SONY

MSX

Initiation au MSX-BASIC

HIT BIT

Initiation au MSX-BASIC

SOMMAIRE

Point de départ 4
En guise d'introduction 4
Milou, le petit chien savant 5
Votre ordinateur Sony 6

Mise en service de votre ordinateur 8
Les instructions et les entrées 8
Quelques trucs impossibles pour Milou 12
Impression de la réponse 16

Des nombres, des lettres et des variables 19
Nombres ou lettres: ça revient au même 20
Quand une lettre devient une variable 21

Faites votre premier programme 30
Planification d'un programme 32
Ecriture d'un programme: simple comme bonjour 35
Correction d'un programme:
chercher la petite bête 37
Exécution d'un programme 39

Les graphiques: une constellation d'étoiles 42
Un programme pour graphique 42
Nombres aléatoires 48
Un peu de couleur sur les programmes 51

Sauvegarde des programmes 59
Branchement d'un magnétophone 60
Pour faire parler l'ordinateur 61
Vérification de la sauvegarde sur bande 62
Chargement d'un programme 64

Décisions et jeux 65
Réussi ou raté: l'ordinateur est l'arbitre 67
Simplification d'un programme 69

Des graphiques en mouvement 72
Afficher, effacer, afficher, effacer 75

Pour obtenir un ensemble cohérent 80
Avec de la couleur et du son 80
Jeu de devinette+son+vidéo 82

Le jeu de l'appareil à jetons 87
Qu'est-ce qu'une variable en tableau? 88
Pour faire un appareil à jetons 90
Des variables en chaîne 96
Pour arrêter une boucle par INKEY 99
Quel est le score? 101
Les touches de finition 107
Pour faciliter la lecture d'un programme 109
Insertion de titres dans un programme 110
Que de chemin parcouru! 113

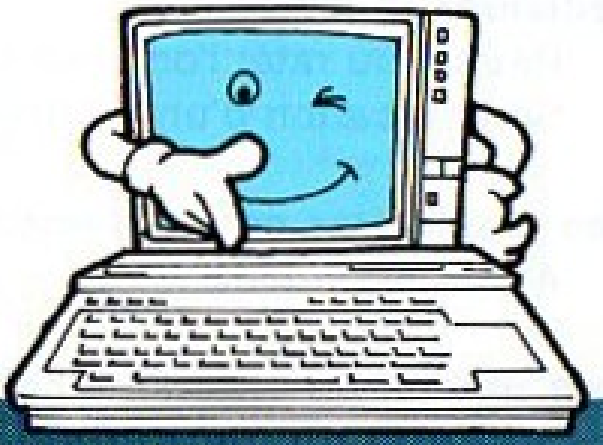
Pratique des instructions en BASIC 115
PRINT 115
INPUT 118
FOR—NEXT 120
IF—THEN et IF—THEN—ELSE 123
DIM (Les variables en tableau) 125

Index alphabétique 130

Ce manuel est destiné à une utilisation exclusive avec un système MSX de Sony.

POINT DE DEPART

- En guise d'introduction
- Milou, le petit chien savant
- Votre ordinateur Sony



EN GUISE D'INTRODUCTION

Cet ouvrage se propose de vous aider au cours de votre itinéraire vers le BASIC.

Que signifie le terme BASIC? Il s'agit d'un langage simple, destiné à "dialoguer" avec un ordinateur. Si l'on parle de "langage BASIC", c'est, en partie, parce qu'il n'existe guère de façon plus simple, plus fondamentale, de s'adresser à un ordinateur.

Mais pourquoi faut-il dialoguer avec un ordinateur, direz-vous? Parce que votre ordinateur Sony peut vous aider à réaliser bien des choses: vous amuser avec des jeux, résoudre des problèmes, d'autres choses encore. Ce manuel vous montrera comment, en quelques heures à peine, vous pourrez prendre plaisir à utiliser votre ordinateur Sony.

Dans un premier temps, nous allons voir comment votre ordinateur écoute et vous adresse la parole (en langage BASIC). Ensuite, nous utiliserons ce langage particulier pour demander à l'ordinateur de réaliser certains petits "tours" intéressants, très simples au départ, puis un peu plus complexes. Avant de refermer cet ouvrage, vous vous amuserez, sur votre ordinateur et avec lui, à un jeu de "machine à sous", tel que ceux que vous avez sûrement vus dans les maisons de jeux vidéo.

La lecture de cet ouvrage contribuera à vous familiariser avec le BASIC; vous serez ainsi à même de lire et d'utiliser les programmes, présentés dans les revues spécialisées, de partager vos découvertes avec vos amis qui possèdent également un ordinateur. Ajoutons qu'il existe diverses autres publications qui vous permettront de vous perfectionner en BASIC.

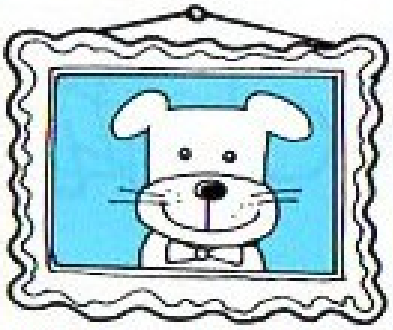
Au cours de votre itinéraire, vous rencontrerez toute une série d'instructions et de termes spéciaux qui ont d'ailleurs été regroupés en annexe dans l'**Index alphabétique**. Vous pourrez donc facilement vérifier la signification de l'un d'eux en vous référant à la fin de ce livre.

Et pour terminer, sachez que les ordinateurs ont vraiment "bon caractère". Ils sont simples, sympathiques et patients, même si vous répétez les mêmes erreurs au cours de votre cheminement. Rassurez-vous: vous ne risquez nullement de "détraquer" votre ordinateur Sony si, par mégarde, vous appuyez sur la touche qu'il ne fallait pas. Comme pour toute activité nouvelle, vous devrez procéder à des essais, essuyer certains échecs, franchir trois pas puis reculer de deux pour, finalement, avancer et réussir. Quoi qu'il en soit, vous constaterez que votre compagnon a vraiment une patience d'ange.

Pour clore ce préambule, retenez que votre ordinateur Sony n'a qu'un désir, celui de devenir votre ami et de vous aider dans une foule de situations.

MILOU, LE PETIT CHIEN SAVANT

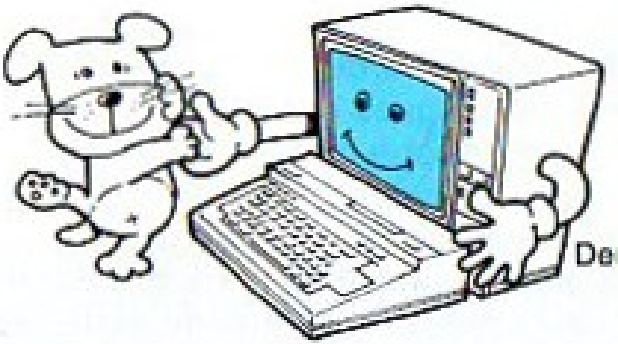
Avant de se mettre en marche, il est bon de se faire accompagner. Pour cheminer en BASIC, vous aurez un ami en la personne de Milou qui, comme la plupart des toutous, est doux et fidèle. Si Milou est un chien comme beaucoup d'autres, il est plutôt malin et, en réponse à vos ordres, il effectuera des dizaines de petits trucs.



Milou

Pourquoi introduire un toutou comme Milou dans un livre, traitant de BASIC et d'ordinateurs? Tout d'abord, parce que les chiens sont excellents pour suivre les **instructions** qui leur sont données et pour effectuer des tours. Ensuite parce que, Milou et votre ordinateur forment une paire de bons amis et que votre ordinateur, vous le comprendrez bientôt, excelle aussi quand on lui demande d'exécuter des **instructions** et de faire des tours, même s'ils n'agissent pas de la même façon.

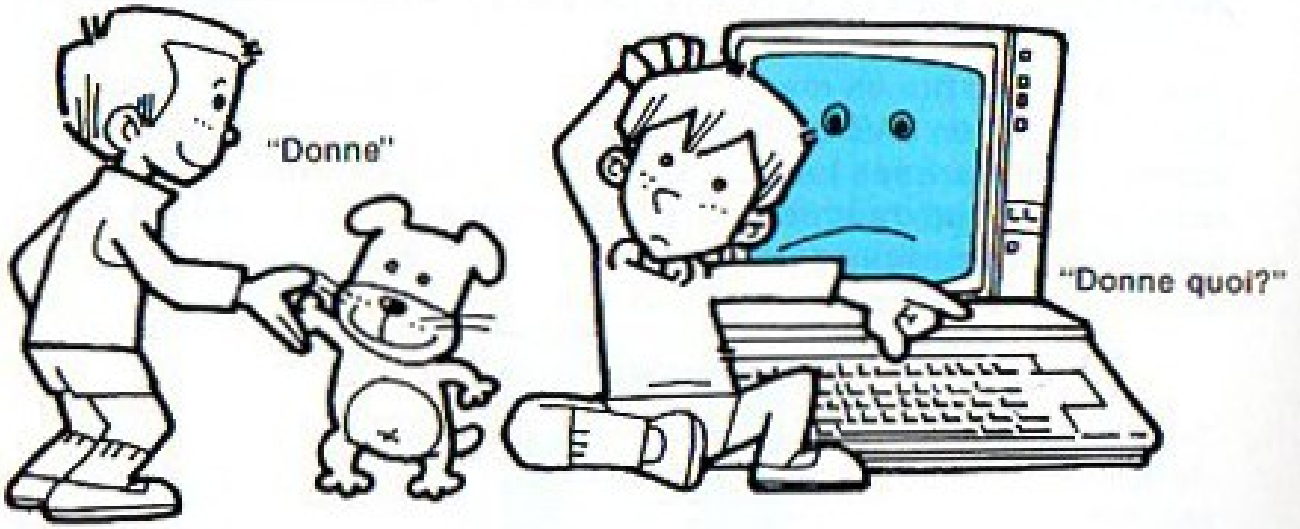
C'est en réfléchissant à la manière d'agir de Milou et de l'ordinateur que l'on comprendra le mieux comment un ordinateur écoute, parle et réfléchit en BASIC.



Deux bons amis

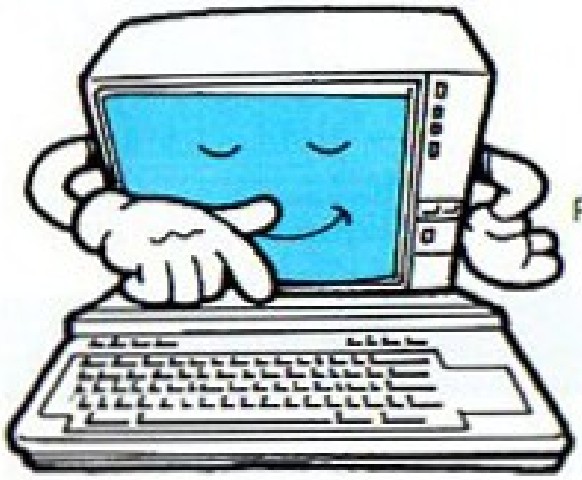
VOTRE ORDINATEUR SONY

Comme tous les chiens, ,Milou excelle à suivre les ordres: "Assis", "Demi-tour", "Cherche", "Rapporte", etc., c'est son fort!
Mais qu'en est-il de votre ordinateur Sony? Essayez par exemple "Donne la patte".



Milou avait compris et il vous a présenté la patte. Mais, que pouvait faire l'ordinateur qui n'a ni mains ni oreilles!

Mais alors, comment demander à un ordinateur de faire quelque chose? En fait, les "oreilles" de l'ordinateur, sa manière d'entendre vos ordres, c'est son **clavier.**



Parlez avec vos doigts

Avant de vous lancer dans le BASIC, et avant de passer à la page suivante, veuillez lire **le mode d'emploi** de l'ordinateur pour savoir comment fonctionne le clavier. Il n'est pas requis de mémoriser toutes les touches, mais vous devriez vous familiariser avec le **curseur**, savoir ce qu'il représente et comment il se déplace. Vous devriez aussi essayer de taper quelques **lettres,** des **chiffres** et des **graphiques** sur l'écran.

MISE EN SERVICE DE VOTRE ORDINATEUR

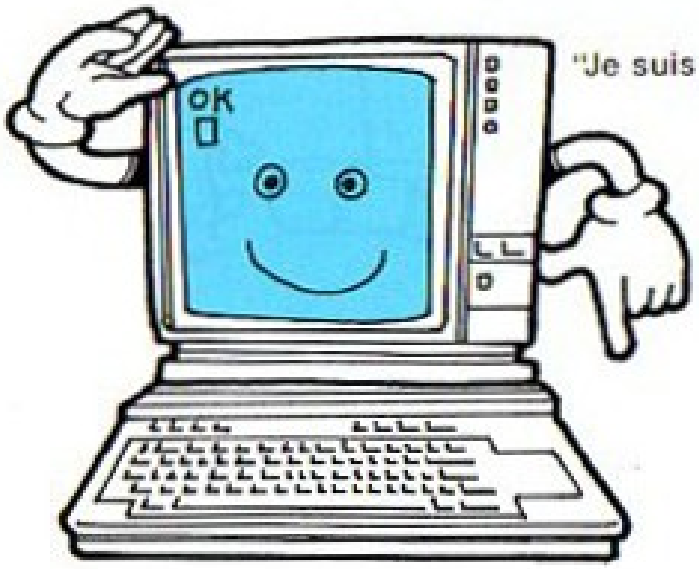
Que faire pour...

- Arrêter l'ordinateur
- Donner des ordres
- Taper en couleur
- Utiliser la touche majuscule
- Faire exécuter de l'arithmétique à l'ordinateur

Vous avez lu le "Mode d'emploi" et le clavier n'a plus guère de secret pour vous. Voyons maintenant ce que l'ordinateur peut faire pour vous aider.

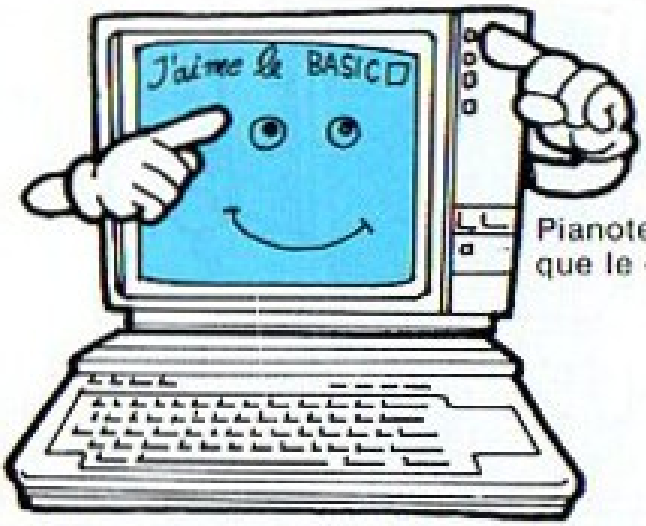
LES INSTRUCTIONS ET LES ENTREES

Nous avons mentionné précédemment que le clavier formait "les oreilles", utilisées par l'ordinateur pour entendre vos messages.
Mais, comment peut-on savoir que l'ordinateur écoute? Il vous fait savoir, par un amical "Ok", quand il est prêt à agir.



"Je suis prêt; à votre service!"

Lorsque vous pianotez sur le clavier, les lettres et les chiffres tapés apparaissent sur l'écran. Le curseur se déplace vers le centre de l'écran, mais le message "Ok" est laissé sur place.

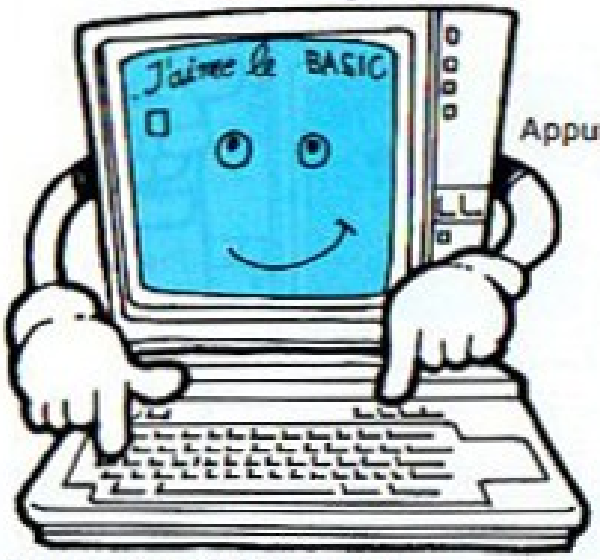


Pianotez quelques mots et remarquez que le curseur se déplace.

Si c'est le cas, nous devons attirer l'attention de l'ordinateur.
A cet effet, appuyez sur la touche [CTRL] avec un doigt, puis sur la touche [STOP] avec un autre. Lorsque vous appuyez simultanément sur [CTRL] et [STOP], l'ordinateur cesse d'écouter vos anciennes instructions et il accorde toute son attention à vos nouvelles commandes.

Une instruction n'est pas la même chose qu'une **entrée**. Une entrée, ce sont des lettres et des chiffres qui constituent une information que vous fournissez à l'ordinateur, comme toutes les lettres nécessaires pour écrire "J'aime le BASIC" sur l'écran.
L'entrée ne dit pas directement à l'ordinateur de "faire" quelque chose. Pour cela, nous avons besoin des "**instructions**", données par certaines touches ou certains mots tapés, qui disent à l'ordinateur ce qu'il doit faire avec la donnée entrée. Pour concrétiser, disons que, sur une calculatrice de poche par exemple, "2", " + " et "2" sont l'entrée dans l'addition 2 + 2 = 4.
Dans ce cas, la touche "=" est une **instruction** qui dit à l'ordinateur d'ajouter 2 à 2 et de donner le résultat, c'est-à-dire la sortie: 4. C'est exactement comme ça que travaille un ordinateur.

Autre détail important: les lettres "Ok" apparaissent **toujours** au-dessus du curseur quand une **instruction** a été entrée correctement. Autrement dit, ce que vous avez fait est parfait!

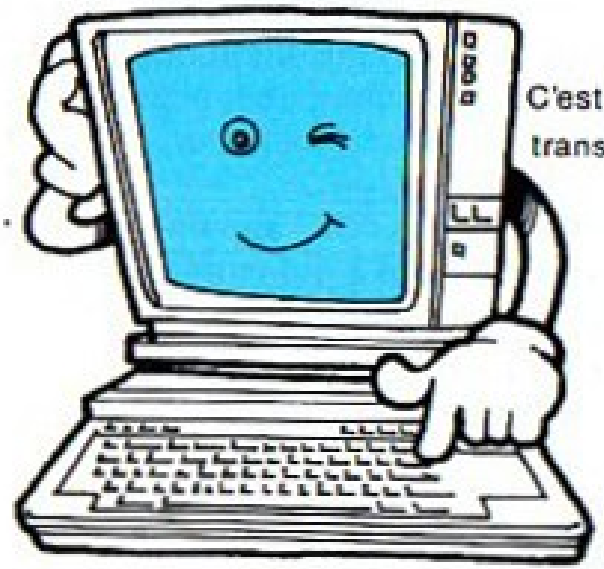


Appuyez sur CTRL et STOP.

Désormais, l'ordinateur est prêt à recevoir vos ordres que vous entrerez, indifféremment, en minuscules ou en majuscules. Demandons lui, par exemple, de donner la main en tapant "DONNE LA PATTE".

```
DONNE LA PATTE
```

Et bien quoi? Pas de réponse?
Bien sûr, car nous avons oublié de dire à l'ordinateur quand il doit exécuter l'instruction. La touche qui permet ceci se trouve sur la droite et elle est marquée "RETURN". **Chaque fois que vous donnez un ordre à votre ordinateur,** vous le mettrez à exécution par la touche RETURN [1], légèrement plus grande que les autres, précisément parce que vous l'emploierez plus souvent.

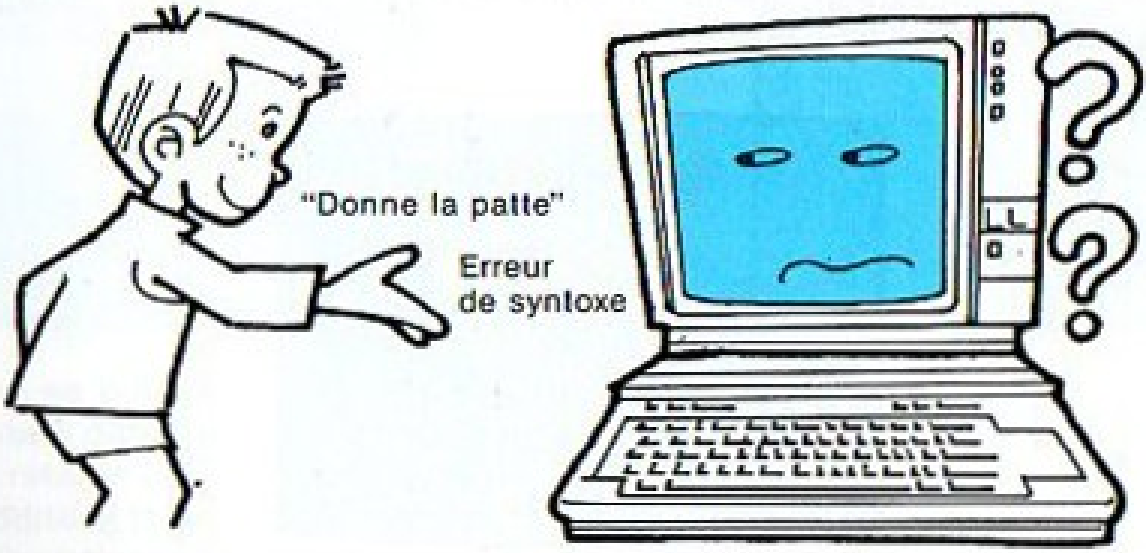


C'est la touche RETURN qui transmet les instructions.

1) La touche ↵ sur certains claviers est la même que la touche RETURN dans ce manuel.

Appuyons sur RETURN et maintenant nous obtenons une réponse. Non pas une poignée de main, bien sûr, mais un "bip", une tonalité et une réponse de ce genre:

Ce que nous obtenons, c'est un **message d'erreur**, qui signifie que l'ordinateur a bel et bien entendu l'ordre donné, mais qu'il ne le comprend pas. Faut-il s'en étonner puisque les ordinateurs ne sont pas particulièrement conçus pour serrer la main des humains. Le message "Syntax error," signifie que s'est glissée une erreur de langage, de langage BASIC, bien entendu.



Notre brave Milou, lui, est capable de comprendre "Assis", "La patte" et bien d'autres mots, tout simplement parce qu'il les a appris de quelqu'un. Or, notre ordinateur a reçu une autre formation et les instructions qu'il comprend sont différentes. Elles s'expriment en langage, appelé BASIC. Au cours de votre itinéraire, en parcourant les pages de cet ouvrage, vous les apprendrez progressivement et vous verrez qu'il est facile de dialoguer avec l'ordinateur. Essayons-en quelques-unes, dès le début, pour se faire la main.

QUELQUES TRUCS IMPOSSIBLES POUR MILOU ——

Voici, pour commencer, un petit truc tout simple, bien que ce ne soit peut-être pas l'avis de Milou ...

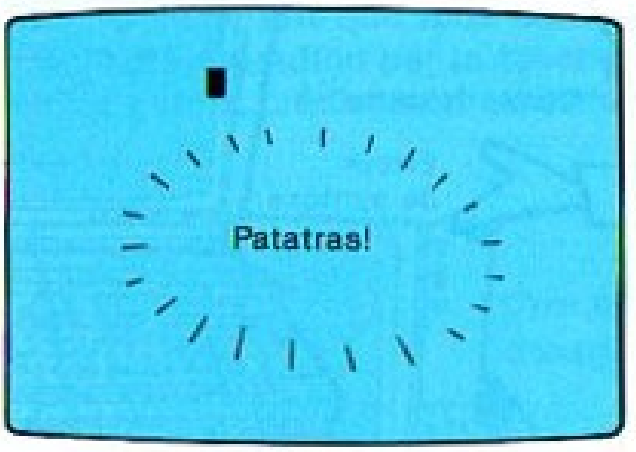
```
WIDTH 10
```

Il n'est pas bien difficile de taper ceci dans l'ordinateur par l'intermédiaire du clavier, mais n'oubliez pas d'appuyer sur la barre d'espacement pour placer un **espacement** entre le mot WIDTH et le chiffre 10.

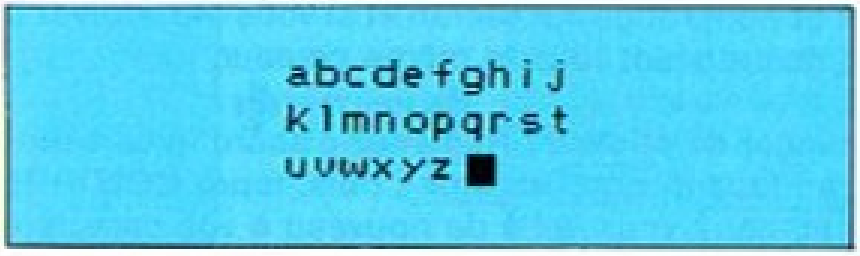
Ne vous inquiétez pas si une faute de frappe s'est glissée; souvenez-vous des explications du mode d'emploi sur le déplacement du curseur et apportez les corrections éventuelles. Votre ordinateur oublie toutes vos erreurs dès que vous les avez corrigées. Si votre écran affiche ceci:

```
WIDTH 10■
```

vous êtes en mesure de donner un ordre. Appuyez sur la touche RETURN et regardez ce qui se passe...



Les lettres ont disparu et le curseur se trouve près du milieu de l'écran. A quoi donc a servi l'instruction WIDTH 10? Pour le découvrir, essayons de pianoter quelque chose sur le clavier, de fournir une entrée quelconque, par exemple toutes les lettres de l'alphabet sans aucun espacement ni retour à la ligne par la touche RETURN.



Quel est le nombre de lettres par ligne? Dix. Sans aucune autre instruction, l'ordinateur a compris que nous voulions que la largeur de chaque ligne soit limitée à dix caractères seulement. "WIDTH 10" est une des instructions qu'il comprend; quand il reçoit cet ordre, il fait en sorte que chaque ligne de l'écran ait exactement dix caractères de largeur.

Passons à un autre essai, mais au préalable, appuyez simultanément sur les touches CTRL et STOP pour "effacer" les dernières entrées et préparer l'ordinateur à recevoir l'instruction **suivante:**

```
WIDTH 37
```

Appuyez maintenant sur la touche RETURN.

Que se passe-t-il encore? L'écran est à nouveau vide et le curseur est revenu dans le coin supérieur gauche. Vous venez de dire à l'ordinateur de rétablir la longueur d'une ligne à 37 caractères et, jusqu'à ce que vous modifiiez l'instruction, tout ce que vous taperez dorénavant se présentera en lignes de 37 caractères (soit dit en passant, chaque fois que vous mettez votre appareil sous tension, la longueur d'une ligne à l'écran est automatiquement fixée à 37 caractères, à moins que vous ne décidiez de la modifier).

Essayons maintenant une nouvelle instruction qui laissera Milou perplexe. Tapez tout d'abord CTRL et STOP, puis:

```
COLOR 8
```

(N'oubliez pas d'appuyer sur la touche [RETURN].) Tout à coup, chacune des lettres devient rouge sur l'écran et si vous introduisez quelques mots nouveaux, ils prennent tous la même couleur.

Et pour changer de couleur? Ramenez d'abord le curseur sur la gauche de l'écran en tapant [CTRL] et [STOP]; puis, tapez COLOR 15 et actionnez la touche [RETURN]: vous voilà de nouveau à vos caractères en blanc.

Il est facile de comprendre que, pour l'ordinateur, 15 signifie blanc et 8 veut dire rouge; mais ne serait-il pas plus facile de taper COLOR RED ou COLOR WHITE? Pour nous, les humains, et même pour le savant Milou, les mots sont plus pratiques que les nombres. Mais ce n'est pas le cas pour les ordinateurs qui adorent les chiffres par dessus tout et c'est pourquoi nous utilisons des chiffres pour nous faire comprendre de l'ordinateur. Comme vous le constaterez, le langage BASIC fait souvent appel à des chiffres pour remplacer des mots de notre langage.

Lorsque vous aurez l'habitude de dialoguer avec l'ordinateur, vous utiliserez, vous aussi, des nombres pour lui faire retenir des noms, des endroits, des couleurs ou des phrases de tous genres. Cette démarche pourrait être comparée à notre façon de donner des numéros pour désigner la voiture ou le domicile de diverses personnes; pour un ordinateur, cette façon de s'exprimer par des chiffres est la plus naturelle. En peu temps, vous vous servirez aussi de nombres et de lettres dans vos programmes en BASIC.

Nous savons déjà que 8 signifie rouge et 15 blanc. De combien de couleurs disposons-nous en fait?

code	couleur	code	couleur	code	couleur	code	couleur
0	transparent	4	bleu foncé	8	rouge moyen	12	vert foncé
1	noir	5	bleu clair	9	rouge clair	13	magenta
2	vert moyen	6	rouge foncé	10	jaune foncé	14	gris
3	vert clair	7	bleu ciel	11	jaune clair	15	blanc

Nous avons donc sous la main une palette de 16 couleurs avec lesquelles il est bien agréable de s'amuser. En guise de jeu, essayez donc ceci:

```
COLOR 4
```

Des caractères bleu foncé sur un fond bleu foncé! Aurait-on utlisé de l'encre sympathique? Tout semble avoir disparu!

Référez-vous au tableau du code des couleurs et faites apparaître d'autres coloris sur l'écran. Vous pouvez ainsi adapter les couleurs à votre humeur du jour, pourquoi pas?

Et voici quelque chose d'un peu plus compliqué.

```
COLOR 1000
```

(N'oubliez pas RETURN.) Le chiffre 1000 ne figure pas dans la liste des couleurs et l'odrinateur le sait. Une brève tonalité vous dit que l'ordinateur va vous répondre. Il vous dit "Illegal function call", ce qui est sa manière à lui dire "N'essayez pas de me tromper!"

En réalité, vous ne parviendrez jamais à tromper vptre ordinateur Sony. Supposons que vous vouliez taper

```
COLOR 6
```

mais que, par erreur, vous tapiez

```
CILOR 6
```

Sans remarquer la faute de frappe, vous actionnez la touche RETURN et, de nouveau, l'odrinateur fait entendre une tonalité. Cette fois, son message est "Syntax error."

Il arrive à tout le monde, même aux professionnels chez Sony, de faire des fautes de frappe. Heureusement, cela a été prévu et c'est pourquoi nous disposons des touches [BS] (espace arrière), [DEL] (effacement), [INS] (insertion) et des touches de déplacement du curseur. Au besoin, faites un petit retour au mode d'emploi et en peu de temps, vous aurez appris comment corriger vos fautes de frappe éventuelles.
(Voir la remarque en page 18.)

Quelle est l'étendue du vocabulaire en BASIC?
Vous venez d'apprendre deux instructions en BASIC, à savoir WIDTH et COLOR et vous avez vu que votre ordinateur Sony répond avec fidélité aux ordres correctement donnés. Il fait exactement ce qu'on lui dit, pourvu que l'orthographe soit respectée, ou bien il se fait entendre et s'explique si vous écrivez mal ou si vous lui demandez quelque chose qu'il n'est pas sensé faire. Combien existe-t-il de mots d'instruction, en tout, en BASIC? Une centaine environ et cela suffit pour dire à l'ordinateur comment jouer de la musique, tracer des dessins, résoudre des problèmes mathématiques et bien d'autres choses encore. Vous trouverez tous ces termes dans le "Manuel de référence de programmation". Bien sûr , quelques-uns d'entre eux suffisent pour commencer cet itinéraire et vous auriez tort de vous décourager en pensant qu'une centaine de mots, c'est un gros fardeau pour un débutant.

En cours de route, votre ordinateur Sony **vous** apprendra aussi certaines choses. Il connait, en effet, quelque 35 **messages d'erreur** différents, comme "Syntax error" ou "Illegal function call" que nous avons déjà rencontrés. D'habitude, il vous dira ce qui ne lui convient pas si vous lui donnez un ordre qui dépasse son entendement.

IMPRESSION DE LA REPONSE

Notre Milou adore jouer des tours, comme "donner la patte", par exemple. Ceux qu'affectionne l'ordinateur sont différents. C'est le cas, tout particulièrement des mathématiques, où vraiment il est très fort.

Introduisons ce qui suit:

```
PRINT 3+5
```

Avez-vous trouvé le signe plus (+)? Il se trouve au-dessus de "=" et il apparaît sur l'écran si vous appuyez **simultanément** sur la touche [SHIFT] et sur "=".

Appuyez maintenant sur la touche [RETURN] pour entrer cette instruction dans l'ordinateur.

```
PRINT 3+5
 8
Ok
■
```

Et voilà! Vous vous êtes servi de l'ordinateur pour résoudre un problème et imprimer la réponse. Essayons-en un autre:

```
PRINT 100-10
```

Vous l'aurez remarqué sans doute, la touche SHIFT est inutile pour le signe moins (–).

Utilisez maintenant la touche RETURN pour entrer la commande.

```
PRINT 3+5
 8
Ok
PRINT 100-10
 90
Ok
■
```

Evidemment, la réponse de l'ordinateur est 90. Mais, direz-vous, il s'agit là de problèmes très simples. Et pourtant, l'ordinateur exécute avec la même facilité des additions ou des soustractions, comportant des millions ou des milliards.

Qu'en est-il des multiplications? Rien de plus simple si l'on tient compte que l'ordinateur utilise un **signe spécial** dans ce cas. Au lieu d'écrire 7 × 9, on entrera 7 * 9.

Tapez, par exemple, PRINT 7 * 9 et appuyez sur RETURN.

```
PRINT 7*9
 63
Ok
■
```

Pour effectuer une division, on se sert de la touche [/]. Par conséquent, au lieu de PRINT 120 ÷ 40, vous taperez:

```
PRINT 120/40
```

Et, bien sûr, l'ordinateur vous donne la réponse exacte quand vous appuyez sur [RETURN].

```
PRINT 120/40
 3
Ok
■
```

Vous pouvez maintenant passer à des opérations plus importantes, comme (7 654 321 × 18), ou des problèmes complexes tels que (2 + 9) × (6 – 8)?

Amusez-vous dorénavant et essayez toutes sortes d'opérations. L'ordinateur vous donnera parfois des réponses incompréhensibles, mais qu'à cela ne tienne! Avancez pas à pas, réfléchissez aux résultats obtenus et, en quelques minutes, vous réaliserez de petits prodiges.

Le fort en maths!

[REMARQUE]

Certaines touches de votre ordinateur MSX peuvent différer des touches présentées dans ce manuel.
Se référer au tableau ci-dessous.

Sur le clavier	Dans ce manuel
[↵]	[RETURN]
[⇧]	[SHIFT]
[←]	[BS]
[⇪]	[CAP]
[SUP]	[DEL]
[EFE]	[HOME]

DES NOMBRES, DES LETTRES ET DES VARIABLES

Que faire pour...

- Imprimer des mots et des phrases
- Créer une variable (un symbole avec une valeur changeante)
- Utiliser les variables en mathématiques
- Donner des noms différents aux variables

Nous avons donc utilisé l'instruction PRINT pour demander à l'ordinateur de résoudre des problèmes mathématiques et, jusqu'à présent, il s'est comporté comme n'importe quelle calculatrice. Il nous faut maintenant aborder certaines autres formes de calcul que peut effectuer cet appareil. Découvrons tout d'abord certains traits nouveaux de l'instruction PRINT et livrons-nous à quelques exercices en tapant

```
PRINT "JEAN ET MARIE"
```

Quand tout a été entré, appuyez sur la touche RETURN.

```
PRINT "JEAN ET MARIE"
JEAN ET MARIE
Ok
■
```

En clair, disons que l'ordinateur "lit" les guillemets et comprend comme suit l'instruction PRINT: je dois imprimer ce qui se trouve entre les guillemets, mais pas les guillements proprement dits.
Voici quelques autres exemples:

```
PRINT "ABC TO XYZ"
ABC TO XYZ
PRINT "How are you?"
How are you?
```

Il va sans dire que vous devez actionner chaque fois la touche RETURN. A ce stade, vous avez sûrement pris l'habitude de le faire et désormais, ceci ne sera donc plus mentionné: tapez sur la touche RETURN chaque fois que vous désirez entrer une instruction.

NOMBRES OU LETTRES: CA REVIENT AU MEME —

Essayons maintenant quelque chose de différent.

```
PRINT "3+5"
```

Ceci ressemble étrangement à ce que nous avons vu dans les pages précédentes. A ce moment-là, nous avions demandé à l'ordinateur **PRINT 3+5** et, complaisamment, il avait répondu 8, parce qu'il n'y avait pas de guillemets.

Avec des guillemets, voici ce que nous obtenons:

```
PRINT "3+5"
3+5
Ok
■
```

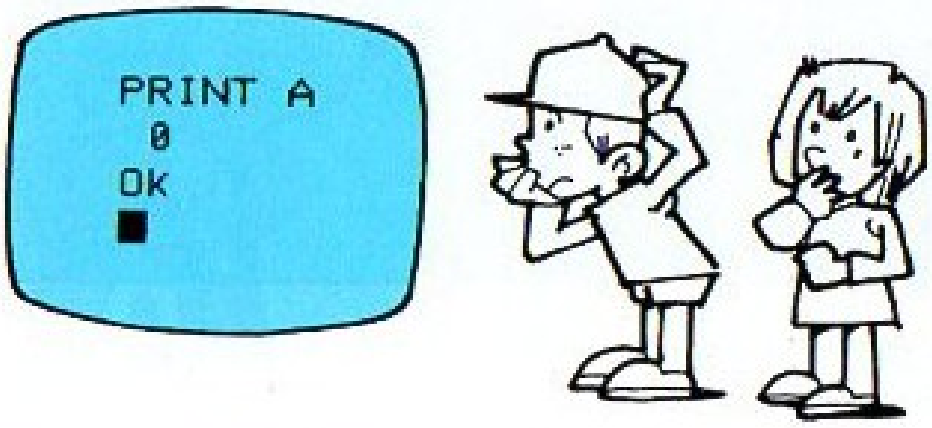
Dans ce cas, l'instruction **PRINT, utilisée avec les guillemets** dit à l'ordinateur de lire ce qui se trouve entre les guillemets et de l'imprimer simplement sur l'écran. A cause de la présence des guillemets, l'ordinateur lit "3+5" comme trois symboles à imprimer, et non pas comme un problème de mathématiques à résoudre.

Les deux se font de la même manière.

Voici de qui se passe quand un nombre est compris entre des guillemets.

QUAND UNE LETTRE DEVIENT UNE VARIABLE

Nous allons maintenant aborder une autre chose intéressante à propos de l'instruction PRINT. Que se passe-t-il quand une lettre sans guillemets arrive après PRINT?



Qu'avons-nous obtenu? Pas de tonalité! Pas de message "Syntax error"! En revanche, nous avons simplement un 0 et le message Ok, qui nous dit que l'ordinateur a exécuté l'instruction et qu'il attend l'ordre suivant. Mais, quelle **a été** notre instruction?

Puisqu'il n'y a pas de message d'erreur, c'est que PRINT A est une instruction parfaitement acceptable. Que signifie-t-elle pour l'ordinateur? Pour le comprendre, essayons ceci:

```
LET A=3
```

Lorsque nous actionnons la touche [RETURN], aucun message d'erreur n'apparaît mais, à part le gentil message Ok, il ne se passe pas grand chose non plus. Puisque l'ordinateur a accepté notre instruction, il doit se passer quelque chose à l'intérieur.

Que se passe-t-il là-dedans?

A présent, mettons ces deux instructions ensemble:

```
LET A=3
Ok
PRINT A
 3
Ok
■
```

Vous avez sans doute déjà une idée de ce qui s'est passé, mais essayons encore une fois:

```
LET A=379
Ok
PRINT A
 379
Ok
■
```

Il y a quelques instants, le **A** était **3**, et maintenant c'est **379**. Si nous le désirons, nous pouvons le modifier encore, simplement en écrivant **LET A=** et en faisant suivre n'importe quel autre nombre. Ceci est ce que l'on appelle une **variable.**

Par conséquent, ce genre de **A** n'est plus réellement la première lettre de l'alphabet; plutôt, quand elle apparaît après PRINT mais sans guillemets, elle devient comme une sorte de récipient, pouvant contenir n'importe quel nombre que nous voulons lui confier.

On pourrait donc comparer une variable à un chapeau de magicien parce que nous pouvons, à volonté, y placer des valeurs que nous retirerons, comme le magicien, chaque fois que nous le voudrons.

Si nous tapons LET A=3, nous plaçons un 3 dans le chapeau et nous pouvons l'en retirer simplement en tapant PRINT A. Nous pouvons ensuite changer en 379 la valeur qui se trouve dans le chapeau par une instruction toute simple (LET A=379) et la valeur apparaîtra alors quand le souhaiterons.

Ainsi, LET est une instruction qui donne une valeur spécifique à notre variable. En BASIC, les variables sont très utiles et elles seront fréquemment employées dans cet ouvrage. Toutefois, pour la facilité des travaux, nous pouvons écrire cette instruction plus rapidement en laissant tomber le mot LET:

```
A=3
```

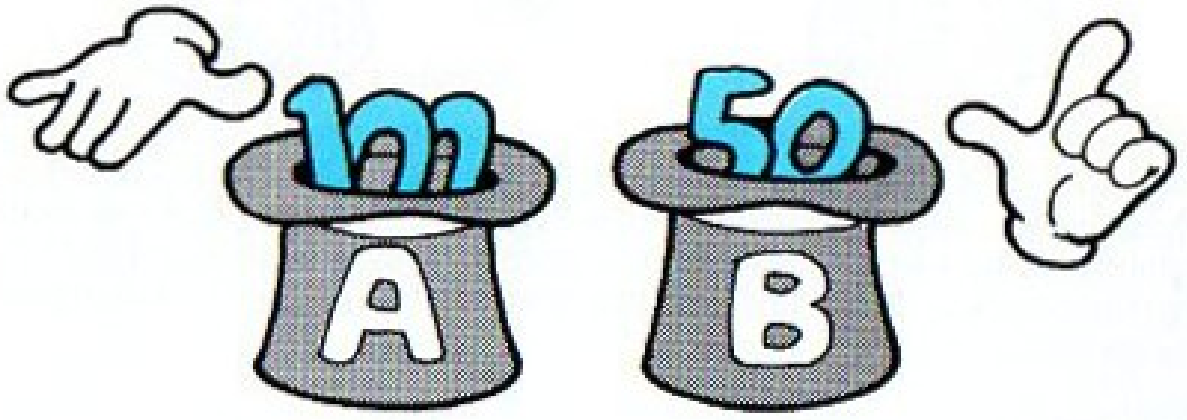
a la même signification que LET A=3.

Pour être sûr que vous n'avez rien oublié, procédons à une petite révision:

```
A=100                Donnez la valeur 100 à A
Ok
PRINT A              Affichez A
 100                 Et voilà
Ok
PRINT "A"            Affichez la lettre "A" (et pas la variable A)
A                    Et voilà
Ok
B=50                 Donnez une valeur à B
Ok
PRINT B              Affichez B
 50                  C'est fait
Ok
■
```

Précédemment, nous avons donné l'instruction PRINT A et l'ordinateur nous a répondu 0. Ceci veut dire que toute variable a la valeur zéro jusqu'à ce que nous lui donnions une valeur différente.

Une variable peut être représentée à l'écran par n'importe quelle lettre. Nous avons déjà utilisé A et B et nous leur avons donné des valeurs différentes.



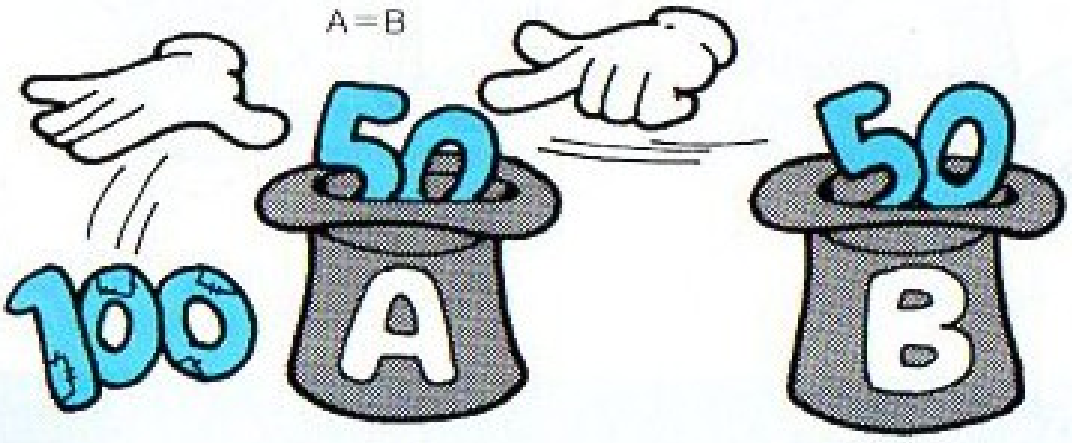
Voyons à présent ce que nous pouvons faire avec ces variables.

Nous avons déjà dit à l'ordinateur que A=100 et que B=50. Donnez maintenant ces trois instructions: A=B, PRINT A et PRINT B. Vous avez certainement deviné le résultat, mais par souci de méthode, essayons quand même.

```
A=B
Ok
PRINT A
 50
Ok
PRINT B
 50
Ok
■
```

Observez bien la première ligne. En tapant A=B, nous avons dit à l'ordinateur d'attribuer la valeur de B à A. Précédemment, nous avions changé la valeur de A de 3 en 379 et l'ancienne valeur avait tout simplement disparu.

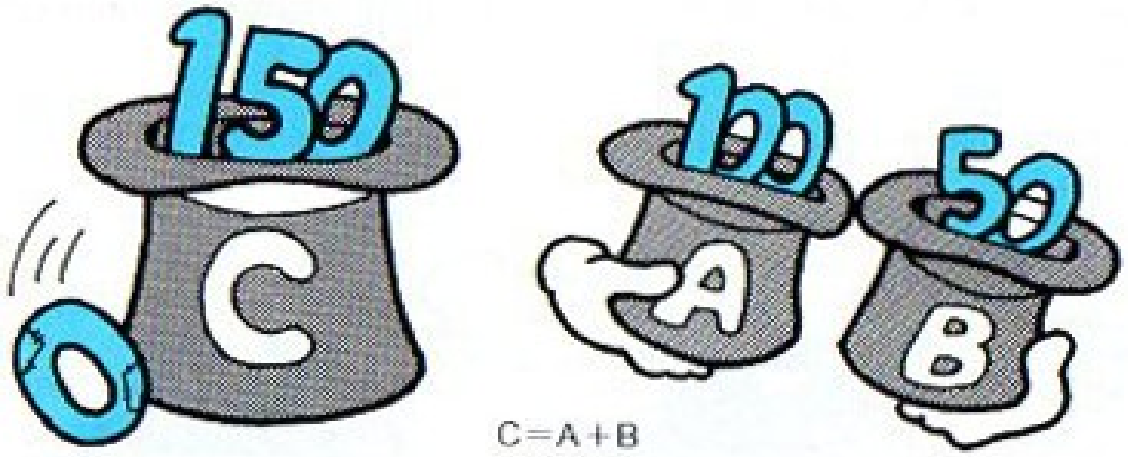
La même chose s'est produite ici: nous avons changé A de 100 en B, c'est-à-dire en 50.



Maintenant, nous pouvons nous livrer à un peu d'exercices mathématiques. Tapez ces quatre instructions:

```
A=100
Ok
B=50
Ok
C=A+B
Ok
PRINT C
 150
Ok
■
```

Ici, nous disposons de trois récipients, à savoir A, B et C et l'ordinateur effectue le calcul pour savoir combien il doit placer dans le récipient... ou le chapeau C.

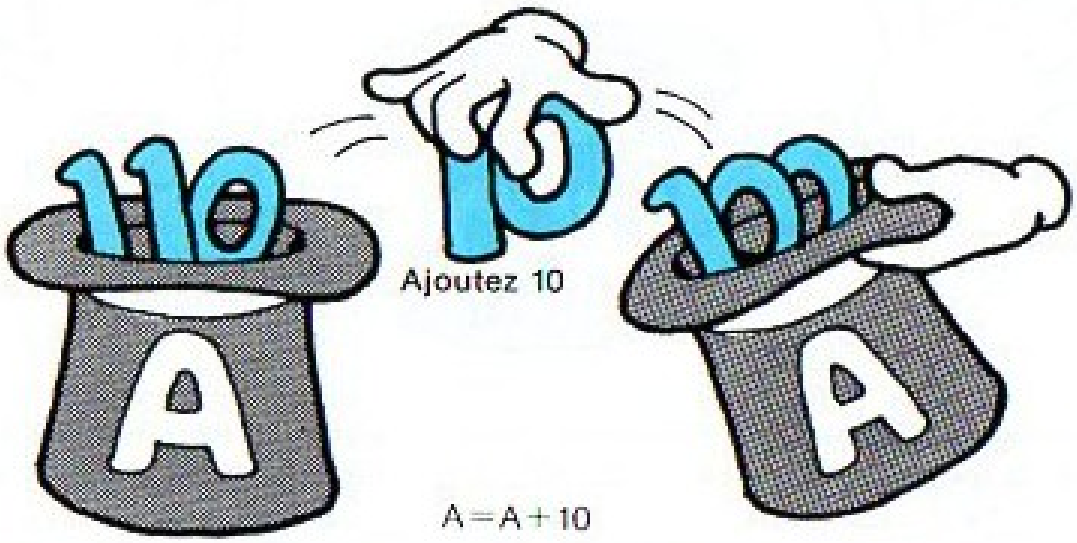


C=A+B

Mais voici quelque chose de plus intéressant encore:

```
A=A+10
Ok
PRINT A
 110
Ok
■
```

Ceci semble-t-il confus? Expliquons. Le A avait une " valeur affectée" de 100. Mais en lisant cette instruction, l'ordinateur a compris notre information comme ceci: "Que désormais A ait une valeur qui soit 10 de plus qu'auparavant." Et c'est pourquoi A est maintenant 110.



Sans doute souhaitez-vous essayer la seconde instruction à nouveau? Et à nouveau? Avant chaque fois, vous devriez être capable de déduire aisément ce que sera le résultat. Votre ordinateur, lui **se souviendra toujours** de la valeur affectée à la variable.

Toujours à propos des variables

Vous savez à présent ce que sont les variables et comment elles fonctionnent. Nous devons voir maintenant comment elles s'emploient et nous les utiliserons dans des programmes et des jeux. Les variables se livrent, en effet, à de nombreuses applications. Ainsi, une variable pourra enregistrer le score d'un jeu, car il s'agit d'un nombre qui varie chaque fois que des points sont ajoutés ou soustraits. D'autres pourrons signifier la position changeante d'un vaisseau spatial ou encore la vitesse d'une voiture de course.

Nous avons déjà utilisé trois lettres (A, B et C) comme variables, mais vous pouvez utitliser pratiquement **n'importe** quelle lettre ou groupe de lettres.

```
UFO=5
Ok
PRINT UFO
 5
Ok
■
```

Comme vous le voyez, un groupe de trois lettres sans espace entre elles, comme UFO dans ce cas, peut représenter un nombre **simple**.

Il importe également de remarquer que chaque lettre différente ou groupe de lettres, comme A, B ou AB, a une signification différente.

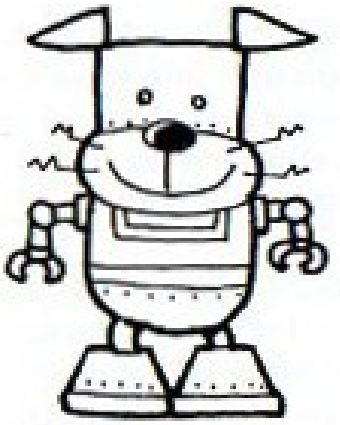
```
A=3
Ok
B=5
Ok
PRINT A
 3
Ok
PRINT B
 5
Ok
PRINT AB
 0
Ok
■
```

Ici, nous avons affecté la valeur 3 à la lettre A et B est égal à 5; toutefois, aucune valeur n'a été entrée pour la variable AB et, par conséquent, l'ordinateur affiche AB avec la valeur 0 (et **non pas** 35!). Rappelez-vous: une variable a toujours la valeur de 0 jusqu'à ce que vous lui donniez une valeur différente.

Le nom que vous donnez à une variable peut comporter deux, trois, dix... ou même 100 **caractères** qui peuvent être aussi bien des **chiffres** que des **lettres**. Cependant, **l'ordinateur ne lira que les deux premiers caractères**, dont le premier doit toujours être une lettre. Ceci revient à dire que, par la machine, POA, POB, PO1 et PO2 seront considérés comme la même valeur simple. Par conséquent, il vous faudra être un peu prudent dans le choix du nom de vos variables.

Ajoutons que les mots, utilisés dans le langage BASIC, **ne peuvent plus** servir comme nom d'une variable. Si vous donnez à votre ordinateur Sony un des termes du BASIC, il l'interprètera comme une **instruction** et non pas comme une variable. C'est ainsi, par exemple, que LET, GOTO ou PRINT ne peuvent pas devenir le nom d'une variable, de même que GOTOA, BLET ou l'une quelconque des instructions que vous recontrerez prochainement dans ce livre.

FAITES VOTRE PREMIER PROGRAMME



Que faire pour...

- Planifier un programme
- Ecrire un programme simple
- Mémoriser les programmes par l'ordinateur
- Corriger un programme
- Exécuter un programme
- Lire les programmes terminés sur l'écran
- Effacer un programme de la mémoire de l'ordinateur

Jusqu'à présent, nous avons étudié les rudiments de l'emploi d'un ordinateur: faire apparaître des lettres et des nombres sur l'écran, utiliser certains symboles, compris par l'ordinateur parce qu'ils font partie de son "langage" propre et donner certaines instructions (commandes) simples. Autrement dit, nous avons dit à l'ordinateur de **faire** certaines choses sur l'écran, de la manière que nous voulions qu'il les fasse. Nous avons donc appris comment l'homme et l'ordinateur communiquent entre eux.

Cependant, ce genre de travail aurait tout aussi bien pu être obtenu avec une bonne calculatrice de poche. Elle aussi peut ajouter, ou soustraire, ou faire différentes opérations mathématiques, tout en apportant les résultats sur son affichage électronique.

Ce qui différencie un **ordinateur**, et ce qui le rend d'ailleurs si utile pour les hommes de science ou beaucoup d'autres gens, c'est qu'il va plus loin. Bien plus loin! Votre ordinateur Sony peut:

- emmagasiner dans sa **mémoire** une longue série de vos **instructions spéciales**;
- utiliser ces instructions pour exécuter de nombreuses tâches différentes, des jeux, des tours, chaque fois que vous le désirez;
- comprendre le résultat de chaque fonction et utiliser ce résultat lors de la fonction suivante; et
- combiner les résultats de toutes les démarches précédentes, à mesure qu'il effectue les dernières.

En d'autres termes, votre ordinateur peut, automatiquement, suivre un **programme** complet de fonctions et le faire bien plus rapidement que n'importe quelle calculatrice.

La **programmation** est un processus qui consiste à fournir des instructions à l'ordinateur, pour lui dire quelles fonctions il doit remplir et dans quel ordre. Ceci vous permet d'utiliser un ordinateur pour créer vos propres jeux électroniques, pour tracer vos propres figures vidéo ou écrire votre propre musique électronique. Et c'est précisément ce que nous allons entamer au cours de ce chapitre où vous allez réaliser votre premier programme tout simple.

Faire un programme d'ordinateur n'est pas une chose difficile. Vous savez déjà comment donner certaines instructions à l'ordinateur et, au fond, un programme n'est rien d'autre qu'une série d'instructions. La chose importante dans un programme, c'est le **classement**, c'est-à-dire l'organisation ou l'ordonnancement: le **numéro** de l'instruction et l'**ordre** dans lequel les instructions sont données.

Nous allons expliquer les programmes un peu plus clairement en utilisant un autre exemple où travaille un chien savant. Pour l'instant, Milou est au repos, mais nous allons essayer quelques nouveaux tours d'adresse avec un **Super Milou**!

Super Milou est une autre sorte de chien savant et bien gentil: il s'agit d'un robot! (Il faut dire que, de nos jours, tant de chose sont automatisées: alors, pourquoi pas un Super Milou?)

Mon nom est Super Milou et
je peux suivre des programmes!

Comme vous le savez, Milou est capable de faire pas mal de trucs, mais ceux-ci restent toutefois fort simples. Si vous jetez une balle, il va la chercher, il la prend entre ses crocs, vous la rapporte et la dépose devant vous. Il comprend vos ordres, tels que "Donne la patte"; en effet, il la lève et vous la donne. Nous aimons bien Milou pour son intelligence et d'autres raisons, mais en une fois, il ne peut faire que des séquences de trois ou quatre instructions. Pour qu'il continue, vous devez lui indiquer un nouveau truc, lui enseigner un nouveau "programme".

PLANIFICATION D'UN PROGRAMME

Avec Super Milou, c'est différent. Il a avalé un microprocesseur qui lui permet de se souvenir de diverses choses et de prendre des décisions; il peut donc effectuer beaucoup plus de prouesses que Milou en un seul truc, tout comme une calculatrice peut résoudre bien plus de problèmes mathématiques que vous pendant la même durée. Ainsi par exemple, vous pouvez demander à Super Milou de faire ceci:

1. Aller chercher une balle et la rapporter!
2. Si la balle est blanche, tourner trois fois sur lui-même et aboyer!
3. Si la balle est noire, se coucher pendant 10 secondes!

La première démarche est simple, même pour Milou, mais les deux autres sont nettement plus complexes: le chien doit savoir identifier si la balle est noire ou blanche, puis se servir de cette information pour décider quel tour d'adresse il doit effectuer et le faire correctement.

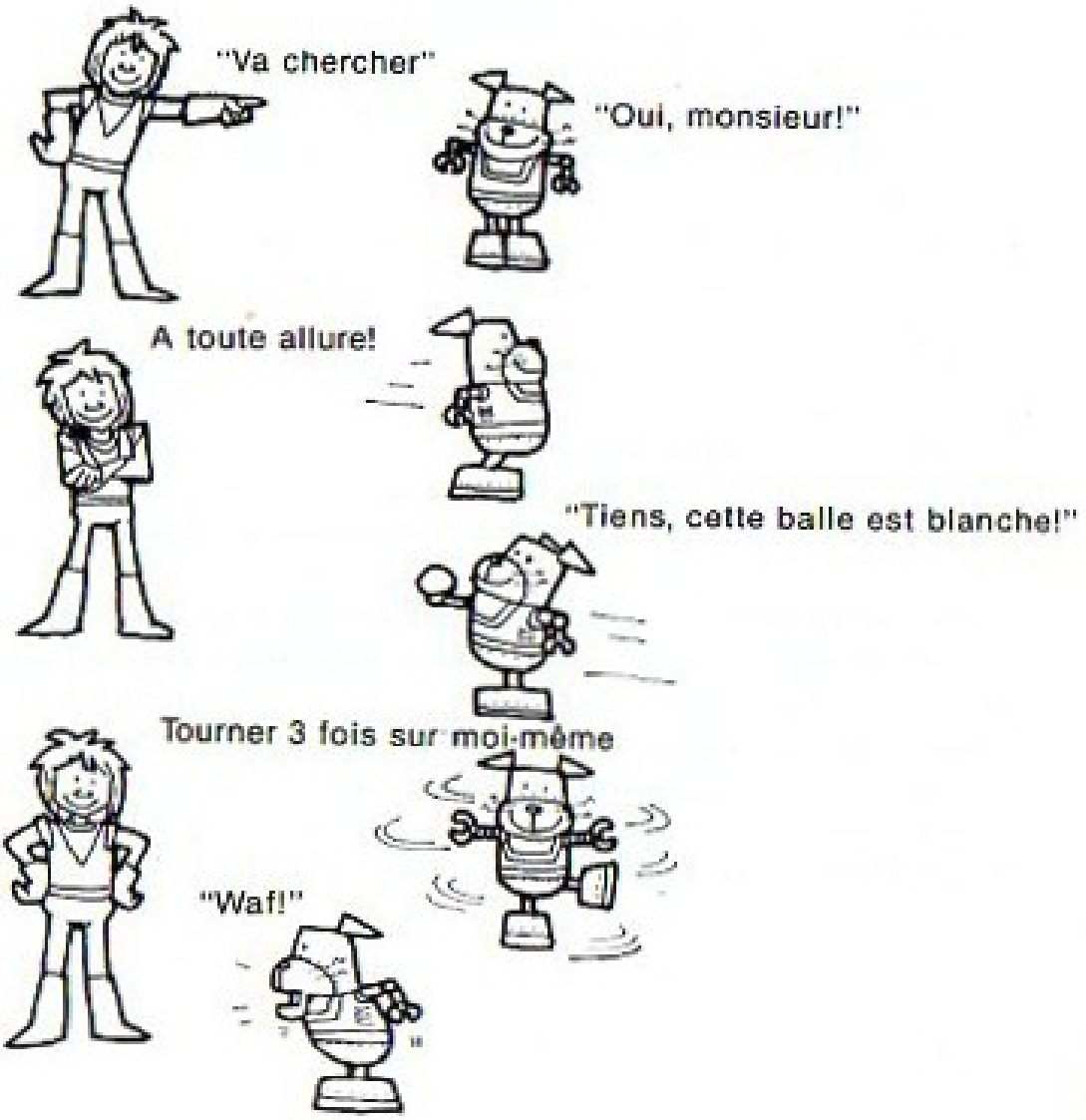
En outre, chaque démarche nécessite un calcul méticuleux: le chien doit savoir exactement combien de fois il doit tourner sur lui-même, ou exactement combien de secondes il doit rester couché, Et pour faire correctement ce qui lui est demandé, il doit savoir, à chaque instant et avec précision, combien de tours ou de secondes se sont déjà écoulées et combien il en reste avant d'aboyer ou de rapporter la balle. Ainsi, Super Milou effectue sans cesse des jugements, il prend constamment des décisions et se livre sans arrêt à des calculs pendant qu'il réalise ses prouesses.

Il nous faut signaler une autre différence importante entre Milou et Super Milou. Notre petit toutou familial est un animal vivant; comme il a de la cervelle, comme nous les humains, il peut faire certaines choses sans avoir reçu d'ordre de personne. Si Milou aperçoit un chat, il aboie; s'il voit un incendie, il se sauve. Mais Super Milou est une machine. S'il peut effectuer des choses complexes, il est incapable de "réfléchir" à leur sujet ou de se souvenir de ce qu'il doit faire dans certains cas. En effet, Super Milou ne fera que ce qu'il a été "programmé" pour faire et, de lui-même, il ne fera rien si un être humain ne lui a pas ordonné de remplir son rôle: des moteurs, des senseurs, des composants divers ou des microprocesseurs, selon une séquence bien déterminée.

Par conséquent, lorsque vous écrivez un programme pour Super Milou, vous devez penser à tout ce que la machine devra faire: **chaque** moindre démarche devra être détaillée et vous devrez lui attribuer une place dans la séquence. Un peu comme ceci:

1. Recevoir et identifier l'instruction "Trouver la balle et la rapporter!".
2. Utiliser des senseurs d'image pour observer et localiser la balle en mouvement.
3. Faire agir ses pattes mécaniques pour se diriger vers la balle.
4. Utiliser ses senseurs d'image pour savoir si la balle est "noire" ou "blanche".
5. Mouvoir ses membres pour saisir la balle et la ramener.
6. Choisir s'il y a lieu de "tourner sur lui-même" ou de "se coucher".
7. Mouvoir ses membres en fonction de l'action requise.
8. Calculer le nombre de tours ou de secondes passés, tout en effectuant son action.
9. Aboyer / Ne pas aboyer.
10. Arrêter tout mouvement lorsque l'action est terminée.
11. Se préparer pour recevoir l'instruction suivante du programme.

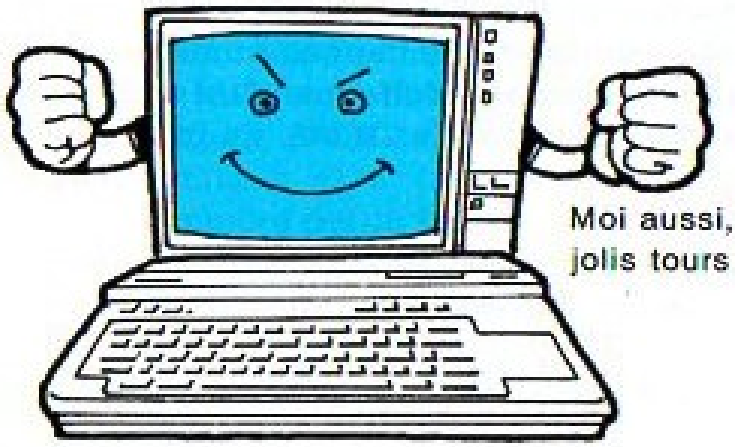
Vous devrez vérifier toutes ces démarches pour être sûr de ne rien avoir oublié, de ne pas avoir fait d'erreur, puis vous devrez les entrer dans le microprocesseur de Super Milou. Si vous avez pensé à tout et avez tout classé dans la séquence adéquate, Super Milou fera merveille chaque fois que vous lui demanderez de recommencer son tour d'adresse. Vous pouvez "faire tourner le programme" une fois, dix fois, ou dix mille fois: Super Milou fera toujours exactement ce qui lui a été demandé.



Ainsi, il existe deux règles essentielles dans l'écriture d'un programme: penser à **tout** ce que l'ordinateur doit faire pour finir exactement le travail que nous voulons lui confier, et classer toutes les démarches dans **l'ordre adéquat.** La seule "intelligence" particulière qui est requise de vous, c'est tout simplement un peu de réflexion logique parce que, comme les autres machines, les ordinateurs fonctionnent comme votre cerveau, c'est-à-dire logiquement.

Passons maintenant à la rédaction de votre premier programme pour l'ordinateur Sony et vous verrez combien, en définitive, tout ceci est bien simple.

ECRITURE D'UN PROGRAMME: SIMPLE COMME BONJOUR



Moi aussi, je fais de jolis tours d'adresse!

Dans un programme d'ordinateur, une "séquence" signifie seulement que les instructions sont écrites dans l'ordre un-deux-trois. Rappelez-vous le tour d'adresse de Super Milou: il avait trois parties séparées que nous avions fait précéder d'un chiffre.

1. Trouver la balle et la rapporter!
2. Si la balle est blanche, tourner trois fois sur soi-même et aboyer!
3. Si la balle est noire, se coucher pendant dix secondes!

Ces démarches se trouvent dans un ordre logique: nous voulons lui faire rapporter la balle avant de faire autre chose et c'est pourquoi l'instruction 1 doit venir en premier lieu (les instructions 2 et 3 pourraient être inversées sans changer le tour).

Nous allons entrer ces instructions dans l'ordinateur de la même façon; nous allons les écrire sur l'écran dans l'ordre précis où elles devront être exécutées et nous donnerons à chacune d'elles un numéro d'ordre. Notre premier "travail" pour l'ordinateur, notre programme, sera d'afficher les mots "JEAN ET MARIE" sur l'écran.

```
10 PRINT "JEAN"
20 PRINT "ET"
30 PRINT "MARIE"
```

Vous connaissez déjà l'instruction PRINT et, ici, nous allons l'utiliser trois fois: une fois pour chacun des trois mots que nous désirons imprimer. Bien entendu, notre séquence est la même que l'ordre des mots. La seule différence par rapport à notre manière précédente d'écrire les instructions PRINT, c'est que chacune commence par un nombre: 10, 20 et 30. Cette nuance est très importante car ces nombres disent à l'ordinateur **quelle instruction exécuter ensuite**, pendant qu'il exécute chaque démarche du programme. En d'autres mots, ils indiquent le chemin à suivre.

Tapez maintenant la première instruction: 10 PRINT "JEAN". Et n'oubliez pas d'appuyer une fois sur RETURN à la fin.

1 0 [ ] P R I N T [ ] " J E A N " RETURN

Qu'obtenons-nous sur l'écran?

```
10 PRINT "JEAN"
■
```

Parfait! Votre première instruction est entrée, le curseur s'est déplacé à la ligne suivante et vous êtes prêt pour entrer les deux instructions suivantes, exactement de la même façon.

```
10 PRINT "JEAN"
20 PRINT "ET"
30 PRINT "MARIE"
■
```

Et voilà votre programme... Il est presque prêt à être utilisé.

CORRECTION D'UN PROGRAMME: CHERCHER LA PETTITE BETE

Avant de "passer à l'exécution" de notre programme, il nous reste une chose très importante à faire: **confirmer** que chaque partie du programme a été entrée correctement. Rappelez-vous que votre ordinateur Sony n'est rien d'autre qu'une machine et qu'il ne peut lire que les commandes écrites en langage BASIC. Comme tous ses collègues, cet ordinateur est très pointilleux et il faut mettre les points sur les "i", pourrait-on dire! La moindre petite faute, un caractère ou un espace de trop ou manquant et le voilà bloqué car l'instruction ne ressemble plus, dans ce cas, à un mot du BASIC qu'il connaît. Ou bien tout le programme s'arrête, ou bien l'ordinateur fait ce qu'il ne doit pas. Ainsi, à titre d'exemple:

```
10 PRIMT "JEAN"
```

Erreur

En informatique, des erreurs de ce genre sont appelées "**bug**", même en français, c'est-à-dire un insecte ou une petite bête qui s'est glissée dans le programme. L'instruction PRINT doit être orthographiée correctement; faute de quoi l'ordinateur ne s'y retrouvra pas et ne comprendra pas vos désirs.

Si vous découvrez une erreur dans le programme à ce stade, estimez-vous heureux, car il vaut mieux la trouver maintenant que plus tard et sa correction est très simple.

Vous savez déjà comment déplacer le curseur à l'aide des quatre touches du clavier, marquées de petites flèches. Comme votre curseur se trouve au bas du programme, vous devez d'abord le faire remonter jusqu'à la ligne "10" en agissant sur la touche. Ensuite, vous déplacez le curseur de six positions vers la droite à l'aide de la touche qui convient, de sorte qu'il se trouve sur la lettre erronée, le M.

```
10 PRIMT "JEAN"
```

Amenez le curseur ici.

Pour corriger, il suffit de taper la lettre correcte.

```
10 PRINT "JEAN"
```

Appuyer sur la touche [N].

Appuyez une fois sur [RETURN]... et la petite bête est exterminée.

```
10 PRINT "JEAN"
20 PRINT "ET"
```

Le curseur revient ici après la poussée sur [RETURN].

Si vous décelez d'autres erreurs, corrigez-les en déplaçant le curseur de la même façon. (Au besoin, vous consulterez le **mode d'emploi** pour vous rafraîchir la mémoire sur la manière d'ajouter un caractère ou un espace oublié, ou d'en retrancher un superflu.) Une fois corrigées toutes les erreurs, servez-vous des touches marquées par des flèches pour ramener le curseur au bas de l'écran qui doit alors ressembler à ceci:

```
10 PRINT "JEAN"
20 PRINT "ET"
30 PRINT "MARIE"
■
```

Vous n'aviez probablement pas fait de faute dans l'écriture de ce programme simple, mais il est quand même très important de **vérifier** chaque étape. Bientôt, vos programmes se composeront de nombreuses instructions dont beaucoup seront plus longues que celles-ci. Or, la moindre "petite bête" peut bloquer tout un programme et n'importe qui, même les informaticiens chez Sony, peut être distrait et faire des fautes d'orthographe ou de frappe.

Quoi qu'il en soit, une chose est certaine: si vous avez fait une erreur, votre ordinateur Sony vous le dira, tôt ou tard. Un **message d'erreur** apparaîra sur l'écran, ou le programme sera interrompu avant d'être achevé, ou encore l'ordinateur fera quelque chose qui n'était pas dans vos intentions. Vous aurez alors à revoir toute la liste des instructions, à localiser la faute (la "bug") et à la corriger, un travail qui peut être très fastidieux. Par conséquent, il est très vivement conseillé de "débuguer" (rechercher la petite bête) avant qu'elle n'ait fait des dégâts dans le programme, c'est-à-dire au stade de l'écriture des instructions.

Règles d'or de la programmation

- **Elaborer dans son esprit chaque étape du programme nécessaire** pour que l'ordinateur accomplisse ce qui est envisagé.
- Prendre soin de lister toutes les instructions **dans l'ordre logique** qui conduira au résultat recherché.
- Au début de chaque ligne d'instruction, placer un **numéro d'ordre.**
- Après l'entrée des instructions, **vérifier soigneusement chacune d'elles** et, le cas échéant, corriger toute erreur ("bug") qui s'y trouverait.

EXECUTION D'UN PROGRAMME

Vous avez entré les instrutions et apporté les corrections éventuelles; il vous faut maintenant **exécuter** le programme et c'est la fonction de l'instruction **RUN** que nous introduisons ici sur l'écran:

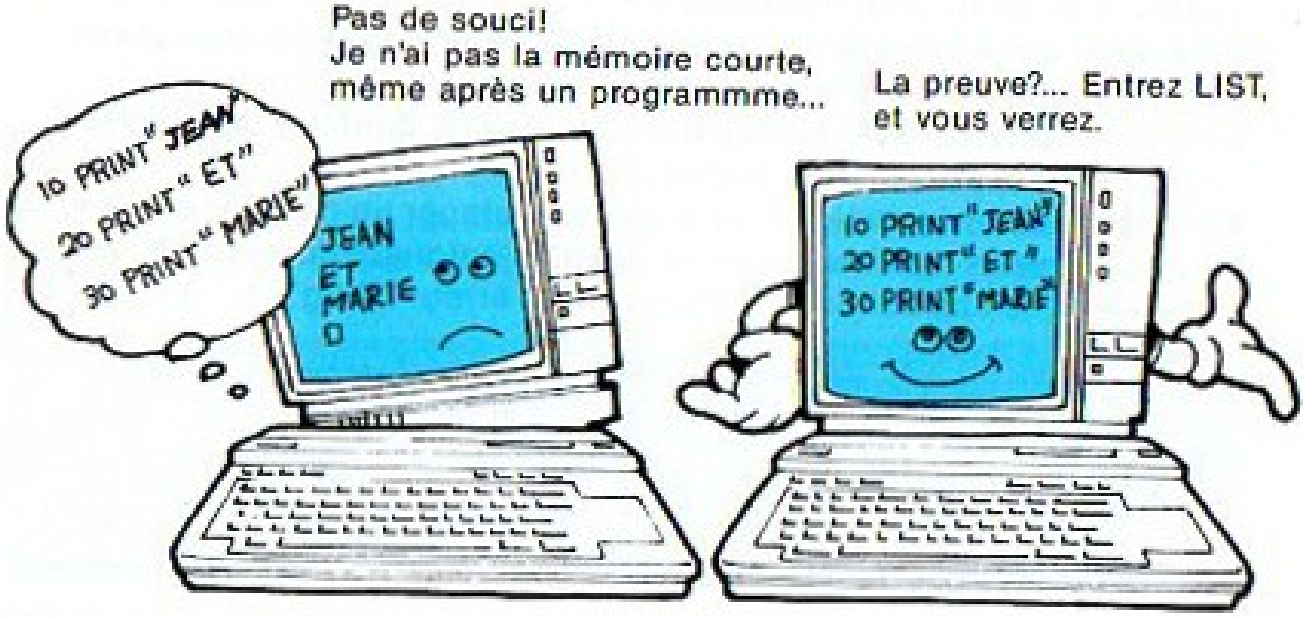
```
RUN
```

Et voici ce que l'ordinateur fait de votre programme lorsque vous l'avez entré **et avez appuyé** sur [RETURN]. Votre écran doit afficher ce qui suit:

```
10 PRINT "JEAN"
20 PRINT "ET"
30 PRINT "MARIE"
RUN
JEAN
ET
MARIE
Ok
■
```

Tapez à nouveau l'instruction RUN, juste après le curseur, et appuyez une nouvelle fois sur [RETURN]. Le résultat est le même, n'est-ce pas? Le résultat de **ce** programme sera toujours le même, à moins que vous ne changiez ou effaciez les commandes. Essayez-le plusieurs fois et voyez vous-même. JEAN ET MARIE réapparaissent chaque fois, ce qui veut dire que votre ordinateur Sony **se souvient** du programme.

Vous pouvez d'ailleurs vérifier la mémoire à tout moment en entrant **LIST** (suivi de [RETURN]). Désormais, nous utiliserons **l'instruction LIST** chaque fois que nous écrirons un programe.



Un des aspects les plus précieux de votre ordinateur Sony, c'est qu'il a une bonne mémoire et se souviendra de vos instructions. Il vous suffit de le lui demander en plaçant un **numéro devant l'instruction.**

```
10 PRINT "JEAN"
```

Si un numéro de ligne est entré, il s'agit d'un programme.

Cette instruction est précédée d'un numéro et l'ordinateur sait, par conséquent, qu'elle fait partie d'un programme et qu'il doit la mémoriser. 10 est un **"numéro de ligne"**. Comme numéro de ligne, vous disposez d'un choix considerable, allant de 0 à 65529.

Pendant combien de temps votre ordinateur se souviendra-t-il des instructions de vos programmes? Eternellement, pourrait-on dire; ou plus exactement, jusqu'à ce que vous lui disiez de les oublier. A cet effet, l'instruction à utiliser est:

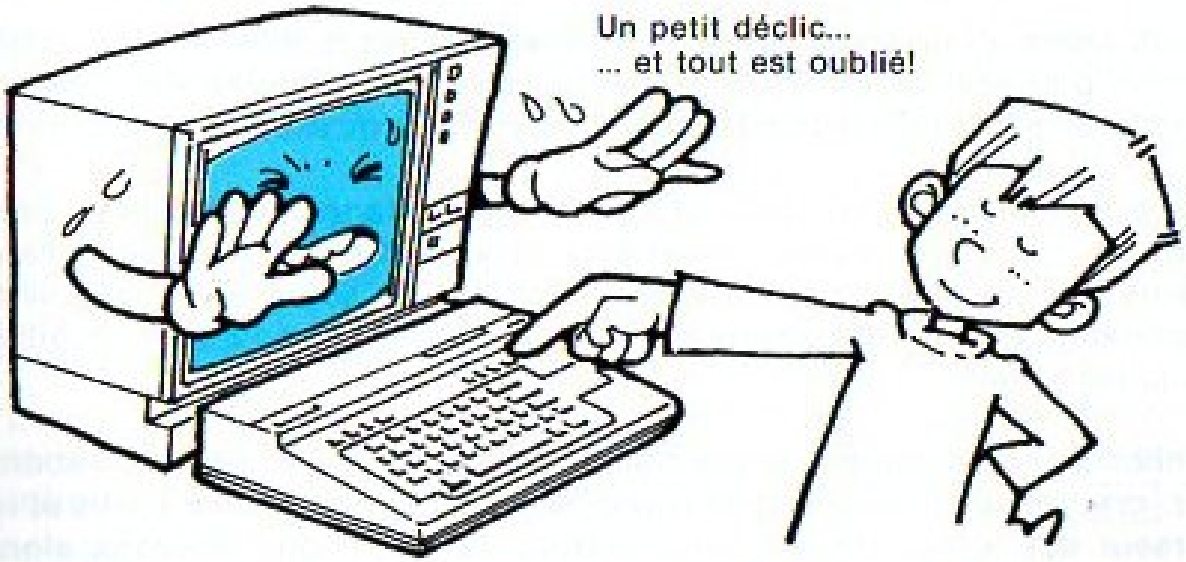
NEW

Vérifions si l'ordinateur garde bien en mémoire les données qui lui sont confiées en entrant quelques instructions numérotées et en tapant LIST. Est-ce que l'ordinateur les a bien listées? Oui? Parfait. Maintenant, entrons NEW.

```
LIST
10 PRINT "JEAN"
20 PRINT "ET"
30 PRINT "MARIE"
Ok
NEW
Ok
■
```

Les instructions apparaissent encore sur l'écran, mais l'ordinateur les a oubliées. Essayez LIST une nouvelle fois: le programme a disparu de la mémoire.

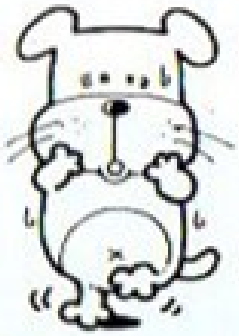
Il existe deux manières de faire oublier les instructions par l'ordinateur. La première consiste à appuyer sur le bouton RESET et l'autre, à couper l'alimentation de l'appareil. **Mais n'ayez pas recours à ces deux extrémités** sauf si vous êtes sûr et certain que l'ordinateur peut oublier ce qu'il y a dans sa mémoire.



Dans quelque temps, nous apprendrons comment **sauvegarder** les programmes, de manière à pouvoir les utiliser à nouveau plusieurs semaines, voire des années plus tard, grâce à un magnétophone. D'ici là, votre ordinateur oubliera tout lorsque vous le mettrez hors tension.

LES GRAPHIQUES: UNE CONSTELLATION D'ETOILES

Que faire pour...



- Utiliser plus de mots du BASIC
- Vider l'écran
- Tracer des points et des lignes
- Faire répéter une action par l'ordinateur
- Utiliser une variable
- Utiliser un nombre aléatoire
- Utiliser de nouvelles instructions de couleur

UN PROGRAMME POUR GRAPHIQUE

Voyons quelques nouveaux termes en BASIC et apprenons à quoi ils servent. Tapez ces instructions:

```
10 SCREEN 2
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
60 GOTO 20
```

Nous avons ici un **programme**, une série d'étapes que l'ordinateur peut utiliser pour réaliser quelque chose. Puisque ce programme se trouve en mémoire, donnez l'instruction RUN et voyons ce qu'exécute la machine.

Toutes les lettres ont disparu et quatre points blancs se trouvent à présent sur l'écran. Le curseur n'est plus là et rien n'apparaît sur l'écran, même si vous appuyez sur une des touches. Que se passe-t-il? Et bien, l'ordinateur est en train d'exécuter le programme. Mais pourquoi travaille-t-il si lentement?

Tâchons de comprendre ce que signifie le programme. Appuyons d'abord sur CTRL et STOP pour arrêter le programme. Les points sont partis et le curseur est revenu. Maintenant, entrons LIST et nous pouvons alors analyser le programme, ligne par ligne. Observons d'abord la ligne 10.

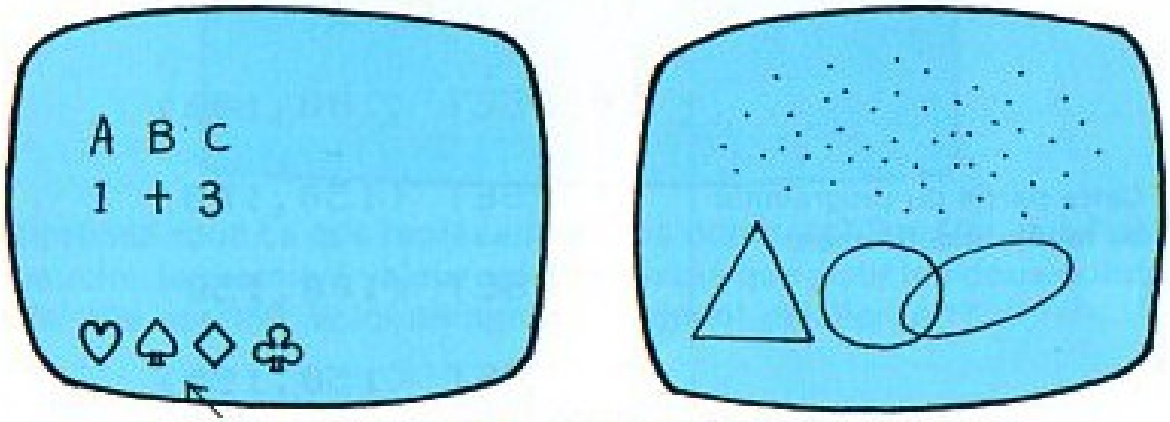
SCREEN 2

est l'instruction qui vous permet de commencer à tracer des **graphiques** (dessins) sur l'écran. Par graphiques, on entend des points, des lignes, des cercles et d'autres figures ou motifs qui sont affichés sur l'écran. (Les informations autres que les graphiques, comme les nombres, les lettres et les symboles, sont appelées des **caractères.**)

Les instructions SCREEN disent à l'ordinateur sous quelle forme l'information sera affichée à l'écran. Il existe, en fait, deux sortes d'instructions SCREEN.

SCREEN 0 ou SCREEN 1 servent pour les **caractères**

SCREEN 2 est destiné aux **graphiques**



(Les symboles sur le clavier sont des caractères.)

Ainsi donc, SCREEN 2 fait afficher des graphiques sur l'écran. Lorsque vous avez donné l'instruction **RUN**, les caractères ont disparu, précisément parce que SCREEN 2 avait donné ordre à la machine d'afficher des graphiques. En appuyant sur CTRL et STOP, vous pouvez annuler l'instruction des graphiques et les caractères réapparaissent. La première ligne du programme étant clarifiéedans votre esprit, nous pouvons passer à la suivante.

```
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
```

Un peu d'explication ne nuira pas: PSET est l'instruction qui permet de placer des **points** sur l'écran et les chiffres entre parenthèses ( ) disent à l'ordinateur où ils doivent être placés.

Se répéter, se répéter et toujours se répéter

La dernière ligne du programme présente une instruction très importante:

```
60 GOTO 20
```

GOTO signifie "aller à" et cette instruction demande donc à l'ordinateur de **retourner** à la ligne 20. Quand il repasse à la ligne 20,

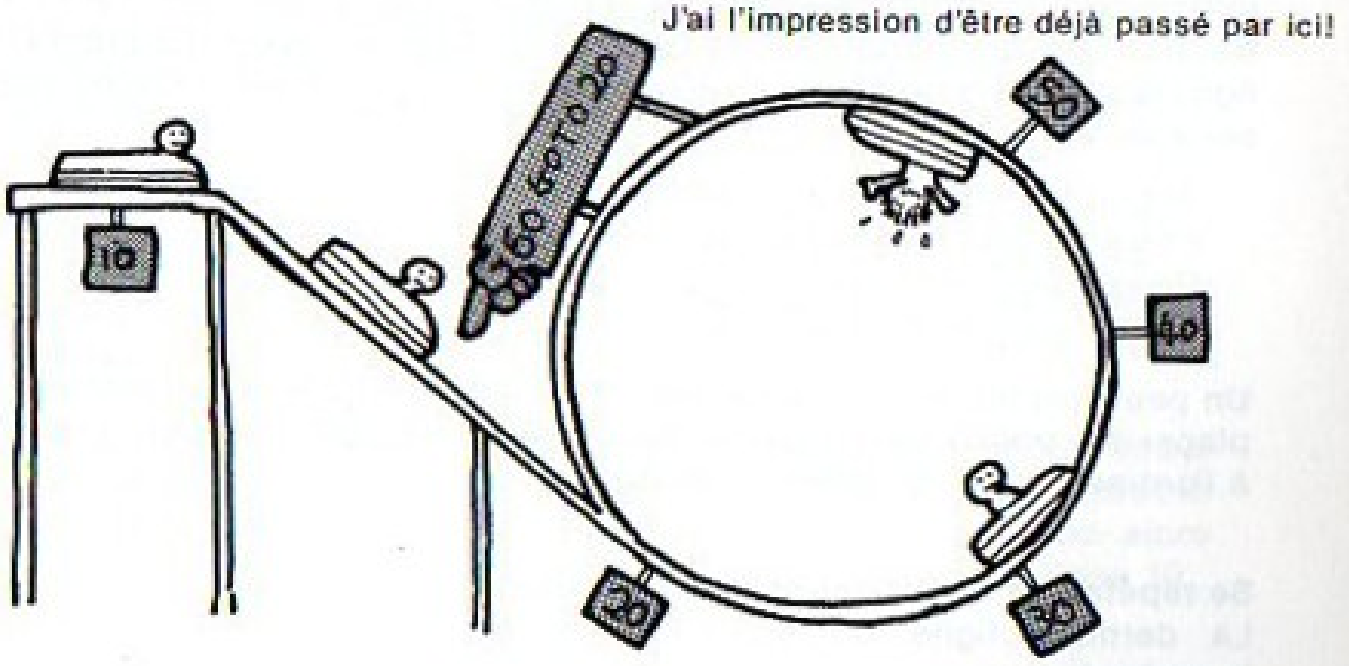
```
20 PSET (100,100)
```

tout naturellement, l'ordinateur répète le programme. A la ligne 60, il revient à la ligne 20, répète le programme, revient à la ligne 20, répète le programme...et ainsi de suite.

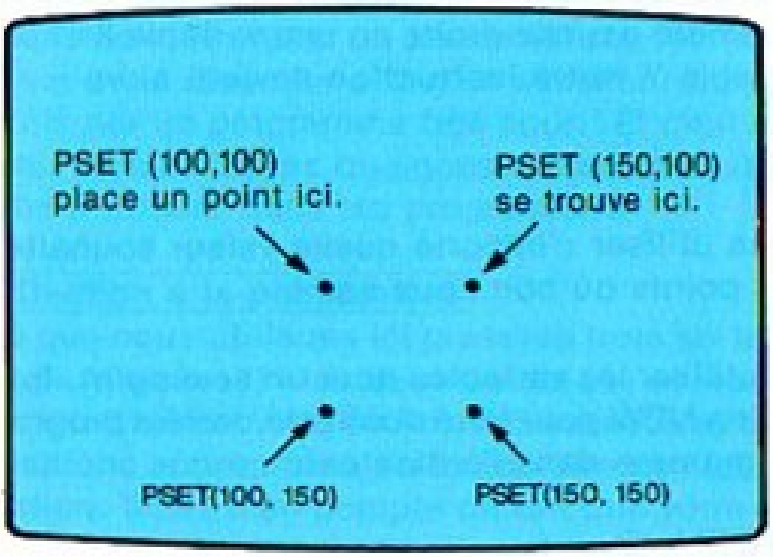
Cette partie du programme se répète indéfiniment.

```
10 SCREEN 2
20 PSET (100,100)
30 PSET (150,100)
40 PSET (100,150)
50 PSET (150,150)
60 GOTO 20
```

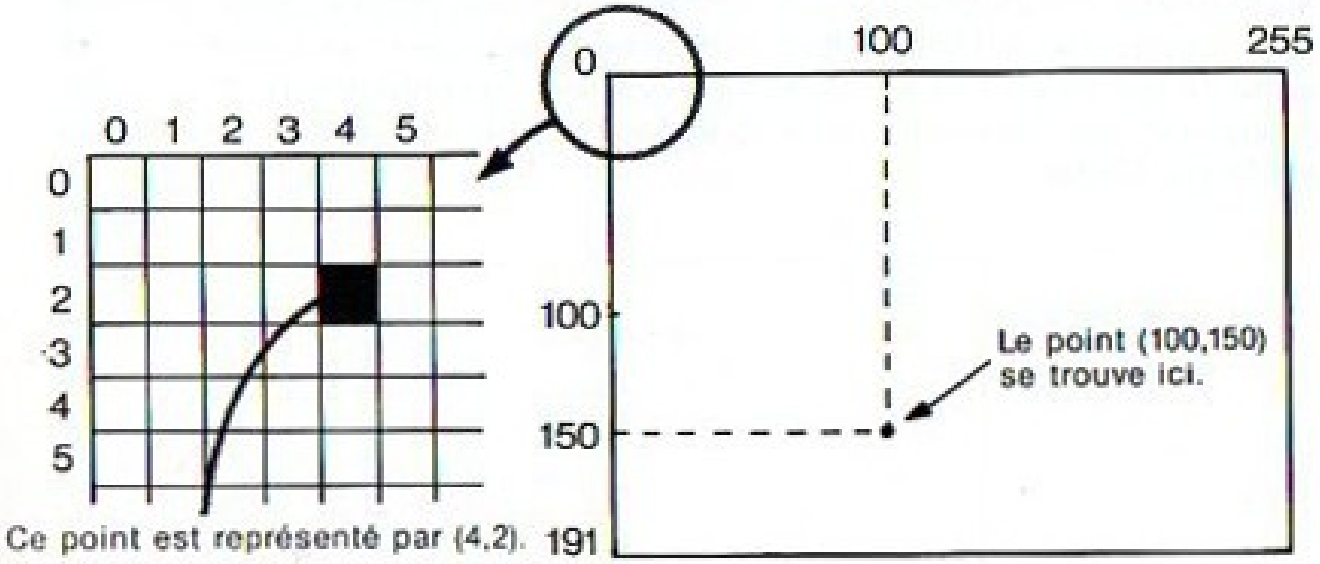
Dans un programme, une section qui se répète porte le nom de **boucle** et il s'agit d'un outil ordinaire et bien pratique en programmation (heureusement, toutes les boucles ne se répètent pas à l'infini!).



Voyons à présent comment ces instructions PSET placent des points sur l'écran.



Comprenez-vous ce que représentent ces nombres? Pour les deux points de gauche, la première valeur est 100, tandis que pour les deux points de droite, elle est 150. Voici un agrandissement de l'écran.



Pour dire à l'ordinateur où il doit placer un point, vous devez lui fournir ses deux coordonnées c.à.d. ses deux numéros de position, le premier indiquant la position dans le sens latéral gauche-droite et le second, dans le sens vertical du haut vers le bas. Sur l'écran, il existe 256 positions différentes de la gauche vers la droite (de 0 à 255) et 192 du haut en bas (de 0 à 191). Par conséquent, il est possible de tracer un point n'importe où sur l'écran, entre (0,0) et (255,191).

Les graphiques et les variables

Vous vous souvenez certainement que tout nombre peut être remplacé par une variable. Ceci se vérifie aussi pour les nombres de PSET. Si nous changeons le numéro gauche-droite en une variable X et le numéro haut-bas en une variable Y, notre instruction devient alors

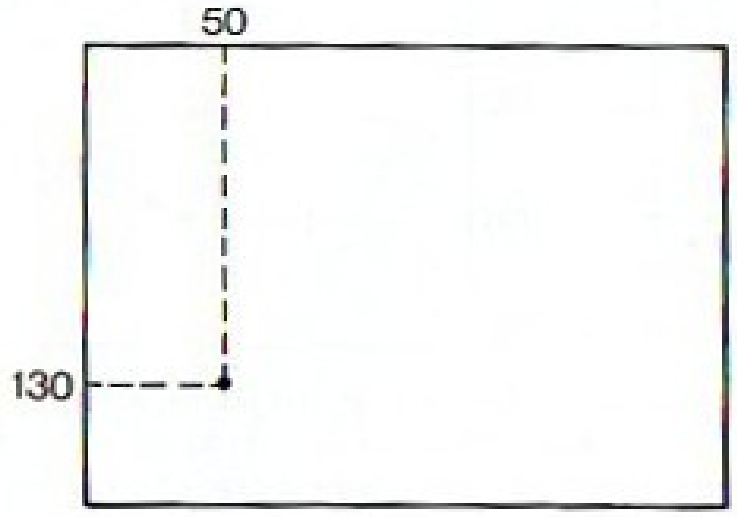
```
PSET (X,Y)
```

et nous pouvons utiliser n'importe quelle valeur souhaitée pour X et Y pour tracer des points où bon nous semble.

Voici comment utiliser les variables pour un seul point. Tout d'abord, utilisons l'instruction NEW pour faire oublier le dernier programme et entrer ce nouveau programme dans l'ordinateur:

```
10 SCREEN 2
20 X=50:Y=130
30 PSET (X,Y)
40 GOTO 30
```

Mémorisons ce programme par une poussée sur RETURN, puis donnons l'instruction RUN. L'ordinateur commence alors par l'instruction de la ligne 10 qui vide l'écran pour faire place aux **graphiques**. A la ligne 20, il apprend deux variables, et à la ligne 30, il les utitlise pour placer un point sur l'écran.



Etant donné que X est 50 et Y est 130, un point blanc apparaît sur l'écran à la position (50, 130). Ensuite, l'ordinateur prend connaissance de la ligne 40 qui le renvoie à la ligne 30. Cela signifie que le seul moyen d'arrêter cette boucle est d'actionner les touches CTRL et STOP.

Pourquoi, direz-vous, avons-nous fait appel à des variables au lieu d'écrire simplement PSET (50, 130)? Et pourquoi fallait-il recommencer indéfiniment à partir de la ligne 40? Certes, ces démarches ne sont pas requises si nous désirons placer un seul point sur l'écran pour un instant seulement. Mais nous disposons là d'un excellent moyen d'afficher de nombreux points par un programme très court. Et c'est d'ailleurs ce que nous allons voir ensuite, après quelques remarques sur la ponctuation et l'orthographe à utiliser dans les programmes.

Remarque: Attention à la ponctuation

Le programme que nous utilisons ici présente trois sortes différentes de **ponctuation**: une virgule (,) , les deux points (:) et les parenthèses ( ). Ces signes de ponctuation sont, en fait, des **caractères** et **ils doivent être placés aux positions appropriées**, au même titre que les lettres dans les mots d'instruction. Tenez bien compte du fait que votre ordinateur Sony ne pourra pas comprendre une instruction si une faute d'orthographe s'y est glissée et que la ponctuation est une partie de l'orthographe.

En BASIC, chaque signe de ponctuation a sa propre signification. La virgule indique qu'un nombre est terminé et qu'un autre va suivre. Les deux points signifient la fin d'une instruction et qu'il y aura une autre instruction sur la même ligne. Enfin, les parenthèses sont nécessaires quand deux ou plusieurs nombres, ou deux ou plusieurs variables sont utilisés ensemble; ces parenthèses sont **toujours par paire**, tout d'abord celle de gauche "(", puis celle de droite ")". Plus loin, dans ce livre, nous rencontrerons certains autres signes de ponctuation, utilisés dans les instructions en langage BASIC.

Comme nous l'avons déjà dit, il arrive à tout le monde de faire des erreurs d'orthographe. Or, quand une de ces erreurs a été commise, il est souvent impossible, pour l'ordinateur, de comprendre l'instruction et il ne lui reste plus qu'à afficher un message d'erreur. Ainsi par exemple,

```
Syntax error in 30
```

signifie qu'il y a une faute d'orthographe à la ligne 30.

NOMBRES ALEATOIRES

Utilisez l'instruction NEW pour vider la mémoire de l'ordinateur, puis entrez-y ce programme:

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)
50 GOTO 20
```

Lorsque vous passez à l'exécution par l'instruction RUN, l'écran se remplit progressivement de points, jusqu'à ce que vous l'arrêtiez... vous l'aviez deviné, par les touches CTRL et STOP.

Que signifient tous ces points sur mon visage?

Donnons quelques mots d'explication à propos des deux nouvelles lignes du programme.

```
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
```

Il est facile de voir que ces lignes donnent des valeurs aux variables X et Y. Mais quel est le sens de ces valeurs? Tout d'abord, considérons RND(1). RND est une abréviation du mot anglais **random** qui veut dire "au hasard" ou "aléatoire" et c'est une des **fonctions** en BASIC. Pour mieux comprendre cette **fonction de nombre aléatoire,** essayez ceci:

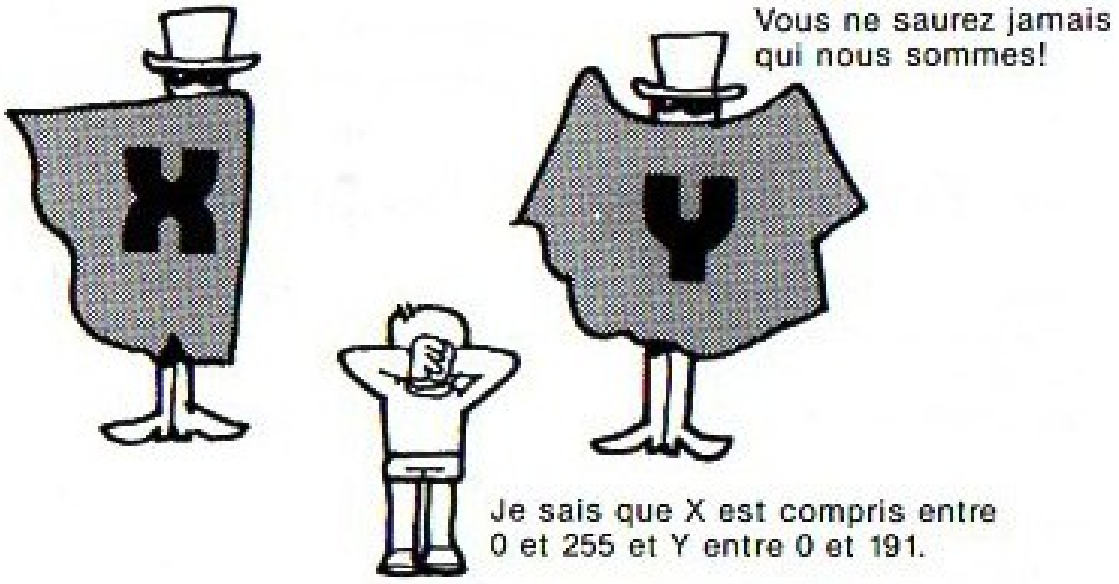
```
X=RND(1):PRINT X
```

Appuyez sur RETURN après avoir entré cette instruction: vous verrez apparaître un nombre à 14 décimales, inférieur à 1, mais supérieur à zéro et, bien sûr, précédé d'un **point décimal** (représentant la virgule). Par exemple:

```
X=RND(1):PRINT X
 .5952194399462 3
Ok
■
```

Essayons encore une fois. Le nombre a-t-il changé? Faisons un nouvel essai. Votre ordinateur dispose d'une longue liste de nombres aléatoires et, l'un après l'autre, il en choisit un à utiliser pour X.

Revenons à la ligne 20 du programme où nous avons (RND(1) * 256), ce qui signifie que le nombre aléatoire est multiplié par 256. Si nous multiplions ces fractions par 256, les résultats contiendront également des fractions. Or, nos positions pour un "point" sur l'écran doivent être des nombres **entiers**, tels que 1 ou 30 ou 192, parce que les numéros de position sur l'écran ont été fixés uniquement en nombres entiers. C'est pourquoi l'instruction est précédée de **INT** qui est également une abréviation du mot anglais "integer" qui signifie un **nombre entier.** Cette instruction a pour but de supprimer le point décimal (la virgule) et les chiffres qui le suivent pour transformer le résultat en un nombre entier. Ainsi, nous pouvons comprendre la valeur de X qui change chaque fois que le programme accomplit sa boucle. Chaque fois que l'ordinateur lit X=INT(RND(1) * 256), il arrondit X à un nombre entier, compris entre 0 et 255, mais vous ne savez jamais quel nombre ce sera. Et, de la même manière, il fait de Y un nombre entier chaque fois qu'il lit la ligne 40.



Nous comprenons déjà la ligne suivante,

```
40 PSET (X,Y)
```

même si nous ne savons pas ce que seront les valeurs de X et Y. L'ordinateur ne nous révèle pas ces valeurs parce qu'il n'y a pas d'instruction PRINT. En revanche, il continue d'afficher de nouveaux points quelque part sur l'écran, aux endroits où les nombres entiers gauche-droite et haut-bas lui disent d'en placer.

Jusqu'à présent, nous nous sommes limités à expliquer comment un point apparaît sur l'écran. D'où viennent les autres? C'est ici, précisément, qu'intervient la "boucle". La ligne suivante,

```
50 GOTO 20
```

fait tourner la programme et le ramène à

```
20 X=INT(RND(1)*256)
```

De nouveaux nombres sont ainsi donnés chaque fois à X et à Y, de sorte que

```
40 PSET (X,Y)
```

place un point à un nouvel endroit. De cette façon, des points apparaissent les uns après les autres sur l'écran.

Les programmes: deux sortes d'instructions

Comme nous l'avons dit, un programme est **une série d'instructions dans un ordre fixe, numéroté,** série entrée dans la mémoire de l'ordinateur. Un programme peut comporter des centaines, voire des milliers de lignes ou d'instructions mais, dans la plupart des cas, il n'est pas nécessaire de s'encombrer à ce point. Ne venons-nous pas d'afficher des milliers de points sur l'écran avec un programme de 5 lignes seulement?

En fait, une de ces cinq lignes d'instruction, à savoir GOTO 20, n'a rien fait faire de concret à l'ordinateur, mais elle lui a dit **où aller ensuite**. C'est cette ligne qui a permis à notre court programme de faire tant de choses, grâce à une boucle qui se répète indéfiniment.

Nous allons voir à présent qu'il existe deux sortes d'instructions:

1. **Instructions disant à l'ordinateur ce qu'il doit faire;** c'est le cas de:

PRINT PSET WIDTH COLOR etc.

et

2. **Instructions changeant l'ordre d'un programme,** comme

GOTO IF—THEN FOR—NEXT etc...

Les instructions du second groupe sont peu nombreuses et nous les étudierons plus en détail dans les chapitres suivants. Elles sont, en fait, des outils extrêmement pratiques et, comme GOTO que nous venons d'utiliser, elles servent à rendre **très puissants** des programmes **très réduits.**

UN PEU DE COULEUR SUR LES PROGRAMMES ——

Pourquoi ne pas rendre notre programme des points un peu plus agréable, en y ajoutant un peu de couleur? Pour commencer, affichons le programme qui se trouve dans la mémoire de l'ordinateur à l'aide de l'instruction LIST (si vous avez débranché votre machine, vous devrez, au préalable, entrer à nouveau le programme).

```
LIST
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)
50 GOTO 20
```

A ce stade, changeons la ligne 40 en: 40 PSET (X,Y), 2

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y)■
50 GOTO 20
```

Déplacez le curseur à cette position et entrez ,2 (n'oubliez pas la virgule qui a son importance, vous le savez!). Appuyez ensuite une fois sur RETURN pour effectuer l'entrée et ramener de nouveau le curseur à la fin du programme. Maintenant, donnez l'instruction RUN.

Les points blancs ont changé en vert moyen, parce que 2 est le numéro de code pour cette couleur. En vous référant au Tableau des couleurs de la page 14, vous pourrez choisir entre diverses couleurs, comme un peintre sur sa palette. Mais vous pouvez aussi faire d'une couleur une variable, de sorte que l'ordinateur change les points d'une couleur en une autre.

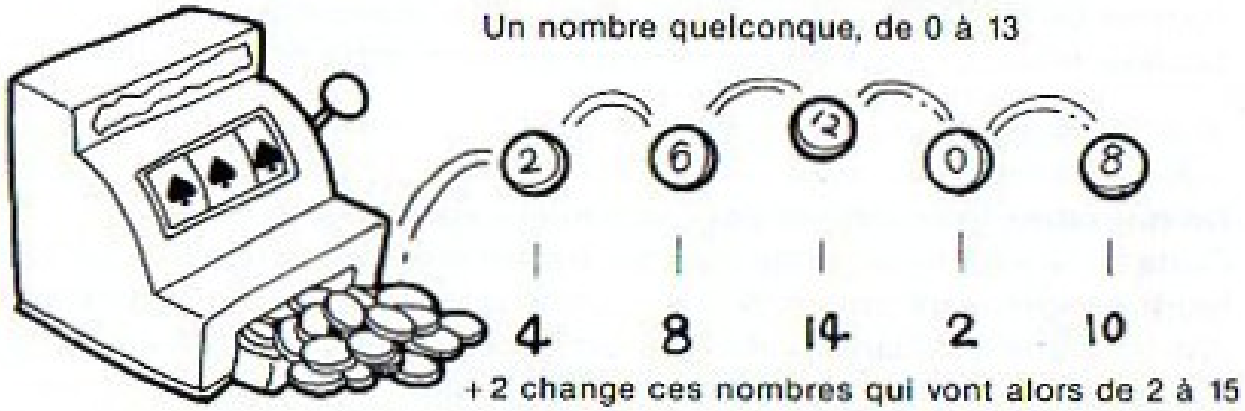
Revenons à nos moutons, ou plutôt à la ligne 40 et changeons le code de couleur en la variable C.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y),C
50 GOTO 20
```

A présent, nous devons donner une valeur à C, qui est notre instruction pour la couleur que nous souhaitons. Mais, ce serait bien plus drôle si, de nouveau, nous laissions à l'ordinateur le soin de donner cette valeur, au hasard. Voici l'instruction qui remplira cette tâche, mais ne l'entrez pas encore maintenant.

```
35 C=INT(RND(1)*14)+2
```

Que signifie cette ligne? Il s'agit d'une instruction de nombre aléatoire, comme celle de X et Y, avec une donnée supplémentaire à la fin. Elle choisit un nombre aléatoire (au hasard) entre 0 et 13, puis elle y ajoute 2, d'où (+2). Cela revient à dire que le nombre est compris entre 2 et 15; mais la palette possède 16 couleurs, de 0 à 15. Pourquoi cette instruction laisse-t-elle tomber les couleurs 0 et 1? Pour la bonne raison que 0 correspond à "transparent" et qu'il serait bien difficile de voir des points transparents, tandis que 1 correspond à "noir" et que des points noirs sont invisibles si le fond de l'écran devient noir également.



Où devons-nous placer cette ligne dans le programme? Elle doit évidemment se trouver avant que les points ne soient affichés à la ligne 40 et c'est pourquoi notre nouvelle ligne aura le numéro 35 (c'est d'ailleurs la raison pour laquelle nos lignes d'instructions portent les numéros 10, 20, 30... ce qui nous laisse des espaces pour en insérer de nouvelles, par la suite, le cas échéant). A présent, introduisons la nouvelle ligne dans notre programme.

```
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
40 PSET (X,Y),C
50 GOTO 20
35 C=INT(RND(1)*14)+2
■
```

30, 40, 50, 35. Cela n'est-il pas un peu confus, pour l'ordinateur? Pas le moins du monde, car il s'y connaît en chiffres et il a vite fait d'arranger les lignes dans l'ordre numérique. Vous voulez le vérifier? Qu'à cela ne tienne! Entrez simplement l'instruction LIST à nouveau et voyez le résultat:

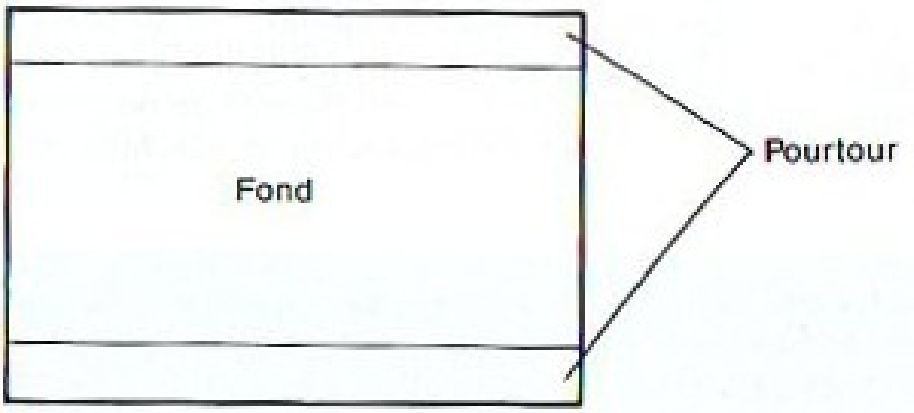
```
LIST
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
35 C=INT(RND(1)*14)+2
40 PSET (X,Y),C
50 GOTO 20
```

Désormais, tout est prêt pour transformer les points en **étoiles**, mais comme celles-ci ne sont visibles que la nuit, notre écran devra être de couleur noire. Et c'est pourquoi nous entrons cette nouvelle ligne:

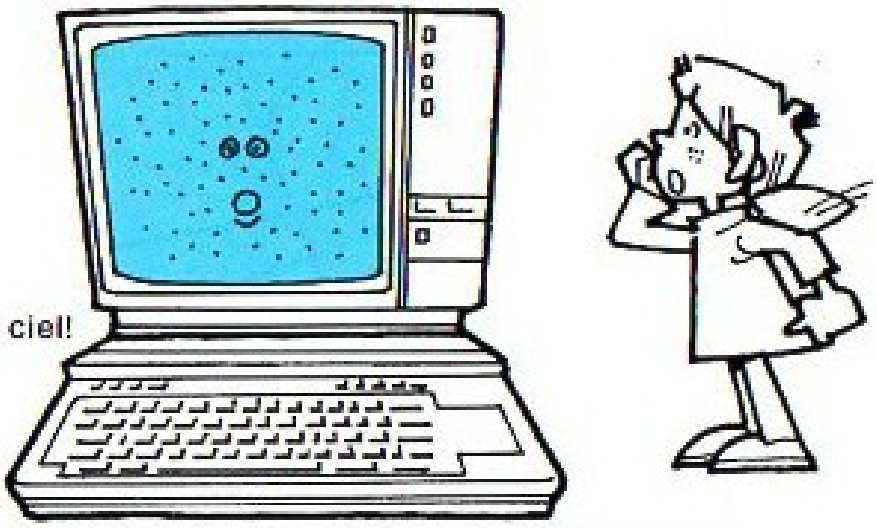
```
5 COLOR 15,1,1
```

De nouvelles instructions pour la couleur et les lignes

Cette ligne est une nouvelle manière d'utilisser l'instruction COLOR (couleur). Elle présente trois codes: le 15 détermine la couleur des **caractères** (ou la couleur de l'avant-plan), le premier 1 donne la couleur du fond (l'arrière-plan de l'écran), tandis que le dernier 1 décide la couleur du **pourtour** (dans ce cas, le fond et le pourtour ont la même couleur. N'importe quel chiffre peut convenir, mais essayons ceux-ci).

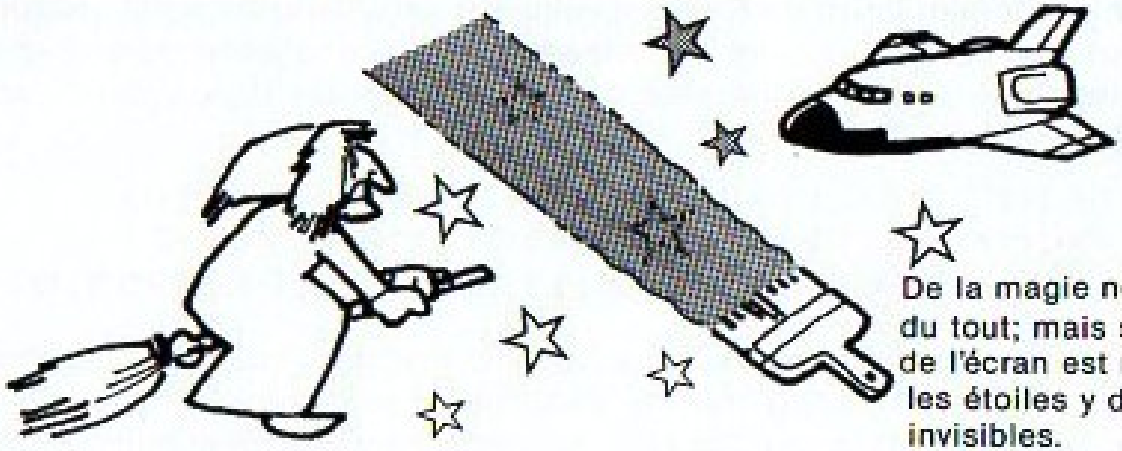


Maintenant, nous sommes parés: alors passons à RUN.



Que d'étoiles dans le ciel!

Mais ce n'est pas fini et nous pouvons rendre ces étoiles encore plus naturelles. Tout d'abord, il conviendrait de contrôler le nombre des étoiles que nous créons, afin de ne pas être encombré par toute une galaxie. Comme d'habitude, nous utilisons pour cela CTRL et STOP.



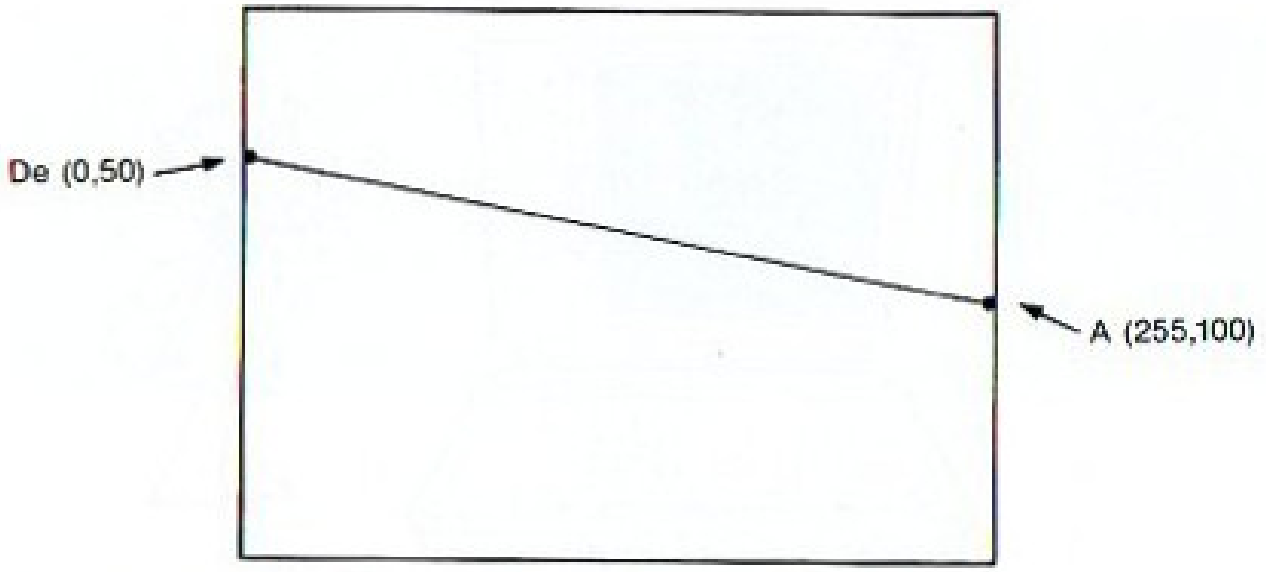
De la magie noire? Pas du tout; mais si une partie de l'écran est rendue noire, les étoiles y deviennent invisibles.

Bien entendu, tracer des **lignes noires** sur l'écran est une manière de rendre noire une partie de l'écran. L'instruction PSET est réservée aux points; pour les lignes, nous utilisons LINE. Voici donc une instruction qui dit à l'ordinateur où il doit tracer une ligne et quelle couleur lui donner.

```
LINE(0,50)-(255,100),2
```

Soit dit en passant, pour le [-] (le trait d'union) entre les deux valeurs, utilisez le signe moins.

Le trait d'union signifie "de/à" et cette instruction trace une droite **de** la position (0,50) **à** la position (255,100). Vous savez, bien sûr, que la dernière partie de cette instruction correspond à la couleur vert moyen. Si vous utilisez cette instruction LINE dans votre programme, vous obtiendrez une droite verte, comme celle-ci:



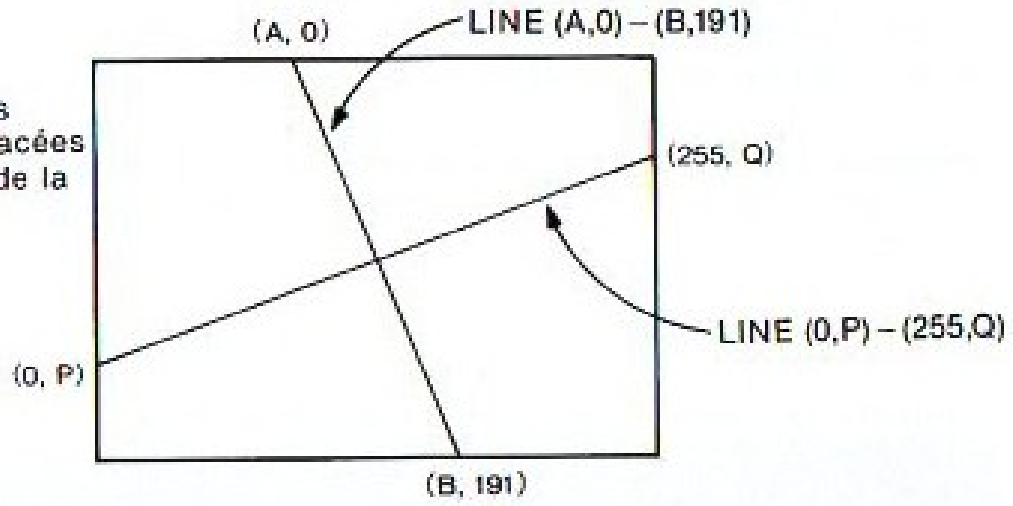
Maintenant que vous comprenez l'instruction LINE, vous allez pouvoir l'utiliser pour améliorer votre programme à étoiles. Nous souhaitons utiliser une droite noire, pour rendre certaines étoiles invisibles, n'est-ce pas? Cela signifie que nous devrons changer le code de couleur en 1. En outre, nous allons utiliser de nouveau les variables, de sorte qu'apparaisse une série de droites au hasard ainsi qu'une série de points au hasard. A cet effet, introduisez ces trois nouvelles lignes dans l'ordinateur:

```
42 A=INT(RND(1)*256):B=INT(RND(1)*256)
44 P=INT(RND(1)*192):Q=INT(RND(1)*192)
46 LINE (A,0)-(B,191),1:LINE (0,P)-(255,Q),1
```

Aucune de ces trois instructions ne peut prendre place sur une seule ligne de l'écran, mais peu importe. L'ordinateur se charge lui-même de les écrire sur deux lignes qui, pour lui, ne forment qu'**une ligne** puisque chacune n'a qu'**un numéro de ligne** (42, 44 ou 46).

La ligne numéro 46 comprend deux instructions LINE, utilisant quatre nouvelles variables. Les lignes 42 et 44 sont les instructions de nombres aléatoires que nous connaissons déjà et elles affectent des valeurs aux nouvelles variables A, B, P et Q. Voici comment apparaîtra une paire de droites noires aléatoires:

Les droites noires invisibles sont tracées par l'instruction de la ligne 46.



Chaque fois que vous introduisez les nouvelles lignes d'instruction 42, 44 et 46 dans l'ordinateur, n'oubliez pas RETURN, puis entrez LIST pour voir l'intégralité du programme.

```
5 COLOR 15,1,1
10 SCREEN 2
20 X=INT(RND(1)*256)
30 Y=INT(RND(1)*192)
35 C=INT(RND(1)*14)+2
40 PSET (X,Y),C
42 A=INT(RND(1)*256):B=INT(RND(1)*256
)
44 P=INT(RND(1)*192):Q=INT(RND(1)*192
)
46 LINE (A,0)-(B,191),1:LINE (0,P)-(2
55,Q),1
50 GOTO 20
```

Veuillez remarquer que les nouvelles lignes d'instruction arrivent avant la boucle GOTO de la ligne 50, ce qui signifie qu'elles se répéteront indéfiniment, comme les étoiles. Chaque fois que l'ordinateur accomplit un tour, de la ligne 20 à la ligne 46, il crée de nouveaux points dans des endroits différents, puis de nouvelles lignes dans des endroits différents. Si un point se trouve à l'endroit où est tracée une droite noire, ce point devient invisible. A ce stade, donnez l'instruction RUN et que voyez-vous?... des étoiles **scintillantes.**

Si vous désirez voir réellement ces droites, de manière à mieux les comprendre, il vous suffit de changer de 1 en 2 le code de couleur de l'instruction 46. Vous voyez alors apparaître des droites vertes sur l'écran.

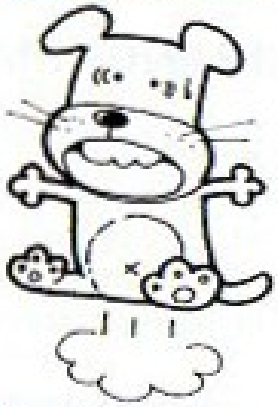
Remarque: Si vous arrêtez ce programme, l'écran reste noir. Mais vous pouvez retrouver le pourtour et l'écran bleu foncé original grâce à cette instruction:

```
COLOR 15,4
```

15 signifie des lettres blanches et 4 donne un écran et un pourtour bleu foncé.

En outre, bien que ceci n'ait pas encore été expliqué jusqu'ici, ajoutons que l'écran revient automatiquement à l'état SCREEN 0 lors de la remise en service du BASIC. Dans cet état, la couleur du pourtour est toujours la même que celle du fond. Si l'on désire obtenir des couleurs différentes pour le pourtour et le fond, on utilisera, par exemple, l'instruction SCREEN 1, puis l'instruction COLOR 15, 4, 7. Pour repasser à nouveau à l'écran original, utiliser l'instruction SCREEN 0.

SAUVEGARDE DES PROGRAMMES



Que faire pour...

- Raccorder l'ordinateur à un magnétophone
- Transférer un programme sur le magnétophone
- Confirmer que le programme est bien préservé
- Charger un programme du magnétophone dans l'ordinateur

Vous avez sans doute déjà vu une photo d'un gros ordinateur, une de ces machines impressionnantes, destinées à diriger une fusée, à gérer une entreprise ou à contribuer à la recherche médicale. Vous aurez sans doute remarqué qu'elles ressemblent à des magnétophones géants. En fait, c'est bien ce qu'elles sont, en partie. Sur ces grandes machines, on parle de "dérouleurs de bande" ou de "lecteurs de bande" et ils ont pour but d'enregistrer et de mémoriser les informations utilisées par l'ordinateur.

Une partie des informations qu'ils conservent, ce sont des programmes, les instructions pour la machine, un peu comme nos instructions PSET, SCREEN, PRINT, etc... que nous connaissons bien. Ils emmagasinent également des **données**, des éléments comme le score de jeux, les réponses à des problèmes ou des formules scientifiques.

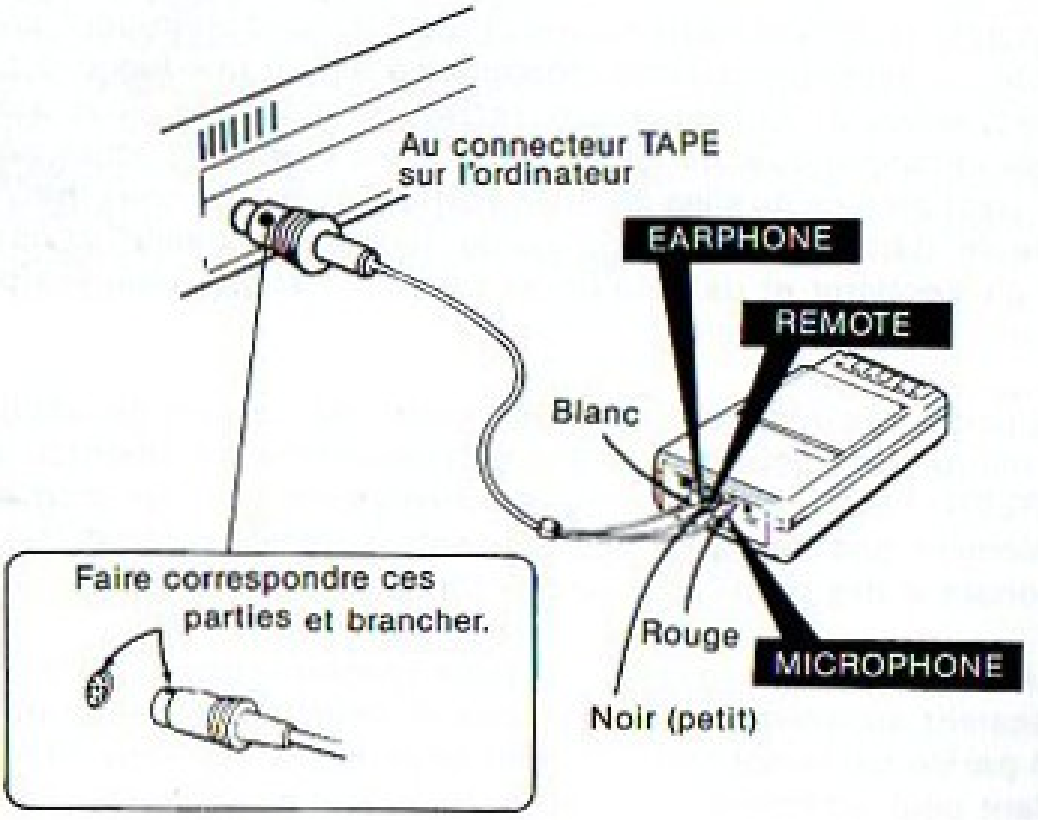
L'informatique est un secteur en plein épanouissement et ce qu'accomplissaient auparavant des machines énormes peut désormais être réalisé par de petits appareils comme votre ordinateur Sony. Si petit qu'un enfant peut aisément le porter, votre ordinateur Sony est, en fait, plus rapide et plus puissant que les anciens mastodontes qui coûtaient des milliers de dollars et ne pouvaient pas prendre place dans un grand living.

Aussi n'est-il pas étonnant que votre ordinateur Sony puisse mémoriser des programmes et des informations sur bande. Comme cette machine est plus petite qu'un ordinateur central, elle utilise un lecteur de bande nettement plus petit. Votre ordinateur Sony peut, en effet, utiliser un magnétocassette ordinaire, servant aussi pour la lecture de bandes musicales ou pour la dictée.

BRANCHEMENT D'UN MAGNETOPHONE

La connexion d'un magnétophone à l'ordinateur est très simple; elle se fait à l'aide du câble spécial, fourni avec l'ordinateur Sony. Une de ses extrémités présente une monture métallique ronde où des broches sont placées au centre. Ce bout se branche dans la prise marquée TAPE sur l'arrière du clavier.

L'autre extrémité du câble se divise en trois parties, de couleur rouge, blanche et noire. Le cordon rouge se branche sur la prise MICROPHONE du magnétophone el le blanc sur la prise marquée EARPHONE. Si le magnétophone utilisé est équipé d'une prise REMOTE, on y branchera le bout noir du câble (si votre magnétophone n'est pas doté d'une connexion REMOTE, le bout noir peut être laissé débranché).



Désormais, l'ordinateur et le magnétophone sont prêts à entrer en conversation.

POUR FAIRE PARLER L'ORDINATEUR

Votre ordinateur révèlera au magnétophone ce qu'il possède en mémoire si vous lui donnez cette instruction:

CSAVE "nom du fichier"

Le "C" au début de CSAVE signifie cassette, tandis que SAVE veut dire ceci: sauvegarder le programme qui se trouve dans la mémoire de l'ordinateur en l'enregistrant sur la cassette. Le "nom de fichier" (n'oubliez pas les guillemets) peut être, à votre choix, n'importe quel nom que vous souhaitez donner au programme, de sorte que vous-même, l'ordinateur et le magnétophone puissiez le différencier des autres.

Un **nom de fichier** peut comporter jusqu'à six caractères qui pourront être des lettres, des chiffres ou des signes graphiques, mais le **premier** caractère doit obligatoirement être une **lettre** de l'alphabet.

Le terme STAR (étoile) pourrait être un nom de fichier approprié pour le programme que nous avons effectué dans le chapitre précédent. Il présente quatre lettres, commence par une lettre et est facile à retenir puisqu'il indique le contenu du programme en question, les étoiles.

Sauvegardons à présent ce programme:

```
CSAVE "STAR"
```

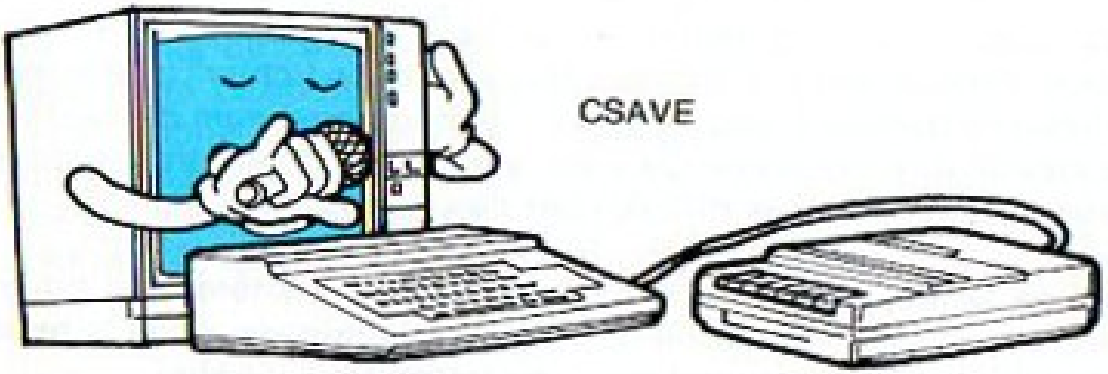
N'appuyez pas encore sur [RETURN] car, dans ce cas, l'ordinateur commencerait à envoyer les informations à préserver. Tout d'abord, nous devons nous assurer que l'ordinateur est prêt pour cette démarche.

Si le magnétophone est muni d'une connexion REMOTE et que le câble y est branché, on placera le magnétophone en mode d'enregistrement.

Vous remarquerez, cependant, que la bande ne commence pas encore à défiler; ceci vient du fait que l'ordinateur "contrôle" votre magnétophone. Si vous appuyez maintenant sur [RETURN], la bande se mettra en service, le programme s'enregistrera, puis le magnétophone arrêtera la bande. Lorsque l'enregistrement est achevé, ce qui suit apparaît sur l'écran:

```
CSAVE "STAR"
Ok
■
```

Si le magnétophone utilisé n'est pas pourvu de la connexion REMOTE, **appuyez sur la touche RECORD**, ce qui, dans un premier temps, fera tourner la bande. **Attendez** alors quelques secondes pour que la vitesse de défilement se stabilise, puis appuyez sur RETURN. Lorsque le message OK apparaît sur l'écran, **appuyez sur la touche STOP** du magnétophone et les démarches sont ainsi presque terminées.



A ce stade, votre programme se trouve en deux endroits. En effet, il subsiste encore dans la mémoire de l'ordinateur et, si tout s'est bien passé, il est également sauvegardé sur la bande.

VERIFICATION DE LA SAUVEGARDE SUR BANDE —

Avant de faire autre chose, **il importe** de s'assurer que le **magnétophone** a bel et bien sauvegardé le programme. Votre ordinateur Sony se chargera de ce travail en comparant le contenu de sa mémoire avec l'enregistrement. Tout d'abord, vous devez **débrancher** le câble au niveau de la prise "REMOTE", puis **rebobiner** la bande jusqu'au point juste avant l'endroit où le programme a été sauvegardé. **Réglez alors le volume** du magnétophone à environ la moitié. **Rebranchez** maintenant la fiche REMOTE et tapez l'instruction suivante:

```
CLOAD? "STAR"
```

Si la connexion "REMOTE" n'a pas été faite, actionnez la touche RETURN **avant** de placer le magnétophone en mode de lecture. Si l'appareil utilisé possède la fiche "REMOTE", réglez celui-ci en mode de lecture et la bande commencera automatiquement lorsque vous appuyerez sur RETURN. Lorsque l'ordinateur a "trouvé" le début du programme, l'écran affiche ce qui suit:

```
Found:STAR
```

A mesure que l'ordinateur reproduit pour lui-même la bande, il compare, point par point, son contenu avec celui de sa propre mémoire et affiche alors Ok si tout est en ordre (n'oubliez pas l'arrêt manuel de la bande à la fin, si le magnétophone n'autorise pas le contrôle à distance).

Nous venons ici de voir un processus important, car il vous permet d'éviter la perte de tout un programme réalisé, mais il peut aussi présenter parfois certaines difficultés. Si le message "Found" (trouvé) n'apparaît pas, c'est peut-être parce que vous n'avez pas rebobiné suffisamment la bande.

Si le signe OK **n'apparaît pas**, augmentez le volume du magnétophone et procédez à un nouvel essai. S'il n'est pas plus concluant que le précédent, il faut incriminer une interférence électronique ou le fait que le programme n'a pas été enregistré convenablement. Dans ce cas, vérifiez les connexions et recommencez à partir de CSAVE "STAR".

Si, cette fois-ci, tout a fonctionné normalement et que vous voyez le message OK, vous savez que votre programme a été sauvegardé. Vous pouvez alors étiqueter la bande, la ranger et l'utiliser à nouveau chaque fois que vous aurez besoin de ce programme à l'avenir.

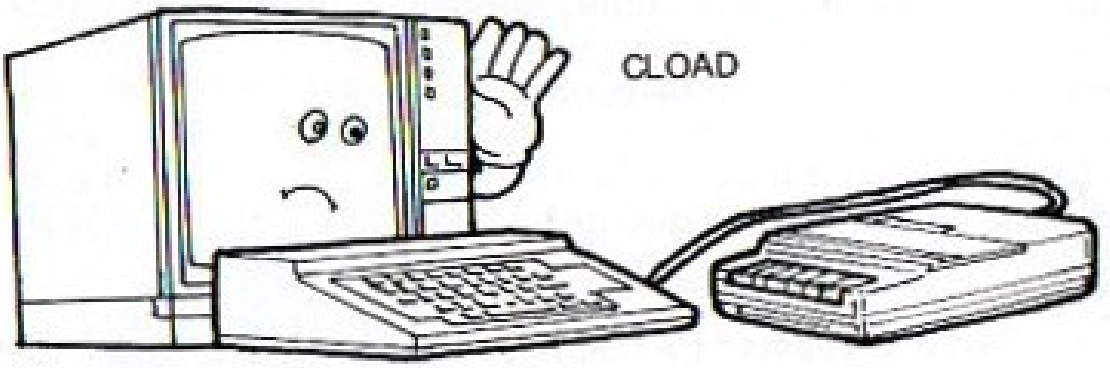
Lorsque vous avez acquis la certitude que votre programme est bien enregistré sur la bande, libre à vous de taper NEW ou d'appuyer sur la touche RESET. Cette démarche effacera la **mémoire de l'ordinateur**, mais pas le contenu de **la mémoire externe**, gardé sur la bande.

CHARGEMENT D'UN PROGRAMME

Une utilisation ultérieure de cette bande est appelée ici "chargement d'un programme", et il est temps de nous livrer à un peu de pratique. Le processus est très semblable à la vérification que nous venons d'effectuer. Assurez-vous que le magnétophone est réglé pour **lire** la bande pour l'ordinateur et que la bande est rembobinée à l'endroit exact, c'est-à-dire juste au début du programme enregistré. Donnez alors cette instruction:

```
CLOAD "STAR"
```

Remarquez que, cette fois-ci, il n'y a aucun point d'interrogation (?). Bien entendu, si vous avez utilisé un nom de fichier différent, tapez celui-ci entre les guillemets à la place du mot STAR.



Lorsque l'ordinateur a fini d'écouter la bande, son écran affiche ceci:

```
CLOAD "STAR"
Found:STAR
Ok
■
```

Assurez-vous que la bande est arrêtée et vous êtes alors en mesure d'**exécuter** (RUN) votre programme sur l'ordinateur.

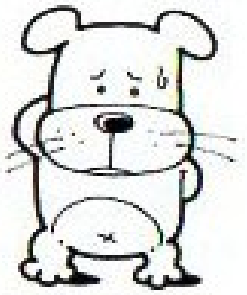
Occasionnellement, l'ordinateur affichera

```
Device I/O error
```

I/O signifie "input/output", les termes anglais pour "entrée /sortie" et, dans ce cas, ils se réfèrent à la connexion avec le magnétophone. Modifiez légèrement le volume et essayez une nouvelle fois la démarche du chargement.

DECISIONS ET JEUX

Que faire pour...



- Faire prendre des décisions par l'ordinateur
- Programmer une formule conditionnelle
- Utiliser un jeu sur l'odinateur
- Simplifier un programme
- Améliorer un programme

Rappelez-vous notre Milou, le petit chien savant, très adroit pour les tours d'adresse, tels que "Assis", "Rapporte" ou "Donne la patte". Jusqu'à présent, votre ordinateur Sony a accompli des tâches toutes simples, répondant à vos ordres, tels que "PRINT", "GOTO", "PSET", etc.

Nous avons ensuite fait connaissance de Super Milou, capable de **prendre certaines décisions** en se demandant "si la balle est noire" ou "si la balle est blanche". Vous savez déjà que votre ordinateur Sony est assez intelligent pour exécuter ces tours d'adresse et maintenant nous allons commencer à rédiger des "super-programmes" qui prennent des décisions.

Tout d'abord, effacez votre dernier programme à l'aide de l'instruction NEW, puis entrez ceci:

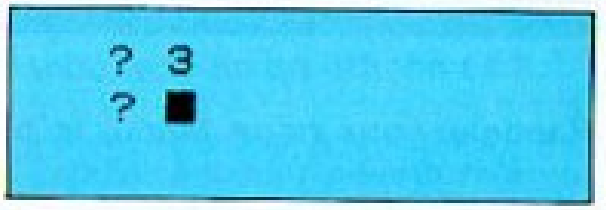
```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN GOTO 50
40 GOTO 10
50 PRINT "Réussi!"
60 GOTO 10
```

Après avoir tapé correctement ce programme, donnez l'instruction RUN et appuyez sur RETURN.

```
RUN
? ■
```

Auparavant, l'ordinateur nous avait déjà répondu par un point d'interrogation, mais cette fois-ci, le curseur se trouve sur la même ligne que le point d'interrogation. Cela signifie qu'il attend que **vous entriez** quelque chose, à cause de l'instruction INPUT de la ligne 20. Vous pouvez entrer quelque chose,—dans ce programme-ci, un nombre quelconque de 1 à 5,—en appuyant tout simplement sur une touche, comme vous le savez déjà. Aussi, donnons par exemple le chiffre 3 à l'ordinateur (que se passera-t-il si nous entrons une lettre ou un signe graphique?).

Appuyez sur [3] et [RETURN]. Que voyez-vous apparaître maintenant sur l'écran?



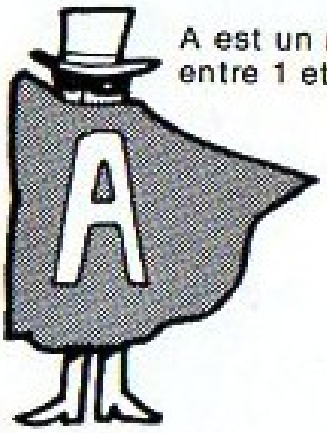
Les deux affichages ont un point d'interrogation à la fin, ce qui signifie, vous le savez, que l'ordinateur attend que vous introduisiez un autre nombre. Et un autre. Et un autre. Et encore un autre. Voilà précisément le jeu: êtes-vous parvenu à deviner juste? avez-vous obtenu "Réussi" ou un échec? (Selon la loi de la chance, vous devriez obtenir environ 1 Réussi sur cinq essais, ou plus ou moins 2 Réussi sur dix essais, soit près de 20 pour cent.) Si vous êtes fatigué de ce jeu avant l'ordinateur (il ne se fatigue jamais!), il vous suffit d'appuyer sur [CTRL] et [STOP].

Que se passait-il donc au cours de ce jeu? Observons bien le programme. Sa première ligne, nous la connaissons bien: il s'agit de **l'instruction de nombre aléatoire**. Chaque fois que l'ordinateur a terminé la boucle (ligne 60), il choisit un nombre compris entre 1 et 5 et en fait la valeur de la variable A.

Ici, nous rencontrons un élément nouveau, à la ligne 20:

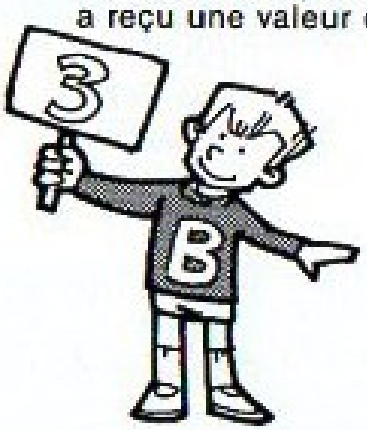
```
20 INPUT B
```

B est une seconde variable et elle attend que vous lui donniez une valeur par l'intermédiaire du clavier. Après que vous avez tapé un nombre, les deux variables ont une valeur et l'ordinateur passe alors à la ligne suivante.



A est un nombre compris entre 1 et 5.

B, dans notre premier essai, a reçu une valeur de 3.



REUSSI OU RATE: L'ORDINATEUR EST L'ARBITRE —

La ligne 30 nous rappelle Super Milou, parce que le premier mot est IF (si).

```
30 IF A=B THEN GOTO 50
```

Le mot IF (si) est **toujours** accompagné de THEN (alors). Par exemple: "SI vous avez faim, ALORS vous devez manger" ou encore "SI vous avez un ordinateur Sony, ALORS vous allez vous amuser".

IF [une condition] THEN [quelque chose]

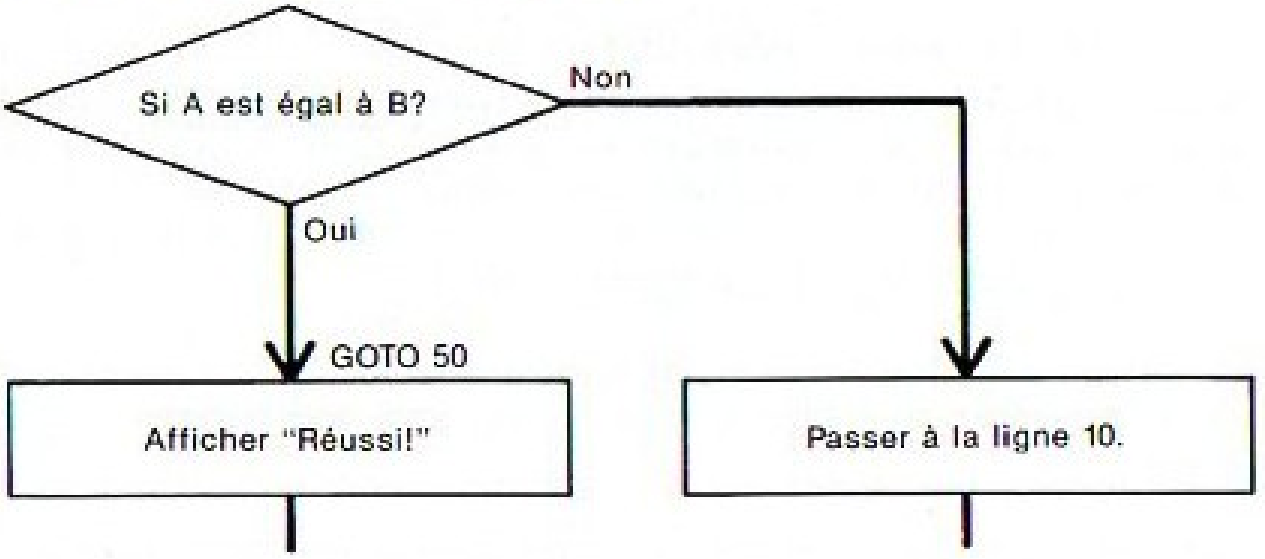
Quelle peut être la **condition** des variables A et B? Elles peuvent être égales (A=B), ou A peut être plus grand (A>B), ou B peut être plus grand (A<B).

"Quelque chose" peut être une instruction quelconque. Dans le cas qui nous intéresse ici il s'agit de GOTO 50.

C'est ce que nous appelons une **formule conditionnelle**. En langage BASIC, il en existe six et certaines d'entre elles peuvent s'écrire de deux façons différentes.

Symbole	Signification	Exemple	
=	Egal à	IF A=B	Si A est égal à B
>	Supérieur	IF A>B	Si A est supérieur à B
<	Inférieur	IF A<B	Si A est inférieur à B
>=	Supérieur ou égal à	IF A>=B	Si A est supérieur ou
=>		IF A=>B	égal à B.
<=	Inférieur ou égal à	IF A<=B	Si A est inférieur ou
=<		IF A=<B	égal à B
<>	Différent de	IF A<>B	Si A est différent
><		IF A><B	de B

Ainsi donc, la ligne 30 représente une formule conditionnelle. Lorsque la partie IF est vérifiée (A=B), l'ordinateur passe à l'instruction (GOTO 50). Par contre, lorsque A et B ne sont pas égaux, il passe à la ligne suivante du programme, soit la ligne 40, sans lire l'instruction THEN. Evidemment, la ligne 40 renvoie l'ordinateur au début, pour produire un nouveau nombre aléatoire que vous devez deviner.



A présent, vous comprenez que "Réussi!" ne peut être affiché que si A et B (B étant le nombre que vous avez donné en devinant) sont égaux. "Réussi!" signifie que le nombre que vous avez entré pour la variable B est le même que la valeur que l'ordinateur avait donné à la variable A.

Après avoir affiché "Réussi!", l'ordinateur passe à la ligne 60 et le jeu recommence. Que A et B soient égaux ou différents, le jeu se poursuit.

SIMPLIFICATION D'UN PROGRAMME

Dans toute la mesure du possible, on essaiera toujours de simplifier au maximum les programmes de l'ordinateur: outre l'avantage de la simplicité de leur compréhension, on réduit ainsi les chances d'erreur en les tapant. Examinons d'un peu plus près notre programme de devinette des nombres.

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN GOTO 50
40 GOTO 10
50 PRINT "Réussi!"
60 GOTO 10
```

Nous savons maintenant que si A=B, l'ordinateur affichera "Réussi!". Aussi simple que cela puisse paraître, ce programme comporte, en fait, deux instructions. La première, qui vient après THEN à la ligne 30, est GOTO 50. La seconde à la ligne 50, est PRINT "Réussi!". Une action simple devrait consister en une seule instruction, et c'est pourquoi nous pouvons changer la ligne 30 en

```
IF A=B THEN PRINT "Réussi!"
```

Par conséquent, nous n'avons plus besoin des instructions des lignes 50 et 60, n'est-ce pas?

```
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN PRINT "Réussi!"
40 GOTO 10
```

Ce qui est bien plus simple, vous en conviendrez! Si A et B sont égaux, après avoir imprimé "Réussi!", l'ordinateur passera à la ligne 40, puis GOTO 10. Si A et B ne sont pas égaux, il fera la même chose, mais sans imprimer "Réussi!" dans ce cas. Ainsi, notre jeu fonctionne maintenant avec quatre instructions uniquement.

A présent, apportez les modifications sur l'écran. Changez tout d'abord l'instruction de la ligne 30 de GOTO 50 en PRINT "Réussi!". Effacez ensuite les lignes 50 et 60 de cette façon: amenez le curseur à la position sous la ligne 60 et entrez 50. Si vous appuyez sur RETURN, la ligne 50 sera effacée de la mémoire de l'ordinateur. Faites alors la même chose avec 60 pour effacer la ligne 60.

Les lignes 50 et 60 ont-elles disparu de votre écran? Non, pas encore; toutefois, elles ne se trouvent plus dans la mémoire. Si vous donnez l'instruction LIST, l'ordinateur affichera ce qui se trouve alors dans sa mémoire, à savoir:

```
LIST
10 A=INT(RND(1)*5)+1
20 INPUT B
30 IF A=B THEN PRINT "Réussi!"
40 GOTO 10
```

Ce qui est bien plus clair, n'est-il pas vrai?

Quelques raffinements

Nous avons donc rendu notre programme de devinette aussi simple que possible et nous pouvons maintenant penser à l'améliorer en y ajoutant des éléments pratiques et amusants. Bien entendu, nous tâcherons de rendre ces améliorations aussi simple que possible également.

Quand A est égal à B, l'ordinateur vous le dit en affichant "Réussi!". Cependant, quand les deux valeurs ne sont pas égales, il ne vous donne pas de message avant de recommencer. Le programme serait plus facile à comprendre pour vos amis si nous changions comme suit la ligne 30:

```
30 IF A=B THEN PRINT "Réussi!" ELSE PRINT
"Raté"
```

ELSE signifie "si non". Cette instruction est pratique quand elle est placée après une formule conditionnelle parce qu'elle clarifie cette démarche de programme.

IF [formule conditionnelle] THEN [instruction 1]
ELSE [instruction 2]

Comme cette nouvelle instruction comporte plus de 37 caractères, elle "débordera" sur la ligne suivante et fera disparaître provisoirement la ligne 40. Mais ne vous inquiétez pas, car elle subsiste dans la mémoire de l'ordinateur. Pour vous en convaincre, utilisez l'instruction LIST et la voilà!

Il y a une autre amélioration qui rendra votre programme plus compréhensible pour vos amis. Nous pouvons changer la ligne 20 en:

```
20 INPUT "Devinez (1-5) ";B
```

Vous comprendrez la signification de cette nouvelle instruction si vous l'entrez sur l'écran et que vous donnez ensuite l'instruction RUN.

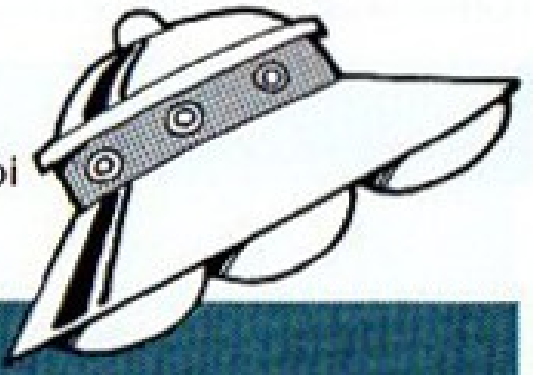
```
RUN
Devinez (1-5) ? ■
```

Comme vous le voyez, tout devient facile à comprendre et bien plus intéressant que le message "?" que nous voyions apparaître auparavant. Désormais, vous pouvez présenter ce jeu de devinette à vos amis et chacun en comprendra immédiatement le fonctionnement.

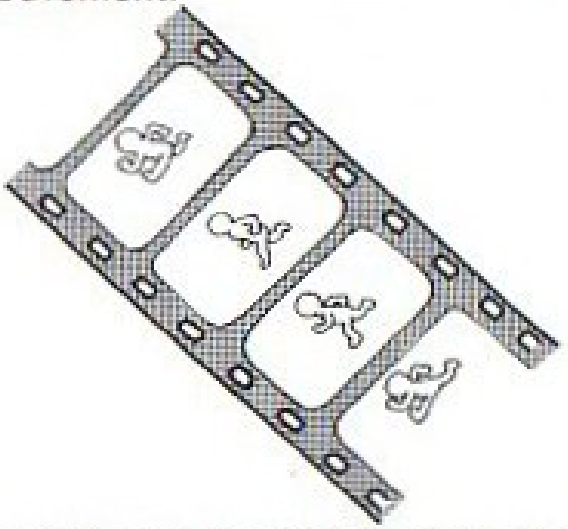
DES GRAPHIQUES EN MOUVEMENT

Que faire pour...

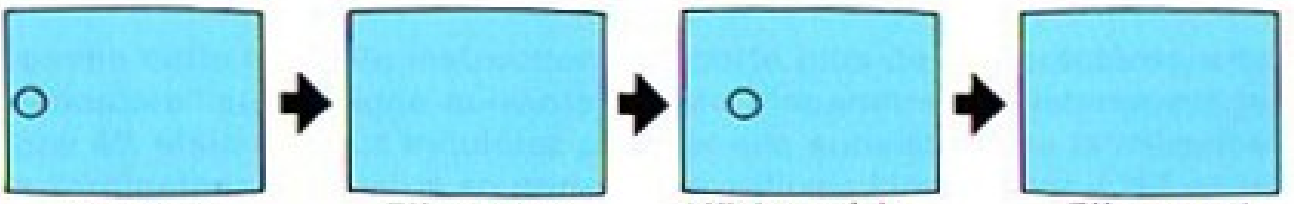
- Déplacer les caractères sur l'écran
- Faire un film d'OVNI
- Raccourcir un programme par emploi de variables

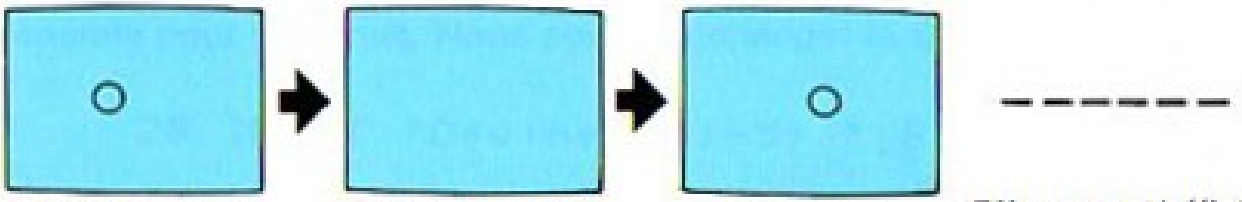


Comme chacun le sait, un film de cinéma est constitué d'une série d'images fixes, défilant rapidement les unes après les autres. Comme chacune de ces images sont légèrement différentes, nous avons l'impression d'un mouvement.



Le même principe se vérifie pour la télévision. Sur l'écran, l'image change de nombreuses fois par seconde et c'est ainsi que nous obtenons le mouvement. L'écran de votre ordinateur Sony est presque le même que celui d'un téléviseur et, par conséquent, il peut servir pour créer des images qui bougent. Voici d'ailleurs comment procéder:





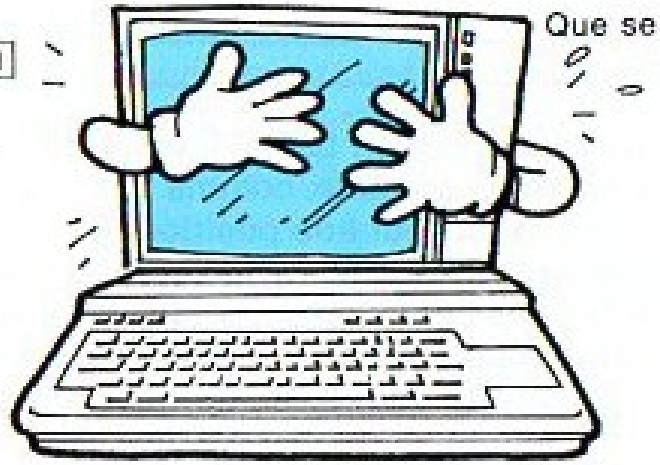
Effacement/affichage répété de façon continue

Pour déplacer le repère "○" à travers l'écran, nous le faisons apparaître sur la gauche, puis nous l'effaçons, puis un peu plus vers la droite, nouvel effacement... et ainsi de suite. Pour effectuer une démarche de cette série, nous devons dire à l'ordinateur d'effacer, puis de localiser la position correcte et ensuite d'y imprimer quelque chose.

```
CLS
LOCATE 36,15:PRINT "Z"
```

Ici, nous rencontrons deux nouvelles instructions, suivies de PRINT. CLS provient de "Clear Screen" (Vider l'écran). Si elle est entrée telle quelle, tout ce qui se trouve sur l'écran disparaîtra.

CLS + RETURN



Que se passe-t-il sur mon visage?

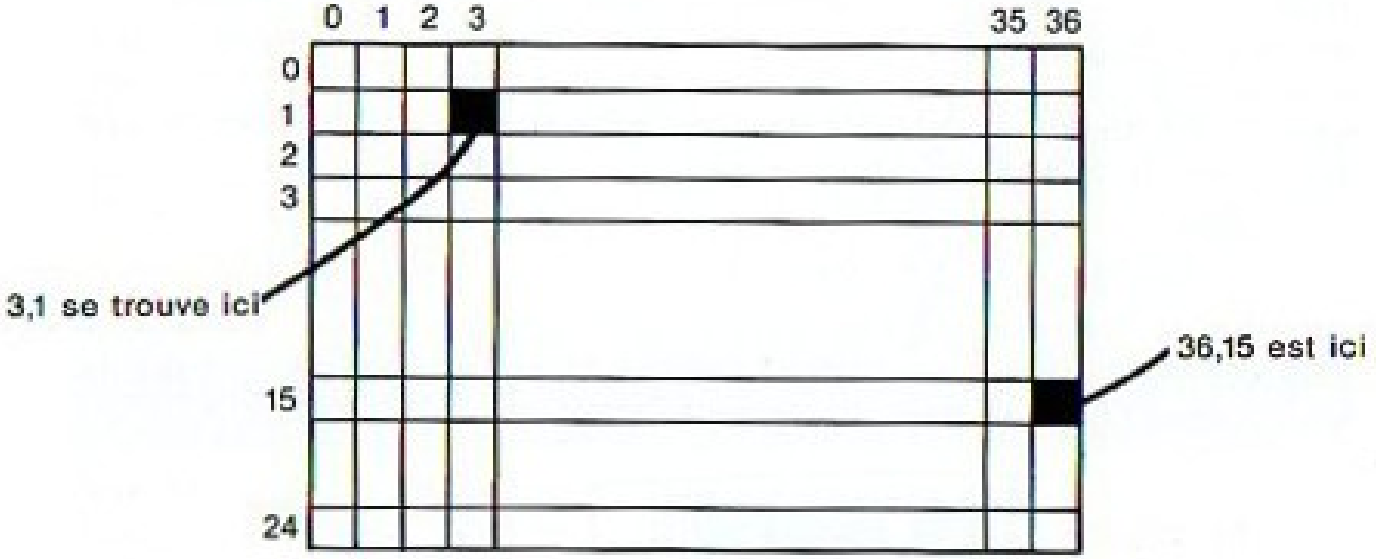
Pour comprendre ce que signifie l'instruction LOCATE, introduisons ensemble les trois instructions ci-dessus.

```
LOCATE 36,15:PRINT "Z"


Ok                                   Z
■
```

L'instruction LOCATE 36, 15 dit au curseur de se déplacer de 36 espaces, puis de descendre à la ligne 15. Les deux points (:) avertissent l'ordinateur qu'une autre instruction se trouve sur la même ligne. Et enfin, l'instruction PRINT lui ordonne d'afficher Z à l'endroit où se trouve le curseur.

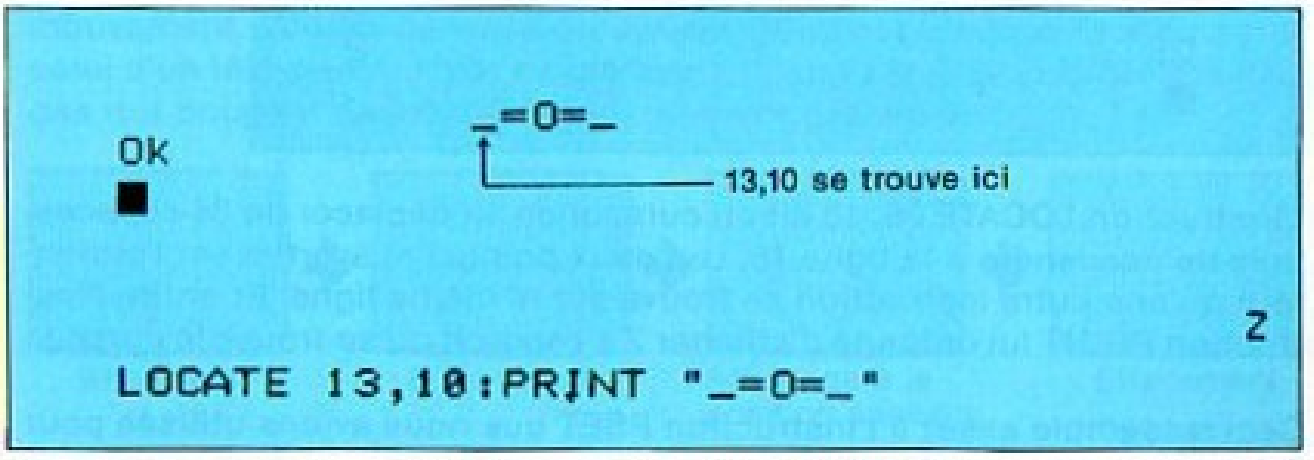
Ceci ressemble assez à l'instruction PSET que nous avions utilisée pour faire apparaître des étoiles scintillantes, mais à une différence près: le caractère Z est plus grand qu'un point et Z est un caractère, alors qu'un point était un graphique. Pour les caractères, par conséquent, nous disons LOCATE 36, 15: PRINT "quelque chose"; par contre, pour les graphiques, nous avions dit PSET (36, 15), par exemple. Le résultat est le même. Dans un cas comme dans l'autre, nous pouvons penser que 36, 15 est une **adresse**.



L'instruction PRINT qui suit l'instruction LOCATE et les deux points (:) n'est nullement limitée à un seul caractère, comme Z. En fait, **n'importe quel nombre** de lettres ou de mots peut être positionné par l'instruction LOCATE. Celle-ci dit simplement au curseur où il doit aller avant de commencer à imprimer (PRINT). Vous pourrez utiliser LOCATE pour commencer un paragraphe au milieu d'une page, pour commencer un jeu en bas de l'écran, ou pour poser n'importe quoi à l'endroit que vous souhaitez.

Utilisons quelque peu notre imagination. Voulez-vous faire apparaître un OVNI? (Vous le savez, un OVNI est un "objet volant non indentifié", un de ces vaisseaux spatiaux venus d'une autre planète et qui utilisent probablement aussi des ordinateurs). Avec 5 caractères tout simplement, vous pouvez créer un OVNI:

```
LOCATE 13,10:PRINT "_=0=_"
```



Ce dessin vous donne-t-il l'impression d'un OVNI? Vous pouvez essayer de lui donner une autre forme en utilisant les caractères et les signes de votre clavier. Pour cet OVNI, nous avons utilisé tout simplement le trait de souligné (_), le signe égale (=) et la lettre O.

Essayons maintenant autre chose:

```
LOCATE 13,10:PRINT "     "
```

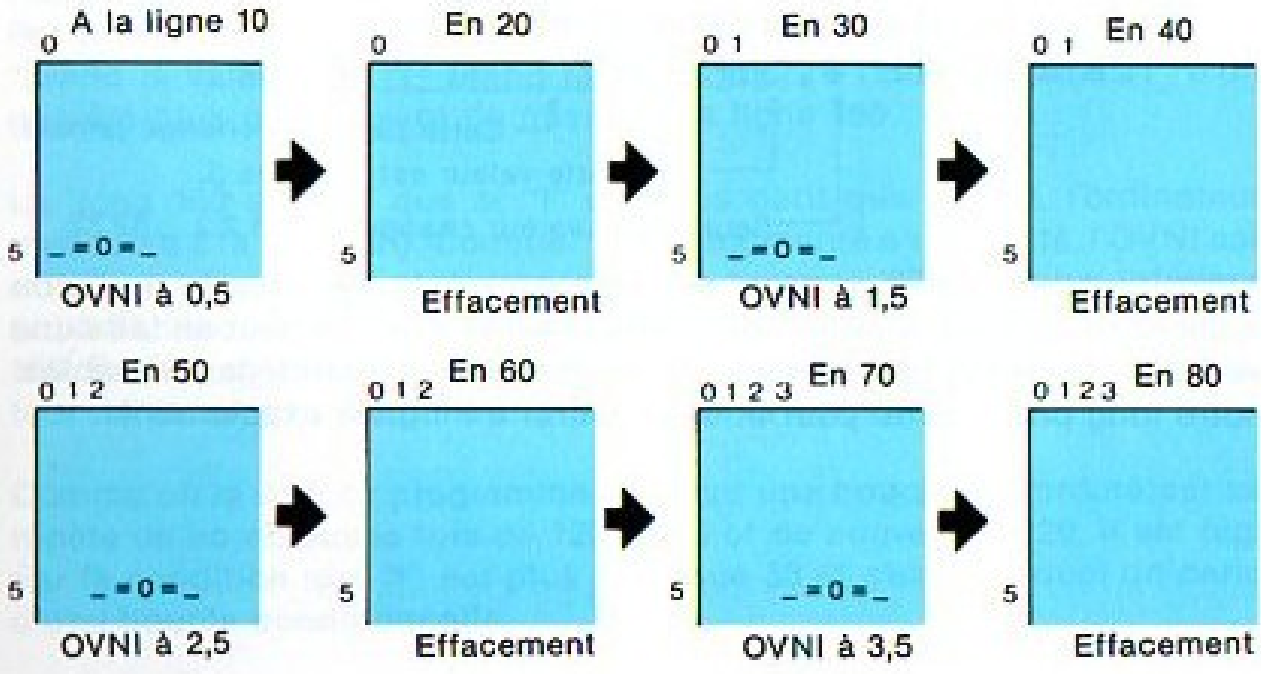
Si vous appuyez sur RETURN, l'OVNI disparaît. Les cinq espaces ont remplacé les cinq caractères. A vrai dire l'**espacement** est un moyen pratique pour effacer des lettres ou des mots quand on ne désire pas effacer tout l'écran en utilisant CLS.

AFFICHER, EFFACER, AFFICHER, EFFACER...

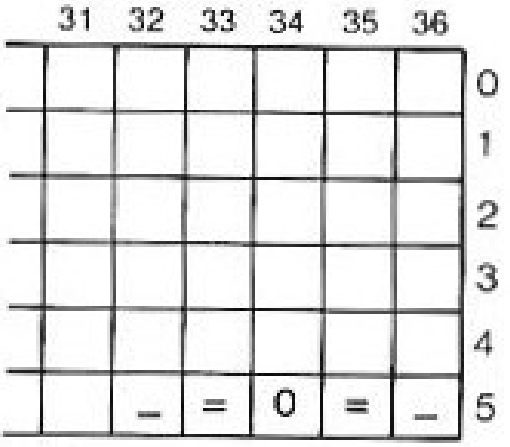
Ainsi donc, les instruction LOCATE et CLS sont le moyen utilisé pour créer un "film". Voici un programme permettant de faire bouger votre OVNI (ne le tapez pas encore, mais cherchez comment il doit fonctionner).

```
5 CLS
10 LOCATE 0,5:PRINT "_=O=_"
20 LOCATE 0,5:PRINT "     "
30 LOCATE 1,5:PRINT "_=O=_"
40 LOCATE 1,5:PRINT "     "
50 LOCATE 2,5:PRINT "_=O=_"
60 LOCATE 2,5:PRINT "     "
70 LOCATE 3,5:PRINT "_=O=_"
80 LOCATE 3,5:PRINT "     "
```

Voici à quoi doivent ressembler ces instructions sur votre écran:



La première ligne, à savoir CLS (vider l'écran), et ensuite chaque paire de lignes (10 et 20, 30 et 40, 50 et 60, 70 et 80) placent l'OVNI à une adresse différente. Si nous continuons jusqu'au bord droit de l'écran, de cette façon:



la dernière instruction sera

```
650 LOCATE 32,5:PRINT "_=0=_"
660 LOCATE 32,5:PRINT "     "
```

Tout ce programme comporterait donc 67 lignes. Drôlement fastidieux, direz-vous, et il serait sûrement plus intéressant de regarder la télé!

Mais n'abandonnez pas de si tôt! Car il existe un moyen bien plus simple. Sans doute avez-vous déjà remarqué qu'après la ligne 5, les instructions se ressemblent et c'est dans ce sens que nous devons chercher. En fait, elles se présentent sous forme de paires presque identiques.

```
10 LOCATE 0,5:PRINT "_=0=_"
20 LOCATE 0,5:PRINT "     "
```

Cette partie ne change jamais.

Cette valeur est toujours 5.

Seule cette valeur change: 0,0,1,1,2,2,....32,32

Ainsi, seul le premier nombre de l'adresse change et il varie de façon systématique. Cela signifie, vous l'aviez deviné, que l'on peut en faire une **variable**. De cette façon, vous comprenez tout l'intérêt des variables: notre long programme peut être raccourci à 7 lignes exactement.

```
100 CLS
110 I=0
120 LOCATE I,5:PRINT "_=O=_"
130 LOCATE I,5:PRINT "     "
140 I=I+1
150 IF I<33 THEN GOTO 120
160 END
```

Déblayez la mémoire de l'ordinateur par l'instruction NEW et entrez ensuite ce programme (cette fois-ci, nous avons commencé les numéros des lignes à 100 de manière à pouvoir ajouter quelques lignes supplémentaires par la suite). Regardez maintenant votre OVNI à l'écran: il vole vraiment!

Examinons à présent le programme, ligne par pigne, et essayons de comprendre **pourquoi** il vole. La ligne 100 déblaie l'écran (pour que l'OVNI ne se cogne pas contre des obstacles!) La ligne 110 nomme la variable et lui attribue une valeur. Etant donné que "I" signifie la position gauche-droite et que nous souhaitons commencer du côté gauche, "I" recevra la "valeur initiale" de zéro. Quant aux lignes 120 et 130, il s'agit de la paire d'affichage-effacement que nous avons déjà rencontrée, mais dans ce cas, elle utilise la variable "I".

La ligne 140 change la variable et est énoncée comme suit: I=I+1. Rappelez-vous qu'il ne s'agit pas d'arithmétique, mais de langage BASIC. Per conséquent, le signe "=" ne signifie pas "égale", mais veut dire "prend la valeur de". "I" prend la valeur de "I+1", ce qui rend "I" d'un numéro plus grand avant de passer à la ligne 150.

La ligne 150 signifie que si "I" est plus petit que (<) 33, l'ordinateur repassera à la ligne 120. Comme "I" est augmenté d'une unité, l'OVNI est affiché un espace de plus vers la droite. Lorsque "I" n'est plus inférieur (<) à 33, l'ordinateur ne repassera pas à "GOTO 120", mais il ira à la ligne 160. En fait, même sans END (fin), le programme s'arrêterait car l'ordinateur ne recevrait plus d'instruction après la ligne 150.

Comme on le voit, ce programme contient une **boucle** où l'ordinateur se répète de nombreuses fois de 120 à 150 et de nouveau à 120. Il est régi par la condition que "I" est plus petit que 33 et c'est pourquoi on parle d'une **boucle conditionnelle**.

Un meilleur moyen d'arriver aux mêmes résultats
Examinons une nouvelle fois notre programme

```
100 CLS
110 I=0
120 LOCATE I,5:PRINT "_=0=_"
130 LOCATE I,5:PRINT "     "
140 I=I+1
150 IF I<33 THEN GOTO 120
160 END
```

← La valeur initiale est zéro

← La valeur augmente d'une unité à chaque boucle; la boucle s'arrête à 33

Les parties les plus importantes de ce programme sont les lignes 110, 140 et 150. Elles disent, en effet, à l'ordinateur où il doit commencer (à 0) et de répéter la boucle 33 fois. Ce genre de répétition est souvent utilisé dans divers programmes et c'est pourquoi le BASIC a une manière spéciale de le faire, grâce aux deux instructions appelées **FOR** et **NEXT**.

```
100 CLS
110 FOR I=0 TO 32
120 LOCATE I,5:PRINT "_=0=_"
130 LOCATE I,5:PRINT "     "
140 NEXT I
150 END
```

← Répéter de 0 à 32

← Augmenter d'une unité (1) la valeur de "I"; revenir à la ligne 120.

Ces deux instructions, FOR à la ligne 110 et NEXT à la ligne 140, sont les spécialistes des boucles. Elles utilisent seulement deux lignes, alors qu'il en fallait trois auparavant.

La chose importante à se rappeler ici est cette formule:

FOR [variable] = [valeur initiale] TO [valeur finale]

et

NEXT [variable]

Dans le cas précis de notre programme, il s'agit de

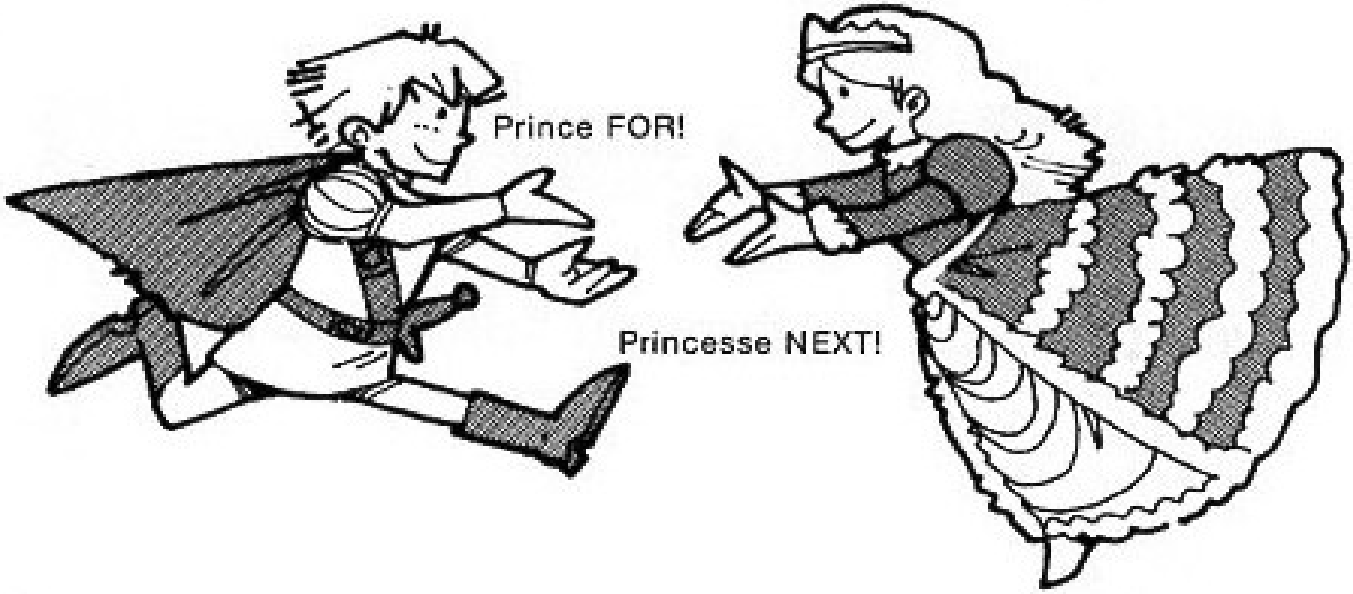
```
110 FOR I=0 TO 32
```

et

```
NEXT I
```

L'instruction FOR dit à l'ordinateur le nombre de boucles qu'il doit effectuer. NEXT est une commande très pratique parce qu'elle compte les démarches et fait passer l'ordinateur à la ligne suivante après que la variable atteint la valeur finale.

FOR et NEXT vont **toujours** ensemble et elles forment une paire.



Si le prince FOR ne parvient pas à trouver la princesse NEXT, l'ordinateur sera tellement malheureux qu'il affichera un message d'erreur et s'arrêtera jusqu'a ce que vous corrigiez cette situation.

POUR OBTENIR UN ENSEMBLE COHERENT

Que faire pour...

- Ajouter de la couleur et du son aux programmes
- Réunir deux programmes
- Etablir un organigramme
- Améliorer vos programmes

AVEC DE LA COULEUR ET DU SON

Nous sommes maintenant prêts pour utiliser certaines fonctions que nous avons déjà apprises auparavant, de manière à rendre notre OVNI mobile plus coloré et plus vivant. Introduisez ce programme dans votre ordinateur:

```
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=0=_"
130 LOCATE I,Y:PRINT "     "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 100
```

Répétition par FOR-NEXT

On pourrait maintenant voir l'OVNI qui vole aussi longtemps qu'on le souhaite car tout le programme est une boucle répétitive, mais appuyons sur CTRL et STOP de manière à mieux voir comment fonctionnent les instructions.

La ligne 100 accomplit deux choses: elle détermine la couleur (fond noir, caractères blancs) et elle vide l'écran. La ligne 103 fournit une nouvelle variable, à savoir Y, et lui donne une **valeur aléatoire** de 0 à 21. La ligne 106 lui est semblable, en ce sens qu'elle fournit une valeur aléatoire, entre 2 et 15, à une nouvelle variable C, puis fait de cette variable le numéro de la couleur (nous éliminerons la couleur 1 car des caractères noirs seraient invisibles!).

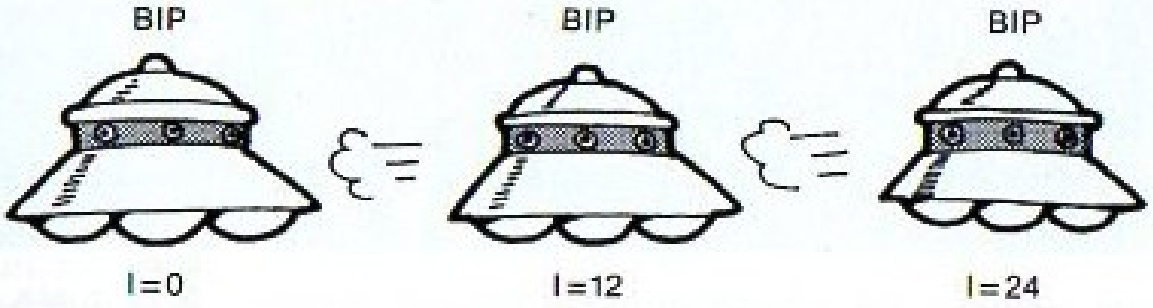
A partir de la ligne 110, nous trouvons la séquence FOR-NEXT et cette fois-ci, la variable Y fait partie de l'**adresse** pour chaque instruction LOCATE. L'emplacement horizontal (gauche-droite) est I, qui change **régulièrement** de 0 à 32, mais l'emplacement vertical (haut-bas) a été rendu **aléatoire** (RND) à la ligne 103; Y peut devenir n'importe quelle valeur de 0 à 21. Et c'est ainsi que des OVNI de couleurs différentes apparaissent à des altitudes différentes.

Observons, pour suivre, les ligne 133 et 136:

```
A=I MØD 12
IF A=Ø THEN BEEP
```

Nous y rencontrons deux mots nouveaux. BEEP signifie le "bip", la tonalité électronique produite par les OVNI, vous vous en doutiez. Mais cette tonalité est audible uniquement si A est zéro et A a la valeur de I MOD 12. Alors, que signifie ce terme MOD? Ce mot est une abréviation de **module** et il indique le nombre laissé quand un nombre est divisé par un autre. I MOD 12 est le nombre en sus lorsque I est divisé par 12. Si I est 15, alors I MOD 10 est 3. Mais si I est 24, I MOD 12 est 0.

Si I augmente de 0 à 32, I MOD 12 devient trois fois 0 pour chaque OVNI, à savoir à 0, 12 et 24. Ceci revient à dire que chaque OVNI fait entendre trois tonalités (bip) au cours de son passage à travers l'écran.



JEU DE DEVINETTE+SON+VIDEO

Un des aspects les plus intéressants de la programmation consiste à réunir deux programmes en un seul. Combiner deux ou plusieurs programmes peut s'avérer très pratique et amusant, comme nous allons le découvrir ici.
Nous revoici en pays de connaissance:

Programme de devinette

```
10 A=INT(RND(1)*5)+1
20 INPUT "Devinez";B
30 IF A=B THEN PRINT "Réussi!" ELSE P
RINT "Raté"
40 GOTO 10
```

Programme d'OVNI

```
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=O=_"
130 LOCATE I,Y:PRINT "      "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 100
```

Si nous plaçons tout simplement ces deux programmes, tels quels et l'un après l'autre, dans l'ordinateur, ils ne feront pas bon ménage. En effet, le programme se déroulera jusqu'à la ligne 40, puis il repassera à la ligne 10 et nos OVNI n'apparaîtront jamais. Le premier changement que nous devons apporter, par conséquent, est de réunir les OVNI au premier programme et le meilleur endroit pour effectuer cette connexion, c'est là où nous avons deviné juste. C'est-à-dire à la ligne 30 (A=B). Ecrivons donc une nouvelle ligne 30:

```
30 IF A=B THEN GOTO 100 ELSE PRINT "Raté"
```

Mais un autre problème subiste: comment arrêter les OVNI lorsque quelqu'un a deviné le nombre correct? Nous ne voulons pas regarder indéfiniment les OVNI; nous ne désirons pas non plus arrêter l'ordinateur; ce que nous souhaitons, c'est revenir à notre jeu des devinettes. Pour cela, nous changerons la ligne 150 comme ceci:

```
150 GOTO 10
```

Voici donc comment se présente notre nouveau programme combiné:

```
10 A=INT(RND(1)*5)+1
20 INPUT "Devinez";B
30 IF A=B THEN GOTO 100 ELSE PRINT "R
até"
40 GOTO 10
100 COLOR 15,1:CLS
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=O=_"
130 LOCATE I,Y:PRINT "     "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 10
```

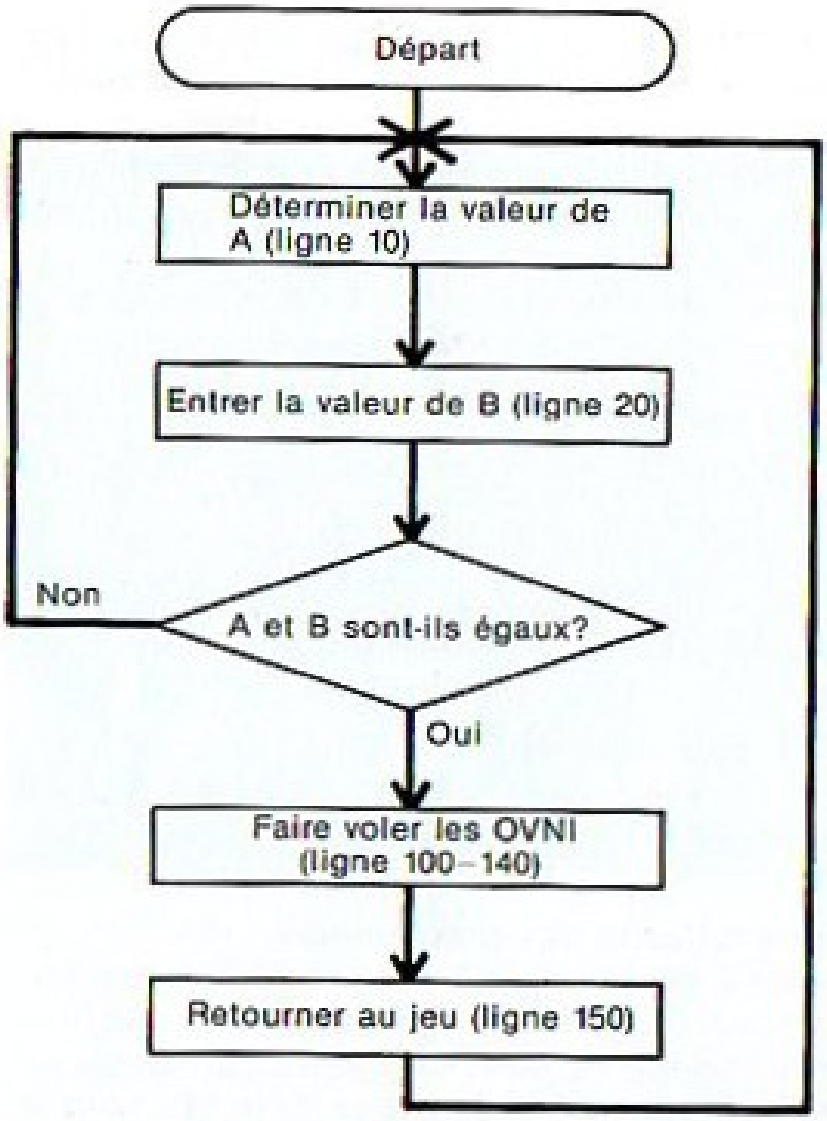
Si Réussi! passer à OVNI

Revenir aux devinettes

L'organigramme: un moyen de réfléchir aux programmes

Nos programmes commencent à s'allonger et à se compliquer, comportant des boucles et des fonctions multiples. Lorsque des informaticiens veulent créer des programmes complexes, ils établissent au préalable un plan soigné, avant même de commencer à rédiger des instructions et le moyen dont ils se servent porte le nom d'**organigramme**.

Cet ouvrage ne fait pas appel à des organigrammes et, pour tout dire, vous ne devez pas du tout les comprendre pour utiliser votre ordinateur Sony. Libre à vous, par conséquent, de passer au chapitre suivant; toutefois, si vous êtes quelque peu curieux, voici comment établir un organigramme pour notre programme jeu des devinettes/OVNI.



Le "pavé" supérieur est le point de départ du programme et chacun des autres pavés indique une opération qui nécessite une instruction (ou plusieurs). La ligne montre le parcours de l'action, ou la logique du programme. Remarquez ensuite que le "pavé" en forme de losange possède deux lignes qui en sortent; il a pour but de montrer qu'une condition (IF) est vérifiée par l'ordinateur, qui doit décider de quel côté aller. Notez également que la ligne "Non" et le pavé du bas forment tous deux des boucles, de façon à répéter le jeu.

Amélioration du jeu des devinettes

La différence entre un programme passable (un qui fonctionne) et un **excellent** programme, ce sont les améliorations qui y sont apportées après sa rédaction. Voici, par exemple, trois choses que nous pourrions améliorer dans notre programme de devinettes.

● Après chaque OVNI, le nombre suivant est de la même couleur que l'OVNI lui-même. Si les nombres étaient tous blancs, ne seraient-ils pas plus faciles à lire?
● Le jeu de devinette se déplace de haut en bas sur l'écran, commençant toujours à la ligne en-dessous du dernier OVNI. Ne serait-il pas possible de le rendre plus stable?
● Lorsque vole le premier OVNI, la couleur du fond passe du bleu foncé au noir, puis elle reste noire. N'aurait-on pas avantage à garder toujours la même couleur de fond?

Par quelques simples additions ou modifications, nous pouvons trouver solution à ces petits problèmes.

```
5 COLOR 15,1
7 CLS:LOCATE 0,0          <-- Ajouter
10 A=INT(RND(1)*5)+1
20 INPUT "Devinez";B
30 IF A=B THEN GOTO 100 ELSE PRINT "R
até"
40 GOTO 10
100 CLS                   <-- Changer
103 Y=INT(RND(1)*22)
106 C=INT(RND(1)*14)+2:COLOR C
110 FOR I=0 TO 32
120 LOCATE I,Y:PRINT "_=O=_"
130 LOCATE I,Y:PRINT "     "
133 A=I MOD 12
136 IF A=0 THEN BEEP
140 NEXT I
150 GOTO 5                <-- Changer
```

Avant de poursuivre la lecture, essayez de comprendre, par vousmême, ces quelques modifications...

Avez-vous compris? Les nouvelles lignes 5 et 7 apportent élégamment une solution à tous nos problèmes. En effet, la ligne 5 demande un arrière-fond noir et des lettres blanches au départ, tandis que la ligne 7 maintient le jeu au sommet de l'écran.

Cependant, rappelez-vous bien ceci: **un changement à un endroit occasionne souvent des changements en d'autres endroits.** Dans notre cas, l'instruction COLOR de la ligne 5 provenait de la ligne 100 qui doit donc être changée, simplement par CLS. De plus, comme le programme commence désormais à la ligne 5 au lieu de la ligne 10, la ligne 150 doit être modifiée, pour que les deux nouvelles lignes soient exécutées.

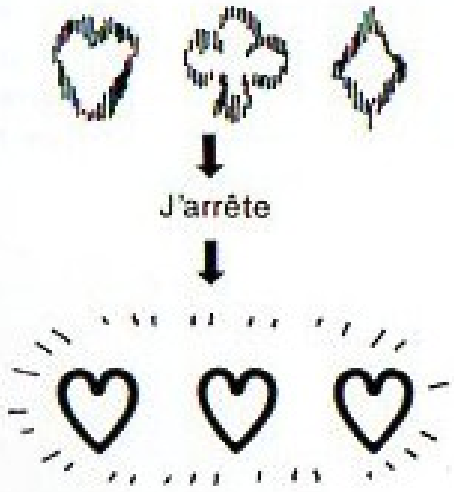
Maintenant, vous avez réalisé un bon programme, bien conçu. En êtes-vous satisfait? Ou souhaiteriez-vous le rendre encore plus intéressant?... Ce serait possible, par exemple, si les OVNI ne volaient pas toujours en ligne droite... Alors essayez! Votre ordinateur Sony est prêt à faire toutes sortes de choses, pourvu que vous lui disiez comment s'y prendre.

LE JEU DE L'APPAREIL A JETONS

Que faire pour....

- Programmer un jeu de machine à jetons qui...
  - —Change l'affichage jusqu'à ce que vous l'arrêtiez
  - —Compare les affichages
  - —Enregistre le score en fonction de votre mise
- Utiliser des variables en tableau
- Utiliser l'instruction ON-GOTO
- Utiliser des variables en chaine
- Arrêter un programme en appuyant sur une touche
- Déterminer un format d'impression
- Utiliser plus d'une condition dans une instruction IF—THEN
- Lister un long programme en sections

Lorsque, dans notre programme précédent, vous aviez deviné le nombre correct, votre récompense était un OVNI volant. A présent, votre ordinateur Sony va devenir un de ces "appareils à jetons", comme on en trouve dans les casinos de Las Vegas. Chaque fois que vous jouerez, vous gagnerez... ou vous perdrez! (Une lapalissade, direz-vous!) Certes, votre ordinateur ne débitera pas d'argent comme à Las Vegas, hélas, mais il tiendra parfaitement compte de votre score en points.



Voici les règles du jeu: tout d'abord, l'ordinateur vous demande combien de points vous souhaitez miser. Ensuite, il change rapidement les symboles affichés dans ses trois "fenêtres" jusqu'à ce que vous l'arrêtiez. A ce moment, il vous dit votre nouveau score: si les trois symboles sont les mêmes, vous gagnez trois fois votre mise. Si seulement deux sont les mêmes, vous récupérez votre mise. Mais si les trois sont différents, vous perdez le double de ce que vous aviez misé. Au début, vous disposez d'un enjeu de 100 points et vous gagnez si vous arrivez à 300. Par contre, si votre score tombe à zéro, vous perdez et le jeu est terminé.

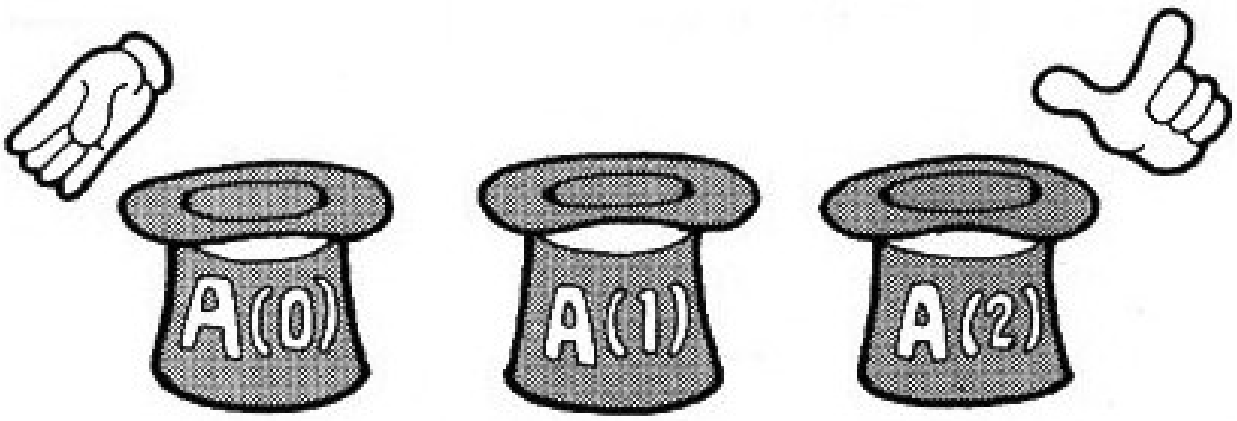
Etes-vous d'accord? Passons alors à la programmation!

La partie principale de ce jeu, ce sont les trois "fenêtres". Dans chacune d'elles, quatre symboles (♥, ♠, ♦, ♣) apparaissent et changent continuellement et avec une telle rapidité qu'il est impossible de les distinguer; votre seul recours est alors de deviner. Bien entendu, notre programme fera appel à des variables pour changer les symboles de chaque fenêtre.

QU'EST-CE QU'UNE VARIABLE EN TABLEAU?

Comme nous avons déjà programmé plusieurs fois des variables, nous savons comment une lettre ou un nom peut avoir une valeur changeante. Nous avons utilisé des noms tels que A, B et Q, et nous leur avons donné des valeurs qui changeaient à chaque fois qu'une boucle était répétée, ou en fonction de la position d'un OVNI, ou de façon aléatoire.

Cette fois, nous avons besoin d'une variable qui puisse devenir un quelconque des quatre symboles. Les trois fenêtres ont chacune le même jeu de symboles, ce qui revient à dire que nous pourrons utiliser la même variable pour les trois. Cependant, chaque fenêtre fonctionne séparément: parfois, les trois correspondront, le plus souvent, elles ne le feront pas. Pour cette situation, nous utiliserons notre variable en trois endroits différents, trios de la même variable. Chacune est appelée A, mais elle reçoit aussi un numéro (entre parenthèses), de manière à ce que nous puissions savoir quelle variable convient à chaque fenêtre.



A(0), A(1) et A(2) sont des **variables en tableau** (on parle aussi de variables indicées). Tout cela vous semble-t-il confus? Préféreriez-vous utiliser trois noms différents? Dans quelques instants, vous comprendrez que ces variables en tableau facilitent considérablement la programmation, parce qu'elles s'avèrent très flexibles.

Notre ensemble de variables a une **largeur** de trois variables et sa **longueur** équivaut au nombre de variables différentes que nous lui donnons. Nous pouvons penser qu'il s'agit de variables **multidimensionnelles**, ce qui contribuera d'ailleurs à retenir l'instruction en BASIC pour effectuer une variable en tableau, à savoir l'instruction **DIM**. Dans ce programme, nous allons utiliser la ligne d'instruction

```
10 DIM A(2)
```

pour effectuer les trois variables indicées A(0), A(1) et A(2). (D'autres exemples d'instructions pour variables indicées seraient: **DIM A(10)**, qui accomplit II variables de A(0) à A(10); ou DIM A(2), B(2), qui accomplit six variables, de A(0) à A(2) et de B(0) à B(2).)

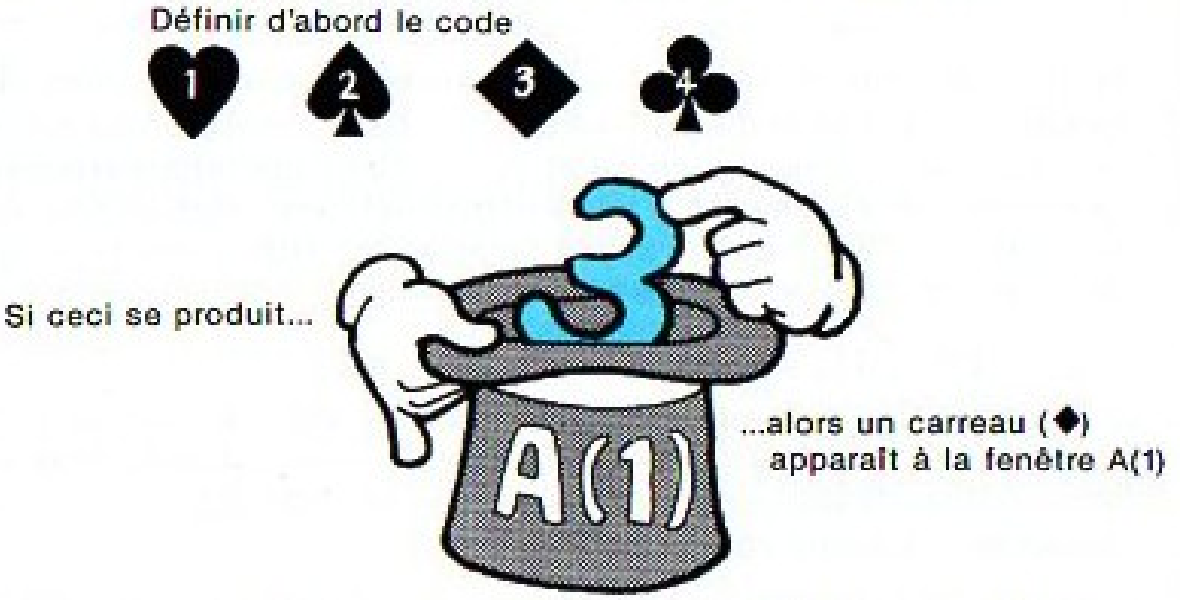
Autres chose à retenir à propos des variables indicées: le nombre entre parenthèses peut également devenir une variable, telle que A(I), ce qui en facilite la programmation.

POUR FAIRE UN APPAREIL A JETONS

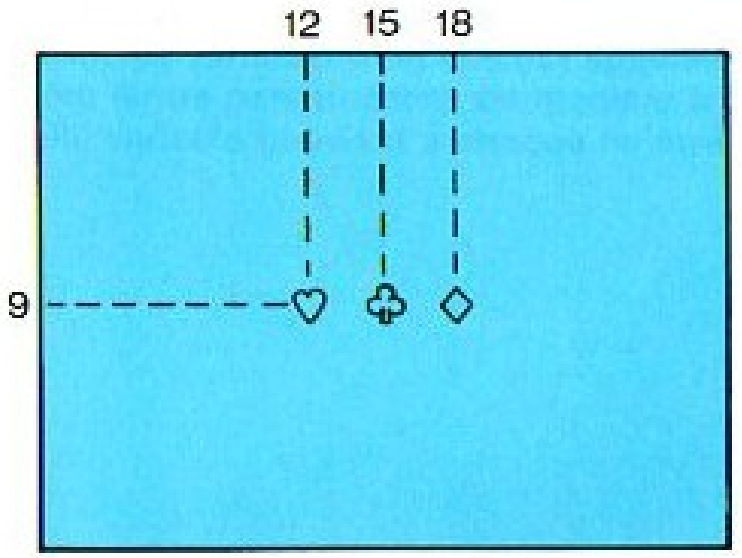
Nous voulons, avant tout, que nos variables indicées représentent les différents symboles du jeu de cartes, à savoir coeur (♥), pique (♠), carreau (♦) et trèfle (♣). La première chose dont nous avons besoin, ce sont des numéros de code pour ces divers symboles. Vous vous souvenez que nous avons utilisé des codes pour représenter les couleurs et votre ordinateur Sony sait déjà quel numéro de code correspond à quelle couleur. A présent, nous devons définir notre propre code pour les symboles.

♥...1 ♠...2
♦...3 ♣...4

Désormais, nous savons que lorsque la valeur de 3 est affectée à la variable A(1), par exemple, un carreau apparaîtra dans cette fenêtre.

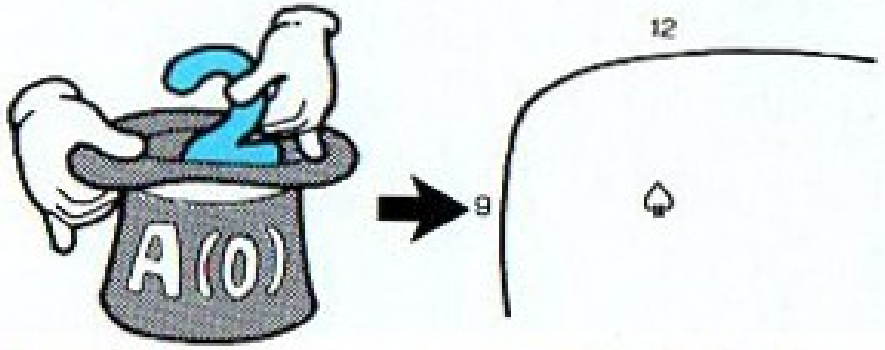


Ensuite, nous devons décider où se trouveront les fenêtres sur l'écran. Plaçons-les au centre.

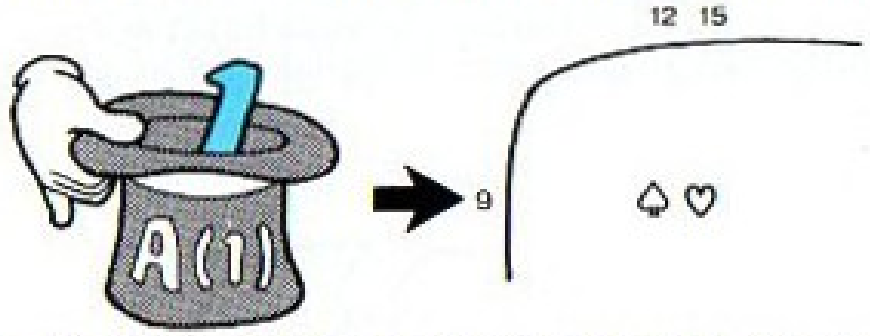


Nous avons placé la fenêtre A(0) en 12,9; la fenêtre A(1) se trouve en 15,9 et le fenêtre A(2) est placée en 18,9.

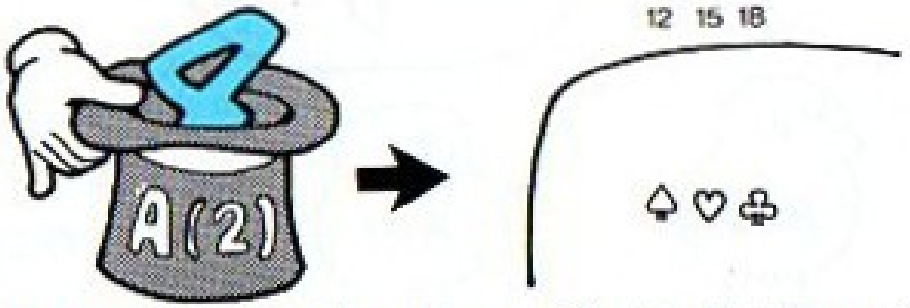
De cette façon, nous avons terminé la planification du programme pour notre "appareil à jetons". Passons maintenant en revue le système que nous venons de concevoir.



Si A(0) prend la valeur de 2, 2 est le code pour (♠) Ainsi, (♠) apparaît en 12,9.



Si A(1) prend la valeur de 1, 1 est le code pour (♥) Ainsi, (♥) apparaît en 15,9.



Si A(2) prend la valeur de 4, 4 est le code pour (♣) Ainsi, (♣) apparaît en 18,9.

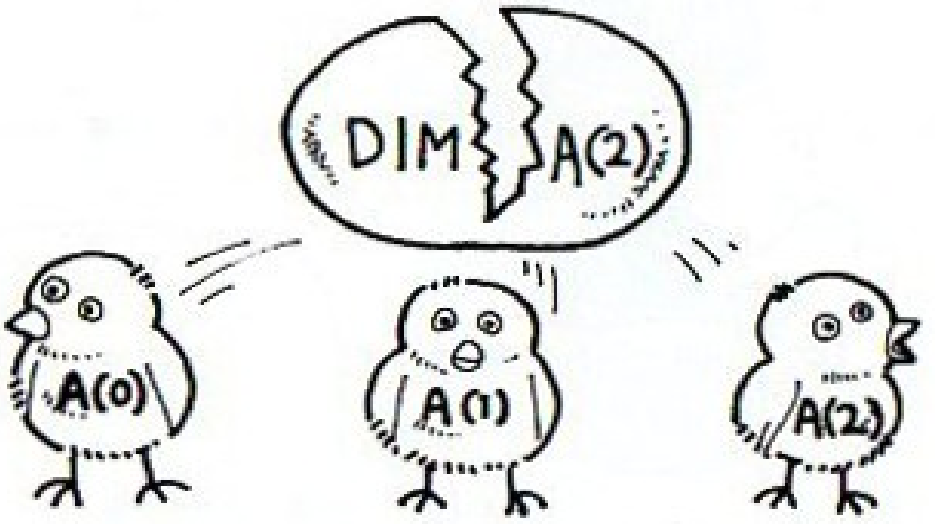
Ce système fait donc apparaître les symboles dans les fenêtres. Et comme chaque variable change très rapidement, aussi vite que les OVNI changeaient de position dans notre programme précédent, il vous sera impossible de distinguer clairement les symboles au passage et vous serez, par conséquent, obligé de deviner.

Nous avons planifié notre programme, nous en comprenons le déroulement; nous sommes prêts, par conséquent, pour les instructions.

```
10 DIM A(2)
20 CLS
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 110
80 PRINT "♠":GOTO 110
90 PRINT "♦":GOTO 110
100 PRINT "♣"
110 NEXT I
120 GOTO 30
```

Comme nous l'avons expliqué dans la section précédente, la ligne 10 crée nos trois variables en tableau. Chaque fois que des variables de ce genre sont utilisées, nous devons le signaler à l'ordinateur **au début** du programme.

```
10 DIM A(2)
```



Bien sûr, la ligne 20 a pour mission de déblayer l'écran. Quant à la ligne 30, elle commence par un FOR et nous savons que, si elle apparaît, elle doit toujours être accompagnée de NEXT quelque part dans le programme. La paire FOR-NEXT se trouve à la ligne 30 et à la ligne 110 et nous savons, par conséquent, que les instructions comprises entre les deux seront répétées sous forme de boucle.
Après FOR, la ligne 30 donne une nouvelle variable I qui sera utilisée entre parenthèses de la variable indicée.

La ligne 40 ne doit guère vous sembler nouvelle. Elle ressemble, en effet, à l'instruction de **nombre aléatoire** que nous avons déjà utilisée dans notre jeu des devinettes. Lorsque I prend la **valeur initiale** de 0, la ligne 40 devient

```
A(0)=INT(RND(1) * 4)+1
```

Ceci signifie que A(0) peut être n'importe quel nombre, compris entre 1 et 4, n'est-ce pas? A(0) correspond à la fenêtre de gauche et les chiffres 1-2-3-4 sont les codes attribués aux symboles ♥, ♠, ♦, ♣. En d'autres termes, lorsque I prend la valeur de 0, la ligne 40 dit à l'ordinateur de choisir un des 4 symboles pour la fenêtre de gauche.

Ensuite, il nous fallait dire à l'ordinateur où placer les trois fenêtres d'affichage sur l'écran.

```
50 LOCATE I * 3 + 12,9
```

Lorsque I a la valeur 0, pour la fenêtre de gauche, le curseur se trouve en 12,9. Quand I a la valeur de 1, pour la fenêtre du centre, l'emplacement deviendra 15,9. Enfin, pour la fenêtre de droite, I sera 2 et l'emplacement sera $2 \times 3 + 12,9$, c'est-à-dire 18,9.

A ce stade, l'ordinateur a choisi un symbole et le curseur s'est déplacé à la fenêtre voulue. Cependant, l'ordinateur n'affichera pas le symbole, à moins que nous ne lui disions de le faire. De là toute l'importance de la ligne 60.

```
60 ON A(I) GOTO 70,80,90,100
```

Ici, nous rencontrons l'instructions GOTO pour quatre lignes à la fois: 70, 80, 90, 100. Comment l'ordinateur pourra-t-il savoir à quelle ligne il doit aller? C'est la partie ON A(I) de cette instruction qui prend la décision. Voici d'ailleurs la forme qui est toujours employée pour une instruction ON-GOTO:

ON [quelque chose] GOTO [ligne 1] , [ligne 2] , [ligne 3] ...

Autrement dit, par exemple: si quelque chose est 1, aller à la ligne 1; si quelque chose est 2, aller à la ligne 2; si quelque chose est 3, aller à la ligne 3... Pour nous exprimer encore d'une autre façon, disons que la valeur de "quelque chose" décide automatiquement à quelle ligne doit se rendre l'ordinateur.

ON-GOTO est un agent de la circulation.

Pour donner un exemple, disons que A(0) a la valeur 3 (le code pour ♦). La ligne 60 donne ordre à l'ordinateur de passer à la ligne 90, quand la variable a cette valeur.

```
90 PRINT "♦":GOTO 110
```

Que se passe-t-il si 2 est la valeur pour la variable A(0)? Dans ce cas, la ligne 60 envoie l'ordinateur à

```
80 PRINT "♠":GOTO 110
```

De cette façon, nous comprenons comment l'ordinateur choisit un des quatre symboles et l'affiche dans une des fenêtres.

Après chacune des instructions PRINT, l'ordinateur passe à la ligne 110. (Il n'existe pas d'instruction GOTO 110 à la ligne 100 parce que l'ordinateur passe automatiquement à la ligne suivante.)

La ligne 110 dit à l'ordinateur de changer, de 0 en 1, la valeur de I. Ensuite, il retourne à la ligne 40. Pouvez-vous déduire ce qui de passera ensuite? Réfléchissez quelques instants au programme et essayez de comprendre par vous-même avant de lire les explications suivantes.

D'accord? Quand I est 1, la ligne 50 devient

```
LOCATE 1 * 3 + 12,9
```

ce qui signifie LOCATE 15,9. En d'autres termes, la position où le symbole sera affiché s'est déplacée de 12,9 à 15,9, c'est-à-dire de la fenêtre gauche à la fenêtre centrale. Un des symboles apparaîtra sur l'écran en 15,9. Ensuite, l'ordinateur changera I en 2 et il reviendra à la ligne 40. Et cette fois, la position deviendra 18,9 ou la fenêtre de droite. En moins d'une seconde, l'ordinateur a fait trois fois le "tour" du programme.

Ceci est ce qui apparaîtra peut-être sur l'écran, mais nous ne pourrons jamais savoir quels symboles l'ordinateur choisira au hasard.

Les lignes de 30 à 120 forment une boucle. Votre ordinateur en fait le tour de façon répétée, se servant des variables pour choisir et afficher les symboles dans la fenêtre de gauche, du centre et de droite... Etant donné que le programme forme une boucle sans fin, les symboles apparaissent en "mouvement rotatif" constant, comme sur un véritable appareil à jetons d'un casino.

Si vous n'avez pas encore essayé votre programme, entrez-le à présent dans votre ordinateur en veillant soigneusement à ne pas commettre de fautes. Lorsque vous donnerez l'instruction RUN, vous constaterez que nous sommes parvenus à créer un appareil à jetons en nous servant de variables en tableau (dans le mode SCREEN 0, cependant, le côté droit de chaque symbole n'est pas affiché).

Que faut-il faire pour arrêter cette boucle sans fin et connaître le score obtenu? Et comment l'ordinateur connaît-il le score exact? Pour que notre machine à jetons devienne un vrai jeu de hasard, nous devons ajouter quelques instructions au programme. A cet effet, nous allons faire appel à certaines variables qui comportent des lettres au lieu des valeurs numériques, ce que l'on appelle des **variables en chaîne**. Sur ce point, quelques explications complémentaires ne feront pas de tort.

DES VARIABLES EN CHAINE

Nous avons déjà vu qu'une variable peut prendre la valeur de n'importe quel **nombre**, tel que 1, 2, 3, 191 ou 252. Les variables peuvent **également** recevoir la valeur de n'importe quelle **lettre**, ou d'une suite de lettres qui forment une **chaîne**.

Effectuons quelques exercices pour voir comment une variable s'emploie pour signifier une lettre ou une chaîne. Tout d'abord, tapez **PRINT A$** et appuyez sur [RETURN].

```
PRINT A$

Ok
■
```

Rien ne se passe. Aucune tonalité et pas de message d'erreur, indiquant une faute de frappe, mais l'ordinateur n'imprime rien de tout. Entrons à présent cette instruction:

```
A$="ABC"
PRINT A$
```

Appuyez sur [RETURN] et votre écran devrait montrer ce qui suit:

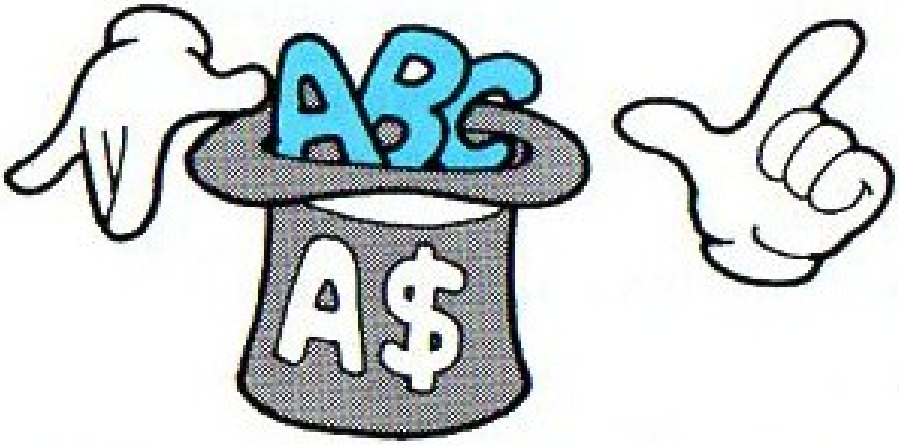
```
A$="ABC"
Ok
PRINT A$
ABC
Ok
■
```

Vous pouvez voir que A$ est utilisé comme variable et que sa valeur est "ABC". Comparons-la avec une autre instruction:

```
A=345
Ok
PRINT A
 345
Ok
■
```

Ceci n'a guère de secret pour nous, n'est-ce pas? La variable A prend la valeur du nombre 345 et il n'y a plus de symbole $ (signe du dollar) à la fin du nom de variable A.

Avez-vous deviné la règle? **Lorsque la variable se termine par un signe $, elle a la valeur d'une lettre ou d'une suite de lettres (appelée une "chaîne"). Sans le signe $ à la fin, la variable prend la valeur d'un nombre.** (On dit aussi variable alphanumérique ou variable chaîne de caractères dans le premier cas, et variable numérique dans le second.)



Autre chose importante à ne pas oublier: la valeur doit se trouver entre **guillemets**, comme ceci:

A$="ABC"

Essayez ces exemples et vous comprendrez clairement la différence entre des variables numériques et des variables alphanumériques, dites aussi variables chaîne de caractères.

```
A=123                    A prend la valeur de 123
Ok
B=234                    B prend la valeur de 234
Ok
PRINT A+B                Affiche A+B
 357                     La somme est 357
Ok
A$="123"                 A$ prend la valeur de "123"
Ok
B$="234"                 B$ prend la valeur de "234"
Ok
PRINT A$+B$              Affiche A$+B$
123234                   La somme est "123234"
Ok
A$=987                   A$ prend la valeur de 987
Type mismatch            Bip! Pas possible.
Ok
■
```

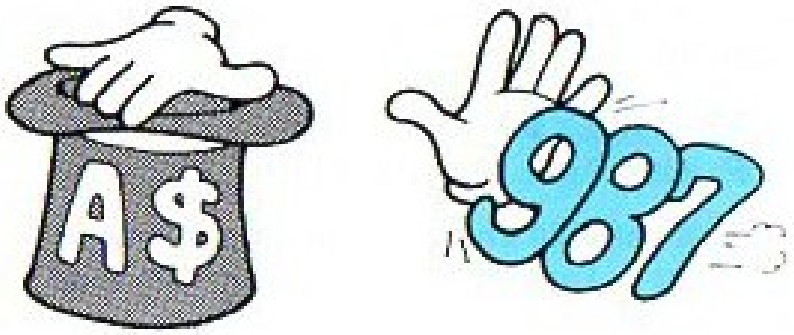
Lorsque nous entrons A$="123", la valeur de la variable A$ devient 123, non pas le nombre cent-vingt-trois, mais les **caractères** un-deux-trois. A la ligne suivante, B$ prend la valeur de 234 (les caractères deux-trois-quatre). Pourquoi? Parce que nous avons utilisé les guillemets et le signe $. Ceci a fait savoir à l'ordinateur que nous voulions utiliser une variable alphanumérique (une variable chaîne de caractères); par conséquent, il a lu 123, non pas comme un nombre, mais comme des carctères, tout comme s'il s'agissait de lettres.

Lorsque l'ordinateur ajoute A$+B$, il les réunit ensemble pour former une suite de caractères. Un exemple tout simple serait "ABC"+"BCD"= "ABCBCD".

Finalement, nous avons entré A$ = 987 et l'ordinateur nous a répondu par un message d'erreur:

Type mismatch (vous avez tapé deux choses qui ne concordent pas)

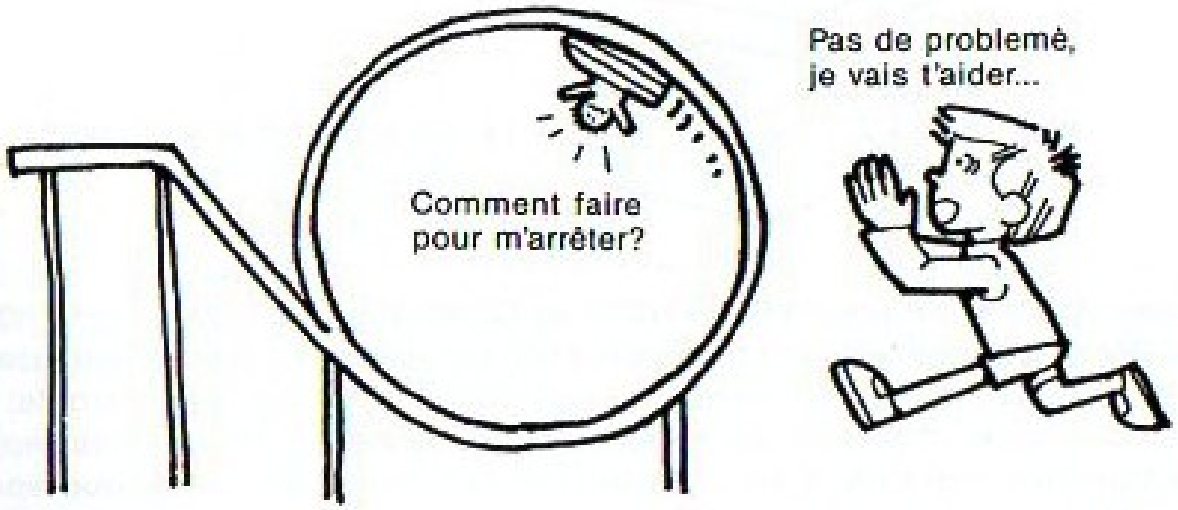
Sans les guillemets, 987 est un nombre (neuf-cent-quatre-vingt-sept) qui **ne concorde pas** avec la variable chaîne de caractères, dotée du signe $.



A présent, nous savons que si le nom d'une variable se termine par un signe $, il s'agit d'une variable chaîne (alphanumérique), qu'elle doit être affectée d'une lettre ou d'une suite de lettres, et que la valeur doit se trouver entre "guillemets". Maintenant, revenons à notre programme du jeu de l'appareil à jetons.

POUR ARRETER UNE BOUCLE PAR INKEY

La boucle FOR-NEXT, utilisée dans notre jeu des devinettes, s'était arrêtée d'elle-même lorsque sa variable avait atteint la dernière valeur. Cependant, la ligne 120, l'instruction GOTO 30, signifie que notre boucle ne s'arrêtera jamais dans le cas du jeu de l'appareil à jetons.



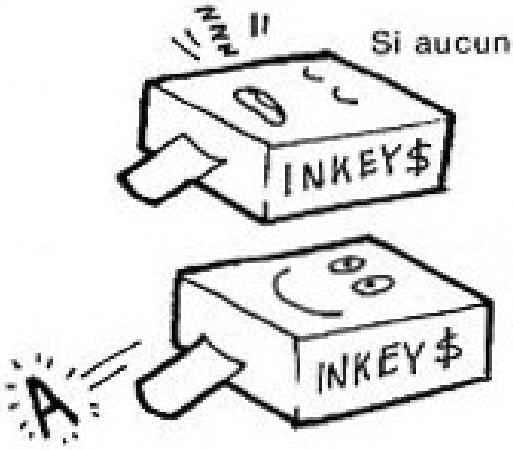
Evidemment, il y a toujours la solution d'appuyer sur CTRL et STOP pour arrêter l'ordinateur. Mais ceci arrête tout le programme et si nous donnons l'instruction RUN pour le remettre en marche, nous retombons dans notre "cercle vicieux", une boucle sans fin. Or, nous souhaitons arrêter la machine pour regarder les symboles, pour voir s'ils correspondent et pour savoir combien de points nous avons gagné ou perdu. Ensuite, nous voulons que le programme puisse recommencer, pour continuer à jouer.

Un bon moyen d'arrêter notre machine infernale est d'ajouter ces deux lignes au programme:

```
103 K$=INKEY$
106 IF K$="A" THEN END
```

Bien sûr, K$ est une variable chaîne, et nous pouvons voir à la ligne 106 que si la valeur K$ est A, l'appareil mettra une commande END à son affichage rotatif.

Que signifie alors **INKEY$**? Il s'agit d'une fonction du langage BASIC, formée de la première partie des mots "input" from the "keyboard" (en français: entrée par le clavier), c'est-à-dire n'importe laquelle des touches actionnées. La ligne 103 signifie, par conséquent, que **K$ prend la valeur de n'importe quelle touche de lettre actionnée**. Si vous n'appuyez pas sur une touche, l'ordinateur continuera, sans donner une valeur à K$. Si vous appuyez sur [B], ou [C] ou [X], l'ordinateur continuera de faire tourner l'appareil à jetons; mais si vous appuyez sur [A], la ligne 106 ordonnera à l'ordinateur de terminer (END).



Si aucune touche n'est actionnée, je ne fais rien.

Si [A] est actionné, je donnerai A.

Ainsi donc, notre boucle FOR-NEXT et GOTO 30, qui va de la ligne 30 à la ligne 120, possède l'instruction END en son sein. Mais l'ordinateur n'entendra pas l'instruction (et il n'arrêtera pas la machine infernale) si vous ne le lui dites pas, en actionnant la touche [A]. En d'autres mots, la machine continuera de changer les symboles jusqu'à ce que vous l'arrêtiez.

```
30 FOR I=0 TO 2
        !
        !
        !
        !
        !
        !
        !
103 K$=INKEY$
106 IF K$="A" THEN END
110 NEXT I
120 GOTO 30
```

La boucle FOR-NEXT qui affiche les 3 symboles dans les fenêtres

Le programme s'arrête (END) si la touche [A] est actionnée.

QUEL EST LE SCORE?

Lorsque l'appareil à jetons s'arrête de changer les symboles du jeu de cartes, nous souhaitons savoir combien de points nous avons gagné (ou perdu!). Cela signifie que nous devons placer les règles dans le programme. Les avez-vous encore en tête? Nous avions commencé avec un **enjeu** de 100 points. Ensuite, nous avions misé une partie de notre avoir, un nombre quelconque de 1 à 100. Ajoutons cette information à notre programme.

```
23 P=100:LOCATE 2,18:PRINT USING "ENJ
EU:####";P
26 LOCATE 3,20:INPUT "MISE";B
```

Observons tout d'abord la ligne 23. Comme notre enjeu total changera chaque fois que nous aurons effectué notre mise, nous allons utiliser une variable, P. L'enjeu P a une valeur initiale de 100 points. Pour afficher le score, nous utilisons l'instruction LOCATE, puis l'instruction PRINT.

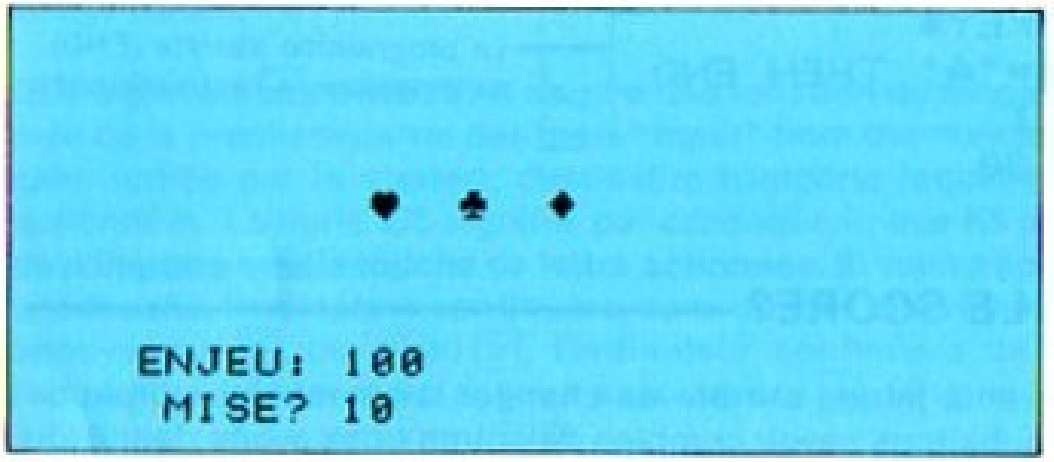
```
PRINT USING "ENJEU:# # # #";P
```

Cette nouvelle instruction PRINT dit à l'ordinateur comment **éditer** ou mettre en forme l'information qu'il affiche. Le signe # indique des "digits" et l'instruction dit donc ceci à l'ordinateur: afficher P en utilisant les quatre espacements vides sur l'écran après "ENJEU:". Comme le jeu commence avec un enjeu de 100 points, le premier affichage en 2,18 sera donc:

ENJEU: ␣ 100

4 chiffres

La ligne 26 dit à l'ordinateur d'entrer votre mise sur l'écran, juste en-dessous de l'enjeu. Votre mise changera chaque fois et, par conséquent, cette ligne utilise la variable B pour indiquer votre mise. L'instruction INPUT ordonne à l'ordinateur d'attendre que la valeur de B, votre mise, soit entrée par l'intermédiaire du clavier. Si vous misez 10 points, cette valeur sera affichée comme suit



Après les lignes 23 et 26, l'ordinateur passe à la section FOR-NEXT. Entrons, à présent, les nouvelles lignes d'instruction 23, 26, 103 et 106 dans notre programme.

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "ENJ
EU:#####";P
26 LOCATE 3,20:INPUT "MISE";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$="A" THEN END
110 NEXT I
120 GOTO 30
```

↑ Ajouter

← Changer

← Ajouter

Remarquez que nous devons changer l'instruction GOTO 110 des lignes 70, 80 et 90 en GOTO 103. Savez-vous ce qui se passerait si nous oubliions ce point?

A présent, la machine est prête pour recevoir votre mise, mais nous n'avons pas encore dit à l'ordinateur comment changer notre enjeu, c'est-à-dire comment tenir compte du score. Nous allons ajouter certaines nouvelles instructions à la fin du programme, en commençant par la ligne 130. Cependant, revenons tout d'abord à la ligne 106.

```
106 IF K$="A" THEN END
```

Lorsque nous disons à la machine d'arrêter de changer les symboles, en appuyant sur la touche [A], nous ne souhaitons pas pour autant que le programme s'achève (END). Nous désirons plutôt que l'ordinateur nous dise notre score. C'est pourquoi nous devrons changer comme suit la ligne 106:

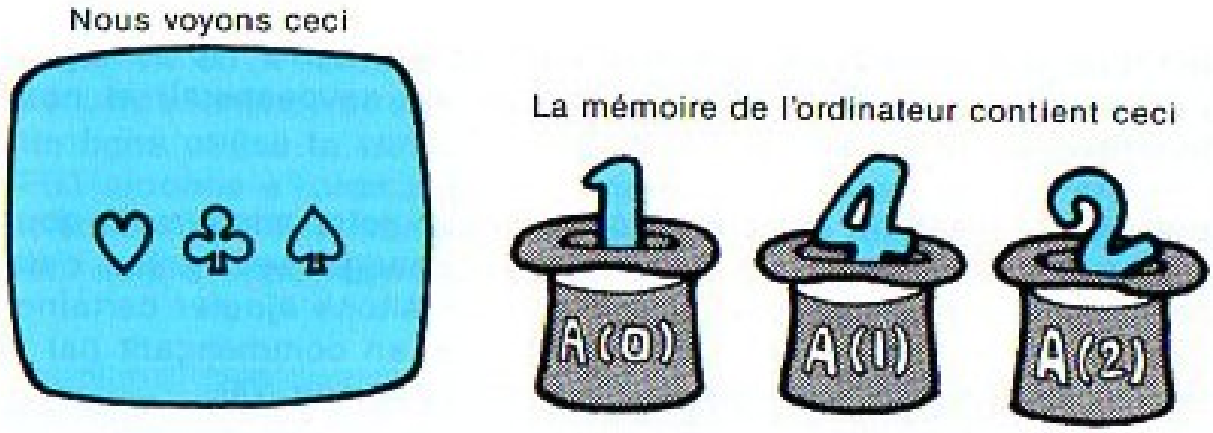
```
106 IF K$="A" THEN GOTO 130
```

Ainsi nous saurons si nous avons gagné ou perdu en regardant les symboles sur l'écran. Si les trois sont identiques, nous triplons notre mise (comme notre enjeu était de 100 et que nous avions misé 10 points, notre score sera de 130 si les trois symboles sont les mêmes). Si deux des symboles sont les mêmes, nous gagnons notre mise (et le nouveau score devient alors 110). Par contre, si les trois symboles sont différents, nous perdons deux fois notre mise (et le nouveau score devient 80, dans ce cas). Ainsi, par exemple, si nous voyons

♥ ♣ ♠

nous pouvons en déduire que nous avons perdu des points.

Mais notre ordinateur ne regarde pas les symboles sur l'écran comme nous. Il utilise les variables qui se trouvent dans sa mémoire et, à l'instant, il sait si A(0), A(1) et A(2) sont égaux en vérifiant les valeurs de ces variables indicées.



Si l'écran affiche:

♥ ♥ ♥

Savez-vous ce qui se trouvera dans la mémoire de l'ordinateur?

Comme vous le voyez, il n'est guère difficile pour l'ordinateur de se rendre compte si deux, trois ou aucun des symboles sont identiques et ceci revient à dire que les instructions de notre programme, relatives au score, ne sont pas très complexes.

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 8,18:PRINT USING "####";P
160 GOTO 26
```

La première ligne de ces nouvelles instructions de calcul du score est la ligne 130. Par conséquent, la partie du programme qui concerne le score ne sera utilisée qu'après une action sur la touche [A] et l'arrêt de la machine à sous.

```
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
```

Remarquez que l'instruction IF-THEN a **deux** conditions, connectées avec **AND**. Si **toutes les deux** sont vérifiées, c'est-à-dire A(0)=A(1) et A(0)=A(2), alors toutes les trois variables indicées sont égales et les trois symboles doivent être identiques sur l'écran. Si ceci est vrai

```
THEN P=P+B*3:GOTO 150
```

Ceci revient à dire que notre mise sera multipliée par 3 et qu'elle sera ajoutée à l'enjeu P; puis, l'ordinateur passera à la ligne 150. Si vous avez misé 10 points avec un enjeu de 100, le nouveau score devient 130.

Par contre, si les trois variables indicées ne sont pas les mêmes, l'ordinateur ne suivra pas l'instruction THEN, mais il passera, en revanche, à la ligne suivante du programme.

```
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
```

Nous avons ici une instruction IF comportant trois conditions et elles sont réunies par **OR**. Ceci signifie que si **une** paire quelconque des variables égale, (ou bien, si deux symboles sont égaux),

```
THEN P=P+B
```

votre mise est ajoutée à l'enjeu et le nouveau score devient 110. Par ailleurs, si aucune des trois conditions n'est remplie (si tous les symboles sont différents), l'ordinateur passera à

```
ELSE P=P-B*2
```

Et vous perdez! Autrement dit, votre mise est multipliée par 2 et est soustraite à l'enjeu initial, le nouveau score devenant donc 80.

Désormais l'ordinateur connaît le score et il doit ensuite l'afficher.

```
150 LOCATE 8,18:PRINT USING "####";P
```

Cette ligne est facile à comprendre parce qu'elle ressemble beaucoup à l'instruction de format PRINT de la ligne 23: d'abord LOCATE, puis le format PRINT USING des quatre digits, et ensuite le nom de la variable qui accepte une nouvelle valeur. Dès lors, chaque fois que l'on arrêtera la machine à jetons, l'ordinateur calculera le nouveau score et il l'affichera à la position 8,18.

Que se passe-t-il ensuite? Le jeu continue et, par conséquent, vous devez faire une nouvelle mise. La ligne 160 renvoie l'ordinateur à la ligne 26 où il attend que vous entriez votre mise par l'intermédiaire du clavier.

Entrez à présent les nouvelles instructions de calcul du score dans le programme qui doit se présenter comme suit:

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "ENJ
EU:####";P
26 LOCATE 3,20:INPUT "MISE";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 K$=INKEY$
106 IF K$="A" THEN GOTO 130
110 NEXT I
120 GOTO 30
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 8,18:PRINT USING "####";P
160 GOTO 26
```

Changer

Ajouter

Attendez un instant! Ce programme est trop long pour apparaître entièrement sur l'écran et la première partie disparaît lorsque vous entrez l'instruction LIST. Pour parer à cette difficulté, nous devons dire à l'ordinateur **ce qu'il doit** lister, de cette façon:

```
LIST 10-100
```

Avec cette instruction, les lignes de 10 à 100 sont affichées. Pour voir la dernière partie, entrez

```
LIST 110-
```

Utilisez maintenant les instructions LIST pour vérifier très soigneusement si vous n'avez pas fait d'erreurs. Si vous êtes certain que le programme est correct, donnez l'instruction RUN et vous pouvez alors vous amuser avec votre nouveau jeu. Bonne chance!

LES TOUCHES DE FINITION

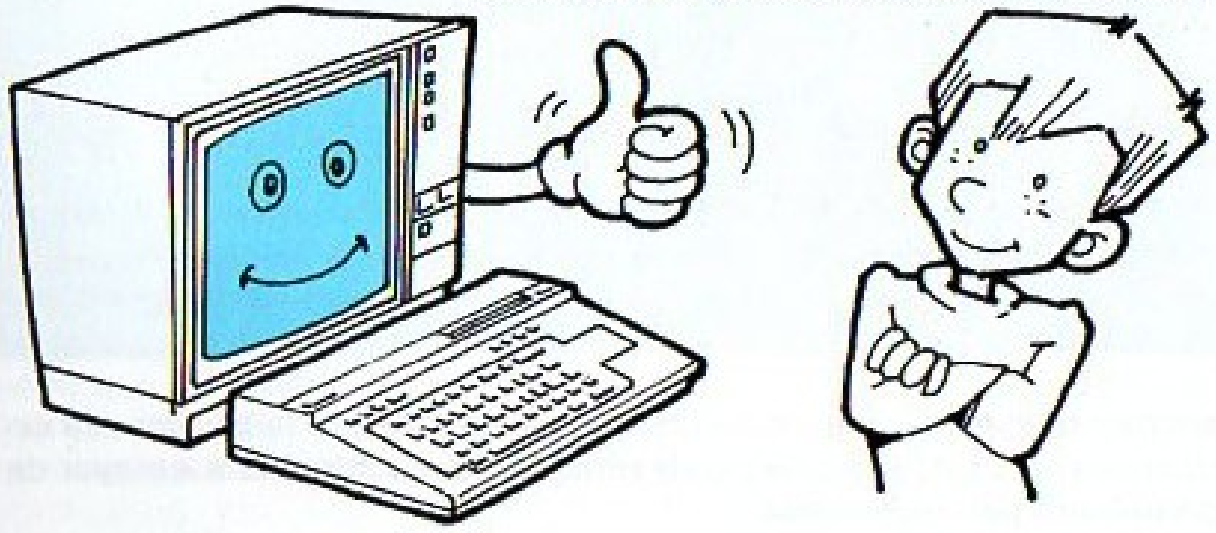
Votre ordinateur Sony s'est transformé en une machine à jetons qui fonctionne très bien. Il accomplit certaines choses relativement complexes: il vous demande de miser, change les symboles du jeu de cartes dans les trois fenêtres et calcule votre score lorsque vous décidez d'arrêter les symboles. Pour toutes ces diverses choses, notre programme ne comporte que 20 lignes d'instructions, ce qui n'est pas beaucoup quand on pense qu'il est nécessaire de dire **chaque fois** à un ordinateur qu'il doit afficher, ou calculer, ou s'arrêter ou attendre la mise, etc.

Mais il manque encore quelque chose. En effet, au début, nous avions dit que vous gagneriez lorsque votre score atteindrait 300, ou que vous perdriez s'il tombait jusqu'à 0. Si vous avez joué quelques instants sur la machine, votre score doit avoir dépassé 300 ou, si vous êtes malchanceux, être retombé à moins de 0, ou peut-être les deux, successivement.

Pourquoi ne pas améliorer encore ce bon programme, en ajoutant quelques instructions qui décideront de la victoire ou de la défaite. En même temps, nous apporterons certaines autres améliorations pour en faire un programme **parfait**.

- Nous pouvons rendre le jeu plus intéressant en permettant d'effectuer une autre mise différente, seulement après avoir actionné la barre d'espacement du clavier.
- Parce que sa position est plus centrale sur le clavier, il serait plus pratique d'uitliser la barre d'espacement au lieu de la touche [A] pour arrêter les symboles.
- Après avoir effectué une nouvelle mise, l'ancienne se trouve encore sur l'écran, ce qui est quelque peu embrouillé.

Encore un effort et le programme sera parfait!



Voici les changements qu'il conviendrait d'apporter à notre programme.

```
10 DIM A(2)
20 CLS
23 P=100:LOCATE 2,18:PRINT USING "ENJ
EU:####";P
24 LOCATE 7,20:PRINT "          "      <-- Ajouter
26 LOCATE 3,20:INPUT "MISE";B
30 FOR I=0 TO 2
40 A(I)=INT(RND(1)*4)+1
50 LOCATE I*3+12,9
60 ON A(I) GOTO 70,80,90,100
70 PRINT "♥":GOTO 103
80 PRINT "♠":GOTO 103
90 PRINT "♦":GOTO 103
100 PRINT "♣"
103 A$=INKEY$
106 IF K$=" " THEN GOTO 130
110 NEXT I          <-- Changer
120 GOTO 30
130 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 150
140 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
150 LOCATE 8,18:PRINT USING "####";P
152 GOTO 170
154 K$=IKEY$                          <-- Ajouter
156 IF K$=" " THEN GOTO 24
160 GOTO 154                          <-- Changer
170 IF P<300 AND P>0 THEN GOTO 154
180 IF P>=300 THEN LOCATE 3,5:PRINT "
VOUS AVEZ GAGNE!"
190 IF P<=0 THEN LOCATE 3,5:PRINT "VO
US AVEZ PERDU!"
200 END
                                      <-- Ajouter
```

Comprenez-vous ces modifications? Avant de passer à la lecture des explications suivantes, veuillez consacrer quelques minutes à essayer de comprendre par vous-même.

La ligne 170 commence un nouveau jeu d'instructions, concernant votre victoire ou votre défaite. Si le score est inférieur à 300 ET AND supérieur à 0, le jeu continue et l'ordinateur répète le programme. Si le score est 300 ou supérieur, le jeu est terminé et "VOUS AVEZ GAGNE !" apparaît dans le haut de l'écran. Si le score est inférieur à 0, "VOUS AVEZ GAGNE!" Puis, que vous ayez gagné ou perdu, l'ordinateur passe à la ligne 200 et le jeu arrive à la fin (END).

La ligne 24 efface la mise précédente au début de chaque manche du jeu. Elle déplace le curseur à l'endroit où est affiché la mise et, à la place, elle crée six espaces vides.

La ligne 106 présente un changement tout simple. Nous voulons arrêter la machine par la barre d'espacement et non plus par la touche [A] et il convient de se souvenir ici que l'ordinateur considère un **espacement** (" ") comme un **caractère**.

Les lignes 160, 154 et 156 signifient que le jeu recommencera lorsque nous appuyons sur la barre d'espacement.

Tels sont donc les quatre changements que nous souhaitons apporter: victoire/défaite, et les trois améliorations mineures.

La ligne 152 ne change pas réellement le jeu, mais elle est nécessaire en raison des autres modifications, apportées au programme. Cette ligne se trouve juste après que le nouveau score est affiché et elle fait passer l'ordinateur en aval à la nouvelle section victoire/défaite de la ligne 170. Aussi longtemps que vous n'avez pas gagné ou perdu, la ligne 170 ramène le programme à la ligne 154 et le jeu continue lorsque vous appuyez sur la barre d'espacement.

POUR FACILITER LA LECTURE D'UN PROGRAMME

Au début de la rédaction de notre programme de la machine à jetons, nous avons utilisé les numéros des lignes en dizaines: 10, 20, 30, 40... Ensuite, pour compléter le programme, nous avons été amenés à utiliser certaines des lignes "libres" pour ajouter de nouvelles instructions, au point que la numérotation des lignes est devenue quelque peu confuse: 90, 100, 103, 106, 110, 120. Pour faciliter la lecture de ce programme, rien ne nous empêche de **renuméroter** les lignes.

Entrons cette instruction:

```
RENUM
```

Les lignes restent dans le même ordre, mais leur numéros changent en dizaines comme nous pouvons le voir en donnant l'instruction LIST:

De cette façon, les numéros des lignes sont bien plus faciles à lire. Mais qu'est-il advenu de nos instructions GOTO qui comportaient des numéros de ligne? Pas d'inquiétude à cet égard! En effet, observez, par exemple la ligne 100 du nouveau programme:

```
100 PRINT "♥":GOTO 140
```

Cette instruction GOTO (GOTO 140) était GOTO 103 avant d'utiliser l'instruction RENUM. A présent, la ligne 103 est devenue la ligne 140 et GOTO 103 a été changé en GOTO 140. Par conséquent, parce que votre ordinateur Sony est malin, le jeu ne sera pas changé lorsque vous utiliserez, dorénavant, le programme de votre machine à jetons.

INSERTION DE TITRES DANS UN PROGRAMME

Supposons que vous veuillez montrer votre programme à un ami, ou que vous souhaitiez le relire, plusieurs mois après l'avoir rédigé. Pour une personne peu avertie ou pour quelqu'un à qui vous ne désirez pas dévoiler vos "mots de passe", la lecture des vingtaine ou trentaine de lignes doit certainement être quasi incompréhensible. Pour parer à cette difficulté, le langage BASIC vous permet d'insérer des "remarques" au milieu d'un programme, comme ceci:

```
5 REM **JEU DE LA MACHINE A JETONS**
7 REM
10 DIM A(2)
20 CLS
30 P=100:LOCATE 2,18:PRINT USING "ENJ
EU:####";P
40 LOCATE 7,20:PRINT "      "
50 LOCATE 3,20:INPUT "MISE";B
55 '
60 ' **MISE EN MARCHE DE LA MACHINE**
62 '
```

```
65 FOR I=0 TO 2 ---Ligne précédente 60
70 A(I)=INT(RND(1)*4)+1
80 LOCATE I*3+12,9
90 ON A(I) GOTO 100,110,120,130
100 PRINT "♥":GOTO 140
110 PRINT "♠":GOTO 140
120 PRINT "♦":GOTO 140
130 PRINT "♣"
140 K$=INKEY$
150 IF K$=" " THEN GOTO 180
160 NEXT I
170 GOTO 60
175 '
180 ' **CALCULER LA MISE **
182 '
185 IF A(0)=A(1) AND A(0)=A(2) THEN P
=P+B*3:GOTO 200------Ligne précédente 180
190 IF A(0)=A(1) OR A(1)=A(2) OR A(0)
=A(2) THEN P=P+B ELSE P=P-B*2
200 LOCATE 8,18:PRINT USING "####";P
210 GOTO 250
220 K$=INKEY$
230 IF K$=" " THEN GOTO 40
240 GOTO 220
245 '
250 ' **GAGNE OU PERDU**  Ligne précédente 250
252 '
255 IF P<300 AND P>0 THEN GOTO 220
260 IF P>=300 THEN LOCATE 3,5:PRINT "
VOUS AVEZ GAGNE!"
270 IF P<=0 THEN LOCATE 3,5:PRINT "VO
US AVEZ PERDU!"
280 END
```

Nos "remarques" constituent les **titres** du programme ou **les soustitres** de ses diverses parties. Si elles allongent le programme, elles en facilitent aussi la lecture et la compréhension. De plus, elles ne modifient nullement le jeu puisque l'ordinateur ne les lit pas; leur seule raison d'être est la facilité de celui ou de ceux qui utilisent le programme.

Examinons de plus près certaines des nouvelles lignes de remarques:

```
5 REM **JEU DE LA MACHINE A JETONS**
7 REM
```

L'instruction REM dit à l'ordinateur d'**ignorer** les caractères qui suivent l'instruction proprement dite. Vous pouvez donc y insérer ce que vous voulez, sans changer nullement le programme. A la ligne 5, nous trouvons donc le titre de ce programme. La ligne 7, par contre, est vide après l'instruction REM. De cette façon, vous ménagez un certain espace vierge autour du titre, un peu comme pour la page du titre d'un livre. Cet espace libre et les astérisques avant et après le titre facilitent la localisation de ce dernier parmi toutes les lignes des instructions du BASIC.

Regardons, pour suivre, les lignes 55, 60 et 62. Vous remarquez ici qu'il existe une façon plus brève pour donner l'instruction REM, par ' (l'apostrophe). Bien entendu, ces lignes ne font pas réellement partie du programme et l'ordinateur ne les lit pas parce qu'elles commencent par '. Les sous-titres, les astérisques et les espaces vierges sont destinés aux êtres humains, et non à l'ordinateur.

L'ordinateur **ne voit pas** les numéros de ligne parce qu'ils précèdent le ' (ou REM). Ainsi par exemple, à la ligne 150, nous voyons:

```
150 IF K$=" " THEN GOTO 180
```

Mais la ligne 180 est une remarque:

```
180 ' **CALCULER LA MISE**
```

Comme l'ordinateur ne rencontre pas d'instruction à la ligne 180, il passe à la suivante. Puisque la ligne 182 ne comporte pas non plus d'instruction, il continue à la ligne 185. Par contre, le **lecteur**, c'est-à-dire vous-même ou votre ami, passe de la ligne 150 à la ligne 180 et sait immédiatement que cette section a pour but de calculer la mise.

QUE DE CHEMIN PARCOURU!

Toutes nos félicitations! Après avoir effectué les exercices et avoir utilisé les jeux, proposés dans cet ouvrage, vous avez déjà acquis une bonne connaissance d'un nouveau langage, le BASIC. En quelques jours à peine, voire quelques heures peut-être, vous vous êtes donc familiarisé avec des instructions très simples, comme PRINT 3 + 5. Désormais, vous êtes à même de comprendre et d'**utiliser** des lignes complexes, telles que A(I)=INT(RND(1) * 4) + 1, ou PRINT USING "# # # #";P. Maintenant que vous avez appris son langage, vous pourrez dialoguer avec votre ordinateur Sony qui, pendant de longues années, vous rendra des services fidèles, tout en contribuant à votre amusement et à votre instruction.

A mesure que vous vous exercerez à la rédaction et à l'utilisation des programmes, vous parviendrez à jongler avec les chiffres, les graphiques, les instructions, les fonctions et les jeux. Ce n'est plus qu'une question de temps pour que vous deveniez un **spécialiste** et sachez que vos progrès seront amplifiés si vous les partagez avec votre famille et vos connaissances.

Nous vous invitons à continuer la pratique des instructions et à en apprendre quelques nouvelles dans la partie suivante de cet ouvrage. Par la suite, en consultant le **Manuel de référence** de la programmation en MSX-BASIC, vous pourrez découvrir toute une nouvelle série d'instructions en BASIC. Elles vous permettront de vous livrer à diverses activités avec l'ordinateur, en combinant couleurs, graphiques, sons et jeux.

Avant de créer **vos propres programmes**, nous vous conseillons de passer quelques instants à leur planification et, éventuellement, à la rédaction d'un organigramme. Une fois que vous avez déterminé quelles **fonctions** et quels types d'**instructions** vous utiliserez, référez-vous à l'index alphabétique à la fin de ce manuel pour trouver et revoir les explications. Procédez selon votre plan, écrivez certaines commandes et exécutez-les. Votre ordinateur Sony vous aidera de deux façons: il effectuera toujours avec **exactitude** ce que vous lui demanderez de faire et il oubliera vos erreurs éventuelles si vous utilisez l'instruction NEW.

Discutez de vos programmes avec vos amis et montrez leur ce que vous pouvez accomplir avec votre ordinateur car deux idées valent souvent mieux qu'une et leurs conseils vous seront utiles. Si, par bonheur, ils disposent également d'un ordinateur, vous pourrez échanger vos programmes, soit en vous servant d'un magnétophone ou en les écrivant sur papier.

Souvenez-vous que le meilleur moyen de vous amuser réellement avec votre ordinateur Sony est de le mettre à contribution pour tenter de nouvelles choses. Il n'existe aucune limite à ce que vous pouvez dire à l'ordinateur en langage BASIC. Il nous reste à vous souhaiter un excellent voyage dans le monde merveilleux de votre imagination. Bonne route!

PRATIQUES DES INSTRUCTIONS EN BASIC

PRINT

```
10 PRINT "JEAN"
20 PRINT "ET"
30 PRINT "MARIE"
RUN
JEAN
ET
MARIE
```

Vous rappelez-vous ce programme tout simple? Essayons de le modifier légèrement.

```
10 PRINT "JEAN";
20 PRINT "ET";
30 PRINT "MARIE"
RUN
JEANETMARIE
```

Ajouter

Rien qu'en plaçant un ; (point-virgule) après "JEAN" et "ET", l'ordinateur présente les trois mots réunis sur une seule ligne. A vrai dire, toutes les instructions peuvent ainsi être placées sur une seule ligne de programme:

```
10 PRINT "JEAN";"ET";"MARIE"
RUN
JEANETMARIE
```

Détail important

Bien sûr, la lecture de trois mots accolés est presque impossible et il nous faut donc essayer autre chose.

```
10 PRINT "JEAN",
20 PRINT "ET",
30 PRINT "MARIE"
RUN
JEAN          ET
MARIE
```

Détail important

14 caractères

En utilisant une , (virgule) au lieu du ; (point-virgule), les premières lettres de chaque mot sont imprimées à 14 espaces de distance. La même chose se produit si vous entrez l'instruction sur une ligne.

```
10 PRINT "JEAN","ET","MARIE"
RUN
JEAN          ET
MARIE
```

Détail important

14-caractères

Essayons à présent cette instruction, en plaçant un seul espace entre chaque mot et le second guillemet.

```
10 PRINT "JEAN ";"ET ";"MARIE"
RUN
JEAN ET MARIE
```

N'oubliez jamais ceci: un **espacement** est un caractère important.

Revoyons, pour suivre, les différentes façons d'imprimer les caractères numériques.

```
10 PRINT "3+5=";   ← Afficher les caractères
20 PRINT 3+5       ← Calculer, afficher la réponse
RUN
3+5=  8
```

La ligne 10 ordonne d'afficher les caractères, tandis que la ligne 20 demande de calculer le résultat. Cependant, ces deux fonctions différentes sont réunies par le ; (point-virgule) à la fin de la ligne 10. Ceci a signifié à l'ordinateur qu'il devait les imprimer ensemble sur une ligne, tout comme on le ferait normalement sur une feuille de papier (remarquez que, dans la réponse, un espace vierge se trouve devant le 8; il a pour but de permettre d'afficher le signe moins (−) si la réponse est un **nombre négatif**, inférieur à zéro, comme dans l'exemple 3 − 5 = −2).

Utilisons maintenant l'instruction INPUT et l'instruction PRINT dans le même programme tout simple.

```
10 INPUT A
20 INPUT B
30 PRINT A,B,A+B
RUN
? 3          ← Quelle valeur? Tapez 3
? 5          ← Quelle valeur? Tapez 5
 3                          5
  8
```

Valeur de A — Valeur de A+B — Valeur de B

L'instruction INPUT dit à l'ordinateur de vous **demander** quelle valeur donner à la variable, et d'attendre que vous **introduisiez** une valeur par l'intermédiaire du clavier. Après que vous avez actionné RETURN, l'ordinateur passe à l'instruction suivante. Et la ligne 30 place les trois valeurs sur l'écran.

Voici une méthode différente et plus claire d'accomplir les mêmes choses.

```
10 INPUT "A=";A
20 INPUT "B=";B
30 C=A+B
40 PRINT "A+B=";C
RUN
A=? 3
B=? 5
A+B= 8      ← Résultat de la ligne 40
```

Les lignes 10 et 20 disent à l'ordinateur quelle question il doit afficher, au lieu d'utiliser seulement le ? (point d'interrogation). La ligne 30 fait une nouvelle variable C et lui donne la valeur de A + B. La ligne 40 dit à la machine d'imprimer la réponse et le problème, commençant par A + B =. Les trois lignes après RUN sont bien plus faciles à comprendre que les trois chiffres esseulés à la fin de l'exemple précédent.

La lecture de l'affichage est parfois facilitée lorsqu'il comporte des **lignes vierges** entre les mots ou entre les groupes de mots.

```
10 PRINT "JEAN"
20 PRINT
30 PRINT "ET"
40 PRINT
50 PRINT "MARIE"
RUN
JEAN
                    Une ligne vierge entre chaque
ET
                    Une ligne vierge entre chaque
MARIE
```

Pour insérer une ligne vierge, il suffit d'entrer l'instruction PRINT seule sur la ligne, comme pour les lignes 20 et 40 dans l'exemple.

INPUT

```
10 INPUT "A est ";A
20 INPUT "B est ";B
30 PRINT "A+B=";A+B
40 PRINT "A-B=";A-B
RUN
A est ? 15
B est ? 3
A+B= 18
A-B= 12
```

Voici un autre programme qui combine les instructions INPUT et PRINT. Comme on le comprend aisément, il dit à l'ordinateur de **demander** des valeurs pour A et B et d'**afficher** ensuite deux calculs. Réunissons maintenant les deux instructions INPUT et donnons quelques explications complémentaires.

```
10 INPUT "A et B sont ";A,B
20 PRINT "A=";A,"B=";B
30 PRINT
40 PRINT "A*B=";A*B
50 PRINT "A/B=";A/B
RUN
A et B sont ? 15,3
A= 15          B= 3

A*B= 45
A/B= 5
```

A la ligne 10, vous pouvez entrer deux valeurs pour deux variables différentes, mais la , (virgule) entre les deux valeurs (en l'occurence, 15 et 3) est **très** importante. Si l'ordinateur affichait 153 au lieu de 15,3, il serait bien difficile de s'y retrouver. (L'instruction PRINT de la ligne 20 a aussi une ponctuation importante, A propos, savez-vous ce que la , et le ; signifient?)

A présent, nous allons ajouter une variable en chaîne à notre instruction INPUT.

```
10 INPUT "Nom";N$
20 PRINT N$;" est grand."
RUN
Nom? Jean
Jean est grand.
```

Lorsque le nom de la variable se termine par le signe $, cette variable "prend la valeur de" une lettre ou un mot. Voici donc un programme qui fait appel à une variable alphabétique et à une variable numérique.

```
10 INPUT "Nom";N$
20 INPUT "Age";Y
30 PRINT N$;" a ";Y;" ans."
RUN
Nom? Jean
Age? 10
Jean a 10 ans.
```

FOR—NEXT

```
10 CLS
20 FOR I=0 TO 36
30 LOCATE I,10
40 PRINT "$"
50 NEXT I
```

⬇

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Dans ce programme, la fonction de répétition (boucle) de l'instruction FOR-NEXT fait afficher par l'ordinateur des signes $ sur l'écran de la position 0,10 à la position 36,10.

```
10 CLS
20 FOR I=0 TO 36 STEP 3
30 LOCATE I,10
40 PRINT "$"
50 NEXT I
```

⬇

```
$  $  $  $  $  $  $  $  $  $  $  $  $
```

Que signifie **STEP 3** sur la ligne 20? Comme on peut le constater, le curseur s'est déplacé de **trois pas** après chaque signe $. Autrement dit, la valeur de I change désormais en incréments de 3 et le signe $ apparaît ainsi aux positions 0,10 et 3,10 et 6,10. $ couvrent presque la même surface de l'écran que dans le programme précédent, de 0,10 à **36**,10.

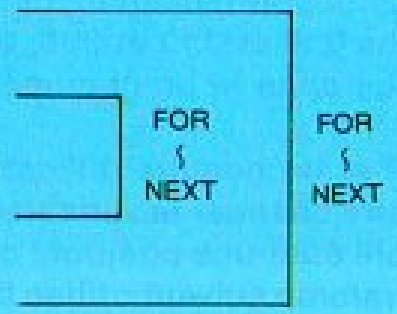
En ajoutant **STEP et un nombre** à l'instruction FOR, par conséquent, il est possible de **faire changer la variable par pas.** Comme dans le dernier exemple, la variable était une position, celle-ci a changé selon un pas de variation. Le programme suivant utilise la directive STEP pour changer la façon de calculer de l'ordinateur.

```
10 FOR I=50 TO 0 STEP -5
20 PRINT I,
30 NEXT I
RUN
 50              45
 40              35
 30              25
 20              15
 10              5
 0
```

I=50 TO 0 a pour effet que l'ordinateur compte comme il le fait à chaque boucle FOR-NEXT. Pour I, il existe 51 valeurs entre 50 et 0. Mais l'instruction **STEP** – **5** fait diminuer les valeurs par pas de – 5 et, par conséquent, il n'existe plus que 11 valeurs pour I.

FOR—FOR—NEXT—NEXT: une boucle à l'intérieur d'une boucle

```
10 FOR I=0 TO 2
20 PRINT "I=";I
30 FOR J=0 TO 4
40 PRINT "J=";J;
50 NEXT J
60 PRINT
70 NEXT I
RUN
I= 0
J= 0  J= 1  J= 2  J= 3  J= 4
I= 1
J= 0  J= 1  J= 2  J= 3  J= 4
I= 2
J= 0  J= 1  J= 2  J= 3  J= 4
```



Ce programme montre comment il est possible de placer une boucle FOR-NEXT à l'intérieur d'une autre boucle FOR-NEXT plus large. La variable I est contrôlée par les première et dernière lignes du programme, tandis que la variable J est contrôlée par les lignes 30 et 50. Il en résulte que, pour chaque valeur I, l'ordinateur effectue quatre petites boucles pour trouver toutes les valeurs J. I change de 0 à 2, et à chaque I, J change de 0 à 4.

Mais voici une autre façon d'utiliser une boucle.

```
10 INPUT N,S
20 CLS
30 FOR I=0 TO N STEP S
40 LOCATE 5,10:PRINT "I=";
50 PRINT USING "####";I
60 FOR J=0 TO 500
70 NEXT J
80 NEXT I
```

Ce programme utilise les instructions INPUT, FOR-NEXT et STEP pour faire changer la variable I en boucles de 0 jusqu'à N, par pas de S. Bien entendu, nous communiquerons les valeurs à N et S par l'intermédiaire du clavier.

Mais à quoi servent la variable J et sa courte boucle FOR-NEXT des lignes 60 et 70? J n'est jamais imprimé. Cette variable n'est jamais utilisée, mais l'ordinateur trouvera les valeurs 501 pour J chaque fois, avant de continuer le calcul et d'imprimer I. C'est ce que l'on appelle une **boucle fictive**. Elle exige un certain temps, juste ce qu'il faut à l'ordinateur pour accomplir 501 boucles FOR-NEXT, mais elle ne fait rien d'autre.

La raison d'être de la boucle J est donc de placer un **intervalle de temporisation** dans la boucle I plus grande. Lorsque vous exécuterez ce programme, vous constaterez que l'ordinateur effectue une pause après chaque affichage. Désormais vous savez à quelle vitesse votre ordinateur Sony est capable d'accomplir 501 choses simples (plutôt vite, vous en conviendrez!), et comment vous servir d'une boucle fictive pour insérer certains temps morts dans votre programme. Etes-vous capable d'écrire une boucle fictive qui occupera l'ordinateur pendant une durée plus longue?

IF—THEN et IF—THEN—ELSE

Voici un autre jeu de devinettes, où vous devez essayer de faire coïncider le nombre, choisi par l'ordinateur. Toutefois, ce jeu est légèrement différent du jeu de devinette et de celui de la machine à jetons que nous avons rencontrés dans ce livre. Dans les cas précédents, il s'agissait de jeux de hasard pur, mais cette fois, vous pouvez utiliser votre adresse dans une certaine mesure pour deviner plus rapidement le nombre correct. Si vous devinez intelligemment, vous devriez pouvoir trouver le nombre en six ou sept essais. Par contre, si vous devinez sans réfléchir, il vous faudra bien plus d'essais, de coups dans l'eau.

```
10 X=INT(RND(1)*100)+1
20 INPUT "Devinez";A
30 IF A=X THEN GOTO 70
40 IF A>X THEN PRINT "PLUS PETIT"
50 IF A<X THEN PRINT "PLUS GRAND"
60 GOTO 20
70 PRINT "BRAVO!"
```

Les lignes 30, 40 et 50 disent à l'ordinateur de comparer votre supposition à la valeur de X, et de vous donner ensuite un indice pour vous aider à faire votre devinette suivante.

Décelez-vous les changements apportés à ce programme suivant?

```
10 X=INT(RND(1)*100)+1
20 INPUT "Devinez";A
30 IF A=X THEN 70
40 IF A>X THEN PRINT "PLUS PETIT"
50 IF A<X THEN PRINT "PLUS GRAND"
60 GOTO 20
70 PRINT "BRAVO!"
```

En réalité, ce programme est le même que le précédent, même si la ligne 30 a été modifiée. IF-THEN a la même signification que IF-THEN-GOTO. En outre, le programme serait encore le même avec: 30 IF A=X GOTO 70. Autrement dit, IF-GOTO a la même signification que IF-THEN-GOTO et, par conséquent, tant THEN que GOTO peut être éliminé (mais pas les deux à la fois!).

A présent, nous allons réunir en une seule les deux lignes de ce programme:

```
10 X=INT(RND(1)*100)+1
20 INPUT "Devinez";A
30 IF A=X THEN 70 ELSE
   IF A>X THEN PRINT "PLUS PETIT"
50 IF A<X THEN PRINT "PLUS GRAND"
60 GOTO 20
70 PRINT "BRAVO!"
```

Les deux instructions IF-THEN des lignes 30 et 40 sont devenues une seule instruction IF-THEN-ELSE. A présent, il n'y a plus de ligne 40, mais la signification est la même. Si ceci est possible, il doit, tout autant, être possible de réunir les lignes 30 et 50 de ce programme en une seule ligne.

```
10 X=INT(RND(1)*100)+1
20 INPUT "Devinez";A
30 IF A=X THEN 70 ELSE
   IF A>X THEN PRINT "PLUS PETIT"
   ELSE PRINT "PLUS GRAND"
60 GOTO 20
70 PRINT "BRAVO!"
```

Pour vous assurer que chacun de ces quatre programmes est bel et bien le même, vous pouvez exécuter chacun d'eux sur votre ordinateur.

DIM (Variables en tableau)

Lorsque nous nous sommes servis des variables en tableau ou indicées pour notre programme de la machine à jetons, nous avons utilisé l'instruction

DIM A(2)

pour obtenir les trois variables:

A(0) A(1) A(2)

Ensuite, nous avons changé les nombres entre paranthèses en une variable, (I), et dit à l'ordinateur que I=0 TO 2. Nous avions encore les mêmes trois variables en tableau, et l'ordinateur les a changées en une boucle FOR-NEXT.

Les variables en tableau peuvent comporter plus d'un seul caractère entre les parenthèses. Ainsi par exemple, si nous entrons

DIM A(3,4)

nous obtiendrons 20 variables:

A (0, 0)	A (1, 0)	A (2, 0)	A (3, 0)
A (0, 1)	A (1, 1)	A (2, 1)	A (3, 1)
A (0, 2)	A (1, 2)	A (2, 2)	A (3, 2)
A (0, 3)	A (1, 3)	A (2, 3)	A (3, 3)
A (0, 4)	A (1, 4)	A (2, 4)	A (3, 4)

Comprenez-vous bien ces 20 variables? Le premier nombre entre parenthèses peut revêtir quatre valeurs, et le second 5 valeurs, ce qui revient à 4 fois 5 égalent 20. (A propos, combien de variables aurions-nous si nous écrivions DIM A(9,9) ?)

Nous vous avons présenté les 20 variables, non pas sous forme d'une simple liste, mais sous forme d'un **tableau** (vous pouvez les concevoir comme un arrangement bi-dimensionnel, où la première valeur change de gauche à droite, et la seconde valeur change du haut en bas.) Comme vous pouvez le comprendre, ces variables en tableau, dites aussi indicées, sont très pratiques pour les programmes, destinés à effectuer des diagrammes ou des tables. Pour utiliser ces variables bi-dimensionnelles en tableau dans un programme, nous devrons remplacer les nombres entre parenthèses par des variables, telles que A(I,J).
Voici un exemple d'un vrai diagramme:

	Lecture	Ecriture	Math.	Total
1er trimestre	68	88	70	226
2e trimestre	73	53	91	217
3e trimestre	92	98	82	272
Total année	233	239	243	715
Moyenne	77	79	81	238

Ce diagramme illustre les points obtenus par un élève dans trois branches au cours des trois trimestres de l'année scolaire. Avec les totaux et les moyennes, il y a exactement 20 nombres différents dans ce tableau. Cela signifie que nous pouvons utiliser la même variable en tableau DIM (3,4) pour effectuer autant de variables pour ce tableau. Réfléchissons maintenant aux moyens d'utiliser les programmes pour affecter des valeurs réelles à nos variables en tableau A(I,J). En voici un, entre autres:

```
100 FOR J=0 TO 2
110 INPUT A(0,J)
120 NEXT J
```

Ces trois lignes servent pour les points obtenus en Lecture, qui deviennent les variables A(0,0), A(0,1) et A(0,2). Chaque fois que l'ordinateur arrive à la ligne 110 dans la boucle FOR-NEXT, il nous demande d'entrer les valeurs (68,73,92).

Ensuite, pour calculer le total des points obtenus en Lecture, nous entrons:

```
200 T=0
210 FOR J=0 TO 2
220 T=T+A(0,J)
230 NEXT J
240 A(0,3)=T
```

..... T est la variable pour le total en lecture

De la même manière, nous pouvons entrer les points en Ecriture au 1er trimestre dans la variable A(1,0), les points en Math au 1er trimestre dans la variable A(2,0). Pour entrer le Total du 1er trimestre pour ces trois branches dans la variable A(3,0), le programme suivant fera l'affaire.

```
300 T1=0
310 FOR I=0 TO 2
320 T1=T1+A(I,0)
330 NEXT I
340 A(3,0)=T1
```

..... T1 est la variable pour le total de 1er trimestre

Pour programmer les autres points et totaux, ainsi que les moyennes, nous pouvons utiliser des boucles FOR-NEXT du même genre avec des variables en tableau sous la forme de A(I,J).
Voici quel sera l'aspect de l'ensemble du programme:

```
10 ' ** PROGRAMME POINTS SCOLAIRES **
20 '
30 DIM A(3,4)
40 CLS
50 '
60 ' ** ENTRER LES POINTS OBTENUS **
70 FOR I=0 TO 2
80 FOR J=0 TO 2
90 ON I+1 GOTO 100,120,140
100 LOCATE 0,J:PRINT "Lecture, Trimes
tre ";J+1;
110 INPUT A(0,J):GOTO 160
120 LOCATE 0,J+3:PRINT "Ecriture, Tri
mestre ";J+1;
130 INPUT A(1,J):GOTO 160
140 LOCATE 0,J+6:PRINT "Math, Trimest
re ";J+1;
150 INPUT A(2,J)
160 NEXT J
170 NEXT I
180 '
190 ' ** CALCULER LES TOTAUX **
195 ' **   ET LES MOYENNES   **
200 FOR J=0 TO 2
210 T=0
220 FOR I=0 TO 2
230 T=T+A(I,J)
240 NEXT I
250 A(3,J)=T
260 NEXT J
270 FOR I=0 TO 3
280 T=0
290 FOR J=0 TO 2
300 T=T+A(I,J)
310 NEXT J
320 A(I,3)=T
330 A(I,4)=INT(A(I,3)/3)
340 NEXT I
```

```
350 '
360 ' ** FAIRE UN TABLEAU **
370 CLS
380 LOCATE 5,0
390 PRINT "Lect Ecrit Math  TOTAL"
400 FOR S=1 TO 3
410 LOCATE 2,S+1:PRINT S
420 NEXT S
430 LOCATE 1,5:PRINT "TTL"
440 LOCATE 0,6:PRINT "MOYE"
450 FOR I=0 TO 3
460 FOR J=0 TO 4
470 LOCATE I*6+4,J+2
480 PRINT A(I,J)
490 NEXT J
500 NEXT I
```

Pour la facilité de notre planification, nous divisons le programme en trois sections, à savoir: Entrée Points, Calcul Totaux et Moyennes, et Réalisation Tableau. En outre, si vous écrivez sur papier les places où chaque variable est entrée, il vous sera plus facile de vous rappeler comment fonctionne le programme.

Bien entendu, les variables en tableau peuvent servir à bien d'autres choses que des "machines à jetons" ou des graphes. Et comme vous l'avez sans doute deviné, le nombre des variables entre parenthèses n'est pas limité à deux: A(P,Q,R) ou B(X,Y,Z,XY,XZ,YZ).... ou toute autre combinaison, jusqu'à concurrence de 255 variables différentes! Les informaticiens font fréquemment appel à des variables en tableau ou indicées pour concevoir des jeux vidéo, pour gérer les finances d'une grande entreprise ou pour de nombreuses autres applications complexes. Comme vous vous en rendrez compte en poursuivant vos investigations avec des amis ou dans d'autres manuels, votre ordinateur Sony se prête à une utilisation de ces variables pour bien d'autres choses encore.

Vous avez pris un excellent départ, car vous avez déjà maîtrisé l'emploi des variables en tableau dans programmes différents.

Alors, continuez à explorer les richesses de votre ordinateur Sony en vous livrant à toutes sortes d'exercices.

Bonne chance!

INDEX ALPHABETHIQUE

A
Adresse 73
Apostrophe..... 112

B
Boucle..... 44, 77, 122
Boucle conditionnelle..... 77
Boucle fictive 123

C
Caractères 29, 43
CLS 73
Classement..... 31
Clavier 7
COLOR 14-15, 54
CSAVE..... 61
CTRL (touche CTRL) 9

D
Deux-points (:) 47
DIM 89, 125
Dollar (signe $)..... 97

E
ELSE (IF-THEN-ELSE) 70, 123
Entrées 9
Espacement..... 12, 75, 116

F
Formule conditionnelle 67, 70
FOR-NEXT 78-79, 100, 120

G
Graphiques 42
GOTO 44
Guillemets (") 20, 97

I
IF-THEN 67, 123
IF-THEN-GOTO 124
INKEY$ 100
INPUT 66, 118
INT (nombre entier) 49
Instructions..... 5, 9, 51

L
LINE..... 55
LIST..... 40, 106
LOCATE 73

M
Messages d'erreur
..... 11, 16, 39, 47
MOD (Module) 81

N
NEW..... 40
Nom de fichier..... 61
Nombres aléatoires..... 48, 93
Nombre négatif 117
Numéro de ligne..... 40

O
ON-GOTO 93

P
Parenthèses ( ) 47
Petite bête 37
Point décimal 48
Ponctuation 47
PRINT 16, 19-20, 115
PRINT USING 101
Programmation 31, 39
PSET 44

R
REM 112
Remarques 112
RENUM 110
Renuméroter 109
RETURN (Touche RETURN) 10
RND (1) 48-49, 93
RUN 39

S
SCREEN 43
SHIFT (Touche SHIFT) 16
Signe dollar ($) 97
Sous-titres 111
STEP 121
STOP (Touche STOP) 9

T
Tableau des couleurs 14
Titres 111
TO 78
Touche CTRL 9
Touche RETURN 10
Touche SHIFT 16
Touche STOP 9
Trait d'union (-) 55

U
USING (PRINT USING) 101

V
Valeur initiale 78, 93
Variables 21, 27, 76
Variable alphanumérique 97
Variables en tableau
(ou indicées) 88-89, 125
Virgule (,) 47