APPRENDRE LE BASIC

SUR ORIC

PAR

Henri COHEN SOLAL

Agrégé de l'Université

ASN Diffusion

Z.I. La Haie Griselle - 94470 BOISSY-ST-LÉGER

INTRODUCTION

Ce cours de programmation en BASIC s'adresse à tous ceux qui n'ont aucune connaissance en informatique et qui souhaitent s'initier à la programmation en langage BASIC.

Ce cours est composé de 7 leçons. Chacune d'elles comporte deux sections de cours - exercices et une section de test. Chaque leçon possède un ou plusieurs paragraphes de RESUME.

Si vous laissez passer un laps de temps trop important entre deux séances de travail, il est conseillé de reprendre au début de la dernière leçon étudiée.

Chargement d'une section : Si vous désirez appeler la LECON3 Section 1 :

1) *Introduire la cassette* contenant le titre LECON3 S1 ;

2) *Taper* CLOAD"LECON3 S1",S suivi d'une pression de la touche <RETURN>

un espace

Remarques sur le chargement :

1) Le chargement d'un programme en mode S (SLOW) dure 4 à 5 minutes suivant les sections. Il a cependant l'avantage d'assurer une meilleure transmission des données.

2) Il n'y a qu'un seul "espace" dans le titre LECON3 S1. Si le titre n'est pas conforme lorsque vous appuyez sur <RETURN>, vous n'aurez pas accès à la leçon souhaitée et vous serez obligé d'appuyer sur la touche <RESET>, sous l'appareil, avant de recommencer les opérations.

3) Les numéros de leçon et de section ont été enregistrés vocalement afin de faciliter la localisation du programme recherché.

Remarques générales :

1) Ne pas confondre la lettre O avec le zéro noté Ø.

2) Le passage des minuscules aux majuscules, et inversement, se fait en tapant <CTRL><T> (tenir enfoncée la touche <CTRL> puis appuyer sur la touche <T>).

3) Vous pouvez lors de vos exercices choisir la couleur du fond ainsi que la couleur d'écriture, à l'aide des instructions PAPER et INK (voir chapitre 4 du manuel ORIC).

4) Les paragraphes délicats sont signalés par le symbole  .

S O M M A I R E

LECON 1

1. QUELQUES INSTRUCTIONS EN B.A.S.I.C.

PRINT, RUN, LIST, LET, INPUT, IF...THEN...ELSE..., GOTO, GOSUB, FOR...STEP...NEXT, REPEAT...UNTIL,...

2. MESSAGE D'ERREUR

Voir appendice J du manuel ORIC la liste des messages d'erreurs possibles.

3. EXERCICES SUR PRINT (PRINT = AFFICHER ou IMPRIMER)

Avant de commencer les exercices, sachez qu'*un caractère* introduit par erreur *peut être immédiatement effacé* par pression de la touche <DEL> (touche au-dessus de la touche <RETURN>), et si vous maintenez appuyée la touche <DEL> vous effacerez les caractères précédents : (DEL ---> DELETE = EFFACER).

Vous pouvez *effacer tout l'écran* en tenant appuyée la touche <CTRL> puis en pressant la touche <L>.

Exécuter les différents ordres en essayant de prévoir ce que l'ordinateur va afficher (y compris les blancs).

```
PRINT "J'APPRENDS LE BASIC"                (appuyer sur <RETURN>
PRINT "   J'APPRENDS", "LE BASIC"           pour enregistrer l'ordre)
PRINT "J'APPRENDS"; " ";"LE BASIC"
PRINT "3+1=12"
PRINT"ON PEUT TOUT METTRE SAUF DES GUILLEMETS !"
```

Si vous avez eu des difficultés, regardez le résumé §4 puis reprenez les exercices.

4. RESUME LECON1 S1

instructions	PRINT"texte" fait afficher le texte (lettres, chiffres ou symboles) PRINT"texte 1";"texte 2" fait afficher texte 1 et texte 2 *accolés* PRINT"texte 1","texte 2" fait afficher les textes *séparés de 3 blancs*
touches	<DEL> permet d'effacer le dernier caractère entré <CTRL><L> permet d'effacer l'écran <RETURN> permet d'enregistrer dans l'ordinateur ce que l'on vient d'écrire.

5. HIERARCHIE DES OPERATIONS

Si vous n'utilisez pas de parenthèses pour les opérations à effectuer, l'ordinateur commencer par traiter multiplications et divisions, ensuite additions et soustractions.

Ainsi, dans l'exemple PRINT 5+1*2, l'ordinateur commence par calculer 1*2 (=2) puis 5 + ce résultat qui donne 7.

Pour être sûr de vos résultats, utilisez donc les parenthèses.

6. LE MODE PROGRAMME

Dans le programme qui vient d'être affiché, l'ordinateur exécute les instructions à partir du plus petit numéro (ici l'instruction 1Ø) et ce, dans l'ordre, jusqu'au numéro le plus grand (ici l'instruction 5Ø).

Les instructions sont ici *numérotées de 1Ø en 1Ø*. C'est une précaution qu'il faut prendre dès le début car cela permet en cas d'oubli, d'insérer de nouvelles instructions. Si l'on veut insérer entre PAPA et MAMAN le mot PEPE, il suffit de taper 15 PRINT"PEPE". Même si cette instruction a été tapée après l'instruction 5Ø *l'ordinateur remettra dans l'ordre toutes les instructions* avant de les exécuter.

7. COMMENTAIRES SUR LE PROGRAMME

```
1Ø PRINT"PAPA"
2Ø PRINT"MAMAN"
3Ø PRINT
4Ø PRINT"ROBERT,"ALAIN"
5Ø END
```

En lisant la *ligne* 1Ø l'ordinateur imprime PAPA *puis va à la ligne* (car il n'y a ni; ni,)
En lisant la *ligne* 2Ø il imprime MAMAN puis va à la ligne.
A la *ligne* 3Ø il n'imprime rien puis va à la ligne (ce *procédé est donc une manière de faire sauter une ligne* !).
A la *ligne* 4Ø il imprime ROBERT, laisse 3 blancs (à cause de la virgule) puis imprime ALAIN et va à la ligne (car ni, ni; après "ALAIN").
A la *ligne* 5Ø il lit END c'est-à-dire FIN et il s'arrête.

A la rencontre de l'instruction *END l'ordinateur s'arrête* et sort du MODE PROGRAMME (il affiche alors *Ready* c'est-à-dire Prêt).

Lire le par. 8 avant d'appuyer sur <RETURN>.

8. LES INSTRUCTIONS NEW, LIST, RUN (NEW = NOUVEAU LIST = LISTE)

Avant de commencer à taper un programme, il faut *vider la mémoire de l'ordinateur* de tout autre programme qui s'y trouverait. On tape pour cela l'instruction NEW.

A tout moment, lors de l'écriture d'un programme, on peut demander la liste de toutes les instructions que l'on vient d'écrire. On tape alors LIST et l'ordinateur *affiche la liste* ou "le listing" *de toutes les instructions dans l'ordre* (plus petit numéro au plus grand). Arrêt et redémarrage possibles en appuyant sur la touche <ESPACE> si le listing est trop long.

Lorsque l'on veut *faire exécuter* un programme que l'on vient d'introduire dans l'ordinateur, taper RUN et l'exécution commencera.

9. EXERCICES SUR LES OPERATIONS

Examiner les commandes ci-dessous une par une.
Prévoir la réponse de l'ordinateur.
Vérifier votre réponse en tapant la commande (suivie naturellement de <RETURN>).

```
PRINT 2*5+3
PRINT (2*5)+3
PRINT 2*(5+3)
PRINT 3+1Ø/2
PRINT (5-3)*(5Ø/1Ø)
PRINT "2*5+3"
PRINT "5*2=";1Ø
PRINT "5*2=";5*2
```

10. EXERCICES SUR LE MODE PROGRAMME

Relire le par. 8.
Pour corriger une ligne une fois le programme tapé, réécrire complètement la ligne (nous verrons plus loin un moyen plus rapide de correction).

Examiner chacun des programmes ci-dessous.
Prévoir la réponse de l'ordinateur.
Vérifier votre prévision en tapant NEW puis le programme (ne pas oublier <RETURN> à la fin de chaque ligne) puis RUN pour l'exécution.

Programme 1

```
10 PRINT"**********"
20 PRINT"* ROBERT *"
30 PRINT"**********"
40 END
```

Programme 2

```
10 CLS          (instruction de nettoyage de l'écran)
20 PRINT"JANVIER","FEVRIER","MARS"
30 PRINT
40 PRINT"       JANVIER","FEVRIER","MARS"
50 PRINT"JANVIER";"FEVRIER","MARS"
60 PRINT"JANVIER"
70 PRINT"FEVRIER"
80 PRINT"MARS"
90 END
```

Une fois ces programmes assimilés, essayer d'inventer des programmes de votre choix. Passer ensuite à la LECON1 S3, pour tester vos connaissances.

11. RESUME LECON1 S2

Utiliser des parenthèses pour les opérations (+ - x /).
PRINT 2 * 3 imprime le *résultat* de l'opération.
Numéroter de 10 en 10 les instructions.
END instruction de fin de programme.
NEW nettoie les mémoires (à faire avant de taper un programme).
LIST donne le listing des instructions.
RUN fait exécuter le programme qui est en mémoire.
CLS nettoie l'écran.

(CLS ---> CLEAR SCREEN = NETTOYAGE ECRAN)

LECON 2

1. EXEMPLES DE VARIABLES NUMERIQUES

A, B, C,..., Z, AA, AB,..., A1, A2,... sont des variables numériques. Remarquer que *le deuxième caractère peut être un chiffre* (ex: A3), mais que le premier caractère doit être une lettre.

2. MANIPULATION DES VARIABLES (LET = POSONS)

Dans l'écriture A = B+2, la variable (ou boite) dans laquelle on met un résultat doit toujours se trouver à GAUCHE SUIVIE du SIGNE =. Ici on met dans A le contenu de B auquel on a additionné 2.

On peut même prendre le contenu d'une variable (par exemple A), le modifier en ajoutant par exemple 3 et remettre le résultat dans la variable A. Ceci en écrivant :

A = A+3 qui est absurde en arithmétique, mais qui se lit en BASIC : dans A, on met le contenu de A additionné de 3.
Ainsi, si A contenait 4, après l'instruction A=A+3, A contiendra 7.

3. MOTS INTERDITS (MOTS RESERVES)

Le langage B.A.S.I.C. utilisant un certain vocabulaire, LET, END, OR, RUN,... ces mots ne doivent pas apparaître comme noms de variable, ni à l'intérieur d'un nom de variable. On dit que ce sont des MOTS INTERDITS.

Ainsi CALIFORNIE en est un, car il contient OR (le "ou" en BASIC).

4. LE ROLE D' INPUT (INPUT signifie ENTRER)

Dans le programme suivant :

```
10 CLS          l'ordinateur nettoie l'écran;
20 INPUT A      imprime un (?). Si l'on tape 15 il range 15 dans A;
30 B=2*A        calcule 2*A (donc 2*15 c'est-à-dire 30) qu'il range dans B;
40 PRINT B      imprime le contenu de B (ici 30);
50 END          s'arrête.
```

Ce programme sert donc à calculer le double d'un nombre *que l'on tape au clavier.*

Si l'on ne tape rien sur le clavier, rien ne se passera et l'ordinateur attendra... .

Remarque : la variable écrite après INPUT peut être "collée" au mot INPUT; on peut donc écrire INPUT A ou INPUTA.

5. RESUME DE LA LECON2 S1

```
A=3    3 est rangé dans la variable numérique A.
Seuls les 2 premiers caractères sont pris en compte.
Il y a des mots interdits.
PRINT A   affiche le contenu de A.
CLEAR     est une instruction qui met à zéro toutes les variables (c'est aussi le
          cas par RUN ou NEW).
```

```
INPUT A   range dans A, le nombre tapé au clavier.
INPUT A,B range dans A puis dans B, les 2 nombres entrés.
INPUT "texte";A  affiche le texte puis range dans A...
```

6. EXERCICES SUR LES VARIABLES NUMERIQUES

Examiner chacun des programmes ci-dessous.
Prévoir la réponse de l'ordinateur.
Vérifier votre prévision en tapant NEW puis le programme (ne pas oublier <RETURN> à la fin de chaque ligne) puis RUN pour l'exécution.

Programme 1

```
10 A=5
20 PRINT A
30 END
```

Programme 2

```
10 PRINT A
20 END
```
(pour le résultat penser à NEW)

Programme 3

```
10 A=3
20 B=5
30 PRINT A+B
40 PRINT A*B
50 END
```

Programme 4

```
10 CLS
20 A=5
30 PRINT"A CONTIENT ";A
40 PRINT"B CONTIENT ";B
50 END
```

Programme 5

```
10 CLS
20 I=5
30 PRINT I
40 I=I+1
50 PRINT I
60 END
```

Programme 6

```
10 CLS
20 A=3
30 B=A+1
40 PRINT A,B
50 END
```

Programme 7

```
10 CLS
20 BATEAU=10
30 BABA=20
40 PRINT BATEAU
50 PRINT BARIL
60 END
```
(bien regarder les 2 premiers caractères).

7. EXERCICES SUR INPUT

Même démarche que pour le par. 6, mais en plus vous répondrez par un nombre au ? créé par INPUT. (Essayer de répondre par une lettre et regarder ce qui se passe...).

Programme 1

```
10 CLS
20 INPUT A
30 PRINT"VOUS AVEZ TAPE ";A
40 END
```

Programme 2

```
10 CLS
20 INPUT"COMBIEN DE KGS DE BANANES ";K
30 F=7
40 P=F*K
50 PRINT"A 7 FRANCS LE KG CELA COUTE ";P
60 END
```

Programme 3

```
10 CLS
20 INPUT"ENTRER 2 NOMBRES ";A,B
30 S=A+B
40 P=A*B
50 PRINT"LA SOMME=";S
60 PRINT"LE PRODUIT=";P
70 END
```

8. INSTRUCTION PRINT TAB( ) (TAB ---> TABULATION)

L'instruction ne permet pas de revenir en arrière. On peut avoir :

```
1Ø PRINT TAB(5)"PAPA";      (le ; empêche le retour à la ligne)
2Ø PRINT TAB(15)"MAMAN"
```

Cela affichera PAPA à partir de la 5ème colonne et MAMAN à partir de la 15ème sur la même ligne (à cause du ;).

9. INSTRUCTION INT( ) (INT ---> INTEGER = ENTIER)

Cette instruction pourra être utilisée pour faire des arrondis grossiers. Lorsque les calculs ne tombent pas justes, l'ordinateur peut afficher jusqu'à 9 chiffres après la virgule.

Remarque sur la virgule : sur les ordinateurs on n'écrit pas *10,5 mais 10.5*

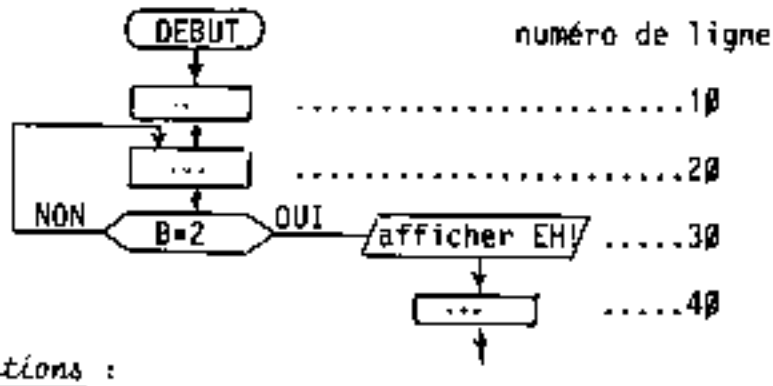
10. L'INSTRUCTION IF...THEN...ELSE... (SI...ALORS...SINON...)

Découpage de la ligne 3Ø

```
3Ø IF B=5 THEN   PRINT"EH!"   ELSE   2Ø
```

- B=5 : condition à tester
- PRINT"EH!" : ce qu'ORIC exécute si la condition est *vraie*
- 2Ø : ce qu'ORIC exécute si la condition est *fausse* (n'est pas vraie)

Cela peut se schématiser par un ORGANIGRAMME sous la forme suivante (lecture du haut vers le bas)



Explications :

Lignes 1Ø et 2Ø : ce sont des instructions (par exemple INPUT...)
Ligne 3Ø : on TESTE si B est égal à 2
si OUI alors on affiche EH! puis suite du programme
SINON, on retourne exécuter la ligne 2Ø.

11. REPONSES A DONNER AU CLAVIER

Programme 1

```
1Ø INPUT A
2Ø IF A=2 THEN END ELSE PRINT"GAGNE"
```

Programme 2

```
1Ø INPUT"ENTRER UN NOMBRE";X
2Ø IF X<Ø OR X>1Ø THEN PRINT X ELSE 1Ø
```

12. RESUME LECON2 S2

```
PRINT TAB(15)"AB"   affichera AB à partir de la 15ème colonne
INT(X)      calcule la partie entière du nombre X
                    |numéro X                |numéro Y
IF condition THEN   | ou            ELSE   | ou
                    |instruction ?           |instruction ??
                                                |ligne n°X
              si la condition est vraie alors exécution |  ou
                                                |instruction ?
                                                |ligne n°Y
              si la condition est fausse alors exécution |  ou
                                                |instruction ??
<      inférieur à
<=     inférieur ou égal à
>      supérieur à
>=     supérieur ou égal à
<> ou >< différent de
OR     ou
AND    et
<CTRL><C>   permet d'arrêter un programme alors qu'il s'exécute (par exemple
            lorsqu'un programme "boucle").
```

13. EXERCICES SUR PRINT TAB( ) et INT( )

Examiner chacun des programmes ci-dessous.
Prévoir la réponse de l'ordinateur.
Vérifier votre prévision (par RUN et une réponse à INPUT éventuellement).

Attention : Sur certains ORIC, il faut taper TAB(N+12) pour obtenir TAB(N) donc par exemple TAB(25) pour afficher à la 13ème colonne. Corriger les programmes ci-dessous si vous avez un tel ORIC.

Programme 1

```
10 CLS
20 PRINT
30 PRINT TAB(17)"ORIC"
40 PRINT TAB(17)"****"
50 END
```

Programme 2

```
10 CLS
20 PRINT
30 PRINT TAB(16)"******"
40 PRINT TAB(16)"*ORIC*"
50 PRINT TAB(16)"******"
60 END
```

Programme 3

```
10 CLS
20 PRINT"FOURNIER";
30 PRINT TAB(15)"Henri"
40 PRINT"DUPONT";
50 PRINT"Alain"
60 PRINT"MOREAU";
70 PRINT"Bertrand"
80 END
```

Programme 4

```
10 CLS
20 PRINT INT(3)
30 PRINT INT(3.9)
40 PRINT INT(-2.1)
50 END
```

14. EXERCICES SUR IF...THEN...ELSE...

Programme 1

```
10 CLS
20 PRINT TAB(13)"MAXIMUM DE 2 NOMBRES"
30 INPUT"ENTRER 2 NOMBRES";A,B
```

```
4Ø IF A<B THEN MAX=B  ELSE  MAX=A
5Ø CLS
6Ø PRINT"LE MAXIMUM DE ";A;" ET ";B;" EST ";MAX
7Ø END
```

Programme 2

```
1Ø CLS
2Ø PRINT" PRIX D'UN ARTICLE SUIVANT QUANTITE"
3Ø PRINT
4Ø INPUT"NOMBRE D'ARTICLES COMMANDES";X
5Ø IF X>1ØØ THEN PRIX=5ØØ  ELSE PRIX=6ØØ
6Ø PRINT
7Ø PRINT"LE PRIX PAR ARTICLE EST ";P;" FRANCS"
8Ø END
```

Programme 3 : jeu de découverte d'un nombre (2 joueurs)

```
1Ø CLS
2Ø INPUT"INTRODUIRE UN NOMBRE SANS QUE L'AUTRE JOUEUR NE LE VOIE";X
3Ø CLS
4Ø INPUT"PROPOSER UN NOMBRE";Y
5Ø IF Y=X THEN PRINT"GAGNE" ELSE 4Ø
6Ø END
```

LECON 3

1. LE SEPARATEUR D'INSTRUCTIONS " : "

Il convient de *ne pas abuser de ce séparateur*, d'une part pour une question de clarté, d'autre part, car on ne pourra plus accéder à une instruction que par le début de la ligne. Par exemple :
IF X>0 THEN 30 branchera (dans le 1er programme) l'ordinateur à l'instruction C=10.
Dans le deuxième programme il ne sera pas possible d'accéder directement à C=10.

2. L'INSTRUCTION REM (REM = REMARQUE)

Voici le même programme avec plus de détails :

```
10 REM ************************
20 REM   SOMME DE DEUX NOMBRES                    <--- titre
30 REM ************************
40:
50 REM A et B contiennent les 2 nombres           <--- les variables
60 REM  S= la somme de A et B
70:
75 CLS:REM  nettoyage écran                       <--- les entrées
80 INPUT"Introduire 2 nombres";A,B
90:
100 S=A+B                                         <--- le calcul
110:
120 PRINT:PRINT:REM pour affichage                <--- la sortie
130 PRINT"La somme de";A;" et de ";B;" est ";S
140 END
```

Remarquer la ligne 40: qui permet d'aérer le listing.

3. INSTRUCTION GOTO (GOTO = ALLER EN)

Pour arrêter un programme qui boucle (volontairement ou involontairement) on peut utiliser <CTRL><C> (en tenant la touche <CTRL> appuyée, presser la touche <C>) ou utiliser la touche <RESET> (sous l'appareil) ou encore arrêter l'ordinateur (on perd alors le programme).

L'instruction GOTO était apparue sans être écrite dans la leçon 2. En effet l'instruction :

```
30 IF B=6 THEN PRINT"EH!"  ELSE 20
```

peut s'écrire

```
30 IF B=6 THEN PRINT"EH!"  ELSE GOTO 20
```

mais après THEN ou après ELSE l'instruction GOTO n'est pas obligatoire.

Lire le par. 4 avant d'appuyer sur <RETURN>.

4. IF... THEN... (sans ELSE)

L'instruction IF...THEN... peut être utilisée sans ELSE.

```
30 IF condition  THEN   instructions 1   <--- ce qu'ORIC exécute si la condition
40 instructions 2       <--- ce qu'ORIC exécute si la                est vraie
50 instructions 3            condition n'est pas vérifiée
```

Faire attention aux instructions 1. En effet si la condition est vérifiée, l'ordinateur exécutera les instructions 1. Si celles-ci ne contiennent pas de GOTO, l'ordinateur après avoir exécuté les instructions 1, passera à la ligne suivante et exécutera les instructions 2 (alors que celles-ci ne doivent être en général exécutées que si la condition n'est pas vérifiée).

Pour empêcher l'exécution des instructions 2, si la condition est vérifiée, il suffit à la fin des instructions 1 de mettre GOTO 5Ø (afin de sauter la ligne 4Ø)

Exemple :

```
3Ø IF B=5 THEN PRINT"JUSTE":GOTO 5Ø
4Ø PRINT"FAUX"
5Ø ......
```

5. REPONSE ET PROGRAMME

```
1Ø LET A=5
2Ø IF A<>5 THEN PRINT"!!"ELSE PRINT"?"
3Ø GOTO 1Ø
```

6. INSTRUCTION ON..GOTO..,..,.. (SUR..ALLER EN..,..,..)

```
1Ø INPUT X
2Ø IF X=1 THEN 1ØØ
3Ø IF X=2 THEN 3ØØ
4Ø IF X=3 THEN 5ØØ
```

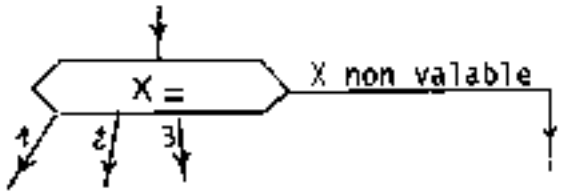
```
1Ø INPUT X
2Ø ON X GOTO 1ØØ,3ØØ,5ØØ
```

Dans les deux programmes si on tape 1, alors X=1 et alors l'ordinateur va à la ligne 1ØØ (premier de la série de nombres 1ØØ,3ØØ,5ØØ). Si on tape 3 l'ordinateur ira en 5ØØ (5ØØ est le 3ème nombre de la série 1ØØ,3ØØ,5ØØ).

Si X=Ø ou X plus grand que le nombre de branchements possibles, l'ordinateur passe à l'instruction suivante. (Si X<Ø il s'arrête avec message d'erreur).

Si X n'est pas entier, l'ordinateur prend sa partie entière.

Sur un organigramme on peut représenter ON X GOTO par :



7. RESUME LECON3 S1

```
 :  permet d'écrire plusieurs instructions sur une même ligne.
REM permet des commentaires, permet des titres dans le listing.
GOTO 2Ø branchement inconditionnel à la ligne 2Ø.
<RESET> permet d'arrêter un programme (sous l'appareil).
IF condition THEN instructions 1        Si condition vérifiée alors instructions 1
instructions 2                          sinon instructions 2
ON A GOTO a,b,c,      si A=1 aller à la ligne a
                      si A=2 aller à la ligne b
                      si A=3 aller à la ligne c
```

8. EXERCICES

Examiner, prévoir, vérifier, *réexaminer si besoin était.*

Programme 1

```
1Ø INPUT A
2Ø PRINT A
3Ø GOTO 1Ø
```

Programme 2

```
1Ø A=5
2Ø GOTO 4Ø
3Ø PRINT"le double de 5 est";2*A
4Ø END
```

Programme 3

Reprendre le programme 1 de la leçon 2 par. 14 en ajoutant la ligne suivante :
65 GOTO 30.

Programme 4

```
10 REM ************************
20 REM VERIFICATION D'UNE ENTREE
30 REM ************************
40 INPUT"Entrer un nombre entre 1 et 10";X
50 IF X<1 OR X>10 THEN 40:REM vérification entrée
60 PRINT X,"est bien compris entre 1 et 10"
70 END
```

Programme 5

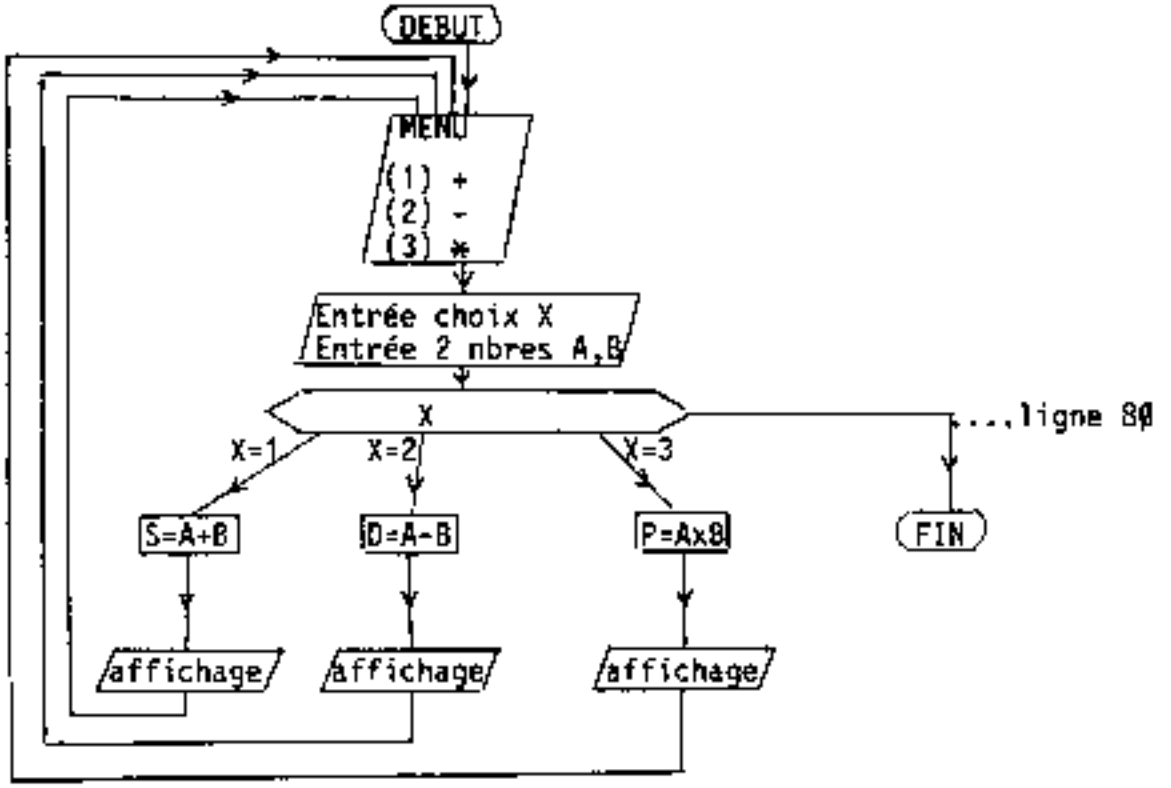
```
10 REM amélioration du jeu "découverte d'un nombre"
20 CLS:INPUT"Introduire un nombre sans que l'autre joueur ne le voie";X
30 CLS
35 INPUT"PROPOSEZ UN NOMBRE";Y
40 IF Y<X THEN PRINT"TROP PETIT":GOTO 35
50 IF Y>X THEN PRINT"TROP GRAND":GOTO 35
60 PRINT"BRAVO VOUS AVEZ TROUVE"
70 END
```

Commentaires : les lignes 40 et 50 indiquent au joueur la position du nombre qu'il a proposé. Remarquer que si Y=X les lignes 40 et 50 ne sont pas exécutées (puisque les conditions ne sont pas vérifiées) et ce sera la ligne 60 qui sera exécutée. Il n'est donc pas nécessaire d'écrire IF Y=X...

Programme 6

```
10 REM Opérations au choix sur 2 nombres
20 CLS:PRINT"MENU":PRINT:PRINT:REM titre et saut de 2 lignes
30 PRINT"(1) ADDITION"
40 PRINT"(2) SOUSTRACTION"
50 PRINT"(3) MULTIPLICATION":PRINT
60 INPUT"VOTRE CHOIX";X
70 PRINT:INPUT"Entrer 2 nombres";A,B
80 ON X GOTO 100,150,200:REM aiguillage
90 END
100 S=A+B:REM début de l'aiguillage 1
110 PRINT:PRINT"La somme des 2 nombres est";S
120 PRINT:PRINT:GOTO 30
150 D=A-B:REM début de l'aiguillage 2
160 PRINT:PRINT"La différence des 2 nombres donne";D
170 PRINT:PRINT:GOTO 30
200 P=A*B:REM début aiguillage 3
210 PRINT:PRINT"La multiplication des 2 nombres donne";P
220 PRINT:PRINT:GOTO 30
```

Commentaires : ce programme est l'expression de l'organigramme suivant :



9. BUT DE LA SECTION 2

Eviter de réécrire des lignes identiques ou très semblales.

Exemple 1

```
10 ....
..
..
40 A1=(3.1)*(3.1)*(3.1)+(3.1)*(3.1)+(3.1)+5 : PRINT A1
50 A2=(3.2)*(3.2)*(3.2)+(3.2)*(3.2)+(3.2)+5 : PRINT A2
60 A3=(3.3)*(3.3)*(3.3)+(3.3)*(3.3)+(3.3)+5 : PRINT A3
70 A4=(3.4)*(3.4)*(3.4)+(3.4)*(3.4)+(3.4)+5 : PRINT A4
80 ....
..
..
```

Commentaire : sur cet exemple les lignes 40 à 70 font le même calcul mais avec des valeurs différentes. Nous allons éviter ce genre de répétition.

Exemple 2

```
10 ....
..
..
30 PRINT:PRINT:PRINT"Lire le paragraphe 1"
40 ....
..
..
70 PRINT:PRINT:PRINT"Lire le paragraphe 2"
..
..
150 PRINT:PRINT:PRINT"Lire le paragraphe 3"
..
200 PRINT:PRINT:PRINT"Lire le paragraphe 3"
..
```

Commentaire : ici les lignes 3Ø, 7Ø, 15Ø et 2ØØ exécutent la même fonction au numéro de paragraphe près. Nous verrons comment éviter cette réécriture.

10. INSTRUCTION DEF FN..(..) (FN = FONCTION)

La fonction définie par cette instruction *doit toujours commencer par* FN. Exemples : FNA(X) FNB(Y) FNC(X).

Le nom de la variable entre parenthèses *importe peu*. On dit que c'est une variable muette. Exemples : DEF FN A(X)=X*X et DEF FN A(Y)=Y*Y définissent la même fonction FN A.

Exemple 1 : soit le programme

```
1Ø DEF FNP(C) = 4*C
2Ø PRINT FNP(2)
3Ø PRINT FNP(3)
4Ø PRINT FNP(4)
5Ø END
```

à la ligne 1Ø la fonction FNP(C) est définie par 4*C (périmètre du carré)
à la ligne 2Ø l'ordinateur remplace C par 2, calcule FNP(2) c'est-à-dire 4*2(il trouve 8) et l'affiche à cause du PRINT
à la ligne 3Ø il remplace C par 3 dans FNP(C) (il trouve 4*3=12) et l'affiche
à la ligne 4Ø même processus, il affiche 16.

Donc si l'on tape RUN on obtient :

```
8
12
16
```

Exemple 2 : en revoyant le par. 9, exemple 1 on constate que l'on a intérêt à définir une fonction DEF FN A(X)=X*X*X*X*X+X+5 et le programme pourra s'écrire :

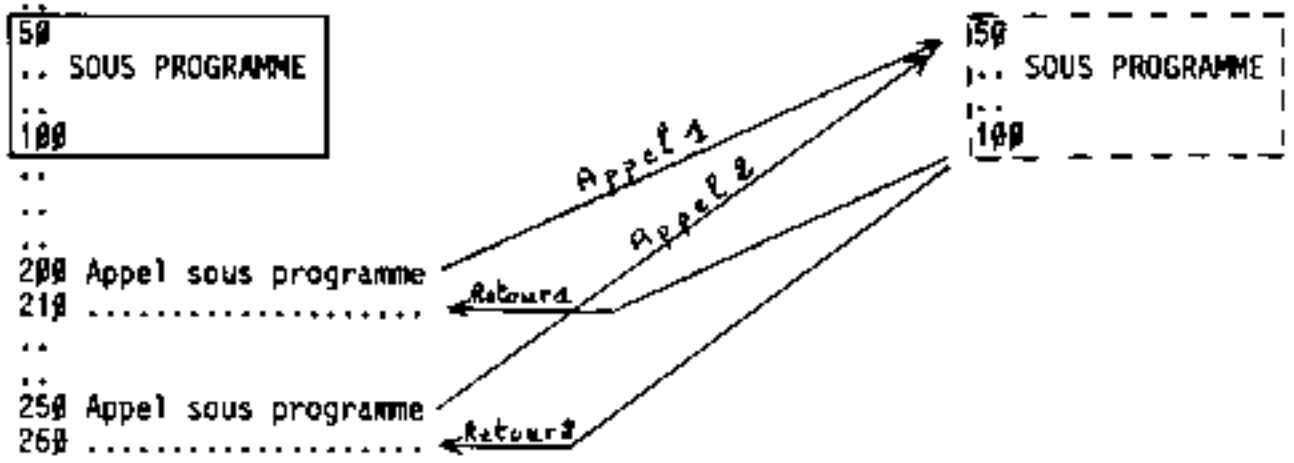
```
1Ø ....
..
..
4Ø DEF FN A(X)=X*X*X*X*X+X+5
5Ø PRINT FN A(3.1):PRINT FN A(3.2):PRINT FN(3.3):PRINT FN A(3.4)
8Ø ....
..
..
```

A la ligne 5Ø l'ordinateur commence par remplacer X par 3.1, puis il calcule FN A(3,1) et enfin il l'affiche. Il recommence le même processus pour 3.2 , pour 3.3 et 3,4.

On peut se servir des fonctions mathématiques d'ORIC (COS,SIN,LOG,ATN...) pour définir une fonction. Exemple : DEF FN A(X)=(EXP(-X))*SIN(X) (e^{-X} sinx).

11. LA NOTION DE SOUS-PROGRAMME

```
1Ø ....
..
..
5Ø
.. SOUS PROGRAMME
..
1ØØ
..
..
..
2ØØ Appel sous programme
21Ø .....
..
..
25Ø Appel sous programme
26Ø .....
..
..
```

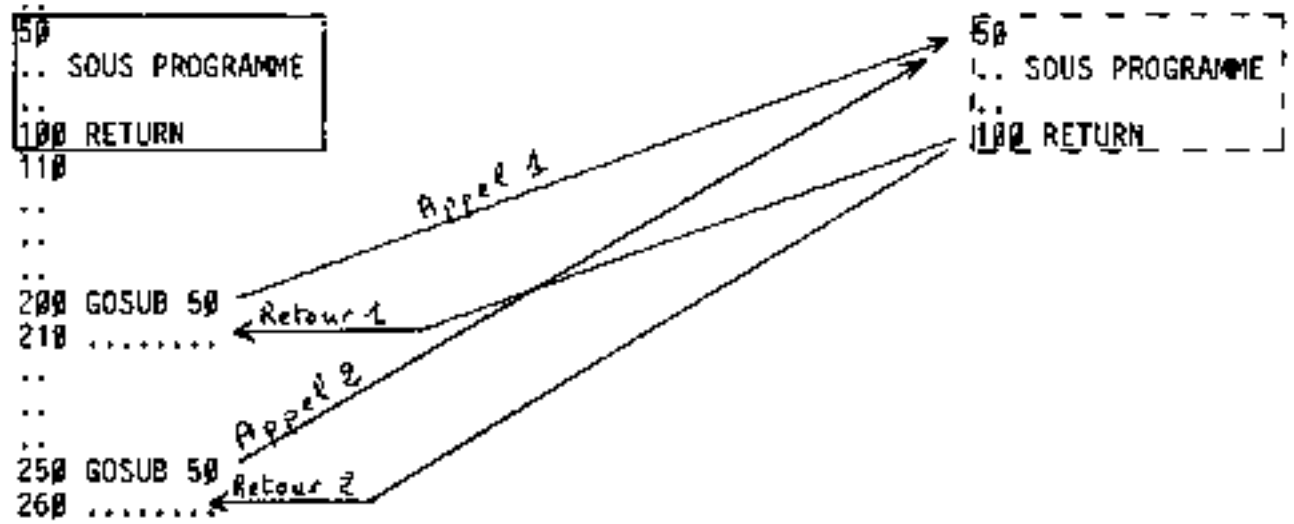


Commentaires : le programme (principal) contient le sous programme (lignes 5Ø à 1ØØ). Ce sous programme est appelé une première fois à la ligne 2ØØ. Une fois le sous programme exécuté, l'ordinateur REVIENT à la PREMIERE INSTRUCTION QUI SUIT L'APPEL du sous programme (ici à la ligne 21Ø). Ce processus se répète à la ligne 25Ø : appel, exécution du sous programme, retour au programme principal (cette fois ligne 26Ø).

12. L'INSTRUCTION GOSUB..RETURN (GOSUB --->GO SUBROUTINE=ALLER au SOUS PROGRAMME RETURN=RETOUR)

Le schéma du par. 11 se traduit par :

```
1Ø ....
..
..
5Ø
.. SOUS PROGRAMME
..
1ØØ RETURN
11Ø
..
..
..
2ØØ GOSUB 5Ø
21Ø .....
..
..
..
25Ø GOSUB 5Ø
26Ø .....
..
..
```



Commentaire : quand l'ordinateur rencontre GOSUB 5Ø à la ligne 2ØØ il mémorise l'endroit où il a trouvé GOSUB 5Ø, il va exécuter le sous programme, puis lorsqu'il rencontre RETURN, revient au bon endroit qu'il avait mémorisé.

En revoyant l'exemple 2 du *paragraphe 9*, on peut utiliser un sous programme très court contenant la phrase "Lire le paragraphe".

```
1Ø
..
..
3Ø P=1:GOSUB 3ØØ
..
..
7Ø P=2:GOSUB 3ØØ
..
..
15Ø P=3:GOSUB 3ØØ
..
..
2ØØ P=4:GOSUB 3ØØ
..
..
3ØØ PRINT:PRINT:PRINT"Lire le paragraphe ";P <--- SOUS PROGRAMME
31Ø RETURN
```

Commentaire : une petite difficulté a été résolue ; celle du numéro du par. A la ligne 3Ø on voulait faire afficher "Lire le paragraphe 1" or le numéro du par. doit changer aux appels suivants; à la ligne 7Ø (par.2), à la ligne 15Ø (par.3),à la ligne 2ØØ (par.4). Pour lever la difficulté on a appelé P le numéro du paragraphe à lire. A la ligne 3Ø on a indiqué le contenu de P *avant* d'appeler le sous programme (afin qu'à la ligne 3ØØ l'ordinateur affiche le *bon* numéro du paragraphe).

13. L'INSTRUCTION ON..GOSUB.....

Les deux programmes suivants sont équivalents :

```
1Ø INPUT X
2Ø IF X=1 THEN GOSUB 1ØØ
3Ø IF X=2 THEN GOSUB 2ØØ
4Ø IF X=3 THEN GOSUB 3ØØ
..
..
1ØØ REM sous programme 1
..  ..
..  RETURN
..
..
2ØØ REM sous programme 2
..  ..
..  RETURN
..
..
3ØØ REM sous programme 3
..  ..
..  RETURN
```

```
1Ø INPUT X
2Ø ON X GOSUB 1ØØ,2ØØ,3ØØ
3Ø ...
..
..
..
1ØØ REM sous programme 1
 .. ..
 .. RETURN
 ..
 ..
2ØØ REM sous programme 2
 .. ..
 .. RETURN
 ..
 ..
3ØØ REM sous programme 3
 .. ..
 .. RETURN
```

Remarquer les instructions RETURN qui permettent dans le programme de droite le *retour à la ligne 3Ø*.

14. RESUME LECON3 S2

DEF FN A(X)	permet de définir la fonction FN A(X)
GOSUB ..	appel d'un sous programme devant contenir comme dernière instruc-
RETURN	tion RETURN
ON A GOSUB a,b	si A=1 appel du sous programme en ligne a
	si A=2 appel du sous programme en ligne b
PI	mot réservé contenant le nombre =3,14..
ZAP	tir laser } (commandes propres à ORIC)
EXPLODE	explosion }

15. EXERCICES LECON3 S2

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
10 REM Aire d'un cercle
15 CLS
20 DEF FN A(R)=INT(PI*R*R)
30 PRINT FN A(1)
40 PRINT FN A(2)
50 PRINT FN A(4/3)
60 END
```

Commentaire : réessayer en remplacant la ligne 20 par
DEF FN A(R)=PI*R*R

Programme 2

```
10 REM arrondi à 1 ou 2 décimales
20 DEF FN A1(X)=INT(X*10+0.5)/10
30 DEF FN A2(X)=INT(X*10*10+0.5)/(10*10)
40 CLS
50 INPUT B
60 PRINT B;"arrondi à 1 et 2 décimales="
70 PRINT FN A1(B)
80 PRINT FN A2(B)
90 END
```

Commentaire : B nombre avec décimales!!
B=6.179 par exemple.

Pour une question de rapidité on conseille de 'placer" *les sous programmes "au début" du programme principal.*

Si l'ordinateur rencontre un RETURN sans avoir été envoyé par un GOSUB, il affiche un message d'erreur RETURN WITHOUT GOSUB c'est-à-dire RETURN sans GOSUB. C'est souvent parce que l'on a mis GOTO 50 au lieu de GOSUB 50.

Programme 3

```
10 REM test de multiplication
20:
30 CLS
40 INPUT"Entrer 2 nombres";A,B
50 P=A*B
60 PRINT:PRINT"Que vaut";A;"*";B;
70 INPUT R
80 IF R=P THEN GOSUB 120:REM S/P bonne réponse
90 IF R<>P THEN GOSUB 180:REM S/P mauvaise réponse
100 GOTO 40
110:
120 REM S/P bonne réponse
130 PRINT:PRINT
140 ZAP:ZAP
150 PRINT"BRAVO!"
160 RETURN:REM fin de sous programme
170:
180 REM S/P mauvaise réponse
190 PRINT:PRINT
200 EXPLODE:EXPLODE
210 PRINT"FAUX"
220 RETURN:REM fin de sous programme.
```

Commentaires : à la ligne 100 il fallait mettre END (ou comme ici GOTO 40), sinon à un retour de sous programme, l'ordinateur viendrait exécuter la ligne 120.

Par souci de clarté pour ce premier exercice de sous programme, les deux sous programmes ont été mis à la fin.

Les deux lignes 80 et 90 peuvent être remplacées par
80 IF R=P THEN GOSUB 120 ELSE GOSUB 180.

A la ligne 60 le ; après le 8 permet une réponse sur la même ligne que la question.

Programme 4

```
10 REM Calcul d'un prix T.T.C.
20:
30 REM PH=prix hors taxe
40 REM PT=prix T.T.C.
50 REM C=code T.V.A.
60 DEF FNA2(X)=INT(X*100+0.5)/100
70:
80 GOTO 250:REM pour sauter les sous programmes
90:
100 REM S/P taux 7%
110 PT=PH*1.07
120 PRINT FNA2(PT)
130 RETURN
140:
150 REM S/P taux 18.6%
160 PT=PH*1.186
170 PRINT FNA2(PT)
180 RETURN
190:
200 REM S/P taux 33%
210 PT=PH*1.33
220 PRINT FNA2(PT)
230 RETURN
240:
250 CLS
260 INPUT"Prix H.T.";PH
270 PRINT
280 INPUT"Code TVA:7%--->1;18.6%--->2;33%--->3";C
290 PRINT:PRINT
300 ON C GOSUB 100,150,200:REM aiguillage multiple
310 GOTO 260
```

Commentaires : remarquer les différents "modules".
L'instruction de la ligne 80 est toujours nécessaire lorsque les sous programmes sont "au début".
On peut remplacer les lignes 120,170 et 220 par GOSUB 95 et créer un sous programme d'affichage arrondi

```
95 REM S/P affichage arrondi
96 PRINT FNA2(PT)
97 RETURN
```

Ceci nous montre qu'*un sous programme peut lui-même appeler un autre sous programme.*

LECON 4

1. LA FONCTION RND(1) (RND ---> RANDOM = HASARD)

Chaque fois que RND(1) est utilisée, elle fournit un nombre aléatoire (au hasard) compris entre 0 et 1. En fait on a $0 \leqslant RND(1) < 1$.

Remarquons que $0 \leqslant 6*RND(1) < 6$ et que $6*RND(1)+1$ est tel que $1 \leqslant 6*RND(1)+1 < 7$ et qu'alors la partie entière de ce nombre vérifie :

$1 \leqslant INT(6*RND(1)+1) \leqslant 6$

C'est donc un entier choisi au hasard entre 1 et 6. Cette fonction INT(6*RND(1)+1) simule donc un lancer d'un dé.

Si l'on cherche à générer des nombres aléatoires compris entre 2 entiers A et B (A<B) on pose
X=INT(A+(B-A+1)*RND(1)) et on aura $A \leqslant X \leqslant B$.

2. LES VARIABLES INDICEES : A(I)

On peut par exemple utiliser des variables indicées si l'on a 10 notes à mettre en mémoire. On utilisera N(1) pour la 1ère note, N(2) pour la 2ème,..., N(10) pour la 10ème. On aurait pu utiliser N(0), N(1), ..., N(9).

Voici un programme permettant d'entrer 4 notes et d'en faire la moyenne :

```
10 INPUT N(1):INPUT N(2):INPUT N(3):INPUT N(4)
20 M=(N(1)+N(2)+N(3)+N(4))/4
30 PRINT"La moyenne des 4 notes est";M
```

Comme pour les autres variables l'ordinateur ne prend en compte que les 2 premières lettres se trouvant avant les parenthèses ; ainsi PARIS(1) et PAPA(1) sont identiques.

3. LA DECLARATION DIM (DIM = DIMENSION)

Il n'est pas nécessaire d'utiliser DIM si on n'utilise que A(0)...A(10). Un oubli de la déclaration DIM entraîne un message d'erreur si on veut utiliser A(11). Même processus si on veut utiliser A(50) alors que l'on a déclaré DIM A(49).

La déclaration s'écrit :

```
10 DIM A(20)
20 ...
..
..
```

4. PROGRAMMES COMPTEUR

Programme 1

```
10 C=0
20 C=C+1:PRINT C
30 IF C=4 THEN END
40 GOTO 20
```

Programme 2

```
10 C=6
20 PRINT C:C=C-1
30 IF C=0 THEN END
40 GOTO 20
```

5. RESUME LECON4 S1

RND(1)	génère un nombre aléatoire de [0;1[
A(I)	variable indicée (indice I)
DIM	déclaration de dimension de variables indicées (nécessaire si I ⩾11)
COMPTEUR	permet de dénombrer un phénomène.

6. EXERCICES LECON4 S1

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1 : Amélioration du jeu découverte d'un nombre (programme 5 par. 8 leçon 3). Cette fois-ci le nombre (compris entre 1 et 100) est choisi au hasard par l'ordinateur !

```
10 REM Découverte d'un nombre
20 X=INT(1+100*RND(1))
30 CLS:PRINT"J'ai choisi un entier entre 1 et 100"
40 PRINT:PRINT"Trouvez-le"
50 INPUT Y
60 IF Y<X THEN PRINT"TROP PETIT":GOTO 50
70 IF Y>X THEN PRINT"TROP GRAND":GOTO 50
80 PRINT"VOUS L'AVEZ TROUVE.BRAVO"
90 END
```

Programme 2 : Ajoutons au programme 1 un "compteur" permettant de savoir en combien d'essais on trouve le nombre choisi par l'ordinateur. Il suffit de le modifier ainsi :

```
50 C=C+1:INPUT Y
80 PRINT"VOUS L'AVEZ TROUVE EN ";C;" ESSAIS"
```

Commentaire : le compteur est incrémenté (augmenté) à chaque passage dans l'INPUT.

Programme 3

```
10 REM Moyenne de 4 notes avec arrondi à 2 décimales
20 DEF FN A2(X)=INT(X*100+0.5)/100
30 REM DIM non nécessaire car A(1)..A(4)
40 INPUT A(1):INPUT A(2):INPUT A(3):INPUT A(4)
50 M=(A(1)+A(2)+A(3)+A(4))/4
60 PRINT "La moyenne est "; FN A2(M)
70 END
```

Commentaire : ne pas confondre A2 et A(2).

Programme 4 : Toujours une moyenne, mais pour 12 notes.

```
10 REM Moyenne de 12 notes
20 DEF FNA2(X)=INT(X*100+0.5)/100
30 DIM A(12)
40 I=I+1:REM  compteur
50 INPUT A(I)
60 IF A(I)<0 OR A(I)>20 THEN 50:REM vérification entrée
65 S=S+A(I):REM somme cumulée
70 IF I<12 THEN 40 : REM retour compteur + entrée
80 M=S/12
90 PRINT"La moyenne est ";FNA2(M)
100 END
```

Commentaire : Pour 12 notes, il convient d'utiliser une boucle qui doit être parcourue 12 fois (pour entrer les 12 notes). Le compteur (ligne 4Ø) et le test du nombre de boucles (ligne 7Ø) suffisent.
A la ligne 6Ø on a testé si la note entrée n'était pas erronée.
A la ligne 65 on profite de la boucle pour additionner à chaque passage la note introduite.

Programme 5 : Programme de tri de 20 nombres.

```
1Ø REM Tri RIPPLE
2Ø DIM A(2Ø)
3Ø REM A(I)=table des 20 nombres
4Ø REM INV=témoin d'inversion
5Ø REM X=variable auxiliaire de transit
6Ø REM I=compteur
7Ø:
8Ø I=I+1:A(I)=INT(1+5Ø*RND(1))
9Ø IF I<2Ø THEN 8Ø
1ØØ:
11Ø INV=Ø
12Ø I=I+1
13Ø IF A(I+1)<A(I)THEN X=A(I):A(I)=A(I+1):A(I+1)= A(I):REM INVERSION DE
   A(I) et A(I+1):INV=1
14Ø IF I<19 THEN 12Ø
15Ø IF INV=1 THEN 11Ø
16Ø:
17Ø J=J+1:PRINT A(J):IF J<2Ø THEN 17Ø
18Ø END
```

Commentaire : Les lignes 8Ø et 9Ø servent à entrer 2Ø nombres aléatoires compris entre 1 et 5Ø.
Le tri (qui n'est pas un tri rapide) se déroule comme suit :
1ère étape : comparaison de A(1) et A(2) : le plus grand placé en A(2)
2ème étape : comparaison de A(2) et A(3) : le plus grand placé en A(3)
.....
.....
.....
19ème étape : comparaison de A(19) et A(2Ø) : le plus grand placé en A(2Ø)

On recommence s'il y a eu inversion (grâce au témoin d'inversion INV). A la fin du processus A(1) est le plus petit et A(2Ø) le plus grand.

Programme 6 : Introduction de 2Ø nombres au choix. On modifie alors les lignes 8Ø et 9Ø

```
8Ø I=I+1:INPUT A(I)
9Ø IF I<2Ø THEN 8Ø.
```

7. LES REPETITIONS

Exemple 1 : Si on a à introduire 2Ø nombres au clavier, il y a donc une instruction INPUT à répéter 2Ø fois. Nous sommes dans le 1er cas : le nombre de répétitions est CONNU.

Exemple 2 : On a à introduire des nombres compris entre 0 et 10, sans savoir, à chaque utilisation du programme, combien de nombres on introduit exactement en tout. On peut convenir de répéter une instruction INPUT jusqu'à ce que l'on introduise un nombre <0 (ce nombre sera volontairement introduit afin d'arrêter les INPUT). Nous sommes dans le 2ème cas : le nombre de répétitions est INCONNU et on a *une condition d'arrêt* (le nombre introduit est <0).

8. L'INSTRUCTION FOR..TO../NEXT (FOR=POUR TO=A NEXT=LE SUIVANT)

```
..                                  ..
..                                  ..
30 FOR I=1 TO 20                    30 POUR I=1 A 20
40: INPUT A(I)                      40  INPUT A(I)
50 NEXT I                           50 LE I SUIVANT
60 ...                              60 ...
..                                  ..
```

Comprenons le processus

Ligne 30 : "POUR I=1 A 20"
ORIC commence par donner à I la valeur 1 et passe à la prochaine instruction.

Ligne 40 : l'INPUT est exécuté.

Ligne 50 : "LE I SUIVANT"
ORIC passe à la valeur suivante de I. Or I contenait la valeur 1. La ligne 50 fait passer la valeur de I de 1 à 2. Ensuite ORIC compare cette valeur à la valeur finale de I donnée à la ligne 30. Cette valeur est 20, elle n'est pas dépassée par la valeur actuelle de I (qui est 2). ORIC retourne alors aux instructions à répéter ; ici la ligne 40.

Ligne 40 : l'INPUT est exécuté.

Ligne 50 : "LE I SUIVANT"
I valait 2, il passe maintenant à 3. Or 3 est encore inférieur ou égal à la valeur limite de I, qui est 20. ORIC retourne à la ligne 40.

Ligne 40 : ...

Ligne 50 : ...

...

Le processus se répète 20 fois (jusqu'à ce que I arrive à 20) alors, à la

Ligne 50 : "LE I SUIVANT"
I valant 20, il passe à 21. Or 21 > 20 qui est la valeur limite de I. ORIC passe alors à l'instruction suivante (ici la ligne 60).

On a donc le schéma suivant :

```
FOR I = ? TO ??
    +----------+
    |          |<--- groupe d'instructions à REPETER autant de fois
    |          |     que I varie de ? à ??
    +----------+
NEXT I
```

9. L'INSTRUCTION FOR.TO./NEXT avec STEP (STEP = PAS de)

Le processus d'une boucle FOR.TO./NEXT avec présence de STEP 2 est en fait identique au processus sans STEP, à la seule différence suivante : lorsqu'ORIC rencontre l'instruction NEXT I, il prend l'ancienne valeur de I et l'augmente de 2 (au lieu de 1). Naturellement, le nombre de répétitions n'est plus le même.

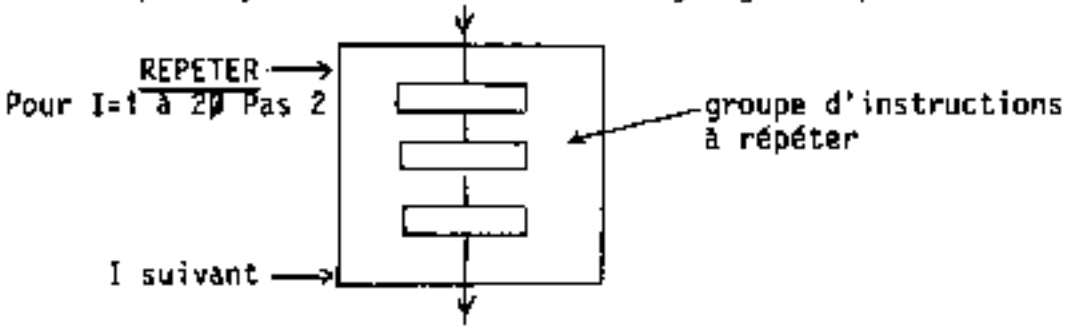
Valeurs de S dans STEP S :

S = 1 : "pas" de 1. (On peut ne pas écrire du tout STEP 1)

S < 0 : "pas" négatif. FOR I = 20 TO 10 STEP -1 fera varier I de 20 à 10 en passant par 20,19,18, ...,11,10.

S non entier : Il peut être utile d'utiliser un pas de 0.5 (par exemple pour un tracé de courbes).

On peut symboliser cela dans un organigramme par :



Remarque : les blancs ne sont pas nécessaires entre FOR I...
30FORI=1TO20STEP2 sera aussi bien accepté que 30 FOR I=1 TO 20 STEP 2.

10. REPONSES A DONNER AU CLAVIER

Programme 1

```
30 FOR X=5 TO 10
..
..
60 NEXT X
```

Programme 2

```
30 FOR Y=1 TO 10 STEP 2
..
..
70 NEXT Y
```

11. L'INSTRUCTION REPEAT/UNTIL .. (REPEAT = REPETER / UNTIL = JUSQU'A)

Ici le processus est encore plus simple : le groupe d'instructions à répéter (ici I=I+1 et INPUT A(I))est exécuté tant que A(I) n'est pas <0. (Le compteur I=I+1 a pour but de changer la variable d'entrée dans l'INPUT à chaque boucle).

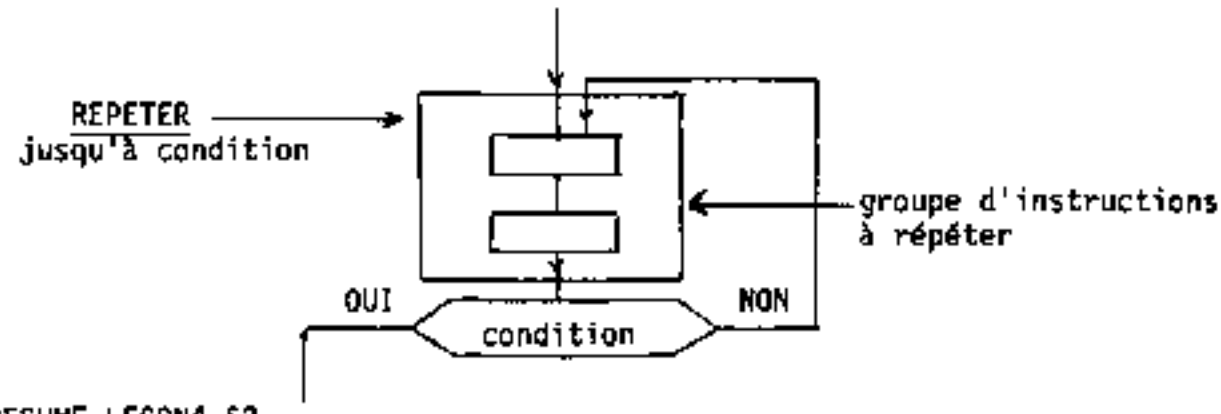
On a donc le schéma suivant :

REPEAT

<--- groupe d'instructions à REPETER, JUSQU'A ce que la condition soit vérifiée

UNTIL Condition

qui est la traduction dans un organigramme de :



12. RESUME LECON4 S2

FOR.TO.STEP./NEXT	Répétition des instructions entre FOR et NEXT, le PAS étant donné par STEP.
REPEAT/UNTIL Condition	Répétition des instructions entre REPEAT et UNTIL jusqu'à ce que la condition soit vérifiée.
KEY$	Mémorise la touche pressée sans arrêter le programme.
WAIT N	Suspend l'exécution du programme pendant N/100 secondes.
PING	Son de cloche.
SHOOT	Tir de fusil.

13. EXERCICES LECON4 S2

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
10 FOR I=4 TO 7
20:  PRINT"ORIC"
30 NEXT I
```

Programme 2

```
10 PRINT"ORIC"
20 FOR I=1 TO 2000
30 NEXT I
40 PRINT"ORIC"
```

Programme 3

```
10 PRINT"ORIC"
20 WAIT 300
30 PRINT"ORIC"
```

Commentaire : L'instruction WAIT N permet d'arrêter l'exécution du programme pendant N/100 secondes. Cela est plus pratique que la boucle "vide" du programme 2.

Programme 4

```
10 REPEAT
20:  ZAP:WAIT 50
30:  PING:WAIT 50
40 UNTIL KEY$="A"
```

Programme 5

```
10 REPEAT
20:  FOR I=1 TO 10
30:       SHOOT
40:       WAIT 10
50:  NEXT I
60 UNTIL KEY$="S"
```

Programme 6

```
10 REM tracé de SINUS
20 REM X=compteur
30 REM N=nombre de subdivisions
40 CLS
50 INPUT"nbre de subdivisions de
       (0,2 PI)";N
60 CLS
70 FOR X=0 TO 2*PI STEP(2*PI)/N
80:  PRINT TAB(20+19*SIN(X))"*"
90 NEXT X
100 END
```

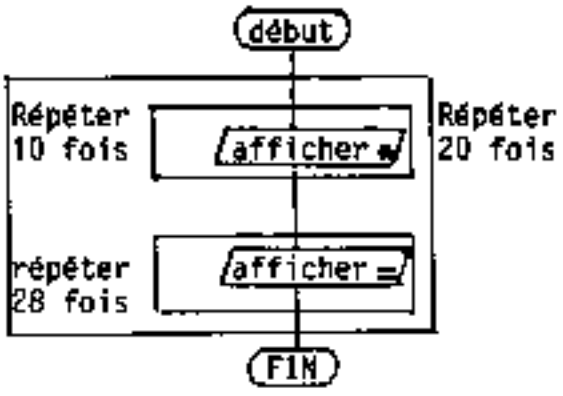
Programme 7

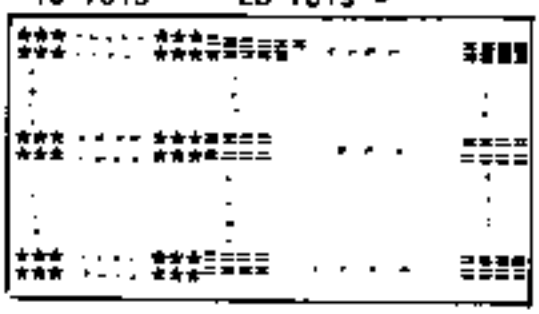
```
10 REM somme de 11 nombres
20 DIM A(11)
30 REM S=somme cumulée
40 CLS
50 PRINT"entrer les 11 nombres"

60 FOR I=1 TO 11
70   INPUT A(I)
80   S=S+A(I)
90 NEXT I
100 PRINT:PRINT
110 PRINT"la somme est";S
120 END
```

Programme 8

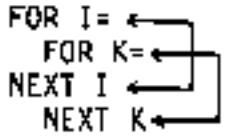
On veut réaliser la figure suivante
on utilise l'organigramme






```
10 CLS
20 FOR I=1 TO 20
30   FOR J=1 TO 10
40    PRINT"*";
50   NEXT J
60   FOR K=1 TO 28
70    PRINT"=";
80   NEXT K
90 NEXT I
100 END
```

Commentaire : nous avons des boucles FOR.TO./NEXT imbriquées (l'une dans l'autre).
Ne jamais utiliser la même variable compteur dans ce cas.
Ne jamais utiliser de boucles "à cheval" l'une sur l'autre



Programme 9

```
10 REM Maximum d'une liste de nombres positifs
15 M=0
20 REPEAT
30 INPUT X
40 IF X>M THEN M=X
50 UNTIL X=0
60 PRINT"Maximum=";M
70 END
```

LECON 5

1. LES CHAINES DE CARACTERES

Une chaîne peut contenir un ou plusieurs ESPACES. La chaîne "On a" est constituée du O, du n, d'un espace, du a.

Une chaîne est écrite ENTRE GUILLEMETS. Ainsi 123 est le nombre cent vingt trois alors que "123" est la chaîne constituée du 1, du 2, du 3.

2. VARIABLES ALPHANUMERIQUES (ou VAR. CHAINE de CARACTERES)

Tout comme les variables numériques, seuls les 2 premiers caractères sont pris en compte (le 1er devant être une lettre majuscule). LIBELLE$ et LIT$ représentent ainsi la même variable.

Toute variable alphanumérique est considérée A PRIORI comme VIDE, (c'est-à-dire la chaîne "" et non " " qui contient un espace). C'est le cas chaque fois que l'on met sous tension l'ORIC si on utilise CLOAD ou RUN ou CLEAR ou NEW.

3. RESUME LECON5 S1

Chaîne de caractères = succession de caractères du clavier.	
LET A$="EH"	affectation de la chaîne "EH" à la variable alphanumérique A$.
A$(I)	est une variable alphanumérique indicée.
DIM A$(5Ø)	déclaration permettant de réserver de la place mémoire pour A$(Ø), A$(1),..., A$(5Ø).
"ORDI"+"NATEUR"	donne la chaîne concaténée "ORDINATEUR".
<	permet la comparaison de deux chaînes, en utilisant l'ordre alphabétique.
LEN(A$)	calcule puis mémorise la longueur de la chaîne A$ (LEN→LENGTH = LONGUEUR).

4. EXERCICES LECON5 S1

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
1Ø A$="ORIC":B$="1"
2Ø C$=" ":REM 1 espace
3Ø D$="":REM chaîne vide
4Ø CLS
5Ø PRINT A$+C$+B$
6Ø PRINT A$+D$+B$
7Ø END
```

Moralité : la chaîne C$ n'est pas vide!

Programme 2

```
1Ø CLS
2Ø INPUT"Votre nom";NOM$
3Ø INPUT"Votre prénom";PRENOM$
4Ø PRINT:PRINT
5Ø PRINT"Vous vous appelez ";
   NOM$, PRENOM$
6Ø END
```

Programme 3

```
10 REM titre centré
20 CLS
30 INPUT"Quel titre";TITRE$
40 CLS
50 PRINT TAB((38-LEN(TITRE$))/2) TITRE$
60 END
```

Commentaire : revoir le paragraphe 13 de la leçon 2 pour corriger éventuellement le programme ci-dessus (TAB(12+N)...TAB(N)).

Programme 4

```
10 REM affichage d'un mot souligné puis encadre
20 CLS:INPUT"Introduire un mot ";A$
30 PRINT:PRINT:PRINT TAB(15)A$
40 PRINT TAB(14)" ";REM Positionnement pour soulignement
50 FOR I=1 TO LEN(A$):PRINT"*";:NEXT I
60 PRINT:PRINT:PRINT:PRINT
70 PRINT TAB(13)" ";
80 FOR I=1 TO 2+LEN(A$):PRINT"*";:NEXT I:PRINT
90 PRINT TAB(14)"*";:PRINT A$;:PRINT"*"
100 PRINT TAB(13)" ";
110 FOR I=1 TO 2+LEN(A$):PRINT"*";:NEXT I
120 END
```

Commentaire :

ligne 30 : affichage du mot introduit

ligne 40 : permet de positionner le "curseur" pour un prochain affichage, juste sous la 1ère lettre du mot A$ (le; est nécessaire)

ligne 50 : soulignement (remarquer le; . Essayer sans ce;)

lignes 70 et 80 : positionnement et surlignement

```
**************  <--- lignes 70 et 80
*     A$     *  <--- ligne 90
**************  <--- lignes 100 et 110
```

Programme 5

```
10 REM  introduction de 2 mots et leur affichage dans l'ordre alphabétique
20 CLS:INPUT"le 1er mot";A$:INPUT"le 2ème mot";B$
30 IF A$<B$ THEN M1$=A$ : M2$=B$ ELSE M1$=B$:M2$=A$
40 PRINT:PRINT;
50 PRINT"Dans l'ordre alphabétique on a ";M1$,M2$
60 END
```

Programme 6

```
10 REM Tri de 11 noms introduits
20 DIM NOM$(11)
30 CLS:PRINT"introduire chaque nom suivi de <RETURN>"
40 FOR I=1 TO 11:INPUT NOM$(I):NEXT I
50 INV=0
60 FOR I=1 TO 10
70 IF NOM$(I+1)<NOM$(I) THEN X$=NOM$(I):NOM$(I)=NOM$(I+1):NOM$(I+1)=X$:INV=1
80 NEXT I
90 IF INV=1 THEN 50
100 FOR I=1 TO 11:PRINT NOM$(I):NEXT I
110 END
```

Commentaire : c'est la technique du tri RIPPLE, déjà vue à la leçon 4, paragraphe 6, programme 5. On a utilisé ici des boucles FOR - NEXT qui n'avaient pas encore été vues à la leçon 4. A part cela, le principe est identique.

Tester le programme en utilisant à la fois des majuscules et des minuscules et vous aurez des surprises.

5. LEFT$(A$,n) et RIGHT$(A$,n) (LEFT = GAUCHE RIGHT = DROITE)

De façon analogue à LEFT$(A$,n), la fonction RIGHT$(A$,n) permet d'extraire de A$, la chaîne constituée de ses n DERNIERS caractères.

Si A$="ORIC 1" alors PRINT RIGHT$(A$,3) affichera C 1.

6. STR$ (nombre) et VAL (chaîne) (STR-→STRING = CHAINE VAL-→VALUE = VALEUR)

Remarquer que toute fonction dont le résultat est une chaîne est affectée d'un $.

Si A$ contient des chiffres et des lettres, ou des lettres exclusivement, VAL(A$) n'a pas d'intérêt.

PRINT VAL("AVRIL 1983") affichera 0.
PRINT VAL("12AVRIL") affichera 12

Précisons que VAL ignore les espaces.

7. LE CODE ASCII

Une correspondance entre les caractères du clavier et leur code ASCII est donnée Appendice D du Manuel ORIC.

Ainsi PRINT ASC("[") affichera 91.

Le code ASCII sert aussi à la fonction de comparaison des chaînes <. En fait l'ordinateur compare le code ASCII de chaque lettre. Remarquer que les lettres minuscules ont un code supérieur à celui des majuscules. On comprend maintenant pourquoi il ne fallait pas mixer majuscules et minuscules pour faire du tri.

8. RESUME LECON5 S2

LEFT$(A$,n)	mémorise les n premiers caractères de A$
RIGHT$(A$,n)	mémorise les n premiers caractères de A$
MID$(A$,i,n)	mémorise n caractères de A$, à partir du ième
STR$(nombre)	associe au nombre la chaîne constituée par ce nombre
VAL(chaîne)	associe à la chaîne de chiffres sa valeur numérique
ASC(caractère)	mémorise le code ASCII du caractère
CHR$(entier de [32,127])	associe le caractère correspondant à l'entier (suivant le code ASCII).

9. EXERCICES LECON5 S2

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
10 REM affichage clavier
15 CLS
20 FOR I=32 TO 127
30 PRINT CHR$(I);
40 NEXT I
50 END
```

Programme 2

```
10 REM affichage d'un mot à l'envers
20 INPUT"Introduire un mot ";R$
30 PRINT:PRINT
40 FOR X=LEN(R$) TO 1 STEP-1
50:  PRINT MID$(R$,X,1);
60 NEXT X
70 END
```

Commentaire : lignes 40 à 60 on extrait les lettres une à une en commençant par la fin et on les affiche.

Programme 3

```
10 REM compteur de A dans un mot
20 REM I=compteur des A
30 CLS
35 INPUT"Introduire un mot";R$
40 PRINT:PRINT
50 I=0
60 FOR X=1 TO LEN(R$)
70 IF MID$(R$,X,1)="A"THEN I=I+1
75 NEXT X
80 PRINT"il y a ";I;" A";" dans";R$
90 END
```

Commentaire : lignes 60 à 75 si la lettre extraite est un A, on incrémente (augmente) le compteur, sinon on passe à une nouvelle extraction de lettre.

Programme 4

```
10 REM recherche d'un nom par ses pre-
      mières lettres
20 CLS
30 PRINT"Introduire 8 noms":PRINT
40 FOR I=1 TO 8:INPUT N$(I):NEXT
50 CLS
60 PRINT"Introduisez les premières lettres
   d'un nom que vous recherchez"
70 PRINT:INPUT R$:PRINT
75 L=LEN(R$)
80 FOR I=1 TO 8
90 IF R$=LEFT$(N$(I),L) THEN PRINT N$(I)
100 NEXT I
110 END
```

Commentaire : on compare les premières lettres proposées aux premières lettres des noms que vous avez introduits grâce à LEFT$.

Programme 5

```
10 REM création d'une chaîne aléatoire de 4 lettres
20 FOR I=1 TO 4:A(I)=INT(65+26*RND(1)):NEXT I
30 FOR J=1 TO 4:A$(J)=CHR$(A(J)):NEXT J
40 B$="":FOR K=1 TO 4:B$=B$+A$(K):NEXT K
50 PRINT"Voilà une chaîne aléatoire";B$
60 END
```

Commentaire : on a volontairement décomposé en 3 parties. Ligne 20 constitution de 4 nombres aléatoires entre 65 et 90 (codes ASCII de A et Z). Ligne 30 constitution des lettres correspondant à ces codes. Ligne 40 concaténation des lettres pour former une chaîne.

LECON 6

1. L'INSTRUCTION GET (GET = OBTENIR)

Soit le programme :

```
1Ø GET A$
2Ø IF A$="1" THEN INK2 ELSE INK7
```

à la ligne 1Ø, ORIC
s'arrête (comme pour INPUT) ;
n'affiche pas de point d'interrogation ;
attend l'introduction d'un caractère (n'importe lequel y compris <RETURN>).
Dès qu'une touche est pressée, ORIC, sans afficher le caractère introduit, passe à l'instruction suivante (ici la ligne 2Ø), sans attendre <RETURN>.

L'instruction GET ne permet pas les variables numériques.

GET B entrainera un message d'erreur. Pour obtenir l'équivalent d'un GET numérique, il suffit de tester si le caractère introduit dans A$ correspond bien à un chiffre (Ø à 9) et ensuite de convertir A$ en un nombre à l'aide de la fonction VAL (ce sera l'objet d'un exercice).

L'instruction GET A$ comme KEY$ (LECON4 S2) ne saisissent qu'un caractère du clavier, mais GET A$ fait stopper le programme pour attendre un caractère, alors que KEY$ permet le déroulement du programme en attendant la frappe d'un caractère.

L'instruction GET est elle aussi utilisée dans des jeux.

2. L'INSTRUCTION READ.,./DATA (READ = LIRE DATA = DONNEES)

Une instruction READ suppose toujours l'existence d'une déclaration DATA (n'importe où dans le programme).

Il peut exister plusieurs lignes DATA. A la première instruction READ rencontrée, l'ordinateur ira lire les données demandées (à la première ligne DATA). A la seconde instruction READ, ORIC ira lire les données suivantes sur la même ligne DATA (s'il reste des données) ou éventuellement à la prochaine ligne DATA.

Exemple :

```
DATA 1,2,3,4,5,6
...
...
READ A,B,C      (1er READ rencontré). Lecture des 3 premières données de DATA
...             (affectation de 1 à A, de 2 à B, de 3 à C).
...
...
READ X,Y        (2ème READ rencontré). Lecture des 2 données suivantes, donc
                affectation de 4 à X et de 5 à Y.
```

Si une demande de lecture est faite, alors que toutes les données ont déjà été lues, un message d'erreur (OUT OF DATA = MANQUE DE DONNEES) est affiché et ORIC s'arrête.

Les données et les variables contenues dans READ doivent être synchronisées (une donnée "chaîne" ne peut être affectée à une variable numérique). Une instruction READ J avec une ligne DATA "LUNDI",8 entraînera un message d'erreur puisque "LUNDI", qui est une chaîne, ne peut être affectée à J qui est une variable numérique.

Remarquons enfin qu'une donnée chaîne peut être écrite dans une ligne DATA sans guillemets. Ainsi DATA JANVIER,.,. et DATA "JANVIER",.,. contiennent la même première donnée.

3. REPONSE AU CLAVIER

Programme

```
1Ø DATA 5,4,3,2,1
2Ø READ A
3Ø PRINT A
4Ø GOTO 2Ø
```

4. L'INSTRUCTION RESTORE (RESTORE = REVENIR au DEBUT)

Si l'on veut faire lire puis afficher les 3 premiers jours de la semaine, on peut écrire :

```
1Ø DATA Lundi,Mardi,Mercredi
2Ø READ A$,B$,C$
3Ø PRINT A$,B$,C$.
```

Si on veut utiliser plusieurs fois ces 3 lignes dans un programme, on peut en faire un sous-programme (en rajoutant 4Ø RETURN) qu'on appellera chaque fois que nécessaire. Montrons que cela est insuffisant :

Au premier appel du sous-programme (par un GOSUB 2Ø), la ligne 2Ø entraînera la lecture des 3 premières données et la ligne 3Ø les affichera.

Au deuxième appel du sous-programme, la ligne 2Ø demandera la lecture de 3 données suivantes. N'ayant plus de données à lire (les 3 seules données ont été lues au 1er appel), ORIC affichera un message d'erreur.

Pour éviter cette situation, il suffit d'indiquer que l'on veut relire depuis le début. On ajoute donc la ligne 15 (15 RESTORE) et le sous-programme ainsi modifié s'écrit :

```
1Ø DATA Lundi,Mardi,Mercredi
15 RESTORE              <--- on se replace au début de la liste des données
2Ø READ A$,B$,C$
3Ø PRINT A$,B$,C$
4Ø RETURN
```

On voit donc que la manipulation READ.,. DATA.,. RESTORE est quelque peu délicate. Cette instruction ne devra donc être utilisée que si les données sont stables.

5. RESUME LECON6 S1

GET A$	ne prend qu'un seul caractère, n'attend pas <RETURN>, n'affiche pas ce que l'on tape. Pas de GET A.
DATA/READ	utilisable pour données stables. Les données dans DATA doivent être synchronisées avec les variables contenues dans READ.
RESTORE	permet de se replacer au début de la liste des données.
SPC(n)	permet avec PRINT de créer n espaces blancs entre deux affichages.

6. EXERCICES LECON6 S1

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
10 REM boucle sans fin
20 GET A$
30 PRINT A$
40 GO TO 20
```

Commentaire :<RESET> est bien utile..

Programme 2

```
10 REM GET numérique
20 PRINT"Entrer un chiffre(1 ou 2)"
30 GET A$
40 IF A$< >"1" OR A$< >"2" THEN 30 ELSE
   A = VAL(A$)
50 PRINT A;"+5=";A+5
60 END
```

Commentaire : le caractère introduit à la ligne 30 ne s'affiche pas. VAL a permis de numériser le caractère qui était dans A$.

Programme 3

```
10 REM Autre GET numérique
20 CLS:PRINT"Entrer un nombre de 3 chiffres"
30 GET A$:PRINT A$;
40 GET B$:PRINT B$;
50 GET C$:PRINT C$
60 A=VAL(A$):B=VAL(B$):C=VAL(C$)
70 N=A*100+B*10+C
80 PRINT"Le nombre introduit était";N
90 END
```

Commentaire : aux lignes 30 à 50 les PRINT assurent l'affichage (puisque GET ne le fait pas), les ; permettent l'affichage des 3 chiffres sur la même ligne et accolés. La ligne 60 numérise les entrées. La ligne 70 utilise le fait que A est le chiffre des centaines, B celui des dizaines et A celui des unités.

Programme 4

```
10 REM lecture et dépassement
20 DATA 6,7
30 DATA 8,9,10
40 READ X
50 PRINT X
60 GO TO 40
```

Programme 5

```
10 REM lecture sans dépassement
20 DATA 6,7
30 DATA 8,9,10
40 FOR I=1 TO 5
50 READ X(I)
60 PRINT X(I)
70 NEXT I
80 END
```

Commentaire : on connait le nombre de données

Programme 6

```
10 REM autre lecture sans dépas-
      sement
20 DATA 6,7
30 DATA 8,9,10
40 REPEAT
50 I=I+1:READ X(I)
60 PRINT X(I)
70 UNTIL X(I)=10
80 END
```

Commentaire : on connait la dernière donnée

Programme 7

```
10 REM lecture par 2 données
20 DATA 11,12,13,14,15
30 DATA 16,17,18,19,20
40 FOR I=1 TO 5
50 READ X(I),Y(I)
60 PRINT X(I),Y(I)
70 NEXT I
80 END
```

Programme 8

```
10 REM utilisation d'une donnée sur deux
20 DATA 11,12,13,14,15
30 DATA 16,17,18,19,20
40 FOR I=1 TO 5
50 READ X(I),Y(I)
60 PRINT X(I)
70 NEXT I
80 END
```

Commentaire : toutes les données sont lues, mais une donnée sur deux est réellement utilisée (ici affichage).

Programme 9 : Transformation d'une date numérique (15-4-1983) en une chaine (15 AVRIL 1983)

```
10 DATA JANVIER,FEVRIER,MARS,AVRIL,MAI,JUIN
20 DATA JUILLET,AOUT,SEPTEMBRE,OCTOBRE,NOVEMBRE,DECEMBRE
30 DIM M$(12)
40 REM M$(I) contiendra le nom du I ième mois
50:
60 FOR I=1 TO 12:READ M$(I):NEXT
70:
75 CLS
80 INPUT"JOUR(1 à 31)";J
90 INPUT"MOIS(1 à 12)";M
100 INPUT"ANNEE";A
110:
120 DATE$=STR$(J)+"  "+M$(M)+"  "+STR$(A)
130 PRINT:PRINT:PRINT DATE$
140 END
```

Commentaire : à la ligne 60 lecture des 12 mois (chaines de caractères donc M$(I)). Lignes 80 à 100 introduction de la date numérique. Ligne 120 date sous forme de chaine DATE$. Ligne 130 affichage final. Les lignes 120 et 130 auraient pu être remplacées par 120 PRINT:PRINT:PRINT J SPC(2)M$(M) SPC(2)A.

7. LES 3 CANAUX

Grâce à ses 3 canaux de sortie, il est donc possible (en utilisant PLAY) d'écouter des notes sur le canal 1 pendant un certain temps, puis d'utiliser les canaux 1 et 2 simultanément afin de réaliser un "accord" ou un effet sonore.

8. L' INSTRUCTION SOUND C,P,V (SOUND = SON)

Pour utiliser l'instruction SOUND C,P,V il faut donner des valeurs aux 3 "paramètres" C, P et V.

Paramètre C : on lui donne soit les valeurs 1, 2 ou 3 (suivant le CANAL choisi) pour émettre un son "PUR", soit 4, 5 ou 6 pour émettre des sons "BRUITES" émis sur le canal 1 (si on choisit C=4), sur le canal 2 (si on choisit C=5) et sur le canal 3 (si on choisit C=6).

Donc C = (canal)	1, 2 ou 3	son pur sur canal 1, 2 ou 3	
	4	son bruité sur canal	1
	5		2
	6		3

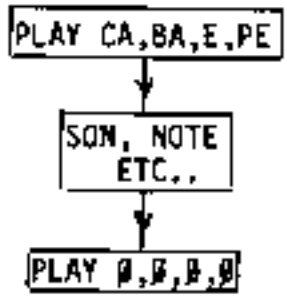
Paramètre P : c'est le paramètre Période. Plus la période est grande, plus le son sera grave. Pour P=55Ø on obtiendra ainsi un son plus grave que pour P=Ø.

Paramètre V : ce paramètre permet de régler le Volume sonore (de 1 à 15). Si on choisit V=Ø, le volume sonore sera modulé par l'instruction PLAY.

Les effets sonores seront obtenus d'une part, en choisissant P et C et d'autre part, par la répétition rapide ou non de certains sons.

9. L'INSTRUCTION PLAY CA,BA,E,PE (PLAY = JOUER)

On a le schéma :



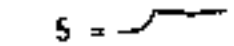
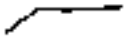
L'instruction PLAY CA, BA,E,PE utilise 4 paramètres : CA,BA, E et PE.

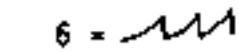
Paramètre CA : c'est simplement le ou les canaux autorisés en son pur. La codification est :

Valeur de CA ⟶	Canaux Autorisés	Valeur de CA ⟶	Canaux Autorisés
Ø	aucun	4	3
1	1	5	3 et 1
2	2	6	3 et 2
3	1 et 2	7	tous

Paramètre BA : c'est l'analogue de CA pour le bruitage ; donc réservé à SOUND lorsque le paramètre C vaut 4, 5 ou 6. C'est la même codification que pour CA c'est-à-dire, si on pose BA=4 seuls les sons bruités du canal 3 seront autorisés à être émis (ce qui correspond à C=6 pour SOUND).

Paramètre E : c'est "l'enveloppe", c'est-à-dire la forme, dont la codification est :

1 =
3 = 
5 = 
7 = 

2 =
4 = 
6 = 

Paramètre PE : c'est la Période de l'Enveloppe. C'est en quelque sorte "la longueur" de l'enveloppe, avant sa répétition. PE peut varier de 0 à 32767.

10. L'INSTRUCTION MUSIC C,O,N,V

Les quatres paramètres ne sont pas tous inconnus.

Paramètre C : C indique le Canal de sortie de la note (1, 2 ou 3).

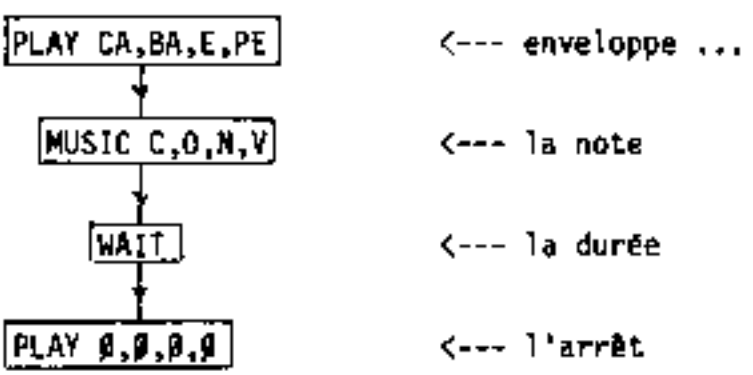
Paramètre O : il indique en quel octave la note sera jouée. Il y a 7 octaves à votre disposition numérotés de 0 à 6 (du plus grave au plus aigu...).

Paramètre N : N indique la note jouée par MUSIC avec la codification suivante :

Valeur de N ⟶	Note	Valeur de N ⟶	Note
1	DO	7	FA #
2	DO #	8	SOL
3	RE	9	SOL #
4	RE #	10	LA
5	MI	11	LA #
6	FA	12	SI

Paramètre V : pour le Volume, même codification que pour SOUND.

La DUREE de la note se fait en utilisant l'instruction WAIT; d'où le schéma :



11. RESUME LECON6 S2

```
SOUND C,P,V      C = Canal  1,2,3 (pur)
                            4,5,6 (bruit)      P = Période (en augmentant, son plus grave)

                 V = Volume (de 1 à 15)  (contrôlé par PLAY si V=0)

PLAY CA,BA,E,PE  CA = canaux autorisés      BA = canaux de bruitage autorisés
                 E  = enveloppe             PE = période enveloppe (0 à 32767)

MUSIC C,O,N,V    C = Canal                  O = Octave (0 à 6)
                 N = Note                   V = Volume
```

12. EXERCICES LECON6 S2

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1 : un affichage un peu spécial ...

```
10 CLS
20 INPUT"Introduire un mot";A$
30 PRINT:PRINT
40 FOR I=1 TO LEN(A$)
50:    PRINT MID$(A$,I,1);
60:    WAIT 30
70:    PLAY 1,1,1,20
80:    SOUND 1,3000,0
90:    PLAY 0,0,0,0
100 NEXT I
110 END
```

Programme 2 : une mitrailleuse

```
10 PLAY 0,1,4,18000
20 FOR I=1 TO 30
30:    SOUND 1,1,3
40:    SOUND 4,1,10
50:    PLAY 1,1,0,0
60:    FOR J=1 TO 40:NEXT
70:    PLAY 0,0,0,0
80 NEXT I
```

Commentaire : à la ligne 30 un son pur est envoyé dans le canal 1. A la ligne 40 un son bruité est envoyé dans le canal 1. PLAY est positionnée ici après les SOUND.

Programme 3 : un bombardement un jour de pluie ...

```
10 PLAY 2,1,4,20000
20 FOR I=1 TO 100
30:    SOUND 2,I,4
40:    FOR J=1 TO 4:NEXT
50     PLAY 0,0,0,0
60 NEXT I
70 FOR K=1 TO 6
80     EXPLODE
90 NEXT K
100 END
```

Commentaire : essayez la commande SOUND 2,1,4 seule. Vous l'entendrez à peine, alors que la répétition très rapide donne un effet surprenant !

Programme 4 : ai-je du bon tabac ?

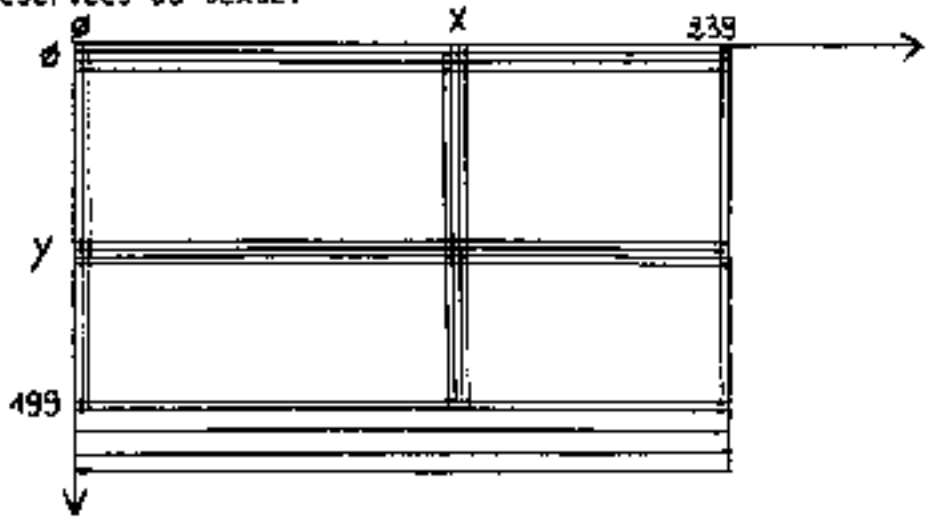
```
10 DATA 1,20,3,20,5,20,1,20,3,40,3,20,
   5,20,6,40,6,40,5,40
20 FOR I=1 TO 10
30:    READ N,D
40:    PLAY 2,0,7,2000
50:    MUSIC 2,3,N,3
60:    WAIT D
70:    PLAY 0,0,0,0
80 NEXT I
90 END
```

Commentaire : les DATA contiennent note, durée, note, durée... Le couple 1,20 signifie DO, prévu joué pendant 2 dixièmes de seconde. A chaque boucle, une note et sa durée sont lues puis jouées.

LECON 7

1. L'INSTRUCTION HIRES (HIRES ---> HIGH RESOLUTION = HAUTE RESOLUTION)

L'ordre HIRES présente l'écran comme une grille de points avec 3 lignes réservées au texte.



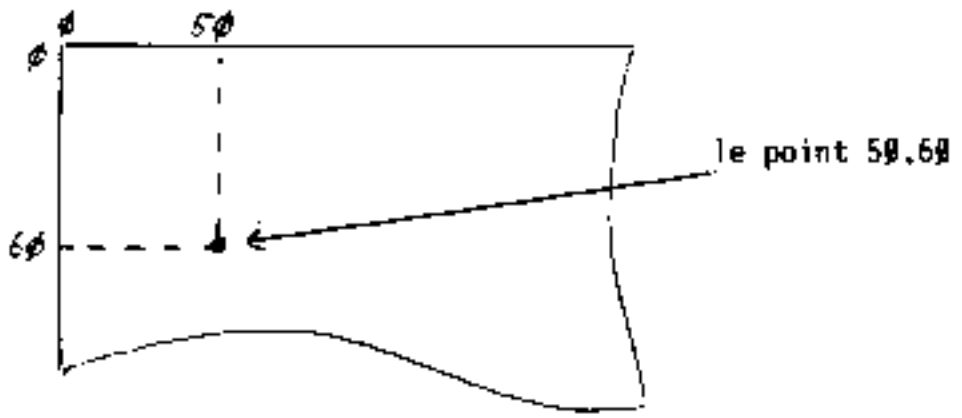
Le rectangle de points a 240 points horizontaux (numérotés de 0 à 239) et 200 points verticaux (numérotés de 0 à 199); donc le rectangle contient 44000 points. C'est sur cette "feuille" que nous dessinerons.

Chaque point de cette feuille a des coordonnées X,Y. X est sa position horizontale: $0 \leq X \leq 239$ et Y est sa position verticale: $0 \leq Y \leq 199$.
Faire attention : le point 0,0 est en haut à gauche !

Les trois lignes de texte peuvent servir pour des questions, des commentaires.

Lorsque HIRES est exécuté, la couleur du fond devient NOIRE et celle des dessins sera BLANCHE. Naturellement, vous pouvez changer ces couleurs à l'aide de PAPER et INK.

2. L'INSTRUCTION CURSET X,Y,CT (CURSET ---> SET CURSOR = PLACER le CURSEUR)



On peut donc tracer une courbe point par point, en utilisant l'instruction CURSET X,Y,1. Naturellement X et Y devront toujours vérifier $0 \leq X \leq 239$ et $0 \leq Y \leq 199$, sinon un message d'erreur apparaîtra.

Pour obtenir certains effets ou pour des jeux,on peut utiliser les autres valeurs de CT = Couleur du Tracé.

CT=0 la couleur du tracé sera la couleur du fond (invisible dans un premier temps mais visible si changement de couleur)

CT=1 couleur choisie pour INK

CT=2 couleurs inversées

CT=3 le point est "présent" sans être tracé !

3. L'INSTRUCTION CIRCLE R,CL (CIRCLE = CERCLE)

Donc, pour tracer un cercle de centre 110,100 de rayon 50 on écrit :

```
CURSET 110,100,3  <--- le 3 pour que le point 110,100 ne soit pas visible
CIRCLE 50,1       <--- le 1 pour que le tracé se fasse avec la couleur de INK.
```

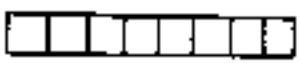
Le paramètre CL est le même que dans CURSET ; c'est la Couleur du Tracé.

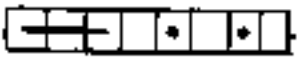
Faire attention au choix du rayon : l'unité est le point.
Un cercle de rayon 1 ou 2 sera quasiment réduit à un point ! Un rayon trop grand risque de faire "sortir" le cercle du cadre de la feuille graphique, ce qui entraînera un message d'erreur;

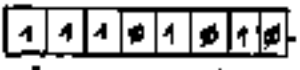
La feuille n'étant pas carrée, le cercle tracé par l'instruction CIRCLE est en fait une ellipse, comme vous le constaterez en exercices.

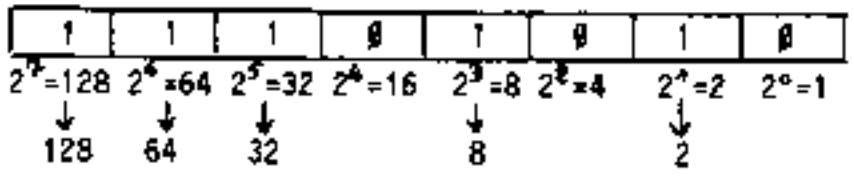
4. L'INSTRUCTION PATTERN n (PATTERN = MODELE)

Pour définir un modèle on trace 8 carrés ;


puis on dessine dessus le modèle souhaité. Par exemple, pour un modèle — . . on obtient :


On associe alors aux cases vides un 0 et autres cases un 1, ce qui donne :


Cela représente un nombre écrit en binaire (base deux). On écrit alors ce nombre en base 10.
Rappelons le processus : on écrit en dessous de chaque case en allant de DROITE à GAUCHE les puissances de 2 en commençant par 2^0.



on multiplie les 1 par les nombres correspondants.

On additionne ces nombres 128+64+32+8+2=234

Il suffit alors d'écrire PATTERN 234 pour que tout cercle tracé se fasse suivant le modèle — . . (qui par répétition donne — . . — . . — . .).

5. RESUME LECON7 S1

HIRES	permet le passage en mode haute résolution 240x200
TEXT	permet le retour à l'écran "texte"
CURSET X,Y,CT	place un point en X,Y dans la Couleur de Tracé CT
CIRCLE R,CT	trace un cercle de rayon R, dans la Couleur de Trace CT avec pour centre la dernière position donnée par CURSET
PATTERN n	permet de définir un MODELE de tracé.

6. EXERCICES LECON7 S1

Pour examiner le listing alors qu'on est en HIRES, il faut taper TEXT et ensuite LIST (sinon le listing défilera dans la "fenêtre" de 3 lignes au bas de l'écran).

Le tracé d'une courbe se faisant point par point, l'ordre CURSET sera souvent à l'intérieur d'une boucle FOR..TO./STEP./NEXT avec un PAS (STEP) adapté à la finesse du tracé souhaité.

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
10 REM segment horizontal
20 HIRES
30 FOR X=10 TO 90
40:   CURSET X,100,1
50 NEXT X
60 PRINT"C'est un segment"
70 GET A$
80 END
```

Commentaire : refaites l'exercice sans la ligne 70. Utilisez une représentation de l'écran sur papier pour faciliter la compréhension.

Programme 2

```
10 HIRES
20 FOR X=90 TO 130
30:   CURSET X,115,1
40:   CURSET X,85,1
50 NEXT X
60 FOR Y=85 TO 115
70:   CURSET 90,Y,1
80:   CURSET 130,Y,1
90 NEXT Y
100 CURSET 110,100,3
110 CIRCLE 50,1
120 END
```

Commentaire : nous verrons dans la section 2 une manière plus rapide de tracer des segments de droite.

Programme 3 : reprendre le programme 2 en rajoutant

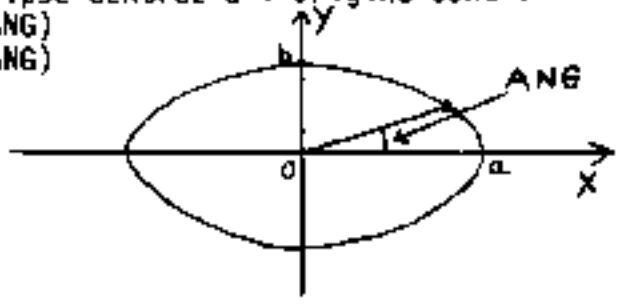
```
115 PATTERN 224
117 CIRCLE 90,1
```

Commentaire :
en ligne 115 modèle(—)(—)(—)

Programme 4

```
10 REM ellipse
15 HIRES
20 FOR ANG=0 TO 2*PI STEP 0.05
30:   X=100*COS(ANG):Y=40*SIN(ANG)
40:   CURSET 120+X,100+Y,1
50 NEXT ANG
```

Commentaire : les équations paramètriques d'une ellipse centrée à l'origine sont :
X=a cos(ANG)
Y=b sin(ANG)



Suite commentaire Programme 4 : on a choisi, à la ligne 30, a et b assez grand pour que l'ellipse soit "visible", mais sans être trop grand afin de ne pas "sortir" de l'écran. A la ligne 40, on a centré l'ellipse au centre de l'écran 120,100. L'orientation de l'axe Y vers le bas (sur l'ordinateur) fait que la partie "basse" de l'ellipse sera tracée avant la partie haute.

Programme 5

```
10 REM spirale
15 HIRES
20 R=15
30 REPEAT
40:   CURSET 110+X,100+Y,1
50:   ANG=ANG+.1:R=R+.2
60:   X=R*COS(ANG):Y=R*SIN(ANG)
65:   PRINT 100+Y
70 UNTIL 110+X<0 OR 110+X>239 OR 100+Y<0 OR 100+Y>199
80 END
```

Commentaire : à la ligne 60 nous avons les équations d'un cercle (a*b dans l'ellipse). A la ligne 50, on augmente l'angle et le rayon (d'où la spirale). La répétition se fait jusqu'à ce que le point à tracer soit en dehors de l'écran.

Programme 6

```
10 REM fonction sinus
20 HIRES
30 FOR X=0 TO 2*PI STEP 0.05
40 X1=15*X:Y1=-5 0*SIN(X)
50 CURSET 20+X1,100+Y1,1
60 NEXT X
70 END
```

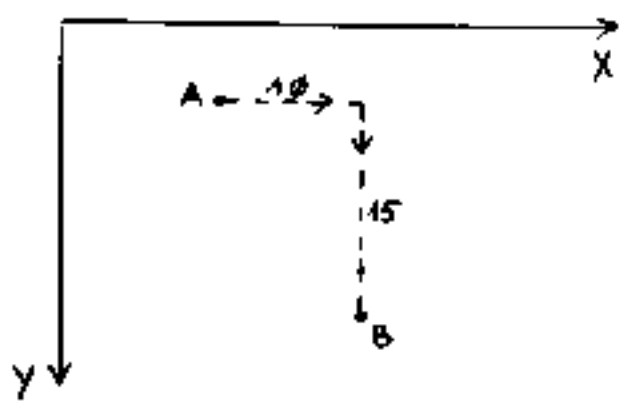
Commentaire : la ligne 40 permet un grossissement (échelle) pour rendre visible la fonction. Le signe — pour Y1 tient compte du fait que l'axe des Y est dirigé vers le bas.

Programme 7

```
10 REM fonction 100e^-x/5 sin(3x)
20 HIRES
30 DEF FN A(X)=100*EXP(-X/5)*SIN(3*X)
40 FOR X=0 TO 2*PI STEP.05
50 X1=35*X
60 CURSET X1,100-FNA(X),1
70 NEXT
80 END
```

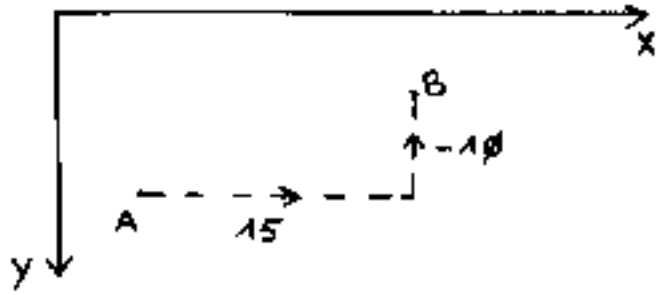
Commentaire : utilisation de DEF FN.

7. LES COORDONNEES RELATIVES



Les coordonnées de B relativement à A sont 10,15.

Faire attention au sens de l'axe Y.



Les coordonnées de B relativement à A sont 15,-10 (-10 car le déplacement est opposé au sens des Y).

Il est bon de savoir que si A a pour coordonnées XA,YA et B pour coordonnées XB,YB alors les coordonnées de B relativement à A sont XB-XA, YB-YA.

8. L'INSTRUCTION CURMOV X,Y,CT (CURMOV--->MOVE CURSOR=DEPLACER le CURSEUR)

L'utilisation de CURMOV est très agréable en mode direct. En mode programme il faudra prendre garde de "sortir" de l'écran. Cette instruction peut aussi être utilisée pour des jeux, en réponse à des pressions de touches fléchées.

9. L'INSTRUCTION DRAW X,Y,CT (DRAW=TRACER)

Cette instruction évitera les boucles FOR.TO.STEP./NEXT contenant la commande CURSET. De plus, l'exécution de DRAW est nettement plus rapide que l'utilisation d'une boucle FOR/NEXT.

Attention : ne pas "sortir" de la page graphique.

10. L'INSTRUCTION CHAR X,0,CT (CHAR--->CHARACTER=CARACTERE)

Si à première vue cette instruction est d'une manipulation lourde (code ASCII, déplacement du curseur pour chaque lettre), il suffira d'utiliser un sous-programme qui sera appelé chaque fois qu'une écriture en haute résolution sera nécessaire.

Le paramètre CT est le même paramètre que pour CURSET... .

Le 0 venant après le paramètre X peut être remplacé par 1. Dans ce cas, ce sont les caractères graphiques incorporés à ORIC qui seront affichés.

Rappelons que les caractères du clavier que l'on peut visualiser sont ceux dont le code ASCII est compris entre 32 et 127.

11. RESUME LECON7 S2

Les coordonnées de B RELATIVEMENT à A sont XB-XA,YB-YA.	
CURMOV X,Y,CT	place le point dont les coordonnées relatives au dernier point placé sont X,Y.
DRAW X,Y,CT	trace le segment de droite joignant le dernier point placé, au point de coordonnées relatives X,Y.
CHAR X,0,CT	affiche en mode HIRES, le caractère dont le code ASCII est X ($32 \leq X \leq 127$).

12. EXERCICES LECON7 S2

Rappelons que si vous possèdez un écran couleur, vous pouvez après chaque commande HIRES des programmes ci-dessous choisir la couleur de fond et la couleur des dessins.

Examiner, prévoir, vérifier, réexaminer si besoin était.

Programme 1

```
10 REM Tracé d'axes
20 CURSET 0,100,3
30 DRAW 239,0,3
40 CURSET 0,199,3
50 DRAW 0,-199,1
60 END
```

Commentaire : il faut placer le curseur au bon endroit avant d'utiliser DRAW. Reprendre ce programme en rajoutant 15 PATTERN 170.

Programme 2

```
10 écriture en haute résolution
20 HIRES
30 INPUT"Entrer un court texte";T$
40 PRINT
50 INPUT"Coordonnées du départ?";X0,Y0
60 CURSET X0,Y0,3
70 FOR I=1 TO LEN(T$)
80 CHAR ASC(MID$(T$,I,1)),0,1
90 CURMOV 8,0,0
100 NEXT I
110 END
```

Commentaire : la ligne 60 permet de placer le curseur sans écrire. La ligne 80 prend le Ième caractère de T$, calcule son code ASCII (avec ASC) et grâce à CHAR affiche le Ième caractère. La ligne 190 permet un déplacement horizontal du curseur pour se préparer à l'affichage du prochain caractère. Les lignes 70 à 100 forment une boucle FOR/NEXT qui extrait les caractères de T$ un par un et ensuite les affiche.
On pourrait améliorer ce programme en vérifiant que le texte ne "sorte" pas de l'écran !

Programme 3

```
10 REM tracé d'axes amélioré
20 HIRES
30 CURSET 0,100,3
40 DRAW 230,0,3
50 CURMOV 0,-3,3
60 CHAR 62,0,1
70 CURMOV -5,10,3
80 CHAR 120,0,1
90 CURSET 110,199,3
100 DRAW 0,-190,1
110 CURMOV -2,-6,3
120 CHAR 94,0,1
130 CURMOV 10,5,3
140 CHAR 121,0,1
150 END
```

Commentaire : les lignes 50 et 60 permettent le fléchage de l'axe horizontal. Les lignes 70 et 80 affichent x sous ce fléchage. La même démarche a été faite pour l'axe vertical.
Suggestion : faire une graduation des axes.

Programme 4 : tracé d'axes et sinusoïde.
Reprendre le programme 3 en ajoutant les instructions suivantes :

```
150 REPEAT
160 X1=15*X:Y1=-50*SIN(X)
170 CURSET 120,100,3
180 CURMOV X1,Y1,1
190 X=X+0.05
200 UNTIL X>2*PI
210 END
```

Commentaire : il est bon de comparer au programme 6 de la section 1. Les répétitions par REPEAT/UNTIL au lieu de FOR/NEXT importent peu.
Par contre, ici on évite le problème des changements d'origine puisqu'à chaque répétition on fixe l'origine en 120,100 et on travaille relativement à cette origine grâce à CURMOV.

Programme 5

```
10 REM DRAW et ses effets
20 HIRES
30 REPEAT
40 X=100*COS(A):Y=40*SIN(A)
50 X1=120+X:Y1=100+Y
60 CURSET 160,100,3
70 DRAW X1-160,Y1-100,1
80 A=A+0.05
90 UNTIL A>2*PI
100 END
```

Commentaire : à la ligne 40 on trouve les équations d'une ellipse lorsque son centre est 0,0. A la ligne 50 le centre de cette ellipse est ramené au centre de l'écran 120,100. Les lignes 60 et 70 tracent un segment joignant le point 160,100 au bord de cette ellipse. Le tout est répété jusqu'à ce que l'ellipse soit couverte.

Programme 6

```
10 REM Le rebond d'une balle
20 HIRES
30 X=120:Y=110
40 DX=2+4*RND(1):DY=2+5*RND(1)
50 DRAW 0,199,1:DRAW 239,0,1:DRAW 0,-199,1:DRAW -239,0,1
60 CURSET X,Y,3:CIRCLE 5,1
70 IF X+DX-5<0 OR X+DX+5>239 THEN DX=-DX:PING
80 IF Y+DY-5<0 OR Y+DY+5>199 THEN DY=-DY:PING
90 CIRCLE 5,0:REM cercle effacé
100 X=X+DX:Y=Y+DY:GOTO 60.
```

Commentaire : Le mouvement d'une balle est assuré en dessinant un cercle et en effaçant l'ancien cercle.

A la ligne 50 un cadre pour l'écran est tracé. A la ligne 60, est tracé un cercle de rayon 5 dont les coordonnées du centre ont été fixées à la ligne 30. Le nouveau cercle à tracer, a son centre qui a pour coordonnées relatives au centre précédent DX,DY. Avant de tracer ce nouveau cercle, il faut s'assurer qu'il ne sorte pas de l'écran. Aux lignes 70 et 80, si le cercle à venir risque de sortir (de l'écran), on change l'accroissement, créateur de ce dépassement, en son opposé (et on rajoute PING). Le problème de la sortie étant réglé, il faut effacer l'ancien cercle avant le tracé du nouveau. L'effacement est effectué à la ligne 90 (CT=0=couleur du fond!). A la ligne 100 on prépare les nouvelles coordonnées du centre et on effectue le tracé à la ligne 60.

CONCLUSION

Ce cours a permis de connaître les diverses possibilités du langage B.A.S.I.C..La haute résolution et le son mis à part, c'est un BASIC que l'on trouve sur la plupart des micro-ordinateurs. Dans ce cours d'initiation, la gestion de fichiers n'a pas été abordée.

Afin de mener à bien la programmation d'un problème sur micro-ordinateur, il est recommandé de suivre la démarche suivante :

- Poser clairement le problème à résoudre.
- Analyser le problème afin d'en déduire un organigramme clair qui vous guidera par la suite.
- Programmer si possible par "module" afin de faciliter la lecture et la maintenance du programme. Lire à ce sujet le chapitre 12 du manuel ORIC.
- Ne pas oublier lors des essais que le BASIC d'ORIC est interprété et qu'ainsi on peut faire afficher en mode direct le contenu d'une variable afin de localiser l'origine d'une erreur.

Avec de l'expérience vous pourrez bien maîtriser le langage BASIC et ainsi développer des programmes assez complexes.

Pour ceux qui souhaiteront aller plus loin, la pratique de BASIC permettra une meilleure compréhension de tout autre langage de programmation.