Les variables alphanumériques et leur traitement

Dans les chapitres précédents, nous avons traité essentiellement des variables numériques (c'est-à-dire des cases mémoires qui contiennent des nombres).

Pourtant, si nous voulons entrer dans l'ordinateur des textes quelconques (suite de lettres, de signes, de chiffres), il nous faudra travailler avec des variables dites alphanumériques.

Comment l'ordinateur les reconnaît-il ?
On leur donne un nom particulier dont les règles sont les mêmes que pour les variables numériques mais pour les distinguer, la dernière lettre du nom d'une variable alphanumérique est un $.

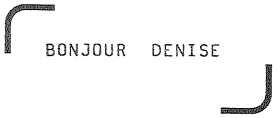
Exemple :

```
10 PRINT "ENTREZ VOTRE PRENOM"
20 INPUT N$
30 PRINT "BONJOUR";N$
```

. L'instruction 10 nous affiche le message "Entrez votre prénom"

. L'instruction 20 nous demande d'entrer un prénom. L'ordinateur stockera dans la variable N$ les caractères que nous allons taper au clavier. Supposons que nous tapions DENISE

- L'ordinateur affichera :



Une variable se terminant par $ peut également contenir des valeurs numériques.

Exemple :

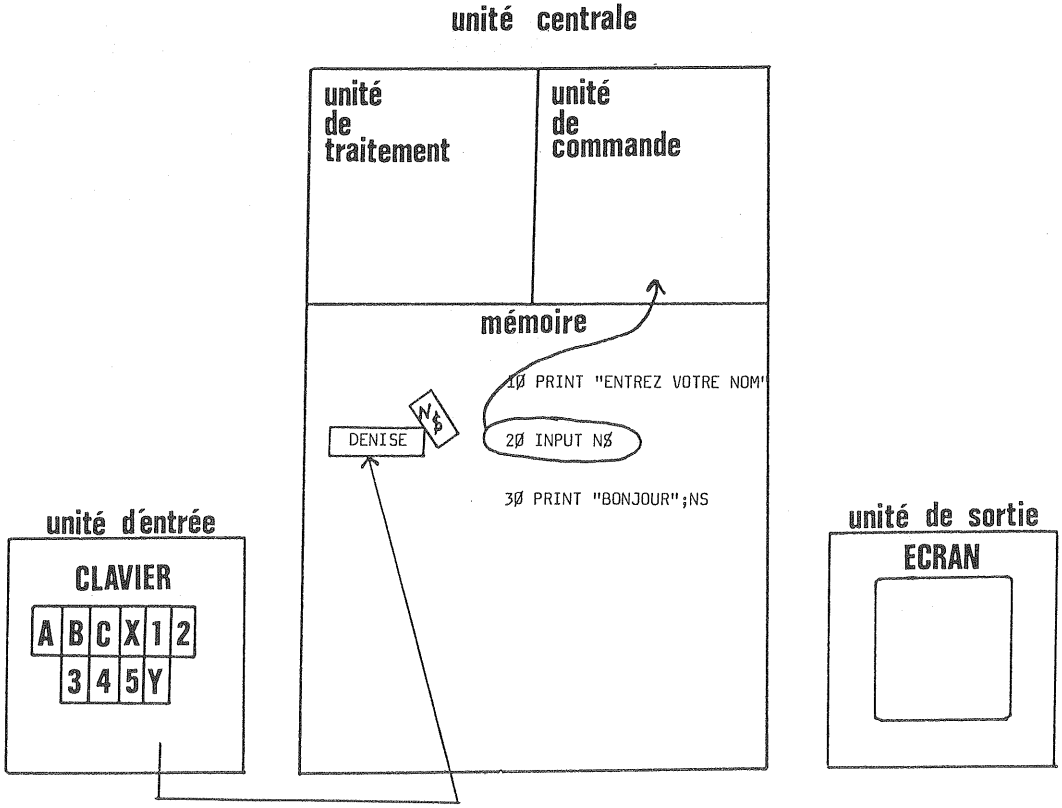
```
1Ø PRINT "DONNEZ LE CODE POSTAL ET LA VILLE"
2Ø INPUT V$
3Ø PRINT V$
```

Si à l'instruction 2Ø, nous avons tapé

67200 | ESPACE | STRASBOURG

La variable V$ contiendra les chiffres du code postal, le caractère [espace], le nom de la ville et nous affichera donc à l'instruction 3Ø exactement ce que nous avons saisi

soit 67200 | ESPACE | STRASBOURG


unité centrale
unité de traitement
unité de commande
mémoire
1Ø PRINT "ENTREZ VOTRE NOM"
N$
DENISE
2Ø INPUT N$
3Ø PRINT "BONJOUR";NS
unité d'entrée
CLAVIER
A B C X 1 2
3 4 5 Y
unité de sortie
ECRAN

Exercices :

```
1Ø PRINT "REPONDEZ PAR OUI OU NON"
2Ø PRINT"ETES-VOUS JOLI(E)"
3Ø INPUT J$
4Ø PRINT "ETES-VOUS RICHE"
5Ø INPUT R$
6Ø PRINT "ETES-VOUS INTELLIGENT (E)"
7Ø INPUT I$
8Ø IF J$="OUI" AND R$="OUI" AND I$="OUI" THEN PRINT
   "JE VEUX VOUS EPOUSER":END
9Ø PRINT "AU REVOIR"
```

```
1Ø PRINT "REPONDRE PAR OUI OU PAR NON"
2Ø PRINT "ETES-VOUS INTELLIGENT"
3Ø INPUT I$
4Ø IF I$="NON" THEN PRINT "LISEZ VITE CE MANUEL"
5Ø IF I$="OUI" THEN PRINT "VOUS AVEZ SUREMENT LU CE MANUEL"
```

Remarque concernant les exercices qui vont suivre :

Pour le bon déroulement des programmes proposés, tapez l'ordre suivant à chaque fois que vous allumez la machine :

CLEAR 600 [RETURN]

Cette instruction sera vue plus en détail dans le tome V.

Nous pouvons effectuer différents traitements sur une variable alphanumérique.

1) Isoler des caractères

à gauche :

la fonction LEFT$ nous permet d'extraire un ensemble de caractères placés à gauche d'une variable alphanumérique. Elle s'écrit LEFT$ (nom de la variable, longueur)

Exemple :
```
10 PRINT "CODE POSTAL ET VILLE"
20 INPUT V$
30 C$=LEFT$(V$,5)
40 PRINT C$
```

Si nous avons introduit 67200 [ESPACE] STRASBOURG

pour V$, C$ contiendra uniquement les 5 premiers caractères de V$, soit le code postal.

L'ordinateur affichera donc 67200 à la ligne 40.

à droite :

la fonction RIGHT$ nous permet d'extraire un ensemble de caractères placés à droite d'une variable alphanumérique. Elle s'écrit RIGHT$ (nom de la variable, longueur)

Exemple :
```
10 PRINT "CODE POSTAL ET VILLE"
20 INPUT V$
30 C$=RIGHT$(V$,5)
40 PRINT C$
```

Si nous avons introduit 67200 [ESPACE] STRASBOURG

pour V$, C$ contiendra les 5 derniers caractères de V$, soit BOURG

L'ordinateur affichera donc BOURG à la ligne 40.

au milieu :

la fonction MID$ nous permet d'extraire un ensemble de caractères placés à l'intérieur d'une variable alphanumérique.

Elle s'écrit MID$ (nom de la variable, position de début, longueur)

Exemple :
```
10 PRINT "CODE POSTAL ET VILLE"
20 INPUT V$
30 C$=MID$(V$,7,4)
40 PRINT C$
```

Si nous avons introduit 672ØØ | ESPACE | STRASBOURG

pour V$, C$ contiendra les 4 caractères positionnés derrière le 7e, ce dernier inclus soit STRA.

L'ordinateur affichera donc STRA à l'instruction 4Ø.

FONCTION LEN

La fonction LEN donne le nombre de caractères se trouvant dans une variable alphanumérique (LEN vient de LENGTH en anglais qui veut dire longueur).

Elle s'écrit LEN (nom de la variable)

Exemple :

```
1Ø PRINT "CODE POSTAL ET VILLE"
2Ø INPUT V$
3Ø C=LEN(V$)
4Ø PRINT C
```

Si nous avons introduit 672ØØ [ESPACE] STRASBOURG

pour V$, C prendra la valeur 16, car il y a 16 caractères dans la variable V$

C représente bien sûr un nombre !

C$=LEN(V$) est une expression fausse, la machine "se plante" et affiche une erreur de syntaxe.

Remarque : Lorsqu'il n'y a aucun caractère dans une variable alphanumérique, la longueur est égale à zéro, on dit que la chaîne de caractères est vide ; l'instruction V$="" par exemple met la variable V$ à vide !

CONCATENATION DE CHAINES DE CARACTERES .

Les variables alphanumériques peuvent "s'additionner" c'est-à-dire se coller l'une derrière l'autre.

Bien sûr, il ne s'agit pas d'une addition arithmétique.

Mais prenons un exemple :

```
10 X$="BON"
20 Y$="J"
30 Z$="OUR"
40 S$=X$+Y$+Z$
50 PRINT S$
```

Dans cet exemple, la variable S$="BONJOUR"

Evidemment, ce programme pourrait s'écrire plus simplement :

```
10 S$="BONJOUR"
20 PRINT S$
```

Vous venez de remarquer au passage (instruction 10) qu'il est possible de placer une valeur dans une variable alphanumérique :

alors que INPUT S$ placera dans S$ la chaine de caractères lus au clavier, S$="BONJOUR" place directement le mot BONJOUR dans la variable S$

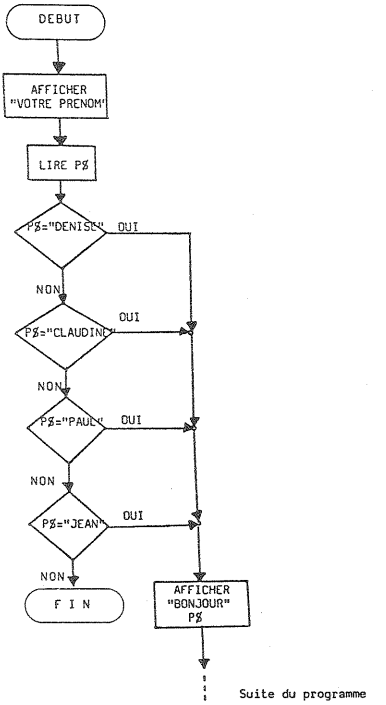
Tout de suite quelques exercices d'application !

PROGRAMME DE PROTECTION

On imagine un programme pour lequel l'exécution n'est accessible qu'à certaines personnes.

L'ordinateur demande le prénom (ou le code secret) de la personne qui veut travailler avec le programme. Si la personne n'est pas autorisée, le programme s'arrête ; si la personne est autorisée, l'ordinateur lui affiche bonjour et passe à la suite.

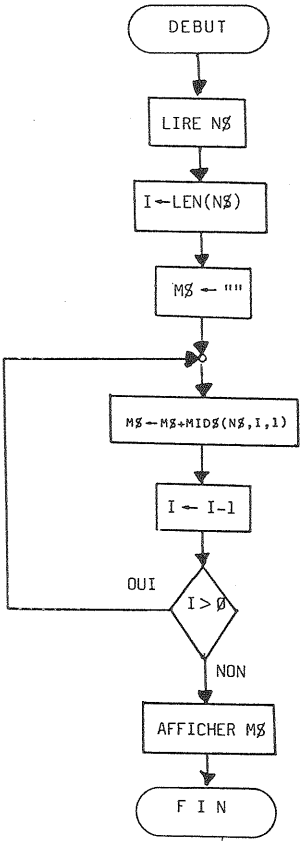
Organigramme :



```
1Ø PRINT "VOTRE PRENOM"
2Ø INPUT P$
3Ø IF P$="DENISE" THEN GOTO 9Ø
4Ø IF P$="CLAUDINE" THEN GOTO 9Ø
5Ø IF P$="PAUL" THEN GOTO 9Ø
6Ø IF P$="JEAN"THEN GOTO 9Ø
7Ø PRINT "TOP SECRET"
8Ø END
9Ø PRINT "BONJOUR";P$
1ØØ .
    .
    .          suite du programme
```

TROUVEZ DES PALINDROMES

Vous indiquez une phrase ou un mot à l'ordinateur qui doit vous l'afficher à l'envers.



```
1Ø PRINT "DONNEZ VOTRE PHRASE"
2Ø INPUT N$
3Ø I=LEN(N$)
4Ø M$=""
5Ø M$=M$+MID$(N$,I,1)
6Ø I=I-1
7Ø IF I > Ø THEN GOTO 5Ø
8Ø PRINT M$
9Ø END
```

Explications :

Instruction 1Ø : affiche le message

Instruction 2Ø : Introduction de la phrase ou du mot.
Supposons qu'on ait tapé : DENISE
Dans ce cas, la variable N$ vaut DENISE

Instruction 3Ø : I=LEN(N$) affecte à la variable I, la valeur 6

Instruction 4Ø : M$=""
La variable M$, dans laquelle se trouvera votre mot retourné, est mise à vide (c'est-à-dire qu'elle ne contient aucun caractère)

Instruction 5Ø : M$=M$+MID$(N$,I,1)
Sachant que M$ est vide
N$ vaut DENISE
et que I = 6

nous obtenons la nouvelle valeur de M$
M$="" + la lettre E soit M$ vaut E

Instruction 6Ø : I=I-1 donne I=5

Instruction 7Ø : Comme I est supérieur à Ø nous retournons à l'instruction 5Ø

Instruction 5Ø : M$=M$+MID$(N$,I,1)
Sachant que M$ vaut E
N$ toujours DENISE
et I=5

nous obtenons M$="E" + la lettre S
soit M$ vaut ES

Instruction 6Ø : I=I-1 donne I=4

Instruction 7Ø : Nous continuons ainsi jusqu'à ce que I=Ø
M$ prendra successivement les valeurs :

E

E S

E S I

E S I N

E S I N E

E S I N E D

Tapez les mots : ETE
KAYAK
ELUPARCETTECRAPULE

Les mots qui sont identiques lorsqu'on les retourne sont appelés palindromes.

Nous pourrions demander à l'ordinateur de nous signaler quelque chose lorsqu'il s'agit d'un palindrome.

Ainsi pourrions-nous ajouter une ligne :

```
85 IF M$=N$ THEN PRINT "PALINDROME"
```

Transformation des variables numériques ⟷ alphanumériques

Nous avons vu jusqu'à présent qu'il existe deux types de variables sur lesquelles on fait faire deux types de traitements différents

1) les variables numériques utilisées avec les calculs arithmétiques

2) les variables alphanumériques utilisées avec les traitements vus dans le paragraphe précédent.

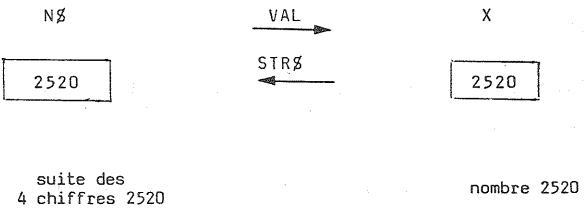
Transformer une variable numérique en alphanumérique ou l'inverse est parfois intéressant pour profiter des différents types de traitements.

L'instruction VAL permet de transformer une variable alphanumérique en nombre.

L'instruction STR$ permet de transformer une variable numérique en chaîne de caractères.

Exemples :

Si dans N$ nous avons la valeur 2520, il s'agit de la suite des 4 chiffres 2520. Si dans X nous avons 2520 il s'agit du nombre 2520



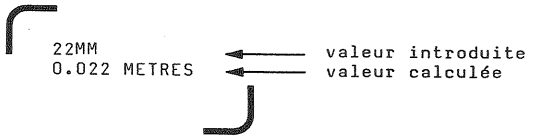
La syntaxe des instructions VAL et STR$ est la suivante :

VAL (nom de la variable alphanumérique)

STR$ (nom de la variable numérique)

Exemple : Conversion d'unités de longueur

Supposons que nous introduisons des chiffres et une unité de longueur et que l'ordinateur doive convertir les chiffres en mètre.



Il faut introduire une chaîne de caractères qui comprend des chiffres suivis de 2 lettres pour les unités.

```
10 INPUT A$
20 L=LEN(A$)
30 IF RIGHT$(A$,2)="MM"then X=0.0001*VAL(LEFT$(A$,L-2))
40 IF RIGHT$(A$,2)="CM"then X=0.001*VAL(LEFT$(A$,L-2))
50 IF RIGHT$(A$,2)="DM"then X=0.01*VAL(LEFT$(A$,L-2))
60 PRINT X;"METRES"
```

On pourra utiliser cet exemple pour des conversions de monnaie, unités de surfaces ...

EXERCICES

Essayez de savoir ce que font ces programmes :

```
1Ø  INPUT X          ——————→   Donnez un nombre de 3 chiffres
2Ø  X$ = STR$(X)
3Ø  X1$=MID$(X$,2,1)
4Ø  X2$=MID$(X$,3,1)
5Ø  X3$=MID$(X$,4,1)
6Ø  X1=VAL(X1$):X2=VAL(X2$):X3=VAL(X3$)
7Ø  X1=X1*1ØØ:X2=X2*1Ø
8Ø  PRINT X;"EST EGAL A:"
9Ø  PRINT X1;"+";X2;"+";X3
```

Au lieu de travailler sur des variables alphanumériques, on peut travailler directement sur les variables numériques :

```
1Ø  INPUT X                  Donnez un nombre de 3 chiffres
2Ø  X1=INT(X/1ØØ)
3Ø  X2=INT(X/1Ø)-(X1*1Ø)
4Ø  X3=X-((X1*1ØØ)+ X2*1Ø))
7P  X1=X1*1ØØ:X2=X2*1Ø
8Ø
9Ø    idem à l'exercice précédent
```

Que va afficher l'ordinateur ?

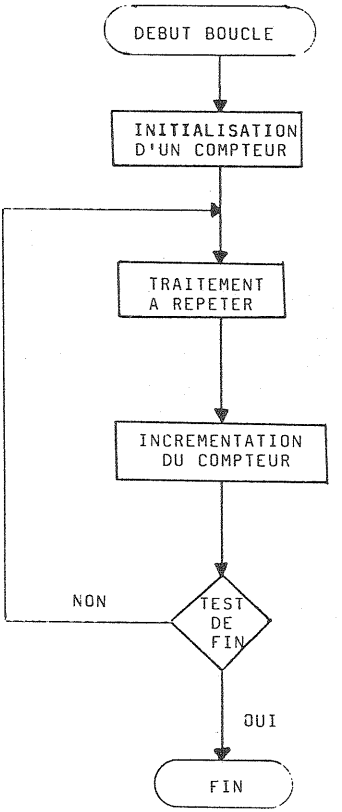
```
1Ø X$="BIERE"
2Ø Y$="TNT"
3Ø Z$="4Ø"
4Ø A1$=LEFT$(Y$,1)+RIGHT$(X$,2)
5Ø A2$=MID$(X$,1,3)
6Ø A3=(VAL(Z$)/2)
7Ø A3$=STR$(A3)
8Ø R$=A1$+"S "+A2$ +"N : "+A3$+"/"+A3$


9Ø PRINT R$
```

Boucle automatique

appelons quelques notions déjà vues sur les boucles.

Afin de répéter une série d'instructions, nous avions utilisé le principe de boucle, d'après le schéma suivant :



Pour permettre à une série d'instructions d'être exécutées un certain nombre de fois, le BASIC dispose de la boucle FOR... NEXT

Elle s'écrit :

```
FOR    variable = valeur initiale TO valeur finale
       compteur
                                STEP incrément

       :
       :
       :

NEXT   variable
```

Une variable est utilisée comme compteur.
La première valeur du compteur représente la valeur initiale.
Les lignes d'instructions après la ligne FOR sont exécutées jusqu'à l'instruction NEXT.
Le compteur est alors incrémenté de la valeur incrément (qui se trouve après STEP).
Si le compteur est différent de la valeur finale, l'ordinateur réexécute les instructions après FOR .... jusqu'à ce que le compteur soit égal ou supérieur à la valeur finale, moment où l'ordinateur exécute les instructions après NEXT.

Exemple :

```
10 FOR I=1 TO 10 STEP 2
20 PRINT I
30 NEXT I
```

Traduction littérale :

. pour I variant de 1 à 10 additioner 2 à I
. afficher I
. passer au I suivant

Remarques :

1) Si l'instruction STEP est omise, le pas d'incrémentation est automatiquement de +1

2) Le pas d'incrémentation peut être négatif

ex. :
```
1Ø FOR I = 1Ø TO 1 STEP-1
2Ø PRINT "BONJOUR"
3Ø NEXT I
```

Bien sûr dans ce cas, la valeur finale doit être inférieure à la valeur initiale.

3) A chaque instruction FOR doit correspondre une instruction NEXT

Exemple :

Reprenons l'exercice du tome III où nous devions afficher une table de multiplication X sur une longueur L.

Grâce à FOR NEXT le programme est réduit à :

```
1Ø PRINT "TABLE"
2Ø INPUT T
3Ø PRINT "LONGUEUR"
4Ø INPUT L
5Ø FOR I=1 TO L
6Ø X=I*T
7Ø PRINT I;"*";T;"=";X
8Ø NEXT I
9Ø END
```

Dans un programme, le nombre de boucles automatiques n'est pas limité.

Néanmoins, il faut tenir compte de certaines règles importantes.

```
10 FOR I = 1 TO 10
20  :
30  :   instructions à répéter I fois
40 NEXT I
50 FOR A = 1 TO 20
60  :
70  :   instructions à répéter A fois
80 NEXT A
```

est
autorisé

```
10 FOR I = 1 TO 10
20 FOR A = 1 TO 20
30  :
40  :   instructions à répéter A*I fois
50 NEXT A
60 NEXT I
```

est
autorisé

```
10 FOR I=1 TO 10
20 FOR A=1 TO 20
30  :
40  :
50 NEXT I
60 NEXT A
```

est
interdit

Les boucles n'ont pas le droit de se chevaucher. La première boucle ouverte doit être la dernière fermée.

Exercice

A votre avis, combien de fois le mot "Bonjour" sera t-il affiché à l'écran ?

```
1Ø FOR I=1 TO 5
2Ø FOR A=1 TO 10
3Ø PRINT "BONJOUR"
4Ø NEXT A
5Ø NEXT I
```

Pour I=1, la boucle FOR A va afficher 10 fois "Bonjour"
Pour I=2, la boucle FOR A va afficher 10 fois "Bonjour"
Pour I=3, la boucle FOR A va afficher 10 fois "Bonjour"

:

Le programme affichera donc au total 5 x 1Ø c'est-à-dire 5Ø fois le mot Bonjour.

Combien d'étoiles seront affichées à l'écran ?

```
1Ø FOR I=1 TO 5
2Ø FOR J=1 TO 10
3Ø FOR K=1 TO 8
4Ø PRINT "X";
5Ø NEXT K
6Ø NEXT J
7Ø NEXT I
```

8 fois X 10 fois x 5 fois

Afficher tous les nombres pairs de 2Ø à Ø

```
1Ø FOR N=2Ø TO Ø STEP-2
2Ø PRINT N
3Ø NEXT N
```

L'ordinateur affichera :

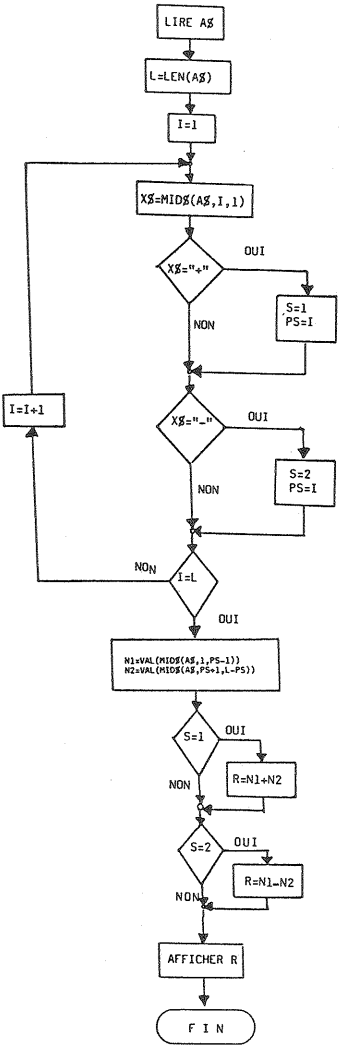
20
18
16
14
12
10
8
6
4
2
0

CALCUL AUTOMATIQUE

Jusqu'à présent, lorsque nous voulions faire la somme ou la différence entre 2 nombres, il fallait introduire un nombre après l'autre séparés par RETURN

Nous voulons à présent introduire ces 2 nombres sur une seule ligne et que l'ordinateur fasse automatiquement le calcul.

Si l'on introduit 2520-2025 par exemple, l'ordinateur affiche 495.
Si l'on introduit 143+5 par exemple, l'ordinateur affiche 148.


LIRE Ag
L=LEN(Ag)
I=1
Xg=MIDg(Ag,I,1)
Xg="+"
OUI
S=1
PS=I
NON
Xg="-"
OUI
S=2
PS=I
NON
I=L
NON
OUI
I=I+1
N1=VAL(MIDg(Ag,1,PS-1))
N2=VAL(MIDg(Ag,PS+1,L-PS))
S=1
OUI
R=N1+N2
NON
S=2
OUI
R=N1-N2
NON
AFFICHER R
F I N

A$ = l'expression à calculer

L = longueur de cette expression

I = compteur varie de 1 à L

X$ = élément de A$

S = 1 si somme
    2 si différence

PS = position du signe

N1 = 1er nombre

N2 = 2ème nombre

R = résultat du calcul

```
 1Ø INPUT A$
 2Ø L=LEN(A$)
 3Ø FOR I=1 TO L
 4Ø X$=MID$(A$,I,1)
 5Ø IF X$="+" THEN S=1:PS=I
 6Ø IF X$="-" THEN S=2:PS=I
 7Ø NEXT I
 8Ø N1=VAL(MID$(A$,1,PS-1))
 9Ø N2=VAL(MID$(A$,PS+1,L-PS))
1ØØ IF S=1 THEN R=N1+N2
11Ø IF S=2 THEN R=N1-N2
12Ø PRINT R
13Ø END
```

Evidemment il faudra encore améliorer ce programme pour qu'il marche dans tous les cas.

Excellent exercice pour montrer l'utilité de la transformation d'une suite alphanumérique en nombre.

Tableaux

Problème :

Supposons que nous ayons à entrer 1Ø nombres dans l'ordinateur. Nous devrons utiliser 1Ø variables différentes et le programme s'écrira par exemple :

```
  1Ø INPUT A
  2Ø INPUT B
  3Ø INPUT C
  4Ø INPUT D
  5Ø INPUT E
  6Ø INPUT F
  7Ø INPUT G
  8Ø INPUT H
  9Ø INPUT I
 1ØØ INPUT J
```

Nous nous rendons bien compte que ce programme est long et qu'il faut utiliser beaucoup de noms de variables différents.

Et si on avait à introduire 1ØØ, 5ØØ ou même 1ØØØ données (pour des valeurs statistiques par exemple !) ?

Il serait pratique de disposer d'un tableau tel que chaque élément du tableau se distingue par son numéro d'ordre dans le tableau

13
5
20
26
12
5
18

le nombre 13 est le premier élément du tableau

le nombre 5 est le 2ème élément du tableau

le nombre 20 est le 3ème élément du tableau

le nombre 26 est le 4ème élément du tableau

le nombre 12 est le 5ème élément du tableau

etc

Hé bien, ceci est possible grâce au basic !

Toutes les variables mentionnées jusqu'ici étaient des variables simples c'est-à-dire une case mémoire qui ne peut contenir qu'une seule valeur à la fois ! Nous allons introduire maintenant les "variables indicées", qui sont des éléments de tableaux.

Un tableau est une collection de variables de même type ayant un même nom.

On distingue les variables par la valeur de l'indice apparaissant entre les parenthèses qui suivent le nom du tableau.

Exemple : soit V le nom d'un tableau

V(1) sera le premier élément d'un tableau
V(2) sera le deuxième élément du tableau
⋮

Les numéros 1, 2, 3 ... sont appelés indexes ou indices.

Pour définir un tableau, il faut

1) lui donner un nom (les règles seront les mêmes que pour les variables simples)
   Si le tableau doit contenir des variables alphanumériques, on écrira $ à la fin du nom

2) établir la limite supérieure de l'indice du tableau par l'instruction DIM
   Un tableau doit être d'abord dimensionné pour que l'ordinateur puisse réserver en mémoire la place nécessaire.

   10 DIM V(15) signifie que le tableau V aura au maximum 15 éléments

   L'utilisation de V(16), c'est-à-dire la 16ème case du tableau V provoquera alors une erreur.

Remarque :

1) Certains ordinateurs commencent les tableaux à partir de l'indice Ø

   V(Ø) existe alors.

2) Si DIM n'est pas spécifié, certains ordinateurs prendront la valeur maximum de l'indice = 1Ø

Tableaux à une dimension ou vecteurs

Supposons qu'on veuille stocker dans un tableau les valeurs suivantes :

13 5 20 26 12 5 18 23 -2 4.3

Il faudra définir un tableau numérique de nom V par exemple.

On y stockera les valeurs ci-dessus et on pourra accéder à chaque élément en donnant l'indice adéquat.

ELEMENTS	CONTENU
V(1)	13
V(2)	5
V(3)	20
V(4)	26
V(5)	12
V(6)	5
V(7)	18
V(8)	23
V(9)	- 2
V(10)	4.3

Exemple :

Lire une série de 20 éléments (numériques) et les stocker dans un vecteur V.

```
1Ø DIM V(1Ø)
2Ø FOR I=1 TO 1Ø
3Ø INPUT X          }      on peut simplement écrire
4Ø V(I)=X           }          INPUT V(I)
5Ø NEXT I
```

Augmentez de 1 chacun des éléments du tableau ci-dessus

```
60 FOR I=1 TO 10
70 V(I)=V(I)+1
80 NEXT I
```

Affichez les nouvelles valeurs du tableau à l'écran

```
90 FOR I=1 TO 10
100 PRINT V(I)
110 NEXT I
```

On peut également lire une série de 5 noms et les stocker dans un vecteur N$.

```
10 DIM N$(5)
20 FOR I=1 TO 5
30 PRINT "NOM:"
40 INPUT N$(I)
50 NEXT I
```

Le tableau contiendra par exemple les valeurs ci-dessous :

ELEMENT	CONTENU
N$(1)	DENISE
N$(2)	CLAUDINE
N$(3)	JEAN
N$(4)	PAUL
N$(5)	JACQUELINE

Tableaux à plusieurs dimensions

Supposons qu'on ait le tableau suivant :

NOM	MATRICULE	SEXE
DENISE	135	F
CLAUDINE	202	F
JEAN	612	M
PAUL	518	M
JACQUELINE	512	F

Ce tableau contient 5 lignes et 3 colonnes soit au total 15 éléments.

Si nous définissons un tableau alphanumérique A$ de 5 lignes et 3 colonnes, nous pourrons y stocker toutes les valeurs ci-dessus.

On pourra accéder à chacun des éléments en spécifiant les indices adéquats.

INDICE	1	2	3
1	DENISE	135	F
2	CLAUDINE	202	F
3	JEAN	612	M
4	PAUL	518	M
5	JACQUELINE	512	F

Chaque élément de la table A$ sera défini par 2 indices
séparés par une virgule et entre parenthèses.
Le premier indice désignera la ligne du tableau.
Le deuxième indice désignera la colonne du tableau.

Par exemple :

A$(1,1) désignera DENISE
A$(3,2) désignera 612
A$(5,3) désignera le caractère F

Remarque :

On peut définir des tableaux ayant plus de 2 dimensions mais
ils sont rarement utilisés.

Exercice :

Nous allons écrire le programme qui charge les valeurs
ci-dessus dans le tableau A$.

```
10 DIM A$(5,3)
20 FOR I=1 TO 5
30 PRINT "NOM"
40 INPUT A$(I,1)
50 PRINT "MATRICULE"
60 INPUT A$(I,2)
70 PRINT "SEXE"
80 INPUT A$(I,3)
90 NEXT I
```

Instruction de lecture spéciale

Il peut souvent être intéressant de disposer d'une série de constantes dans un programme (fichiers constants en mémoire).

Exemple :

Lorsque l'ordinateur possède la valeur numérique d'un mois, il est parfois important de disposer également du mois en clair.

Il faudrait donc pouvoir disposer en mémoire de la liste des mois en clair sans être obligé de les fournir à chaque fois à l'ordinateur.

Les instructions READ DATA nous permettent de stocker des valeurs constantes dans un programme afin que l'ordinateur puisse les lire :

READ veut dire lire
DATA veut dire donnée

Le programme exposé ci-dessous s'écrira :

```
10 DIM M$(12)
20 FOR I=1 TO 12
30 READ M$(I)
40 DATA "JANVIER","FEVRIER","MARS","AVRIL"
50 DATA "MAI","JUIN","JUILLET","AOUT","SEPTEMBRE"
60 DATA "OCTOBRE","NOVEMBRE","DECEMBRE"
65 NEXT I
70 PRINT "NUMERO DU MOIS"
80 INPUT N
90 PRINT M$(N)
100 END
```

L'instruction READ est une instruction de lecture comme le INPUT ; mais READ lit les données stockées dans l'instruction DATA.

Voyons en détail comment se déroulent les instructions READ et DATA

```
1Ø READ A
2Ø READ B
3Ø READ C
4Ø PRINT "A=";A
5Ø PRINT "B=";B
6Ø PRINT "C=";C
7Ø DATA 1,2,3
```

Que s'est-il passé ?

La 1ère instruction READ a lu la 1ère valeur de Data
La 2ème instruction READ a lu la 2ème valeur de Data
La 3ème instruction READ a lu la 3ème valeur de Data

Remarque : L'instruction DATA peut se mettre n'importe où dans le programme

Une autre application :

Chargez dans un tableau le nombre de jours compris dans chaque mois

```
10 DIM V(12)
20 FOR I=1 TO 12
30 READ V(I)
40 NEXT I
    :
    :
    :
1000 DATA 31,29,31,30,31,30,31,31,30,31,30,31
```

A la suite du programme ci-dessus, supposons que le programme lise une date au clavier :

```
100 PRINT "JOUR"
110 INPUT J
120 PRINT "MOIS"
130 INPUT M
140 REM CONTROLE SUR LE MOIS
150 IF M > 12 THEN PRINT "ERREUR MOIS":GOTO 120
160 REM CONTROLE SUR LE JOUR
170 IF J > V(M) THEN PRINT "ERREUR JOUR":GOTO 100
```

Protégeons notre programme des vilains curieux !

chargement du tableau

```
10 DIM P$(5)
20 FOR I=1 TO 5
30 READ P$(I)
40 NEXT I
50 DATA "DENISE","CLAUDINE","PAUL","JEAN","JACQUELINE"
60 PRINT "VOTRE PRENOM"
70 INPUT N$
80 FOR I=1 TO 5
90 IF P$(I)=N$ THEN GOTO 130
100 NEXT I
110 PRINT "TOP SECRET"
120 END
130 PRINT "BONJOUR";N$
    :
    :      (Suite du programme)
```

ANNUAIRE TELEPHONIQUE

Puisque grâce à l'instruction READ DATA nous pouvons stocker en mémoire un fichier, pourquoi pas stocker notre fichier d'amis ?

Exemple :

Notre fichier est constitué des personnes suivantes

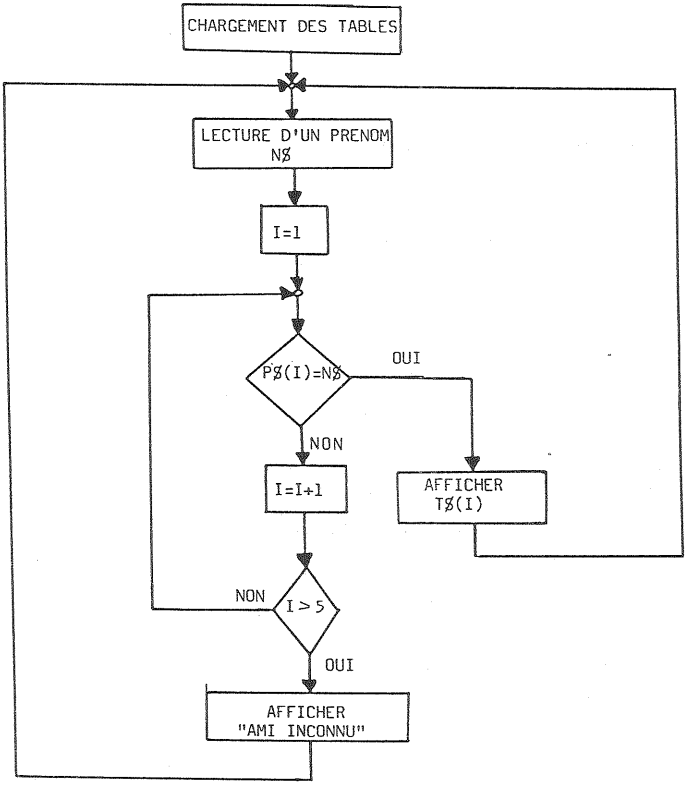
PRENOMS	Nº TELEPHONE
DENISE	13 52 02
CLAUDINE	61 25 18
PAUL	252 02 02
JEAN	55 12 91
JACQUELINE	912 52 08

Utilisons 2 tableaux en mémoire :

P$: table alphabétique comprenant tous les prénoms

T$: table alphanumérique comprenant tous les numéros de téléphone

PRINCIPE DU PROGRAMME :



```
                10 REM ANNUAIRE TELEPHONIQUE
                20 DIM P$(5):DIM T$(5)
chargement      30 FOR I=1 TO 5
   des          40 READ P$(I):READ T$(I)
 tableaux       50 NEXT I
                60 DATA "DENISE","13 52 01","CLAUDINE","61 25 18"
                70 DATA "PAUL","252 02 02","JEAN","55 12 91"
                80 DATA "JACQUELINE","912 52 08"
   lecture      100 REM RECHERCHE D'UN NUMERO
     d'un       110 PRINT "PRENOM"
   prénom       120 INPUT N$
recherche dans  130 FOR I=1 TO 5
la table des    140 IF P$(I)=N$ THEN GOTO 180
   prénoms      150 NEXT I
numéro non      160 PRINT "AMI INCONNU"
trouvé,         170 GOTO 100
retour au
début
affichage du    180 PRINT "SON NUMERO EST";T$(I)
numéro et       190 GOTO 100
retour au
début
```

Nous allons essayer de refaire le programme de lecture du numéro de mois et affichage du mois en clair sans utiliser de tableau.

Le tableau devient :

```
1Ø PRINT "NUMERO DE MOIS"
2Ø INPUT N1
3Ø FOR I=1 TO N1
4Ø READ M$
5Ø NEXT I
6Ø PRINT M$
7Ø DATA "JANVIER","FEVRIER","MARS","AVRIL","MAI"
8Ø DATA "JUIN","JUILLET","AOUT","SEPTEMBRE"
9Ø DATA "OCTOBRE","NOVEMBRE","DECEMBRE"
    :
    :
    :
```

Ce programme marche parfaitement et est avantageux parce qu'il n'utilise pas de tableau en mémoire, on a donc gagné de la place en mémoire, ce qui est très intéressant.

Mais où est le problème ?

Supposons qu'à un autre endroit du programme on veuille à nouveau lire un numéro de mois et afficher le mois en clair.

Que se passe-t-il ?

On écrira par exemple :

```
    :
    :
    :
12ØØ PRINT "AUTRE NUMERO DE MOIS"
121Ø INPUT N2
122Ø FOR I=1 TO N2
123Ø READ M$
124Ø NEXT I
125Ø PRINT M$
    :
    :
    :
```

Chaque fois que l'ordinateur exécute une instruction READ il passe à la donnée (DATA) suivante.

L'ordinateur ne revient pas automatiquement à la lère donnée de data.

Ainsi dans le programme précédent, supposez que pour le mois n° 1 on tape 6, l'ordinateur nous affichera bien JUIN.

Puis pour l'autre numéro de mois, tapons 7, l'ordinateur continuera de lire les données à partir de juillet, août, septembre ... décembre et il nous affichera une erreur car il ne trouvera plus de Data.

Heureusement, Basic dispose de l'instruction RESTORE qui permet de revenir à la première donnée de Data.

Dans le programme précédent, il suffisait donc d'ajouter une ligne 1215 RESTORE pour que ça marche comme sur des roulettes !

Sous-programme

Supposons que l'ordinateur doive lire 3 variables, effectuer sur chacune d'elles un calcul identique et afficher le résultat.

```
 1Ø INPUT A
 2Ø X=SIN(A)+COS(A+3)-LOG(A*2)+13.52026
 3Ø PRINT X
 4Ø INPUT A
 5Ø X=SIN(A)+COS(A+3)-LOG(A*2)+13.52026
 6Ø PRINT X
 7Ø INPUT A
 8Ø X=SIN(A)+COS(A+3)-LOG(A*2)+13.52026
 9Ø PRINT X
1ØØ END
```

Nous remarquons que les instructions de lecture, calcul et affichage sont identiques.

Si un même traitement se répète plusieurs fois, on peut écrire un sous-programme :

```
 1Ø GOSUB 1ØØ
 2Ø GOSUB 1ØØ
 3Ø GOSUB 1ØØ
 4Ø END
1ØØ INPUT A
11Ø X=SIN(A)+COS(A+3)-LOG(A*2)+13.52026
12Ø PRINT X
13Ø RETURN
```

L'instruction GOSUB se branche à une adresse (numéro de l'instruction comme le GOTO), poursuit le traitement à partir de cette adresse ; et lorsqu'il rencontre l'instruction RETURN, il revient à la lère instruction qui suit le GOSUB qui a été exécuté.
GOSUB vient de l'anglais GO SUBROUTINE (aller au sous-programme)
RETURN veut dire RETOUR au programme principal.

Afin de mieux comprendre, GOSUB et RETURN
voyons le programme suivant :

```
 1Ø GOSUB 1ØØ
 2Ø PRINT "RETOUR"
 3Ø END
1ØØ PRINT "SOUS-PROGRAMME"
11Ø PRINT "EN"
12Ø PRINT "COURS"
13Ø RETURN
```

Si on exécute ce programme, les messages s'afficheront dans l'ordre suivant :

```
SOUS-PROGRAMME
EN
COURS
RETOUR
```

REMARQUES :

- un sous-programme peut être appelé autant de fois qu'il est nécessaire
- le nombre de sous-programmes n'est limité que par la place mémoire
- on peut emboîter des GOSUB, la seule limite étant la mémoire disponible à cet effet
- il est conseillé de bien séparer les sous-programmes du programme principal afin de permettre une meilleure lisibilité du programme

  Suggestions : . mettre des commentaires au début de chaque sous-programme

  . regrouper tous les sous-programmes à la fin du programme

Branchement calculé

Soit un programme permettant d'effectuer différents traitements suivant le choix de l'utilisateur.

Au début du programme, s'affiche un menu :

```
        MENU

1. Traitement 1
2. Traitement 2
3. Traitement 3
4. Traitement 4
5. Traitement 5
6. Traitement 6
7. Traitement 7
8. Traitement 8

                    Votre choix ?
```

L'utilisateur devra taper 1, 2, 3, 4, 5, 6, 7 ou 8 suivant le traitement qu'il veut effectuer.

Suivant la réponse tapée, le programme se branche à différents numéros de ligne du programme.

Le programme pourra s'écrire de la manière suivante :

```
    5 REM AFFICHAGE DU MENU
   1Ø PRINT "MENU"
   2Ø PRINT "1.TRAITEMENT 1"
   3Ø PRINT "2. TRAITEMENT 2"
   4Ø PRINT "3. TRAITEMENT 3"
   5Ø PRINT "4. TRAITEMENT 4"
   6Ø PRINT "5. TRAITEMENT 5"
   7Ø PRINT "6. TRAITEMENT 6"
   8Ø PRINT "7. TRAITEMENT 7"
   9Ø PRINT "8. TRAITEMENT 8"
  1ØØ PRINT "VOTRE CHOIX"
  11Ø REM LECTURE DU CHOIX C
  12Ø REM ET BRANCHEMENTS
  13Ø INPUT C
  14Ø IF C=1 THEN GOTO 1ØØØ
  15Ø IF C=2 THEN GOTO 2ØØØ
  16Ø IF C=3 THEN GOTO 3ØØØ
  17Ø IF C=4 THEN GOTO 4ØØØ
  18Ø IF C=5 THEN GOTO 5ØØØ
  19Ø IF C=6 THEN GOTO 6ØØØ
  2ØØ IF C=7 THEN GOTO 7ØØØ
  21Ø IF C=8 THEN GOTO 8ØØØ
  22Ø PRINT "ERREUR"
  23Ø GOTO 13Ø

1ØØØ REM TRAITEMENT 1
        :
        :
        :

2ØØØ REM TRAITEMENT 2
        :
        :
        :

3ØØØ REM TRAITEMENT 3
        :
        :
        :

4ØØØ REM TRAITEMENT 4
        :
        :
        :

5ØØØ REM TRAITEMENT 5
        :
        :
        :

6ØØØ REM TRAITEMENT 6
        :
        :
        :

7ØØØ REM TRAITEMENT 7
        :
        :
        :

8ØØØ REM TRAITEMENT 8
        :
        :
        :
```

Trop long !

Nous voyons que pour effectuer les différents branchements, il faut effectuer un test pour chaque valeur de C.
Et si on avait beaucoup plus de valeurs possibles ?
L'instruction ON .... GOTO... permet d'éviter tous ces tests et donc de réduire la longueur du programme et d'accélérer son exécution.

Les lignes 14Ø, 15Ø, 16Ø, 17Ø, 18Ø, 19Ø, 2ØØ, 21Ø peuvent se réduire en une seule :

```
14Ø ON C GOTO 1ØØØ, 2ØØØ, 3ØØØ, 4ØØØ, 5ØØØ, 6ØØØ, 7ØØØ, 8ØØØ
```

Syntaxe de l'instruction :

ON expression GOTO liste de numéros de lignes

Ex. : ON C GOTO 1ØØØ, 2ØØØ, 3ØØØ, 4ØØØ, 5ØØØ, 6ØØØ, 7ØØØ, 8ØØØ
se branche à la ligne dont le numéro est le Cième nombre après le GOTO.

c'est-à-dire :

si C=1, on va en 1ØØØ
si C=2, on va en 2ØØØ
si C=3, on va en 3ØØØ
si C=4, on va en 4ØØØ
si C=5, on va en 5ØØØ
si C=6, on va en 6ØØØ
si C=7, on va en 7ØØØ
si C=8, on va en 8ØØØ

Si C=Ø ou si on cherche a sélectionner une ligne qui n'est pas citée (C supérieur ou égal à 9), l'ordinateur exécute l'instruction située après ON... GOTO

ON... GOSUB...

L'instruction ON... GOSUB est identique à ON... GOTO sauf que c'est un appel de sous-programme qui est effectué (GOSUB).

Le retour (RETURN) se fait à l'instruction qui suit le ON... GOSUB.

Exemple :

```
    10 INPUT I
    20 ON I GOSUB 1000, 2000, 3000
    30 PRINT "RETOUR"
    40 END
  1000 PRINT "SOUS-PROGRAMME 1"
  1010 RETURN
  2000 PRINT "SOUS-PROGRAMME 2"
  2010 RETURN
  3000 PRINT "SOUS-PROGRAMME 3"
  3010 RETURN
```

Exercices

LA PHRASE MYSTERIEUSE

Chargez 3 tableaux ; dans le premier on donne des sujets, dans le 2ème on donne des verbes, dans le 3ème des compléments d'objets directs.

W1$		W2$		W3$	
1	RENE	1	MANGE	1	l'ordinateur
2	IRENE	2	RAMASSE	2	lentement
3	LE VOISIN	3	REGARDE	3	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

L'ordinateur doit vous afficher 5 phrases au hasard.

```
1Ø DIM W1$(1O):DIM W2$(1Ø):DIM W3$(1Ø)
2Ø PRINT "DONNEZ 1Ø SUJETS"
3Ø FOR I=1 TO 1Ø
4Ø INPUT W1$(I)
5Ø NEXT I
6Ø PRINT "DONNEZ 1Ø VERBES"
7Ø FOR I=1 TO 1Ø
8Ø INPUT W2$(I)
9Ø NEXT I
1ØØ PRINT "DONNEZ 1Ø COMPLEMENTS"
11Ø FOR I=1 TO 1Ø
12Ø INPUT W3$(I)
13Ø NEXT I
145 FOR I=1 TO 5
15Ø X1=INT(RND(1)*1Ø)+1
16Ø X2=INT(RND(1)*1Ø)+1
17Ø X3=INT(RND(1)*1Ø)+1
19Ø PRINT W1$(X1);W2$(X2);W3$(X3)
2ØØ NEXT I
21Ø END
```

La fonction RND(1) donne un nombre aléatoire (au hasard) compris entre Ø et 1.

Cette fonction sera vue plus en détail dans le tome V.

CALCUL MENTAL

Ce programme vous propose 7 additions, vous devez donner la réponse (juste si possible).

. J et F sont 2 variables où l'ordinateur va cumuler le nombre de réponses justes (J) et le nombre de réponses fausses (F) ; ces 2 variables sont mises à zéro (instructions 6 et 7)

. la boucle, qui permet de tourner 7 fois va de l'instruction 1Ø à l'instruction 1Ø2

. dans l'instruction 2Ø et 3Ø, l'ordinateur créé 2 nombres entiers au hasard compris entre Ø et 1ØØ : X et Y

. dans l'instruction 6Ø il se calcule la réponse (R) qu'il ne vous donne pas, bien entendu !

. puis c'est à vous de donner la réponse (S) dans l'instruction 75

. instruction 8Ø et 9Ø :

  - si R=S (c'est-à-dire si votre réponse est identique à son calcul à lui) il augmente J de 1

  - si R est différent de S (c'est-à-dire votre réponse est fausse, si on considère que lui ne se trompe pas !) il augmente E de 1

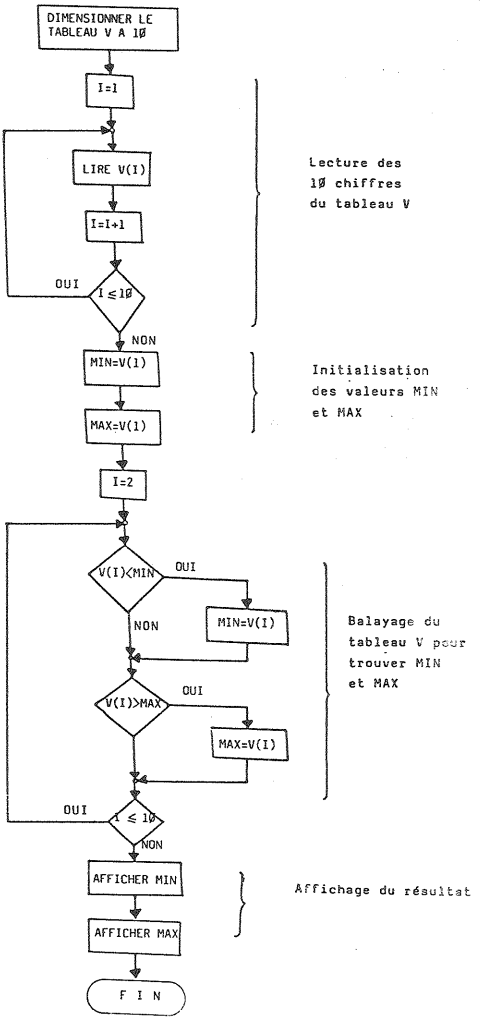
. après 7 tours dans la boucle, il vous affiche le nombre de réponses justes (J) et le nombre de réponses fausses (F) dans l'instruction 11Ø et 12Ø

```
 5 REM CALCUL MENTAL
 6 J=Ø
 7 F=Ø
1Ø I=1
2Ø X=INT(RND(1)*1ØØ)
3Ø Y=INT(RND(1)*1ØØ)
4Ø PRINT "ADDITIONNEZ CES 2 NOMBRES"
5Ø PRINT X;" + ";Y
6Ø R=X+Y
7Ø PRINT "QUEL EST VOTRE RESULTAT"
75 INPUT S
8Ø IF R=S THEN J=J+1
9Ø IF R<>S THEN F=F+1
1ØØ I=I+1
1Ø2 IF I <=7 THEN GOTO 2Ø
11Ø PRINT J;" REPONSES SONT JUSTES"
12Ø PRINT F ;" REPONSES SONT FAUSSES"
```

CHERCHEZ LE MINIMUM ET LE MAXIMUM ...

Je vous propose d'écrire un programme qui lit 1Ø chiffres et affiche quelles sont les valeurs minimales et maximales qu'on a introduites.

Organigramme :



PROGRAMME BASIC :

```
10  REM MINIMUM ET MAXIMUM D'UN TABLEAU
20  DIM V(10)
30  FOR I=1 TO 10
40  INPUT V(I)
50  NEXT I
60  MIN=V(1)
70  MAX=V(1)
80  FOR I=2 TO 10
90  IF V(I) < MIN THEN MIN=V(I)
100 IF V(I) > MAX THEN MAX=V(I)
110 NEXT I
120 PRINT "MINIMUM=";MIN
130 PRINT "MAXIMUM=";MAX
140 END
```

. instructions 20 à 50 :
  lecture de 10 nombres

. dans les instructions 60 et 70, on considère que le
  1er nombre est MIN et MAX

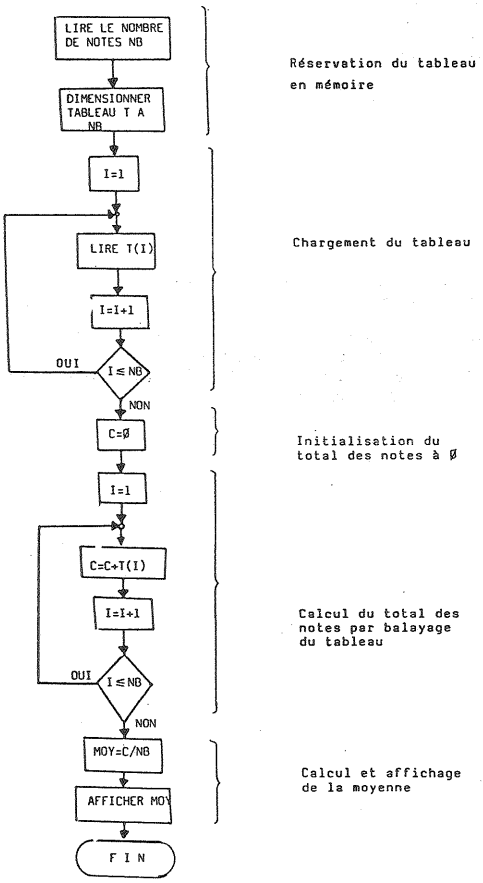
. dans les instructions 80 à 110 on recherche si dans la
  table V il y en a un plus petit que MIN ou un plus grand
  que MAX ; si oui on le garde ... à la fin on aura donc
  le plus petit et le plus grand

VALEUR MOYENNE D'UN TABLEAU

Moyenne des notes d'un élève

On dispose d'un tableau contenant les notes d'un élève.
On désire calculer la moyenne des notes de cet élève.

Organigramme :



```
10 REM MOYENNE DES NOTES
20 PRINT "NOMBRE TOTAL DE NOTES"
30 INPUT NB
40 DIM T(NB)
50 FOR I=1 TO NB
60 INPUT T(I)
70 NEXT I
80 REM CALCUL DE LA MOYENNE
90 REM A PARTIR DU TABLEAU
100 C=0
110 FOR I=1 TO NB
120 C=C+T(I)
130 NEXT I
140 MOY=C/NB
150 PRINT "LA MOYENNE EST";MOY
```

En cette période de pénurie, faites donc calculer à l'ordinateur votre consommation moyenne d'essence !

Supposons que nous ayons cherché 4 fois de l'essence.

La 1ère fois nous avons pris 11 litres, nous avons roulé 135 km
La 2ème fois nous avons pris 16 litres, nous avons roulé 202 km
La 3ème fois nous avons pris 39 litres, nous avons roulé 612 km
La 4ème fois nous avons pris 35 litres, nous avons roulé 518 km

Pour calculer la consommation moyenne (CM), il faut calculer
la consommation totale (CT) = 11+16+39+35=101 litres
et la distance totale parcourue (DP) :
135+202+612+518=1467 km

La consommation moyenne (CM) est donnée par la formule :

$$CM = \frac{CT \times 100}{DP}$$

Dans notre exemple, $\frac{101 \times 100}{1467} = 6,8$ pour 100 km

Comment faire avec l'ordinateur ?

Nous allons remplir, non pas notre réservoir d'essence
mais 2 tableaux dans la mémoire de l'ordinateur :

1) Un tableau L qui contient le nombre de litres d'essence consommés

2) Un tableau K qui contient le nombre de kilomètres effectués

Tableau L

L(1)	11
L(2)	16
L(3)	39
L(4)	35

Tableau K

K(1)	135
K(2)	202
K(3)	612
K(4)	518

Au début du programme, l'ordinateur nous demandera le nombre de fois où nous avons cherché de l'essence (NB) puis il demandera toutes les valeurs (litres et Km) et il nous affichera notre consommation moyenne.

Dans le cas de l'exemple NB=4

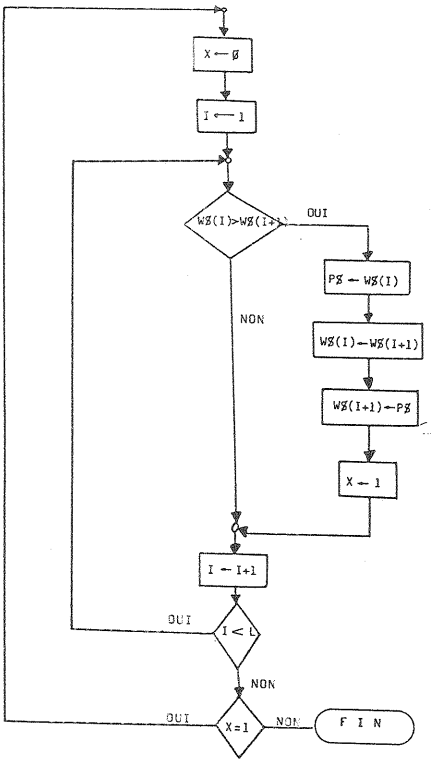
On appellera CT = la consommation totale
DP = la distance parcourue
CM = la consommation moyenne

```
10 REM CONSOMMATION MOYENNE
20 PRINT "NOMBRE DE RELEVES"
30 INPUT NB
40 DIM L(NB):DIM K(NB)
50 FOR I=1 TO NB
60 PRINT "LITRES"
70 INPUT L(I )
80 PRINT "KM"
90 INPUT K(I)
100 NEXT I
110 REM CALCUL
120 CT=0:DP=0
130 FOR I=1 TO NB
140 CT=CT+L(I)
150 DP=DP+K(I)
160 NEXT I
170 CM=CT*100/DP
180 PRINT "CONSOMMATION";CM
190 END
```

PROGRAMMES DE TRI

Principe

Triez un vecteur W$ de longueur L en ordre croissant



Le principe du tri consiste à balayer le vecteur et d'échanger deux cases si elles ne sont pas ordonnées ; lorsqu'on fait un échange on met X à 1 et si X est à 1 on recommence. Lorsque X est resté à Ø, cela veut dire que nous venons de balayer notre vecteur sans faire d'échange, et donc qu'il est trié.

REMARQUE :

Il est possible de trier des lettres, des chiffres, des nombres, des mots.
Il existe plusieurs algorithmes de tri

Exemple : recherche successive du minimum

Programme BASIC chargeant du texte dans un tableau et triant ce tableau dans l'ordre alphabétique

```
5 REM CHARGEMENT
1Ø PRINT "NOMBRE DE MOTS"
2Ø INPUT L
3Ø DIM W$(L)
4Ø FOR I=1 TO L
5Ø INPUT W$(I)
6Ø NEXT I
7Ø REM TRI
8Ø X=Ø
9Ø FOR I=1 TO L
1ØØ IF W$(I) <=W$(I+1) THEN 15Ø
11Ø P$=W$(I)
12Ø W$(I)=W$(I+1)
13Ø W$(I+1)=P$
14Ø X=1
15Ø NEXT I
16Ø IF X=1 THEN 8Ø
17Ø REM FIN DU TRI
18Ø FOR I=1 TO L
19Ø PRINT W$(I)
2ØØ NEXT I
```

chargement du tableau à L éléments

T R I

affichage du tableau trié

Encore une autre application des tableaux :

En tapant le mois en chiffre, l'ordinateur doit l'afficher en clair.

```
1Ø DIM W$(12)
2Ø W$(1)="JANVIER"
3Ø W$(2)="FEVRIER"
:
:
:
13Ø W$(12)="DECEMBRE"
14Ø INPUT "MOIS EN CHIFFRES",M
15Ø PRINT W$(M)
16Ø GOTO 14Ø
```

et pour plus de précautions, rajoutez :

```
145 IF M >12 GOTO 17Ø
17Ø PRINT "CE MOIS N'EXISTE PAS"
18Ø END
```

Remarque :

Nous avons introduit une nouvelle forme de l'instruction INPUT permettant d'afficher un message et d'introduire des données.

```
INPUT"MESSAGE";VARIABLE
```

Exercice

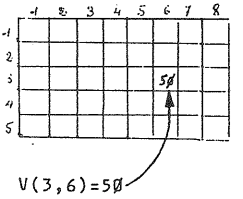
L'ordinateur doit trier les lettres de votre nom dans l'ordre croissant.

```
1Ø INPUT "DONNEZ VOTRE NOM";N$
2Ø L=LEN (N$)
3Ø DIM W$(L)
4Ø FOR I=1 TO L
5Ø W$(I)=MID$(N$,I,1)
6Ø NEXT I
7Ø X=Ø
8Ø FOR I=1 TO L-1
9Ø IF W$(I+1) <=W$(I) THEN GOTO 14Ø
1ØØ P$=W$(I)
11Ø W$(I)=W$(I+1)
12Ø W$(I+1)=P$
13Ø X=1
14Ø NEXT I
15Ø IF X=1 THEN GOTO 7Ø
16Ø M$=""
17Ø FOR I=1 TO L
18Ø M$=M$+W$(I)
19Ø NEXT I
2ØØ PRINT M$
21Ø END
```

Et enfin de quoi régaler les matheux !

Exemple de matrice à 2 dimensions :

DIM V(5,8)



Rappel : V(I,J)

Le 1er indexe donne le N° ligne
Le 2ème indexe donne le N° colonne

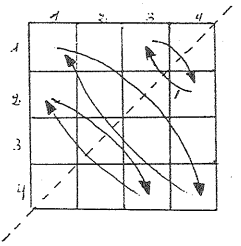
Exercices :

1) Mettre à zéro une matrice MAT de deux dimensions de NXM (c'est-à-dire N lignes, M colonnes)

```
10 FOR I=1 TO N
20 FOR J=1 TO M
30 MAT(I,J)=0
40 NEXT J
50 NEXT I
```

2) Dans une matrice carrée MAT de dimension N X N
faire l'échange par rapport à la première diagonale

Exemple : matrice 4 X 4 (N=4)



Analyse du problème

coordonnées des cases à inverser

I	J		I'	J'
1	1	⟷	4	4
1	2	⟷	3	4
1	3	⟷	2	4
2	1	⟷	4	3
2	2	⟷	3	3
3	1	⟷	4	2
(*)	(°)			

{ I : ligne
J : colonne

(*). Le balayage se fera pour tous les I allant de 1 à 3,
c'est-à-dire de 1 à (N-1)

(°). Le balayage se fera pour tous les J allant de 1 à (N-I)

On remarque également que I+J'=N+1 et J+I'=N+1 donc I'=N+1-J et J'=N+1-I

En résumé, pour la matrice de dimension N X N on balaye pour tous les I allant de 1 à (N-1) et tous les J allant de 1 à (N-I) et la case MAT (I,J) sera à échanger avec MAT ((N+1-J),(N+1-I)).

PROGRAMME

```
1Ø FOR I=1 TO N-1
2Ø FOR J=1 TO N-I
3Ø X = MAT(I,J)                          }
4Ø MAT(I,J)=MAT((N+1-J),(N+1-I))         }  Echange
5Ø MAT((N+1-J),(N+1-I))=X                }
6Ø NEXT J
7Ø NEXT I
```

Un petit programme amusant ...

```
   5 REM PROGRAMMER LES RESULTATS
   6 REM            DU
   7 REM         L O T O
   8 REM
  1Ø A=Ø:B=Ø:C=Ø:D=Ø:E=Ø:F=Ø
  2Ø GOSUB 1ØØØ
  3Ø A=N
  4Ø GOSUB 1ØØØ
  5Ø B=N
  6Ø GOSUB 1ØØØ
  7Ø C=N
  8Ø GOSUB 1ØØØ
  9Ø D=N
 1ØØ GOSUB 1ØØØ
 11Ø E=N
 12Ø GOSUB 1ØØØ
 13Ø F=N
 14Ø GOSUB 1ØØØ
 15Ø PRINT "LE TIRAGE EST"
 16Ø PRINT A;B;C;D;E;F
 17Ø PRINT "LE NUMERO COMPLEMENTAIRE EST";N
 18Ø END
1ØØØ N=INT(RND(1)*49)+1
1Ø1Ø IF A=N THEN GOTO 1ØØØ
1Ø2Ø IF B=N THEN GOTO 1ØØØ
1Ø3Ø IF C=N THEN GOTO 1ØØØ
1Ø4Ø IF D=N THEN GOTO 1ØØØ
1Ø5Ø IF E=N THEN GOTO 1ØØØ
1Ø6Ø IF F=N THEN GOTO 1ØØØ
1Ø7Ø RETURN
```

Remarque :

La ligne 1ØØØ calcule un nombre entier compris entre 1 et 49.

La fonction RND(1) génère un nombre réel aléatoire compris entre Ø et 1.

En multipliant ce nombre par 49, on obtient un nombre compris entre Ø et 49.

On prend la valeur entière de ce nombre pour éliminer les virgules.

En additionnant 1, on obtient un nombre pouvant avoir toutes les valeurs de 1 à 49.