MISC10/FREN.COV

AMSTRAD

BASIC

guide d' initiation

AMSTRAD
BASIC
guide d'initiation

1ère Partie
Premiers Pas

Amstrad France
143 Grande Rue
92310 SEVRES

Amstrad BASIC

Guide Pratique

Première partie: Notions de base

SOFT 411 ISBN 1 85084 000 B

☐ Programmation: Dave Atherton

☐ Ecrit par Dave Collier et George Tappenden

☐ Production: Peter Hill et Ray Smith

☐ Adaptation: Anne Thomas et Patrick Feldstein

SOMMAIRE

PREFACE

Ce manuel constitue la première partie d'un cours de programmation en langage Basic à partir de l'ordinateur couleur personnel Amstrad CPC464. Les deux datacassettes qui accompagnent ce manuel contiennent des programmes d'ordinateur qui font partie intégrante de ce cours.

La datacassette A contient:

- Des programmes destinés à vous aider à acquérir les principes d'une programmation simple et amusante.
- Des jeux destinés à vous distraire et à vous aider à vous habituer à l'ordinateur.

La datacassette B contient:

- Des test d'auto-évaluation pour s'assurer de sa propre compréhension des concepts décrits dans chaque chapitre.

La deuxième partie de ce manuel, Basic Avancé, traite de techniques de programmation plus élaborées.

1 DE QUOI S'AGIT-IL?

Pour lire ces lignes, vous êtes probablement l'heureux propriétaire d'un ordinateur Amstrad CPC464. Le caractère exceptionnel de l'affichage et l'excellente qualité du son vous ont déjà dévoilé une partie du nouveau monde fascinant qui s'ouvre à vous. Mais pour exploiter toutes les facultés de votre ordinateur, vous vous êtes déjà rendu compte de la nécessité d'apprendre le langage Basic du CPC464.

BASIC

Il s'agit du langage ordinateur le plus connu. C'est également le langage idéal pour le débutant puis qu'il lui suffit de quelques heures d'étude pour créer son propre programme, sans parler des multiples satisfactions qui en découlent: créer son propre programme peut déjà constituer un hobby en soi.
Mais qu'est-ce donc que programmer?
Vous devez tout d'abord vous rappeller qu'un ordinateur peut faire beaucoup de choses mais qu'il ne peut pas penser; c'est à vous de penser pour lui. Penser pour l'ordinateur, c'est à dire mettre en oeuvre des moyens pour parvenir à une fin, revient à respecter la suite d'instructions que constitue un programme. En composant un programme, vous pouvez obtenir un jeu d'arcades, un traitement de textes ou même une machine qui s'occupe de vos comptes.

Vous vous apercevez sûrement que les programmes conçus pour le CPC464 doivent être modifiés pour fonctionner dans d'autres types d'ordinateurs. Ceci parce que le langage Basic Amstrad utilisé par le CPC464 contient de nombreuses commandes et de nombreuses fonctions uniques qu'un équipement moins sophistiqué n'est pas en mesure d'offrir.

Tout comme tout autre langage, le Basic dispose d'un vocabulaire particulier; ce vocabulaire se réume un "mots-clé" que vous vous apprêtez à apprendre ainsi que leur signification sur le CPC464. A chaque fois qu'un mot-clé est mentionné dans ce manuel, il figure automatiquement sur la marge extérieur afin que vous puissez le consulter facilement lorsque vous avez besoin de vous remettre à jour.

Methode du manuel

Chacun des chapitres proposés représente environ une à deux soirées de travail. Il contient en général:

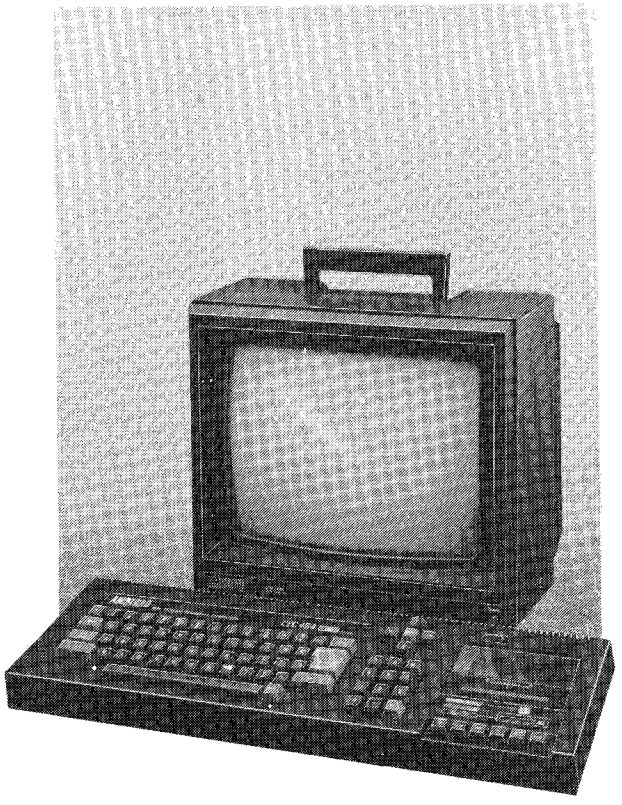
- Une explication écrite.
- Un travail pratique sur l'ordianteur.
- Des exemples vous permettant de créer vos programmes vous-même.

Ces exercices visent à consolider vos connaissances, ils sont en outre appuyés par un test d'auto-évaluation à la fin de chaque chapitre.

Ne sautez pas de chapitre. Chaque nouvelle information est introduite de façon progressive et repose sur les données des chapitres précédents. Au cas où un chapitre ou une partie de chapitre vous paraît un peu compliqué, parcourez le rapidement une fois ou deux avant de le travailler lentement. Vérifiez votre compréhension à l'aide des tests d'auto-évaluation.

Une fois cette partie du cours absorbée, vous devriez être capable de composer des programmes simples et sûrs pour vous-même. La deuxième partie du cours, *Basic Plus*, explique certaines fonctions plus élaborées du Basic Amstrad et enseigne la composition de programmes plus recherchés.

2 INSTALLATION ET DEMARRAGE



Avant tout, déballez votre CPC464. Si cela est déjà fait, vous pouvez sauter les quelques paragraphes qui suivent:

Les differents emballages devraient contenir les articles suivantes:

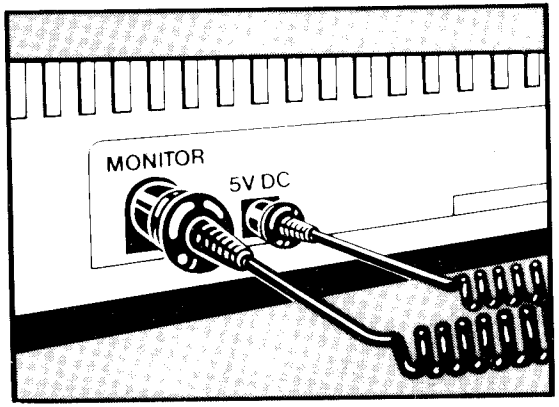
- Le CPC464, Micro-Ordinateur Couleur Personnel
- Le Moniteur GT64, le Moniteur Couleur CTM640
- Le Modulateur MP1, alimentation courant (facultatif).
- Le Guide de l'Utilisateur de L'Amstrad CPC464.
- La Cassette de Démonstration.

Placez le CPC464 et son moniteur (ou le modulateur MP1/alimentation courant et récepteur TV familial) sur un bureau ou une table suffisamment spacieux dans une pièce tranquille de la maison. Introduisez avec soin les deux fils de l'avant du moniteur ou MP1 dans leur fiches situées à l'arrière de CPC464 (voir schéma). Ne branchez pas tout de suite le moniteur MP1 sur le conducteur principal 13 amp. Veillez à ce que les fiches soient à leur place à l'arrière du CPC464 mais n'utilisez pas trop de puissance. Evitez de brancher et débranchez trop souvent les prises et fiches car elles risquent de s'abîmer.

Si possible, branchez votre CPC464 une fois pour toutes (au moins tout au long de cette étude).

Si vous utilisez un récepteur TV familial avec votre CPC464, veillez à ce que vous disposiez de deux fiches 13 amp. Ou utilisez une prise multiple afin de brancher la TV et le MP1 en même temps. Puis connectez le fil coaxial du MP1 à la prise-antenne.

Procurez vous un siège confortable et installez vous à un mètre de la TV ou du moniteur (travailler trop prés de l'écran peut fatiguer considérablement, surtout les yeux).



Placez ce manuel de façon à pouvoir le lire rapidement en frappant sur le clavier tout en regardant l'écran. Il est également conseillé de garder à sa portée le "Guide de l'Utilisateur de l'Amstrad CPC464".

Puis branchez sur le conducteur principal et allumez. L'écran devrait afficher les mots suivants:

```
Amstrad 64K Microcomputer  (v1)

© 1984 Amstrad Consumer Electronics plc
         and Locomotive Software Ltd.

BASIC 1.0

Ready
■
```

Si ces mots n'apparaissent pas (caractères jaunes sur fond bleu pour le CTM640, moniteur couleur), éteignez et recommencez. Si vous disposez d'un téléviseur, vérifiez si c'est bien un modèle du type PERITEL. Si vous ne parvenez toujours pas à vos fins, vérifiez si le voyant lumineux "ON" du CPC464 est bien

allumé (rouge). Sinon, veillez à ce que:

- Le bouton MARCHE/ARRET de CPC464 soit bien en marche.
- Il n'y ait aucune panne de courant.
- Le fil soit bien branché sur le CPC464.

Si rien n'y fait, rentrez en contact avec votre vendeur.

RUN

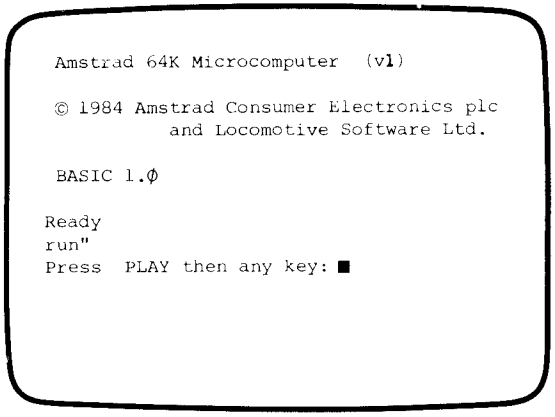
HELLO

Signifie que l'ordinateur est prêt à recevoir des commandes ou un programme. Le petit carré noir est connu sous le nom de "curseur" et vous indique l'emplacement de votre prochain caractère au fur et à mesure que vous tapez.

Nous sommes donc partis. Introduisez la cassette *"Basic: Notions de Base", datacassette A* dans le lecteur de cassettes (datacorder) et tapez ce qui suit:

run" **[ENTER]**

Pour taper le caractère ", maintenez enfoncée l'une des touches SHIFT au choix et appuyez sur " (au-dessus du 2, rang supérieur gauche). En appuyant sur la touche ENTER, vous signalez au CPC464 que vous avez terminé votre message et que vous attendez la suite. Si votre commande a été faite correctement, le CPC464 répond par une nouvelle ligne au message de l'écran:



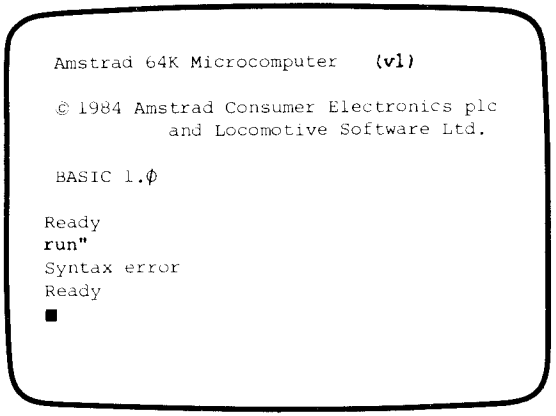
Vérifiez si la bande est rembobinée à son début en appuyant sur la touche REW du lecteur de cassettes, puis suivez les instructions. Appuyez sur la touche PLAY puis sur la touche ENTER. Vous entendez le son aigu du speaker incorporé dés que le programme est enregistré par le CPC464.

Si vous faites une faute de frappe (avant d'appuyer sur la touche ENTER), ayez recours à la touche

[DEL]

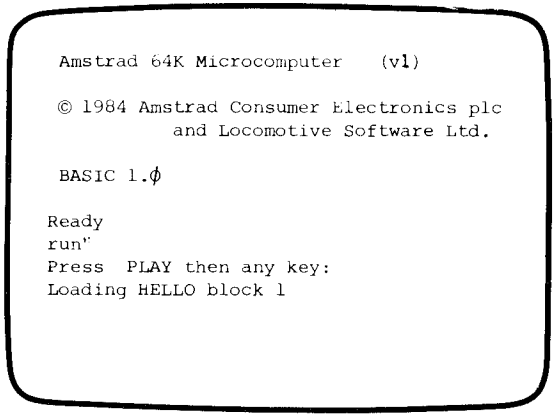
Cette touche sert à effacer: le curser recule et le dernier caractère tapé disparaît. Vous pouvez alors retaper le caractère souhaité.

Si vous avez fait une faute de frappe et avez déjà appuyé sur la touche ENTER, ne vous inquiétez pas; Le CPC464 ajoutera simplement une ligne sur l'écran comme l'indique l'exemple suivant:



Si un tel message apparaît, il vous suffit de tout recommencer et de retaper la ligne.

Si tout est en règle (et si vous avez composé la commande correctement), le message suivant est affiché rapidement sur l'écran:



Ce message est presque simultanément suivi par le programme HELLO qui vous accueille dans le monde de l'ordinateur.

Lorsque vous aurez parcouru le programme HELLO plusieurs fois, jetez un oeil en haut à gauche du clavier et vous apercevez une touche rouge; il s'agit de la touche ESCape; pressez la deux fois:

[ESC] [ESC]

Le CPC464 répondra par le message suivant:

```
Break in 14Ø
Ready
■
```

Le nombre doit être 140. Et cela ne veut pas dire que l'on vient de vous octroyer à vous ou votre ordinateur 140 secondes, heures ou jours de repos! Ni que votre domicile se trouve au 140 de votre rue à partir d'aujourd'hui. Oubliez plutôt ce numéro; nous le verrons en détails plus tard. Dans tous les cas, vous en avez terminé avec le programme et vous vous apprêtez à border la suite de ce chapitre.

A propos, observez que le CPC464 marque la différence entre le 0 (le chiffre) et le O (la lettre) en barrant le caractère lorsque celui-ci n'est pas alphabétique.

Jouons à un jeu

A priori, vous pouvez penser que vous avez mieux à faire que de consacrer votre temps et votre argent à des jeux pour ordinateur; pourtant ils s'avèrent beaucoup plus intéressants qu'ils ne paraissent. D'abord, vous faites connaissance avec votre machine sans passer par des exercices ennuyeux et ensuite les jeux vous permettent de vous rendre compte que les ordinateurs sont aussi faits pour vous distraire.

A présent, passons à quelquechose de différent; tapez le message suivant:

load "simon" **[ENTER]**

N'oubliez pas de maintenir enfoncée la touche SHIFT pour taper le caractère ". Le CPC464 répond de la façon suivante:

Press PLAY then any key:

Suivez de nouveau les instructions. Appuyez sur le bouton PLAY du lecteur de cassettes puis appuyez sur la touche ENTER du clavier principal. Si vous avez tapé le nom correctement et si tout est en ordre, le CPC464 met en marche la cassette du lecteur de cassettes et charge un programme appellé SIMON. Chaque programme porte un nom pour vous permettre de la trouver facilement. Cette fois-ci, le programme sera recherché dans la mémoire du CPC464 et ne fera rien avant que vous le commandez.

Le CPC464 donne le message suivant:

Loading SIMON, block 1

Changé en:

Loading SIMON, block 2

Ne vous inquiétez pas. Une fois la cassette arrêtée, un autre message est donné:

Ready

C'est le moment d'informer le CPC464 de ce que vous souhaitez qu'il fasse du message maintenant en mémoire. Tapez:

run **[ENTER]**

Et amusez vous bien

LOAD

Test

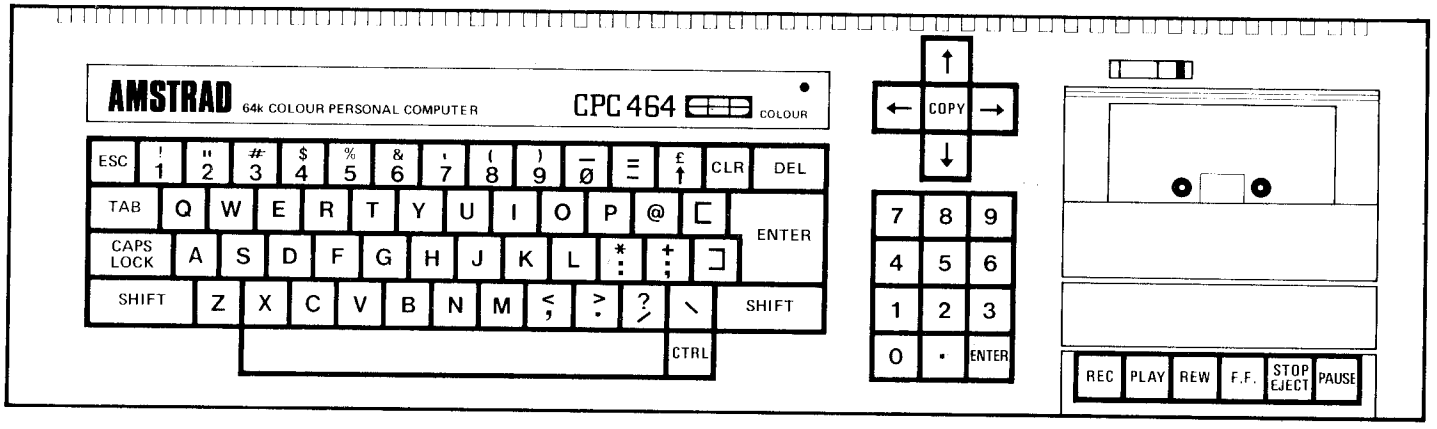
Si vous êtes fatigué de jouer à SIMON, arrêtez en appuyant deux fois sur la touche ESC consiste à charger votre premier test d'auto-évaluation (SAT2, datacassette B) en suivant la même méthode que celle employée pour SIMON. Tout au long de ce programme, il vous sera posé des questions sur ce chapitre afin que vous sachiez s'il vous faut revoir certains points.

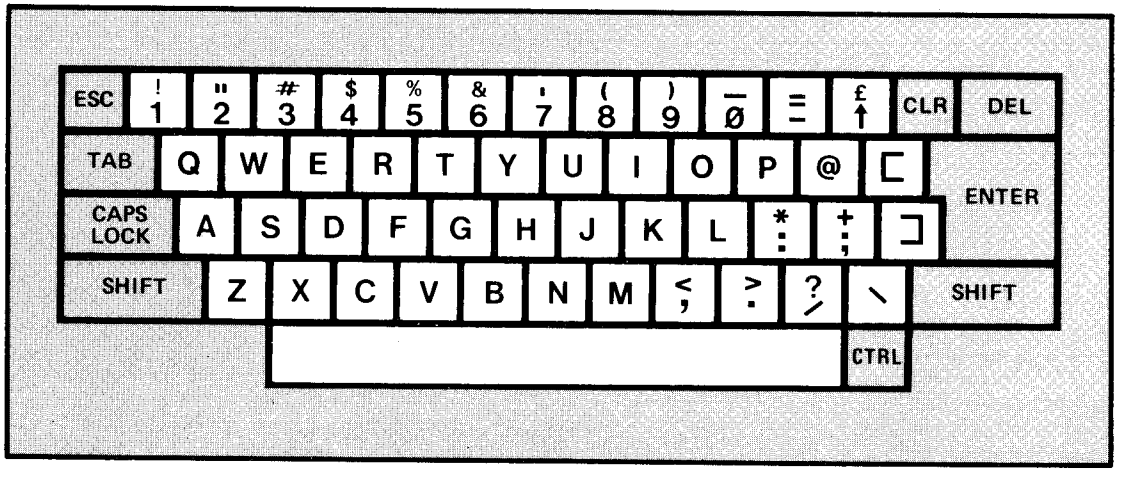
3 LE CLAVIER

Nous avons déjà signalé la nécessité d'apprendre le langage Basic pour communiquer avec le CPC464. Comme vous pouvez voir, vous transmettez des ordres au CPC464 en composant des mots et des nombres sur le clavier. Ce chapitre vous explique tout cela en détail. Vous allez apprendre à connaître la position de chaque touche et à utiliser chacune d'entre elles à bon escient.

Le CPC464 dispose de cinq groupes de touches distincts:

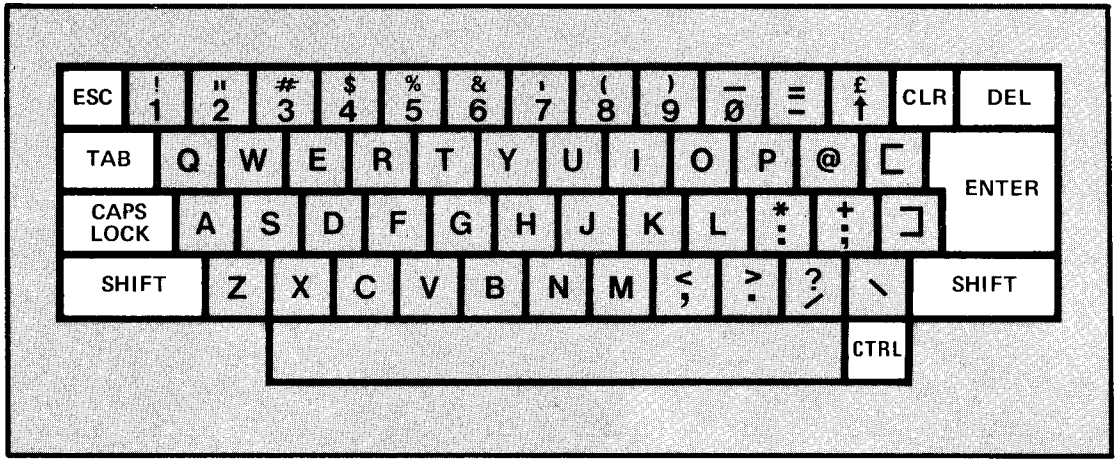
- Les touches de caractères
- Les touches de commandes
- Le clavier numérique
- Les touches du curseur
- Les touches de commande du datacorder





Clavier principal: Les touches de caracteres

Vous les avez déjà utilisées; Si vous savez vous servir d'une machine à écrire, vous savez déjà tout. Cette partie du clavier comprend lettres, nombres, de nombreux caractères de ponctuation (connus sous le nom de caractères spéciaux) et une barre-espace pour les blancs. Si vous maintenez enfoncée une des touches plud d'une demi-seconde, le caractère se répètera comme si vous aviez réappuyé et continuera de se répéter jusqu'à ce que vous lâchiez la touche en question.



Clavier principal: Les touches de commande

[ESC]

ESC signifie ESCape. Si vous appuyez sur cette touche alors qu'un programme est en cours, cela arrête le CPC464, mais vous pouvez reprendre le programme en appuyant sur une autre touche du clavier. Si vous appuyez sur la touche ESC une deuxième fois, le CPC464 se mettra en position "Ready".

[TAB]

En pressant la touche TAB, vous apercevez sur l'écran une flèche tournée vers la droite. Mais on n'utilise pas cette touche pour cette partie du cours, vous apprendrez à vous en servir dans la deuxième partie.

[SHIFT]

SHIFT change les signes lorsque vous appuyez sur les touches de caractères. En appuyant su SHIFT, vous obtenez les lettres majuscules à partir des touches de lettres et les signes supérieurs des touches à partir des touches de nombres et de caractères spéciaux.

[CAPS] [LOCK]

Appuyez de nouveau sur CAPS LOCK. Le résultat est le même que lorsque vous maintenez SHIFT enfoncé, à la seule différence que vous conservez les signes inférieure sur les touches de nombres et de caractères spéciaux.

[CLR]

La touche CLR (CLeaR) ressemble assez à la touche DEL; elle efface le caractère, mais pas celui situé à gauche du curseur comme le fait la touche DEL. Elle "mange" le caractère placé sous le curseur sans changer de position, et tout ce qui figure à droite du curseur se déplace automatiquement vers la gauche (un espace).

[DEL]

Vous vous rappellez peut-être cette touche mentionnée au chapitre précédent. Lorsque vous tapez une ligne, vous effacez le caractère situé à gauche du curseur en appuyant sur DEL. Cette touche recule également le curseur qui prend la place du caractère effacé.

[ENTER]

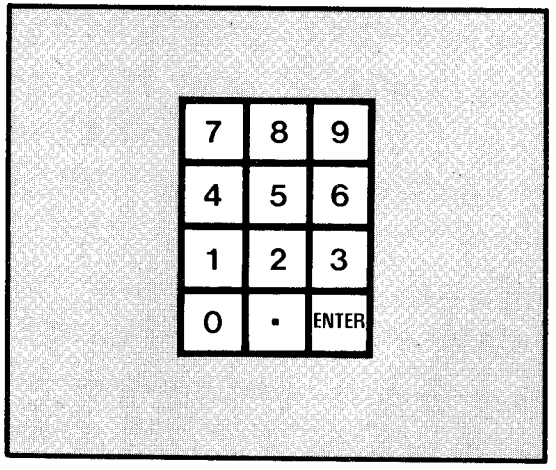
Cette touche peut être comparée la touche de retour de chariot d'une machine à écrire électrique. Vous devez la presser à la fin de chaque ligne tapée afin que le CPC464 sache que vous avez terminé. Normalement, le curseur passe de la fin du dernier mot ou nombre tapé au début de la ligne inférieure suivante. Quelquefois, le CPC464 peut aussi émettre:

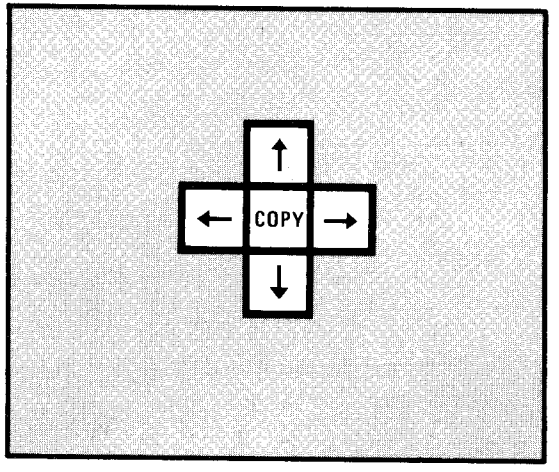
```
Syntax error
```

En Basic, cela signifie "je ne comprends pas", et cela suppose que le message n'est pas correct. Mais nous rentrerons dans les détails plus avant dans le cours.

[CTRL]

Cette touche signifie ConTRoLe. En la maintenant enfoncée, vous obtenez une autre gamme de signes à partir des touches de lettres et de quelques-unes des touches de chiffres (en plus de leurs signes supérieurs). Elle ordonne également au CPC464 de réaliser certaines opérations lorsqu'elle est utilisée en même temps que d'autres touches de commande. Mais nous verrons cela plus loin.



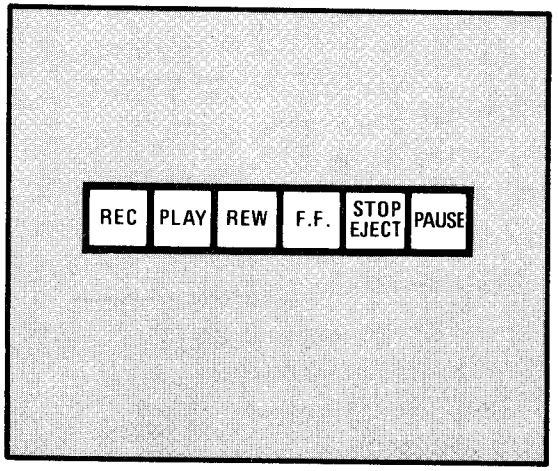


Le clavier numerique

L'agencement fonctionnel de ces touches permet la frappe de nombreux chiffres. Exception faite de la touche supplémentaire ENTER, elles ne différent pas des touches situées au premier rang du clavier principal. Mais les touches SHIFT et CTRL ne le modifient pas si elles ne sont pas programmées pour. Cette programmation spéciale n'est pas expliquée dans cette partie du cours mais vous pourrez voir plus avant dans ce chapitre que la touche ENTER a une autre fonction.

Le curseur

Les touches-flèche servent à déplacer le curseur vers la direction souhaitée. La fonction de la touche COPY sera expliquée plus loin dans ce cours.



Les commandes du lecteur de cassettes

Il n'existe qu'une différence notable entre ces touches et les touches d'un appareil enregistreur. Une fois les touches PLAY (ou PLAY et REC) enfoncées, le lecteur de cassettes ne se mettra pas en route avant d'en avoir reçu l'ordre par le CPC464.

Travaux d'application

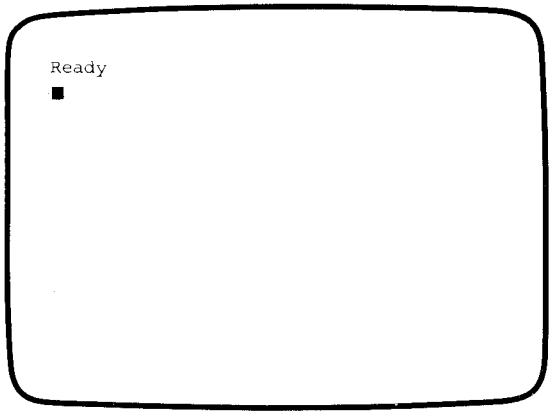
Vérifiez si la cassette est bien introduite dans le datacorder. Si votre écran affiche toujours vos derniers travaux SAT2 du chapitre précédent, vous pouvez vous en débarrasser en tapant le message suivant:

CLS

cls **[ENTER]**

Cela signifie: "Effacez l'écran". Et c'est exactement ce qui se passe. Essayez. Quelquesoit le message affiché sur l'écran auparavant, le résultat est le même.

RUN



Magique n'est-ce pas? A présent, vous pouvez taper le message suivant:

run "letters" **[ENTER]**

Maintenant, il vous suffit d'appuyer sur les touches du clavier pour observer ce qui se passe. Tâchez d'appuyez sur plus d'une touche à la fois.

Ce programme vous permet de voir sur l'écran de nombreux signes qui ne figurent pas sur la partie supérieure des touches. Vous les obtenez en maintenant la touche CTRL enfoncée pendant que vous appuyez sur les touches de chiffres et de lettres, même si toutes les touches ne disposent pas de "caractères cachés".

Une fois un programme amorcé, celui-ci ne s'arrête pas avant de se terminer. Ou bien s'il doit se répéter en permanence comme LETTERS, vous devez le stopper. Vous avez déjà vu plus haut qu'il suffit pour cela d'appuyer deux fois sur la touche ESC. Vous pouvez le faire d'une autre façon en "forçant une reprise". Vous obtiendrez le même résultat que si vous arrêtiez le courant et remettiez l'appareil en marche. Faites de la manière suivante: maintenez enfoncées les touches

[CTRL] and **[SHIFT]**

and press

[ESC]

Comme vous pouvez le voir, le CPC464 retourne au "Hello" qui était affiché lorsque vous avez mis l'appareil en marche pour le première fois. A partir de maintenant, cette opération s'appellera CTRL/SHIFT/ESC.

Attention: Lorsque vous composez cette commande, le CPC464 "oublie" tout programme retenu préalablement en mémoire.

Ça va jusqu'à maintenant? Le programme qui suit est trés simple, mais d'abord apprenons une autre façon de le passer.
Maintenez enfoncée la touche

[CTRL]

Appuyez sur la touche

[ENTER]

(du clavier numérique cette fois-ci et non pas celle du clavier principal).

Le CPC464 répondra exactement comme si vous aviez tapé une ligne complète de commandes. Vous les reconnaissez? Rappellez vous, nous vous avions signalé plus tôt que la touche ENTER avait une autre fonction!

REPEAT NAME (Veuillez répéter le nom) (programme que vous venez de charger). Mais ce programme ne va pas vous amuser éternellement...aussi faites le CTRL/SHIFT/ESC pour remettre le CPC464 à zéro, puis chargez et passez le programme suivant:

KEYBOARD (clavier)

Il s'agit d'un programme d'entraînement qui vous permet de vous habituer aux touches du CPC464. Très vite, vous commencerez à vous rappeller la position des touches. Il vaut parfois la peine de revenir à ce programme de temps en temps afin d'améliorer votre vitesse.

Test

Avant de passer au chapitre suivant, chargez et passez le SAT3.

CHAQUE CHOSE A SA PLACE

Vous avez pu voir grâce à la cassette démonstration et aux programmes précédents que le CPC464 dispose de graphiques superbes. Ce chapitre a pour but de vous apprendre le b.a ba dont vous aurez besoin pour tracer sur l'écran les lignes et formes de vos propres créations.

Le CPC464 affiche tous les caractères et graphiques dans une fenêtre sur l'écran du moniteur. Les côtés, le haut et le bas sont appelés les bords; on ne les utilise jamais. Mais vous pouvez en changer la couleur. Passez le message suivant:

BORDER

border Ø **[ENTER]**

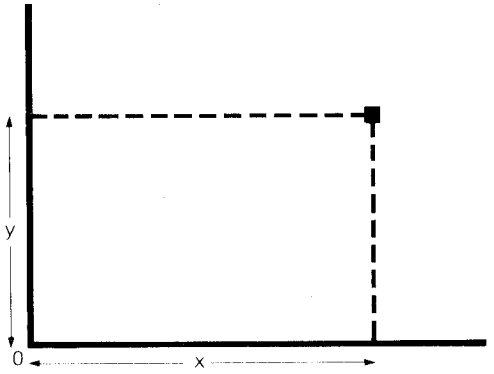
C'est noir, n'est-ce pas? Maintenant essayez ça

border 26 **[ENTER]**

Et nous voilà à l'autre extrême. A présent, vous pouvez vous amuser à essayer tous les nombres intermédiaires. Si vous avez lu le *Guide de l'Utilisateur,* vous savez déjà que le CPC464 vous offre un choix de 27 couleurs différentes. Même si vous ne possédez pas de TV couleur ou le moniteur couleur CTM640, vous ne perdrez pas votre temps en exécutant cet exercice qui consiste à passer d'un nombre à l'autre: l'ordre ascendant des nombres correspond aux différents tons de gris entre le noir et le blanc.

Coordonnes

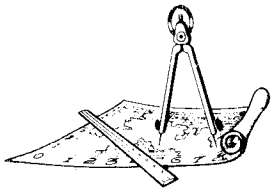
Chaque ligne, forme ou caractère qui paraît à l'écran est constitué de points minuscules. L'emplacement de ces points est signalé par les coordonnées x et y.



L'abscisse est appelée x et l'ordonnée y.

Passez le programme suivant. Il s'appelle DRAW (dessin). Vous pouvez tracer des lignes droites sur l'écran en maintenant enfoncée la touche TAB et en appuyant sur l'une des touches du curseur (flèches). Si vous voulez vous déplacer sans rien tracer, appuyez simplement sur les touches du curseur. Vous pouvez accélérer l'opération en enfonçant la touche SHIFT en même temps.

Comme vous pouvez le voir, le programme indique l'abscisse x et l'ordonnée y de l'emplacement du curseur (en cours). Si vous laissez le curseur se déplacer vers le haut sans l'arrêter, il finira par disparaître de la fenêtre. Prenez alors que valeur de y à ce moment. Faites de même pour x. Vous découvrirez alors que l'écran-graphique du CPC464 mesure 400 points de haut et 640 points de large.



Essayez de placer un carré façon inhabituelle sur l'écran. Voici un exemple:

Placez un carré de 50x50 sur l'angle en bas à gauche de l'écran avec x=300 et y=150.

Tout d'abord, déplacez le curseur sur l'abscisse x jusqu'à 300. Puis déplacez-le sur l'ordonné y jusqu'à 150. Si le carré fait 50x50 de côtés, nous devons tracer une ligne de ce point à x=300 +50=350. Tracez cette ligne. Si vous êtes allé trop loin, vous pouvez "manger" la ligne en déplaçant le curseur vers la direction opposée jusqu'à obtention de la valeur souhaitée de x. On compte à presente:

x=35Ø y=15Ø

Maintenant, tracez une ligne verticale jusqu'à ce que y=150+50=200. Puis retranchez de x 50: x=350–50=300, et enfin faites de même pour y, y=200–50=150. Vous revenez ainsi au point de départ.

Ne vous inquiétez pas si vous n'obtenez pas un résultat absolument identique, c'est l'exercice qui compte.

A la mode

MODE

Avant d'étudier les précoordonnés de plus prés, observons une autre caractéristique de l'affichage écran. Lorsque vous commencez à mettre le CPC464 en marche, le programme "hello" apparait en caractères deux fois plus importants que ceux de cette page environ. Il existe en fait trois tailles de caractères sélectionnes grâce au mot clé MODE.

Essayez. Revenez au "hello" en forçant la mise à zéro (CTRL/SHIFT/ESC, vous vous rappellez?). Puis tapez.

Mode 0 **[ENTER]**

Vous pouvez remarquer que la taille des caractères a doublé. A présent, faites:

Mode 1 **[ENTER]**

Vous êtes maintenant revenu à la taille de départ. A présent, faites:

Mode 2 **[ENTER]**

Vous obtenez maintenant des caractères diminués de moitié. Nous avons donc trois modes:

- Mode 0: 20 caractères par ligne.
- Mode 1: 40 caractères par ligne.
- Mode 2: 80 caractères par ligne.

A noter que seule la largeur change et que l'écran compte toujours 25 lignes de caractères.

Le mot clé "mode" peut également modifier les graphiques de l'écran. Mais pour ce qui concerne ces derniers, nous ne parlons pas de caractères mais de pixels.

Le pixel est le plus petit point que vous puissiez tracer sur l'écran. Nous avons vu que l'écran graphique est de 640 points sur 400, mais la largeur du pixel change selon le mode:

- Mode 0: 4 points de large
- Mode 1: 2 points de large
- Mode 2: 1 point de large

Retour en position

Revenons donc aux précoordonnés. Nous verrons plus loin comment placer le texte et les nombres sur l'écran mais pour l'instant, observons les graphiques. Le programme suivant s'appelle COORGEOM, mais avant de le changer, voici un mot clé trés utile: CAT.

CAT est le diminutif de catalogue et sert à trouver les noms des programmes de la data-cassette. Rembobinez la cassette jusqu'au début en enfonçant le bouton REW puis faites:

cat **[ENTER]**

Le CPC464 répond en émettant:

Press PLAY and then any key ■
(appuyez sur PLAY avant de continuer)

C'est exactement comme si vous veniez de commander LOAD ou RUN, mais au lieu de charger un programme, le CPC464 va afficher un message signifiant qu'il a trouvé:

DRAW block 1 $

Les programmes sauvegardés sont rangés dans les blocs de 2000 caractères. Les programmes longs peuvent occuper de nombreux blocs individuels. Le CPC464 ne se contente pas d'indiquer sur l'écran un message spécial pour chaque bloc, il veille également à ce qu'il n'y ait pas d'erreur d'enregistrement et inscrit OK en fin de ligne.

Si vous demandez au CPC464 de charger un programme en mentionnant son nom, vous obtenez également des messages "found" (trouvé) pour les programmes sauvegardés préalablement mais aucune vérification n'est enterprise.

Vous devez avoir trouvé le programme COORGEOM: prennez en connaissance.

Jeu No. 3: Bombardier

Et voici le jeu que vous attendiez! C'est l'occasion d'affronter un vaisseau spatial extra-terrestre. Transmettez les précoordonnés à votre robot-bombardier et éjectez la bombe droit sur votre cible, sinon vous êtes un homme mort.

Test

Si vous avez survécu à cette bataille, vérifiez vos progrès en passant le SAT4 avant de poursuivre.

5 EXECUTION D'UN DESSIN

DRAW

PLOT

Le mot-clé "PLOT" constitue la première commande en Basic que nous allons étudier en détails. Jusqu'à présent, vous avez transmis des messages comme RUN ou BORDER sans savoir s'il s'agissait de commandes ou s'ils devaient suivre des règles précises.

Lorsque vous composez des commandes, vous avez souvent besoin d'ajouter des "arguments" au mot-clé afin de détailler l'opération envisagée. Pour le cas du PLOT, les arguments donnent la position désirée sur l'écran comme spécifié par les coordonnés x et y. Par exemple, faites:

plot 319,199 **[ENTER]**

Vous obtenez un point jaune d'un pixel presque exactement en milieu d'écran. Si vous ne vous rappelez pas laquelle est la ligne x et laquelle est y, souvenez vous de ce petit conseil: vous devez franchir la porte d'une maison avant d'en grimper les escaliers, soit gauche-droite avant haut-bas. Après exécution de la commande PLOT, le CPC464 laisse le curseur des symboles graphiques sur les coordonnés x et y sauf contre-ordre.

Passons à présent à une autre commande:

draw 0,0 **[ENTER]**

La commande DRAW a la même structure que PLOT, mais les arguments indiquent le point à partir duquel la ligne doit être tracée, spécifié bien sûr par les coordonnées x et y. La ligne diagonale qui paraît sur l'écran démarre au centre (x=319, y=199) et descend jusqu'au point gauche en bas (x=0, y=0). Maintenant, essayez de tracer le même carré qeu nous avions au chapitre précédent. Voici les commandes:

move 300,150 **[ENTER]**
draw 350,150 **[ENTER]**
draw 350,200 **[ENTER]**
draw 300,200 **[ENTER]**
draw 300,150 **[ENTER]**

La commande MOVE place le curseur des symboles graphiques sur les précoordonnés x et y spécifiés par les arguments sans rien ajouter sur l'écran.

Le programme du carre

MOVE

Jusqu'à présent, nous avons transmis des ordres exécutés directement par le CPC464. Voyons maintenant comment faire entrer et ranger un programme qui sera exécuté ultérieurement. mais avant cela, il nous faut débarrasser le CPC464 en faisant:

new **[ENTER]**

NEW

Pour la mémoire du CPC464, l'opération équivaut à effacer un tableau à l'aide d'une éponge mouillée. Même si l'écran ne s'efface pas, NEW fait disparataire tout programme préalablement chargé ou seulement passé. Transmettez cet ordre uniquement si vous ne risquez pas de perdre un programme précieux!

Donc en route avec notre programme. Il s'agit de nouveau du même carré, avec la seule différence que quelques nombres ont été ajoutés aux lignes:

```
10 clg
20 move 300,150
30 draw 350,150
40 draw 350,200
50 draw 300,200
60 draw 300,150
```

La ligne 10 constitue une nouvelle commande à votre disposition. La commande CLS efface l'écran et déplace le curseur-textes au coin en

CLG

LIST

haut à gauche; la commande CLG efface également l'écran mais deplace le curseur-graphiques au coin en bas à gauche (x=0, y=0). La ressemblance frappante entre ces deux commandes vous étonne peut-être. Cependant, dans la deuxième partie du cours, vous apprendrez à traiter textes et graphiques en même temps sur l'écran et vous comprendrez toute leur utilité.

Passez les six lignes de ce programme exactement comme indiqué et appuyez sur ENTER à la fin de chaque ligne.

Les numéros de ces lignes vous sont indispensables pour informer le CPC464 de l'ordre de vos commandes. Sauf avis contraire, il exécutera les commandes dans cet ordre. Les numéros de lignes montent par dizaines, ceci pour pouvoir insérer d'autres lignes si nécessaire. L'ordre dans lequel vous les introduisez importe peu. En fait, si vous vous trompez pour une de ces lignes et ne vous en apercevez qu'aprés avoir appuyé sur ENTER, vous pouvez effacer la ligne incorrecte en mémoire simplement en la réinscrivant.

Lorsque tout vous semble aller, passez le programme en faisant:

run **[ENTER]**

Joli, n'est-ce pas? Le CPC464 a gardé votre programme en mémoire et seulement au moment où vous le lui avez commandé (run). Et il est bien là: vous pouvez vérifier en faisant:

list **[ENTER]**

Comment changer de couleur

Les trois commandes de symboles graphiques que nous venons de voir ont un argument supplémentaire optionnel, l'INK (encre). Essayez de réinscrire la ligne 50 du programme du carré comme suit:

```
50 draw 300,200,3
```

Lorsque vous exécutez le programme cette fois, le côte gauche ainsi que le haut du carré vont être rouges. ceci s'explique par le 3 que nous avons ajouté; en effet, il a passé l'information suivante: la commande DRAW doit être exécutée en utilisant INK numéro 3. Le CPC464 poursuivra avec cette encre jusqu'à ce qu'un nouvel ordre de couleur soit transmis.

Le choix des couleurs pouvant être spécifié dans la commande dépend du mode utilisé.

Voici le maximum des différentes encres pour chaque mode:

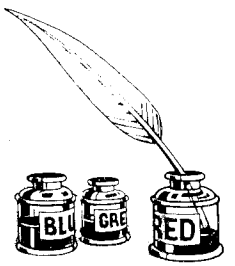
- Mode 0 - 16 encres
- Mode 1 - 4 encres
- Mode 2 - 2 encres

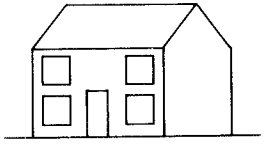
A présent, essayez d'inscrire ce qui suit:

```
ink 3,0 [ENTER]
```

Vous pouvez immédiatement remarquer que la ligne rouge devient noire. Faites ce que vous avez fait pour BORDER au chapitre précédent et essayez d'autres couleurs pour le numéro 3 de INK. Vous pouvez aussi tenter de changer les INKS (encres) 0,1 et 2.

INK





REM

PAPER

Maison

Le programme suivant de la datacassette A s'appelle HOUSE (maison). Il utilise toutes les commandes déjà apprises plus deux. La première est REM (remark). Lorsque le CPC464 reçoit cette commande, il ignore le reste de la ligne. Ceci vous permet d'ajouter des commentaires et des explications aux programmes afin que vous et les autres puissiez comprendre complètement le programme (si vous l'avez oublié depuis quelques temps).

L'autre commande nouvelle est PAPER; elle vous permet de spécifier la couleur de fond sur la fenêtre de l'écran. L'argument pour PAPER doit être le numéro de l'une des INKS spécifié pour le mode en cours. Si vous avez changé INK au moment où cela vous était suggéré, vous avez pu voir qu'il s'agissait de la commande automatiquement sélectionnée pour le PAPER lorsque le CPC464 a été mis en marche pour la première fois.

Avant de passer le programme, lisez attentivement la liste ci-aprés.

```
10 REM drawing a house (dessiner une maison)
20 MODE 0
30 CLS
40 REM ** start ** (début)
50 Border 12
60 INK 0,12:REM Yellow (jaune)
70 INK 1,3:REM red (rouge)
80 INK 2,6:REM bright red (rouge vif)
90 INK 3,9:REM green (vert)
100 PAPER 0
110 REM draw front (dessiner la façade)
120 MOVE 100,50 (déplacement)
130 DRAW 100,250,1
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM draw side (dessiner le coté)
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM draw gable end (dessiner le pignon)
240 REM already at start point (déjà au point de départ)
250 REM so no need for a MOVE (donc pas besoin de se déplacer)
260 DRAW 500,350
```

```
270 DRAW 600,250
280 DRAW 400,250
290 REM draw roof (dessiner le toit)
300 REM only two lines needed (seulement deux droites)
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM draw door (dessiner la porte)
350 MOVE 225,50
360 DRAW 225,140,2
370 DRAW 275,140
380 DRAW 275,50
390 REM draw windows (dessiner les fenêtres)
400 REM left hand bottom (fond partie gauche)
410 MOVE 120,70
420 DRAW 120,130,3
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM left hand top (dessus partie gauche)
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM right hand top (dessus partie droite)
```

```
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
570 DRAW 320,170
580 REM right hand bottom (fond partie droite)
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
```

Test

Peut-être souhaitez-vous reparcourir le chapitre avant de passer le test SAT5. Nous avons vu jusqu'à présent beaucoup de choses trés importantes en peu de temps, mais l'heure est venue d'aborder la programmation proprement dite.

6 CHIFFRES, LETTRES ET MOTS

PRINT

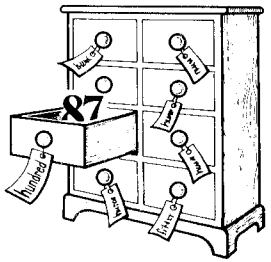
LET

Le temps est venu d'apprendre quelques mots-clé qui permettent d'afficher nombres et mots sur l'écran. Commençons par PRINT (impression). Essayez de taper sur le clavier:

print 2+2 **[ENTER]**

Comme vous le devinez, la réponse paraît à la ligne suivante, à savoir 4 bien sûr.
A présent essayez:

print "Hello" **[ENTER]**



Dans ce cas, nous devons inscrire le mot entre guillemets. La raison à cela vous paraîtra évidente plus tard. En attendant, passons à un autre mot-clé.

Letting

Quelquefois, le monde de l'ordinateur peut sembler vraiment bizarre. Les mots-clé du Basic sont des mots anglais normaux mais leur signification peut changer pour le CPC464. L'un de ces mots-clé est LET. En algèbre ordinaire, vouz pouvez écrire:

Let x=5

Cependant, vous ne pouvez pas écrire:

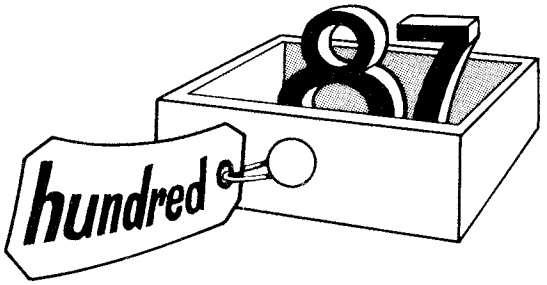
Let x=x+5

En Basic, cela passe sans problème. Cela signifie "prendre l'ancienne valeur de x, ajouter 5 et considérer ce chiffre comme la nouvelle valeur de x". Mais pourquoi x? Eh bien il s'agit d'une variable. Et cette caractéristique va s'avérer trés utile. Les variables peuvent contenir tout ce que vous souhaitez. Rien en Basic ne vous

empêchera d'écrire:

Let hundred = 87

Si vous n'aimez pas le mot "hundred" (cent), vous pouvez utiliser le h ou le c puisqu'il ne s'agit là que d'étiquettes. Lorsque pour définir une variable vous utilisez le mot LET, le CPC464 se comporte comme s'il écrivait le nom sur une boîte vide pour y glisser ensuite la valeur donnée aprés le signe "=".



Essayez de taper cela sur votre CPC464:

Let hundred = 87 **[ENTER]**

Vous obtiendrez la réponse "Ready" (prêt). Puis tapez:

print hundred **[ENTER]**

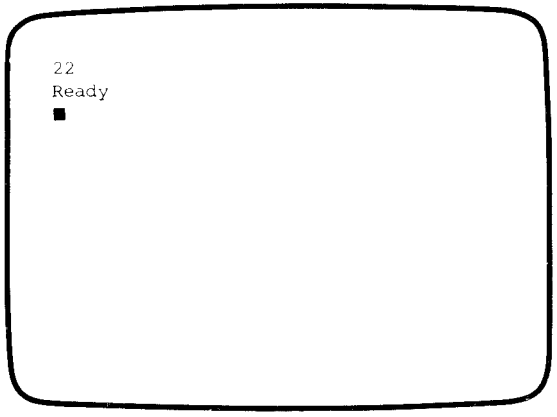
Vous obtiendrez alors sur votre écran:

```
Ready
let hundred=87
Ready
print hundred
87
Ready
■
```

A présent, nous pouvons donc passer au second programme. Les cinq lignes suivantes doivent être tapées exactement comme ceci:

```
10 cls
20 let a=15
30 let b=7
40 let a=a+b
50 print a
```

N'oubliez pas d'appuyer sur la touche ENTER à la fin de chaque ligne. A present, essayez de passer ce programme. Le résultat suivant devrait être obtenu:



Ceci peut paraître évident, mais pas ce qui suit:

```
10 let a=5
20 let b=10
30 let a=b
40 let b=a
50 print a+b
```

Essayez pour vous même. Pouvez vous dire pourquoi la réponse est 20? Pour vous aider, pensez à des nombres transférés d'une à l'autre. Rappelez-vous que c'est la variable située à gauche du signe "=" qui change et que le CPC464 va parcourir le programme une ligne après l'autre.

Variables alphanumeriques

Nous avons vu que nous pouvions définir des variables par des nombres et les utiliser en tant que telles. Le Basic permet également de définir des nombres pouvant contenir une série de lettres, chiffres et caractères spéciaux. On appelle ces variables des variables alphanumériques.

La seule différence notable entre une variable alphanumérique et une variable ordinaire réside dans le signe "$" que vous devez ajouter à la fin de votre variable alphanumérique. N'oubliez jamais de mettre le caractère qu'elle doit contenir entre guillemets, sinon le CPC464 croirait qu'il s'agit d'un autre type de variable.
Passez:

let a$=Hello **[ENTER]**

Vous obtiendrez le message-erreur:

Type mismatch

Le CPC464 pensait que vous essayiez de faire entrer dans la variable a$ la valeur numérique d'une autre variable appelée "hello". Vous auriez dû faire:

let a$="Hello" **[ENTER]**

Cette fois-ci, vous pouvez taper:

print a$ **[ENTER]**

Essayez.

A présent passez le programme suivant; assurez vous du résultat avant d'introduire le message:

```
10 let a$="5"
20 let b$="12"
30 let a=5
40 let b=12
50 let c=a+b
60 let c$=a$+b$
70 print c
80 print c$
```

Qu'y a-t-il dans un nom?

Vous pouvez vous servir de n'importe quel nom comme variable à l'exception d'un mot-clé du Basic. Par exemple, le CPC464 n'acceptera pas:

let save=s+p **[ENTER]**

Vous obtiendrez le message "syntax error". Il vous faudrait transformer ce nom en "savers" ou "savings" par exemple. Le guide de l'utilisateur se termine par une liste complète des mots-clé, vous pouvez la consulter pour savoir quels mots éviter.

Vous avez probablement remarqué que le CPC464 accepte les commandes inscrites en haut ou en bas de l'écran. Mais quelque soit l'emplacement de votre message, l'ordinateur présentera toujours ses mots-clé en haut de l'écran lorsqu'un programme est imprimé. Si vous conservez tous les noms des variables en bas de l'ecran, il vous sera facile de les extraire d'une liste.

Sauvegarde

Jusqu'à présent, vous avez chargé des programmes soit à l'aide des deux datacassettes fournies avec ce cours, soit en les passant sur le clavier. Vous allez maintenant pouvoir sauvegarder le programme en cours pour la postérité ou, plus exactement, pour pouvoir le réutiliser dans ce chapitre et dans le suivant sans avoir besoin de le recommencer depuis le début. Sortez la datacassette du datacorder et remplacez-la par une cassette vierge: les cassettes ordinaires suffisent mais ne vous procurez pas une bande de plus de 60 mn car elle serait trop fine. Veillez également à ce que la tête de cassette soit de bonne qualité.

Rembobinez la cassette jusqu'au début et passez le message suivant:

SAVE

save "variables" **[ENTER]**

Si vous le souhaitez, vous pouvez utiliser un autre nom que "variables" et ajouter des espaces entre les mots. Cependant, n'oubliez jamais les guillemets. Le CPC464 répond:

Press REC and PLAY and then any key:
(appuyer sur REC et PLAY avant de continuer)

Suivez les instructions, vous obtiendrez alors le message suivant:

Saving VARIABLES block 1
(sauvegarde des variables en bloc 1)

Le CPC464 démarrera la lecture de cassettes et vous entendrez le transfert de votre programme sur la cassette. Pendant ce temps, le curseur disparaît de l'écran mais revient lorsque l'opération est terminée et que le CPC464 envoit le message "ready". Il est prudent de sauvegarder deux exemplaires du programme en cas d'accident, aussi répétez ce que vous venez d'exécuter afin de sauvegarder une deuxième fois. N'oubliez pas de relâcher la touche REC après l'opération.

Impression: Quelques details supplementaires

Si vous venez de passer le programme ci-dessus, le CPC464 conserve toujours les variables en mémoire. Sinon, passez et chargez le programme une autre fois afin que nous puissions utiliser les variables pour étudier un ou deux détails supplémentaires au sujet de la commande PRINT.
Passez:

print a,b,c, **[ENTER]**

Vous voyez que les trois nombres figurent sur la même ligne de l'écran mais qu'ils sont séparés de 13 lignes les uns des autres. Cette façon agréable d'imprimer les chiffres marche cependant uniquement pour les modes 1 et 2. Essayez la même chose pour le mode 0 et vous comprendrez pourquoi. Si vous ne souhaitez pas d'espace entre les nombres, frappez:

print a;b;c **[ENTER]**

Cette fois, les nombres figurent sur la même ligne mais ne sont pas espacés du tout, aussi nous devons souvent ajouter des espaces, comme pour l'exemple suivant:

print "the value of c is ";c;" not ";c$

Nous pouvons placer chiffres et lettres où nous voulons sur l'écran grâce au mot-clé "LOCATE". La structure de commande est:
locate x,y

Cela vous paraît déjà connu, n'est-ce pas? Poutant, méfiez-vous; les précoordonnés x et y n'ont rien à voir avec les précoordonnés de graphiques. LOCATE déplace le curseur-textes vers la position donnée par les arguments de la commande. Les précoordonnés de texte débutent au coin supérieur gauche de l'écran (dont les précoordonnés sont x=1 et y=1) et sont comptés de gauche à droite et de haut en bas de l'écran. Faites:

75 locate 20,13 **[ENTER]**

Débarrasez l'écran à l'aide de la touche CLS. En repassant ce programme avec cette nouvelle ligne, vous pouvez voir que la valeur de c$ est imprimée en milieu d'écran et commence à la position 20, ligne 13.

Histogramme

Le programme suivant de la datacassette est appelé histogramme. Il vous représente visuellement quatre nombres entre 0 et 290 (un peu comme des résultats d'élection ou d'enquête d'opinion). Ce type de programme doit attendre à certains moments que les nombres passent en mémoire par le biais du clavier avant de continuer. Le mot-clé pour cela est INPUT, et voici un exemple de commande-type:

INPUT

```
1Ø INPUT name$
```

Le CPC464 attend patiemment à la ligne 10 jusqu'à ce que vous inscriviez quelque chose et que vous appuyiez sur la touche ENTER. L'information est alors introduite dans la variable alphanumérique "name$" et le processus continue.

A présent, passez BARCHART (histogramme) avant d'étudier la liste ci-aprés. Vous apercevez un point d'interrogation (?) suivi du curseur: le CPC464 attend le INPUT. Le mot "prompt" apparaît également sur l'écran afin que l'utilisateur sache à quel genre d'information le CPC464 s'attend. Faites ceci en inscrivant un message après le mot-clé INPUT (donnée d'entrée) comme suit:

```
INPUT "Amount (1-29Ø)";a
```

Le message de "prompt" doit être placé entre guillemets (") et séparé de la variable par un point-virgule (;).

Vous pouvez constater que nombre des lignes du BARCHART (histogramme) comprennent plusieurs commandes séparées par deux points (:). Le but de ces deux points équivaut a commencer une commande avec le nombre d'une ligne et la terminer par ENTER. C'est une bonne façon de garder une série de commandes ensemble, surtout si elles sont trés courtes.

Voici un très bon usage des deux points:

```
5Ø b=50:REM bar size
```

Ajouter un REM à une commande pour en expliquer sa fonction rend la lecture du programme plus facile plus tard.

```
10 Rem 3D Bar Chart (histogramme)
20 REM by Dave Atherton
30 MODE 1
40 BORDER 14:INK 0,14:INK 1,0:INK 2,3:INK 3,24
50 b=50:REM bar size
60 LOCATE 1,23:INPUT "Amount (1-290)" ;a
70 x=100:PLOT x,55,1
80 DRAW x-b*2,55:DRAW x-b*2, a+55
90 DRAW x-b,a+55+b:DRAW x+b,a+55+b
100 DRAW x,a+55:DRAW x-b*2,a+55
110 MOVE x+b,a+b+55:DRAW x+b,b+55
120 DRAW x,55:DRAW X,55+a
130 LOCATE 1,23
140 PRINT"
150 LOCATE 1, 23: INPUT "Amount (1-290)";a
160 x=260:PLOT x,55,2
170 DRAW x-b*2,55:DRAW x-b*2,a+55
180 DRAW x-b,a+55+b:DRAW x+b,a+55+b
190 DRAW x,a+55:DRAW x-b*2,a+55
200 MOVE x+b ,a+b+55:DRAW x+b,b+55
210 DRAW x,55:DRAW x,55+a
220 LOCATE 1,23
230 PRINT"
240 LOCATE 1,23:INPUT "Amount (1-290)";a
250 x=420:PLOT x,55,3
260 DRAW x-b*2,55:DRAW x - b*2,a+55
270 DRAW x-b, a+55+b:DRAW x+b,a+55+b
```

```
280 DRAW x,a+55:DRAW x-b*2,a+55
290 MOVE x+b, a+b+55:DRAW x+b,b+55
300 DRAW x,55:DRAW x,55+a
310 LOCATE 1,23
320 PRINT''
330 LOCATE 1,23:INPUT  "Amount (1-290)";a
340 x=580:PLOT x,55,1
350 DRAW x-B*2,55:DRAW x-b*2,a+55
360 DRAW x-b,a+55+b:DRAW x+b,a+55+b
370 DRAW x,a+55:DRAW x-b*2,a+55
380 MOVE x+b,a+b+55:DRAW x+b, b+55
390 DRAW x,55:DRAW x,55+a
```

Jeu No.4

Certaines personnes utilisent des mots qui peuvent paraître trés savants, surtout dans le monde de l'ordinateur. Notre générateur automatique BUZZWORD va vous aider à les maîtriser.

Test

Passez le SAT6 afin de vérifier si vous avez besoin de relire certains points du chapitre avant de poursuivre.

7 MISE AU POINT

Le contenu de ce chapitre comprend principalement des corrections et des changements de programmes. L'idéal sera de corriger le programme introduit au chapitre précédent. Pour cela, rembobinez la cassette et rechargez-le en mémoire en faisant:

load "variables" **[ENTER]**

"Variables": Il s'agit du nom que vous avez donné au programme lorsque vous avez utilisé le mot-clé SAVE pour le sauvegarder dans la datacassette.

Comment changer de lignes

Lorsque vous introduisez un message par le biais du clavier, vous êtes presque sûr de faire des fautes, même si vous vous contentez de copier une page imprimée. L'erreur la plus courante réside dans la frappe de mauvaises touches. En général, vous vous en apercevez tout de suite, aussi, avant d'appuyer sur la touche ENTER, reculez et effacez les caractères non désirés avec la touche DEL.

Lorsqu'une ligne est en train d'être tapée, (autrement dit ligne en cours), vous pouvez déplacer le curseur vers l'avant ou vers l'arrière sur cette ligne en utilisant les touches du curseur. Vous pouvez introduire de nouveaux caractères et vous pouvez en faire disparaître grâce aux touches CLR et DEL.

Le chapitre précédent vous a également montré que vous pouviez modifier des lignes complètes simplement en les recommençant. Le CPC464 inscrit automatiquement la nouvelle ligne sur l'ancienne.

Tout en inscrivant vos propres programmes, vous vous apercevez qu'ils ne passeront pas toujours correctement au départ. Le CPC464 vous signalera vos erreurs lorsque vous taperez vos lignes et lorsque vous essayerez de passer le programme. Mais il ne pourra vous signaler, par exemple, que vous avez oublié certaines commandes ou omis de déplacer le curseur sur une autre position avant d'inscrire une ligne. Voilà pourquoi les nombres de lignes que nous avons utilisés s'ajoutent par dizaines. Comme vous l'avez remarqué au chapitre précédent, nous avons ajouté une ligne entre 70 et 80 en faisant:

```
75 locate 20,13
```

Dans les cas extrêmes, vous ne pouvez plus vous arrêter d'ajouter jusqu'à 9 lignes à une position quelle qu'elle soit. De la même façon, vous pouvez effacer des lignes en introduisant une ligne-espace. La ligne que nous venons d'ajouter peut disparaître de la façon suivante:

```
75 [ENTER]
```

Le CPC464 élimine alors la ligne 75 du programme. Essayez et repassez le programme (LIST) pour vérifier ce qui s'est passé.

Editing

Lorsque vous avez besoin de modifier un programme déjà introduit ou chargé en mémoire, vous pouvez vous servir de la fonction EDIT; ceci grâce à l'exemple que nous venons de voir:

EDIT

```
edit 40 [ENTER]
```

Comme vous le remarquez, la ligne 40 est affichée sur l'écran et le curseur est placé sur le premier caractère de la ligne. Faites alors appel aux touches-flèche du curseur (gauche et droite) afin de positionner le curseur sur la partie à modifier comme si vous travaillez sur une ligne en cours; essayez vous-même en remplaçant la valeur de b (15) par 26. Placez le curseur sur le 1 de 15 et appuyez deux fois sur la touche CLR. Le 15 est affacé. A présent, inscrivez 26 et pressez ENTER. Le curseur n'a pas besoin de se trouver en fin de ligne. A présent, si vous éditez (LIST) ou passez (RUN) le programme, vous pouvez remarquer que le CPC464 a modifié la ligne de la façon désirée.

Un moyen plus simple d'éditer des lignes consiste à utiliser la touche COPY. Un deuxième curseur appelé le curseur copy est utilisé pour extraire des lignes ou des parties de lignes placées n importe où sur l'écran.

Vous pouvez obtenir le curseur-copy en maintenant la touche SHIFT enfoncée et en appuyant sur l'une des touches du curseur. Editez de

nouveau le programme (LIST) et essayez, en plaçant le curseur-copy au début d'une des lignes. Vous pouvez voir que le curseur normal n'a pas bougé. Appuyez sur la touche COPY pour copier chaque caractère sur la ligne en cours avant de reprendre.

Une fois habitué à vous servir du curseur-copy, vous n'utiliserez surement plus beaucoup la commande EDIT. Le résultat est le même, à la seule différence que vous avez l'avantage de pouvoir débarrasser une ligne de caractères que vous ne voulez plus.

Let

Frapper LET à chaque début de ligne au chapitre précédent a pu vous paraître astreignant. Le temps est venu de s'expliquer. Vous n'avez pas besoin de ce mot-clé dans le Basic d'Amstrad. Essayez de rééditer VARIABLES afin d'éliminer tous les LET. Repassez le programme modifié et vous verrez que cela ne fait aucune différence!

Comment sauter des lignes

Nous avons vu plus haut que les numéros des lignes permettent au CPC464 de connaître l'ordre de rangement des commandes; mais nous avons aussi remarqué qu'elles n'étaient pas forcément rangées dans cet ordre. Quelquefois, il arrive que nous voulions revenir à un point ou aller plus avant dans le texte. Le mot-clé à utiliser pour cela est GOTO (se déplacer vers). On émet la commande en ajoutant un numéro de ligne; par exemple:

```
7Ø goto 13Ø
```

Cette ligne ordonne au CPC464 de passer de la ligne 70 directement à la ligne 130. Vous pouvez faire de même à l'envers:

```
13Ø goto 7Ø
```

Le programme retourne alors à la ligne 70 en passant par chaque ligne au-dessus de 130 jusqu'à ce que vous arretiez ou appuyiez sur la touche ESCAPE.

Et apres?

Nous sommes appelés à émettre des décisions aussi banales soient-elles à chaque heure de notre vie (comme le choix du thé et du café, ou celui des chaussures à porter avant de sortir). Une fois le choix fixé entre le thé et le café ou rien du tout ou entre les chaussures noires ou les chaussures rouges ou rien du tout, nous agissons en fonction de notre décision. Le CPC464 dispose de champs d'action alternatifs par le biais du mot-clé IF.

GOTO

IF

La ligne suivante pourrait appartenir à un programme visant à vérifier le montant de la somme d'argent que vous avez épargné, la variable "money" (argent) étant la balance courante:

```
IF money=Ø THEN PRINT "hard up"
```

THEN

Il est interessant de noter que l'ordre PRINT sera uniquement exécuté si "money" est égal à zéro. Sinon, le CPC464 continue à la ligne suivante du programme.

Faites la même chose d'une façon encore plus astucieuse en introduisant un autre mot-clé, ELSE de la manière suivante:

ELSE

```
IF money>Ø THEN PRINT "Rich" ELSE PRINT "Hard up"
```

Le signe ">" signifie "supérieur à" ou "plus que"; vous pourriez savoir si vous devez bien de l'argent. Si "money" (argent) est inférieur ou égal à zéro, le CPC464 ignore ce qui est inscrit entre THEN et ELSE et exécute ce qui est inscrit après ELSE. ELSE vous offre une alternative avant de passer à la ligne suivante du programme.

Nous avons étudié un programme appelé BARCHART au chapitre précédent; une phrase disait:

```
Amount (Ø-29Ø)?
```

Si vous avez essayé d'introduire un nombre supérieur à 90, le programme n'a pas tenu compte de votre ordre. Afin d'éviter cela, vous pouvez limiter la variable à sa valeur maximum. Par exemple:

```
IF a>29Ø THEN a=29Ø
```

Comme vous le voyez, la commande IF...THEN ne se limite pas à PRINT et la valeur de a peut toujours être inscrite sur l'écran.

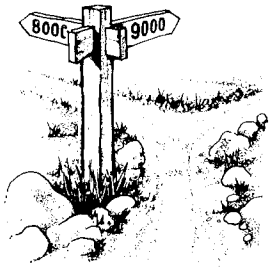
Deplacements

Lorsque vous avez recours à GOTO, le mot-clé IF peut vous être très utile. Il agit comme un panneau indicateur pour guider le CPC464 vers la partie du programme par laquelle il doit poursuivre. A supposer que vous vouliez que l'on s'arrête d'introduire des nombres supérieurs à 295 dans le programme BARCHART, les lignes suivantes se présenteront de la manière ci-dessous:

```
60 INPUT "Amount (0-290)";a
65 IF a> 290 GOTO 60
```

Jusqu'à ce qu'un nombre inférieur ou égal à 290 soit introduit, le CPC464 peut se diriger au mauvais endroit à la moindre erreur de votre part. Nous voici arrivés à point nommé à un mot classique du jargon-ordinateur:

Le bug



Les machines font des fautes, les hommes commettent des erreurs. Nous avons déjà dit que les ordinateurs ne pensent pas et que le programmeur doit établir ses propres lignes de pensée; s'il se trompe, et cela peut très bien arriver aux meilleurs d'entre eux, il ne peut s'en rendre compte qu'une fois le programme passé dans l'appareil. Le CPC464 exécute les ordres reçus même si ceux-ci ne constituent pas toujours ce que le programmer avait prévu.

Le "de-bugging" est le processus qui consiste à parcourir un programme en en corrigeant les erreurs de logique ou de compréhension. Ne vous sentez pas honteux si vous introduisez un programme contenant des bugs (erreurs sur le ROM). Bien sur, les programmeurs chevronnés rencontrent moins souvent des bugs dans leurs programmes, mais il ne s'agit là que d'une question de connaissances et de pratique. Vous vous rendrez bientot compte que les bugs se montrent de moins en moins souvent dans vos programmes.

Vérifier son programme avant de le passer constitue un bon entraînement. On peut le faire par "contrôle sur papier" en inscrivant sur une feuille toutes les valeurs et variables ainsi que les produits des sommes.
Prenons un exemple:

```
10 a=0
20 print a
30 a=a+1
40 if a<4 then goto 20
```

Avant d'introduire ceci dans le CPC464, incrivez sur papier les trois titres suivants:

Step Line number Val. of a

Parcourez à présent le programme et exécutez chaque instruction exactement comme le ferait le CPC464. Vous devriez arriver à:

Séquence	*Numéro de ligne*	*Valeur de a*
1	*10*	*0*
2	*20*	*0*
3	*30*	*1*
4	*40*	*1*
5	*20*	*1*
6	*30*	*2*
7	*40*	*2*
8	*20*	*2*
9	*30*	*3*
10	*40*	*3*
11	*20*	*3*
12	*30*	*4*
13	*40*	*4*

Cette technique du contrôle-papier vous force à comprendre le fonctionnement de la machine, vous pouvez vous rendre compte des problèmes, beaucoup plus facilement. Lorsque vous venez d'écrire un programme, votre plus grande difficulté peut résider dans le fait que vous savez exactement ce qui doit suivre et ne pouvez comprendre pourquoi les choses ne se passent pas comme prévu!

Une autre technique consiste à ajouter des instructions en PRINT à certains points du programme afin de voir la valeur de la variable à ce point. Par exemple, ajoutons la ligne

suivante au programme ci-dessus:

```
35 print "Line 35 a= ";a
```

Vous pouvez également avoir recours à la commande STOP. Par exemple:

```
35 stop
```

Le CPC464 interrompera le cours d'exécution du programme à ce point et vous pouvez lui ordonner d'imprimer la variable ou les variables qui vous interessant grâce aux commandes d'impression directe. On revient au début du programme en faisant:

```
cont [ENTER]
```

Cela signifie CONTinuer; n'oubliez pas que cela ne marche pas si des lignes sont ajoutées ou effacées après l'arrêt (STOP) du programme.

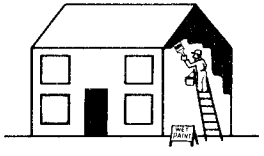
Renovation

Oui, nous voici de nouveau à la maison. Cette fois-ci, elle aurait bien besoin d'une couche de peinture par-ci par-là. Cherchez la carte des couleurs et chargez le programme DECO. Il ne s'agit pas d'un jeu mais plutôt d'une façon amusante d'utiliser toutes les commandes apprises jusqu'ici. Vous souhaitez peut-être modifier votre programme pour lui donner une touche plus personnelle. Il vous suffit pour cela de mettre en pratique les techniques étudiées en début de chapitre.

STOP

Si vous rencontrez des mots-clé inconnus dans le listing du DECO présenté aux prochaines pages, ne vous inquiétez pas: tout vous paraitra plus clair trés bientôt.

CONT



```
10 REM deco
20 MODE 0
30 CLS
40 REM ** START **
50 BORDER 12 (bord 12)
60 INK 0,12:REM YELLOW (encre 0, rem jaune)
70 INK 3,3:REM red (rouge)
80 INK 6,6:REM bright red (rouge vif)
90 INK 9,9:REM green (vert)
100 PAPER 0 (papier)
110 REM draw front (dessin de la façade)
120 MOVE 100,50 (Déplacement)
130 DRAW 100,250,3
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM draw side (dessin du coté)
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM draw gable end (dessin du bout du pignon)
240 REM already at start point (déjà en position de départ)
250 REM so no need for a MOVE (donc pas besoin de déplacement)
260 DRAW 500,350
270 DRAW 600,250
280 DRAW 400,250
```

```
290 REM draw roof (dessin du toit)
300 REM only two lines needed (besoin de seulement deux lignes)
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM draw door (dessin de la porte)
350 MOVE 225,50
360 DRAW 225,140,6
370 DRAW 275,140
380 DRAW 275,50
390 REM draw windows (dessin des fenetres)
400 REM left hand bottom (partie en bas à gauche)
410 MOVE 120,70
420 DRAW 120,130,9
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM left hand top (partie en haut à gauche)
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM right hand top (partie en haut à droite)
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
```

```
570 DRAW 320,170
580 REM right hand bottom (partie en bas à droite)
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
640 REM *** DECO ***
650 r$=CHR$ (18)
660 LOCATE 1,25
670 PRINT "Type a colour (1-15)"; (frappe d'une couleur)
680 FOR i=1 TO 15
690 INK i,i:NEXT i
700 LOCATE 1,1:PRINT r$;
710 INPUT "Roof colour";r (couleur du toit)
720 FOR i=107 TO 399 STEP 2
730 MOVE i,252:DRAW i+96,349,r
740 NEXT i
750 LOCATE 1,1:PRINT r$;
760 INPUT ""gable end'';g (bout du pignon)
770 FOR i=252 TO 346
780 MOVE i+154,i:DRAW 848-i,i,g
790 NEXTi
800 LOCATE 1,1:PRINT r$;
810 INPUT "End wall";e (bout du mur)
820 FOR i=52 TO 248 STEP 2
830 MOVE 404,i:DRAW 598,i,e
840 NEXT i
```

```
850 LOCATE 1,1:PRINT r$;
860 INPUT "front";f (façade)
870 FOR i=52 TO 248 STEP 2
880 MOVE 104,i:DRAW 398,i,f
890 NEXT i
900 LOCATE 1,1:PRINT r$;
910 INPUT "Door";d (porte)
920 FOR i=52 TO 138
930 MOVE 229,i:DRAW 268,i,d
940 NEXT i
950 LOCATE 1,1:PRINT r$;
960 INPUT "Window";w (fenêtre)
970 FOR j=0 TO 100 STEP 100
980 FOR i=70+j TO 130+j
990 MOVE 120,i:DRAW 180,i,w
1000 MOVE 320,i:DRAW 380,i
1010 NEXT i
1020 NEXT j
1030 END
```

Test

Passez le SAT7 afin de vérifier votre compréhension de ce chapitre. Ne vous effrayez pas si vous voyez que vous avez besoin de consulter les pages précédentes avant de répondre aux questions. Bon nombre de programmeurs utilisant eux-mêmes un manuel de référence.

8 DES AMENAGEMENTS DANS LA MAISON

Dans la programmation-ordinateur existent des tâches que l'on doit constamment répéter. Le CPC464, comme tout ordinateur, aime beaucoup les répétitions. Lorsque nous lui donnons le programme voulu, il le répètera jusqu'à ce qu'on lui ordonne de s'arrêter. Revenons donc à la maison afin de voir certains détails sans avoir à passer par de nombreuses programmations.

Chargez le programme suivant sur la cassette, MANSION. Apparement, il s'agit de la même vieille maison, mais cette fois nous allons voir si nous pouvons produire des affichages visuels trés recherchés en utilisant les principes de la séquence-boucle et du "sous-programme".

Procedure boucle

Aussi surprenant que cela puisse vous paraître, vous avez déjà eu affaire à une boucle au chapitre précédent. Même si nous ne l'avons pas encore appelé ainsi, le programme suivant contient une boucle:

```
10 a=0
20 print a <----------
30 a=a+1             |
40 if a<4 then goto 20 (si s<4 retourner au 20)
```

Ce programme signifie qu'il faut imprimer (print) la valeur de a, revenir à la ligne 20 (refermer une boucle) et la refaire jusqu'à ce que a soit supérieur à 3.

Nous aurions pu composer le programme comme suit:

```
10 for a=0 to 3
20 print a
30 next a
```

Si vous vous attardez sur le listing du programme MANSION imprimé plus loin, vous verrez qu'il ne se termine plus à la ligne 640. Nous allons tout d'abord observer les nouvelles lignes, de 640 à 680 inclus.

Si vous passez le programme, vous apercevez la maison comme d'habitude. Lorsqu'il atteint la commande STOP, vous pouvez passer le morceau suivant en faisant:

CONT **[ENTER]**

Essayez. Que pensez vous de la grille? Le chien du voisin ne peut même plus entrer dans le jardin!

La clé de cette opération se trouve à la ligne 650; mais regardons plutôt:

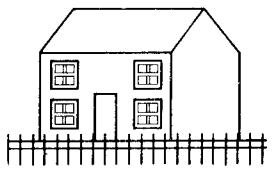
650 FOR F=0 TO 620 STEP 20

la commande FOR signale au CPC464 qu'il est sur le point d'exécuter une boucle. F=0 signifie que la séquence-boucle commence. "TO 620" indique où elle se termine. L'instruction STEP 20 signale la distance entre les grilles en suggérant au CPC464 d'augmenter "F" de 20 à chaque fois qu'il exécute une boucle.

La ligne 660 utilise la valeur en cours de la variable F pour déplacer le curseur de graphiques vers l'emplacement de la grille suivante et tracer la ligne de cet emplacement.

L'instruction qui achève la boucle est NEXT, comme vous pouvez le voir à la ligne 670. Cela signifie: Donnez moi la nouvelle valeur de F", ceci en utilisant le "step" décrit plus tôt.

Donc, en résumé, le CPC464 parcourt une boucle qui va de F=20, 40, 60, 80, etc..... jusqu'au nombre suivant qui est 640, soit la fin de la boucle passée. Le CPC464 arrive alors à NEXT (suivant) et exécute la ligne No680 pour dessiner les barreaux horizontaux de la grille.



FOR

STEP

NEXT

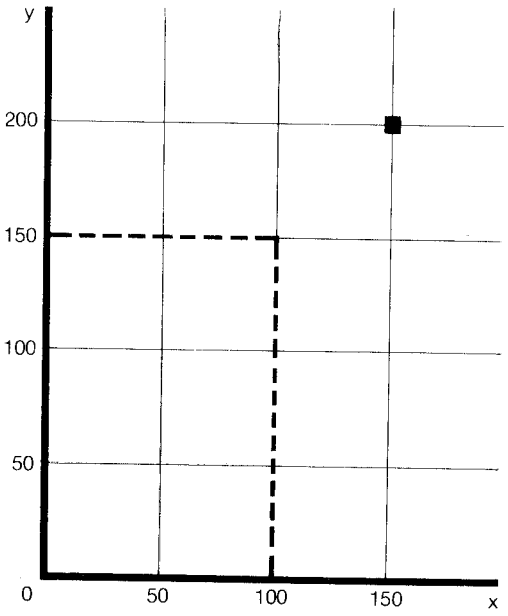
PLOTR

MOVER

DRAWR

Relativite

Avant de passer à l'autre morceau, terminons juste notre étude commandes de graphiques avec les versions relatives de PLOT, DRAW et MOVE. Vous vous souviendrez que les arguments des 3 commandes sont les précoordonnés x et y mesurés à partir de l'origine des graphiques, soit 0,0.



Pour les commandes PLOTR, DRAWR ET MOVER, les arguments sont les déplacements désirés de x et y en commençant par la position en cours du curseur de graphiques. Sur le schéma ci-dessus, vous pouvez voir que l'instruction suivante déplacerait le curseur-graphique sur x=150, y=200:

mover 5Ø,5Ø **[ENTER]**

Même chose pour les commandes DRAWR et PLOTR. Leurs arguments concernant toujours les précoordonnés relatifs en partant de la position du curseur en cours et en évitant au programmeur de recalculer les précoordonnés à chaque fois. Leur utilité va prendre toute son importance avant la fin du chapitre.

Execution des fenetres

Bien sûr, vous aurez sûrement remarqué que nous n'en avons pas encore terminé avec la maison. Après la boucle du dessin de la grille figure une autre commande STOP à la ligne 685, suivie de quelques autres lignes qui comprennent une nouvelle commande: GOSUB. Passez à la fin du programme en inscrivant CONT.

Eh bien, il était temps de garnir les fenêtres de vitres; l'opération a été accomplie par "sous-programmation", terme relatif à la programmation-ordinateur utilisé pour une suite d'instructions auxquelles on fait souvent appel (autant de fois nécessaire). Si nous n'avions pas réalisé de sous-programmation pour les vitres, il aurait fallu répéter plus de 16 fois les mêmes instructions. GOSUB signifie "passez (GO) à la sous-programmation (subroutine)" et est toujours suivi du numéro de ligne et de la première instruction de sous-programmation.

GOSUB

```
10 REM mansion (maison)
20 MODE 0
30 CLS
40 REM ** start ** (début)
50 BORDER 12 (bord)
60 INK 0,12:REM yellow (encre jaune)
70 INK 3,3:REM red (rouge)
80 INK 6,6:REM bright red (rouge vif)
90 INK 9,9:REM green (vert)
100 PAPER 0 (papier)
110 REM draw front (dessin de la façade)
120 MOVE 100,50
130 DRAW 100,250,3
```

```
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM draw side (dessin du côté)
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM draw gable end (dessin du bout du pignon)
240 REM already at start point (déjà au point de départ)
250 REM so no need for a move (donc inutile de se déplacer)
260 DRAW 500,350
270 DRAW 600,250
280 DRAW 400,250
290 REM draw roof (dessin du toit)
300 REM only two lines needed (besoin de seulement deux lignes)
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM draw door (in red) (dessin de la porte, en rouge)
350 MOVE 225,50
360 DRAW 225,140,6
370 DRAW 275,140
380 DRAW 275,50
390 REM draw windows (in green) (dessin des fenêtres, en vert)
400 REM left hand bottom (partie en bas à gauche)
410 MOVE 120,70
```

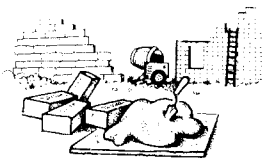
```
420 DRAW 120,130,9
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM left hand top (partie en haut à gauche)
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM right hand top (partie en haut à droite)
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
570 DRAW 320,170
580 REM right hand bottom (partie en bas à droite)
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
635 STOP
640 REM fence (grille)
650 FOR F=0 TO 620 STEP 20
660 MOVE F,0:DRAW F,60,15
670 NEXT F
680 MOVE 0,45:DRAW 620,45
```

```
685 STOP
690 REM frames in windows (cadres des fenêtres)
700 REM use a square drawing subr (séquence)boucle du dessin d'un carré)
710 size=18
720 MOVE 130,78:GOSUB 900
730 MOVE 156,78:GOSUB 900
740 MOVE 130,103:GOSUB 900
750 MOVE 156,103:GOSUB 900
760 MOVE 130,178:GOSUB 900
770 MOVE 156,178:GOSUB 900
780 MOVE 130,203:GOSUB 900
790 MOVE 156,203:GOSUB 900
800 MOVE 330,78:GOSUB 900
810 MOVE 356,78:GOSUB 900
820 MOVE 330,103:GOSUB 900
830 MOVE 356,103:GOSUB 900
840 MOVE 330,178:GOSUB 900
850 MOVE 356,178:GOSUB 900
860 MOVE 330,203:GOSUB 900
870 MOVE 356,203:GOSUB 900
880 END (fin)
890 REM subroutine for square (sous-programmation pour carré)
900 DRAWR 0,size,9
910 DRAWR size,0
920 DRAWR 0,-size
930 DRAWR -size,0
940 RETURN
```

A présent, vous comprenez pourquoi nous avons dû aborder les commandes de graphiques relatives avant d'aller plus avant dans le chapitre Si nous avons recours à une sous-programmation standard pour exécuter une tâche particulière, et dans le cas présent pour le tracé d'un carré, un moyen doit être trouvé pour décrire cette tache dans un langage universel. Pour ce qui concerne notre sous-programmation de vitres aux fenêtres, vous pouvez voir que l'utilisation de la commande DRAWR et de la variable "size" (taille) nous permet de dessiner un carré de n'importe quelle taille et ce sur n'importe quelle partie de l'écran.

On devrait aborder une sous-programmation avec la commande REM au moins une fois afin de pouvoir décrire ce qu'elle exécute. Il serait sot de ne pas le faire pour un long programme contenant de nombreuses sous-programmations. Nous verrons au chapitre 9 toute l'importance de la question et apprendrons quelles autres informations doivent être données. Quoi qu'il en soit, vous devez terminer une sous-programmation par la commande RETURN.

Lorsque le CPC464 en vient à une commande GOSUB, il indique où il en est dans le programme avant d'exécuter les commandes de sous-programmation. Le RETURN à la fin de la sous-programmation ressemble un peu à l'instruction NEXT pour le cas d'une boucle FOR, mais tandis que NEXT vous ramène au début de la séquence-boucle, RETURN renvoie le CPC464 à l'instruction qui suit le GOSUB qui avait ordonné la sous-programmation.



Un programme peut se servir de sous-programmations pour dessiner une maison de la même façon qu'un entrepeneur fait appel à des sous-traitants pour en construire une. Les maçons, menuisieurs, plombiers et carreleurs doivent exécuter des tâches spécifiques. Et ces spécialistes peuvent employer d'autres personnes afin qu'elles exécutent une partie de leurs travaux. Par exemple, un maçon aura certainement besoin d'un ouvrier pour qu'il lui prépare son mortier car il y a longtemps qu'il ne le fait pas lui-même. De plus, ce même ouvrier peut aider tout autre spécialiste.

De la même façon, vous pouvez réaliser des programmes. Une sous-programmation commandée par GOSUB peut avoir recours à une autre sous-programmation pour la seconder dans une partie de son travail, etc...Le CPC464 est en mesure de suivre la trace de nombreux niveaux de GOSUB et saura toujours où renvoyer une de ces sous-parties.

RETURN

Comment terminer

Les sous-programmations figurent toujours à la fin d'un programme. Mais que se passe-t-il une fois la dernière commande exécutée? Eh bien si vous ne prenez aucune précaution, le programme passe directement au début de la première sous-programmation et le résultat peut s'avérer malencontreux sous ses apparences plutôt comiques. Mais pour éviter une telle éventualité, passez une commande END. Cela est très simple, comme vous pouvez le voir dans MANSION:

END

```
800 END
```

Lorsque le CPC464 aboutit à ce genre de ligne, il interrompt le cours du programme et retourne à READY.

A présent voici une autre façon d'introduire une séquence-boucle:

```
800 GOTO 800
```

Le programme continuera son cours jusqu'à ce que vous appuyiez soit sur la touche ESC soit sur la touche qui rembobine complètement la cassette, cela est utile-vous ne voulez pas voir apparaitre READY sur l'écran.

Exercices

Essayez d'utiliser la sous-programmation de la fenêtre pour placer une grande fenêtre panoramique sur l'extremité droite de la maison face au jardin.

Composez une sous-programmation permettant de dessiner une cheminée puis utilisez la pour placer cette cheminée sur la gauche, la droite ou le milieu du toit.

Test

Cette fois, vous serez non seulement interrogé sur des points de ce chapitre, mais vous aurez également à effectuer certaines révisions d'autres chapitres étudiés. Vous aurez le bon sens de reparcourir ce manuel afin de trouver vos réponses. Il n'est bien entendu pas question de se rappeler tout ce qui a été vu jusqu'à présent en si peu de temps.

9 ANALYSE DE PROGRAMMES

Il n'est pas exagéré d'affirmer que la programmation peut recouvrir une infinité de besoins. Souvenez-vous de ce que nous avons déjà mentionné plus d'une fois: les ordinateurs ne pensent pas. Le travail de programmation d'un ordinateur consiste en l'analyse minutieuse d'un problème ou d'une tâche et en la conversion de l'information obtenue en un programme logique et cohérent qui n'utilise pas trop d'espace-mémoire et qui parvient assez rapidement à ses fins.

Bien entendu, inutile de vous attendre à de tels résultats sans notions ni pratique. Cependant, ce chapitre peut vous aider à acpuérir certaines habitudes et techniques de bonne programmation qui vous aideront pour les années à venir.

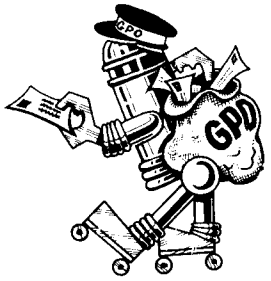
Travaux a partir d'objectifs fixes

Pour tout programme de plus de 20 lignes, il n'existe qu'une façon d'aborder une tâche. Vous devez le découper en morceaux faciles à consulter. Quelque soit votre niveau, amateur ou professionnel, vous ne pouvez pas garder en tête tous les différents aspects d'un programme long en même temps.

Aussi, lorsque vous commencez un programme, vous devez avant tout le diviser en plusieurs parties et prendre note des caractéristiques de chaque partie. Il vaut la peine de démarrer en parallèle en "livre-projets" qui vous rappellera tous les paramètres de chaque programme et sous-programme. Sinon, vous pourriez oublier à mi-chemin la raison pour telle ou telle exécution, voire la spécification exacte du but de la sous-programmation. A partir d'une description générale d'un programme et de ses objectifs, vous devez le découper en une suite logique et ordonnée de tâches.

Et comment parvenir à cela? Eh bien vous disposez de plusieurs méthodes, mais nous allons seulement vous en expliquer une. Nous allons avoir recours au Programming Development Language (PDL, programmation d'un langage de développement). Aidons nous d'un exemple connu: un facteur doit prendre un certain nombre de décisions et executer un certain nombre d'actions lors de sa tournée. Imaginez que vous avez créé un robot-facteur qui doive faire la même chose. Comme les robots sont controlés par ordinateur, vous devez leur communiquer des ordres en permanence. Mais jetez donc un coup d'oeil sur les lignes suivantes:

Programme du robot-facteur




```
Dirigez vous vers la première maison de la rue
For (pour) pour chaque maison de la rue
      If (si) s'il y a quelque chose pour cette maison
      Then repeat (alors répétez l'opération)
            Sortez votre paquet du sac
            Until (jusqu'à) ce que vous ayez tout en main
            pour cette maison précise
            If (si) si vous rencontrez un portail
            Then (alors) alors ouvrez-le
                  Franchissez-le
                  Refermez le portail
            While (pendant) pendant votre trajet
                  Dirigez vous vers la porte
            If (au cas où) au cas où vous devez remettre une
            lettre recommandée, une enveloppe taxée ou un colis
            Then (alors) sonnez
                  Attendez la réponse
                  For (pour) pour chaque paquet
                        In case (en cas de):
                              Lettre recommandée: recueillez
                              la signature
                              Enveloppe taxée: réclamez
                              l'argent
                              Othwise (sinon): remettez
                              votre paquet
            Else (autrement): glissez tout dans la boite à
            lettres.
            While (pendant) pendant votre trajet
                  Dirigez vous vers l'entrée
            If (si) s'il y a un portail
            Then (alors) alors ouvrez le
                  Franchissez-le
                  Refermez le portail
            Else (autrement) autrement dirigez vous vers la
      Next (suivante) maison suivante.
```

Tout d'abord, on peut remarquer que le programme est doté d'un petit nombre de blocs de composition. Il s'agit pour certains d'entre eux de commandes en Basic, mais pas pour d'autres. Vous pouvez également remarquer les alinéas du texte qui indiquent les actions qui se font à l'intérieur de la séquence-boucle ou à partir des blocs IF THEN ELSE (si, alors, sinon). Faites bien attention à cela lorsque vous composez vos propres programmes de cette façon ou vous risquez de vous perdre!

Le terme de "routine" mérite peut-être une explication: toute suite d'actions (ou commandes) qui exécutent une tâche particulière sont regroupées ensemble pour former une "routine". Observez la deuxième ligne du programme:

For (pour) pour chaque maison de la rue

Ceci constitue le début d'une routine, il peut être répété pour n'importe quelle maison dans n'importe quelle rue, du moment que le programme s'occupe du reste du problème.

Entrant dans cette routine plusieurs sous-routines (ou sous-programmations): vous avez là la véritable signification du terme. En voici un exemple très typique:

While (pendant) pendant votre trajet, dirigez vous vers la porte.

Vous pouvez voir qu'une sous-programmation peut s'occuper de la marche à partir de n'importe quel point jusqu'au point de destination. Soulignons cependant que l'on ne s'attarde pas à détailler minutieusement chaque action à cette étape. Si vous continuez de découper ce programme, vous pourriez voir que le seul mot "dirigez" représente des pages et des pages de travail. Mais à ce niveau, une description simple et générale suffit.

Lorsque toutes les tâches sont successivement passées en revue, nous pouvons aboutir au point final où toutes les actions du programme correspondent à des actions individuelles se rapportant au capteur du robot ou aux mécanismes de contrôle. Si vous trouvez tout ceci assomant, lisez plutôt ces lignes réconfortantes: la rédaction du programme que l'on vient de voir nécessite plusieurs années de travail soutenu pour une équipe moyenne de programmeurs. C'est pourquoi vous pouvez l'éliminer de votre liste de projets en toute légitimé. Nous avons seulement choisi cet exemple pour vous montrer à quel point les tâches les plus complexes peuvent être exprimées simplement.

Exercices

Plusieurs paramètres font défaut dans le programme du robot. Vérifiez si vous pouvez créer des sous-programmes pouvant résoudre les problèmes suivants:

- Pas de réponse au son de la sonnerie
- Absence de boîte à lettres
- Bras trop chargés pour ouvrir le portail
- Rues comportant uniquement les noms des maisons

Vous pouvez sûrement penser à d'autres problèmes. Ne vous découragez pas si vous ne parvenez pas à les résoudre tous. Ce que l'être humain peut accomplir sans efforts peut s'avérer extrêment compliqué en soi.

Creation de blocs

Voici quelques blocs de notre programme de facteur ainsi que des exemples de leur traduction en langage Basic Amstrad.

For (pour) pour chaque maison de la rue
Next (suivante) maison suivante

```
800 FOR house = firsthouse TO last house (de la première à la dernière maison)
.....
870 NEXT
```

Repeat (répéter l'opération)
Sortez le paquet du sac
Until (jusqu'à) ce que vous ayez tout en main pour cette maison

```
1000 GOSUB 12000: REM get packet from bag
.....
1110 IF waslastpacket=0 THEN GOTO 1000
```

Cela suppose que la sous-programmation de la ligne 12000 fixe la variable "waslastpacket".

While (pendant) votre parcours
Dirigez vous vers la porte

```
2000 GOSUB 13000:REM see if we are at the door (vérifier qu'on est bien arrivé à
la porte).
2010 IF atdoor <> 0 then 2040
2020 GOSUB 14000:REM walk towards door
2030 GOTO 2000
2040 REM end of while
```

La sous-programmation de la ligne 13000 définit la variable à "atdoor" (à la porte).

In case: (au cas où):
Lettre recommandée: Recueillez une signature
Enveloppe taxée: Réclamez l'argent
Otherwise (autre cas): Remettez le paquet

```
3000 GOSUB 15000:REM get type of packet (type de paquet)
3010 IF packettype=recd THEN GOSUB 16000:GOTO 3040
3020 IF packettype=excess THEN GOSUB 17000:GOTO 3040 (enveloppe taxée)
3030 GOSUB 18000:REM hand over packet (remettez le paquet)
3040 REM end of "in case" (fin de "au cas où").
```

La sous-programmation à la ligne 15000 définit la variable "packettype" et les variables "recd" et "excess" ont été définies aux valeurs appropriées.

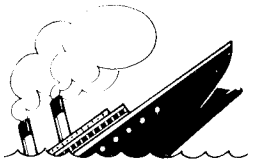
If (si) si vous avez une lettre recommandée, une enveloppe taxée ou un colis
Then (alors) alors sonnez
Else (sinon) sinon glissez tout dans la boite à lettres

Voici une traduction possible:

```
4000 GOSUB 10000 :REM see if recorded/excess/parcel
.....
4070 IF needring<> 0 THEN GOSUB 20000 ELSE GOSUB 21000
```

Peut-être souhaitez vous garder les routines d'origine proches des commandes IF-THEN-ELSE pour plus de clarté, plutôt que d'utiliser des "sous-routines", la traduction serait donc:

```
4000 IF needring =0 THEN GOTO 4100
.....REM get answer etc. (réponse)
4090 GOTO 4200
4100 REM don't need to ring (inutile de sonner)
4200 REM end of "IF needring" (fin de IF "vous devez sonner")
```



Travail de routine

Voici tout ce que vous devez mettre au point d'écrire une routine:

- Nom de la routine (pour vous même et non pour le CPC464).
- Noms des variables qui doivent être définies avant l'usage de la routine.
- Impact des routines sur les variables.
- Impacts secondaires de la routine.

Au cours de vos programmes, vous devriez utiliser des feuilles de papier séparées afin de les relier plus tard pour former votre livre de projets. Inscrivez en haut de chaque feuille les renseignements ci-dessus. Vous en tirerez deux avantages:

1 Vous pourrez voir en un clin d'oeil ce que vous avez l'intention d'écrire.

2 Il vous sera plus facile d'identifier deux sous-programmations presque identiques afin que vous les combiniez au lieu d'accomplir deux fois la même chose.

Vous souvenez vous de la routine qui a permis de dessiner des pans de vitres sur les fenêtres? Elle commençait par un REM.

```
REM subroutine for square
```

La seule variable qui entre en jeu avant l'intervention de la sous-programmation est "size" (taille), et la sous-programmation utilise simplement la variable mais ne la change pas. ASu bout du compte, la routine n'a eu aucun impact sur les autres parties du programme et le CPC464 lui-même.

Une telle méthode de création des routines les rend "inattaquables". Vous connaisez exactement les démarches à suivre pour les utiliser, vous connaisez les services qu'elles peuvent vous rendre ainsi que leur impact. Il est ainsi très facile d'éliminer les bugs (erreurs sur le ROM) de tels programmes, sans compter qu'une telle conception vous évite la mauvaise surprise de voir tout l'ensemble de votre programme se perdre dans les profondeurs... et couler en catastrophe aussi allègrement que l'a fait un beau jour le Titanic.....

La création des programmes par cette méthode de routine implique également que lorsque vous aurez terminé, vous vous retrouverez avec votre morceau de papier qui vous dira exactement quoi écrire.

Documentation

En tant que programmeur amateur, vous travaillez probablement en solitaire. Vous allez donc avoir tendance à vous sentir très fier de vos programmes et éprouver une certaine réticence à les divulguer ou à accepter les critiques d'autrui. Pour les programmeurs professionels, il en va tout autrement; la plupart d'entre eux travaillent en équipe sur un programme donné et doivent donc être capables de lire et de comprendre tout ce que les autres ont écrit sans difficultés.

Le langage Basic est accessible et souple, il n'insiste sur aucune structure ou forme rigide d'un programme. Bien sûr, tout cela est parfait lorsque vous débutez, mais dès que les tâches deviennent un peu plus complexes, il apparaît nécessaire de réécrire des programmes que vous aviez terminé un mois plus tôt simplement parce que vous ne vous y retrouvez plus ou parce que vous ne parvenez pas à les modifier.

Le secret de tous les succès reside dans la commande REM- REM encore et toujours. Lorsque votre programme comporte une bifurcation ou un GOSUB, composez simplement le REM pour exprimer la raison et le but de ce cheminement. Vous pouvez faire GOSUB 4000 par exemple, sans problèmes jusqu'à ce que vous trouviez cinquante autres sous-programmations. Lorsque c'est le cas, vous devez écrire (il s'agit bien sûr une fois de plus d'un exemple):

GOSUB 4000:REM Move cursor to position
(mettre le curseur en position)

Nous avons essayé de définir le rôle d'une routine et les variables qui entrent en jeu. Vous pouvez le faire à l'intérieur d'un programme en introduisant l'information requise dans une liste de REM en début de chaque routine. Cela n'empiète pratiquement pas sur le temps du traitement et la lecture du tout n'en parait que facilitée, pour vous comme pour d'autres programmeurs.

10 SUPER LE SON

Lorsque nous avons signalé que les programmes de Basic conçus pour le CPC464 ne marchaient pas forcément sur les autres ordinateurs, nous pensions en particulier aux commandes du son. La portée et la puissance des commandes de son du Basic CPC464 sont telles que peu d'ordinateurs sont en mesure de l'égaler.

Mais ceci peut également créer un problème. Il serait tout à fait possible de rédiger un second manuel destiné uniquement aux commandes de son. Si bien que comme ce chapitre doit se contenter de vous faire acquérir les notions de base du langage Basic Amstrad, il se voit obligé de passer à côté de bien des renseignements complémentaires.

Ceci dit, chargez le programme suivant de la cassette. Il se nomme ZAPPOW et vous donnera une idée de tout ce que l'on peut réaliser à partir de ces commandes de son. Une liste de ce programme se trouve à votre disposition à la fin de ce chapitre pour que vous puissiez emprunter les parties qui vous plaisent et composer vos propres programmes.

Mise au point

Comme vous l'avez sûrement déjà deviné, la commandes de son s'appelle "SOUND". Faites:

sound 1,478 **[ENTER]**

C'était le Do. Le nombre 478 s'appelle le "point" et détermine la note qui va être jouée. Une liste complète ainsi que leurs associations de points sont à votre disposition dans le Guide de l'Utilisateur de l'Amstrad CPC464. On utilise le son "canal" 1. La deuxième partie de ce cours explique comment utiliser les deux autres canals.

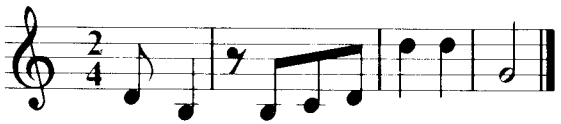
sound 1,478,200 **[ENTER]**

Cette fois-ci, lorsque vous appuyez sur la touche ENTER, on entendra le son pendant exactement deux secondes. Le numéro 200 définit la durée du son par centaines de secondes. Si vous omettez cette partie de la commande, le CPC464 suppose que la durée souhaitée est de 20 secondes. A présent, vous souhaitez peut-être composer un programme pour écouter un air connu. Par exemple:

```
10 rem A Familiar Tune
20 sound 1,213
30 sound 1,253,60
40 sound 1,0,40
50 sound 1,253
60 sound 1,239
70 sound 1,213
80 sound 1,127,40
90 sound 1,0,1
100 sound 1,127,40
110 sound 1,159,60
120 end
```

Observez les lignes 60 et 90: pour obtenir un "silence" (terminologique musicale pour pause) entre deux notes, vous devez passer une commande dont le point est zèro-en d'autres termes. Il s'agit d'un silence. A la ligne 40, nous avons un silence à deux temps. A la ligne 90, le silence est extrêmement bref afin que la note répétée ne soit pas confondue avec sa soeur jumelle.

Un Air Connu

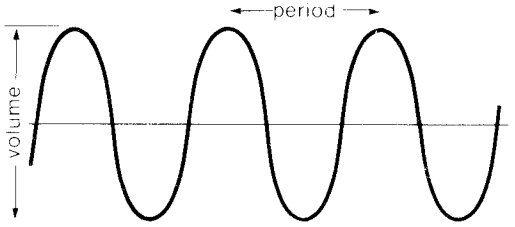


Ces notes qui reproduisent l'air du programme sont destinées à ceux d'entre vous qui maîtrisent un peu la musique.

SOUND

Les sons du BASIC

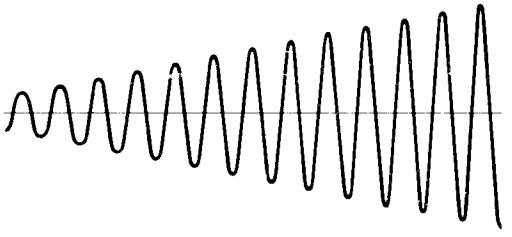
Avant de poursuivre, regardons un peu comment les sons fonctionnent. Dans sa forme la plus simplifiée, un son peut être représenté graphiquement par une courbe sinusale comme celle-ci:



A présent, vous savez ce que la partie "durée" du son signifie exactement. C'est le temps qui s'écoule entre deux courbes de son, mesuré toutes les huit microsecondes (une microseconde = un millionième de seconde). Plus le temps écoulé entre deux courbes est court, plus le son est aigu, et plus le temps est long, plus le son est grave. La hauteur de ces courbes est appelée volume, et nous pouvons le spécifier par notre commande son. Essayez de passer de nouveau notre ligne de Do, mais cette fois-ci, ajoutez un 2 à la fin comme suit:

sound 1,478,200,2 **[ENTER]**

Les choses se calment, n'est-ce pas? On peut spécifier le volume entre 0 et 7, 0 signifiant "pas de volume" (oui, cela peut être très utile) et 7 signifiant "volume maximum". Le son que nous venons de voir a commencé et s'est terminé par le même volume. Mais dans la réalité, cela ne se passe pas de la même façon. Il est tout à fait normal d'avoir affaire à un son qui change de volume en cours de route.

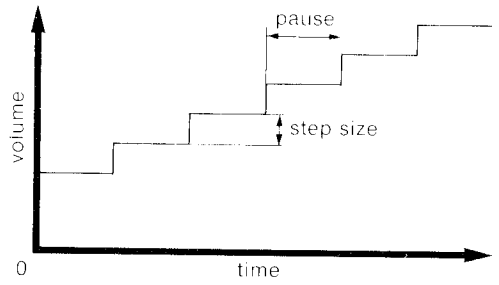


ENV

Nous pouvons obtenir ceci sur le CPC464 grâce à la commande ENV qui vous remet une "enveloppe" qui peut varier le volume d'un son tout au long de sa durée. Cpmposez les deux lignes suivantes et passez les:

```
10 env 1,6,1,30
30 sound 1,478,1800,1,1
```

Vous avez sûrement besoin d'étudier de près le prochain graphique pour comprendre comment cela marche. Ne vous inquiétez pas si vous remarquez que la ligne 20 manque. Nous y reviendrons en temps voulu.



Vous pouvez avoir à votre disposition jusqu'à 15 différentes enveloppes de volume dans votre programme et le premier No1 de la commande ENV ci-dessus l'identifie simplement comme la première enveloppe. Le numéro suivant, 6, signifie que vous voulez changer le volume en 6 registres identiques. Le chiffre suivant, 1, indique de combien vous voulez augmenter le volume. S'il s'agissait du chiffre-1, le volume diminuerait. Enfin, le 30 définit la longeur de chaque registre en centaines de secondes.

Faites attention au volume spécifié dans la commande SOUND. En fait, le CPC464 peut fixer le volume à 16 niveaux différents lorsque l'enveloppe de volume est spécifié. L'éventail s'échelonne alors entre 1 et 15 et les niveaux 0,2,4,6,8,10,12, et 14 correspondent aux registres de 0 à 7, registres que vous obtenez lorsque l'enveloppe de volume n'est pas spécifié.

A présent, observons comment les commandes SOUND et ENV fonctionnent ensemble dans notre exemple. Le volume est fixé à 1 dans la commande SOUND, et ENV1 offre six regsitres permettant d'augmenter le volume d'un niveau, de façon à ce que le volume final équivale à 1+6=7. Remarquez que le premier registre arrive dés la commande SOUND exécutée, aussi le volume initial est égal à 1+1=2. A noter également que la multiplication des six registres de l'enveloppe par la durée ne devrait pas prendre plus de temps que la durée spécifiée par la commande SOUND, sinon la fin de l'enveloppe nous échapperait.

A présent, penchons nous sur le problème de la ligne manquante No20. De la même manière que les niveaux de volume d'une commande SOUND sont modifiés par une enveloppe "volume", la fréquence d'un son peut être modifiée par une enveloppe "ton". Pour cela, utilisez la commande ENT. La ligne 20 est maintenant à sa place:

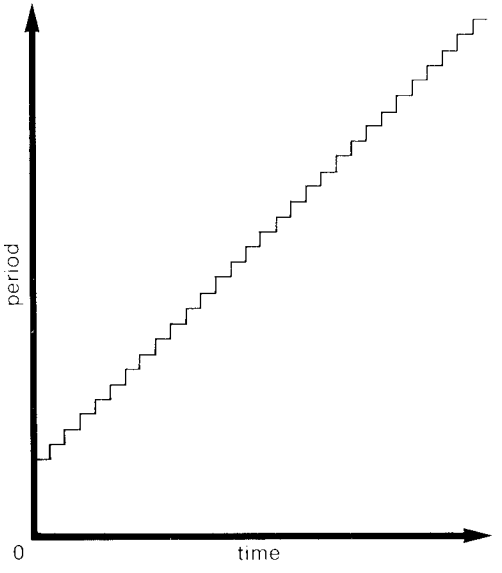
ENT

```
10 env 1,6,1,30
20 ent 1,180,1,1
30 sound 1,478,180,1,1,1
```

Comme vous le voyez, notre commande SOUND a déjà acquis un autre 1, ceci pour indiquer qu'il faut utiliser ENT1.

La structure de la commande ENT ressemble de prés à celle de la commande ENV, comme

vous pouvez le remarquer sur le graphique ci-dessous. Mais cette fois-ci, c'est le point du son qui se trouve augmenté registre par registre jusqu'à 180 registres, chacun durant un centième de secondes.



Si ce n'est pas déjà fait, passez le programme ci-dessus plusieurs fois. Puis changez la valeur de registre des commandes ENV et ENT pas -1 au lieu de 1, changez le volume initial en fixant la commande de son à 7 puis repassez le.

Quelques sons bruyants

Le CPC464 peut ajouter un petit bruit à chaque son pour l'enrichir. Mais peut-être pensez vous que la commande son est déjà truffée de surprises puisqu'on lui a ajouté les numérations ENV et ENT. Mais reste une dernière chose (cette fois-ci c'est bien vrai) que vous pouvez lui ajouter: il s'agit du bruit. Editez la ligne 30 de votre programme pour avoir:

```
30 sound 1,478,180,1,1,1,5
```

Le passage du programme à présent va vous paraître trés différent. Essayez de vous habituer aux 31 bruits distincts qui peuvent être spécifiés en insérant la numération exacte dans cette partie de la commande. Si vous le gardez interrompu ou si vous choisissez 0, vous n'obtenez aucun bruit.

Exercices

Ajoutez à l'air connu les enveloppes "ton" et "volume" puis changez les afin d'obtenir des sons différents. Puis ajoutez différents bruits à quelques unes des commandes du son.

Composez votre programme en reprenant quelques mesures d'un air connu. Même si vous ne connaissez pas grand chose en musique, vous pouvez y arriver avec quelques efforts supplémentaires. Si vous avez déjà quelques bases, ne soyez pas trop ambitieux au départ. Une berceuse suffira amplement pour démarrer.

Recreation

Eh bien il ne s'agit pas tout à fait de récréation. Le programme suivant de la datacassette s'appelle ORGAN. Vous pouvez non seulement reprendre des airs à partir du clavier, mais aussi analyser le programme pour comprendre comment il est crée. A présent le langage Basic doit vous être suffisamment connu pour vous permettre d'imprimer un programme ou comprendre ce qu'on attend de lui. cependant la programmation de celui-ci en particulier est assez élaborée. N'ayez pas peur d'en modifier le contenu pour observer ce qui se passe. Mais comme promis, voici la liste du ZAPPOW:

```
10 REM Zappow
20 REM by Dave Atherton
30 MODE 1
35 INK 0,1:INK 1,25
40 LOCATE 13,4:PRINT"SOUND DEMO"
50 LOCATE 10,7:PRINT"1. Explosion"
60 LOCATE 10,8:PRINT"2. Dog Barking" (aboiement de chien)
70 LOCATE 10,9:PRINT"3. Siren" (sirène)
80 LOCATE 10,10:PRINT"4. Toilet Flush" (chasse d'eau du cabinet)
90 LOCATE 10,11:PRINT"5. Cuckoo" (cocorico)
100 LOCATE 10,12:PRINT"6. Machine Gun" (arme à feu)
110 LOCATE 10,13:PRINT"7. Space Invader" (envahisseur de l'espace)
120 LOCATE 5,17:PRINT"Select a sound from 1 to 7" (choix d'un son de 1 à 7)
130 IF INKEY$="" THEN 130
140 A$=INKEY$:IF a$="" THEN 140
150 IF a$<"1" OR a$>"7" THEN 140
160 a=VAL(a$)
170 LOCATE 20,19:PRINT a$
180 IF a=1 THEN GOSUB 280
190 IF a=2 THEN GOSUB 330
200 IF a=3 THEN GOSUB 380
210 IF a=4 THEN GOSUB 430
220 IF a=5 THEN GOSUB 480
230 IF a=6 THEN GOSUB 530
240 IF a=7 THEN GOSUB 580
250 FOR J=0 TO 1000:NEXT
260 LOCATE 20,19:PRINT""
270 GOTO 130
```

```
280 REM Explosion
290 ENV 1,11,-1,25
300 ENT 1,9,49,5,9,-10,15
310 SOUND 1,145,255,0,1,1,12
320 RETURN
330 REM Dog Bark (aboiement de chien)
340 ENV 1,4,7,10
350 ENT 1,7,-8,3,6,24,2
360 SOUND 1,120,33,8,1,1,3
370 RETURN
380 REM Siren
390 ENV 1,2,9,45
400 ENT 1,2,9,45
410 SOUND 1,150,90,6,1,1
420 RETURN
430 REM Toilet Flush (chasse d'eau)
440 ENV 1,3,-2,85
450 ENT 1,5,-1,51
460 SOUND 1,150,254,11,1,1,8
470 RETURN
480 REM Cuckoo (cocorico)
490 ENV 1,4,12,11
500 ENT 1,5,12,8
510 SOUND 1,165,40,13,1,1
520 RETURN
530 REM Machine Gun
540 ENV 1,21,-5,4
550 ENT 1
```

```
560 SOUND 1,162,82,15,1,1,11
570 RETURN
580 REM Space Invader (anvahisseur de l'espace)
590 ENV 1,4,30,19
600 ENT 1,9,49,5,1,-10,26
610 SOUND 1,136,68,15,1,1,0
620 RETURN
```

Test

Pour être en mesure de répondre aux questions du SAT10, ne vous faites pas trop d'illusions: vous devrez à coup sûr reparcourir ce chapitre. Si vous aviez déjà en tête ce qui a trait aux commandes SOUND aux CPC464, vous ne seriez pas en train de lire ce manuel.

11 DETAILS SUR LES NOMBRES

Nous l'avons déjà observé plusieurs fois, le CPC464, comme tout autre ordinateur, excelle dans l'exécution des tâches répétitives. Mais il fait également ses preuves lorsqu'il aborde les nombres. En fait, c'est là qu'il se distingue vraiment. Tout ce que nous avons vu jusque là a été rendu possible grâce au traitement interne et ultra rapide d'une quantité de nombres. Même les fonctions graphique et son marchent de cette façon.

Si vous n'êtes pas très bon en maths, vous trouverez ce chapitre quelque peu ardu. Mais rien ne vous empêche d'essayer de le travailler. L'arithmétique simple ne doit pas vous poser trop de problèmes et le Basic présente les choses très clairement. Dans tout les cas, ce cours ne couvre pas les points les plus compliqués.

L'arithmetique Basic

Les opérations du CPC464 sont réalisées grâce aux quatre fonctions arithmétiques connues de tous. Elles sont représentées par des symboles différents:

Fonction	*Signification*	*BASIC*
+	Addition	+
–	Soustraction	–
x	Multiplication	x
÷	Division	÷

Donc les opérations en Basic vous paraitront un peu différentes de celles dont vous vous etiez habitué jusqu'à maintenant.

Au chapitre, nous avions écrit:

```
print 2+2
```

Vous pouvez faire la même chose avec les sommes suivantes en vous reportant à la liste des fonctions ci-dessus et en utilisant les symboles Basic adéquates:

```
3+7          17-8
15 x 4       15÷3
8÷2          20 x 17
```

Pour des opérations plus compliquées, nous

devons informer le CPC464 que certaines d'entre elles doivent être exécutées avant d'autres. La fraction suivante, par exemple, peut paraître parfaitement claire en arithmétique ordinaire:

15 x 7
7 – 2

En basic, ceci ne peut s'écrire:

15*7/7-2 (=13, faux!)

Nous devons signaler que la multiplication 15*7 et la soustraction 7-2 doivent être effectuées AVANT la division. Vous obtenez ceci en les mettant entre parenthèses comme ceci:

(15*7)/(7-2) (=21, juste!)

Vous devriez maintenant pouvoir calculer les opérations suivantes:

30	15 x 6
3+2	9
24 – (4 x 3)	5 x (2+8)

Ces règles s'appliquent aux nombres autant qu'aux variables. Passez au programme suivant et changez plusieurs fois la ligne 40 afin de pouvoir calculer les autres opérations ci-dessus. N'oubliez pas leurs symboles en Basic et placez les paranthèses au bon endroit.

```
10 a=2
20 b=5
30 c=10
40 print a+b+c
```

a x a	b x c
	b – a
c – (a x b)	
c	c – b
b – a	c + b

Les deux premières opérations sont assez claires, mais quel résultat avez vous l'intention d'obtenir pour les trois autres? Vous venez d'avoir un aperçu de la façon dont le CPC464 travaille les mathématiques. Pour les trois dernières opérations, si vous comptez les chiffres à gauche et à droite du point décimal, vous remarquez qu'ils sont toujours neuf. C'est la façon dont le CPC464 affiche normalement les nombres sur l'écran. Cependant ceci n'est pas parfait: quelquefois le résultat d'une suite de calculs ne sera pas absolument exact. Au lieu de 5,0, il peut vous arriver d'obtenir:

5,0000001

ou bien:

4,99999999

Subtil, n'est-ce pas? Heureusement, nous n'avons pas souvent besoin d'une telle précision. En

fait, nous voulons quelquefois uniquement les nombres entiers vu que la différence entre 276,25 et 276 est plutôt moindre. Pour obtenir un nombre entier, le mot-clé à utiliser est ROUND. Voici un exemple-type de commande:

ROUND

```
x=round (26*17)/1.6
```

Essayez. Si la partie droite du point décimal est inférieure à 0.5 (0.5), le résultat est arrondi au nombre entier inférieur. Si par contre cette partie est supérieure ou égale à 0.5, elle est arrondie au nombre entier supérieur.

Autre chose à ne pas oublier: le CPC464 peut afficher des nombres de 9 chiffres et garder d'autres chiffres en mémoire. Par exemple, vous pouvez vous retrouver avec la valeur suivante:

5.000000001 (10 chiffres)

Le CPC464 insistera sur le fait qu'il s'agit de 5.0 à chaque fois que vous le lui demanderez. Mais, chose importante, si la commande suivante ressemble à:

```
IF a=5 THEN GOTO ....
```

le GOTO ne sera jamais exécuté.

Afin d'éviter ce genre de problème, il existe une technique de programmation-standard qui consiste à ne jamais user du modèle ci-dessus mais plutôt du suivante:

```
vsmall=0.000001
IF ABS(a-5)<vsmall THEN GOTO ....
```

Ne vous laissez pas impressionner par le mot clé ABS. Il vous indique simplement la différence entre "a" et 5, comme vous le verrez en deuxième partie.

Le programme suivant de la datacassette A constitue un bon moyen d'avoir les tables de multiplication sous son nez, c'est aussi un exemple facile de mathématiques que propose le CPC464. Demandez lui de vous présenter la table de 23x87. Aprés avoir passé ce programme, il vaudrait la peine de l'imprimer sur l'écran du moniteur afin de voir à quoi il ressemble vraiment.

Logique elementaire

Nous avons mentionné bien des choses aux chapitres précédents sans vraiment rentrer dans les détails, simplement parce que nous ne voulions pas vous embrouiller avec trop d'explications en même temps. L'un de ces détails réside dans l'utilisation d'"opérateurs logiques".

Revenons à notre étude de séquences-boucles; nous nous sommes servis de signes tels "<" et ">". Il existe une liste complète de ces symboles appelés "opérateurs logiques".

< Inférieur
> Supérieur
= Egal à
< > Différent de<= Inférieur ou égal à
= Egal à
< > Différent de
<= Inférieur ou égal à
>= Supérieur ou égal à

Même si ces termes ne vous sont pas trés familiers, leurs signification, elle, est tout à fait claire. Lorsque vous devez comparer deux variables afin de savoir quand achever une séquence-boucle ou bien pour faire votre choix entre deux solutions, ces opérateurs vous donnent les moyens de bien choisir. Les commandes IF vues au chapitre 7 sont complètement liées opérateurs.

Vous vous devez de garder en mémoire une des subtilités des opérateurs logiques: il est important pour eux de savoir si un nombre est positif ou négatif, car si a=5 et b=3, alors:

a>b

Mais si a=-5 et b=3:

a<b

Les variables alphanumeriques

Vous souvenez vous que nous avons commencé par dire que le CPC464 ne se distinguait vraiment que par les nombres? Pourquoi alors, vous demandez vous, peut il ranger et traiter des caractères de variables alphanumériques, comme nous l'avons vu au chapitre six?

Eh bien chaque caractère est identifié par un nombre appellé son "code-caractère", si bien que les caractères sont uniquement considérés comme une suite de nombres. Ainsi, des opérateurs logiques peuvent tester des variables alphanumériques pour leur ordre alphabétique, "A" étant par hypothèse inférieur à "Z". (pas la valeur numérique du code-caractère). Mais passons plutôt à la commande suivante:

```
If "Apple"<"Orange" THEN PRINT "Lemon"
```

Le résultat est "lemon" puisque dans l'alphabet "apple" passe avant "orange". Toutes les lettres majuscules sont inférieures aux lettres figurant au bas des touches et une variable alphanumerique courte est inférieure à une variable alphanumerique plus longue commençant par les mêmes lettres:

"aardvark"<"abbey"	Vrai
"love"< "locksmith"	Vrai
"box"<"BOX"	Faux
"Zoo"< "Zookeeper"	Faux
"apple" "Apples"	Vrai

Dans ce cas, le CPC464 teste la valeur "numerique" de chaque lettre d'un mot et la compare avec la valeur numérique de la lettre située à la même place dans le mot se trouvant de l'autre coté de l'opérateur.

Des maisons et des jardins

La maison commence à prendre vraiment forme à présent. Elle est déjà garnie d'une fenêtre, d'une cheminée et d'une grille pour empêcher le chien des voisins de rentrer. Il ne vous reste plus qu'à entreprendre quelques travaux dans le jardin.

S'agissant d'un type de machine fonctionnel, ne comptez pas sur le CPC464 pour vous réaliser un beau parterre de fleurs. Cependant, il peut vous aider à confectionner un coin-jardinnage. Mais chargez plutôt le prochain programme à partir de la datacassette A. Il s'intitule GARDEN (le jardin). Le jardin mesure 6 mètres sur 4 et nous allons cultiver des rangs de 4 m de long.

La largeur de chacun de ces rangs dépendra du type de légume semé, tout comme le poids de chaque récolte par mètre carré. Le programme vous demandera combien de rangs vous souhaitez pour chaque sorte de légume et vous laissera savoir s'il vous reste encore de la place pour un rang ou deux.

```
10 REM Garden
20 MODE 1
30 INK 0,0:BORDER 0
40 INK 1,26
50 Length=6
60 CLS
```

En admettant que votre terre soit riche et que l'année soit bonne, vous disposez d'un tableau indiquant le poids prévisible de chaque légume. Pour ce programme, voici la liste ci-après. Vous pouvez remarquer que les calculs sont très simples et se basent sur le tableau suivant:

Légume	*Largeur du rang (en mètres)*	*Production par rang (in Kg)*
Oignons	0.30	9
Carottes	0.30	3
Pommes de terre	1.10	50
Choux	0.60	8
Haricots	1.10	30
Navets	0.50	11

(N.dT.: En Anglais, 0.30 équivaut à 0,30).

Si vous n'aimez pas les navets, pourquoi ne pas modifier le programme en y insérant ce que VOUS aimez?

```
70 PRINT "GARDEN" (jardin)
80 PRINT "Length remaining: "length;"metres"
        (longueur cultivable restante: longueur mètres)
90 PRINT "Which vegetable do you want to grow"
        (Quel légume voulez vous cultiver?)
100 PRINT
110 PRINT''                    width          yield          rows"
                               (largeur)      (rendement)    (rangs)
120 PRINT"1. ONIONS            0.3m           9Kg             ";onions
130 PRINT"2. CARROTS           0.3m           3Kg             ";carrots
140 PRINT"3. POTATOES          1  m           50Kg            ";potatoes
150 PRINT"4. CABBAGES          0.6m           8Kg             ";cabbages
160 PRINT"5. BEANS             1  m           30Kg            ";beans
170 PRINT"6. PARSNIPS          0.5m           1Kg             ";parsnips
180 PRINT
190 PRINT "Enter a number between 1 and 6'' (chiffre entre 1 et 6)
200 PRINT "or 7 to show total output''
          (ou 7 pour montrer la totalité de la production)
210 INPUT veg
220 IF veg<=0 OR veg>7 THEN GOTO 190
230 IF veg=7 THEN GOTO 320
240 IF veg=1 THEN GOSUB 470
250 IF veg=2 THEN GOSUB 540
260 IF veg=3 THEN GOSUB 610
270 IF veg=4 THEN GOSUB 680
280 IF veg=5 THEN GOSUB 750
290 IF veg=6 THEN GOSUB 820
300 PRINT "-------------------------------------"
310 GOTO 60
320 REM summary (résumé)
340 PRINT "SUMMARY"
350 PRINT "GARDEN OUTPUT IN KILOS"  (production totale en Kg)
```

```
370 PRINT
380 PRINT"Onions :";onions*9;"Kg"
390 PRINT"Carrots :";carrots*3;"Kg"
400 PRINT"Potatoes :";potatoes*50;"Kg"
410 PRINT"Cabbages :";cabbages*8;"Kg"
420 PRINT"Beans :";beans*30;"Kg"
430 PRINT"Parsnips :";parsnips*11;"Kg"
450 GOTO 450
460 :
470 REM Onions
480 PRINT"Onions"
490 rowwidth=0.3
500 GOSUB 890
510 onions=rows
520 RETURN
530 :
540 REM Carrots
550 PRINT"Carrots"
560 rowwidth=0.3:produce=3
570 GOSUB 890
580 carrots=rows
590 RETURN
600 :
610 REM Potatoes
620 PRINT"Potatoes"
630 rowwidth=1
640 GOSUB 890
650 potatoes=rows
```

```
660 RETURN
670 :
680 REM Cabbages (choux)
690 PRINT""Cabbages''
700 rowwidth=0.6 (largeur du rang)
710 GOSUB 890
720 cabbages=rows
730 RETURN
740 :
750 REM Beans
760 PRINT""Beans''
770 rowwidth=1
780 GOSUB 890
790 beans=rows
800 RETURN
810 :
820 REM Parsnips
830 PRINT""Parsnips''
840 rowwidth=0.5
850 GOSUB 890
860 parsnips=rows
870 RETURN
880 :
890 REM Details
900 INPUT ""How many rows do you want to plant'';rows
        (combien de rangs souhaitez vous)
910 testlength=length-rows*rowwidth
920 IF test length<0 THEN PRINT""no room'':GOTO 900
930 length=testlength
950 RETURN
```

Test

Il vous faut absolument passer le SAT11 afin de vous entraîner à manipuler les opérateurs mathématiques du CPC464. La plupart des programmes nécessitent un certain nombre de notions de mathématiques de base pour réaliser n'importe quoi, aussi n'hésitez pas à passer le plus de temps possible à comprendre cet aspect du CPC464. N'oubliez pas que la plupart des programmes de jeux disponibles pour le CPC464 vous obligent à effectuer des calculs mathématiques à toute vitesse!

12 JEUX

La plupart des jeux, et pas uniquement les jeux-ordinateurs, vous poussent à vous mesurer physiquement et mentalement avec l'adversaire, avec les évènements et même le temps. Il arrive que vous deviez jouer au plus fin avec les trois paramètres en même temps. Un jeu d'échecs ou d'arcade vous amusera de cette façon pendant de nombreuses heures sans que vous ayez besoin de connaître quoi que ce soit sur la programmation des jeux.

Cependant, si vous souhaitez analyser des jeux, vous avez besoin de quelques conseils. Retournez à vos livres de mathématiques. Il vous est impossible de programmer le rebondissement d'une balle sur un mur si vous ne vous souvenez pas de la formule qui vous permet de calculer un ricochet, une vitesse ou une trajectoire. Bien entendu, nous n'allons pas aborder ce genre de détails à ce stade du cours, mais vous devez vous rendre compte que les deux jeux proposés ne représentent qu'une infime partie de tout ce que le CPC464 est capable de réaliser.

RND

Effets hasard

Lorsque vous avez joué au "Bombardier" du Chapitre Quatre, le vaisseau ennemi faisait sans cesse irruption en divers endroits. En fait, il obéissait tout simplement à la commande RND (Random hasard) qui permet l'entrée en jeu de l'élément "hasard" dans un programme-ordinateur.
On peut avoir recours à la commande RND de la façon suivante:

```
move rnd*639,rnd*399
```

Il vous est donné un nombre décimal entre 0 et 1 que vous devez multiplier par le plus grand nombre que vous souhaitez insérer dans la commande ou la routine. Dans le cas ci-dessus, le curseur-graphique sera positionné au hasard sur l'écran puisque nous avons mulitplié le maximum des coordonnées x et y par RND.

Essayez vous-même de rédiger un programme succinct faisant intervenir cet élément "hasard" dans l'emplacement d'un petit carré sur l'écran.

Mesure du temps

Autre fonction trés utile au CPC464, le temps (TIME). Dés que l'on met l'appareil en marche ou que l'on recommence à s'en servir, celui-ci comptabilise son temps toutes les trois cents secondes et conserve son calcul dans TIME. Cette fonction n'entre pas en jeu uniquement au moment où un programme est en train d'être chargé ou sauvegardé sur la datacassette. Voici un exemple d'application:

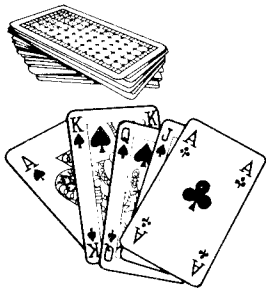
Remarquez le nouveau mot-clé qui figure aux lignes 20 et 60, INKEY$. Celui-ci ressemble beaucoup à INPUT à la seule différence qu'il ne vous donne qu'un caractère et que ENTER ne doit pas suivre obligatoirement. Dans ce cas précis, il sert à détecter le moment où l'on appuie sur une touche (n'importe laquelle à l'exception de SHIFT, CTRL, CAPS LOCK et ESC).

TIME

INKEY$

```
10 print "Press any Key" (frappez une touche de votre choix)
20 if inkey$="" " goto 20
30 for t=1 to rnd*5000 :next
40 a=time
50 print"again"
60 if inkey$="" "goto 60
70 b=time
80 print "Reaction time="; (b-a)/300;"seconds"
90 end
```

Il s'agit là d'échantillonner la valeur de TIME avant et aprés l'apparition du mot "again" (de nouveau) sur l'écran, le temps de réaction s'inscrivant entre les deux positions.



Le vingt-et-un

Vous vous apercevez bien vite combien il est agréable de jouer avec le CPC464 au solitaire plutôt que d'y jouer tout seul. Le programme suivant de la datacassette A se nomme BLACKJACK (le vingt-et-un). Au cas ou vous ne connaissez pas, le jeu consiste à demander des cartes jusqu'à ce que leur total approche de trés prés le nombre 21. Si vous obtenez plus de 21, vous perdez la main. Sinon, le CPC464 se sert une main lui-même et tente de vous battre sans pour autant vous faire perdre.
Voici les autres règles:

- Cinq cartes dont le total atteint 21 ou moins gagnent automatiquement
- Les as peuvent valoir 1 ou 11 points, au choix
- Les cartes de figure valent 10 points

Voici le listing du BLACKJACK. A noter que plusieurs commandes qui y figurent ne seront abordées qu'en deuxième partie du manuel.

```
10 REM **BLACKJACK**
20 REM
30 REM **STARTUP**
40 MODE 1:BORDER 4
50 INK 0,17:INK 1,0
60 LOCATE 16,5:PRINT "BLACKJACK"
70 LOCATE 10,12
80 PRINT "Press a key to start" (frappez une touche pour commencer)
90 suit$=CHR$(226)+CHR$(227)+CHR$(228)+CHR$(229)
100 card$="A23456789TJQK"
110 C5$="5 card trickøI win"
120 myace=0:yourace=0
130 mygames=0:yourgames=0
140 WHILE INKEY$="":WEND
```

```
150 CLS
160 REM ** YOUR TURN ** (votre tour)
170 yourcards=0:yourace=0
180 yourhand=0
190 LOCATE 20,1
200 PRINT"Games Me:";my games;
210 PRINT"You:";yourgames
220 y=20:x=5
230 yourcards=0
240 GOSUB 770
250 yourcards=yourcards+1
260 IF value=1 THEN yourace=yourace+1
270 yourhand=yourhand+value
280 GOSUB 830:x=x+5
290 IF yourhand>21 THEN GOTO 690
300 GOSUB 930
310 oneacehand=yourhand
320 IF yourace>=1 THEN oneacehand=yourhand+10
350 IF yourcards=5 THEN 440
360 IF yourhand=21 THEN 440
370 IF oneacehand=21 THEN 420
380 IF yourace=0 AND yourhand<=11 THEN 240
390 IF yourcards=i THEN 240
400 INPUT"Want another card(Y/N)";q$
        (nouvelle carte demandée (oui ou non)
410 IF UPPER$(q$)="Y" THEN GOTO 240
420 IF oneacehand<21 THEN yourhand=oneacehand
440 REM ** MY TURN ** (mon tour)
450 y=10:X=5
```

```
460 myhand=0:mycards=0:myace=0
470 GOSUB 770
480 mycards=mycards+1
490 IF value=1 THEN myace=myace+1
500 myhand=myhand+value
510 GOSUB 830:x=x+5
520 FOR delay=0 TO 1000:NEXT
530 IF myhand>21 GOTO 720
540 IF mycards=5 THEN GOSUB 930:PRINT C5$:GOTO 710
550 IF yourcards=5 THEN 470
560 mineA=myhand
570 IF myace>=1 AND myhand<12 THEN mineA=myhand+10
600 IF myhand>=yourhand THEN 640
610 IF mineA>=yourhand THEN myhand=mineA:GOTO 640
630 GOTO 470
640 REM ** TEST RESULTS ** (résultats)
650 GOSUB 930
660 PRINT"I have";my hand;
670 PRINT"and you have";your hand
680 IF myhand<yourhand GOTO 730 ELSE GOTO 700
690 GOSUB 930:PRINT "You have bust!" (perdu!)
700 PRINT "I win" (je gagne)
710 mygames=mygames+1:GOTO 140
720 GOSUB 930:PRINT "I have bust!" (j'ai perdu!)
730 PRINT "You win" (vous gagnez)
740 yourgames=yourgames+1:GOTO 140
750 END
760 REM ** GENERATE CARD **
```

```
770 card=INT(RND*13)+1
780 suit=INT (RND*4)+1
790 value=card
800 IF value>10 THEN value=10
810 RETURN
820 REM ** PRINT CARD **
830 LOCATE x,y
840 PRINT CHR$(24);"  ";CHR$(24)
850 LOCATE x,y+1
860 PRINT CHR$(24);"  ";
870 PRINT MID$(card$,card,1);
880 PRINT MID$(suit$,suit,1);
890 PRINT "  "; CHR$ (24)
900 LOCATE x,y+2
910 PRINT CHR$(24);"  ";CHR$(24)
920 RETURN
930 LOCATE 1,24:PRINT SPACE$(40)
940 LOCATE 1,24:RETURN
```

Simple Simon

CHR$

Vous le constatez vous-même ce n'est-ce pas si simple que ça! Vous avez de nouveau affaire à certaines commandes qui ne sont traitées qu'en deuxième partie. Cependant vous aimeriez sûrement connaître tout de suite un de ces mots-clé, CHR$.

Au Chapitre Trois, nous avions vu que certains caractères inhabituels pouvaient apparaître sur l'écran sans pour cela figurer sur les touches du clavier. La fonction CHR$ (CHaracteRs) vous permet de les faire intervenir par le biais de leur propre code. Par exemple, il vous est possible d'imprimer un petit vaisseau spatial sur l'ecran en faisant:

```
print chr$ (239)
```

La liste complète des caractères du CPC464 se trouve à votre disposition dans le *Guide de l'Utilisateur de l'Amstrad CPC464.*

```
10 cr$=CHR$(13)
20 REM Simon
30 REM **** INSTRUCTIONS ****
40 MODE 1:BORDER 20:INK 0,20:INK 1,1
50 LOCATE 16,2:PRINT CHR$(24);"Simon";CHR$(24)
60 PRINT:PRINT
70 PRINT "In this game, you have to watch the"
80 PRINT "flashing circles and remember the"
90 PRINT "pattern. When the sequence ends you"
100 PRINT "must copy it out on the cursor keys"
110 PRINT "The sequence increases by one after"
120 PRINT "each correct attempt. PRINT"
130 PRINT "For example, a circle at the top of"
140 PRINT "the screen should be indicated by"
150 PRINT "the up cursor. The cursor keys are"
160 PRINT "above the numeric key pad, and are"
170 PRINT "marked as follows:"PRINT
```

(Ce jeu consiste a observer les cercles clignotent et a les memoriser. A la fin de la sequence, vous devez les recopier sur les touches du curseur. A chaque fois que vous reussissez, la sequence augmente. Par exemple un cercle en haut de l'ecran doit être indique par le curseur superieur. Les touches du curseur se trouvent au-dessus du clavier numerique et sont signalees de la façon suivante:)

```
180 PRINT TAB(20);CHR$(240)
190 PRINT TAB(19);CHR$(242);"  ";CHR$(243)
200 PRINT TAB(20);CHR$(241)
210 LOCATE 7,22:PRINT"Press ENTER to continue" (appuyez sur ENTER
pour continuer)
220 LOCATE 5,24:PRINT"there will be a short pause!" (attendez vous a
une brève pause!)
230 WHILE INKEY$< cr$:WEND
240 REM **** SET-UP ****
250 MODE 0
260 WINDOW 7,14,10,16 (fenêtre)
270 b=17:f=3:REM Background/Foreground (arrière-plan, premier plan)
280 BORDER b
290 INK 0,17
300 FOR i=1 TO 15:INK i,b:NEXT
310 x=320:y=70:c=2;GOSUB 940
320 y=330:c=1:GOSUB 940
330 x=120:y=200:c=3:GOSUB 940
340 x=520:c=4:GOSUB 940
350 INK 5,F:PEN 5
360 RANDOMIZE TIME (L'élément "hasard" intervient pour le facteur
temps)
370 a$=" "
380 REM **** DISPLAY SEQUENCE **** (séquence affichage)
390 a$=a$+CHR$(RND*3+1)
400 FOR i=1 TO LEN(a$)
410 FOR j=1 TO 200:NEXT
420 x=ASC(MID$(a$,i,1))
430 INK x,2*x+1
440 SOUND 1,10+x*100
450 FOR j=1 TO 200:NEXT
```

```
460 INK x,b
470 NEXT
480 FOR i=1 TO 100:NEXT
490 REM **** GET ANSWER ****
500 FOR i=1 TO LEN(a$)
510 WHILE k$>" ":k$=INKEY$:WEND
520 FOR L=1 TO 2000:k$=INKEY$
530 IF k$>" "THEN 560
540 NEXT
550 k$=" "
560 k=ASC(k$)-239:IF k<1 ORk>4 THEN520
570 x=ASC(MID$(a$,i,1))
580 IF k<>x GOTO 730:REM wrong
590 INK x,2*x+1
600 SOUND 1,10+x*100
610 FOR j=1 TO 80:NEXT
620 INK x,b
630 FOR j=1 TO 20:NEXT
640 NEXT
650 REM **** RIGHT! **** (juste)
660 CLS:PRINT"RIGHT!"
670 PRINT:PRINT:PRINT" SCORE:"
680 PRINT:PRINT"  ";LEN(A$)
690 FOR j=1 TO 600:NEXT
700 LOCATE 1,1:PRINT''
710 GOTO 390
720 REM **** WRONG **** (faux)
730 SOUND 1,2000
```

```
740 CLS:PRINT  "Wrong"
750 FOR j=1 TO 300:NEXT
760 PRINT:PRINT "Sequence  was:"
770 FOR i=1 TO LEN(a$)
780 x=ASC(MID$(a$,i,1))
790 INK x,2*x+1
800 SOUND 1,10+x*100
810 FOR j=1 TO 200:NEXT
820 INK x,b
830 FOR J=1 TO 200:NEXT
840 NEXT
850 REM **** END & RESTART **** (fin et recommencement)
860 CLS
870 PRINT"  You"
880 PRINT"  scored"
890 PRINT:PRINT"    ";LEN(a$)
900 PRINT:PRINT"  PRESS"
910 PRINT" ENTER"
920 WHILE INKEY$<>CR$:WEND
930 GOTO 360
940 REM **** CIRCLES **** (cercles)
950 r=60
960 FOR i=-r TO r STEP 2
970 h=SQR(r*r-i*i)
980 MOVE x-h,i+y:DRAW x+h,i+y,c
990 NEXT
1000 RETURN
```

Test

Avant de passer à la deuxième partie de ce manuel, entraînez-vous le plus possible à rédiger vos programmes et effectuez le dernier test, le SAT12. Celui-ci non seulement regroupe toutes les questions-clé de TOUS les chapitres étudiés mais vous signale également très clairement les passages que vous devez revoir à tout prix!
Bonne chance!

LISTE DES MOTS-CLE

Voici une liste de tous les mots-clé du Basic Amstrad qui apparaissent dans ce manuel. Ce livre étant destiné avant tout aux débutants, il ne traite pas de toutes les variations ou extensions possibles. La deuxième partie de ce livre, intitulée *MORE BASIC,* aborde de nouveau mots-clé ainsi que des techniques de programmation plus élaborées.

Vous trouverez une description de tous les mots-clé dans le *Guide de l'Utilisateur de l'Amstrad CPC464.*

Chapitre Deux

RUN
LOAD

Chapitre Trois

CLS
RUN

Chapitre Quatre

BORDER
MODE
CAT

Chapitre Cinq

CLG
INK
DRAW
LIST
MOVE
NEW
PAPER

PLOT
REM

Chapitre Six

INPUT
LET
LOCATE
PRINT
SAVE

Chapitre Sept

CONT
EDIT
ELSE
GOTO
IF
STOP
THEN

Chapitre Huit

DRAWR
END
FOR
GOSUB
MOVER
NEXT
PLOTR
RETURN
STEP

Chapitre Dix

ENT
ENV
SOUND

Chapitre Onze

ROUND

Chapitre Douze

CHR$
INKEY$
RND
TIME

LISTE DES PROGRAMMES

La datacassette A contient les programmes ci-aprés dans l'ordre de référence du manuel. La datacassette B contient les tests d'auto-évaluation (SAT) que vous êtes tenus de passer à chaque fin de chapitre (à l'éeception des chapitres un et neuf).

Chapitre Deux

HELLO
SIMON (voir également le chapitre douze)

Chapitre Trois

LETTERS (lettres)
REPEAT NAME (répétition d'un nom)
KEYBOARD (clavier)
HANGMAN (le pendu)

Chapitre Quatre

DRAW
COORGEOM
BOMBER (le bombardier)

Chapitre Cinq

HOUSE (maison)

Chapitre Six

BARCHART (histogramme)
BUZZWORD

Chapitre Sept

DECO

Chapitre Huit

MANSION (maison)

Chapitre Dix

ZAPPOW
ORGAN

Chapitre Onze

MULT TABLES (tables de multiplications)
GARDEN (jardin)

Chapitre Douze

BLACKJACK (le vingt-et-un)
PONTOON
SIMON

INDEX