sinclair

ZX81 BASIC

COURS DE PROGRAMMATION

sinclair

ZX81 BASIC

COURS DE PROGRAMMATION

1re Edition

par Steven Vickers

Illustration de la couverture réalisée par John Harris à l'intention de Sinclair

ZX81

Chapitre 1

Chapitre 1

RACCORDEMENT DU ZX81

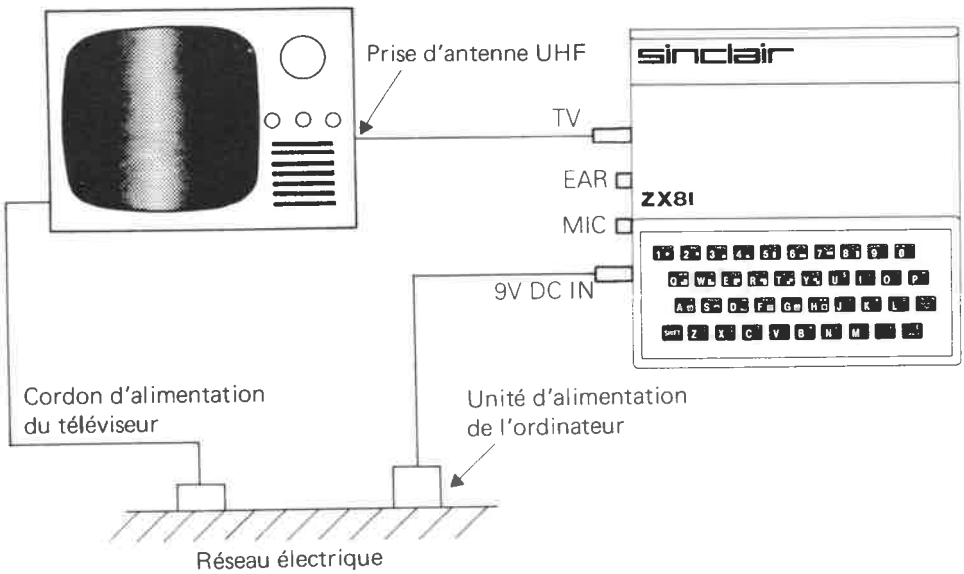
Lorsque vous avez déballé le ZX81, vous avez trouvé :

1. Ce manuel.

2. L'ordinateur. Le ZX81 comporte trois embases de jacks (avec les légendes 9V DC IN, EAR & MIC), une prise d'antenne et une partie exposée de son circuit imprimé pour vous permettre de connecter d'autres équipements. Le ZX81 ne comporte pas de commande marche/arrêt ; pour le mettre sous tension, il suffit de le raccorder à l'alimentation électrique.

3. Alimentation.
   L'unité d'alimentation transforme le courant du secteur pour le mettre à la valeur utilisée par le ZX81. Si, par erreur, vous connectez l'alimentation à la mauvaise embase de l'ordinateur, ce dernier ne sera pas endommagé. Si vous voulez utiliser votre propre alimentation, elle doit fournir une tension non régulée de 9 V DC à 700 mA et comporter un jack de 3,5 mm à extrémité positive.

4. Un câble d'antenne d'1,20 m pour relier l'ordinateur à un téléviseur.



5. Une paire de conducteurs mesurant 30 cm environ équipés de prises jack de 3,5 mm à leurs deux extrémités. Ces conducteurs permettent de relier l'ordinateur à un magnétophone.

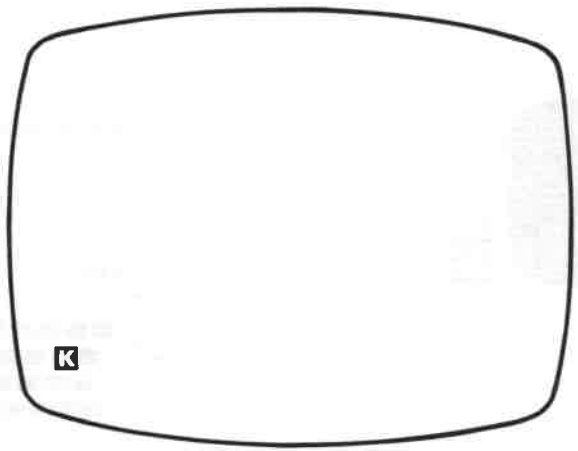
Vous devez également utiliser un téléviseur ; l'ordinateur ZX81 peut travailler sans téléviseur, mais vous ne pourrez pas voir ce qu'il est en train de faire ! Ce téléviseur doit être du type UHF ; en France, si le poste ne peut pas recevoir la 2e chaîne, il ne peut pas être utilisé avec le ZX81.

Ultérieurement, vous devrez utiliser un magnétophone à cassette. En effet, lorsque vous mettez le ZX81 hors tension, toute l'information qu'il contient est définitivement perdue. La seule façon de la préserver est de l'enregistrer sur une cassette — cette opération vous sera expliquée au chapitre 16. Bien entendu, vous pouvez également acheter des bandes qui ont été préparées par d'autres personnes et exécuter leurs programmes.

Lorsque vous avez tout rassemblé (à l'exception du lecteur de cassette), il faut connecter les éléments en respectant le schéma ci-dessus.

Si votre téléviseur comporte deux prises d'antenne (UHF et VHF), il faut utiliser la prise UHF.

Mettre sous tension et allumer le téléviseur. Maintenant, vous devez régler votre téléviseur. Le ZX81 travaille sur le canal 36 en UHF et, après mise sous tension et réglage, vous devez obtenir l'image suivante :



Pendant l'emploi de l'ordinateur, la plupart des gens mettent la commande de volume à sa position minimale.

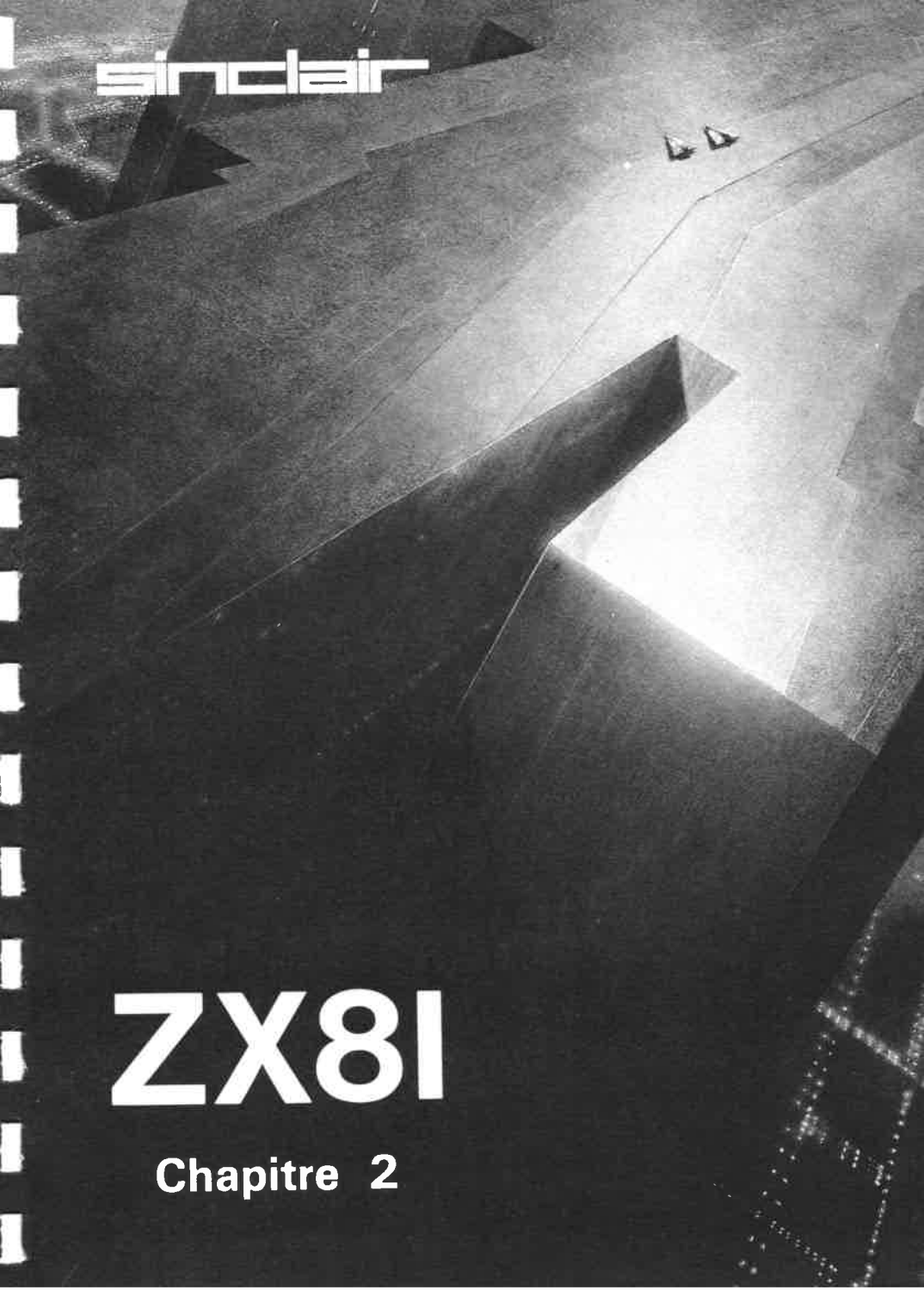
Si la commande d'accord de votre téléviseur est du type variable en continu, vous devez la régler jusqu'au moment où vous obtenez cette image. Toutefois, de nombreux téléviseurs comportent un poussoir distinct pour chaque chaîne. Vous devez donc choisir un poussoir inutilisé et effectuer le réglage correspondant.

Si l'ordinateur se bloque, rappelez-vous qu'il est toujours possible de le remettre à zéro et d'obtenir à nouveau cette image en sortant la prise «9V DC IN» et en la remettant ensuite. Cette opération doit être faite seulement dans les cas exceptionnels puisqu'elle provoque la perte de toute l'information que contient la machine.

Maintenant que votre ordinateur est installé, vous allez vouloir vous en servir. Si vous connaissez déjà le langage BASIC, il vous suffit de lire l'annexe C et de vous reporter au reste du manuel pour préciser certains points qui pourraient demeurer vagues.

Si vous êtes néophyte, l'essentiel de ce manuel a été écrit à votre intention. Ne laissez pas les exercices de côté, beaucoup d'entre eux permettent de traiter des points intéressants qui ne sont pas étudiés dans le texte. Vous devez donc bien les étudier et faire tous ceux qui vous intéressent ou qui semblent porter sur des points que vous ne comprenez pas bien.

Et continuez toujours à utiliser votre machine. Si vous vous posez une question du genre «que va-t-il vaire si je lui dis ceci et cela ?», vous pouvez facilement répondre — il suffit d'entrer les données correspondantes et de voir ce qu'il se passe. Lorsque le manuel vous dit d'entrer quelque chose, vous devez toujours vous poser la question «Que pourrais-je frapper à la place ?» et effectuer ensuite une tentative avec votre réponse. Plus vous écrirez vous-même et mieux vous comprendrez le ZX81. (cette méthode s'appelle la formation non programmée). Quelles que soient les entrées que vous décidez d'essayer, vous ne risquez en aucun cas d'endommager l'ordinateur.



ZX81

Chapitre 2

POUR DIRE A L'ORDINATEUR CE QU'IL DOIT FAIRE

Mettez l'ordinateur sous tension (en le reliant à l'alimentation) ; vous devez maintenant obtenir l'écran blanc avec la lettre K en blanc sur fond noir (voir l'illustration du chapitre 1). Pour que la machine fasse quelque chose, vous devez entrer un message qu'elle comprend ; par exemple, le message :

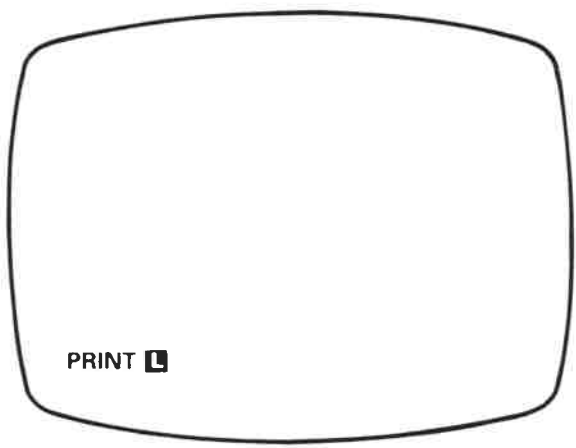
PRINT 2 + 2

lui dit de calculer la somme 2 + 2 et de mettre la réponse sur l'écran du téléviseur.

Un message de ce type qui ordonne à l'ordinateur de faire quelque chose immédiatement est une commande ; dans le cas ci-dessus, il s'agit d'une commande **PRINT**, mais également d'une instruction **PRINT**. Le fait de l'appeler instruction **PRINT** est une référence à la forme sans préciser la manière dont l'ordinateur va l'utiliser. Ainsi, chaque commande revêt la forme d'une instruction, mais d'autres choses aussi revêtent cette forme — par exemple les lignes d'un programme comme nous le verrons au chapitre 8.

Pour entrer cette commande :

1. Entrez d'abord **PRINT**. Attention : Bien que le clavier comporte une touche pour chaque lettre, vous ne devez pas entrer les différentes lettres constituant le mot PRINT. Il vous suffit d'appuyer sur la touche P et le mot apparaît sur l'écran avec un espace de présentation ; l'écran se présente alors comme suit :



Ceci est dû au fait que l'ordinateur attend un mot-clé au début de chaque commande — un mot-clé précise le type de la commande. Les mots-clés sont écrits au-dessus des touches et, en vous reportant au clavier, vous pouvez voir que '**PRINT**' est écrit au-dessus de la touche P ; par conséquent, pour écrire '**PRINT**', vous devez appuyer sur la touche P.

Pour vous indiquer qu'il attend un mot-clé, l'ordinateur met sur l'écran l'indication K qui a constitué votre point de départ. L'écran comporte presque toujours une lettre blanche sur fond noir (vidéo inversée), K ou L, (ou comme nous le verrons plus tard F ou G) appelée Curseur. La lettre K signifie «quelle que soit la touche manœuvrée, je l'interprèterai comme un mot-clé». Comme vous avez pu le constater, lorsque vous avez appuyé sur P pour obtenir **PRINT**, le K est devenu L.

La possibilité d'appuyer sur une touche pour obtenir plus d'un seul symbole est très utilisée sur le ZX81. Dans la suite de ce manuel, les mots pour lesquels existe une touche sont imprimés en **CARACTÈRES GRAS.**

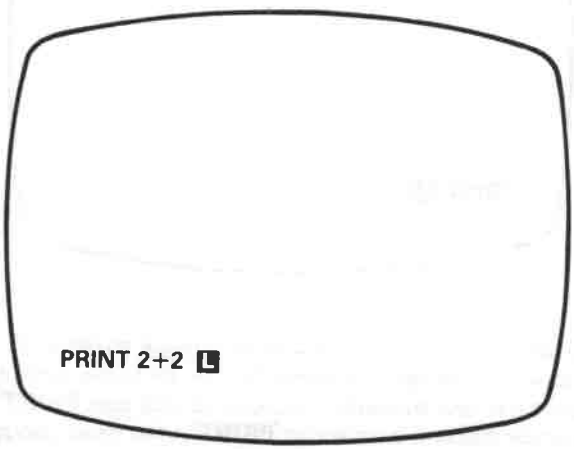
Souvenez-vous qu'il est inutile d'entrer ces mots lettre par lettre car l'ordinateur ne comprendra pas ce que vous voulez faire.

2. Appuyez maintenant sur la touche 2. Ceci ne doit pas provoquer de difficulté. Dans ce cas également, vous allez voir le chiffre 2 apparaître sur l'écran et la lettre L se déplacer d'une position.

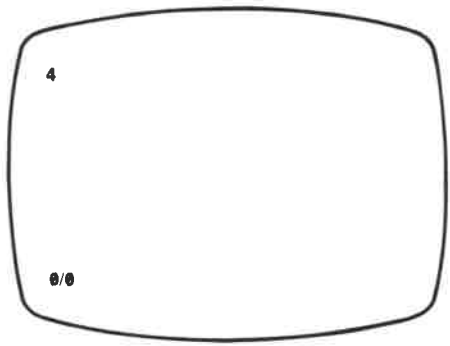
Remarquez aussi qu'un espace est intercalé automatiquement entre **PRINT** et 2 pour faciliter la lecture. Cette insertion intervient assez fréquemment de sorte que vous n'avez pratiquement jamais besoin d'entrer un espace. Toutefois, si vous entrez un espace, il sera affiché sur l'écran, mais il n'aura aucune influence sur la signification du message.

3. Appuyez maintenant sur +. Ce caractère correspond à la position MAJUSCULES. (Ces indications sont en rouge – c'est-à-dire la couleur de **SHIFT** – **MAJUSCULES** – sur la touche – dans le coin supérieur de droite de chaque touche) ; pour obtenir '+', vous devez maintenir la touche SHIFT en position basse et, simultanément, appuyer sur LIST K+ LEN.

4. Entrez à nouveau le chiffre 2. L'écran se présente comme suit :



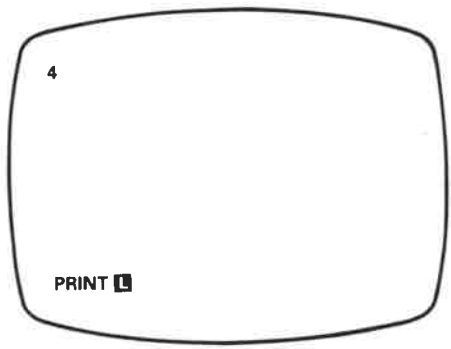
5. Maintenant et c'est là une opération qu'il ne faut jamais oublier, appuyer sur **NEWLINE**, c'est-à-dire sur la touche [FUNCTION NEW LINE] . Cette manœuvre veut dire « message complet » ou encore « bon, c'est maintenant à l'ordinateur de travailler ». L'ordinateur lit le message, en déduit ce qu'il doit faire et le fait. Dans le cas ci-dessus, l'écran devient :



Le résultat est donc 4 ; bien entendu, vous n'avez pas besoin d'un ordinateur pour obtenir cette réponse.

Ø/Ø représente le compte-rendu des opérations effectuées par l'ordinateur. (Remarquez que le chiffre zéro est toujours traversé par une barre oblique pour le distinguer du O. Cette représentation est courante en informatique.) Le premier Ø veut dire « tout va bien, pas de problème ». (A l'annexe B, vous trouverez une liste d'autres codes de compte-rendu contenant, entre autres, ceux qui peuvent apparaître si une opération se déroule mal.) Le deuxième Ø veut dire « la dernière ligne que j'ai effectuée était la ligne Ø». Vous verrez plus loin – lorsque nous passerons à l'écriture des programmes – qu'une instruction peut être désignée par un numéro et mémorisée pour exécution ultérieure : dans ce cas, elle devient une ligne de programme. En réalité, les commandes n'ont pas de numéro mais, pour faciliter ces comptes rendus, l'ordinateur prétend qu'elles sont la ligne Ø.

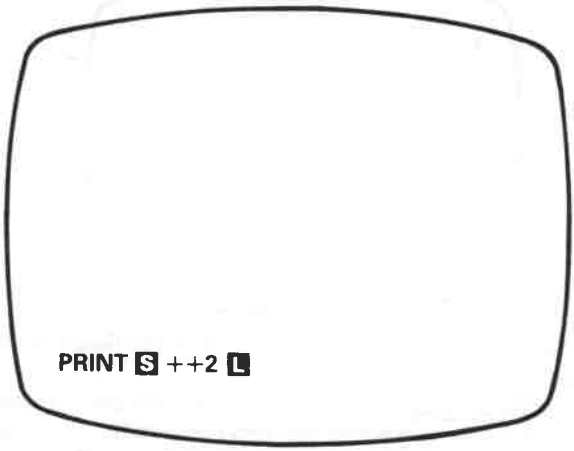
Imaginons un compte rendu qui cache le curseur K – Si vous appuyez sur P (**PRINT**), le compte-rendu disparaît et l'écran se présente comme suit :



Le curseur peut également être utilisé pour corriger des erreurs : appuyez sur ++2 pour obtenir

PRINT ++2

sur la dernière ligne de l'écran. Ceci est incompréhensible et, si vous appuyez sur la touche **NEWLINE**, vous obtenez



Le S est la marque désignant une erreur de syntaxe (la syntaxe est la grammaire des messages, elle précise ceux qui sont autorisés et ceux qui ne le sont pas), elle indique que l'ordinateur est allé jusqu'à **'PRINT'** et que, ensuite, il a décidé que le message que vous lui aviez fourni était incorrect.

Bien entendu, vous voulez maintenant effacer le premier + et le remplacer, par exemple, par 3. Vous devez d'abord déplacer le curseur pour le ramener immédiatement à droite du premier + ; deux touches — ◊ et ◊ — (5 et 8 en position majuscules) permettent de déplacer le curseur à gauche et à droite. Maintenez **SHIFT** en position basse et appuyez deux fois sur la touche ◊. Le curseur se déplace de deux positions vers la gauche et vous obtenez :

PRINT +L+2

Appuyez maintenant sur la touche **RUBOUT** (Ø en position basse) et vous avez :

PRINT L + 2

RUBOUT efface le caractère (ou le mot-clé) présent immédiatement à gauche du curseur.

Si vous appuyez maintenant sur la touche 3, le chiffre "3" est inscrit immédiatement à gauche du curseur ; vous obtenez donc :

PRINT 3L + 2

et si vous appuyez sur **NEWLINE**, vous obtenez la réponse : 5.

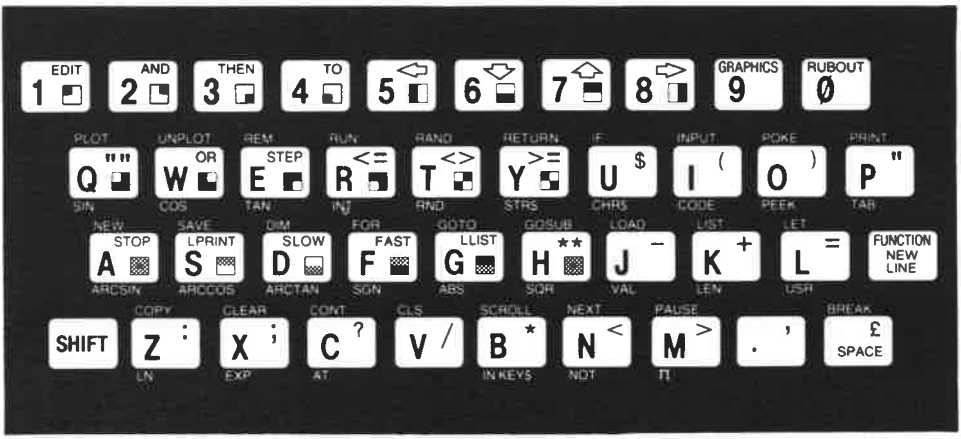
La touche ◊ (8 en position basse) a la même fonction que la touche ◊ mais elle déplace le curseur vers la droite et non vers la gauche.

Résumé

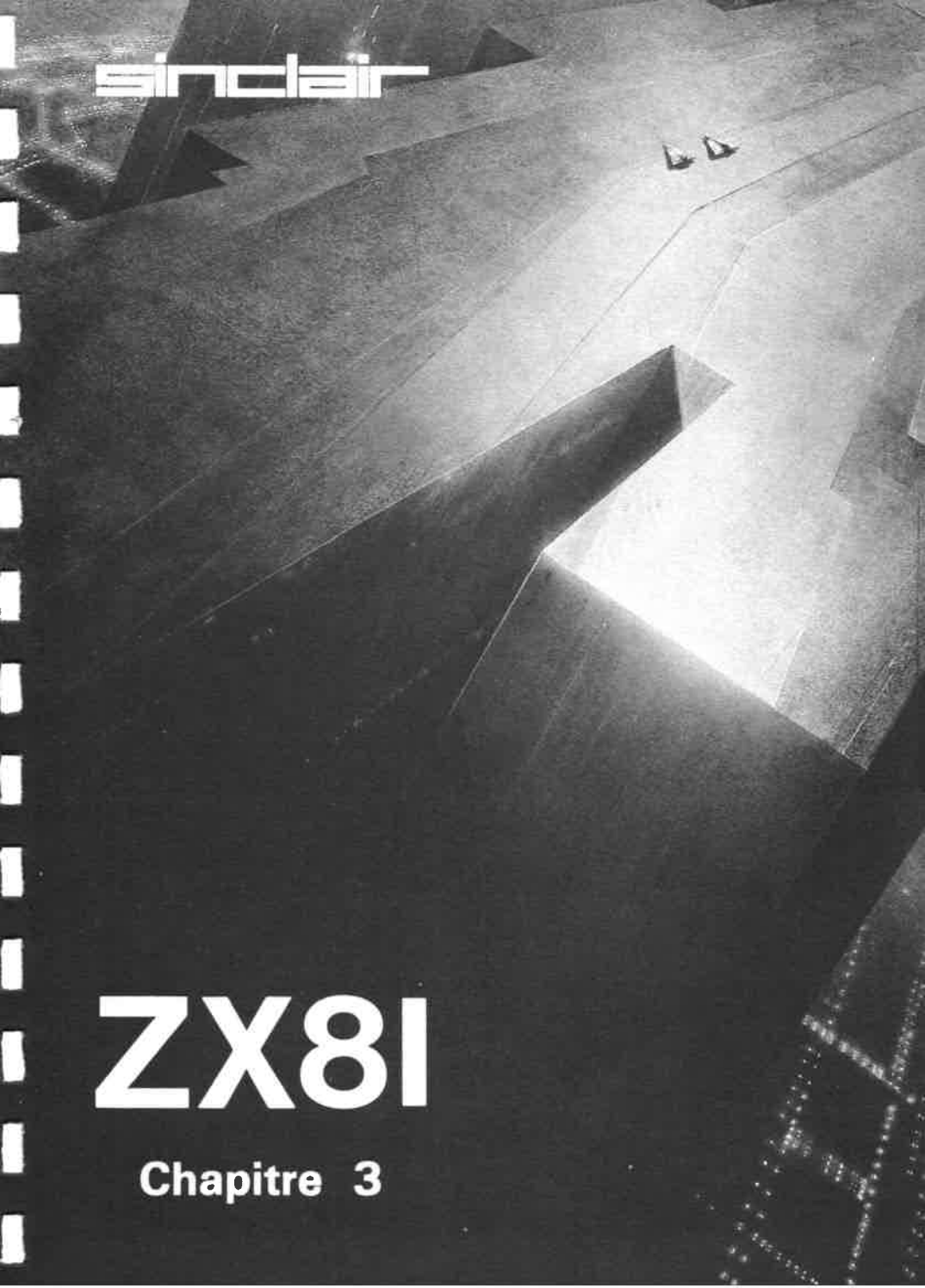
Ce chapitre a expliqué l'entrée des messages destinés au ZX81 ; vous avez pu étudier :
Le système permettant d'entrer un mot en manœuvrant une seule touche,
Les curseurs K et L,
Les comptes-rendus,
Le marqueur d'erreur de syntaxe S,
Et la correction des erreurs par les touches ◊ , ◊ , et **RUBOUT**.

Le clavier

Voici une photographie du clavier.



Rappelez-vous que pour utiliser **SHIFT**, vous devez maintenir cette touche en position basse pendant que vous appuyez sur une autre touche. Ne confondez pas le chiffre Ø et la lettre O.



ZX81

Chapitre 3

Chapter 3

UNE LEÇON D'HISTOIRE

Les messages que vous entrez sont écrits dans un langage spécial appelé BASIC (en anglais Beginners' All-purpose Symbolic Instruction Code). En réalité, l'ordinateur doit réaliser un certain nombre d'opérations pour décomposer les messages écrits en BASIC en opérations élémentaires, mais, tout compte fait, c'est là une partie de son travail. Les messages du BASIC contiennent assez peu de mots anglais (comme **PRINT**) pour qu'il soit facile à un non anglophone de les apprendre.

Le langage BASIC a été conçu au Collège de Dartmouth, dans le New Hampshire, Etats-Unis, en 1964 et, depuis cette époque, c'est le langage informatique le plus utilisé par les débutants et par les passionnés de l'informatique. Ceci résulte du fait que ce langage est particulièrement bien adapté au travail en ligne dans lequel l'utilisateur entre quelque chose dans l'ordinateur et où ce dernier répond immédiatement. Il existe d'autres langages — comme l'ALGOL (en réalité toute une famille d'ALGOLS) et le PASCAL — dont la structure est beaucoup plus élaborée et la puissance plus grande que le BASIC, mais quelques-uns seulement comme APL & POP-2 sont faciles à utiliser en ligne. Nous devons quand même mentionner d'autres langages comme le FORTRAN, PL1 et COBOL.

De nombreux magazines consacrés à l'ordinateur individuel diffusent des programmes en BASIC qui peuvent être intéressants à étudier pour y trouver des idées. Vous devrez certainement procéder à une petite adaptation de ces programmes parce que chaque ordinateur utilisant le langage BASIC emploie son propre dialecte qui est différent de tous les autres.

ZX81

Chapitre 4

POUR EMPLOYER LE ZX81 COMME UNE CALCULATRICE

Mettez votre ordinateur sous tension. Vous pouvez maintenant l'utiliser comme une calculette comme il a été dit au chapitre 2 : Vous entrez **PRINT**, les données du problème à résoudre et, enfin, vous appuyez sur **NEWLINE** (dans la suite, nous omettrons la plupart du temps de vous dire d'appuyer sur la touche **NEWLINE**).

Comme vous êtes en droit de l'espérer, le ZX81 ne se contente pas d'additionner, il sait également soustraire, multiplier (avec le symbole * au lieu du symbole habituel, comme c'est souvent le cas sur les ordinateurs) et diviser (avec le symbole / au lieu de ÷). Essayons maintenant d'effectuer ces opérations.

+, −, * et / sont des opérateurs arithmétiques, tandis que les valeurs numériques sur lesquelles ces symboles agissent sont les opérandes.

L'utilisateur peut également élever un nombre à une puissance par l'opérateur ** (touche H en position majuscules. Nota : Ne pas entrer *— B en position majuscules — deux fois) : entrez :

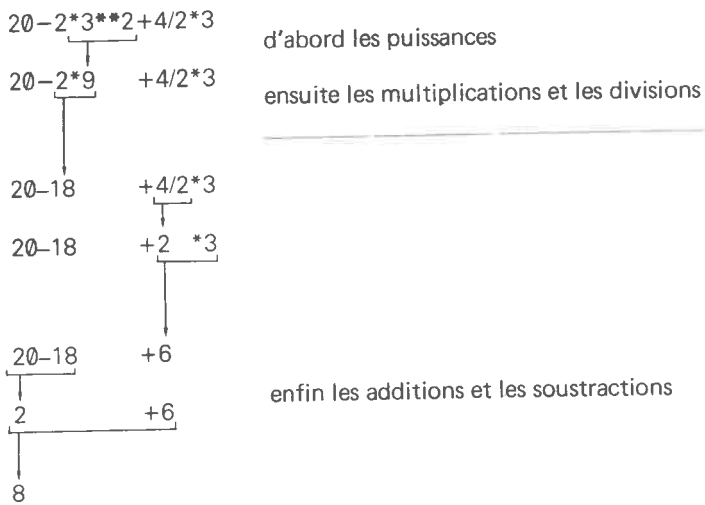
PRINT 2**3 (N'oubliez pas d'appuyer sur **NEWLINE**).

et vous obtenez la réponse 8 (2 élevé à la puissance 3 ou 2^3 ou 2 au cube).

Le ZX81 donne également le résultat de combinaisons d'opérations. Exemple :

PRINT 20−2*3**2+4/2*3

donne le résultat 8. Pour donner ce résultat, l'ordinateur effectue plusieurs opérations puisqu'il calcule d'abord toutes les puissances (**) de gauche à droite, il fait ensuite toutes les multiplications et les divisions (*&/), de gauche à droite également, puis il fait les additions et soustractions (+&−) également de gauche à droite. Ainsi, l'exécution de l'exemple ci-dessus est la suivante :



Nous réalisons ce qui précède en donnant à chaque opérateur une priorité, c'est-à-dire un numéro de 1 à 16. Les opérations qui ont la priorité la plus haute sont évaluées en premier et les opérations de même priorité sont évaluées de gauche à droite.

**	a la priorité 10
* et /	ont la priorité 8
+ & –	ont la priorité 6

Lorsque le signe "–" est utilisé pour rendre négative une valeur (par exemple quand vous écrivez –1), sa priorité devient 9. (Ceci est le moins unaire par opposition au moins binaire de 3–1 ; une opération unaire comporte un seul opérande, tandis qu'une opération binaire en a deux. Noter que sur le ZX81 vous ne pouvez pas utiliser "+" comme opérateur unaire).

Cet ordre est absolument rigide, mais vous pouvez le modifier en utilisant des parenthèses : les expressions entre parenthèses sont évaluées en premier pour être traitées ensuite comme un seul nombre de sorte que :

PRINT 3*2 + 2

donne le résultat 6 + 2 = 8, tandis que

PRINT 3*(2 + 2)

donne le résultat 3*4 = 12.

Une combinaison de ce type est appelée expression ; dans le cas précédent, il s'agit d'une expression arithmétique ou numérique puisque le résultat est un nombre. En général, lorsque l'ordinateur attend que vous lui fournissiez un nombre, vous pouvez lui donner une expression, c'est lui qui calcule alors le résultat.

Vous pouvez écrire des nombres contenant une marque décimale (utilisez le point) et vous pouvez également employer la notation scientifique — c'est souvent le cas sur les calculatrices de poche. Après un nombre normal (avec ou sans marque décimale), vous pouvez écrire une partie exponentielle composée de la lettre E, peut-être de + ou de –, et d'un nombre sans marque décimale. Dans ce cas, E signifie "* 10**" ("fois dix à la puissance de"), et

2.34E0	= 2.34 * 10**0	= 2.34	
2.34E3	= 2.34 * 10**3	= 2340	
2.34E–2	= 2.34 * 10**–2	= 0.0234	et ainsi de suite.

(Essayez d'écrire les expressions ci-dessus sur le ZX81).

Le raisonnement le plus simple est d'imaginer que la partie exponentielle décale la marque décimale vers la droite (exposant positif) ou vers la gauche (exposant négatif).

Vous pouvez également écrire plusieurs choses à la fois en les séparant soit par des virgules (,) soit par des points virgules (; = X en position majuscules). Si vous utilisez

une virgule (,), le nombre suivant est présenté de deux façons possibles : soit commençant à la marge de gauche, soit au milieu de la ligne à la 16e colonne. Si vous utilisez un point virgule, le nombre suivant est affiché immédiatement après le précédent. Essayez d'entrer :

PRINT 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; 10

et

PRINT 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

pour voir quelles sont les différences. Si vous le désirez, vous pouvez mettre des virgules et des points virgules dans une seule instruction **PRINT**.

Résumé

Instructions : **PRINT**, avec virgules et points virgules
Opérateur : +, −, *, /, **
Expressions, notation scientifique

Exercices

1. Essayez :

PRINT 2.34EØ
PRINT 2.34E1
PRINT 2.34E2

et la suite jusqu'à

PRINT 2.34E15

Vous verrez qu'au bout d'un moment, le ZX81 commence aussi à utiliser la notation scientifique. Ceci est dû au fait qu'il n'utilise jamais plus de 14 positions pour écrire un nombre. De même, essayez :

PRINT 2.34E − 1
PRINT 2.34E − 2

et ainsi de suite.

2. Essayez :

PRINT 1,,2,,3,,,4,,,,5

Une virgule provoque toujours un petit déplacement du nombre suivant. Essayez maintenant :

PRINT 1;;2;;3;;;4;;;;5

Pourquoi plusieurs points virgules ne donnent-ils pas un résultat différent qu'un seul ?

3. **PRINT** donne seulement 8 chiffres significatifs. Essayez :

PRINT 4294967295, 4294967295 – 429E7

Ceci prouve que l'ordinateur peut contenir tous les chiffres de 4294967295 même s'il n'est pas disposé à les afficher tous simultanément.

4. Si vous avez des tables de logarithmes, essayez de vérifier cette règle :
Le fait d'élever 1Ø à la puissance d'un nombre est équivalent au logarithme inverse de ce nombre. Par exemple, entrez :

PRINT 1Ø**Ø.3Ø1Ø

et regardez ensuite le logarithme inverse de Ø.3Ø1Ø. Pourquoi les deux résultats ne sont-ils pas exactement égaux ?

5. Le ZX81 travaille en arithmétique en virgule flottante, ce qui veut dire qu'il fait une distinction entre les chiffres d'un nombre (la mantisse) et la position de la marque (l'exposant). Ceci n'est pas toujours exact, même pour les nombres entiers. Entrez :

PRINT 1E1Ø+1–1E1Ø, 1E1Ø–1E1Ø+1

Les nombres sont préservés avec une précision de 9 1/2 chiffres et, par conséquent, 1E1Ø est trop grand pour être mémorisé entièrement. L'imprécision (2 environ) est supérieure à 1, de sorte que pour l'ordinateur 1E1Ø et 1E1Ø +1 semblent égaux.
Voici un exemple encore plus curieux, entrez :

PRINT 5E9+1–5E9

Ici, l'imprécision de 5E9 est seulement de 1 environ et le 1 ajouté est en réalité arrondi à 2. Dans ce cas, les nombres 5E9+1 et 5E9+2 semblent égaux pour l'ordinateur.
Le plus grand entier qui peut être mémorisé avec une précision absolue est $2^{32}-1$ (4 294 967 295).

ZX81

Chapitre 5

Fonctions

Mathématiquement, une fonction est une règle permettant de donner un nombre (le résultat) en échange d'un autre (l'argument ou l'opérande) et, en réalité, il s'agit donc d'une opération unaire. Le ZX81 contient certaines fonctions, leurs noms sont les mots écrits sur les touches. Par exemple, **SQR** est la « racine carrée » bien connue, et

PRINT SQR 9

donne le résultat 3, c'est-à-dire la racine carrée de 9. Pour obtenir **SQR**, il faut d'abord appuyer sur la touche **FUNCTION** – touche **NEWLINE** en position majuscules. Le curseur se transforme alors en **F**. Appuyez maintenant sur **SQR** (H) : **SQR** apparaît sur l'écran et le curseur redevient **L**. La même méthode permet d'utiliser tous les mots qui sont écrits sous les touches (ces mots étant presque tous des noms de fonctions).

Essayez

PRINT SQR 2

Vous pouvez vérifier la précision du résultat en utilisant

PRINT SQR 2***SQR** 2

qui doit donner le résultat 2. Notez que les deux **SQR** sont calculés avant * et, en pratique, toutes les fonctions (avec une exception – **NOT**) sont élaborées avant les cinq opérateurs +, –, *, / et **. Ici également, vous pouvez contourner cette règle en utilisant des parenthèses :

PRINT SQR (2*2)

donne le résultat 2.

Voici d'autres fonctions (voir la liste complète à l'annexe C). Si vos connaissances mathématiques ne vous permettent pas de les comprendre, c'est sans importance, vous pourrez quand même utiliser votre ordinateur.

SGN Fonction de signe (ne pas confondre avec **SIN**). Le résultat est –1, Ø ou +1 selon que l'argument est négatif, nul ou positif.

ABS Valeur absolue ou modulo. Le résultat est que l'argument est rendu positif, c'est-à-dire

ABS –3.2 = **ABS** 3.2 = 3.2

SIN, **COS**, **TAN**, **ASN** arcsin, **ACS** arccos, **ATN** arctan : Fonctions trigonométriques, en radians et non en degrés.

LN logarithme naturel (base 2,718281828459Ø45.. c'est-à-dire e).

EXP fonction exponentielle
SQR racine carrée
INT partie entière. Cette fonction est toujours arrondie par défaut de sorte que **INT** 3.9 = 3 et **INT** –3.9 = – 4. (Un entier est un nombre entier qui peut éventuellement être négatif.)
PI = 3,14159265358979... Longueur de la circonférence d'un cercle par rapport à son diamètre. **PI** n'a pas d'argument. (Dix chiffres seulement sont mémorisés dans l'ordinateur et seuls les huit premiers sont affichés sur l'écran.)
RND Cette fonction est également dépourvue d'argument. Elle donne un nombre aléatoire entre "Ø" (valeur possible) et "1" (valeur impossible).

Pour reprendre le jargon du dernier chapitre, toutes ces fonctions à l'exception de **PI** et **RND** sont des opérations unaires de priorité 11. (**PI** et **RND** sont des opérations nullaires parce qu'elles n'ont pas d'opérande.)

Les fonctions trigonométriques ainsi que **EXP, LN** et **SQR** sont généralement calculées avec une précision de huit chiffres.

RND et **RAND** : Les deux sont sur la même touche, mais alors que **RND** est une fonction, **RAND** est un mot-clé comme **PRINT. RAND** sert à contrôler le caractère aléatoire de **RND**.

RND n'est pas véritablement aléatoire puisque cette fonction suit une séquence fixe de 65 536 nombres qui sont mélangés de façon à sembler aléatoires (**RND** est pseudo-aléatoire). Vous pouvez utiliser **RAND** pour faire démarrer **RND** à un endroit déterminé de la séquence en appuyant sur **RAND** et en entrant ensuite un nombre entre 1 et 65535 puis en manœuvrant **NEWLINE**. Il n'est pas très important de savoir où un nombre donné démarre **RND** car ce même nombre placé après **RAND** fait toujours démarrer **RND** au même endroit. Par exemple, entrez :

RAND 1 (et **NEWLINE**)

et ensuite

PRINT RND

et entrer ces deux lignes alternativement à plusieurs reprises. (N'oubliez pas d'utiliser **FUNCTION** pour obtenir **RND**.) Le résultat de RND sera toujours Ø.ØØ22735596 ce qui n'est pas vraiment une séquence aléatoire.

RAND Ø

(vous pouvez d'ailleurs omettre le Ø) a un comportement légèrement différent : ce mot-clé décide de l'endroit auquel **RND** démarre en fonction du temps écoulé depuis la mise sous tension du ZX81 et, par conséquent, le résultat doit être véritablement aléatoire.
Nota : Dans certains dialectes du BASIC, l'argument d'une fonction doit toujours être entre parenthèses. Ceci ne s'applique pas au BASIC du ZX81.

Résumé

Instruction : **RAND**

Fonctions : **SGN, ABS, SIN, COS, TAN, ASN, ACS, ATN, LN, EXP, SQR, INT, PI, RND**

FUNCTION

Exercices

1. Pour obtenir des logarithmes naturels (base 1Ø) qui sont ceux que vous recherchiez dans une table de logarithmes, divisez le logarithme naturel par **LN** 1Ø. Par exemple, pour calculer log 2 :

PRINT LN 2/**LN** 1Ø

donne le résultat Ø.3Ø1Ø3.

Essayez de faire des multiplications et des divisions avec des logarithmes en vous servant du ZX81 comme d'un ensemble de tables de logarithmes (pour les logarithmes inverses, voir l'exercice 3 du chapitre 2). Contrôler les résultats en utilisant * et / qui sont à la fois plus faciles, plus rapides, plus précis et, en général, doivent être préférés.

2. **EXP** et **LN** sont des fonctions mutuellement inverses ce qui veut dire que si vous appliquez la première et ensuite l'autre, vous revenez à votre première valeur numérique. Exemple :

LN EXP 2 = **EXP LN** 2 = 2

Ceci est également vrai pour les fonctions **SIN** et **ASN**, pour **COS** et **ACS**, ainsi que pour **TAN** et **ATN**. Vous pouvez utiliser cet exercice pour vérifier la précision de calcul de l'ordinateur pour ces fonctions.

3. π radians égalent 180°. Pour passer de degrés en radians, vous divisez par 180 et vous multipliez par π : ainsi

PRINT TAN (45/180***PI**)

donne tan 45° (1).

Pour transformer des radians en degrés, il faut diviser par π et multiplier par 180.

4. Essayez

PRINT RND

quelques fois pour voir de quelle façon varie le résultat. Pouvez-vous détecter une règle de variation ? (c'est peu vraisemblable).

Comment utiliseriez-vous **RND** et **INT** pour obtenir un nombre entier aléatoire entre 1 et 6, afin de représenter le jet d'un dé ? (Réponse : **INT** (**RND** * 6) +1.).

5. Vérifier cette règle :
   Supposons que vous choisissiez un nombre entre 1 et 872 et que vous entriez ensuite :

   RAND suivi de votre nombre (et **NEWLINE**)

   La prochaine valeur de **RND** doit être

   (75 * (votre nombre + 1) − 1) / 65536

6. (Pour les mathématiciens seulement)
   Soit p un facteur premier important et soit une racine primitive modulo p.
   Si b_i est le reste de a^i modulo p ($1 \leqslant b_i < p - 1$), la séquence

$$\frac{b_i - 1}{p - 1}$$

est une séquence cyclique de p − 1 nombres distincts de la fourchette Ø à 1 (1 exclu). Si a a été choisi correctement, les nombres peuvent sembler aléatoires.

65537 est un facteur premier de Mersenne, $2^{16}-1$. Utilisez ce facteur et la loi de réciprocité quadratique de Gauss pour démontrer que 75 est une racine primitive modulo 65537.

Le ZX81 utilise p = 65537 et a = 75 et mémorise après les b_i-1 en mémoire. La fonction **RND** provoque le remplacement en mémoire de $b_i - 1$ par $b_{i+1} - 1$ ce qui donne le résultat $(b_{i+1} - 1) / (p-1)$. **RAND** n (avec $1 \leqslant n \leqslant 65535$) rend b_i égal à n + 1.

7. **INT** est toujours arrondi par défaut. Pour arrondir à l'entier le plus proche, ajoutez d'abord Ø,5. Exemple :

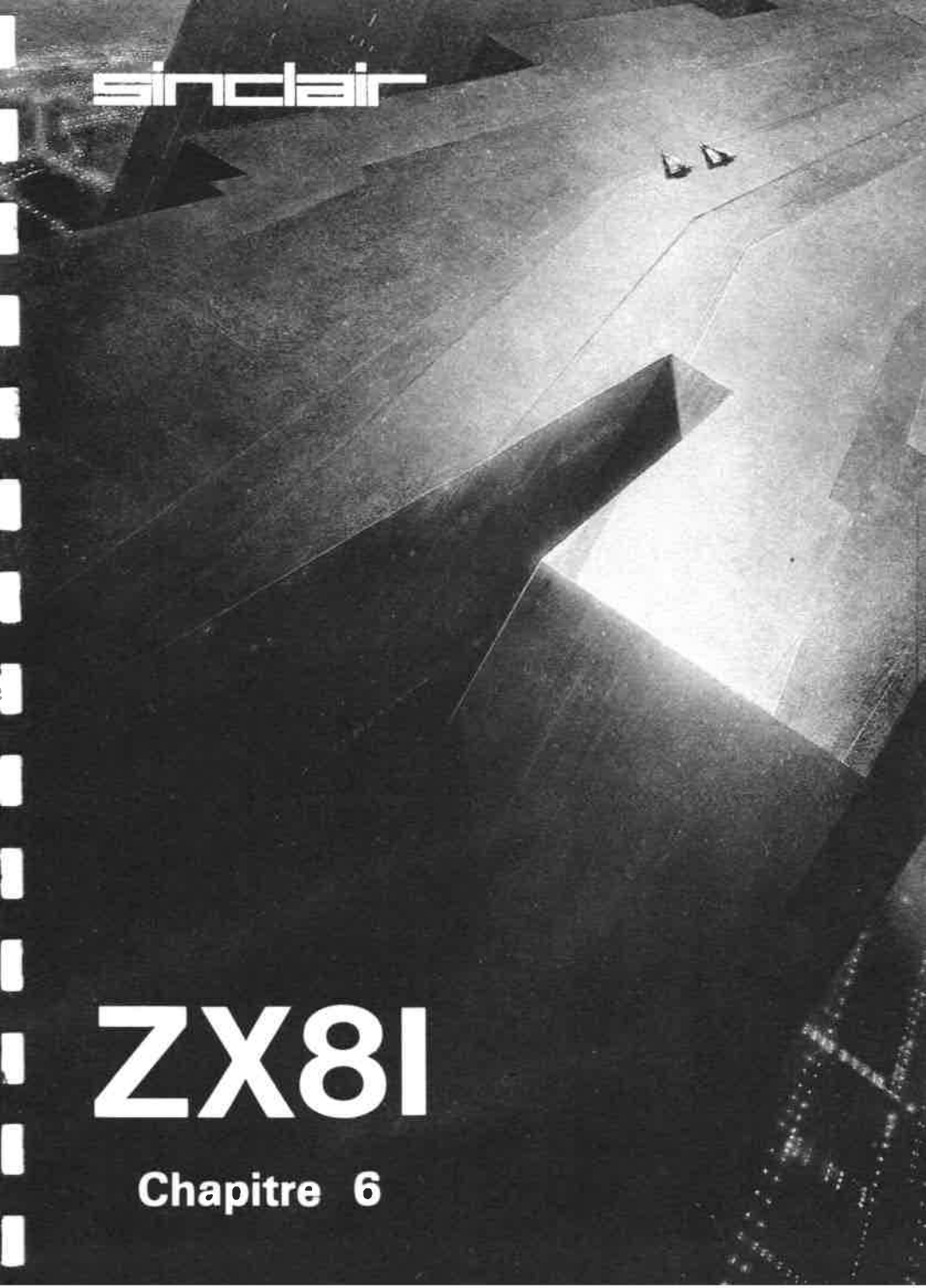
INT (2.9+Ø.5) = 3	**INT** (2.4+Ø.5) = 2
INT (−2.9+Ø.5) = −3	**INT** (−2.4+Ø.5) = −2

Comparez ce qui précède et les résultats que vous obtenez si vous n'ajoutez pas Ø.5.

8. Essayez

   PRINT PI, **PI** −3, **PI** −3.1, **PI** −3.14, **PI** −3.141

Cet exercice montre la précision de mémorisation de π par l'ordinateur.



ZX81

Chapitre 6

Chapitre 6

Variables

Vous vous dites peut-être «Ma calculatrice peut stocker un nombre et s'en souvenir plus tard». Est-ce que le ZX81 peut en faire autant ?

OUI. En réalité, grâce à l'instruction **LET**, le ZX81 peut conserver des centaines de nombres. Supposons que les œufs coûtent 5,80 F la douzaine et que vous vouliez vous souvenir de cette information. Entrez :

LET OEUFS = 5.80 (et bien entendu, **NEWLINE**)

En premier lieu, l'ordinateur a réservé un endroit «dans lui-même» pour que vous puissiez y ranger un nombre et, ensuite, il a désigné cet endroit par le nom «OEUFS» pour que vous puissiez le citer ultérieurement. Cette combinaison d'un espace mémoire et d'un nom est appelée variable. Ensuite, il a rangé le nombre 5,80 dans cet espace : nous disons qu'il a donné la valeur 5,80 à la variable (dont le nom est) OEUFS. OEUFS est une variable numérique puisque sa valeur est un nombre.

Si vous voulez savoir combien coûtent les œufs, il faut entrer

PRINT OEUFS

Si vous voulez connaître le coût d'une demi douzaine d'œufs, vous entrez

PRINT OEUFS/2

et si vous voulez aller plus avant pour connaître le carré du sinus du prix d'un œuf, vous pouvez entrer

PRINT COS(OEUFS/12)**2

Vous devez vous dire que tout ceci est très facile, mais que le prochain problème va être de savoir ce que vous devez faire lorsque vous allez apprendre que les œufs sont passés à 6,10 F la douzaine.

Ne perdez pas de temps, entrez

LET OEUFS = 6.10

Cette opération ne réserve pas un autre espace mémoire, elle remplace l'ancienne valeur 5,80 par la nouvelle 6,10. Maintenant, vous pouvez entrer

PRINT OEUFS

en sachant que l'information que vous allez obtenir correspond bien au dernier prix de ce produit.

Entrez maintenant :

PRINT LAIT

Le compte-rendu que vous allez obtenir est 2/Ø et, si vous vous reportez à l'indication

"2" de l'annexe B, vous verrez que ce chiffre signifie « variable non trouvée ». Puisque vous ne le lui avez pas indiqué, l'ordinateur n'a pas la moindre idée du prix du lait. Entrez maintenant

LET LAIT = 1.85

et tout rentre dans l'ordre.
(Entrez

PRINT LAIT

à nouveau.)

Il n'est pas nécessaire de désigner les variables par des noms de produits alimentaires, vous pouvez utiliser des combinaisons de lettres ou de chiffres mais le premier caractère doit toujours être une lettre. Vous pouvez également utiliser des espaces pour faciliter la lecture, mais ils ne sont pas comptés comme une partie du nom.

Par exemple, des variables peuvent être désignées par les noms suivants :

UN KILO DE POMMES
RADIO 3
RADIO 33
X
K9P

mais les exemples suivants sont interdits :

3 OURS (commence par un chiffre)
TALBOT ? (? n'est ni une lettre, ni un chiffre)
BLANC SUR NOIR (les caractères en vidéo inversée sont interdits).
JEAN-PIERRE (le trait d'union n'est ni une lettre, ni un chiffre).

Entrez maintenant

CLEAR

et

PRINT OEUFS

A nouveau, le compte-rendu que vous obtenez est "2" (Variable non trouvée). **CLEAR** a pour effet de libérer tout l'espace mémoire qui avait été réservé pour des variables et, par conséquent, les variables se présentent maintenant comme si elles n'avaient jamais été définies. La mise hors tension de l'ordinateur a également ce résultat, mais, dans ce dernier cas, la machine ne se souvient absolument de rien lorsqu'elle est remise sous tension.

Une expression peut contenir le nom d'une variable à tous les endroits auxquels peut figurer un nombre.

Nota : Dans certaines versions du langage BASIC, vous pouvez omettre le libellé **LET** et, par exemple, entrer :

OEUFS = 58

Cette possibilité n'existe pas sur le ZX81. De toutes les façons, il serait extrêmement difficile d'entrer ce message.

Dans d'autres versions encore, seuls les deux premiers caractères d'un nom sont vérifiés. Par conséquent, RADIO 3 et RADIO 33 seraient traités comme un seul et même nom ; enfin, dans d'autres encore, le nom d'une variable doit être composé d'une lettre suivie d'un chiffre. Le ZX81 ne comporte aucune des limitations précédentes. Toutefois, dans d'autres versions encore du langage BASIC, si une variable n'est pas encore apparue à gauche de l'instruction **LET**, elle est considérée implicitement comme ayant la valeur Ø. Comme vous l'avez vu ci-dessus avec **PRINT** LAIT, il n'en est pas ainsi dans notre machine.

Résumé

Variables
Instructions : **LET, CLEAR**

Exercices

1. Pourquoi les noms de variables doivent-ils commencer par une lettre ?

2. Si vous n'avez pas l'habitude de travailler avec des puissances (**, H en position majuscules) faites l'exercice.

A son niveau le plus élémentaire, 'A**B' signifie 'A multiplié par lui même B fois' ; bien évidemment, cette expression n'a de sens que si B est un nombre entier positif. Pour trouver une définition applicable aux autres valeurs de B, voyons maintenant la règle :

A ** (B + C) = A**B * A**C

Vous n'avez sans doute pas besoin d'autres renseignements pour être convaincu que cette opération est possible lorsque B et C sont des nombres entiers positifs mais, si nous décidons que cette opération doit donner un résultat exact même lorsque B et C ne sont pas des entiers positifs, nous sommes obligés d'accepter les hypothèses suivantes :

A ** Ø = 1
A ** (–B) = 1/A**B
A ** (1/B) = la Bième racine de A

et

A ** (B*C) = (A**B)** C

Si vous n'avez jamais été confronté auparavant à ce qui précède, ne tentez pas de vous en souvenir ; rappelez-vous seulement que :

A ** −1 = 1/A

et

A ** (1/2) = la racine carrée de A

et, vous vous rendrez compte plus tard que lorsque vous aurez bien assimilé cette partie, le reste vous semblera évident.

Faites maintenant quelques expériences sur ce qui précède en disant à l'ordinateur d'imprimer différentes expressions contenant ** :
par exemple :

PRINT 3**(2+Ø), 3**2*3**Ø
PRINT 4**−1,1/4

3. Entrez

LET E = **EPX** 1

E a la valeur 2.718281828, la base des logarithmes naturels. Vérifiez la règle

EXP un nombre = E ** le nombre

pour différents nombres.

ZX81

Chapitre 7

Chaînes

L'une des opérations que le ZX81 est capable de faire alors que les calculatrices ne le peuvent pas est de travailler sur du texte. Entrez

PRINT "SALUT, JE SUIS VOTRE ZX81." (" est P en position majuscules.)

La formule de politesse entre guillemets est appelée chaîne (ce qui veut dire une chaîne de caractères), elle peut contenir le caractère de votre choix à l'exception du guillemet de la chaîne. (Mais vous pouvez utiliser la prétendue image du guillemet " " (Q en position majuscules) qui est écrite sous la forme " par une instruction **PRINT**.)

L'une des fautes de frappe les plus courantes dans le cas des chaînes est d'oublier l'un des guillemets. Dans ce cas, la machine vous renvoie la marque **S**.

Si vous écrivez des valeurs numériques, vous pouvez utiliser des chaînes pour expliquer la signification des nombres. Par exemple, entrez

LET OEUFS = 6.1

et ensuite

PRINT "LE PRIX DES OEUFS EST" ; OEUFS ; "FRANCS LA DOUZAINE."

(Ne vous inquiétez pas si vous allez plus loin que la fin de la ligne.)

Cette instruction donne trois informations (les éléments de **PRINT**), à savoir la chaîne "LE PRIX DES OEUFS EST", le nombre 6.1 (la valeur de la variable OEUFS) et enfin la chaîne "FRANCS LA DOUZAINE." En réalité, vous pouvez écrire PRINT un nombre quelconque d'éléments, un nombre quelconque de chaînes et de nombres (ou d'expressions), mais il ne faut pas oublier de noter que les espaces d'une chaîne en font partie au même titre que les lettres. Ils ne sont pas ignorés, même à la fin.

Les chaînes vous permettent de faire un grand nombre de choses.

1. Vous pouvez les affecter à des variables. Toutefois, le nom de la variable doit être particulier pour indiquer que sa valeur est une chaîne et non un nombre : la variable doit être désignée par une seule lettre suivie du signe $ (U en position majuscules). Par exemple, entrez

LET A$ = "DEUX PLATS"

et

PRINT A$

2. Vous pouvez les additionner. Cette opération est souvent appelée concaténation, ce qui veut dire chaînage et c'est exactement de cela qu'il s'agit. Essayez

PRINT "G" + "N OBJETS"

Toutefois, vous ne pouvez soustraire, multiplier ou diviser des chaînes ni les élever à des puissances.

3. Vous pouvez appliquer des fonctions à des chaînes pour obtenir des valeurs numériques et vice versa.

LEN s'applique à une chaîne et donne un résultat qui est sa longueur. Par exemple, **LEN** "FROMAGE" = 7.
VAL s'applique à une chaîne et le résultat est ce que donne la chaîne lorsqu'elle est évaluée comme une expression arithmétique. Par exemple (si A = 9), **VAL** "1/2+**SQR**A" = 3.5. Si la chaîne à laquelle **VAL** s'applique contient des variables, deux règles entrent en jeu.

(i) Si la fonction **VAL** fait partie d'une expression plus importante, elle doit être en première position ; par exemple, 1Ø **LET** X = **VAL** + 7"Y" doit être modifiée pour devenir 1Ø **LET** X = **VAL** "Y" +7.

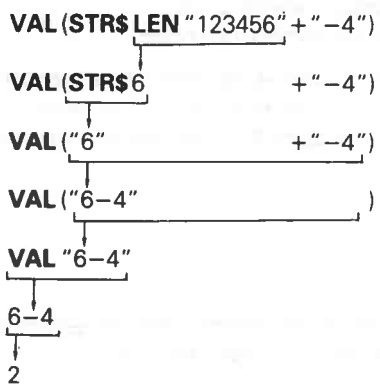
(ii) **VAL** peut seulement figurer dans la première coordonnée d'une instruction **PRINT AT**, **PLOT** ou **UNPLOT** (voir les chapitres 17 et 18).
Exemple :
1Ø **PLOT** 5, **VAL** "X" doit être modifiée pour devenir
1Ø **LET** Y = **VAL** "X"
15 **PLOT** 5, Y

STR$ s'applique à un nombre et le résultat est ce qui apparaîtrait sur l'écran si le nombre était affiché par une instruction **PRINT**. Par exemple **STR$** 3.5 = "3.5".

4. Comme pour les nombres, vous pouvez combiner ces fonctions pour faire des expressions de chaînes, comme

VAL (**STR$** **LEN** "123456" + " –4")

dont l'évaluation est la suivante



Résumé
Chaînes
Opération : + (pour les chaînes)
Fonctions : **LEN, VAL, STR$**

Exercices

1. Entrez

```
LET A$="2+2"
```

et ensuite

```
PRINT A$;"=",VAL A$
```

Tentez maintenant de transformer A$ en quelque chose de plus compliqué ayant la même fonction, par exemple

```
LET A$="ATN 1*4"
```

(Dans ce cas, le résultat doit être π.)

2. La chaîne " " sans caractères est appelée chaîne vide ou nulle. C'est la seule chaîne dont la longueur est Ø. Souvenez-vous que les espaces sont comptés et qu'une chaîne vide n'est pas la même chose qu'une chaîne contenant des espaces.

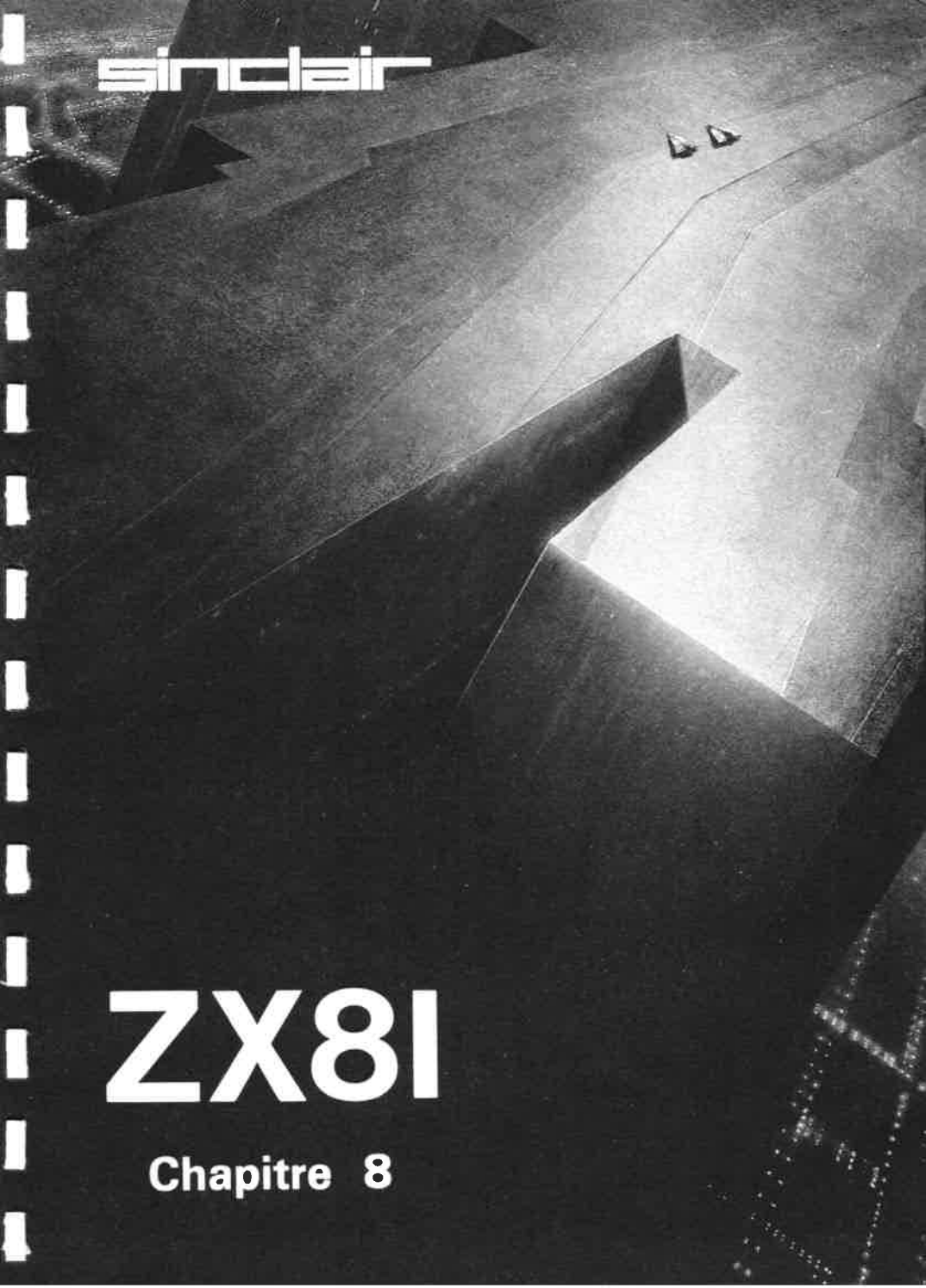
Attention de ne pas vous tromper et de la prendre pour l'image " " (une seule manœuvre, touche Q en position majuscules). Cette marque particulière a été conçue pour tenir compte du fait que vous ne pouvez pas écrire un guillemet classique de chaîne au milieu d'une chaîne (Pourquoi pas ?). Lorsque cette image apparaît dans une chaîne commençant et se terminant par un guillemet (par exemple dans la liste d'un programme), elle se présente sous la forme de deux symboles guillemet pour la distinguer du guillemet normal ; mais, lorsqu'elle est affichée par une instruction **PRINT**, elle n'est plus constituée que d'un symbole guillemet.

Essayez

```
PRINT "X";"";"X","""","""";"";"""
```

3. Si vous avez envie d'humilier votre machine, entrez

```
PRINT "2+2=";2+1
```



ZX81

Chapitre 8

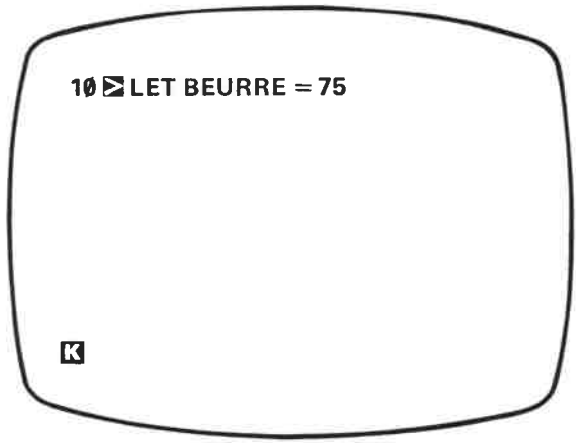
Chapter 8

Programmation de l'ordinateur

Vous allez enfin pouvoir écrire un programme ordinateur. Mettez l'ordinateur hors tension/sous tension pour vous assurer qu'il est bien remis à zéro. Maintenant, entrez

10 **LET** BEURRE=75 (& **NEWLINE**)

Sur l'écran, vous obtenez :



Cette présentation est différente de celle des OEUFS du chapitre 6 ; si vous entrez

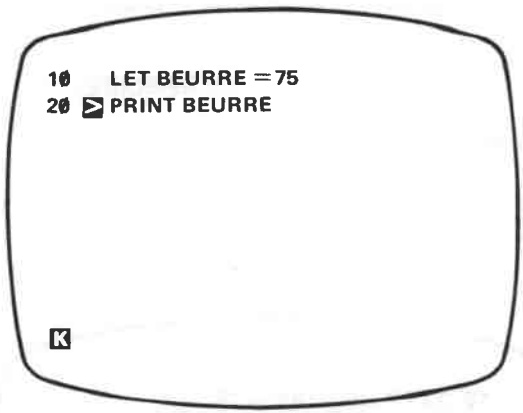
PRINT BEURRE

vous verrez (par le compte-rendu 2) que la variable BEURRE n'a pas été établie. (Appuyez à nouveau sur **NEWLINE**, l'écran va devenir à nouveau semblable à l'illustration ci-dessus).

L'instruction **LET** ayant un numéro (10) placé devant elle, l'ordinateur ne l'a pas exécutée directement, il l'a préservée pour la suite. 10 est son numéro de ligne, il sert aux citations de la même façon que les noms servent à citer les variables. Plusieurs instructions mémorisées de ce type sont appelées programme. Entrez maintenant

20 **PRINT** BEURRE

Et l'écran doit présenter les indications suivantes :



Ceci est la liste de votre programme. Pour que ce programme soit exécuté, vous devez entrer

RUN (n'oubliez pas d'appuyer sur **NEWLINE**)

et le résultat 75 apparaît dans le coin supérieur gauche de l'écran. Dans le coin inférieur gauche, le compte-rendu Ø/2Ø apparaît. Comme vous le savez, Ø veut dire "tout va bien" et 2Ø désigne le numéro de la ligne à laquelle s'est terminée l'exécution. Appuyez à nouveau sur **NEWLINE** pour faire réapparaître la liste.

Notez que les instructions ont été exécutées dans l'ordre de leurs numéros.

Supposons maintenant que vous vous rappelez enfin que vous devez enregistrer le prix de la levure. Il faut entrer :

15 **LET** LEVURE=4Ø

et cette information est enregistrée. Cette opération aurait été beaucoup plus difficile si les deux premières lignes avaient été numérotées 1 et 2 au lieu de 1Ø et 2Ø (les numéros des lignes doivent être des nombres entiers de la fourchette 1-9999), c'est pour cette raison qu'il est toujours recommandé à la première entrée d'un programme de laisser des numéros libres entre les numéros des lignes.

Maintenant, vous devez modifier la ligne 2Ø qui devient

2Ø **PRINT** BEURRE,LEVURE

Vous pourriez entrer la totalité des données, mais il est possible et plus facile d'utiliser celles qui se trouvent déjà dans la machine. Vous voyez le petit signe ▶ à la ligne 15 ? C'est le curseur du programme, il indique la ligne en cours de ce programme. Appuyez sur la touche ⟲ (6 en position majuscules) et le curseur passe à la ligne 2Ø. (La touche permet de le ramener à la position précédente.) Appuyez maintenant sur la touche **EDIT** (1 en position majuscules), et une reproduction de la ligne 2Ø apparaît en bas de l'écran.

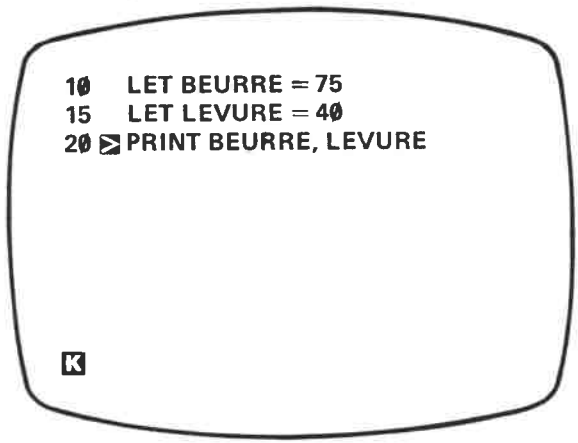
Appuyez sept fois sur la touche ◊ pour amener le curseur ■ à la fin de la ligne et entrez ensuite

LEVURE (sans manœuvrer **NEWLINE**)

Maintenant, la ligne du bas doit être

2Ø PRINT BEURRE,LEVURE ■

Appuyez sur **NEWLINE** et la nouvelle ligne remplace l'ancienne ligne 2Ø. Maintenant, l'écran contient les indications suivantes :



Exécutez (**RUN**) ce programme pour provoquer l'affichage des deux prix.

(Voici un petit truc utile de **EDIT** qui vous permet d'effacer tout le bas de l'écran. Appuyez sur **EDIT** et la ligne en cours est sortie du programme et mise en position basse, remplaçant ce que vous vouliez supprimer. Si vous appuyez maintenant sur **NEWLINE**, la ligne est remise dans le programme qui n'a donc subi aucune modification et le bas de l'écran est complètement vide, à l'exception du curseur.)

Imaginons maintenant que vous soyez distrait et que vous entriez

12 **LET** LEVURE=4Ø

Cette indication est ramenée dans le programme et vous pouvez donc réaliser votre erreur. Pour supprimer cette ligne inutile, entrez

12 (en appuyant ensuite sur **NEWLINE**)

Vous remarquerez sans doute que le curseur du programme a disparu. Vous en déduirez qu'il est probablement caché entre les lignes 1Ø et 15, et par conséquent, si vous appuyez sur ◊, il remonte à la ligne 1Ø alors que si vous appuyez sur ◊, il descend à la ligne 15.

Enfin, entrez

LIST 15

L'écran affiche maintenant :

```
15 LET LEVURE=4Ø
2Ø   PRINT BEURRE,LEVURE
```

La ligne 1Ø ne se trouve plus sur l'écran, mais elle est encore dans votre programme — Vous pouvez vous en assurer en appuyant à nouveau sur **NEWLINE**. Les seuls effets de **LIST** 15 sont de produire une liste à partir de la ligne 15 et d'amener le curseur du programme à cette même ligne 15.

LIST

utilisée seule fait commencer la liste au début du programme.

Résumé

Programmes
Edition des programmes à l'aide ⇦, ⇨, et **EDIT**.
Instructions : **RUN**, **LIST**

Exercices

1. Modifiez le programme afin d'obtenir non seulement les deux prix, mais également des messages précisant à quels produits correspond chaque prix.

2. Utilisez la touche **EDIT** pour changer le prix du beurre.

3. Exécutez le programme et entrer

   PRINT BEURRE,LEVURE

   Les variables sont encore présentes alors que le programme est terminé.

4. Entrez

   12 (et **NEWLINE**)

Dans ce cas également, le curseur du programme demeure caché entre les lignes 1Ø et 15. Appuyez maintenant sur **EDIT** pour faire descendre la ligne 15 ; lorsque le curseur du programme est caché entre deux lignes, l'instruction **EDIT** fait descendre la deuxième. Appuyez sur **NEWLINE** pour vider le bas de l'écran.

Entrez maintenant

```
30
```

Cette fois, le curseur est caché après la fin du programme ; si vous appuyez sur **EDIT**, la ligne 20 est descendue.

5. Mettez une instruction **LIST** dans le programme afin qu'il établisse lui-même sa liste lorsqu'il est exécuté.

ZX81

Chapitre 9

Continuons la programmation

Entrez

NEW

Cette fonction efface les anciens programmes et variables que contient le ZX81. (L'instruction **NEW** ressemble beaucoup à **CLEAR**, mais cette dernière efface seulement les variables.) Maintenant, entrez le programme suivant :

```
1Ø REM CE PROGRAMME EXTRAIT LES RACINES CARRÉES
2Ø INPUT A
3Ø PRINT A, SQR A
4Ø GOTO 2Ø
```

(Vous devrez taper vous-même la plupart des espaces de la ligne 1Ø.)

Exécutez maintenant le programme. L'écran semble se vider et rien ne se produit ; regardez maintenant le curseur qui se trouve dans le coin inférieur gauche ; contrairement au K que vous attendiez, le curseur est un L, ce qui veut dire que la machine est passée en mode entrée. Ceci est la conséquence de l'instruction **INPUT** de la ligne 2Ø. La machine attend que vous entriez un nombre (ou une expression), elle ne va plus rien faire avant cette entrée. Après, vous obtiendrez

2Ø **LET** A= ... ce que vous avez indiqué

Attention, et qu'est donc devenue la ligne 1Ø ?. Il semble que l'ordinateur l'a complètement ignorée. C'est exact. **REM** de la ligne 1Ø veut dire remarque ou rappel et sa seule fonction est de vous rappeler ce que fait le programme. Une instruction **REM** est composée de **REM** suivi de ce que vous voulez, mais elle est ignorée par la machine.

Bien, nous sommes donc en mode entrée pour la ligne 2Ø. Entrez un nombre quelconque, 4 par exemple, et appuyez sur **NEWLINE**. Le chiffre 4 et sa racine carrée apparaissent sur l'écran, mais n'en déduisez pas que tout est terminé. Non, la machine semble vouloir un autre nombre. Ceci résulte de la ligne 4Ø dont le contenu **GOTO** 2Ø veut dire "allez à 2Ø". Au lieu de terminer le programme et de s'arrêter, l'ordinateur retourne à la ligne 2Ø et recommence. Vous devez donc entrer maintenant un autre nombre (2 par exemple, mais n'oubliez pas qu'il doit être positif).

Après quelques exercices similaires, vous allez sans doute vous demander si la machine ne va pas se lasser de ce jeu ; n'en croyez rien. Dès qu'elle manifeste une nouvelle fois son côté insatiable, en demandant un autre nombre, entrez **STOP** (A en position majuscules) ; elle comprendra très bien de quoi il s'agit. L'ordinateur renvoie maintenant le compte-rendu D/2Ø — regardez la signification du D dans la liste des comptes-rendus de l'annexe B. 2Ø est la ligne à laquelle elle attendait une entrée lorsque vous l'avez arrêtée.

Si vous vous rappelez maintenant d'autres nombres dont vous voulez calculer la racine carrée, entrez

CONT

(l'abréviation de CONTINUER) ; l'ordinateur efface l'écran et vous demande un autre nombre.

Avec **CONT**, l'ordinateur se souvient du numéro de ligne figurant dans le dernier compte-rendu qu'il vous a transmis et qui contenait un code autre que Ø et il saute à cette ligne. Puisque le dernier compte-rendu était D/2Ø (et que D est différent de Ø) dans notre cas, **CONT** est identique à **GOTO** 2Ø.

Entrez maintenant plusieurs nombres jusqu'au moment où l'écran est presque plein. Dès que l'écran est plein, l'ordinateur s'arrête et vous envoie le compte-rendu 5/3Ø. 5 signifie "écran plein", et 3Ø est le numéro de l'instruction **PRINT** que la machine tentait d'exécuter lorsqu'elle s'est rendue compte qu'il n'y avait plus de place. Entrez à nouveau

CONT

pour effacer l'écran et, cette fois, continuez les opérations. **CONT** signifie **GOTO** 3Ø.

Notez que l'écran est effacé non pas parce qu'il s'agit d'une instruction **CONT**, mais parce que nous sommes en présence d'une commande. Toutes les commandes (à l'exception de **COPY**, dont la description figure au chapitre 2Ø) provoquent d'abord l'effacement de l'écran.

Lorsque vous en aurez assez de ce petit jeu, arrêtez le programme à l'aide de **STOP** et appuyez sur **NEWLINE** pour obtenir la liste.

Regardez l'instruction **PRINT** de la ligne 3Ø. Chaque fois que le couple de nombres A et **SQR**A est écrit, il apparaît sur une nouvelle ligne parce que l'instruction **PRINT** ne se termine pas par une virgule ou par un point virgule. Dans ce cas, l'instruction **PRINT** suivante commence à écrire sur une nouvelle ligne. (Par conséquent, pour insérer une ligne vide, il faut utiliser une instruction **PRINT** dans laquelle il n'y a rien à écrire — c'est-à-dire **PRINT** seulement.)

Toutefois, une instruction **PRINT** peut se terminer par une virgule ou par un point virgule et, dans ce cas, la prochaine instruction **PRINT** poursuit l'écriture comme si les deux ne composaient qu'une seule instruction longue.

Par exemple, en utilisant des virgules, remplacez 3Ø par

3Ø **PRINT** A,

et exécutez le programme pour voir comment plusieurs instructions **PRINT** successives peuvent s'écrire sur la même ligne tout en étant réparties sur deux colonnes.

Par contre, avec des points virgules et

3Ø **PRINT** A;

tous les éléments sont rassemblés.

Essayez également

3Ø **PRINT** A

Entrez maintenant les lignes suivantes :

```
100 REM CE PROGRAMME MESURE LES CHAINES
110 INPUT A$
120 PRINT A$,LEN A$
130 GOTO 110
```

Ce programme est séparé du dernier mais vous pouvez les conserver tous les deux dans la machine en même temps. Pour exécuter le nouveau programme, entrez

```
RUN 100
```

Ce programme entre une chaîne à la place d'un nombre, et écrit ensuite la chaîne et sa longueur. Entrez

```
CHAT          (et NEWLINE, comme d'habitude)
```

Puisque l'ordinateur attend une chaîne, il écrit deux guillemets de chaîne – à titre de rappel et pour vous économiser aussi quelques manœuvres. Mais vous ne devez pas nécessairement vous limiter à des constantes chaîne (des chaînes explicites avec guillemets de début et de fin et tous leurs caractères entre ces guillemets) ; l'ordinateur évalue toutes les expressions chaîne, par exemple celles contenant des variables de chaîne. Dans ce cas, vous allez peut-être être obligé d'effacer les guillemets que l'ordinateur a mis sur l'écran. Essayez d'effacer les guillemets. Pour faire cette opération, appuyez sur ◊ et **RUBOUT** deux fois ; ensuite, entrez

```
A$
```

Puisque A$ a encore la valeur "CHAT", l'ordinateur répond encore CHAT 4.
Entrez maintenant

```
A$
```

mais, cette fois, sans effacer les guillemets de la chaîne. Maintenant A$ a la valeur "A$" et la réponse est 2.

Si vous voulez utiliser **STOP** pour entrer une chaîne, vous devez d'abord remettre le curseur au début de la ligne à l'aide de la touche ◊.

Revenons maintenant à l'exemple **RUN** 100 précédent. Puisque cet exemple provoque un saut à la ligne 100 n'auriez-vous pas pu utiliser **GOTO** 100 à la place ? Dans le cas qui nous préoccupe, il se trouve que la réponse est positive, mais il y a quand même une différence. **RUN** 100 commence d'abord par effacer toutes les variables (comme **CLEAR** au chapitre 6) et, après cette opération, se comporte comme **GOTO** 100. Par contre, **GOTO** 100 n'efface rien. Et, il est bien possible que dans certains cas, vous souhaitiez exécuter un programme sans effacer les variables. Dans ce cas, il faut utiliser **GOTO** alors que **RUN** pourrait avoir des conséquences désastreuses ; c'est pour cette raison qu'il ne faut pas prendre l'habitude d'appuyer automatiquement sur **RUN** pour exécuter un programme.

Une autre différence réside dans le fait que vous pouvez entrer **RUN** sans numéro de ligne, tandis que **GOTO** doit toujours avoir un numéro.

Ces deux programmes se sont arrêtés parce que vous avez entré **STOP** sur la ligne **INPUT** ; toutefois, il se peut que parfois — par erreur — vous écriviez un programme que vous ne pouvez pas arrêter et qui ne s'arrête pas de lui-même.

Entrez

```
2ØØ GOTO 2ØØ
RUN 2ØØ
```

Ceci semble être tout ce qu'il faut pour créer le mouvement perpétuel si vous ne débranchez pas la machine ; heureusement, il existe un "frein" moins brutal. Il suffit d'appuyer sur la touche **SPACE** qui est surmontée de l'indication "BREAK" (interruption). Le programme s'arrête et lance le compte-rendu D/2ØØ.

Au bout de chaque ligne d'un programme, l'ordinateur tente de savoir si cette touche a été manœuvrée. Dans l'affirmative, la machine s'arrête. La touche **BREAK** peut également être utilisée lorsque vous employez le magnétophone ou l'imprimante.

Vous disposez maintenant d'une série importante d'instructions — **PRINT**, **LET**, **INPUT**, **RUN**, **LIST**, **GOTO**, **CONT**, **CLEAR**, **NEW** et **REM** — qui, pour la plupart, peuvent être utilisées comme commandes ou comme lignes de programme — ceci est vrai pour presque toutes les instructions du BASIC. La seule exception majeure est **INPUT** qui ne peut pas être utilisée comme commande (la machine vous renvoie le compte-rendu 8 si vous essayez ; ceci résulte du fait que dans la machine, la même zone est utilisée pour les commandes et pour les données d'entrée et que, pour une commande **INPUT**, il y aurait une confusion). **RUN**, **LIST**, **CONT**, **CLEAR** et **NEW** ne sont pas très utiles dans un programme, mais il est néanmoins possible de les utiliser.

Résumé

Instructions : **GOTO**, **CONT**, **INPUT**, **NEW**, **REM**, **PRINT**
STOP dans les données d'entrée
BREAK

Exercices

1. Dans le programme de calcul de racines carrées, essayez de remplacer la ligne 4Ø par **GOTO** 5, **GOTO** 1Ø ou **GOTO** 15 — ce remplacement ne devrait pas provoquer de différence perceptible à l'exécution du programme. Si le numéro de ligne d'une instruction **GOTO** cite une ligne qui n'existe pas, le saut se fait au profit de la ligne suivante. Ceci est également vrai pour **RUN**. En réalité, **RUN** sans suffixe numérique signifie **RUN** Ø.

2. Exécutez le programme de calcul de longueur des chaînes et, lorsque l'ordinateur vous demande des informations, entrez

```
X$          (après avoir enlevé les guillemets)
```

Bien entendu, X$ est une variable indéfinie et la machine vous renvoie le compte-rendu 2/11Ø.

Si vous entrez maintenant

LET X$ = "QUANTITE DEFINIE"

(qui comporte son propre compte-rendu Ø/Ø) et

CONT

vous vous rendez compte que vous pouvez utiliser X$ comme donnée d'entrée sans aucun problème.

Le point important de cet exercice est que **CONT** a le même effet que **GOTO** 11Ø. Cette fonction ne tient pas compte du compte-rendu Ø/Ø à cause du code Ø et elle prend son numéro de ligne dans le compte-rendu précédent 2/11Ø. Elle a été conçue dans un esprit utilitaire. Si un programme s'arrête sur une erreur, vous pouvez tenter toutes sortes de manœuvres pour essayer de le remettre en état, et **CONT** peut néanmoins travailler normalement après vos opérations.

3. Essayez ce programme :

```
1Ø INPUT A$
2Ø PRINT A$;" = ";VAL A$
3Ø GOTO 1Ø
```

(voir le chapitre 7, exercice 1.)

Mettez des **PRINT** supplémentaires afin que l'ordinateur précise ce qu'il fait et demande la chaîne d'entrée avec une grande politesse.

4. Ecrivez un programme d'entrée de prix et d'écriture de la TVA (au taux de 15 %). Dans ce cas également, mettez des instructions **PRINT** pour que la machine vous dise ce qu'elle fait. Modifiez ensuite le programme de façon à pouvoir entrer le taux de la TVA (pour permettre un taux zéro ou pour calculer des budgets prévisionnels).

5. Ecrivez un programme dont la fonction sera d'écrire le total courant des nombres que vous entrez. (Suggestion : utilisez deux variables TOTAL — mise à Ø au début et UNITE. Entrez UNITE, ajoutez sa valeur à TOTAL, imprimez les deux résultats et recommencez avec un autre chiffre).

6. Les listes automatiques (celles qui ne sont pas provoquées par une instruction **LIST**) ont pu vous étonner. Si vous entrez un programme de 5Ø lignes composé exclusivement d'instructions **REM**,

```
 1 REM
 2 REM
 3 REM
 :  :
 :  :
49 REM
5Ø REM
```

vous pourrez perfectionner vos connaissances.

La première chose dont il faut vous souvenir est que la ligne en cours (avec ▶) apparaît toujours sur l'écran et, de préférence, près du centre de l'écran.

Entrez

LIST (et **NEWLINE** bien entendu)

appuyez une deuxième fois sur **NEWLINE**. Les lignes 1 à 22 doivent maintenant être sur l'écran. Entrez

23 **REM**

vous devez pouvoir lire maintenant les lignes 2 à 23 sur l'écran ; entrez

28 **REM**

et vous obtenez les lignes 27 à 48. (Dans les deux cas, l'entrée d'une nouvelle ligne a provoqué le déplacement du curseur du programme, entraînant l'élaboration d'une nouvelle liste).

Est-ce que ce qui précède vous paraît un peu arbitraire ? En réalité, la machine essaye de vous donner exactement ce que vous voulez, mais les êtres humains étant des créatures imprévisibles, la machine ne devine pas toujours exactement ce qu'il faut.

L'ordinateur se souvient non seulement de la ligne en cours — celle qui doit apparaître sur l'écran — mais également de la première ligne de l'écran. Lorsqu'il essaye d'établir une liste, il commence par comparer la première ligne et la ligne en cours.

Si la première ligne se trouve après, il est inutile de commencer ici, et la machine utilise donc la ligne en cours comme nouvelle première ligne et établit ensuite sa liste.

Autrement, il tente d'abord de faire la liste en commençant par la première ligne. Si la ligne en cours se trouve aussi sur l'écran, tout va bien ; si la ligne en cours est juste en-dessous de l'écran, il décale la première ligne d'une position vers le bas et retente son opération ; si la ligne en cours est très loin du bas de l'écran, il change la première ligne pour qu'elle devienne la ligne qui précède la ligne en cours.

Pour vous exercer à déplacer la ligne en cours, amusez-vous à entrer

numéro de ligne **REM**

LIST déplace la ligne en cours, mais non la première ligne et, de ce fait, les listes établies ultérieurement peuvent être différentes.

Par exemple, entrez

LIST

pour obtenir la liste de **LIST** et appuyez ensuite à nouveau sur **NEWLINE** pour que la ligne Ø devienne la première ligne. Sur l'écran, vous devez avoir les lignes 1 à 22. Entrez

LIST 22

pour obtenir les lignes 22 à 43 ; lorsque vous appuyez à nouveau sur la touche **NEWLINE**, vous obtenez encore les lignes 1 à 22. Ceci est plus utile pour les programmes courts que pour les programmes longs.

7. Quelle serait l'action de **CONT**, **CLEAR** et **NEW** dans un programme ? Pouvez-vous trouver des emplois intéressants pour ces instructions ?

ZX81

Chapitre 10

Instruction IF (SI...)

Tous les programmes que nous avons étudiés jusqu'à maintenant ont été relativement prévisibles — ils effectuaient directement l'exécution de toutes les instructions et, dans certains cas, revenaient parfois à leur point de départ. Ceci n'est pas toujours utile. En pratique, on attend d'un ordinateur qu'il prenne des décisions et agisse en conséquence ; la machine prend des décisions grâce à l'instruction **IF**.

Remettez l'ordinateur à zéro (avec **NEW**) ; ensuite, entrez et exécutez le petit programme suivant qui est très amusant :

```
10 PRINT "JE VOUS EN RACONTE UNE DROLE"
20 INPUT A$
30 IF A$="NON MERCI" THEN GOTO 200
40 PRINT "COMBIEN DE PATTES A UN CHEVAL ?"
50 INPUT PATTES
60 IF PATTES=6 THEN GOTO 100
70 PRINT "NON, 6, QUATRE VERS L'AVANT" ,"ET DEUX POUR
   RUER".
80 STOP
100 PRINT "OUI,"JE RECOMMENCE ?"
110 GOTO 20
200 PRINT "OK, JE NE RECOMMENCE PAS."
```

Avant de passer à l'étude de l'instruction **IF**, vous devez d'abord examiner les instructions **STOP** de la ligne 80 : une instruction **STOP** arrête l'exécution du programme et provoque le lancement du compte-rendu 9.

Comme vous pouvez le voir maintenant, une instruction **IF** revêt la forme

IF condition **THEN** instruction

Dans d'autres cas, les instructions sont des **GOTO** mais elles pourraient être différentes et même être d'autres instructions **IF**. La condition est quelque chose qui va être résolu pour être confirmé ou infirmé (vrai ou faux) ; si la condition est vérifiée, l'instruction placée après **THEN** est exécutée ; autrement, cette instruction est laissée de côté.

Les conditions les plus utiles comparent deux nombres ou deux chaînes : elles vérifient si les deux nombres sont égaux ou si l'un est plus grand que l'autre ; elles permettent également de vérifier si deux chaînes sont égales ou si l'une précède l'autre en ordre alphabétique. Les opérateurs de comparaison sont les suivants : =, <, >, <=, >=, et <>.

Le signe = que nous avons utilisé deux fois dans le programme (une fois pour des nombres et une fois pour des chaînes) veut dire égal. Ce n'est pas le même que = dans une instruction **LET**.

Le signe < signifie "est inférieur à", par exemple

1<2
−2<−1
et −3<1

sont des comparaisons vraies, mais

1<Ø
et Ø<−2

sont fausses.

Voyons maintenant le fonctionnement de ce qui précède ; écrivons un programme permettant d'entrer des nombres et d'afficher le plus grand parmi tous ceux qui ont été entrés.

```
1Ø  PRINT "NOMBRE","LE PLUS GRAND JUSQU'A MAINTENANT"
2Ø  INPUT A
3Ø  LET LE PLUS GRAND=A
4Ø  PRINT A, LE PLUS GRAND
5Ø  INPUT A
6Ø  IF LE PLUS GRAND<A THEN LET LE PLUS GRAND=A
7Ø  GOTO 4Ø
```

La partie essentielle de ce programme se trouve à la ligne 6Ø qui met à jour LE PLUS GRAND si son ancienne valeur était plus petite que le nouveau nombre A qui vient d'être entré.

> (M en position majuscules) signifie "est supérieur à", c'est-à-dire l'inverse de <. Pour vous souvenir duquel il s'agit, rappelez-vous que la pointe de la flèche désigne le nombre qui est censé être le plus petit.

<= (R en position majuscules — vous ne devez pas entrer < en utilisant les touches < et =) signifie "est inférieur ou égal à" ; par conséquent, la signification est la même que celle de < sauf que la relation demeure vraie lorsque les deux nombres sont égaux : ainsi, 2<=2 est vrai, mais 2<2 ne l'est pas.

>= (Y en position majuscules) signifie "est supérieur ou égal à" c'est-à-dire est semblable à >.

<> (T en position majuscules) signifie "est différent de" ou "n'est pas égal à", c'est-à-dire l'inverse de la signification de = .

Ces six opérateurs de comparaison ont la priorité 5.

Les mathématiciens écrivent habituellement <=, >= et <> sous la forme ≤, ≥ et ≠. Ils écrivent également des relations comme "2<3<4" pour désigner les relations "2<3 et 3<4" mais ce type d'écriture est impossible en BASIC.

Ces relations peuvent être combinées en utilisant les opérateurs logiques **AND**, **OR** et **NOT**.

une relation **AND** une autre relation

est vérifiée quand les deux relations sont vérifiées.

une relation **OR** autre

est vérifiée lorsque l'une des deux relations est vérifiée (ou les deux).

NOT relation

est vérifiée lorsque la relation est fausse et n'est pas vérifiée lorsque la relation est vraie.

Les expressions logiques peuvent être constituées avec des opérateurs de comparaison ainsi qu'avec **AND**, **OR**, et **NOT** de la même façon que les expressions numériques peuvent être composées de nombres et de +, —, etc ; si nécessaire, vous pouvez même utiliser des parenthèses. Les priorités des opérateurs logiques sont 4 pour **NOT**, 3 pour **AND** et 2 pour **OR**.

Puisque (contrairement aux autres fonctions) **NOT** a une priorité relativement basse, son argument ne doit pas comporter de parenthèses sauf s'il contient **AND** ou **OR** ; par conséquent, **NOT** A = B signifie **NOT** (A = B) (c'est-à-dire la même chose que A <> B).

Illustrons ce qui précède en remettant l'ordinateur à zéro et en exécutant le programme suivant.

```
1Ø INPUT F$
2Ø INPUT AGE
3Ø IF F$="X" AND AGE<18 OR F$="AA" AND AGE<14
THEN PRINT
"NE PAS";
4Ø PRINT "LAISSER ENTRER."
5Ø GOTO 1Ø
```

Dans ce programme, F$ représente la classification d'un film —X pour les personnes de 18 ans et plus, AA pour les 14 ans et plus et A ou U pour tous publics ; le programme indique si une personne d'un âge donné peut voir le film.

Enfin, nous pouvons comparer non seulement des nombres, mais également des chaînes. Nous avons vu "=" utilisé dans 'F$="X"', et vous pouvez bien entendu utiliser les cinq autres, <, etc.

Que signifie "inférieur à" pour des chaînes ? En premier lieu, ce symbole ne veut pas dire "plus court que" et c'est là une faute à ne pas commettre. Nous avons précisé qu'une chaîne est inférieure à une autre si elle la précède dans l'ordre alphabétique. Exemples :

"SMITH"	"SMYTHE"
"SMYTHE"	"SMITH"
"BLOGGS"	"BLOGGS-BLACKBERRY"
"MILLE"	"MILLION"
"TCHAIKOWSKY"	"WAGNER"
"DOLLAR"	"ROUBLE"

Toutes les relations ci-dessus sont vérifiées. <= signifie "est inférieur ou égal à" etc., comme pour les nombres.
Nota : Dans certaines versions du langage BASIC — mais pas dans le ZX81 — l'instruction **IF** peut revêtir la forme

IF condition THEN numéro de ligne

La signification de l'instruction ci-dessus est la même que

IF condition **THEN GOTO** numéro de ligne

Résumé

Instructions : **IF**, **STOP**
Opérations : =, <, >, <= , >=, <>, **AND**, **OR**
Fonction : **NOT**

Exercices

1. <> et = sont opposés en ce sens que **NOT** A=B veut dire la même chose que A<>B

et

NOT A<>B veut dire la même chose que A=B

Essayez de vous convaincre que >= est l'opposé de < et que <= est le contraire de > afin de pouvoir vous débarrasser de **NOT** placé devant une relation en transformant cette dernière en son contraire.

Et,

NOT (une première expression logique **AND** une deuxième)

est la même chose que

NOT (la première) **OR NOT** (la deuxième),

et

NOT (une première expression logique **OR** une deuxième)

est identique à

NOT (la première) **AND NOT** (la deuxième).

Grâce à ce qui précède, vous pouvez faire évoluer les **NOT** à l'aide de parenthèses, jusqu'au moment où tous les noms s'appliquent aux relations, ce qui vous permet de vous en débarrasser. Ainsi, logiquement parlant, **NOT** est inutile. Toutefois, **NON** peut néanmoins faciliter votre travail en améliorant la lisibilité d'un programme.

2. Le langage BASIC s'écarte parfois de la langue parlée. Considérons par exemple la clause en langue française " si A n'est pas égal à B ou C". Comment feriez-vous pour écrire cette clause en BASIC ? (La réponse est NON).

'**IF** A <> B **OR** C' ni '**IF** A <> B **OR** A <> C'

Ne vous inquiétez pas si vous ne comprenez pas les exercices 3, 4 et 5 ; les points qu'ils traitent sont d'un niveau relativement élevé.

3. (Pour les experts)
Essayez

PRINT 1=2, 1<>2

dont, au premier abord, vous pourriez attendre une erreur de syntaxe. En réalité, en ce qui concerne l'ordinateur, il n'existe rien de tel qu'une valeur logique.

(i) = , <, >, <=, >= et <> sont des opérations binaires à valeur numérique de priorité 5. Le résultat est 1 (c'est-à-dire vérifié = VRAI) si la relation est confirmée et Ø (c'est-à-dire infirmé = FAUX) si elle ne l'est pas.

(ii) dans

IF condition **THEN** instruction

la condition peut être une expression numérique quelconque. Si sa valeur est Ø, elle est considérée comme fausse et n'importe quelle autre valeur est considérée vraie. Ainsi, l'instruction **IF** signifie exactement la même chose que

IF condition <> Ø **THEN** instruction

(iii) **AND**, **OR** et **NOT** sont également des opérations à valeur numérique.

X **AND** Y a la valeur { X si Y est non nul (ce qui veut dire vrai)
Ø si Y est nul (ce qui veut dire faux)

X **OR** Y a la valeur { 1 si Y est non nul
X si Y est nul

et

NOT X a la valeur { Ø si X est non nul
1 si X est nul

Relisez ce chapitre en tenant maintenant compte des informations ci-dessus afin d'être bien sûr que tout se passe bien.

Dans les expressions X **AND** Y, X **OR** Y et **NOT** X, X et Y prennent habituellement la valeur Ø ou 1, ce qui veut dire vrai ou faux. Mettez au point dix combinaisons différentes et vérifiez qu'elles donnent bien le résultat que vous attendez des fonctions respectives des opérateurs **AND**, **OR** et **NOT**.

4. Essayez le programme suivant :

```
1Ø INPUT A
2Ø INPUT B
3Ø PRINT (A AND A >=B)+(B AND A<B)
4Ø GOTO 1Ø
```

A chaque exécution, le programme écrit le plus grand des deux nombres — A et B — Pourquoi ?

Essayez de vous convaincre que vous pouvez admettre que

X **AND** Y

signifie

'X si Y (autrement, le résultat est Ø) '

et que

X **OR** Y

signifie

'X ou bien Y (et dans ce cas le résultat est 1)'

Une expression qui contient **AND** et **OR** est appelée expression conditionnelle. Exemple d'utilisation de **OR** :

LET PRIX DE DETAIL=PRIX MOINS TVA*(1.15 **OR** V$ ="PRIX HORS TAXE")

Noter la relation entre **AND** et l'addition (parce que sa valeur par défaut est Ø) et la relation entre **OR** et la multiplication (parce que sa valeur par défaut est 1).

5. Vous pouvez également écrire des expressions conditionnelles à valeur de chaîne, mais seulement avec **AND**.

X$ **AND** Y$ a la valeur { X$ si Y est non nul ; si Y est nul

l'expression veut donc dire 'X$ si Y (autrement la chaîne vide)'.

Essayez ensuite le programme ci-dessous dont le rôle est d'entrer deux chaînes et de les mettre en ordre alphabétique.

```
10 INPUT A$
20 INPUT B$
30 IF A$<=B$ THEN GOTO 70
40 LET C$=A$
50 LET A$=B$
60 LET B$=C$
70 PRINT A$;" ";("<" AND A$<B$)+("=" AND A$=B$);" ";B$
80 GOTO 10
```

6. Essayez le programme suivant :

```
10 PRINT "X"
20 STOP
30 PRINT "Y"
```

A l'exécution, ce programme met "X" sur l'écran en lançant le compte-rendu 9/20. Entrez maintenant

CONT

Vous pensez peut-être que le programme va se comporter comme si vous aviez écrit '**GOTO** 20', donc, que la machine va s'arrêter sans mettre "Y" sur l'écran ; ceci ne serait pas très utile et, pour cette raison, la machine est conçue pour que à la suite d'un compte rendu avec le code 9 (l'instruction **STOP** exécutée) le numéro de la ligne soit augmenté de 1 pour une instruction **CONT**. Ainsi, dans notre exemple, '**CONT**' a le même rôle que '**GOTO** 21' (qui, puisqu'il n'y a aucune ligne entre 20 et 30, se comporte comme '**GOTO** 30').

7. Plusieurs versions du BASIC (pas celle du ZX81) contiennent une instruction ON de la forme

ON expression numérique, GOTO, numéro de ligne, numéro de ligne..., numéro de ligne

Dans cette instruction, l'expression numérique est évaluée ; si la valeur de l'expression est 'n', l'instruction a l'effet suivant

GOTO le numéro de la nième ligne

Par exemple,

POUR A GOTO 100, 200, 300, 400, 500

Dans ce cas, si A a la valeur 2, 'GOTO 200' est exécutée. Dans le langage BASIC ZX81, ceci peut être remplacé par

GOTO 100*A

Essayez d'imaginer comment vous pourriez utiliser l'instruction suivante lorsque les numéros des lignes ne sont pas au niveau de la centaine

GOTO une expression conditionnelle.

ZX81

Chapitre 11

Jeu de caractères

Les lettres, chiffres, signes de ponctuation et autres qui peuvent apparaître dans les chaînes sont appelés caractères, ils constituent l'alphabet ou jeu de caractères utilisé par le ZX81. La plupart de ces caractères sont des symboles simples mais certains d'entre eux appelés jetons représentent des mots complets, par exemple **PRINT**, **STOP**, ******, etc.

En tout, il existe 256 caractères ; chaque caractère a un code compris entre 0 et 255. La liste complète des caractères et des codes correspondants est à l'annexe A. Deux fonctions – **CODE** et **CHR$** – permettent de passer des codes aux caractères et des caractères aux codes.

CODE s'utilise sur une chaîne et donne le code du premier caractère de la chaîne (ou Ø si cette dernière est vide).

CHR$ s'applique à un nombre et donne la chaîne à un seul caractère dont le code est ce nombre.

Le programme suivant imprime tout le jeu de caractères.

```
1Ø LET A=Ø
2Ø PRINT CHR$ A;
3Ø LET A=A+1
4Ø IF A<256 THEN GOTO 2Ø
```

Le début de la liste comporte les symboles " , £, $ et ainsi de suite jusqu'à Z qui sont tous sur le clavier et peuvent être entrés avec le curseur L. Plus loin, vous pouvez voir les mêmes caractères en vidéo inversée (caractères noirs sur fond blanc) ; ces caractères peuvent également être obtenus à partir du clavier. Si vous appuyez sur la touche **GRAPHICS** (9 en position majuscules), le curseur devient G ; il signifie mode graphique. Si vous appuyez sur la touche d'un symbole, ce dernier apparaît en vidéo inversée et cette présentation se poursuit jusqu'au moment où vous appuyez à nouveau sur **GRAPHICS** ou simplement sur **NEWLINE**. **RUBOUT** a sa signification habituelle. Veillez à ne pas perdre le curseur G parmi tous les caractères en vidéo inversée que vous venez d'entrer.

Lorsque vous aurez fait quelques expériences, vous aurez quand même le jeu de caractères en haut ; s'il n'en est pas ainsi, exécutez à nouveau le programme. Au début, vous trouverez des espaces et dix formes composées de marques noires, blanches et grises. Vous en trouverez onze autres un peu plus loin. Ces combinaisons sont appelées symboles graphiques ; les symboles graphiques sont utilisés pour faire des dessins et des schémas. Vous pouvez les entrer par le clavier en utilisant le mode graphique (à l'exception de l'espace qui est un symbole normal avec le curseur L ; le carré noir est l'espace en vidéo inversée). Vous devez utiliser les 2Ø touches sur lesquelles sont écrits les symboles graphiques. Par exemple, si vous voulez le symbole ▗ qui est sur la touche T, il faut appuyer sur **GRAPHICS** pour obtenir le curseur G et appuyer ensuite sur la touche T (en position majuscules). La description qui précède du mode **GRAPHICS** doit vous donner à penser que vous allez avoir un symbole en vidéo inversée ; toutefois, T en position majuscules est habituellement <>, c'est-à-dire un jeton, et les jetons n'ont pas d'inverse : pour cette raison, vous obtenez donc le symbole graphique ▗ .

Voici les 22 symboles graphiques.

Symbole	*Code*	*Pour l'obtenir*	*Symbole*	*Code*	*Pour l'obtenir*
	0	K or L **SPACE**		128	G **SPACE**
	1	G 1 en position majuscules		129	G Q en position majuscules
	2	G 2 en position majuscules		130	G W en position majuscules
	3	G 7 en position majuscules		131	G 6 en position majuscules
	4	G 4 en position majuscules		132	G R en position majuscules
	5	G 5 en position majuscules		133	G 8 en position majuscules
	6	G T en position majuscules		134	G Y en position majuscules
	7	G E en position majuscules		135	G 3 en position majuscules
	8	G A en position majuscules		136	G H en position majuscules
	9	G D en position majuscules		137	G G en position majuscules
	10	G S en position majuscules		138	G F en position majuscules

Reportez-vous à nouveau au jeu de caractères. Les jetons constituent un groupe composé de deux blocs faciles à distinguer : un petit groupe de trois (**RND**, **INKEY$** et **PI**) après Z, et un groupe plus important (commençant par l'image du guillemet complet après ▣ et se poursuivant de **AT** jusqu'à **COPY**).

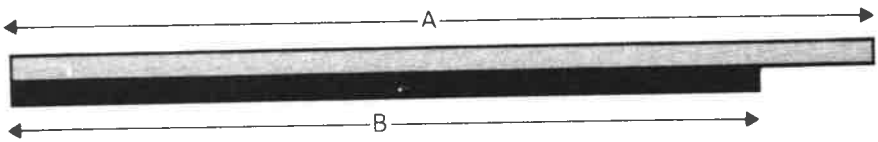
Les autres caractères semblent être des "?". C'est bien sous cette forme qu'ils sont écrits, le vrai point d'interrogation étant entre : et (. Parmi les autres, certains correspondent à des caractères de contrôle comme ◊ , **EDIT** et **NEWLINE**, tandis que les autres sont réservés à des caractères qui n'ont pas de signification spéciale pour le ZX81.

Résumé

Fonctions : **CODE, CHR$**

Exercices

1. Imaginez que l'espace correspondant au symbole d'une touche soit divisé en quatre carrés : ▦ . Si chaque carré peut être blanc ou noir, il y a 2*2*2*2 = 16 possibilités. Trouvez-les dans le jeu de caractères.
2. Imaginez que l'espace d'un symbole soit divisé en deux parties horizontales : ▤ . Si chaque moitié peut être de couleur blanche, noire ou grise, il y a donc 3*3 = 9 possibilités. Trouvez ces possibilités.
3. Les caractères de l'exercice 2 sont conçus pour être utilisés dans les schémas à barres horizontales composés de deux couleurs, gris et noir. Ecrivez un programme réalisant l'entrée de deux nombres A et B (entre Ø et 32) et dessinez un diagramme à barres pour les deux nombres.



Vous devez commencer par écrire "▩", et changer ensuite pour avoir "▨", ou "▬", selon que A est m ou Q ou inférieur à B.

Que fait votre programme si A et B ne sont pas des nombres entiers ? Ou bien s'ils ne sont pas dans l'intervalle 0-32 ? Un bon programme — sympathique pour l'utilisateur est l'expression à la mode qui sert à le désigner — doit faire quelque chose de sensé et d'utile.

4. Le clavier comporte deux caractères entièrement gris sur A et H. Si vous les regardez de très près, vous verrez que celui qui est sur H ressemble à un échiquier miniature tandis que celui qui est sur A ressemble à un échiquier en diagonale. Et écrivez-les l'un à côté de l'autre et vous verrez qu'ils ne se connectent pas correctement. Celui de A se connecte correctement avec ▨ et ▩ (sur S et D), tandis que celui de H se connecte très bien avec ▩ et ■ (sur F et G).

5. Exécutez ce programme :

```
1Ø INPUT A
2Ø PRINT CHR$ A;
3Ø GOTO 1Ø
```

Si vous procédez à des expérimentations avec ce programme, vous vous rendrez compte que pour CHR$, A est arrondi à l'entier le plus voisin ; et, si A n'est pas dans l'intervalle 0 - 255, le programme s'arrête et lance le compte-rendu B.

6. En employant les codes correspondant aux caractères, nous pouvons élargir le concept du "classement alphabétique", pour qu'il s'applique aux chaînes contenant n'importe quels caractères, et pas seulement des lettres. Si, au lieu de nous situer seulement au niveau de l'alphabet à 26 lettres, nous utilisons l'alphabet à 256 caractères dans le même ordre que leurs codes, le principe est exactement le même. Par exemple, les chaînes suivantes sont dans l'ordre alphabétique du ZX81.

```
"  ZACHARY"
"■"
"(ASIDE)"
"123 TAXI SERVICE"
"AASVOGEL"
"AA RDVARK"
"ZACHARY"
"AA RDVARK"
```

Voici la règle. La comparaison porte d'abord sur le premier caractère des deux chaînes. Si ces caractères sont différents, le code de l'un est inférieur à celui de l'autre et la chaîne dont il est le premier caractère est la première des deux. Si les deux caractères sont les mêmes, une comparaison est effectuée sur les caractères suivants. Si, pendant cette succession de comparaisons, l'une des chaînes est épuisée avant l'autre, elle est considérée comme précédente. Autrement, elles doivent être égales.

Entrez à nouveau le programme de l'exercice 4 du chapitre 10 (celui qui entre deux chaînes et les écrit en ordre) et utilisez-le pour faire des expériences.

7. Le programme suivant remplit l'écran de caractères graphiques noirs et blancs présentés de façon aléatoire :

```
1Ø LET A=INT (16*RND)
2Ø IF A>=8 THEN LET A=A+12Ø
3Ø PRINT CHR$ A;
4Ø GOTO 1Ø
```

(Comment ce programme fonctionne-t-il ?).

ZX81

Chapitre 12

Boucles

Supposons maintenant que vous voulez entrer cinq nombres et les additionner. L'une des façons d'opérer (mais ne le faites pas à moins de vous considérer puni) consiste à écrire

```
10 LET TOTAL=0
20 INPUT A
30 LET TOTAL=TOTAL+A
40 INPUT A
50 LET TOTAL=TOTAL+A
60 INPUT A
70 LET TOTAL=TOTAL+A
80 INPUT A
90 LET TOTAL=TOTAL+A
100 INPUT A
110 LET TOTAL=TOTAL+A
120 PRINT TOTAL
```

Cette méthode ne constitue pas une bonne pratique en programmation. Elle est encore à peu près acceptable pour additionner cinq nombres, mais vous pouvez facilement imaginer le caractère fastidieux d'un programme similaire conçu pour additionner 10 nombres et, s'il s'agit d'une centaine ou plus, la tâche devient quasiment impossible.

Il est beaucoup plus agréable de caler une variable pour qu'elle compte jusqu'à cinq et d'arrêter ensuite le programme, comme suit (ce programme est celui que vous devriez entrer) :

```
10 LET TOTAL = 0
20 LET COMPTE = 1
30 INPUT A
40 REM COMPTE = NOMBRE D'ENTREES EFFECTIVES DE A
50 LET TOTAL = TOTAL + A
60 LET COMPTE + COMPTE + 1
70 IF COMPTE <= 5 THEN GOTO 30
80 PRINT TOTAL
```

Il faut tout de suite remarquer qu'il est extrêmement facile de modifier la ligne 70 pour que ce programme puisse additionner dix nombres ou même cent.

Ce type de calcul est tellement utile que deux instructions spéciales ont été conçues pour le faciliter : les instructions **FOR** et **NEXT**. Ces deux instructions sont toujours utilisées ensemble. Avec ces deux instructions le programme que vous avez entré donne exactement le même résultat que :

```
1Ø LET TOTAL = Ø
2Ø FOR C=1 TO 5
3Ø INPUT A
4Ø REM C = NOMBRE D'ENTREES EFFECTIVES DE A
5Ø LET TOTAL = TOTAL + A
6Ø NEXT C
8Ø PRINT TOTAL
```

(Pour obtenir ce programme à partir du précédent, il vous suffit d'éditer les lignes 2Ø, 4Ø, 6Ø et 7Ø. **TO** est 4 en position majuscules.)

Noter que COMPTE est devenu C. La variable de comptage — ou variable de contrôle — d'une boucle **FOR—NEXT** doit être désignée par un nom composé d'une seule lettre.

Ce programme a pour effet que C prend successivement les valeurs 1 (valeur initiale), 2, 3, 4 et 5 (la limite) et que pour chacune, les lignes 3Ø, 4Ø et 5Ø sont exécutées. Ensuite, lorsque C est parvenu à la fin de ses cinq valeurs, la ligne 8Ø est exécutée.

Ce qui précède comporte un autre avantage subtil : le fait que la variable de contrôle ne doit pas augmenter de 1 à chaque pas, la valeur "1" peut être transformée en une autre valeur en utilisant une partie **STEP** dans l'instruction **FOR**. La forme la plus générale de l'instruction **FOR** est la suivante :

FOR variable de contrôle = valeur initiale **TO** limite **STEP** pas

dans laquelle la variable de contrôle est une seule lettre et où la valeur initiale, la limite et le pas sont des expressions numériques. Ainsi, si vous remplacez la ligne 2Ø du programme par la ligne suivante

```
2Ø FOR C=1 TO 5 STEP 3/2
```

C prend les valeurs 1, 2,5 et 4. Notez que vous n'êtes pas obligé de vous limiter à des nombres entiers et que la valeur de contrôle ne doit pas coïncider exactement avec la limite — elle continue sa boucle tant qu'elle est inférieure ou égale à la limite (exercice 4).

Vous devez apporter une attention particulière aux opérations lorsque vous exécutez deux boucles **FOR—NEXT**, l'une dans l'autre. Essayez le programme suivant qui imprime une série complète de dominos à 6 points.

```
1Ø FOR M=Ø TO 6
2Ø FOR N=Ø TO M
3Ø PRINT M;":";N;" ";
4Ø NEXT N
5Ø PRINT
6Ø NEXT M
```

boucle N (lignes 2Ø à 4Ø)

boucle M (lignes 1Ø à 6Ø)

Vous pouvez voir que la boucle N est entièrement contenue dans la boucle M — l'emboîtement des boucles est correct. En revanche, il faut éviter que deux boucles **FOR—NEXT** se recouvrent c'est-à-dire que l'une d'elles ne soit pas entièrement contenue dans

l'autre, comme suit :

```
MAUVAIS  1Ø FOR M=Ø TO 6
         2Ø FOR N=Ø TO M          } boucle M
         3Ø PRINT M;":";N;" ";    }
         4Ø NEXT M                }
         5Ø PRINT                 } boucle N
         6Ø NEXT N                }
```

Deux boucles **FOR–NEXT** doivent être soit l'une à l'intérieur de l'autre, soit complètement séparées. Il faut également éviter de sauter au milieu d'une boucle **FOR–NEXT** à partir d'un point extérieur. La variable de contrôle prend seulement une valeur correcte lorsque son instruction **FOR** est exécutée et, si vous faites une bourde à ce niveau, la prochaine instruction **NEXT** va jeter la machine dans la plus grande confusion. Vous recevrez sans doute le compte-rendu d'erreur 1 ou 2 (pour vous dire qu'une instruction **NEXT** ne contient pas une variable de contrôle reconnue) si vous avez de la chance.

Résumé

Instructions : **FOR, NEXT, TO, STEP**

Exercices

1. Récrivez le programme du chapitre 11 (celui qui imprime le jeu de caractères) en utilisant une boucle **FOR–NEXT** (Réponse au chapitre 13).

2. Par opposition à une variable normale, une variable de contrôle n'a pas seulement un nom à une valeur, mais également une limite, un pas et un numéro de ligne pour revenir en arrière (la ligne suivant l'instruction **FOR** où elle a été établie). Vous devez toujours être convaincu du fait qu'à l'exécution de l'instruction **FOR**, tous ces renseignements sont bien disponibles (sans oublier d'utiliser la valeur initiale comme première valeur prise par la variable et, ensuite que (en utilisant à titre d'exemples, le deuxième programme et le troisième) ces informations sont suffisantes pour la conversion de la ligne

```
NEXT C
```

en deux lignes

```
LET C=C+1

IF C<=5 THEN GOTO 3Ø
```

(En réalité, nous avons un petit peu triché, car il faudrait ici écrire **GOTO** 21 au lieu de **GOTO** 3Ø. Cette écriture différente aurait exactement le même résultat sur notre programme.

3. Exécutez le troisième programme et entrez ensuite :

PRINT C

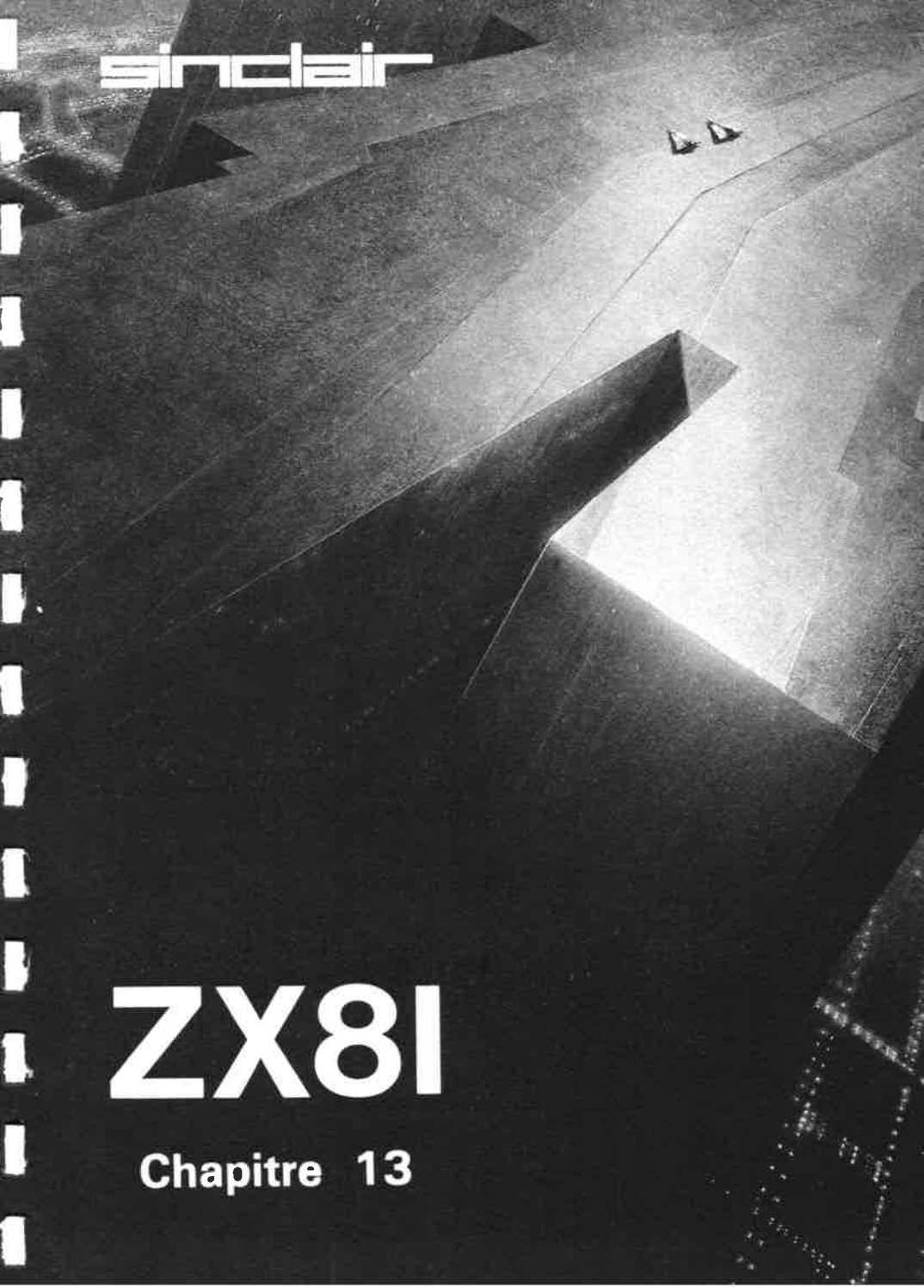
Pourquoi le résultat est-il 6 et non 5 ?
(Réponse : l'instruction **NEXT** de la ligne 60 est exécutée 5 fois et la valeur "1" est ajoutée à C à chaque exécution).
Que se passe-t-il si vous écrivez **STEP** 2 à la ligne 20 ?

4. Modifiez le troisième programme de façon à obtenir qu'au lieu d'additionner automatiquement 5 nombres, il vous demande d'entrer le nombre de nombres que vous voulez additionner. A l'exécution de ce programme, que se passe-t-il si vous entrez Ø, ce qui veut dire que vous ne voulez additionner aucun nombre ? Pourquoi pensez-vous que ceci peut provoquer des problèmes pour la machine alors que vous précisez exactement ce que vous voulez ? (L'ordinateur doit rechercher l'instruction **NEXT** C qui, habituellement, n'est pas nécessaire). En réalité, tout ceci a été prévu.

5. Essayez le programme suivant pour imprimer les nombres 1—1Ø en ordre décroissant.

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

Transformez ce programme en un programme sans boucle **FOR—NEXT** de la même façon que vous convertiriez le programme 3 pour en faire le programme 2 (voir exercice 2). En quoi le pas négatif rend-il légèrement différent ?



ZX81

Chapitre 13

Normale et rapide

Le ZX81 peut travailler à deux vitesses – normale et rapide. A la mise sous tension, l'ordinateur travaille en mode NORMAL, il peut simultanément calculer et mettre les informations sur l'écran. Ce mode convient parfaitement pour les présentations évolutives.

Toutefois, l'ordinateur peut aller quatre fois plus vite ; mais, pour ce faire, il ne se préoccupe plus de la représentation sur l'écran, sauf lorsqu'il n'a rien d'autre à faire. Pour utiliser la vitesse rapide, entrez

FAST

Maintenant, lorsque vous appuyez sur une touche, l'écran clignote – ceci résulte du fait que l'ordinateur arrête d'afficher une image pendant qu'il se concentre uniquement sur la touche que vous avez manœuvrée.

Entrez un programme, par exemple :

```
1Ø FOR N=Ø TO 255
2Ø PRINT CHR$ N;
3Ø NEXT N
```

A l'exécution, tout l'écran prend une couleur grisâtre jusqu'à la fin du programme où le résultat de l'instruction **PRINT** est affiché sur l'écran.

L'image est également présente pendant une instruction **INPUT**, pendant que l'ordinateur attend que vous entriez (**INPUT**) les données. Essayez le programme suivant :

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

Pour revenir en mode normal (calcul et affichage simultanés), entrez

SLOW

Le choix du mode de travail – calcul et affichage pour bien voir ce que vous faites, ou calcul seulement (mode rapide) pour gagner du temps – est quelque chose de personnel, mais, de façon générale, vous utiliserez le mode rapide lorsque

(i) votre programme contient beaucoup de calculs numériques, en particulier s'il n'en écrit pas beaucoup, bien qu'étant en mode calcul et affichage, le temps ne semble pas très long si vous pouvez voir des résultats présentés sur l'écran assez fréquemment, ou

(ii) vous entrez un programme long. Vous avez déjà remarqué que la liste est refaite chaque fois que vous entrez une nouvelle ligne de programme, à la longue, cette opération peut être fastidieuse.

Vous pouvez utiliser des instructions **SLOW** et **FAST** dans des programmes, sans aucun problème.

Par exemple :

```
1Ø SLOW
2Ø FOR N=1 TO 64
3Ø PRINT "A";
4Ø IF N = 32 THEN FAST
5Ø NEXT N
6Ø GOTO 1Ø
```

Résumé

Instructions : **FAST, SLOW**

ZX81

Chapitre 14

18X2

Chapitre 14

Sous-programmes

Il arrive que différentes parties de votre programme soient chargées d'exécuter des travaux assez similaires et, de ce fait, vous êtes amené à entrer les mêmes lignes deux fois ou davantage ; cette entrée répétitive est inutile. Vous pouvez entrer les lignes une seule fois, sous forme d'un sous-programme et, ensuite, les utiliser ou les appeler à n'importe quel autre endroit du programme sans avoir à refaire leur entrée.

Pour ce faire, vous devez utiliser les instructions **GOSUB** et **RETURN**.

Dans **GOSUB** n

n est le numéro de la première ligne du sous-programme ; cette instruction est semblable à **GOTO** n mais dans le cas de **GOSUB** n, la machine se souvient du numéro de ligne de l'instruction GOSUB afin de pouvoir y revenir après avoir exécuté le sous-programme. Pour se souvenir de ce numéro, l'ordinateur place le numéro de la ligne (l'adresse de retour), en haut d'une pile d'adresses (la pile de GOSUB).

RETURN

prélève le premier numéro de ligne de la pile **GOSUB** et passe à la ligne désignée.

Premier exemple

```
10 PRINT "CECI EST LE PROGRAMME PRINCIPAL"
20 GOSUB 1000
30 PRINT "ENCORE"
40 GOSUB 1000
50 PRINT "C'EST TOUT"
60 STOP
1000 REM "LE SOUS-PROGRAMME COMMENCE ICI"
1010 PRINT "CECI EST LE SOUS-PROGRAMME"
1020 RETURN
```

L'instruction **STOP** de la ligne 60 est très importante parce que, si elle était omise, le programme passerait directement au sous-programme et provoquerait une erreur 7 à la détection de l'instruction **RETURN**.

Prenons un exemple moins trivial et supposons que vous voulez écrire un programme qui doit manipuler des livres sterling, des shilling et des pence. (Le système utilisé en Grande-Bretagne avant 1971 ; une livre était divisée en 20 shillings (1 shilling était donc égal à 5 pence) et un shilling était subdivisé en 12 anciens pence. En Angleterre, l'ancien penny était désigné par l'abréviation "d".) Les trois variables sont L, S et D (et peut-être d'autres comme L1, S1, D1, etc.) et les calculs sont très faciles. Ils sont d'abord exécutés séparément sur les livres, les shilling et les pence. Par exemple, pour ajouter deux sommes, vous ajoutez les pence, ensuite les shilling et enfin les livre. Pour doubler un montant quelconque, vous doublez le nombre de pence, le nombre de shilling et le nombre de livre, etc. Lorsque toutes ces opérations ont été exécutées, il faut les ajuster, c'est-à-dire les mettre en forme correcte, afin que le nombre de pence soit compris entre 0 et 11 et le nombre de shilling entre 0 et 19. La dernière phase étant commune à toutes les opérations, elle peut revêtir la forme d'un sous-programme.

Pour le moment laissons de côté la notion de sous-programme ; essayez d'écrire le programme. Soit les nombres arbitraires L, S et D, comment allez-vous les convertir en livres, shillings et pence ? Une partie du problème est que vous allez réfléchir à des cas de plus en plus bizarres.

Le premier exemple qui va sans doute vous venir à l'esprit est du genre £1..25s..17d, que vous devez convertir pour obtenir £2..6s..5d. Pas très difficile. Mais que se passe-t-il dans le cas de nombres négatifs ? Une dette de £1..25s..17d soit £−1..−25s..−17d. peut très bien devenir £−3..13s..7d, ce qui est une façon plutôt curieuse de le dire (comme si les gens se prêtaient seulement des sommes en livres). Et les fractions ? Si vous divisez £1..25s..17d par 2, vous obtenez £0.5..12.5s..8.5.d. Bien que ce résultat donne 8,5 pour les pence, c'est-à-dire entre 0 et 11 et 12,5 pour les shillings c'est-à-dire entre 0 et 19, il n'est pas aussi bon que l'indication £1..3s..2.5d. Essayez maintenant de trouver vos réponses à tout ce qui précède et utilisez-les dans un programme avant de continuer votre lecture.

Voici une solution.

```
1000 REM SOUS-PROGRAMME D'AJUSTEMENT DES L,S,D EN
     FORME NORMALE POUR LES LIVRES, SHILLINGS ET PENCE
1010 LET D=240*L+12*S+D
1020 REM MAINTENANT TOUT LE MONTANT EST EN PENCE
1030 LET E=SGN D
1040 LET D=ABS D
1050 REM NOUS TRAVAILLONS SUR D POSITIF, EN CONSERVANT
     SON SIGNE DANS E
1060 LET S=INT (D/12)
1070 LET D=(D−12*S)*E
1080 LET L=INT (S/20)*E
1090 LET S=S*E−20*L
1100 RETURN
```

Isolément, ceci ne sert pas à grand chose puisqu'il n'y a pas de programme pour établir L, S et D à l'avance et qu'il n'y a pas non plus quelque chose pour les utiliser ensuite. Entrez le programme principal suivant et un autre sous-programme pour écrire L, S et D.

```
10 INPUT L
20 INPUT S
30 INPUT D
40 GOSUB 2000
45 REM ECRIRE LES VALEURS
50 PRINT
60 PRINT "□□□ =";
70 GOSUB 1000
75 REM AJUSTEMENT
80 GOSUB 2000
85 REM ECRIRE LES VALEURS
90 PRINT
100 GOTO 10

2000 REM SOUS PROGRAMME D'ECRITURE DE L,S ET D
2010 PRINT "£";L;"..";S;"S..";D;"D";
2020 RETURN
```

(Rappelez-vous — chapitre 9 — que l'instruction **PRINT** vide de la ligne 50 imprime une ligne vide).

Il est bien évident que le sous-programme d'écriture de la ligne 2000 a permis de diminuer la quantité de programmation ; ceci constitue un emploi très commun des sous-programmes — le raccourcissement des programmes principaux. Toutefois, en réalité, le sous-programme d'ajustement augmente la longueur du programme — incidence d'une **GOSUB** et d'une **RETURN** ; ainsi, il apparaît que la longueur des programmes n'est pas le seul élément critique. Utilisés avec intelligence, les sous-programmes peuvent faciliter la compréhension des programmes pour le participant le plus important — l'être humain.

Grâce aux instructions plus puissantes qu'il contient, le programme principal est simplifié : chaque **GOSUB** représente du BASIC compliqué, mais vous pouvez l'oublier, seul le résultat compte. Ainsi, il devient beaucoup plus facile de comprendre la structure générale du programme.

Par ailleurs, les sous-programmes sont simplifiés pour une raison tout-à-fait différente, ils sont plus courts. Ils utilisent les instructions classiques **LET** et **PRINT**, mais puisqu'elles ne font qu'une partie du travail, elles sont plus faciles à écrire.

L'astuce se situe dans le choix du ou des niveaux auxquels doivent être écrits les sous-programmes. Ils doivent être assez importants pour avoir une incidence notable sur le programme principal, mais demeurer assez petits pour être plus faciles à écrire qu'un programme complet, sans sous-programme. Les exemples suivants (non recommandés) illustrent ce point.

D'abord,

```
  10 GOSUB 1000
  20 GOTO 10
1000 INPUT L
1010 INPUT S
1020 INPUT D
1030 PRINT " ";L;"..";S;"S..";D;"D";TAB 8;"=";
1040 LET D=240*L+12*S+D
   :     :
   :     :
2000 RETURN
```

et ensuite

```
  10 GOSUB 1010
  20 GOSUB 1020
  30 GOSUB 1030
  40 GOSUB 1040
  50 GOSUB 1050
   :     :
   :     :
  30 GOTO 10
1010 INPUT L
1015 RETURN
1020 INPUT S
1025 RETURN
1030 INPUT D
1035 RETURN
1040 PRINT " ";L;"..";S;"S..";D;"D";TAB 8; "= ";
1045 RETURN
1050 LET D=240*L+12*S+D
1055 RETURN
   :     :
   :     :
```

Le premier programme et son seul sous-programme puissant et le second avec ses différents sous-programmes sans importance démontrent des points tout à fait opposés.

Un sous-programme peut aisément en appeler un autre ou même s'appeler lui-même (un sous-programme qui s'appelle lui-même est récurrent) ; par conséquent, vous ne devez pas avoir peur d'en utiliser plusieurs.

Résumé

Instructions : **GOSUB, RETURN**

Exercices

1. Le programme donné comme exemple est presque un calculateur universel des valeurs L, S et D. Comment l'utiliseriez-vous ?
   (i) Pour convertir des livres et des nouveaux pence en livres, shillings et pence ?
   (ii) Pour convertir des guinées (une guinée = 1 livre 1 s.. = 21 s) en livres et shillings ?
   (iii) Pour calculer des fractions de livres ? (par exemple un tiers de livre, soit 6s..8d.).
   Ecrivez également une ligne pour arrondir les pence au farthing (1/4 d) le plus proche.

2. Ajoutez deux instructions au programme :

   4 **LET** AJUSTMENT = 1ØØØ
   7 **LET** LSD ECRIRE = 2ØØØ

et transformez

GOSUB 1ØØØ en **GOSUB** AJUSTEMENT
GOSUB 2ØØØ en **GOSUB** LSD ECRIRE

Le résultat est exactement celui que vous attendiez ; en pratique, le numéro de la ligne d'une **GOSUB** (ou de **GOTO** ou de **RUN**) peut être une expression numérique quelconque. (Attention, ceci est vrai seulement sur le ZX81 et pas sur d'autres machines ; n'oubliez pas que le BASIC du ZX81 n'est pas le BASIC standard).

Ce genre d'opération peut jouer un rôle important au niveau de la lisibilité de vos programmes.

3. Récrivez le programme principal de l'exemple pour qu'il fasse quelque chose d'autre tout en utilisant encore les mêmes sous-programmes.

4. ... **GOSUB** n
   ... **RETURN**

écrites sur des lignes consécutives peuvent être remplacées par

... **GOTO** n

Pourquoi ?

5. Un sous-programme peut comporter plusieurs points d'accès. Par exemple, du fait des modalités d'emploi des instructions **GOSUB** par notre programme principal — **GOSUB** 1ØØØ suivie immédiatement par **GOSUB** 2ØØØ — nous pouvons remplacer nos deux sous-programmes par un sous-programme plus important dont le rôle est d'ajuster L, S et D et d'écrire ces valeurs. Ce sous-programme comporte deux points d'accès — un au début de tout le sous-programme et un autre, plus loin, pour la seule partie écriture.

Procédez aux modifications voulues.

6. Exécutez ce programme :

```
1Ø GOSUB 2Ø
2Ø GOSUB 1Ø
```

Les adresses de retour sont mises dans la pile de **GOSUB** mais elles ne sont jamais prélevées une nouvelle fois et, tôt ou tard, l'ordinateur ne peut pas en mémoriser d'autres. A cet instant, le programme s'arrête et l'ordinateur lance un compte-rendu d'erreur 4 (voir l'annexe B).

Vous pouvez avoir du mal à les extraire sans perdre toutes les données, mais cette opération est néanmoins possible en procédant comme suit :

(i) Supprimez les deux instructions **GOSUB**
(ii) Insérez les deux nouvelles lignes suivantes

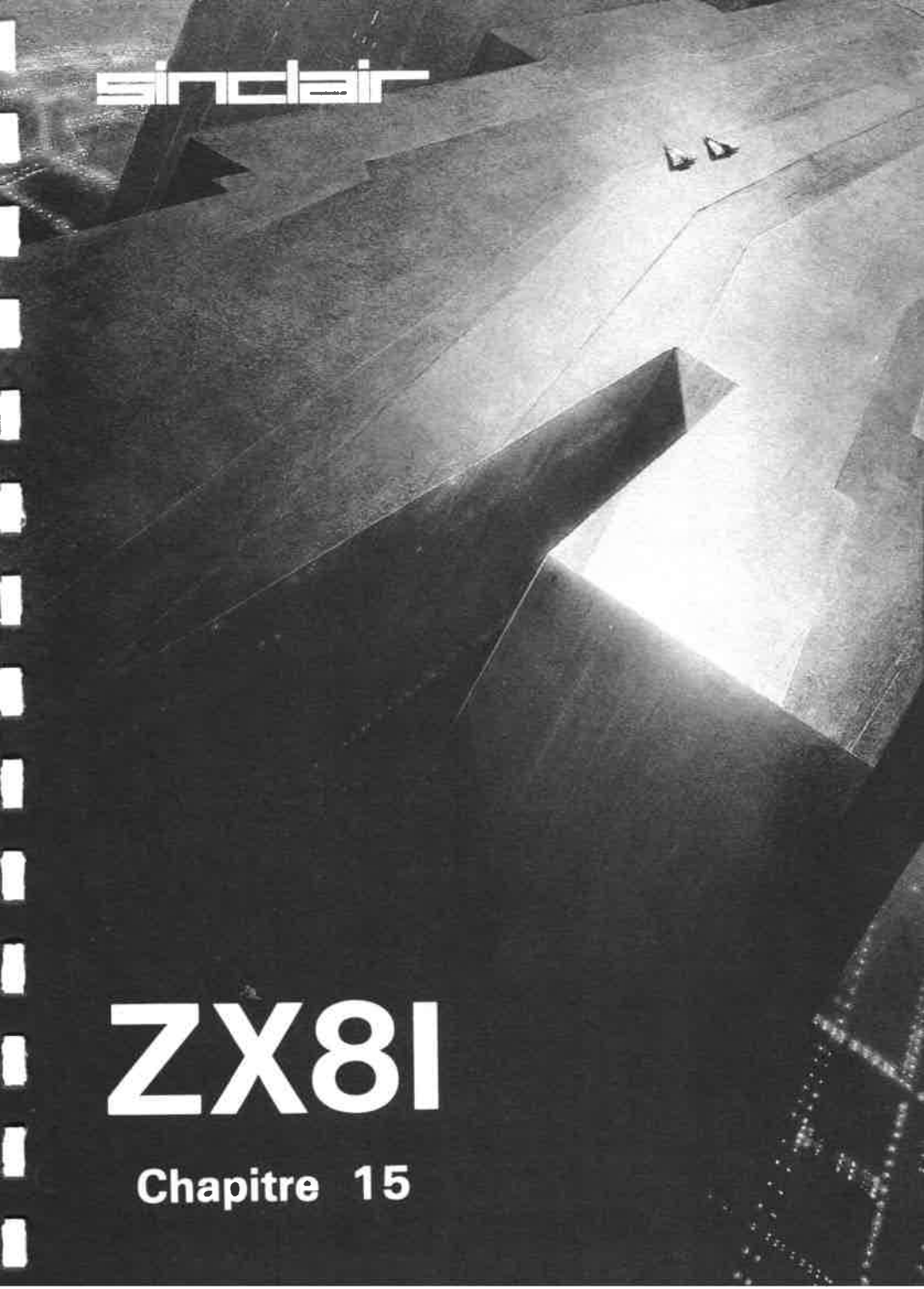
```
11 RETURN
21 RETURN
```

(iii) Entrez

RETURN

Les adresses de retour sont "éjectées" jusqu'au moment où la machine lance l'erreur 7.
(iv) Maintenant, modifiez votre programme pour éviter que cet incident ne se reproduise.

Expliquez en détail.



ZX81

Chapitre 15

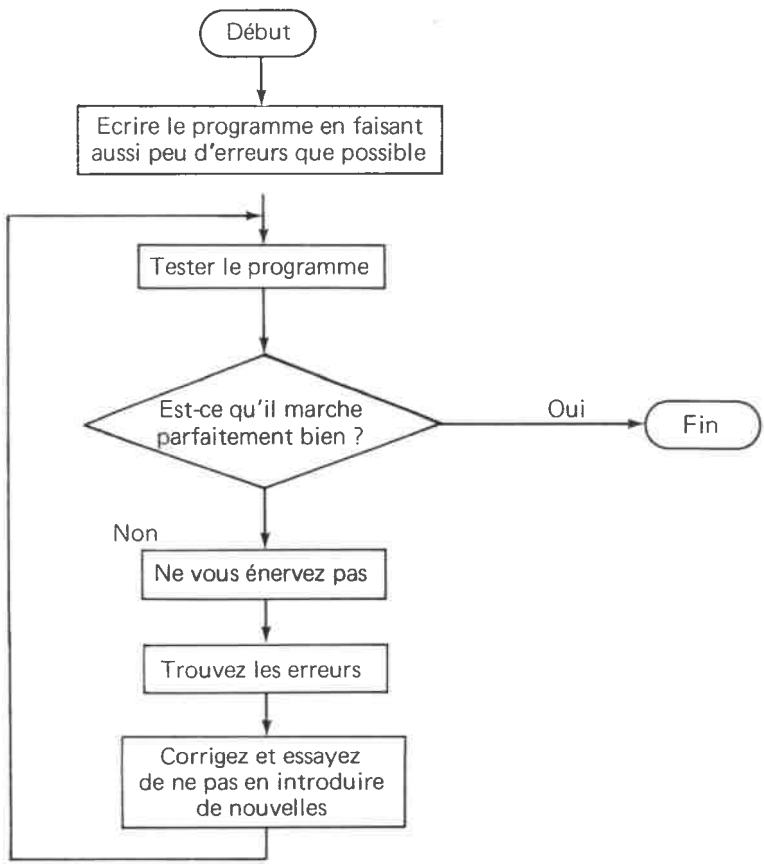
Chapter 15

Pour que vos programmes marchent

Pour savoir programmer les ordinateurs, il ne faut pas se contenter de connaître le rôle de chaque instruction. Vous vous êtes déjà probablement rendu compte du fait que la plupart de vos programmes comportaient des anomalies ou erreurs à leur première entrée : dans certains cas, il peut s'agir de simples fautes de frappe ou d'erreurs dans votre conception de ce que le programme doit faire. Vous pouvez imputer ces erreurs à votre inexpérience — il n'en est rien, vous vous tromperiez vous-même.

CHAQUE PROGRAMME COMMENCE PAR DES ERREURS.

De nombreux programmes se terminent également par des erreurs. Les deux corollaires correspondants sont : d'abord, vous devez tester tous vos programmes immédiatement et, ensuite, il est inutile de perdre votre sang-froid lorsque ceci se produit. Le cheminement général à suivre peut être illustré par un organigramme :



Pour utiliser l'organigramme ci-dessus, il suffit de suivre les flèches reliant les différentes "boîtes" entre elles et de faire ce qui est écrit dans chacune. Nous avons utilisé différents types de dessins pour les divers types d'instructions :

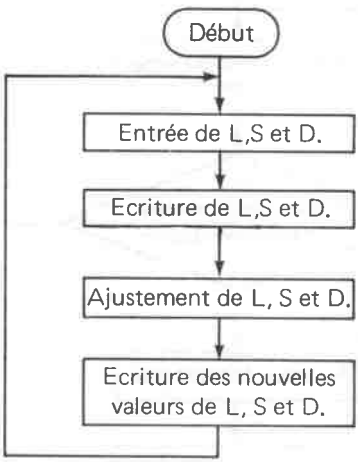
Cette représentation  indique le début ou la fin.

Ce cadre rectangulaire  désigne une instruction simple.

Un losange  vous demande de prendre une décision avant de continuer.

(Ces dessins sont largement utilisés, mais ils ne sont pas obligatoires.)

Bien sûr, les organigrammes ne conviennent pas bien pour décrire les activités de l'homme ; la réflexion en ligne droite n'est pas tout à fait ce qu'il faut pour la créativité ou la souplesse. Par contre, cette structure convient parfaitement pour les ordinateurs. Elle est la mieux adaptée à la description de la structure des programmes, les sous-programmes étant inclus dans les différents types d'encadrés ; par exemple, l'organigramme illustrant le programme de conversion des livres, des shillings et des pence du chapitre précédent pourrait être :



Ainsi, au niveau conceptuel, les cadres rectangulaires correspondent à des **GOSUB** ; en réalité, on constate que certains encadrés — comme celui réservé à l'entrée de L, S et D — sont traduits directement en instructions BASIC, sans sous-programme.

Toutes les aides — par exemple les organigrammes, les sous-programmes et les instructions **REM** — qui améliorent la lisibilité de programmes vous permettent de mieux les comprendre ; et, de ce fait, vous ferez moins d'erreurs. De plus, les sous-programmes

vous aident également à corriger les erreurs que vous avez commises car ils facilitent la vérification des programmes. Vous vous rendrez rapidement compte qu'il est plus facile de tester les sous-programmes individuellement et de vous assurer ensuite qu'ils fonctionnent bien ensemble que de faire le contrôle de tout un programme non-structuré.

Ainsi, les sous-programmes vous aident au stade de l'encadré "trouver les erreurs" et c'est bien là que vous avez le plus besoin d'aide, car ce stade est souvent le plus exaspérant. Pour faciliter votre recherche des erreurs :

(i) Assurez-vous qu'il n'y a pas de fautes de frappe. Ne jamais omettre cette recherche.

(ii) Essayez de savoir ce que les variables doivent devenir à chaque étape — si possible, expliquez ceci dans des instructions **REM**. Vous pouvez contrôler les variables à un point déterminé du programme en insérant une instruction **PRINT** à cet endroit.

(iii) Si le programme persiste à s'arrêter en lançant un compte-rendu d'erreur, vous devez utiliser cette information autant que faire se peut. Regardez le code correspondant au compte-rendu et déterminez ensuite pourquoi l'arrêt est intervenu à cette ligne. Si nécessaire, écrivez les valeurs des variables.

(iv) Vous pouvez également progresser ligne par ligne dans un programme en entrant ses lignes comme des commandes.

(v) Faites semblant d'être vous-même l'ordinateur : exécutez le programme sur une feuille de papier en notant les valeurs des variables.

Lorsque vous avez découvert les erreurs, leur correction ressemble beaucoup à l'écriture du programme original, mais n'oubliez quand même pas de procéder à nouveau au test de votre programme. Il est extrêmement facile de corriger une erreur... et d'en introduire une autre.

Exercices

1. L'organigramme du calculateur LSD ne comporte pas de cadre "fin". Est-ce important ? Si vous vouliez en mettre un, où le mettriez-vous ?
2. Dessinez les organigrammes des programmes de boucles du chapitre 12.

ZX81

Chapitre 16

Chapter 16

Stockage sur bande magnétique

Comme il a été dit au chapitre 1 et comme vous vous en êtes indubitablement aperçu par la suite, la mise hors tension du ZX81 provoque la perte de tous les programmes et variables que votre machine contenait. La seule façon de les préserver est de les enregistrer sur une bande magnétique (une cassette) par l'ordinateur ; ultérieurement, vous pouvez les remettre dans la machine qui se trouve remise à peu près dans le même état que celui qu'elle avait lorsqu'elle a fait l'enregistrement.

Le ZX81 est accompagné d'une paire de conducteurs (v du chapitre 1) dont le rôle est de connecter le ZX81 à un magnétophone à cassette. Vous devez utiliser votre magnétophone, sans oublier que certains fonctionnent mieux que d'autres.

Pour le ZX81, les petits enregistreurs portatifs et bon marché sont au moins aussi bons que les puissantes machines stéréo provoquent d'ailleurs moins de difficultés. Un compteur est un accessoire très utile.

Deuxièmement, votre magnétophone doit comporter une embase d'entrée pour le micro et une embase de sortie pour le casque ou l'écouteur (s'il n'y en a pas, essayez l'embase du haut-parleur extérieur). Ces prises doivent être de préférence prévues pour des jacks 3,5 mm (c'est-à-dire pour correspondre aux jacks des conducteurs fournis) parce que les autres types ne donnent pas un signal assez puissant pour le ZX81.

Toutes les cassettes doivent donner des résultats corrects, mais il est préférable d'utiliser des bandes à faible niveau de bruit.

Vous disposez maintenant d'un magnétophone correct, il suffit de le raccorder à l'ordinateur : l'un des fils doit relier l'embase d'entrée du microphone de l'enregistreur et l'embase portant l'indication "MIC" du ZX81 et l'autre relier l'embase de sortie de l'écouteur de l'enregistreur et l'embase "EAR" du ZX81. Même si vous croisez les conducteurs, vous ne réussirez pas à endommager votre ZX81).

Entrez un programme dans l'ordinateur, par exemple celui du jeu de caractères du chapitre 11. Pour sauvegarder un programme, il faut lui donner un nom qu'il est souhaitable d'insérer dans le programme afin qu'il soit présent dans les listes – le moyen le plus facile est d'utiliser une instruction REM.

Entrez :

```
5 REM "CARACTERES"
```

Maintenant, nous allons faire un essai "à blanc" pour voir ce qui se passe. Entrez :

```
SAVE "CARACTERES"
```

et observez l'écran de votre téléviseur. Pendant 5 secondes environ, il est de couleur grisâtre pour ensuite (pendant 6 secondes environ) présenter une structure évolutive de minces raies noires et blanches ; enfin, l'écran devient blanc et affiche le compte-rendu Ø/Ø. L'ordinateur était en train de transmettre un signal à l'embase "MIC" tout en lançant également le même signal au téléviseur, pour produire l'image que vous avez vue. La partie grise était une entrée en matière silencieuse, tandis que la partie noire et blanche représentait le programme.

Bien entendu, ce qui vous intéresse maintenant est de mettre le signal sur la bande magnétique — faisons cette opération correctement.

Pour sauvegarder un programme

1. Positionnez la bande en un endroit du ruban qui est vierge ou en un endroit que vous êtes prêt à surcharger.

2. Utilisez le microphone pour vous enregistrer en disant "caractères". Ceci n'est pas essentiel, mais cet enregistrement facilitera la recherche du programme ultérieurement. Rebranchez l'ordinateur et le magnétophone.

3. Entrez

SAVE "CARACTERES" (sans appuyer sur **NEWLINE**)

4. Mettez le magnétophone en position Enregistrement.

5. Appuyez sur **NEWLINE**.

6. Observez le téléviseur. Lorsqu'il a terminé son travail (compte-rendu Ø/Ø), arrêtez le magnétophone.

Pour être sûr que ces opérations se sont bien déroulées, vous devez maintenant écouter la bande magnétique au moyen du haut parleur incorporé du magnétophone (vous devrez probablement débrancher le conducteur de l'embase écouteur du magnétophone). Rebobinez la bande pour la remettre à l'endroit auquel vous avez commencé et lisez ensuite cette bande.

D'abord, vous entendrez votre propre voix disant "caractères". Ensuite, vous entendrez un ronronnement légèrement amorti. Ceci ne fait pas vraiment partie de l'enregistrement, il s'agit de la fin du signal destinée au téléviseur (avant d'avoir manœuvré **NEWLINE**) qui a été également transmis au magnétophone.

La partie suivante est composée de 5 secondes de silence, le début du signal destiné à l'enregistreur. Ce silence correspond à la période pendant laquelle l'écran était gris.

Ensuite, pendant 6 secondes environ, vous entendrez une série de tonalités aigues qui à plein volume, doivent être désagréablement ressenties. Ceci constitue l'enregistrement du programme et correspond à la structure de lignes noires et blanches sur le téléviseur.

Enfin, vous allez encore entendre le ronronnement amorti. Si vous n'entendez pas ces différentes bruits, vous devez d'abord vérifier que l'ordinateur et le magnétophone étaient bien connectés. Sur certains enregistreurs, les prises ne donnent pas un bon contact lorsqu'elles sont enfoncées à fond. Vous devez donc essayer dans ce cas de sortir votre prise d'un ou deux mm — il arrive que l'on puisse sentir que la nouvelle position est plus naturelle que l'ancienne.

Supposons maintenant que l'enregistrement semble bien concorder à la description ci-dessus ; vous allez essayer de remettre le programme dans l'ordinateur.

Pour charger un programme désigné par un nom

1. Rebobinez la bande pour la remettre à l'endroit auquel vous avez commencé.

2. Assurez-vous que l'embase "EAR" de la machine est bien connectée à l'embase du casque de l'enregistreur à cassette.

3. Mettez la commande de volume du magnétophone à environ 3/4 de sa puissance maximale ; si votre magnétophone est équipé de commandes de tonalité, réglez-les en mettant les aigües en position haute et les graves en position basse (pour avoir un son plus sifflant).

4. Entrez

LOAD "CARACTERES" (sans appuyer sur **NEWLINE**)

5. Mettez l'enregistreur en marche.

6. Appuyez sur **NEWLINE**.

Vous allez de nouveau pouvoir observer sur l'écran des images de l'enregistrement mais cette fois elles seront différentes, elles revêtiront la forme d'une structure noire et blanche. Les deux parties — le silence et le programme — seront moins faciles à distinguer, mais vous devez quand même être capable de constater que la partie programme comporte des lignes plus lartes et mieux définies (essayez aussi à un moment quelconque de faire l'exercice 1).

Après quinze secondes environ, le chargement doit être terminé, la machine doit être arrêtée avec le compte-rendu Ø/Ø sur l'écran. Autrement, il faut appuyer sur la touche **BREAK** (espace) pour débloquer la machine.

Dans la plupart des cas, le problème se situe au niveau de la commande de volume : le volume doit être :

(i) Assez fort pour que la partie programme soit bien reçue par l'ordinateur.

(ii) Pas trop fort, car un excès de puissance risque de déformer la partie programme. (Ceci est extrêmement rare.)

et (iii) Assez bas pour que la partie silence soit reconnue comme telle par l'ordinateur.

Le meilleure réglage consiste à agir sur la commande de volume pour la régler à un niveau assez fort mais sans rendre bruyante la partie silencieuse ; vous pouvez faire ce réglage en écoutant cet enregistrement par le haut-parleur. Si le silence est vraiment trop bruyant, il se peut que vous ayez d'autres problèmes :

Certains enregistreurs forment une boucle de feedback avec le ZX81. Ceci peut se produire seulement lorsque les conducteurs EAR et MIC sont connectés simultanément et, dans ce cas, le remède consiste à effectuer une sauvegarde (**SAVE**) après avoir déconnecté le fil "EAR".

Certains magnétophones peuvent enregistrer le ronflement du réseau électrique. Pour supprimer ce ronflement, il suffit de les alimenter par des piles électriques.

D'autres enregistreurs (en particulier les modèles anciens et usés) sont intrinsèquement bruyants. Ce problème peut être partiellement résolu en utilisant une bande de meilleure qualité, mais ceci ne devrait pas être nécessaire.

Essayez aussi de nettoyer la tête du magnétophone, il se peut qu'elle soit sale.

Enfin, il se peut également que la mise en place de la prise dans l'embase du casque provoque le même problème que celui que nous avons mentionné pour l'embase du micro.

Si vous avez mis un programme sur bande et que vous ne vous souvenez pas de son nom, vous pouvez quand même le charger. (Essayez avec le programme "caractères" que vous avez utilisé auparavant.)

Pour charger un programme non désigné par un nom

1. Rebobinez la cassette pour que la bande amorce silencieuse soit en face de la tête du magnétophone.

2. Contrôlez tout votre équipement et réglez les commandes comme vous l'avez déjà fait. Vous vous rendrez peut-être compte que vous devez faire plus attention au niveau du volume que lorsque vous connaissez le nom du programme.

3. Entrez

LOAD "" (sans appuyer sur **NEWLINE**)

4. Mettez le magnétophone en marche.

5. Appuyez sur la touche **NEWLINE**.

6. Les autres opérations sont les mêmes que précédemment.

Dans ce cas, le nom du programme dont vous voulez le chargement est la chaîne vide et l'ordinateur charge le premier programme qu'il trouve. Notez également que lorsque vous sauvegardez un programme, vous ne pouvez pas le désigner par une chaîne vide ; si vous tentez de le désigner par un nom composé d'une chaîne vide, l'ordinateur vous signale une erreur F.

Les instructions **LOAD** et **SAVE** peuvent aussi être utilisées dans des programmes. Avec **SAVE**, le programme se sauvegarde dans un état tel que, dès qu'il est rechargé, il reprend immédiatement l'exécution à la ligne suivant l'instruction **SAVE**.

Par exemple, entrez

```
 5 REM "INUTILE"
10 PRINT "C'EST TOUT CE QU'IL FAIT"
20 STOP
100 SAVE "INUTILE"
110 GOTO 10
```

Connectez le magnétophone et entrez

RUN 100 (sans appuyer sur la touche **NEWLINE**).

et mettez le magnétophone en position enregistrement. Appuyez sur **NEWLINE**. Lorsque

le programme s'est sauvegardé, il continue à fonctionner. Vous constaterez plus tard que le E final du mot inutile de la ligne 1ØØ a été transformé, il est maintenant présenté en vidéo inversée, mais vous n'avez pas à vous en inquiéter.

Pour procéder au chargement, rebobinez la bande pour la ramener avant le début du programme et entrez

LOAD "INUTILE" (sans **NEWLINE**),

et procédez maintenant à la transmission du contenu de la bande vers l'ordinateur et appuyez ensuite sur **NEWLINE**. Une fois le chargement terminé, la machine continue l'exécution à la ligne 11Ø, vous n'avez absolument pas à intervenir.

Notez que l'insertion de l'instruction **SAVE** à la fin du programme signifie que pour l'exécuter sans **SAVE**, il vous suffit d'entrer **RUN** — vous n'avez pas à contourner l'instruction **SAVE**.

Toutefois, vous ne devez pas lancer une sauvegarde (une instruction **SAVE**) de l'intérieur d'une routine **GOSUB** — ça ne marche pas correctement.

Ne mettez pas de caractères en vidéo inversée dans le nom d'un programme. La partie du nom qui se trouve après le caractère inversé serait perdue.

Le nom d'un programme ne doit pas être composé de plus de 124 caractères.

Le nom figurant dans **LOAD** ou dans **SAVE** ne doit pas être une constante chaîne, ce peut être n'importe quelle expression du type A$ ou CHR$ 1ØØ.

Résumé

Pour sauvegarder un programme sur bande magnétique
Pour charger un programme désigné à partir d'une bande
Chargement du premier programme disponible d'une bande
Sauvegarde d'un programme qui réalisera lui-même son chargement et son exécution.

Instructions : **SAVE, LOAD**

Nota

Vous ne pouvez pas charger des programmes qui ont été sauvegardés sur un autre type d'ordinateur ou sur le ZX80 avec son propre langage BASIC. Vos programmes sauvegardés ne peuvent pas être chargés dans d'autres types d'ordinateurs ou dans le ZX80 avec le BASIC qui lui est propre. Par contre le ZX80 utilisé avec le BASIC du ZX81 est bien évidemment compatible avec le ZX81 ; les programmes sauvegardés sur l'une des machines peuvent être mis dans l'autre. Si vous chargez un programme provenant d'un ZX80, votre ZX81 se trouve automatiquement en mode rapide.

Exercices

1. Réalisez une bande composée de plusieurs programmes courts, commencez la transmission vers l'ordinateur et entrez

LOAD "PAS LE NOM D'UN PROGRAMME"

Sur le téléviseur, vous devez voir aisément la différence entre les parties vides de la bande (structure relativement floue de blancs et noirs) et les programmes (lignes nettement plus définies). Les deux structures diffèrent de ce que vous pouvez observer pendant la sauvegarde. Si vous diminuez le volume pendant le passage d'un programme, vous verrez que l'image se présente sous forme de "vides" dès que le signal devient trop faible pour être interprété comme un programme.

2. Réalisez une bande sur laquelle, à son chargement, le premier programme imprime un menu (une liste de tous les autres programmes de la bande) qui vous demande de choisir un programme et le charge.

3. Entrez à nouveau le programme "CARACTERES" et entrez ensuite

LET X=7

pour que — bien que cela ne figure pas dans le programme — l'ordinateur contienne maintenant une variable X de valeur 7. Sauvegardez le programme, mettez l'ordinateur hors tension, puis sous tension (pour être sûr qu'il n'y a aucune possibilité de tricher) et rechargez le programme. Entrez

PRINT X

et vous obtenez la réponse 7. L'instruction **SAVE** n'a pas préservé seulement le programme, elle agit également sur toutes les variables — X y compris.

Si vous voulez sauvegarder les variables à l'exécution du programme, vous devez vous souvenir qu'il faut utiliser **GOTO** et non **RUN** (comme il a été dit au chapitre 9). Vous pouvez éviter d'avoir à vous souvenir de cette caractéristique en agissant de manière à obtenir que le programme s'exécute lui-même, c'est-à-dire en employant **SAVE** comme ligne du programme.

4. Entrez un programme très long et débranchez momentanément votre alimentation. Ce genre d'incident intervient parfois spontanément ; ce n'est plus une erreur, mais un incident. Vous ne pouvez faire qu'une chose — pleurer. Si cet incident se reproduit trop souvent, il y a probablement quelque chose d'anormal et il vaut mieux sauvegarder votre programme incomplet sur bande — par exemple quand vous en avez écrit la moitié.

ZX81

Chapitre 17

Affinons l'écriture

Vous vous rappelez que l'instruction **PRINT** comporte une liste de rubriques dont chacune est une expression (ou peut-être même rien) séparées par des virgules ou des points-virgules. Il existe deux autres types de rubriques utilisées dans **PRINT** pour dire à l'ordinateur non pas ce qu'il doit écrire, mais où le faire. Par exemple : **PRINT AT** 11,16;"*" provoque l'affichage d'une étoile au milieu de l'écran.

AT ligne, colonne

déplace la position d'écriture (**PRINT**) (l'endroit auquel la rubrique suivante va être écrite) pour la mettre à la ligne et à la colonne précisées. Les lignes sont numérotées de Ø (en haut) à 21 et les colonnes de Ø (à gauche) à 31.

TAB colonne

provoque le passage de la position d'écriture (**PRINT**) à la colonne indiquée. L'écriture se fait à la même ligne ou bien, si un espace arrière doit intervenir, provoque le passage à la ligne suivante. Notez que l'ordinateur réduit le numéro de la colonne modulo 32 (c'est-à-dire qu'il divise ce numéro par 32 et se sert du reste) ; par conséquent **TAB** 33 signifie la même chose que **TAB** 1.

Par exemple

PRINT TAB 3Ø;1;**TAB** 12;"CONTENU";**AT** 3,1;"CHAPITRE";**TAB** 24;"PAGE"

(Vous pourriez écrire de cette façon le titre d'une page "CONTENU" 1 étant le numéro de la page).

Quelques détails :

(i) Il est souhaitable de terminer les nouvelles rubriques par des points-virgules comme nous l'avons fait ci-dessus. Vous pouvez utiliser des virgules (ou même ne rien mettre à la fin des instructions), mais ceci vous oblige, une fois établie la position d'écriture, à la redéplacer immédiatement — ce qui n'est pas vraiment très utile.

(ii) Bien que **AT** et **TAB** ne soient pas des fonctions, vous devez appuyer sur la touche de fonction (NEWLINE en position majuscules) pour les obtenir.

(iii) Vous ne pouvez pas écrire sur les deux lignes du bas (22 et 23) de l'écran parce qu'elles sont réservées aux commandes, aux données d'entrée (**INPUT**), aux comptes-rendus, etc. L'expression "ligne du bas" veut donc habituellement dire ligne 21.

(iv) Vous pouvez utiliser **AT** pour modifier la position d'écriture, même lorsque quelque chose est déjà écrit sur l'écran ; l'indication écrite est surchargée.

Deux autres instructions sont liées à **PRINT** — **CLS** et **SCROLL**.

CLS efface l'écran (c'est tout).

SCROLL fait remonter tout le contenu de l'écran d'une ligne (la ligne du haut est perdue) et déplace la position d'écriture (**PRINT**) qui est mise au début de la ligne du bas.

Pour visualiser ces opérations, exécutez ce programme :

```
1Ø SCROLL
2Ø INPUT A$
3Ø PRINT A$
4Ø GOTO 1Ø
```

Résumé

Rubriques de **PRINT** : **AT, TAB**
Instructions : **CLS, SCROLL**

Exercices

1. Essayez d'exécuter ce programme :

```
1Ø FOR I=Ø TO 2Ø
2Ø PRINT TAB 8*I;I;
3Ø NEXT I
```

Il vous permet de comprendre exactement ce que veut dire la réduction modulo 32 du numéro associé à **TAB**. Vous pouvez obtenir un exemple plus élégant en remplaçant par un 6 le 8 de la ligne 2Ø.

ZX81

Chapitre 18

ZX81

Chapter 13

Les graphiques

Passons maintenant à l'étude de quelques-unes des fonctions les plus élégantes du ZX81 ; ces fonctions font appel à des éléments d'image. L'écran que vous pouvez employer est composé de 22 lignes et de 32 colonnes, c'est-à-dire 22*32=7Ø4 positions dont chacune contient 4 éléments d'image.

Un élément d'image est caractérisé par deux valeurs numériques, ses coordonnées. La première — la coordonnée x — précise sa position par rapport à la colonne de gauche (rappelez-vous que X va de gauche à droite) et la deuxième — la coordonnée y — précise sa distance par rapport à la ligne du bas. Généralement, les coordonnées sont écrites sous forme de deux valeurs numériques entre parenthèses. Exemples : (Ø,Ø) (63,Ø), (Ø,43) et (63,43) représentent le coin inférieur de gauche, le coin inférieur de droite, le coin supérieur de gauche et le coin supérieur de droite.

L'instruction

PLOT coordonnée-x, coordonnée-y

met en noir les éléments d'image correspondant aux coordonnées alors que l'instruction

UNPLOT coordonnée-x, coordonnée-y

provoque leur effacement.

Essayez le programme suivant dit programme de la "varicelle" :

```
1Ø PLOT INT (RND*64), INT (RND*44)
2Ø INPUT A$
3Ø GOTO 1Ø
```

Ce programme inscrit sur l'écran un point aléatoire chaque fois que vous appuyez sur NEWLINE.

Voici un programme plus utile. Il trace la courbe de la fonction **SIN** (une sinusoïde) pour ses valeurs entre 0 et 2π.

```
1Ø FOR N=Ø TO 63
2Ø PLOT N, 22+2Ø*SIN (N/32*PI)
3Ø NEXT N
```

Le suivant dessine la courbe de **SQR** (une partie de parabole) entre Ø et 4.

```
1Ø FOR N=Ø TO 63
2Ø PLOT N,2Ø*SQR (N/16)
3Ø NEXT N
```

Notez que les coordonnées des éléments d'image sont assez différentes du numéro de ligne et de colonne d'une rubrique **AT**. A la fin de ce chapitre vous trouverez un schéma qui vous aidera probablement à déterminer les coordonnées des éléments d'images ainsi que les numéros des lignes et des colonnes.

Exercices

1. Il existe trois différences entre les nombres figurant dans une rubrique **AT** et les coordonnées d'un élément d'image ; quelles sont ces différences ?

Si une position d'écriture **PRINT** correspond à **AT** L,C (ligne et colonne) démontrez que les quatre éléments de cette position ont les coordonnées suivantes : Coordonnées x 2*C ou 2*C+1 et coordonnées y 2*(21−L) ou 2*(21−L)+1. (Reportez-vous au schéma.)

2. Vous pouvez transformer le programme précédent en un grignoteur ; c'est-à-dire que l'écran devient d'abord entièrement noir (un carré noir est un espace en vidéo inversée) et que la machine blanchit ensuite des points aléatoires. Si vous disposez seulement d'une mémoire d'1K — la machine standard sans mémoire supplémentaire — vous allez manquer de place dans la mémoire, et vous devrez donc modifier le programme pour qu'il utilise seulement une partie de l'écran.

3. Modifiez le programme de traçage de la fonction **SIN** pour que, avant de dessiner la courbe, le programme trace une ligne horizontale de "−" pour former l'axe des x et une ligne verticale de "+" pour l'axe des y.

4. Ecrivez d'autres programmes pour tracer d'autres fonctions comme **COS**, **EXP**, **LN**, **ATN**, **INT** etc. Dans chaque cas, assurez-vous que le schéma peut s'inscrire sur l'écran ; vous devez donc vous assurer de ce qui suit :

(i) sur quelle plage allez-vous faire évoluer les fonctions (cette plage correspond à la fourchette Ø−2π pour la courbe **SIN**).

(ii) sur l'écran, où allez-vous mettre l'axe des x (qui correspond à 22 sur la ligne 2Ø du programme de traçage de **SIN**).

(iii) comment allez-vous placer l'axe des y du dessin (correspondant à 2Ø sur la ligne 2Ø du programme de traçage de **SIN**).

Vous vous rendrez rapidement compte que la fonction **COS** est la plus facile, comme la fonction **SIN**.

5. Exécutez le programme suivant :

```
1Ø PLOT 21,21
2Ø PRINT "GUILLEMETS GRAS"
3Ø PLOT 46,21
```

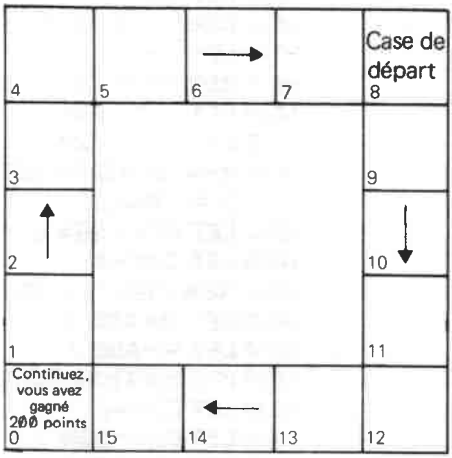
PLOT fait passer à la position d'écriture (**UNPLOT** également).

6. Ce sous-programme dessine une ligne relativement droite entre l'élément d'image (A,B) et l'élément (C,D). Utilisez-le dans un programme principal qui donne les valeurs de A, B, C et D.

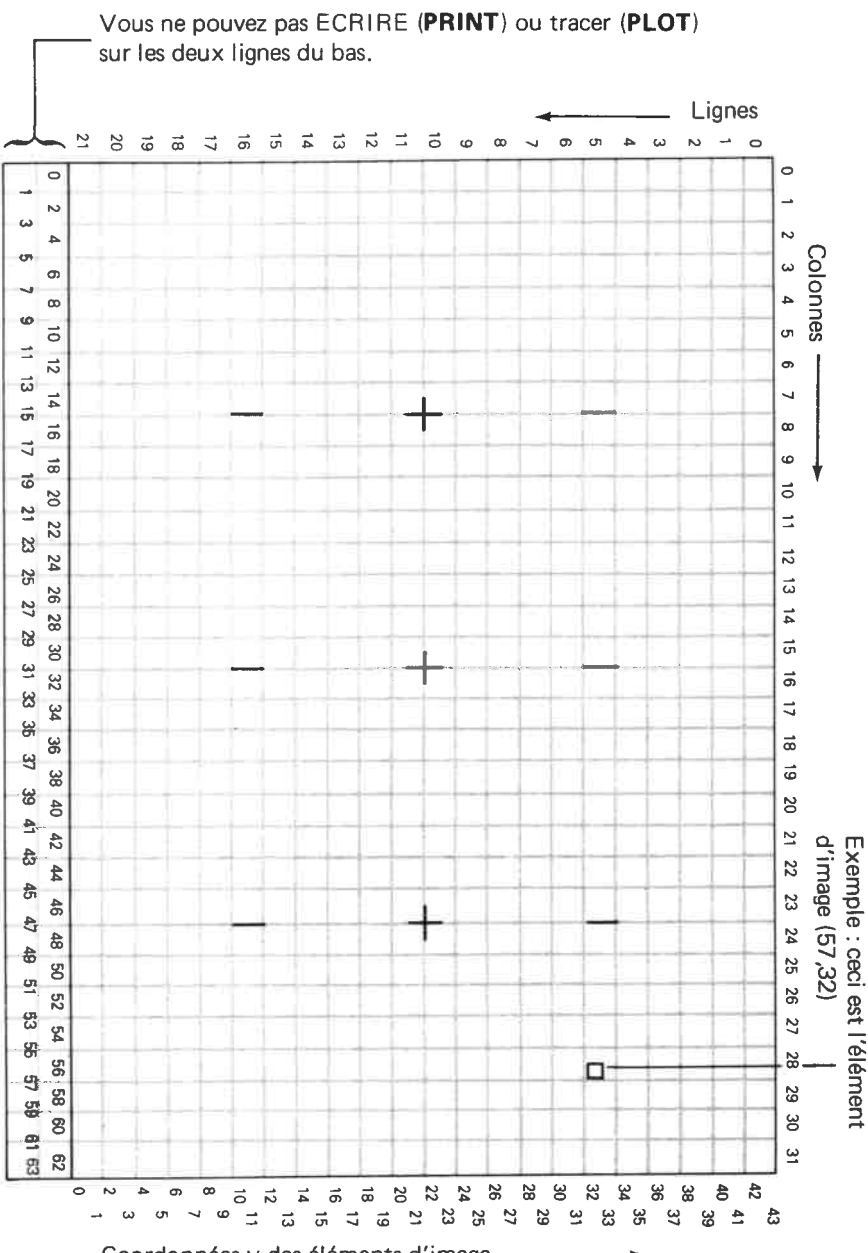
(Si vous ne disposez pas d'un module de mémoire supplémentaire, vous devrez probablement omettre les instructions **REM**).

```
1000 LET U=C-A
1005 REM U INDIQUE LE NOMBRE DE PAS A FAIRE
1010 LET V=D-B
1015 REM V INDIQUE LE NOMBRE DE PAS PARCOURU
1020 LET D1X=SGN U
1030 LET D1Y=SGN V
1035 REM (D1X,D1Y) EST UN SEUL PAS DANS LA DIRECTION
     DIAGONALE
1040 LET D2X= SGN U
1050 LET D2Y=0
1055 REM (D2X,D2Y) EST UN SEUL PAS A GAUCHE OU A DROITE
1060 LET M=ABS U
1070 LET N=ABS V
1080 IF M>N THEN GOTO 1130
1090 LET D2X=0
1100 LET D2Y=SGN V
1105 REM MAINTENANT (D2X,D2Y) EST UN SEUL PAS VERS LE
     HAUT OU LE BAS
1110 LET M=ABS V
1120 LET N=ABS U
1130 REM M EST LE PLUS GRAND DE ABSU ET ABSV, N EST LE
     PLUS PETIT
1140 LET S=INT (M/2)
1145 REM NOUS VOULONS PASSER DE (A,B) A (C,D) EN M PAS PAR
     N PAS VERS LE HAUT-BAS OU A DROITE-GAUCHE DE
     GRANDEUR D2, ET PAR M-N PAS DIAGONAUX DE LONGUEUR
     D1, DISTRIBUES AUSSI REGULIEREMENT QUE POSSIBLE
1150 FOR I=0 TO M
1160 PLOT A,B
1170 LET S=S+N
1180 IF S<M THEN GOTO 1230
1190 LET S=S-M
1200 LET A=A+D1X
1210 LET B=B+D1Y
1215 REM PAS DIAGONAL
1220 GOTO 1250
1230 LET A=A+D2X
1240 LET B=B+D2Y
1245 REM PAS VERS LE HAUT-BAS OU A DROITE-GAUCHE
1250 NEXT I
1260 RETURN
```

La dernière partie (la ligne 115Ø et les suivantes) mélange les M-N PAS de longueur D1 régulièrement avec les N PAS de longueur D2. Imaginez un jeu de Monopoly comportant M carrés en bord, ces cases étant numérotées de 0 à M—1. La case sur laquelle vous êtes à un moment quelconque est le numéro S, à partir du coin opposé à GO. Chaque jet de dés vous fait avancer de N cases autour du jeu et, sur la ligne droite de l'écran, vous faites un pas de gauche-droite/ vers le haut-bas (si vous passez sur GO du jeu) ou, autrement, un pas diagonal. Puisque la longueur totale parcourue sur le jeu est M*N pas ou N fois le tour du jeu, vous passez GO N fois et, régulièrement espacés dans vos pas M, il y a N pas gauche-droite/haut-bas.



Ajustez le programme de manière que si un autre paramètre (E) est à 1, la ligne est dessinée en noir (comme dans notre cas) tandis que s'il est à zéro, la ligne est dessinée en blanc (avec **UNPLOT**). Vous pouvez également effacer une ligne que vous venez de tracer.


Vous ne pouvez pas ECRIRE (**PRINT**) ou tracer (**PLOT**) sur les deux lignes du bas.
Lignes
Colonnes
Exemple : ceci est l'élément d'image (57,32)
Coordonnées x des éléments d'image
Coordonnées y des éléments d'image

ZX81

Chapitre 19

Temps et mouvement

Vous voudrez parfois que le programme ait une certaine durée et, pour l'obtenir, vous pourrez utiliser l'instruction **PAUSE** qui est particulièrement utile en mode rapide :

PAUSE n

arrête les calculs et affiche l'image pendant n trames de la télévision (à raison de 50 trames par seconde en France et de 60 aux Etats-Unis). Ce système peut prendre la valeur maximale 32767, ce qui vous donne tout juste un peu moins de 11 minutes. Si n est plus grand que 32767, l'instruction veut dire "**PAUSE** pour toujours".

Une pause peut toujours être arrêtée par la manœuvre d'une touche (notez qu'un espace ou le signe £ provoque également une interruption). Vous devez appuyer sur la touche après le commencement de la pause.

A la fin de la pause, l'écran clignote. Si une instruction **PAUSE** est insérée dans un programme qui sera exécuté en mode rapide, ou sur l'ancien ZX80 équipé du module de mémoire morte 8K, l'instruction **PAUSE** doit être suivie par **POKE** 16437,255. L'instruction **PAUSE** semblera opérer correctement sans ce complément, mais ceci provoquera probablement l'effacement de votre programme.

Ce programme calcule l'heure indiquée par la grande aiguille (dans notre cas, un point placé en bord) d'une horloge.

```
 5 REM DESSINONS L'AVANT DE L'HORLOGE
10 FOR N=1 TO 12
20 PRINT AT 10-10*COS (N/6*PI),10+10*SIN (N/6*PI);N
30 NEXT N
35 REM DEMARRONS L'HORLOGE
40 FOR T=0 TO 10000
45 REM T EST LE TEMPS EN SECONDES
50 LET A=T/30*PI
60 LET SX=21+18*SIN A
70 LET SY=22+18*COS A
200 PLOT SX,SY
300 PAUSE 42
310 POKE 16437,255
320 UNPLOT SX,SY
400 NEXT T
```

(Vous devez mettre les instructions **REM** seulement si vous disposez d'un module de mémoire supplémentaire.)

L'horloge s'arrête au bout de 2 heures 3/4 environ à cause de la ligne 40 mais vous pouvez aisément la faire fonctionner plus longtemps. Notez l'action de contrôle de la ligne 300. Vous pouvez penser que **PAUSE** 50 va faire avancer l'horloge une fois par seconde, mais les calculs prenant un peu de temps, il faut le prévoir. La meilleure méthode est de procéder par essais pour synchroniser l'horloge de l'ordinateur sur une horloge

réelle et ajuster ensuite la ligne 3ØØ jusqu'au moment où les deux horloges concordent. (Il est difficile, sinon impossible de faire cette opération très précisément ; un ajustement d'une trame par seconde représente 2 %, soit une demi heure par jour.)

La fonction **INKEY$** (sans argument) lit le clavier. Si vous appuyez exactement sur une touche (ou bien sur la touche SHIFT et sur une autre touche), vous obtenez comme résultat le caractère que cette touche donne en mode **L** ; autrement, le résultat est une chaîne vide. Les caractères de contrôle n'agissent pas de la façon habituelle, ils donnent des résultats comme **CHR$** 118 pour NEWLINE — leur écriture est de la forme "?".

Essayez ce programme qui opère comme une machine à écrire.

```
10 IF INKEY$<>"" THEN GOTO 10
20 IF INKEY$="" THEN GOTO 20
30 PRINT INKEY$;
40 GOTO 10
```

Dans cette séquence, la ligne 1Ø attend que vous n'agissiez plus sur le clavier et la ligne 2Ø attend que vous appuyiez sur une nouvelle touche.

Rappelez-vous que contrairement à **INPUT**, **INKEY$** ne vous attend pas. Vous n'avez donc pas à appuyer sur NEWLINE, mais, par contre, si vous n'avez rien entré, vous avez raté une possibilité.

Exercices

1. Que se passe-t-il si vous ratez la ligne 1Ø dans le programme machine à écrire ?

2. Pourquoi ne pouvez-vous pas entrer un espace ou £ dans le programme machine à écrire ?

Voici un programme modifié qui vous donne un espace si vous appuyez sur la touche de déplacement du curseur vers la droite (8 en position majuscules).

```
10 IF INKEY$<>"" THEN GOTO 10
20 IF INKEY$="" THEN GOTO 20
30 LET A$=INKEY$
40 IF A$=CHR$ 115 THEN GOTO 110
90 PRINT A$;
100 GOTO 10
110 PRINT " ";
120 GOTO 10
```

Notez de quelle manière nous mettons **INKEY$** dans A$ dans la ligne 3Ø. Il est impossible de rater cette opération et de remplacer A$ par **INKEY$** aux lignes 4Ø et 9Ø, mais il y a toujours le risque que **INKEY$** puisse changer entre deux lignes.

Complétez la programmation de manière que la manœuvre de **NEWLINE** (**CHR$** 118) vous donne une nouvelle ligne.

3. Une autre façon d'utiliser **INKEY$** est en association avec **PAUSE** comme dans cet autre programme machine à écrire.

```
10 PAUSE 40000
20 POKE 16437,255
30 PRINT INKEY$;
40 GOTO 10
```

Pour être sûr que ce programme a fonctionné, pourquoi est-il essentiel qu'une pause ne se termine pas si elle vous trouve déjà en train d'appuyer sur une touche lorsqu'elle commence ?

Cette méthode a un inconvénient — l'écran clignote ; toutefois, en mode rapide, c'est la seule façon de faire cette opération. Exécutez ce programme en mode rapide et remarquez que l'ordinateur profite d'une pause pour mettre l'image sur l'écran.

4. Celui-ci va vous faire sortir de vos gonds. L'ordinateur affiche un nombre que vous (ou toute autre victime innocente) devez ré-entrer. Pour commencer, il faut savoir que vous disposez d'une seconde pour faire cette entrée, mais que, si vous vous trompez, vous disposez d'un temps plus long pour le nombre suivant, alors que si vous réussissez, vous disposez d'un temps plus court pour la prochaine entrée. L'idée recherchée est d'aller aussi vite que possible et d'appuyer ensuite sur la touche Q pour connaître votre score — le plus fort est le meilleur.

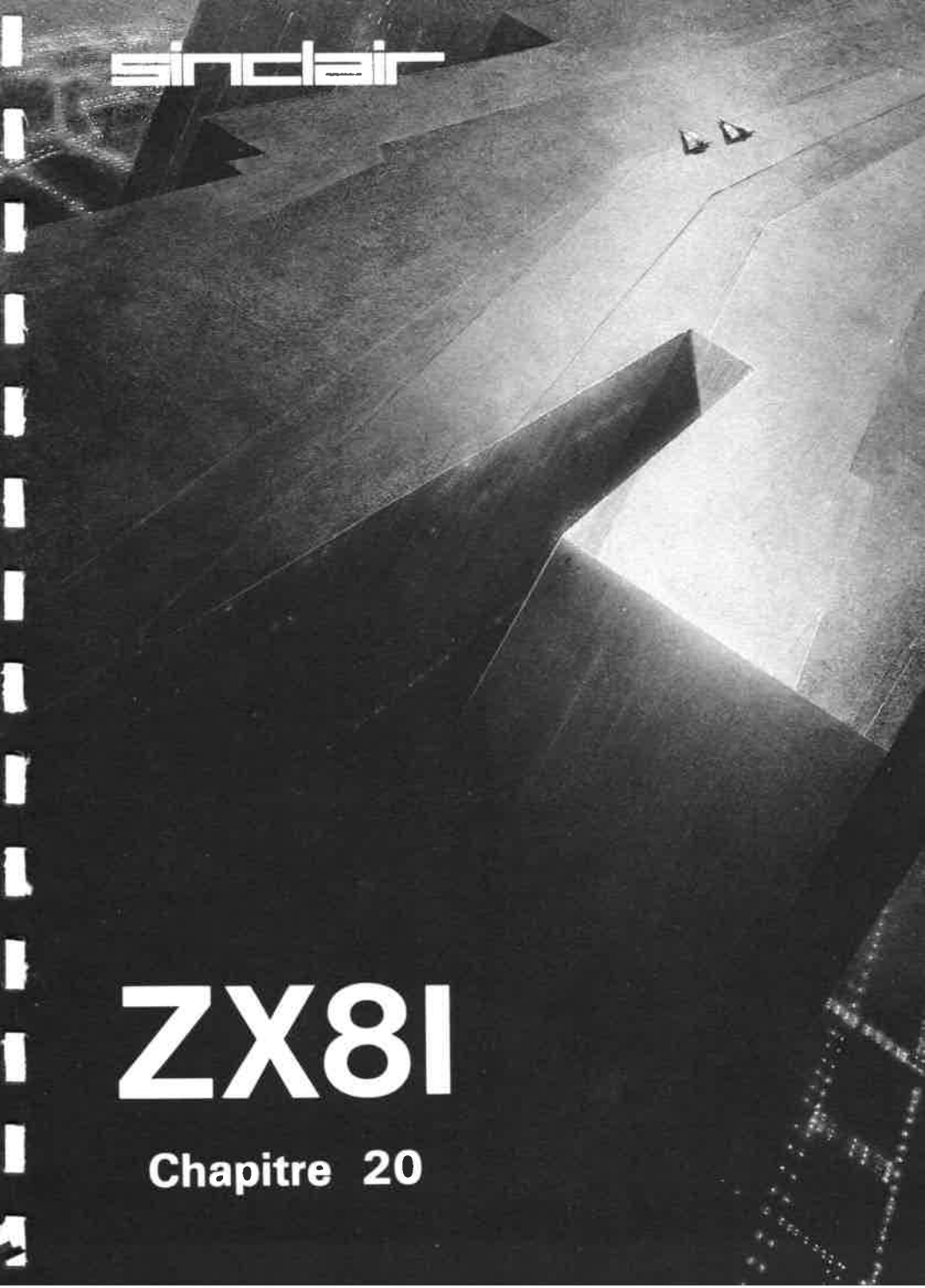
```
10 LET T=50
15 REMT=NOMBRE DE TRAMES PAR GO—INITIALLEMENT 50
   PAR SECONDE
20 SCROLL
30 LET A$=CHR$ INT (RND*10+CODE "0")
35 REM A$ EST UN NOMBRE ALEATOIRE
40 PRINT A$
45 PAUSE T
50 POKE 16437,255
60 LET B$=INKEY$
70 IF B$="Q" THEN GOTO 200
80 IF A$=B$ THEN GOTO 150
90 PRINT "RATE"
100 LET T=T*1.1
110 GOTO 20
150 PRINT "OK"
160 LET T=T*0.9
170 GOTO 20
200 SCROLL
210 PRINT "VOTRE SCORE EST"; INT (500/T)
```

5. (Seulement si vous disposez d'un module de mémoire supplémentaire.) Vous devez utiliser la routine linéaire du chapitre 15 pour modifier le programme de la deuxième aiguille de l'horloge, afin que ce programme affiche deux aiguilles — celle des heures et celles de minutes — et les dessine chaque minute. (Vous devez rendre l'aiguille des heures

plus courte.) Si vous êtes ambitieux, faites le nécessaire pour que tous les quarts d'heure la machine fasse quelque chose de particulier.

6. Pour vous amuser, essayez la séquence suivante :

```
10 IF INKEY$ = "" THEN GOTO 10
20 PRINT AT 11,14; "AIE"
30 IF INKEY$<>"" THEN GOTO 30
40 PRINT AT 11,14; "   "
50 GOTO 10
```



ZX81

Chapitre 20

Chapter 20

L'imprimante de l'ordinateur ZX81

Si vous avez une imprimante ZX81, elle est accompagnée de ses consignes d'utilisation. Ce chapitre est consacré aux principales instructions BASIC qui permettent de la faire fonctionner.

Les deux premières – **LPRINT** et **LLIST** – sont semblables à **PRINT** et **LIST** avec une différence néanmoins, elles utilisent l'imprimante à la place du téléviseur. (La lettre L est un accident historique. Lorsque le langage BASIC a été inventé, il actionnait généralement une machine à écrire électrique à la place d'un téléviseur et, pour cette raison, **PRINT** voulait véritablement dire imprimer. Si vous aviez voulu à cette époque des résultats imprimés importants, vous auriez utilisé une imprimante rapide, imprimant ligne par ligne, connectée à l'ordinateur et une instruction **LPRINT** voulant dire en langue anglaise "line printer **PRINT**".)

Par exemple, essayez ce programme :

```
10 LPRINT "CE PROGRAMME:"...
20 LLIST
30 LPRINT,,"ECRIT LE JEU DE CARACTERES" ...
40 FOR N=0 TO 255
50 LPRINT CHR$ N ;
60 NEXT N
```

La troisième instruction **COPY** écrit une copie du contenu de l'écran du téléviseur. Par exemple, mettez sur l'écran la liste du programme ci-dessus et entrez ensuite :

```
COPY
```

Vous pouvez toujours arrêter l'imprimante en appuyant sur la touche **BREAK** (espace).

Si vous exécutez ces instructions sans connecter l'imprimante, vous allez perdre votre sortie et passer à l'instruction suivante. Toutefois, il se peut qu'un blocage intervienne et, dans ce cas, la touche **BREAK** permet de revenir à l'état normal.

Résumé

Instructions : **LPRINT**, **LLIST**, **COPY**

Nota : Aucune des instructions ci-dessus ne figure dans le BASIC standard, bien que **LPRINT** soit parfois utilisée dans d'autres ordinateurs.

Exercices

1. Essayez ce programme :

```
10 FOR N=31 TO 0 STEP -1
20 LPRINT AT 31-N,N,CHR$ (CODE "0"+N);
30 NEXT N
```

Vous verrez apparaître une série de lettres présentées diagonalement à partir du coin supérieur de droite et jusqu'au bas de l'écran ; ensuite, le programme s'arrête et lance le compte-rendu 5.

Transformez maintenant '**AT** 31–N,N' de la ligne 2Ø en '**TAB** N'. Le programme résultant a exactement le même effet que le précédent.

Transformez maintenant **PRINT** de la ligne 2Ø en **LPRINT**. Cette fois, le compte-rendu d'erreur 5 n'est pas lancé (il ne doit pas intervenir avec l'imprimante) et la forme précédente va contenir dix lignes supplémentaires composées de chiffres.

Transformez maintenant '**TAB** N' en '**AT** 21–N,N' en utilisant **LPRINT**. Cette fois, vous obtenez une seule ligne de symboles. La différence résulte du fait que la sortie causée par **LPRINT** n'est pas écrite directement, elle est mise dans un buffer qui conserve une image d'une ligne de longueur composée de ce que l'ordinateur va imprimer lorsque le moment est venu de faire cette opération. L'impression a lieu :

(i) Lorsque le tampon est plein,

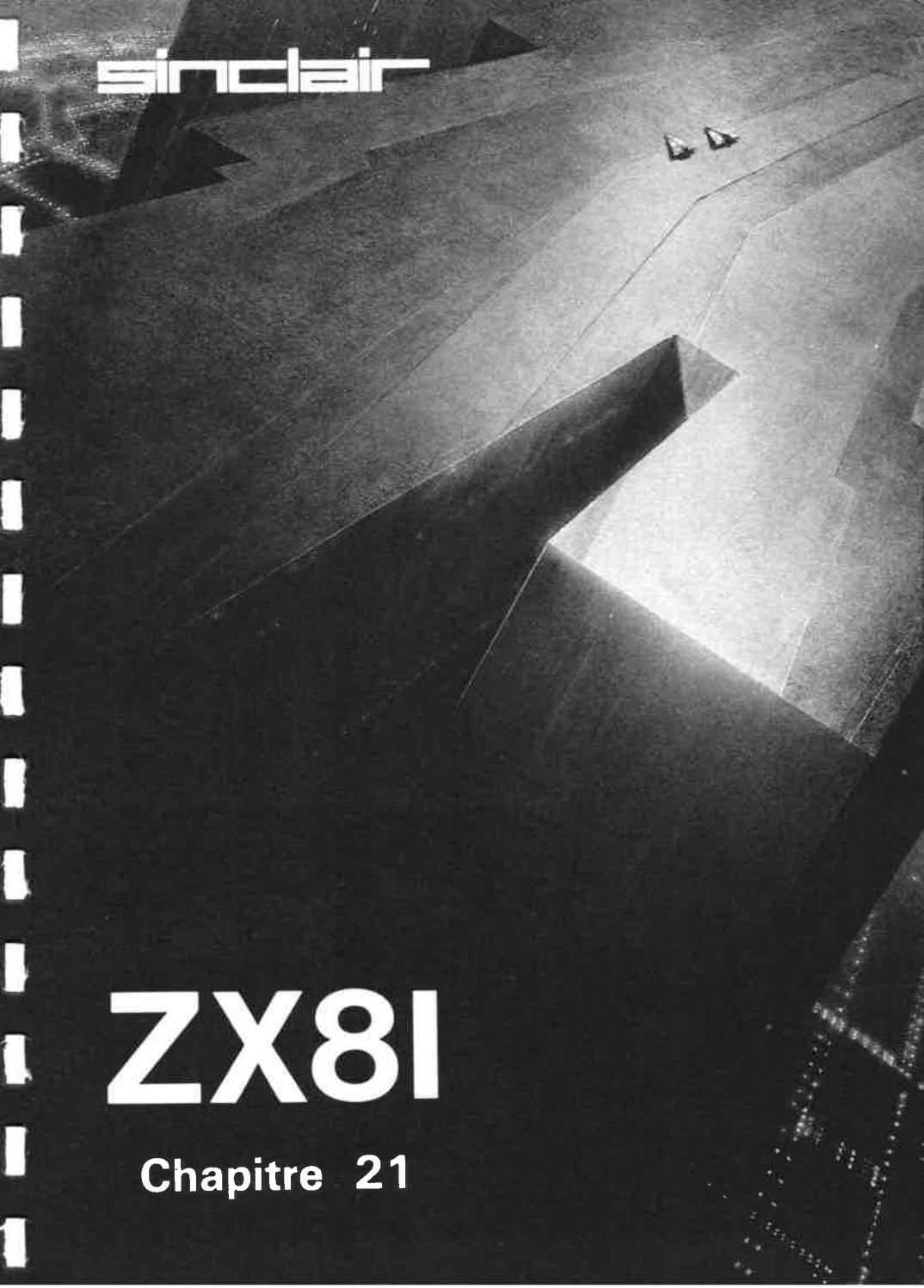
(ii) Après une instruction **LPRINT** qui ne se termine pas par une virgule ou par un point virgule,

(iii) Lorsqu'une virgule ou une rubrique de **TAB** exige une nouvelle ligne, ou

(iv) A la fin du programme, s'il reste encore des données à imprimer.

(iii) Nous allons vous expliquer maintenant pourquoi notre programme avec **TAB** opère comme il le fait. Comme pour **AT**, le numéro de la ligne est ignoré et la position **LPRINT** (comme la position **PRINT** mais pour l'imprimante et non pour le téléviseur) est transformée en numéro de colonne. Une rubrique **AT** ne provoque jamais la transmission d'une ligne à l'imprimante. (En réalité, le numéro de ligne écrit après **AT** n'est pas entièrement ignoré ; il doit être compris dans l'intervalle –21/+21 ; si cet intervalle n'est pas respecté, l'ordinateur lance un message d'erreur. C'est pour cette raison qu'il est toujours plus sûr de spécifier la ligne 0. La rubrique '**AT** 21–N,N' de la version finale de notre programme serait préférable – mais moins informative – si elle était remplacée par '**AT** Ø , N'.).

2. Faites un schéma imprimé de la fonction **SIN** en exécutant le programme du chapitre 18 et en employant ensuite **COPY**.



ZX81

Chapitre 21

Chapitre 27

Les sous-chaînes

Si nous partons d'une chaîne, une sous-chaîne est composée de plusieurs caractères consécutifs de cette chaîne dans leur ordre initial. Ainsi "CHAINE" est une sous-chaîne de "CHAINE LONGUE", mais "B ABC" et "BORD FRANC" ne sont pas liées.

Une notation particulière le découpage permet de décrire les sous-chaînes, elle peut être appliquée aux expressions chaîne arbitraires. Format général :

expression chaîne (début **TO** fin)

de sorte que, par exemple :

"ABCDEF" (2 **TO** 5) = "BCDE"

Si vous omettez 'début', la valeur implicite correspondante est 1 ; si vous omettez 'fin', la valeur implicite est celle de la longueur de la chaîne. Ainsi :

"ABCDEF" (**TO** 5) = "ABCDEF" (1 **TO** 5) = "ABCDE"
"ABCDEF" (2 **TO**) = "ABCDEF" (2 **TO** 6) = "BCDEF"

et

"ABCDEF" (**TO**) = "ABCDEF" (1 **TO** 6) = "ABCDEF"

(la dernière peut également s'écrire "ABCDEF" (), si toutefois cela vaut quelque chose.)

Une forme légèrement différente évite d'utiliser **TO** et ne comporte qu'un seul nombre :

"ABCDEF" (3) = "ABCDEF" (3 **TO** 3) = "C"

Bien que début et fin doivent habituellement citer des parties existantes de la chaîne, cette règle est outrepassée par une autre : si le début se trouve au-delà de la fin, le résultat est une chaîne vide. Ainsi :

"ABCDEF" (5 **TO** 7)

donne le compte-rendu d'erreur 3 (erreur d'indice) parce que la chaîne contient seulement 6 caractères et que 7 est au-delà de cette valeur mais

"ABCDEF" (8 **TO** 7) = ""

et

"ABCDEF" (1 **TO** Ø) = ""

Le début et la fin ne doivent jamais être négatifs ; autrement, la machine lance l'erreur B.

Le programme suivant a pour rôle de rendre B$ égal à A$ en omettant les espaces de fin.

```
10 INPUT A$
20 FOR N=LEN A$ TO 1 STEP -1
30 IF A$(N)<>" " THEN GOTO 50
40 NEXT N
50 LET B$=A$( TO N)
60 PRINT """";A$;""","""";B$;""""
70 GOTO 10
```

Notez que si A$ est composé uniquement d'espaces, à la ligne 50 nous avons N = 0 et A$ (**TO** N) = A$ (1 **TO** 0) = " ".

Pour les variables chaîne, nous pouvons non seulement extraire des sous-chaînes, mais également procéder à des affectations. Par exemple, entrez

```
LET A$ ="LAURE AIME UN CANARD"
```

et ensuite

```
LET A$ (5 TO 8) = "******"
```

et

```
PRINT A$
```

Remarquez que puisque la sous-chaîne A$ (5 **TO** 8) contient seulement 4 caractères, seules les quatre premières étoiles ont été utilisées. Ceci est une caractéristique de l'affectation des sous-chaînes : la sous-chaîne doit avoir après l'opération exactement la même longueur qu'avant celle-ci. Pour être sûr que ceci se produit bien, la chaîne pour laquelle l'affectation intervient est coupée sur la droite si elle est trop longue, ou remplie d'espaces si elle est trop courte — ceci s'appelle l'affectation de Procrustre ; elle a été nommée du nom de l'aubergiste Procruste qui s'assurait que ses hôtes étaient bien de la même taille que leur lit, soit en les raccourcissant en leur coupant les pieds, soit encore en les allongeant avec un instrument de torture.

Si vous essayez maintenant

```
LET A$ () = "COR BLIMEY"
```

et

```
PRINT A$;"."
```

vous verrez que la même chose se reproduit (cette fois avec une insertion d'espaces) parce que A$ () est compté comme une sous-chaîne.

```
LET A$ = "COR BLIMEY"
```

fait le travail correctement.

Le découpage peut être considéré comme ayant la priorité 12 ; par conséquent :

```
LEN "ABCDEF" (2 TO 5) = LEN ("ABCDEF" (2 TO 5)) = 4
```

Les expressions chaînes compliquées doivent être mises entre parenthèses avant d'être découpées. Par exemple :

```
"ABC"+"DEF"(1 TO 2)  = "ABCDE"
("ABC"+"DEF")(1 TO 2) = "AB"
```

Résumé

Découpage avec **TO**. Notez que cette notation n'est pas standard.

Exercices

1. Certains langages BASIC (mais pas celui du ZX81) ont des fonctions appelées LEFT$, RIGHT$, MID$ et TL$.

LEFT$ (A$,N) donne la sous-chaîne de A$ composée des N premiers caractères.

RIGHT$ (A$,N) donne la sous-chaîne de A$ composée des caractères à partir du Nième.

MID$ (A$,N1,N2) donne la sous-chaîne de A$ composée de N2 caractères à partir du N1ème.

TL$ (A$) donne la sous-chaîne de A$ composée de tous ses caractères à l'exception du premier.

Comment écririez-vous ce qui précède en BASIC ZX81 ; est-ce que vos réponses pourraient opérer sur des chaînes de longueur Ø ou 1 ?

2. Essayez cette suite de commandes :

```
LET A$="X"+"Y"
LET A$(2)=CHR$ 11          (le caractère guillemet de chaîne)
LET A$(4)=CHR$ 11
PRINT A$
```

A$ est maintenant une chaîne contenant des guillemets de chaîne ! Bien entendu, rien ne vous empêche de faire cette opération si vous persévérez assez longtemps, mais, bien évidemment, si vous aviez initialement entré :

```
LET A$="X"+"Y"
```

la partie située à droite du signal égal aurait été traitée comme une expression, donnant à A$ la valeur "XY".

Entrez maintenant :

```
LET B$="X""+""Y"
```

Vous vous rendrez compte que même si A$ et B$ ont le même aspect lorsqu'elles sont écrites, elles ne sont pas égales — essayez

```
PRINT A$=B$
```

Alors que B$ contient simplement des caractères image guillemet (avec le code 192), A$ contient de véritables caractères guillemet de chaîne (code 11).

3. Exécutez ce programme :

```
10 LET A$="LEN ""ABCD"""
100 PRINT A$;" = ";VAL A$
```

Ce programme est faux parce que **VAL** ne traite pas l'image de guillemet "" comme un guillemet de chaîne.

Ajoutez des lignes supplémentaires entre 10 et 100 pour remplacer les images de guillemet de A$ par des guillemets de chaîne (que vous devez appeler **CHR$** 11) et essayez à nouveau.

Faites le même type de modification sur le programme du chapitre 9, exercice 3, et faites ensuite des expérimentations avec ce programme.

4. Ce sous-programme supprime toutes les apparitions de la chaîne "CARTHAGO" de A$.

```
1000 FOR N=1 TO LEN A$-7
1020 IF A$(N TO N+7)="CARTHAGO" THEN LET
A$(N TO N+7)="********"
1030 NEXT N
1040 RETURN
```

Ecrivez un programme donnant plusieurs valeurs à A$ (par exemple "DELENDA EST CARTHAGO"") et qui utilise le sous-programme.

ZX81

Chapitre 22

Chapitre 22

Tableaux

Supposons que vous disposiez maintenant d'une liste de nombres par exemple le nombre de percepteurs décédés chaque année pendant l'année fiscale en cours. Pour les mettre dans l'ordinateur, vous pourriez établir une seule variable pour chaque mois mais ceci serait très compliqué. Vous pourriez décider d'appeler les variables HELAS PLUS DE 1, HELAS PLUS DE 2, etc. jusqu'à HELAS PLUS DE 12, mais le programme qui permettrait d'écrire ces douze nombres serait long et ennuyeux à entrer. Il serait beaucoup plus agréable d'entrer le programme suivant :

```
 5 REM CE PROGRAMME NE VA PAS MARCHER
1Ø FOR N=1 TO 12
2Ø PRINT HELAS PLUS DE N
3Ø NEXT N
```

Mais c'est impossible.

Voici maintenant un mécanisme qui vous permet d'utiliser cette idée en employant des tableaux. Un tableau est composé d'une série de variables, ses éléments, portant tous le même nom et distingués seulement par un numéro (l'indice) écrit entre parenthèses après le nom. Dans notre exemple, le nom serait A (comme les variables de contrôle des boucles **FOR-NEXT**, le nom d'un tableau doit être composé d'une seule lettre) et les douze variables seraient alors A(1), A(2), etc. jusqu'à A(12).

Les éléments d'un tableau sont appelés variables indicées par opposition aux variables simples que vous connaissez déjà. Avant d'utiliser un tableau, vous devez lui réserver de l'espace dans l'ordinateur en utilisant une instruction DIM (l'abréviation de dimensions).

```
DIM A(12)
```

établit un tableau appelé A de dimension 12 (c'est-à-dire qu'il existe 12 variables indicées A(1)..., A(12)), et initialise les 12 valeurs à Ø. Cette instruction supprime également un tableau appelé A s'il existait auparavant. (Ce n'est pas le cas pour une variable simple. Un tableau et une variable numérique simple du même nom peuvent co-exister et il ne doit pas y avoir de confusion entre les deux parce que la variable du tableau a toujours un indice.)

L'indice peut être une expression numérique arbitraire ; vous pouvez donc écrire :

```
1Ø FOR N=1 TO 2
2Ø PRINT A(N)
3Ø NEXT N
```

Vous pouvez également créer des tableaux à plusieurs dimensions. Dans un tableau à deux dimensions, vous avez besoin de deux nombres pour désigner l'un des éléments — comme le numéro de la ligne et celui de la colonne pour préciser la position d'un caractère sur l'écran de télévision — et, dans ce cas, le tableau est vraiment un tableau. Vous pouvez également imaginer que les numéros de la ligne et de la colonne (les deux dimensions) citent une page imprimée et que vous pouvez avoir une autre dimension pour

les numéros des pages. Bien entendu, nous parlons de tableaux numériques ; dans ce cas, les éléments ne sont pas des caractères imprimés comme dans un livre, ce sont des valeurs numériques. Pensez aux éléments d'un tableau à trois dimensions C ; pour les décrire, il faut indiquer C (numéro de la page, numéro de la ligne, numéro de la colonne).

Par exemple, pour créer un tableau B à deux dimensions (3 et 6) vous utilisez une instruction **DIM**

DIM B(3,6)

Ainsi, vous obtenez 3*6=18 variables indicées

B(1,1), B(1,2), . . . , B(1,6)
B(2,1), B(2,2), . . . , B(2,6)
B(3,1), B(3,2), . . . , B(3,6)

Le même principe s'applique à un nombre quelconque de dimensions.

Bien que vous puissiez désigner un nombre et un tableau par le même nom, vous ne pouvez pas désigner deux tableaux par le même nom, même s'ils ont un nombre différent de dimensions.

Il existe aussi des tableaux de chaînes. Les chaînes d'un tableau diffèrent des chaînes simples lorsqu'elles sont de longueur fixe et que l'affectation est toujours du type de Procruste — une autre façon de les imaginer est de dire qu'il s'agit de tableaux (avec une dimension supplémentaire) de caractères seuls. Le nom d'un tableau de chaîne est une seule lettre suivie de $; un tableau de chaînes et une variable chaîne simple ne peuvent pas avoir le même nom (contrairement au cas des nombres).

Supposons maintenant que vous vouliez un tableau A$ de cinq chaînes. Vous devez décider de la longueur que devront avoir les chaînes. Supposons que 1Ø caractères suffisent pour chacune. Vous dites alors :

DIM A$ (5,1Ø) (entrez cette instruction)

Elle établit un tableau de 5 * 1Ø caractères, mais vous pouvez également imaginer que chaque ligne est une chaîne :

A$(1) =	A$(1,1)	A$(1,2)	. . .	A$(1,1Ø)
A$(2) =	A$(2,1)	A$(2,2)	. . .	A$(2,1Ø)
:	:	:	:	:
A$(5) =	A$(5,1)	A$(5,2)	. . .	A$(5,1Ø)

Si vous donnez le même nombre d'indices (deux dans ce cas) qu'il existe de dimensions dans l'instruction **DIM**, vous obtenez un seul caractère ; mais, si vous manquez le dernier, vous obtenez une chaîne de longueur fixe. Par exemple, A$(2,7) est le septième caractère de la chaîne A$ (2) ; grâce à la notation "découpée", nous pouvons également écrire ceci sous la forme A$(2) (7). Entrez maintenant

LET A$(2)="1234567891Ø"

et **PRINT** A$(2), A$(2,7)
Vous obtenez

1234567890 7

Pour le dernier indice (celui que vous pouvez rater) vous pouvez également effectuer un découpage pour que, par exemple :

A$(2,4 **TO** 8) = A$(2) (4 **TO** 8) = "45678"

Rappelez-vous :

Dans un tableau de chaînes, toutes les chaînes ont la même longueur fixe.

L'instruction **DIM** comporte un nombre supplémentaire (le dernier) pour spécifier cette longueur.

Lorsque vous écrivez une variable indicée pour un tableau de chaînes, vous pouvez ajouter un nombre supplémentaire (le facteur de découpage) pour qu'il corresponde au nombre supplémentaire écrit dans l'instruction **DIM**.

Résumé

Tableaux (la façon dont le ZX81 traite les tableaux de chaînes est légèrement différente de la méthode standard).

Instructions : **DIM**

Exercices

1. Etablissez le tableau M$ de douze chaînes dans lequel M$ (N) est le nom du Nième mois. (Pour vous aider, noter que l'instruction **DIM** sera **DIM** M$ (12,9). Essayez cette instruction en écrivant tous les M$ (N) (utilisez une boucle).

Ecrivez

```
PRINT "NOUS SOMMES AU MOIS DE ";M$(5);"EMPS";
"OU LES ENFANTS JOYEUX JOUENT
```

Que pouvez-vous faire au sujet de tous ces espaces ?

2. Vous pouvez avoir des tableaux de chaînes sans indication de dimension. Entrez

```
DIM A$(10)
```

et vous vous rendrez compte que A$ se comporte exactement comme une variable chaîne, sauf que sa longueur est toujours 10 et que l'affectation est toujours du type de Procruste.

3. Qui a besoin de READ, DATA et RESTORE ?

La plupart des langages BASIC (mais non celui du ZX81) comportent trois instructions appelées : READ, DATA et RESTORE.

Une instruction DATA est une liste d'expressions et l'ensemble des instructions DATA du programme donne une longue liste d'expressions, la liste des DATA.

Les instructions READ permettent d'affecter ces expressions une à une à des variables :

```
READ X
```

a, par exemple, l'effet d'affecter l'expression courante de la liste de DATA à la variable X et de passer ensuite à l'expression suivante pour la prochaine instruction READ.

(RESTORE permet de revenir au début de la liste des DATA.)

En théorie, vous pouvez toujours remplacer les instructions READ et DATA par des LET ; toutefois, l'une de leurs principales fonctions est l'initialisation des tableaux, comme dans le programme suivant :

```
 5 REM CE PROGRAMME NE PEUT PAS FONCTIONNER EN BASIC
   ZX81
1Ø DIM M$(12,3)
2Ø FOR N=1 TO 12
3Ø READ M$(N)
4Ø NEXT N
5Ø DATA "JAN","FEV","MAR","AVR"
6Ø DATA "MAI","JUN","JUI","AOU"
7Ø DATA "SEP","OCT","NOV","DEC"
```

Si vous voulez exécuter ce programme une seule fois, il est souhaitable de remplacer la ligne 3Ø par une instruction **INPUT** et le programme devient :

```
1Ø DIM M$(12,3)
2Ø FOR N=1 TO 12
3Ø INPUT M$(N)
4Ø NEXT N
```

et vous éviterez ainsi une partie du travail de frappe. Toutefois, si vous voulez sauvegarder le programme, vous ne voudrez certainement pas entrer les mois chaque fois que vous l'exécutez.

Nous vous suggérons la méthode suivante :

(i) Initialisez le tableau à l'aide d'un programme semblable au précédent.

(ii) Editez le programme d'initialisation (n'utilisez pas **NEW** puisque vous voulez préserver le tableau).

(iii) Entrez le reste du programme et sauvegardez le tout. Vous préserverez également les variables, y compris le tableau.

(iv) Lorsque vous rechargerez le programme, vous chargerez également le tableau.

(v) Lorsque vous exécuterez le programme, n'utilisez pas **RUN** qui efface les variables. Utilisez **GOTO** avec un numéro de ligne.

Vous pouvez également mettre en œuvre la technique **LOAD** et exécution du chapitre 16 (exercice 3). Ensuite, à l'étape (iii) ci-dessus, vous mettrez une instruction **SAVE** dans le programme et l'étape (v) se trouve alors entièrement supprimée.

sinclair

ZX81

Chapitre 23

Lorsque l'ordinateur est plein

Le ZX81 n'a qu'une mémoire interne limitée qu'il n'est guère difficile de remplir. En général, la meilleure indication de l'effet de remplissage est un compte-rendu d'erreur 4 ; d'autres choses peuvent néanmoins se passer et certaines sont particulièrement bizarres. Le comportement exact de la machine dépend de la présence d'un module supplémentaire de mémoire ; nous allons commencer par supposer que vous n'en avez pas. Si vous en avez un, enlevez-le (après la première mise hors tension de l'ordinateur).

Le fichier d'affichage, c'est-à-dire la partie intérieure de l'ordinateur où ce dernier met l'image de la télévision, est conçu pour occuper seulement la place de ce qui a été imprimé jusqu'à maintenant : une ligne de l'écran composée de 32 caractères au maximum et d'un caractère **NEWLINE**. Ceci veut dire que vous pouvez tomber en panne de mémoire en écrivant quelque chose et que le risque le plus grand se situe au moment de l'établissement d'une liste. Entrez

```
NEW
DIM A(150)
10 FOR N=1 TO 15
20 PRINT N
```

Et voici la première surprise : la ligne 10 disparaît de la liste. La liste doit nécessairement contenir la ligne en cours — la ligne 20, mais il n'y a pas de place pour les deux lignes. Entrez maintenant :

```
30 NEXT N
```

Encore une fois, la place disponible dans la liste est seulement celle de la ligne 30. Entrez maintenant :

```
40 REM X          (sans appuyer sur NEWLINE)
```

et vous voyez la ligne 30 disparaître et la ligne 40 sauter en haut de l'écran. Elle n'a pas été mise dans le programme — vous disposez encore du curseur L que vous pouvez déplacer. Tout ce que vous avez constaté est l'existence d'un mécanisme curieux qui donne à la moitié inférieure de l'écran 24 lignes qui lui donnent priorité par rapport à la moitié supérieure. Entrez maintenant

```
XXXXXX            (toujours pas de NEWLINE)
```

et le curseur disparaît — il n'y a pas de place pour le mettre sur l'écran. Entrez un autre X sans **NEWLINE**, et l'un des X disparaît. Entrez **NEWLINE**. Tout disparaît alors que le programme est encore dans la machine, comme vous pouvez le constater en supprimant la ligne 10 et en utilisant les touches ⇦ et ⇨. Maintenant entrez

```
10 FOR N=1 TO 15
```

à nouveau — cette ligne vient se placer en haut de l'écran comme l'a fait la ligne 40. Mais, lorsque vous appuyez sur **NEWLINE**, elle n'est pas entrée, bien qu'il n'y ait pas

de message d'erreur ou de marque **S** pour vous dire que quelque chose ne va pas. Ceci résulte du fait qu'il n'y a pas de place pour vérifier la syntaxe d'une ligne ; ce curieux phénomène se manifeste seulement pour les lignes contenant des nombres (autres que le numéro de ligne au début).

Le seul remède est de rendre de la place disponible d'un façon ou d'une autre ; commencez par supprimer la ligne 1Ø qui ne veut pas rentrer. Appuyez sur **EDIT**, l'écran se vide puisqu'il n'y a pas de place pour faire descendre la ligne.

Lorsque **EDIT** n'a pas son effet habituel, il est parfois possible de faire de la place en entrant un nombre d'espaces jusqu'au moment où le curseur monte sur l'écran.

Appuyez sur **NEWLINE** et vous obtenez à nouveau une partie de la liste. Supprimez maintenant la ligne 4Ø (que vous ne vouliez pas vraiment de toute façon) en entrant

4Ø (et **NEWLINE**)

Essayez maintenant d'entrer à nouveau la ligne 1Ø — ça marche toujours pas ! Effacez-la une nouvelle fois. Vous devez quand même trouver de l'espace quelque part. N'oubliez pas que la raison pour laquelle la ligne 1Ø a été rejetée est probablement qu'il n'y avait pas de place pour vérifier la syntaxe des deux valeurs numériques, 1 et 15 : par conséquent, si vous supprimez la ligne 2Ø du programme, vous aurez peut-être la place requise pour entrer la ligne 1Ø et quand même assez de place pour ré-entrer la ligne 2Ø (qui ne contient aucun nombre) ensuite. Essayez en entrant

```
2Ø
1Ø FOR N=1 TO 5
2Ø PRINT N
```

et le programme est bien entré correctement.

Ecrivez

GOTO 1Ø

et, une fois encore, vous vous rendez compte que la ligne est refusée lorsque sa syntaxe ne peut pas être vérifiée. Toutefois, si vous effacez et entrez

RUN

tout marche bien. (**RUN** efface le tableau, rendant ainsi un important espace disponible).

Entrez maintenant la même séquence qu'auparavant, de **NEW** à la ligne 3Ø et, ensuite

4Ø **REM** XXXXXXXXXXXX

(12 caractères X), qui, en fin de compte, a l'aspect 4ØRE. Lorsque vous appuyez sur **NEWLINE**, la liste sera seulement composée de la ligne 3Ø et, en pratique, la ligne 4Ø est complètement perdue. Ceci est dû au fait que cette ligne était trop longue pour le programme. Les conséquences sont encore un peu plus mauvaises lorsque la ligne est une version allongée d'une ligne qui se trouve déjà dans le programme car dans cette situation, vous perdez à la fois l'ancienne ligne du programme et la nouvelle qui devait la remplacer.

Le meilleur remède consiste à acheter un module supplémentaire de mémoire et à le connecter à l'arrière de votre ordinateur.

Le module Sinclair 16K équipe votre machine d'une mémoire 16 fois plus importante que celle de l'ordinateur de base.

Nota : Si vous avez un module 3K ZX80, vous ne pouvez pas l'utiliser avec le ZX81.

Avec le module de mémoire, le comportement constaté est très différent puisque le fichier d'affichage est rempli d'espaces afin que chaque ligne ait une longueur de 32 caractères (Notez que **SCROLL** va à l'encontre de ceci — voir le chapitre 27). Maintenant, les opérations d'écritures et d'établissement de listes ne risquent pas de provoquer une pénurie de mémoire et vous n'aurez plus à subir de listes raccourcies et de sauts ; mais il vous arrivera quand même de perdre des lignes ou de les voir se bloquer et, ici également, le seul remède est de trouver de l'espace de réserve.

Si vous disposez d'une carte d'extension mémoire, connectez-la et refaites les entrées de ce chapitre à l'aide de

DIM A(3Ø69)

pour remplacer **DIM** A(15Ø).

En résumé, toute l'histoire est assez simple et la morale est, si vous le pouvez, de ne pas bourrer votre ordinateur de données. Cette histoire comporte une deuxième morale, souvent, les choses ne sont pas aussi mauvaises qu'elles paraissent.

1. Si une liste est tronquée ou si les choses commencent à sauter à droite et à gauche, vous subissez les conséquences d'un manque d'espace.

2. Si **NEWLINE** semble sans effet à la fin d'une ligne, il n'y a probablement pas assez de place pour traiter un nombre. Effacez la ligne à l'aide de **EDIT-NEWLINE** ou **RUBOUT.**

3. Il est possible qu'une manœuvre de **NEWLINE** provoque la perte d'une ligne complète.

Dans tous les cas curieux décrits ci-dessus, le remède est le même. Ne vous affolez pas, recherchez de l'espace inoccupé.

Le premier moyen à envisager est **CLEAR**. Si vous utilisez quelques variables et que vous n'avez pas peur de les perdre, il faut utiliser **CLEAR**.

Autrement, il faut rechercher les instructions inutiles du programme, par exemple, les **REM** et en supprimer une partie.

Résumé

Lorsque la mémoire se remplit, des choses bizarres peuvent avoir lieu. Habituellement, ces choses ne sont pas fatales.

ZX81

Chapitre 24

Chapitre 24

Comptons sur nos doigts

Le chapitre suivant va nous permettre d'aller voir un petit peu ce qui se passe dans l'ordinateur ; avant de regarder les particularités physiques de la machine, il est quand même préférable de décrire de quelle façon les ordinateurs calculent : les ordinateurs travaillent en système binaire, ce qui veut dire qu'ils n'ont pas de doigts, seulement des pouces.

Dans la plupart des langues européennes, le comptage se fait suivant une structure plus ou moins régulière de dizaines — par exemple, en anglais et en français, bien que le début soit un peu erratique, la structure se stabilise rapidement en groupes réguliers :

vingt, vingt-et-un, vingt-deux, ..., vingt-neuf
trente, trente-et-un, trente-deux, ... trente-neuf
quarante, quarante-et-un, quarante-deux, ... quarante neuf
etc.

cette structure est rendue encore plus systématique par les nombres arabes que nous utilisons. Toutefois, la seule raison pour laquelle nous utilisons la dizaine est que nous avons cinq doigts à chaque main, soit dix doigts en tout.

Supposons maintenant que les martiens aient trois doigts de plus à chaque main (dans la mesure où on peut les appeler doigts) : au lieu d'utiliser notre système décimal à base 10, ils utilisent un système hexadécimal (hex, en abrégé) à base seize. Ils ont donc besoin de six chiffres hexadécimaux supplémentaires en plus de dix que nous utilisons et, il se trouve qu'ils les écrivent sous la forme A, B, C, D, E et F. Quel est le chiffre après F ? Puisqu'avec nos dix doigts nous écrivons 10 pour dix, les martiens qui travaillent en base 16 écrivent 10 pour seize. Leur système numérique commence comme ceci :

Hex	*Français*
Ø	Zéro
1	Un
2	Deux
:	:
:	:
9	Neuf

exactement comme le nôtre, mais il se poursuit comme suit :

A	Dix
B	Onze
C	Douze
D	Treize
E	Quatorze
F	Quinze
10	Seize
11	Dix-sept
:	:
:	:
19	Vingt-cinq
1A	Vingt-six

Hex	*Français*
1B	Vingt-sept
:	:
:	:
1F	Trente-et-un
20	Trente-deux
21	Trente-trois
:	:
:	:
9E	Cent cinquante-huit
9F	Cent cinquante-neuf
AØ	Cent soixante
A1	Cent soixante-et-un
:	:
:	:
FE	Deux cent cinquante-quatre
FF	Deux cent cinquante-cinq
1ØØ	Deux cent cinquante-six

Lorsque vous utilisez la notation hexadécimale et que vous voulez préciser ce fait de façon claire, vous devez écrire "h" à la fin de chaque valeur numérique et dire "hex". Par exemple, pour cent cinquante huit, vous devez écrire "9Eh" et dire "neuf E hex".

Mais vous vous demandez peut-être qu'est-ce que tout ce qui précède a à voir avec les ordinateurs. Voilà : les ordinateurs se comportent comme s'ils n'avaient que deux doigts représentés respectivement par une basse tension (hors = Ø) et par une haute tension (en = 1). Ce système s'appelle le système binaire et les deux chiffres binaires sont appelés bits : ainsi, un bit est soit à la valeur Ø soit à la valeur 1.

Dans les différents systèmes, le comptage commence comme suit

Français	*Décimal*	*Hexadécimal*	*Binaire*
Zéro	0	0	0 or 0000
Un	1	1	1 or 0001
Deux	2	2	10 or 0010
Trois	3	3	11 or 0011
Quatre	4	4	100 or 0100
Cinq	5	5	101 or 0101
Six	6	6	110 or 0110
Sept	7	7	111 or 0111
Huit	8	8	1000
Neuf	9	9	1001
Dix	10	A	1010
Onze	11	B	1011
Douze	12	C	1100
Treize	13	D	1101
Quatorze	14	E	1110
Quinze	15	F	1111
Seize	16	10	10000

Le point important est que seize est égal à deux élevé à la quatrième puissance, ce qui veut dire que les conversions entre l'hexadécimal et le binaire deviennent très simples.

Pour convertir une valeur hexadécimale en binaire, vous devez transformer chaque chiffre hexadécimal en quatre bits à l'aide de la table précédente. Pour passer du binaire à l'hexadécimal, divisez le nombre binaire en groupes de quatre bits en commençant par la droite et transformez ensuite chaque groupe pour en faire le chiffre hexadécimal correspondant.

Pour cette raison, bien qu'à strictement parler, les ordinateurs utilisent un système binaire pur, les humains écrivent souvent en notation hexadécimal les nombres destinés à l'ordinateur.

De l'ordinateur, les bits sont les plus souvent groupés en série de huit, chaque série étant appelée octet. Un octet peut représenter un nombre quelconque entre zéro et deux cent cinquante-cinq (11111111 en binaire ou FF en hexadécimal) ou, bien entendu, un caractère quelconque du jeu de caractères du ZX81. Sa valeur peut être écrite avec deux chiffres hexadécimaux.

Deux octets peuvent être associés pour constituer ce que nous appelons techniquement un mot. Un mot peut être écrit avec 16 bits ou quatre chiffres hexadécimaux, il représente un nombre entre Ø et (en décimal) $2^{16-1} = 65535$.

Un octet a toujours huit bits, mais les mots varient d'un ordinateur à l'autre.

Résumé

Systèmes décimal, hexadécimal et binaire
Bits et octets (ne les confondez pas) et mots.

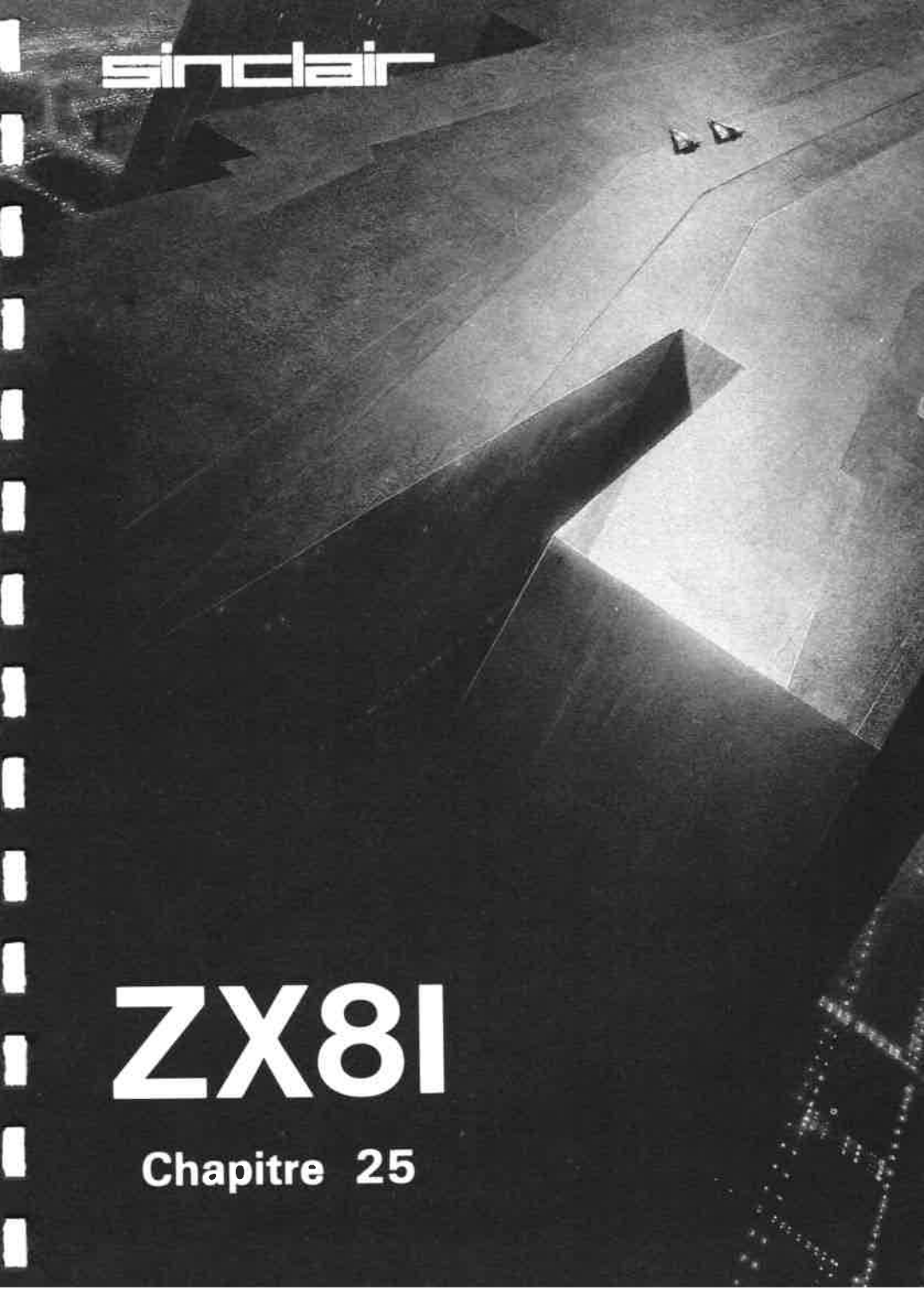
Exercices

1. La devise martienne est la livre que vous divisez en seize onces. Comment feriez-vous pour effectuer une conversion afin de passer de livres et onces à des onces et vice-versa.
   (i) Lorsque tous les nombres sont écrits en système décimal ?
   (ii) Lorsque tous les nombres sont écrits en système hexadécimal ?

2. Comment feriez-vous pour convertir des nombres décimaux en hexadécimaux et vice versa ? (Une indication : l'exercice 1).

   Ecrivez des programmes assurant la conversion de valeurs numériques pour qu'elles deviennent des chaînes donnant leur représentation hexadécimale et vice versa (c'est ce que font **STR$** et **VAL** avec les représentations décimales).

3. Supposons que les Vénusiens aient 8 doigts au total, sans pouces, quelle utilité peut avoir leur système octal (base 8) dans le cas des ordinateurs ?



ZX81

Chapitre 25

Comment fonctionne l'ordinateur

L'illustration suivante montre l'intérieur du ZX81 (ne la séparez pas du reste du manuel, car il serait très difficile de la remettre ensuite dans cet ouvrage).

Comme vous pouvez le voir, chaque élément comporte une abréviation composée des lettres TLA.

Les petits bouts de plastique montés sur de nombreuses pattes métalliques sont les merveilleuses puces de silicium qui ont rendu possible les montres digitales et le Sinclair ZX81. Chaque morceau de matière plastique contient un bout de silicium dont la taille est voisine de celle du curseur K et cet élément de silicium est connecté aux pattes métalliques par des conducteurs.

Le cerveau qui assure l'exécution des opérations est la puce microprocesseur, souvent appelée UC (unité centrale). Celui que vous observez est un processeur Z80 (en réalité un Z80A qui est plus rapide que le précédent).

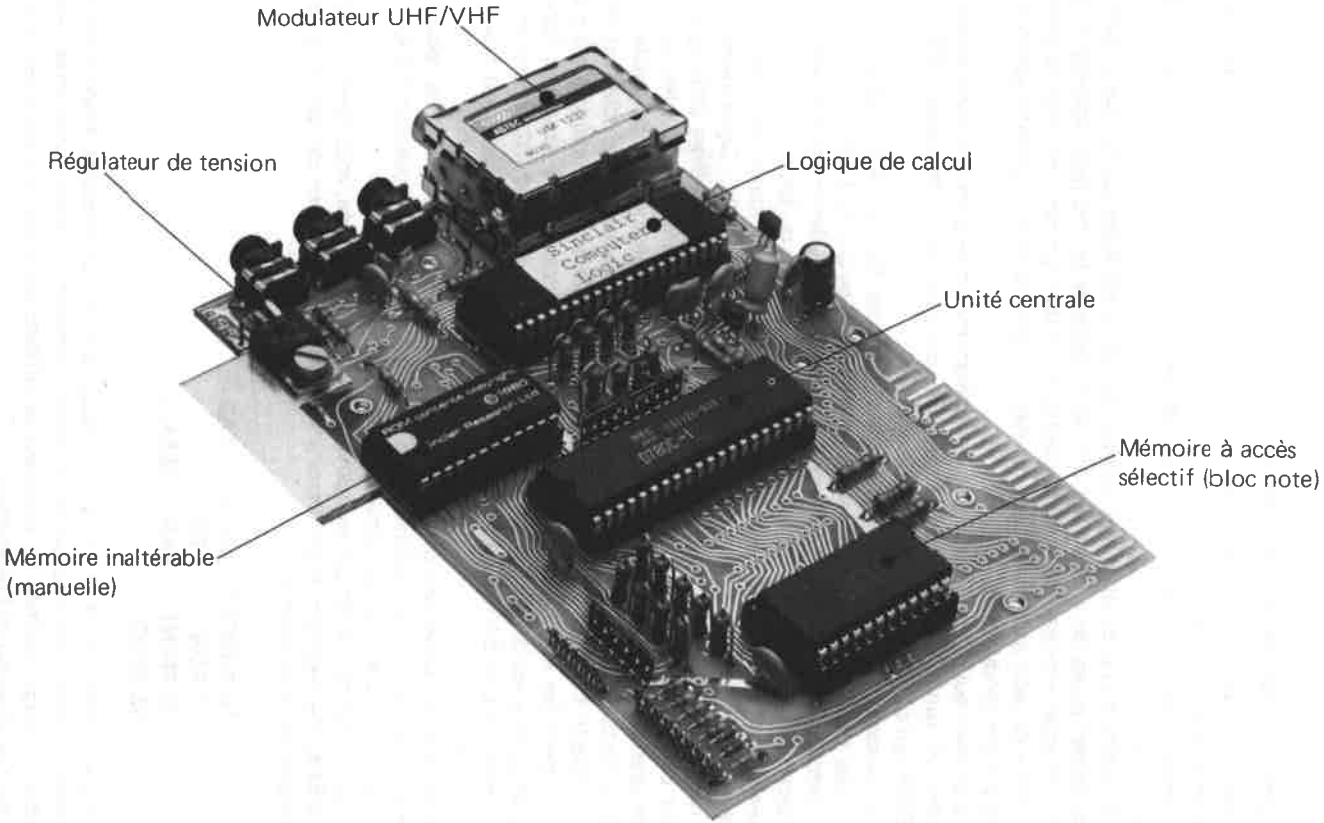
Le processeur pilote tout l'ordinateur, exécute les opérations arithmétiques, est averti des touches que vous manœuvrez, décide de ce qui doit être fait en conséquence et coordonne l'image sur le téléviseur. Bien qu'il soit extrêmement intelligent, il ne peut pas faire tout ceci seul. Laissé à lui-même, il ne sait rien du BASIC, de l'arithmétique en virgule flottante ou des téléviseurs, il doit donc recevoir toutes ses instructions d'une autre puce — la mémoire morte ou mémoire inaltérable (ROM). Cette mémoire contient une longue liste de toutes les instructions qui permettent de réaliser un programme ordinateur complet pour dire au processeur ce qu'il doit faire dans tous les cas prévisibles. Ce programme n'est pas écrit en langage BASIC, il est rédigé en code machine Z80, qui revêt la forme d'une longue série d'octets. (Souvenez-vous qu'un octet est un nombre de l'intervalle Ø-255). Chaque octet est associé à une adresse indiquant où il se trouve dans la mémoire inaltérable ; le premier octet a l'adresse Ø, le deuxième l'adresse 1 et ainsi de suite jusqu'à 8191. En tout, nous avons donc 8192=8*1024 octets et c'est pour cette raison que le BASIC ZX81 est parfois appelé BASIC 8K. Un K est égal à 1Ø24 ou 2^{10} ou 400 h.

Bien que les puces des différentes machines soient similaires, cette séquence particulière d'instructions est unique au ZX81, elle a été écrite spécialement à son intention par une petite société de mathématiciens de Cambridge.

Pour connaître l'octet qui se trouve à une adresse donnée, vous devez utiliser la fonction **PEEK**. Par exemple, le programme suivant écrit les 21 premiers octets de la mémoire inaltérable (et leurs adresses).

```
1Ø PRINT "ADRESSE";TAB 8;"BYTE"
2Ø FOR A = Ø TO 2Ø
3Ø PRINT A; TAB 8; PEEK A
4Ø NEXT A
```

La puce suivante est la mémoire à accès sélectif ou mémoire RAM. C'est là que le processeur range l'information qu'il désire conserver — votre programme BASIC, les variables, l'image destinée au téléviseur et différentes notes (les variables système) qui précisent la situation de l'ordinateur.


Modulateur UHF/VHF
Régulateur de tension
Logique de calcul
Unité centrale
Mémoire à accès sélectif (bloc note)
Mémoire inaltérable (manuelle)

Comme la mémoire inaltérable, la mémoire RAM est organisée en octets qui ont chacun une adresse : les adresses sont comprises dans l'intervalle 16384-17407 (et parfois même jusqu'à 32767 — si votre machine est équipée du module de mémoire 16K). Comme dans le cas de la mémoire inaltérable, vous pouvez connaître les valeurs des octets en utilisant **PEEK**, mais la différence importante par rapport à la mémoire inaltérable est que vous pouvez également modifier les valeurs. (Bien entendu, la mémoire inaltérable est inaltérable comme le dit son nom.)
Entrez

POKE 17ØØ,57

Cette opération a pour effet que l'octet qui est à l'adresse 17ØØ a la valeur 57. Si vous entrez maintenant

PRINT PEEK 17ØØ

la machine vous renvoie le nombre 57. (Tentez la même opération avec d'autres valeurs pour vous assurer que cette méthode est bien générale.)

Notez que l'adresse doit être dans l'intervalle Ø - 65635 ; la plupart des adresses citent des octets de la mémoire inaltérable ou qui n'existent pas et sont donc sans effet. La valeur doit être comprise dans l'intervalle —255/+255 et, si elle est négative, la valeur 256 lui est ajoutée.

La possibilité d'établir des valeurs vous permet de manipuler l'ordinateur si vous savez bien tirer partie de cette fonction. Toutefois, les connaissances requises sont nettement au-delà de ce qui peut être décrit dans un manuel de présentation comme celui-ci.

La dernière puce c'est la puce logique — la logique de calcul. Cette puce a également été conçue et fabriquée spécialement pour le ZX81, elle est connectée aux autres puces d'une façon astucieuse afin que ces dernières fassent plus qu'elles ne feraient normalement.

Le modulateur transforme les signaux de l'ordinateur destinés au téléviseur pour leur faire revêtir une forme convenant à ce dernier et le régulateur convertit les 9V non régulés mais lissés de l'alimentation en 5V régulés.

Résumé

Les puces
Instructions : **POKE**
Fonctions : **PEEK**

ZX81

Chapitre 26

Pour utiliser le langage machine

Ce chapitre est écrit à l'intention de ceux qui comprennent le code ou langage machine Z80, le jeu d'instructions utilisé par le processeur Z80. Si vous ne faites pas partie de cette catégorie de lecteurs, mais que vous souhaiteriez être dans ce groupe, plusieurs ouvrages existent déjà sur ce sujet ; deux manuels de présentation ont déjà été publiés en anglais ('Programming the Z80' écrit par Rodnay Zaks et publié par Sybex et "Z8Ø and 8Ø8Ø Assembly Language programming" de Rathe Sparcklen, publié par Hyden).

Le nec plus ultra est composé de deux ouvrages : "Z8Ø Assembly Language Programming Manual" et "Z8Ø-CPU, Z8ØA-CPU Technical Manual" publiés par Zilog, mais il serait présomptueux de les recommander à des débutants.

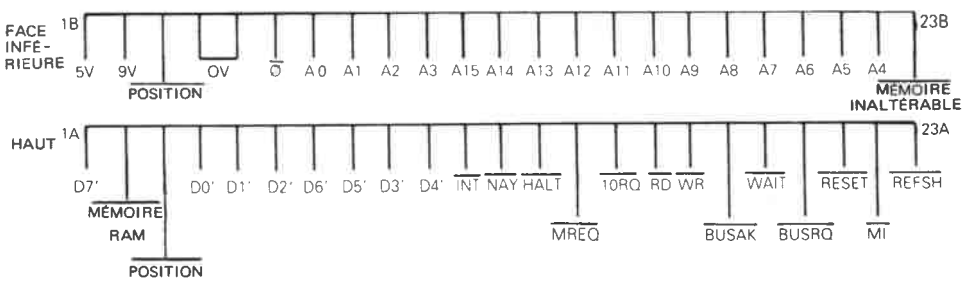
Les routines en langage machine peuvent être exécutées à partir d'un programme BASIC par la fonction **USR**. L'argument de la fonction **USR** est l'adresse de début de la routine, le résultat correspondant étant un entier sans signe composé de deux octets, c'est-à-dire le contenu de la paire de registres bc au retour. L'adresse de retour vers le BASIC est conservée de la façon habituelle et le retour se fait par une instruction du Z8Ø.

Il existe certaines limites pour les routines **USR** :

(i) Au retour, les registres iy et i doivent contenir les 4ØØØ h et 1 Eh.

(ii) La routine d'affichage utilise les registres a', f', ix et r et, de ce fait, une routine **USR** ne doit pas utiliser ces registres si les fonctions de calcul et d'affichage opèrent simultanément (il n'est même pas recommandé de lire les registres af').

Les bus de commande, des données et des adresses sont tous visibles à l'arrière de ZX81 et, par conséquent, vous pouvez avec le ZX81 faire à peu près tout ce que vous pouvez faire avec le Z8Ø. Il se peut néanmoins que le matériel du ZX81 vous gêne dans certains cas, en particulier pour le calcul et l'affichage simultané. Voici un schéma des connexions visibles à l'arrière de la machine.



Un élément en langage machine qui se trouve au milieu de la mémoire risque d'être surchargé par le système BASIC. Les endroits les plus sûrs sont :

(i) Dans une instruction **REM** : Entrez une instruction **REM** composée d'assez de caractères pour contenir votre langage machine que vous pouvez alors examiner. Cette partie doit devenir la première ligne du programme pour qu'elle ne soit pas déplacée. Eviter les instructions d'arrêt qui seront reconnues comme la fin de l'instruction **REM**.

(ii) Dans une chaîne : Etablissez une chaîne assez longue et affectez ensuite un octet de code machine à chaque caractère. Les chaînes peuvent toujours se déplacer dans la mémoire.

Dans l'annexe A consacrée au jeu de caractères, vous trouverez les caractères et les instructions du Z80 écrits côte à côte en ordre pour vous faciliter le travail lorsque vous entrez du code.

(iii) En haut de la mémoire. Lorsque le ZX81 est mis sous tension, il procède au contrôle de la quantité de mémoire présente et place ensuite la pile machine en haut de la mémoire ; il n'y a donc pas d'espace à cet endroit pour les routines **USR**. La machine met l'adresse du premier octet non-existant (par exemple 17K ou 17Ø8 avec 1K de mémoire) dans une variable système appelée RAMTOP, dans les deux octets dont les adresses sont 16388 et 16389. Par contre, **NEW** n'effectue pas un contrôle mémoire complet ; la vérification porte seulement sur ce qui précède l'adresse dans RAMTOP. Ainsi, si vous examinez l'adresse d'un octet existant dans RAMTOP, pour **NEW** toute la mémoire à partir de cet octet est en dehors du système BASIC et est donc laissée tranquille. Par exemple, supposons que vous ayez une mémoire de 1K et que vous veniez de mettre l'ordinateur sous tension :

PRINT PEEK 16388+256***PEEK** 16389

vous indique l'adresse (17Ø8) du premier octet non-existant.

Supposons maintenant que vous disposiez d'une routine **USR** de longueur 2Ø octets. Vous voulez porter RAMTOP à 17388 = 236 + 256*67 (comment réaliserez-vous cette opération sur la machine ?) ; entrez

POKE 16388,236
POKE 16389,67

et ensuite **NEW**. Maintenant, les vingt octets de mémoire compris entre l'adresse 17388 et l'adresse 17Ø7 vous sont réservés, vous pouvez en faire ce que vous voulez. Si vous entrez à nouveau **NEW**, les 20 octets ci-dessus ne sont pas affectés.

Le haut de la mémoire est un bon endroit pour les routines **USR**, un endroit sûr (même vis-à-vis de **NEW**) et statique. Son principal inconvénient est que le contenu de cette partie n'est pas sauvegardé par **SAVE**.

Résumé

Fonctions : **USR**
Instructions : **NEW**

Exercices

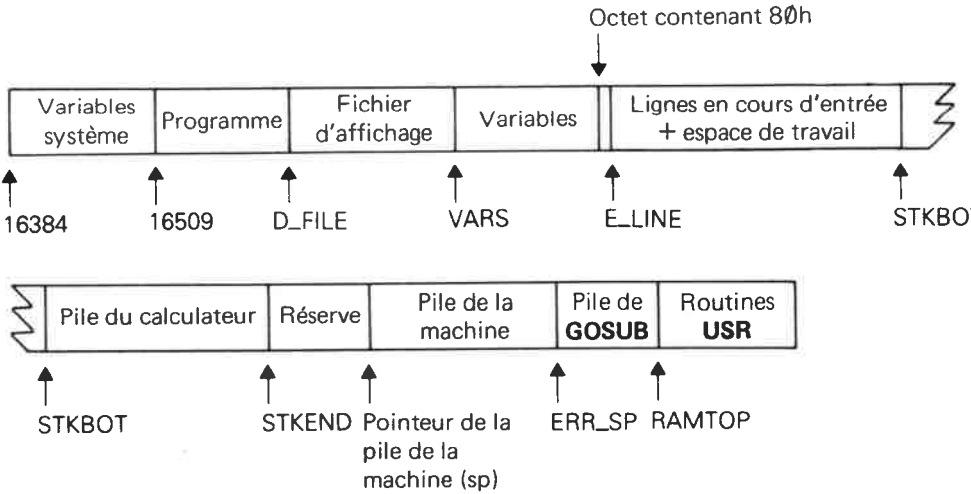
1. Rendez RAMTOP égale à 167ØØ et exécutez ensuite **NEW**. Vous aurez ainsi un aperçu de ce qui se passe lorsque la mémoire se remplit complètement.

ZX81

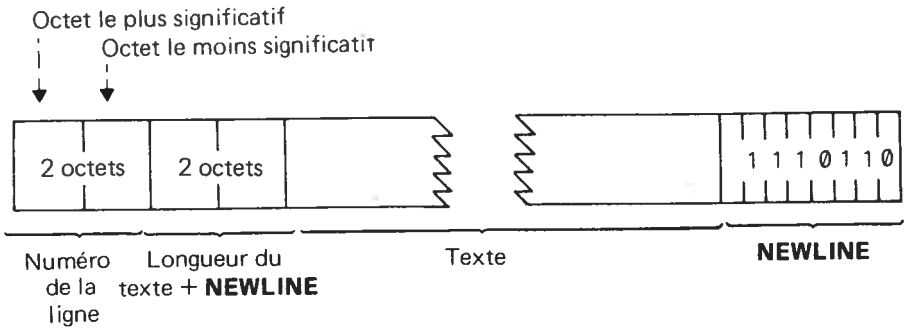
Chapitre 27

Organisation de la mémoire

La mémoire est subdivisée en plusieurs zones qui permettent de stocker différents types d'informations. La taille des zones est calculée pour concorder exactement avec l'information qu'elles contiennent réellement et, si à un point donné, vous en insérez davantage (par exemple en ajoutant une variable ou une ligne de programme) de l'espace est rendu disponible en décalant tout ce qui se trouve au-dessus de ce point. En revanche, si vous supprimez les informations, tout ce qui se trouve au-dessus est amené vers le bas.



Les variables système contiennent différentes informations dont le rôle est de dire à l'ordinateur dans quel état il est. Ces variables sont reprises exhaustivement au prochain chapitre, mais pour le moment, il vous suffit de noter qu'il y en a certaines (appelées D–FILE, VARS, E–LINE etc.) qui contiennent les adresses des limites entre les différentes zones dans la mémoire. Ces variables ne sont pas des variables BASIC, leurs noms ne sont pas reconnus par la machine.



Noter que, contrairement à tous les autres cas de nombres à deux octets dans le Z8Ø, le numéro de ligne (ceci s'applique également aux variables de contrôle **FOR-NEXT**), est stocké dans l'ordre octet le plus significatif/octet le moins significatif : en d'autres termes, dans l'ordre suivant lequel vous les écririez.

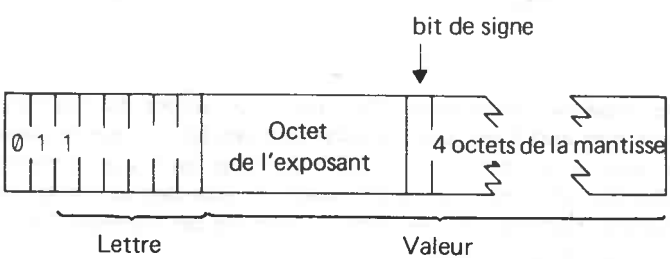
Dans le programme, une constante numérique est suivie de sa forme binaire avec le caractère **CHR$** 126 suivi de cinq octets pour le nombre lui-même.

Le fichier d'affichage est la copie en mémoire de l'image présentée sur le téléviseur. Ce fichier commence par un caractère **NEWLINE** et comporte ensuite 24 lignes de texte se terminant chacune par **NEWLINE**. Le système est conçu pour qu'une ligne de texte n'ait pas besoin de tout l'espace requis pour mettre 32 caractères : les espaces de fin peuvent être omis. Ceci permet d'économiser de la mémoire lorsque la quantité de mémoire disponible est faible.

Lorsque la mémoire totale du système (conformément à la variable système RAMTOP) est inférieure à 3 1/4K, un écran vide — tel qu'il a été établi au début ou par **CLS** — est composé de 25 **NEWLINE** seulement. Lorsque la mémoire est plus grande, un écran vide est garni de 24*32 espaces et, dans l'ensemble, il reste à sa taille maximale ; toutefois, **SCROLL** et d'autres conditions spécifiques provoquant l'agrandissement de la partie inférieure de l'écran à plus de deux lignes peuvent perturber ce qui précède par l'introduction de lignes courtes en bas.

Les variables ont deux formats différents en fonction de leur nature.

Nombre dont le nom est composé d'une seule lettre :



Nombre dont le nom comporte plus d'une lettre :

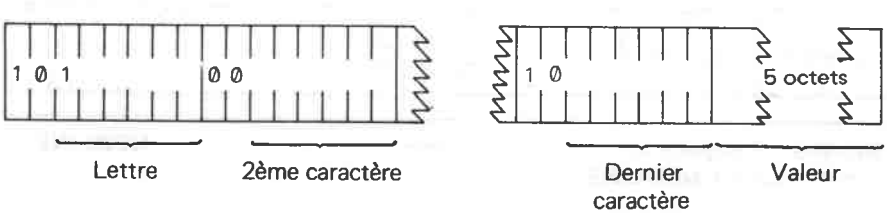
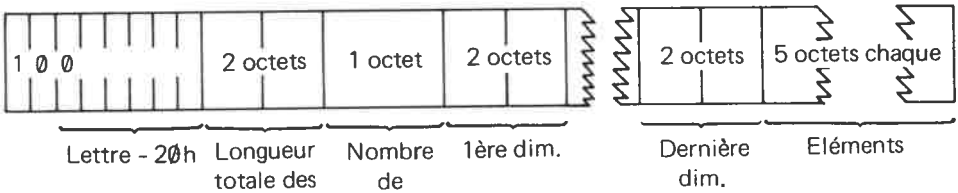


Tableau des nombres



L'ordre des éléments est le suivant :

d'abord les éléments pour lesquels le premier indice est 1
ensuite, les éléments pour lesquels le premier indice est 2
les éléments pour lesquels le premier indice est 3

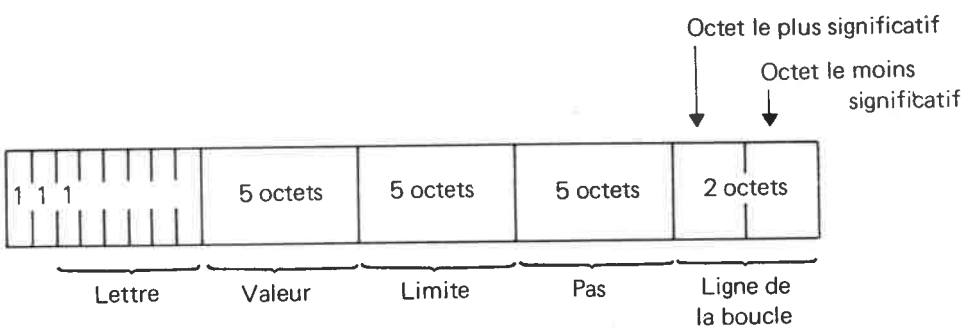
et ainsi de suite pour toutes les valeurs possibles du premier indice.

Les éléments accompagnés d'un premier indice donné sont classés de la même façon avec le deuxième indice, etc. jusqu'à la fin.

Exemple : les éléments du tableau B 3 x 6 du chapitre 22 sont rangés dans l'ordre suivant :

B (1,1), B (1,2),... B (1,6), B (2,1), B (2,2),... B (2,6), B (3,1), B (3,2),... (3,6)

Variable de contrôle d'une boucle **FOR-NEXT** :



Chaîne :

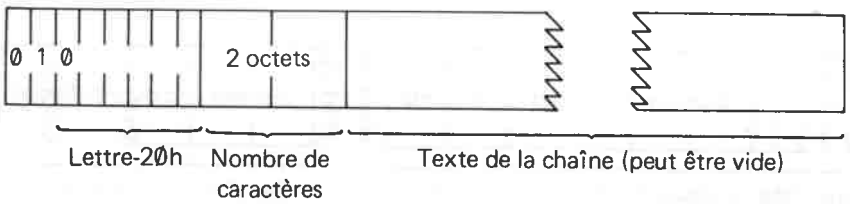
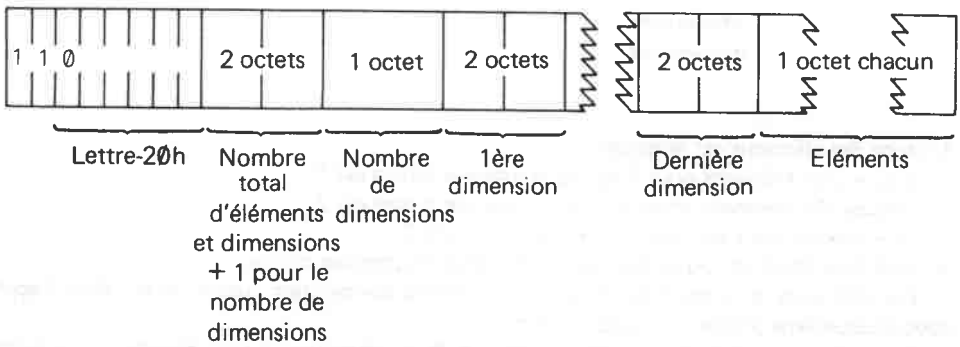


Tableau de caractères :



La partie qui commence à E—LINE contient la ligne en cours d'entrée (sous forme d'une commande, d'une ligne de programme ou **INPUT**) ainsi qu'une certaine quantité d'espace de travail.

Le calculateur est la partie du système BASIC traitant de l'arithmétique et les nombres sur lesquels il travaille sont pour l'essentiel dans la pile du calculateur.

La partie réserve contient l'espace demeuré inutilisé à cet instant.

La pile de la machine est celle utilisée par la puce Z8Ø pour conserver les adresses de retour, etc.

La pile **GOSUB** a été mentionnée au chapitre 14.

L'espace que doivent occuper les routines **USR** doit être mis de côté par vous-même à l'aide de **NEW** comme il a été dit au chapitre précédent.

ZX81

Chapitre 28

Chapitre 28

Les variables système

Les octets de l'intervalle 16384 - 16508 de la mémoire sont mis de côté pour emploi spécial par le système. Vous pouvez les examiner pour découvrir différentes choses concernant le système et certaines d'entre elles peuvent être modifiées utilement.

Elles sont listées ci-dessous avec leur utilisation.

Appelées variables système, ces variables sont désignées par des noms, mais il ne faut pas les confondre avec les variables utilisées par le BASIC. L'ordinateur ne reconnaît pas les noms comme désignant les variables système, ces noms sont donnés seulement comme mnémoniques à l'usage des êtres humains.

Les abréviations de la colonne 1 signifient :

X La variable ne doit pas être modifiée car cette opération risquerait de bloquer le système.
N La modification de cette variable n'aura pas d'effet durable.
S Cette variable peut être sauvegardée par **SAVE**.

Le nombre inscrit dans la colonne 1 est le nombre d'octets de la variable. Pour deux octets, le premier est le moins significatif, c'est-à-dire l'inverse de ce que vous attendiez sans doute. Ainsi, pour transformer une valeur x en une variable à deux octets, à l'adresse n, il faut utiliser

POKE n,v−256***INT** (v/256)
POKE n+1,**INT** (v/256)

et pour connaître sa valeur, il faut utiliser

PEEK n + 256***PEEK** (n+1)

Notes	*Adresse*	*Nom*	*Contenu*
1	16384	ERR−NR	1 moins le code de compte-rendu. Commence à 255 (pour −1) et, par conséquent, si **PEEK** 16384 donne un résultat, ce résultat est 255. **POKE** 16384,n peut servir pour forcer un arrêt sur erreur : Ø ⩽ n ⩽ 14 donne l'un des comptes-rendus habituels, 15 ⩽ n ⩽ 34 ou 99 ⩽ n ⩽ 127 donne un compte-rendu non standard et 35 ⩽ n ⩽ 98 risque fortement de perturber le fichier d'affichage.
X1	16385	FLAGS	Les différents drapeaux qui contrôlent le système BASIC.
X2	16386	ERR−SP	Adresse de la première rubrique de la pile machine (après les retours de **GOSUB**).

Notes	*Adresse*	*Nom*	*Contenu*
2	16388	RAMTOP	Adresse du premier octet au-dessus de la zone du système BASIC. Vous pouvez effectuer une POKE pour obtenir du nouvel (**NEW**) espace de réserve au-dessus de cette zone (voir le chapitre 26) ou pour que **CLS** établisse un fichier d'affichage minimal (chapitre 27). Une fonction POKE sur RAMTOP est sans effet tant que l'une des deux n'a pas été exécutée.
N1	16390	MODE	Curseur K, L, F ou G spécifié.
N2	16391	PPC	Numéro de la ligne de l'instruction en cours d'exécution. POKE est sans effet durable, sauf dans la dernière ligne du programme.
S1	16393	VERSN	Ø identifie le BASIC ZX81 dans les programmes sauvegardés.
S2	16394	E–PPC	Numéro de la ligne en cours, avec le curseur du programme.
SX2	16396	D–FILE	Voir le chapitre 27.
S2	16398	DF–CC	Adresse de la position d'écriture **PRINT** dans le fichier d'affichage. Peut être modifiée pour que la sortie à écrire soit transmise ailleurs.
SX2	16400	VARS	Voir le chapitre 27.
SN2	16402	DEST	Adresse de la variable en affectation.
SX2	16404	E–LINE	Voir le chapitre 27.
SX2	16406	CH–ADD	Adresse du caractère suivant à interpréter. Le caractère après l'argument de **PEEK** ou **NEWLINE** à la fin d'une instruction **POKE**.
S2	16408	X–PTR	L'adresse du caractère précédant le marqueur **S**.
SX2	16410	STKBOT	Voir le chapitre 27.
SX2	16412	STKEND	Voir le chapitre 27.
SN1	16414	BERG	Registre b du calculateur.
SN2	16415	MEM	Adresse de la zone utilisée comme mémoire du calculateur (habituellement MEMBOT, mais pas toujours).
S1	16417	Inutilisée	
SX1	16418	DF–SZ	Nombre de lignes (y compris une ligne vide) de la partie inférieure de l'écran.
S2	16419	S–TOP	Le numéro de la plus haute ligne du programme dans une liste automatique.
SN2	16421	LAST–K	Précise les touches qui ont été manœuvrées.
SN1	16423		Etat d'anti-rebond du clavier.

Notes	*Adresse*	*Nom*	*Contenu*
SN1	16424	MARGIN	Nombre de lignes vides en-dessus ou en-dessous de l'image : 55 en Grande Bretagne, 31 aux Etats-Unis.
SX2	16425	NXTLIN	Adresse de la prochaine ligne de programme à exécuter.
S2	16427	OLDPPC	Numéro de la ligne à partir de laquelle **CONT** a effectué son saut.
SN1	16429	FLAGX	Différents drapeaux.
SN2	16430	STRLEN	Longueur de la destination du type de chaîne en affectation.
SN2	16432	T–ADDR	Adresse de la rubrique suivante dans la table de syntaxe (généralement inutile).
S2	16434	SEED	Base de **RND**. Cette variable est établie par **RAND**.
S2	16436	FRAMES	Compte les trames affichées sur le téléviseur. Le bit 15 est à 1. Les bits Ø à 14 sont décrémentés à chaque trame transmise au téléviseur. Cette particularité peut être utilisée pour la synchronisation mais **PAUSE** en fait également usage. **PAUSE** remet le bit 15 à Ø et met la durée de la pause dans les bits Ø à 14. Lorsque ces bits ont été ramenés à Ø, la pause est terminée. Si la pause est arrêtée par la manœuvre d'une touche, le bit 15 est à nouveau mis à 1.
S1	16438	COORDS	Coordonnée x du dernier point tracé **PLOT**.
S1	16439		Coordonnée y du dernier point tracé **PLOT**.
S1	1644Ø	PR–CC	Octet le moins significatif de l'adresse de la position suivante à laquelle **LPRINT** doit écrire (dans PRBUFF).
SX1	16441	S–POSN	Numéro de colonne de la position de **PRINT**.
SX1	16442		Numéro de ligne de la position de **PRINT**.
S1	16443	CDFLAG	Différents drapeaux. Le bit 7 est en (1) en mode calcul et affichage.
S33	16444	PRBUFF	Tampon de l'imprimante (le 33ème caractère est **NEWLINE**).
SN30	16477	MEMBOT	Zone de mémoire du calculateur ; sert pour ranger des nombres qui ne sont pas pratiques à mettre dans la pile du calculateur.
S2	165Ø7	Inutilisée	

Exercices

1. Essayez ce programme

```
1Ø FOR N=Ø TO 21
2Ø PRINT PEEK (PEEK 164ØØ+256*PEEK 164Ø1+N)
3Ø NEXT N
```

Ce programme vous indique les 22 premiers octets de la zone des variables : essayez d'adapter la variable de contrôle N à la description du chapitre 27.

2. Dans le programme ci-dessus, transformez la ligne 2Ø en

```
2Ø PRINT PEEK (165Ø9+N)
```

Le nouveau programme vous indique les 22 premiers octets de la zone programme. Faites-les concorder avec le programme à proprement parler.

Le jeu de caractères

Le jeu de caractères ci-dessous est le jeu complet du ZX80 avec ses codes en décimal et en hexadécimal. Si vous voulez bien considérer que les codes sont les instructions en langage machine Z80, la colonne de droite vous donne les mnémoniques correspondant en langage d'assemblage. Comme vous le savez sans doute si vous comprenez ces choses, certaines instructions du Z80 sont composées avec le suffixe CBh ou EDh ; ces instructions complexes sont reprises dans les deux colonnes de droite.

Code	*Caractère*	*Hex*	*Assembleur Z80*	*— après CBh*
0	espace	00	nop	rlc b
1		01	ld bc,NN	rlc c
2		02	ld (bc),a	rlc d
3		03	inc bc	rlc e
4		04	inc b	rlc h
5		05	dec b	rlc l
6		06	ld b,N	rlc (hl)
7		07	rlca	rlc a
8		08	ex af,af'	rrc b
9		09	add hl,bc	rrc c
10		0A	ld a,(bc)	rrc d
11	"	0B	dec bc	rrc e
12	£	0C	inc c	rrc h
13	$	0D	dec c	rrc l
14	:	0E	ld, c,N	rrc (hl)
15	?	0F	rrca	rrc a
16	(	10	djnz DIS	rl b
17	)	11	ld de,NN	rl c
18	>	12	ld (de),a	rl d
19	<	13	inc de	rl e
20	=	14	inc d	rl h
21	+	15	dec d	rl l
22	–	16	ld d,N	rl (hl)
23	*	17	rla	rl a
24	/	18	jr DIS	rr b
25	;	19	add hl,de	rr c
26	,	1A	ld a,(de)	rr d
27	.	1B	dec de	rr e
28	0	1C	inc e	rr h
29	1	1D	dec e	rr l
30	2	1E	ld e,N	rr (hl)
31	3	1F	rra	rr a
32	4	20	jr nz,DIS	sla b
33	5	21	ld hl,NN	sla c
34	6	22	ld (NN),hl	sla d

Code	*Caractère*	*Hex*	*Assembleur Z80*	*— après CBh*	*— après EDh*
35	7	23	inc hl	sla e	
36	8	24	inc h	sla h	
37	9	25	dec h	sla l	
38	A	26	ld, h,N	sla (hl)	
39	B	27	daa	sla a	
40	C	28	jr z,DIS	sra b	
41	D	29	add hl,hl	sra c	
42	E	2A	ld hl,(NN)	sra d	
43	F	2B	dec hl	sra e	
44	G	2C	inc l	sra h	
45	H	2D	dec l	sra l	
46	I	2E	ld,l,N	sra (hl)	
47	J	2F	cpl	sra a	
48	K	30	jr nc,DIS		
49	L	31	ld sp,NN		
50	M	32	ld (NN),a		
51	N	33	inc sp		
52	O	34	inc (hl)		
53	P	35	dec (hl)		
54	Q	36	ld (hl),N		
55	R	37	scf		
56	S	38	jr c,DIS	srl b	
57	T	39	add hl,sp	srl c	
58	U	3A	ld a,(NN)	srl d	
59	V	3B	dec sp	srl e	
60	W	3C	inc a	srl h	
61	X	3D	dec a	srl l	
62	Y	3E	ld a,N	srl (hl)	
63	Z	3F	ccf	srl a	
64	**RND**	40	ld b,b	bit 0,b	in b,(c)
65	**INKEY$**	41	ld b,c	bit o,c	out (c),b
66	**PI**	42	ld b,d	bit 0,d	sbc hl,bc
67		43	ld b,e	bit 0,e	ld (NN),bc
68		44	ld b,h	bit 0,h	neg
69		45	ld b,l	bit 0,l	retn
70		46	ld b,(hl)	bit 0,(hl)	im 0
71		47	ld b,a	bit 0,a	ld i,a
72	inutilisés	48	ld c,b	bit 1,b	in c,(c)
73		49	ld c,c	bit 1,c	out (c),c
74		4A	ld c,d	bit 1,d	adc hl,bc
75		4B	ld c,e	bit 1,e	ld bc,(NN)
76		4C	ld c,h	bit 1,h	

Code	*Caractère*	*Hex*	*Assembleur Z80*	*– après CBh*	*– après EDh*
77		4D	ld c,l	bit 1,l	reti
78		4E	ld c,(hl)	bit 1,(hl)	
79		4F	ld c,a	bit 1,a	ld r, a
80		50	ld d,b	bit 2,b	in d,(c)
81		51	ld d,c	bit 2,c	out (c),d
82		52	ld d,d	bit 2,d	sbc hl,de
83		53	ld d,e	bit 2,e	ld (NN),de
84		54	ld d,h	bit 2,h	
85		55	ld d,l	bit 2,l	
86		56	ld d,(hl)	bit 2,(hl)	im 1
87		57	ld d,a	bit 2,a	ld a,i
88		58	ld e,b	bit 3,b	in e,(c)
89		59	ld e,c	bit 3,c	out (c),e
90		5A	ld e,d	bit 3,d	adc hl,de
91		5B	ld e,e	bit 3,e	ld de,(NN)
92		5C	ld e,h	bit 3,h	
93		5D	ld e,l	bit 3,l	
94	inutilisés	5E	ld e,(hl)	bit 3,(hl)	im 2
95		5F	ld e,a	bit 3,a	ld a, r
96		60	ld h,b	bit 4,b	in h,(c)
97		61	ld h,c	bit 4,c	out (c),h
98		62	ld h,d	bit 4,d	sbc hl,hl
99		63	ld h,e	bit 4,e	ld (NN),hl
100		64	ld h,h	bit 4,h	
101		65	ld h,l	bit 4,l	
102		66	ld h,(hl)	bit 4,(hl)	
103		67	ld h,a	bit 4,a	rrd
104		68	ld l,b	bit 5,b	in l,(c)
105		69	ld l,c	bit 5,c	out (c),l
106		6A	ld l,d	bit 5,d	adc hl,hl
107		6B	ld l,e	bit 5,e	ld de,(NN)
108		6C	ld l,h	bit 5,h	
109		6D	ld l,l	bit 5,l	
110		6E	ld l,(hl)	bit 5,(hl)	
111		6F	ld l,a	bit 5,a	rld
112	montée curseur ⇧	70	ld (hl),b	bit 6,b	
113	descente curseur ⇩	71	ld (hl),c	bit 6,c	
114	curs. vers la gauche ⇦	72	ld (hl),d	bit 6,d	sbc hl,sp
115	curs. vers la droite ⇨	73	ld (hl),e	bit 6,e	ld (NN),sp
116	**GRAPHICS**	74	ld (hl),h	bit 6,h	
117	**EDIT**	75	ld (hl),l	bit 6,l	
118	**NEWLINE**	76	halt	bit 6,(hl)	

Code	*Caractère*	*Hex*	*Assembleur Z80*	*— après CBh*	*— après EDh*
119	**RUBOUT**	77	ld (hl),a	bit 6,a	
120	mode ⊠ \ ⊡	78	ld a,b	bit 7,b	in a,(c)
121	**FUNCTION**	79	ld a,c	bit 7,c	out (c),a
122	inutilisé	7A	ld a,d	bit 7,d	adc hl,sp
123	inutilisé	7B	ld a,e	bit 7,e	ld sp,(NN)
124	inutilisé	7C	ld a,h	bit 7,h	
125	inutilisé	7D	ld a,l	bit 7,l	
126	nombre	7E	ld a,(hl)	bit 7,(hl)	
127	curseur	7F	ld a,a	bit 7,a	
128	■	80	add a,b	res 0,b	
129	▟	81	add a,c	res 0,c	
130	▙	82	add a,d	res 0,d	
131	▄	83	add a,e	res 0,e	
132	▗	84	add a,h	res 0,h	
133	▐	85	add a,l	res 0,l	
134	▚	86	add a,(hl)	res 0,(hl)	
135	▖	87	add a,a	res 0,a	
136	▒	88	adc a,b	res 1,b	
137	▓	89	adc a,c	res 1,c	
138	▓	8A	adc a,d	res 1,d	
139	" inversé	8B	adc a,e	res 1,e	
140	£ inversé	8C	adc a,h	res 1,h	
141	$ inversé	8D	adc a,l	res 1,l	
142	: inversé	8E	adc a,(hl)	res 1,(hl)	
143	? inversé	8F	adc a,a	res 1,a	
144	(inversé	90	sub b	res 2,b	
145	) inversé	91	sub c	res 2,c	
146	> inversé	92	sub d	res 2,d	
147	< inversé	93	sub e	res 2,e	
148	= inversé	94	sub h	res 2,h	
149	+ inversé	95	sub l	res 2,1	
150	— inversé	96	sub (hl)	res 2,(hl)	
151	* inversé	97	sub a	res 2,a	
152	/ inversé	98	sbc a,b	res 3,b	
153	; inversé	99	sbc a,c	res 3,c	
154	, inversé	9A	sbc a,d	res 3,d	
155	. inversé	9B	sbc a,e	res 3,e	
156	0 inversé	9C	sbc a,h	res 3,h	
157	1 inversé	9D	sbc a,l	res 3,l	
158	2 inversé	9E	sbc a,(hl)	res 3,(hl)	
159	3 inversé	9F	sbc a,a	res 3,a	
160	4 inversé	A0	and b	res 4,b	ldi

Code	*Caractère*	*Hex*	*Assembleur Z80*	*– après CBh*	*– après EDh*
161	5 inversé	A1	and c	res 4,c	cpi
162	6 inversé	A2	and d	res 4,d	ini
163	7 inversé	A3	and e	res 4,e	outi
164	8 inversé	A4	and h	res 4,h	
165	9 inversé	A5	and l	res 4,1	
166	A inversé	A6	and (hl)	res 4,(hl)	
167	B inversé	A7	and a	res 4,a	
168	C inversé	A8	xor b	res 5,b	ldd
169	D inversé	A9	xor c	res 5,c	cpd
170	E inversé	AA	xor d	res 5,d	ind
171	F inversé	AB	xor e	res 5,e	outd
172	G inversé	AC	xor h	res 5,h	
173	H inversé	AD	xor l	res 5,l	
174	I inversé	AE	xor (hl)	res 5,(hl)	
175	J inversé	AF	xor a	res 5,a	
176	K inversé	B0	or b	res 6,b	ldir
177	L inversé	B1	or c	res 6,c	cpir
178	M inversé	B2	or d	res 6,d	inir
179	N inversé	B3	or e	res 6,e	otir
180	O inversé	B4	or h	res 6,h	
181	P inversé	B5	or l	res 6,l	
182	Q inversé	B6	or (hl)	res 6,(hl)	
183	R inversé	B7	or a	res 6,a	
184	S inversé	B8	cp b	res 7,b	lddr
185	T inversé	B9	cp c	res 7,c	cpdr
186	U inversé	BA	cp d	res 7,d	indr
187	V inversé	BB	cp e	res 7,e	otdr
188	W inversé	BC	cp h	res 7,h	
189	X inversé	BD	cp l	res 7,l	
190	Y inversé	BE	cp (hl)	res 7,(hl)	
191	Z inversé	BF	cp a	res 7,a	
192	" "	C0	ret nz	set 0,b	
193	**AT**	C1	pop bc	set 0,c	
194	**TAB**	C2	jp nz,NN	set 0,d	
195	inutilisé	C3	jp NN	set 0,e	
196	**CODE**	C4	call nz,NN	set 0,h	
197	**VAL**	C5	push bc	set 0,l	
198	**LEN**	C6	add a,N	set 0,(hl)	
199	**SIN**	C7	rst 0	set 0,a	
200	**COS**	C8	ret z	set 1,b	
201	**TAN**	C9	ret	set 1,c	
202	**ASN**	CA	jp z,NN	set l,d	

Code	*Caractère*	*Hex*	*Assembleur Z80*	*— après CBh*
203	**ACS**	CB		set l,e
204	**ATN**	CC	call z,NN	set 1,h
205	**LN**	CD	call NN	set 1,l
206	**EXP**	CE	adc a,N	set 1,(hl)
207	**INT**	CF	rst 8	set 1,a
208	**SQR**	D0	ret nc	set 2,b
209	**SGN**	D1	pop de	set 2,c
210	**ABS**	D2	jp nc,NN	set 2,d
211	**PEEK**	D3	out N,a	set 2,e
212	**USR**	D4	call nc,NN	set 2,h
213	**STR$**	D5	push de	set 2,l
214	**CHR$**	D6	sub N	set 2,(hl)
215	**NOT**	D7	rst 16	set 2,a
216	**	D8	ret c	set 3,b
217	**OR**	D9	exx	set 3,c
218	**AND**	DA	jp c,NN	set 3,d
219	<=	DB	in a,N	set 3,e
220	>=	DC	call c,NN	set 3,h
221	<>	DD	préfixe les ins-	
			tructions avec ix	set 3,1
222	**THEN**	DE	sbc a,N	set 3,(hl)
223	**TO**	DF	rst 24	set 3,a
224	**STEP**	E0	ret po	set 4,b
225	**LPRINT**	E1	pop hl	set 4,c
226	**LLIST**	E2	jp po,NN	set 4,d
227	**STOP**	E3	ex (sp),hl	set 4,e
228	**SLOW**	E4	call po,NN	set 4,h
229	**FAST**	E5	push hl	set 4,l
230	**NEW**	E6	and N	set 4,(hl)
231	**SCROLL**	E7	rst 32	set 4,a
232	**CONT**	E8	ret pe	set 5,b
233	**DIM**	E9	jp (hl)	set 5,c
234	**REM**	EA	jp pe,NN	set 5,d
235	**FOR**	EB	ex de,hl	set 5,e
236	**GOTO**	EC	call pe,NN	set 5,h
237	**GOSUB**	ED		set 5,l
238	**INPUT**	EE	xor N	set 5,(hl)
239	**LOAD**	EF	rst 40	set 5,a
240	**LIST**	F0	ret p	set 6,b
241	**LET**	F1	pop af	set 6,c
242	**PAUSE**	F2	jp p,NN	set 6,d
243	**NEXT**	F3	di	set 6,e

Code	*Caractère*	*Hex*	*Assembleur Z80*	*— après CBh*
244	**POKE**	F4	call p,NN	set 6,h
245	**PRINT**	F5	push af	set 6,l
246	**PLOT**	F6	or N	set 6,(hl)
247	**RUN**	F7	rst 48	set 6,a
248	**SAVE**	F8	ret m	set 7,b
249	**RAND**	F9	ld sp,hl	set 7,c
250	**IF**	FA	jp m,NN	set 7,d
251	**CLS**	FB	ei	set 7,e
252	**UNPLOT**	FC	call m,NN	set 7,h
253	**CLEAR**	FD	préfixe les ins-turctions avec iy	set 7,l
254	**RETURN**	FE	cp N	set 7,(hl)
255	**COPY**	FF	rst 56	set 7,a

Codes des comptes-rendus

La table suivante est une liste des codes des comptes-rendus accompagnée d'une description générale et d'une liste des cas dans lesquels chaque compte-rendu peut intervenir. L'annexe C contient une description plus détaillée de la signification des comptes-rendus pour chaque instruction ou fonction.

Code	*Signification*	*Situation(s)*
Ø	Achèvement réussi, ou saut à un numéro de ligne plus grand que les numéros existants. Un compte-rendu avec un code Ø ne modifie pas le numéro de ligne utilisé par **CONT**.	Non spécifique
1	La variable de contrôle n'existe pas (n'a pas été établie par une instruction **FOR**) mais il existe une variable standard portant le même nom.	**NEXT**
2	La variable qui a été utilisée est indéterminée. Pour une variable simple, ceci intervient si la variable est utilisée avant d'avoir été l'objet d'une affectation dans une instruction **LET**. Pour une variable indicée, ceci se produit si la variable est utilisée avant d'avoir été dimensionnée dans une instruction **DIM**. Pour une variable de contrôle, ceci se produit si la variable est utilisée avant d'avoir été établie comme variable de contrôle dans une instruction **FOR**, lorsqu'il n'existe pas de variable simple standard portant le même nom.	Non spécifique
3	Indice hors intervalle. Si l'indice est en dehors de l'intervalle autorisé (négatif ou supérieur à 65535), l'erreur B est lancée.	Variables indicées
4	Pas assez de place en mémoire. Noter que le numéro de la ligne écrit dans le compte-rendu (après le symbole /) peut être incomplet sur l'écran à cause de la pénurie de mémoire : par exemple, 4/2Ø peut se présenter sous la forme 4/2. Voir le chapitre 23. Pour **GOSUB**, voir l'exercice 6 du chapitre 14.	**LET**, **INPUT**, **DIM**, **PRINT**, **LIST**, **PLOT**, **UNPLOT**, **FOR**, **GOSUB**. Parfois pendant l'évaluation d'une fonction.
5	Plus de place sur l'écran. **CONT** permet de dégager de la place en effaçant l'écran.	**PRINT**, **LIST**.
6	Dépassement de capacité arithmétique : les calculs ont donné un nombre supérieur à $1Ø^{38}$ environ.	N'importe quelle opération arithmétique.
7	Il n'y a pas de **GOSUB** correspondant à une instruction **RETURN**.	**RETURN**

Code	*Signification*	*Situation(s)*
8	Vous avez tenté d'utiliser **INPUT** comme commande (ceci est interdit).	**INPUT**
9	L'instruction **STOP** a été exécutée. **CONT** ne tente pas une nouvelle exécution de l'instruction **STOP**.	**STOP**
A	Argument interdit pour certaines fonctions.	**SQR**, **LN**, **ASN**, **ACS**
B	Entier hors intervalle. Lorsqu'un entier est requis, l'argument en virgule flottante est arrondi à l'entier le plus voisin. Si le résultat est en dehors d'un intervalle autorisé, l'erreur B est lancée.	**RUN**, **RAND**, **POKE**, **DIM**, **GOTO**, **GOSUB**, **LIST**, **LLIST**, **PAUSE**, **PLOT**, **UNPLOT**, **CHR$**, **PEEK**, **USR**
	Pour l'accès des tableaux, voir également le compte-rendu 3.	Accès de tableau
C	Le texte de l'argument (chaîne) de **VAL** n'est pas une expression numérique autorisée.	**VAL**
D	(i) Programme interrompu par **BREAK**.	A la fin de n'importe quelle instruction ou dans **LOAD**, **SAVE**, **LPRINT**, **LLIST** ou **COPY**.
	(ii) La ligne **INPUT** commence par **STOP**.	**INPUT**
E	Inutilisé	
F	Le nom donné pour le programme est une chaîne vide.	**SAVE**

Le ZX81 pour ceux qui comprennent le BASIC

Généralités

Si vous connaissez déjà le BASIC, vous n'aurez certainement pas beaucoup de problèmes pour utiliser le ZX81 ; toutefois, il faut quand même signaler une ou deux idiosyncrasies.

(i) Les mots ne doivent pas être orthographiés, ils disposent d'une touche qui leur est propre — ce point est traité au chapitre 2 (pour les mots-clés et les touches en position majuscules) et au chapitre 5 (pour le nom des fonctions). Dans le texte, ces mots sont écrits en **CARACTÈRES GRAS**.

(ii) Le BASIC du ZX81 est dépourvu des instructions READ, DATA et RESTORE (à ce sujet, voyez l'exercice 3 du chapitre 22), des fonctions définies par l'utilisateur (FN et DEF ; mais **VAL** peut parfois être utilisée) et des lignes à plusieurs instructions.

(iii) Les fonctions de traitement des chaînes sont complètes mais non standard (voir le chapitre 21 ainsi que le chapitre 22 pour les tableaux de chaînes).

(iv) Le jeu de caractères du ZX81 est spécifique à cette machine.

(v) De façon générale, l'affichage sur le téléviseur ne correspond pas à un topogramme mémoire.

(vi) Si vous avez l'habitude d'utiliser **PEEK** et **POKE** sur une autre machine, n'oubliez pas que toutes les adresses sont différentes sur le ZX81.

Vitesse

La machine travaille à deux vitesses — normale et rapide, c'est-à-dire mode calcul et affichage et mode rapide.

En mode calcul et affichage, l'image présentée sur le téléviseur est générée en continu et les calculs sont faits pendant les parties vides en haut et en bas de l'image.

En mode rapide, l'image est mise hors fonction pendant les calculs, elle apparaît seulement à la fin d'un programme, pendant que la machine attend une entrée de donnée **INPUT** ou pendant une pause ; voir l'instruction **PAUSE**.

Le mode rapide travaille environ 4 fois plus vite, il doit être utilisé pour les programmes comportant de nombreux calculs et non des résultats volumineux ou bien pour entrer des programmes longs.

Les instructions **FAST** et **SLOW** permettent de passer d'une vitesse à l'autre.

Le clavier

Les caractères du ZX81 ne comprennent pas seulement les symboles (lettres, chiffres, etc.) mais également les jetons composés (mots-clés, noms de fonctions, etc.). Dans ce qui suit, ils sont imprimés en majuscules **CARACTÈRES GRAS**, et leur entrée se fait toujours par le clavier et non lettre à lettre. Pour permettre ce gain de temps, certaines touches ont jusqu'à cinq significations différentes qui s'obtiennent en partie en manœuvrant les touches en position majuscules (c'est-à-dire en appuyant sur la touche majuscules **SHIFT** en même temps que sur la touche requise)et en partie par la sélection du mode de travail de la machine.

Le mode est indiqué par le curseur, une lettre en vidéo inversée, dont le rôle est de montrer la position à laquelle sera inséré le prochain caractère provenant du clavier.

Le mode **K** (K représente mot-clé en anglais — keyword) est mis en œuvre automatiquement lorsque la machine attend une commande ou une ligne de programme (plutôt que des données d'entrée) et, à partir de sa position sur la ligne, le mode sait s'il doit attendre un numéro de ligne ou un mot-clé. Ceci est vrai en début de ligne, immédiatement après quelques chiffres au début de la ligne ou tout de suite après **THEN**. En position normale, la prochaine touche manœuvrée sera interprétée soit comme un mot-clé (écrits pour la plupart en-dessus des touches), soit comme un chiffre.

Le mode **L** (la lettre L représente le mot lettre) est habituellement utilisé à d'autres moments. En position normale, la touche suivante est interprétée comme le symbole principal qu'elle porte.

Dans les modes **K** et **L**, les touches en position majuscules sont interprétées comme le caractère rouge subsidiaire inscrit dans le coin supérieur de droite de la touche.

Le mode **F** (fonctions) entre en jeu après une manœuvre de **FUNCTION** (**NEWLINE** en position majuscules) et dure seulement pendant une manœuvre d'une touche. Cette touche est interprétée comme un nom de fonction ; les noms de fonction sont écrits sous les touches.

Le mode **G** (mode graphique) est mis en œuvre lorsque la touche **GRAPHICS** (9 en position majuscules) a été utilisée, il demeure en activité jusqu'à la prochaine dépression de cette touche. Une touche en position normale donne la vidéo inversée de son interprétation en mode **L**. Une touche en position majuscules donne le même résultat s'il s'agit d'un symbole, mais si en position majuscules, cette touche donne normalement un jeton, en mode graphique, elle donne le symbole graphique inscrit dans le coin inférieur de droite de la touche.

L'écran

L'écran est composé de 24 lignes contenant chacune 32 caractères ; de plus, il est divisé en deux parties. La partie supérieure peut avoir 22 lignes au maximum pour afficher une liste ou le résultat d'un programme. La partie inférieure deux lignes au moins — sert pour l'entrée des commandes, des lignes de programme et des données ainsi que pour la présentation des comptes-rendus.

Entrées par le clavier : ces données sont affichées sur la partie inférieure de l'écran au fur et à mesure de leur entrée, chaque caractère (symbole simple ou jeton composé) étant inséré immédiatement avant le curseur. Le curseur peut être déplacé vers la gauche en manœuvrant la touche ◊ (5 en position majuscules) ou vers la droite avec ◊ (8 en position majuscules). Le caractère qui est avant le curseur peut être supprimé à l'aide de **RUBOUT** (Ø en position majuscules). (Nota : une ligne complète peut être supprimée en appuyant sur **EDIT** (1 en position majuscules) et ensuite sur **NEWLINE** ; ou encore, s'il s'agit de données d'entrée **INPUT**, en manœuvrant seulement **EDIT**).

Lorsque vous utilisez **NEWLINE**, la ligne est exécutée, elle est mise dans le programme ou utilisée comme donnée d'entrée **INPUT** suivant le cas, sauf si elle contient une erreur de syntaxe. Dans ce cas, le symbole **S** apparaît juste avant l'erreur.

Au fur et à mesure de l'entrée de lignes de programme, une liste est affichée sur la moitié supérieure de l'écran. Les modalités d'élaboration de la liste sont plutôt compliquées, l'explication plus détaillée de cette opération figure à l'exercice 6 du chapitre 9. La dernière ligne à entrer s'appelle la ligne en cours ou ligne courante, elle est indiquée par le symbole ▶, mais une modification peut intervenir en manœuvrant les touches ⇦ (6 en position majuscules) et ⇨ (7 en position majuscules). Si vous appuyez sur **EDIT** (1 en position majuscules), la ligne en cours est descendue c'est-à-dire mise dans la partie inférieure de l'écran, elle peut maintenant être éditée.

Lorsqu'une commande est exécutée, ou bien à l'exécution d'un programme, l'écran est d'abord effacé et le résultat est ensuite affiché sur sa partie supérieure où il demeure jusqu'à l'entrée d'une ligne de programme ou jusqu'au moment où vous manœuvrez **NEWLINE** avec une ligne vide, ou encore jusqu'à la manœuvre de ⇦ ou ⇨. La partie inférieure présente alors un compte-rendu de la forme m/n où m est un code précisant ce qui est arrivé (voir l'annexe B) et n est le numéro de la dernière ligne exécutée — ou Ø pour une commande. Le compte-rendu demeure sur l'écran jusqu'à la prochaine manœuvre d'une touche (et indique le mode K).

Dans certains cas, la touche **SPACE** joue le rôle de **BREAK**, c'est-à-dire qu'elle arrête l'ordinateur en lançant le compte-rendu D. Ceci est reconnu :

(i) à la fin d'une instruction pendant l'exécution d'un programme,

(ii) pendant que la machine recherche un programme sur bande,

ou (iii) pendant que l'ordinateur utilise l'imprimante (ou encore lorsqu'il essaye par erreur de l'utiliser, alors qu'elle n'est pas reliée au système).

Le BASIC

Les nombres sont stockés avec une précision de 9 à 1Ø chiffres. Le plus grand nombre utilisable est $1Ø^{38}$ environ et le plus petit nombre (positif) est $4*1Ø^{-39}$ environ.

Dans le ZX81, les nombres sont stockés en binaire en virgule flottante avec un octet exposant e ($1 \leqslant e \leqslant 255$) et quatre octets de mantisse m ($1/2 \leqslant m < 1$). Ceci représente le nombre m6 2^{e-128}.

Puisque nous avons la relation $1/2 \leqslant m < 1$, le bit le plus significatif de la mantisse m est toujours 1. Par conséquent, en pratique, nous pouvons le remplacer par un bit pour indiquer le signe — Ø pour les nombres positifs, 1 pour les négatifs.

Zéro bénéficie d'une représentation spéciale dans laquelle les 5 octets sont à Ø.

Les variables numériques ont des noms de longueur arbitraire ; ces noms commencent par une lettre et se poursuivent par des lettres et des chiffres. Tous les caractères sont significatifs de sorte que, par exemple, LONGNAME et LONGNAMETOO sont des noms différents. Les espaces sont ignorés.

Les variables de contrôle des boucles **FOR—NEXT** sont désignées par des noms d'une seule lettre.

Les tableaux numériques sont désignés par des noms d'une seule lettre, le nom d'un tableau numérique peut être le même que celui d'une variable simple. Ils peuvent avoir arbitrairement plusieurs dimensions de longueur arbitraire. Les indices commencent à 1.

La longueur des chaînes est variable. Le nom d'une chaîne est composé d'une seule lettre, suivi de $.

Les tableaux de chaînes peuvent avoir arbitrairement plusieurs dimensions de taille arbitraire. Le nom est une seule lettre suivie de $, mais le nom d'un tableau de chaînes ne peut pas être le même que celui d'une chaîne. Toutes les chaînes d'un tableau donné doivent être de la même longueur fixe ; cette longueur est spécifiée comme dimension supplémentaire finale dans l'instruction **DIM**. Les indices commencent à 1.

Découpage : Des sous-chaînes de chaînes peuvent être spécifiées en utilisant des séparateurs.

Un séparateur peut être

(i) vide

ou

(ii) une expression numérique

ou

(iii) une expression numérique facultative **TO** expression numérique facultative

et s'utilise pour exprimer une sous-chaîne, soit par

(a) une expression de chaîne (séparateur)

soit

(b) par une variable de tableaux de chaînes (indice,... indice, séparateur)

qui signifie la même chose que

variable de tableau de chaînes (indice, ... indice) (séparateur)

En (a), supposons que l'expression chaîne a la valeur s$.

Si le séparateur est vide, le résultat est s$ qui est considérée comme une sous-chaîne.

Si le séparateur est une expression numérique de valeur m, le résultat est le mième caractère de s$ (une sous-chaîne de longueur 1).

Si le séparateur est de la forme (iii), supposons que la première expression numérique a la valeur m (la valeur par défaut est 1), et la deuxième, n (la valeur par défaut est la longueur de s$).

Si $1 \leqslant m \leqslant n$ longueur de s$, le résultat est la sous-chaîne de s$ qui commence au mième caractère et se termine au nième.

Si $\emptyset \leqslant n < m$, le résultat est la chaîne vide.

Autrement, le compte-rendu d'erreur 3 est lancé.

Le découpage est exécuté avant l'évaluation des fonctions ou des opérations, sauf si des parenthèses modifient cet ordre.

Les sous-chaînes peuvent faire l'objet d'affectations (voir **LET**).

L'argument d'une fonction peut être dépourvu de parenthèses si c'est une constante ou une variable (éventuellement indicée ou séparée).

Fonction	Type de l'opérande (x)	Résultat
ABS	nombre	Valeur absolue
ACS	nombre	Arc cosinus en radians. Erreur A si x n'est pas dans l'intervalle $-1/+1$.
AND	opération binaire, opérande de droite toujours un nombre	
	opérande numérique de gauche :	A **AND** B $= \begin{cases} A \text{ si } B \neq \emptyset \\ \emptyset \text{ si } B = \emptyset \end{cases}$
	opérande chaîne de gauche	A$ **AND** B $= \begin{cases} A\$ \text{ si } B \neq \emptyset \\ "" \text{ si } B = \emptyset \end{cases}$
ASN	nombre	Arc sinus en radians. Erreur A si x n'est pas dans l'intervalle $-1/+1$.
ATN	nombre	Arc tangente en radians.
CHR$	nombre	Le caractère dont le code est x, arrondi à l'entier le plus proche. Erreur B si x n'est pas dans l'intervalle $\emptyset/255$.
CODE	chaîne	Le code du premier caractère dans x (ou $\emptyset$ si x est la chaîne vide).
COS	nombre (en radians)	Cosinus
EXP	nombre	e^x
INKEY$	néant	lit les claviers. Le résultat est le caractère représentant (en mode L) la touche manœuvrée s'il y en a une exactement, ou, autrement, la chaîne vide.
INT	nombre	Partie entière (toujours arrondie par défaut).
LEN	chaîne	Longueur
LN	nombre	Logarithme naturel (base e). Une erreur A intervient si $x \leqslant \emptyset$.
NOT	nombre	$\emptyset$ si $x \neq \emptyset$, 1 si $x = \emptyset$. **NOT** a la priorité 4.
OR	opération binaire les deux opérandes sont des nombres	A **OR** B $= \begin{cases} 1 \text{ si } B \neq \emptyset \\ A \text{ si } B = \emptyset \end{cases}$ **OR** a la priorité 2.

Fonction	Type de l'opérande	Résultat
PEEK	nombre	La valeur de l'octet en mémoire dont l'adresse est x (arrondi à l'entier le plus proche). L'erreur B est lancée si x est en dehors de l'intervalle Ø/65535.
PI	néant	π (3.14159265).
RND	néant	Le prochain nombre pseudo-aléatoire y dans une séquence générée en prenant les puissances de 75 (modulo 65537, en soustrayant 1 et en divisant par 65536.) $Ø \leqslant y \leqslant 1$.
SGN	nombre	Signe : le signe (−1, Ø ou +1) de x.
SIN	nombre (en radians)	Sinus
SQR	nombre	Racine carrée L'erreur B intervient si $x \leq Ø$.
STR$	nombre	La chaîne de caractères qui serait affichée si x était écrit.
TAN	nombre (en radians)	Tangente
USR	nombre	Appelle le sous-programme en langage machine dont l'adresse de début est x (arrondie à l'entier le plus proche). Au retour, le résultat est le contenu de la paire de registres bc. L'erreur B est lancée si x n'est pas dans l'intervalle Ø/65535.
VAL	chaîne	Evalue x (sans ses guillemets de séparation) comme une expression numérique. L'erreur C intervient si x contient une erreur de syntaxe ou donne une valeur de chaîne. D'autres erreurs sont possibles, selon l'expression.
	nombre	Négation

La liste suivante reprend les opérations binaires :

+	Addition (de nombres) ou concaténation (de chaînes)	
−	Soustraction	
*	Multiplication	
/	Division	
**	Elévation à une puissance. L'erreur B est lancée si l'opérande de gauche est négatif.	
=	Egal	Les deux opérandes doivent être du même type. Le résultat est un nombre, 1 si la comparaison est vérifiée et Ø si elle ne l'est pas.
>	Supérieur à	
<	Inférieur à	
<=	Inférieur ou égal à	

>= ⎫ Supérieur ou égal à
<> ⎭ Différent de.

Les priorités des fonctions et des opérations sont les suivantes :

Opération	Priorité
Indiçage et découpage	12
Toutes les fonctions sauf **NOT** et le moins unaire	11
**	10
Moins unaire	9
*,/	8
+,– (–binaire)	6
=,>,<,<=,>=,<>	5
NOT	4
AND	3
OR	2

Instructions

Dans la liste suivante,

α	représente une seule lettre
v	représente une variable
x,y,z	représentent des expressions numériques
m,n	représentent des expressions numériques arrondies à l'entier le plus proche.
e	représente une expression
f	représente une expression valorisée en chaîne
s	représente une instruction

Notez que les expressions arbitraires sont autorisées partout (sauf le numéro de ligne qui doit être au début d'une instruction).

Toutes les instructions à l'exception de **INPUT** peuvent être utilisées comme commandes ou dans des programmes (bien qu'elles puissent être plus utiles dans un cas que dans l'autre).

CLEAR Supprime toutes les variables et libère l'espace qu'elles occupaient.

CLS (Clear Screen = effacement écran) efface le fichier d'affichage. Voir au chapitre 27 les explications relatives au fichier d'affichage.

CONT Si p/q était le dernier compte-rendu avec une indication autre que zéro, **CONT** a l'effet suivant

⎧ **GOTO** q si p ≠ 9
⎩ **GOTO** q+1 si p = 9 (instruction **STOP**)

COPY

Envoie à l'imprimante (si elle est connectée) une copie du contenu de l'écran ; autrement, cette fonction n'agit pas.
Contrairement à toutes les autres commandes, une commande **COPY** n'efface pas d'abord l'écran. Les espaces sont interdits avant **COPY**.
Le compte-rendu D est lancé si la touche **BREAK** est manœuvrée.

DIM$\alpha(n_1, \ldots, n_k)$

Supprime le tableau désigné par le nom α et établit un tableau α de nombres à k dimensions $n_1, \ldots, n_k$. Met toutes les valeurs à zéro.
L'erreur 4 intervient s'il n'y a pas assez de place pour caser le tableau. Un tableau demeure indéterminé jusqu'au moment où il est dimensionné dans une instruction **DIM**.

DIM$\alpha\$(n_1, \ldots, n_k)$

Supprime le tableau ou la chaîne désigné par le nom $\alpha\$$ et établit un tableau $\alpha\$$ de caractères à k dimensions $n_1, \ldots n_k$. Met toutes les valeurs à " ". Ceci peut être considéré comme un tableau de chaînes de longueur fixe n_k à k^{-1} dimensions $n_1, \ldots n_{k-1}$.
L'erreur 4 est lancée s'il n'y a pas de place pour caser le tableau. Un tableau est indéterminé jusqu'au moment où il est dimensionné dans une instruction **DIM**.

FAST

Lance le mode rapide dans lequel le fichier d'affichage est seulement mis sur l'écran à la fin du programme, pendant l'entrée **INPUT** de données ou pendant une pause.

FORα=x **TO** y

FORα=x **TO** y **STEP** z

FORα=x **TO** y **STEP** 1.

Supprime n'importe quelle variable simple α, et établit une variable de contrôle à la valeur x, avec la limite y, au pas z et dont l'adresse de boucle est 1 de plus que le numéro de ligne de l'instruction **FOR** (—1 s'il s'agit d'une commande). Vérifie si la valeur initiale est supérieure (si pas $\geqslant \emptyset$) ou inférieure (si pas $< \emptyset$) à la limite et, dans l'affirmative, saute à l'instruction **NEXT** α au début d'une ligne.
Voir **NEXT** α.
L'erreur 4 intervient s'il n'y a pas de place pour la variable de contrôle.

GOSUB n

Met le numéro de ligne de l'instruction **GOSUB** dans la pile ; puis, comme **GOTO** n.
L'erreur 4 peut intervenir s'il n'y a pas assez de **RETURN**.

GOTO n

Saute à la ligne n (ou bien, s'il n'y en a pas, à la première ligne suivante).

IF x **THEN** s

Si x est vérifié, (non-nul) s est exécuté. La forme "**IF** x **THEN** numéro de ligne" est interdite.

INPUTv

Arrête (sans message particulier) et attend que l'utilisateur entre une expression ; la valeur de cette expression est affectée à v. En mode rapide, le fichier d'affichage est mis sur l'écran. **INPUT** ne peut pas être utilisée comme commande ; l'erreur 8 intervient si vous essayez cette opération. Si le premier caractère de la ligne d'entrée **INPUT** est **STOP**, le programme s'arrête et lance le compte-rendu D.

LETv=e

Affecte la valeur de e à la variable v.
LET ne peut pas être omise.
Une variable simple est indéterminée jusqu'au moment où son affectation intervient par une instruction **LET** ou **INPUT**.
Si v est une variable chaîne indicée ou une variable chaîne séparée (sous-chaîne), l'affectation est du type de Procruste ; la valeur chaîne de e est tronquée ou remplie avec des espaces à droite pour lui donner la même longueur que la variable v.

LIST

LIST Ø

LIST N

Etablit la liste du programme sur la télévision à partir de la lettre n et fait de n la ligne en cours.
L'erreur 4 ou 5 est lancée si la liste est trop longue pour être mise sur l'écran ; **CONT** refait exactement la même chose.

LLIST

LLIST Ø

LLISTn

Comme **LIST**, mais avec l'imprimante au lieu du téléviseur. Doit demeurer sans effet si l'imprimante n'est pas connectée. Arrêt avec lancement du compte-rendu D si la touche **BREAK** est manœuvrée.

LOADf

Recherche sur la bande magnétique un programme appelé f et le charge avec ses variables.
si f = "", le programme chargé est le premier programme disponible.
Si vous utilisez **BREAK** ou si une erreur de bande est détectée, (i) si un programme n'a pas encore été lu sur la bande, un arrêt intervient avec un compte-rendu D et l'ancien programme ;
(ii) si une partie d'un programme a déjà été mise dans la machine, il y a exécution de **NEW**.

LPRINT...

Comme **PRINT**, mais avec l'imprimante au lieu du téléviseur. Une ligne de texte est transmise à l'imprimante
(i) lorsque l'écriture déborde d'une ligne à la suivante,
(ii) après une instruction **LPRINT** qui ne se termine pas par une virgule ou par un point virgule,

(iii) lorsqu'une virgule ou une rubrique de **TAB** exige une nouvelle ligne ou encore

(iv) à la fin du programme, s'il reste encore des choses non écrites.

Dans une rubrique **AT**, seul le numéro de colonne a un effet. Le numéro de ligne est ignoré (sauf que les mêmes cas d'erreur que pour **PRINT** ont lieu si le numéro est en dehors de l'intervalle autorisé). Une rubrique **AT** ne transmet jamais une ligne de texte à l'imprimante.

Cette instruction doit être sans effet si l'imprimante n'est pas connectée. Arrêt avec le compte-rendu D si la touche **BREAK** est manœuvrée.

NEW Provoque un démarrage "à neuf" du système BASIC et supprimant les programmes et les variables et utilisant la mémoire jusqu'à l'octet (non compris) dont l'adresse est dans la variable système RANBOT (octets 16388 et 16389).

NEXTα

(i) Trouve la variable de contrôle α.
(ii) Ajoute son pas à sa valeur.
(iii) Si le pas $\geqslant \emptyset$ et la valeur $>$ la limite, ou
si le pas $< \emptyset$ et la valeur $<$ la limite,
et passe ensuite à la ligne de la boucle.

L'erreur 1 est lancée s'il y a une variable simple α.
L'erreur 2 intervient s'il n'y a ni variable simple ni variable de contrôle α.

PAUSEn Arrête le calcul et met le fichier d'affichage sur l'écran pendant n trames (à raison de 50 trames par seconde) ou jusqu'au moment où une touche est manœuvrée.
$\emptyset \leqslant n \leqslant 65535$ ou autrement, lancement de l'erreur B si $n \geqslant 32767$, la durée de la pause n'est pas comptée et celle-ci dure jusqu'au moment où vous manœuvrez une touche.

PLOTm,n Fait devenir noir l'élément d'image (|m|,|n); met la position d'écriture **PRINT** à celle qui est immédiatement après cet élément d'image.
$\emptyset \leqslant |m| \leqslant 63{,}\emptyset$ | n| $\leqslant 43$; autrement l'erreur B est lancée.

POKEm,n Ecrit la valeur n à l'octet de mémoire dont l'adresse est m.
$\emptyset \leqslant m \leqslant 65535$, $-255 \leqslant n \leqslant 255$; autrement, l'erreur B est lancée.

PRINT . . . Les " . . ." représentent une séquence de rubriques de **PRINT** séparées par des virgules ou des points virgules ; ces rubriques sont écrites sur le fichier d'affichage pour présentation sur le téléviseur. La position (ligne et colonne) à laquelle le prochain caractère doit être écrit est appelée position d'écriture (**PRINT**).

Une rubrique **PRINT** peut être
(i) vide, c'est-à-dire rien,
(ii) une expression numérique.

Un signe moins est écrit au début si la valeur est négative. Soit maintenant x le modulo de la valeur.

Si $x \leqslant 1Ø^{-5}$ ou $x \geqslant 1Ø^{13}$, l'écriture se fait en notation scientifique. La partie mantisse contient 8 chiffres au maximum (sans zéros de fin) et la marque décimale (omise s'il n'y a qu'un chiffre) est placée après le premier chiffre. La partie exposant est E suivi de + ou – suivi d'un ou de deux chiffres.

Autrement, x est écrit en notation décimale normale avec 8 chiffres significatifs au maximum, pas de zéros de fin après la marque décimale. Une marque décimale placée immédiatement au début est toujours suivie d'un zéro de sorte que, par exemple, .Ø3 et Ø.3 sont imprimés de cette façon.

Ø est imprimé sous forme d'un seul chiffre Ø.

(iii) une expression chaîne.

Les jetons de la chaîne sont agrandis, éventuellement au moyen d'un espace avant ou après.

Le caractère image guillemet s'écrit sous la forme ".

Les caractères inutilisés et les caractères de contrôle sont écrits sous la forme ?.

(iv) **AT** m,n

La position d'écriture passe à la ligne |m| (comptée à partir du haut) et la colonne n (comptée à partir de la gauche).

$Ø \leqslant |m| \leqslant 21$, autrement, l'erreur 5 est lancée si $|m| = 22$ ou 23, autrement l'erreur B est lancée.

$Ø \leqslant |n| \leqslant 31$, autrement, l'erreur B intervient.

(v) **TAB** n

n est réduit modulo 32. Ensuite, la position d'écriture **PRINT** est mise à la colonne n, sur la même ligne, sauf s'il entraîne un déplacement en arrière ; dans ce cas, la position d'écriture passe à la ligne suivante.

$Ø \leqslant n \leqslant 255$, autrement, l'erreur B est lancée.

L'insertion d'un point virgule entre deux rubriques laisse la position d'écriture inchangée, de sorte que la deuxième rubrique est placée immédiatement après la première. Par contre, une virgule déplace la position d'écriture **PRINT** d'une position au moins et, ensuite, du nombre de positions requises pour qu'elle soit dans la colonne zéro ou dans la colonne 16, avec une nouvelle ligne si nécessaire.

A la fin d'une instruction **PRINT** qui ne se termine pas par un point virgule ou par une virgule, une nouvelle ligne intervient.
L'erreur 4 (hors mémoire) peut intervenir si la capacité de mémoire est de 3K ou moins.
L'erreur 5 signifie que l'écran est plein.
Dans les deux cas, le remède est **CONT**, qui efface l'écran et déclenche la continuation des opérations.

RAND
RAND Ø

RAND n
Etablit la variable système (appelée SEED = base) qui sert à générer la valeur suivante de **RND**. Si n ≠ Ø SEED est mise à la valeur n ; si n = Ø, elle est mise à la valeur d'une autre variable système (appelée FRAMES = trames) qui compte les trames déjà mises sur le téléviseur et doit être relativement aléatoire.
L'erreur B intervient si n n'est pas dans l'intervalle Ø 65535.

REM . . .
Pas d'effet. " . . . " peut être une séquence quelconque de caractères à l'exception de **NEWLINE**.

RETURN
Extrait un numéro de ligne de la pile de **GOSUB** et saute à la ligne qui le suit.
L'erreur 7 intervient s'il n'y a pas de numéro de ligne dans la pile. Votre programme doit contenir une erreur ; les **GOSUB** ne correspondent pas exactement aux **RETURN**.

RUN
RUN Ø

RUN n
CLEAR, et ensuite **GOTO** n.

SAVE f
Enregistre sur bande le programme et les variables et les désigne f. **SAVE** ne doit pas être utilisée dans une routine **GOSUB**.
L'erreur F intervient si f est la chaîne vide, ce qui est interdit.

SCROLL
Fait remonter le fichier d'affichage d'une ligne, provoquant la perte de la première ligne et insérant une ligne vide en bas.
NB : La nouvelle ligne est véritablement vide, puisqu'elle n'est composée que d'un caractère **NEWLINE** sans aucun espace (voir le chapitre 27).

SLOW
Met l'ordinateur en mode calcul et affichage pour lequel le fichier d'affichage est présenté en continu et l'exécution des calculs correspond aux espaces en haut et en bas de l'image.

STOP
Arrête le programme et lance le compte-rendu 9. **CONT** reprend l'exécution à la ligne suivante.

UNPLOT m,n
Comme **PLOT**, mais enlève la couleur d'un élément d'information au lieu de le rendre noir.

Index

Cet index reprend les touches du clavier et la façon de s'en servir – **K**, **L**, **F** ou **G** en position majuscules ou non) ainsi que leurs codes.

Habituellement, une indication n'est citée qu'une fois par chapitre ; par conséquent, lorsque vous avez trouvé une référence, vous devez lire le reste du chapitre, exercices compris.

IMP. AGGATOR PARIS (11) 222 60 20

DIRECO INTERNATIONAL
30, avenue de Messine
75008 Paris