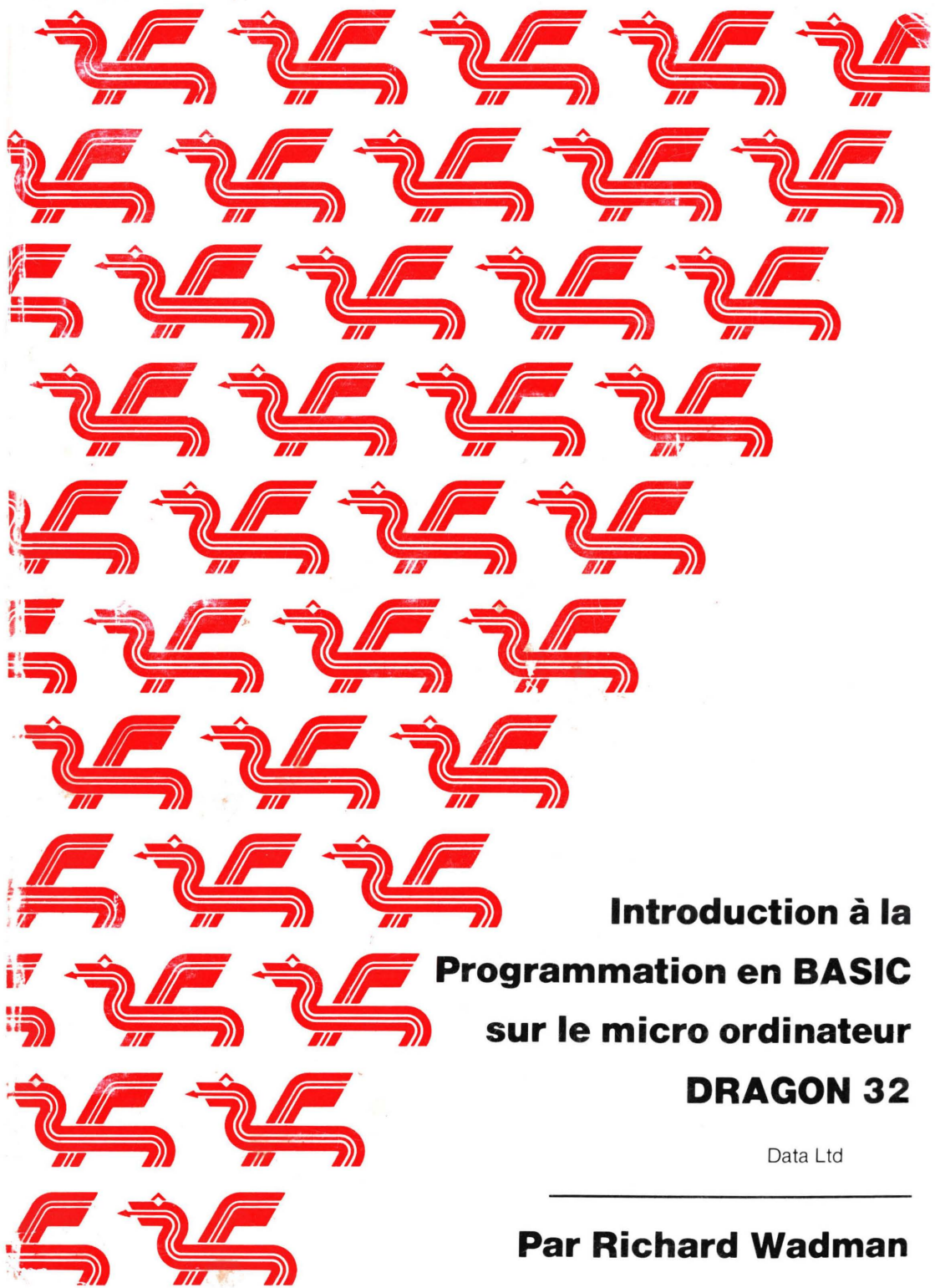


Introduction à la Programmation en BASIC sur le micro ordinateur DRAGON 32

Data Ltd

Par Richard Wadman

Introduction à la Programmation en BASIC sur le micro ordinateur DRAGON 32

Data Ltd

Par Richard Wadman

Première impression 1982

Remerciements:
L'auteur de Dragon Data Ltd souhaitent remercier le Professeur A.G. Hawkes du University College de Swansea pour sa collaboration.

MODE D'EMPLOI

Vous avez ouvert l'emballage et trouvé ce manuel. La boîte doit aussi contenir:

1. Votre ordinateur Dragon 32
2. Un bloc secteur
3. Un cable de raccordement TV.

En plus de ces trois éléments vous allez aussi avoir besoin d'un poste de télévision. Votre ordinateur Dragon 32 fonctionnera aussi bien sur un poste noir et blanc que sur un poste couleur. Cependant, pour profiter de toutes les possibilités graphiques en couleurs que vous offre votre Dragon 32 il faut le raccorder sur un poste couleur. Voilà! C'est tout ce qu'il vous faut pour mettre votre Dragon 32 en route.

Vous pouvez toutefois améliorer les capacités de votre appareil en ajoutant les options suivantes:

1. Un magnétophone à cassettes pour stocker programmes et données.
2. Une imprimante.
3. Des manettes de jeu.
4. Des lecteurs de disquettes pour un stockage massif de programmes et données.

Aucune de ces options n'est absolument nécessaire mais le magnétophone à cassettes vous évitera de devoir réécrire plusieurs fois le même programme.

LEGENDE

1. **PRISE TV**
   A accorder à un poste de télévision standard, sur la prise d'antenne ou sur la prise Péritel.

2. **BOUTON RESET**
   S'emploie pour remettre l'ordinateur à son point de départ.
   Arrête immédiatement le programme ou toute opératio d'entrée/sortie en cours. Tout programme contenu en mémoire est cependant conservé après l'usage de reset.

3. **PRISE MANETTE DE JEU,** gauche.

4. **PRISE MANETTE DE JEU,** droite.
   Pour N° 3 et N° 4, les prises DIN pentapolaires pour le raccordement des manettes de jeu sont disponibles en accessoires optionels.

5. **ENTREE/SORTIE DU MAGNETOPHONE A CASSETTES**
   Prise DIN pentapolaire pour raccordement du magnétophone à cassette. Le câble de raccord est disponible en accessoire optionel.

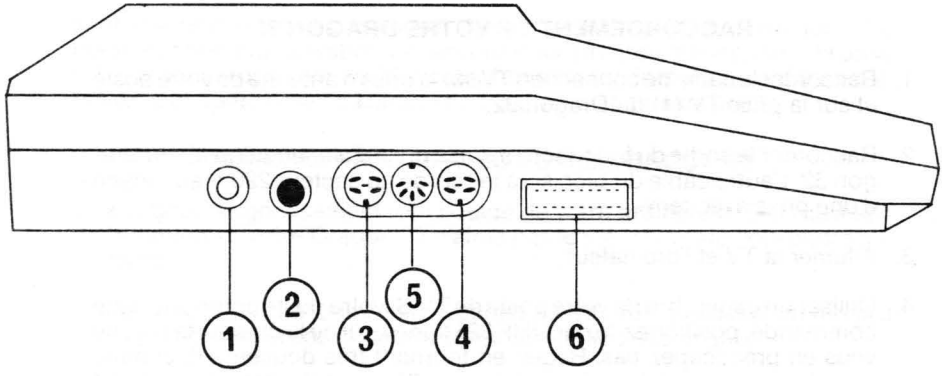
6. **PRISE POUR IMPRIMANTE PARALLELE**
   Raccordement d'une imprimante type centronics par un cable aux normes centronics.

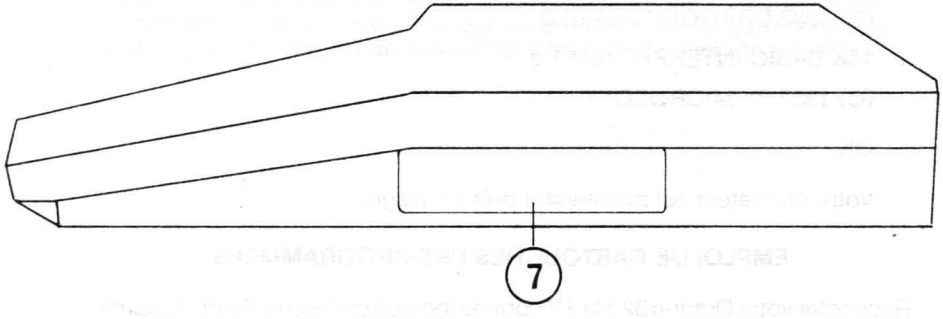
7. **FENTE D'ENGAGEMENT DES CARTOUCHES PRE-PROGRAMMES**
   S'utilise pour les cartouches de jeux. Les cartouches *doivent* être engagées uniquement lorsque l'ordinateur est débranché.

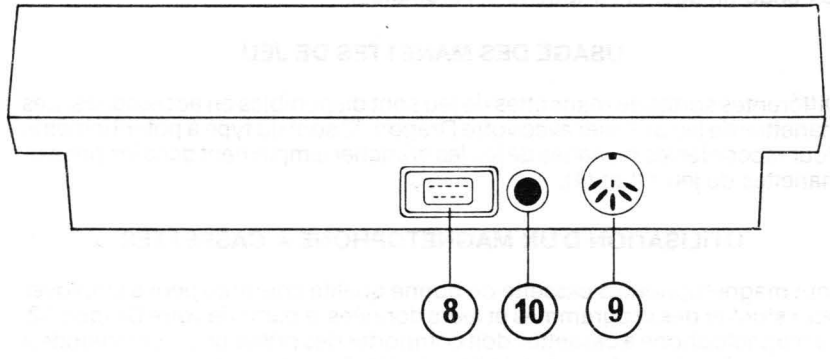
8. **ALIMENTATION ELECTRIQUE**
   Pour le raccordement du bloc secteur fourni.

9. **PRISE MONITEUR**
   Pour raccordement d'un moniteur couleur.

10. **BOUTON MARCHE/ARRET**
    Contrôle l'alimentation électrique de l'ordinateur.


1
2
3
5
4
6


7


8
10
9

RACCORDEMENT DE VOTRE DRAGON 32

1. Raccorder le cable de connection TV sur la prise d'antenne de votre poste et sur la prise TV (1) du Dragon 32.

2. Raccorder la sortie du bloc secteur sur la prise d'alimentation (8) du Dragon 32. L'autre câble du bloc sera raccordé au secteur 220 V au moyen d'une prise avec terre.

3. Allumer la TV et l'ordinateur.

4. Utiliser un canal libre de votre poste de TV. Si votre poste comporte cette commande, positionez-le sur UHF 625 lignes; s'il ne la comporte pas, ne vous en préoccupez pas. Régler en tournant très doucement, comme pour trouver une station, jusqu'à ce que l'écran présente un carré vert entouré de noir, (ou carré gris pâle entouré de noir si le poste de TV est en noir et blanc).

   Dans le carré vert le texte suivant apparaît:

   (C) 1982 DRAGON DATA LTD

   16K BASIC INTERPRETER 1.Ø

   (C) 1982 BY MICROSOFT

   OK

   Votre ordinateur est maintenant prêt à l'usage.

EMPLOI DE CARTOUCHES PRE-PROGRAMMEES

Raccorder votre Dragon 32 à la TV comme indiqué ci-dessus. *Avant* d'allumer l'ordinateur, insérer la cartouche, l'étiquette vers le haut, dans la fente (7).

Assurez vous avant d'insérer *ou* de retirer une cartouche pré-programmée que votre ordinateur est *éteint, courant coupé.* L'inobservation de ceci pourrait endommager la cartouche et l'ordinateur.

USAGE DES MANETTES DE JEU

Différentes sortes de mannettes de jeu sont disponibles en accessoires. Les manettes de jeu à utiliser avec votre Dragon 32 sont du type à potentiomètre. Pour raccorder les manettes de jeu les brancher simplement dans les prises à manettes de jeu (3) et (4).

UTILISATION D'UN MAGNETOPHONE A CASSETTES

Tout magnétophone à cassette de bonne qualité courante peut s'employer pour stocker des programmes et leurs données, à partir de votre Dragon 32. Le magnétophone à cassettes doit comporter des prises pour: commande à distance, écouteur et une entrée accessoire. Des cordons de raccordement

sont disponibles en accessoires standards pour pratiquement tous les magnétophones à cassettes. Le raccordement à l'ordinateur se fait à la prise (5). Les raccordements au Magnétophone dépendent du type d'appareil. Voyez le chapitre 4 de ce Manuel de Programmation à ce sujet.

SOINS A PRENDRE AVEC VOTRE DRAGON 32

1. Eloignez soigneusement tout liquide de votre ordinateur. Bien qu'il soit britannique, votre Dragon 32 n'aime pas le thé; ni le café non-plus d'ailleurs.

2. Assurez-vous que tous les fils et cables sont à l'abri. Si quelqu'un venait à se prendre les pieds dedans, cela pourrait faire des dégâts.

3. Assurez-vous que toutes les prises sont fermement assujetties dans leur logement *avant* d'allumer l'appareil.

4. Arrêtez tout et débranchez l'appareil quand il n'est pas en service.

5. Pour nettoyer le bati et le clavier, d'abord débrancher l'appareil de son alimentation. Utiliser un chiffon légèrement humide, essuyez le bati et le clavier. N'employez *pas* de nettoyants à base d'alcool ou de solvants.

TABLE DES MATIERES

CHAPITRE

CHAPITRE

INTRODUCTION

Ce livre a été conçu pour ceux qui désirent apprendre à programmer leur ordinateur DRAGON 32 en utilisant le language BASIC.

Le langage BASIC est un language de programmation extrêmement performant et tout à la fois très facile à assimiler. Il est constitué de moins de 100 "mots", ce qui représente moins de 1% du vocabulaire d'une personne moyenne.

La programmation peut paraître ardue, à première vue, comme toute nouvelle discipline. En fait il ne s'agit que de résoudre un problème en une suite logique d'étapes. Il y a un secret, le voici: prendre son temps et s'assurer que l'on comprend bien chaque étape, avant de passer à la suivante. Ne vous faites pas de souci si vous faites des erreurs, cela fait normalement partie du processus d'apprentissage. Vous ne détraquerez jamais votre ordinateur par des erreurs de programmation; trouvez l'erreur, rectifiez là et poursuivez. Essayez vos propres idées: "que se passera-t-il si je tente ceci?" C'est une excellente méthode pour vous rendre compte de quoi l'appareil est capable. Entrez et faites tourner chaque exemple, non seulement pour voir ce qu'il fait mais aussi pourquoi il le fait.

Avec du temps et de la patience, vous verrez que votre ordinateur n'est pas qu'une machine à faire tourner les cartouches de jeu. La programmation peu devenir une activité en soi, absorbante et pleine de joies. En plus du plaisir que les résultats vous apporteront.

CHAPITRE UN

MISE EN ROUTE

Le clavier

Vous avez installé l'ordinateur selon le mode d'emploi et vous êtes prêt à démarrer. Alors, allumez. Votre écran de TV doit afficher un carré vert avec un message. (S'il ne le fait pas, éteignez et vérifiez tous les raccords et prises). Le message dépend du type d'appareil, si les lecteurs de disquettes sont raccordés, etc... Alors ne nous en préoccupons pas. La dernière ligne toutefois est toujours la même:

OK

OK, c'est le message que l'ordinateur vous envoie pour vous dire qu'il est prêt à recevoir vos instructions. Vous devez attendre le message avant de taper quoi que ce soit. Sous le OK se trouve un carré qui clignote - c'est ce qu'on appelle le curseur. Il vous montre où vous en êtes sur la ligne.

Maintenant regardez le clavier: il ressemble à une machine à écrire, avec quelques touches de plus. Et il se comporte comme une machine à écrire: appuyez sur quelques touches et vous verrez les lettres apparaître sur l'écran. Remarquez que le curseur se déplace après chaque lettre pour vous montrer où vous êtes arrivé. Si vous appuyez sur [SHIFT] et Ø (zéro) en même temps et si vous continuez à tapez, vous verrez alors que les lettres sur l'écran sont devenues vertes sur un fond noir. (Tout au long de ce livre nous utiliserons Ø pour représenter zéro, afin de le distinguer de la lettre O. L'ordinateur tient absolument à cette différence). Ces lettres sont des minuscules: a, b, c, d: MAIS elles n'apparaîtront sous cette forme que sur une IMPRIMANTE. Toutes vos instructions ou commandes à l'ordinateur doivent être en majuscules. Appuyez donc a nouveau sur [SHIFT] et Ø ensemble et tapez encore quelques lettres. Vous devriez être maintenant revenu à des caractères noirs sur fond vert.

Maintenant, essayez la touche [←]. C'est la touche de rappel arrière; lorsque vous vous en servez le curseur recule sur la ligne et la lettre se trouvant la plus proche sur sa gauche disparaît. C'est bien pratique pour rectifier les fautes de frappe: reculez jusqu'à l'erreur et re-tapez.

Pour effacer complètement l'écran appuyez sur [CLEAR]: tout le texte disparaît et le curseur remonte au coin haut gauche. Clear n'efface que l'écran, les données contenues dans l'ordinateur ne sont pas affectées.

Exercez vous quelques temps à utiliser le clavier, afin de vous accoutumer à la position des touches et à la façon de corriger les erreurs. Ensuite effacez l'écran et assurez vous que vous êtes bien en mode de lettres capitales.

LE DRAGON COMME CALCULATEUR

Votre ordinateur peut fonctionner de deux façons différentes - en commande *directe* il exécutera immédiatement vos instructions - en commande *différée* il stockera une série d'instructions et les traitera sous forme de programme. En commande directe l'ordinateur se comporte comme une calculatrice.

L'ordinateur comprend un language appelé BASIC. Et en BASIC, il y a un certain nombre de mots particuliers pour dire à l'ordinateur d'exécuter certaines choses. Par exemple: PRINT veut dire "imprimez sur l'écran ce qui va suivre" PRINT est un COMMANDEMENT.

Essayez ceci: tapez PRINT 12 + 7 puis appuyez sur la touche [ENTER]. L'écran va alors afficher

19
OK

ou encore; tapez PRINT 12 + 8/2 puis apuyez sur [ENTER]

16
OK

PRINT 12 + 7 est une INSTRUCTION

Si vous aviez fait une faute de frappe vous auriez probablement reçu le message -

? SN ERROR

Ceci veut dire 'erreur de syntaxe' et signifie que l'ordinateur ne reconnait pas quelque chose. Généralement parce qu'un mot qu'il connait a été mal orthographié. L'ordinateur affichera des messages d'erreur lorsqu'il ne comprend pas l'instruction ou lorsqu'il la comprend mais juge que ce qui a été demandé est illogique ou impossible.

Tapez PRINT 3/Ø puis appuyez sur [ENTER]

Le message sera:

?/Ø ERROR

ce qui veut dire qu'une tentative a été faite de diviser par séro, ce qui n'est pas possible.

Les messages d'erreur sont assez brefs, pour économiser de la place. Vous en trouverez la liste complète avec les causes probables, dans l'appendice C.

Pour revenir aux erreurs de syntaxe, l'ordinateur est vraiment maniaque au sujet de l'orthographe de son vocabulaire BASIC. Si vous détectez une erreur avant d'appuyer sur [ENTER] vous pouvez utiliser le rappel arrière [←] et corriger. Autrement il n'y a pas d'autre alternative que de retaper toute la ligne correctement.

REGLES ARITHMETIQUES

Jusqu'à présent, nos exemples n'ont demandé à l'ordinateur que de faire 2 "opérations arithmétiques", à savoir l'addition (+) et la division (/). Le mot "opération" signifie quelque chose que nous demandons à l'ordinateur de faire. Il y a six opérations simples que l'ordinateur peut faire en arithmétique et il y a des régles très strictes quant à la façon de les exécuter. Dans l'exemple ci-dessus:

PRINT 12 + 8/2

la réponse peut être 16 ou 1Ø; le choix n'est pas évident.

12 plus 8 divisé par 2 font 1Ø. Mais 8 divisé par 2 plus 12 égal 16.

La réponse donnée par l'ordinateur est 16 parce que tel est l'ordre qu'il choisit pour traiter son arithmétique. Les opérations et l'ordre de priorité dans lequel elles sont traitées est décrit ci-après:

1. *Moins négatif*

Il s'agit d'un signe moins que l'on emploie pour indiquer un nombre négatif.

PRINT - 3 + 2

L'ordinateur va en premier lieu appliquer le signe moins au nombre.
Donc - 3 + 2 donne - 1. Si l'ordinateur faisait l'addition d'abord, - 3 + 2 feraient - 5, ce qui n'est pas le cas!

2. *Elevation à la puissance*

5 à la puissance 4, (5^4), c'est 5 x 5 x 5 x 5

Après avoir appliqué les signes moins, l'ordinateur fait les élévations à la puissance.

PRINT 4 + 3 ↑ 2

est calculé d'abord par le carré de 3 (3 x 3 = 9) puis en additionant 4 pour parvenir au total de 13. S'il y a plus d'une élévation à la puissance à exécuter, elles sont calculées de gauche à droite. Essayez un exemple:

PRINT 2 ↑ 3 ↑ 2 ↑ 3 (la touche [↑] à gauche du clavier s'emploie pour l'élévation à la puissance).

Cette expression est calculée en multipliant 2 par lui-même 3 fois (2 x 2 x 2 = 8) puis ce résultat est multiplié par lui-même (8 x 8 = 64), puis ce résultat est multiplié par lui-même 3 fois (64 x 64 x 64 = 262144).

3. *Multiplication*

Le signe que l'ordinateur utilise pour la multiplication est * afin de ne pas créer de confusion avec la lettre x. On utilise les touches [SHIFT] et [[:].

exemple:
PRINT 5 * 2 + 3

Cela donne 13 (5 x 2 = 10 plus 3 = 13).

4. *Division*

Le signe que l'ordinateur utilise pour la division est / donc 3 : 2 s'écrira 3/2.

PRINT 5/2 + 3

donne 5,5 (5 : 2 = 2,5 plus 3 = 5,5).

La multiplication et la division ont la même préséance, c'est-à-dire le même ordre de priorité. Lorsque des opérations arithmétiques ont la même préséance ils sont calculés de gauche à droite.

PRINT 5 + 2 * 3 + 4/2

donne 13 (2 x 3 = 6, 4 : 2 = 2, 5 + 6 + 2 = 13).

5. *Addition*

Le signe de l'addition est +

6. *Soustraction*

Le signe de la soustraction est -

L'addition et la soustraction ont la même préséance donc il sont aussi calculés de gauche à droite, après que les opérations à plus haute priorité aient été traitées.

Pour résumer, les ordres de préséance qu'applique l'ordinateur dans ses opérations arithmétiques sont:

En premier lieu -	(caractérisant les nombres négatifs)
Ensuite ↑	(les exponentielles, de gauche à droite)
En troisième lieu * /	(multiplication et division, de gauche à droite)
Enfin + -	(addition et soustraction, de gauche à droite)

Vous trouverez ci-desous quelques expressions arithmétiques. Calculez chacune d'elles mentalement (ou avec un crayon et du papier), puis essayez la sur l'ordinateur. Si votre résultat est différent de celui de l'ordinateur, essayer de comprendre pourquoi. A moins que vous n'ayez déjà beaucoup d'expérience sur le maniement des ordinateurs nous vous conseillons très vivement de faire vraiment ce petit exercice. La majeure partie des "erreurs de l'ordinateur" sont en fait des erreurs de programmation: le programmeur n'a pas utilisé les mêmes règles que l'ordinateur... Nous ne vous donnons pas les résultats, car si vous avez tapé ce qui suit correctement, les réponses que vous donnera l'ordinateur seront exactes.

```
PRINT 3 + 2
PRINT 4 + 6 - 2 + 1
PRINT 8 * 4
PRINT 4 ↑ 2 + 1
PRINT 5/4 - 1
PRINT 5 - 4/2
PRINT 6 * - 2 + 6/3 + 8
PRINT 4 + - 2
PRINT 2 * 2 + 3 * 4
PRINT 8/2/2/4
PRINT 20/2 * 5
PRINT 8 * 2/2 + 5 * 3 * 2 ↑ 2
```

Il n'est pas nécessaire de taper chaque fois PRINT, on dispose d'une abréviation: c'est le signe [?].

? 3 + 2 c'est équivalent à PRINT 3 + 2

A la fin de chaque ligne, vous le savez déjà, il faut frapper la touche [ENTER].

Le message OK vous informe que l'ordinateur est prêt.
La touche [ENTER] informe l'ordinateur que vous êtes prêt.

Après vous êtes torturé l'esprit à propos des ordres de préséance, apprenez qu'il est possible de les modifier. Ceci se fait en utilisant les parenthèses. Prenons, par exemple 14 divisé par 4 plus 3, si vous écrivez - 14/4 + 3 la réponse serait 6,5 parce que vous aurez divisé 14 par 4 puis ajouté 3 au résultat. Ce n'est pas ce que vous vouliez. Ecrivez donc plutôt

14/ (4 + 3) et la réponse sera 2.

Les parenthèses modifient l'ordre de préséance, la régle étant simple: calculer en premier lieu ce qui est entre parenthèse. S'il y en a plusieurs, travailler de l'intérieur vers l'extérieur.

12/(3 + (1 + 2) ↑ 2) se calcule de la façon suivante:

a)	12/(3 + 3 ↑ 2)	(1 + 2) est exécuté le premier
b)	12/(3 + 9)	3 ↑ 2 ensuite,
c)	12/12	(3 + 9) ensuite,
d)	1	la division en dernier.

Voici quelques autres expressions à calculer. Ici encore, si vous n'êtes pas vraiment rompu à la façon dont travaillent les ordinateurs, les quelques minutes que vous prendra ce travail vous permettront de tirer meilleur parti de votre ordinateur.

```
? 44/(2 + 2)
? (44/2) + 2
? 4 + (-5*2)
? 100(200/(2*(9-5)))
? 42/((9/3) + 1.75 + (5/4))
```

IMPRESSION DE MOTS

Jusqu'à présent nous n'avons utilisé que des nombres dans les instructions PRINT que nous avons donné à l'ordinateur. Le profane considére souvent ce dernier comme un "avaleur de chiffres" mais ce n'est pas sa seule fonction. Un ordinateur peut aussi s'utiliser pour traiter des caractères. Par caractères, nous entendons les lettres de A à Z, les nombres de 0 à 9, les signes de ponctuation et tous les caractères spéciaux avec lesquels nous ferons connaissance par la sute. Le BASIC permet de traiter des groupes de caractères que l'on appelle *chaînes.*

Une *chaîne* peut être composée de caractères les plus divers - même un espace peut être un caractère important dans une chaîne. Pour aviser l'ordinateur qu'il traite une chaîne, et non un nombre, la chaîne est mise entre guillements (""). Voici des exemples de chaînes.

"LE TITRE", "Z+*?!", "MR J.P. SMITH"

"AXY 479W", "01-479-6172"

Les deux dernières chaînes pourraient représenter un numéro d'immatriculation de voiture et un numéro de téléphone, ils contiennent tous deux des *caractères numériques* mais nous ne saurions nous en servir pour faire de l'arithmétique. Pour le BASIC, un assemblage de chiffres inséré entre guillemets n'est pas un nombre; la chaîne "12345" est une chose totalement différente du nombre 12345. En fait les chaînes et les nombres sont entreposés en des endroits tout à fait différents, dans la mémoire de l'ordinateur.

Essayez ceci:

PRINT "2 + 5 ="; 2 + 5

La première partie de l'instruction PRINT apparaît sur l'écran exactement comme elle est exprimée dans la chaîne. (Les gruillemets servant à contenir la chaîne n'en font *pas* partie). La deuxième partie de la commande est calculée en tant qu'expression *numérique*, l'écran affichera donc

2 + 5 = 7

Comme nous l'avons dit, l'espacement peut être un caractère important d'une chaîne. Dans les instructions BASIC, les espaces n'ont aucune importance, mais rendent la ligne que plus facile à lire. Dans les chaînes, par contre, il ont leur rôle à jouer. Lorsqu'une chaîne est imprimée sur l'écran, elle est reproduite exactement comme le groupe de caractères défini dans les guillemets. Dans ce livre, lorsque nous estimerons qu'un espace est nécessaire, nous l'indiquerons par le signe ∇. Mais attention, ceci n'est pas un signe figurant au clavier et veut seulement dire qu'il y a lieu de taper un espace à cet endroit. Nous ne ferons figurer ce signe que dans les chaînes, tous les autres espaces étant facultatifs.

Vous devriez maintenant pouvoir employer votre ordinateur pour résoudre de petits problèmes dans le genre de ceux figurant à la fin du présent chapitre. Nous nous permettons de vous répéter que si vous n'avez pas l'habitude d'utiliser un ordinateur il serait bon que vous fassiez ces exercices.

1) André mesure 168 centimètres. Quelle est sa taille en pouces anglais (inches)?
(1 inches = 2,54cm).

2) Une recette de cuisine nécessite Ø,75 Kg de farine.
Combien de livres anglaises de farine vous faut-il?
(1 Kg = 2,2 livres).

3) Votre voiture consomme 123 litres d'essence au cours d'un voyage. Au départ, le compteur indique 16.550 Km et à la fin du voyage le compteur indique 17.950 Km.
Quelle est la consommation du véhicule, exprimée en litres aux 100 Km.

4) Vous placez 15.000 Francs dans un compte d'épargne qui rapporte 11% l'an. Combien en retirerez vous après 5 ans?
$(A=P(1+R/1ØØ)^N$ P = somme de départ R = taux d'intérêt annuel N = nombre d'années A = montant disponible après N années.

REPONSES

1) 66.14 inches

2) 1.65 livres

3) 8,786 litres aux 100 km

4) 25.275 francs.

CHAPITRE DEUX

LE SENS DES MOTS

CONSTANTES

Tous les exemples que nous avons utilisé jusqu'ici ne contenaient que des *constantes.* Une *constante* est tout à fait conforme à son nom - quelque chose qui ne change pas - 3,145 est une constante, si on le modifie pour en faire 3,146 cela devient une autre constante, c'est tout. Les constantes sont bien utiles dans les programmes informatiques, mais pas aussi utiles que les *variables.*

VARIABLES

Une *variable* est quelque chose qui peut changer de valeur.

Dans l'équation X + 5 = Y

X et Y sont des variables puisque tous deux peuvent prendre diverses valeurs, la fonction restant exacte. Les variables dans un ordinateurs sont des emplacements dans sa mémoire. On pourrait les comparer à un ensemble de cases ou de boîtes. Pour les reconnaître il est nécessaire de les étiquetter (leur donner des noms comme X ou Y). Ces variables, dans votre ordinateur, sont de deux sortes - *numériques* ou *chaînées* et de deux dimensions: *simples* ou en forme de *tableau.* Nous connaissons déjà la différence entre les constantes numériques et les constantes chaînées, donc les variables numériques contiendront toujours des nombres et les variables chaînées toujours des chaînes de caractères (alpha numérique). Pour l'instant nous ne nous occuperons que des variables simples; les variables tabulées seront traitées ultérieurement.

NOMMER LES VARIABLES

Pour donner un nom à une variable *numérique* vous pouvez employer n'importe quelle combinaison d'une lettre suivie ou non d'une lettre ou d'un chiffre.

N, AA, X, TI, Y, Z9, L5, BZ, PQ, K9

sont des exemples de noms de variables numériques valables.

En fait, votre ordinateur permet à un nom de variable numérique d'avoir n'importe quelle longueur mais il ne reconnaît que les deux premiers caractères du nom; il acceptera sans rechigner des noms de variables numériques tels que

MAISON, MARBRE, MALDEMER

mais les identifiera comme une seule et même variable: MA. Cela s'applique également aux variables chaînées.

NOMS DES VARIABLES NUMERIQUES

Une variable *numérique* ne peut contenir que des chiffres.

Le nom d'une variable numérique peut se composer de toute combinaison de lettres et de chiffres mais *doit* commencer par une lettre.

Comme l'ordinateur ne reconnait que les deux premiers caractères du nom d'un variable, les noms tels que:

ETAT, ETAGE, ETANGS

seront tous reconnus comme étant le même: ET

Les noms de variables qui sont plus longs que deux caractères sont pratiques pour vous rappeler ce que vous y avez mis.

NOMBRE, COMPTOIR, SOMME

seront acceptés mais utiliseront plus de mémoire dans l'ordinateur que NO, CO, SO.

D'autre part:

FRANC$, FRAI$, FRUIT$

sont considérés comme FR$

Pour nommer une variable *chainée* on utilise donc le même système *mais son nom doit obligatoirement se terminer par* $.

A$, P7$, MN$, ZØ$

sont des exemples de noms de variables chaînées valables.

ATTRIBUTION DE VALEURS AUX VARIABLES

Comment utilise-t-on ces variables? Tapez les exemples suivants:

```
A = 5                    n'oubliez pas d'appuyer sur la
B = 2                    touche [ENTER] après chaque ligne
C = A + B
D = D + 3
PRINT A, B, C, D         (tapez bien les virgules)
```

Votre écran va afficher

```
5         2
7         3
```

La première ligne veut dire: stocker la valeur 5 dans une variable nommée A, la ligne suivante stocke 2 dans la variable B. (L'ordinateur décide lui-même où caser ces valeurs dans sa mémoire; il vous suffit de lui fournir leur noms). La troisième ligne dit: "trouvez les valeurs contenues dans les variables A et B, additionnez les puis casez le total dans une variable nommée C." Après que l'ordinateur ait exécuté cette instruction, les variables A et B contiennent toujours leur valeur initiale (soit 5 et 2 dans ce cas) et C contient la somme, soit 7. La quatrième ligne pourra sembler bizarre à ceux qui connaissent l'algèbre. Ceci résulte du fait qu'en BASIC, le signe "égal" (=) ne signifie pas la même chose qu'en mathématique. Le signe = en BASIC signifie: prenez le second membre de cette expression, calculez en la valeur si nécessaire, et placez la dans la variable définie par le premier membre de l'expression.

Une telle ligne s'appelle une *déclaration d'attribution* et le côté *gauche* de cette déclaration d'attribution doit *toujours* être une variable. Une expression dans le genre de 2 = B + C est valable en algèbre *mais pas* en language BASIC.

Pour en revenir à la déclaration D = D + 3, elle signifie que le contenu actuel de la variable D (qui pour l'instant est zéro puisque nous n'y avions encore rien mis), se voit ajouter 3 puis devient la valeur modifiée de la variable D. Autrement dit:

ajoutez 3 à la variable D. Ceci peut sembler un peu étrange mais c'est d'un usage très utile (et très courant) dans les programmes informatiques. Par ailleurs cela démontre une autre caractéristique des variables: elles ne peuvent contenir qu'une seule valeur à la fois. Si vous attribuez une valeur à une variable dont le nom apparaît à *gauche* d'une déclaration d'attribution, la valeur attribuée remplace l'ancienne valeur qui est donc perdue. Vous pouvez toutefois recopier cette valeur avant de la perdre, dans une autre variable, ou en utilisant une expression du genre C = A + B.

Maintenant vous tapez:

```
A = B                    souvenez-vous de taper [ENTER]
B = 17                   après chaque ligne.
D = D + 2
PRINT A, B, C, D
```

la réponse sera

```
2          17
7           5
```

La variable A contient maintenant la copie de B, en provenance de l'exemple précédent, la variable B contient une nouvelle valeur (17), et les contenus précédents de A et B sont perdus. La variable C n'a pas changé. La variable D vaut maintenant 5; puisqu'elle détenait 3 depuis l'exemple précédant et qu'il lui a été ajouté 2.

Les variables chaînées se comportent exactement de la même façon (sauf que vous devez vous souvenir que leur nom doit toujours se terminer par le signe $). Essayez cet exemple

```
A$ = "CECI ∇ EST ∇ UNE ∇"
B$ = "TRES ∇"
C$ = "LONGUES ∇ CHAINE"
D$ = A$ + B$ + B$ + B$ +B$ +B$ + B$
D$ = D$ + C$
PRINT D$
```

Votre écran affichera:

```
CECI EST UNE TRES TRES TRES TRES
TRES TRES LONGUE CHAINE
```

Aux lignes 1, 2 et 3 nous avons attribué leur valeur aux variables A$, B$ et C$. A la ligne 4 nous avons additionné six fois B$ à A$. Avec les variables de chaîne, le signe plus (+) ne signifie pas la même chose qu'avec les variables numériques. Il signifie: ajoutez au bout de la chaîne précédente. (Pour ceux qui aiment les mots savants, cela s'appelle "concatenation").

A la ligne 5 nous ajoutons C$ à la fin du D$ que l'on venait de constituer. Ceci est une façon d'utiliser l'ordinateur pour construire les phrases.

LES CHAINES ET LES NOMBRES NE SE MELANGENT PAS

Souvenez vous bien qu'il faut séparer les variables numériques des variables chaînées. Seuls les nombres peuvent être stockés dans les variables numériques et les variables de chaînes ne peuvent contenir que des chaînes de caractères (alphanumériques). Donc des déclarations de ce type:

D = "CHAINE"

A$ = 6

B = A$ * 2

donneront lieu au message d'erreur: ?TM ERROR signifiant qu'il y a incompatibilité dans l'écriture (Type mismatch)

Le signe plus (+) est le *seul* opérateur arithmétique utilisable avec les variables chaînées, tous les autres (-, *, /, ↑) donneront lieu a un message d'erreur.

PRECISIONS SUR LES COMMANDEMENTS

Dans la suite de ce volume vous constaterez que certaines pages sont encadrées. Celles-ci vous préciseront les détails relatifs à chaque commandement, au fur et à mesure que nous les aborderons. A la fin de la plupart de ces pages encadrées figure un petit programme modèle qui explicite l'usage du commandement en question. Etudiez attentivement ces petits programmes, travaillez-les et essayez de découvrir comment l'ordinateur va les traiter avant de les faire tourner. Ces programmes ont été conçus pour démontrer comment opère une instruction donnée mais ils contiennent également des astuces pratiques que vous trouverez utiles à inclure plus tard dans vos propres programmes.

NOMS DES VARIABLES CHAINEES

Les variables chaînées ne peuvent contenir que des chaînes.

Le nom d'une variable chaînée peut se composer de toute combinaison de lettres et de chiffres, mais *doit* débuter par une lettre et se terminer par le signe $.

De même qu'avec les noms des variables numériques, l'ordinateur ne reconnaîtra que les deux premiers caractères du nom de la variable; c'est ainsi que,

ANTENNE$, ANIS$, AN2$

seront tous trois considérés comme une seule et même variable: AN$

CHAPITRE TROIS

LE PROGRAMME

Jusqu'à présent, votre ordinateur n'a rien fait d'autre que de vous renvoyer la ligne que vous veniez d'y entrer... Maintenant nous allons commencer à construire un *programme* d'ordinateur.

Un *programme* est un ensemble d'instructions qui commandent à l'ordinateur de faire quelque chose. Un programme en BASIC consiste en un certain nombre de *lignes*. Une *ligne* est constituée de deux parties: d'abord un *numéro de ligne* et ensuite une ou plusieurs *instructions*. S'il y a plus d'une instruction sur une ligne, chaque instruction doit être séparée par deux points(:). Une *instruction* dit à l'ordinateur ce qu'il doit faire; nous en avons déjà utilisées de semblables auparavant.

PRINT A$

Voici un programme en BASIC:

```
10 CLS0
20 PRINT "QUEL EST VOTRE NOM?"
30 INPUT NOM$
40 I = RND (255): J = RND (9) - 1
50 CLS J
60PRINT @ 200 + J, NOM$;
70 SOUND I, 2
80 GOTO 40
```

Comme vous le constatez, ce programme contient de nouveaux termes de BASIC. Ne vous en occupez pas pour l'instant, ils seront expliqués par la suite. Remarquez la structure d'un programme en BASIC - une séquence de *lignes*, chaque ligne étant constituée d'un *numéro de ligne* et d'une *instruction* au moins. (Il y en a deux dans la ligne 40).

CHARGEMENT D'UN PROGRAMME

Pour introduire un programme dans la mémoire de l'ordinateur nous devons tout d'abord vider celle-ci de ce qu'elle pouvait contenir avant. Pour ce faire, on tape NEW puis [ENTER].

Ensuite entrer chaque ligne, exactement comme elles sont écrites ci-dessus, en tapant [ENTER] à la fin de chaque ligne. Vous remarquerez qu'après [ENTER] il ne se passe rien, car une ligne qui débute par un numéro n'est pas exécutée; elle est stockée et c'est tout. Quand un programme "tourne", il part au numéro de ligne le plus bas, exécute cette ligne, puis passe à la ligne suivante dans l'ordre croissant des numéros de lignes, et ainsi de suite. En raison de ce que la séquence d'exécution d'un programme dépend des numéros de ligne, vous pouvez rentrer les lignes dans l'ordre où vous le voulez, l'ordinateur les classera de lui-même dans l'ordre numérique croissant.

Essayez de taper ce programme sur votre ordinateur. Si vous faites une erreur avant d'avoir tapé [ENTER] utilisez la touche de rappel arrière, comme nous vous l'avons indiqué précédemment. Si vous ne constatez une erreur qu'après avoir entré la ligne, alors ré-écrivez là entièrement et faites là entrer à nouveau. L'ordinateur ne conservera que la version la plus récente de cette ligne.

Après que vous ayez entré tout le programme, pour voir comment l'ordinateur l'a stocké dans la mémoire - tapez LIST et appuyez sur [ENTER]. Remarquez que LIST n'est pas précédé d'un numéro de ligne. C'est un ordre direct, il s'exécute immédiatement. Vérifiez bien le programme pour vous assurer que tout est juste (sinon re-tapez les lignes qui sont entachées d'erreur.

Maintenant, enfin, nous sommes prêts à faire tourner le programme. Pour cela, tapez RUN puis [ENTER].

L'ordinateur entre aussitôt en action.

L'écran va se dégager puis un message apparaîtra en haut de l'écran, vous demandant votre nom. Tapez votre nom et appuyez sur [ENTER].

L'écran alors va s'éclairer de différentes couleurs et votre nom va apparaître, sautant de ci delà au milieu de l'écran. De curieux bruits accompagneront toute cette activité (si vous avez pensé à ouvrir le son du téléviseur).

Ceci va durer éternellement à moins que vous n'y mettiez fin. La méthode la plus radicale pour arrêter le déroulement d'un programme est de couper le courant... Mais ce n'est pas très satisfaisant, car le programme sera perdu. La meilleure façon de faire est d'appuyez sur la touche BREAK.

PAS A PAS

Maintenant que vous avez vu ce que fait ce programme, nous allons vous expliquer comment il le fait, ligne par ligne

10 CLS

Puisque c'est la ligne qui a le plus petit numéro, c'est celle ci qui sera traitée la première. Le commandement CLS veut dire: dégager l'écran. (en Anglais clear screen), et mettre le fond à sa couleur habituelle: le vert.

20 PRINT "QUEL EST VOTRE NOM"

C'est le commandement PRINT (en Anglais imprimer) que nous avons déjà vue. Cette ligne affiche le message en haut de l'écran.

30 INPUT NOM$

Le commandement INPUT dit à l'ordinateur de s'arrêter et d'attendre que vous entriez quelque chose, qu'il affectera à la variable qui lui a été désignée.

LIST

Le commandement LIST affiche le programme en cours sur l'écran. Il n'est pas précédé d'un numéro de ligne.

Si le programme est trop long pour tenir tout entier sur l'écran, son listage peut être arrêté en frappant les touches SHIFT et @ en même temps, (mais il faut faire vite). Le listage peut être repris en frappant n'importe quelle autre touche du clavier.

Pour n'obtenir qu'une partie du programme vous pourrez utilisez

LIST n_1 - n_2

n_1 et n_2 étant deux numéros de lignes (n_2 doit être plus grand que n_1).

LIST 40 - 100

affichera toutes les lignes de programmation de la ligne 40 à la ligne 100.

LIST - 80

affichera toutes les lignes depuis le début du programme jusqu'à la ligne N°80.

LIST 120 -

affichera toutes les lignes à partir de la ligne N°120, jusqu'à la fin du programme.

RUN

Le commandement RUN s'emploie pour mettre en route l'exécution d'un programme.

Il n'a pas de numéro de ligne.

Si vous désirez faire démarrer un programme de n'importe quel endroit, autre que le début, vous pourrez le faire en tapant

RUN *numéro de ligne*

le *numéro de ligne* étant celui d'où vous souhaitez faire partir le programme.

RUN 25Ø

NEW

Le commandement NEW vide la mémoire et met toutes les variables à zéro.

Vous remarquerez qu'il n'y a pas de numéro de ligne.

C'est une bonne chose de taper NEW avant d'entrer un programme, pour vous assurer que rien ne traîne dans la mémoire qui pourrait déranger le nouveau programme que vous allez rentrer.

DECLARATION D'ATTRIBUTION

La *déclaration d'attribution* s'utilise pour attribuer une valeur a une *variable*.

La forme que prend la déclaration d'attribution est:

LET *Variable* = expression de la valeur.

La partie LET de la déclaration fait partie du vocabulaire BASIC standard (LET veut dire SOIT, en Anglais) mais n'est pas nécessaire sur votre ordinateur. Donc LET n'apparaîtra pas dans les programmes du présent manuel.

La *variable* faisant partie de la déclaration peut porter n'importe quel nom.

L'*expression* faisant partie de la déclaration peut être une constante, une autre variable, ou un mélange des deux reliés par des opérateurs (+, –, *, etc.). Comme les variables de chaînes de caractères et les variables numériques ne peuvent pas être mélangées, la *variable* et l'*expression* doivent être de la même sorte.

Le signe égal (=) ne signifie pas la même chose que le signe égal en algèbre; il convient de l'interpréter comme voulant dire: 'attribué à'. Donc la même variable peut apparaître des deux côtés de la déclaration comme dans l'exemple suivant:

```
40 X = X + 1
```

qui indique à l'ordinateur qu'il y a lieu d'ajouter 1 à la valeur actuelle de X et de remettre la nouvelle valeur dans X.

```
10 S = 0: N = 0: CLS 5
20 PRINT @ 72, "ENTREZ UN NOMBRE";
30 INPUT X: CLS 5
40 S = S + X: N = N + 1
50 PRINT @ 194, "VOUS AVEZ ENTRE"; N; "NOMBRES";
60 PRINT @ 262, "LA MOYENNE EST", S/N;
70 GOTO 20
```

(Souvenez-vous que l'ordinateur, dans ce cas, ne retiendra comme nom de la *variable* que NO$ et ignorera les autres lettres. Il est toutefois souvent utile d'utiliser un nom de variable qui soit plus long; c'est plus compréhensible, et peut vous aider à retrouver ce que contient cette variable.

40 I = RND (255): J = RND (9) - 1

Cette ligne 40 montre que plus d'une instruction peuvent figurer sur la même ligne. (Vous noterez la présence de deux points (:) qui les séparent). Vous découvrez pour la première fois le commandement RND. Il crée un nombre au hasard, comme si on le tirait au lotto. Le nombre entre parenthèses, après RND dit à l'ordinateur dans quelle gamme de nombres il doit faire sa sélection. Donc I = RND (255) veut dire qu'il y a lieu de choisir un nombre entier dans la gamme de 1 à 255, puis de placer ce nombre dans une variable appelée I. Dans la deuxième déclaration la gamme du hasard est maintenant limitée de 1 à 9; toutefois, après que le nombre au hasard ait été trouvé, on lui soustrait 1 avant de l'introduire dans J. Cela signifie que J sera un nombre de 0 à 8.

50 CLS (J)

Cette ligne vide l'écran. Cette fois cependant, le fond de couleur dépendra de la valeur de J. Il y a neuf couleurs disponibles qui sont numérotées de 0 à 8. C'est cette ligne qui provoque les changements de couleur de l'écran.

60 PRINT @ 200 + J, NOM$;

C'est une version plus élaborée de notre vieille connaissance, le commandement PRINT. Cette ligne indique à l'ordinateur qu'il doit imprimer le contenu de NOM$ (dans le cas présent, votre nom), en commençant à un endroit défini de l'écran. Dans cet exemple la position est à 200 + J, ce qui est donc quelque part entre 200 et 208. La position 200 se situe à la ligne 7, 8 espaces après le début. (Reportez vous à la grille des PRINT @ pour vous rendre compte comment l'écran est divisé). En raison de ce que la valeur de J change, votre nom semble sauter le long de la ligne.

70 SOUND I, 2

Ceci est la ligne qui produit les bruit étranges. Le commandement SOUND (sound c'est son en Anglais) dit à l'ordinateur d'utiliser son générateur de sons; il lui dit quel son produire et de quelle durée, au moyen des deux chiffres qui suivent SOUND.

80 GOTO 40

GOTO (en Anglais go to: aller à) veut tout simplement dire: "allez à la ligne ci-après indiquée", (donc dans ce programme, allez à la ligne 40).

PRINT

Le commandement PRINT demande à l'ordinateur d'afficher sur l'écran quelque chose qui y est contenu.

Ce "quelque chose" peut être une donnée, la valeur variable, le résultat d'un calcul ou le résultat de toute autre activité de l'ordinateur dont on veut prendre connaissance: on l'appelle une "sortie". Si l'instruction PRINT comporte plusieurs éléments, ils doivent être séparés par une virgule (,) ou un point-virgule (;).

La virgule provoquera l'affichage de la sortie sur deux colonnes, chacune large de quinze caractères. (Si la longueur du premier élément dépasse 15 caractères il sera quand même imprimé entièrement. Le texte de l'élément suivant apparaîtra à la ligne suivante).

Le point virgule comporte la sortie. Les chaînes seront imprimées les unes à la suite des autres et les valeur numériques seront précédées et suivies d'un espace. Le point virgule maintient à la place qu'elle occupe la position d'impression et repart de là quand le programme atteint le PRINT suivant.

Lorsque PRINT n'est pas suivi d'aucune instruction, il donne lieu a une ligne "blanche" (vide).

```
10 CLS
20 PRINT
30 PRINT "▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ le commandement PRINT
40 PRINT "▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ ▽ UNE DEMONSTRATION
50 PRINT
60 PRINT "COLONNE UN", "COLONNE DEUX"
70 PRINT 14.2, 13.7
80 PRINT 1, 2, 7, 11
90 PRINT
100 A$ = "COMPACTEE": B = 3
110 PRINT "SECONDE LIGNE"
130 PRINT A$; "SORTIE DE LA LIGNE"; B
140 PRINT
150 PRINT "CECI APPARAITRA";
160 PRINT "▽ COMME UNE LIGNE UNIQUE"
170 PRINT "DEMONSTRATION"
180 PRINT "TERMINEE"
```

INPUT

Quand un programme atteint une instruction INPUT, il s'arrête et attend qu'on lui entre quelque chose au moyen du clavier. Après le commandement INPUT doit donc figurer un ou plusieurs noms de *variables* séparés par des virgules.

25 INPUT A,B,F$,H7

L'instruction ci-dessus vous demande de rentrer 4 éléments. Ceux-ci peuvent être fournis, soit l'un après l'autre en frappant la touche [ENTER] après chacun d'eux, soit en séparant chaque élément par une virgule. Par exemple:

146.2, 78.1, STRING, 3 [ENTER]

C'est toujours une bonne chose que d'imprimer un message avant l'instruction de INPUT afin de vous rappeler ce qui est demandé. Ceci peut être fait au moyen d'une instruction de PRINT ou inclus dans l'INPUT, comme ci-après:

35 INPUT "DEUX NOMBRES SVP"; N1, N2

A noter, la présence d'un point-virgule séparant la chaîne de la liste des entrées. (en Anglais input veut dire entrée)

Vous devez prendre garde à faire des entrées correctes (des chaînes pour les variables chaînées et des nombres pour les variables numériques), autrement le programme s'arrêtera et affichera: ?REDO (recommencez!), et vous devrez refaire votre entrée.

Il n'est pas nécessaire de mettre les chaînes entre guillemets lorsqu'elles sont entrées en INPUT et affectées à une variable de chaîne. Donc dans l'exemple ci-dessus on peut rentrer "STRING" ou aussi STRING, et ça marche dans les deux cas.

Dans le programme de démonstration qui suit vous remarquerez que:

a) à la ligne 170, INPUT est utilisé pour arrêter le programme jusqu'à ce que l'on soit prêt. A$ ne contiendra rien.

b) des variables chaînées (C$, P$ et T$) sont utilisées pour éviter de devoir taper le même message plusieurs fois

```
10 CLS: T$ = "CECI EST UNE DEMONSTRATION DE INPUT"
20 P$ = "FRAPPEZ LA TOUCHE ENTER POUR CONTINUER"
30 C$ = "ENTRER 4 NOMBRES,"
40 PRINT: PRINT T$: PRINT
50 PRINT C$; "FRAPPER LA"
60 PRINT "TOUCHE ENTER APRES CHACUN"
70 INPUT A, B, C, D
80 PRINT: PRINT "VOUS AVEZ ENTRE CES VALEURS: -"
```

```
 90 PRINT A; B; C; D: PRINT
120 PRINT "MAINTENANT ∇"; C$; " ∇ SEPAREES"
130 PRINT "PAR DES VIRGULES ET FRAPPER ENTER."
140 INPUT A, B, C, D
150 PRINT: PRINT "CETTE FOIS LES NOMBRES ETAIENT:-"
160 PRINT A; B; C; D
170 PRINT: PRINT P$: INPUT A$: CLS
180 PRINT: PRINT T$: PRINT
190 INPUT "ENTRER UNE CHAINE ET UN NOMBRE"; B$,N
200 PRINT: PRINT "LA CHAINE QUE VOUS AVEZ ENTRE ETAIT: -";
    PRINT
210 PRINT B$
220 PRINT: PRINT "ET LE NOMBRE ETAIT: -"; N
230 PRINT: PRINT "FIN DE DEMONSTRATION"
```

Ainsi donc le programme retourne à la ligne 40 ou il sélectionne un nouveau nombre au hasard pour I et pour J. Comme la couleur du fond est conditionnée par J, elle change dès que le programme atteint la ligne 50. Et comme I a changé également, le son va changer à la ligne 70. Quand le programme atteint la ligne 80, il retourne à nouveau à la ligne 40 où il sélectionne etc. etc. (et il tourne ainsi en rond... indéfiniment, jusqu'à ce que vous fassiez appel à la touche BREAK).

QUELQUES CHANGEMENTS

C'était le premier programme. Il ne faisait pas grand chose, mais c'est un début. Il nous a permis de démarrer et de vous présenter de nouveaux commandements. Une explication plus détaillée de ceux-ci se trouve dans les pages encadrées qui jalonnent ce volume. (Ces pages contiennent également de petits programmes de démonstration. Vous devriez essayer de faire tourner chacun de ces programmes).

Si ce premier programme ne vous a pas impressionné, vous pouvez tenter d'y faire des changements. Faire des changements dans un programme qui est déjà tapé est une chose très simple. Parceque l'ordre dans lequel se déroule un programme en BASIC est établi par les numéros de lignes, une ligne peut être changée en écrivant une autre ligne comportant le même numéro que celle qui se trouve déjà dans le programme. La nouvelle ligne remplace l'ancienne. Essayez d'entrer une nouvelle ligne, portant le numéro 70.

70 SOUND I, K

D'autre part, une nouvelle ligne peut être insérée dans le programme en lui donnant un numéro qui la placera à l'endroit souhaité

Ainsi, tapez

45 K = RND (20)

Comme cette ligne porte le numéro 45 elle viendra tout naturellement s'insérer entre les lignes 40 et 50

(Quand vous écrivez vos propres programmes, vous pouvez numéroter les lignes comme vous voulez de 0 à 63999. Il est généralement souhaitable de numéroter les lignes de 10 en 10. A savoir: 10, 20, 30, cela vous laisse de la place pour insérer de nouvelles lignes si nécessaire).

Essayez de faire tourner le programme modifié. (tapez RUN), la nouvelle ligne 45 sélectionne un autre nombre au hasard, cette fois entre 1 et 20. La ligne 70 modifiée utilise ce nombre aléatoire, (dans la variable K), pour modifier la durée du son.

CONSTRUCTION D'UN PROGRAMME

Bien qu'il soit facile de s'installer au clavier et de taper des lignes de programmation, les difficultés ont tendance à apparaître ultérieurement. Surtout quand le programme devient plus long. Taper un programme au clavier devrait être la *dernière* partie de la réalisation d'un nouveau programme.

RND

La commande RND génére des nombres aléatoires (pris au hasard).

RND est une *fonction*. Une *fonction*, en BASIC c'est quelque chose qui prend un ou plusieurs nombres et leur fait subir une opération dont le résultat est une valeur unique. Les nombres utilisés par la *fonction* sont appelés *arguments* et sont toujours mis entre parenthèses après le nom de la fonction. Le résultat d'une fonction est dit être *restitué* au programme.

La *fonction* RND *restitue* un nombre au hasard, compte tenu cependant de l'*argument de fonction*.

Si la valeur de l'*argument* est Ø (RND(Ø)) la fonction restituera une valeur comprise entre Ø et 1.

Si la valeur de l'argument est supérieure à Ø (RND (6)) la fonction restituera un nombre entier compris entre 1 et la valeur de l'argument. (RND (6)) restituera soit 1, soit 2, ou 3, ou 4, ou 5, ou 6, au hasard:

```
1Ø CLS
2Ø PRINT @ 8, "TAPEZ UN NOMBRE";: INPUT N
3Ø CLS: PRINT @ 194, "3 NOMBRES AU HASARD POUR N = ";N
4Ø  PRINT @ 27Ø, RND (N)
5Ø PRINT @ 3Ø2, RND (N)
6Ø PRINT @ 334, RND (N)
7Ø GOTO 2Ø
```

CLS

Le commandement CLS s'utilise pour vider l'écran et pour établir un fond de couleur. Le fond normal est vert. Si vous utilisez CLS sans rien spécifier d'autre c'est cette couleur qui apparaîtra.

Pour changer la couleur du fond, ajoutez un chiffre entre Ø et 8 après CLS, (CLS 2).

Les couleurs disponibles sont:

Ø - Noir	1 - VERT	2 - Jaune
3 - Bleu	4 - Rouge	5 - Blanc
6 - Cyan (bleu primaire)	7 - Magenta (rouge primaire)	8 - Orange

La nuance exacte de ces couleurs sera influencée par votre poste de télévision.

Vous remarquerez toutefois que quelle que soit la couleur du fond, l'ordinateur imprimera toujours les textes en noir ou en vert.

```
1Ø CLS
2Ø PRINT @ Ø, "DEMONSTRATION DES COULEURS DE FOND"
3Ø PRINT @ 192, "TAPER UN CHIFFRE ENTRE Ø ET 8"
4Ø INPUT C
5Ø CLS C
6Ø PRINT @ 288, "VOICI LA COULEUR DE FOND NO:-"; C
7Ø GOTO 2Ø
```

Commencez avec un papier et un crayon et écrivez ce que vous avez l'intention de faire. Divisez le problème à résoudre en plusieurs sections. Beaucoup de problèmes que l'on souhaite traiter par ordinateur peuvent se diviser en trois partie au moins:

1) Préparation, titres, instructions, données à traités;
2) calculs;
3) affichage des résultats.

Maintenant attaquez vous à chaque section séparément, en la divisant éventuellement en sous-rubriques jusqu'à ce qu'il ne reste à faire qu'une action simple du genre "ajouter 1 au compteur". Ordonnez ces différentes actions en séquence logique, écrivez-les en entier, puis passez à la section suivante. Quand vous avez terminé ceci, vous vous retrouvez avec une série d'étapes (certaines d'entre elles étant très sommaires) composant chaque section. Quiconque lirait le résultat de ce travail devrait - si c'est bien fait -pouvoir le résoudre en se servant uniquement de l'arithmétique élémentaire. (Ce dispositif s'appelle un algorithme en langage informatique). Maintenant tout ce qu'il vous reste à faire c'est de traduire votre plan en une langue que l'ordinateur puisse comprendre. Si votre travail a été fait à la perfection, chaque étape deviendra ainsi une ligne de programmation BASIC.

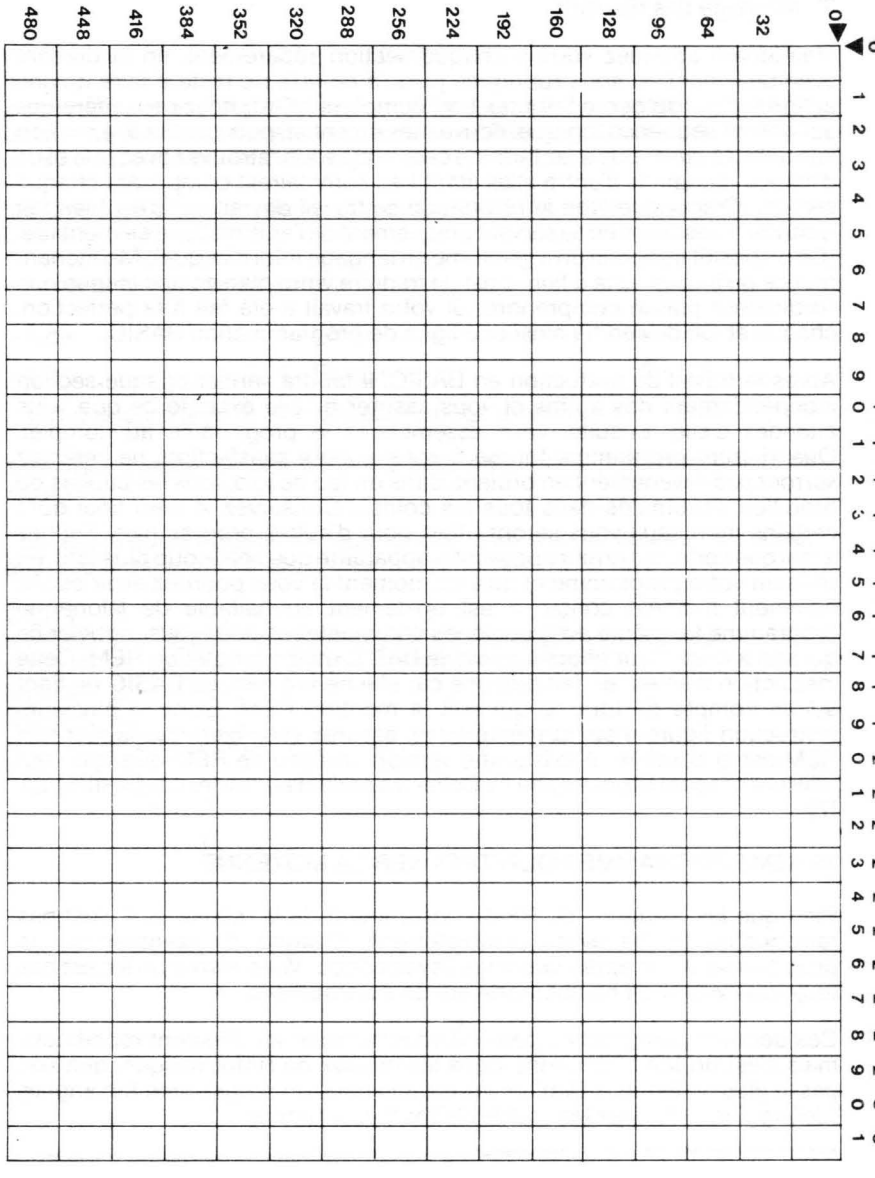
Après le travail de traduction en BASIC, il faudra vérifier chaque section indépendament des autres et vous assurer qu'elle exécute ce que vous attendez d'elle. Ensuite, vous assemblerez le programme au complet. Quand votre programme tourne à votre entière satisfaction, ne célébrez *surtout pas* l'événement en brûlant, dans un feu de joie, tous les papiers de brouillon accumulés dans tous les coins... Conservez le plan final dont certains morceaux vous seront utiles pour d'autres programmes. Sachez aussi que certaines erreurs peuvent n'apparaître que beaucoup plus tard, en utilisant votre programme et qu'à ce moment là vous pourriez avoir oublié comment il a été conçu. Il est également souhaitable de jaloner le programme lui-même avec quelques commentaires pour vous rappeler ce qui sert à quoi. Pour effectuer cela, le BASIC utilise la notation REM. Cette instruction n'en est en fait pas une car elle ne fait rien. Le BASIC ne tient aucun compte de tout ce qui suit la mention REM. Donc si plusieurs instruction figurent sur la même ligne, assurez vous bien que la mention REM est la dernière. Il existe une version abrégée de REM que l'on peut utiliser en tapant l'apostrophe ('). Celle-ci s'obtient au clavier par [SHIFT] ET [7].

```
10 REM PROGRAMME POUR TROUVER LA MOYENNE
```

Bien que les mentions de REM consomment de la mèmoire, il n'est pas raisonnable de les éviter complètement. Essayez de comprendre un programme un an après que vous l'ayez conçu. Vous verrez qu'il n'est pas toujours évident s'il ne comporte aucun commentaire.

Ces derniers paragraphes peuvent vous sembler terriblement rébarbatifs, mais c'est un fait bien connu dans les milieux de l'informatique, que l'on passe plus de temps à éliminer les erreurs dans un programme (en anglais "debugging"; éliminer les "petites bêtes") qu'à l'écrire.

Tableau des Print @



PRINT @

La commande PRINT @ s'utilise pour positioner une sortie à un endroit défini de l'écran.

Pour ce faire, l'écran est décomposé en une grille de 16 x 32 définissant 512 positions. Reportez vous au tableau des PRINT @ pour voir comment ces positions sont numérotées.

La forme du commandement PRINT @ est la suivante:

PRINT @ *expression, liste d'affichage.*

L'*expression* peut-être un nombre, une variable ou une expression arithmétique, tant que la valeur reste comprise entre Ø et 511.

La liste d'affichage est la même que pour PRINT; ce peut être des nombres, des variables, des chaînes ou des expressions, séparés par des virgules ou des point virgules.

Si vous considérez votre écran comme étant divisé en 16 lignes horizontales, l'instruction

PRINT @ 32 * (LINE-1), A

imprimera la valeur de A en début d'une ligne imaginaire sur votre écran, la position de cette ligne dépendant de la valeur donnée à la variable LINE. (qui doit être comprise entre 1 et 16).

L'exemple suivant est volontairement embrouillé, pour un résultat relativement simple. Essayez à titre d'exercice de comprendre ce qu'il va faire. Ensuite faites tourner ce programme et essayez d'obtenir le même résultat par une programmation plus directe.

Vous releverez l'usage du point virgule à la fin e certaines déclarations, pour éviter que le reste de la ligne ne soit effacé. (Pour vous en convaincre, essayez d'entrer une ligne 11Ø sans le point virgule final).

```
1Ø CLS: P$ = "PRINT @":N = 1
2Ø COLONNE = 12: A$ = "SUR L'ECRAN"
3Ø PRINT @ 32 * (COLONNE - 1) + 19, "A PLACER";
4Ø PRINT @ 448 + 9, "N'IMPORTE OU";
5Ø PRINT @ 262, "CECI";: PRINT @ 267, "MONTRE";
6Ø PRINT @ 32 * (COLONNE - 1) + 11, "EST UTILISE";: PRINT @ 13,
"PAGE";
7Ø PRINT @ 448 + 18, A$;
8Ø PRINT @ 273, "COMMENT ▽"; P$
```

```
 90PRINT @ 32 * (COLONNE - 1) + 7, "PEUT";
100 PRINT @ 134, P$; " ∇ DEMONSTRATION";
110 PRINT @ 451, "ELEMENTS";
120 PRINT @ 18, N
130 GOTO 130
```

La dernière ligne (130) installe l'ordinateur dans une boucle sans fin qui ne fait rien. Ceci empêche le signal OK d'apparaître quand le programme est achevé. Vous appuyerez sur la touche BREAK pour interrompre le programme.

SOUND

Le commandement SOUND crée une tonalité de hauteur (ou si vous préférez de fréquence) et de durée définies. Cela nécessite deux arguments:-

50SOUND P,D

P est un nombre compris entre 1 et 255. Le ton le plus bas est 1, le plus haut est 255. Do cinquième sur le piano est P = 89.

D est un nombre compris entre 1 et 255. D = 16 rendra un ton d'une durée approximative d'une seconde.

```
10 CLS
20 PRINT @ 6, "DEMONSTRATION DU SON"
30 PRINT @ 64, "ENTRER UN CHIFFRE ENTRE 1 ET 255,";
40 PRINT @ 96, "POUR LA HAUTEUR DE LA NOTE,";: INPUT P
50 PRINT @ 192, "ENTRER UN NOMBRE POUR LA DUREE DE LA NOTE";
60 INPUT D: CLS (RND (9) - 1)
70 SOUND P, D
80 GOTO 10
```

GOTO

Le commandement GOTO doit avoir la forme:-

GOTO *numéro de ligne*

Le *numéro de ligne* doit être un nombre (pas une variable) et doit exister dans le programme. Si le *numéro de ligne* n'est pas trouvé, le programme s'arrêtera sur le commentaire ?UL ERROR (le UL = Undefined Line, soit ligne NON définie).

L'instruction GOTO est exécutée immédiatement, il n'y a donc pas lieu de metre une autre instruction sur la même ligne, car le programme ne l'atteindrait jamais.

```
20 CLS
30 GOTO 60
40 PRINT "A LA LIGNE 40"
50 GOTO 80
60 PRINT "A LA LIGNE 60"
70 GOTO 40
80 PRINT "FIN DU PROGRAMME"
```

Si vous travaillez selon ces critères, il y a moins de chances que votre programme comporte des erreurs et s'il y en avait quand même, vous les trouverez plus facilement.

UN EXEMPLE DE PROGRAMME

PROBLEME:- utiliser l'ordinateur pour simuler le jet de deux dès.

1re partie - Affichage titres et instructions

a) dégager l'écran,
b) imprimer le titre du programme,
c) imprimer les instructions pour l'utilisateur,
d) imprimer les entêtes, 1er dé, 2é dé.

2e partie - Trouver les valeurs des deux dès.

(quand vous jetez un dé, n'importe lequel des six côtés apparaît au hasard)
a) le premier dé est un nombre au hasard entre 1 et 6,
b) le second dé est un nombre au hasard entre 1 et 6.

3e partie - Afficher la réponse.

a) imprimer la valeur du premier dé, du deuxième dé.

4e partie - Arrêter le programme et le répéter si il y a lieu.

a) arrêter le programme,
b) demander un nouveau jet de dés,
c) répéter les parties 1, 2 et 3 selon nécessité.

En traduisant ceci en BASIC (avec quelques commentaires) nous obtenons:-

```
10 'PROGRAMME DE JET DE DES
20 '
30 'PREMIERE PARTIE
40 CLS: REM DEGAGER L'ECRAN
50 PRINT "JET DE DES": 'TITRE
60 PRINT
70 PRINT "UTILISER LA TOUCHE BREAK POUR ARRETER LE
   PROGRAMME": 'INSTRUCTIONS
80 PRINT
90 PRINT "1ER DE", "2E DE": 'ENTETES
100 PRINT
110 REM FIN DE LA PREMIERE PARTIE
```

REM

Le commandement REM s'utilise pour insérer des commentaires dans un programme. L'ordinateur ne tient compte de rien de ce qui suit REM (ou sa version abrigée ') sur cette ligne

```
10 REM CECI EST UNE LIGNE DE COMMENTAIRE
35 D = B*B-4*A*C: 'TROUVER LE NOMBRE SIGNIFICATIF
```

```
120 REM
130 REM DEUXIEME PARTIE
140 D1 = RND(6): ' JET 1ER DE
150 D2 = RND(6): ' JET 2E DE
160 ' FIN DE LA 2E PARTIE
170 '
180 ' TROISIEME PARTIE
190 PRINT D1, D2: ' AFFICHAGE RESULTAT
200 ' FIN 3E PARTIE
210 '
220 ' QUATRIEME PARTIE
230 INPUT "APPUYER SUR ENTER POUR JET DE DES"; A$:
240 GOTO 40: ' RETOUR AU DEPART
250 ' FIN QUATRIEME PARTIE
```

Ce programme n'a pas vraiment besoin de tous ces commentaires, écrivez maintenant votre propre version du même programme.

Sachez qu'il n'y a pas de bonne (ou mauvaise) version d'un programme. La bonne version est une version qui marche. Il peut y avoir des solutions plus élégantes ou plus efficaces de résoudre le même problème. En général, un programme plus court tourne plus vite et occupe moins de place en mémoire.

Nous terminons cette partie avec une version plus élaborée du même programme. Notez au passage comment le découpage d'un programme en plusieurs parties permet de l'assembler dans un ordre différent tout en obtenant le même résultat.

```
10  PROGRAMME DE JET DE DES
20  '
 30 ' PREMIERE PARTIE
 40 CLS0: PRINT @ 8, "JET DE DES";
 50 PRINT @ 167, "PREMIER ∇";: PRINT @ 178, "DEUXIEME";
 60 PRINT @ 200, "DE";: PRINT @ 211, "DE";
 70 PRINT @ 450, "TOUCHE BREAK POUR FIN";
 80 PRINT @ 358, "TOUCHE ENTER POUR JET";
 90 PRINT @ 358, "TOUCHE ENTER POUR JET";
100 INPUT A$
110 ' DEUXIEME PARTIE
120 D1 = RND(6): D2 = RND(6)
130 ' TROISIEME PARTIE
140 PRINT @ 265, D1;: PRINT @ 276, D2;
150 GOTO 90
```

CHAPITRE QUATRE

CONSERVATION ET REMANIEMENT DES PROGRAMMES

Installation du magnétophone

Certains programmes commencent à être assez longs et il est fastidieux de devoir taper le même programme chaque fois que l'on veut s'en servir. Il est pourtant assez simple de conserver les programmes sur une cassette de magnétophone et de les ramener dans la mémoire de l'ordinateur chaque fois que c'est nécessaire. Pour faire cela il vous faut un magnétophone à cassettes et un raccord. Ce raccord se compose d'un câble avec une prise DIN (ronde pentapolaire) à un bout et trois fiches de type jack à l'autre bout.

Tout magnétophone à cassettes de qualité moyenne peut être employé, à condition qu'il ait la possibilité de:

a) enregistrer à partir d'une source extérieure (prise type jack habituellement identifiée AUX ou LINE IN ou MIC).

b) sortir sur un haut-parleur extérieur ou écouteur (prise type jack identifiée EAR ou MONIT ou L/S ou SPKR (parfois par le croquis d'une oreille).

c) se mettre en marche et s'arrêter par commande à distance (prise jack plus petite habituellement identifiée REM ou TELEC, et disposée à côté du jack du paragraphe a).

d) fonctionner sur le courant du secteur. Ceci n'est pas essentiel mais des piles usées peuvent affecter sérieusement le succès des opérations de conservation et de reprise des programmes.

Pour raccorder le magnétophone à l'ordinateur, engager la prise DIN dans l'entrée marquée TAPE sur le côté gauche de l'ordinateur.

Les trois prises à l'autre bout du câble sont à raccorder au magnétophone comme suit:

1) Le petit jack dans la prise marquée REM ou TELEC (C'est une commande marche/arrêt à distance).

2) Le grand jack avec un fil gris dans la prise marquée AUX ou LINE IN ou MIC.

3) L'autre grand jack, avec fil noir, dans la prise marquée EAR (oreille).

Allumer le magnétophone, engager une cassette et rebobiner jusqu'au début de la bande. Maintenant mettre le volume à 6 (ou à un peu plus de la moitié de sa puissance maximale). Vous êtes maintenant prêt a stocker vos programmes.

STOKAGE D'UN PROGRAMME SUR BANDE

Tapez un programme, faites le tourner pour vous assurer qu'il marche bien puis procédez de la façon suivante:

1) Appuyez simultanément sur les touches PLAY et RECORD (soit marche et enregistrement, soit encore sur certains appareils → et la touche rouge) jusqu'à ce qu'ils s'enclenchent.

2) Si vous êtes en début de bande, faites avancer l'amorce à la main jusqu'à disparition, puis tapez sur l'ordinateur, l'instruction

CSAVE "PROGRAM 1" et appuyez sur [ENTER]

Le nom "PROGRAM 1" peut être remplacé par tout libellé de votre choix, (il doit commencer par une lettre et sa taille ne doit pas dépasser 8 caractères). Quand vous aurez tapé [ENTER] le moteur du magnétophone se mettra en route et votre programme va s'enregistrer. Après un moment, vous verrez apparaître à l'écran le message OK et le moteur du magnétophone s'arrêtera.

Le programme sera toujours dans la mémoire de l'ordinateur, mais une copie aura été déposée sur la bande. Vous avez ainsi stocké sur bande le programme PROGRAM 1 ou tout autre nom que vous lui aurez donné).

CHARGEMENT DE PROGRAMMES DANS LA MEMOIRE

Pour ramener en mémoire un programme conservé sur cassette, tapez d'abord NEW de façon à éliminer tout programme existant dans la mémoire. La marche à suivre est la suivante:-

1) Rembobinez la bande au début.
2) Appuyez sur le bouton PLAY ou → jusqu'à ce qu'il s'enclenche
3) Tapez l'instruction

CLOAD "PROGRAM 1" et appuyer sur [ENTER]

Le moteur du magnétophone va patir et la lettre S va apparaître au coin haut gauche de l'écran. Cela indique que l'ordinateur recherche le programme désigné. Quand il l'aura trouvé, le S se transformera en

F PROGRAM 1

Quand le message OK apparaît, le moteur du magnétophone s'arrête; le programme est chargé dans l'ordinateur. Pour vérifier sa présence, tapez LIST.

Si le programme n'est pas là, ou si vous recevez le message I/O ERROR, sur l'écran, il est possible que le réglage de votre magnétophone (volume, grave/aigüe s'il y en a) doive être modifié. Vérifiez les cables et les fiches et recommencez en modifiant le réglage du volume jusqu'à résultat satisfaisant.

CSAVE
CLOAD
SKIPF

Le commandement CSAVE conserve un programme sur une cassette de magnétophone. Le nom du du programme doit avoir huit caractères ou moins.

CSAVE "PROGRAM"

(En Anglais to save, conserver; CSAVE, c'est Cassette/Save).

Pour conserver des données sur cassettes, employer le paramétre supplémentaire A, l'information sera alors stockée en code ASCII. Elle pourra ensuite être lue par une instruction INPUT # - 1.

CSAVE "DATA", A

La commande CLOAD charge un programme désigné, de la cassette dans la mémoire. (en Anglais to load, charger; CLOAD c'est Cassette/Load).

CLOAD "PROGRAM"

La commande SKIPF saute au programme suivant celui qui est désigné, ou a la fin de ce dernier. (en Anglais to skip, sauter, dans le sens de sauter un passage dans un livre)

SKIPF "PROGRAM"

CONSERVER PLUSIEURS PROGRAMMES

Pour conserver plusieurs programmes sur la même bande, vous devez veiller à ne pas les enregistrer sur ceux qui s'y trouvent déjà. Il faut donc que la bande soit positionnée après la fin du dernier programme enregistré. Ceci se fait de la manière suivante:-

1) Rembobinez la bande à son début.
2) Appuyez sur PLAY ou → jusqu'à ce que la touche s'enclanche.
3) Tapez l'instruction.

SKIPF "PROGRAM 1"

Le moteur va partir, l'ordinateur va chercher (S) le programme, le trouver (F), le parcourir entièrement puis arrêter le moteur et vous restituer le message OK.

4) Appuyez sur la touche STOP.
5) Appuyez sur PLAY ou RECORD en même temps, donnez un nom au nouveau programme et conservez le par CSAVE.

Après avoir conservé ce programme, la bande est positionnée après sa fin donc vous pouvez directement taper d'autres programmes et les conserver, si vous le souhaitez.

CONSEILS POUR UN ENREGISTREMENT FIABLE

1) Faites marcher le magnétophone sur le secteur afin que sa vitesse demeure constante.
2) Utilisez des casettes neuves et de bonne qualité. Bien qu'il puisse vous sembler que des cassettes de longue durée (C120) soient plus pratiques, il vaut mieux utiliser des bandes plus courtes (C30 ou C12).
3) Lancez toujours la recherche en début de bande. Ne vous fiez pas au compteur du magnétophone.
4) Ne laissez pas les touches PLAY ou RECORD enfoncées inutilement. Faites STOP dès que vous avez fini de conserver ou de charger un programme.
5) Rebobinez les cassettes avant de les ranger.
6) Etiquettez vos cassettes tout de suite après avoir conservé un programme. Pour les programmes importants cassez la languette de protection pour éiter un effacement malencontreux.

Même si vous êtes très soigneux, des accidents peuvent arriver. Certains programmes représentent, en termes de temps et d'efforts, un investissement considérable. Conservez donc une deuxième copie sur une autre cassette.

L'EDITEUR

Plus le programme devient long, plus il y a des chances de faire des erreurs de frappe. Jusqu'à ce stade, le seul reméde était de re-taper toute la ligne. Ceci ne sera désormais plus nécessaire car nous allons vous présenter l'EDITEUR.

L'EDITEUR vous permettra de vous déplacer vers l'avant ou l'arrière, le long d'une ligne en changeant, enlevant ou insérant des caractères. Pour appeler l'EDITEUR, tapez

EDIT *numéro de ligne* [ENTER]

numéro de ligne étant le numéro de la ligne de programme sur laquelle vous entendez travailler. La ligne sera affichée, en entier sur l'écran. Puis le numéro de ligne sera imprimé avec le curseur clignotant à côté de lui.

DEPLACEMENT LE LONG D'UNE LIGNE

Le curseur est maintenant en début de ligne. Pour avancer le long de la ligne, appuyez sur la barre d'espacement. Le curseur se déplace en laissant affichés les caractères sur lesquels il est passé. Pour retourner en arrière, nous pouvons utilisr le rappel arrière [←]. Vous pouvez accélérer le mouvement en tapant un nombre puis la touche voulue, (c'est à dire [5] [SPACE] décalera le curseur de 5 caractères vers l'avant; [3] [←] décalera le curseur de 3 en arrière). En utilisant ces deux touches vous pouvez positionner le curseur sur tout caractère figurant dans la ligne. Vous ne verrez pas ce caractère parce que le curseur clignotera en surimpression. Deux autres instructions permettent de sauter directement à un endroit déterminé de la ligne. Pour allonger la ligne, tapez [X]. Le curseur sautera directement en fin de ligne et il vous suffira de taper ce que vous vouliez ajouter. En utilisant le dispositif de recherche, vous pourrez déplacer directement le curseur sur un caractère visé. Tapez [S] suivi du caractère que vous voulez atteindre. ([S] [A] déplacera le curseur sur le *premier* A qui se présentera sur la ligne). S'il y a plus d'un A sur cette ligne et que vous vouliez atteindre le troisième, alors [3] [S] [A], y positionnera le curseur.

MODIFICATIONS

Maintenant que le curseur a été mis en position par l'usage de la barre d'espacement, du rappel arrière ou de la touche [S], vous pouvez:-

a) *Effacer* un caractère en tapant [D]. Ceci effacera le caractère sous le curseur. Pour effacer plus d'un caractère, [5] [D] effacera 5 caractères, à partir de la position du curseur.

b) *Changer* un caractère en tapant [C], suivi du nouveau caractère. ([C] [F] changera le caractère qui se trouve sous le curseur et le transformera en F.) [3] [C] remplacera les trois prochains caractères par ceux qui auront été tapés.

c) *Insérer* des caractères en tapant [I] suivi des caractères que vous voulez insérer. Lorsque vous tapez [I], l'éditeur se met en mode d'insertion.

EDIT

Le commandement EDIT s'utilise pour changer le contenu d'une ligne déterminée.

EDIT *numéro de ligne*

Lorsque l'ordinateur est en mode EDIT toutes les instructions d'édition ci après peuvent être employées.

L	Affiche une ligne donnée
C *caractère*	Change un caractère
n C *caractère*	Change les *n* caractères suivants en nouveaux caractères
I	Insére des caractères
D	Efface les caractères existants
n D	Efface les *n* caractères suivants
H	Efface le reste de la ligne à partir de la position du curseur et se met en mode d'insertion
X	Allonge la ligne, se déplace à la fin et se met en mode d'insertion
S *caractère*	Cherche la première position ou ce caractère apparaît
K	Efface le reste de la ligne à partir de la position occupée par le curseur
n K *caractère*	Efface la ligne jusqu'à la nième apparition d'un caractère
n [BARRE D'ESPACEMENT]	Avance le curseur de *n* espaces, si *n* est omis, avance de 1
n [←]	Fait reculer le curseur de *n* espaces. Si *n* est omis, recule de 1
[SHIFT] [↑]	Met fin au mode d'insertion et retourne au mode d'édition
[ENTER]	Met fin à l'opération d'édition, rentre la ligne en mémoire et retourne au clavier.

Dans cette situation, tout ce qui est tapé est inséré dans la ligne. Pour quitter le mode d'insertion et retourner à l'éditeur, vous taperez [SHIFT] et [↑] ensemble.

a) Affichez une *liste* de la ligne dans sa forme actuelle en tapant [L]. La ligne sera affichée avec tous les changements que vous aurez faits jusqu'ici. Après l'affichage d'une ligne, le curseur revient au début.

b) Quitter l'éditeur en appuyant sur [ENTER]. Ceci placera la ligne modifiée dans le programme en mémoire et vous restituera le commentaire OK. Vous pouvez quitter l'éditeur à tout moment au moyen de [ENTER].

Voici un exemple d'un travail d'édition. Cela peut avoir l'air un peu compliqué, à première vue, mais vous serez surpris de la rapidité avec laquelle vous y deviendrez accoutumé, avec un minimum de pratique. Les touches que vous devez appuyer sont entre crochets.

Il s'agit d'écrire correctement la phrase anglaise "THE ARO NO MISTAKES IN THIS LINE" (Il n'y a pas d'erreurs dans cette ligne, mais elle a été malencontreusement écrite comme on le voit à la ligne 10

[E] [D] [I] [T] [1] [Ø] [ENTER]

10 PRINT ∇ "THEIR ∇ ARE ∇ MANY ∇ MISTOOK ∇ TIN ∇ LINE

10 ■	(■ donne la position du curseur)
[1] [Ø] [SPACE]	avance de dix vers l'avant, (ou [2] [S] [I])
[C] [R] [C] [E]	change le I en R et le R en E
[S] [M]	passer au début de MANY
[2] [D]	effacer M et A
[SPACE] [C] [O]	sauter le N et changer le Y en O
[S] [O]	passer au premier O de Mistook
[D] [C] [A]	effacer le premier O et changer le second en A
[SPACE] [I] [E] [S]	passer après K et insérer ES
[SHIFT] [↑]	quitter le mode d'insertion
[S] [L]	passer au L de Line
[I] [T] [H] [I] [S] [SPACE]	insérer THIS et un espace avant LINE
[SHIFT] [↑]	quitter le mode d'insertion
[X]	passer en bout de ligne (vous êtes maintenant en mode insertion).
[SPACE] [N] [O] [W] ["]	ajouter NOW et les guillemets pour fermer la chaîne
[SHIFT] [↑]	quitter le mode d'insertion
[L]	afficher la forme actuelle de la ligne

10 PRINT ∇ "THERE ARE NO MISTAKES IN THIS LINE NOW"

10 ■

[ENTER]	mettre la ligne dans le programme et quitter l'éditeur.

Il y a deux autres commandements d'éditeur. Ils doivent être utilisés avec précautions.

a) [K] (pour kill = anéantir). En tapant [K] on efface le reste de la ligne, à partir de la position du curseur. [K] suivi d'un caractère effacera toute la ligne, depuis le curseur jusqu'à la première apparition de ce caractère. [3] [K] [A] effacera jusqu'à la troisième apparition du caractère A.

b) [H] (pour hack = hacher). En tapant [H] on annule le reste de la ligne, à partir de la position du curseur et on passe en mode d'insertion. C'est utile pour re-taper la fin d'une ligne.

Si votre frappe est parfaite et si vous ne faites jamais d'erreur de programmation, vous pouvez ignorer ce chapitre. Pour les autres, l'EDITEUR nous facilitera beaucoup la vie, à partir de maintenant. Tapez l'un des exemples qui vous ont été donnés, éventuellement conservez le sur bande puis transformez le en un autre programme en vous servant de l'EDITEUR.

D'AUTRES INSTRUCTIONS DE SYSTEME

Des instructions comme LIST et RUN sont des instructions de système. Elles ne font pas partie du programme mais donnent instructions à votre ordinateur de faire immédiatement quelque chose. En voici quelques autres qui rendent la programmation plus facile.

DEL (delete = effacer)

Pour effacer une ligne d'un programme on peut taper le numéro de la ligne suivi de [ENTER], Ca va pour une ou deux lignes, mais s'il faut en éliminer 30 ou 40?...

DEL *numéro de ligne - numéro de ligne*

va effacer un bloc entier de lignes, en commençant par le premier numéro indiqué jusqu'au deuxième numéro indiqué, *inclus*.

DEL 100 - 250

va effacer toutes les lignes de 100 à 250, inclus.

La commande DEL peut ègalement s'utiliser sous d'autres formes.

DEL 20	effacement de la ligne 20 uniquement
DEL 30 -	effacement de toutes les lignes, de la ligne 30 à la fin du programme
DEL - 200	effacement de toutes les lignes du début du programme jusqu'à et y compris 200
DEL -	effacera le programme entier.

DEL

L'instruction DEL s'utilise pour annuler certaines lignes déterminées dans le programme en mémoire.

DEL *numéro de ligne 1* - numéro de ligne 2

Cette instruction annulera toutes les lignes à partir de la ligne 1 jusqu'à et y compris la ligne 2. L'instuction peut aussi s'utiliser autrement:-

DEL -	Annule le programme entier
DEL - 100	Annule les lignes du début à la ligne 100
DEL 300 -	Annule les lignes de 300 jusqu'à la fin
DEL 40	Annule la ligne 40 seulement
DEL 100 - 200	Annule les lignes entre 100 et 200

RENUM

L'instruction RENUM permet de renuméroter tout ou partie des lignes de programmation. RENUM change également les numéros dans les instructions de branchements (GOTO etc...) afin que le programme puisse continuer à bifurquer vers les endroits prévus.

RENUM *nouvelle ligne, ligne de départ, incrément*

Cette commande va renuméroter toutes les lignes en commençant par la *ligne de départ*, à laquelle elle donnera comme numéro celui de *nouvelle ligne* et en augmentant le numéro de chaque ligne suivante du chiffre contenu dans *incrément*. La commande peut aussi marcher autrement, sous forme ci-après:-

RENUM	Renumérote tout le programme. Les lignes seront renumérotées de 10 en 10, soit 10, 20, 30...
RENUM 100	Renumérote tout le programme à partir du numéro 100, soit 100, 110, 120...
RENUM 100,50,5	Renumérote en commençant à l'ancienne ligne 50. Les porteront les numéros 100, 105, 110...
RENUM,,20	Renumérote le programme entier. Leslignes porteront les numéros 10, 30, 50

Il y a lieu de remarquer que si un paramètre a été omis et qu'un paramètre suivant est utilisé, il faut mettre une virgule. L'on ne peut pas employer RENUM pour intervertir les lignes.

RENUM (renumber = renumeroter)

L'instruction RENUM renumérotera tout ou partie des lignes dans votre programme, il changera également les numéros des GOTO, GOSUB, IF THEN, ON GOTO et ON GOSUB afin que les branchements s'effectuent à la même place qu'avant. Nous ferons connaissance avec ces déclarations plus tard.

RENUM, *nouvelle ligne, ligne de départ, incrément*

Nouvelle ligne est le nouveau numéro à donner à la première ligne qui sera renumérotée.
Ligne de départ est le numéro de ligne à partir duquel vous voulez commencer à renuméroter et l'*incrément* est l'intervalle numérique imposé entre lignes consécutives. Tous ou certains de ces paramètres peuvent être omis. Si vous omettez *nouvelle ligne*, c'est 10 qui sera utilisé. Si vous omettez *ligne de départ*, le programme tout entier sera renuméroté.
En omettant l'*incrément* le renumérotage se fera de 10 en 10.

RENUM	renumérote tout le programme en 10, 20, 30...
RENUM 100,50,5	renumérote à partir de 50 en 100, 105, 110... Toutes les lignes avant 50 seront inchangées
RENUM 110, 2	renumérote le programme entier en 110, 112, 114
RENUM,,5	renumérote le programme entier en 10, 15, 20...

Il est à noter que si vous omettez l'un des paramètres mais voulez spécifier le suivant, il y a lieu de mettre une virgule.

TRON-TROFF

Trace on et Trace off, suivre le programme à la trace.

S'il vous arrive quelques fois d'avoir des difficultés avec un programme, il peut être utile de savoir quel trajet il suit à travers les lignes d'instructions. L'usage de la possibilité de traçage va vous permettre de faire cela.
En tapant TRON *avant* de faire partir le programme, vous mettez le traceur en route. Le numéro de ligne où le programme se trouve va maintenant s'afficher sur l'écran au fur et à mesure que le programme y arrive. Ceci vous permettra de vous rendre compte si le programme bifurque comme vous l'avez prévu, aux bons endroits.
Pour arrêter le traceur, tapez TROFF

STOP

Un programme peut être arrêté en un point de son déroulement en incluant une ligne comportant l'instruction STOP.

185 STOP

Cette ligne fera s'arrêter le programme lorsqu'il atteindra la ligne 185. Un message apparaîtra sur l'écran vous indiquant que le programme s'est arrêté et à quelle ligne. Vous pouvez maintenant examiner le contenu de toute variable en utilisant PRINT ou ?.
Pour faire repartir le programme taper CONT (continue) et le programme reprendra à partir de la ligne suivant le STOP.

Vous pouvez maintenant renuméroter les programmes, éliminer les lignes inutiles, changer le contenu des lignes, consever le résultat sur cassette. Comme nous savons maintenant travailler les programmes, nous allons commencer à en construire qui fassent des choses plus intéressantes que des écrans de couleurs ou des bruits bizarres.

TRACE

Le déroulement du *programme* peut être suivi en utilisant le traceur. Lorsque chaque ligne est atteinte, le numéro de celle-ci s'affiche sur l'écran. Le traceur doit être mis en service avant de faire partir le programme.

TRON	met le traceur en service
TROFF	met le traceur hors service

Tous deux sont des instructions directes et ne nécessitent pas de numéro de ligne.

END
STOP
CONT

L'instruction END termine l'exécution d'un programme et rend le contrôle au clavier.

L'instruction STOP arrête l'exécution d'un programme à la ligne contenant l'instruction STOP. Un message: BREAK AT N apparaît alors sur l'écran pour indiquer que l'arrêt est intervenu à la ligne N. Pour faire repartir le programme, utiliser CONT (continue), sans numéro de ligne. Le programme continuera son déroulement à partir de la ligne suivant le STOP. Un programme ne continuera pas après la déclaration END. (Fin). Il faudra le faire repartir par RUN, c'est à dire à son début.

CHAPITRE CINQ

BIFURCATIONS ET EMBRANCHEMENTS

A la fin du chapitre 3, nous avons discuté la façon de construire un programme en différentes sections. Maintenant nous allons nous occuper de la façon dont il faudra organiser le parcours du programme pour qu'il assemble les différentes sections. Cela se fait au moyen de *branchement*. Nous avons déjà rencontré l'instruction de *branchement* GOTO. C'est un branchement inconditionel, puisque dès que le programme rencontre GOTO, il saute immédiatement à la ligne désignée, et continue à partir de là. Le programme n'a pas de choix: GOTO lui signifie de se rendre tout de suite quelque part, pas peut être ou éventuellement. Jusqu'à présent nous avons principalement utilisé GOTO pour retourner au début du programme. Bien que la possibilité de répéter sans cesse une partie du programme soit très utile, il est peu probable que nous voulions faire cela indéfiniment. (De plus il n'est pas de bonne pratique d'avoir a compter sur la touche BREAK pour arêter un programme). Heureusement, le language BASIC nous fournit plusieurs commandements possibles pour nous permettre de contrôler le déroulement du programme.

SELECTION DES OPTIONS

Le premier de ces commandements est constitué par une extension de notre vieil ami GOTO. C'est ON...GOTO qui prend la forme suivante:-

ON *expression numérique* GOTO *liste de numéros de lignes*

L'expression numérique est évaluée et, si nécessaire, ramenée à un nombre entier (c'est à dire que les chiffres après la décimale sont ignorés). Le contrôle du programme est alors transféré à l'un des numéros de lignes sur la liste. Si l'expression s'évalue à 1, le programme part au premier numéro de la liste, à 2, au deuxième etc... Si la valeur de l'expression est inférieure à 1 ou supérieure au nombre de numéros de lignes figurant dans la liste, l'ordinateur ignorera l'instruction et continuera à la ligne suivante. Toutefois, une valeur négative de l'expression provoquera l'arrêt du programme avec un message d'erreur. Il est habituellement recommandé de vérifier que la valeur de l'expression numérique reste dans les limites de validité avant d'atteindre la déclaration ON...GOTO. Voici quelques exemples

```
140 ON P GOTO 200, 300, 400
210 ON X - 4 GOTO 20, 40, 700, 10, 690
185 ON B*C/D - E GOTO 115, 285, 900, 40
```

L'emploi de ON...GOTO est un moyen utile pour sélectionner parmi plusieurs options, car il peut être considéré comme un branchement conditionnel. Nous nous en servirons de cette manière dans l'exemple figurant à la prochaine section.

ON...GOTO

Le commandement ON...GOTO opére un branchement multidirectionnel vers des lignes dont les numéros sont désignés.

ON *expression* GOTO *numéros de lignes*

L'expression est évaluée (et au besoin ramenée à un nombre entier). Le programme bifurque alors vers le numéro de ligne figurant dans la liste dont la position ordinale est égale à la valeur de l'expression. Si la valeur de l'expression donne quatre, la commande ON...GOTO sélectionnera le quatrième numéro de ligne dans sa liste. La valeur de l'expression ne doit pas être négative, sinon il en résultera un message d'erreur. Si la valeur de l'expression donnait zéro ou qu'elle soit supérieure au nombre de numéros de ligne dans la liste, la déclaration de ON...GOTO sera ignorée et le programme continuera à la prochaine ligne.

```
10 CLS: PRINT "EQUATION DU SECOND DEGRE"
20 INPUT "A,B,C"; A,B,C: IF A = 0 THEN 20
30 R = - B/(2*A): D = B*B-4*A*C:S = SGN(D)
40 P = SQR(D*S)/(2*A)
50 ON S + 2 GOTO 80, 60, 70
60 PRINT "RACINES DOUBLES": PRINT R,R: END
70 PRINT "RACINES REELLES": PRINT R + P, R - P: END
80 PRINT "RACINES COMPLEXES": PRINT R,R: PRINT P, -P: END
```

DECISIONS

Une forme encore plus utile du branchement conditionnel est l'emploi de l'instruction IF...THEN. C'est probablement l'instruction la plus performante du language BASIC. Comme telle, elle peut prendre la forme la plus simple, comme la plus complexe. Dans sa forme la plus simple:-

IF *condition* THEN *numéro de ligne*

elle exprime que si (IF) la condition est remplie, alors (THEN) l'ordinateur doit passer à la ligne indiquée, sinon il doit continuer le programme à la ligne suivante.

```
120 IF D > 9 THEN 250
180 IF A$ = "YES" THEN 600
```

Dans la ligne 120, ci-dessus, le programme transférera le contrôle à la ligne 250 si, *et seulement si* la valeur de D est supérieure à 9. Si D est plus petit ou égal à 9, le programme continuera à la ligne suivante. Dans la ligne 130 le branchement vers la ligne 600 ne s'effectuera que si la chaîne A$ est constituée par les caractères YES. La correspondance dans ce cas doit être tout à fait conforme. YESS, YEA, YER, ou OK ne feront pas l'affaire (même 'yes' en minuscule ne marchera pas) et le programme n'effectuera pas le branchement.

Dans sa forme complète la déclaration IF...THEN se présente comme suit:-

IF *condition* THEN *action 1* ELSE *action 2*

(Else veut dire autrement, en Anglais)
Ce qui veut dire pour l'ordinateur 'SI la condition est remplie, ALORS effectuer l'action 1, si la condition n'est pas remplie alors effectuer l'action 2'.

```
210 IF P = 3 THEN PRINT "VRAI" ELSE PRINT "FAUX"
```

Dans cet exemple, VRAI va seulement apparaître si P est égal à 3, si P a *toute* autre valeur, FAUX va être affiché. Dans les deux cas, le programme continuera à la ligne suivante. Il est possible que les deux actions à entreprendre, soient constituées de plus d'une instruction.

```
210 IF P = 3 THEN PRINT "VRAI": R = R + 1: GOTO 560
    ELSE PRINT "FAUX": L = L + 1
```

Si P est égal à 3 l'ordinateur imprimera VRAI, ajoutera 1 à la variable R et continuera le programme à la ligne 560. Pour toute autre valeur il imprimera FAUX, ajoutera 1 à L et continuera à la ligne suivante.

L'ordinateur acceptera pour l'*action* 1 et l'*action* 2 toute instruction conforme au BASIC, y compris d'autres IF...THEN, si vous le souhaitez. La seule contrainte c'est que l'instruction IF...THEN au complet doit tenir sur une seule ligne. (Une ligne peut contenir un maximum de 256 caractères, y compris le numéro de ligne).

Nous avons jusqu'ici examiné ce qui se passe après qu'une condition ait été évaluée. La décision a prendre par l'ordinateur, repose sur la vérification de la condition. Une condition peut être VRAIE ou FAUSSE, rien d'autre. (L'ordinateur donne la valeur 1 à une condition vraie et la valeur Ø à une condition fausse). Une condition simple a la forme

expression 1 relation expression 2

expression 1 et *expression 2* sont des expressions habituelles, du language BASIC. Les deux expressions doiventêtre du même type. (toutes deux numériques ou toutes deux en chaînes). Une *relation* peut être l'une quelconque des suivantes:

SIGNIFICATION	SYMBOLE	EXEMPLE
Egale a	=	60 IF X = Y + 2 THEN 100
Plus petit que	<	110 IF A*B+2<C/2 THEN 20
Plus grand que	>	185 IF A$>B$ THEN PRINT A$
Plus petit ou égal à	<=	220 IF 4*W9<=B/Z9 THEN A = A -1
Plus grand ou égal à	>=	415 IF B7>=Ø THEN X = Ø
Différent de	<>	80 IF A$ <> "OUI" THEN 999

Sachez qu'une relation peut tout aussi bien s'appliquer à des chaînes qu'à des valeurs numériques. Quand on compare des chaînes, chaque caractère est vérifié à son tour, et dès lors, des instructions IF...THEN peuvent être utilisées pour effectuer des comparaisons d'ordre alphabétique.

La condition "A" < "B" est vraie, parce que la lettre A vient avant la lettre B dans l'alphabet. La condition "AAA" < "AA" est fausse parce que AAA apparaîtrait après AA, dans un dictionnaire, par exemple.

Les conditions peuvent être encore élargies en combinant deux ou plusieurs conditions par l'usage des opérateurs AND, OR

270 IF A = 4 AND B = 7 THEN 500

L'opérateur AND (= et, en Anglais) veut dire que les deux conditions doivent être vraies, en même temps pour que toute l'expression soit considérée comme vraie. Dans l'exemple, si A est égal à 4 et que B ait toute autre valeur que 7, l'ensemble sera considéré comme faux et le programme continuera à la ligne suivante.

L'opérateur OR (= ou, en Anglais) veut dire que si l'une des conditions est vraie, la totalité de l'expression sera considérée comme vraie.

320 IF D<3 OR F+G>25 THEN 100

Le programme va bifurquer vers la ligne 100 si D est plus petit que 3, quelle que soit la valeur de P + G. Alternativement, il bifurquera vers la ligne 100 si F + G est plus grand que 25, quelle que soit la valeur de D.

IF...THEN...ELSE

La forme complète du commandement IF est la suivante.

IF *condition* THEN *action 1* ELSE *action 2*

L'instruction vérifie la *condition* qui sera VRAIE ou FAUSSE. Si elle est vraie l'action 1 sera effectuée, si elle est fausse, c'est l'action 2 qui le sera.

Une *condition* est composée d'une expression, une relation et une expression. Les expressions peuvent être toutes expressions BASIC du même type (c'est à dire deux expressions numériques ou deux expressions de chaînes). Une relation est tout opérateur de la liste ci-dessous:

= Egal à	<> Différent de
< Plus grand que	< Plus petit que
>= Plus grand ou égal à	<= Plus petit ou égal à

Les conditions peuvent également être combinées, au moyen des opérateurs logiques AND, OR, NOT.

condition AND *condition:* VRAI uniquement si les deux conditions sont vraies.
condition OR *condition:* VRAI si l'une des condition est vraie.
NOT *condition:* VRAI si la condition est fausse.

L'*action 1* et l'*action 2* peuvent se composer de toute instruction BASIC, y compris une autre instruction IF...THEN.

La partie ELSE de l'instruction est optionnelle et peut donc être omise. Dans ce cas le programme continue à la ligne suivante si la condition est FAUSSE.

```
10 CLS: PRINT @ 9, "JEU DE DEVINETTE": N = RND(100):T = 0
20 INPUT "DEVINEZ MON CHIFFRE"; G: T = T + 1
30 IF G = N THEN PRINT "JUSTE EN"; T; "ESSAIS": END
40 IF G>N THEN PRINT "NON C'EST PLUS PETIT" ELSE PRINT
   "NON C'EST PLUS GRAND"
50 GOTO 20
```

INKEYS

Le commandement INKEYS est une *fonction.* Son rôle est de scruter le clavier pour voir si une touche a été frappée; si tel est le cas le caractère correspondant est *restitué.*

INKEYS peut s'utiliser pour placer un caractère unique dans une chaîne; il n'est pas nécessaire de faire suivre par [ENTER].

```
10 CLS0: PRINT @ 5, "APPUYEZ SUR UNE TOUCHE ET JE";
20 PRINT @ 38, "VOUS DIRAI LAQUELLE C'ETAIT ▽";
30 B$ = "VOUS N'APPUYEZ SUR AUCUNE TOUCHE ▽▽"
40 C$ = "LA TOUCHE QUE VOUS AVEZ APPUYEE ETAIT:- ▽"
50 A$ = INKEY$
60 IF A$ = ""THEN PRINT @ 193, B$; ELSE PRINT @ 193, C$; A$;
70 FOR D = 1 to 600: NEXT D: GOTO 50
```

Le programme qui va suivre est une version simplifiée d'un programme d'enseignement courant. Les commentaires, dans le programme indiquent ce que chaque section du programme va faire. Notez au passage, l'utilisation de ON...GOTO pour sélectionner les options et de IF...THEN pour vérifier l'arithmétique. A la ligne 800 nous vous présentons un mot nouveau: INKEYS. Ce commandement scrute le clavier pour voir si une touche a été frappée. Si c'est le cas, le caractère correspondant à la touche sera stocké dans A$. Après une entrée sur INKEYS, [ENTER] n'est pas nécessaire. Voir à ce sujet la page encadrée concernant INKEYS.

Le programme à l'air beaucoup plus long qu'il n'est en réalité en raison de ce que plus de la moitié en est consacré à des commentaires. Pour économiser de l'espace mémoire dans l'ordinateur nous ne mettrons plus à partir de maintenant autant de commentaires dans les programmes.

```
 10 REM EXERCICE D'ARITHMETIQUE
 20 REM
 30 REM METTRE LES COMPTEURS A ZERO & TROUVER NIVEAU DE
    DIFFICULTE
 40 REM
 50 R = 0: W = 0: CLS
 60 T$ = "EXERCICE D'ARITHMETIQUE"
 70 PRINT @ 6, T$
 80 PRINT @ 64, "▽ ▽ QUEL NIVEAU DE DIFFICULTE SOUHAITEZ
    VOUS?"
 90 PRINT @ 128, "▽▽ UN NOMBRE ENTRE 1 ET 10": INPUT L:L1 = 10*L -
    1
100 REM
110 REM AFFICHER LES OPTIONS
120 REM
130 CLS: PRINT @ 6, T$
140 PRINT @ 71, "1. ▽ ADDITION."
150 PRINT @ 103, "2. ▽ SOUSTRACTION."
160 PRINT @ 135, "3. ▽ MULTIPLICATION."
170 PRINT @ 167, "4. ▽ DIVISION."
180 REM
190 REM POSITION DE L'IMPRESSION
200 REM
210 P = 224: Q = 352
220 REM
230 REM SELECTIONNER LES OPTIONS
240 REM
250 PRINT @ P, "▽ ▽ LEQUEL VOULEZ-VOUS ESSAYER ▽";: INPUT
260 REM
270 REM SELECTIONNER DEUX NOMBRES POUR UN PROBLEME
280 REM
```

```
290 N1 = RND (L1): N2 = RND (L1)
300 REM
310 REM BRANCHEMENT SUR L'OPTION
320 REM
330 ON A GOTO 390, 460, 530, 600
340 REM
350 REM FAUSSE OPTION - ESSAYEZ A NOUVEAU
360 REM
370 CLS: SOUND 160,3: GOTO 130
380 REM
390 REM SECTION ADDITION
400 REM
410 PRINT @ P, "∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ADDITION."
420 PRINT @ Q, "∇ ∇ COMBIEN FONT"; N1; "PLUS ∇"; N2;: INPUT N4
430 N3 = N1 + N2
440 IF N4 = N3 THEN 690 ELSE 730
450 REM
460 REM SECTION SOUSTRACTION
470 REM
480 PRINT @ P, "∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ SOUSTRACTION."
490 PRINT @ Q, "COMBIEN FONT"; N1; "MOINS"; N2;: INPUT
500 N3 = N1 - N2
510 IF N4 = N3 THEN 690 ELSE 730
520 REM
530 REM SECTION MULTIPLICATION
540 REM
550 PRINT @ P, "∇ ∇ ∇ ∇ ∇ MULTIPLICATION."
560 PRINT @ Q, "COMBIEN FONT"; N1; "MOINS"; N2;: INPUT N4
570 N3 = N1 * N2
580 IF N4 = N3 THEN 690 ELSE 730
590 REM
600 REM SECTION DIVISION
610 REM
620 PRINT @ P, "∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ DIVISION
630 PRINT @ Q, "COMBIEN FONT"; N1; "DIVISE PAR"; N2;: INPUT N4
640 N3 = N1/N2
650 IF N3 = N4 THEN 690 ELSE 730
660 REM
670 REM REPONSE JUSTE
680 REM
```

```
690 R = R + 1: PRINT @ P, "∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ JUSTE.": GOTO 770
700 REM
710 REM REPONSE FAUSSE
720 REM
730 W = W + 1: PRINT @ P, "∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ ∇ FAUX.":
    PRINT @ P + 32,
    "∇ ∇ ∇ ∇ ∇ LA BONNE REPONSE EST ∇"; N3: GOTO 770
740 REM
750 REM VOIR S'IL Y A REPETITION ET DEGAGER LES LIGNES
760 REM
770 FOR D = 1 TO 600: NEXT D
780 PRINT @ P, "∇": PRINT @ P + 32, "∇": PRINT @ Q, "∇"
790 PRINT @ P, "VOULEZ VOUS RECOMMENCER? (O/N)"
800 A$ = INKEY$: IF A$ = ""THEN 800
810 IF A$ = "Y" THEN 130
820 REM
830 REM DONNER LES RESULTATS ET FIN
840 REM
850 CLS: PRINT @ 128, "VOUS AVEZ EU"; R; "JUSTE(S) ET";
    W; FAUTE(S)"
860 END
```

RECOMMENCEZ ENCORE ET ENCORE

Dans de nombreux cas un programme aura besoin de répéter plusieurs fois une même séquence de lignes. L'instruction de branchement suivante va nous permettre de le faire. Entrez ce petit programme et faites le tourner par RUN.

```
10 CLS
20 FOR I = 1 TO 50
30 PRINT @ 198, "COMPTEUR I ="
40 PRINT @ 214, I
50 NEXT I
60 PRINT @ 396, "FIN DE BOUCLE"
```

Et comme ça tourne trop vite pour qu'on puisse voir quelque chose, ajoutez la ligne,

```
45 FOR J = 1 to 100: NEXT J
```

Comme vous le voyez, le programme compte de 1 à 50. Les lignes qui sont répétées sont les lignes 30, 40 et 45. Une séquence de répétition comme celle-ci s'appelle une 'boucle' parce que le programme revient sur lui-même, dans le cas présent, à la ligne 20. Les instructions qui

contrôlent la boucle sont à la ligne 2Ø, (le haut de la boucle) et à la ligne 5Ø (le bas de la boucle). La ligne 2Ø signifie: "Pour (FOR) toutes les valeurs de I entre 1 et 5Ø, par pas de 1, exécuter toutes les instructions suivantes du programme, jusqu'à rencontrer l'instruction NEXT (passez à la valeur suivante)". La variable I se comporte comme un compteur et prend successivement les valeurs 1, 2, 3, 4.... 5Ø. Nous avons ajouté la ligne 45 parce que le comptage se faisait trop vite. Cette ligne constitue elle même une boucle dont la variable est J et n'a rien d'autre à faire qu'à compter de 1 à 1ØØ. Le temps que cela prend est suffisant pour ralentir suffisament l'autre boucle pour que nous puissions lire les nombres sur l'écran. Si maintenant nous changeons la ligne 2Ø en:

```
2Ø FOR I = 1 TO 5Ø STEP 2
```

le compteur va maintenant compter à partir de 1, de deux en deux (soit 1, 3, 5, 7... 49). Ajouter STEP nous permet de décider de l'incrément du compteur, c à d par tranches de combien il doit compter. (en Anglais STEP veut dire pas). Si le mot STEP ne figure pas, le compteur en déduit que vous voulez compter un par un. Maintenant modifiez l'exemple en tapant les lignes suivantes.

```
15 INPUT "DEBUT, FIN, PAS"; A, B, C
2Ø FOR I = A to B STEP C
7Ø PRINT @ 428, "AND I ="; I
```

Faites tourner le programme par RUN pour différentes valeurs de début, fin et pas. Essayez les valeurs suivantes.

DEBUT	FIN	PAS
5Ø	1	-1
-5Ø	5Ø	5
1	1Ø	Ø.5
2.6	3.9	Ø.1
1	-2	1

Vous remarquerez que vous pouvez obtenir un comptage en avant ou en arrière avec le pas que vous souhaitez. La seule règle est que si le pas est positif le début doit être plus petit que la fin et si le pas est négatif alors le début doit être plus grand que la fin. Dans le dernier des exemples mentionnés ci-dessus (1, -2, 1) vous remarquerez que si vous transgressez la régle, (puisqu'il n'est pas possible de compter de 1 à -2 par pas de + 1) la boucle sera quand même exécutée une fois.

Les boucles peuvent être insérées les unes dans les autres (la boucle de la ligne 45 est insérée à l'intérieur de l'autre boucle qui commence à la ligne 2Ø). Vous devez vous assurer que la fermeture des boucles s'effectue dans l'ordre approprié.

```
20 FOR J = 1 TO 10
30 FOR J = - 2 TO 4 STEP 0.6
40 FOR K = 1 TO 0 STEP -0.1
.
.
.
.
.
100 NEXT K
110 NEXT J
120 NEXT I
```

Chaque instruction FOR doit avoir un NEXT qui lui corresponde, la variable placée après NEXT indiquant à quel FOR elle appartient. Les boucles doivent être fermées dans l'ordre inverse de celui dans lequel elles ont été ouvertes. Si les trois lignes de programmation au-dessus étaient,

```
100 NEXT J
110 NEXT K
120 NEXT I
```

le programme s'arrêterait en affichant une message d'erreur car la boucle J et la boucle K se chevauchent.

Si toutes les boucles se ferment à la même place, les instructions NEXT peuvent se combiner ainsi:

```
100 NEXT K, J, I
```

mais l'ordre des variables doit être respecté.

Les variables utilisées pour créer une boucle, (dans l'exemple I, A, B et C) peuvent être utilisées à l'intérieur de la boucle, mais vous n'êtes pas autorisé à modifier le début, la fin ni le pas. Le compteur peut être changé en cours de route en figurant à gauche d'une déclaration d'attribution mais ce n'est pas la une bonne pratique.

D'autres instructions de branchements comme GOTO ou IF...THEN peuvent être utilisés à l'intérieur d'une boucle pour en sortir avant sa fin. Par contre vous ne pouvez pas sauter à l.intérieur d'une boucle; une boucle *doit* commencer par l'instruction FOR.

Le programme suivant illustre l'usage de boucles imbriquées. Il simule une montre digitale. Pour le rendre vraiment précis il se peut que vous ayez à régler la boucle de retard de la ligne 220. A remarquer: l'usage d'INKEY$ pour arrêter et faire partir la montre, aux lignes 90, 150, 170.

```
10 CLS0: PRINT @ 10, "MONTRE DIGITALE";
20 ' POSITIONS DE PRINT
```

FOR...NEXT...STEP

Le commandement FOR...NEXT se compose de deux instructions,

FOR *variable numérique* = expression TO expression
STEP expression

et

NEXT *variable numérique*

Les instructions FOR...NEXT sont liées et contrôlent le nombre de fois qu'une partie du programme sera exécutée. Cette opération s'appelle un bouclage (looping).

Les *expressions* sont évaluées et la boucle compte à partir de la valeur de la première expression jusqu'à (TO) la valeur de la seconde expression, avec un incrément (pas) donné par la troisième expression. La valeur en cours du compteur est logée dans la *variable numérique*. La boucle sera toujours exécutée au moins une fois, même si le domaine et le pas exprimés sont incompatibles. Si la partie STEP de l'instruction est omise, la valeur + 1 sera adoptée.

Des boucles peuvent être imbriquées les unes dans les autres, mais doivent l'être en bon ordre.

Vous pouvez bifurquer hors d'une boucle FOR...NEXT avant sa fin, par GOTO, IF...THEN ou d'autres déclarations similaires. Toutefois vous ne pouvez jamais faire aboutir un branchement à l'intérieur d'une boucle.

```
10  CLS: CLEAR: DIM L(1000)
20  INPUT "ENTRER UN NOMBRE"; N:IF N > 1000 OR N < 2 THEN 20
30  CLS: PRINT "NOMBRES PREMIERS ENTRE 2 ET"; N
40  FOR I = 2 TO N: IF L (I) <0 THEN 80
50 PRINT I;
60 IF I>SQR(N) THEN 80
70  FOR J = I TO N STEP I; L(J) =-1: NEXT J
80  NEXT I
```

```
30 P = 261: Q = 196
40 ' GROUPE DES INSTRUCTIONS
50 PRINT @ 386, "FRAPPER LA BARRE D'ESPACEMENT POUR
   ARRET & DEPART";
60 PRINT @ 424, "ET 'R' POUR REPARTIR";
70 'AFFICHAGE ET ATTENTE POUR DEPART
80 PRINT @ Q, "HEURES";: PRINT @ Q + 10, "MINS";: PRINT @
   Q + 22, "SECS";
90 A$ = INKEY$: IF A$ <> "∇" THEN 90
100 ' DEPART BOUCLE HEURES MINS & SECS
110 FOR ∇ H = 0 TO 23: FOR M = 0 TO 59: FOR S = 0 TO 59
120 SOUND 220,1
130 'BOUCLE DES IOEMES DE SEC
140 FOR T = 0 TO 9
150 A$ = INKEY$: IF A$ <> "∇" THEN 200
160 ' VERIFICATION SI UNE TOUCHE APPUYEE
170 A$ = INKEY$: IF A$ = "R" THEN 110
180 IF A$ <> "∇" THEN 170
190 ' AFFICHAGE DE L'HEURE
200 PRINT @ P,H;: PRINT @ P + 10,M;: PRINT @ P + 19,S;
   ".";T;
210 ' AJUSTAGE DE L'HEURE
220 FOR D = 1 TO 13: NEXT D
230 NEXT T
240 ' FIN DE LA BOUCLE DES IOEMES
250 NEXT S,M,H
260 ' FIN DES BOUCLES SECS, MINS & HEURES
270 PRINT @ 448, "ARRETE"
280 END
```

LES SOUS PROGRAMMES

A ce stade, vous avez déjà pris conscience de ce que les programmes informatiques ont une structure générale et qu'ils s'assemblent à partir de blocs plus petits. Certains de ces blocs peuvent être nécessaires plusieurs fois et à différents endroits du programme. La structure d'un programme peut souvent être simplifié en traitant ces situations en *sous-routines.* (en Anglais: *sub*routine). Une sous-routine,comme son nom l'indique, est une partie auxiliaire d'un programme, ou si l'on veut: un programme dans le programme. L'avantage principal d'une *souśroutine* est de pouvoir être *appelée* et d'effectuer à ce moment la séquence de lignes qu'elle contient, pour revenir ensuite à l'endroit d'où elle a été appelée. Pour appeler une sous-routine, vous employer la déclaration,

GOSUB, *numéro de ligne*

le *numéro de ligne* étant le numéro de la ligne du programme où la sous-routine débute. Comme la sous-routine doit retourner à l'endroit d'où elle a été appellée, elle doit se terminer par l'instruction

RETURN

Le GOSUB se comporte d'une façon analogue au GOTO. La différence est que avec GOTO le programme se branche sur une ligne spécifiée et continue à partir de là; il ne revient pas à moins qu'il ne rencontre un autre GOTO.
Tapez l'exemple suivant et faites le tourner.

```
 10 CLS: PRINT "DANS LE PROGRAMME PRINCIPAL"
 20 GOSUB 50
 30 PRINT "DE RETOUR DANS LE PROGRAMME PRINCIPAL"
 40 END
 50 PRINT "DANS LA PREMIERE SOUS-ROUTINE"
 60 GOSUB 90
 70 PRINT "A NOUVEAU DANS LA PREMIERE SOUS-ROUTINE"
 80 RETURN
 90 PRINT "DANS LA DEUXIEME SOUS-ROUTINE"
100 RETURN
```

Le programme se déroule en traitant les lignes dans l'ordre suivant: 10, 20, 50, 60, 90, 100, 70, 80, 30, 40.

Vous remarquerez comment la première sous-routine (lignes 50-80) appelle la seconde sous-routine (lignes 90, 100). L'instruction END (fin) que nous avons glissée dans le programme en marque effectivement la fin. Elle s'emploie pour indiquer à quel endroit le programme se termine vraiment. Retirez la ligne 40 et faites tourner cet exemple à nouveau. Vous allez obtenir ?RG ERROR IN 80. Parce que le programme a rencontré un RETURN sans avoir reçu ordre de se rendre à une sous-routine. Le programme est alors tombé à travers le fond de la première sous-routine. Ceci nous enseigne qu'il faut toujours protéger les sous-routines en s'assurant que la seule façon d'y entrer soit un GOSUB et que la seule façon d'en sortir soit un RETURN. Vous pouvez, tant qu'il vous plaît, utiliser GOTO, IF...THEN et d'autres branchements analogues, à l'intérieur d'une sous-routine mais elles ne doivent pas causer de branchement vers un numéro de ligne qui soit à l'extérieur de la routine.

Comme pour ON...GOTO, on peut effectuer un branchement multiple avec les sous-routines et dans une forme similaire.

ON *expression* GOSUB *liste de numéros de ligne*

Les programmeurs très expérimentés tiennent généralement un fichier de sous-routines; cela leur permet de construire de nouveaux programmes à partir de "pièces détachées" disponibles, pour ainsi dire. C'est généralement une bonne idée que de donner aux sous-routine

des numéros de lignes élevés, 10000-, afin qu'elles puissent s'intégrer dans tout programme sans devoir être renumérotées.

Nous vous présenterons dans les prochains chapitres de nombreux exemples de sous-routines. C'est pourquoi nous ne vous en donnons pas ici.

GOSUB
RETURN
ON...GOSUB

Le commandement GOSUB transfère le contrôle du programme au bébut d'une sous-routine du programme. Le RETURN rend le contrôle du programme à la ligne qui suit immédiatement l'instruction GOSUB.

GOSUB est suivi par un numéro de ligne qui est celui de la première ligne de la sous-routine.

GOSUB 1600

Une sous-routine doit contenir au moins une instruction RETURN.

Le commandement ON...GOSUB permet un branchement multiple sur plusieurs sous-routines de la même façon que ON...GOTO.

ON *expression* GOSUB *liste de numéros de lignes*

Si l'*expression* est négative le programme s'arrêtera avec un message d'erreur. Si l'*expression* est zéro ou plus grande que le nombre d'éléments dans la liste des numéros de ligne, la déclaration sera ignorée et le programme continuera à la ligne suivante.

```
 10 CLS: INPUT "ENTRER DEUX NOMBRES"; A,B
 20 INPUT "MAINTENANT ENTRER UN NOMBRE DE 1 A 4"; C
 30 ON C GOSUB 100,200,300,400
 50 PRINT C; "N'EST PAS ENTRER 1 ET 4": GOTO 20
100 PRINT "ADDITION."; A; "PLUS"; B; "IS"; A + B
110 RETURN
200 PRINT "SOUSTRACTION." A; "MOINS"; B; "IS"; A - B
210 RETURN
300 PRINT "MULTIPLICATION."; A; "FOIS"; B; "IS"; A * B
310 RETURN
400 PRINT "DIVISION"; A; "DIVISE PAR"; B; IS"; A/B
410 RETURN
```

CHAPITRE SIX

NOUVELLES DIMENSIONS

Au chapitre 2, lorsque nous avons discuté les types de variables possibles, nous avons dit que les variables se présentaient en deux formats les *simples* et les *tableaux*. Jusqu'à présent nous n'avons utilisé que les variables *simples*.

Si vous souhaitez que votre ordinateur tienne un fichier de vos livres, ou de vos disques, vous ne tarderiez pas à être à cours de variables pour désigner chaque article. Sans compter que ce serait un programme très difficile à écrire pour retrouver chaque nom... sous un autre nom! C'est là qu'intervient la grande utilité des *tableaux* (en Anglais: arrays)

LISTES ET TABLES

Les variables en tableau sont tout particulièrement utiles pour traiter des listes d'éléments. Nous pourrions par exemple dreser une liste de livres de la façon suivante:

1. Titre 1
2. Titre 2
3. Titre 3

.
.
.

Maintenant pour retrouver les titres des livres, nous pourrions demander le numéro 8 de la liste. Dans notre programme nous allons donner un nom à ce tableau de variables (le tableau qui contient les titres), puis en se référant à ce nom, suivi de son numéro il sera facile de le retrouver dans la liste. Donc d'abord il faut donner un nom au tableau.

Le nom d'une variable en *tableau* suit les mêmes règles que les variables *simples*. L'ordinateur reconnait la différence entre les unes et les autres parce que les variables en tableau contiennent *toujours* dans leur désignation un numéro entre parenthèses qui suit leur nom. Ce numéro, c'est l'indice qui permet de retrouver cette donnée précise dans le tableau. (Pensez au livre N° 8 de tout à l'heure).

A(5) se réfère au 5e article du tableau numérique du nom de A.
D7$(28) se réfère au 28e article du tableau de chaînes du nom de D7$.

Le numéro entre parenthèses qui suit le nom de la variable s'appelle un *indice*. Les variables en tableau sont dites variables *indicées*. Pour établir un tableau vous devez dire à l'ordinateur comment vous voulez nomer votre tableau et quelle sera sa taille. Ceci se fait avec l'instruction DIM.

DIM *nom du tableau* (nombre), *nom du tableau* (nombre, nombre)

DIM est l'abréviation de dimension: l'instruction DIM donne un nom au tableau et établi la taille *maximale* qu'il peut prendre. Le *nombre* peut

être tout nombre positif, ou une variable simple a condition que cette variable simple ait déjà reçu une valeur. N'assigner aux tableaux que la taille indispensable car les très grands tableaux consomment beaucoup de la place disponible dans la mèmoire.

```
10 DIM A(22), NA$(40)
```

La ligne ci-dessus demande à l'ordinateur d'établir un tableau nommé A, de 22 nombres et un tableau de chaînes appelé NA$ de dimension 40. (En fait, les tailles sont 23 et 41, parce que les indexs commencent à 0).

Pour vous référer à un élément contenu dans un tableau, dans une expression, il vous suffit de mentionner le numéro d'index entre parenthèses, après le nom.

```
25 A(4) = 7.0
32 A(M) = B*C/A
```

La ligne 25 donnera la valeur 7 à l'élément 4 du tableau A. La ligne 32 va évaluer l'expression et mettre le résultat dans l'élément M du tableau A. (Vous remarquerez que le A du côté droit de la ligne 32 est une variable *simple* appelée A et qu'elle n'a rien à voir avec le tableau A figurant du côté gauche).

Les tableaux peuvent aussi avoir deux dimensions. La ligne,

```
10 DIM T(10,5), TB$(12,4)
```

va créer un tableau numérique, T, qui sera un tableau de 10 lignes et 5 colonnes. Le tableau de chaînes TB$, aura 12 lignes et 4 colonnes. Par exemple, un enseignant peut vouloir tenir les résultats d'une interrogation de 25 élèves, sur 6 sujets différents. Un tableau nommé INTERO (25,6) lui servirait à cet usage. Pour accéder à l'un des éléments du tableau il faut deux indexations.

```
25 PRINT INTERO (10,3)
```

indiquerait à l'affichage les notes du 10e éléve sur le 3e sujet. Bien sûr les indexations doivent correspondra à des éléments qui existent. Si l'on essye d'obtenir INTERO (30,7) il y aura message d'erreur, puisque l'on sort des limites du tableau.

Le moment est venu de donner un exemple utilisant des tableaux. Le programme qui suit est court mais il va demander un effort de reflexion si vous voulez comprendre comment il marche. Il s'agit de mélanger un paquet de 52 cartes. Chaque carte porte un numéro de 1 à 52, dans le tableau X. Les éléments sont tirés au hasard du tableau X et placés dans le tableau Y. Quand le programme se termine, Y contient des nombres de 1 à 52, mais dans le désordre (c'est à dire "mélangés".) Il n'est pas possible d'utiliser directement le générateur de nombres aléatoires RND car il serait succeptible de tirer la même carte plusieurs fois. La ligne 50 imprime le tableau correspondant aux cartes mélangées. Vous remarquerez - c'est important - qu'il utilise une expression comme indexation du tableau.

DIM

Le commandement DIM s'utilise pour dimensioner des tableaux. Les tableaux peuvent avoir une ou deux dimensions et peuvent être numériques ou alphanumériques. Les noms des tableaux suivent les mêmes régles que les noms des variables simples.

DIM *nom du tableau* (n), *nom du tableau* (n,n)

Si la taille maximale du tableau n'excéde pas 1Ø, la déclaration DIM est superflue.

Pour se référer à un élément de tableau dans une expression, le nom doit être suivi de l'*indice* entre parenthéses.

A(14,K) N9(B), L$(14,4)

```
10 DIM X(52), Y(52): CLS
20 FOR I = 1 TO 52: X(I) = I: NEXT I
30 FOR I = 52 TO 1 STEP - 1
40 J = RND(I): Y(I) = X(J) = X(J): X(J) = X(I): NEXT I
50 FOR I = 1 TO 13: FOR J = 1 TO 4: PRINT Y (4*(I-1)+J);: NEXT
   J,I: END
```

Les éléments d'un tableau peuvent être traités et déplacés de la même façon. L'une des applications les plus répandues de cette technique est la mise en ordre alphabétique. Dans l'exemple suivant, une liste de mots est triée par la sous-routine dont le début est à la ligne 200. De nombreux livres ont été écrits sur les triages par l'informatique. La méthode que nous employons ici s'appelle le triage par échange. Ce n'est peut-être pas la meilleure, mais elle a le mérite d'être la plus simple. Si elle ne vous est pas familière, essayez à la main de trier les lettres D, B, A, E, C.

```
 10 CLS: DIM W$(50)
 20 INPUT "COMBIEN DE MOTS"; N
 30 CLS: PRINT "ORIGINAL"
 40 FOR I = 1 TO N: PRINT I; ".∇ ∇";
 50 INPUT W$(I): NEXT I
 60 GOSUB 200
 70 PRINT @ 18, "TRIE"
 80 FOR I = 1 TO N
 90 PRINT @ 18 + 32*I, I; ".∇ ∇"; W$(I)
100 NEXT I:END
200 M = N
210 F = 0: FOR I = 1 TO M - 1
220 IF W$(I) <= W$(I + 1) THEN 240
230 T$ = W$(I): W$(I) = W$(I + 1): W$(I + 1) = T$: F = 1
240 NEXT I: IF F = 1 THEN M = M - 1: GOTO 210
250 RETURN
```

QUELLE EST SA FONCTION?

Vous souvenez-vous de RND? Nous appellons RND une *fonction*. Ce n'est pas la seule fonction, il y en a d'autres... Au sens de l'informatique, une fonction est un sous-programme spécial qui est capable de *restituer* une valeur unique lorsqu'une série d'*arguments* lui sont communiqués. En language BASIC, une *fonction* prend la forme suivante,

Nom de la fonction *(arguments)*

et peut s'utiliser dans une expression de la même manière que les autres opérateurs arithmétiques (↑, *, /, +, -). Toutefois, les fonctions ont priorité de traitement sur tous les autres opérateurs, sauf les parenthèses.

Les *arguments* d'une fonction sont des valeurs que l'on donne à la fonction, afin qu'elle donne en retour le résultat. Les arguments peuvent être des constantes, des variables ou des expressions.

RND(1Ø), RND(X) ou RND(A★2+F)

sont tous des arguments valables. Vous remarquerez que l'argument est *toujours* inclus entre parenthèses.

Votre ordinateur vous fourni un certain nombre de fonctions, comme RND, qui font partie du language BASIC. Les fonctions intégrées à votre ordinateur peuvent se répartir en cinq classes différentes. Nous allons prendre chaque classe séparément, vous présenter la liste de ses composants ainsi qu'une courte explication et un exemle pour chaque fonction. A partir de maintenant vous verrez ces fonctions apparaître dans nos programmes, c'est pourquoi nous ne vous présenterons pas un exemple de programme pour chacune, et d'ailleurs il y en a beaucoup trop pour que ce soit possible.

Fonctions de classe I

Ce sont des fonctions numériques, surtout utilisées en mathématiques. Elles ont un argument *numérique* et répondent par une valeur *numérique*. Les fonctions de la classe I ne peuvent être utilisées que dans des expressions *numériques*. Pour ceux d'entre vous qui ne seraient pas très familiers avec la trigométrie, vous pourrez vous reporter à l'appendice D de cet ouvrage.

Nom de la fonction	Operation	Exemple
ABS (X)	Valeur absolue de X	1ØØ A = ABS (D★2 - C)
ATN (X)	Arctangeante X, en *radians*. L'inverse de TAN (X)	11Ø PRINT "ANGLE =";
COS (X)	Cosinus X, X étant un angle mesuré en *radians*.	51Ø F7 = COS (X + 4)
EXP (X)	Elévation de la base e des logarithmes naturels à la puissance x, (e^x). L'inverse de LOG (X)	215 Q = EXP (-A★A)
FIX (X)	Donne la partie entière de X, (c'est à dire élimine tous les chiffres après la virgule)	172 N = FIX (Z★.Ø5)
INT (X)	Elimine les décimales de X, si X est positif, comme FIX. Si X est négatif, arrondira vers le bas. (C'est à dire INT (-12.ØØ1) fera -13.	28Ø P = INT (1ØØ★X)

Nom de la fonction	Operation	Exemple
JOYSTK (X)	Indique la position actuelle, des manettes de jeux, comme suit: X = Ø, position horizontale de la manette gauche X = 1, position verticale de la manette gauche X = 2, position horizontale de la manette droite X = 3, position verticale de la manette droite	1ØØ A = JOYSTK (Ø): B = JOYSTK (1)
LOG (X)	Logarithme naturel de X. La valeur de l'argument doit être supérieure à zéro. L'inverse de EXP(X)	617 L1 = 5.2★LOG(W4)
PEEK(X)	Donne le contenu de l'emplacement de la mémoire dont l'adresse est X	55 P = PEEK (65280)
POINT (X, Y)	Vérifie si un point graphique en basse résolution est "allumé" ou "éteint". X doit être compris entre Ø et 63 (horizontal) et Y entre Øet 31 (vertical). Restitue la valeur Ø si le point est éteint, -1 si en mode texte; 1 à 8, si en service, selon le N° de couleur.	5Ø IF POINTS (5,A) = C THEN 21Ø
POS(X)	Donne la position actuelle du PRINT. Les seuls arguments possibles sont, Ø pour l'affichage à l'écran, -2 pour l'imprimante	168 IF POS (Ø) > 3Ø THEN PRINT A$
PPOINT (X,Y)	Vérifie si un point en haute résolution graphique est allumé ou éteint. Restitue Ø si éteint. Autrement, restitue le numéro de code couleur de ce point.	115 C = PPOINT (A1, A2)

Nom de la fonction	Operation	Exemple
	(X de Ø à 255, Y de Ø - 191)	
RND(X)	Donne un nombre entier au hasard entre 1 et X. X = Ø restitue un nombre au hasard entre Ø et 1.	22Ø PRINT @ RND (51Ø); "★";
SGN (X)	Donne le signe de l'argument. X négatif restitue - 1. X = Ø restitue Ø X = positif restitue + 1	412 Y = RND (ABS (N))★SGN (N)
SIN (X)	Donne le sinus de X, X étant un angle en *radian.*	2Ø5 S = SIN (K★PI/18Ø)
SQR (X)	Racine carrée de l'argument ($\sqrt{x}$), X ne doit pas être négatif. Si X négatif, la fonction restitue $--\sqrt{ABS(x)}$	33Ø C = SQR (A★A + B★B)
TAN (X)	Tangeante de X. X étant un angle en radian. L'inverse de ATN (X)	84Ø R5 = B/TAN (EQ-5)

Fonctions de la classe II

Les fonctions de la classe II ont un argument *numérique* mais restituent une valeur de *chaîne.* Elles ne peuvent s'utiliser que dans des expressions de *chaîne.*

Nom de la fonction	Operation	Exemple
CHR$ (X)	Répond par le caractère dont le N° de code est représenté par X. X doit être compris entre Ø et 255. Voir Appendice A pour liste des N° de code.	2Ø M$ = CHR$ (143) + CHR$ (128)

Nom de la fonction	Operation	Exemple
HEX$ (X)	Calcule la valeur en hexadécimal d'un nombre décimal X.	42 PRINT HEX$ (3Ø)
STR$ (X)	Convertit une expression numérique en chaîne alpha-numérique équivalente	175 A$ = STR$ (12.49)

Fonctions de la classe III

Les fonctions de la classe III sont des fonctions de *chaîne.* Leurs arguments (il y en a généralement au moins deux) sont composés d'une chaîne et d'un nombre. Toutes répondent par une valeur de *chaîne* et doivent donc faire partie d'une expression de chaîne.

Nom de la fonction	Operation	Exemple
LEFT$ (X$, N)	Donne les N *premiers* caractères de la chaîne X$	
MID$ (X$,M,N)	Donne N caractères de la chaîne X$, en commençant à la position M. Si N ne figure pas, toute la chaîne à droite de M est restituée. M doit être plus grand que Ø	76Ø K$ = MID$ (W$,I,4)
RIGHT$ (X$,N)	Restitue les N *derniers* caractères de la chaîne X$	34Ø T$ = RIGHT$ (Q$,B+7)
STRING$ (N,C)	Restitue une chaîne de longueur N, constituée par le caractère défini par C. L'argument C peut être soit un nombre représentant le code ASC II du caractère, soit le caractère lui-même entre guillements.	4ØØ A$ = STRING$ (5,67) 41Ø PRINT STRING$ (32,"★")

Fonctions de la classe IV

Ce sont des fonctions mixtes, similaires à celles de la classe II. Elles ont un argument de *chaîne* et restituent une valeur *numérique* et n'apparaissent donc que dans les expressions numériques.

Nom de la fonction	Operation	Exemple
ASC (X$)	Donne le N° de code ASCII du *premier* caractère de la chaîne figurant dans l'argument.	715 P = ASC (F$) -64
INSTR (P, S$, T$)	Scrute la chaîne S$ à la recherche de la chaîne T$, en partant de la position P dans la chaîne examinée. Répond Ø si T$ n'est pas trouvé; autrement, indique la position de T$ dans S$.	212 F = INSTR(N,X$,"AB")
LEN (X$)	Donne la longueur (en nombre de caractères) de la chaîne X$. Même les espaces sont comptés. Si la chaîne est vide, la réponse est Ø	845 N = LEN(N$)
VAL (X$)	Convertit la représentation alphanumérique de chiffres en représentation numérique. C'est la fonction inverse de STR$ déjà vu. Si la chaîne commence par une lettre, la réponse est Ø	92 Z = VAL (AB$)

Fonctions de la classe V

Les fonctions de la classe V sont des fonctions de système. Elle ne comportent pas d'argument.

Nom de la fonction	Operation	Exemple
INKEY$	Vérifie ce qui se passe au clavier et indique la touche qui a été frappée, s'il y en a une. Restitue une chaîne, doit donc être utilisé dans une expression de chaîne.	146 P3$ = INKEY$

Nom de la fonction	Operation	Exemple
MEM	Trouve la quantité de mémoire encore libre dans l'ordinateur.	PRINT MEM
TIMER	Donne le contenu du "Timer" (Timer = horloge temps réel), une valeur comprise entre Ø et 65535. Pour remettre le TIMER à zéro, faire: TIMER = Ø	62 T1 = TIMER - T 63 TIMER = Ø

Les fonctions à créer soi même.

En plus des fonctions fournies par le système, il vous est possible de créer jusqu'à 26 fonctions numérique de votre crû. La forme de l'instruction est:
DEF FN lettre *(variable factice)* = *formule*
La *lettre* est n'importe quelle lettre et A à Z. La *variable factice* est une lettre qui sera remplacée par l'argument de fonction lorsque la fonction sera appellée. La *formule* est une expression BASIC composée de la *variable factice* elle-même et/ou d'autres variables. D'autres fonctions, soit crées par l'utilisateur ou figurant dans le système peuvent également faire partie de l'expression. Une fonction *ne peut pas* être appelée par elle même.

L'équation $y = ((x-3)^2 + (x-4)^4)/x^3$, se traduira directement en une fonction définie par l'utilisateur, de la manière suivante:

```
25 DEF FNY (X) = ((X - 3) ↑ 2 + (X -4) ↑ 4)/X↑3
```

X est ici une *variable factice*, pas un nom de variable. Quand la fonction est appelée, plus loin, quelque part dans le programme, elle sera remplacée par l'argument. Pour utiliser la fonction, il vous suffit de l'inclure dans une expression, de la même manière qu'une fonction fournie par le système.

```
150 Y(I) = FNY (X) + FNY (W)
```

Les fonctions peuvent également s'utiliser pour fournir des 'routines de service' (des opérations qui reviennent souvent). Vous avez probablement remarqué que les nombres affichés sur votre écran on une tendance à être un peu désordonnés. L'ordinateur essaie de se rendre utile en vous envoyant un nombre avec le maximum de décimales, donc le maximum de précision. Quelques fois, cette précision n'est pas nécessaire, quelques fois elle est franchement inopportune. Dans le cas de montants d'argent par exemple, un nombre excessif de décimales est plutôt une plaie.

La fonction suivante peut s'employer pour obtenir l'affichage de chiffres, avec le nombre de décimales souhaité (D).

10 DEF FND (X) = INT (X * 10 ↑ D + 0.5)/10 ↑ D

Vous remarquerez que X est une variable factice, D n'est *pas* une variable factice, la valeur de D doit être fournie depuis l'extérieur de la fonction, par un INPUT, par exemple. Pour utiliser la fonction,

205 DEF FND (A) etc

Comme toutes les fonctions trigonométriques utilisent des arguments en radians, une fonction pour convertir les degrés en radians peut-être utile. Cette conversion sera effectuée par:

10 DEF FNR (X) = X/57.295779

Un résultat plus précis peut être obtenu en utilisant plutôt:

10 DEF FNR (X) = X ★ ATN (1.0)/45

La constante π peut être créée par:

20 DEF FNP (X) = 4.0 ★ ATN (1.0)

A noter dans ce cas: la variable factice n'a aucun role, elle est seulement là parce qu'il faut mettre un argument.

Comme vous ne pouvez évidemment pas appeler une fonction qui n'a pas encore été définie, il est recommandé de mettre toutes vos définitions de fonctions au début d'un programme.

ALTERNATIVES A INPUT

La seule façon que nous avons utilisée jusqu'à présent pour donner des valeurs aux variables est l'instruction INPUT. Elle est très pratique, mais vous avez peut être constaté que INPUT n'acceptait pas certain caractères. Si vous débutez une chaîne par des espaces, ceux-ci sont perdus, si vous tapez une virgule, tout ce qui vient après est perdu. Il existe une alternative pour parer à ces problèmes, c'est LINE INPUT.

LINE INPUT *"texte"*; variable de chaîne

(en Anglais, line = ligne). Le LINE INPUT se comporte de la même manière que l'INPUT, sauf qu'il accepte tout, y compris les espaes et les virgules. Le *texte* est comme pour INPUT et la *variable de chaîne* peut être toute variable de chaîne. Il ne peut y avoir qu'une variable pour chaque instruction LINE INPUT.

25 LINE INPUT "TAPER UNE LIGNE DE TEXTE"; L$

Souvent dans un programme, il est nécessaire de dterminer un certain nombre de constantes avant que le programme puisse vraiment se mettre à travailler. Vous pourriez, bien entendu, entrer ces données chaque fois que

DEF FN

La commande DEF FN s'utilise pour définir une fonction numérique créée par l'utilisateur.

DEF FN *nom (variable factice)* = formule

Le nom peut être toute lettre de A à Z

La *variable factice* peut être n'importe quelle lettre; elle sera remplacée par l'argument dès que la fonction sera utilisée. Une seule *variable factice* peut être utilisée.

La *formule* décrit l'opération à effectuer en termes utilisant la variable factice et/ou d'autres variables.

Les fonctions définies par l'utilisateur doivent être contenues dans une ligne de programmation. Une fonction définie par l'utilisateur peut utiliser d'autres fonctions, (fonctions définies ou fonctions système) dans sa formule, mais une fonction ne peut pas faire appel à elle même.

Une fonction doit être définie avant d'être utilisée, elle devrait donc figurer en début de programme.

D'autres fonctions mathématiques peuvent être créées par l'utilisateur, en voici quelques unes.

```
10 DEF FNS (X) = 1/COS(X): 'SECANTE
20 DEF FNI (X) = - ATN (X/SQR( - X*X+1))+1.5708
30 DEF FNH (X) = - EXP(X)/(EXP(X)+EXP(-X))*2+1
40 DEF FNM (A) = INT((A/B-INT(A/B))*B+0.5)*SNG(A/B)
50 B = 8: PRINT FNM (13)
```

LINE INPUT

Le commandement LINE INPUT, fait entrer une ligne entière dans une variable de chaîne, y compris les virgules et les espaces de début, qui ne sont pas acceptés par INPUT.

LINE INPUT *"texte"; variable de chaîne*

Le texte est constitué par tout message inclus entre guillemets. Il est facultatif et s'il figure, il doit être séparé de la *variable de chaîne* par un point-virgule (;). La *variable de chaîne* peut être toute variable de chaîne. Il ne peut apparaître qu'une seule variable dans l'instruction LINE INPUT. La longueur maximale que peut atteindre un LINE INPUT est de 255 caractères.

```
10 CLEAR 500: CLS
20 LINE INPUT "ENTREZ VOTRE NOM EN ENTIER"; N$
30 LINE INPUT "ET VOTRE ADRESSE"; A$
```

vous faites tourner le programme, en utilisant INPUT. Il y a toutefois une méthode beaucoup plus pratique qui consiste à utiliser les instructions READ et DATA. Celle-ci s'emploient toujours ensemble et prennent la forme:

READ, *liste de variables*
DATA, *liste de valeurs*

(en Anglais, read, = lire et data = données)
L'instruction READ se comporte comme l'instruction INPUT à la différence qu'au lieu d'arrêter le programme et d'attendre que l'utilisateur entre une donnée, il va chercher la donnée dans l'instruction DATA qui fait partie du programe. L'instruction DATA peut figurer n'importe où dans le programme. S'il y en a plus d'une, READ commence par celle qui a le numéro de ligne le plus petit et continue dans l'ordre numérique croissant.

```
10 DATA 1, 2, 3, 4, 5
20 FOR I = 1 TO 3
30 READ A: PRINT A: NEXT I
40 READ D,G: PRINT D,G
```

L'exemple ci-dessus va lire le premier élément dans la liste des DATA (1), l'imprimer, lire le second (2), l'imprimer, etc... Après la lecture de chaque élément, le curseur se déplace à l'élément suivant. La ligne 40 va lire les deux derniers éléments. Si maintenant vous ajoutez la ligne,

```
50 READ X: PRINT X
```

et faites à nouveau tourner le programme, vous obtiendrez ?OD ERROR IN 50. (out of data, erreur à 50). Cela veut dire que READ n'a plus d'éléments dans DATA parce qu'il les a tous lus. Ajoutez maintenant une autre ligne,

```
45 RESTORE
```

et faites à nouveau tourner le programme. Cette fois cela marche, X a la valeur 1. L'instruction RESTORE a remis le curseur au début de DATA. (To restore, remettre en place). Les chaînes peuvent aussi être traitées par des instructions DATA, mais si il y a une succession mixte de variables dans READ, il est indispensable qu'elle corresponde chaque fois a une valeur de même nature dans la succession des DATA.

Si vous avez écrit un programme comportant beaucoup de chaînes, vous avez peut être déjà obtenu le message d'erreur ?OM ERROR. (OM out of memory). Cela veut dire que la partie de la mémoire attribuée au stockage des chaînes était insuffisant. Pour attribuer plus de mémoire aux chaînes, employez l'instruction CLEAR.

```
10 CLEAR 1000
```

Cette instruction réservera 1000 octets de mémoire pour le stockage des chaînes. Comme CLEAR remet *toutes* les variables à zéro, employer ceci uniquement au tout début d'un programme.

READ
DATA
RESTORE

Le commandement READ lit l'élément suivant dans une ligne de DATA et l'attribue à une variable spécifiée dans la liste.

READ *liste des noms de variables*

La ligne DATA stocke des données dans le programme et peut être une ligne numérotée, n'importe où dans le programme.

DATA *liste de valeurs*

Les variables numériques et les variables de chaînes peuvent toutes deux être utilisées dans READ et DATA, à condition que leur séquence soit identique. Une chaîne doit se trouver attribuée à une variable de chaîne et une valeur numérique a un variable numérique.

Le commandement RESTORE remet le curseur de lecture au premier élément de la ligne de DATA portant, dans le programme, le numéro le plus faible.

RESTORE

```
10 CLS: PRINT: PRINT: PRINT
20 READ A,B: IF A = - 9999 THEN RESTORE: GOTO 20
30 PRINT A; "∇ + ∇ 5 ∇ EST ∇";: INPUT C
40 IF C = B THEN PRINT "JUSTE" ELSE PRINT "FAUX"
50 FOR D = 1 TO 600: NEXT D: GOTO 10
60 DATA 8,13,12,17,5,10,27,32,14,19,3,8
70 DATA 7,12,6,11,1,6,-9999,-9999
```

CLEAR

Le commandement CLEAR efface toutes les variables et réserve de l'espace pour stocker les chaînes.

CLEAR 500

réservera 500 octets pour stocker des variables de chaîne.

La commande CLEAR peut également s'utiliser pour fixer la plus haute adresse de la mémoire autorisée en BASIC, afin de réserver la place située plus haut pour le language machine.

CLEAR 200, 14000

va réserver 200 octets pour le stockage des chaînes alphanumériques et établir la plus haute adresse admise pour le BASIC à 14000. Les routines en language machine peuvent maintenant être stockées à partir de 14001 et au dessus.

Si CLEAR n'est pas utilisé, 200 octets de place pour le stockage des chaînes sont automatiquement attribués.

PAUSE POUR FAIRE LE POINT

Le contenu de ce chapitre, constitue avec celui des chapitre 1, 2, 3 et 5 le cœur du language BASIC. Bien qu'il y ait encore des instructions de programmation à venir dans cet ouvrage, elles ne constituent, dans un certain sens, qu'un vernis de finition.

Tout ce que nous avons vu et assimilé jusqu'à présent va être utilisé fréquenment dans les chapitres suivants car c'est là l'ossature de n'importe quel programme. Bien que vous soyez très probablement avide de commencer a dessiner avec votre ordinateur, il serait peut-être souhaitable de passer quelque temps à ce stade, pour vous assurer que vous avez bien compris comment les choses se passent. Cela n'en rendra votre apprentissage des fonctions graphiques que plus facile.

Revoyez les exemples et essayez de les faire fonctionner avec vos propres idées. Faites des tentatives, des essais: vous le savez, aucune erreur de programmation ne détraquera DRAGON 32, s'il n'est pas d'accord, il vous le dira, c'est tout...

Nous terminons cette section avec deux exemples. Le premier est une extension du programme de mélange de cartes du chapitre précédent. Le programme distribue une 'main' de jeu de cartes. Remarquez les points suivants.

a) le mélange des cartes apparaît maintenant dans uns sous-routine, à la ligne 90
b) l'usage de READ et de DATA pour charger les tableaux DIM, au début
c) les lignes 130 et 140 pour trouver une couleur de cartes et quelle carte dans la couleur

```
 10 DIM X(52), PAQUET(52), CARTE$(13), COULEUR(4)
 20 FOR I = 0 TO 3: READ COULEUR(I): NEXT I
 30 DATA PIQUE, CARREAU, TREFLE, CŒUR
 40 FOR I = 1 TO 13: READ CARTE$(I): NEXT I
 50 DATA AS, DEUX, TROIS, QUATRE, CINQ, SIX, SEPT
 60 DATA HUIT, NEUF, DIX, VALET, DAME, ROI
 70 CLS: INPUT "COMBIEN DE CARTES FAUT-IL DONNER?"; N
 80 GOSUB 190
 90 ST = 1
100 EN = ST+N-1: IF EN >52 THEN GOTO 80
110 CLS: PRINT @ 10, "VOTRE MAIN"; PRINT; PRINT
120 FOR I = ST TO EN
130 S = INT(PAQUET(1)-1)/13)
140 C = PAQUET(I) -S*13
150 PRINT TAB (8); CARTE$(C); "∇ DE ∇"; COULEUR(S)
160 NEXT I: ST = ST + N
170 PRINT @ 448, "UNE AUTRE DONNE, OUI OU NON ∇;: INPUT A$
180 IF A$ = "OUI" THEN 100 ELSE END
190 FOR 19 = 1 TO 52: X(19) = 19: NEXT 19
200 FOR 19 = 52 TO 1 STEP -1
210 J9 = RND(19): PAQUET(19) = X(J9): X(J9) = X(I9)
220 NEXT 19: RETURN
```

Le second exemple utilise presque toutes les fonctions de chaînes disponibles. Le programme scrute le texte qui lui est communiqué et communique le nombre de fois que chaque lettre y apparaît. Ce type de programme s'utilise fréquement pour déchiffrer les messages codés. Avec quelques modifications, il peut être utilisé pour rechercher un mot déterminé, ou une suite de caractères.

```
 10 CLEAR 1000: CLS: READ A$
 20 DATA ABCDEFGHIJKLMNOPQRSTUVWXYZ
 30 PRINT "TAPEZ UNE LIGNE DE TEXTE": PRINT
 40 LINE INPUT L$
 50 FOR I = A TO LEN (A$): CLS
 60 T$ = MID$(A$,I,1): C = 0: P = 1: P$ = L$
 70 F = INSTR(P,L$,T$)
 80 IF F>0THEN C = C+1 ELSE 140
 90 P$ = LEFT$(P$,F-1) + STRING$(LEN(T$),CHR$(128))
100 IF F>LEN(L$) TEHN 120
110 P$ = P$ + RIGHT$(L$,LEN(L$)-F)
120 P = F + LEN(T$)
130 IF P<= LEN(L$) - LEN(T$) + 1 THEN 70
140 PRINT P$
150 PRINT @ 354, "TROUVE";C;"APPARITIONS DE ";T$
160 PRINT @ 416, "APPUYEZ SUR LA BARRE D'ESPACEMENT POUR
    CONTINUER, N POUR ARRET"
170 Z$ = INKEY$: IF Z$ = ""THEN 170
180 IF Z$ = "N" THEN 200
190 NEXT 1
200 CLS: END
```

CHAPITRE SEPT

DES POINTS ET DES IMAGES

Lorsque votre ordinateur affiche quelque chose à l'écran, ce qui se passe, en fait, c'est qu'il allume ou non chaque point du tube cathodique et construit ainsi l'image que vous voyez. Si le point est allumé, il apparaît comme un point de couleur, si le point est éteint, il est noir. Toutes les lettres que nous avons affichées sont constituées par des points de lumière. La taille du point dépend de la *définition* dans laquelle l'ordinateur travaille. En *base définition* appelée aussi *basse résolution,* les points sont grands; en *haute définition,* ou *haute résolution,* les points sont petits. Dans le premier cas, l'écran est divisé en un nombre relativement peu élevé de points, dans le second ce nombre est beaucoup plus élevé.

Votre ordinateur vous offre la possibilité de travailler en cinq définitions différentes allant de 512 à 49.152 points sur l'écran. Ceci vous donne une souplesse très grande dans vos images. Nous allons commencer par créer des images dans la définition la plus basse, puis nous passerons à des niveaux plus élevés. Les méthodes utilisées pour afficher des dessins à l'écran et y créer du mouvement sont assez semblables, quelle que soit la définition choisie.

IMPRIMER DES IMAGES

Vous vous souvenez qu'au chapitre 3, lorsque nous vous avons présenté l'instruction PRINT @, nous vous avions décrit l'écran comme une grille de 16 x 32 éléments. Ceci nous permettait de placer un caractère donné n'importe ou sur l'écran, en lui assignant une position. En utilisant la fonction CHR$ (voir les fonctions au chapitre six), nous pouvons créer des caractères graphiques spéciaux. Le programme suivant va afficher tous les caractères que CHR$ met à votre disposition.

```
10 FOR I = 1 to 255: CLS0
20 PRINT @ 100, "CHR$(";I;")";
?0 PRINT @ 120, CHR$ (I);
40 FOR D = 1 TO 600: NEXT D, I: CLS
```

Les nombres de 1 à 31 sont affectés à des caractères de contrôle et ne produisent donc pas grand chose. De 32 à 127, apparaissent les caractères du clavier. Les codes de 128 à 255 sont des caractères graphiques spéciaux. (Une liste complète des caractères disponibles est données à l'appendice A). Ces caractères graphiques sont des modules de couleur qui peuvent être assemblés en formes simples. La forme la plus simple est un rectangle de couleur. Par exemple, CHR$ (143) donne un rectangle vert (qui est la couleur 1). Si vous ajoutez 16, CHR$ (159) donne un rectangle jaune - qui est la couleur 2 - etc... Il y a 16 modules, de CHR$(128) à CHR$(143) qui sont composés de vert et noir.

Pour obtenir le même module, mais en couleur différente, ajoutez le nombre voulu de fois 16 au code:

+ 16 jaune	+ 32 bleu	+ 48 rouge
+ 64 blanc	+ 80 cyan	+96 magenta
	+ 112 orange	

Le programme suivant montre ce qui se produit quand on augmente le numéro de code de 16 en 16. Vous pouvez aussi l'utiliser pour ajuster la teinte de la couleur de votre poste de TV, employez C = 143.

```
10 CLS0: INPUT "ENTRER UN CODE DE 128 A 143"; C
20 FOR I = 1 TO 14: FOR J = C TO 255 STEP 16
30 FOR K = 1 TO 4: PRINT CHR$(J);: NEXT K,J,I
40 GOTO 40
```

Comme vous le constatez sur l'exemple ci-dessus, les caractères obtenus par CHR$ peuvent s'imprimer directement sur l'écran. Mais comme ce sont des caractères, ils peuvent également être placés dans des variables de chaînes. Ceci est plus pratique car il est alors possible de les manipuler plus facilement

Nous allons maintenant commencer à construire un dessin et nous dessinerons un château; d'abord parce que c'est une forme assez simple et ensuite parce que la démonstration permettra de voir comment, à partir d'une forme simple on peut arriver à ajouter de plus en plus de détails. Entrez au clavier et faites tourner chaque partie du programme, au fur et à mesure que nous vous la communiquons, de façon à ce que vous puissiez voir les diverses étapes par lesquelles le dessin va passer.

D'abord nous allons construire un mur, en travers de l'écran, il aura 32 blocs de largeur et nous lui donnerons une hauteur de six blocs.

```
10 CLEAR 500: CLS0
20 FOR I = 1 TO 6: FOR J = 1 TO 32
30 MUR$ = MUR$ + CHR$ (207): NEXT J,I
40 PRINT @ 256, MUR$;
200 GOTO 200
```

La première ligne réserve de l'espace pour les chaînes de caractères que nous allons employer. Les lignes 20 et 30 'construisent' le mur avec des blocs de couleur blanche, CHR$ (207), le tout est stocké dans une chaîne appelée MUR$ qui se présente sur l'écran sous la forme d'un bloc de couleur, continu. La dernière ligne (200) ne sert qu'à faire rester le dessin sur l'écran.

Maintenant nous ajoutons les crénaux: ceci se fait avec des blocs alternés de blanc et de noir. Il ne faudra qu'une rangée, cette fois.

```
50 FOR I = 1 TO 16: B$ = B$ + CHR$(128) + CHR$(207): NEXT I
60 PRINT @ 224, B$;
```

La ligne 5Ø construit les crénaux et la ligne 6Ø les imprime en dessus du mur.

Maintenant il nous faut une tour. La tour est construite avec les mêmes matériaux que le mur, donc nous allons prendre quelques briques de MUR$

```
7Ø P = 1Ø
8Ø FOR I = 1 TO 3: PRINT @ 128 + 32*I+P, LEFT$ (MUR$,1Ø);: NEXT I
```

La ligne 8Ø prend dix briques dans MUR$ et construit 3 rangées au milieu du mur. Si vous voulez essayer de mettre la tour à un autre endroit, changez P. La tour fait 1Ø blocs de large donc P peut avoir n'importe quel valeur entre Ø et 22.

Maintenant nous mettons des crénaux sur la tour.

```
9Ø PRINT @ 128 + P, LEFT$ (B$,1Ø);
```

Ceci termine notre château de base. Nous pouvons lui ajouter des meurtrières en utilisant un caractère différent venant surimprimer ceux qui figurent à l'écran.

```
1ØØ FOR I = 2 TO 32 STEP 4: PRINT @ 288 + I, CHR$(2Ø6);: NEXT I
```

Nous avons également besoin d'une porte et pour ce faire, nous allons imprimer des modules noirs, aux endroits appropriés.

```
11Ø G$ = CHR$ (128) + CHR$ (128) + CHR$ (128)
12Ø FOR I = 353 TO 417 STEP 32: PRINT @ I+P+5, G$;: NEXT I
```

La porte est stockée sous G$ et imprimée à la ligne 12Ø (utilisant le P dans PRINT @ afin que la porte reste dans l'axe de la tour).

Mais un château dont la porte est toujours ouverte n'est pas sûr! Une herse défendant l'entrée serait bien utile, et l'usage d'un autre caractère, (142+112) va nous permettre de réaliser a peu près ce qu'il faut.

```
13Ø P$ = CHR$ (254) + CHR$ (254) + CHR$ (254)
14Ø FOR I = 353 TO 417 STEP 32: PRINT @ I+P+5, P$;
15Ø FOR K = 1 TO 3ØØ: NEXT K,I
```

La ligne 13Ø construit une herse orange (pourquoi pas?) qui est engendrée du haut vers le bas par la ligne 14Ø. La boucle d'attente K fait "descendre" la herse lentement.

Nous allons maintenant quitter le château mais vous pouvez encore lui rajouter tout ce qu'il vous plaira: pourquoi pas un fossé plein d'eau bleue, un drapeau, en haut de la tour et ainsi de suite.

ANIMATION DES IMAGES

Dans l'exemple du château, les lignes 140 et 150 nous montrent comment on peut créer du mouvement dans un dessin, en imprimant des morceaux successivement. Ce type de mouvement est presque entièrement limité à des ouvertures et fermetures de portes... Une meilleure méthode consiste à imprimer une image entièrement, puis à l'effacer et a la réimprimer dans une position légèrement différente. Comme l'image va être redessinée chaque fois, son dessin doit être contenu dans une sous-routine. Mais en premier lieu il faut exécuter le dessin ; nous allons le réaliser en un bloc de 3 sur 4. La 'ligne' du haut sera la tête, puis le corps, puis enfin les jambes.

```
 10 CLEAR 500: CLS0
 20 M1$ = CHR$ (128) + CHR$ (193) + CHR$ (194) + CHR$ (128)
 30 M2$ = CHR$ (196) + CHR$ (207) + CHR$ (207) + CHR$ (200)
 40 M3$ = CHR$ (128) + CHR$ (202) + CHR$ (197) + CHR$ (128)
```

L'image est maintenant contenue dans les trois variables de chaînes M1$, M2$, M3$. Suit la sous-routine qui imprimera les chaînes dans l'ordre voulu.

```
500 P = 32*Y + X
510 PRINT @ P, M1$;: PRINT @ P + 32, M2$;
520 PRINT @ P + 64, M3$;: RETURN
```

Ceci va imprimer l'image sur trois lignes directement les unes en dessous des autres, en commençant à l'endroit déterminé par X et Y. (Souvenez-vous de la grille X, Y. Y va horizontalement de 0 à 31 et Y verticalement de 0 à 15). Pour que le programme tourne il faudra entrer toutes les lignes indiquées ci-dessous, jusqu'à 160.

Maintenant pour faire bouger le personnage, il faut changer la position de PRINT, donc changer X ou Y. Ceci peut être fait au moyen du clavier. Nous utiliserons IKEY$ pour lire le clavier et pour les directions, les touches qui s'imposent sont celles qui portent des flèches. Ces touches elles aussi ont des numéros de code, tout comme les lettres.

[←] CHR$ (8)
[→] CHR$ (9)
[↓] CHR$ (10)
[↑] CHR$ (94)

Nous allons utiliser X pour y loger la position horizontale de l'image et Y pour y loger sa position verticale. Si la touche retour arrière [←] est frappée nous voulons un déplacement vers la gauche, donc il y a lieu de soustraire une unité à la valeur actuelle de X. Mais attention, il ne faut pas dépasser le cadre de l'écran.

```
 90 GOSUB 500
100 A$ = INKEY$: IF A$ = ""THEN 100
120 IF A$ = CHR$(8) THEN X = Y -1: IF X <0 THEN X = 0
```

La ligne 100 scrute le clavier jusqu'à ce qu'une touche soit utilisée. Si c'est [←] alors la ligne 120 ôte un à la valeur de X et vérifie si le cadre de l'écran n'est pas dépassé; si tel est le cas elle bloque le mouvement à la limite gauche. Le déplacement à droite, en haut ou en bas s'exécute de manière similaire.

```
130 IF A$ = CHR$ (9) THEN X = X + 1: IF X > 28 THEN X = 28
140 IF A$ = CHR$ (94) THEN Y = Y - 1: IF Y < 0 THEN Y = 0
150 IF A$ = CHR$ (10) THEN Y = Y + 1: IF Y > 13 THEN Y = 13
160 GOSUB 500: GOTO 100
```

A la ligne 130, nous établissons que la valeur maximale autorisée pour X est 28. Elle ne saurait en fait dépasser 31 mais souvenez vous que notre image a quatre blocs de large. La même chose s'applique à la variable Y; il faut laisser assez de place pour trois lignes. La ligne 160 va à la sous-routine d'impression puis revient pour scruter le clavier et voir si une touche a été utilisée ou pas. Si vous lancez le programme maintenant, vous allez pouvoir promener l'image sur l'écran; mais il va y avoir une belle pagaille parce que l'image précédante ne s'efface pas. Pour la faire disparaître il va nous falloir une chaîne d'effacement et une sous-routine pour l'imprimer.

```
 50 BL$ = CHR$ (128) + CHR$ (128) + CHR$ (128) + CHR$ (128)

600 P = 32*Y + X: PRINT @ P, BL$;
610 PRINT @ P + 32, BL$;: PRINT @ P + 64, BL$
620 RETURN
```

La nouvelle sous-routine (600) fait exactement la même chose que l'autre, sauf que cette fois elle imprime un bloc de carrés noirs. Tout ce qu'il nous reste à faire maintenant est d'effacer l'image juste avant d'imprimer la nouvelle.

```
110 GOSUB 600
```

Vous pouvez maintenant déplacer l'image comme vous le voulez sur l'écran. Sous cette forme, ce programme n'est qu'un exemple, mais le déplacement de sujets de cette façon peut s'incorporer dans des jeux ou des programmes éducatifs pour enfants.

UNE NOUVELLE DEFINITION

Nous allons maintenant passer au niveau de définition suivant. (souvenez-vous que résolution et définition veulent dire la même chose). Ce niveau s'exprime sur une grille de 32 x 64, ce qui donne 2.048 points sur l'écran. Ce niveau et le précédant, de 16 x 33 points, sont les niveaux de basse résolution, ils peuvent au besoin être utilisés en même temps. Pour allumer ou éteindre les points sur l'écran il y a deux commandements:

SET (X,Y,C) et RESET (X,Y)

Le commandement SET allume le point X, Y dans la couleur C. X doit être compris entre Ø et 63 et Y entre Ø et 31. Ce sont les positions horizontales et verticales comme précédemment. C est un nombre de Ø à 8 et exprime la couleur à donner au point que l'on allume.

La commande RESET éteint le point X, Y. En utilisant ces commandements on peut simuler le mouvement par allumage et extinction successives. Essayez le programme suivant.

```
1Ø CLSØ: X1 = Ø: Y1 = Ø: XI = 2: YI = 2

2Ø X2 = X1 + XI: IF X2 > 63 OR X2 < Ø THEN XI = -XI:
   SOUND 18Ø,1: GOTO 2Ø

3Ø Y2 = Y1 + YI: IF Y2 > 31 OR Y2 < Ø THEN YI = -YI:
   SOUND 18Ø,1: GOTO 3Ø

4Ø SET (X2, Y2,8): RESET (X1, Y1): X1 = X2: Y1 = Y2: GOTO 2Ø
```

Vous voyez bien ce qu'il fait: mais comment le fait-il? En partant du point X1, Y1, le programme augmente X1 d'une petite quantité XI et Y1 de YI et ceci crée un nouveau point X2, Y2. Le nouveau point est allumé et l'ancien (X1, Y1) éteint par la ligne 4Ø. Le point X2, Y2 devient alors l'ancien point, le programme retourne à la ligne 2Ø pour créer un nouvel X2, Y2 et ainsi de suite. Ce processus fait se déplacer la "balle" à travers l'écran. Quand la "balle" atteint le bord de l'écran, le signe de l'incrément est inversé, cela veut dire que X (par exemple) commence maintenant à décroître provoquant un changement de direction. Il y a "rebond" sur les bords. Si vous changez la valeur de l'incrément, (XI et YI à la ligne 1Ø) vous déplacerez la balle à des vitesses différentes. Ce type de programme constitue la base de la majorité des jeux de balle sur ordinateur mais ils sont généralement écrits en language machine, pas en BASIC.

Les jeux de tir sont une autre utilisation classique de points en mouvement. Ce type de jeu nécessite un mouvement à travers l'écran et la possibilité de "tirer" un projectile. On pourrait bien sûr utiliser les fléches du clavier, comme tout à l'heure mais une méthode beaucoup plus intéressante est de faire usage des *manettes de jeu*. ("Joysticks" en Anglais).

Les manettes de jeu se branchent dans les prises sur le côté de votre ordinateur et permettent un guidage beaucoup plus précis du mouvement que les touches du clavier. La position indiquée par le joueur, sur les manettes de jeu est lue par la fonction JOYSTK. JOYSTK(Ø) restitue la position horizontale de la manette de jeu gauche et JOYSTK(1) la position verticale. JOYSTK(2) et JOYSTK(3) font la même chose pour la manette de jeu droite. Comme la valeur restituée dans chaque cas sera toujours comprise entre Ø et 63, il va être nécessaire de mettre cette valeur à l'échelle voulue pour qu'elle s'adapte à la finesse de résolution dans laquelle on travaille.

SET

Le commandement SET s'utilise pour allumer un point déterminé de basse définition dans une couleur déterminée.

SET (*x*, *y*, *c*)

x, *y* sont les coordonnées du point sur l'écran. *x* doit être situé entre Ø et 63, *y* entre Ø et 31.

c est le code de couleur choisi et doit donc être compris entre Ø et 8.

```
1Ø CLSØ: SET (5,27,8): SET (6, 27,8)
2Ø FOR X = Ø TO 6: FOR Y = 28 TO 3Ø
3Ø SET (X,Y,8): NEXT Y,X
4Ø FOR X = 7 TO 63: FOR D = 1 TO 2ØØ: NEXT D
5Ø FOR Y = 27 TO 3Ø: IF Y = 27 THEN RESET (X-2,Y)
6Ø SET (X,Y,8): RESET (X-7,Y): NEXT Y,X
7Ø GOTO 7Ø
```

RESET

Le commandement RESET s'utilise pour effacer (éteindre) un point précédemment allumé par la commande SET. Il s'utilise pour le mode graphique en basse définition.

RESET (*x*,*y*)

x, *y* sont les coordonnées du point à éteindre. *x* doit être situé entre Ø et 63 et *y* de Ø à 31.

En fait, le point est ramené à la couleur du fond ; visuellement il est donc effacé. Voir SET pour le programme d'exemple.

```
10 CLS0: FOR I = 0 TO 3
20 PRINT @ 74 + 32*I, "JOYSTK(";I;") ▽ ▽"; JOYSTK(I);
30 NEXT I: FOR D = 1 TO 400: NEXT D: GOTO 10
```

Faites tourner le programme ci-dessus et manipulez les manettes de jeu. Vous allez voir les valeurs changer selon les déplacements des manettes. Vous pouvez aussi utiliser le bouton qui est sur la manettes. Ajouter pour cela la ligne:

```
25 P = PEEK (65280): PRINT @ 202, "VALEUR DU BOUTON ▽ ▽"; P;
```

La fonction PEEK demande à l'ordinateur d'aller consulter un endroit bien défini de sa mémoire. L'adresse mémoire 65280 contient le résultat de la lecture du bouton des manettes de jeu. Ce sera soit 127, ou 255 pour l'instant. Si vous appuyez sur le bouton de *gauche* cette valeur changera à 125 ou 253, appuyez sur le bouton de *droite* et la valeur deviendra 126 ou 254. (Si les deux sont actionnés en même temps, le nombre sera 124 ou 252).

Nous allons maintenant nous mettre au travail sur un programme de jeu. Une bataille entre deux engins spaciaux. Nous allons utiliser les manettes pour déplacer les engins et le bouton pour faire feu.

Tout d'abord les engins: nous allons utiliser une méthode similaire à celle utilisée pour fabriquer l'image dans l'exemple précédent. Chaque engin est un bloc de 2 x 3, l'un jaune, l'autre bleu, le tout stocké dans les tableaux S$ et S2$.

```
10 CLEAR 500: FOR I = 0 TO 5: READ S(I): NEXT I
20 DATA 128, 131, 128, 134, 140, 137
30 FOR Y = 0 TO 1: C = (Y + 1)*16
40 S$(Y) = CHR$(S(0) + C) + CHR$(S(1) + C) + CHR$(S(2) + C)
50 S2$(Y) = CHR$(S(3) + C) + CHR$(S(4) + C) + CHR$(S(5) + C)
60 NEXT Y
```

Si la forme des engins ne vous plaît pas, vous pouvez très bien les dessiner autrement en changeant les données (DATA) à la ligne 20.

Maintenant nous devons lire les données fournies par les manettes, vérifier que nous restons dans les limites de l'écran et établir à quel endroit imprimer les engins.

```
 70 FOR Y = 0 TO 1: A(Y) = JOYSTK(Y*2):
 80 B(Y) = INT(JOYSTK(1 + Y*2)/2)
 85 IF A(Y) >58 THEN A(Y) = 58
 90 IF A(Y) <2 THEN A(Y) = 2
100 IF B(Y) >27 THEN B(Y) = 27
110 L(Y) = INT(B(Y)/2)*32 + INT(A(Y)/2):NEXT Y
```

La position des manettes est analysée par la ligne 70. Les limites sont

définies dans les lignes 80-100 (pensez à la dimension des engins). Le résultat final est ensuite transformé en une valeur pour l'instruction PRINT @. (C'est un exemple de la possibilité de mélanger les deux niveaux de basse résolution - les manettes travaillent dans l'une des définitions et l'instruction PRINT @ dans l'autre).

Maintenant il faut afficher les engins spaciaux puis retrouner voir si ceux-ci ont été déplacés, par les manettes.

```
120 CLS0: FOR Y = 0 TO 1: PRINT @ 0,Z(0);: PRINT @ 26,Z(1);
130 PRINT @ L(Y),S$(Y);: PRINT @ L(Y) + 32,S2$(Y);: NEXT Y

170 A$ = INKEY$: IF A$ = ""THEN 70
180 CLS: END
```

La ligne 120 affiche également la "marque" (le score) - mais pour l'instant nous ne nous en occupons pas. Pour terminer le jeu, taper n'importe quelle touche au clavier, autrement le programme retourne à 70 et analyse à nouveau la position des manettes de jeu.

Faites tourner le programme tel qu'il en est à ce stade et vérifiez que les engins spaciaux se déplacent bien, partout sur l'écran.

La prochaine étape consiste à faire tirer les armes et à afficher l'"éclair de plasma"! C'est la partie la plus délicate à réaliser parce qu'il va falloir analyser les boutons des manettes et savoir lequel des joueurs à tiré. De plus, comme les engins se déplacent sans cesse et partout sur l'écran nous devons connaître la direction de la salve. Pour que le programme reste simple, nous décidons que la salve qui part de l'engin qui tire suivra une ligne horizontale vers sa cible au niveau correspondant à la position verticale de l'engin attaquant.

```
140 P = PEEK(65280)
150 IF P = 125 OR P = 253 THEN F = 0: T = 1: GOSUB 200
160 IF P = 126 OR P = 254 THEN F = 1: T = 0: GOSUB 200
```

Ces lignes lisent les boutons des manettes et analysent d'où est parti le coup. La salve sera matérialisée sur l'écran par la sous-routine 200.

```
200 V1 = B(F): H1 = A(F): H2 = A(T): ST = 1
210 IFH1>H2 THEN ST = -1
220 FOR H = H1 + ST*5 TO H2 + 2 STEP ST

240 SET (H,V1,4): SOUND 200,1: RESET (H-2*ST,V1)
250 NEXT H: RETURN
```

Vous remarquerez que le pas s'inverse à la ligne 210, si les positions gauche et droite sont inversées. Le mouvement du "plasma" est créé par la ligne 240.

Tout ce qu'il reste à faire maintenant c'est de vérifier si le tir à touché au but et si tel est le cas, à faire les bruits appropriés et a enregistrer la marque.

C'est la fonction POINT qui vérifie si le coup a porté. La forme de cette fonction est: POINT (X,Y), X, Y étant le point que vous voulez vérifier. La fonction resitue Ø si le point est éteint et le chiffre du code couleur si le point est allumé.

Comme l'écran est noir et que les joueurs tirent dans la bonne direction (en tous cas ils essaient), il suffit de vérifier si un point de couleur est allumé dans la ligne de feu. Si nous ajoutons la ligne suivante à notre sous-routinne 2ØØ

```
23Ø IF POINT (H,V1)>Ø THEN GOSUB 3ØØ: RETURN
```

chaque fois qu'un coup porte au but, le programme ira à la sous-routine 3ØØ (celle où nous enregistrons la marque, etc...) et quand il reviendra, il repartira à son point de départ.

```
3ØØ Z(F) = Z(F) + 1
31Ø FOR K = 1 TO 15: I = RND(5)-2:J = RND(4)-2
32Ø SET(H+I*ST, V1+J,8): SOUND(RND(95)),1
33Ø NEXT K: RETURN
```

Cette sous-routine compte les coups et tient la marque, dessine l'explosion, et crée le bruit.

Bien qu'il ne se compose que de 28 lignes, ce programme est suffisant pour réaliser un jeu qui comporte déjà beaucoup de mouvement. Nous vous laissons le soin, pour vous exercer et pour vous amuser, de le transformer en un jeu du niveau de ceux du commerce.

Ceci a été un chapitre long et parfois difficile mais il contient la plupart des éléments nécessaires aux fonctios graphiques de votre ordinateur, quel que soit le niveau de définition que vous employez.

CHAPITRE HUIT

PASSAGE A LA HAUTE RESOLUTION

Nous passons maintenant aux écrans à haute résolution qui sont complètement séparés des écrans à basse résolution. Les deux écrans à basse résolution peuvent être utilisés en même temps et sont affichés sur ce que l'on appelle l'"écran de texte". L'écran à haute résolution ne peut pas être mêlé à l'écran de texte. Vous pouvez par contre passer de l'un à l'autre mais il n'est pas possible d'écrire du texte sur l'écran haute résolution, ni de dessiner des graphiques haute résolution sur l'écran de texte.

Quand quelque chose est dessiné en haute résolution, l'ordinateur inscrit les instructions d'affichage dans une partie spéciale de sa mémoire qui s'appelle la 'RAM vidéo'. La RAM vidéo donne lecture de son contenu à la TV, ou il est couverti en images. Un certain nombre de "pages" généralement quatre est réservé dans la RAM vidéo pour cet usage. A mesure que le nombre de détails que vous ajoutez augmente, le nombre d'instructions requises augmente aussi et nécessite donc plus de place; il vous faudra en conséquence réserver plus de pages. Ceci peut se faire en utilisant le commandement PCLEAR, suivi du nombre de pages que vous souhaitez réserver, (8 au maximum).

PCLEAR 8

Etant donné que chaque page occupe 1.536 emplacements de mémoire, ne réservez strictement que ce dont vous avez besoin. La quantité de mémoire disponible dans votre ordinateur est une constante: plus vous en réservez pour les pages graphiques, moins il vous en reste pour le programme. PCLEAR se comporte de la même façon que CLEAR et doit donc être utilisé au début d'un programme.

LE MODE

La quantité d'espace qu'il vous faut réserver dépend du niveau de résolution que vous voulez utiliser. Un désavantage des définitions élevées est qu'il n'est plus possible d'utiliser toute la gamme de couleurs disponible dans les basses résolutions. Les couleurs disponibles et la résolution dépendent du *mode* dans lequel vous décidez de travaillez. Le mode est fixé par le commandement PMODE.

PMODE *mode, page de départ*

mode est un chiffre de Ø à 4 et *page de départ* est la 'page' de la RAM vidéo par laquelle vous voulez commencer. Comme précédemment les écrans sont divisés en grilles. Mais maintenant il n'y a qu'une seule grille de (256 x 192). Bien que la résolution change selon le mode choisi, on se réfère

PCLEAR

PCLEAR s'utilise pour réserver des pages graphiques dans les modes de haute définition.

PCLEAR *n*

n doit être compris entre 1 et 8. Si aucune instruction PCLEAR n'est donnée, il sera automatiquement réservé 4 pages.

Comme chaque page graphique occupe 1.536 octets de mémoire, ne réservez strictement que ce qu'il vous faut.

NOMBRE PMODE	TAILLE GRILLE	TAILLE POINT	PAGES UTILISEES	JEU DE COULEURS DISPONIBLE	
				ECRAN 1,0	ECRAN 1,1
Ø	128 x 96	■■ ■■	1	Noir (Ø), Vert (1)	Noir (Ø), Blanc (5)
1	128 x 96	■■ ■■	2	Vert (1), Jaune (2) Bleu (3), Rouge (4)	Blanc (5), Cyan (6) Magenta (7), Orange (8)
2	192 x 128	■■	2	Noir (Ø), Vert (1)	Noir (Ø), Blanc (5)
3	192 x 128	■■	4	Vert (1), Jaune (2) Bleu (3), Rouge (4)	Blanc (5), Cyan (6) Magenta (7), Orange (8)
4	256 x 192	■	4	Noir (Ø), Vert (1)	Noir (Ø), Blanc (5)

SCREEN

Le commandement SCREEN s'utilise pour mettre l'affichage en mode texte ou graphique. (Screen en Anglais, écran).

SCREEN *type, gamme de couleurs*

type est soit Ø pour le texte et les graphiques en basse résolution, soit 1 pour la haute résolution.

la *gamme de couleurs* est soit Ø soit 1. La gamme de couleurs pour le texte est Ø: noir sur vert ou 1: noir sur orange. Pour les graphiques haute résolution la *gamme de couleurs* disponible dépend du mode choisi, selon le tableau ci-après.

PMODE	SCREEN 1,Ø	SCREEN 1,1
Ø	Noir, vert	Noir, blanc
1	Vert, jaune bleu, rouge	Blanc, vert magenta, orange
2	Noir, vert	Noir, blanc
3	Vert, jaune bleu, rouge	Blanc, cyan magenta, orange
4	Noir, vert	Noir, blanc

toujours aux points de l'écran en utilisant la grille 256 x 192. La différence réside seulement dans la taille du point obtenu. La sélection du mode décide également des couleurs dont vous pouvez disposer. Chaque mode offre deux gammes de couleurs. La gamme de couleur est choisie par le commandement SCREEN qui sélectionne également le type d'écran.

SCREEN *type, gamme de couleurs*

Le *type* Ø correspond à l'écran de texte et 1 à l'écran de haute résolution. La *gamme de couleur* est également Ø ou 1. En l'absence d'indication, c'est SCREEN Ø, Ø qui est établi automatiquement. C'est ce que nous avons appliqué jusqu'à présent, et cela donne un écran de texte en noir sur fond vert. (Il est possible d'utiliser SCREEN Ø,1 qui donne un texte noir sur fond orange, mais chaque fois que l'ordinateur imprime, il va retourner à noir sur fond vert). Pour afficher l'écran en haute définition vous devez donner à *type* la valeur 1. Le tableau de la page ci-contre vous indique toutes les possibilités offertes par chaque cas.

Comme vous pouvez le constater, la résolution et la couleur sont liées. Vous remarquerez également que lorsque le mode augmente, de Ø à 4, le nombre de pages nécessaires augmente aussi.

Donc pour afficher un écran en mode graphique PMODEØ il ne faut qu'une page de mémoire, tandis que avec PMODE 3 et PMODE 4 il faut 4 pages. Quand la gamme de couleur a été sélectionnée, l'ordinateur choisit la couleur dont le numéro est le plus faible comme couleur de fond et le numéro le plus élevé dans la gamme choisie comme couleur d'écriture, d'"encre". Par exemple, avec PMODE 3 et SCREEN 1,Ø en place, l'ordinateur va dessiner en rouge sur un fond vert. Vous pouvez changer l'encre et le fond avec le commandement COLOR.

COLOR, *encre, fond*

Encre et *fond* sont les numéros de couleur choisis dans la gamme admise pour ce mode.

DES VIEILLES CONNAISSANCES

Vous vous souvenez des commandements CLS et SET, RESET et POINT, utilisés en base résolution. Ils ont leur équivalents en haute résolution; ce sont: PCLS, PSET, PRESET et PPOINT. Leurs usages sont les mêmes qu'avant: PCLS vide l'écran haute résolution et si il est suivi d'un numéro de code de couleur, il crée un fond de cette couleur. PSET allume un point et PRESET l'éteint. PPOINT vérifie si le point est allumé ou éteit. L'exemple suivant passe tour à tour en revue chaque mode et chaque gamme de couleur disponibles. Il affiche des points de couleurs choisies au hasard, sur

une grille rectangulaire. Des endroits vides sont visibles dans la grille; ils résultent soit de ce que la couleur choisie par le hasard à cet endroit se trouve être la couleur de fond, soit de ce que cette couleur n'est pas disponible dans la gamme choisie.

```
10 FOR P = 0 TO 4: PMODE P, 1
20 FOR S = 0 TO 1: SCREEN 1,S
30 PCLS: FOR I = 50 TO 150 STEP 20
40 FOR J = 50 TO 150 STEP 20
50 C = RND(8):PSET(I,J,C): NEXT J,I
60 FOR D = 1 TO 1000: NEXT D,S,P
```

Regardez bien la taille du point; elle vous montre quelle est la résolution disponible dans ce mode.

TIRER DES LIGNES

Bon, nous savons mettre des ponts sur l'écran... Et ensuite? La chose la plus évidente à faire de deux points est de les joindre par une ligne. Heureusement, il y a une instruction qui fait exactement cela, LINE. (Line en Anglais, ligne). Effacez les lignes 40 et 50 de l'exemple précédent, changez la ligne 30 en,

```
30 PCLS: LINE(10,180)-(252,10),PSET
```

et faites tourner le programme. Une ligne se dessine à travers l'écran du bas à gauche vers le haut à droite. L'instruction de la ligne 30 signifie: tirez une ligne du point de départ (10,180) au point d'arrivée (245,10) dans la couleur d'encre (PSET). Si vous changez PSET en PRESET, la ligne sera tirée mais dans la couleur du fond: donc invisible. Mais c'est un truc utile à connaître car on peut l'utiliser pour effacer une ligne. PSET et PRESET font partie intégrante de l'instruction LINE mais elles n'ont rien à voir avec les commandements qui allument ou éteignent des points.

Il n'est pas toujours nécessaire de spécifier le point de départ de LINE. Sans mention de point de départ, la ligne débutera au dernier point d'arrivée. (Si la commande LINE entre en scène pour la première fois dans le programme, le dernier point d'arrivée sera supposé être 128,96: le centre de l'écran). Ajoutez maintenant une ligne supplémentaire à l'exemple en cours.

```
40 LINE - (130,180),PSET
```

Une ligne est maintenant tirée à partir du dernier point de ligne (245,10) jusqu'au point 130, 180 dans le bas de l'écran.

COLOR

Le commandement COLOR s'utilise pour changer les couleurs de fond et d'encre automatiquement utilisées en modes graphiques de haute définition en l'absence d'instructions contraires

COLOR *encre, fond*

Les numéros d'*encre* et de *fond* sont tous les deux des chiffres compris entre Ø et 8, représentant la couleur souhaitée. Les deux couleurs doivent faire partie de la gamme de couleurs disponibles dans le mode choisi.

PCLS

Le commandement PCLS s'utilise pour vider l'écran et y établir une couleur de fond donnée, en mode de haute résolution.

PCLS *c*

c est le code de la couleur de fond souhaitée. Ce doit être l'une des couleurs disponibles dans la gamme du mode choisi. Si la couleur demandée n'est pas disponible ou si *c* est omis, le fond prendra la couleur prévue normalement.

Voir l'encadré CLS pour les codes couleur.

PSET

C'est la version haute résolution de la commande SET.

PSET (*x,y,c)*

allume le point (*x,y*) en la couleur *c*. *x* doit être entre Ø et 255, *y* doit être situé entre Ø et 191.

c'est une couleur, de Ø à 8 et doit être l'une de celles qui sont disponibles.

PRESET

C'est la version haute résolution de la commande RESET.

PRESET (*x, y*)

Eteint le point (*x,y*) en le rétablissant dans la couleur de fond. *x* doit être situé entre Ø et 255 et *y* entre Ø et 191.

Pour dessiner un carré ou un rectangle, on pourrait tracer quatre lignes mais une extension de la commande LINE permet de procéder autrement. Utiliser l'EDITEUR pour ajouter un B à la ligne 30 qui devient maintenant:

```
30 PCLS: LINE(10,180) - (245,10),PSET,B
```

et au lieu d'une ligne diagonale, vous obtenez un rectangle. Donc pour dessiner un rectangle, il suffit de spécifier la position des deux coins diagonalement opposés et d'ajouter B à l'instruction LINE. Retournez maintenant à l'EDITEUR et ajoutez F à la fin de la ligne 30, ce qui doit vous donner:

```
30 PCLS: LINE(10,180) - (245,10),PSET,BF
```

Le F que nous venons d'ajouter veut dire: remplisser le cadre (le rectangle) avec la couleur de l'encre. Une instruction aussi performante mérite d'être utilisée, donc profitons en pour dessiner quelque chose.

Comme la dernière fois, nous opérons dans la construction mais cette fois nous mettrons une maison en chantier. Faites tourner le programme après chaque section de façon à observer la progression de la construction.

Tout d'abord nous allons choisir la résolution et dessiner le corps de la maison.

```
10 PMODE 3,1: SCREEN 1,0:PCLS
20 LINE (60,48) - (200,144),PSET,B

260 GOTO 260
```

Maintenant il nous faut un toit.

```
40 LINE (60,48) - (130,20),PSET
50 LINE - (200,48),PSET
```

Et un garage, avec une porte.

```
70 LINE (200,144) - (255,94),PSET,B
90 LINE (210,144) - (245,104),PSET,BF
```

Nous pouvons utiliser la même technique pour mettre une porte à la maison.

```
100 LINE (160,144) - (188,105),PSET,BF
```

Pour dessiner des fenêtres, il nous faut un rectangle avec deux lignes pour former des croisillons.

```
110 LINE (85,132) - (135,108),PSET,B
120 LINE (110,108) - (110,132),PSET
130 LINE (85,120) - (135,120),PSET
```

Les fenêtres du premier étage sont obtenues par le même système.

```
140 LINE (90,84) - (125,64),PSET,B
150 LINE (90,74) - (125,74),PSET
160 LINE (110,84) - (110,64),PSET
```

LINE

Le commandement LINE s'utilise pour dessiner des lignes et des rectangles en mode de haute résoltion graphique.

LINE (*x*1, *y*1) - (*x*2, *y*2), *a, b*

*x*1, *y*1 sont les coordonnées du point de départ de la ligne.
*x*2, *y*2 sont les coordonnées du point d'arrivée de la ligne.
a représente soit PSET ou PRESET. Si PSET est utilisé, la ligne sera dessinée dans la couleur de l'encre. Si PRESET est utilisé, la ligne sera dessinée dans la couleur du fond (donc effacée). *b* est un paramétre optionel. S'il est utilisé ce doit être B ou BF. Si c'est B, un rectangle sera dessiné au lieu d'une ligne; le coin supérieur du rectangle sera *x*1, *y*1 et le coin inférieur droit *x*2, *y*2. Si BF est utilisé, le rectangle sera dessiné et entièrement colorrié dans la couleur de l'encre.

```
10 PMODE 4,1: SCREEN 1,1: PCLS 5: COLOR 0,5
20 FOR I = 1 TO 1000
30 X = X + L * SIN(R): Y = Y + L * COS(R)
40 IF X<-128 OR X>128 THEN 90
50 IF Y<-96 OR Y>95 THEN 90
60 LINE - (X + 128,Y + 96),PSET
70 R1 = R1+60: R = R1/57.29578: L = L+0.5
80 NEXT 1
90 GOTO 90
```

```
170 LINE (155,64) - (175,84),PSET,B
180 LINE (165,84) - (165,64), PSET
190 LINE (155,74) - (175,74),PSET
```

et pour compléter l'ensemble: une cheminée,

```
200 LINE (150,40) - (160,15),PSET,BF
```

Ce petit programme montre comment on peut obtenir un dessin très rapidement en n'ayant recours qu'à une seule instruction. Ceci suppose bien sûr, que lon sache exactement à quel endroit faire tomber les lignes. La façon la plus facile de repérer ces points est de tirer une copie de la grille des graphiques de l'appendice B et de dessiner ce que l'on souhaite obtenir sur cette feuille. Il suffit, ensuite de lire les coordonnées de chaque point.

UNE COUCHE DE COULEUR

Notre maison à peut être l'air un peu terne... Ce qu'il lui faudrait c'est une bonne couche de couleur, pour lui donner un air gai. Nous allons dire à DRAGON de prendre ses pinceaux et de se mettre à l'œuvre. Le commandement PAINT permet de peindre n'importe quelle forme avec n'importe quelle couleur disponible. (en Anglais PAINT = peinture). Tout ce que vous avez à faire c'est de lui dire à quel endroit commencer, quelle couleur utiliser et quelle est la couleur de la bordure à laquelle la peinture devra s'arrêter.

PAINT *(x, y), a, b*

x, y étant les coordonnées du point de départ, *a* et *b* les codes des couleurs de la peinture et de la bordure. Ajoutez la ligne suivante dans le programme de la maison.

```
 30 PAINT (90,90),2,4
```

Ceci veut dire, commencer au point (90,90), peindre en jaune (couleur 2) jusqu'à ce que l'on rencontre une bordure de couleur rouge (4). Faites tourner le programme et voyez ce qu'il fait. Maintenant annulez la ligne précédente et réintroduisez la avec le numéro de ligne 195. Faites tourner le programme à nouveau.

```
195 PAINT (90,90),2,4
```

Voyez comment il s'arrête à la limite des fenêtres qui n'étaient pas là avant. Peignez maintenant le garage de la même façon.

```
 80 PAINT (210,140),2,4
```

Maintenant le toit. Si vous omettez de spécifier une couleur, ou une référence de bordure dans une instruction PAINT, la couleur en cours de l'encre sera utilisée dans les deux cas.

```
 60 PAINT (130,25)
```

Et nous terminons l'ensemble par une lignes d'horizon et le ciel.

```
210 LINE (0,64) - (60,64),PSET
220 LINE (200,64) - (255,64),PSET
230 PAINT (0,54),3,4
```

PAINT

Le commandement PAINT s'utilise en mode de haute résolution graphique, pour remplir une forme avec une couleur spécifiée.

PAINT *(x, y), c, b*

x, y sont les coordonnées du point à partir duquel la peinture doit commencer.

c, est le code de couleur de la peinture à utiliser. Il doit être compris entre Ø et 8 et doit être l'une des couleurs disponibles dans la gamme du mode utilisé. Si *c* est omis, la couleur courante de l'encre sera utilisée.

b est le code de couleur de la bordure à laquelle la peinture doit s'arrêter. Il doit également être compris entre Ø et 8. La peinture continuera à s'étendre au delà de toute bordure d'une autre couleur. Si *b* est omis, la couleur courante de l'encre sera utilisée.

Voir CIRCLE pour un exemple d'utilisation.

Notre maison paraît maintenant plus riante. S'il vous plaît de continuer les améliorations, pourquoi ne pas rajouter un chemin d'accès et une clôture. Ou alors prener le temps de faire quelques dessins et peintures suivant vos propres idées.

FAIRE DES RONDS

Nous avons des lignes, des carrés et des rectangles, il nous faut maintenant des cercles. L'instruction CIRCLE dessine des cercles, des ellipses et des arcs.

CIRCLE (x, y) *rayon, couleur, rapport hl, début, fin*

Le point *x, y* est le centre du cercle; le *rayon* c'est le rayon du cercle mesuré en points d'écran. La *couleur* est l'une des couleurs disponibles dans le mode utilisé, (si la couleur est omise, la couleur de l'encre est utilisée). Les autres paramétres serviront à dessiner des ellipses et des arcs, nous les traiterons ultérieurement. Voyons d'abord ce qui se passe pour les cercles.

```
10 FOR P = 0 TO 4: PMODE P,1
20 SCREEN 1,1: PCLS
30 FOR R = 120 TO 10 STEP - 10
40 CIRCLE (128,96),R: NEXT R
50 FOR D = 1 TO 500: NEXT D,P
```

Ceci va dessiner des cercles autour du centre de l'écran. Les cercles sont difficiles à dessiner et pour obtenir un résultat vraiment précis vous devrez utiliser PMODE 4.

Vous remarquerez que si un cercle dépasse les limites de l'écran, il n'y a pas de problème. Si vous essayer de tirer une ligne vers un point qui n'est pas sur l'écran, il se pourrait qu'elle ne soit pas dessinée du tout, surtout dans les modes des résolutions les plus élevées. Essayer d'insérer

```
42 LINE - (300,40),PSET
```

et voyez le résultat.

Le commandement PAINT peut également s'utiliser pour remplir les cercles,

```
45 PAINT (128,96)
```

va remplir la partie centrale.

En utilisant le paramétre de *rapport hl* vous allez pouvoir changer les cercles en ellipses. Le *rapport hl* signifie rapport entre hauteur et largeur. La largeur, dans CIRCLE reste toujours la même: deux fois le rayon. La hauteur peut être modifiée par le *rapport hl*; si celui-ci est supérieur à I alors le 'cercle' sera plus haut que large. Une valeur inférieure à I écrasera le cercle dans l'autre sens, plus large que haut. Donc la largeur le long de l'axe X (horizontal) reste

CIRCLE

Le commandement CIRCLE dessine des cercles, des ellipses et des arcs. Il ne peut s'utiliser que dans les modes de haute définition graphique.

CIRCLE *(x, y), r, c, hl, début, fin*

x	est la coordonnée X du centre du cercle (de Ø à 255)
y	est la coordonnée Y du centre du cercle (de Ø à 191)
r	est le rayon du cercle, mesuré en points d'écran
c	est le code de couleur (de Ø à 8), ce doit être l'une des couleurs disponibles. Si *c* est omis, utilisation de la couleur de l'encre.
hl	est le rapport hauteur-largeur (de Ø à 255). S'utilise pour dessiner des ellipses. Si *hl* est omis, 1 est utilisé.
début	est le début de l'arc de cercle (de Ø à 1). La position Ø est située à 3 heures. Si *début* est omis, Ø est utilisé.
fin	c'est la fin de l'arc du cercle (de Ø à 1). Le dessin s'effectue dans le sens des aiguilles d'une montre, à partir du point *début*. La position .5 correspond à 9 heures. Si *début* est omis, 1 est utilisé.

```
10 PMODE 3,1: SCREEN 1,0: PCLS
20 CIRCLE (180,156), 28,3: PAINT (180,156),3,3
30 CIRCLE (110,156), 28/3: PAINT (110,156),3,3
40 CIRCLE (144,80), 68,4,1,0,.5
50 LINE (212,80) - (76,80),PSET: LINE - (48,32), PSET
60 PAINT (144,82): CIRCLE (144,80),70,4,.8,.79,1
70 LINE (160,80) - (160,28),PSET: PAINT (210,75)
80 GOTO 80
```

constante, seule la hauteur sur l'axe des Y (vertical) varie. Lorsque le *rapport hl* est Ø, le 'cercle' en est réduit à une ligne horizontale et lorsque le *rapport hl* est très grand, l'on se rapporche d'une ligne verticale (plutôt un rectangle long et mince). La valeur maximum autorisée est 255. Transformez les lignes 3Ø et 4Ø de l'exemple en cours:

```
3Ø FOR H = Ø.5 TO 3 STEP Ø.5
4Ø CIRCLE (128,96),4Ø,,H: NEXT H
```

Vous remarquerez à la ligne 4Ø les virgules supplémentaires. La raison en est que nous avons omis le paramétre *couleur*.

La variante finale de CIRCLE est la possibilité de dessiner des arcs (parties de cercle). Pour utiliser cette possibilité, vous devez indiquer le début et la fin de l'arc. Les valeurs de début et de fin doivent toutes deux avoir une valeur comprise entre Ø et 1. Le point de départ du cercle se situe à la position 3 heures sur une pendule. Le dessin se déroule dans le sens des aiguilles d'une montre à partir du point de début. Un départ à Ø,25 et une fin à Ø,75 par exemple, vont dessiner un arc allant de 6 heures à 12 heures, soit la moitié gauche 'un cercle. Pour dessiner la moitié supérieure: départ à Ø,5 et fin à 1,Ø. Le programme suivant utilise des arcs de cercle pour former un motif.

```
1Ø PMODE 4,1: SCREEN 1,1: COLOR Ø,5: PCLS
2Ø FOR R = 15 TO 6Ø STEP
3Ø CIRCLE (128,96 + R), R,, 1,.5,1
4Ø CIRCLE (128,96 - R),R,,1,Ø,.5
5Ø CIRCLE (128 - R,96), R,, 1, .75, .25
6Ø CIRCLE (128 + R,96), R,,1,.25,.75
7Ø FOR D = 1 TO 5ØØ: NEXT D
8Ø NEXT R
9Ø GOTO 9Ø
```

TOURNER LES PAGES

Pour "animer" un dessin, un procédé bien connu consiste à dessiner sur des pages successives des images qui se modifient légérement d'une page à l'autre et a les feuilleter très rapidement. Vous vous souvenez d'avoir réservé un certain nombre de pages de mémoire au moyen de la commande PCLEAR et d'avoir décidé par PMODE sur quelle page vous alliez écrire. Bien sûr vous devez penser à la définition que vous utilisez. En PMODE 3 et PMODE 4 chaque écran graphique nécessite 4 pages donc on ne peut en fait, que passer de la page 1 à la page 5. En PMODE 1 et PMODE 2 qui ne nécessitent que 2 pages, on peut passer des pages 1 à 3 à 5 à 7. L'exemple suivant montre comment cela se fait; entrez par INPUT toutes les valeurs de PMODE.

```
10 PCLEAR 8: P MODE 3, 4: PCLS
20 INPUT "MODE"; M: ON M GOTO 40, 40, 50, 50
30 S = 1: GOTO 60
40 S = 2: GOTO 60
50 S = 4
60 FOR P = 1 TO 8 STEP S: P MODE M, P: PCLS
70 LINE (128,0) - (128,(P-1)*15),PSET
80 SCREEN 1,1: FOR I = 1 TO 1000: NEXT I,P
90 FOR P = 1 TO 8 STEP S: GOSUB 150: NEXT P
100 IF M >2 THEN D = 4: S1 = 3 ELSE D = 7: S1 = S
110 FOR P = D TO 1 STEP - S1: GOSUB 150: NEXT P
120 GOTO 90
150 PMODE M,P: SCREEN 1,1
160 FOR T = 1 TO 20: NEXT T: RETURN
```

Les lignes 60 à 80 dessinent un motif modifié, sur les différentes pages. Ce travail se fait sans affichage à l'écran, puisque aucune instruction SCREEN n'a encore été donnée. Le reste du programme affiche chaque page tour à tour, faisant tourner les pages vers l'avant, puis vers l'arrière, pour donner l'illusion du mouvement comme dans les dessins animés. Vous constatez que plus on emploie de pages, plus le mouvement est continu.

Une autre façon d'utiliser les pages de mémoire est donnée par le commandement PCOPY.

PCOPY *page d'origne* TO *page destinataire*

Vous pouvez copier le contenu de toute page, sur une autre page, à la condition que la page ait été préalablement réservée au moyen de PCLEAR. PCOPY peut aussi s'utiliser pour caser des duplicatas sur une page PMODE 3 ou PMODE 4. Le programme suivant vous montre comment PCOPY s'utilise pour cet usage. Vous remarquerez qu'il y a lieu de faire attention à l'endroit ou l'on place l'image.

```
10 PCLEAR 8: PMODE 3, 4: PCLS
20 LINE (100,20) - (140,40),PSET, BF
30 CIRCLE (50,25),20
40 CIRCLE (200,50),20
50 FOR D = 3 TO 1 STEP - 1
60 PCOPY 4 TO D: NEXT D
70 FOR P = 4 TO 1 STEP - 1: PMODE 3, P
80 SCREEN 1,1: FOR I = 1 TO 1000: NEXT I, P
90 GOTO 90
```

En PMODE 3 (et 4), l'affichage se compose de quatre pages, la page 1 étant le quart du haut de l'écran, la page 2 le suivant, etc... Donc en copiant le

contenu de la page 4 sur la page 1, vous avez dupliqué le quart haut de l'affichage dans le quart bas. Avec PMODE 1 et PMODE 2 on peut obtenir le même effet mais cette fois l'écran sera divisé en moitié et non plus en quarts.

Ceux d'entre vous qui sont avides de continuer à se documenter sur les fonctions graphiques peuvent passer directement au chapitre 1Ø. Les autres vont maintenant prendre part à un petit interméde musical.

PCOPY

PCOPY est un commandement de haute résolution graphique qui s'utilise pour copier le contenu d'une page graphique sur une autre page graphique.

PCOPY *page d'origine* TO *page destinataire*

page d'origine et *page destinataire* doivent être des chiffres compris entre 1 et 8 et doivent se référer à des pages préalablement réservées par le commandement PCLEAR. L'espace requis pour stocker un affichage d'écran diffère selon chaque mode. Il faut en tenir compte en utilisant PCOPY.

PCOPY 3 TO 5

CHAPITRE NEUF

LA PISTE SONORE

ADDITION DU SON

Les graphiques et d'autres programmes peuvent être rendus plus attrayants en leur ajoutant le son car la communication entre l'ordinateur et le monde extérieur devient alors aurdio-visuelle. Nous avons déjà fait usage du commandement SOUND, particulièrement dans l'exemple du chapitre sept. Une méthode plus facile pour ajouter du son est de le fournir vous-même. Non: on ne va pas vous demander de chanter quelque chose pendant que votre programme se déroule à l'écran, enfin pas tout à fait... Votre ordinateur utilise un magnétophone à cassettes pour stocker les programmes; ce même magnéthophone peut aussi jouer un enregistrement sonore. Les instructions MOTOR ON et AUDIO OFF vont le permettre, couplées avec les instructions AUDIO ON et AUDIO OFF qui mettent en circuit ou hors circuit la sortie cassette vers le haut-parleur de la TV. Ceci revient à dire qu'en mettant ces instructions dans votre programme, vous pourrez avoir un fond musical pour vos graphiques. Où pour une application plus sérieuse, une bande pré-enregistrée pourrait fournir des instructions et des questions dans un programme d'enseignement. L'exemple suivant montre que c'est facile à réaliser. Si vous n'avez pas de bande de son sous la main, utilisez une bande contenant un programme. Les bruits étranges que vous entendrez correspondent au language que les ordinateurs utilisent pour communiquer entre-eux!

```
10 CLS: PRINT @ 135, "APPUYER SUR LA BARRE D'ESPACEMENT"
20 PRINT @ 195, "POUR ARRETER OU FAIRE PARTIR LE
   MAGNETOPHONE"
30 A$ = INKEY$: IF A$ < > "∇" THEN 30
40 IF F = 0 THEN MOTOR ON: AUDIO ON: F = 1 ELSE
   MOTOR OFF: AUDIO OFF: F = 0
50 GOTO 30
```

Rembobinez la bande entièrement et appuyer sur le bouton PLAY ou →, puis faites tourner le programme informatique. En appuyant sur la barre d'espacement vous pouvez faire partir, arrêter, repartir le magnétophone.

Ce système est très utile pour les programmes d'enseignement comportant des questions et réponses. On peut réaliser un programme de ce genre qui serve à plusieurs enseignements différents, rien qu'en changeant la bande contenant les questions. De même, vos animations graphiques peuvent être agrémentées d'un fond sonore ou d'effets de son divers.

JOUEZ UN AIR DE MUSIQUE

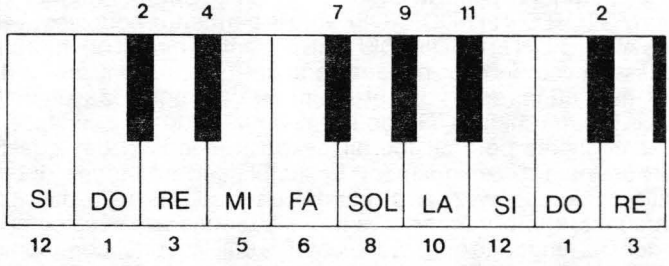
Vous pouvez également obtenir de votre ordinateur qu'il joue de la musique. Le commandement PLAY traduit le contenu d'une chaîne de sons.

PLAY *chaîne*

La *chaîne* peut être une constante de chaîne ou une variable de chaîne. Mais pas n'importe quelle chaîne: il faut une *chaîne musicale* construite à partir de: *note, octave, longueur de la note, tempo et pauses.* La *note* c'est de toute évidence, la note que vous voulez jouer. Si vous étiez anglais, ce serait très facile car il vous suffirait de programmer les lettres qui en anglais représentent les notes. Malheureusement, pour nous c'est un peu plus compliqué mais le tableau de correspondance ci-après va vous y aider.

C = DO	D = RE	E = MI
F = FA	G = SOL	A = LA
B = SI		

Donc pour la *note* vous utilisez A, B, D, E, F, G. Pour indiquer une dièse, vous utilisez # ou + (F# ou F+ pour fa dièse), et pour une bémol - (B- pour si bémol). L'ordinateur ne reconnaîtra pas B# ni C- car si dièse et do bémol ne figurent pas dans la gamme musicale normale. Une autre façon de programmer des notes consiste à utiliser le numéro représentant sa position dans la gamme de 12 tons.



Les notes et leurs équivalents numériques sont indiqués sur le clavier ci-dessus.

L'instruction PLAY peut être utilisée en instruction directe, ce qui est bien utile pour vérifier une chaîne musicale, avant de la rentrer dans un programme. Comme tous les bons musiciens, nous allons commencer par nous exercer aux gammes.

PLAY "CDEFGABCCBAGFEDC" la gamme en do

PLAY "GABCDEF# GGF# EDCBAG" la gamme en sol

Bon... la gamme de do est a peu près juste mais la gamme de sol ne vaut rien. Cela provient de ce que les gammes se prolongent dans une *octave* différente et cela doit, être dit à l'ordinateur. Pour lui faire sélectionner une *octave* employer O suivi d'un nombre entre 1 et 5. L'octave O2 (dans laquelle se trouve le do moyen) est automatiquement sélectionnée lorsque l'ordinateur est mis en marche. L'octave en cours d'exécution est conservée jusqu'à l'apparition d'une instruction différente. Il est donc plus prudent de toujours spécifier quelle octave on veut utiliser, avant de commencer. Essayons les gammes de nouveau.

PLAY "O3CDEFGABO4CCO3BAGFEDC"

PLAY "O3GABO4CDEF# GG# EDCO3BAG"

Pour jouer la gamme de do en utilisant les chiffres au lieu des lettres.

PLAY "O3; 1; 3; 5; 6; 8; 10; 12; O4; 1; 1; O3; 12; 10; 8; 6; 5; 3; 1"

Vous remarquerez la présence du séparateur (;) utilisé dans la chaîne. Vous pouvez utiliser le point-virgule où vous le voulez mais avec les nombres il est indispensable pour éviter la confusion.

AUDIO

Le commandement AUDIO contrôle la liaison entre la sortie son du magnétophone à cassettes vers le haut parleur du téléviseur. AUDIO ON ouvre le passage du son du magnétophone vers le téléviseur. AUDIO OFF interrompt la liaison.

MOTOR

Le commandement MOTOR contrôle la marche du magnétophone à cassettes. MOTOR ON met le moteur en marche. MOTOR OFF l'arrête.

Pour que le commandement ait de l'effet, il faut que le bouton de marche du magnétophone ait été engagé.

Etant donné qu'une chaîne musicale est quand même une chaîne, elle peut être manipulée comme toute autre chaîne, en utilisant les opérations de chaînes habituelles. L'exemple suivant va jouer la gamme en do sur toute l'étendue de l'instruction PLAY.

```
10 A$ = "CDEFGAB": FOR I = 1 to 5
20 B$ = "O" + STR$(I) + A$
30 PRINT B$: PLAY B$: NEXT I
```

En utilisant le même système, mais avec les nombres au lieu des lettres, nous pouvons jouer toute la gamme chromatique.

```
10 FOR J = 1 TO 5: A$ = "O" + STR$(I) + ";"
20 FOR J = 1 TO 12: PLAYS A$ + STR$(J): NEXT J,I
```

Dans la majeure partie des airs, les notes sont rarement de la même longueur, il nous faut donc établir la durée de chaque note. Ceci s'obient par l'usage du paramétre *longueur de la note* dans la chaîne musicale (L). La lettre L est suvie d'un nombre entre 1 et 255. En général, le nombre utilisé représente une longueur courrament employée en musique. L 1 est une note entière, L 2 une demi, L 4 un quart, etc... Il est donc possible d'obtenir 1/255eme d'une note mais peu de compositeurs l'utilisent! Ceux d'entre vous qui ont l'habitude des partitions musicales connaissent la "note pointée". Le point signifie qu'il faut augmenter la durée d'une note de la moitié de sa valeur normale. Pour obtenir cet effet avec la commande PLAY, vous mettrez un point, (ou autant de points que vous voulez) après le nombre, dans le paramètre L.

L 4. = I/4 + I/8 = une note de 3/8

Nous en savons maintenant assez pour jouer un petit air. Tapez soigneusement ce qui suit,

```
5 CLEAR 500
10 A$ = "O2L4GG;L2BGDL4BB;L2BGL4GB;
        O3L2DDL4CO2B;L1AL4AB;
        O3L2CCO2L4BA;L2BGL4GB;
        L2ADL4F# A;L1G;"
20 B$ = A$ + A$: PLAY B$
```

Les séparateurs (;) sont utilisés pour figurer ici les barres de séparations de la portée; elles ne sont pas indispensables.

Vous devriez pouvoir reconnaître l'air, 'Clementine', mais il est joué trop lentement, ce qui va nous conduire à aborder le paramétre: *tempo*. La lettre T, suivie d'un nombre entre 1 et 255 va régler le tempo. Plus le nombre est élevé, plus l'air est joué vite. Essayez de changer la ligne 20 en

```
20 B$ = A$ + A$: PLAY "T6" + B$
```

PLAY

Le commandement PLAY s'utilise pour créer une séquence musicale. L'argument est une expression de chaîne, une constante de chaîne ou une variable de chaîne. Sa forme est,

PLAY *musique* (to play = jouer)

musique étant construit des éléments suivants

note Une lettre de "A" à "G" ou un chiffre de 1 à 12. (voir à ce sujet le tableau de correspondance entre les notes en Anglais 'A, B, C, D' et les notes en Français 'DO, RE, MI etc...).

octave 'O' suivi d'un nombre entre 1 et 5. Sans indication l'ordinateur adopte O2. Toutes les valeurs non spécifiées sont déterminées a une certaines valeur préétablie dès que l'ordinateur est mis sous tension.

longueur de la note 'L' suivi d'un nombre entre 1 et 255. Sans indication, valeur L4

tempo 'T' suivi d'un nombre entre 1 et 255. Sans indication, valeur T2.

volume 'V' suivi d'un nombre entre 1 et 31. Sans indication V15

longueur de la pause 'P' suivi d'un nombre de 1 à 255

exécution de sous-chaînes 'X' suivi d'une variable de chaîne et d'un point virgule (;)

Une dièse s'indique par '+' ou '#' et le bémol s'indique par '–'

La *longueur de la note* peut être modifiée par addition d'un point (.) après le chiffre, (L2.) pour représenter une note pointée.

L'octave, le volume, le tempo et la longueur de la note peuvent être modifiés en utilisant l'un des suffixes suivants:

+	ajoute un à la valeur courante
-	soustrait un à la valeur courante
>	multiplie la valeur courante par deux
<	divise la valeur courante par deux

```
10 X$ = "O3L4EF# L4.EL8AAG# ABL4O+C# O-B"
20 A$ = "XX$;O4C# O+DC# O-BL2AXX$;
   O+C# DEL8DO-BL<AG# L<AL4.BL8O+C# L4
   DO-BL.4O+C# L9DL<EC# L4.EL8EEEEEL1
   EL4.EL8DC# EDO-BL<AG# L<A"
30 PLAY "T2V20" + A$
```

Livrez vous à une expérience en changeant le 6 en une autre valeur qui corresponde à une vitesse différente d'exécution de ce même air.

La plupart des airs de musiques nécessitent l'insertion de pauses entre les phrases musicales et il faut aussi varier parfois l'intensité de certains passages. Le paramétre *pause* est composé de la lettre P suivie par un nombre. Elle suit le même principe que le paramétre (L) sauf que l'on ne peut pas mettre de point après le chiffre. Pour insérer une pause de la même longueur qu'une note de longueur L4. il faut utiliser P4 P8 . Le paramétre de *volume* nous permet de faire varier l'intensité du son en insérant la lettre V, suivie d'un chiffre entre Ø et 31. Plus le nombre augmente, plus le volume augmente aussi. L'exemple ci-dessous utilise le paramétre *volume* pour produire un crescendo.

```
10 A$ = "V10O2L4GG;L1GP4V14L4GGG;
        L1GP4V18L4GGG;L2BL4BBBV22L2BL4BBB;
        V26O3L2DL4DDDL2DL4DDD;
        V30L1GL2.F# L4C# ;L2EDCO2A;
        L1GL2AL4.DL8A;L2B"
20 PLAY "T5" + A$
```

Un morceau de musique contient souvent un passage qui se répéte à différents endroits, un refrain par exemple. Au lieu de ré-écrire ce passage plusieurs fois, il vaut mieux le mettre dans une variable de chaîne séparée. Le commandement X d'*exécution de sous-chaînes* permet à cette sous-chaîne de faire partie de la séquence normale d'exécution. Le X doit être suivi du nom de la variable de chaîne *et d'un point-virgule* (;) comme dans l'exemple ci-après:

```
10 X$ = "O3L2GBO4C;DL4CO3BAG;"
20 Y$ = "L2ADD;L1.A"
30 Z$ = "L2ADD;L1.G"
40 A$ = "XX$;XY$;XX$;XZ$;L2BGG;O4CO3L4
        BAGF# ;XY$;XX$;XZ$;"
50 PLAY "T8" + A$
```

Nous aurions pu utiliser une sous-chaîne dans l'exemple de l'air 'Clementine'. Pour le faire, changer la ligne 20 en

```
20 PLAY "T6XA$;XA$;"
```

Le point virgule *doit* suivre le signe # de la sous-routine et avec l'usage de nombres pour les notes (à la place de lettres), ce sont les seuls cas ou l'usage du point virgule est indispensable.

Il y a une option supplémentaire qui peut être utilisée avec les paramétres volume (V), octave (O), tempo (T) et longueur de la note (L). Au lieu d'un

chiffre, à la suite de la lettre, on peut utiliser l'un des suffixes suivants:

+	ajoute un à la valeur courante
-	soustrait un de la valeur courante
>	multiplie la valeur courante par deux
<	divise la valeur courante par deux

Le dernier exemple de gamme, peut être ré-écrit en utilisant cette option supplémentaire,

```
10 PLAY "O1C": FOR I = 1 TO 4: PLAY "DEFGABO + C": NEXT I
```

Où se procurer de la musique? Vous pouvez bien sûr en composer vous-même, mais pour le commun des mortels les partitions de musique pour un seul instrument: flûte, trompette ou autre seront des sources fort utiles.

Ceux qui n'ont pas de penchant naturel à faire de la musique sur leur ordinateur, ne doivent pas ignorez pour autant le commandement PLAY. Il peut être utilisé pour des effets très efficaces dans des programmes de jeux. Essayez n'importe lequel des exemples de ce châpitre avec le paramétre 'tempo' réglé à T255: vous verrez le résultat.

Pour terminer, un exemple qui exploite toute la technologie moderne de l'instruction PLAY pour interpréter, une chanson vieille de 400 ans...

```
10 A$ = "O3L2E;L1GL2AL2.BL4O+C# L2O-B;
   L1AL2F# L2.DL4EL2F# ;L1GL2EL2.EL4DL2E;
   L1F# V10L2DV8L1O-BV6L2O+E;L1GL2AL2.B
   L4O+C# L2O-B;L1AL2F# L2.DL4EL2F# ;
   L2.GL4F# L2EV8L2.D# V10L4C# V15L2D# ;
   L1.EL1EP1;"
20 B$ = "O4L1.DL2.DL4C# O-L2B;L1AL2F# L2.D
   L4EL2F# ;L1GL2EL2.EL4DL2E;L1F# L2D
   O-L1BO+L2B;0+L1DL2DL2.DL4C# 0-L2B;
   L1AL2F# L2.DL4EL2F# ;L2.GV10L4F# L2
   EV6L2.D# L4C# V4L2D# ;V15L1.EL2EP1;"
30 PLAY "T10XA$;XB$;XA$;XB$;"
```

CHAPITRE DIX

ENCORE DES GRAPHIQUES

Au châpitre 8, nous avons montré comment les commandements CIRCLE et LINE pouvaient s'utiliser pour confectionner des figures régulières comme des rectangles, des cercles, des ellipses ou des arcs. Bien que ces commandements soient très pratique, il faut parfois faire preuve d'énormément d'ingéniosité pour construire des formes irrégulières ou très détaillés en les utilisant. La meilleure façon de traiter des figures un peu complexes est de les dessiner.

Quand vous dessinez une figure sur une feuille de papier, vous commencez à un endroit donné et déplacez le crayon d'une certaine distance vers le haut, puis vers la droite, etc... Le commandement DRAW vous permet d'effectuer la même opération sur l'écran. Le commandement DRAW à la forme

DRAW *chaîne*

chaîne représente soit une constante de chaîne soit une variable de chaîne, contenant un jeu de sous-commandements pour DRAW. Le procédé est très proche de celui du commandement PLAY du chapitre précédent.

Habituellement la première chose à faire pour n'importe quel dessin est de prendre position au point de départ.

M*x, y* veut dire qu'il faut prendre position aux coordonnées contenues dans *x, y*. Par exemple, M 128,96 fixera la position de départ du dessin au centre de l'écran. Quand vous vous déplacez vers un point, il peut être utile de faire un déplacement à blanc, c'est à dire sans rien desiner; en d'autres termes, en levant le crayon pour qu'il ne marque pas. En procédant autrement il y a un risque d'avoir des traits indésirables sur le dessin. On effectue un déplacement à blanc en utilisant la lettre B; toute instruction de dessin placée après la lettre B se fera à blanc. BM 128,96 signifie: se déplacer au centre de l'écran sans rien dessiner.

Lorsque vous avez décidé de votre point de départ, vous pouvez vous déplacer vers le haut (U), vers le bas (D), vers la droite (R) ou vers la gauche (L), d'autant de points que vous le souhaitez. La séquence U20R20D20L20 tirera une ligne de 20 points vers le haut - à partir du pont de départ - puis 20 points vers la droite, puis 20 points vers le bas, puis 20 points vers la gauche. Vous aurez obtenu un cadre. Le moment est venu de vous présenter un exemple,

```
10 PMODE 3,1: PCLS: SCREEN1,1
20 DRAW"C8;"BM120,96;U26;R13;D26;L13"
80 GOTO 80
```

Le point virgule dans la chaîne est utilisé comme séparateur. L'ordinateur n'en fait rien; on peut donc ne pas le mettre, mais il nous permet d'y voir plus clair. Cet exemple, dessine un rectangle près du centre de l'écran.

En plus des lignes verticales et horizontales, on peut aussi dessiner des diagonales. Il faut alors faire usage des sous-commandements E, F, G et H: E 12 par exemple dessinera une ligne diagonale longue de 12 points et à 45 degrés de la verticale. Tous les angles sont mesurés à partir de la verticale, comme suit:

E 45 degrés F 135 degrés
G 225 degrés H 315 degrés

Ceci permet donc de tracer des diagonales dans 4 directions. Ajoutez la ligne,

```
40 DRAW "L6;U6;E6;BR13;F6;D6;L6;BU26;H6;G6"
```

au programme ci-dessus et notre rectangle va devenir une fusée! L'ordinateur se souvient de sa dernière position donc la ligne 40 continuera à dessiner à partir de cet endroit. Compulsez maintenant la chaîne de la ligne 40 pour voir comment elle est faite. La dernière position de dessin est au coin bas gauche du rectangle, et BR 13 veut dire: "se délacer de 13 points sur la droite sans rien écrire."

La fusée a été dessinée dans la couleur d'encre, faute d'indication contraire mais nous pouvons changer la couleur en utilisant C. La lettre C, sera suivie d'un nombre de 0 à 8, correspondant au code de l'une des couleurs disponibles. Utilisez l'éditeur pour changer la ligne 40 et transformer la comme suit:

```
40 DRAW "C7;L6;U6;E6;BR13;F6;D6;L6;BU26;H6;G6"
```

Et nous avons maintenant une fusée bicolore! Le dessin peut être peint exactement de la même manière que les autres formes mais comme C change la couleur d'encre il faut faire attention à ne pas tout couvrir de couleur.

Ce dessin est peut-être un peu petit, nous allons donc l'agrandir à l'échelle par l'utilisation du paramétre S. Le S veut dire 'à l'échelle' (en Anglais scale) et il permet d'agrandir ou de réduire l'échelle un dessin entier ou partie de celui-ci par unités d'un quart. Donc S 1 réduit le dessin à l'échelle d'un quart. S 2 le réduit à deux quarts (la moitié!) S 8 le fait passer à huit quarts: il double l'échelle. S peut être suivi de tout chiffre de 1 à 62. Sans indication de S, on a 4/4 soit la taille d'origine. Rajoutez à l'exemple en cours:

```
15 DRAW "S12"
```

et la fusée a maintenant trois fois sa taille d'origine

Une autre option est également disponible, c'est le paramétre d'angle A qui permet une rotation de tout ou partie du dessin. Toutes les lignes placées après A seront dessinées avec un déplacement angulaire donné par A*n*, *n* étant un nombre de 0 à 3, comme suit:

0 0 degré 1 90 degrés
2 180 degrés 3 270 degrés

Changez le programme en cours par les lignes suivantes:

```
18 FOR I = Ø TO 3: DRAW"A" + STR$(I);PCLS
5Ø FOR D = 1 TO 1ØØ: NEXT D,I
```

Notre fusée tourne mais elle change également de couleur! C'est parce que l'ordinateur ne se souvient pas seulement de la dernière position mais également du dernier réglage de couleur C et A. Ce problème peut être vite réglé en introduisant C8 au début de la chaîne de la ligne 3Ø.

La ligne 2Ø montre que tout comme avec PLAY, les chaînes utilisées par DRAW peuvent comporter des fonctions de chaînes. Et tout comme dans PLAY, vous pouvez exécuter des sous-chaînes avec la fonction X suivie d'une variable de chaîne, XA$, et d'un point-virgule (;).

```
1Ø PMODE 3,1: SCREEN 1,1: PCLS
2Ø S$ = "L8E4F4"
3Ø D$ = "AØ; XS$;A1;XS$;A2;XS$;A3;XS$;"
4Ø DRAW "S24" + D$
5Ø GOTO 5Ø
```

Un triangle est stocké dans la sous-chaîne S$, il est ensuite utilisé dans la ligne 3Ø afin de construire une forme. Il est à noter que dans ce seul cas le point virgule est indispensable, à la suite du signe de chaîne ($).

Le dernier paramétre est N qui indique à l'ordinateur de ne pas prendre en compte la dernière position du dessin, comme nouveau point de départ pour le trait suivant. NUIØL5, dessinera une ligne de 1Ø points vers le haut, *puis retournera au point de départ de la ligne* et dessinera 5 points vers la droite (une forme en L).

```
1Ø PMODE 3,1: SCREEN 1,1: PCLS
2Ø DRAW "BM128,96;NU25ND25NR25NL25;NE17NF17NG17 NH17"
3Ø GOTO 3Ø
```

L'exemple ci-dessus va dessiner des lignes vers l'extérieur à partir du centre, en revenant toujours au centre pour le point de départ de la ligne suivante.

Il peut vous arriver de vouloir dessiner un autre dessin, proche de celui que vous venez de terminer. Vous savez où le nouveau dessin doit venir se placer, par rapport au précédent mais vous souhaitez vous dispenser de claculer les nouvelles coordonnées. Ceci peut se faire avec le *mouvement relatif.* L'instruction de mouvement (M) permet cette manœuvre très facilement: tout ce que vous avez à faire est de spécifier la distance en plus ou en moins, vis à vis du point actuel: M + 5, -1Ø par exemple. Souvenez vous d'utiliser B pour éviter des lignes indésirables.

```
25 DRAW "BM - 25, - 25; U1ØR25D1ØL25"
```

Ajoutez la ligne ci-dessus au dernier exemple et un rectangle sera dessiné au dessus du dernier dessin. La dernière position était 128,96 en raison du paramétre N. Nous avons maintenant déplacé la position de 25 points à gauche et 25 points vers le haut (souvenez-vous, y = Ø est en haut de l'écran), et le dessin du rectangle commence donc au point (1Ø3,71).

Les résultats d'une commande DRAW peuvent être combinés avec des figures provenant des commandes LINE et CIRCLE mais souvenez-vous que les manipulations sur l'échelle (S), la couleur (C) ou l'angle (A) n'auront d'effets que sur la partie du dessin contenue dans DRAW.

Les commandes PSET et PRESET peuvent s'utiliser en corrélation avec PAINT pour caser des détails supplémentaires et de la couleur. Soyez toutefois prudent avec PAINT car des changements dans la partie DRAW du dessin peuvent mettre la couleur aux mauvais endroits.

DEPLACEMENTS SUR L'ECRAN

Ayant réalisé votre chef d'œuvre en utilisant LINE, CIRCLE ou DRAW, vous voulez maintenant le déplacer sur l'écran. Nous pourrions bien entendu effectuer cette opération en effaçant le dessin et en le redessinant chaque fois, comme nous l'avons fait précédemment. Mais cela prendrait pas mal de temps pour peu que le dessin soit relativement complexe; les deux prochaines instructions vont régler cette question. Tout ce que vous avez à faire est d'utiliser GET pour prendre une copie de votre dessin, et PUT, pour mettre cette copie ailleurs. L'instruction GET vous permet de prendre une copie d'une surface rectangulaire de l'écran en la mémorisant dans une variable en tableau, laquelle réaffichera ultérieurement l'image sur l'écran par l'instruction PUT. (GET = prenez, PUT = mettez)

GET (x 1, y 1) - (x2,y2), *nom du tableau,* G

x1, y1 et x2, y2 représentent les coordonnées du coin haut à gauche et du coin bas à droite de la surface rectangulaire contenant le dessin que vous voulez stocker. Le nom du tableau est le *nom d'un tableau* qui doit avoir été *déclaré préalablement.* (Si vous ne vous souvenez plus trop bien des tableaux, reportez vous au chapitre six). La taille du tableau doit correspondre à la taille du rectangle que l'on veut y mettre. La première dimension a donner au tableau correspond à la *largueur* du rectangle (x2 - x1), la seconde dimension correspond à la *hauteur* (y2 - y1). Le dernier paramétre, G, est facultatif, il conditionne la quantité de détails que l'on stocke. En PMODE Ø, 1 ou 3, G est indispensable sans quoi les mouvements horizontaux via PUT peuvent être inexacts. Nous allons utiliser notre fusée de tout à l'heure pour étudier ce procédé. D'abord nous allons calculer la dimension du tableau nécessaire à caser cette fusée. Le dessin débute à 12Ø,96, l'empennage gauche est six points plus loin, la fusée et l'empennage droit sont à 13 + 6, donc le dessin fait 25 points de large, de 114,96 à 139, 96. La hauteur est de 26, plus le nez en forme de cône qui fait 5 à sa pointe; donc

la hauteur est de 31. Arrondissons dans les deux sens et nous voici avec un tableau rectangulaire de 30 x 40 dont le coin haut gauche est à 112,60 et le coin bas droit à 142,100.

```
10 PMODE 3,1: SCREEN 1,1: PCLS: DIM R(29,39)
20 R$ = "C8BM120,96;U26R13D26L13;C7L6U6E6BR13
   F6D6L6BU26H6G6"
30 DRAW R$
40 GET (112,60) - (142,100),R,G

100 GOTO 100
```

L'exemple ci-dessus dessine notre fusée comme avant (tout d'un trait cette fois) et la stocke dans le tableau R. Vous voyez que nous ne dimensionons que 29 x 39, car les tableaux commencent à zéro.

Ayant stocké le dessin il nous faut maintenant le remettre à l'écran, par l'instruction PUT. PUT revêt une forme similaire à GET, à savoir:

PUT *(x1, y1) - x2, y2, nom du tableau, action*

x1, y1 et x2, y2 sont comme avant les coordonnées du rectangle, mais cette fois il s'agit de la zone où l'on entend mettre le dessin et pas la zone d'ou il a été pris. Le *nom du tableau* c'est le nom de la variable en tableau qui contient le dessin stocké. Le paramétre *action* est facultatif et ne s'utilise que lorsque le paramétre G a été utilisé dans GET. L'*action* doit être l'un des mots suivants et cela conditionne la façon dont le résultat est affiché à sa nouvelle position.

PSET — Placer chaque point tel qu'il est dans le tableau d'origine. En d'autres termes, afficher le dessin tel qu'il est.

PRESET — Eteindre chaque point qui est allumé dans le tableau d'origine. Cela va soit annuler le dessin, soit inverser les couleurs, le résultat dépendant des couleurs de fond et d'encre.

AND — Compare les points du tableau d'origine avec les points de sa copie. Si les points sont allumés tous les deux on a un point allumé. Si l'un des deux points est éteint, le point est éteint. Ceci veut dire que si un dessin est superposé sur un autre seuls les points qui coïncident seront visibles.

OR — Compare les points comme précédemment, Si un point du tableau d'origine est allumé ou un point de sa copie est allumé le point sera allumé. Ceci a pour effet de superposer un dessin sur un autre.

NOT — Ceci inverse chaque point de la zone d'affichage ce qui a pour résultat de présenter le dessin sur un fond ayant la couleur d'encre.

GET

L'instruction GET ne peut s'utiliser que dans les modes de haute résolution graphique. GET copie le contenu graphique d'une zone écran rectangulaire spécifiée et le stocke dans un tableau. Le tableau doit avoir été préalablement établi et ce a la bonne dimension.

GET (x1, y1) - (x2, y2), *nom du tableau,* G

x1, y1 et x2, y2 correspondent aux coordonnées des coins supérieur gauche et inférieur droit du rectangle sur l'écran.

nom du tableau est le nom du tableau établi préalablement et ayant les dimensions voulues pour contenir le rectangle.

G défini la quantité de détail à stocker. Ce paramétre est facultatif.

Voir le descriptif de PUT pour l'exemple d'usage.

Vous devez toujours utiliser PUT dans le même mode que GET, autrement vous risquez d'obtenir des résultats pour le moins inattendus. Nous allons maintenant revenir à notre exemple et installer par PUT, notre fusée à un autre endroit. Ajoutez ces lignes supplémentaires.

```
50 Y = 150:FOR X = 10 TO 210 STEP 40
60 PUT(X, Y) - (X + 30, Y + 40), R, PSET

80 NEXT X
```

Vous devez maintenant avoir une rangée de fusées, allignées au bas de l'écran. Pour faire bouger une fusée le long du bas de l'écran, ajoutez la ligne suivante

```
70 FOR D = 1 TO 200: NEXT D: PCLS
```

Et en utilisant les manettes de jeu, en corrélation avec GET et PUT vous pouvez déplacer votre dessin à volonté.

```
10 PMODE 3,1: SCREEN 1,1: PCLS: DIM S(48,48)
20 DRAW "BM24,12,S8;C4;E2H2D4D8R8H8G8R2
   NR6F3R6E3"
30 GET (0,0) - (48,48),S
40 A$ = INKEY$: A$ = ""THEN 40
50 PCLS;A = JOYSTK(0)*3.25: B = JOYSTK(1)*2.25
60 PUT (A,B) - (A + 48, B + 48), S: GOTO 50
```

L'exemple ci-dessus exécute un dessin dans le coin en haut à gauche de l'écran. Quand vous appuyez sur n'importe quelle touche, l'écran est effacé et le dessin peut être déplacé en manœuvrant la manete de jeu gauche.

Ceci termine nos explications concernant les capacités graphiques qui sont a votre disposition. Les exemples que nous avons présentés sont limités aux nécessités d'explication de cet ouvrage et ne représentent en aucune façon la limite de ce qui est faisable. Dessiner des images ou des graphiques en tous genres n'est pas difficile à condition de bien s'organiser à l'avance et en dessinant préalablement les formes à obtenir sur la grille des graphiques plutôt que directement sur l'écran.

DRAW

L'instruction DRAW dessine une ligne, ou une série de lignes, selon les instructions contenues dans une chaîne. Elle ne fonctionne que dans les modes de haute résolution graphique.

DRAW *chaîne*

La *chaîne* peut être une constante ou une variable de chaîne et elle peut contenir n'importe lesquelles des sous-instructions suivantes.

M*x; y*	Se rendre à la position de dessin en x, y
U*n*	Vers le haut de *n* points (U pour up)
D*n*	Vers le bas de *n* points (D pour down)
L*n*	Vers la gauche de *n* points (L pour left)
R*n*	Vers la droite de *n* points (R pour right)
E*n*	A 45 degrés de *n* points
F*n*	A 135 degrés de *n* points
G*n*	A 225 degrés de *n* points
H*n*	A 315 degrés de *n* points
X	Exécution d'une sous-chaîne et retour
C	Etablir une couleur définie pour la ligne
A*k*	Déplacer la prochaine ligne d'un angle:
	k = Ø Ø degrés — *k* = 1 90 degrés
	k = 2 180 degrés — *k* = 3 270 degrés
S*k*	Dessin à l'échelle par unités d'1/4, *k* de 1 à 62
	k = 1: échelle d'un quart, *k* = 8: échelle du double
	Sans indication: *k* = 4 (quatre quarts donc 1/1)
N	Ne pas retenir la dernière position de dessin
*B	Ne rien dessiner, se déplacer seulement

Les mouvements relatifs peuvent être précisés par ces 8 paramétres dans la forme suivante

M *x décaler, y décaler*

x décaler et *y décaler* étant les nombres qui spécifient la distance qui sépare la position actuelle de la nouvelle à obtenir. Les deux chiffres *doivent* être précédés soit d'un signe plus (+), soit d'un signe moins (-).

Un exemple d'usage de DRAW est donné au descriptif de l'instruction PUT.

PUT

L'instruction PUT s'utilise pour afficher le contenu d'un tableau graphique ayant été constitué avec l'instruction GET.

PUT doit être utilisé dans le même mode que celui utilisé quand le tableau a été créé, autrement les résultats sont imprévisibles.

PUT *(x1, y1)* - (x2, y2), nom du tableau, action

x1, y1 sont les coordonnées du coin haut gauche de la surface d'affichage *x2, y2* correspond au coin bas droit. Le *nom du tableau* se réfère au tableau défini à l'avance et contenant les éléments graphiques. Le paramétre *action* est facultatif mais *doit* être utilisé dans le cas où le paramétre G était présent dans l'instruction GET.

PSET	Allume les points de destination comme ceux du tableau d'origine
PRESET	Eteint chaque point qui était allumé dans le tableau d'origine
AND	Compare les points du tableau d'origine avec les points de destination. Si allumés tous les deux, on a un point allumé, si non un point éteint
OR	Compare les points comme précédemment. Si l'un des deux points est allumé, le point sera allumé à l'écran
NOT	Inverse l'état de chaque point dans la zone de destination quel que soit la situation dans le tableau d'origine

La zone d'affichage de destination doit être de la même taille que la zone de départ, autrement l'écran affichera des incohérances.

```
10 PCLEAR 4; PMODE 3,1: PCLS: SCREEN 1,1: DIM W(30,30)
20 DRAW "BM10,12; S 8; R1U3R1D2R2U2R1D3R1D2L1
   D2R1D1L2U3L4D3L2U1R1U2L1U2R1U2BR1BD1D2R2
   U2NL2R2D2L2U2"
30 PAINT (11,13), 6,5: GET (0,0) - (30,30), W
40 A$ = INKEY$: IF A$ = "" THEN 40
50 PCLS: FOR C = 0 TO 100 STEP 20
60 FOR A = 0 TO 200 STEP 20
70 PUT (A,C) - (30 + A,30 + C), W
80 PUT (A,C + 30) - (30 + A,60 + C), W
90 PUT (A,C + 60) - (30 + A,90 + C), W
100 PLAY "T255; ABFGBA": PCLS: NEXT A, C
```

CHAPITRE ONZE

LA TOUCHE FINALE

COMPLEMENTS A PRINT

Bien que les possibilités offertes par PRINT et PRINT @ vous donnent un large contrôle sur la manière d'afficher les résultats, il existe encore une possibilité supplémentaire: le commandement PRINT USING. Ce commandement permet de spécifier exactement comment chaque ligne doit apparaître à l'écran. C'est tout spécialement utile pour produire des tableaux ou des travaux comptables.

PRINT USING *format; rôle des sorties*

Le *format* est une constante de chaîne ou une variable de chaîne contenant les instructions sur la manière dont les *sorties* doivent apparaître à l'écran. Le *rôle des sorties* c'est l'ensemble des éléments qui doivent "sortir" de l'ordinateur vers l'écran, donc les constantes et variables habituelles des instructions PRINT.

Les instructions *format* sont constituées par des 'spécificateurs de champ' qui sont des caractères spéciaux indiquant à l'ordinateur le nombre de positions de print qu'il doit utiliser pour afficher un nombre ou une chaîne.

Le spécificateur #

Ce caractère s'utilise pour indiquer la position de chaque chiffre dans un nombre.

```
PRINT USING "# # #.# #"; A
```

L'instruction ci-dessus va imprimer le contenu de A avec 3 chiffres avant la décimale et 2 après. S'il y a plus de deux chiffres après la décimale, le nombre sera arrondi en conséquence. Les positions inutilisées du côté gauche de la décimale seront affichées comme espaces; si le nombre est trop grand pour entrer dans le champ qui lui est imparti l'ordinateur esayera de s'en tirer pour le mieux et imprimera le nombre avec un signe % devant pour informer de l'anomalie.

```
PRINT USING "# # #.# #"; 13.4695
∇ 13.47
PRINT USING "# # #.# #"; 1492.878
% 1492.88
PRINT USING "# # #.# #"; 146
146.ØØ
PRINT USING "# # #"; 18.76
∇ 19
```

Le spécificateur ★ (astérisque)

La grande majorité des comptables n'aiment pas écrire des nombres avec des espaces devant, surtout sur les chèques. Si vous placez *deux* astérisques au début de votre champ numérique, les positions inutilisées seront remplies avec des astérisques.

```
PRINT USING "★★# # #.# #"; 1.492
★★★★1.49
```

Le spécificateur +

Quand on place le signe + au début d'un champ numérique cela oblige le signe du nombre à apparaître.

```
PRINT USING "+# # #.# #"; 14.7
∇+ 14.7Ø
PRINT USING "+★★# # #.# #"; -7,4
★★★★-7.4Ø
```

Si le signe plus (+) est placé après le champ numérique cela obligera le signe à apparaître *après* le nombre.

```
PRINT USING "# # #.# #+"; 27.86
∇ 27.86 +
PRINT USING "# # #.# #+";-1.6
∇∇ 1.6Ø-
```

Si le signe moins est placé après un nombre, les nombres négatifs apparaîtront avec le signe moins à leur suite, tandis que les nombres positifs seront suivis d'un espace.

```
print using "★★# # #.# #-";-12.418
★★★12.42-
PRINT USING "# # #.# #-"; 47.25
∇ 47.25 ∇
```

Le spécificateur ↑↑↑↑

Ceci permet aux nombres d'être exprimés en notation exponentielle. Les quatre fléches doivent être placées derrière le champ numérique.

```
PRINT USING "# # #.# # ↑↑↑↑"; 123456
∇ 1.2346E+Ø5
```

Le spécificateur !

Ce spécificateur s'utilise avec les chaînes. Il n'imprimera que le premier caractère de chaîne qui se présentera.

```
PRINT USING "!"; "CREDIT"
C
```

Le spécificateur %

Pour imprimer des chaînes, il est nécessaire de spécifier la largeur du champ dans lequel elle doivent apparaître. Ceci se fait au moyen de deux signes % séparés par un certain nombre d'espaces. La largeur du champ attribué sera égale au nombre d'espaces plus deux. Si la chaîne est plus longue que le champ disponible, seuls les *n* premiers caractères seront affichés. *n* étant l'équivalent de la longueur du champ.

```
PRINT USING "% ∇ ∇ ∇ ∇ ∇ %"; "DEBIT"
DEBIT ∇ ∇
PRINT USING "% ∇ %"; "SOLDE"
SOL
```

Le spécificateur $

Le signe dollar est utilisé pour représenter de l'argent; placé devant une valeur numérique il introduira un signe dollar dans la sortie.

```
PRINT USING " $ # # # . # #"; 2,87
$ ∇ ∇ 2.87
```

Si l'on emploie deux signes dollar ($ $) le signe dollar apparaîtra juste avant le nombre.

```
PRINT USING " $ # # # . # #"; 2.87
∇ ∇ ∇ $ 2.87
```

Utilisé en même temps que deux astérisques, le signe dollar produira le résultat suivant,

```
PRINT USING "★ ★ $ # . # #"; 14.9
★ $ 14.90
```

Les espaces et autres caractères figurant dans la chaîne de format figureront également dans la sortie. (pour rappel, "sortie", en anglais OUTPUT, c'est ce que l'ordinateur "sort", vers l'écran, l'imprimante ou autre périphérique)

```
PRINT USING "MEAN ∇ ∇ # # . # # ∇ ∇ ∇ TOTAL ∇ ∇ # # # . # #";
3.4,40.8
MEAN ∇ ∇ ∇ 3.40 ∇ ∇ ∇ TOTAL ∇ ∇ ∇ 40.80"
```

Si *le rôle des sorties* contient plus d'éléments que les places disponibles dans le *format*, le *format* est recommencé à son début

```
PRINT USING " # # # . # # ∇ ∇ ∇"; 7.84,142.5,.234
∇ ∇ 7.84 ∇ ∇ ∇ 142.50 ∇ ∇ ∇ ∇ ∇ 0.23
```

Bien entendu, si l'on fait usage de l'écran, la longueur d'une ligne produite par PRINT USING reste limitée à 32 positions d'écriture. Tout ce qui dépasse sera reporté à la ligne suivante. Pour ceux qui disposent d'une imprimante la longueur de ligne utilisable est bien plus grande. Au moins 80 caractères par ligne sur la plupart des appareils. Pour l'imprimante, la forme est:

PRINT # - 2, USING *format, rôle des sorties*

format et *rôle des sorties* étant les mêmes que précédemment. Le - 2 veut dire: "sortie vers l'imprimante, pas sur l'écran". Si vous voulez afficher à l'écran et imprimer à la fois, il faudra formuler deux instructions PRINT USING.

ENTREES ET SORTIES SUR CASSETTE

Jusqu'à présent toutes les données nécessaires à nos programmes ont été rentrées par le clavier ou ont été lues par READ dans des lignes de DATA. Toutes les sorties ont été faites à l'écran. Mais vous pouvez aussi utiliser votre magnétophone à cassettes pour stocker des données, tout comme vous pouvez stockez les programmes. Les données ainsi conservées peuvent alors être relues à une date ultérieure. Le magnétophone est raccordé et installé exactement de la même façon que pour stocker et restituer des programmes. Il va seulement falloir dire à l'ordinateur qu'il doit travailler sur des fichiers de données et ceci se fera au moyen de la commande OPEN.

OPEN *a*, # - 1, nom du fichier

Le *a* doit être soit "O" soit "I" (O pour output, en anglais, sortie; I pour Input, en anglais, entrée) "O" s'utilise pour les sorties, cela veut dire que les données *sortent* de l'ordinateur vers la cassette. "I" s'utilise pour les entrées, les données *entrent dans* l'ordinateur, venant de la cassette.

Le # - 1 indique à l'ordinateur que vous vous sevez d'un magnétophone à cassettes. Le *nom du fichier* c'est le nom que vous voulez donner à ce fichier (tout nom, commençant par une lettre et composé de 8 caractères au maximum, est valable).

La tâche suivante est d'enregistrer les données sur la bande magnétique. Cela se fera au moyen d'une instruction PRINT, de la forme suivante.

PRINT # - 1, *rôle des sorties*

La seule différence avec le PRINT habituel est la précision, qui dit à l'ordinateur de communiquer le contenu de la sortie au magnétophone et non à l'écran.

Quand vous avez fini d'enregistrer les données, il *faut* fermer le fichier au moyen du commandement CLOSE.

CLOSE # - 1

PRINT USING

Le commandement PRINT USING permet d'élargir le contrôle sur la manière dont sont disposés les résultats sur l'écran, (ou sur l'imprimante).

PRINT USING: *format: rôle des sorties*

format est une constante de chaîne ou une variable contenant les 'spécificateurs de champ' indiquant comment le *rôle des sorties* doit se disposer. Le rôle des sorties est une liste de variables numériques ou de chaîne, ou aussi de constantes, séparées par des virgules.

Les 'spécificateurs de champ' sont les suivants:

Caractères	Action	Exemple	Résultat
#	Formattage des nombres	"# # # #"; 147.2	**147**
.	Point décimal	"# # . # #"; 34.678	**34.68**
,	Afficher une virgule à la gauche de chaque troisième caractère	"######.#";123456	
★★	Remplir les espaces de tête avec des astérisques	"★★# # #.# # #"; 1.47	★★★★ 1.47Ø
$	Place le signe dollar en tête du nombre	"$# # # #.# #"; 12.689	$ ∇ ∇ 12.69
★★$	Signe dollar, flottant	"★★$####.##"; 12.689	★★★★ $12.69
+	En première position, produit l'affichage du signe en tête. En dernière position, le signe apparaît après le nombre	"# #.# # +"; -12.689	12.69-
↑↑↑↑	Imprime en notation exponentielle	"# #.# # ↑↑↑↑"; 12.689	1.27E+Ø1
!	N'imprime que le premier caractère d'une chaîne	"!"; "CREDIT"	C
% espaces %	Délimitation du champ, pour les chaînes. La longueur du champ est égale au nombre d'espaces plus 2	"% ∇ ∇ ∇ ∇ ∇ %"; "SOLDE"	SOLDE

Chaque 'spécificateur de champ' peut être séparé par des espaces qui apparaîtront comme espaces à l'affichage.

```
10 CLS: INPUT "ENTRER LE DERNIER SOLDE"; B:C = 0: D = 0
20 CLS: T$ = " ∇∇∇∇% ∇∇∇% ∇∇∇%∇∇∇%∇∇∇%∇∇∇∇%"
30 L$ = "∇∇∇### .## ∇∇### .## ∇∇### .## +"
40 PRINT USING T$; "DEBIT", "CREDIT", "SOLDE"
50 PRINT USING L$; D, C, B
60 OPEN "I", # - 1, "CHEQ"
70 IF EOF (- 1) THEN 110
80 INPUT # - 1, A: D = 0: C =0
90 IF A < 0 THEN D = ABS (A) ELSE C = A
100 B = B + C - D: PRINT USING L$; D, C, B: GOTO 70
110 CLOSE # - 1: END
```

SORTIE SUR IMPRIMANTE

Pour ceux qui disposent d'une imprimante raccordée à la prise I/0, voici les variantes de certaines instructions qui permettent de diriger la sortie vers l'imprimante et non vers l'écran.

PRINT # - 2, *rôle des sorties*
PRINT # - 2, USING *format; rôle des sorties*

Le *format* et le *rôle des sorties* sont les mêmes que pour l'affichage à l'écran.

POS (- 2) restituera la position courante de la tête d'impression.

LLIST enverra la liste d'un programme, directement à l'imprimante. Son usage est exactement le même que celui de LIST.

En utilisant la combinaison de [SHIFT] [0] vous obtiendrez l'écriture en "minuscules" sur l'imprimante. L'option "minuscules" ne peut s'employer que dans les chaînes et les REM, toutes les instructions à l'ordinateur devant être faites en "majuscules".

Pour relire les données à partir du magnétophone, vous procédez de la même façon, à la différence que cette fois le fichier sera ouvert par INPUT et au lieu de PRINT vous utiliserez,

INPUT # - 1, *rôle des entrées*

Le commandement CLOSE est le même dans les deux cas. (To close, fermer). Les exemples ci-dessous vont vous montrer pratiquement comment faire. Premièrement, mettre en place le magnétophone et mettre la bande à l'endroit où vous voulez situer le fichier. (employez SKIPF). Ensuite, mettre l'appareil en position d'enregistrement en vous servant des touches PLAY et RECORD ensemble.

```
10 CLS: PRINT "CREATE PHONE LIST"
20 OPEN "O", # - 1, "PHONE": PRINT "ENTER XXX, XXX TO END"
30 PRINT @ 128, ""; : INPUT "NAME ∇";N$
40 INPUT "TELEPHONE NO."; T$:IF
N$ = "XXX" OR T$ = "XXX" THEN 60
50 PRINT # - 1, N$, T$,: PRINT @ 128, "": GOTO 30
60 CLOSE # - 1: END
```

Quand vous faites tourner le programme, la bande avancera et enregistrera le fichier. Chaque fois que vous écrivez un nom et un numéro de téléphone, il sera communiqué à la bande magnétique (le PRINT @ 128 figurant à la ligne 30 ne sert qu'à dégager cette ligne sur l'écran). Ceci va continuer jusqu'à ce que vous entriez XXX, XXX, le fichier sera alors fermé et le programme se terminera.

Maintenant il ne nous reste plus qu'à relire ce fichier. La différence principale entre la procédure d'entrée et la procédure de sortie, c'est qu'à l'entrée il ne faut pas tenter de lire la bande au delà de la fin du fichier. Ce problème est réglé par l'instruction supplémentaire EOF. Cette instruction vérifie si la fin du fichier que vous lisez est atteinte ou non.

Rembobinez la bande à son début et cette fois, n'appuyez que sur le bouton PLAY du magnétophone.

```
10 CLS: PRINT "READ PHONE LIST"
20 OPEN "I", # - 1, "PHONE"
30 PRINT "NAME", "NUMBER"
40 IF EOF (- 1) THEN 60
50 INPUT # - 1, A$, B$: PRINT A$, B$: GOTO 40
60 CLOSE # - 1: END
```

Quand vous faites tourner le programme, la bande du magnétophone va partir et l'ordinateur va rechercher le fichier "TELEPHONE". (Il se peut que vous ayiez à attendre un peu, si la bande est à son début et que le fichier soit plus loin). Ensuite il lira les noms et numéros de téléphone et les affichera sur l'écran. Il est à noter qu'il n'est pas indispensable d'utiliser les mêmes noms de variables que ceux utilisés lors de l'écriture des données. Toutefois *vous devez* utiliser le même type de variable (numérique ou chaîne). Quand la fin du fichier est atteinte, le fichier est fermé et le programme se termine.

L'instruction EOF doit apparaître avant l'INPUT # - 1, autrement vous recevrez un message d'erreur IE, qui veut dire: "essai de lire un fichier au delà de sa fin.

N'omettez pas de fermer le fichier par CLOSE car sinon vous risquez des problèmes, spécialement si cet oubli intervient à l'enregistrement d'un fichier.

ENCORE UN MOT

Vous êtes maintenant en bonne voie, de devenir expert en programmation BASIC et commencez peut être a songer à l'étape suivante: le language machine. Ceci est la langue maternelle de l'ordinateur et jusqu'à présent vous ne lui avez parlé que par le truchement d'un interprète parlant BASIC.

Pourquoi s'en préoccuper? Parce que le language machine fait travailler l'ordinateur beaucoup plus vite, occupe en général moins de place en mémoire et de plus permet de faire des choses qui ne sont pas réalisables en Basic.

La meilleure chose a faire c'est de commencer par se procurer un manuel traitant du language machine et se référant spécialement aux microprocesseurs de série 6800. Par exemple

Basic Microprocessors and the 6800, par *RON BISHOP,* édité par *Hayden Book C° Inc.*

Une fois que vous aurez acquis les connaissances de base en language machine, vous pourrez utiliser ceraines instructions que DRAGON 32 met à votre disposition pour faire usage de routines en language machine. Vous trouverez ci-dessous un bref descriptif de ces possibilités.

USR*n* vous permet d'appeler jusqu'à dix (0 à 9) routines en language machine. Sa forme d'utilisation est:

USR*n (argument)*

l'*argument* est une chaîne ou une expresion numérique.

Quand le programme rencontre un appel à USR, le contrôle est transféré à l'adresse donnée dans l'introduction DEF USR*n*. L'adresse spécifie ou se trouve le point d'entrée dans la routine en language machine.

DEF USR*n* s'utilise pour définir l'adresse d'une fonction USR*n*. Sa forme est la suivante,

DEF USR*n* = *adresse*

n est compris entre 0 et 9 et correspond au *n* figurant dans USR. L'*adresse* doit être comprise entre 0 et 65535 et doit contenir l'adresse d'entrée pour USR*n*

CLEAR *s, h* L'instruction CLEAR doit s'utiliser pour réserver de la mémoire à l'usage des fonctions USR. Le *s* se référe comme pécédemment à la quantité de mémoire réserver.

Le *h* est la plus haute adresse laissée à la disposition de BASIC. A partir de *h* + 1 la mémoire est maintenant uniquement réservée aux routines en language machine.

POKE. Le commandement POKE s'utilise pour placer une valeur dans un endroit bien défini de la mémoire.

POKE *adresse, valeur*

l'*adresse* est comme précédemment et la *valeur* doit être comprise entre Ø et 255.

VARPTR. (contraction de *variable* Pointer) Un index relatif a une variable BASIC peut être utilisé comme argument par une fonction USR. Ceci peut permettre à une fonction USR d'accéder au contenu d'un tableau.

VARPTR *(nom de variable)*

Le *nom de la variable* est la variable BASIC à laquelle vous voulez accéder. VARPTR s'utilise comme partie d'un argument d'USR de la manière suivante

USRØ (VARPTR (X))

CSAVEM *nom, départ, fin, entrée*
CLOADM *nom, transfert*

nom représente le nom du fichier sur la bande, *départ* c'est l'adresse de départ de la routine en mémoire, *fin* c'est la dernière adresse occupée par la routine et *entrée* est le point d'entrée dans le programme. *transfert* dans CLOADM permet de recharger la routine dans la mémoire à une adresse donnée par *départ* + *transfert*.

Une fois chargé, le contrôle peut être transféré à la routine par l'instruction EXEC

EXEC *adresse*

L'*adresse* est le point de départ de la routine; si *adresse* n'est pas cité l'ordinateur utilisera le *départ* de la dernière instruction CLOAD.

ANNEXE A

CODE DE CARACTERES A.S.C.I.I.
(Décimal)

TOUCHE	Sans SHIFT	Avec SHIFT
[BREAK]	3	3
[CLEAR]	12	92
[ENTER]	13	13
[SPACEBAR]	32	32
!	33	-
"	34	-
≠	35	-
$	36	-
%	37	-
&	38	-
'	39	-
(	40	-
)	41	-
★	42	-
+	43	-
,	44	-
-	45	-
.	46	-
/	47	-
Ø	48	18
1	49	-
2	50	-
3	51	-
4	52	-
5	53	-
6	54	-
7	55	-
8	56	-
9	57	-
:	58	-
;	59	-
<	60	-
=	61	-
>	62	-

TOUCHE	Sans SHIFT	Avec SHIFT
?	63	-
@	64	19
A	97	65
B	98	66
C	99	67
D	100	68
E	101	69
F	102	70
G	103	71
H	104	72
I	105	73
J	106	74
K	107	75
L	108	76
M	109	77
N	110	78
O	111	79
P	112	80
Q	113	81
R	114	82
S	115	83
T	116	84
U	117	85
V	118	86
W	119	87
X	120	88
Y	121	89
Z	122	90
↑	94	95
I	10	91
←	8	21
→	9	93

Les caractères obtenus sans SHIFT, deviennent des minuscules en faisant [SHIFT] [0] (voir page 1)

Les caractères suivants s'obtiennent comme suit, au moyen de la fonction CHR$:

[CHR$ (123)
/ CHR$ (124)
] CHR$ (125)
↑ CHR$ (126)
← CHR$ (127)

Les caratères de 128 à 255 sont des caractères graphiques, comme ci-après.

CARACTERES GRAPHIQUES

Pour obtenir les caractères ci-dessus, utiliser CHR$ avec code approprié. Pour les avoir dans d'autres couleurs ajouter le nombre voulu au numéro de code. Par exemple, PRINT CHR$ (142 + 112) donne le caractère 142, mais le vert est remplacé par l'orange.

+ 16 jaune
+ 32 bleu
+ 48 rouge
+ 64 blanc
+ 80 cyan (bleu de Prusse)
+ 96 magenta
+ 112 orange

ANNEXE B

ECRANS PRINT ET GRAPHIQUES

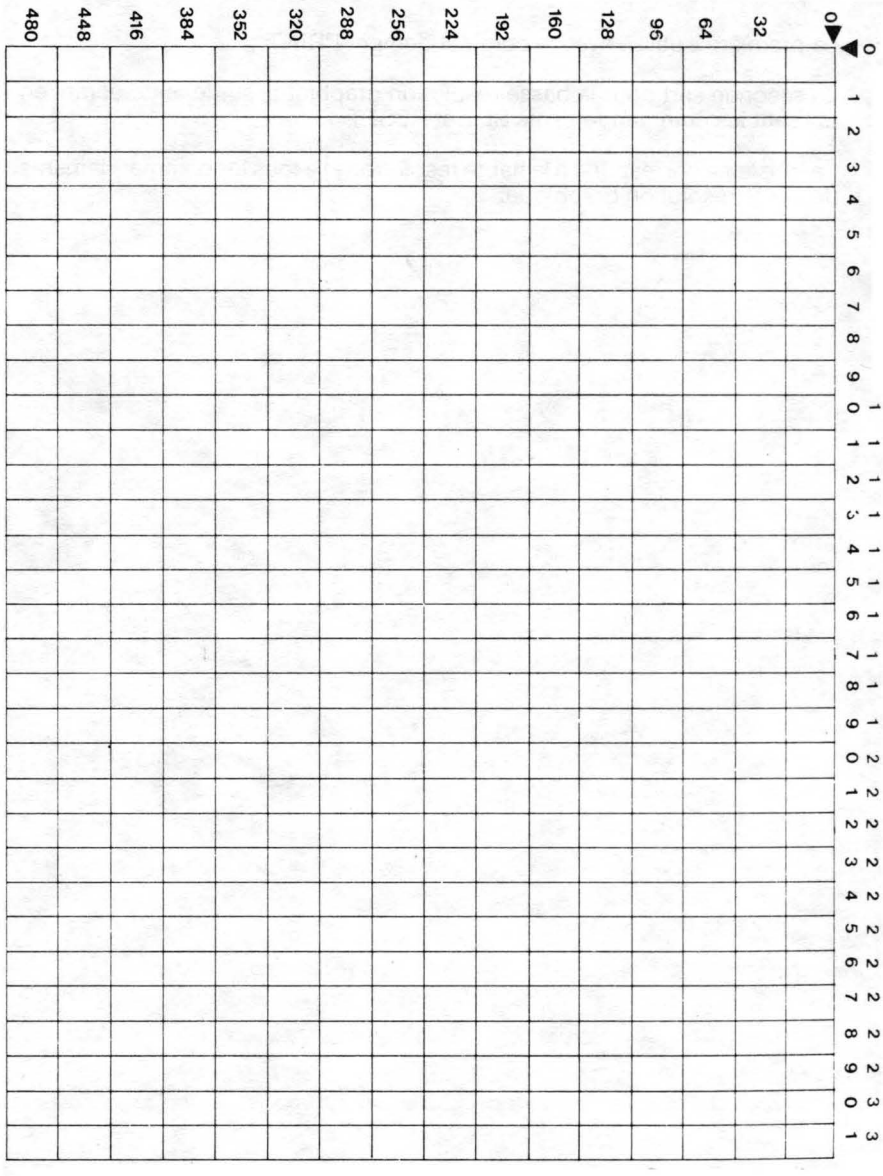
Les grilles suivantes sont utiles pour concevoir des dessins et des dispositions d'affichage.

La première s'utilise avec le commandement PRINT @

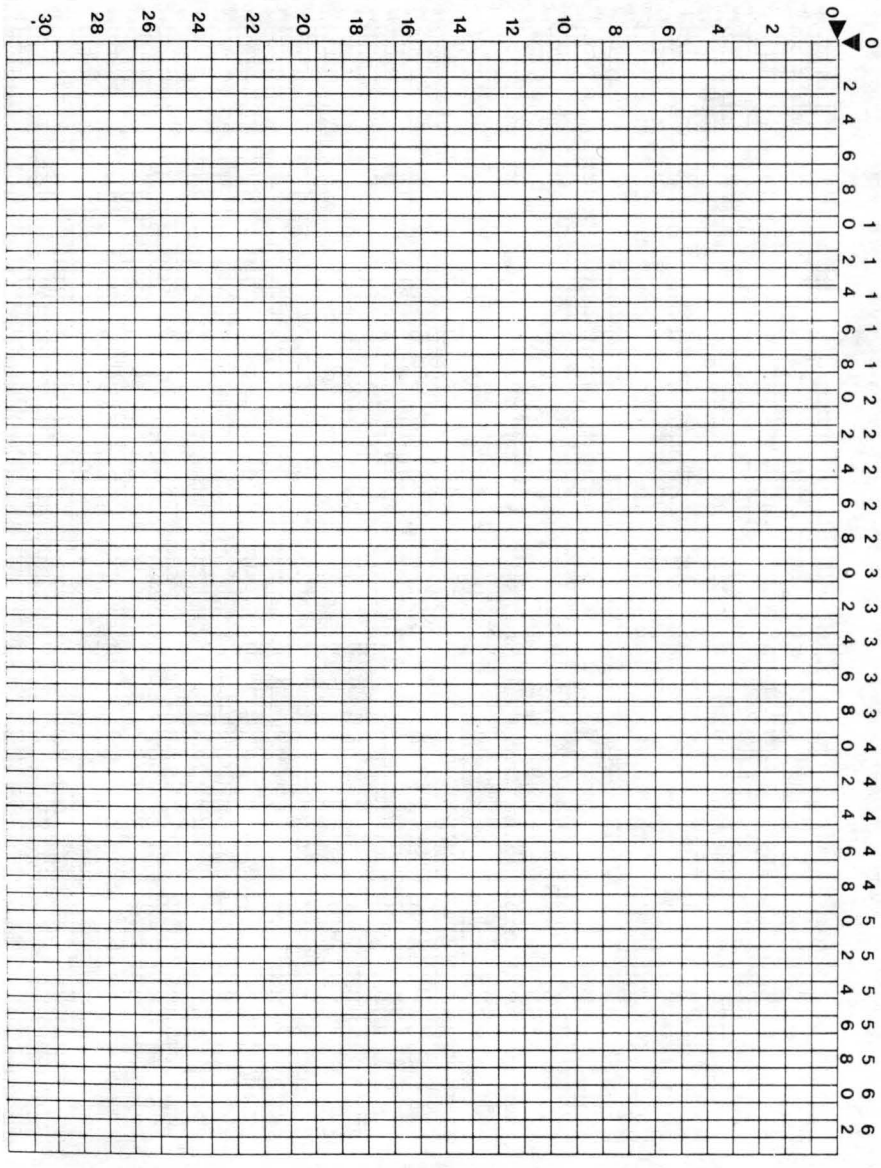
La seconde sert pour la basse résolution graphique, sur le texte-écran, en utilisant les commandements SET et RESET.

La troisième correspond a la haute résolution et a tous les commandements de haute résolution graphique.

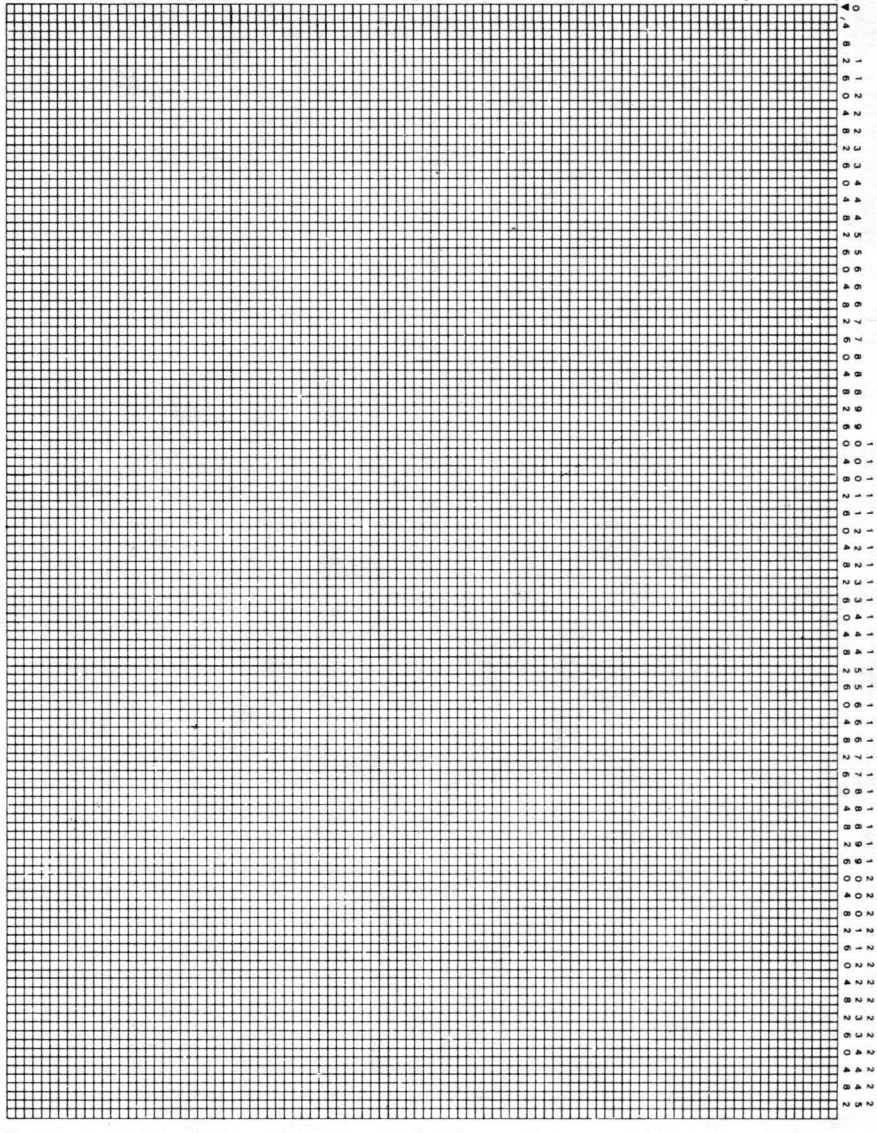
Grille Print @



Grille Basse Résolution



Grille Haute Résolution



ANNEXE C

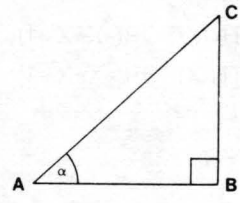
CODE D'ERREURS

CODE	EXPLICATION
/Ø	Division par zéro. Impossible
AO	Tentative d'ouverture d'un fichier qui est déjà ouvert. Se produit en général après utilisation de RESET pour arrêter un programme utilisant des fichiers. Couper le courant et rallumer.
BS	Indice érroné. En général provient de ce que l'indice est plus grand que la dimension donnée au tableau.
CN	Impossible de continuer. Tentative d'utiliser CONT alors que la fin, END, du programme a été atteinte.
DD	Tentative de redimensionner un tableau. Un tableau ne peut être dimensionné qu'une seule fois dans un programme.
DS	Instruction directe. Se produit en général si l'on tente de CLOAD un fichier de données.
FC	Appel d'une fonction dans des conditions non autorisées. En général le paramètre se situe hors limites ou encore, la variable n'est pas du type correct.
FD	Données ne correspondant pas au fichier. Causé par tentative de lire des données de chaîne dans des variables de chaîne lors de l'utilisation de fichiers sur cassette.
FM	Erreur de mode dans l'utilisation d'un fichier. Tentative d'entrer dans un fichier ouvert (OPEN) pour la sortie ou l'inverse. Voir page 130: "I" et "O"
ID	Utilisation illégale d'une instruction. Tentative d'utiliser de manière directe une instruction qui ne peut s'utiliser que dans un programme, comme INPUT, DEF FN.
IE	Tentative d'entrer dans un fichier alors qu'on en a atteint le fin. Vérifier la chose par IF EOF (- 1). Voir page 133.
IO	Erreur entrée/sortie. Magnétophone mal réglé ou bande magnétique défectueuse.
LS	Chaîne trop longue. Maximum 255 cractères.
NF	NEXT utilisé sans être précédé de FOR. Se produit en général quand des instructions NEXT sont inversées, dans des boucles imbriquées.
NO	Fichier pas ouvert. Entrées et sorties dans un fichier ne sont possible qu'après son ouverture par OPEN.
OD	Il n'y a plus de données disponibles. Une instruction READ a déjà lu toutes les données contenues dans DATA.
OM	Il n'y a plus de mémoire disponible. Toute la mémoire a déjà été utilisée ou réservée.
OS	Il n'y a plus d'espace disponible pour des chaînes. Utiliser CLEAR pour créer plus d'espace, s'il y en a de disponible en mémoire.

CODE	EXPLICATION
OV	Overflow (débordement). Le nombre est trop grand pour être traité par l'ordinateur (ABS (X) > 1E38)
RG	RETURN sans GOSUB correspondant. Très vraisemblablement le programme s'est achevé sans avoir d'instruction à ce sujet et dès lors est tombé dans la sous-routine: Correctif, utiliser END. Autre prossibilité: un branchement érroné a conduit le programme à l'intérieur de la sous-routine.
SN	Erreur de syntaxe. En général, erreur typographique ou ponctuation incorrecte.
ST	Formule de chaîne trop complexe. Divisez l'opération en plusieurs étapes.
TM	Non concordance des genres: essai d'attribuer des valeurs des chaînes a des variables numériques ou l'inverse.
UL	Ligne non définie. Instruction de branchement vers une ligne qui n'existe pas.

ANNEXE D

FONCTIONS TRIGONO METRIQUES



Dans un triangle rectangle ABC,

AB s'appelle le côté *adjacent* à l'angle α
BC s'appelle le côté *opposé* à l'angle α
AB s'appelle l'hypothénuse.

Les SINUS, COSINUS et TANGEANTE de l'angle se définissent comme suit:

$$\text{SIN } \alpha = \frac{\text{côté opposé}}{\text{hypothénuse}}$$

$$\text{COS } \alpha = \frac{\text{côté adjacent}}{\text{hypothénuse}}$$

$$\text{TAN } \alpha = \frac{\text{côté opposé}}{\text{côté adjacent}}$$

TOUTES les fonctions trigonométriques en BASIC sont considérées comme étant exprimées en *radians*. Un *radian* est une mesure d'angle en unités circulaire. Il y a 360 degrés, ou 2π *radians*, dans un cercle. (π = PI = rapport constant du diamètre à la circonférence, soit 3/1415926). Pour convertir de l'un à l'autre, procéder comme suit:

degrés/(180/π) = radians

radians★(180/π) = degrés

L'inverse d'une fonction c'est l'application de cette fonction en sens inverse. Par exemple, la tangeante d'un cercle de 1,5 radians est,

TAN(1.5) = 14.101419

L'inverse de la fonction TAN (tangeante) c'est ATN. (arc tangeante). Ceci s'utilise pour connaître l'angle, la tangeante étant connue

ATN(14.101419) = 1.5

L'inverse des fonctions SIN et COS ne sont pas disponibles en BASIC, mais elles peuvent être calculées en utilisant la fonction ATN, dans les formules suivantes.

ARC SIN X =	(ATN(X/SQR(-X*X+1))
ARC COS X =	(ATN(X)SQR(-X*X+1))+1.57Ø8

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INDEX

INFORMATION SUPPLEMENTAIRE

DRAGON 32
INFORMATION SUPPLEMENTAIRE

Ce texte contient quelques informations supplémentaires qui, nous l'espérons, vous permettront de tirer le maximum de votre micro-ordinateur Dragon 32. Le paragraphe concernant le cable de liaison au magnétophone est tout particulièrement important en raison de ce que les informations qu'il contient sont différentes de celles contenue dans le manuel fourni avec votre Dragon 32.

Vous trouverez ci-après des informations concernant les réglages à apporter éventuellement à votre poste de télévision ainsi que d'intéressants détails au sujet du raccordement d'une imprimante. Pour terminer, vous trouverez un tableau des adresses mémoire de Dragon 32.

Télévision

Lorsque vous aurez raccordé votre ordinateur au poste de télévision et que vous l'aurez allumé, choisissez un canal disponible et ajustez-le, ainsi qu'il est décrit dans le manuel. Si l'image du téléviseur n'était pas stable, ajustez la stabilité verticale de l'image au moyen du bouton prévu, sur votre téléviseur, à cet effet. (Au besoin, consulter le mode d'emploi du téléviseur).

Pour obtenir une image qui vous convienne, ajuster le contraste, la luminosité et le dosage couleur à votre idée.

Magnétophone à cassettes

Pour raccorder un magnétophone à cassettes en utilisant le câble de liaison Dragon, mettre la prise pentapolaire DIN dans la prise marquée TAPE (bande) sur le côté gauche de l'ordinateur. Les trois fiches, à l'autre bout du câble se raccordent au magnétophone à cassettes, comme suit:

(i) La fiche jack la plus petite, avec un fil bleu se raccorde à la petite prise correspondante du magnétophone, identifiée généralement REM ou TELC. D'habitude elle se trouve à côté de la prise pour le microphone.

(ii) La prise jack avec le fil rouge se raccorde à la prise généralement identifié AUX ou MIC ou parfois LINE IN. Si vous avez le choix entre AUX et MIC, préférez AUX.

(iii) La prise jack avec le fil blanc, va dans la prise identifiée EAR ou MONIT ou L/S ou SPKR ou parfois aussi par le dessin d'une petite oreille.

Quand vous utilisez les rebobinages rapides, avant ou arrière de votre magnétophone, il peut se révéler nécessaire de débrancher momentanément la prise REM, pour permettre au magnétophone de fonctionner.

Assurez vous bien que vous l'avez re-branchée ensuite.

Au lieu de débrancher comme indiqué plus haut, vous pouvez aussi taper MOTOR ON, puis appuyer sur ENTER, avant d'exécuter le rebobinage rapide. Après vous pourrez taper MOTOR OFF, puis ENTER, pour faire usage du magnétophone en corrélation avec l'ordinateur.

Si vous ré-utilisez une cassette ayant déjà servi, ayez soin de l'effacer à vide, avant d'enregistrer de nouveaux programmes ou de nouvelles données dessus.

Imprimante

Le raccord d'imprimante existant à gauche du Dragon 32 est prévu pour une imprimante utilisant une interface parallèle, type centronics. (C'est l'entrée n° 6 sur l'illustration du manuel). Les connections aiguilles sont les suivantes:

PIN 1	Print Strobe	PIN 2	+ 5 volts
PIN 3	Data bit 0	PIN 4	+ 5 volts
PIN 5	Data bit 1	PIN 6	0 volts
PIN 7	Data bit 2	PIN 8	0 volts
PIN 9	Data bit 3	PIN 10	0 volts
PIN 11	Data bit 4	PIN 12	0 volts
PIN 13	Data bit 5	PIN 14	0 volts
PIN 15	Data bit 6	PIN 16	0 volts
PIN 17	Data bit 7	PIN 18	0 volts
PIN 19	ACK	PIN 20	BUSY

La position des aiguilles à chiffre impair est sur la ligne supérieure de la partie du connecteur, PIN 1 étant situé à droite (conneteur face à vous). Les aiguilles à chiffre pair sont sur la ligne du bas, l'aiguille PIN 2 étant située à droite.

Cartouches

Il est essentiel que le courant d'alimentation de l'ordinateur soit coupé, lors de l'insertion ou du débranchement d'une cartouche, sur le côté droit de l'ordinateur.

DRAGON 32
ORGANISATION DE LA MEMOIRE

Adresse décimale	Contenu	Adresse hexadécimale
Ø - 1Ø23	Utilisation système	Ø - 3FF
255	RAM(MEV) page directe	ØFF
1Ø23	RAM page étendue	3FF
1Ø24 - 1535	mémoire texte écran	4ØØ - 5FF
	mémoire écran graphique	
1536 - 3Ø71	page 1	6ØØ - BFF
3Ø72 - 46Ø7	page 2	CØØ - 11FF
46Ø8 - 6143	page 3	12ØØ - 17FF
6144 - 7679	page 4	18ØØ - IDFF
768Ø - 9215	page 5	IEØØ - 23FF
9216 - 10751	page 6	24ØØ - 29FF
10752 - 12287	page 7	2AØØ - 2FFF
12288 - 13823	page 8	3ØØØ - 35FF
13824 - 32767	Stokage programme et variables	36ØØ - 7FFF
32768 - 49151	Interpréteur Basic	8ØØØ - BFFF
49152 - 65279	Mémoire cartouches	CØØØ - FEFF
65280 - 65375	Entrée/Sortie	FFØØ - FF5F
65376 - 655Ø3	bits de contrôle SAM	FF6Ø - FFDF
655Ø4 - 65535	Vecteurs MPU	FFEØ - FFFF

DRAGON 32/64

Guide de référence rapide

OPERATEUR MATHEMATIQUES ET LOGIQUES

Symboles	Operation	Priorité
↑	élévation à la puissance	1
*	multiplication	3
/	division	3
+	addition	4
-	soustraction	4
>	plus grand que	5
<	plus petit que	5
=	égal à	5
<>	pas égal à	5
>=	plus grand ou égal à	5
<=	plus petit ou égal à	5
NOT	NON logique	6
AND	ET logique	6
OR	OU logique	6

Le seul opérateur mathématique utilisable avec les chaînes de caractères est + (concatenation).

Les opérateurs relationels peuvent être utilisés pour comparer les chaînes de caractères aussi bien que les nombres.

TOUCHES DE CONTROLES

[←]	Rappel arrière - efface le dernier caractère
[SHIFT] [←]	Efface la ligne en cours
[BREAK]	Interruption d'un programme en cours - Redonne le contrôle au clavier.
[CLEAR]	Efface l'écran
[ENTER]	Entrée - Fin de la ligne en cours
[SHIFT @]	Pause sur programme en cours. Pour continuer appuyer sur n'importe quelle touche
[SHIFT] [Ø]	Passe de majuscules à minuscules
[SPACE BAR]	Espace

CLEAR *n, h*

Réserve *n* bytes pour le stockage de caractères et efface les variables. *h* spécifie la plus haute adresse accessible au BASIC. CLEAR 500

CLS *c*

Teinte l'écran dans la couleur désirée *c*.

0	noir
1	vert
2	jaune
3	bleu
4	rouge
5	blanc
6	bleu de prusse
7	magenta
8	orange

DATA

Permet de stocker dans un programme des données qui seront utilisées par READ.
DATA 4, 7, 14.-, 2, SUJET

DEF FN

Défini une fonction numérique.
DEF FNA (X) = X ★ X + 3 ★ X

DEF USR *n*

Défini le point d'entrée pour la fonction USR *n*, *n* = 0 - 9.
DEF USR 2 = 14000

DIM

Dimension de un ou plusieurs tableaux.
DIM X (40), A$ (7, 6), B (10, 2)

END

Termine un programme.
END

EXEC adresse

Transfère le contrôle à un programme machine à l'adresse indiquée. Si l'adresse est omise, le contrôle s'effectue par CLOADM.
EXEC 45043.

FOR TO STEP
NEXT

Crée une boucle a exécuter dans l'intervalle indiqué. STEP en indique l'incrément. Si STEP est omis, l'incrément est 1.
FOR X = 1 TO 10...NEXT X
FOR A = 1.3 TO 7.6 STEP 0.1...NEXT A
FOR G = 50 TO 10 STEP - 10...NEXT G

GOSUB

Appelle une sous-routine commençant à la ligne indiquée.
GOSUB 500.

GOTO — Branchement immédiat à la ligne spécifiée.
GOTO 45

IF condition THEN action 1 ELSE action 2
Teste la condition. Si vraie, réalise l'action 1, sinon réalise l'action 2
IF A = 3 THEN 300
IF B - C > 0 THEN X = X + 1 ELSE Y = Y - 1

INPUT — Arrête le programme en attendant une entrée par le clavier, INPUT "ENTREZ VOTRE NOM"; N$
INPUT A, B, C, D

LET — Assigne une valeur à une variable; peut être omis
LET X = 47.4

LINE INPUT — Permet l'entrée d'une ligne par le clavier, y compris les virgules. Doit être achevé par [ENTER].
LINE INPUT "TITRE"; T$

ON...GOSUB — Branchement multiple aux lignes spécifiées.
ON I GOSUB 100, 200, 300, 400

ON...GOTO — Branchement multiple aux lignes spécifiées.
ON K GOTO 245, 187, 310

POKE adresse, valeur
Place la valeur à l'adresse indiquée de la mémoire. La valeur doit être comprise entre 0 et 255.
POKE 25852,0

PRINT — Affiche à l'écran la liste d'éléments qui suivent PRINT. Les virgules déplacent l'élément suivant de 16 colonnes, les; ne donnent pas lieu à décalage.
PRINT "LA REPONSE" PRINT A, B PRINT "DEJA"; T; "ESSAIS"

PRINT TAB — Déplace le curseur au numéro de colonne spécifié.
PRINT TAB (10); "CREDIT"

PRINT USING — Permet de formatter une entrée.

#	nombre de chiffres
$	dollar devant un nombre
★★	rempli espaces vides de début de ligne avec des astérisques
↑↑↑↑	imprime en format exponentiel
% espace %	encombrement des chaînes (longueur en nombre d'espace) + 2
+	ordonne d'imprimer le signe

PRINT USING "#.#"; A, B
PRINT USING " % ∇ %"; A$

PRINT @ place — Affiche à l'écran à l'emplacement spécifié. (Ø-511).
PRINT 8, "PAGE"; N

READ — Affecte l'élément suivant d'une liste DATA a une variable spécifiée.
READ A, B, C$

REM — Tout ce qui suit REM est ignoré par l'ordinateur. Permet de placer des commentaires dans un programme.

RESTORE — Ramène le pointeur de DATA au premier élément de la première ligne de DATA.
RESTORE

RETURN — Renvoie le programme, à partir de la sous-routine à la ligne suivant immédiatement le GOSUB.
RETURN

STOP — Arrête l'exécution d'un programme. Utiliser CONT pour repartir.
STOP

INSTRUCTIONS POUR LE SON

PLAY chaîne de caractère

Joue la musique transformée en chaîne de caractères, comme ci-après:
A-G (ou 1-12), note. A = LA; B = SI; C = DO; D = RE etc...

On, octave — n = 1 à 5
Vn, volume — n = 1 à 31
Ln longueur de la note — n = 1 à 255
Tn, tempo — n = 1 à 255
Pn, pause — n = 1 à 255
XA$; exécute le segment de chaîne dans A$
(#) dièse et (-) bémol
PLAY "Ø3L2GBØ4CXY$;"

SOUND ton, durée

Exécute le son spécifique d'un ton compris entre 1 et 255, pour la durée spécifiée.
SOUND 18Ø,5

INSTRUCTIONS POUR LE CONTROLE DU MAGNETOPHONE

AUDIO — Permet de passer ou non le son du magnétophone sur la télévision.
AUDIO ON (ouvert) AUDIO OFF (fermé)

CLOAD — Transfère un programme de la bande magnétique à la mémoire. Le premier programme rencontré est transféré, sauf si un nom est spécifié.
CLOAD, CLOAD "NOM"

CLOADM — Permet le transfert d'une bande magnétique à la mémoire d'un programme en code machine. Un décalage par rapport a l'adresse de départ peut être spécifié. CLOADM, CLOADM "NOM".
CLOADM "NOM", 25Ø0

CLOSE — Fermeture d'un fichier ouvert.
CLOSE, CLOSE - 1

CSAVE... — Permet la conservation sur bande magnétique. (Attention le nom du programme ne doit contenir que 8 caractères au maximum).
Si l'on précise A, le programme est conservé en format ASC II
CSAVE, CSAVE "NOM", CSAVE "NOM", A.

CSAVEM nom, départ, fin, longueur
Conserve un programme en code machine, sur une cassette.
CSAVEM "NOM", 4E, 6F, 5F

EOF (-1) — Restitue 'vrai' si la fin du fichier sur cassette a déjà été lue.
IF EOF (-1) THEN 48Ø

INPUT # - 1 — Prend en INPUT des données de la cassette (le fichier doit être ouvert)
INPUT # - 1, A, B, C

MOTOR — Contrôle du moteur du magnétophone.
MOTOR ON (marche) MOTOR OFF (arrêt)

OPEN a, # - 1, nom
Etablit la liaison avec le magnétophone. a = "I" (entrées) a = "O" (sorties). Le nom est le nom du fichier de données.
OPEN "I". # - 1, "DATA2"

PRINT # - 1 — Inscrit des données sur la cassette. Le fichier doit avoir été ouvert.
PRINT # - 1, X9, LN (J)

SKIPF — Saute en fin du programme spécifié.
SKIPF "NOM"

CONTROLE DE L'IMPRIMANTE

LLIST — Imprime sur l'imprimante, les lignes de programmation dont le numéro est spécifié.
LLIST LLIST 10-95

OPEN "O", # - 2, nom du fichier
Ouvre la liaison de sortie vers l'imprimante.
OPEN "O", # - 2, "SORTIE"

PRINT # - 2 — Imprime une liste d'éléments sur l'imprimante.
PRINT # - 2, W, A$; X

INSTRUCTIONS DE SYSTEME

CONT — Continue un programme interrompu par STOP ou BREAK

DEL — Supprime les lignes de programme spécifiées.
DEL 100 - 350 DEL 100- DEL -80

EDIT — S'utilise pour modifier le contenu d'une ligne spécifiée.
EDIT 115 [ENTER]
En mode d'édition, on peut utiliser les sous-ordres suivants:

- *n*C — Change n caractères
- *n*D — Supprime n caractères
- I — Insertion de caractères
- H — Supprime le reste de la ligne et permet l'insertion
- L — Liste l'état de la ligne en cours
- *n*Sc — Place le curseur au nième caractère c
- X — Place le curseur en fin de ligne pour l'allonger
- *n* ← — Déplace le curseur vers la gauche, de n caractères
- *n* SPACE BAR — Déplace le curseur de n caractères vers la droite

LIST — Affiche les lignes indiquées, sur l'écran.
LIST LIST 10 - 95 LIST - 200

NEW — Efface le programme de la mémoire

RENUM nouvelle ligne, ligne de départ, incrément
Permet de renuméroter automatiquement les lignes.
Aussi branchements GOTO et GOSUB.
RENUM RENUM 100, 50, 10 RENUM,,20

RUN — Exécute un programme à partir de la ligne ayant le numéro le plus bas ou la ligne spécifiée.
RUN RUN 200

TROFF — Arrête l'action du traceur de programme
TROFF

TRON — Met en route le traceur du programme. Affichage du numéro de la ligne en cours d'exécution.

VARIABLES SIMPLES

Type	Nom	Intervalle
Numérique	AB	Ø à $\pm$ $1Ø^{38}$
Alphanumérique	AB$	Ø à 255 caractères

A est une lettre, B optionel pouvant être lettre ou chiffre. Si le nom utilisé a plus de 2 caractères, seuls les 2 premiers comptent.

TABLEAUX DE VARIABLES

Type	Nom
Numérique	AB (3, 4, 5)
Alphanumérique	AB$ (17, 8, 2)

Les noms et les limites sont les mêmes que pour les variables simples. La taille des tableaux est limitée par la place en mémoire.

CARACTERES SPECIAUX

' Abréviation de REM

? Abréviation de PRINT

: Séparateur d'instructions sur une même ligne

$ Derrière un nom de variable, la transforme en chaîne alphanumérique.

CODES D'ERREUR

/Ø	division par Ø
AO	fichier déjà ouvert
BS	indice érroné
CN	continuer impossible
DD	tentative de re-dimensionner un tableau
DS	instruction directe dans un fichier
FC	Appel illégal d'une fonction
FD	donnée incorrecte
FM	utilisation erronée d'un mode fichier
ID	instruction directe incorrecte
IE	essai d'entrée dans un fichier au delà de sa fin
I/O	erreur d'entrée/sortie
LS	chaîne de caractère trop longue
NF	NEXT sans FOR
NO	fichier non-ouvert
OD	plus de DATA disponibles pour READ
OM	plus de place en mémoire
OS	plus d'espace disponible pour chaînes
OV	nombre trop grand
RG	RETURN sans GOSUB
SN	erreur de syntaxe
ST	formule alphanumérique trop complexe
TM	non concordance de genre
UL	ligne inexistante

INSTRUCTIONS GRAPHIQUES

CIRCLE (x, y) r, c, h, s, e

Trace un cercle de centre x, y - de rayon r et de couleur c, h est le rapport hauteur, largeur (ellipse). Le cercle peut commencer à s et terminer à e - arc de cercle - (Ø à 1: Ø est à 3 heures).
CIRCLE (128,96),25
CIRCLE (1ØØ,5Ø),3Ø, 4, 1, Ø, Ø.5

COLOR encre, fond

Défini les couleurs du fond et de l'encre dans les limites disponibles.
COLOR Ø,5

DRAW chaîne

Trace une ligne en suivant les spécifications contenues dans la chaîne.

M x, y	place le curseur en x, y
Un	trace n points vers le haut
Dn	trace n points vers le bas
Rn	trace n points vers la droite
Ln	trace n points vers la gauche
En	trace n points à 45°
Fn	trace n points à 135°
Gn	trace n points à 225°
Hn	trace n points à 315°
Ak	déplace d'un angle de k: si Ø = Ø°, 1 = 90°, 2 = 180° et 3 = 270°
Sk	echelle de 1/4 à 15 par k = 1 à 62
N	n'initialise pas la position de dessin
B	mouvement à blanc (sans écrire)
C	couleur de la ligne
X	exécution d'une partie de la chaîne

GET (x1, y1) - (X2, Y2) tableau, G

Lit le contenu graphique d'un rectangle de l'écran de dimension x1, y1 (sommet) x2, y2 (bas) et le place en tableau. G, en option permet de spécifier des détails complémentaires

LINE (x1, y1) - (x2, y2), a, b

Trace une ligne de x1, y1 à x2, y2. Si x1, y1 est omis, la ligne démarre du dernier point utilisé. a doit être PSET (sélection de couleur de l'encre) ou PRESET (sélection de la couleur du fond). b est optionel mais peut être B qui trace un rectangle dont la diagonale est (X1, y1) (x2, y2) ou BF (traçage du rectangle en le colorant en couleur d'encre.
LINE (1Ø,2Ø) - (35,15, PSET
LINE Ø,Ø) - (128,96), PSET,BF.

PAINT (x, y) c, b

Colore le dessin à partir du point (x, y) en couleur C et s'arrêtant a des limites définies par la couleur b.
PAINT (128,96),1,4

PCLEAR n — Réserve n pages de mémoire graphique (n = 8)
PCLEAR 6

PCLS c — Vide l'écran de haute résolution et le teinte dans la couleur c
PCLS 5

PCOPY a TO b

Recopie une page graphique a sur la page b
PCOPY 3 TO 4

PMODE mode, page

Sélectionne le mode de haute résolution (Ø à 4) et la première page en mémoire (1 à 8)
PMODE 4,1

PRESET (x, y) — Eface un point (x, y) en le mettant dans la couleur du fond.
PRESET (15,37)

PSET (x, y, c) — Allume le point (x, y) dans la couleur c
PSET (12Ø,95,4)

PUT (x1, y1) - (x2, y2), tableau, action

Affiche les détails d'un graphisme contenu dans le tableau dans un rectangle défini par x1, y1 (sommet) et x2, y2 (bas).
Action est optionel et peut être PSET, PRESET, AND, OR ou NOT
PUT (A, C) - (A + 24, C + 24),;

RESET (x, y)

Remet un point basse résolution graphique dans la couleur du fond.
RESET (10,12)

SCREEN (type, couleur)

Sélectionne le type d'écran (texte = 0 graphique = 1) et la gamme de couleur (0 ou 1)
SCREEN 1,1

SET (x, y, c)

Allume un point en x, y en basse résolution, dans la couleur c.
SET (15,30,8)

FONCTIONS DE CHAINE

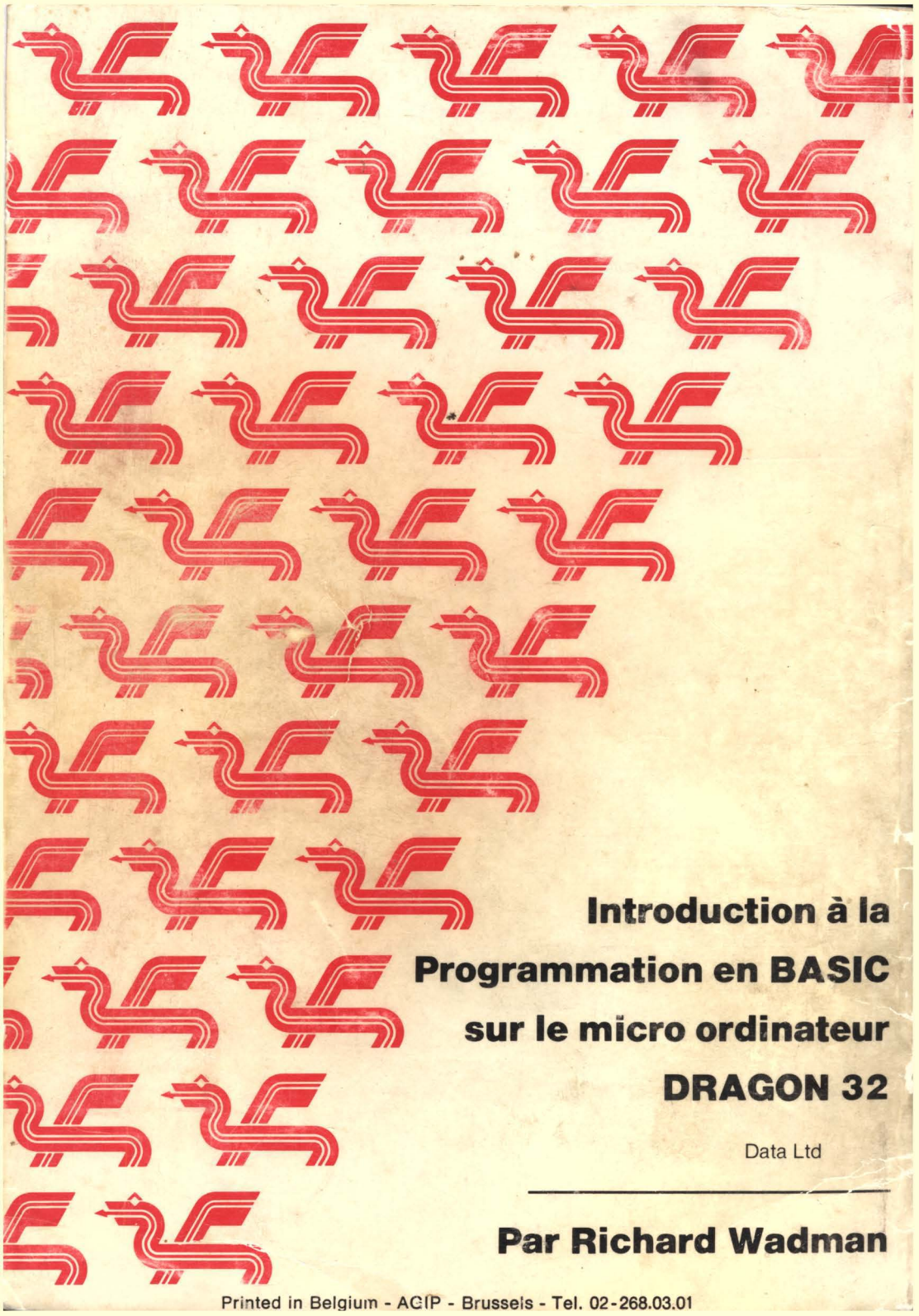
ASC (X$)	Restitue le code ASCII du premier caractère dans la chaîne. X = ASC (A$)
CHR$ (N)	Restitue le caractere qui correspond à la valeur du code ASCII spécifié. A$ = CHR$ (143)
HEX$ (N)	Restitue sous forme de chaîne la valeur hexadécimale de l'argument. PRINT HEX$ (46)
INKEY$	Analyse le clavier et restitue la touche qui a été préssée. K$ = INKEY$
LEFT$ (X$, N)	Restitue les N premiers caractères de la chaîne X$ B$ = LEFT$ (A$,9)
LEN (X$)	Restitue la longueur de la chaîne X$ K = LEN (K$)
MID$ (X$, N, M)	Restitue la partie de X$ de longueur M, à partir de la position N B$ = MID$ (A$, 4, 1)
RIGHT$ (X$, N)	Restitue N derniers caractères de X$ B$ = RIGHT$ (A$, 3)
STRING$ (N, X$)	Restitue une chaîne de caractères constituée de N fois le premier caractère de X$. Le code ASCII peut remplacer X$ L$ = STRING$ (32, "+") PRINT STRING$ (5,41)
STR$ (N)	Remplace N par sa représentation en chaîne de caractères X$ = STR$ (14.4)
VAL (X$)	Convertit les nombres contenus dans une chaîne X$ en valeur numérique. P = VAL (D$)

FONCTIONS NUMERIQUES

Arguments des fonctions

a	valeur numérique $\pm 10^{38}$
b	valeur entière entre Ø - 65535
r	angle en radians
x et y	abscisse et ordonnée d'un point en basse résolution x = Ø - 63 y = Ø - 31
w et z	abscisse et ordonnée en haute résolution w = Ø - 255 z = Ø - 191
ABS (a)	Valeur absolue de a AY = ABS (Y)
ATN (a)	Restitue l'arc tangeante en radians X = ATN (a)
COS (a)	Restitue le cosinus de l'angle a, à exprimé en radians W = COS (2★B)
EXP (a)	Restitue la valeur exponentielle e^a. Q = EXP (-5.6)
FIX (a)	Restitue la valeur arrondie de a Z = FIX (13.43)
INT (a)	Convertit un nombre en sa partie entière A = INT (5.7)
JOYSTK (a)	Restitue les coordonnées horizontales et verticlaes des manettes de jeu. Ø = horizontal gauche 1 = vertical gauche 2 = horizontal droit 3 = vertical droit H = JOYSTK (O) V = JOYSTK (P)
LOG (a)	Restitue le logarithme naturel Z = LOG (7.842)
MEM	Restitue la place disponible en mémoire PRINT MEM
PEEK (n)	Restitue le contenu de la mémoire à l'adresse indiquée n PEEK (66Ø82)
POINT (x, y)	Teste le point x, y; restitue le code de la couleur si le point est allumé et Ø si il est éteint. -1 s'il s'agit d'un caractère T = POINTS (15,25)

POS (d)	Restitue la position actuelle de d. P = POS (-2)
PPOINT (w, x)	Même utilisation que POINT, en haute résolution graphique. Restitue le code couleur si allumé et Ø si éteint. P = PPOINT (186,54)
RND (n)	Génère un nombre entier au hasard entre 1 et n. Si n est Ø le nombre est compris entre Ø ou 1. X = RND (18)
SQR (a)	Restitue la racine carrée de a X = SQR (A + 7 ★ C)
SGN (a)	Restitue le signe du nombre, soit 1 = positif, soit Ø = zéro, soit - 1 = négatif G = SGN (4 ★ H/3)
TAN (r)	Restitue la tangante de l'angle celui-ci étant exprimé en radians Z = TAN (2 ★ A3)
TIMER	Restitue la valeur de l'horloge en temps réel ou permet son réglage. T = TIMER TIMER = Ø
USR (n)	Appelle le code machine à l'adresse n spécifiée F = USR (D)
VAPTZ (var)	Restitue l'adresse du pointeur à la variable spécifiée Z = USR (VARPTR (F))



Introduction à la Programmation en BASIC sur le micro ordinateur DRAGON 32

Data Ltd

Par Richard Wadman

Printed in Belgium - ACIP - Brussels - Tel. 02-268.03.01