# BASIC PLUS

## 80 ROUTINES SUR AMSTRAD

MICHEL MARTIN

# BASIC PLUS

## 80 ROUTINES SUR AMSTRAD

**Autres ouvrages relatifs à l'Amstrad**

— 102 programmes pour Amstrad — Jacques Deconchat
— Amstrad en famille — Jean-François Sehan
— Super jeux Amstrad — Jean-François Sehan
— La découverte de l'Amstrad CPC 464 et 664 — Daniel-Jean David
— Exercices en Basic pour Amstrad — Maurice Charbit

**A paraître :**

— Périphériques et gestion de fichiers sur Amstrad — Daniel-Jean David
— Simulation et intelligence artificielle — René Descamps

**Ouvrage de référence**

— Les mots de la micro — Alan Freedman, adapté par Bernard Sauteur

Pour tout problème rencontré dans les ouvrages P.S.I.
vous pouvez nous contacter au numéro ci-dessous :

Numéro Vert/Appel Gratuit en France

16 (05) 21 22 01

(Composer tous les chiffres, même en région parisienne)


ISBN : 2-86595-286-X

MICHEL MARTIN

# BASIC PLUS

## 80 ROUTINES SUR AMSTRAD

ÉDITIONS DU P.S.I.
1985

# Sommaire

# Avertissement

Ce livre s'adresse aux possesseurs de CPC 464 ou CPC 664 qui ont déjà pratiqué le BASIC et qui veulent aller plus loin.

L'ouvrage propose un ensemble de 80 programmes utilitaires qui vous permettront de manipuler simplement le générateur sonore, le graphisme haute résolution, ou de rajouter des fonctions au BASIC Amstrad.

L'ouvrage se compose de cinq parties :

- les fonctions évoluées de l'Amstrad ;
- améliorations du BASIC Amstrad ;
- le générateur sonore ;
- les modes graphiques haute résolution ;
- utilitaires d'ordre général.

□ **"Les fonctions évoluées de l'Amstrad"** donne un ensemble d'outils pour mieux utiliser l'Amstrad.

□ **"Améliorations du Basic Amstrad"** permet d'ajouter des instructions au BASIC de cet ordinateur, comme MOD, DIV, DUMP, etc.

□ **"Le générateur sonore"** détaille la structure du synthétiseur sonore, montre comment créer des bruits d'animation et facilite l'écriture de morceaux de musique sur une, deux ou trois voix.

□ **"Les modes graphiques haute résolution"** détaille la structure de l'écran graphique, montre comment écrire des jeux d'animation, comment redéfinir les caractères standard et donne divers utilitaires graphiques.

- **"Utilitaires d'ordre général"** propose un ensemble de programmes facilitant les entrées/sorties écran, un mini-éditeur de textes 80 colonnes et divers autres utilitaires.

- Un **avant-propos** donne une méthode pour faciliter l'écriture et la lisibilité des programmes BASIC.

# BASIC : partir sur de bonnes bases

## LES ORIGINES DU BASIC

Le langage BASIC a été créé en 1964 aux États-Unis. La signification du sigle BASIC (Beginners All Purpose Symbolic Instruction Code, ou encore Codage d'instructions à usage général pour débutants) nous montre que ses auteurs le désignaient avant tout comme un langage simple, utilisable par les débutants.

Aujourd'hui, le BASIC a bien évolué et s'impose aussi bien sur la plupart des micro-ordinateurs familiaux que dans certaines applications professionnelles. Tout en gardant sa simplicité d'emploi, il a su devenir puissant et souple.

Le langage BASIC se compose d'un certain nombre de mots clés ou instructions (154 pour le CPC 464). Deux machines différentes utilisant le langage BASIC n'auront pas exactement le même vocabulaire. En effet, certaines instructions varient d'un ordinateur à l'autre, en raison des possibilités particulières de chaque appareil. Ainsi, un ordinateur qui ne possède pas de générateur sonore n'a rien à faire de commandes de génération de son. Ou encore, un ordinateur ne travaillant qu'avec deux couleurs n'a rien à faire d'instructions de manipulation de couleurs.

Malgré cela, la plupart des mots clés du BASIC sont indépendants de l'ordinateur utilisé, de sa marque et de son pays de fabrication. On peut donc parler de l'universalité du BASIC, en sachant très bien que ce terme n'est pas entièrement vérifié...

Le BASIC ainsi défini, il peut sembler facile d'écrire un programme en mettant bout à bout un ensemble de mots clés.

Les paragraphes qui suivent montrent qu'il n'en est rien, et que l'écriture d'un programme BASIC demande méthode et réflexion.

## MIEUX PROGRAMMER EN DÉFINISSANT SES OBJECTIFS

Considérons un programmeur débutant qui a l'intention d'écrire un programme de sa conception. Très souvent, il s'y prendra de la façon suivante : il s'assiéra devant la machine et commencera à taper ce qu'il croit être un programme, sans avoir pris le temps d'y réfléchir.

Deux cas peuvent se présenter :

- le programme à écrire est court et/ou simple ;
- le programme à écrire est long et/ou complexe.

Dans le premier cas, et avec un peu d'expérience, le programmeur pourra arriver à ses fins, après quelques tentatives infructueuses.
Dans le second cas, le programmeur peinera pour écrire un programme sans structure et très difficilement compréhensible pour lui-même ou pour d'autres programmeurs. Il pourra même ne plus comprendre ce qu'il a écrit, et arrêter son travail, découragé.

Heureusement, il existe une méthode pour permettre aux programmeurs, débutants ou expérimentés, d'écrire des programmes clairs et facilement compréhensibles, même si la tâche qu'ils réalisent est longue et complexe.

La clé de cette méthode réside dans l'"analyse du problème" que l'on se propose de résoudre par l'écriture d'un programme. Cette analyse débouche sur une définition des objectifs.

La plupart des programmes peuvent être mis sous la forme suivante :

ENTREE → TRAITEMENT → SORTIE

La sortie du programme est l'objectif à atteindre, le résultat à calculer ou à afficher.

L'entrée du programme représente les données nécessaires à la réalisation des sorties.

Le traitement représente l'ensemble des opérations à faire subir aux entrées pour obtenir les sorties désirées.

La définition des objectifs consistera à classer et décrire l'ensemble des entrées/sorties et le traitement. Plus cette description sera détaillée, plus il sera ensuite facile d'écrire les instructions correspondantes en langage BASIC.

Par exemple, pour un programme de jeu :

- les entrées peuvent être les différentes commandes de déplacement et de tir entrées au clavier ;
- les sorties peuvent être le déplacement de divers objets sur l'écran ;
- le traitement peut être la définition graphique des objets ; l'interprétation des commandes entrées par le joueur, le déplacement effectif des objets.

Arrivé à ce point, deux choses très importantes restent à faire : organiser ses objectifs en définissant une structure et hiérarchiser les fonctions du programme.

## MIEUX PROGRAMMER EN ORGANISANT SES OBJECTIFS : CONCEPT DE STRUCTURE

Les entrées/sorties et le traitement étant définis, il est nécessaire de donner une structure au traitement. Autrement dit, il faut organiser la tâche plus ou moins complexe que constitue le traitement en un ensemble de sous-tâches plus simples, chaque sous-tâche étant définie par un traitement spécifique.

Prenons l'exemple suivant : le programme à réaliser a pour objectif le tracé d'une courbe représentée par une équation quelconque de la forme $Y = f(x)$.

L'analyse des objectifs nous montre que les entrées sont :

- domaine d'étude ;
- équation de la courbe.

La sortie est : affichage du min et du max sur la courbe.

Le traitement est : résolution et affichage de l'équation entre les bornes du domaine d'étude choisi.

Si nous poursuivons l'analyse, le traitement peut se décomposer en :

- Recherche de l'échelle en Y (pour faire apparaître la courbe sur tout l'écran),
- Recherche du pas du tracé (distance entre deux points de l'axe Ox),
- Tracé d'un point aux coordonnées X, Y.

Le traitement se décompose ici en trois sous-tâches élémentaires.

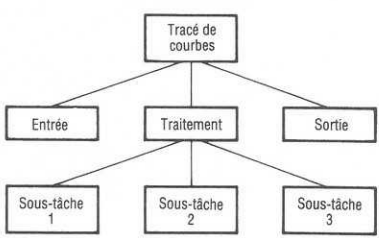
L'écriture d'un module ou procédure pour chaque sous-tâche constitue la structure du traitement.

Arrivé à ce niveau, il ne reste plus qu'une étape avant le codage puis l'écriture du programme sur la machine.

## MIEUX PROGRAMMER EN DÉFINISSANT DES NIVEAUX : CONCEPT DE HIÉRARCHIE

Nous avons vu qu'il était nécessaire de diviser la tâche traitement en sous-tâches plus simples.

L'exemple précédent peut se schématiser comme suit :




Cette représentation ou "graphe hiérarchique" suffit à elle seule pour définir la structure du programme.

Pour cet exemple précis, la programmation BASIC sera la suivante :

```
10 REM Trace de courbes
20 :
30   GOSUB 1000 'Entree
40   GOSUB 2000 'Traitement
50   GOSUB 3000 'Sortie
60 :
70 END
80 REM ------------------------------------
1000 REM Entree
1010 :
1020 REM Saisie de l'equation et du
1030 REM domaine de definition
1040 :
1050 RETURN
1060 REM ------------------------------------
2000 REM Traitement
2010 :
2020 GOSUB 10000 'Echelle Oy
2030 GOSUB 11000 'Pas Ox
2040 FOR I=1 TO NBPTS STEP PAS
2050   GOSUB 12000 'Trace d'un point
2060 NEXT I
2070 :
2080 RETURN
2090 REM ------------------------------------
3000 REM Sortie
3010 :
3020 REM Affichage des Min et Max
3030 :
3040 RETURN
3050 REM ------------------------------------
10000 REM Echelle Oy
10010 :
10020   REM Recherche de l'echelle
10030 :
10040 RETURN
10050 REM ------------------------------------
11000 REM Recherche du pas Ox
11010 :
11020   REM Calcul du pas
11030 :
11040 RETURN
11050 REM ------------------------------------
12000 REM Affichage d'un point
12010 :
12020   REM Point aux coordonnees X,Y
12030 :
12040 RETURN
```

Cet exemple simple fait appel à une structure à trois niveaux hiérarchiques. Des programmes plus complexes peuvent demander quatre et même cinq niveaux hiérarchiques. Les sous-tâches se décomposent alors en blocs plus élémentaires qui se décomposent eux-mêmes en d'autres blocs plus élémentaires.

## VERS UNE PROGRAMMATION EFFICACE

La dernière étape constitue l'écriture du programme. Les objectifs étant définis, le traitement structuré et hiérarchisé, vous pouvez probablement gagner du temps en tapant directement votre programme au clavier, plutôt que de l'écrire sur une feuille, puis le recopier.

Deux derniers conseils :

- Affectez à chaque niveau hiérarchique un groupe de n° de lignes. Ainsi, écrivez par exemple :
  le niveau 1 entre les lignes 10 et 100 ;
  le niveau 2 entre les lignes 100 et 1000 ;
  le niveau 3 entre les lignes 1000 et 10000 ;
  etc.
- Documentez votre programme en insérant un nombre non négligeable de remarques (REM) concernant les entrées, sorties et traitement de chaque module.

Ces tâches peuvent vous paraître fastidieuses mais, si vous les suivez méthodiquement, vous vous féliciterez de la facilité avec laquelle vous pourrez insérer une modification six mois après la création du programme...

# Chapitre 1 | Les fonctions évoluées de l'Amstrad

Ce chapitre donne la manière de se servir au mieux des fonctions BASIC de l'Amstrad, ou de ses routines internes.

Les fonctions simples, communes à la plupart des BASIC, ne sont pas reprises ici. L'étude porte sur cinq sujets :

- les fonctions logiques ;
- la fonction DEF FN ;
- le chaînage de deux programmes avec passage d'arguments ;
- les fonctions spécifiques Amstrad ;
- les routines Assembleur utiles.

## LES FONCTIONS LOGIQUES

### Fonctions de l'Amstrad

**AND**

| A | B | A AND B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 1 | 0 | 0 |

**Utilisation de AND**

IF A=4 AND C=5 THEN PRINT

Une ligne sera sautée si A=4 et C=5.

**OR**

| A | B | A OR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 0 | 1 |

**Utilisation de OR**

IF A=4 OR C=5 THEN 20

Le débranchement à la ligne 20 se fera si A=4 ou C=5.

**NOT**

| A | NOT A |
|---|---|
| 0 | 1 |
| 1 | 0 |

**Utilisation de NOT**

IF NOT A=B THEN PRINT

Si A=B n'est pas vrai, autrement dit si A<>B, alors une ligne est sautée.

**XOR**

| A | B | A XOR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Utilisation de XOR**

IF A XOR B THEN 10

Le débranchement à la ligne 10 se fera si A=0 et B=1 ou si A=1 et B=0.

**Remarque :** Les fonctions **AND, OR, NOT** et **XOR** s'étendent également aux nombres entiers compris entre −32768 et 32767. Le résultat est alors égal à la fonction appliquée à chaque bit des deux nombres impliqués.

D'autres fonctions logiques sont parfois utilisées sur des machines différentes. Voici la liste des plus courantes et les programmes qui permettent de les simuler en logique booléenne (sortie à 1 ou à 0).

### Fonction NOR (Non OU)

| A | B | A NOR B |
|---|---|---|
| 0 | 0 | 1 → |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 1 | 0 | 0 |

Si A et B sont faux, alors A NOR B est vrai.

*Utilisation*

Soit un jeu dans lequel un avion à hélice est animé. Si l'hélice ne tourne pas et s'il n'y a plus de carburant, alors l'avion s'écrase.

Si HELICE=1 indique que l'hélice tourne et CA=1 indique qu'il reste du carburant, alors IF HELICE=1 NOR CA=1 THEN GOSUB ECRASEMENT lancera la procédure d'écrasement.

### *Le programme*

A et B contiennent les variables à tester.
X contient A NOR B à la sortie du programme.

```
1 REM  ******************************
2 REM  *                            *
3 REM  * SIMULATION DE LA FONCTION  *
4 REM  * LOGIQUE 'NOR'              *
5 REM  *                            *
6 REM  ******************************
7 REM  *                            *
8 REM  * Entree : Var A et B        *
9 REM  * Sortie : Resultat dans X   *
10 REM *                            *
11 REM ******************************
12 :
20 CLS:INPUT"Premier parametre  ";A
30 INPUT"Deuxieme parametre ";B
40 GOSUB 100 : REM A NOR B
50 PRINT A" NOR "B" = "X
60 END
70 REM ------------------------------
100 REM Calcul de la fonction NOR
101 :
110 X=0
120 IF A=0 AND B=0 THEN X=1
130 :
140 RETURN
```

### *Analyse du programme*

Ligne 110 : Sortie par défaut.
Ligne 120 : Calcul de la fonction NOR.

## Fonction NAND (Non ET)

| A | B | A NAND B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |

Si A vrai et B vrai, alors A NAND B faux ; tous les autres cas donnent A NAND B vrai.

### *Utilisation*

Le même exemple que précédemment peut être repris :

IF HELICE=1 NAND CA=1 THEN GOSUB DEFAILLANCE

En effet, si l'hélice ne tourne plus (HELICE = 1 faux), ou s'il n'y a plus de carburant (CA=1 faux), alors la procédure DEFAILLANCE est lancée.

### *Le programme*

A et B contiennent les variables à tester.
X contient A NAND B à la sortie du programme.

```
1 REM  ******************************
2 REM  *                            *
3 REM  * SIMULATION DE LA FONCTION  *
4 REM  * LOGIQUE 'NAND'             *
5 REM  *                            *
6 REM  ******************************
7 REM  *                            *
8 REM  * Entree : Var A et B        *
9 REM  * Sortie : Resultat dans X   *
10 REM *                            *
11 REM ******************************
12 :
20 CLS:INPUT"Premier parametre  ";A
30 INPUT"Deuxieme parametre ";B
40 GOSUB 100 : REM A NAND B
50 PRINT A" NAND "B" = "X
60 END
70 REM ------------------------------
100 REM Calcul de la fonction NAND
101 :
110 X=1
120 IF A=1 AND B=1 THEN X=0
130 :
140 RETURN
```

### *Analyse du programme*

Ligne 110 : Sortie par défaut.
Ligne 120 : Calcul de la fonction NAND.

## Fonction IMP (Implique)

| A | B | A IMP B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 0 | 0 |

Seul le cas A vrai, B faux donne A IMP B faux.

### *Utilisation*

Soit une serrure. Si vous possédez la clé (CLE=1) et si la serrure est fermée (FE=0), alors vous pouvez ouvrir la porte.

IF NOT ((CLE=1) IMP (FE=1)) THEN GOSUB OUVRIR

### *Le programme*

A et B contiennent les variables à tester.
X contient A IMP B à la sortie du programme.

```
1 REM  ******************************
2 REM  *                            *
3 REM  * SIMULATION DE LA FONCTION  *
4 REM  * LOGIQUE 'IMPLIQUE          *
5 REM  *                            *
6 REM  ******************************
7 REM  *                            *
8 REM  * Entree : Var A et B        *
9 REM  * Sortie : Resultat dans X   *
10 REM *                            *
11 REM ******************************
12 :
20 CLS:INPUT"Premier parametre  ";A
30 INPUT"Deuxieme parametre ";B
40 GOSUB 100 : REM A IMP B
50 PRINT A" IMP "B" = "X
60 END
70 REM ------------------------------
100 REM Calcul de la fonction IMP
101 :
110 X=1
120 IF A=1 AND B=0 THEN X=0
130 :
140 RETURN
```

### *Analyse du programme*

Ligne 110 : Sortie par défaut.
Ligne 120 : Calcul de la fonction IMP.

## Émulateur de fonctions logiques

Pour terminer ce paragraphe sur les fonctions logiques, voici un programme mettant en œuvre une fonction booléenne aussi complexe que vous le voulez, constituée des opérateurs booléens AND, OR, NOT, XOR.

### *Le programme*

Donnez le nombre de variable(s) de la fonction à définir, puis entrez en 75 une fonction booléenne appelée FNA(F).

Exécutez le programme à partir de la ligne 50. La table de vérité de la fonction booléenne est alors affichée.

**Remarque :** Les variables booléennes de la fonction FNA(F) sont appelées V (i), où i varie de 1 à NV (nombre de variables).

```
1   REM ******************************
2   REM *                            *
3   REM *     MINI     EMULATEUR     *
4   REM *  DE  FONCTIONS  LOGIQUES    *
5   REM *                            *
6   REM ******************************
7   REM *                            *
8   REM * Entree : Fonction logique  *
9   REM *           Nbre de variables *
10  REM * Sortie : Table de verite   *
11  REM *                            *
12  REM ******************************
13  :
20   CLS:PRINT"Definissez votre fonction booleenne"
30   PRINT"en 75 puis tapez 'RUN 50'"
40   STOP
50   REM ------------------------------
60   INPUT "Nombre de variables ";NV
70   CLS
75   DEF FNA(F)=(V(0) AND V(1)) OR V(2)
80   FOR P=0 TO 2^NV-1
90     K1=P
100    FOR I=NV-1 TO 0 STEP -1
110      K2=K1-2^I
120      IF K2<0 THEN V(NV-1-I)=0
130      IF K2>=0 THEN V(NV-1-I)=1:K1=K2
140    NEXT I
150    FOR I=0 TO NV-1
160      PRINT V(I);
170    NEXT I
180    PRINT "!";FNA(F)
190 NEXT P
```

***Analyse du programme***

Lignes 20 à 40 : Définition.
Ligne 75 : Définition de la fonction.
Ligne 80 : Boucle sur le nombre de variables.
Lignes 100 à 170 : Affichage de l'entrée.
Ligne 180 : Affichage de la sortie.

## LES FONCTIONS RÉSERVÉES A L'UTILISATEUR : DEF FN

La fonction BASIC **DEF FN** permet de définir simplement une fonction mathématique qui sera utilisée dans la suite du programme.

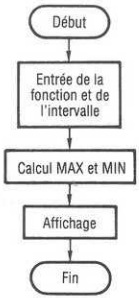
Le but de cette fonction est double :

- exécuter une seule fois la définition de cette formule et l'utiliser par la suite sous forme réduite (gain de place et de temps) ;
- permettre à un utilisateur non programmeur de définir la fonction qui l'intéresse.

***Exemple d'utilisation***

Soit une fonction mathématique. Nous voulons connaître son maximum et son minimum dans un intervalle donné.

Ce problème se symbolise comme suit :



***Programme***

```
1000 REM ****************************************
1010 REM * Recherche du minimum et du maximum    *
1020 REM * d'une fonction sur un intervalle donne*
1030 REM ****************************************
1040 :
1050 CLS:PRINT"Tapez '1150 DEF FNA(X)=' suivi de l
a"
1060 PRINT "fonction a etudier."
1070 PRINT:PRINT"Tapez ensuite 'RUN 1100'"
1080 STOP
1090 :
1100 REM Saisie de l'intervalle
1110 PRINT:PRINT"Entrez l'intervalle d'etude :"
1120 PRINT:INPUT"Minimum ";MI
1130 PRINT:INPUT"Maximum ";MA
1140 :
1150 DEF FNA(X)=SIN(X)
1160 :
1170 REM Calcul des minimum et Maximum
1173 :
1175 U=-1E+15 : V=1E+15
1180 FOR X=MI TO MA STEP (MA-MI)/100
1190   T=FNA(X)
1200   IF T>U THEN U=T : REM Maximum
1210   IF T<V THEN V=T : REM Minimum
1220 NEXT X
1230 :
1240 REM Affichage des resultats
1250 PRINT:PRINT"Le maximum est :";U
1260 PRINT:PRINT"Le minimum est :";V
1270 END
```

***Analyse du programme***

| | |
|---|---|
| Lignes 1050 à 1080 | : Présentation. |
| Lignes 1100 à 1140 | : Saisie de l'intervalle d'étude. |
| Ligne 1150 | : Définition de la fonction. |
| Lignes 1160 à 1230 | : Calcul des min et max. |
| Lignes 1240 à 1270 | : Affichage des min et max. |

# CHAÎNAGE DE DEUX PROGRAMMES AVEC PASSAGE DE PARAMÈTRES

Lorsqu'un programme occupe une place très importante en mémoire, il est parfois nécessaire de le scinder en plusieurs programmes plus courts.

Ces programmes sont alors chaînés pour réaliser l'ensemble des tâches initialement prévues.

Le problème suivant se pose : les résultats calculés dans l'un des programmes peuvent être nécessaires dans un autre programme de la chaîne. Comment préserver ces résultats ? (L'instruction "LOAD" efface en effet toutes les valeurs des variables BASIC.)

Si les programmes sont chaînés de manière immédiate, aucun problème ne se pose car le BASIC Amstrad a prévu cette éventualité. Il suffit de faire "CHAIN MERGE" <Programme chaîné>. Les variables du programme appelant ne sont pas détruites. Par contre, si le chaînage est différé dans le temps (il peut être nécessaire d'effectuer des opérations complémentaires avant le chaînage), la mémorisation sur un support magnétique des variables qui ne doivent pas être détruites s'avère nécessaire.

Supposons que le programme N calcule des valeurs qui sont nécessaires au programme N+1, et que le programme N+1 soit chaîné au programme N.

Nous allons prendre le cas très général dans lequel le programme N calcule :

- des chaînes,
- des entiers,
- des réels,
- des tableaux,

nécessaires au programme N+1.

Les deux programmes qui suivent sont à incorporer respectivement dans les programmes N et N+1. Ils sont appelés par "GOSUB 10000".

Le premier, situé dans le programme N, sauvegarde les chaînes, entiers, réels et tableaux nécessaires.

Le second, situé dans le programme N+1, récupère les chaînes, entiers, réels et tableaux sauvegardés par le premier.

### *Exemple d'appel dans le programme N*

```
10 A$="CHAINE 1":B$="CHAINE 2"
20 E1%=12:E2%=324
30 R1=-12.54:R2=6546.546
40 T$(1)="QQQQ":T$(2)="QWERTY"
50 T(1)=3:T(2)=-8.5:T(3)=45
60 GOSUB 10000
70 END
```

### *1er programme*

```
10000 REM ************************************
10001 REM *                                  *
10002 REM * PASSAGE DE PARAMETRES PAR FICHIER *
10003 REM *                                  *
10004 REM ************************************
10005 :
10010 OPENOUT "PASSAGE"
10020 PRINT #9,A$:PRINT #9,B$:REM Chaines
10030 PRINT #9,E1%:PRINT #9,E2%:REM Entiers
10040 PRINT #9,R1:PRINT #9,R2:REM Reels
10050 PRINT #9,T$(1):PRINT #9,T$(2):REM T$
10060 PRINT #9,T(1):PRINT #9,T(2):PRINT #9,T(3)
:REM T
10070 CLOSEOUT
10080 :
10090 RETURN
```

### *Analyse du programme*

Ligne 10010 : Ouverture du fichier séquentiel "PASSAGE" en écriture.
Ligne 10020 : Ecriture de chaînes.
Ligne 10030 : Ecriture d'entiers.
Ligne 10040 : Ecriture de réels.
Ligne 10050 : Ecriture de tableau alphanumérique.
Ligne 10060 : Ecriture de tableau de réels.
Ligne 10070 : Fermeture du fichier "PASSAGE".

### *Exemple d'appel dans le programme N+1*

```
10 GOSUB 10000
20 PRINT A$,B$
30 PRINT E1%,E2%
40 PRINT R1,R2
50 PRINT T$(1),T$(2)
60 PRINT T(1),T(2),T(3)
70 :
80 END
```

*2e programme*

```
10000 REM ************************************
****
10001 REM *
   *
10002 REM * LECTURE DE PARAMETRES DANS UN FICHI
ER *
10003 REM *
   *
10004 REM ************************************
****
10005 :
10010 OPENIN "PASSAGE"
10020 INPUT #9,A$:INPUT #9,B$
10030 INPUT #9,E1%:INPUT #9,E2%
10040 INPUT #9,R1:INPUT #9,R2
10050 INPUT #9,T$(1):INPUT #9,T$(2)
10060 INPUT #9,T(1):INPUT #9,T(2):INPUT #9,T(3)
10070 CLOSEIN
10080 :
10090 RETURN
```

*Analyse du programme*

Ligne 10010 : Ouverture du fichier séquentiel "PASSAGE" en lecture.
Ligne 10020 : Lecture de chaînes.
Ligne 10030 : Lecture d'entiers.
Ligne 10040 : Lecture de réels.
Ligne 10050 : Lecture de tableau alphanumérique.
Ligne 10060 : Lecture de tableau de réels.
Ligne 10070 : Fermeture du fichier "PASSAGE".

## LES FONCTIONS SPÉCIFIQUES AMSTRAD

### Contrôle des couleurs affichées sur l'écran : les instructions BORDER, PAPER, PEN et INK

Si vous éprouvez quelques difficultés pour comprendre le maniement des instructions BORDER, PAPER, PEN et INK, le programme suivant peut vous aider.

Mais sachez avant d'exécuter ce programme que :

- le nombre de couleurs pouvant apparaître simultanément à l'écran est intimement lié au mode d'affichage ;
- quel que soit le mode, la couleur du bord (BORDER) peut être une des 27 couleurs du CPC ;
- INK permet d'affecter à un stylo (PEN) une des 27 couleurs disponibles ;
- PEN permet de choisir l'encre d'écriture, après l'avoir initialisée par l'instruction INK.

***Programme***

```
10 REM Demonstration sur les ordres BORDER, PAP
ER, PEN et INK
20 REM en MODE 0, MODE 1 et MODE 2.
30 :
1000 REM ******************************
1010 REM * DEMONSTRATION DE  PEN      *
1020 REM ******************************
1030 MODE 0
1040 PRINT"En mode 0, on a 16"
1050 PRINT"couleurs d'encre :"
1060 PRINT:PRINT"   Couleur     PEN":PRINT
1070 FOR I=0 TO 15
1080   PEN I
1090   READ A$:PRINT A$
1100   LOCATE 15,I+6:PRINT I
1110 NEXT I
1120 DATA Bleu,Jaune vif,Turquoise vif,Rouge vi
f,Blanc brillant,Noir
1130 DATA Bleu vif,Magenta vif,Turquoise,Jaune,
Bleu pastel,Rose
1140 DATA Vert vif,Vert pastel,Bleu/Jaune vif,R
ose/Bleu ciel
1150 PRINT:PRINT"Appuyez sur 1 touche"
1160 A$=INKEY$:IF A$="" THEN 1160
1170 :
1180 MODE 1
1190 PRINT"En mode 1 on a 4 couleurs d'encre :"
1200 PRINT:PRINT"   Couleur     PEN"
1210 FOR I=0 TO 3
1220   PEN I
1230   READ A$:PRINT A$
1240   LOCATE 15,I+6:PRINT I
1250 NEXT I
1260 DATA Bleu,Jaune vif,Turquoise vif,Rouge vi
f
1270 PRINT:PRINT"Appuyez sur 1 touche"
```

```
1280 A$=INKEY$:IF A$="" THEN 1280
1290 :
1300 MODE 2
1310 PRINT"En mode 2 on a 2 couleurs d'encre :"
1320 PRINT:PRINT"   Couleur     PEN":PRINT
1330 FOR I=0 TO 1
1340   PEN I
1350   READ A$:PRINT A$
1360   LOCATE 15,I+6:PRINT I
1370 NEXT I
1380 DATA Bleu,Jaune vif
1390 PRINT:PRINT"Appuyez sur 1 touche"
1400 A$=INKEY$:IF A$="" THEN 1400
1410 :
2000 REM ******************************
2010 REM * DEMONSTRATION DE  PAPER    *
2020 REM ******************************
2030 MODE 1
2040 PRINT"De meme, le MODE 0 permet 16 couleur
s"
2050 PRINT"de papier; Le MODE 1 permet 4 couleu
rs"
2060 PRINT"de papier et le MODE 2 permet 2 coul
eurs";
2070 PRINT"de papier."
2080 :
2090 PRINT:PRINT"En MODE 0, et en se reportant
dans le"
2100 PRINT"tableau 2 du guide de l'utilisateur,
"
2110 PRINT"on voit que PAPER 12 (par ex.) donne
"
2120 PRINT"la couleur 18, donc Vert vif (dans l
e"
2130 PRINT"tableau 1)
2140 PRINT:PRINT"Appuyez sur une touche pour co
ntinuer"
2150 A$=INKEY$:IF A$="" THEN 2150
2160 MODE 0:PAPER 12:CLS
2170 PRINT"Effectivement,"
2180 PRINT"PAPER 12 donne la"
2190 PRINT"couleur Vert Vif."
2200 PRINT:PRINT"Appuyez sur 1 touche";
2210 PRINT"pour continuer"
2220 A$=INKEY$:IF A$="" THEN 2220
2230 :
2240 MODE 1
2250 PRINT"De meme, en MODE 1, PAPER 2 donnera
la"
```

→

```
2260 PRINT"couleur 20 donc Turquoise vif, et "
2270 PRINT"en MODE 2, PAPER 1 donnera la couleu
r "
2280 PRINT"24 donc Jaune Vif a l'ecran."
2290 PRINT:PRINT"Appuyez sur une touche pour co
ntinuer"
2300 A$=INKEY$:IF A$="" THEN 2300
3000 REM ******************************
3010 REM * DEMONSTRATION DE  BORDER   *
3020 REM ******************************
3030 MODE 1
3040 PRINT"Quel que soit le mode, BORDER donne
la"
3050 PRINT"couleur de la bordure de l'ecran."
3060 PRINT"Cette couleur peut etre choisie parm
i"
3070 PRINT"les 27 couleurs de base."
3080 PRINT:PRINT"Appuyez sur une touche pour co
ntinuer"
3090 A$=INKEY$:IF A$="" THEN 3090
3100 FOR I=0 TO 26
3110   BORDER I
3120   LOCATE 10,10:PRINT"BORDER";I
3130   FOR J=1 TO 800:NEXT J
3140 NEXT I
3150 PRINT:PRINT"Appuyez sur une touche pour co
ntinuer"
3160 A$=INKEY$:IF A$="" THEN 3160
3170 :
4000 REM ******************************
4010 REM * DEMONSTRATION DE   INK     *
4020 REM ******************************
4030 BORDER 1:CLS
4040 PRINT"Si le mode 0 permet d'afficher 16"
4050 PRINT"couleurs differentes sur l'ecran,"
4060 PRINT"le mode 1 quatre couleurs differente
s"
4070 PRINT"et le mode 2 deux couleurs different
es,"
4080 PRINT"il est possible de choisir ces coule
urs"
4090 PRINT"dans la palette des 27 couleurs "
4100 PRINT"disponibles par l'instruction"
4110 PRINT:PRINT"INK X,Y avec X=No de Stylo (PE
N)"
4120 PRINT"          et Y=Couleur du Stylo"
4130 PRINT:PRINT"Par ex., en mode 1, pour que l
e stylo"
```

```
4140 PRINT"2 ait la couleur Vert Pastel, on fer
a "
4150 PRINT"INK 2,22 puis PEN 2 pour selectionne
r"
4160 PRINT"le stylo 2."
4170 INK 2,22:PEN 2
4180 PRINT:PRINT"L'ecriture est maintenant Vert
 Pastel."
```

***Analyse du programme***

Lignes 1000 à 1410 : Démonstration de PEN.
Lignes 1030 à 1170 : en mode 0.
Lignes 1180 à 1280 : en mode 1.
Lignes 1300 à 1400 : en mode 2.
Lignes 2000 à 2300 : Démonstration de PAPER.
Lignes 2040 à 2220 : en mode 0.
Lignes 2240 à 2300 : en mode 1.
Lignes 3000 à 3170 : Démonstration de BORDER.
Lignes 4000 à 4180 : Démonstration de INK.

## RANDOMIZE

L'instruction "RANDOMIZE <Expression Numérique>" permet de fixer la séquence pseudo-aléatoire du générateur de nombres aléatoires de l'Amstrad.

Lorsque vous désirez utiliser le générateur aléatoire dans un programme, l'instruction "RANDOMIZE" permet d'avoir des séquences aléatoires différentes.

Si vous voulez utiliser la première forme possible pour RANDOMIZE, il vous faut fixer l'expression numérique (pour deux expressions numériques identiques, la séquence générée sera la même).

Pour fixer l'argument du RANDOMIZE, vous pouvez demander à la personne qui exécute le programme d'entrer un nombre N, et faire RANDOMIZE N ; mais voici deux programmes qui vous permettront d'éviter cette démarche un peu lourde.

Le premier programme attend l'appui sur une touche. Tant que le clavier n'est pas actionné, la variable J est incrémentée. Dès qu'une action clavier est détectée, l'instruction RANDOMIZE J est exécutée.

***Programme***

```
100 REM Utilisation du RANDOMIZE
110 :
120 CLS:PRINT"Appuyez sur une touche":PRINT
130 :
140 A$=INKEY$
150 J=J+1
160 IF A$="" THEN 140
170 :
180 RANDOMIZE J
190 :
200 FOR I=1 TO 10
210   PRINT INT(RND*10);
220 NEXT I
230 :
240 END
```

***Analyse du programme***

Ligne 120 : Message à l'écran.
Lignes 140 à 160 : Attente d'une action clavier.
Ligne 180 : RANDOMIZE.
Lignes 200 à 230 : Affichage d'une séquence pseudo-aléatoire.

Le deuxième programme se sert de l'instruction TIME comme RANDOMIZE. L'instruction "RANDOMIZE TIME" donnant d'assez mauvais résultats, une expression un peu plus complexe a été choisie.

***Programme***

```
100 REM Utilisation de TIME comme RANDOMIZE
110 :
120 RANDOMIZE RND(TIME)*1000
130 FOR I=1 TO 10
140   PRINT INT(RND*10);
150 NEXT I
160 :
170 END
```

***Analyse du programme***

Ligne 120 : RANDOMIZE.
Lignes 130 à 150 : Affichage d'une séquence pseudo-aléatoire.

## INT, CINT, CREAL, FIX, ROUND

Si vous êtes un peu déconcerté devant l'abondance des instructions permettant d'extraire les parties fractionnaire ou entière d'un nombre réel, voici un programme qui pourra vous fixer les idées sur leur utilisation.

### *Programme*

```
100 REM INT, CINT, CREAL, FIX, ROUND
110 :
120 CLS
121 PRINT"  Test de INT,CINT,CREAL,FIX et ROUND
":PRINT:PRINT:PRINT
130 INPUT "Entrez un nombre ";N
140 PRINT
150 PRINT"INT("N")     ="INT(N)
160 PRINT"CINT("N")    ="CINT(N)
170 PRINT"CREAL("N")   ="CREAL(N)
180 PRINT"FIX("N")     ="FIX(N)
190 PRINT"ROUND("N",2)="ROUND(N,2)
200 PRINT"ROUND("N",4)="ROUND(N,4)
210 PRINT
220 LOCATE 1,20:INPUT"Un autre essai (O/N) : ";
R$
230 IF R$<>"O" AND R$<>"N" THEN PRINT CHR$(7):G
OTO 220
240 IF R$="O" THEN 100
250 END
```

### *Analyse du programme*

| | |
|---|---|
| Ligne 130 | : Saisie clavier d'un nombre réel. |
| Lignes 140 à 210 | : Affichage du résultat des fonctions INT, CINT, CREAL, FIX et ROUND. |
| Lignes 220 à 240 | : Poursuite ou arrêt du programme. |

## TAG

Si certains ordinateurs familiaux demandent des acrobaties innommables pour afficher du texte sur un écran graphique, l'Amstrad ne fait pas partie de ceux-là.

L'instruction TAG permet d'afficher des caractères alphanumériques à partir de la position courante du curseur.

Jugez plutôt de la simplicité d'emploi de TAG :

```
100 REM Demonstration de TAG
110 :
120 CLS
130 FOR A=1 TO 360 STEP 3
140   MOVE 270+190*COS(A),200+190*SIN(A)
150   TAG:PRINT "CPC464";
160 NEXT A
```

Les deux programmes suivants affirment la souplesse d'utilisation de TAG. Le premier programme montre comment graduer les deux axes d'un repère Oxy.

***Programme***

```
100 REM Exemple d'utilisation de TAG
110 :
120 CLS
130 REM Axes
140 :
150 MOVE 60,20:DRAW 570,20
160 MOVE 60,20:DRAW 60, 390
170 :
180 REM Graduations
190 :
200 FOR I=1 TO 10
210   MOVE 60+I*50,10
220   DRAWR 0,20
230 NEXT I
240 FOR I=1 TO 10
250   MOVE 50,20+I*35
260   DRAWR 20,0
270 NEXT I
280 :
290 REM Annotations
300 :
310 FOR I=1 TO 10
320   MOVE I*50-3+40,15
330   TAG:PRINT I;
340 NEXT I
350 FOR I=1 TO 10
360   MOVE -10,25+I*35
370   TAG:PRINT I;
380 NEXT I
```

### *Analyse du programme*

Lignes 130 à 170 : Axes Oxy.
Lignes 180 à 280 : Graduations.
Lignes 290 à 380 : Annotations sur les axes.

Le deuxième programme est une extension du premier. En effet, il permet de tracer, de graduer et d'annoter les deux axes d'un repère Oxy paramétrable. En d'autres termes, c'est vous qui choisissez les coordonnées d'origine du repère, ses dimensions, le nombre de graduations sur chaque axe, et les valeurs à porter en face des graduations. Toutes ces informations sont transmises dans un tableau T(10).

### *Programme*

```
5 MODE 2
10 FOR I=1 TO 10:READ T(I):NEXT I
20 DATA 90,60,400,200,5,6,12,5,-0.09,0.02
30 CLS:GOSUB 10190
40 END

10000 REM ************************************
10010 REM *                                  *
10020 REM * TRACE DE REPERES Oxy PARAMETRABLE *
10030 REM *                                  *
10040 REM ************************************
10050 REM *                                  *
10060 REM * Entree : T(1)=Abcisse origine     *
10070 REM *          T(2)=Ordonnee origine    *
10080 REM *          T(3)=Dimension Ox        *
10090 REM *          T(4)=Dimension Oy        *
10100 REM *          T(5)=Nombre de reperes Ox *
10110 REM *          T(6)=Nombre de reperes Oy *
10120 REM *          T(7)=Val. de X a l'origine *
10130 REM *          T(8)=Increment de X      *
10140 REM *          T(9)=Val. de Y a l'origine *
10150 REM *          T(10)=Increment de Y     *
10160 REM *                                  *
10170 REM ************************************
10180 :
10190 XO=T(1):YO=T(2):LX=T(3):LY=T(4)
10200 NX=T(5):NY=T(6):DX=T(7):IX=T(8)
10210 DY=T(9):IY=T(10) 'Initialisation
10220 :
```

→

```
10230 REM Definition des fleches
10240 :
10250 SYMBOL AFTER 129
10260 SYMBOL 130,&C0,&60,&30,&18,&18,&30,&60,&C0
10270 SYMBOL 131,0,0,0,&18,&3C,&66,&C3,&81
10280 REM Trace des axes
10290 :
10300 MOVE X0,Y0:DRAWR LX,0
10310 MOVE X0,Y0:DRAWR 0,LY
10320 :
10330 REM Trace des fleches
10340 :
10350 MOVE X0+LX,Y0+6:TAG:PRINT CHR$(130);
10360 MOVE X0-4,Y0+LY:TAG:PRINT CHR$(131);
10370 :
10380 REM Trace des graduations
10390 :
10400 FOR I=1 TO NX
10410   MOVE X0+(LX/(NX+1))*I,Y0-5
10420   DRAWR 0,10
10430 NEXT I
10440 :
10450 FOR I=1 TO NY
10460   MOVE X0-5,Y0+(LY/(NY+1))*I
10470   DRAWR 10,0
10480 NEXT I
10490 :
10500 REM Calcul des echelles
10510 :
10520 MX=ABS(DX+IX*NX) 'Maximum en X
10530 MY=ABS(DY+IY*NY) 'Maximum en Y
10540 :
10550 I=0
10560 IF MX<1 THEN 10650
10570 :
10580 WHILE MX>1
10590   I=I+1
10600   MX=MX/10
10610 WEND
10620 PX=I-1
10630 GOTO 10710
10640 :
10650 WHILE MX<1
10660   I=I+1
10670   MX=MX*10
10680 WEND
10690 PX=-I
10700 :
```

```
10710 I=0
10720 IF MY<1 THEN 10810
10730 :
10740 WHILE MY>1
10750   I=I+1
10760   MY=MY/10
10770 WEND
10780 PY=I-1
10790 GOTO 10860
10800 :
10810 WHILE MY<1
10820   I=I+1
10830 MY=MY*10
10840 WEND
10850 PY=-I
10860 :
10870 REM Annotations
10880 :
10890 FOR I=0 TO NX
10900   MOVE X0+I*(LX/(NX+1))-20,Y0-20
10910   A=INT(((DX+IX*I)/10^PX)*10)/10
10920   TAG:PRINT A;
10930 NEXT I
10940 :
10950 FOR I=0 TO NY
10960   MOVE X0-50,Y0+I*(LY/(NY+1))+5
10970   A=INT(((DY+IY*I)/10^PY)*10)/10
10980   TAG:PRINT A;
10990 NEXT I
11000 :
11010 REM Exposants
11020 :
11030 MOVE X0+LX+20,Y0+10:TAG:PRINT"x^"PX;
11040 MOVE X0-20,Y0+LY+20:TAG:PRINT"x^"PY;
11050 :
11060 RETURN
```

***Analyse du programme***

Lignes 10190 à 10210 : Conversion des entrées.
Lignes 10230 à 10270 : Définition des caractères graphiques.
Lignes 10280 à 10310 : Tracé des axes.
Lignes 10330 à 10360 : Tracé des flèches.
Lignes 10380 à 10480 : Tracé des graduations.
Lignes 10500 à 10850 : Calcul des échelles.
Lignes 10870 à 10990 : Annotations sur les axes.
Lignes 11010 à 11060 : Facteurs multiplicatifs sur les axes.

# LES BONNES ADRESSES DES ROUTINES ASSEMBLEUR

Les routines graphiques dont la description suit ont servi à l'élaboration du programme "Remplissage de surfaces délimitées par un contour fermé".

Ces renseignements sont tirés du « CPC 464 FIRMWARE » qui contient l'explication détaillée des routines de la ROM 464.

## GRA TEST ABSOLUTE #BBFO

Teste l'état d'un pixel à une position absolue.

Entrée : DE = abscisse X du point.
HL = coordonnée Y du point.

Sortie : A = INK du point considéré.

BC, DE, HL, flags écrasés.

## GRA TEST RELATIVE #BBF3

Teste l'état d'un pixel à une position relative.

Entrée : DE = abscisse relative X du point.
HL = coordonnée relative Y du point.

Sortie : A = INK du point considéré.

BC, DE, HL, flags écrasés.

## GRA PLOT ABSOLUTE #BBEA

Réalise l'instruction PLOT à une position absolue.

Entrée : DE = abscisse X du point.
HL = coordonnée Y du point.

AF, BC, DE, HL écrasés.

### GRA PLOT RELATIVE #BBED

Réalise l'instruction PLOT à une position relative.

Entrée : DE = abscisse relative X du point.
HL = coordonnée relative Y du point.

AF, BC, DE, HL écrasés.

### GRA SET PEN #BBDE

Choisit la couleur de l'encre en HIRES.

Entrée : A = encre choisie.

AF écrasé.

### GRA LINE ABSOLUTE #BBF6

Trace une ligne vers une position absolue.

Entrée : DE = abscisse X du point à joindre.
HL = coordonnée Y du point à joindre.

AF, BC, DE, HL écrasés.

### GRA LINE RELATIVE #BBF9

Trace une ligne vers une position relative.

Entrée : DE = abscisse relative X du point à joindre.
HL = coordonnée relative Y du point à joindre.

AF, BC, DE, HL écrasés.

### GRA MOVE ABSOLUTE #BBC0

Positionne le curseur graphique à une position absolue.

Entrée : DE = abscisse X demandée.
HL = coordonnée Y demandée.

AF, BC, DE, HL écrasés.

## GRA MOVE RELATIVE #BBC3

Positionne le curseur graphique à une position relative.

Entrée : DE = abscisse demandée.
HL = coordonnée demandée.

AF, BC, DE, HL écrasés.

Voici quelques autres adresses à usage général :

- **CALL &BBFF** Initialise l'écran.
- **CALL &BC14** Efface l'écran.
- **CALL &BB03** Initialise le clavier.
- **CALL &BB06** ou **CALL &BB18** Attend l'appui sur une touche.

Parmi les ordres BASIC évolués, spécifiques à l'Amstrad, qui ne seront pas développés ou exploités par la suite, signalons :

- **BIN$** pour convertir un nombre de base 10 en base 2.
- **DI** et **EI** qui permettent respectivement l'annulation et le rétablissement des interruptions (autres que BREAK) au niveau BASIC.
- **DEG** et **RAD** pour choisir le mode de calcul trigonométrique en DEGrés ou en RADians.
- **EOF** variable booléenne qui teste si l'unité de cassette est à la fin du fichier courant.
- **HIMEM** qui donne l'adresse de l'octet le plus élevé de la mémoire utilisée par BASIC.
- **INSTR** qui donne la position d'une sous-chaîne dans une chaîne.
- **JOY** pour lire la position du joystick.
- **KEY** et **KEY DEF** qui permettent, pour la première, de définir une touche fonction parmi 32, et pour la seconde, de changer la valeur alphanumérique produite par la frappe d'une touche.
- **LOWER$** et **UPPER$** qui permettent, pour la première, de transformer les caractères majuscules de l'argument en caractères minuscules, et pour la seconde, de faire l'opération inverse.
- **SPEED INK** pour associer les temps d'apparition des couleurs liées aux commandes INK et BORDER.

- **SPEED KEY** pour définir la vitesse de répétition des touches au clavier.
- **SPEED WRITE** pour définir la vitesse de sauvegarde cassette.
- **STRING$** pour concaténer n fois le même argument alphanumérique.
- **WIDTH** pour choisir le nombre de caractères par ligne sur une sortie imprimante.

# Chapitre 2 | Améliorations du Basic Amstrad

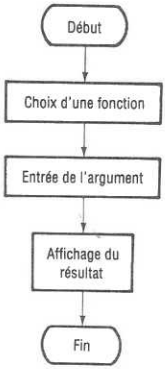
## FONCTIONS SINUSOÏDALES ET HYPERBOLIQUES

### Fonctions sinusoïdales

L'Amstrad est assez pauvre en fonctions sinusoïdales. En effet, il possède seulement les fonctions **ATN** (Arctangente), **COS** (Cosinus) et **SIN** (Sinus).

Le programme qui suit lui rajoute les fonctions : Sécante, Cosécante, Cotangente, Sinus inverse, Cosinus inverse, Sécante inverse, Cosécante inverse, Cotangente inverse.

Sa structure est très simple :



***Programme***

```
1 REM ******************************
2 REM *                            *
3 REM *   FONCTIONS  SINUSOIDALES  *
4 REM *                            *
5 REM ******************************
6 :
7 DEF FNA(A)=1/COS(A)
8 DEF FNB(A)=1/SIN(A):DEF FNC(A)=1/TAN(A)
9 DEF FND(A)=ATN(A/SQR(-A*A+1))
10 DEF FNE(A)=-ATN(A/SQR(-A*A+1))+1.5708
11 DEF FNF(A)=ATN(SQR(A*A-1))+(SGN(A)-1)*1.5708
12 DEF FNG(A)=ATN(1/SQR(A*A-1))+(SGN(A)-1)*1.5708
13 DEF FNH(A)=-ATN(A)+1.5708
14 :
20 CLS:PRINT"Choisissez votre fonction :"
30 PRINT:PRINT"1)Secante"
40 PRINT"2)Cosecante"
50 PRINT"3)Cotangente"
60 PRINT"4)Sinus inverse"
70 PRINT"5)Cosinus inverse"
80 PRINT"6)Secante inverse"
90 PRINT"7)Cosecante inverse"
100 PRINT"8)Cotangente inverse"
110 PRINT:INPUT"Fonction choisie ";F
120 PRINT:INPUT"Argument ";A
130 ON F GOSUB 200,210,220,230,240,250,260,270
140 PRINT:PRINT"Resultat :";X
150 END
160 REM ------------------------------
200 REM Calcul Secante
201   X=FNA(A)
202 RETURN
203 REM ------------------------------
210 REM Calcul Cosecante
211   X=FNB(A)
212 RETURN
213 REM ------------------------------
220 REM Calcul Cotangente
221   X=FNC(A)
222 RETURN
223 REM ------------------------------
230 REM Calcul Sinus Inverse
231   X=FND(A)
232 RETURN
233 REM ------------------------------
240 REM Calcul Cosinus Inverse
241   X=FNE(A)
```

```
242 RETURN
243 REM ------------------------------------
250 REM Calcul Secante Inverse
251   X=FNF(A)
252 RETURN
253 REM ------------------------------------
260 REM Calcul Cosecante Inverse
261   X=FNG(A)
262 RETURN
263 REM ------------------------------------
270 REM Calcul Cotangente Inverse
271   X=FNH(A)
272 RETURN
```

### *Analyse du programme*

| | |
|---|---|
| Lignes 6 à 13 | : Définition des fonctions. |
| Lignes 20 à 120 | : Menu. |
| Ligne 130 | : Déroutement sur la procédure correspondant à la fonction choisie. |
| Ligne 140 | : Affichage du résultat. |
| Lignes 200 à 272 | : Routines de calcul des fonctions. |

## Fonctions hyperboliques

L'Amstrad est totalement dépourvu des fonctions hyperboliques usuelles. Le programme qui suit permet d'intégrer les fonctions suivantes :

- Sinus hyperbolique, Cosinus hyperbolique, Tangente hyperbolique, Sécante hyperbolique, Cosécante hyperbolique, Cotangente hyperbolique, Sinus hyperbolique inverse, Cosinus hyperbolique inverse, Tangente hyperbolique inverse, Sécante hyperbolique inverse, Cosécante hyperbolique inverse, Cotangente hyperbolique inverse.

### *Programme*

```
1 REM ********************************
2 REM *                              *
3 REM * LES FONCTIONS HYPERBOLIQUES  *
4 REM *                              *
5 REM ********************************
6 :
7 DEF FNA(A)=(EXP(A)-EXP(-A))/2
8 DEF FNB(A)=(EXP(A)+EXP(-A))/2
9 DEF FNC(A)=-EXP(-A)/(EXP(A)+EXP(-A))*2+1
10 DEF FND(A)=2/(EXP(A)+EXP(-A))
```

```
11 DEF FNE(A)=2/(EXP(A)-EXP(-A))
12 DEF FNF(A)=EXP(-A)/(EXP(A)-EXP(-A))*2+1
13 DEF FNG(A)=LOG(A+SQR(A*A+1))
14 DEF FNH(A)=LOG(A+SQR(A*A-1))
15 DEF FNI(A)=LOG((1+A)/(1-A))/2
16 DEF FNJ(A)=LOG((SQR(-A*A+1)+1))/A
17 DEF FNK(A)=LOG(SGN(A)*SQR(A*A+1)+1)/A
18 DEF FNL(A)=LOG((A+1)/(A-1))/2
19 :
20 CLS:PRINT"Choisissez votre fonction :"
30 PRINT:PRINT"1)Sinus Hyperbolique"
40 PRINT"2)Cosinus Hyperbolique"
50 PRINT"3)Tangente Hyperbolique"
60 PRINT"4)Secante Hyperbolique"
70 PRINT"5)Cosecante hyperbolique"
80 PRINT"6)Cotangente Hyperbolique"
90 PRINT"7)Sinus Hyperbolique Inverse"
100 PRINT"8)Cosinus Hyperbolique Inverse"
110 PRINT"9)Tangente Hyperbolique Inverse"
120 PRINT"10)Secante Hyperbolique Inverse"
130 PRINT"11)Cosecante Hyperbolique Inverse"
140 PRINT"12)Cotangente Hyperbolique Inverse"
150 PRINT:INPUT"Fonction choisie ";F
155 PRINT:INPUT"Argument ";A
160 ON F GOSUB 200,210,220,230,240,250,260,270,
280,290,300,310
170 PRINT:PRINT"Resultat :";X
180 END
185 :
190 REM ------------------------------------
200 REM Sinus Hyperbolique
201   X=FNA(A)
202 RETURN
203 REM ------------------------------------
210 REM Cosinus Hyperbolique
211   X=FNB(A)
212 RETURN
213 REM ------------------------------------
220 REM Tangente Hyperbolique
221   X=FNC(A)
222 RETURN
223 REM ------------------------------------
230 REM Secante Hyperbolique
231   X=FND(A)
232 RETURN
233 REM ------------------------------------
240 REM Cosecante Hyperbolique
241   X=FNE(A)
242 RETURN
```

```
243 REM ------------------------------------------------
250 REM Cotangente Hyperbolique
251   X=FNF(A)
252 RETURN
253 REM ------------------------------------------------
260 REM Sinus Hyper. Inverse
261   X=FNG(A)
262 RETURN
263 REM ------------------------------------------------
270 REM Cosinus Hyper. Inverse
271   X=FNH(A)
272 RETURN
273 REM ------------------------------------------------
280 REM Tangente Hyper. Inverse
281   X=FNI(A)
282 RETURN
283 REM ------------------------------------------------
290 REM Secante Hyper. Inverse
291   X=FNJ(A)
292 RETURN
293 REM ------------------------------------------------
300 REM Cosecante Hyper. Inverse
301   X=FNK(A)
302 RETURN
303 REM ------------------------------------------------
310 REM Cotangente Hyper. Inverse
311   X=FNL(A)
312 RETURN
```

***Analyse du programme***

Lignes 6 à 18 : Définition des fonctions.
Lignes 20 à 155 : Menu.
Ligne 160 : Déroutement sur la procédure correspondant à la fonction choisie.
Ligne 170 : Affichage du résultat.
Lignes 200 à 312 : Routines de calcul des fonctions.

La structure de ce programme est identique à celle du programme précédent.

# FONCTIONS MOD ET DIV

## Fonction MOD (modulo)

Définition : Reste arithmétique découlant de la division de A par B.

Exemple : 8 MOD 5 = 3 (reste entier de 8 divisé par 5).

Programme : Fournir les deux nombres A et B pour lesquels on veut connaître la fonction MOD.
C = A MOD B est la sortie du programme.
Ce programme est appelé par "GOSUB 10000".

### *Exemple*

```
10 A=8:B=5
20 GOSUB 10000
30 PRINT C:REM A MOD B
40 END
```

### *Programme*

```
10000 REM ********************************
10001 REM *                              *
10002 REM * SIMULATION DE LA FONCTION MOD *
10003 REM *                              *
10004 REM ********************************
10005 REM *                              *
10006 REM * Entree : A et B              *
10007 REM * Sortie : C = A MOD B         *
10008 REM *                              *
10009 REM ********************************
10010 :
10020 C=INT(A-INT(A/B)*B)
10030 :
10040 RETURN
```

### *Analyse du programme*

Ligne 10020 : Calcul de la fonction MOD.

## Fonction DIV

Définition : Résultat de la division entière de A par B.

Exemples : 8 DIV 5 = 1 ; 14 DIV 3 = 4.

Programme : Fournir les deux nombres A et B pour lesquels on veut connaître la fonction DIV.
C = A DIV B est la sortie du programme.
Ce programme est appelé par "GOSUB 10000".

***Exemple***

```
10 A=13:B=2
20 GOSUB 10000
30 PRINT C:REM A DIV B
40 END
```

***Programme***

```
10000 REM ********************************
10001 REM *                              *
10002 REM * SIMULATION DE LA FONCTION DIV *
10003 REM *                              *
10004 REM ********************************
10005 REM *                              *
10006 REM * Entree : A et B              *
10007 REM * Sortie : C = A DIV B         *
10008 REM *                              *
10009 REM ********************************
10010 :
10020 C=INT(A/B)
10030 :
10040 RETURN
```

***Analyse du programme***

Ligne 10020 : Calcul de la fonction DIV.

## OPÉRATIONS EN DOUBLE PRÉCISION

### Addition en double précision

Définition : Le Basic de l'Amstrad permet de faire des additions de nombres réels avec un nombre total de chiffres (avant et après la virgule) n'excédant pas 9.
Ce programme permet de faire des additions sur des nombres réels positifs avec 9 chiffres maximum avant la virgule, et 9 chiffres maximum après la virgule, d'où le nom d'addition en double précision.

Programme : Les deux nombres positifs à additionner sont placés dans les chaînes A$ et B$.
Le résultat en double précision est exprimé dans la chaîne C$.
Le programme est appelé par "GOSUB 10000".

***Exemple***

```
10 A$="98765432.56":B$="76564532.43578543"
20 GOSUB 10000
30 PRINT C$
40 END
```

***Programme***

```
10000 REM ************************************
*
10001 REM *
*
10002 REM * ADDITION EN DOUBLE PRECISION
*
10003 REM *
*
10004 REM ************************************
*
10005 REM *
*
10006 REM * Entree : A$, B$ Nombres a
*
10007 REM *           additionner
*
10008 REM * Sortie : C$=A$+B$
*
10009 REM *
*
10010 REM ************************************
*
10011 :
10020 REM Extraction partie entiere, partie ree
lle
10030 :
10040 J=0:E1=0:D1=0
10050 FOR I=1 TO LEN(A$)
10060   IF MID$(A$,I,1)="." THEN J=I
10070 NEXT I
10075 IF J=0 THEN J=LEN(A$)+1
10080 FOR I=1 TO J-1
10090   X=ASC(MID$(A$,I,1))-48
10095   E1=E1+X*10^(J-I-1)
10100 NEXT I
```

```
10105 IF J-1=LEN(A$) THEN D1=0:GOTO 10140
10110 FOR I=J+1 TO LEN(A$)
10120   D1=D1+(ASC(MID$(A$,I,1))-48)*10^(LEN(A$
)-I)
10130 NEXT I
10140 J=0:E2=0:D2=0
10150 FOR I=1 TO LEN(B$)
10160   IF MID$(B$,I,1)="." THEN J=I
10170 NEXT I
10180 FOR I=1 TO J-1
10190   X=ASC(MID$(B$,I,1))-48
10195   E2=E2+X*10^(J-I-1)
10200 NEXT I
10205 IF J-1=LEN(B$) THEN D2=0:GOTO 10240
10210 FOR I=J+1 TO LEN(B$)
10220   D2=D2+(ASC(MID$(B$,I,1))-48)*10^(LEN(B$
)-I)
10230 NEXT I
10235 L1=LEN(STR$(INT(D1))):L2=LEN(STR$(INT(D2)
))
10236 IF L1>L2 THEN D2=D2*10^(L1-L2)
10237 IF L2>L1 THEN D1=D1*10^(L2-L1)
10238 D1=INT(D1):D2=INT(D2)
10240 :
10250 REM Addition
10260 :
10280 D$=STR$(INT(D1+D2))
10290 IF LEN(D$)=L1 OR LEN(D$)=L2 THEN R=0:GOTO
 10310
10300 R=1:REM Mise a 1 de la retenue
10305 D$=RIGHT$(D$,LEN(D$)-1-R)
10310 :
10320 G$=STR$(INT(E1+E2+R))
10330 C$=G$+"."+D$
10340 :
10350 RETURN
```

***Analyse du programme***

Lignes 10020 à 10240 : Extraction des parties entières et décimales des nombres à additionner.
Lignes 10250 à 10350 : Addition.

**Remarque :** Pour ne pas présenter un programme trop complexe, seule l'addition de nombres réels positifs a été envisagée. Le lecteur pourra, à moindres frais, transformer ce programme pour permettre l'addition de nombres positifs ou négatifs. Enfin, en partant du même principe (manipulation de chaînes), il sera intéressant d'écrire des programmes de soustraction, multiplication, division, extraction de racine carrée, etc.

# CHANGEMENT DE BASE

Définition : Base = nombre d'unités d'ordre n nécessaires pour former une unité d'ordre n+1. Ainsi, en base N, les nombres s'échelonnent de 0 à N−1, puis de 10 à 1(N−1), etc.

## Conversion base B → base 10

Définition : Un nombre en base B est converti en le même nombre exprimé en base 10.

Programme : Fournir le nombre en base B rangé par unités dans le tableau A.
Fournir la base B dans laquelle est exprimé ce nombre.
Le résultat est le même nombre exprimé en base 10 dans N.
Ce programme est appelé par "GOSUB 10000".

### *Exemple*

```
10 DIM A(100)
20 A(1)=2:A(2)=3:A(3)=1:B=7
30 GOSUB 10000:REM Conversion
40 PRINT N:REM Nombre converti
50 END
```

### *Programme*

```
10000 REM ************************************
10001 REM *                                  *
10002 REM *       CHANGEMENT  DE  BASE       *
10003 REM *       BASE B  ->  BASE 10        *
10004 REM *                                  *
10005 REM ************************************
10006 REM *                                  *
10007 REM * Entree : A=Nombre a convertir     *
10008 REM *           B=Base de conversion   *
10009 REM * Sortie : N=Nombre converti        *
10010 REM *                                  *
10011 REM ************************************
10012 :
10020 FOR I=1 TO 100
10030   IF A(I)<>0 THEN J=I
10040 NEXT I
10050 :
10060 FOR I=1 TO J
10070   N=N+A(I)*B^(J-I)
10080 NEXT I
10090 :
10100 RETURN
```

### *Analyse du programme*

Lignes 10020 à 10040 : Calcul du nombre de chiffres du nombre exprimé en base B.
Lignes 10060 à 10080 : Conversion.

## Conversion base 10 → base B

Définition : Un nombre en base 10 est converti en le même nombre exprimé en base B.

Programme : Fournir le nombre N en base 10.
Fournir la base désirée B.
Le résultat est le même nombre exprimé en base B dans le tableau A où chaque case représente une unité.
Ce programme est appelé par "GOSUB 10000".

### *Exemple*

```
10 DIM A(100)
20 N=122:B=16:GOSUB 10000
30 FOR I=J TO 1 STEP -1
40   PRINT A(I);
50 NEXT I
60 END
```

### *Programme*

```
10000 REM ************************************
10001 REM *                                  *
10002 REM *       CHANGEMENT  DE  BASE       *
10003 REM *       BASE 10 ->  BASE B         *
10004 REM *                                  *
10005 REM ************************************
10006 REM *                                  *
10007 REM * Entree : N=Nombre a convertir     *
10008 REM *          B=Base de conversion    *
10009 REM * Sortie : A=Nombre converti        *
10010 REM *          J=Nombre de chiffres    *
10011 REM *          significatifs dans A    *
10012 REM *                                  *
10013 REM ************************************
10014 :
10020 R=N
10030 I=I+1
10040 A(I)=R-INT(R/B)*B:R=INT(R/B)
10050 IF R>=B THEN 10030
10060 A(I+1)=R
10070 FOR I=1 TO 100
10080   IF A(I)<>0 THEN J=I
10090 NEXT I
10100 :
10110 RETURN
```

***Analyse du programme***

Lignes 10020 à 10060 : Conversion.
Lignes 10070 à 10090 : Calcul du nombre de chiffres du nombre exprimé en base B.

## OPÉRATIONS SUR LES MATRICES

Les habitués du calcul matriciel seront bien déçus en constatant que le BASIC Amstrad ne comporte aucune instruction sur ce sujet. Que ces gens-là se rassurent : voici un ensemble d'utilitaires qui vont permettre de manipuler simplement les matrices. Ces utilitaires sont élémentaires. Ils autorisent les opérations de base (addition de matrices, inversion de matrices, saisie de matrices, etc.) nécessaires à toute résolution informatique d'un problème matriciel.

Tous ces utilitaires sont regroupés en un seul programme. Un exemple d'appel des utilitaires est donné après leur description détaillée.

### MATCON

Définition : La matrice X(A,B) est remplie de constantes.

Programme : Fournir la dimension A,B de la matrice et la constante C.
Sortie : La matrice X(A,B) contient la constante C.
Ce programme est appelé par "GOSUB 10000".

***Analyse du programme***

Lignes 10120 à 10160 : Mise à C de la matrice X.

### MATIDN

Définition : La matrice carrée X(A,A) est rendue égale à l'identité.
La diagonale de X est formée de 1, les autres éléments sont à 0.

Programme : Fournir la dimension A de la matrice X.
Sortie : La matrice X est rendue égale à l'identité.
Ce programme est appelé par "GOSUB 11000".

***Analyse du programme***

Ligne 11130 : Si la ligne I = la colonne J, alors IDN(I,J)=1.
Ligne 11140 : Si la ligne I <> la colonne J, alors IDN(I,J)=0.

## MATINPUT

Définition : MATINPUT est utilisé pour attribuer des valeurs, entrées au clavier, aux éléments d'une matrice A*B.

Programme : Fournir la dimension A,B de la matrice.
La matrice X(A,B) est lue.
Ce programme est appelé par "GOSUB 12000".

***Analyse du programme***

Lignes 12110 à 12150 : Boucle de lecture de la matrice.

## MATPRINT

Définition : Affiche les valeurs contenues dans une matrice A*B.

Programme : Fournir la dimension de la matrice A,B et la matrice à afficher.
Ce programme est appelé par "GOSUB 13000".

***Analyse du programme***

Lignes 13110 à 13160 : Affichage de la matrice X de dimension A,B.

## MATREAD

Définition : Permet de lire le contenu de la matrice X en consultant les DATA contenues dans le programme.

Programme : Fournir la dimension A,B de la matrice, et les DATA correspondantes.
Sortie : La matrice X contient les éléments lus en DATA.
Le programme est appelé par "GOSUB 14000".

***Analyse du programme***

Lignes 14110 à 14150 : Lecture de la matrice X stockée en DATA.

## MATTRN

Définition : Donne la transposée d'une matrice de dimension A*B.
Par cette transformation, la 1re ligne de X devient la 1re colonne de Y, la 2e ligne de X devient la 2e colonne de Y, etc.

Programme : Fournir la dimension de la matrice : A,B et la matrice à transposer X.
Sortie : Y = Matrice transposée de X.
Ce programme est appelé par "GOSUB 15000"

***Analyse du programme***

Lignes 15120 à 15160 : Transposition de la matrice.

## MAT+

Définition : Additionne 1 à 1 les éléments de deux matrices de même dimension.

Programme : Fournir la dimension des matrices à additionner : A,B et les matrices à additionner X et Y.
Sortie : Z = X + Y.
Ce programme est appelé par "GOSUB 16000".

***Analyse du programme***

Lignes 16120 à 16160 : Calcul de la matrice Z = X + Y.

## MAT−

Définition : Soustrait 1 à 1 les éléments de deux matrices de même dimension.

Programme : Fournir la dimension des deux matrices à soustraire : A,B et les deux matrices à soustraire X, Y.
Sortie : Z = X – Y.
Ce programme est appelé par "GOSUB 17000".

***Analyse du programme***

Lignes 17120 à 17160 : Calcul de la matrice Z = X – Y.

**MAT***

Définition : Multiplie 1 à 1 les éléments de deux matrices de même dimension.

Programme : Fournir la dimension A,B des matrices à multiplier et les matrices à multiplier.
Sortie : Z = X * Y.
Ce programme est appelé par "GOSUB 18000".

***Analyse du programme***

Lignes 18120 à 18190 : Calcul de la matrice Z = X * Y.

**MATINV**

Définition : Une matrice carrée A*A peut être inversée si son déterminant est non nul. Dans ce cas, le résultat de cette fonction est la matrice inverse.

Programme : Fournir la dimension A de la matrice carrée, et la matrice X à inverser.
Sortie : Déterminant de la matrice et, s'il est non nul, son inverse.
Ce programme est appelé par "GOSUB 19000".

***Analyse du programme***

Lignes 19130 à 19460 : Calcul de la matrice inverse.
Ligne 19320 : Affichage du message "pas d'inverse".

### *Exemple d'utilisation des fonctions MAT XXX*

```
10 REM Operations sur les Matrices
20 :
30 DIM X(10,10),Y(10,10),Z(10,10)
40 :
52 :PRINT
1000 REM Test de la fonction MATCON
1010 :
1020 A=3:B=2:C=52:GOSUB 10000
1030 CLS:PRINT"Test MATCON. C=52":PRINT
1040 FOR I=1 TO A
1050   FOR J=1 TO B
1060     PRINT X(I,J);
1070   NEXT J
1080   PRINT
1090 NEXT I
1100 :
1110 REM Test de la fonction MATIDN
1120 :
1130 A=6:GOSUB 11000
1140 PRINT:PRINT"Test MATIDN.":PRINT
1150 FOR I=1 TO A
1160   FOR J=1 TO A
1170     PRINT X(I,J);
1180   NEXT J
1190   PRINT
1200 NEXT I
1210 :
1220 REM Test de la fonction MATINPUT
1230 :
1240 PRINT:PRINT"Test MATINPUT.":PRINT
1250 A=4:B=3:GOSUB 12000
1260 :
1270 REM Test de la fonction MATPRINT
1280 :
1290 PRINT:PRINT"Test MATPRINT.":PRINT
1300 X(1,1)=1:X(1,2)=2:X(1,3)=3
1310 X(2,1)=4:X(2,2)=5:X(2,3)=6
1320 A=2:B=3:GOSUB 13000
1330 :
1340 REM Test de la fonction MATREAD
1350 :
1360 DATA 1,2,3,4,5,6,7,8
1370 A=2:B=4:GOSUB 14000
1380 PRINT:PRINT"Test MATREAD.":PRINT
1390 FOR I=1 TO A
1400   FOR J=1 TO B
1410     PRINT X(I,J);
```

```
1420    NEXT J
1430    PRINT
1440 NEXT I
1450 :
1460 REM Test de la fonction MATTRN
1470 :
1480 X(1,1)=1:X(1,2)=2:X(1,3)=3
1490 X(2,1)=4:X(2,2)=5:X(2,3)=6
1500 A=2:B=3:GOSUB 15000
1510 PRINT:PRINT"Test MATTRN.":PRINT
1520 FOR I=1 TO B
1530    FOR J=1 TO A
1540       PRINT Y(I,J);
1550    NEXT J
1560    PRINT
1570 NEXT I
1580 :
1590 REM Test de la fonction MAT+
1600 :
1610 X(1,1)=1:X(1,2)=2:X(1,3)=3
1620 X(2,1)=4:X(2,2)=5:X(2,3)=6
1630 Y(1,1)=6:Y(1,2)=5:Y(1,3)=4
1640 Y(2,1)=3:Y(2,2)=2:Y(2,3)=1
1650 A=2:B=3:GOSUB 16000
1660 PRINT:PRINT"Test MAT+":PRINT
1670 FOR I=1 TO A
1680    FOR J=1 TO B
1690       PRINT Z(I,J);
1700    NEXT J
1710    PRINT
1720 NEXT I
1730 :
1740 REM Test de la fonction MAT-
1750 :
1760 X(1,1)=1:X(1,2)=2:X(1,3)=3
1770 X(2,1)=4:X(2,2)=5:X(2,3)=6
1780 Y(1,1)=6:Y(1,2)=5:Y(1,3)=4
1790 Y(2,1)=3:Y(2,2)=2:Y(2,3)=1
1800 A=2:B=3:GOSUB 17000
1810 PRINT:PRINT"Test MAT-":PRINT
1820 FOR I=1 TO A
1830    FOR J=1 TO B
1840       PRINT Z(I,J);
1850    NEXT J
1860    PRINT
1870 NEXT I
1880 :
1890 REM Test de la fonction MAT*
1900 :
```

→

```
1910 X(1,1)=1:X(1,2)=2:X(1,3)=3
1920 X(2,1)=4:X(2,2)=5:X(2,3)=6
1930 Y(1,1)=6:Y(1,2)=5
1940 Y(2,1)=4:Y(2,2)=3
1950 Y(3,1)=2:Y(3,2)=1
1960 A=2:B=3:GOSUB 18000
1970 PRINT:PRINT"Test MAT*":PRINT
1980 FOR I=1 TO A
1990   FOR J=1 TO A
2000     PRINT Z(I,J);
2010   NEXT J
2020   PRINT
2030 NEXT I
2040 :
2050 REM Test de la fonction MATINV
2060 :
2070 X(1,1)=3:X(1,2)=-0.5:X(1,3)=-0.5
2080 X(2,1)=-1:X(2,2)=0:X(2,3)=1
2090 X(3,1)=-1:X(3,2)=0.5:X(3,3)=-0.5
2100 A=3:GOSUB 19000
2110 PRINT:PRINT"Test MATINV":PRINT
2120 FOR I=1 TO A
2130   FOR J=1 TO A
2140     PRINT Y(I,J);
2150   NEXT J
2160   PRINT
2170 NEXT I
2180 :
2190 END
```

***Programme***

```
10000 REM ****************************************
10010 REM *                                      *
10020 REM *        FONCTION    MATCON            *
10030 REM *                                      *
10040 REM ****************************************
10050 REM *                                      *
10060 REM * Entree : A,B Dimension de la matrice*
10070 REM *          C   Constante               *
10080 REM * Sortie : X   Matrice forcee a C      *
10090 REM *                                      *
10100 REM ****************************************
10110 :
10120 FOR I=1 TO A
10130   FOR J=1 TO B
10140     X(I,J)=C
10150   NEXT J
10160 NEXT I
10170 RETURN
10180 :
```

```
11000 REM ****************************************
11010 REM *                                      *
11020 REM *        FONCTION     MATIDN           *
11030 REM *                                      *
11040 REM ****************************************
11050 REM *                                      *
11060 REM * Entree : A Dimension de la matrice   *
11070 REM * Sortie : X Matrice Identite          *
11080 REM *                                      *
11090 REM ****************************************
11100 :
11110 FOR I=1 TO A
11120   FOR J=1 TO A
11130     IF I=J THEN X(I,J)=1
11140     IF I<>J THEN X(I,J)=0
11150   NEXT J
11160 NEXT I
11170 RETURN
11180 :
12000 REM ****************************************
12010 REM *                                      *
12020 REM *        FONCTION     MATINPUT         *
12030 REM *                                      *
12040 REM ****************************************
12050 REM *                                      *
12060 REM * Entree : A,B Dimension de la matrice*
12070 REM * Sortie : X   Matrice lue             *
12080 REM *                                      *
12090 REM ****************************************
12100 :
12110 FOR I=1 TO A
12120   FOR J=1 TO B
12130     PRINT "Ligne"I"Colonne"J;:INPUT X(I,J)
12140   NEXT J
12150 NEXT I
12160 RETURN
12170 :
13000 REM ****************************************
13010 REM *                                      *
13020 REM *        FONCTION     MATPRINT         *
13030 REM *                                      *
13040 REM ****************************************
13050 REM *                                      *
13060 REM * Entree : A,B Dimension de la matrice*
13070 REM * Sortie : X   Matrice affichee        *
13080 REM *                                      *
13090 REM ****************************************
13100 :
13110 FOR I=1 TO A
13120   FOR J=1 TO B
13130     PRINT X(I,J);
13140   NEXT J
13150   PRINT
13160 NEXT I
13170 RETURN
13180 :
```

```
14000 REM ****************************************
14010 REM *                                      *
14020 REM *        FONCTION        MATREAD       *
14030 REM *                                      *
14040 REM ****************************************
14050 REM *                                      *
14060 REM * Entree : A,B Dimension de al matrice*
14070 REM * Sortie : X   Matrice lue             *
14080 REM *                                      *
14090 REM ****************************************
14100 :
14110 FOR I=1 TO A
14120   FOR J=1 TO B
14130     READ X(I,J)
14140   NEXT J
14150 NEXT I
14160 RETURN
14170 :
15000 REM ****************************************
15010 REM *                                      *
15020 REM *        FONCTION     MATTRN           *
15030 REM *                                      *
15040 REM ****************************************
15050 REM *                                      *
15060 REM * Entree : A,B Dimension de la matrice*
15070 REM *          X   Matrice a transposer    *
15080 REM * Sortie : Y   Matrice transposee      *
15090 REM *                                      *
15100 REM ****************************************
15110 :
15120 FOR I=1 TO A
15130   FOR J=1 TO B
15140     Y(J,I)=X(I,J)
15150   NEXT J
15160 NEXT I
15170 RETURN
15180 :
16000 REM ****************************************
16010 REM *                                      *
16020 REM *         FONCTION         MAT+        *
16030 REM *                                      *
16040 REM ****************************************
16050 REM *                                      *
16060 REM * Entree : A,B Dimension des matrices *
16070 REM *          X,Y Matrices a additionner *
16080 REM * Sortie : Z = X + Y                   *
16090 REM *                                      *
16100 REM ****************************************
16110 :
16120 FOR I= 1 TO A
16130   FOR J=1 TO B
16140     Z(I,J)=X(I,J)+Y(I,J)
16150   NEXT J
16160 NEXT I
16170 RETURN
16180 :
```

```
17000 REM ****************************************
17010 REM *                                      *
17020 REM *          FONCTION      MAT-          *
17030 REM *                                      *
17040 REM ****************************************
17050 REM *                                      *
17060 REM * Entree : A,B Dimension des matrices *
17070 REM *          X,Y Matrices a soustraire   *
17080 REM * Sortie : Z = X - Y                   *
17090 REM *                                      *
17100 REM ****************************************
17110 :
17120 FOR I=1 TO A
17130   FOR J=1 TO B
17140     Z(I,J)=X(I,J)-Y(I,J)
17150   NEXT J
17160 NEXT I
17170 RETURN
17180 :
18000 REM ****************************************
18010 REM *                                      *
18020 REM *          FONCTION      MAT*          *
18030 REM *                                      *
18040 REM ****************************************
18050 REM *                                      *
18060 REM * Entree : A,B Dimension des matrices *
18070 REM *          X,Y Matrices a multiplier   *
18080 REM * Sortie : Z = X * Y                   *
18090 REM *                                      *
18100 REM ****************************************
18110 :
18120 FOR I=1 TO A
18130   FOR J=1 TO A
18140     Z(I,J)=0
18150     FOR K=1 TO B
18160       Z(I,J)=Z(I,J)+X(I,K)*Y(K,J)
18170     NEXT K
18180   NEXT J
18190 NEXT I
18200 RETURN
18210 :
19000 REM ****************************************
19010 REM *                                      *
19020 REM *         FONCTION      MATINV         *
19030 REM *                                      *
19040 REM ****************************************
19050 REM *                                      *
19060 REM * Entree : A Dimension matrice         *
19070 REM *          X Matrice a inverser        *
19080 REM * Sortie : Y Matrice inversee           *
19090 REM *          R Determinant               *
19100 REM *                                      *
19110 REM ****************************************
19120 :
19130 FOR I=1 TO A
19140   FOR J=1 TO A
```

→

```
19150      Y(I,J)=X(I,J)
19160    NEXT J
19170  NEXT I
19180  B=A-1
19190  FOR K=1 TO A
19200    P=Y(K,1)
19210    IF P<>0 THEN 19340
19220    FOR I=K+1 TO A
19230      Z(K)=I
19240      IF Y(I,1)=0 THEN 19310
19250      FOR J=1 TO A
19260        R=Y(K,J)
19270        Y(K,J)=Y(I,J)
19280        Y(I,J)=R
19290      NEXT J
19300      GOTO 19200
19310    NEXT I
19320    PRINT"Pas d'inverse !!"
19330    GOTO 19470
19340    FOR J=1 TO B
19350      Y(K,J)=Y(K,J+1)/P
19360    NEXT J
19370    Y(K,A)=1/P
19380    FOR I=1 TO A
19390      IF I=K THEN 19450
19400      R=Y(I,1)
19410      FOR J=1 TO B
19420        Y(I,J)=Y(I,J+1)-R*Y(K,J)
19430      NEXT J
19440      Y(I,A)=-R*Y(K,A)
19450    NEXT I
19460  NEXT K
19470  RETURN
```

## FONCTIONS DEEK, DOKE

### Fonction DEEK

Définition : Donne le contenu de la mémoire sur deux octets (double PEEK).

Programme : Fournir l'adresse du premier octet à connaître A.
Sortie : B = 1er octet, C = 2e octet.
Ce programme est appelé par "GOSUB 10000".

***Exemple***

```
10 A=10 : REM Adresse du DEEK
20 GOSUB 10000
30 PRINT B,C
40 END
```

***Programme***

```
10000 REM ************************************
10001 REM *                                    *
10002 REM *       FONCTION       DEEK          *
10003 REM *                                    *
10004 REM ************************************
10005 REM *                                    *
10006 REM * Entree : A = Adresse du DEEK       *
10007 REM * Sortie : B = 1er Octet             *
10008 REM *            C = 2eme Octet          *
10009 REM *                                    *
10010 REM ************************************
10011 :
10020 B=PEEK(A)
10030 C=PEEK(A+1)
10040 :
10050 RETURN
```

***Analyse du programme***

Ligne 10020 : 1er octet dans B.
Ligne 10030 : 2e octet dans C.

**Fonction DOKE**

Définition : Place deux octets en mémoires consécutives (double POKE).

Programme : Fournir l'adresse du 1er octet et les 2 octets à stocker. Ce programme est appelé par "GOSUB 10000".

***Exemple***

```
10 A=1000:B=56:C=57:REM Adresse du DOKE et DATA
20 GOSUB 10000
30 END
```

***Programme***

```
10000 REM ***********************************
10001 REM *                                   *
10002 REM *       FONCTION       DOKE         *
10003 REM *                                   *
10004 REM ***********************************
10005 REM *                                   *
10006 REM * Entree : A = Adresse du DOKE      *
10007 REM *          B = 1er Octet            *
10008 REM *          C = 2eme Octet           *
10009 REM *                                   *
10010 REM ***********************************
10011 :
10020 POKE A,B
10030 POKE A+1,C
10040 :
10050 RETURN
```

***Analyse du programme***

Ligne 10020 : Stockage du 1er octet.
Ligne 10030 : Stockage du 2e octet.

## FONCTION DUMP : VISUALISATION DU CONTENU DE LA MÉMOIRE

Définition : Fournit le contenu hexadécimal d'une portion de mémoire.

Programme : Fournir le début et la fin de la mémoire dont on veut connaître le contenu.
Le programme liste le contenu de la mémoire, 8 octets/8.
Ce programme est lancé par "RUN".

***Programme***

```
10000 REM ***********************************
10001 REM *                                   *
10002 REM *       DUMP       MEMOIRE          *
10003 REM *                                   *
10004 REM ***********************************
10005 :
10010 CLS:INPUT"<Debut>, <Fin>";D,F:LIGNE=D
10020 A=INT(LIGNE/256):GOSUB 10200:A$=B$
10025 A=LIGNE-A*256:GOSUB 10200:A$=A$+B$+"   "
```

```
10030 FOR I=0 TO 7
10040   A=PEEK(LIGNE+I)
10050   GOSUB 10200 : REM Dec->Hex
10060   A$=A$+B$+" "
10070 NEXT I
10080 A$=A$+" "
10090 FOR I=0 TO 7
10100   A=PEEK(LIGNE+I)
10110   IF A<58 AND A>47 THEN A$=A$+CHR$(A):GOT
O 10125
10115   IF A<91 AND A>64 THEN A$=A$+CHR$(A):GOT
O 10125
10120   A$=A$+"."
10125 NEXT I
10130 PRINT A$
10140 LIGNE =LIGNE+8
10150 IF LIGNE <=F THEN 10020
10160 END
10170 REM ********************************
10200 REM Sous-programme de conversion
10210 REM Decimal->Hexadecimal
10220 :
10230 REM Entree : A en Decimal
10240 REM Sortie : B$ en hexadecimal
10250 :
10260 B=INT(A/16):C=A-B*16
10270 IF B<=9 THEN B$=CHR$(B+48)
10280 IF B>9  THEN B$=CHR$(B+55)
10290 IF C<=9 THEN B$=B$+CHR$(C+48)
10300 IF C>9  THEN B$=B$+CHR$(C+55)
10310 RETURN
```

***Analyse du programme***

Ligne 10010 : Saisie du début et de la fin du DUMP.
Lignes 10020 à 10025 : Initialisations.
Lignes 10030 à 10150 : Calcul des données à afficher.
Lignes 10200 à 10310 : Conversion Décimal → Hexadécimal.

## FONCTION IN

Définition : Permet de savoir si un élément se trouve dans un ensemble.

Programme : Deux programmes sont fournis :
- un pour les chaînes de caractères ;
- un pour les nombres entiers ou réels.

Fournir l'ensemble A$ ou A,
le nombre d'éléments de l'ensemble,
l'élément à rechercher.

Sortie : R=−1 si l'élément se trouve dans l'ensemble, 0 sinon.

### *1er exemple*

```
10 A$(1)="TOTO":A$(2)="AIME"
20 A$(3)="LES":A$(4)="GATEAUX"
30 N=4:REM 4 elements dans A$
40 B$="AIME":REM Chaine recherchee
50 GOSUB 10000
60 PRINT R:REM R=B$ IN A$
70 END
```

### *1er programme*

```
10000 REM ************************************
10001 REM *                                  *
10002 REM * FONCTION IN (VERSION CHAINE)     *
10003 REM *                                  *
10004 REM ************************************
10005 REM *                                  *
10006 REM * Entree : A$=Ensemble de chaines *
10007 REM *          N=Nb d'elements de A$  *
10008 REM *          B$=Chaine a comparer    *
10009 REM * Sortie : R = B$ IN A$            *
10010 REM *                                  *
10011 REM ************************************
10012 :
10020 R=0
10030 FOR I=1 TO N
10040   IF B$=A$(I) THEN R=-1
10050 NEXT I
10060 :
10070 RETURN
```

### *2e exemple*

```
10 A(1)=1:A(2)=12:A(3)=-3
20 N=3:REM 3 Elements dans A
30 B=50:REM Element recherche
40 GOSUB 10000
50 PRINT R:REM R=B IN A
60 END
```

### *2e programme*

```
10000 REM ************************************
10001 REM *                                    *
10002 REM * FONCTION IN (VERSION NUMERIQUE)    *
10003 REM *                                    *
10004 REM ************************************
10005 REM *                                    *
10006 REM * Entree : A =Ensemble de nombres    *
10007 REM *           N=Nb d'elements de A     *
10008 REM *           B =Nombre a comparer     *
10009 REM * Sortie : R = B IN A                *
10010 REM *                                    *
10011 REM ************************************
10012 :
10020 R=0
10030 FOR I=1 TO N
10040   IF B=A(I) THEN R=-1
10050 NEXT I
10060 :
10070 RETURN
```

### *Analyse du programme*

Ligne 10020 : Valeur par défaut du résultat.
Lignes 10030 à 10060 : Fonction IN.

# Chapitre 3

# Le générateur sonore

## QU'EST-CE QU'UN SON ?

Pour répondre à cette question, faisons un parallèle simple : un caillou jeté dans une mare produit une onde qui a pour centre le point d'impact du caillou et qui s'en éloigne à vitesse constante. De même, un corps sonore qui a subi un choc émet un mouvement de vibration, ou d'ondulation.

L'air qui entoure ce corps participe au mouvement, et forme autour de lui des ondes qui s'éloignent du corps à vitesse constante et qui parviennent à l'oreille.

Les sons perceptibles ont une fréquence de vibration comprise entre 16 et 15000 périodes/Sec (ou Hertz). On parlera d'infra-son pour une fréquence inférieure à 16 Hz (Hertz), et d'ultra-son pour une fréquence supérieure à 15000 Hz.

## LE SYNTHÉTISEUR DE SON DE L'AMSTRAD

L'Amstrad possède un synthétiseur de son intégré (référencé AY 3-8912).

Le synthétiseur sonore comporte trois voies indépendantes, entièrement programmables. Ainsi :

- la fréquence,
- la forme de l'onde (carrée, triangulaire, dent de scie ou bruit),
- l'enveloppe (attaque, décroissance, stabilisation, extinction),
- la durée du son,

d'un son quelconque sont programmables.

L'instruction de base pour activer le générateur sonore est l'instruction SOUND C,P,D,V,EV,ET,NP avec :

**C** = N° de canal (1..255).
**P** = Période du signal (0..4095).
**D** = Durée du signal (−32768 ..32767).
**V** = Volume (0..15).
**EV** = Enveloppe de volume (0..15).
**ET** = Enveloppe de ton (0..15).
**NP** = Période de bruit (0..4095).

Pour produire une note n'ayant aucune sonorité particulière, on pourra se contenter de l'ordre SOUND C,P,D; et de l'ordre SOUND 1,P,D si la note utilise un seul canal.

Le programme suivant montre comment activer une note en donnant sa période et sa durée. Ce programme fondamental, quoique très simple, servira par la suite.

***Exemple***

```
10 CLS:PRINT"SON ELEMENTAIRE":PRINT
20 INPUT"Periode";P
30 INPUT"Duree  ";D
40 GOSUB 10000
50 GOTO 10
```

***Programme***

```
10000 REM ****************************************
10001 REM *                                        *
10002 REM *          SON       ELEMENTAIRE         *
10003 REM *                                        *
10004 REM ****************************************
10005 REM *                                        *
10006 REM * Entree : P=Periode de la note          *
10007 REM *            D=Duree   de la note        *
10008 REM *                                        *
10009 REM ****************************************
10010 :
10020 SOUND 1,P,D
10030 :
10040 RETURN
```

***Analyse du programme***

Ligne 10020 : Activation de l'ordre SOUND.

## MÉTRONOME

Cet appareil, bien connu des musiciens, bat la mesure, et aide le débutant à suivre le rythme d'une partition de musique. Un métronome est gradué en battements/minute. Ainsi, s'il est positionné sur 60, il produira 60 battements par minute. L'ordre SOUND possède un argument <Durée>. Aussi, il est très facile de réaliser l'équivalent d'un métronome mécanique à peu de frais.

***Programme***

```
1000 REM ***************************************
*
1010 REM *
*
1020 REM *          M E T R O N O M E
*
1030 REM *
*
1040 REM ***************************************
*
1050 :
1060 CLS:PRINT"                    METRONOME":LOCATE
8,10
1070 INPUT"Tempo (10..300)";T
1080 PRINT
1090 PRINT"Taper sur ENTER pour lancer le Metro
nome"
1100 A$=INKEY$:IF A$="" THEN 1100
1110 :
1120 SOUND 1,2000,10,7
1130 SOUND 1,0,(60/T)*100-10
1140 GOTO 1120
```

***Analyse du programme***

| | |
|---|---|
| Ligne 1070 | : Entrée du Tempo. |
| Lignes 1090 à 1100 | : Activation du métronome. |
| Ligne 1120 | : Production d'un battement. |
| Ligne 1130 | : Silence. |

## GAMME CHROMATIQUE

Pour vous faire une idée de l'étendue des fréquences du synthétiseur sonore, voici un programme qui joue la gamme chromatique sur 8 octaves.

### *Programme*

```
10 REM Gamme sur une voix
11 :
15 CLS:PRINT"Gamme Chromatique":PRINT
20 D=100 'Duree de la note
30 FOR I=0 TO 95
35 IF INT(I/12)=I/12 THEN J=J+1:PRINT"Octave"J
40   READ P
50   GOSUB 10000 'Activation de la note
60 NEXT I
70 :
80 END
90 REM ****************************************
**
100 DATA 3822,3608,3405,3214,3034,2863,2703,255
1,2408,2273,2145,2025
110 DATA 1911,1804,1703,1607,1517,1432,1351,127
6,1204,1136,1073,1012
120 DATA 956,902,851,804,758,716,676,638,602,56
8,536,506
130 DATA 478,451,426,402,379,358,338,319,301,28
4,263,253
140 DATA 239,225,213,201,190,179,169,159,150,14
2,134,127
150 DATA 119,113,106,100,95,89,84,80,75,71,67,6
3
160 DATA 60,56,53,50,47,45,42,40,38,36,34,32
170 DATA 30,28,27,25,24,22,21,20,19,18,17,16
10000 REM ****************************************
**
10001 REM *
 *
10002 REM *         SON       ELEMENTAIRE
 *
10003 REM *
 *
10004 REM ****************************************
**
10005  REM *
  *
10006 REM * Entree : P=Periode de la note
 *
10007 REM *           D=Duree   de la note
 *
10008 REM *
 *
10009 REM ****************************************
**
10010 :
10020 SOUND 1,P,D
10030 :
10040 RETURN
```

### *Analyse du programme*

| | |
|---|---|
| Lignes 10 à 20 | : Initialisation. |
| Lignes 35 à 60 | : Activation d'une note. |
| Lignes 100 à 170 | : DATA correspondant aux fréquences des notes à jouer. |
| Lignes 10000 à 10040 | : Ordre SOUND. |

## CPC-PIANO

Vous jouez du piano ? Non ! Alors il est temps de vous y mettre. Ce programme simule le clavier d'un piano. Les touches du clavier reprennent la disposition des tons et 1/2 tons d'un clavier de piano.

Les touches 1 à 8 sélectionnent les octaves, et la touche 0 arrête le programme. Musiciens, à vos claviers...

### *Programme*

```
10 REM CPC-piano
11 :
20 GOSUB 1000 'Initialisation
30 GOSUB 2000 'Piano
40 :
50 END
60 REM ************************************
*
1000 REM Initialisation des notes
1010 :
1020 DIM T(96)
1030 FOR I=1 TO 96
1040   READ T(I)
1050 NEXT I
1060 :
1070 DATA 3822,3608,3405,3214,3034,2863,2703,25
51,2408,2273,2145,2025
1080 DATA 1911,1804,1703,1607,1517,1432,1351,12
76,1204,1136,1073,1012
1090 DATA 956,902,851,804,758,716,676,638,602,5
68,536,506
1100 DATA 478,451,426,402,379,358,338,319,301,2
84,268,253
1110 DATA 239,225,213,201,190,179,169,159,150,1
42,134,127
1120 DATA 119,113,106,100,95,89,84,80,75,71,67,
63
1130 DATA 60,56,53,50,47,45,42,40,38,36,34,32  →
```

```
1140 DATA 30,28,27,25,24,22,21,20,19,18,17,16
1150 :
1160 DIM G(26)
1170 FOR I=1 TO 26
1180   READ G(I)
1190 NEXT I
1200 :
1210 DATA 1,0,0,5,4,6,8,10,0,12,13,0,0,0,0,0,0,
0,3,7,11,0,2,0,9,0
1220 :
1230 RETURN
1240 REM ************************************
*
2000 REM CPC-piano
2010 :
2011 CLS:PRINT"CPC-piano..."
2020 A$=INKEY$
2030 IF A$="" THEN 2020 ' Boucle d'attente
2040 :
2050 A=ASC(A$)
2055 IF A=48 THEN 2140
2060 IF A>56 THEN 2080
2070 OCT=A-49 'Choix de l'octave
2080 IF A>90 OR A<65 THEN 2020 'Touche inconnue
2090 :
2100 IF G(A-64)<>0 THEN SOUND 1,T(G(A-64)+OCT*1
2),20,5
2110 :
2120 GOTO 2020
2130 :
2140 RETURN
```

***Analyse du programme***

Lignes 10 à 50 : Programme principal.
Lignes 1000 à 1230 : Lecture des DATA correspondant aux notes possibles.
Lignes 2000 à 2140 : Saisie clavier et activation de l'ordre SOUND.

## ÉDITEUR MUSICAL

Si vous possédez des partitions de musique, ou si vous êtes à même d'en écrire, ce programme peut s'avérer très intéressant. Après quelques efforts pour entrer toutes les notes sur une, deux ou trois voix, vous pouvez écouter le morceau ainsi reconstitué. Il a même été prévu une commande pour immortaliser vos œuvres sur fichier séquentiel.

***Programme***

```
10 REM ****************************************
*
20 REM *
*
30 REM *    E D I T E U R      M U S I C A L
*
40 REM *
*
50 REM ****************************************
*
60 :
70 GOSUB 500 'Initialisation
80 GOSUB 1000 'Menu principal
90 :
100 END
110 REM ------------------------------------
500 REM Initialisaton
510 :
520 DIM NN(300,3),DU(300,3),T1(96)
530 :
540 FOR I=1 TO 96
550   READ T1(I)
560 NEXT I
570 DATA 3822,3608,3405,3214,3034,2863,2703,255
1,2408,2273,2145,2025
580 DATA 1911,1804,1703,1607,1517,1432,1351,127
6,1204,1136,1073,1012
590 DATA 956,902,851,804,758,716,676,638,602,56
8,536,506
600 DATA 478,451,426,402,379,358,338,319,301,28
4,268,253
610 DATA 239,225,213,201,190,179,169,159,150,14
2,134,127
620 DATA 119,113,106,100,95,89,84,80,75,71,67,6
3
630 DATA 60,56,53,50,47,45,42,40,38,36,34,32
640 DATA 30,28,27,25,24,22,21,20,19,18,17,16
650 :
660 RETURN
1000 REM Menu principal
1010 :
1015 CLS:PRINT"EDITEUR MUSICAL":PRINT
1020 PRINT"1)Modification du volume"
1030 PRINT"2)Ecriture du morceau"
1040 PRINT"3)Execution du morceau"
1045 PRINT"4)Sauvegarde K7"
1050 PRINT:INPUT"Votre choix";C
1060 IF C>4 OR C<1 THEN 1000
```

→

```
1070 :
1080 ON C GOSUB 10000,11000,12000,13000
1090 :
1100 GOTO 1000
1110 :
1120 REM ------------------------------
10000 REM Modification du volume sonore
10001 :
10010 CLS:PRINT"Modification du volume sonore":
PRINT
10020 INPUT"Volume (0..7)";V
10030 :
10040 RETURN
10050 REM ------------------------------
11000 REM Ecriture du morceau
11010 :
11020 CLS:PRINT"Eciture du morceau":PRINT
11030 PRINT"Notes : de DO1 a SI8":PRINT
11040 PRINT"Duree : 1-Croche  2-Noire"
11050 PRINT"          3-Blanche 4-Ronde"
11060 PRINT:INPUT"Voie (1,2,3 ou 4-Fin)";VX
11070 IF VX>4 OR VX<1 THEN PRINT CHR$(7):GOTO 1
1060
11080 IF VX=4 THEN RETURN
11090 :
11100 PRINT:INPUT"Note ";N$
11110 GOSUB 20000 'Decodage
11120 IF B=1 THEN PRINT CHR$(7):GOTO 11100
11130 :
11140 PRINT:INPUT"Duree";D
11150 IF D>4 OR D<1 THEN PRINT CHR$(7):GOTO 111
40
11160 :
11170 PRINT:INPUT"Temps";T
11180 IF T>300 OR T<1 THEN PRINT CHR$(7):GOTO 1
1170
11190 :
11200 NN(T,VX)=NO:DU(T,VX)=D*25 'Memorisation

11210 :
11220 GOTO 11000
11230 :
11240 REM ------------------------------
12000 REM Execution du morceau
12010 :
12020 I=0
12030 I=I+1
12040 IF NN(I,1)=0 AND NN(I,2)=0 AND NN(I,3)=0
THEN 12130
```

```
12050 IF DU(I,1)<>0 THEN SOUND 1,NN(I,1),DU(I,1
),V
12060 IF DU(I,2)<>0 THEN SOUND 2,NN(I,2),DU(I,2
),V
12070 IF DU(I,3)<>0 THEN SOUND 4,NN(I,3),DU(I,3
),V
12080 :
12090 SOUND 1,0,20:SOUND 2,0,20:SOUND 4,0,20
12100 :
12110 GOTO 12030
12120 :
12130 RETURN
12140 REM ----------------------------
13000 REM Sauvegarde Cassette
13010 :
13020 CLS:PRINT"Sauvegarde K7":PRINT
13030 INPUT"Nom de la sauvegarde";N$
13040 :
13050 I=0
13060 I=I+1
13070 IF NN(I,1)<>0 OR NN(I,2)<>0 OR NN(I,3)<>0
 THEN 13060
13080 :
13090 OPENOUT N$
13100 PRINT #9,I 'Nombre de donnees
13110 FOR J=1 TO I
13120   PRINT #9,NN(J,1)
13125   PRINT #9,DU(J,1)
13130   PRINT #9,NN(J,2)
13135   PRINT #9,DU(J,2)
13140   PRINT #9,NN(J,3)
13145   PRINT #9,DU(J,3)
13150 NEXT J
13160 CLOSEOUT
13170 :
13180 RETURN
20000 REM Decodage de la note tapee
20010 :
20015 B=0:C=0 'Initialisation
20016 :
20020 L$=LEFT$(N$,2) 'Note
20030 R=ASC(RIGHT$(N$,1))-48 'Octave
20040 IF R<1 OR R>8 THEN B=1:GOTO 20170
20050 :
20060 IF L$="DO" THEN C=1
20070 IF L$="RE" THEN C=3
20080 IF L$="MI" THEN C=5
20090 IF L$="FA" THEN C=6
20100 IF L$="SO" THEN C=8
```

→

```
20110 IF L$="LA" THEN C=10
20120 IF L$="SI" THEN C=12
20130 IF C=0 THEN B=1:GOTO 20170
20135 FOR I=1 TO LEN(N$)
20136    IF MID$(N$,I,1)="#" THEN C=C+1
20137 NEXT I
20140 :
20150 D=C+(R-1)*12:ND=T1(D) 'Note
20160 :
20170 RETURN
```

***Analyse du programme***

| | |
|---|---|
| Lignes 10 à 100 | : Programme principal. |
| Lignes 500 à 660 | : Initialisation du programme. |
| Lignes 1000 à 1110 | : Menu principal. |
| Lignes 10000 à 10040 | : Modification du volume sonore. |
| Lignes 11000 à 11230 | : Ecriture du morceau. |
| Lignes 12000 à 12130 | : Exécution du morceau. |
| Lignes 13000 à 13180 | : Sauvegarde cassette (CPC 464) ou disquette (CPC 664). |
| Lignes 20000 à 20170 | : Décodage de la note demandée. |

**Remarque :** Avant de demander l'exécution du morceau, il est nécessaire d'initialiser le volume sonore...

## LECTURE DE FICHIERS MUSICAUX

Ce programme lit les morceaux stockés sur cassette et les exécute autant de fois que vous le désirez jusqu'à l'appui sur la touche BREAK.

***Programme***

```
1000 REM ************************************
*
1010 REM *
*
1020 REM *     LECTURE DE FICHIERS MUSICAUX
*
1030 REM *
*
1040 REM ************************************
*
1050 :
1060 CLS:PRINT"      LECTURE DE FICHIERS MUSICAU
X":LOCATE 1,10
1070 INPUT"Nom du fichier";N$
```

```
1080 :
1090 OPENIN N$
1100 INPUT #9,NB 'Nombre de donnees
1105 DIM NN(NB,3),DU(NB,3)
1110 :
1120 FOR I=1 TO NB
1130   INPUT #9,NN(I,1)
1140   INPUT #9,DU(I,1)
1150   INPUT #9,NN(I,2)
1160   INPUT #9,DU(I,2)
1170   INPUT #9,NN(I,3)
1180   INPUT #9,DU(I,3)
1190 NEXT I
1195 CLOSEIN
1200 :
1210 CLS:PRINT"Appuyez sur une touche pour lanc
er"
1220 PRINT"le morceau."
1230 :
1240 A$=INKEY$:IF A$="" THEN 1240
1250 :
1260 REM Execution du morceau
1270 :
1280 I=0
1290 I=I+1
1300 IF NN(I,1)=0 AND NN(I,2)=0 AND NN(I,3)=0 T
HEN 1380 ' Fin du morceau
1310 IF DU(I,1)<>0 THEN SOUND 1,NN(I,1),DU(I,1)
1320 IF DU(I,2)<>0 THEN SOUND 2,NN(I,2),DU(I,2)
1330 IF DU(I,3)<>0 THEN SOUND 4,NN(I,3),DU(I,3)
1340 SOUND 1,0,20:SOUND 2,0,20:SOUND 4,0,20 'Si
lence
1350 :
1360 GOTO 1290 ' Prochaine note
1370 :
1380 GOTO 1210 ' Fin du morceau
```

***Analyse du programme***

Lignes 1000 à 1070 : Initialisation.
Lignes 1090 à 1195 : Lecture du fichier musical.
Lignes 1260 à 1380 : Exécution du morceau.

## MORCEAUX SUR UNE ET DEUX VOIX

La technique consiste à mettre les notes à jouer et les durées associées en DATA. Dans la suite, nous avons pris une durée de 40 pour une noire, 80 pour une blanche et 20 pour une croche.

## Morceaux sur une voix

La première DATA donne le nombre de paires Note/Durée à lire. Une boucle de lecture Note/Durée active l'ordre SOUND 1,NO,DU.

### *Programme*

```
10000 REM ************************************
**
10010 REM *
 *
10020 REM *      MORCEAU   SUR   UNE   VOIX
 *
10030 REM *
 *
10040 REM ************************************
**
10050 :
10060 READ NB 'Nombre de donnees
10070 FOR I=1 TO NB
10080   READ NO,DU
10090   IF NO<>0 THEN SOUND 1,NO,DU
10100   SOUND 1,0,1 'Silence
10110 NEXT I
10120 :
10130 DATA 14,119,40,106,40,95,40,119,40,119,40
,106,40,95,40,119,40
10140 DATA 95,40,89,40,80,80,95,40,89,40,80,80
10150 :
10160 END
```

### *Analyse du programme*

Ligne 10060 : Lecture du nombre de notes.
Lignes 10070 à 10110 : Boucle de lecture.
Lignes 10130 à 10140 : Données Note/Durée.

## Morceaux sur deux voix

La technique employée est la même que pour "Morceaux sur une voix". Cependant, les données contiennent en plus l'information CANAL (CA) qui va activer l'ordre SOUND CA,NO,DU.

***Programme***

```
10000 REM ************************************
**
10010 REM *
 *
10020 REM *      MORCEAU   SUR   DEUX   VOIX
   *
10030 REM *
 *
10040 REM ************************************
**
10050 :
10060 READ NB 'Nombre de donnees
10070 FOR I=1 TO NB
10080   READ CA,NN,DU
10090   SOUND CA,NN,DU
10100   SOUND 1,0,1:SOUND 2,0,1 'Silence
10110 NEXT I
10120 :
10130 DATA 22,1,119,40,1,106,40,1,95,40,1,119,4
0
10140 DATA 1,119,40,1,106,40,1,95,40,1,119,40
10150 DATA 17,95,40,10,119,40,1,89,40,2,106,40,
1,80,80,2,95,40,2,119,40
10160 DATA 1,95,40,2,119,40,1,89,40,2,106,40,1,
80,80,2,95,40,2,119,40
```

***Analyse du programme***

Ligne 10060 : Lecture du nombre de notes.
Lignes 10070 à 10110 : Boucle de lecture.
Lignes 10130 à 10160 : Données Canal/Note/Durée.

## PROGRAMMATION DE MORCEAUX DE MUSIQUE

Ce programme reprend les bases données dans "Morceaux sur une voix". Trois airs célèbres sont stockés en DATA sous la forme :

<Nb de DATA>,<Durée>,<Période>,

***Programme***

```
10 REM Programmation de morceaux
11 :
20 CLS:PRINT"PROGRAMMATION DE MORCEAUX DE MUSIQ
UE"
30 PRINT:PRINT"Voulez-vous entendre :"
40 PRINT:PRINT"1)Histoires sans paroles"
50 PRINT"2)La truite de Schubert"
60 PRINT"3)Classique"
70 PRINT:INPUT"Entrez votre choix (1, 2 ou 3)";
C
80 IF C>3 OR C<1 THEN 10
90 :
100 RESTORE
110 IF C=1 THEN X=0 'ier morceau
120 IF C=2 THEN X=139 '2eme morceau
130 IF C=3 THEN X=292 '3eme morceau
140 :
150 IF X<>0 THEN FOR I=1 TO X:READ A:NEXT I

160 :
170 READ NB 'Nombre de notes
180 FOR I=1 TO NB
190   READ D,P 'Lecture Duree et Periode
195 IF C=2 THEN D=D*20 ELSE D=D*14
200   GOSUB 10000 'Activation Note
205   P=0:D=0.6:GOSUB 10000
210 NEXT I
220 :
230 GOTO 10
240 :
250 REM ****************************************
300 REM Pop Corn
310 DATA 69,1,47,1,60,1,80,1,95,1,119,1,95,1,80
,1,95,1,80,2,71,1,95,3,80
320 DATA 1,47,1,60,1,80,1,95,1,119,1,95,1,80,1,
95,1,80,1,60,1,63,1,53,3,60
330 DATA 1,47,2,40,1,47,2,60,1,47,1,60,3,80,3,7
1,1,47,2,40,1,47,2,60,1,47,1,60
340 DATA 1,80,1,60,1,60,1,63,3,60
350 DATA 1,47,1,60,1,80,1,95,1,119,1,95,1,80,1,
95,1,80,2,71,1,95,3,80
360 DATA 1,47,1,60,1,80,1,95,1,119,1,95,1,80,1,
95,1,80,1,60,1,63,1,53,3,60
370 REM ----------------------------------
380 REM La truite de Schubert
390 DATA 76,1,119,1,89,1,89,1,71,1,71,2,89,2,11
9,1,119
```

```
395 DATA 1,80,1,89,1,95,1,106,3,119,1,119,1,80,
1,89,1,95,1,106,3,119
400 DATA 1,119,1,89,1,89,1,71,1,71,2,89,2,119
410 DATA 1,89,1,95,1,106,1,95,1,89,1,127,3,119,
1,119,1,95,1,95,1,89,1,95
420 DATA 1,106,1,95,2,89,2,119,1,89,1,95,1,95,1
,89,1,95,1,106,1,95,3,89
430 DATA 1,89,1,106,1,106,1,89,1,89,1,89,2,119,
1,119,1,119,1,95,1,80,1,95,3,89
440 DATA 1,89,1,106,1,106,1,89,1,89,1,89,2,119,
1,119,1,119,1,95,1,80,1,95,3,89
450 REM ------------------------------
460 REM Classique
470 DATA 73,2,106,1,80,1,84,2,80,2,106,2,71,2,1
06,2,67,2,80,2,60,2,80
475 DATA 2,53,2,80,2,50,2,80
480 DATA 1,84,1,80,1,71,1,84,2,106,1,53,1,60,1,
63,1,60,1,53,1,63
490 DATA 1,80,1,71,1,63,1,80,2,89,1,100,1,106,2
,100,1,60,1,67,1,71,1,67,1,60
500 DATA 1,71,1,89,1,80,1,71,1,89,2,100,1,106,1
,119,2,106,1,67,1,71,1,80
510 DATA 1,71,1,67,1,80,1,100,1,89,1,80,1,100,2
,106,1,119,1,134,2,119,2,100
520 DATA 2,106,2,80,1,84,1,80,1,71,1,84,2,84,1,
80,1,84,3,80
10000 REM ************************************
**
10001 REM *
 *
10002 REM *         SON       ELEMENTAIRE
 *
10003 REM *
 *
10004 REM ************************************
**
10005  REM *
  *
10006 REM * Entree : P=Periode de la note
 *
10007 REM *           D=Duree   de la note
 *
10008 REM *
 *
10009 REM ************************************
**
10010 :
10020 SOUND 1,P,D
10030 :
10040 RETURN
```

***Analyse du programme***

| | |
|---|---|
| Lignes 10 à 80 | : Présentation. |
| Lignes 100 à 150 | : Initialisation en fonction du morceau choisi. |
| Ligne 170 | : Lecture du nombre de notes. |
| Lignes 180 à 210 | : Boucle de lecture. |
| Lignes 300 à 520 | : DATA Durée/Période. |
| Lignes 10000 à 10040 | : SOUND. |

## EFFETS SONORES, BRUITS D'ANIMATION SPÉCIFIQUES

Voici quelques programmes, généralement très courts, qui simulent divers bruits et que vous pourrez incorporer très simplement dans vos programmes.

### Cheval au galop

Ce programme utilise le paramètre "Période de bruit" de la commande SOUND pour reproduire le bruit que fait un cheval au galop. Si le son produit ne vous convient pas, modifiez la valeur de la "Période de bruit" en lignes 10, 30 et 40, qui est fixée à 25 dans le programme.

**Remarque :** Les paramètres "Enveloppe de volume" et "Enveloppe de ton" sont présents mais sans effet, puisqu'aucune commande ENV ou ENT n'a été faite au préalable.

```
5  REM Cheval au galop
6  :
10 SOUND 1,600,10,5,1,1,25
20 FOR I=1 TO 120:NEXT I
30 SOUND 1,600,5,5,1,1,25
35 FOR I=1 TO 80:NEXT I
40 SOUND 1,600,10,5,1,1,25
45 FOR I=1 TO 220:NEXT I
50 GOTO 10
```

### Loco

Pour les amateurs de circuits ferroviaires, voici un programme qui reproduit le son d'une locomotive à vapeur. Ici encore, le paramètre "Période de

bruit" de la commande SOUND est utilisé, mais avec une période de ton et une durée différentes de celles du programme "Cheval au galop".

```
10 REM Loco
11 :
20 J=500
30 SOUND 1,400,20,5,1,1,25
40 IF J>250 THEN J=J-30
50 FOR I=1 TO J:NEXT I
60 GOTO 30
```

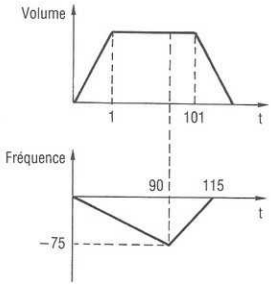
## Passage de bolide

Dans ce programme, nous utilisons les commandes de définition d'enveloppe de volume (ENV) et d'enveloppe de ton (ENT).

Comme son nom l'indique, l'enveloppe de volume va moduler la puissance sonore parvenant sur le haut-parleur en fonction du temps.

Comme son nom ne l'indique pas forcément, l'enveloppe de ton va moduler la fréquence de la note dans le temps.

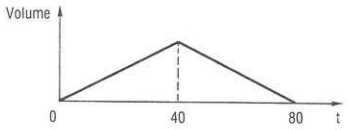
Pour le passage d'un bolide, le son croît, se stablise puis décroît ; la fréquence décroît lentement, puis croît rapidement, ce qui correspond aux schémas suivants :




```
10 REM Passage de bolide
11 :
20 ENV 1,1,15,1,1,0,100,15,-1,8
30 ENT 1,15,-5,6,5,15,5
40 SOUND 1,1200,-1,0,1,1
```

### Friture sur la ligne

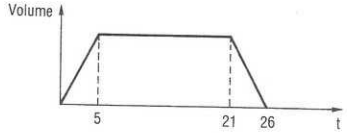
Ce programme reproduit les parasites que l'on entend (parfois !) sur les lignes téléphoniques. Pour ce faire, une enveloppe de volume est définie en forme de triangle de courte durée.




```
10 REM Friture
11 :
20 ENV 1,40,3,1,40,-3,1
30 SOUND 1,4095,-100,0,1
```

### Alarme sur le navire

Ici encore, l'enveloppe de volume est redéfinie pour reproduire le son qui précède l'immersion des sous-marins en cas de danger.




```
10 REM Alarme sur le navire
11 :
20 ENV 1,5,3,1,1,0,16,5,-3,2
30 SOUND 1,300,-10,0,1
```

### Boucles sur SOUND

Ce programme montre comment produire des sons très différents en faisant varier rapidement la période de ton de l'ordre SOUND. Quatre effets sonores différents s'enchaînent automatiquement.

```
10 REM Boucles sur SOUND
11 :
20 CLS:PRINT"Vaisseau au depart"
30 J=0 'RAZ index de boucle
40 FOR I=1 TO 800 STEP 10
50   SOUND 1,I,1,5
60 NEXT I
70 J=J+1:IF J<10 THEN 40
80 :
90 CLS:PRINT"zap..."
100 J=0 'RAZ index de boucle
110 FOR I=1 TO 100
120   SOUND 1,I,1,5
130 NEXT I
140 J=J+1:IF J<4 THEN 110
150 :
160 CLS:PRINT"Bonus jeu d'arcade"
170 J=0 'RAZ index de boucle
180 FOR I=1 TO 100 STEP 10
190   SOUND 1,I,1,5
200 NEXT I
210 J=J+1:IF J<20 THEN 180
220 :
230 CLS:PRINT"Avion en pique"
240 FOR I=1000 TO 2000 STEP 2
250   SOUND 1,I,1,5
260 NEXT I
```

# Chapitre 4

# Les modes graphiques haute résolution

Les micro-ordinateurs CPC 464 et CPC 664 présentent un grand avantage par rapport aux autres micro-ordinateurs familiaux. Cet avantage se situe au niveau du graphisme haute résolution.

Si la plupart des micro-ordinateurs font la différence entre écran graphique et écran texte, Amstrad a su concilier les deux. Ainsi, il est très facile de mélanger texte et graphiques haute résolution sur le même écran.

Trois modes (haute résolution et texte) sont possibles :

- MODE 0 : 160 × 200 pixels, 20 colonnes × 25 lignes, 16 couleurs.
- MODE 1 : 320 × 200 pixels, 40 colonnes × 25 lignes, 4 couleurs.
- MODE 2 : 640 × 200 pixels, 80 colonnes × 25 lignes, 2 couleurs.

## ALLUMAGE D'UN POINT SUR L'ÉCRAN

L'instruction PLOT X,Y,E
avec

X = coordonnée en X du point à allumer (0..639),
Y = coordonnée en Y du point à allumer (0..399),

et

E = couleur d'encre,

permet d'allumer un point élémentaire (ou pixel) sur l'écran.

Nous allons mettre en œuvre l'allumage d'un pixel sur deux applications décrites ci-après :

- tracé de courbes en haute résolution ;
- figures de Moivre.

## Tracé de courbes en haute résolution

Tracé de courbes Y=F(x)

Le lycéen de second cycle est souvent confronté au problème suivant : étudier puis représenter l'évolution d'une courbe monovariable entre deux bornes fixes.

Le programme décrit ici permet de donner le tracé d'une courbe monovariable quelconque de type Y = F(x) entre deux bornes fixées par l'utilisateur.

Si ce programme ne réduit pas à néant l'étude préalable de la fonction (en particulier l'étude des points singuliers), il permet cependant de confirmer ou d'infirmer l'allure générale de la courbe à étudier.

Le programme est très simple. Il se décompose en trois phases :

- saisie de l'équation,
- saisie du domaine de définition,
- passage en mode haute résolution et tracé.

### *Programme*

```
10 REM ****************************************
11 REM *                                      *
12 REM * TRACE DE COURBES D'EQUATION Y=F(X)   *
13 REM *                                      *
14 REM ****************************************
15 :
20 GOSUB 1000 : REM Saisie de l'equation
30 STOP
40 GOSUB 2000 : REM Saisie du domaine
50 GOSUB 3000 : REM Trace
60 :
70 END
80 REM ****************************************
1000 REM Saisie de l'equation
1010 :
1020 CLS
1030 PRINT"Tapez 2060 DEF FNA(X)="
1040 PRINT"suivi de l'equation a etudier."
1050 PRINT:PRINT"Tapez ensuite RUN 40"
1060 :
1070 RETURN
1080 REM ****************************************
2000 REM Saisie du domaine de definition
2010 :
2020 CLS
2030 PRINT"Entrez le domaine d'etude :"
2040 PRINT:INPUT"X min";X1
2050 PRINT:INPUT"X max";X2
```

```
2060 DEF FNA(X)=SIN(X)
2070 :
2080 RETURN
2090 REM ****************************************
3000 REM Trace de la courbe
3010 :
3020 REM Calcul de l'echelle en Y
3030 :
3035 M1=-1E+33:M2=1E+33
3040 FOR X=X1 TO X2 STEP (X2-X1)/100
3050   A=FNA(X)
3060   IF A>M1 THEN M1=A
3070   IF A<M2 THEN M2=A
3080 NEXT X
3090 EX=640/(X2-X1):EY=399/(M1-M2)
3100 PX=(X2-X1)/100
3110 :
3120 REM Trace
3130 :
3140 CLG:REM Effacement d'ecran
3150 FOR X=X1 TO X2 STEP PX
3160   PLOT (X-X1)*EX,(FNA(X)-M2)*EY
3170 NEXT X
3180 :
3190 RETURN
```

### *Analyse du programme*

Lignes 20 à 70 : Programme principal.
Lignes 1000 à 1070 : Saisie de l'équation.
Lignes 2000 à 2080 : Saisie du domaine d'étude.
Ligne 2060 : Définition de la fonction à étudier.
Lignes 3000 à 3100 : Calcul d'échelle.
Lignes 3120 à 3190 : Tracé de la courbe.

Tracé de courbes Y(t), X(t)

Ce programme peut être employé de deux façons différentes :

- Comme aide mathématique (représentation instantanée d'une courbe par l'entrée de ses équations).
- Comme générateur de motifs mathématiques graphiques.

La première voie, si elle n'évite pas à l'étudiant de fastidieuses recherches de tangentes, points d'inflexion et assymptote nécessaires à tout "bon" tracé de courbes, permet cependant de concrétiser rapidement une impression plus ou moins juste sur l'allure de la courbe à étudier.

La deuxième voie permet d'obtenir d'assez jolis dessins par la seule entrée d'équations et du domaine d'étude.

***Programme***

```
10 REM ****************************************
*
11 REM *
*
12 REM * TRACE DE COURBES D'EQUATION Y(t),X(t)
*
13 REM *
*
14 REM ****************************************
*
15 :
20 GOSUB 1000 : REM Saisie des equations
30 STOP
40 GOSUB 2000 : REM Saisie du domaine
50 GOSUB 3000 : REM Trace
60 :
70 END
80 REM ***************************************
1000 REM Saisie des equations
1010 :
1020 CLS
1030 PRINT"Tapez 2060 DEF FNA(X)="
1031 PRINT"et     2061 DEF FNB(X)="
1040 PRINT"suivi des equations X(t),Y(t) a etud
ier"
1050 PRINT:PRINT"Tapez ensuite RUN 40"
1060 :
1070 RETURN
1080 REM ***************************************
*
```

```
2000 REM Saisie du domaine de definition
2010 :
2020 CLS
2030 PRINT"Entrez le domaine d'etude :"
2040 PRINT:INPUT"T min";T1
2050 PRINT:INPUT"T max";T2
2060 DEF FNA(X)=SIN(X)
2061 DEF FNB(X)=COS(X)
2070 :
2080 RETURN
2090 REM ***************************************
*
3000 REM Trace de la courbe
3010 :
3020 TPAS=(T2-T1)/320 'Pas en abcisse
3030 :
3040 REM Pas en X et en Y
3050 :
3060 X1=1E+33:X2=-1E+33:Y1=1E+33:Y2=-1E+33
3070 :
3080 FOR T=T1 TO T2 STEP TPAS
3090   A=FNA(T):B=FNB(T)
3100   IF A<X1 THEN X1=A:X3=T
3110   IF A>X2 THEN X2=A:X4=T
3120   IF B<Y1 THEN Y1=B:Y3=T
3130   IF B>Y2 THEN Y2=B:Y4=T
3140 NEXT T
3150 :
3160 XPAS=(X1-X2)/640:YPAS=(Y1-Y2)/399
3170 :
3180 REM Trace
3190 :
3200 CLG:REM Effacment d'ecran
3210 FOR T=T1 TO T2 STEP TPAS
3220   X=FNA(T):Y=FNB(T)
3230   PLOT INT((1/XPAS)*(X1-X)),INT((1/YPAS)*(
Y-Y2))
3240 NEXT T
3250 :
3260 RETURN
```

***Analyse du programme***

Ce programme est très similaire au précédent.

Lignes 20 à 70 : Programme principal.
Lignes 1000 à 1070 : Saisie des équations.
Lignes 2000 à 2080 : Saisie du domaine d'étude.
Lignes 2060 et 2061 : Définition de la fonction à étudier.

Lignes 3000 à 3160 : Calcul d'échelle.
Lignes 3180 à 3260 : Tracé de la courbe.

Tracé de courbes Y(t), X(t) imbriquées

L'intérêt mathématique de ce programme est très limité. Par contre, l'intérêt graphique est certain. Il consiste à imbriquer N courbes Y(t), X(t) de même équation, ce qui a pour effet de donner une "profondeur" à la représentation.

***Programme***

```
10 REM ****************************************
*
11 REM *
*
12 REM * TRACE DE COURBES D'EQUATION Y(t),X(t)
*
13 REM *
*
14 REM ****************************************
*
15 :
20 GOSUB 1000 : REM Saisie des equations
30 STOP
40 GOSUB 2000 : REM Saisie du domaine
50 GOSUB 3000 : REM Trace
60 :
70 END
80 REM ***************************************
1000 REM Saisie des equations
1010 :
1020 CLS
1030 PRINT"Tapez 2060 DEF FNA(X)="
1031 PRINT"et     2061 DEF FNB(X)="
1040 PRINT"suivi des equations X(t),Y(t) a etud
ier"
1050 PRINT:PRINT"Tapez ensuite RUN 40"
1060 :
1070 RETURN
1080 REM ***************************************
*
2000 REM Saisie du domaine de definition
2010 :
2020 CLS
2030 PRINT"Entrez le domaine d'etude :"
```

```
2040 PRINT:INPUT"T min";T1
2050 PRINT:INPUT"T max";T2
2060 DEF FNA(X)=SIN(X)
2061 DEF FNB(X)=COS(X)
2065 PRINT:INPUT"Nombre de courbes imbriquees";
NC
2066 PRINT:INPUT"Nombre de points par courbe ";
NP
2070 :
2080 RETURN
2090 REM ***************************************
*
3000 REM Trace de la courbe
3010 :
3020 TPAS=(T2-T1)/NP 'Pas en abcisse
3030 :
3040 REM Pas en X et en Y
3050 :
3060 X1=1E+33:X2=-1E+33:Y1=1E+33:Y2=-1E+33
3070 :
3080 FOR T=T1 TO T2 STEP TPAS
3090   A=FNA(T)*NC:B=FNB(T)*NC
3100   IF A<X1 THEN X1=A:X3=T
3110   IF A>X2 THEN X2=A:X4=T
3120   IF B<Y1 THEN Y1=B:Y3=T
3130   IF B>Y2 THEN Y2=B:Y4=T
3140 NEXT T
3150 :
3160 XPAS=(X1-X2)/640:YPAS=(Y1-Y2)/399
3170 :
3180 REM Trace
3190 :
3200 CLG:REM Effacement d'ecran
3205 FOR K=1 TO NC
3210   FOR T=T1 TO T2 STEP TPAS
3220     X=FNA(T)*K:Y=FNB(T)*K
3230     PLOT INT((1/XPAS)*(X1-X)),INT((1/YPAS)
*(Y-Y2))
3240   NEXT T
3245 NEXT K
3250 :
3260 RETURN
```

***Analyse du programme***

Même découpage en modules que le programme précédent.

Remarquez l'imbrication des courbes lignes 3200 à 3245 où K indique le numéro de la courbe tracée.

## Figures de Moivre

Les figures obtenues par le glissement d'une droite sur deux courbes planes portent le nom de "figures de MOIVRE".

De très belles figures peuvent être obtenues sur les micro-ordinateurs graphiques, à moindre effort.

Les programmes proposés ici se servent de l'ordre graphique DRAW.

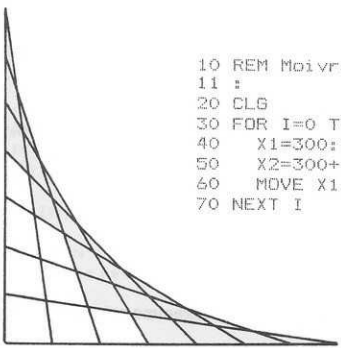
Sept figures sont proposées :

- MOIVRE 1 : les deux courbes planes sont deux droites d'angle 90°.
- MOIVRE 2 : les deux courbes planes sont deux droites d'angle < 90°.
- MOIVRE 3 : composition de quatre figures du type 1.
- MOIVRE 4 : composition de quatre figures de type 1.
- MOIVRE 5 : autre composition de quatre figures de type 1.
- MOIVRE 6 : composition de deux figures de type 2 pour former un triangle.
- MOIVRE 7 : application de "MOIVRE 1" pour former le dessin d'un papillon.

### *Organigramme des programmes*

Chacun des sept programmes possède une structure linéaire qui consiste en une ou plusieurs boucles décrivant le glissement de la droite.

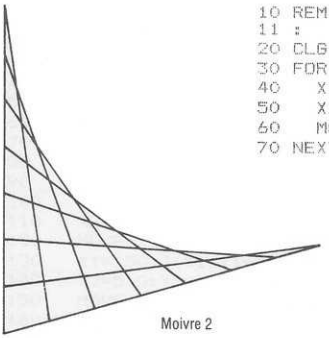
### *Programmes*



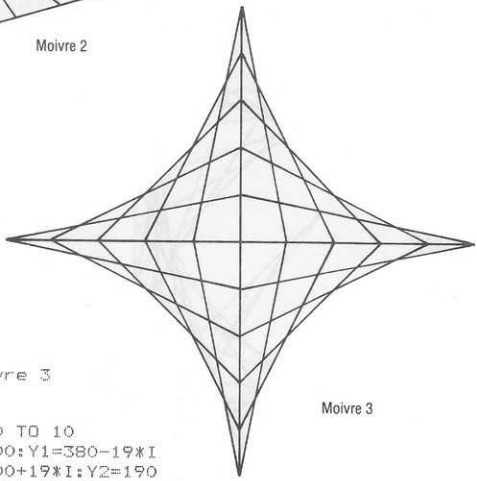
```
10 REM Moivre 1
11 :
20 CLG
30 FOR I=0 TO 10
40   X1=300:Y1=380-19*I
50   X2=300+19*I:Y2=190
60   MOVE X1,Y1:DRAW X2,Y2
70 NEXT I
```

Moivre 1

```
10 REM Moivre 2
11 :
20 CLG
30 FOR I=0 TO 10
40   X1=300:Y1=380-19*I
50   X2=300+19*I:Y2=190+19*I
60   MOVE X1,Y1:DRAW X2,Y2
70 NEXT I
```



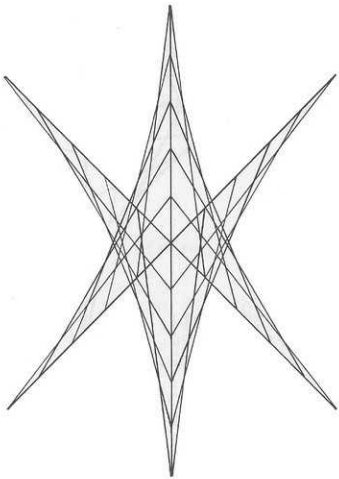
Moivre 2



Moivre 3

```
10 REM Moivre 3
11 :
20  CLG
30  FOR I=0 TO 10
40    X1=300:Y1=380-19*I
50    X2=300+19*I:Y2=190
60    MOVE X1,Y1:DRAW X2,Y2
70    X1=300:Y1=19*I
80    X2=300+19*I:Y2=190
90    MOVE X1,Y1:DRAW X2,Y2
100   X1=300:Y1=380-19*I
110   X2=300-19*I:Y2=190
120   MOVE X1,Y1:DRAW X2,Y2
130   X1=300:Y1=19*I
140   X2=300-19*I:Y2=190
150   MOVE X1,Y1:DRAW X2,Y2
160 NEXT I
```

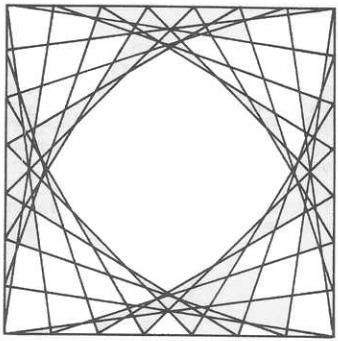
```
10 REM Moivre 4
11 :
20  CLG
30  FOR I=0 TO 10
40     X1=300:Y1=380-19*I
50     X2=300+19*I:Y2=190-19*I
60     MOVE X1,Y1:DRAW X2,Y2
70     X1=300:Y1=19*I
80     X2=300+19*I:Y2=190+19*I
90     MOVE X1,Y1:DRAW X2,Y2
100    X1=300:Y1=380-19*I
110    X2=300-19*I:Y2=190-19*I
120    MOVE X1,Y1:DRAW X2,Y2
130    X1=300:Y1=19*I
140    X2=300-19*I:Y2=190+19*I
150    MOVE X1,Y1:DRAW X2,Y2
160 NEXT I
```



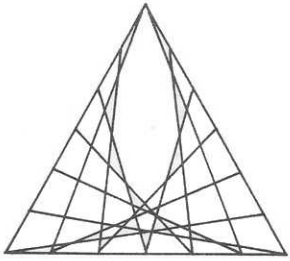
Moivre 4

```
10 REM Moivre 5
11 :
20  CLG
30  FOR I=0 TO 10
40    X1=150+36*I:Y1=0
50    X2=510:Y2=36*I
60    MOVE X1,Y1:DRAW X2,Y2
70    X1=510-36*I:Y1=0
80    X2=150:Y2=36*I
90    MOVE X1,Y1:DRAW X2,Y2
100   X1=510-36*I:Y1=360
110   X2=150:Y2=360-36*I
120   MOVE X1,Y1:DRAW X2,Y2
130   X1=150+36*I:Y1=360
140   X2=510:Y2=360-36*I
150   MOVE X1,Y1:DRAW X2,Y2
160 NEXT I
```

Moivre 5

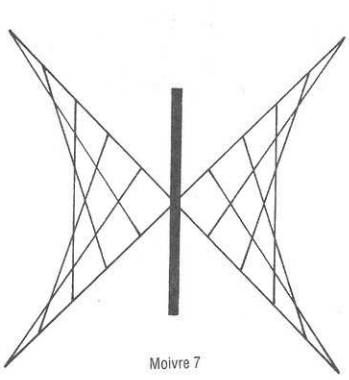


```
10 REM Moivre 6
11 :
20  CLG
30  FOR I=0 TO 10
40    X1=115-5*I:Y1=300-9*I
50    X2=65+10*I:Y2=210
60    MOVE X1,Y1:DRAW X2,Y2
70    X1=115+5*I:Y1=300-9*I
80    X2=165-10*I:Y2=210
90    MOVE X1,Y1:DRAW X2,Y2
100 NEXT
```



Moivre 6

```
10 REM Moivre 7, le Pap
11 :
20  CLG
30  FOR I=0 TO 10
40     X1=65+I*5:Y1=300-I*4
50     X2=115-5*I:Y2=260-4*I
60     MOVE X1,Y1:DRAW X2,Y2
70     X1=165-5*I:Y1=300-I*4
80     X2=115+I*5:Y2=260-I*4
90     MOVE X1,Y1:DRAW X2,Y2
100 NEXT I
110 MOVE 114,280:DRAW 116,280:DRAW 116,240
120 DRAW 114,240:DRAW 114,280
```



Moivre 7

# SIMULATION D'ORDRES GRAPHIQUES ÉVOLUÉS

La plupart des micro-ordinateurs possédant un graphisme haute résolution disposent des ordres **CIRCLE** et **BOX**.

**CIRCLE** : permet de tracer un cercle,
**BOX** : permet de tracer un rectangle.

Les deux programmes qui suivent permettent de simuler ces ordres. Les tracés se font en BASIC. Un "certain temps" est donc nécessaire pour tracer un cercle ou un rectangle...

## CIRCLE

Ce programme trace un cercle si on lui fournit les coordonnées du centre du cercle et le rayon du cercle. Il se sert du fait que l'équation Y(t), X(t) d'un cercle est de la forme :

$$X = X1 + R \cos(t)$$
$$Y = Y1 + R \sin(t)$$

avec X1,Y1 coordonnées du centre du cercle et R rayon du cercle.

Il suffit alors de faire varier t de 0 à $2\Pi$ pour obtenir un tracé complet du cercle.

### *Exemple*

```
10 CLG : REM Effacement d'ecran
20 X1=400:Y1=200:R=100:GOSUB 10000
30 END
```

### *Programme*

```
10000 REM ***********************************
10001 REM *                                   *
10002 REM *       TRACE     DE     CERCLES    *
10003 REM *                                   *
10004 REM ***********************************
10005 REM *                                   *
10006 REM * Entree : X1, Y1 Coord. Centre     *
10007 REM *            R Rayon du cercle      *
10008 REM *                                   *
10009 REM ***********************************
10010 :
10020 FOR I=0 TO 2*PI STEP PI/90
10030   X=X1+R*COS(I)
10040   Y=Y1+R*SIN(I)
10050   PLOT X,Y
10060 NEXT I
10070 :
10080 RETURN
```

***Analyse du programme***

Lignes 10030 à 10040 : Calcul des coordonnées du point courant.
Ligne 10050 : Tracé.

### BOX

Comme son nom l'indique, ce programme permet de tracer des boîtes de forme rectangulaire en précisant abscisse et ordonnée des deux points extrêmes de la boîte : point en haut à gauche et point en bas à droite.

***Exemple***

```
10 CLG : REM Effacement d'ecran
20 X1=30:Y1=100:X2=120:Y2=160:GOSUB 10000
30 END
```

***Programme***

```
10000 REM ********************************
10001 REM *                              *
10002 REM *       TRACE    DE    BOITES   *
10003 REM *                              *
10004 REM ********************************
10005 REM *                              *
10006 REM * Entree : X1,Y1 Cote sup gauche*
10007 REM *          X2,Y2 Cote sup droit *
10008 REM *                              *
10009 REM ********************************
10010 :
10020 MOVE X1,Y1
10030 DRAW X2,Y1:DRAW X2,Y2:DRAW X1,Y2:DRAW X1,Y1
10040 :
10050 RETURN
```

***Analyse du programme***

Ligne 10020 : Positionnement du curseur graphique.
Ligne 10030 : Tracé.

## PALETTE DE COULEURS

Si l'on rapproche petit à petit deux points élémentaires de couleurs différentes, il arrive un moment où l'œil ne fait plus la différence entre les deux

points : il perçoit un seul point dont la couleur est le mélange de la couleur de chaque point.

Ce principe est utilisé ici : la superposition d'un fond de couleur A et d'un motif de couleur B arrive à tromper l'œil qui perçoit une couleur C différente de A et de B, si le motif est bien choisi.

Une démonstration est faite en MODE 0, 1 et 2 sans utiliser l'instruction INK. Il est donc évident que ce programme ne montre pas la totalité des couleurs que l'on peut obtenir par ce procédé.

***Programme***

```
100 REM Palette de couleurs en MODE 1,2 et 3
110 :
120 REM MODE 0
130 :
140 MODE 0
150 FOR I=0 TO 15
160   CLS:PAPER I
170   FOR J=0 TO 15
180     PEN J
190     PRINT CHR$(207);
195     PAPER 0:PEN 1:PRINT"  PAPER"I"PEN"J:PAP
ER I
200   NEXT J
210   PAPER 0:PEN 1:PRINT:PRINT"Appuyez sur 1 t
ouche"
220 A$=INKEY$:IF A$="" THEN 220
230 NEXT I
240 :
250 REM MODE 1
260 :
270 MODE 1
280 FOR I=0 TO 3
290   CLS:PAPER I
300   FOR J=0 TO 3
310     PEN J
320     PRINT CHR$(207);
330     PAPER 0:PEN 1:PRINT"  PAPER"I"PEN"J:PAP
ER I
340   NEXT J
350   PAPER 0:PEN 1:PRINT:PRINT"Appuyez sur une
 touche"
360   A$=INKEY$:IF A$="" THEN 360
370 NEXT I
380 :
390 REM MODE 2
400 :
410 MODE 2
```

→

```
420 FOR I=0 TO 1
430   CLS:PAPER I
440   FOR J=0 TO 1
450     PEN J:PRINT CHR$(207);
460     PAPER 0:PEN 1:PRINT"  PAPER"I"PEN"J:PAP
ER I
470   NEXT J
480   PAPER 0:PEN 1:PRINT:PRINT"Appuyez sur une
 touche"
490   A$=INKEY$:IF A$="" THEN 490
500 NEXT I
```

***Analyse du programme***

| | |
|---|---|
| Lignes 120 à 230 | : Démonstration en MODE 0. |
| Ligne 160 | : Choix de la couleur PAPER. |
| Ligne 180 | : Choix de la couleur PEN. |
| Lignes 250 à 370 | : Démonstration en MODE 1. |
| Ligne 290 | : Choix de la couleur PAPER. |
| Ligne 310 | : Choix de la couleur PEN. |
| Lignes 410 à 500 | : Démonstration en MODE 2. |
| Ligne 430 | : Choix de la couleur PAPER. |
| Ligne 450 | : Choix de la couleur PEN. |

## REDÉFINITION DE CARACTÈRES

### Programmation de caractères graphiques

Si le jeu de caractères standard de l'Amstrad ne vous suffit pas, vous pouvez inventer de nouveaux caractères grâce à l'instruction SYMBOL. La commande "SYMBOL AFTER n" permet de redéfinir 256-n caractères, dont le lième sera affiché par "PRINT CHR$ (256−n+i)", pour i compris entre 0 et n−1.

Un caractère est défini dans une grille de 8 × 8 pixels.

Le programme suivant permet de définir un caractère dans sa grille. Les ordres sont : flèches haut, bas, gauche et droite pour se déplacer dans la grille, P pour allumer un pixel, V pour éteindre un pixel, ENTER quand le caractère est entièrement défini. Le programme fournit alors les arguments de l'ordre SYMBOL.

### *Programme*

```
10 REM Programmation de caracteres graphiques
11 :
20 CLS:DIM T(8,8)
30 FOR I=1 TO 8
40   PRINT "........"
50 NEXT I
60 X=1:Y=1 'Initialsation
70 LOCATE X,Y
80 A$=INKEY$:IF A$="" THEN 80
85 A=ASC(A$)
90 IF A$="P" THEN T(Y,X)=1:PRINT"*"
100 IF A$="V" THEN T(Y,X)=0:PRINT"."
110 IF A=243 AND X<8 THEN X=X+1
120 IF A=242 AND X>1 THEN X=X-1
130 IF A=240 AND Y>1 THEN Y=Y-1
140 IF A=241 AND Y<8 THEN Y=Y+1
150 IF A<>13 THEN 70
160 LOCATE 1,15:PRINT"Donnees correspondantes :
":PRINT
170 FOR I=1 TO 8
180   A=0 'Initialisation
190   FOR J=1 TO 8
200     A=A+(2^(8-J))*T(I,J)
210   NEXT J
220   PRINT A;
230 NEXT I
```

### *Analyse du programme*

| | |
|---|---|
| Lignes 20 à 50 | : Dessin de la grille-caractère. |
| Ligne 80 | : Attente de la frappe d'une touche. |
| Lignes 85 à 150 | : Action en fonction de la touche pressée. |
| Lignes 170 à 230 | : Affichage des arguments de SYMBOL. |

**Remarque :** Une fois votre caractère défini par ce programme, il suffit de le déclarer en tapant (par exemple "SYMBOL 250,1,2,4,8,16,32,64,128". Le caractère sera affiché par PRINT CHR$(250).

## Programmation de caractères graphiques multiples

Ce programme est une extension du précédent. Il vous permet de définir un objet quelconque dans une grille de 6 × 10 caractères, soit 48 × 80 pixels.

Les commandes sont les mêmes que pour le programme précédent.

Quand vous appuyez sur la touche ENTER, la sortie des données (paramètres de SYMBOL) peut se faire à l'écran ou sur imprimante.

***Programme***

```
100 REM Programmation de caracteres graphiques
multiples
110 :
200 REM Initialisation
210 :
220 CLS:MODE 1:INK 0,0:BORDER 0:PEN 1
230 DIM T(48,80)
240 X=1:Y=1 'Position de depart
245 :
250 REM ******************************
300 REM Gestion du curseur
310 :
320 LOCATE INT((X-1)/2)+1,INT((Y-1)/2)+1
330 A$=INKEY$:IF A$="" THEN 330
340 A=ASC(A$)
350 IF A$="P" THEN T(Y,X)=1:CP=1:GOSUB 1000:PRI
NT CHR$(CA) 'Carre plein
360 IF A$="V" THEN T(Y,X)=0:CP=0:GOSUB 1000:PRI
NT CHR$(CA) 'Carre vide
370 IF A=243 AND X<80 THEN X=X+1
380 IF A=242 AND X>1  THEN X=X-1
390 IF A=241 AND Y<48 THEN Y=Y+1
400 IF A=240 AND Y>1  THEN Y=Y-1
405 IF A<>243 AND A<>242 AND A<>241 AND A<>240
AND A<>80 AND A<>86 AND A<>13 THEN PRINT CHR$(7
) 'Action refusee
410 IF A<>13 THEN 320 'Boucle de saisie
430 REM ******************************
500 REM Affichage des resultats
510 :
520 CLS
530 PRINT"Sortie des resultats :"
540 PRINT"1) Sur ecran,"
550 PRINT"2) Sur imprimante."
560 PRINT:INPUT"Votre choix :";R
570 IF R<>1 AND R<>2 THEN 500 'Reponse refusee
580 IF R=1 THEN C=0 ELSE C=8
585 MODE 2 '80 Colonnes
590 :
600 FOR I=1 TO 6
610   FOR J=1 TO 10
615     PRINT#C:PRINT#C,"Ligne"I"Colonne"J":";
```

```
620       FOR K=1 TO 8
630         A=0
640         FOR L=1 TO 8
650           A=A+(2^(8-L))*T(K+(I-1)*8,L+(J-1)*8
)
660         NEXT L
670         PRINT#C,A;
680       NEXT K
700     NEXT J
710 NEXT I
720 :
730 END
740 REM ******************************
1000 REM Carre plein (CP=1) ou vide (CP=0)
1010 :
1015 S=0 'RAZ parametre de calcul
1020 IF INT(X/2)<>X/2 AND INT(Y/2)<>Y/2 THEN GO
SUB 1100
1030 IF INT(X/2)= X/2 AND INT(Y/2)<>Y/2 THEN GO
SUB 1200
1040 IF INT(X/2)<>X/2 AND INT(Y/2)= Y/2 THEN GO
SUB 1300
1050 IF INT(X/2)= X/2 AND INT(Y/2)= Y/2 THEN GO
SUB 1400
1060 CA=128+S 'Caractere a afficher
1070 :
1080 RETURN
1090 REM ******************************
1100 REM Caractere en haut a gauche
1110 :
1120 IF CP=1 THEN S=S+1
1130 IF T(Y,X+1)=1 THEN S=S+2
1140 IF T(Y+1,X)=1 THEN S=S+4
1150 IF T(Y+1,X+1)=1 THEN S=S+8
1160 :
1170 RETURN
1180 REM ******************************
1200 REM Caractere en haut a droite
1210 :
1220 IF CP=1 THEN S=S+2
1230 IF T(Y,X-1)=1 THEN S=S+1
1240 IF T(Y+1,X-1)=1 THEN S=S+4
1250 IF T(Y+1,X)=1 THEN S=S+8
1260 :
1270 RETURN
1280 REM ******************************
1300 REM Caractere en bas a gauche
1310 :
1320 IF CP=1 THEN S=S+4
```

→

```
1330 IF T(Y-1,X)=1 THEN S=S+1
1340 IF T(Y-1,X+1)=1 THEN S=S+2
1350 IF T(Y,X+1)=1 THEN S=S+8
1360 :
1370 RETURN
1380 REM ******************************
1400 REM Caractere en bas a droite
1410 :
1420 IF CP=1 THEN S=S+8
1430 IF T(Y-1,X-1)=1 THEN S=S+1
1440 IF T(Y-1,X)=1 THEN S=S+2
1450 IF T(Y,X-1)=1 THEN S=S+4
1460 :
1470 RETURN
```

***Analyse du programme***

Lignes 100 à 240 : Initialisation.
Lignes 300 à 410 : Gestion des commandes de l'opérateur.
Lignes 500 à 730 : Affichage des résultats.
Lignes 1000 à 1080 : Action suite à une commande P ou V.
Lignes 1100 à 1470 : Sous-programme de calculs sur les caractères à afficher.

## AFFICHAGE DE MOTIFS PROGRAMMÉS

### Affichage monochrome en MODE 1

L'utilitaire précédent ne saurait être complet sans ce programme qui permet de dessiner sur l'écran n'importe quel objet graphique défini par une commande SYMBOL.

A titre d'exemple, un héros de "Heroic fantasy" a été repris pour illustrer le programme.

***Programme***

```
10  REM Affichage monochrome d'objets graphique
s en MODE 1
20 :
30 MODE 1:INK 1,2:INK 2,0:PAPER 1:BORDER 2:PEN
2:CLS
40 X=2:Y=8:U=10:V=10
```

```
50 GOSUB 1000 'Definition du motif
60 GOSUB 2000 'Affichage du motif
70 END
80 REM *********************************
100 REM Donnees correspondant a l'objet a dessi
ner
110 :
120 DATA &FF,&FF,&FC,&F1,&E0,&E0,&D0,&C1
130 DATA 0,0,0,0,&80,&80,0,0
140 DATA &C8,&C7,0,&78,&FC,&BE,&BF,&BD
150 DATA 0,0,0,0,0,0,0,0
160 DATA &BD,&A9,&81,&B9,&42,&42,&42,&43
170 DATA 0,0,0,0,0,0,0,&0A
180 DATA &40,&A0,&E0,&FF,&FF,&FF,&80,&80
190 DATA &EF,&FF,&FF,0,0,0,&80,&80
200 DATA &80,&80,&88,&88,&88,&88,&4A,&FF
210 DATA &80,&80,&80,&80,&80,&80,&80,0
220 DATA &6F,&6F,&6F,&6F,&37,&37,&37,&37
230 DATA &80,&80,&80,&80,&80,&80,&80,&80
240 DATA &3B,&3B,&77,&77,&77,&3B,&37,&17
250 DATA &80,&80,&C0,&C0,&C0,&80,&80,0
260 DATA &16,&2F,&2F,&2F,0,0,0,0
270 DATA 0,0,&E0,&E0,0,0,0,0
1000 REM ***************************************
*
1010 REM *
*
1020 REM *   DEFINITION D'UN MOTIF GRAPHIQUE
*
1030 REM *
*
1040 REM ***************************************
*
1050 REM *
*
1060 REM * Entree : Donnees correspondant au
*
1070 REM *           motif graphique.
*
1080 REM *
*
1090 REM ***************************************
*
1100 :
1110 SYMBOL AFTER 129
1120 FOR I=1 TO 16
1130   FOR J=1 TO 8
1140     READ A(J)
1150   NEXT J
```

```
1160   SYMBOL 128+I,A(1),A(2),A(3),A(4),A(5),A(
6),A(7),A(8)
1170 NEXT I
1180 :
1190 RETURN
2000 REM ************************************
*
2010 REM *
*
2020 REM *     AFFICHAGE D'OBJETS GRAPHIQUES
*
2030 REM *
*
2040 REM ************************************
*
2050 REM *
*
2060 REM * Entree : X,Y Dimension de l'objet
*
2070 REM *           U,V Position de l'objet
*
2080 REM *           DATA graphiques
*
2090 REM * Sortie : Affichage de l'objet
*
2100 REM *
*
2110 REM ************************************
*
2120 :
2130 FOR I=1 TO Y
2140   FOR J=1 TO X
2150     LOCATE U+J-1,V+I-1
2160     PRINT CHR$(129+(J-1)+(I-1)*X)
2170   NEXT J
2180 NEXT I
2190 :
2200 RETURN
```

***Analyse du programme***

| | |
|---|---|
| Lignes 10 à 70 | : Programme principal. |
| Ligne 30 | : Définition des couleurs du fond et du personnage. |
| Ligne 40 | : Position et dimension du personnage. |
| Ligne 50 | : Définition du personnage. |
| Ligne 60 | : Affichage du personnage. |
| Lignes 100 à 270 | : Données graphiques correspondant au personnage. |

Lignes 1000 à 1190 : Définition du personnage (conversion des DATA en caractères accessibles par CHR$).
Lignes 2000 à 2200 : Affichage du personnage.

**Remarque :** L'entrée de la procédure de définition du motif (lignes 1000 à 1190) n'étant pas paramétrée, il vous faudra modifier la ligne 1120 sur laquelle figure le nombre de caractères à définir, si vous voulez créer un personnage différent.

## Animation monochrome en MODE 1

Forts de l'exemple précédent, nous voilà capables d'aborder l'animation graphique qui, dans sa plus simple expression (c'est le cas ici) se bornera à deux images différentes. L'alternance de ces deux images donnera une impression de déplacement.

Le programme a la même structure que le précédent, mais comporte en plus un sous-programme paramétrable qui permet d'effacer un motif graphique sur l'écran.

### *Programme*

```
10  REM Animation monochrome en MODE 1
20 :
30 REM Initialisation
40 :
50 MODE 1:INK 1,2:INK 2,0:PAPER 1:BORDER 2:PEN
2:CLS
55 PRINT"   Animation monochrome en MODE 1"
60 SYMBOL AFTER 129
70 SY=129:N=16:RESTORE 320:GOSUB 1000 'Heros au
 repos
80 SY=145:N=16:RESTORE 490:GOSUB 1000 'Heros en
 marche
90 X=2:Y=8
100 :
110 REM Deplacement du Hero
120 :
130 FOR Z=1 TO 30 STEP 2
140   U=Z:V=10:SY=130:GOSUB 2000
150   GOSUB 3000
160   U=U+1:SY=146:GOSUB 2000
170   GOSUB 3000
180 NEXT Z
190 CLS:END
200 REM ******************************
300 REM Donnees correspondant a l'objet a dessi
ner
310 :
320 DATA &FF,&FF,&FC,&F1,&E0,&E0,&D0,&C1
330 DATA 0,0,0,0,&80,&80,0,0
340 DATA &C8,&C7,0,&7B,&FC,&BE,&BF,&BD
350 DATA 0,0,0,0,0,0,0,0
```

→

```
360 DATA &BD,&A9,&81,&B9,&42,&42,&42,&43
370 DATA 0,0,0,0,0,0,0,&0A
380 DATA &40,&A0,&E0,&FF,&FF,&FF,&80,&80
390 DATA &EF,&FF,&FF,0,0,0,&80,&80
400 DATA &80,&80,&88,&88,&88,&88,&4A,&FF
410 DATA &80,&80,&80,&80,&80,&80,&80,0
420 DATA &6F,&6F,&6F,&6F,&37,&37,&37,&37
430 DATA &80,&80,&80,&80,&80,&80,&80,&80
440 DATA &3B,&3B,&77,&77,&77,&3B,&37,&17
450 DATA &80,&80,&C0,&C0,&C0,&80,&80,0
460 DATA &16,&2F,&2F,&2F,0,0,0,0
470 DATA 0,0,&E0,&E0,0,0,0,0
480 DATA ********************************
490 DATA &FF,&FF,&FC,&F1,&E0,&E0,&D0,&C1
500 DATA 0,0,0,0,&80,&80,0,0
510 DATA &C8,&C7,0,&7B,&FC,&FC,&FE,&FD
520 DATA 0,0,0,0,0,0,&C,&E
530 DATA &FD,&D0,&CD,&F1,&41,&41,&41,&41
540 DATA &9F,&CF,&EE,&F7,&FF,&7E,&3C,&1B
550 DATA &41,&41,&41,&FF,&FF,&FF,&80,&80
560 DATA 0,0,0,0,0,0,&80,&80
570 DATA &80,&80,&88,&88,&88,&88,&4A,&FF
580 DATA &80,&80,&80,&80,&80,&80,&80,0
590 DATA &3D,&3D,&3D,&3D,&3D,&7C,&78,&70
600 DATA &C0,&E0,&E0,&E0,&F0,&70,&70,&70
610 DATA &70,&70,&F8,&F8,&F8,&70,&70,&70
620 DATA &70,&70,&FC,&FC,&FC,&38,&38,&38
630 DATA &70,&F8,&FE,&FE,0,0,0,0
640 DATA &38,&7C,&7F,&7F,0,0,0,0
1000 REM ************************************
*
1010 REM *
*
1020 REM *   DEFINITION D'UN MOTIF GRAPHIQUE
*
1030 REM *
*
1040 REM ************************************
*
1050 REM *
*
1060 REM * Entree : Donnees correspondant au
*
1070 REM *          motif graphique.
*
1080 REM *          SY Sybole de debut de
*
1090 REM *          definition.
*
1100 REM *          N  Nombre de caracteres
*
1110 REM *          redefinis
*
1120 REM *
*
1130 REM ************************************
*
1140 :
1150 FOR I=1 TO N
1160   FOR J=1 TO 8
1170     READ A(J)
1180   NEXT J
1190   SYMBOL SY+I,A(1),A(2),A(3),A(4),A(5),A(6
),A(7),A(8)
1200 NEXT I
```

```
1210 :
1220 RETURN
2000 REM ****************************************
*
2010 REM *
*
2020 REM *     AFFICHAGE D'OBJETS GRAPHIQUES
*
2030 REM *
*
2040 REM ****************************************
*
2050 REM *
*
2060 REM * Entree : X,Y Dimension de l'objet
*
2070 REM *           U,V Position de l'objet
*
2080 REM *           DATA graphiques
*
2090 REM * Sortie : Affichage de l'objet
*
2100 REM *
*
2110 REM ****************************************
*
2120 :
2130 FOR I=1 TO Y
2140   FOR J=1 TO X
2150     LOCATE U+J-1,V+I-1
2160     PRINT CHR$(SY+(J-1)+(I-1)*X)
2170   NEXT J
2180 NEXT I
2190 :
2200 RETURN
3000 REM ****************************************
*
3010 REM *
*
3020 REM *    EFFACEMENT D'OBJETS GRAPHIQUES
*
3030 REM *
*
3040 REM ****************************************
*
3050 REM *
*
3060 REM * Entree : X,Y Dimension de l'objet
*
3070 REM *           U,V Position de l'objet
*
3080 REM * Sortie : Effacement de l'objet
*
3090 REM *
*
3100 REM ****************************************
*
3110 :
3120 FOR I=1 TO Y
3130   LOCATE U,V+I-1
3140   PRINT SPACE$(X)
3150 NEXT I
3160 :
3170 RETURN
```

### *Analyse du programme*

| | |
|---|---|
| Lignes 10 à 55 | : Initialisation. |
| Lignes 60 à 80 | : Définition du héros. |
| Lignes 90 à 190 | : Déplacement du héros. |
| Lignes 300 à 640 | : Données correspondant au héros. |
| Lignes 1000 à 1220 | : Conversion des DATA en caractères accessibles par CHR$. |
| Lignes 2000 à 2190 | : Affichage du personnage. |
| Lignes 3000 à 3170 | : Effacement du personnage. |

## Jeu d'animation monochrome en MODE 1

Beaucoup de jeux comportent un nombre limité de personnages. Les personnages ont eux-mêmes un nombre de positions différentes limité. Le programme suivant montre qu'il est facile de réaliser de tels jeux sur Amstrad.

Le héros qui a été défini précédemment peut se déplacer sur l'écran grâce aux touches-flèches et peut attraper des écus d'or (carrés !) qui tombent du ciel. Pour cela, il doit être bien positionné quand la pièce arrive à son niveau.

### *Programme*

```
10  REM Animation monochrome en MODE 1
20 :
30 REM Initialisation
40 :
50 MODE 1:INK 1,2:INK 2,0:PAPER 1:BORDER 2:PEN
2:CLS
60 X=2:Y=6:V=15:U=1:XP=1:PO=0 'Initialisation

70 SYMBOL AFTER 129
80 SY=129:N=16:RESTORE 270:GOSUB 1000 'Heros au
 repos droit
90 SY=145:N=16:RESTORE 440:GOSUB 1000 'Heros en
 marche droite
100 SY=161:N=16:RESTORE 610:GOSUB 1000 'Heros a
u repos gauche
110 SY=177:N=16:RESTORE 780:GOSUB 1000 'Heros e
n marche gauche
120 SY=193:N=1:RESTORE 950:GOSUB 1000 'Piece ca
rree
130 X=2:Y=8
140 :
```

```
150 REM Deplacement du Heros
160 :
170 A$=INKEY$ 'Lecture clavier
180 IF A$<>"" THEN A=ASC(A$)
190 IF A=243 THEN GOSUB 4000 'Deplacement vers
la droite
200 IF A=242 THEN GOSUB 5000 'Deplacement vers
la gauche
210 GOSUB 6000 'Deplacement des Pieces Carrees
220 IF A<>13 THEN 170 'Boucle de jeu
230 CLS:END
240 REM ******************************
250 REM Donnees correspondant au heros
260 :
270 DATA &FF,&FF,&FC,&F1,&E0,&E0,&D0,&C1
280 DATA 0,0,0,0,&80,&80,0,0
290 DATA &C8,&C7,0,&78,&FC,&BE,&BF,&BD
300 DATA 0,0,0,0,0,0,0,0
310 DATA &BD,&A9,&81,&B9,&42,&42,&42,&43
320 DATA 0,0,0,0,0,0,0,&0A
330 DATA &40,&A0,&E0,&FF,&FF,&FF,&80,&80
340 DATA &EF,&FF,&FF,0,0,0,&80,&80
350 DATA &80,&80,&88,&88,&88,&88,&4A,&FF
360 DATA &80,&80,&80,&80,&80,&80,&80,&80,0
370 DATA &6F,&6F,&6F,&6F,&37,&37,&37,&37
380 DATA &80,&80,&80,&80,&80,&80,&80,&80,&80
390 DATA &3B,&3B,&77,&77,&77,&3B,&37,&17
400 DATA &80,&80,&C0,&C0,&C0,&80,&80,0
410 DATA &16,&2F,&2F,&2F,0,0,0,0
420 DATA 0,0,&E0,&E0,0,0,0,0
430 DATA ********************************
440 DATA &FF,&FF,&FC,&F1,&E0,&E0,&D0,&C1
450 DATA 0,0,0,0,&80,&80,0,0
460 DATA &C8,&C7,0,&78,&FC,&FC,&FE,&FD
470 DATA 0,0,0,0,0,0,&C,&E
480 DATA &FD,&D0,&CD,&F1,&41,&41,&41,&41
490 DATA &9F,&CF,&EE,&F7,&FF,&7E,&3C,&18
500 DATA &41,&41,&41,&FF,&FF,&FF,&80,&80
510 DATA 0,0,0,0,0,0,&80,&80
520 DATA &80,&80,&88,&88,&88,&88,&4A,&FF
530 DATA &80,&80,&80,&80,&80,&80,&80,0
540 DATA &3D,&3D,&3D,&3D,&3D,&3D,&7C,&78,&70
550 DATA &C0,&E0,&E0,&E0,&F0,&70,&70,&70
560 DATA &70,&70,&F8,&F8,&F8,&70,&70,&70
570 DATA &70,&70,&FC,&FC,&FC,&38,&38,&38
580 DATA &70,&F8,&FE,&FE,0,0,0,0
590 DATA &3B,&7C,&7F,&7F,0,0,0,0
600 REM ******************************
610 DATA 0,0,0,0,1,1,0,0
620 DATA &FF,&FF,&3F,&5F,7,7,&B,&83
630 DATA 0,0,0,0,0,0,0,0
640 DATA &13,&E3,0,&1E,&3F,&7D,&FD,&BD
650 DATA 0,0,0,0,0,0,0,&50
660 DATA &BD,&95,&81,&9D,&42,&42,&42,&C2
670 DATA &F7,&FF,&FF,0,0,0,1,1
680 DATA 2,6,6,&FF,&FF,&FF,1,1
690 DATA 1,1,1,1,1,1,1,0
700 DATA 1,1,&11,&11,&11,&11,&52,&FE
710 DATA 1,1,1,1,1,1,1,1
720 DATA &F6,&F6,&F6,&F6,&EC,&EC,&EC,&EC
```

→

```
730 DATA 1,1,3,3,3,1,1,0
740 DATA &DC,&DC,&EE,&EE,&EE,&DC,&EC,&E8
750 DATA 0,0,7,7,0,0,0,0
760 DATA &68,&F4,&F4,&F4,0,0,0,0
770 REM ********************************
780 DATA 0,0,0,0,1,1,0,0
790 DATA &FF,&FF,&3F,&5F,7,7,&B,&83
800 DATA 0,0,0,0,0,0,&30,&70
810 DATA &13,&E3,0,&1E,&3F,&3F,&7F,&BF
820 DATA &F9,&F3,&77,&EF,&FF,&7E,&3C,&18
830 DATA &BF,&BB,&B7,&BF,&11,&81,&81,&81
840 DATA 0,0,0,0,0,0,1,1
850 DATA &81,&81,&81,&FF,&FF,&FF,1,1
860 DATA 1,1,1,1,1,1,1,0
870 DATA 1,1,&11,&11,&11,&11,&52,&FE
880 DATA 3,7,7,7,&F,&E,&E,&E
890 DATA &BC,&BC,&BC,&BC,&BC,&3E,&1E,&E
900 DATA &E,&E,&3F,&3F,&3F,&1C,&1C,&1C
910 DATA &E,&E,&1F,&1F,&1F,&E,&E,&E
920 DATA &1C,&3E,&FE,&FE,0,0,0,0
930 DATA &E,&1F,&7F,&7F,0,0,0,0
940 REM ********************************
950 DATA &FF,&81,&81,&81,&81,&81,&81,&FF
1000 REM ***************************************
*
1010 REM *
*
1020 REM *   DEFINITION D'UN MOTIF GRAPHIQUE
*
1030 REM *
*
1040 REM ***************************************
*
1050 REM *
*
1060 REM * Entree : Donnees correspondant au
*
1070 REM *           motif graphique.
*
1080 REM *           SY Sybole de debut de
*
1090 REM *           definition.
*
1100 REM *           N  Nombre de caracteres
*
1110 REM *           redefinis
*
1120 REM *
*
1130 REM ***************************************
*
1140 :
1150 FOR I=1 TO N
1160   FOR J=1 TO 8
1170     READ A(J)
1180   NEXT J
1190   SYMBOL SY+I,A(1),A(2),A(3),A(4),A(5),A(6
),A(7),A(8)
1200 NEXT I
1210 :
1220 RETURN
```

```
2000 REM ***************************************
*
2010 REM *
*
2020 REM *    AFFICHAGE D'OBJETS GRAPHIQUES
*
2030 REM *
*
2040 REM ***************************************
*
2050 REM *
*
2060 REM * Entree : X,Y Dimension de l'objet
*
2070 REM *          U,V Position de l'objet
*
2080 REM *          DATA graphiques
*
2090 REM * Sortie : Affichage de l'objet
*
2100 REM *
*
2110 REM ***************************************
*
2120 :
2130 FOR I=1 TO Y
2140   FOR J=1 TO X
2150     LOCATE U+J-1,V+I-1
2160     PRINT CHR$(SY+(J-1)+(I-1)*X)
2170   NEXT J
2180 NEXT I
2190 :
2200 RETURN
3000 REM ***************************************
*
3010 REM *
*
3020 REM *   EFFACEMENT D'OBJETS GRAPHIQUES
*
3030 REM *
*
3040 REM ***************************************
*
3050 REM *
*
3060 REM * Entree : X,Y Dimension de l'objet
*
3070 REM *          U,V Position de l'objet
*
3080 REM * Sortie : Effacement de l'objet
*
3090 REM *
*
3100 REM ***************************************
*
3110 :
3120 FOR I=1 TO Y
3130   LOCATE U,V+I-1
3140   PRINT SPACE$(X)
3150 NEXT I
3160 :
3170 RETURN
```

→

```
4000 REM ****************************************
*
4010 REM *
*
4020 REM * DEPLACEMENT DU HEROS VERS LA DROITE
*
4030 REM *
*
4040 REM ****************************************
*
4050 :
4060 IF U>35 THEN RETURN 'Deplacement refuse
4070 :
4080 IF P=1 THEN P=0 ELSE P=1 'Position repos o
u marche
4090 GOSUB 3000 'Effacement position precedente

4100 IF P=1 THEN SY=130 ELSE SY=146
4110 U=U+2:GOSUB 2000 'Deplacement du heros
4120 :
4130 RETURN
5000 REM ****************************************
*
5010 REM *
*
5020 REM * DEPLACEMENT DU HEROS VERS LA GAUCHE
*
5030 REM *
*
5040 REM ****************************************
*
5050 :
5060 IF U<3  THEN RETURN 'Deplacement refuse
5070 :
5080 IF P=1 THEN P=0 ELSE P=1 'Position repos o
u marche
5090 GOSUB 3000 'Effacement position precedente
5100 IF P=1 THEN SY=162 ELSE SY=178
5110 U=U-2:GOSUB 2000 'Deplacement du heros
5120 :
5130 RETURN
6000 REM ****************************************
*
6010 REM *
*
6020 REM * DEPLACEMENT DES PIECES CARREES
*
6030 REM *
*
6040 REM ****************************************
*
6050 :
6060 IF RND<0.8 THEN RETURN 'Aucune action
6070 IF PO>0 THEN LOCATE XP,PO:PRINT" "
6080 IF PO>10 AND ABS(XP-U)<2 THEN PRINT CHR$(7
) '1 point
6090 IF PO>10 THEN PO=0
6100 IF PO=0 THEN XP=INT(RND*30)+5
6110 PO=PO+2
6120 LOCATE XP,PO:PRINT CHR$(194)
6130 RETURN
```

*Analyse du programme*

| | |
|---|---|
| Lignes 10 à 230 | : Programme principal. |
| Lignes 30 à 70 | : Initialisation. |
| Lignes 80 à 120 | : Définition des objets graphiques. |
| Ligne 130 | : Initialisation de la position du héros. |
| Lignes 150 à 230 | : Déplacement des divers objets graphiques. |
| Lignes 250 à 950 | : Données correspondant au héros et à l'écu d'or. |
| Lignes 1000 à 1220 | : Définition des objets graphiques (conversion des DATA en caractères accessibles par CHR$). |
| Lignes 2000 à 2200 | : Affichage d'un objet graphique. |
| Lignes 3000 à 3170 | : Effacement d'un objet graphique. |
| Lignes 4000 à 4130 | : Déplacement du héros vers la droite. |
| Lignes 5000 à 5130 | : Déplacement du héros vers la gauche. |
| Lignes 6000 à 6130 | : Déplacement des écus d'or. |

**Remarque :** En suivant la même démarche, vous pouvez à moindre frais créer de "bons" jeux d'animation. Pour définir les personnages ou objets du jeu, servez-vous du programme "Programmation de caractères graphiques multiples".

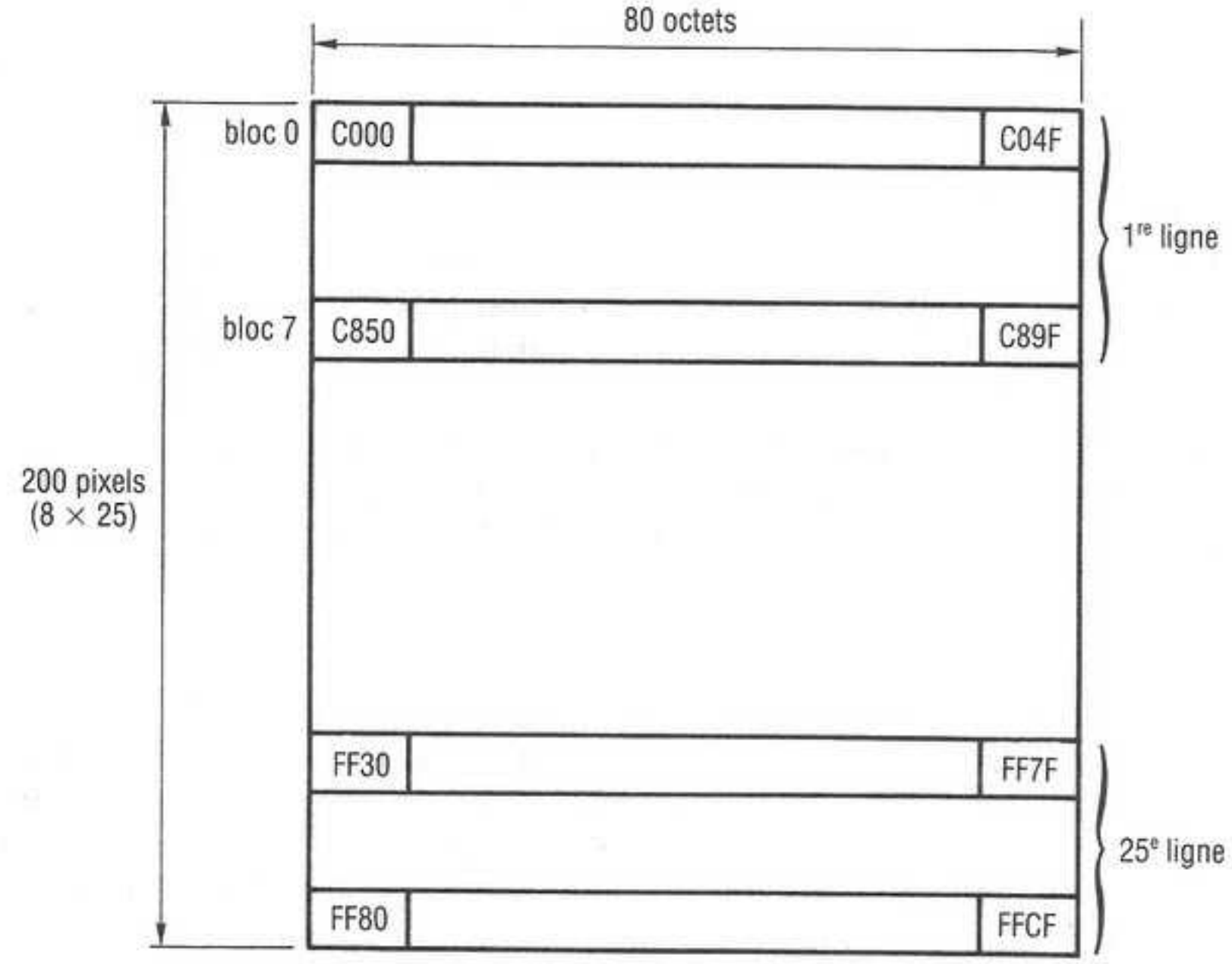
Parmi les programmes qui suivent, certains font appel à des routines écrites en Assembleur. Ces routines sont stockées en mémoire par un programme BASIC où elles figurent sous forme de DATA, et sont appelées par « CALL &ADRESSE ». Si parmi les lecteurs, il y a des personnes désireuses de comprendre le fonctionnement des routines Assembleur présentées, un listing commenté est fourni pour chaque routine et une analyse sommaire est faite. (Le microprocesseur de l'Amstrad est un Z80, et aucune astuce de programmation n'est employée ; le décryptage du programme devrait donc être simplifié pour une personne connaissant bien l'Assembleur Z80.)

## MODIFICATION DU CONTENU DE L'ÉCRAN

### PAPER

Le programme présenté ici permet d'écrire en bloc dans la mémoire d'écran pour reproduire l'effet de la commande "PAPER". Il n'a de raison d'être que d'un point de vue pédagogique (ménageons le lecteur, sans quoi il risque d'être affolé par la progression vertigineuse de la difficulté !).

Avant d'entrer dans le détail du programme, analysons la structure de la mémoire d'écran. Tout d'abord, elle est implantée en &C000. Quel que soit le MODE (0, 1 ou 2), elle est divisée en 8 blocs, 1 bloc constituant une ligne élémentaire par ligne et une ligne élémentaire comportant 80 octets selon le schéma ci-dessous :




**Remarque :** Chaque bloc fait 2 KO soit 2048 octets. Or, on voit que seulement 25 × 80 = 2000 octets sont utilisés par la mémoire d'écran. Les 48 autres octets sont perdus et ne servent à rien.

## Organisation de chaque octet graphique en fonction du mode d'affichage

MODE 0 :

| | LSB | | | | | | | MSB |
|---|---|---|---|---|---|---|---|---|
| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PEN | 8 | 2 | 4 | 1 | 8 | 2 | 4 | 1 |
| Pixel | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

| | | LSB | | | | | | MSB |
|---|---|---|---|---|---|---|---|---|
| | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| MODE 1 : | PEN | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| | Pixel | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |

| | | LSB | | | | | | | MSB |
|---|---|---|---|---|---|---|---|---|---|
| | Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| MODE 2 : | PEN | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Pixel | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

LSB = Least Significative Bit
MSB = Most Significative Bit

Revenons au programme de remplissage d'écran.

Il est constitué d'une boucle dans laquelle on remplit chaque octet graphique avec &0F, ce qui a pour effet, en MODE 1, de remplir l'écran avec la couleur PEN 2.

```
        LDA     0F          * Motif de couleur PEN 2
        LD      HL,C000     * Debut de la memoire d'ecran
        LD      D,40        * Nombre de boucles de FF ecritures
L1      LD      B,0         * FF ecritures
L2      LD      (HL),A      * Ecriture SCRN
        INC     HL          * Passage a l'octet suivant
        DJNZ    L2          * B-1, bis si B<>0
        DEC     D           * 1 boucle de FF ecritures realisee
        JR      NZ,L1       * Bis si non fini
        RET                 * Retour au BASIC
```

Ce programme Assembleur est incorporé dans un programme BASIC. L'appel à la routine ASM est fait en ligne 180.

```
100 REM Remplissage d'ecran
110 :
120 FOR I=0 TO 16
130   READ A:POKE &B000+I,A
140 NEXT I
150 :
160 DATA &3E,&F0,&21,0,&C0,&16,&40,&06,&0,&77,&
23,&10,&FC,&15,&20,&F9,&C9
170 :
180 CALL &B000
```

## Sauvegarde/lecture de contexte

Les deux programmes proposés ici permettent, pour le premier, de mémoriser une page écran, pour le second, de restituer la page mémorisée.

La mémoire d'écran commence normalement en &C000 et occupe &3FFF octets. L'offset (ou déplacement) du 1er caractère par rapport au début de l'écran est alors nul. Malheureusement, dès qu'un SCROLLING d'écran se produit (lors d'un LIST par exemple), l'offset prend une valeur non nulle, ce qui a pour effet de faire commencer la mémoire d'écran &C000 à un endroit différent du début (coin en haut à gauche) de l'écran.

Heureusement, une routine du firmware CPC permet de forcer le début de l'écran en &C000. Cette routine est située en &BBFF. Elle doit être appelée avant toute sauvegarde ou restitution de contexte, et doit être précédée de CLS (voir exemple).

Pour que les deux routines suivantes (sauvegarde et restitution de contexte) fonctionnent normalement, il est nécessaire que les opérations ayant fait suite à l'initialisation de l'écran (CALL &BBFF) n'aient pas produit de scrolling.

Par exemple, pour utiliser le programme de sauvegarde de contexte :

```
CLS : CALL &BBFF
LIST
RUN
```

Pour utiliser le programme de restitution de contexte :

```
CLS : CALL &BBFF
RUN
```

Si l'affichage réalisé par le second programme (restitution de contexte) vous paraît bizarre, c'est que vous n'avez pas respecté les opérations (CLS:CALL &BBFF) d'initialisation avant de lancer le programme de sauvegarde.

Sauvegarde de contexte :

Le programme Assembleur fait une copie de la mémoire d'écran &C000 à &FFFF en mémoire vive à partir de l'adresse &6000.

```
      LD    HL,6000 * @ de debut de sauvegarde
      LD    DE,C000 * @ de debut de mem. ecran
      LD    C,40    * Nombre de boucles de FF ecritures
L1    LD    B,0     * FF ecritures
L2    LD    A,(DE)  * Lecture ecran
```

```
    LD    (HL),A  * Sauvegarde memoire
    INC   HL      * Passage a la memoire suivante
    INC   DE      * Passage a la memoire ecran suivante
    DJNZ  L2      * B-1, bis si B<>0
    DEC   C       * 1 boucle de FF ecritures realisee
    JR    NZ,L1   * Bis si non fini
    RET           * Retour au BASIC
```

Le programme BASIC mémorise la routine de sauvegarde à partir de l'adresse &5FEO et l'exécute.

```
100 REM Sauvegarde de contexte
110 :
120 FOR I=0 TO 19
130   READ A:POKE &5FE0+I,A
140 NEXT I
150 :
160 DATA &21,0,&60,&11,0,&C0,&E,&3F,6,0,&1A,&77
,&23,&13,&10,&FA,&D,&20,&F6,&C9
170 :
180 CALL &5FE0
```

Lecture de contexte :

Le programme Assembleur fait une copie de la mémoire &6000 à &9FFF en mémoire d'écran.

```
      LD    HL,6000 * @ de debut de sauvegarde
      LD    DE,C000 * @ de debut de mem. ecran
      LD    C,40    * Nombre de boucles de FF ecriture
L1    LD    B,0     * FF ecritures
L2    LD    A,(HL)  * Lecture memoire
      LD    (DE),A  * Sauvegarde ecran
      INC   HL      * Passage a la memoire suivante
      INC   DE      * Passage a la memoire ecran suivante
      DJNZ  L2      * B-1, bis si B<>0
      DEC   C       * 1 boucle de FF ecritures realisee
      JR    NZ,L1   * Bis si non fini
      RET           * Retour au BASIC
```

Le programme BASIC mémorise la routine de lecture à partir de l'adresse &5FAO et l'exécute.

```
100 REM Lecture de contexte
110 :
120 FOR I=0 TO 19
130   READ A:POKE &5FA0+I,A
140 NEXT I
150 :
160 DATA &21,0,&60,&11,0,&C0,&E,&3F,6,0,&7E,&12
,&23,&13,&10,&FA,&D,&20,&F6,&C9
170 :
180 CALL &5FA0
```

**Remarque :** Si, entre la sauvegarde et la restitution de contexte, vous faites des opérations de chargement (LOAD), la page sauvegardée risque d'être altérée. En principe, ce problème n'en est pas un puisque ces deux utilitaires sont incorporés dans un même programme, et qu'aucune opération "LOAD" n'est alors nécessaire.

## REMPLISSAGE DE BOÎTES

Il peut être intéressant (par exemple pour tracer un histogramme sous forme de barres) de pouvoir remplir facilement un rectangle d'une dimension quelconque.

Les deux programmes qui suivent réalisent cette tâche. Le premier est écrit en BASIC, le second en Assembleur.

Programme en BASIC :

Il consiste en une suite de DRAWR en partant du bas du rectangle à remplir.

### *Exemple d'appel*

```
10 X1=100:X2=300:Y1=100:Y2=200
20 EN=2 'Couleur d'encre du trace
30 CLS:GOSUB 11000 'Appel de BOXF
40 END
```

### *Programme*

```
10000 REM ****************************************
10010 REM *                                      *
10020 REM *              TRACE  DE  BOITES       *
10030 REM *                                      *
10040 REM ****************************************
11000 FOR I=Y1 TO Y2
11010   MOVE X1,I
11020   DRAWR X2-X1,0,EN
11030 NEXT I
11040 RETURN
```

### *Analyse du programme*

Ligne 11010 : Positionnement du curseur graphique.
Ligne 11020 : Tracé de la droite.

Programme en Assembleur :

Il réalise la même action de remplissage, de bas en haut du rectangle en faisant appel à la routine firmware "LINE RELATIVE".

```
L1    LD      DE,(B040) * Abcisse du MOVE
      LD      HL,(B044) * Ordonnee du MOVE
      CALL    #BBC0     * MOVE absolu
      LD      DE,(B042) * Longueur du LINE en X
      LD      HL,0      * Longueur du LINE en Y
      CALL    #BBF9     * LINE relatif
      LD      HL,(B044) * Ligne a tracer
      INC     HL        * Passage a la ligne suivante
      LD      (B044),HL * Memorisation
      LD      HL,(B046) * Nombre de lignes a tracer
      DEC     HL        * 1 de moins
      LD      (B046),HL * Memorisation
      LDA     A,H
      OR      L
      JR      NZ,L1     * Boucle si Job non termine
      RET               * Retour au BASIC
```

Le programme ASM est inséré dans un programme BASIC sous forme de DATA.

***Exemple d'appel***

```
100 GOSUB 1000 'Initialisation
110 X=100:LX=120:Y=20:LY=200 'Parametre de BOXF
120 CLS:GOSUB 1100
130 END
```

***Programme***

```
1000 REM Mise en memoire de la routine ASM
1010 :
1020 FOR I=0 TO 38
1030   READ A:POKE &B000+I,A
1040 NEXT I
1050 DATA &ED,&5B,&40,&B0,&2A,&44,&B0,&CD,&C0,&
BB,&ED,&5B,&42,&B0
1060 DATA &21,0,0,&CD,&F9,&BB,&2A,&44,&B0,&23,&
22,&44,&B0
1070 DATA &2A,&46,&B0,&2B,&22,&46,&B0,&7C,&B5,&
20,&DA,&C9
1080 :
1090 RETURN
1100 REM ******************************
```

```
1200 REM Procedure BOXF
1210 :
1220 REM Entree : X = Abcisse du bord gauche
1230 REM          LX= Largeur de la boite
1240 REM          Y = Ordonnee du bas de la boi
te
1250 REM          LY= Hauteur de la boite
1260 :
1270 A(2)=INT(X/256):A(1)=X-A(2)
1280 A(4)=INT(LX/256):A(3)=LX-A(4)
1290 A(6)=INT(Y/256):A(5)=Y-A(6)
1300 A(8)=INT(LY/256):A(7)=LY-A(8)
1310 FOR I=0 TO 7
1320   POKE &B040+I,A(I+1)
1330 NEXT I
1340 :
1350 CALL &B000
1360 :
1370 RETURN
```

***Analyse du programme***

Lignes 1000 à 1090 : Mise en mémoire de la procédure ASM.
Lignes 1100 à 1370 : Passage des données à la procédure ASM et exécution.

## REMPLISSAGE D'UNE SURFACE DÉLIMITÉE PAR UN CONTOUR FERMÉ

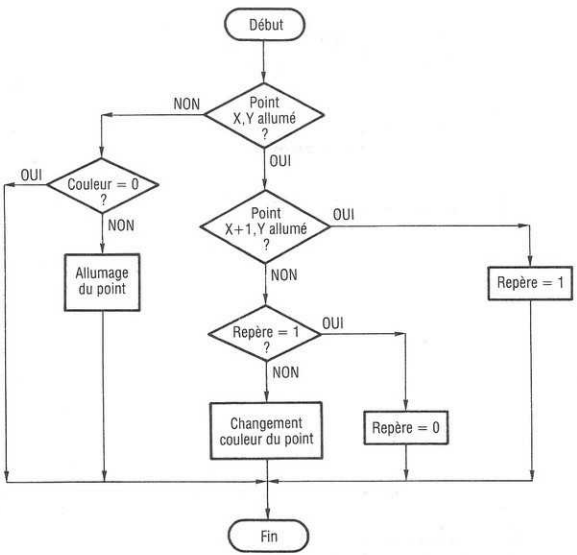
Le micro-ordinateur Amstrad ne possède aucun ordre de remplissage d'une surface délimitée par un contour fermé. L'utilitaire suivant comble cette lacune.

Avant d'entrer dans le détail du programme, analysons la démarche logique utilisée pour réaliser le remplissage.

L'écran possède 640 × 200 pixels. Ces pixels sont accessibles par les instructions graphiques MOVE, PLOT, DRAW, TEST, etc. avec les arguments X (0..639) et Y (0..399). Le tracé d'un contour fermé sur l'écran pourra être repéré par l'instruction TEST(<Coord. X>, <Coord. Y>). En balayant l'écran, pour X de 0 à 639, et pour Y de 0 à 399, nous pourrons connaître l'état coloré de chaque point par le résultat de la fonction TEST.

Il suffira, partant d'une encre PEN 0, identique à la couleur PAPER, de modifier PEN 0 en PEN 1 (et inversement) à chaque occurrence (TEST)<>PAPER, dans un balayage horizontal de l'écran.

Ce qui se schématise par l'organigramme suivant :




L'organigramme précédent nous montre que le changement de couleur sera effectué uniquement si deux points consécutifs ne sont pas allumés. En effet, tout segment de droite horizontal ne demande pas de changement de couleur, et le programme considère comme segment de droite, toute proximité immédiate de deux points allumés. Le programme de démonstration donné par la suite montre les problèmes que l'on peut rencontrer (sur le cercle) pour remplir une surface délimitée par un contour sur lequel il existe des points consécutifs allumés.

```
        LD    A,0
        CALL  #BBDE      * Modif encre HIRES
        LD    HL,0
        LD    (BF00),HL
        LD    (BF02),HL
        LD    (BF04),HL  * Initialisation
20      LD    HL,(BF00)
        INC   HL
        LD    (BF00),HL  * No de point sur la ligne
        LD    BC,&639
        SCF
        SBC   HL,BC      * Est-on en fin de ligne ?
        JR    C,L1       * NON
        LD    HL,0000    * OUI
        LD    (BF00),HL  * RAZ indic ligne
        LD    HL,(BF02)
        INC   HL
        INC   HL
        LD    (BF02),HL  * Indic de colonne
        LD    BC,&399    * Est-on en haut de l'ecran ?
        SCF
        SBC   HL,BC
        JR    NC,L2      * OUI
L1      LD    DE,(BF00)  * NON
        LD    HL,(BF02)
        CALL  #BBF0      * GRA TEST ABS
        CP    1          * Point allume ?
        JR    C,L3       * NON
        LD    DE,(BF00)
        LD    HL,(BF02)
        INC   DE
        CALL  #BBF0      * GRA TEST ABS
        CP    1          * Point + 1 allume ?
        JR    NC,11      * OUI
        LD    A,(BF05)
        ORA   A          * Flag presence = 1 ?
        JR    NZ,12      * OUI
        LD    A,(BF04)
        CP    1
        JR    C,L4
        LD    A,0
        LD    (BF04),A   * RAZ flag presence
        JR    L5
L4      LD    A,1
        LD    (BF04),A   * RA1 flag de presence
L5      CALL  #BBDE      * GRA SET PEN
L3      LD    A,(BF04)
        CP    1          * Presence = 1 ?
        JR    C,10       * NON
        LD    DE,(BF00)
        LD    HL,(BF02)
        CALL  #BBEA      * GRA PLOT ABS
10      JR    20
11      LD    A,1
        LD    (BF05),A
        JR    L3
12      LD    A,0
        LD    (BF05),A
        JR    L3
L2      LD    A,1
        CALL  #BBDE      * GRA SET PEN
        RET              * Retour au BASIC
```

Le programme Assembleur est inséré en DATA dans un programme BASIC sous forme d'une procédure qui doit être appelée par "GOSUB 10000" pour être mise en mémoire et par CALL &BF10 pour être exécutée.

**Remarque :** CALL &BFA0 permet de sélectionner PEN 1 comme prochaine couleur d'affichage graphique.

```
10000 REM ************************************
**********************
10010 REM *
                        *
10020 REM *  MEMORISATION DE LA PROCEDURE ASM D
E REMPLISSAGE       *
10030 REM *
                        *
10040 REM ************************************
**********************
10050 :
10060 FOR I=0 TO 149:READ A:POKE &BF10+I,A:NEXT
 I
10070 DATA &3E,0,&CD,&DE,&BB
10080 DATA &21,0,0,&22,0,&BF,&22,2,&BF,&22,4,&B
F,&2A,0,&BF,&23,&22,0,&BF
10090 DATA 1,&7F,2,&37,&ED,&42,&38,&16,&21,0,0,
&22,0,&BF,&2A,2,&BF,&23,&23,&22,2,&BF
10100 DATA 1,&BF,1,&37,&ED,&42,&30,&5A,&ED,&5B,
0,&BF,&2A,2,&BF,&CD,&F0,&BB
10110 DATA &FE,1,&38,&2B,&ED,&5B,0,&BF,&2A,2,&B
F,&13,&CD,&F0,&BB,&FE,1,&30,&2F
10120 DATA  &3A,5,&BF,&B7,&20,&30
10130 DATA &3A,4,&BF,&FE,1,&38,7,&3E,00,&32,4,&
BF,&18,5,&3E,1,&32,4,&BF
10140 DATA &CD,&DE,&BB,&3A,4,&BF,&FE,1,&38,&A,&
ED,&5B,0,&BF,&2A,2,&BF,&CD,&EA,&BB
10150 DATA &18,&8F
10160 DATA &3E,&01,&32,5,&BF,&18,&E6,&3E,0,&32,
5,&BF,&18,&DF
10170 DATA &3E,1,&CD,&DE,&BB,&C9
10180 RETURN
```

***Exemple d'utilisation***

```
10 GOSUB 10000 'Initialisation
20 MODE 2 'Passage en mode haute resolution
30 :
40 REM Rectangle 1
50 MOVE 300,20:DRAWR 0,50:DRAWR 20,0:DRAWR 0,-5
0:DRAWR -20,0
```

```
60 :
70 REM Triangle
80 FOR I=1 TO 100 STEP 2:PLOT I,I:NEXT
90 FOR I=101 TO 200 STEP 2:PLOT I,200-I:NEXT
100 MOVE 1,1:DRAWR 198,0
110 :
120 REM Cercle
130 DEG
140 FOR a=1 TO 360
150   PLOT 80+80*COS(a),250+80*SIN(a)
160 NEXT a
170 :
180 REM Rectangle 2
190 MOVE 300,200:DRAWR 200,0:DRAWR 0,30:DRAWR -
200,0:DRAWR 0,-30
200 :
210 CALL &BFA0 'Prepositionnement PEN du BOXF
220 CALL &BF10 'Remplissage des contours fermes

230 END
```

**Remarque :** On peut noter que :

- Le cercle n'est pas correctement rempli. Ceci est dû à sa définition très grossière. En effet, il existe plusieurs paires de points consécutifs allumés. Ces paires de points sont considérées comme des droites et ne demandent donc pas de modification de couleur.
- Le problème précédent a été résolu dans le tracé du triangle. L'instruction PLOT remplace avantageusement l'instruction DRAW qui se permet (eh oui !) d'allumer deux points consécutifs sur une même ligne. Essayez donc de remplir un triangle confectionné à l'aide des instructions DRAW, du genre : MOVE 100,100:DRAWR 100,100:DRAWR 100, −100:DRAWR −200,0 et comparez les résultats...

## REPRODUCTION DE FIGURES EN BASSE RÉSOLUTION

Il est assez facile de reproduire un dessin quelconque par une suite de lettres de l'alphabet. Nous prenons ici l'exemple de TITI. Les zones les plus claires sont représentées par la lettre "O", les zones sombres par la lettre "B" et les zones intermédiaires par la lettre "P".

En particulier, le contour du dessin est réalisé par la lettre B.

Le programme consiste en une suite de "PRINT#8,Ligne".

```
10 REM TITI
20 :
40 PRINT#8,SPC(36)"O"
50 PRINT#8,SPC(30)"O     O"
60 PRINT#8,SPC(26)"O   O     O"
70 PRINT#8,SPC(27)"O   O     O"
80 PRINT#8,SPC(27)"O   O     O"
90 PRINT#8,SPC(24)"BPPPPPPPPPPPPPPB"
100 PRINT#8,SPC(19)"BPPPPPPPPPPPPPPPPPPPPPPPPB"
110 PRINT#8,SPC(16)"BPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPB"
120 PRINT#8,SPC(13)"BPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPB"
130 PRINT#8,SPC(11)"BPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPB"
140 PRINT#8,SPC(9)"BPPPPBPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPBPPPB"
150 PRINT#8,SPC(8)"BPPPBPPPPBPPPPPPPPPPPPPPPPPP
PPPPPPPPPPBPPPPPBPPB"
160 PRINT#8,"        BPPBPPPBPPPPPPPPPP PPPPPPPP
PPPPPPPPPPPPPPPBPPPPBPB"
170 PRINT#8,"       BPPBPPBPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPBPPPBPB"
180 PRINT#8,"      BPPPPPBPPPPBPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPBPPPPPB"
190 PRINT#8,"     BPPPPPBPPBPBPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPBPPPBPPPPPBP"
200 PRINT#8,"     PPPPPPPPPPBPPBPPPPPPPPPPPPPPP
PPPPPPPPPPPPPBPBPPPPPPPPPP"
210 PRINT#8,"    BPPPPPPPPBPBPPBPPPPPPPPPPPPPPP
PPPPPPPPPPPPBPBPPPBPPPPPPB"
220 PRINT#8,"    BPPPPPPPPPBPPBPPBPPPPPPPPPPPPP
PPPPPPPPPPPPPBPBPPPPPPPPPB"
230 PRINT#8,"   BPPPPPPPPPBPPBPPBPPPPPPPPPPPPP
PPPPPPPPPPBPBPPPBPPPPPPPB"
240 PRINT#8,"   BPPPPPPPPPBPPB     PPPPPPPPPPP
PPPPPPP    BPPBPPPPPPPPPPB"
250 PRINT#8,"    PPPPPPPPPPB         PPPPPPPPP
PPPPP       BPPPPPPPPPPP"
260 PRINT#8,"    BPPPPPPPPPP         PPPPPPPPP
PPPP        PPPPPPPPPPB"
270 PRINT#8,"     BPPPPPPPP          PPPPPPPPP
PPP         PPPPPPPPPPS"
280 PRINT#8,"      BPPPPPPP           PPPPPPPP
PPP          PPPPPPPPPB"
```

→

```
290 PRINT#8,"        BPPPPPPPP          PPPPPPPPP
PPP              PPPPPPPPPB"
300 PRINT#8,"        BPPPPPPP     OO    PPPPPPPPP
PP     OO     PPPPPPPPPB"
310 PRINT#8,"         BPPPPPP    OOOO   PPPPPPPPP
PP    OOOO    PPPPPPPB"
320 PRINT#8,"          BPPPPP   OOOOO  PPPPPPPPP
PP   OOOOO    PPPPPPB"
330 PRINT#8,"           BPPPPP  OOOOO  PPPPPPPPP
PP   OBBOO PPPPPPPB"
340 PRINT#8,"           BPPPPP  OOOOO  PPPPPPPPP
PP   OBBOO PPPPPPPB"
350 PRINT#8,"            BPPPPP OOBBO  PPPPPPPPP
PP   OBBOO PPPPPPB"
360 PRINT#8,"             BPPPP OOBBB  PPPPPPPPP
PP   BBBOO PPPPPB"
370 PRINT#8,"              BPPPPP OOBB PPPPPPPPPP
PPPBBOO PPPPPPPB"
380 PRINT#8,"              BPPPPPPPPPOBPPPPPPPPPPPP
PPPBOOPPPPPPPPPPB"
390 PRINT#8,"             BPPPPPPPPPPPPPPPP
  PPPPPPPPPPPPPPPB"
400 PRINT#8,"             BPPPPPPPPPPPPPBPP
  BPPPPPPPPPPPPPPB"
410 PRINT#8,"             BPPPPPPPPPPPPBPP
   PPBPPPPPPPPPPPB"
420 PRINT#8,"               BPPPPPPPPPBPPPP
 PPPPPBPPPPPPPPS"
430 PRINT#8,"                   BPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPB"
440 PRINT#8,SPC(26)"BDPPPPPPPDB"
450 PRINT#8,SPC(27)"BPPPPPPPB"
460 PRINT#8,SPC(28)"BPPPPPB"
470 PRINT#8,SPC(29)"PPPPP"
480 PRINT#8,SPC(29)"BPPPB"
490 PRINT#8,SPC(29)"BPPPB"
500 PRINT#8,SPC(26)"BBPBPPPBB"
510 PRINT#8,SPC(25)"BPPPBPPPPBB"
520 PRINT#8,SPC(25)"BPPPPPPPPPB"
530 PRINT#8,SPC(24)"BPPPPPPPPPPBB"
540 PRINT#8,SPC(23)"BPPBPPPPPPPPBPB"
550 PRINT#8,SPC(22)"BPPPBPPPPPPPPPBPPB"
560 PRINT#8,SPC(21)"BPPPPPPPPPPPPPPPBPPPB"
570 PRINT#8,SPC(20)"BPPPPBPPPPPPPPPPPBBPPB"
580 PRINT#8,SPC(21)"BPPPPBPPPPPPPPPPPPPBPB"
590 PRINT#8,SPC(21)"BPPPBPPPPPPPPPPPPPPB"
600 PRINT#8,SPC(21)"BPPBPPPPPPPPPPPPPPPB"
610 PRINT#8,SPC(22)"BBPPPPPPPPPPPPPPPPB"
620 PRINT#8,SPC(22)"BPPPPPPPPPPPPPPPPPB"
```

```
630 PRINT#8,SPC(21)"BBPPPPPPPPPPPPPPPPPPB"
640 PRINT#8,SPC(20)"BBPPPPPPPPPPPPPPPPPPB"

650 PRINT#8,SPC(20)"BBPPPPPPPPPPPPPPPPPPB"
660 PRINT#8,SPC(20)"BBPPPPPPPPPPPPPPPPPB"
670 PRINT#8,SPC(20)"BBPPPPPPPPPPPPPPPPB"
680 PRINT#8,SPC(20)"BBBPPPPPPPPPPPBPPPB"
690 PRINT#8,SPC(22)"BBPPPPB    BPPB"
700 PRINT#8,SPC(25)"BPPB    BPPPB          BBB
   BBB"
710 PRINT#8,"     BBB                BPPB   BPPB
PPPBBBPPPPPPPPPPPPPPPPPFBB"
720 PRINT#8,"   BPPPPPPPBB            BPPPPPBPPP
PPPPPPPPPPPPPPPPPPPPPPBPPB"
730 PRINT#8," BPPPBPPPPPPPPPPPBBBBBBPPPPPPPPPBBP
PPPPPPPPPPPPPPPPPPPPPPPPPBPB"
740 PRINT#8,"BPPBPPPPPPPPPPPPPPPPPPPPPPPPPPPB
 BBPPPPPPPPPPPPPPPPPPPPPPPS"
750 PRINT#8,"BPBPPPPPPPPPPPPPPPPPPPPPPPPPB
     B  PPPPPPPPPPPPPPPPPPPPB"
760 PRINT#8," BPPPPPPPPPPPPPPPPPPPPPPPPPB
          BBPPPPPPPPPPPPPPPB"
770 PRINT#8,"BPPPPPPPPPPPPPPPPPPPPPPPB
               BBBBBBB"
780 PRINT#8,"  BPPPPPPPPPPPPPPPPPB"
790 PRINT#8,"     BBBBBBBBBB"
```

# Chapitre 5 | Utilitaires d'ordre général

## ÉDITEUR DE LIGNES

L'Amstrad possédant un MODE 80 colonnes, il peut être intéressant de saisir du texte (pour écrire un livre par exemple...) dans le but de le reproduire sur imprimante. Le programme "Editeur de Lignes" vous permet de saisir du texte sur 80 colonnes, ligne par ligne, d'insérer ou de supprimer une ligne, de lister tout ou partie des lignes entrées et enfin de sortir le document sur imprimante.

***Programme***

```
10 REM Editeur de lignes
11 :
20 MODE 2 : REM 80 Colonnes
30 CLS
40 :
50 I=0:DIM A$(500):REM 500 Lignes
60 I=I+1:IF I>MX THEN MX=I
70 PRINT I;:INPUT A$(I)
80 IF LEN(A$(I))=0 THEN 110 : REM Commande

90 GOTO 60
100 :
110 REM Commande
120 :
130 PRINT:PRINT"L)iste sur ecran"
140 PRINT"S)upprime une ligne"
150 PRINT"I)nsere une ligne"
160 PRINT"eD)ite"
```

→

```
165 PRINT"E)crit sur imprimante"
170 INPUT"Votre choix";C$
180 IF C$="L" OR C$="S" OR C$="I" OR C$="E" OR
C$="D" THEN 200
190 PRINT"Commande inconnue":GOTO 170
200 :
210 IF C$="L" THEN GOSUB 1000
220 IF C$="S" THEN GOSUB 2000
230 IF C$="I" THEN GOSUB 3000
240 IF C$="D" THEN GOSUB 4000
250 IF C$="E" THEN GOSUB 5000
260 :
270 GOTO 110
280 REM ***************************
1000 REM Listage Ecran
1001 :
1010 INPUT"de";DE:INPUT"a";A
1020 :
1030 FOR I=DE TO A
1040   PRINT I;A$(I)
1050 NEXT I
1060 :
1070 RETURN
1080 REM ***************************
2000 REM Supression de ligne
2001 :
2010 INPUT"Ligne a supprimer";L
2015 IF L>MX OR L=0 THEN 2130
2020 PRINT L;A$(L)
2030 INPUT"Etes-vous sur (O/N)";R$
2040 IF R$<>"O" THEN 2130
2050 :
2060 REM Supression
2080 FOR I=L TO MX
2090   A$(I)=A$(I+1)
2100 NEXT I
2110 A$(MX)="":MX=MX-1
2120 :
2130 RETURN
2140 ***************************
3000 REM Insertion de ligne
3001 :
3010 INPUT"Insertion apres la ligne";L
3020 IF L>MX OR L=0 THEN 3110
3030 PRINT L+1;:INPUT A$
3040 :
3050 REM Insertion
3060 FOR I=MX TO L+2 STEP -1
3070   A$(I)=A$(I-1)
```

```
3080 NEXT I
3090 A$(L+1)=A$:MX=MX+1
3100 :
3110 RETURN
3120 REM ************************
4000 REM Retour a l'edition
4001 :
4010 INPUT"Ligne";L
4020 IF L>=MX THEN 4050
4030 PRINT"Ligne occupee":INPUT"Etes-vous sur (
O/N)";R$
4040 IF R$="N" THEN 4070
4050 E=1:I=L:GOTO 70
4060 :
4070 RETURN
4080 REM ************************
5000 REM Impression
5001 :
5010 INPUT"Impression a partir de";DE
5020 INPUT"jusqu'a";A
5030 IF A>MX THEN A=MX
5040 IF DE=0 THEN DE=1
5050 FOR I=DE TO A
5060   PRINT #8,A$(I)
5070 NEXT I
5080 :
5090 RETURN
```

***Analyse du programme***

| | |
|---|---|
| Ligne 20 | : Passage en MODE 80 colonnes. |
| Lignes 50 à 90 | : Programme principal. |
| Lignes 130 à 270 | : Menu général. |
| Lignes 1000 à 1070 | : Listage du texte sur écran. |
| Lignes 2000 à 2130 | : Suppression de ligne dans le texte. |
| Lignes 3000 à 3110 | : Insertion de ligne dans le texte. |
| Lignes 4000 à 4070 | : Edition écran. |
| Lignes 5000 à 5090 | : Impression écran. |

## CLASSEMENT DE NOMBRES ET DE CHAÎNES

Dans votre vie d'informaticien, vous serez sans doute amené à vouloir classer des données numériques (ou alphanumériques) par ordre croissant ou décroissant (respectivement alphabétique ou inverse alphabétique).

Le langage BASIC fait la différence entre caractères et nombres, c'est ainsi que deux programmes sont proposés :

- un pour le classement de chaînes de caractères,
- un pour le classement de nombres.

Il existe un nombre impressionnant de méthodes de classement par ordre croissant ou décroissant. Celle qui a été retenue ici offre deux avantages : sa simplicité et sa rapidité. Le programme reprend la méthode dite des "inversions" : un tableau contient les entités à classer. La méthode consiste à balayer le tableau en inversant deux entités successives a et b si elles ne répondent pas à la condition "a>b" ou "a<b" pour un classement décroissant, respectivement croissant.

Le tableau sera balayé jusqu'à ce qu'il n'y ait plus aucune inversion dans un balayage complet.

Classement de nombres :

```
10 REM Classement de Nombres
11 :
20 DIM T(100):REM 100 Donnees Max
100 REM Acquisition des donnees
101 :
110 CLS:PRINT"        CLASSEMENT DE NOMBRES"
120 PRINT:PRINT"Entrez les donnees a classer"
130 PRINT"La saisie se terminera par 9999"
140 :
150 I=1:PRINT
160 PRINT "Donnee";I;:INPUT T(I)
170 IF T(I)<>9999 THEN I=I+1:GOTO 160
180 :
200 REM Classement
201 :
210 CLS
220 PRINT"Voulez-vous un classement croissant"
230 INPUT"ou decroissant(C/D)";R$
240 IF R$<>"C" AND R$<>"D" THEN 200
250 :
260 IN=0:REM Indicateur d'inversion
270 FOR J=1 TO I-1
280   C=T(J):D=T(J+1)
290   IF R$="C" AND C>D THEN T(J)=D:T(J+1)=C:IN
=1
300   IF R$="D" AND C<D THEN T(J)=D:T(J+1)=C:IN
=1
310 NEXT J
320 IF IN=1 THEN 260
330 :
```

```
400 REM Affichage des resultats
401 :
410 CLS:PRINT"          DONNEES CLASSEES":PRINT
420 FOR J=1 TO I
430   A=T(J)
440   IF A<>9999 THEN PRINT A
450 NEXT J
```

Classement de chaînes :

```
10 REM Classement de Chaines
11 :
20 DIM T$(100):REM 100 Donnees Max
100 REM Acquisition des donnees
101 :
110 CLS:PRINT"        CLASSEMENT DE CHAINES"
120 PRINT:PRINT"Entrez les donnees a classer"
130 PRINT"La saisie se terminera par 9999"
140 :
150 I=1:PRINT
160 PRINT "Donnee";I;:INPUT T$(I)
170 IF T$(I)<>"9999" THEN I=I+1:GOTO 160
180 :
200 REM Classement
201 :
210 CLS
220 PRINT"Voulez-vous un classement croissant"
230 INPUT"ou decroissant(C/D)";R$
240 IF R$<>"C" AND R$<>"D" THEN 200
250 :
260 IN=0:REM Indicateur d'inversion
270 FOR J=1 TO I-1
280   C$=T$(J):D$=T$(J+1)
290   IF R$="C" AND C$>D$ THEN T$(J)=D$:T$(J+1)
=C$:IN=1
300   IF R$="D" AND C$<D$ THEN T$(J)=D$:T$(J+1)
=C$:IN=1
310 NEXT J
320 IF IN=1 THEN 260
330 :
400 REM Affichage des resultats
401 :
410 CLS:PRINT"          DONNEES CLASSEES":PRINT
420 FOR J=1 TO I
430   A$=T$(J)
440   IF A$<>"9999" THEN PRINT A$
450 NEXT J
```

***Analyse du programme***

Lignes 100 à 180 : Acquisition des données à classer.
Lignes 200 à 330 : Classement.
Lignes 400 à 450 : Affichage des résultats.

**Remarque :** Les programmes de classement de chaînes et de nombres sont différents. En effet, si nous ne faisions qu'un seul programme de classement, il opérerait sur les chaînes, puisque les chaînes peuvent aussi bien contenir de l'alphanumérique que du numérique. Malheureusement, la chaîne "21" est inférieure à la chaîne "3" d'un point de vue alphanumérique, ce qui est faux d'un point de vue numérique (on a en effet 21>3 !).

On est donc contraint de faire deux programmes de classement.

## HORLOGES

Comme tout bon micro-ordinateur familial, l'Amstrad possède une instruction BASIC (ici TIME) qui donne le temps écoulé depuis la mise sous tension de l'appareil (ici en 1/300 sec.). Pour constituer une horloge, il suffit donc d'initialiser le temps à la valeur courante, et de compter le nombre de secondes écoulées depuis ce moment.

Cette opération est faisable sur tout ordinateur possédant une instruction TIME (ou équivalent). Le programme qui suit adopte cette démarche.

***Programme***

```
100 REM Horloge 1
110 :
120 CLS:PRINT"Entrez l'heure sous la forme : HH
/MM/SS"
130 PRINT:PRINT:INPUT H$
140 PRINT:INPUT"Appuyez sur <ENTER> pour valide
r";A$
150 :
160 H=VAL(LEFT$(H$,2))
170 M=VAL(MID$(H$,4,2))
180 S=VAL(RIGHT$(H$,2))
190 :
200 T0=TIME:CLS
210 :
220 V1=(TIME-T0)/300
225 IF V1>=1 THEN T0=TIME
230 IF V1>=1 THEN S=S+1:IF S=60 THEN S=0:M=M+1:
IF M=60 THEN M=0:H=H+1:IF H=24 THEN H=0
240 LOCATE 10,10:PRINT H"/"M"/"S"  "
250 GOTO 220
```

### *Analyse du programme*

Lignes 120 à 130 : Lecture de l'heure courante.
Ligne 140 : Validation de l'heure entrée.
Lignes 160 à 180 : Conversion de la chaîne H$ en HH/MM/SS.
Lignes 220 à 250 : Calcul du temps écoulé et affichage.

Peut-être êtes-vous déçu par la précision de cette horloge ? Rassurez-vous. L'Amstrad possède une instruction qui lui permet de se différencier des autres micro-ordinateurs. Il s'agit de l'instruction EVERY a,b qui permet d'effectuer un débranchement de la ligne courante chaque a × 0.002 m/sec. Si nous prenons a=50, le débranchement se fera chaque (50 × 0.02 = 1 sec.) seconde. L'écriture du programme est donc grandement simplifiée, et la précision est remarquable.

### *Programme*

```
100 REM Horloge 2
110 :
120 CLS:PRINT"Entrez l'heure sous la forme : HH
/MM/SS"
130 PRINT:PRINT:INPUT H$
140 PRINT:INPUT"Appuyez sur <ENTER> pour valide
r";A$
150 :
160 H=VAL(LEFT$(H$,2))
170 M=VAL(MID$(H$,4,2))
180 S=VAL(RIGHT$(H$,2))
185 CLS
190 :
200 EVERY 50,0 GOSUB 300
210 :
220 GOTO 220
230 REM ****************************************
300 REM Procedure de calcul et d'affichage de l
'heure
310 :
320 S=S+1:IF S=60 THEN S=0:M=M+1:IF M=60 THEN M
=0:H=H+1:IF H=24 THEN H=0
330 LOCATE 10,10:PRINT H"/"M"/"S"  "
340 :
350 RETURN
```

### *Analyse du programme*

Lignes 120 à 130 : Initialisation du temps courant.
Lignes 160 à 180 : Conversion de la chaîne H$ en HH/MM/SS.
Ligne 200 : EVERY.
Lignes 300 à 350 : Calcul et affichage de l'heure.

# AFFICHAGE PERSONNALISÉ

Pour donner une touche personnelle à l'affichage de titres ou de messages sur l'écran, voici un utilitaire dans lequel vous choisissez la vitesse d'affichage du message et l'occurrence d'affichage des lettres (aléatoire ou linéaire).

### *Exemple d'affichage*

```
10 A$="TEST DU PROGRAMME MESSAGE SONORE"
20 TV=0:V=90:CLS:GOSUB 10000
30 END
```

### *Programme*

```
10000 REM ************************************
**
10010 REM *
 *
10020 REM *      AFFICHAGE       PERSONNALISE
 *
10030 REM *
 *
10040 REM ************************************
**
10050 REM *
 *
10060 REM * Entree : TV=Vitesse d'affichage fix
e*
10070 REM *           ou aleatoire (TV=0 ou TV=1
)*
10080 REM *           V =Vitesse d'affichage
 *
10090 REM *           entre 1 et 100
 *
10100 REM *
 *
10110 REM ************************************
**
10120 :
10130 FOR I=1 TO LEN(A$)
10140    IF TV=1 THEN TEMPS=(100-V)*4 ELSE TEMPS
=(100-V)*RND*15
10150    B$=MID$(A$,I,1) 'Lettre a afficher
10160    FOR J=1 TO TEMPS:NEXT J 'Pause
10170    PRINT B$; 'Affichage
10180    SOUND 1,80,4,4 'Bip
10190 NEXT I
10200 RETURN
```

### *Analyse du programme*

Ligne 10140 : Calcul du temps passé entre deux lettres.
Ligne 10150 : Sélection de la lettre à afficher.
Ligne 10160 : Pause.
Ligne 10170 : Affichage de la lettre.
Ligne 10180 : Bip sonore.

## UTILITAIRES D'ÉCRAN

### INPUT borné

Ce programme permet d'effectuer un "INPUT" numérique entier ou réel, compris entre deux bornes définies à l'avance. L'INPUT est refusé tant qu'il n'est pas compris entre les deux bornes.

L'appel au programme se fait par "GOSUB 10000".

Au préalable,

- Stocker dans L1 la valeur minimale permise,
- Stocker dans L2 la valeur maximale permise,
- Poser la question nécessitant l'INPUT borné sous la forme PRINT "texte" ; (ligne 25 dans l'exemple).

### *Exemple*

```
10 L1=-10:L2=100 'Valeurs limites
20 CLS:PRINT"Veuillez entrer un nombre"
30 PRINT"compris entre -10 et 100";
40 GOSUB 10000
50 END
```

### *Programme*

```
10000 REM ****************************************
10010 REM *                                        *
10020 REM *     REPONSE  NUMERIQUE  BORNEE         *
10030 REM *                                        *
10040 REM ****************************************
10050 REM *                                        *
10060 REM * Entree : L1 = Minimum                  *
10070 REM *          L2 = Maximum                  *
10080 REM * Sortie   N  = Nombre lu                *
10090 REM *                                        *
10100 REM ****************************************
10110 :
10120 H=POS(#0):V=VPOS(#0) 'Position du curseur
10130 :
10140 LOCATE H,V:PRINT SPACE$(15)
10150 LOCATE H,V:INPUT N
10160 IF N>L2 OR N<L1 THEN 10140
10170 :
10180 RETURN
```

***Analyse du programme***

Ligne 10120 : Lecture et mémorisation de la position du curseur.
Ligne 10140 : Positionnement du curseur.
Ligne 10160 : Test si le nombre entré est compris entre le min. et le max.

## Saisie de réponses prédéfinies

Cet utilitaire permet de faciliter la saisie clavier de questions pour lesquelles le nombre de réponses est fini et restreint.

Une touche quelconque permet de changer de réponse. La touche ENTER valide la réponse choisie.

L'appel au programme se fait par "GOSUB 10000".

Au préalable, mettre dans N le nombre de réponses possibles, dans A$ le libellé des réponses possibles. Poser la question sous la forme PRINT "texte" ; (ligne 30 dans l'exemple).

***Exemple***

```
10 N=3 'Nombre de valeurs
20 A$(1)="Oui":A$(2)="Non":A$(3)="Peut-etre" 'V
aleurs possibles
30 CLS:PRINT"Etes vous d'accord ? ";
40 GOSUB 10000 'Gestion de la reponse
50 LOCATE 1,10:PRINT A$(S)" est votre reponse"
60 END
```

***Programme***

```
10000 REM ****************************************
10010 REM *                                      *
10020 REM * INPUT AVEC REPONSES PREDEFINIES      *
10030 REM *                                      *
10040 REM ****************************************
10050 REM *                                      *
10060 REM * Entree : N = Nb de valeurs possibles*
10070 REM *          A$= Valeurs possibles       *
10080 REM * Sortie : S = Valeur choisie           *
10090 REM *                                      *
10100 REM ****************************************
10110 :
10120 REM Calcul de la longueur maximale de la
reponse
10130 L=0 'Initialisation
10140 FOR I=1 TO N
```

```
10150    A=LEN(A$(I))
10160    IF A>L THEN L=A
10170 NEXT I
10180 :
10190 REM Initialisation
10200 :
10210 H=POS(#0):V=VPOS(#0)
10220 S=1:PRINT A$(S)
10230 :
10240 REM Saisie de la reponse
10250 :
10260 A$=INKEY$:IF A$="" THEN 10260
10270 IF ASC(A$)=13 THEN 10350 'ENTER
10280 :
10290 REM Passage a la reponse suivante
10300 :
10310 S=S+1:IF S>N THEN S=1
10320 LOCATE H,V:PRINT A$(S);SPACE$(L-LEN(A$(S)
))
10330 GOTO 10260 'Boucle de lecture clavier
10340 :
10350 REM ENTER
10360 :
10370 RETURN
```

***Analyse du programme***

Lignes 10120 à 10170 : Calcul de la longueur maximale de la réponse.
Ligne 10210 : Lecture et mémorisation de la position curseur.
Ligne 10260 : Lecture du clavier.
Lignes 10290 à 10330 : Passage à la réponse suivante.

## Effacement d'écran

Ce programme permet d'effacer (remplir de blancs) tout ou partie de l'écran. L'effacement est réalisé à partir d'une position X,Y du curseur, et sur une longueur de N caractères. H et V contiennent respectivement l'abscisse et l'ordonnée de début de zone à effacer. N contient le nombre de caractères à effacer.

***Exemple***

```
10 H=1:V=10 'Position de depart d'effacement
20 N=40 'Nombre de caracteres a effacer
30 GOSUB 10000 'Effacement
40 END
```

### *Programme*

```
10000 REM ***************************************
10010 REM *                                       *
10020 REM * EFFACEMENT D'UNE PARTIE DE L'ECRAN    *
10030 REM *                                       *
10040 REM ***************************************
10050 REM *                                       *
10060 REM * Entree : H=Position horizontale de    *
10070 REM *          debut d'effacement           *
10080 REM *          V=Position verticale de      *
10090 REM *          debut d'effacement           *
10100 REM *          N=Nombre de caractere(s)     *
10110 REM *          a effacer                    *
10120 REM *                                       *
10130 REM ***************************************
10140 :
10150 LOCATE H,V
10160 IF N<255 THEN 10180
10170 PRINT SPACE$(255):N=N-255:GOTO 10160
10180 PRINT SPACE$(N)
10190 RETURN
```

### *Analyse de programme*

Ligne 10150 : Positionnement du curseur en début de zone.
Ligne 10170 : Effacement modulo 255.
Ligne 10180 : Effacement résiduel.

## Éditeur de masques d'écran

Pour faciliter la création d'"écrans de saisie", voici un utilitaire qui permet de fabriquer des "fichiers-écrans de saisie". Ces fichiers sont constitués d'un certain nombre de rubriques positionnées selon votre choix sur l'écran.

### *Programme*

```
100 REM Saisie de masques d'ecran
110 :
120 MODE 1:INK 1,1:PAPER 0:BORDER 0:PEN 2:CLS
130 PRINT SPACE$(5);:PAPER 3:PRINT"SAISIE DE MA
SQUES D'ECRAN":PAPER 0
140 LOCATE 1,5
150 INPUT"Nombre de rubriques :";NR
160 DIM A$(NR),C(NR),L(NR)
170 PRINT:INPUT"Titre du masque :";T$
```

```
180 :
190 FOR I=1 TO NR
200   CLS
210   PAPER 3:PRINT"RUBRIQUE NO";I:PAPER 0
220   LOCATE 1,5:INPUT"Libelle : ";A$(I)
230   PRINT:INPUT"Colonne";C(I)
240   PRINT:INPUT"Ligne  ";L(I)
250 NEXT I
260 :
270 CLS:INPUT"Nom du fichier masque : ";N$
280 OPENOUT N$
290 PRINT#9,NR 'Nombre de rubriques
300 PRINT#9,T$ 'Titre du masque
310 FOR I=1 TO NR
320   PRINT#9,A$(I) 'Rubrique
330   PRINT#9,L(I)  'Ligne d'affichage
340   PRINT#9,C(I)  'Colonne d'affichage
350 NEXT I
360 CLOSEOUT
```

***Analyse du programme***

| | |
|---|---|
| Lignes 100 à 140 | : Présentation. |
| Ligne 150 | : Nombre de rubriques dans le masque. |
| Ligne 170 | : Titre du masque. |
| Lignes 190 à 250 | : Saisie des positions et libellés des rubriques. |
| Lignes 280 à 360 | : Ecriture du fichier masque. |

## Exploitation des fichiers masques d'écran

Cet utilitaire permet d'exploiter les masques d'écran créés par le programme précédent.

Le nom du fichier est inscrit dans la variable N$, et l'utilitaire est appelé par "GOSUB 10000".

Les flèches "vers le bas" et "vers le haut" gèrent le déplacement parmi les rubriques. La rubrique sélectionnée apparaît en inverse vidéo. L'appui sur la touche ENTER valide le choix de la rubrique.

***Exemple***

```
10 N$="MASQUE":GOSUB 10000
20 PRINT"Rubrique choisie :";P
30 END
```

***Programme***

```
10000 REM ************************************
**
10010 REM *
 *
10020 REM *     AFFICHAGE ET GESTION DE MASQUES
 *
10030 REM *
 *
10040 REM ************************************
**
10050 REM *
 *
10060 REM * Entree : N$ = Nom du fichier masque
 *
10070 REM * Sortie : P  = Choix de la rubrique
 *
10080 REM *
 *
10090 REM ************************************
**
10100 :
10110 REM Lecture du masque
10120 :
10130 OPENIN N$
10140 INPUT#9,NR 'Nombre de rubriques
10150 DIM A$(NR),C(NR),L(NR)
10160 INPUT#9,T$ 'Titre
10170 FOR I=1 TO NR
10180   INPUT#9,A$(I)
10190   INPUT#9,L(I)
10200   INPUT#9,C(I)
10210 NEXT I
10220 CLOSEIN
10230 :
10240 REM Affichage du masque
10250 :
10260 L=LEN(T$):L=(40-L)/2:CLS
10270 INK 0,0:PEN 2:BORDER 0
10280 PAPER 3:LOCATE L,1:PRINT T$:PAPER 0
10290 FOR I=1 TO NR
10300   LOCATE C(I),L(I)
10310   PRINT A$(I)
10320 NEXT I
10330 :
10340 REM Gestion du masque
10350 :
```

```
10360 LOCATE C(1),L(1):PAPER 3:PRINT A$(1):PAPE
R 0:P=1
10370 A$=INKEY$:IF A$="" THEN 10370
10380 A=ASC(A$)
10390 IF A<>240 AND A<>241 AND A<>13 THEN 10370
10400 LOCATE C(P),L(P):PRINT A$(P)
10410 IF A=241 THEN 10490
10420 IF A=13 THEN 10550
10430 :
10440 REM Vers le haut
10450 :
10460 P=P-1:IF P=0 THEN P=NR
10470 GOTO 10530
10480 :
10490 REM Vers le bas
10500 :
10510 P=P+1:IF P>NR THEN P=1
10520 :
10530 LOCATE C(P),L(P):PAPER 3:PRINT A$(P):PAPE
R 0:GOTO 10370
10540 :
10550 REM RETURN
10560 CLS
10570 RETURN
```

***Analyse du programme***

Lignes 10130 à 10220 : Lecture du masque N$.
Lignes 10240 à 10320 : Affichage du masque.
Lignes 10340 à 10570 : Gestion du masque.
Lignes 10370 à 10420 : Gestion du clavier.
Lignes 10440 à 10470 : Déplacement vers le haut.
Lignes 10490 à 10530 : Déplacement vers le bas.

## Saisie formatée

Peut-être vous êtes-vous déjà posé le problème suivant : est-il possible de saisir un nombre réel avec N chiffres avant la décimale et P chiffres après ? Cet utilitaire permet de le faire. Après avoir choisi le nombre de chiffres avant et après la virgule dans N et P, le masque de saisie est affiché sous la forme : ##..##. ##..##
N P

Le point décimal est géré automatiquement. La touche ENTER valide l'entrée du nombre. Le nombre stocké est entré dans la variable NL.

### *Exemple*

```
10 N=2:P=4
20 CLS:PRINT"Entrez un nombre reel :";
30 GOSUB 10000
40 END
```

### *Programme*

```
10000 REM ************************************
**
10010 REM *
 *
10020 REM *       SAISIE NUMERIQUE FORMATEE
 *
10030 REM *
 *
10040 REM ************************************
**
10050 REM *
 *
10060 REM * Entree : N=Nombre de chiffres
 *
10070 REM *           avant le point decimal
 *
10080 REM *           P=Nombre de chiffres
 *
10090 REM *           apres le point decimal
 *
10100 REM * Sortie : NL=nombre lu
 *
10110 REM *
 *
10120 REM ************************************
**
10130 :
10140 REM Visualisation de l'entree
10150 :
10160 H=POS(#0):V=VPOS(#0)
10170 LOCATE H,V+1
10180 IF N>0 THEN FOR I=1 TO N:PRINT"#";:NEXT I
10190 IF P<>0 THEN PRINT".";
10200 IF P>0 THEN FOR I=1 TO P:PRINT"#";:NEXT I
10210 LOCATE H,V
10220 :
10230 AV=1:I=0 'Initialisation
10240 A$=INKEY$:IF A$="" THEN 10240
```

```
10250 IF (A$>"9" OR A$<"0") AND A$<>"." AND ASC
(A$)<>13 AND A$<>"-" THEN PRINT CHR$(7):GOTO 10
240
10260 I=I+1 'Nombre de caracteres entres
10270 IF A$="." THEN AV=0:I=0:PRINT".";:B$=B$+A
$:GOTO 10350
10280 IF ASC(A$)=13 THEN 10360
10290 IF AV=1 AND I<=N THEN B$=B$+A$:PRINT A$;

10300 IF AV=1 AND I>=N AND P=0 THEN 10360
10310 IF AV=1 AND I>=N THEN B$=B$+".":PRINT".";
:AV=0:I=0:PRINT CHR$(7);:GOTO 10350
10320 IF AV=0 AND A$="-" THEN PRINT CHR$(7);:I=
I-1:GOTO 10350
10330 IF AV=0 AND I<=P THEN B$=B$+A$:PRINT A$;
10340 IF AV=0 AND I>=P THEN PRINT CHR$(7);:GOTO
 10360
10350 GOTO 10240 'Boucle de saisie
10360 NL=VAL(B$):LOCATE H,V+1:PRINT SPACE$(N+P+
1)
10370 RETURN
```

***Analyse du programme***

Lignes 10140 à 10210 : Visualisation du masque de saisie.
Lignes 10230 à 10350 : Gestion du clavier.
Ligne 10360 : Stockage du nombre entré dans NL et effacement du masque de saisie.

## Affichage programmé

Cet utilitaire permet d'afficher du texte alphanumérique sur l'écran. Le texte à afficher est inscrit dans le tableau A$. Les tableaux H et V donnent les positions horizontale et verticale du texte à afficher. Enfin, la variable N contient le nombre de libellés à afficher.

***Exemple***

```
10 REM Affichage programme
20 :
30 N=3
40 H(1)=10:V(1)=3:H(2)=15:V(2)=15:H(3)=16:V(3)=18
50 A$(1)="Premier message"
60 A$(2)="Deuxieme message"
70 A$(3)="Troisieme message"
80 CLS:GOSUB 10000
90 END
```

***Programme***

```
10000 REM ***************************************
10010 REM *                                       *
10020 REM *         AFFICHAGE  PROGRAMME          *
10030 REM *                                       *
10040 REM ***************************************
10050 REM *                                       *
10060 REM * Entree : N=Nb de donnees a afficher  *
10070 REM *           H=Positions horizontales    *
10080 REM *           V=Positions verticales      *
10090 REM *           A$=Chaines a afficher       *
10100 REM *                                       *
10110 REM ***************************************
10120 :
10130 FOR I=1 TO N
10140   LOCATE H(I),V(I)
10150   PRINT A$(I)
10160 NEXT I
10170 RETURN
```

***Analyse du programme***

Ligne 10140 : Positionnement du curseur.
Ligne 10150 : Affichage du texte.

# Compatibilité CPC 464/CPC 664

Tous les programmes proposés dans ce manuel fonctionnent indifféremment sur CPC 464 et CPC 664.

Le CPC 464 étant une sous-version du CPC 664, les logiciels présentés n'exploitent donc pas à fond les possibilités du 664.

A titre d'information, les nouvelles commandes existant dans le BASIC 1.1 du CPC 664 sont :

- Commande **FILL** pour remplir une surface quelconque.
- Commande **GRAPHICS PEN** et **GRAPHICS PAPER** pour initialiser l'encre et le papier utilisés en mode graphique.
- Le paramètre <**INK MODE**> a été rajouté dans les ordres PEN, MOVE, MOVER, PLOT, PLOTR, DRAW et DRAWR. Ce paramètre permet de réaliser un affichage normal, XOR AND ou OR avec l'affichage déja présent.
- Commande **MASK** pour tracer des lignes pointillées.
- Commande **COPYCHR$** qui renvoie le code ASCII d'un caractère lu sur l'écran.
- Commande **FRAME** permet de synchroniser le balayage de l'écran avec les apparitions de l'image sur ce dernier.
- Commande **CLEAR INPUT** pour vider le buffer clavier.
- Commande **CURSOR** pour contrôler l'affichage (non affichage) du curseur lors des commandes INKEY$ et INPUT.

# Conclusion

Pour conclure ce manuel, précisons deux points :

□ Nous venons de voir par de nombreux exemples comment créer des modules utilitaires dans le but d'augmenter les possibilités de l'Amstrad.

Cette liste d'utilitaires n'est pas exhaustive ; si vous éprouvez le besoin de créer une nouvelle fonction sur votre Amstrad, la méthode développée dans ce livre peut être fructueuse :

- définissez les entrées/sorties du module,
- structurez et hiérarchisez son traitement,

et vous verrez que la tâche est facilitée.

□ Il serait faux et prétentieux de dire que ces programmes s'adaptent sans difficultés à un micro-ordinateur différent de l'Amstrad. En particulier, les techniques développées dans les chapitres III et IV (le générateur sonore et les modes graphiques haute résolution) sont très spécifiques à l'Amstrad et difficilement adaptables à une autre machine.

Par contre, tout le chapitre II (améliorations du Basic Amstrad), qui développe des utilitaires, est facilement adaptable.

Alors, si vous possédez un autre micro-ordinateur que l'Amstrad, et si vous avez ... un peu de courage, je vous souhaite bonne chance !

# Index

# Conseils de lecture

Pour approfondir vos connaissances en BASIC Amstrad et mieux connaître le système des CPC 464 et CPC 664, P.S.I. vous propose une palette d'ouvrages utiles.

## POUR MAÎTRISER LE BASIC AMSTRAD

☐ **BASIC Amstrad, 1 – Méthodes pratiques** – Jacques Boisgontier et Bruno Césard (Éditions du P.S.I.)
**2 – Programmes** (à paraître) – Jacques Boisgontier (Éditions du P.S.I.)
Pour ceux qui ont déjà pratiqué un BASIC, voici deux ouvrages de perfectionnement au BASIC Amstrad, illustrés par de nombreux programmes-exemples.

*A paraître :*

☐ **Super générateur de caractères** – Jean-François Sehan (Éditions du P.S.I.)
L'art et la manière de dessiner des Lutins en BASIC sur Amstrad, MO5 et TO7/70, Commodore 64 et MSX.

## POUR PROGRAMMER EN ASSEMBLEUR ET MIEUX CONNAÎTRE LE SYSTÈME DES CPC 464 ET CPC 664

- **Assembleur de l'Amstrad** – Marcel Henrot (Éditions du P.S.I.)
  Une initiation complète à l'Assembleur du Z80 : un bon complément du *Clefs pour Amstrad.*

- **Clefs pour Amstrad** – Daniel Martin (Éditions du P.S.I.)
  Mémento présentant synthétiquement le jeu d'instructions du Z80, les points d'entrée des routines système, les connecteurs et brochage, etc. Le livre de chevet du programmeur sur Amstrad.

Achevé d'imprimer en septembre 1985
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : septembre 1985

N° d'impression : 508014
N° d'édition : 86595-286-1
ISBN : 2-86595-286-X

# BASIC PLUS

## 80 ROUTINES SUR AMSTRAD

Vous connaissez bien le BASIC de votre Amstrad, et, sans songer encore à l'assembleur, vous souhaitez accroître les capacités de votre CPC 464 ou 664 : "BASIC plus" vous propose 80 routines pour "muscler" votre ordinateur, 80 manières de simuler des fonctions que vous n'auriez jamais cru pouvoir utiliser.

"BASIC plus" vous dévoile les possibilités du synthétiseur de son de l'Amstrad pour programmer un morceau de musique, ou produire des effets spéciaux (jeux d'arcades, bateau, alarme sur le navire).

"Mode graphique haute résolution" vous donne accès à des instructions graphiques évoluées, et vous initie au tracé en haute résolution.

Un dernier chapitre sur l'animation graphique vous montre comment écrire des jeux d'aventure ou d'action en haute résolution.

Au-delà du BASIC Amstrad, découvrez BASIC plus !

9 782865 952861

ÉDITIONS DU P.S.I.
BP 86 - 77402 LAGNY-S/MARNE CEDEX - FRANCE

ISBN : 2 86595-286 X 100 F.F.

PHOTO : L. MAUNOURY