
1180 GOTO 210  
1190 PRINT "UNE RACINE DOUBLE"  
1200 PRINT "X = "; -B/(2* A)  
1210 PRINT "VOULEZ-VOUS RECOMMENCER? -OUI=1 NON=0 -"  
1220 INPUT Z  
1230 IF Z=1 THEN 20  
1240 PRINT "EXERCICE TERMINE"
```

ÉQUATION DU SECOND DEGRÉ (3)

JRUN

EQUATION DU SECOND DEGRE

$$AX^2+BX+C=\emptyset$$

?2, 1, 5

PAS DE RACINE REELLE

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

?1

?3, \emptyset , 4

PAS DE RACINE REELLE

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

? \emptyset , 2, -3

?EXTRA IGNORED

EXERCICE TERMINE

JRUN

EQUATION DU SECOND DEGRE

$$AX^2+BX+C=\emptyset$$

? \emptyset , 2, -3

EQUATION DU PREMIER DEGRE

LA RACINE EST EGALE A : -C/B SOIT X = 1. 5

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

?1

? \emptyset , \emptyset , 4

EQUATION IMPOSSIBLE

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

?1

?4, \emptyset , \emptyset

UNE RACINE DOUBLE

$$X=\emptyset$$

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

?1

?1, 1, -6

DEUX RACINES DISTINCTES

$$X_1=-3 \quad X_2=2$$

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

?1

? \emptyset , \emptyset , \emptyset

EQUATION INDETERMINEE

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

?1

?112, 2, -5

DEUX RACINES DISTINCTES

$$X_1=-2.1583124 \quad X_2=1.1583124$$

VOULEZ-VOUS RECOMMENCER? -OUI=1 NON= \emptyset -

? \emptyset

EXERCICE TERMINE

traitement des chaînes de caractères

1. PRÉSENTATION

On appelle **chaîne**, une succession de caractères entourée des symboles " (guillemets).

Chacun de ces caractères peut être n'importe lequel du clavier de la machine, y compris le blanc.

- Le nom d'une chaîne de caractères est toujours suivi du symbole « dollar » : \$.
- L'ordinateur considère une chaîne de caractères comme une valeur à laquelle il est possible de faire subir différents traitements.

2. TRAITEMENT DES CHAINES DE CARACTÈRES

a) LEN

Format **LEN (A\$)**

Rôle **LEN (A\$)** représente la longueur de la chaîne de caractères **A\$**.

LEN (A\$) est une valeur numérique.

Exemple

```
1Ø INPUT I$
2Ø PRINT LEN (I$)
3Ø GOTO 1Ø
RUN
? BASIC
5
? BASIC EST UN LANGAGE INTÉRESSANT
32
?
```

b) LEFT\$

Format

LEFT\$ (A\$, X)

A\$ = chaîne de caractères.

X = constante ou expression numérique.

Rôle

LEFT\$ (A\$, X) représente les X premiers caractères de gauche de la chaîne de caractères A\$.

Exemple

```
1Ø INPUT A$
2Ø Y $ = LEFT $ (A$, 3)
3Ø PRINT Y$
4Ø GOTO 1Ø
RUN
? CARACTÈRES
CAR
? CARTE
CAR
? BASIC
BAS
?
```

c) RIGHT\$

Format

RIGHT\$ (A\$, X)

A\$ = chaîne de caractères.

X = constante ou expression numérique.

Rôle **RIGHT\$ (A\$, X)** représente les X derniers caractères de A\$.

Exemple
1Ø Y\$ = " LANGAGE BASIC "
2Ø M\$ = RIGHT\$ (Y\$, 6)
3Ø PRINT " LE "; M\$
RUN
LE BASIC

REMARQUE. Si $X \geq \text{LEN}(A\$)$
 $\text{LEFT\$}(A\$, X) = \text{RIGHT\$}(A\$, X) = A\$$

d) MID\$

Format **MID\$ (A\$, X, Y)**
A\$ = chaîne de caractères.
X et Y = constantes ou expressions numériques.

Rôle MID\$ (A\$, X, Y) permet d' « isoler » Y caractères de A\$, à partir du Xème inclus.

Exemple
1Ø INPUT A\$
2Ø D\$ = MID\$ (A\$, 4, 5)
3Ø PRINT " D\$ = "; D\$
RUN
? LE BASIC STANDARD
D\$ = BASIC

e) STRING\$

Format **STRING\$ (X, " C ")**
X = constante ou expression numérique.
C = caractère écrit entre guillemets.

Rôle Représente une chaîne de X caractères C.

Exemples

- 1Ø PRINT " TITRE "
2Ø PRINT STRING\$(5, " * ")
RUN
TITRE

- 1Ø INPUT A\$
2Ø PRINT A\$
3Ø PRINT STRING\$(LEN(A\$), " - ")
RUN
? DÉSIGNATION
DÉSIGNATION

3. INSTRUCTIONS DE CONVERSION

Dans un programme, on a parfois besoin d'exploiter une chaîne de caractères comme une variable numérique ou, inversement, d'exploiter une valeur numérique comme une variable alphanumérique.

Il faut donc pouvoir convertir les variables d'une forme dans une autre.

a) VAL

Format VAL (A\$)

Rôle Convertit une chaîne de caractères en variable numérique.

REMARQUE. Le contenu de la chaîne doit correspondre à la forme utilisable d'une valeur numérique. Par exemple, la chaîne ne devra pas commencer par un blanc :

VAL (" - 435.27 ") est incorrect à cause du blanc ;
par contre VAL (" - 28.236 ") est correct.

Exemple

Supposons que, connaissant la date exacte d'achat, on veuille la date exacte d'échéance pour un achat à 30 jours.

On appelle D\$ la date d'achat et elle est donnée sous la forme XX/XX/XXXX. Le caractère de séparation / l'empêche d'être une zone numérique.

```
1Ø INPUT " DATE D'ACHAT ="; D$
2Ø Y$ = MID$(D$, 4, 2)
3Ø Y = VAL(Y$) : PRINT " Y = "; Y
4Ø Z = Y + 3 : PRINT " Z = "; Z
5Ø PRINT LEFT$(D$, 3) ; Z ; RIGHT$(D$, 5)
RUN
DATE D'ACHAT = ? 12/Ø6/1979
Y = Ø6
Z = Ø9
12/Ø9/1979
```

N.B. Cet exemple n'est vrai que pour des achats effectués avant le mois d'octobre.

b) STR\$

Format

STR\$(X)

X = expression numérique.

Rôle

Convertit une valeur numérique en une chaîne de caractères.

Exemple

```
1Ø INPUT X
2Ø Y$ = STR$(X) : PRINT " Y$ = "; Y$
3Ø C$ = LEFT$(Y$, 3)
4Ø PRINT " C$ = "; C$
RUN
? 12345
Y$ = 12345
C$ = 123
```

c) ASC

Format **ASC (A\$)**

Rôle Cette fonction donne un nombre égal à la valeur ASCII (exprimée en décimal) du premier caractère de la chaîne A\$.

Exemple 1Ø A\$ = " BASIC "
2Ø X = ASC (A\$)
3Ø PRINT " X = "; X
RUN
X = 66

d) CHR\$

Format **CHR\$ (X)**

X = code ASCII exprimé en décimal.
Ø ≤ X ≤ 255

Rôle Cette fonction donne un caractère alphanumérique dont le code ASCII est égal à X.

Exemple 1Ø INPUT X
2Ø PRINT " CHR\$ (X) = "; CHR\$ (X)
3Ø GOTO 1Ø
RUN
? 63
CHR\$ (X) = ?
? 66
CHR\$ (X) = B
? 36
CHR\$ (X) = \$
? 51
CHR\$ (X) = 3
?

4. OPÉRATIONS SUR LES CHAINES DE CARACTÈRES

1. Concaténation

On peut relier des constantes ou variables alphanumériques afin de constituer une nouvelle chaîne unique.

Exemples

- 1Ø A\$ = " BONJOUR "
- 2Ø B\$ = " MADAME "
- 3Ø D\$ = A\$ + " " + B\$
- 4Ø PRINT D\$
- RUN
- BONJOUR MADAME**

- 1Ø INPUT " ADRESSE CONNUE = ", AD\$
- 2Ø Y\$ = LEFT\$(AD\$, 2)
- 3Ø Y = VAL(Y\$) : PRINT " Y = "; Y
- 4Ø Z = Y + (3 * 2) : PRINT " Z = "; Z
- 5Ø N\$ = STR\$(Z)
- 6Ø A\$ = N\$ + RIGHT\$(AD\$, (LEN(AD\$) - 2))
- 7Ø PRINT " NOUVELLE ADRESSE : " ; A\$
- RUN
- ADRESSE CONNUE = ? 18 RUE DE LA RÉPUBLIQUE
- Y = 18
- Z = 24
- NOUVELLE ADRESSE : 24 RUE DE LA RÉPUBLIQUE

2. Comparaison de chaînes de caractères

Il est possible de comparer deux chaînes de caractères entre elles.

La comparaison se fait caractère par caractère en fonction du code ASCII de chacun de ces caractères, par exemple :

" GASTON " < " GÉRARD " .

Tous les symboles de relations logiques sont utilisables avec les chaînes de caractères.

EXEMPLE 8

```

JNEW
J10 PRINT " CE PROGRAMME COMPTE LES VOYELLES "
J20 PRINT " D'UNE CHAINE DE CARACTERES "
J30 PRINT
J40 PRINT " ECRIVEZ UN MOT OU UNE PHRASE "
J50 INPUT A$
J60 FOR I = 1 TO LEN (A$)
J65 B$ = MID$ (A$, I, 1)
J70 IF B$ = " A " OR B$ = " E " OR B$ = " I " OR B$* = " O " OR B$ = " U " OR B$ = " Y " THEN V = V + 1
J80 NEXT I
J90 PRINT " NOMBRE DE VOYELLES =  = ", V
J100 V = 0
J110 GOTO 30

```

erreur

```

JRUN
CE PROGRAMME COMPTE LES VOYELLES
D'UNE CHAINE DE CARACTERES
ECRIVEZ UN MOT OU UNE PHRASE
?LE LANGUAGE BASIC
?TYPE MISMATCH ERROR IN 70 ←

```

J?Ø IF B\$ = "A" OR B\$ = "E" OR B\$ = "I" OR B\$ = "O" OR B\$ = "U" OR B\$ = "Y" THEN V = V + 1

IRUN

CE PROGRAMME COMPTE LES VOYELLES
D'UNE CHAÎNE DE CARACTÈRES

ÉCRIVEZ UN MOT OU UNE PHRASE

?LE LANGAGE BASIC

NOMBRE DE VOYELLES = 6

ÉCRIVEZ UN MOT OU UNE PHRASE

?INFORMATIQUE

NOMBRE DE VOYELLES = 6

ÉCRIVEZ UN MOT OU UNE PHRASE

?A BIENTÔT

NOMBRE DE VOYELLES = 4

ÉCRIVEZ UN MOT OU UNE PHRASE

EXERCICES

1. En introduisant chaque mot d'une phrase par un ordre `INPUT`, écrire l'ordre `PRINT` permettant de reconstituer cette phrase.
2. Écrire le programme `BASIC` qui permet de compter le nombre de fois où un mot donné apparaît dans un texte.
3. Sachant qu'on ne peut pas insérer de virgule dans un texte introduit en mémoire par `INPUT`, trouver la méthode qui permet de remplacer la virgule par le symbole `#` en entrée et de faire apparaître cependant la virgule en sortie.

listes et tables

1. LES LISTES

1. Présentation

Supposons que, dans un programme de calcul de statistiques, par exemple, on utilise les nombres de jours de chaque mois.

Nous aurons donc :

N1	=	31	pour	JANVIER
N2	=	28	pour	FÉVRIER
⋮				
N11	=	30	pour	NOVEMBRE
N12	=	31	pour	DÉCEMBRE

Dans le même programme, on peut également trouver le total des ventes pendant chaque mois, c'est-à-dire T1, T2... T12. Ainsi il est possible de calculer la moyenne des ventes quotidiennes pour chaque mois :

M1	=	T1 / N1		M11	=	T11 / N11
M2	=	T2 / N2		M12	=	T12 / N12

En BASIC, on simplifie l'écriture de ces calculs en représentant les variables N1 à N12 par un seul nom, de même pour T1 à T12 et M1 à M12.

On appellera donc N l'ensemble des 12 valeurs N1... N12.

N désigne donc 12 positions consécutives de la mémoire, numérotées de 1 en 1, de Ø à 11.

N représente donc une liste de valeurs et s'appelle **variable de liste**. Afin de pouvoir utiliser l'une de ces 12 valeurs, la machine doit savoir exactement où elle se trouve. Il faut donc, avant l'exécution du programme, réserver en mémoire la place nécessaire à toutes les valeurs.

2. Réserve

Elle se fait au moyen de l'instruction DIM.

- DIM

Format 1 n DIM A (X)
A = nom de la variable
X = dimension de la variable de liste.

Rôle Cette instruction définit le nombre d'éléments d'une variable de liste.

Exemple 1Ø DIM N (11) définit la variable de liste N
comme l'ensemble de 12 valeurs N (Ø), N (1)... N (11).

REMARQUES

1. L'instruction DIM peut être placée n'importe où dans un programme.

2. Toutes les valeurs de la variable de liste sont initialisées à zéro par l'instruction DIM, lorsque la variable est numérique.

3. La dimension de la variable de liste peut être une expression arithmétique, fonction d'une variable.

Exemple : 1Ø DIM Z (2 * Y) où Y est une variable.

4. La variable de liste peut être alphanumérique.

Exemple : 4Ø DIM A \$ (12) définit un ensemble de 13 valeurs alphanumériques A \$ (Ø), A \$ (1)... A \$ (12).

5. Lorsqu'une variable de liste n'est pas définie par l'instruction DIM, elle est au maximum de dimension 1Ø.

3. Utilisation

Toutes les places étant réservées en mémoire par l'instruction DIM, il est possible d'utiliser chacune des valeurs de la liste en indiquant la variable de liste.

Exemple

$N(4) = 31$, car $N(4)$ représente la 5^e valeur de la liste N . C'est donc le nombre de jours du mois de mai, soit 31.

REMARQUES

1. L'indice est toujours placé entre parenthèses.
2. L'indice peut être une expression mathématique.

Exemple : $V(2 * X + t)$, où x et t sont des variables numériques.

4. Exemple

Moyenne des ventes mensuelles :

```
10 DIM N (11)
20 DIM T (11) : DIM M (11)
30 FOR I = 0 TO 11
40 READ N (I), T (I)
50 NEXT I
60 FOR I = 0 TO 11
70 M (I) = T (I) / N (I)
80 PRINT " MOYENNE - " ; I ; " = " ; M (I)
90 NEXT I
100 DATA 31, 1000, 28, 800, 31, 1200, 30, 1500,
31, 1100, 30, 1500, 31, 950, 31, 800, 30, 1200,
31, 1000, 30, 1100, 31, 1500
```

```
RUN
MOYENNE - 0 = 32.2581
MOYENNE - 1 = 28.5714
MOYENNE - 2 = 38.7096
MOYENNE - 3 = 50
MOYENNE - 4 = 35.4838
MOYENNE - 5 = 50
MOYENNE - 6 = 30.6452
MOYENNE - 7 = 25.8064
MOYENNE - 8 = 40
MOYENNE - 9 = 32.2581
MOYENNE - 10 = 36.6666
MOYENNE - 11 = 48.3871
```

2. LES TABLES

1. Présentation

En BASIC, on peut également décrire et utiliser des tableaux.

Considérons une société qui produit cinq catégories d'articles dans quatre usines différentes.

Le tableau suivant donne les quantités fabriquées de chaque catégorie d'articles par usine, pendant un mois.

Usine \ Article	1	2	3	4
1	150	140	155	160
2	100	110	105	103
3	95	97	100	100
4	105	100	102	105
5	89	95	90	85

Ce tableau contient 20 valeurs (5 lignes de 4 nombres).

On peut le décrire en BASIC par une seule variable appelée **variable de tableau**.

2. Réservation

Elle se fait également par l'instruction DIM.

- | |
|-----|
| DIM |
|-----|

Format 2 n DIM A (X, Y)

A = le nom de la variable

X et Y = les dimensions du tableau

qui comporte : X + 1 lignes de 0 à X et

Y + 1 colonnes de 0 à Y

Rôle Définit la dimension de la variable de tableau et réserve la place mémoire nécessaire.

- DIM A (X, Y) réserve une place en mémoire pour $(X + 1) * (Y + 1)$ éléments en commençant par A (Ø, Ø).

Le tableau ainsi défini comporte donc X + 1 lignes de Y + 1 valeurs, soit $(X + 1) * (Y + 1)$ éléments.

Chacun de ces éléments est repéré par deux indices, le premier pour la ligne et le second pour la colonne.

Par exemple, dans le tableau présenté ci-dessus en 1., l'élément

- A (Ø, Ø) a la valeur 150
- A (2, 3) a la valeur 100
- A (1, 2) a la valeur 105
- A (3, 2) a la valeur 102..., etc.

Sur certains matériels, on peut utiliser les **tableaux de chaînes**, définis par l'instruction `n DIM A$(X, Y)`.

- DIM A\$(X, Y) réserve une place en mémoire pour $(X + 1) * (Y + 1)$ éléments de chaîne, chacun de 255 caractères au maximum.

REMARQUES.

1. Mêmes remarques que pour la variable de liste.
2. Lorsqu'une variable de tableau n'est pas définie par DIM, elle contient au maximum 11 lignes et 11 colonnes.

3. Utilisation

Pour utiliser individuellement une des valeurs d'un tableau, il faut faire suivre la variable de tableau de deux indices, l'un pour la ligne, et l'autre pour la colonne.

Exemple : A (3, 4) représente l'élément du tableau A qui se trouve sur la 4^e ligne et dans la 5^e colonne.

4. Exemple

Fabrication totale d'articles :

```
1Ø DIM Q (4, 3)
2Ø FOR I = Ø TO 4
3Ø FOR J = Ø TO 3
4Ø READ Q (I, J)
5Ø NEXT J
6Ø NEXT I
65 DATA 15Ø, 14Ø, 155, 16Ø, 1ØØ, 11Ø, 1Ø5, 1Ø3, 95,
97, 1ØØ, 1ØØ, 1Ø5, 1ØØ, 1Ø2, 1Ø5, 89, 95, 9Ø, 85
7Ø FOR I = Ø TO 4
8Ø FOR J = Ø TO 3
9Ø QT (I) = QT (I) + Q (I, J)
1ØØ NEXT J
11Ø PRINT " QUANTITÉ ARTICLE "; I + 1; " = "; QT (I)
12Ø NEXT I
```

```
RUN
QUANTITÉ ARTICLE 1 = 6Ø5
QUANTITÉ ARTICLE 2 = 418
QUANTITÉ ARTICLE 3 = 392
QUANTITÉ ARTICLE 4 = 412
QUANTITÉ ARTICLE 5 = 359
```

EXEMPLE 9

Table de valeurs financières $V_n = (1 + i)^n$
où N varie de 1 à 2Ø
et i varie de 1 % à 7 %

```
2 PRINT "TABLE DE VALEURS FINANCIERES"
5 DIM V(2Ø,7)
6 FOR I=1 TO 7
7 PRINT TAB(10*I)I;
8 NEXT I
1Ø FOR N=1 TO 2Ø
15 PRINT TAB(5)N;
2Ø FOR I=1 TO 7
3Ø V (N,I)=(1+I/1ØØ) ^ N
4Ø PRINT TAB(1Ø*I)V(N,I);
5Ø NEXT I
6Ø NEXT N
7Ø END
```

Table de valeurs financières (2)

1	1.01	1.02	1.03	1.04	1.05	1.06	1.07
2	1.0201	1.0404	1.0609	1.0816	1.1025	1.1236	1.1449
3	1.0303	1.06121	1.09273	1.12486	1.15763	1.19102	1.22504
4	1.0406	1.08243	1.12551	1.16986	1.21551	1.26248	1.3108
5	1.05101	1.10408	1.15927	1.21665	1.27628	1.33823	1.40255
6	1.06152	1.12616	1.19405	1.26532	1.3401	1.41852	1.50073
7	1.07214	1.14869	1.22987	1.31593	1.4071	1.50363	1.60578
8	1.08286	1.17166	1.26677	1.36857	1.47746	1.59385	1.71819
9	1.09369	1.1951	1.30477	1.42331	1.55133	1.68948	1.83846
10	1.10462	1.219	1.34392	1.48024	1.6289	1.79085	1.96716
11	1.11567	1.24338	1.38423	1.53945	1.71034	1.8983	2.10486
12	1.12682	1.26825	1.42576	1.60103	1.79586	2.0122	2.2522
13	1.13809	1.29361	1.46853	1.66507	1.88565	2.13293	2.40985
14	1.14947	1.31948	1.51259	1.73168	1.97994	2.26091	2.57854
15	1.16097	1.34587	1.55797	1.80094	2.07893	2.39657	2.75904
16	1.17258	1.37279	1.60471	1.87298	2.18288	2.54036	2.95217
17	1.1843	1.40025	1.65285	1.9479	2.29203	2.69278	3.15883
18	1.19615	1.42825	1.70243	2.02582	2.40663	2.85435	3.37994
19	1.20811	1.45682	1.75351	2.10685	2.52696	3.02561	3.61654
20	1.22019	1.48595	1.80611	2.19112	2.65331	3.20715	3.8697

EXERCICES

1. On donne deux variables de liste S1 et S2.
Chercher les éléments de même rang, égaux dans les deux listes, et les ranger dans une troisième liste.
Écrire le programme BASIC correspondant.
2. Écrire le programme qui permet de faire la somme de deux matrices A et B (ou de deux tableaux de mêmes dimensions).

séquences d'exécution

1. FIN DE PROGRAMME

- END

Format *n* END

Rôle Termine l'exécution du programme.

L'instruction END n'est pas obligatoire. Le programme s'arrête quand il n'y a plus d'instruction à exécuter.

Cependant, elle est intéressante pour « isoler » une zone de sous-programme par exemple.

Exemples

- sans END

```
10 GOSUB 50
20 PRINT " PROGRAMME PRINCIPAL "

50 PRINT " SOUS-PROGRAMME "
60 RETURN

RUN
SOUS-PROGRAMME
PROGRAMME PRINCIPAL
SOUS-PROGRAMME
```

- avec END

```
1Ø GOSUB 5Ø
2Ø PRINT " PROGRAMME PRINCIPAL "
3Ø END

5Ø PRINT " SOUS-PROGRAMME "
6Ø RETURN
```

```
RUN
SOUS-PROGRAMME
PROGRAMME PRINCIPAL
```

2. COMMENTAIRES

- | |
|-----|
| REM |
|-----|

Format n REM commentaires

Rôle Permet d'écrire des commentaires dans un programme BASIC.

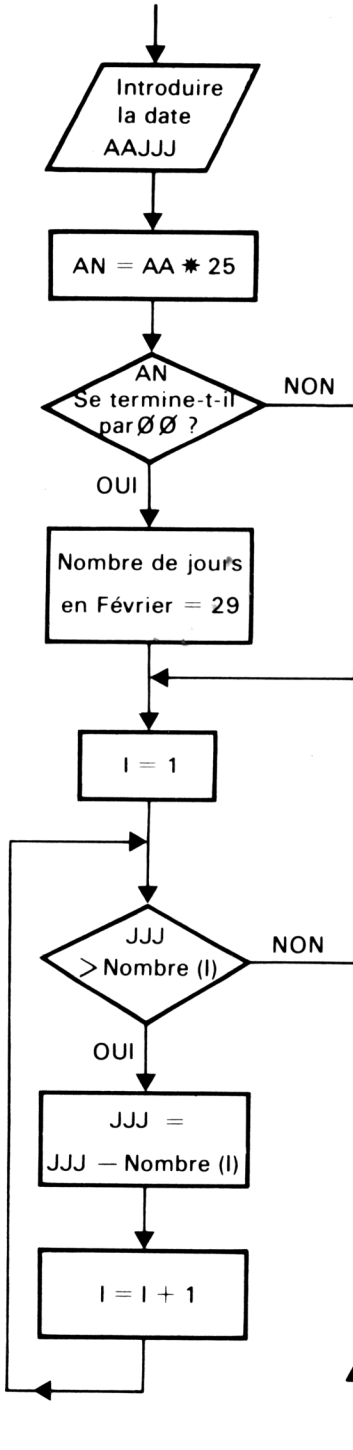
Ces commentaires servent à expliciter le programme. Ils ne sont pas exécutés (ils n'apparaissent donc pas après RUN).

Exemple

```
1Ø REM... CE PROGRAMME CALCULE LA SOMME
   DE DEUX NOMBRES RÉELS...
2Ø INPUT X : REM ** PREMIER NOMBRE RÉEL **
3Ø INPUT Y : REM ** DEUXIÈME NOMBRE RÉEL **
4Ø Z = X + Y : REM // CALCUL //
5Ø PRINT " Z = X + Y = "; Z
6Ø GOTO 2Ø
```

```
RUN
? 5.3621
? 4.2358
Z = X + Y = 9.5979
?
```

TRADUCTION D'UNE DATE



On tape au clavier une date sous la forme numérique suivante :

AAJJJ (79220 par exemple)
où AA est l'année (79, 80, 67...)
et JJJ est le numéro du jour dans l'année (220, 010, ...).

La machine doit donner la date correspondante sous la forme usuelle JJ MOIS 19AA (23 JANVIER 1980 par exemple).

ORGANIGRAMME

Les noms des mois et les nombres de jours par mois sont dans des tables en mémoire.

PROGRAMME BASIC

```
J10 REM ***CE PROGRAMME TRADUIT UNE DATE INTRODUITE SOUS
FORME NUMERIQUE
J20 REM ***A 5 CHIFFRES (2 POUR L'ANNEE ET 3 POUR LE NUMERO DU
JOUR)
J30 REM ***SOUS LA FORME TRADITIONNELLE : NN MOIS 19XX
J40 REM ***EXEMPLE : 77023 JANVIER 1977
J50 REM ***CHARGEMENT DE LA TABLE A TABLE DES NOMBRES DE
JOURS DES MOIS ***
J60 PRINT "INTRODUISEZ LES LONGUEURS EN JOURS DE CHAQUE
MOIS DE L'ANNEE "
J70 PRINT "FAITES 'RETURN' APRES CHACUN "
J80 DIM A(12)
J90 FOR I=1 TO 12
J100 INPUT A(I)
J110 NEXT I
J120 REM ***FIN DE CHARGEMENT DE LA TABLE 1***
J130 REM <<<< CHARGEMENT DE LA TABLE DES MOIS <<<<
J140 DIM B$(12)
J150 FOR J=1 TO 12
J160 READ B$(J)
J170 NEXT J
J180 DATA " JANVIER ", " FÉVRIER ", " MARS ", " AVRIL ", " MAI ", " JUIN ",
" JUILLET ", " AOÛT ", " SEPTEMBRE ", " OCTOBRE ", " NOVEMBRE ", " DÉCEMBRE "
J190 REM <<<< CHARGEMENT TABLE 2<<<<
J200 I=1
J210 PRINT "INTRODUISEZ UNE DATE NUMERIQUE "
J220 INPUT X$
J230 C$=LEFT$(X$,2)
J240 AN=VAL(C$)* 25
J250 E$=STR$(AN)
J260 IF RIGHT$(E$,2)="00" THEN A(2)=29: GOTO 280
J270 A(2)=28
J280 JO=VAL(RIGHT$(X$,3))
J290 IF JO>A(I) THEN 320
J300 PRINT X$ ; " CORRESPOND AU : "; JO ; " "; B$ (I) ; " !19 "; C$
J310 GOTO 200
J320 JO=JO-A(I)
J330 I=I+1
J340 GOTO 290
```

JRUN

INTRODUISEZ LES LONGUEURS EN JOURS DE CHAQUE MOIS DE L'ANNÉE
FAITES 'RETURN' APRES CHACUN

?31

?28

?31

?30

?31
?30
?31
?31
?30
?31
?30
?31

INTRODUISEZ UNE DATE NUMERIQUE

?79024

79024CORRESPOND AU :24 JANVIER

?SYNTAX ERROR IN 300

J300 PRINT X\$;" CORRESPOND AU :"; JO;" "; B\$(I)); "" 19"; C\$

JRUN

INTRODUISEZ LES LONGUEURS EN JOURS DE CHAQUE MOIS DE L'ANNEE

FAITES 'RETURN' APRES CHACUN

?31
?28
?31
?30
?31
?30
?31
?31
?30
?31
?30
?31

INTRODUISEZ UNE DATE NUMERIQUE

?79030

79030CORRESPOND AU:30 JANVIER 1979

INTRODUISEZ UNE DATE NUMERIQUE

?80060

80060CORRESPOND AU:29 FEVRIER 1980

INTRODUISEZ UNE DATE NUMERIQUE

?79060

79060CORRESPOND AU:1 MARS 1979

INTRODUISEZ UNE DATE NUMERIQUE

?64061

64061CORRESPOND AU:1 MARS 1964

INTRODUISEZ UNE DATE NUMERIQUE

?77230

77230CORRESPOND AU:18 AOUT 1977

INTRODUISEZ UNE DATE NUMERIQUE

?65365

65365CORRESPOND AU:31 DECEMBRE 1965

INTRODUISEZ UNE DATE NUMERIQUE

les fichiers

1. PRÉSENTATION

BASIC est le langage utilisé surtout sur les micro-ordinateurs. Par conséquent, nous ne citerons dans ce chapitre que les fichiers sur disquette.

Un fichier est un ensemble d'éléments de même nature, placés les uns à la suite des autres.

Chaque élément ou article du fichier est lui-même un ensemble d'informations le caractérisant.

Les principaux traitements sur les fichiers sont les suivants :

- création du fichier
- consultation du fichier
- fin du fichier
- mise à jour du fichier
 - adjonction d'un ou plusieurs articles
 - suppression d'un ou plusieurs articles
 - modification d'un ou plusieurs articles

Afin de réaliser ces divers traitements, il faut connaître parfaitement les caractéristiques du fichier concerné et en particulier son organisation.

Cette organisation peut être de deux types :

- séquentielle,
- à accès direct.

2. UTILISATION DES FICHIERS EN BASIC

Toutes les opérations sur fichiers se font en BASIC par des instructions, qu'il n'est pas possible de détailler car elles dépendent essentiellement des systèmes.

Cependant, quel que soit le système, un certain ordre doit être respecté dans l'exécution de ces instructions.

1. Fichiers séquentiels

a) *Ouverture du fichier.* En général, l'instruction correspondante est l'ordre **OPEN** qui doit préciser :

- si le fichier est en *entrée* (INPUT) ou en *sortie* (OUTPUT),
- le nom du fichier,
- d'autres paramètres qui diffèrent selon les machines.

b) *Écriture des données.* En général, l'instruction correspondante est l'ordre **PRINT** ou **WRITE** qui doit préciser :

- sur quel fichier on écrit,
- ce que l'on écrit.

c) *Lecture des données.* En général, l'instruction correspondante est l'ordre **INPUT** ou **READ** qui doit préciser :

- le fichier que l'on lit,
- les données que l'on lit.

d) *Fermeture du fichier.* En général, l'instruction correspondante est l'ordre **CLOSE** qui doit préciser le fichier à fermer.

2. Fichiers à accès direct (ou Random)

L'organisation Random d'un fichier permet d'accéder directement et rapidement à n'importe quel enregistrement du fichier. Contrairement à l'organisation séquentielle, qui oblige l'utilisateur à examiner tout le fichier depuis le début jusqu'à la rencontre de l'information recherchée.

a) *Ouverture du fichier.* **OPEN**, qui doit préciser que le fichier est en Random. En effet, on peut aussi bien utiliser ce fichier en *entrée* qu'en *sortie*, c'est-à-dire qu'on peut alterner « lecture » et « écriture » au cours du même traitement.

b) *Écriture des données.* En général, ordre **PUT**.

c) *Lecture des données.* En général, ordre **GET**.

PUT et **GET** doivent préciser :

- le fichier concerné,
- l'enregistrement concerné (éventuellement).

d) *Fermeture du fichier.* En général, ordre **CLOSE**, comme pour les fichiers séquentiels.

EXEMPLE

ÉCRITURE ET LECTURE D'UN FICHIER RANDOM

FICHIER CLIENTS

simplifié au maximum puisqu'il ne comporte
que les noms et prénoms des clients
(20 caractères pour le nom, et 15 pour le prénom)

```
5  REM ... ECRITURE ...
10 OPEN "R", # 1, "CLIENTS", Ø
20 REM *** R SIGNIFIE RANDOM
22 REM *** # 1 SIGNIFIE QUE LE FICHIER EST ASSOCIÉ AU CANAL 1
24 REM *** Ø SIGNIFIE QUE LA DISQUETTE EST SUR L'UNITÉ Ø
30 FIELD # 1, 20 AS NOM $, 15 ASPREN $
32 REM /// CETTE INSTRUCTION ASSIGNE LES 20 PREMIERS CARACTÈRES
34 REM /// DE LA MÉMOIRE TAMPON ASSOCIÉE AU FICHIER A LA VARIABLE
36 REM /// ALPHANUMÉRIQUE NOM $, LES 15 SUIVANTS A LA VARIABLE
38 REM /// ALPHANUMÉRIQUE PREN $
40 INPUT "NOM ET PRÉNOM DU CLIENT : ", N$, P$
50 IF N$ = "FIN" THEN 100
60 LSET NOM $ = N$
62 REM >>> CETTE INSTRUCTION PERMET DE CADRER A GAUCHE
64 REM >>> LA VALEUR DE N$ DANS LA ZONE NOM$
70 LSET PRENOM $ = P$
80 PUT # 1
82 REM --- CETTE INSTRUCTION PERMET D'ÉCRIRE LE CONTENU
84 REM --- DE LA ZONE TAMPON DANS LE FICHIER
90 GOTO 40
100 CLOSE #1
105 REM ... FIN ECRITURE ...
107 PRINT : PRINT
110 REM ... LECTURE ...
```

```

115 OPEN "R", # 1, "CLIENTS", Ø
120 FIELD # 1, 20 ASNOM $, 15 ASPREN $
130 FOR N = 1 TO 10
140 GET # 1, N
145 REM *** IL Y A 10 CLIENTS DANS LE FICHER ***
150 PRINT N, NOM $, PREN $
160 NEXT N
170 CLOSE # 1
180 REM * > * ICI LA LECTURE SE FAIT DE FAÇON
181 REM * > * SEQUENTIELLE
182 REM * > * POUR LIRE UN ENREGISTREMENT PARTICULIER
183 REM * > * DONT ON CONNAIT LE NUMÉRO, ON UTILISERAIT
185 REM * > * L'INSTRUCTION GET AVEC LE NUMÉRO CORRES-
187 REM * > * PONDANT : GET # 1, 6 LIT LE SIXIÈME
189 REM * > * ENREGISTREMENT DU FICHER
190 REM ... FIN LECTURE ...
200 END

```

RUN

```

NOM ET PRENOM DU CLIENT : ? ANATOLE, DANIEL
NOM ET PRENOM DU CLIENT : ? ANDRE, MICHEL
NOM ET PRENOM DU CLIENT : ? BARBET, JEAN
NOM ET PRENOM DU CLIENT : ? BAROT, PIERRE
NOM ET PRENOM DU CLIENT : ? BARQUE, JACQUES
NOM ET PRENOM DU CLIENT : ? CADOT, VINCENT
NOM ET PRENOM DU CLIENT : ? CAZE, MARC
NOM ET PRENOM DU CLIENT : ? DUPONT, YVES
NOM ET PRENOM DU CLIENT : ? DURAND, FRANÇOIS
NOM ET PRENOM DU CLIENT : ? FIN

```

1	ANATOLE	DANIEL
2	ANDRE	MICHEL
3	BARBET	JEAN
4	BAROT	PIERRE
5	BARQUE	JACQUES
6	CADOT	VINCENT
7	CAZE	MARC
8	DUPONT	YVES
9	DURAND	FRANÇOIS

MOTS RÉSERVÉS BASIC

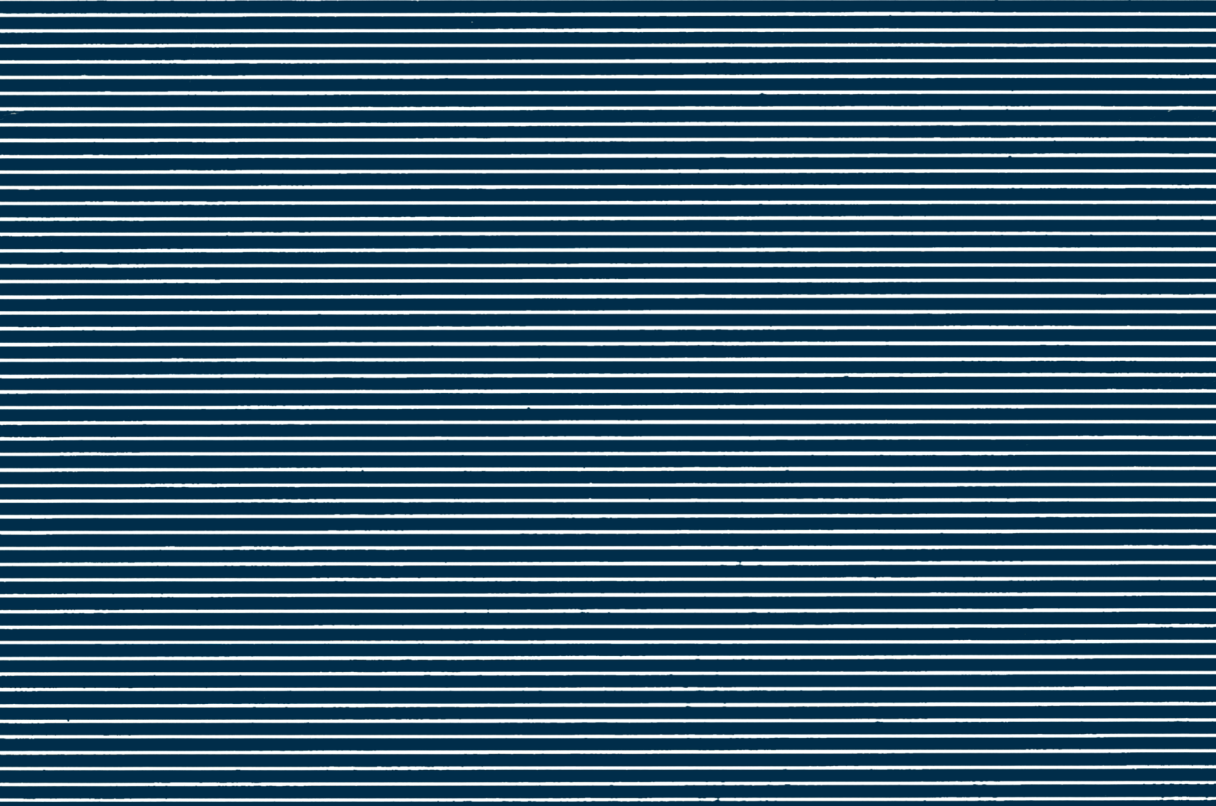
AND	LOAD
BASIC	NAME
CLOSE	NEW
DATA	NEXT
DEF	NOT
DELETE	ON
DIM	OPEN
EDIT	OR
ELSE	POKE
END	PRINT
ERROR	PUT
ERR	READ
FIELD	REM
FILES	RETURN
FN	RUN
FOR	SPACES
GET	STOP
GOSUB	STRING
GOTO	SWAPP
IF	TAB
INPUT	THEN
LET	TO
LIST	WAIT

ANNEXE 2

CODE ASCII

CARAC- TÈRE	HEXA- DÉCIMAL	DÉCIMAL
Espace	2-0	32
!	2-1	33
"	2-2	34
#	2-3	35
\$	2-4	36
%	2-5	37
&	2-6	38
'	2-7	39
(2-8	40
)	2-9	41
*	2-A	42
+	2-B	43
,	2-C	44
-	2-D	45
.	2-E	46
/	2-F	47
Ø	3-0	48
1	3-1	49
2	3-2	50
3	3-3	51
4	3-4	52
5	3-5	53
6	3-6	54
7	3-7	55
8	3-8	56
9	3-9	57
:	3-A	58
;	3-B	59
<	3-C	60
=	3-D	61
>	3-E	62
?	3-F	63
@	4-0	64
A	4-1	65
B	4-2	66
C	4-3	67
D	4-4	68
E	4-5	69
F	4-6	70
G	4-7	71
H	4-8	72
I	4-9	73
J	4-A	74
K	4-B	75
L	4-C	76
M	4-D	77

CARAC- TÈRE	HEXA- DÉCIMAL	DÉCIMAL
N	4-E	78
O	4-F	79
P	5-0	80
Q	5-1	81
R	5-2	82
S	5-3	83
T	5-4	84
U	5-5	85
V	5-6	86
W	5-7	87
X	5-8	88
Y	5-9	89
Z	5-A	90
[5-B	91
\	5-C	92
]	5-D	93
^	5-E	94
_	5-F	95
a	6-1	97
b	6-2	98
c	6-3	99
d	6-4	100
e	6-5	101
f	6-6	102
g	6-7	103
h	6-8	104
i	6-9	105
j	6-A	106
k	6-B	107
l	6-C	108
m	6-D	109
n	6-E	110
o	6-F	111
p	7-0	112
q	7-1	113
r	7-2	114
s	7-3	115
t	7-4	116
u	7-5	117
v	7-6	118
w	7-7	119
x	7-8	120
y	7-9	121
z	7-A	122
{	7-B	123
}	7-D	125



Basic Information

Personal Information

Education

Work Experience

Skills

References

Additional Information

Notes

Comments

History

Settings

Help

Logout

Admin

Footer

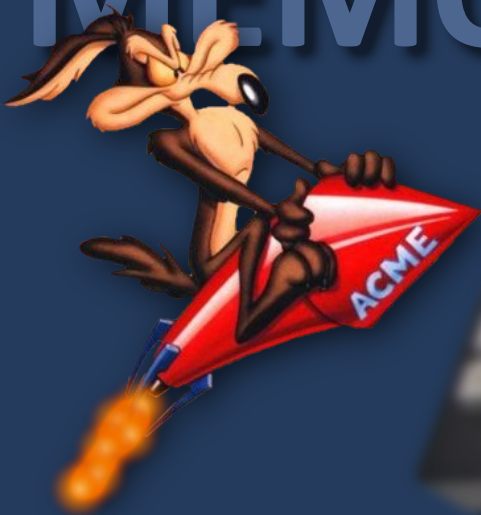


Document **numérisé**
avec amour par :

AMSTRAD

CPC 

MÉMOIRE ÉCRITE



<https://acpc.me/>