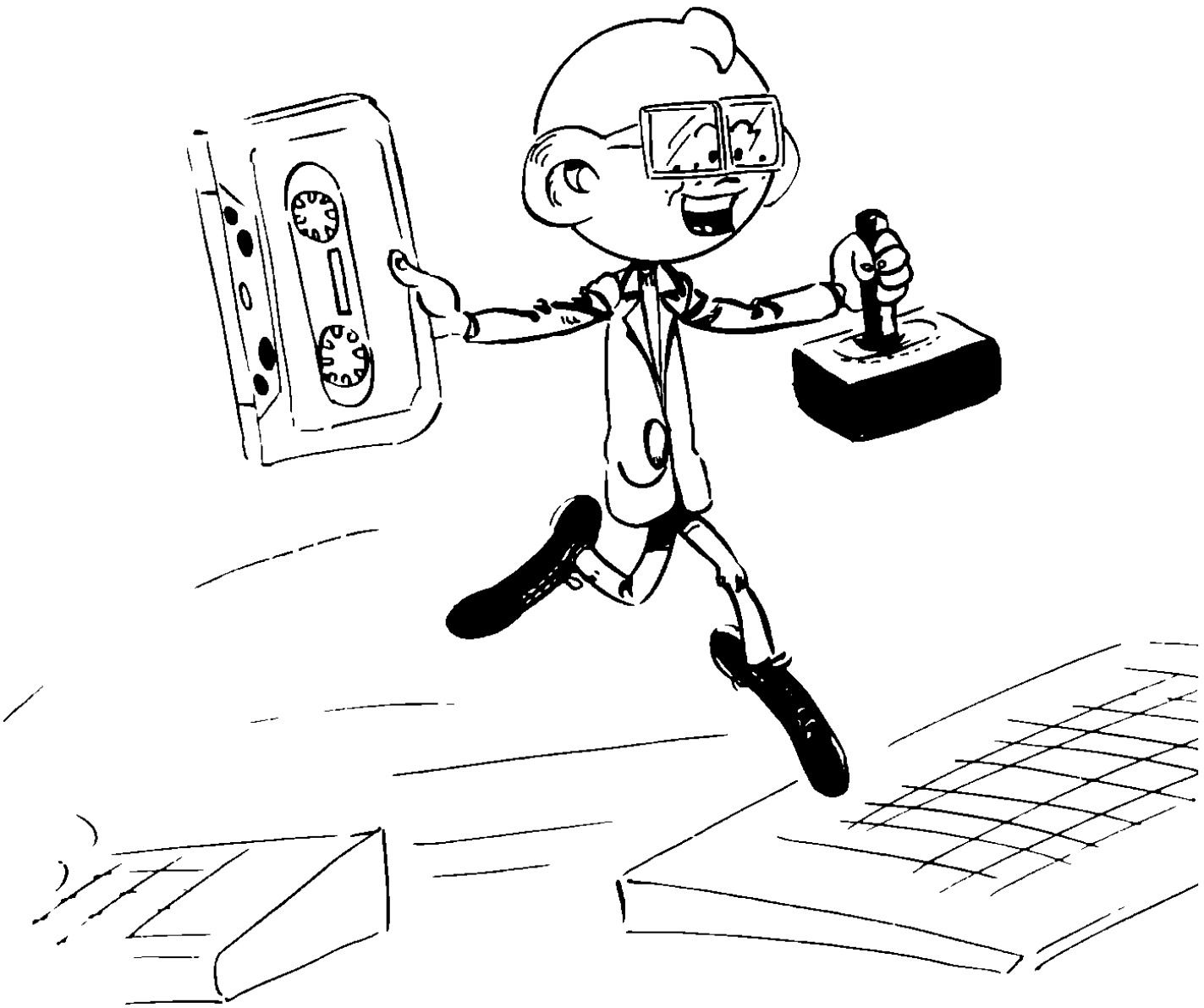
# Le cube informatique

# Initiation au basic



Conception graphique : Logoform - Grenoble
Illustrations : Simon LAVAULT

# SOMMAIRE



4

# LE CUBE INFORMATIQUE D'ERNEST DUCLAVIER

Bonjour, ce drôle de petit bonhomme qui se promène sur les micro-ordinateurs avec des lunettes grandes comme des écrans, c'est ERNEST, ERNEST DUCLAVIER. Nous l'avons rencontré par hasard, un soir, une manette dans une main et une cassette dans l'autre. Au départ, il faisait un peu fort en thème mais s'est révélé très sympathique dès que nous avons abordé le sujet de l'informatique:

"Contrairement à ce que l'on pense, un micro-ordinateur n'est pas une boîte magique qui sait, comprend n'importe quoi et résoud tous les problèmes. Cela reste une machine inerte qui s'anime lorsqu'on tourne un interrupteur mais incapable du moindre effort si l'homme n'intervient pas... heureusement".

Après ces généralités il nous proposé de nous initier au langage BASIC par des explications, des exercices, de la simulation et des jeux et il a sorti un cube de sa poche. Vous l'avez maintenant entre les mains avec en outre un livret de fiches de travail tiré de votre expérience...

Ce qui reste le plus étrange c'est que nous n'avons jamais revu ERNEST depuis ce soir là. On raconte qu'il habite dans les ordinateurs... Ridicule. Il nous arrive pourtant parfois de frotter le CUBE d'ERNEST pour le revoir. Et si l'histoire d'Aladin marchait au XX$^{e}$ siècle ! ! !

## organisation du livret

Ce livret est composé de FICHES toutes sur le même standard qui sont composées de la manière suivante

- TITRE ..... N° □
  - Objet
  - Mots clefs
  - Quelques détails
  - Exercices et jeux

• **TITRE** : les chapitres principaux de l'initiation (voir Sommaire) pour les différents thèmes.

• **N°** | | : est volontairement laissé vierge pour que vous y inscriviez les numéros du compteur de cassette qui correspond aux problèmes.

• **OBJET** : détaille la nature du problème présenté.

• **MOTS CLEFS** : permet le repérage rapide en cas de recherche d'une notion particulière. Ces mots clefs sont rappelés en annexe.

• **QUELQUES DETAILS :** ce sont les notes complémentaires aux explications des programmes : précisions, astuces...

• **EXERCICES ET JEUX :** donnent des informations sur les règles et la nature des exercices proposés pour mieux assimiler les notions abordées.

Ces fiches sont par thème, elles se suivent mais peuvent être exploitées séparement (pour revoir un thème il faut reprendre la cassette à partir du numéro que vous aurez pris soin de noter sur la fiche).

A la fin du livret vous trouverez en index les mots clefs, un sommaire et des conseils qui font le bilan de l'initiation (programmation et correction programme).

Les programmes et le livret se complètent, certaines explications sont sur l'écran, d'autres sur les fiches et il en est de même pour les exercices.

La meilleure méthode pour bien assimiler cette initiation est à notre avis de garder le livret à portée de la main tout en suivant les déroulements des programmes à l'écran.

A la fin de chaque démonstration, à l'écran s'affiche la question "Voulez-vous revoir ces notions ? O/N". Si vous tapez O l'ordinateur reprend ses explications, si c'est N le message "laissez tourner le magnétophone" apparaît et le programme suivant se charge. Nous vous conseillons de noter le numéro de compteur de votre magnétophone à ce moment là. Ainsi vous n'aurez qu'à positionner la bande au bon endroit pour toute étude ultérieure.

# COMMENT UTILISER VOTRE CUBE

Ce cube informatique est constitué de quatre cassettes. Chaque cassette contient plusieurs programmes chaînés entre eux. Chaque cassette est repérée par son numéro d'ordre logique.

## MODE D'EMPLOI

### SI VOUS POSSEDEZ UN MICRO-ORDINATEUR MO-5

Ce logiciel fonctionne avec la configuration suivante :
- Unité centrale MO-5
- Lecteur enregistreur de programmes

Assurez-vous que tous ces éléments sont correctement connectés et mis sous tension.

1. Connectez le lecteur enregistreur de programmes sur le côté de l'unité centrale.

2. Mettez sous tension votre unité centrale.

3. Introduisez la cassette dans le lecteur en choisissant la face correspondante pour le MO-5, vérifiez que la cassette est bien rembobinée complètement, appuyez sur la touche lecture ▷, Il est normal que le lecteur ne démarre pas.

4. Tapez "RUN", puis appuyez sur la touche ENTREE. Le lecteur démarre, le programme est alors chargé et démarrera en fin de chargement automatiquement.

**Tous les programmes étant chaînés entre eux, vous devez laisser enfoncée la touche lecture de votre magnétophone pendant toute la durée de la séance de travail, un programme en appelant un autre.**

**En fin de cassette le programme vous demandera de changer de cassette.**

## SI VOUS POSSEDEZ UN MICRO-ORDINATEUR TO-7 OU TO-7/70

Ce logiciel fonctionne avec la configuration suivante :

- Unité centrale TO-7 ou TO-7/70
- MEMO 7 BASIC
- Lecteur enregistreur de programmes.

Assurez-vous que tous ces éléments sont correctement connectés et mis sous tension.

1. Votre unité centrale étant éteinte, introduisez la MEMO 7 Basic dans la trappe prévue à cet effet.

2. Connectez alors le lecteur enregistreur de programmes sur le côté de l'unité centrale.

3. A la mise sous tension, sur l'écran s'affiche un menu.

4. Introduisez la cassette dans le lecteur en choisissant la face correspondante aux TO-7, vérifiez que la cassette est bien rembobinée à fond, appuyez sur la touche lecture ▷, il est normal que le lecteur ne démarre pas.

5. Sélectionnez l'option 1 pour entrer en mode Basic. Tapez RUN" puis appuyez sur ENTREE.

6. Le lecteur démarre automatiquement, le programme se charge alors et démarre.

# COMMENT N° □
# ÇA MARCHE

## LES DIFFERENTES PARTIES DE VOTRE MICRO-ORDINATEUR
## SIMULATIONS :

- DU CHARGEMENT D'UN PROGRAMME
- DE L'EXECUTION D'UN PROGRAMME

## MOTS-CLEFS

UNITÉ CENTRALE, CLAVIER, ÉCRAN, MAGNÉTOPHONE.

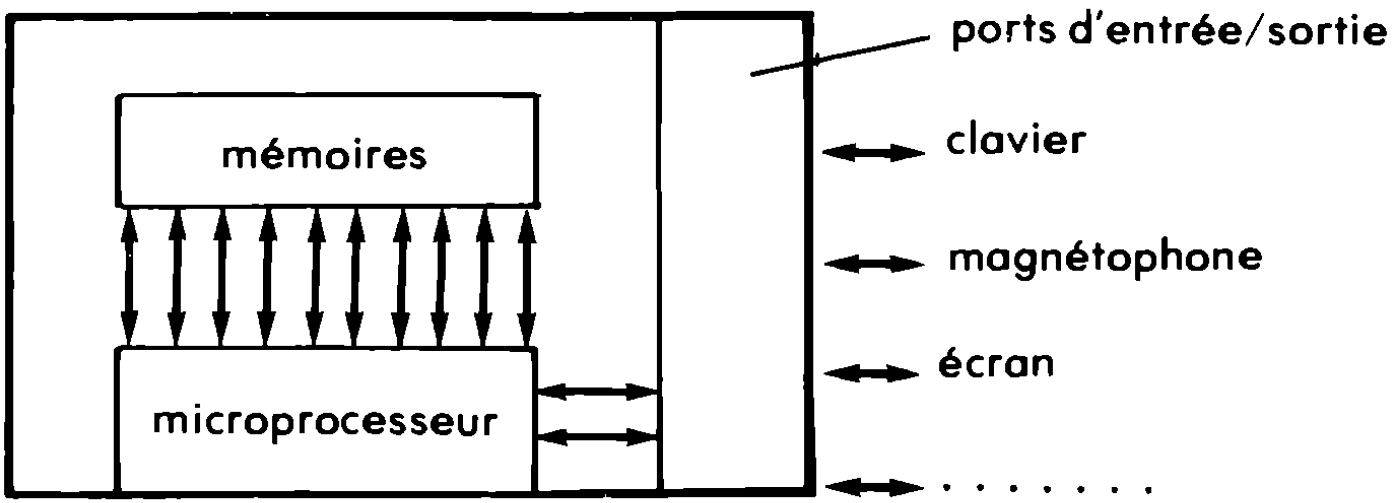
## QUELQUES DÉTAILS

Ce premier programme présente les différentes parties de votre micro-ordinateur, voyons plus précisément leur nature :

**UNITE CENTRALE :** c'est la partie vitale de votre ordinateur. Elle contient les composants électroniques qui constituent le cerveau de la machine :

- le MICROPROCESSEUR qui gère et trie les informations
- Les MEMOIRES qui les stockent
- Les PORTS D'ENTREE SORTIE qui permettent de communiquer avec le MICROPROCESSEUR, par le clavier, le magnétophone ou l'écran.

Résumons par un court schéma.



**ECRAN :** c'est votre téléviseur, ou un moniteur. Vous lui transmettez des signaux via la prise PERITEL ou l'entrée UHF (prise d'antenne) avec un modulateur. Ces signaux contrôlent les faisceaux lumineux qui créent l'image sur l'écran.

**MAGNETOPHONE : (ou LECTEUR DE DISQUETTES).** Il sert au stockage des programmes de façon permanente sur un support magnétique ( par exemple le CUBE INFORMATIQUE). Ainsi une fois un programme créé, il est préférable de le SAUVEGARDER sur une cassette, cela évite d'avoir à le retaper une fois votre ordinateur éteint. Une autre méthode de stockage, plus industrielle est le MODULE électronique qui contient des composants dans lesquels sont gravés des informations.

Au fait, nous parlons d'informations.
A quoi ressemblent-elles ?

Un ordinateur ne comprend que des signaux élémentaires, c'est-à-dire des oui ou non, ce sont des BITS (binary digit). Il ne peut distinguer que ces deux options. Ainsi tous les ordres que vous lui transmettrez suivent un code dit binaire (Ø et 1, oui et non). Les nombres par exemple sont codés de la manière suivante :

| nombre | code |
|---|---|
| Ø | Ø |
| 1 | 1 |
| 2 | 1Ø |
| 3 | 11 |
| 4 | 1ØØ |
| 5 | 1Ø1 |

Une suite de Ø et 1 traduit donc un nombre, l'ordinateur lit un train de huit Ø ou 1 : c'est un OCTET.

En fait, en mémoire et sur les bandes magnétiques, les informations sont des suites de Ø et 1.

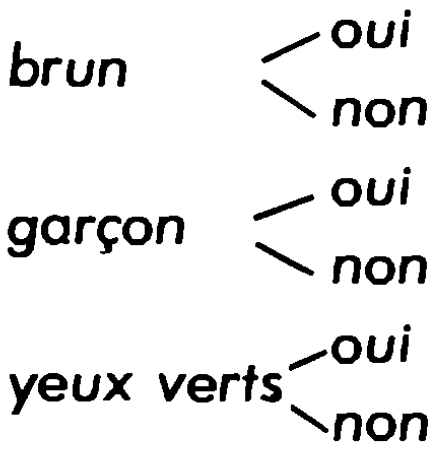
# EXERCICES ET JEUX

*Essayons d'imaginer la traduction d'informations suivant un code binaire en sélectionnant le nombre de BIT nécessaire :*

*Exemple choisir entre 3 options, fromage, dessert, café. On ne peut faire ce choix simplement par Ø ou 1, il faut 2 niveaux de choix (2 bits).*

*fromage : Ø Ø*
*dessert : Ø 1*
*café : 1 Ø*

*Autre exemple : trions les élèves d'une classe pour sélectionner les garçons bruns aux yeux verts. Nous pouvons utiliser 3 bits :*



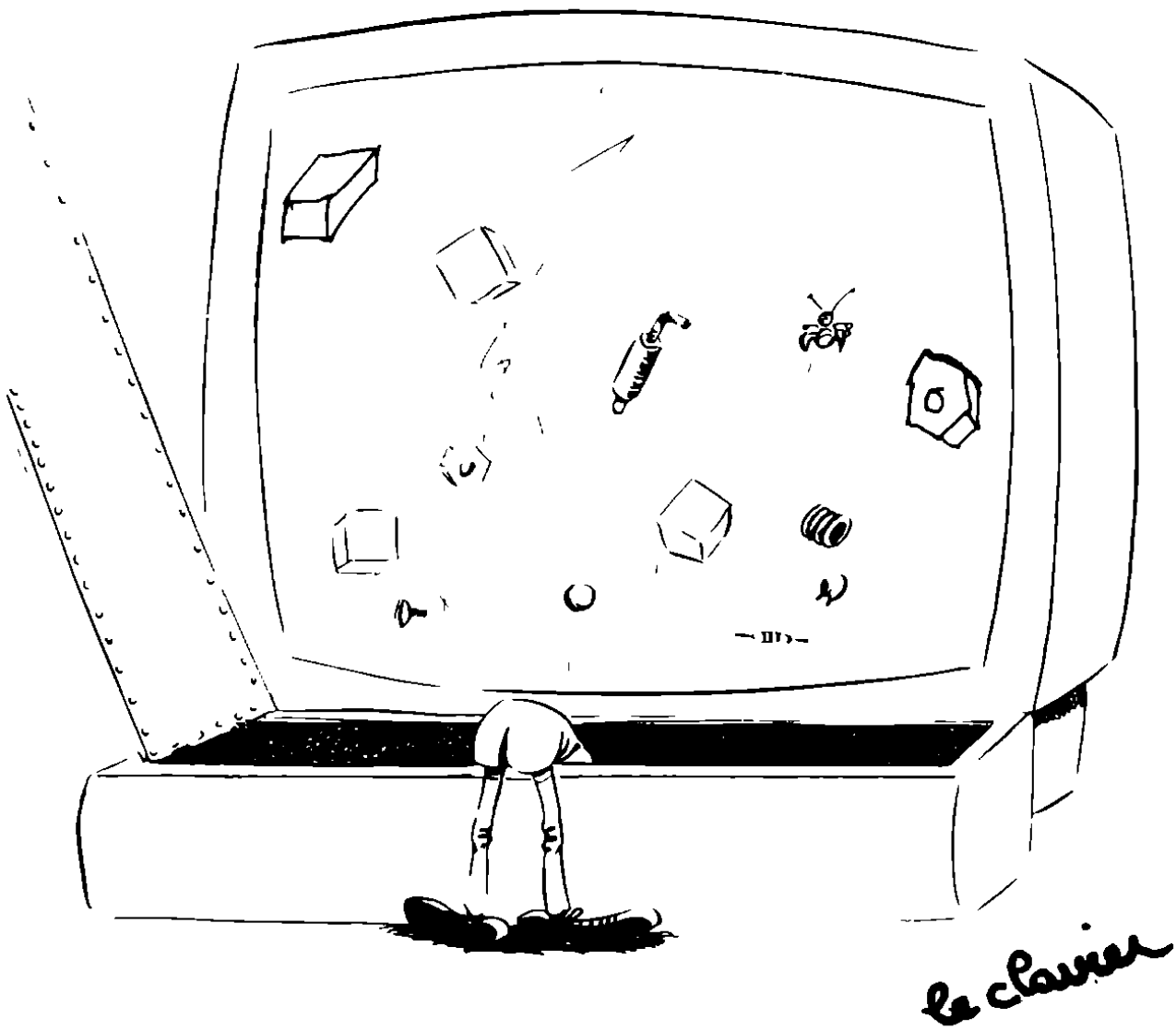
*Ainsi tous les élèves pourront avoir un code du type :*
*0 1 0 = garçon qui n'a pas les yeux verts et n'est pas brun.*
*1 0 0 = fille brune qui n'a pas les yeux verts...*

*Nous pourrons donc trier les élèves avec 3 bits.*

*A vous d'imaginer les codages selon le même principe pour les problèmes suivants :*

*- on a accidentellement mélangé un sac de haricots rouges et blancs, il faut les trier.*
- Il faut distinguer les règnes végétal, animal, minéral.
*- il faut vérifier que toutes les lumières d'une villa de cinq pièces sont éteintes.*

Tous les éléments de votre micro ordinateur sont donc réliés et les transmissions d'informations se font par l'intermédiaire de câbles dans lesquels circulent ces "trains" électriques ! Suivez la petite animation pour voir l'ordre de ces transmissions.


le Canard

# LE CLAVIER N° □

**2 programmes**

**PRESENTATION DU CLAVIER**
**RECONNAISSANCE DES TOUCHES**
**FONCTIONNEMENT DU CLAVIER**
**COUPLAGES DES TOUCHES**
**APPRENTISSAGE DE LA FRAPPE AU CLAVIER**

**MOTS-CLEFS**

CLAVIER, TOUCHES, MINUSCULE, EDITEUR, CONTRÔLE, COMMUNICATION

# QUELQUES DÉTAILS

## PROGRAMME 1

A quoi sert un ordinateur familial : à permettre un dialogue homme-machine. En attendant la transmission d'ordres vocaux ou la connection directe cerveau humain/microprocesseur... Il nous reste principalement le clavier pour dicter nos remarques aux circuits électroniques.

Découvrons donc la géographie du clavier à travers un jeu de reconnaissance de position de touches.

Dans un premier temps, les caractères n'apparaissent pas à l'écran, nous les étudierons juste après. Il vous faut réagir vite car si la bonne réponse n'est pas affichée au bout de quelques secondes, l'ordinateur donne la réponse.

## PROGRAMME 2

Voici maintenant le clavier présenté complètement avec toutes ses touches. On

distingue différents types de touches :

- **les TOUCHES DE CONTROLE** qui envoient directement un ordre au micro-ordinateur :

**STOP** arrête le déroulement d'un programme
**RAZ** nettoie l'écran si un programme ne tourne pas
**CNT** combinée à une lettre a une action spécifique : **CNT + C** arrête l'exécution d'un programme, **CNT + X** efface la ligne sur laquelle se trouve le curseur... (Reportez-vous au manuel d'utilisation).

- **les TOUCHES DE CARACTERES** qui servent à l'écriture de lettres, chiffres ou caractères spéciaux à l'écran. La pression d'une touche A par exemple entraîne l'affichage d'un A. La combinaison **SHIFT** Jaune avec une touche aura pour conséquence l'affichage du signe jaune de la touche s'il existe.

Notez que : la combinaison **SHIFT Jaune/barre d'espacement** donne accès au mode minuscule ou au retour au mode majuscule.

- les **TOUCHES D'EDITION** permettent de déplacer le curseur et de modifier des affichages en insérant (**INS**) ou effaçant (**EFF**) des lettres.

Une touche enfin est remarquable : **ENTREE**, en effet, sa pression entraîne la transmission de la phrase affichée. En effet lorsque vous écrivez quelque chose à l'écran, l'ordinateur dans un premier temps l'ignore si vous ne le validez pas par **ENTREE.** De même si vous modifiez l'affichage d'une phrase sans valider, cette modification ne sera pas prise en compte.

Sur le MO5 une autre possibilité existe, celle d'afficher directement par pression simultanée de la touche **BASIC** et d'une lettre, un ordre BASIC (**BASIC + Z = LOCATE**). C'est une facilité qui évite de taper tous les ordres. Ceci n'est pas possible sur le TO7 ni le TO7 70, ces machines n'ayant pas de langage résident.

# *EXERCICES ET JEUX*

*L'ordinateur vous propose un entraînement à la reconnaissance du clavier en recopiant quelques phrases. Attention, toutes les erreurs sont détectées, y compris celle de mode (minuscule/majuscule).*

*A vos touches !... Vous vous étonnerez !*

# DONNÉES ET VARIABLES

N° □

**3 programmes**

**NOTION DE DONNEES**
**CARACTERES CODES SUIVANT LA NORME ASCII**
**VARIABLES NUMERIQUES ET ALPHANUMERIQUES**
**TRAITEMENT DES DONNEES EN MEMOIRE**

**MOTS-CLEFS**

DONNEES, VARIABLE, CARACTERE, ASCII, SIGNE=

# QUELQUES DÉTAILS

## PROGRAMME 1

La définition de ce qu'est une donnée reste l'exercice le plus difficile pour un professeur. En effet, l'ordinateur ne sait pas gérer autre chose que des trains de huit informations élémentaires du type 10101001. Mathématiquement on démontre facilement qu'il y a 256 possibilités de combiner huit informations élémentaires 0 ou 1. L'ordinateur sait distinguer 256 informations différentes de base. Chaque constructeur a donc la possibilité de décider que 00000001 représentera pour son ordinateur le caractère "A", un autre constructeur choisissant pour le caractère "A" le codage 00001000. Afin d'éviter ce genre de problème, il existe une norme internationale appelée norme **ASCII** qui fixe la correspondance entre les différentes combinaisons de 0 et de 1 et les caractères alphabétiques. Le programme vous permet de voir quels sont les codages retenus par la norme ASCII, et donc comment l'ordinateur stocke les informations en mémoire en fonction de la norme.

Cette norme consiste en trois parties. Chaque octet de la mémoire de votre ordinateur est formé de huit bits (informations élémentaires Ø ou 1), ce qui lui permet théoriquement de distinguer 256 informations différentes. L'ordinateur pour son fonctionnement a besoin de pouvoir distinguer 32 codes de commande que nous ne détaillerons pas ici ; ils représentent les 32 premiers codes ASCII de Ø à 31. Les codes ASCII de 32 à 128 comme vous pouvez le constater dans le programme représentent tous les caractères alphabétiques en majuscules et minuscules plus certains caractères spéciaux.

Tous ces détails sont destinés à vous aider à comprendre la notion de données telle que l'ordinateur la conçoit: il ne comprend que les octets. Lorsque nous introduisons une donnée dans la mémoire de l'ordinateur celui-ci la classe suivant son habitude. Par exemple une donnée alpha-numérique "ABA", sera d'abord ramenée aux codes ASCII des lettres la composant, c'est à dire, "65, 66, 65", chaque code ASCII sera ensuite placé en mémoire tel quel.

Pour que l'ordinateur puisse distinguer au niveau traitement une variable alpha-numérique, on adjoint à ces dernières un suffixe constitué par un signe dollar $. Par exemple :

MA$ représente une variable alpha-numérique
TOTO représente une variable numérique.

Une variable pour l'ordinateur est une zone mémoire qui à l'instant t possède un contenu qui peut être numérique ou alphanumérique. Un certain nombre d'opérations sont possibles sur ces zones mémoires, ce qui fait la puissance d'un ordinateur.

Ces opérations sont évidemment très différentes suivant la nature des variables, il est par exemple inconcevable de diviser une variable alpha-numérique par un nombre.

Ainsi si on a A$ = "CUBE", cela signifie que l'ordinateur range dans une de ses zones mémoires les valeurs des caractères ASCII composant le mot CUBE. Mais pour l'ordinateur, il sera plus facile de manipuler A$ que "CUBE". Une donnée est donc la matière première qu'un

programme va manipuler suivant des ordres précis. Nous détaillerons ceci dans le chapitre suivant.

## PROGRAMME 2

Après avoir appris à reconnaître et distinguer les types de variables, cherchons à comprendre ce qu'en fait votre ordinateur : il les range en mémoire et peut les rappeler, les ajouter, les transformer.

Une simulation vous plonge au cœur de cette mémoire. Le micro-processeur prend des valeurs en mémoire, il opère sur elles et les reloge en mémoire en suivant les instructions programmées.

Cette intermède technique est nécessaire pour comprendre ce que signifient les signes algébriques traditionnels :

= est une équivalence : dans la case nommée A, il y a la valeur 32 : A = 32. A$ = "CALE" signifie que dans les cases A$, on trouve les valeurs 67/65/76/69.

**+ est une réunion :** dans la case C il y a la valeur A et celle de B : C = A + B,

Dans la case C$ on met le contenu de la case A$ (BONJOUR) et de la case B$ (MONSIEUR) :
C$ = A$ + B$ ; dans C$ nous trouverons BONJOUR MONSIEUR.
Cette dernière opération s'apelle la **concaténation**, un chapitre ultérieur lui est consacré.

# *EXERCICES ET JEUX*

## *PROGRAMME 3*

*Le programme suivant est destiné à vous faire jouer à reconnaître les variables et les données. Il s'agit de faire fonctionner une usine en amenant les bonnes fournitures dans les bons ateliers. Pour cela, vous vous aiderez des touches fléchées pour faire descendre, monter, aller à gauche, aller à droite votre petit chariot. La touche ACC vous permet d'interrompre le jeu.*

# ANALYSE N° □

## NOTION DE PROGRAMME
## PRESENTATION DU BASIC
## ANALYSE D'UN PROBLEME

## MOTS-CLEFS

BASIC, PROGRAMME, ANALYSE, COMMUNICATION

## QUELQUES DÉTAILS

Nous voilà prêts pour le vif du sujet. Un ordinateur n'exécute que les actions pour lesquelles on le programme. Si vous désirez changer la couleur de l'écran, vous devez transmettre à l'ordinateur un certain nombre d'ordres qu'il suivra. Nous avons vu que les

informations que comprend la machine sont très élémentaires (trains de huit Ø ou 1). Il n'est question de taper ce type de successions de chiffres, le clavier ne servirait à rien, un contact suffirait (comme pour le morse). On utilise plus communément des langages de programmation dont le **BASIC (Beginners All Purpose Symbolic Instruction Code**, code d'instructions symboliques pour débutants) est un des plus accessibles. Il est composé de mots tirés de l'anglais qui permettent de définir des instructions (**PRINT = Affichage, LIST = Listage du programme, RUN = Exécution d'un programme...**). **Ainsi la "conversation" est plus rapide !**

Les suites ou instructions définissent un certain nombre de tâches élémentaires constituant les éléments du programme. Nous avons symbolisé cela ci-par une tuyauterie faite de petites structures que l'on peut ajuster comme on le désire. Pourquoi des tuyaux ? Parce que tout comme l'eau ou le sable y circulent, les données sont utilisées et modifiées au gré des programmes.

En effet l'ordinateur ne fera que reconnaître, manipuler, trier, renvoyer des données (chiffres ou lettres) pour les mener d'un état A (départ) à un état B (arrivée). Par exemple si un programme calcule une taxe de 15 %, le programme prend la valeur et restitue le résultat. On pourrait aussi multiplier la valeur initiale par 3, ajouter 3, diviser 3, retrancher 1, reprendre la valeur, multiplier par 0,85, soustraire à la valeur initiale... pour obtenir le même résultat. La décomposition d'un problème en tâches élémentaires appartient au programmeur ; c'est à lui d'ANALYSER une action pour la décomposer le plus rationnellement possible.

Pour nous résumer, tout problème peut être scindé en sous-tâches élémentaires pour que l'ordinateur les traite : **c'est l'ANALYSE d'un problème.**

# *EXERCICES ET JEUX*

*Nous vous proposons un petit excercice d'analyse de tâche :*

*Vous devez rendre la monnaie pour un achat de 80 F, le client vous donne 100 F. Vous disposez en caisse de 3 pièces de 5 F, 5 pièces de 2 F, 10 pièces 1 F. Le client vous demande le moins de pièces possibles .*

*A vous de trouver une voie d'analyse du problème et de proposer en plus une solution loufoque.*

# ORDINOGRAMMES N° □

## NOTION D'ORDINOGRAMMES

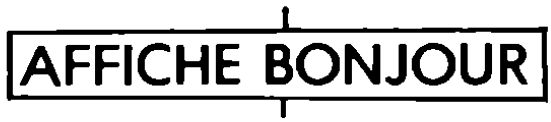
## MOTS-CLEFS

ORDINOGRAMMES, ANALYSE

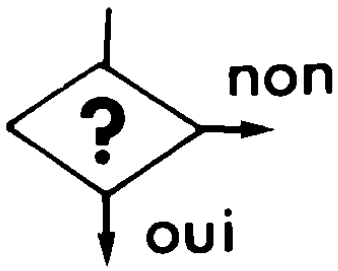
## QUELQUES DÉTAILS

L'ordinateur va nous contraindre à analyser ce que nous voulons voir exécuter. Pour cela il existe un code d'analyse qui permet de faire des schémas : les ORDINOGRAMMES.

Les actions sont repérées par des rectangles :

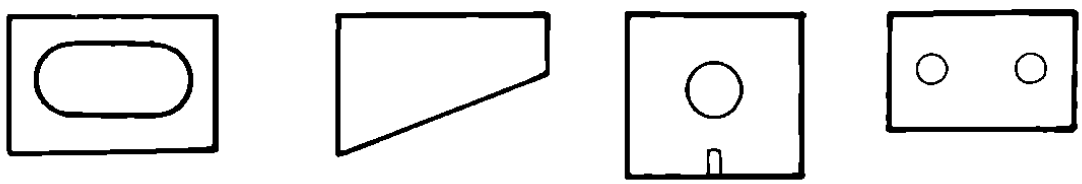



Les questions par des losanges (toujours oui ou non)




Les flèches symbolisent les fonctions entre tâches.

Les sorties sont représentées par des pictogrammes.



ECRAN IMPRIMANTE DISQUE CASSETTE

Cette rationalisation permet de systématiser l'analyse afin de bien associer une tâche élémentaire à une instruction.

A l'écran vous voyez l'association d'ordinogrammes et d'instructions.

Ces instructions se présentent de la manière suivante (exemple) :

120 PRINT "BONJOUR"

120 représente l'adresse de l'instruction.
PRINT "BONJOUR" en est le libellé.

Les instructions sont rangées en mémoire suivant leurs adresses et s'exécutent dans l'ordre de leurs adresses sauf si le libellé de l'instruction en décide autrement.

L'instruction 2 est donc exécutée après l'instruction 1, la 3 après la 2, etc...

Détaillons successivement tous les libellés possibles pour les instructions.

Ces libellés font toujours appel aux fonctions BASIC.

En fait, tout le secret du succès du BASIC réside dans les quelques paragraphes qui vont suivre ! Tout le reste n'est que présentation, astuce et élégance.

Il existe 6 types d'ordres fondamentaux en BASIC :

L'ordre d'affichage **PRINT** (affiche)
il permet de faire afficher sur l'écran une chaîne de caractères quelconque ou des nombres. Sa syntaxe est :

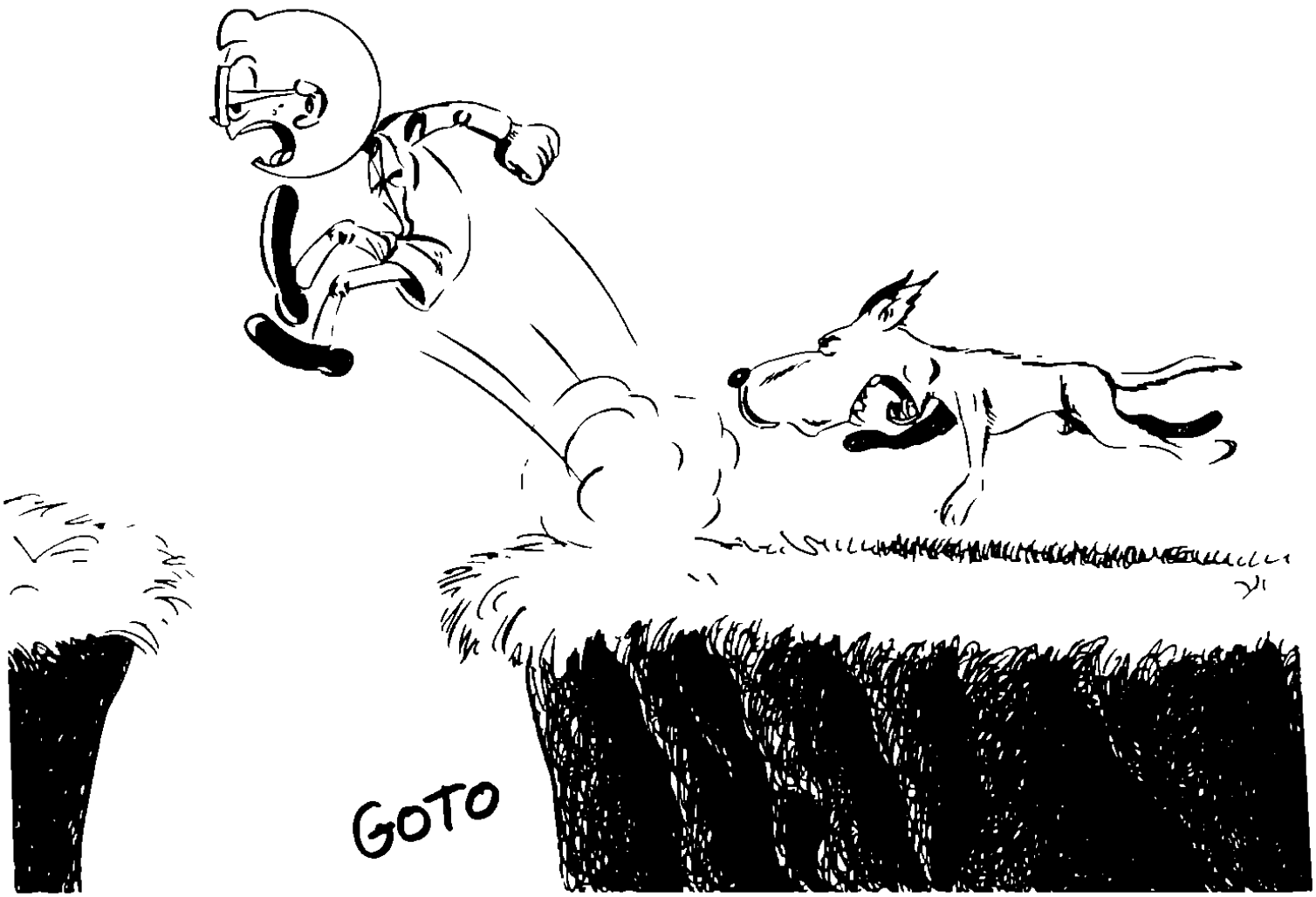
PRINT "BONJOUR" ou PRINT 123 ou PRINT A

N'oubliez pas les guillemets pour les mots ou les chaînes de caractères. L'exécution de cette instruction provoque l'affichage sur l'écran de la phrase située entre les guillemets ou du nombre ou de la la valeur de la variable.

Si vous êtes pressé, vous pouvez taper sur un point d'interrogation à la place de PRINT, le BASIC le reconnaissant comme une abréviation du mot PRINT.

L'ordre d'acquisition **INPUT** (entrée)
Celui-ci autorise la saisie d'une valeur ou d'un mot tapé par l'utilisateur sur le clavier. Pour calculer une taxe sur une valeur à entrer, nous utiliserons ce type d'ordre. C'est un peu "quelle valeur ou quel mot ?". Sa syntaxe est :

INPUT A pour une saisie de nombre


GOTO

INPUT A$ pour une saisie de chaîne de caractères (cassette n°2).

Lorsque votre micro-ordinateur rencontre une telle instruction, il se met en attente d'une intervention extérieure. Cela se traduit par un point d'interrogation suivi d'un curseur clignotant.

## L'ordre de saut **GOTO**

Nous savons qu'un programme se déroule suivant l'ordre croissant des adresses des instructions. L'utilisateur peut toutefois modifier cette exécution en demandant à l'ordinateur de se brancher à une autre adresse. Il faut utiliser pour cela un GOTO. Sa syntaxe est la suivante

GOTO xxx ou xxx représente l'adresse à laquelle on veut que l'ordinateur continue l'exécution du programme.

L'adresse spécifiée peut être antérieure ou postérieure à celle où se trouve l'instruction GOTO.

## L'ordre de test **IF... THEN...**

L'ordinateur ne sait reconnaître que des valeurs binaires. Il en est ainsi de ses choix, il ne connaît pas le "peut-être". Soit une condition est réalisée soit qu'elle ne l'est pas. Certes, il paraît un peu manichéen mais nul n'est parfait. Sa syntaxe est la suivante :

IF (condition) THEN xxxx
où condition est une affirmation qui réclame une réponse parmi deux possibles, soit la condition est vérifiée, soit elle ne l'est pas.
Par exemple IF A = 2 ou
IF A$ = "BONJOUR"

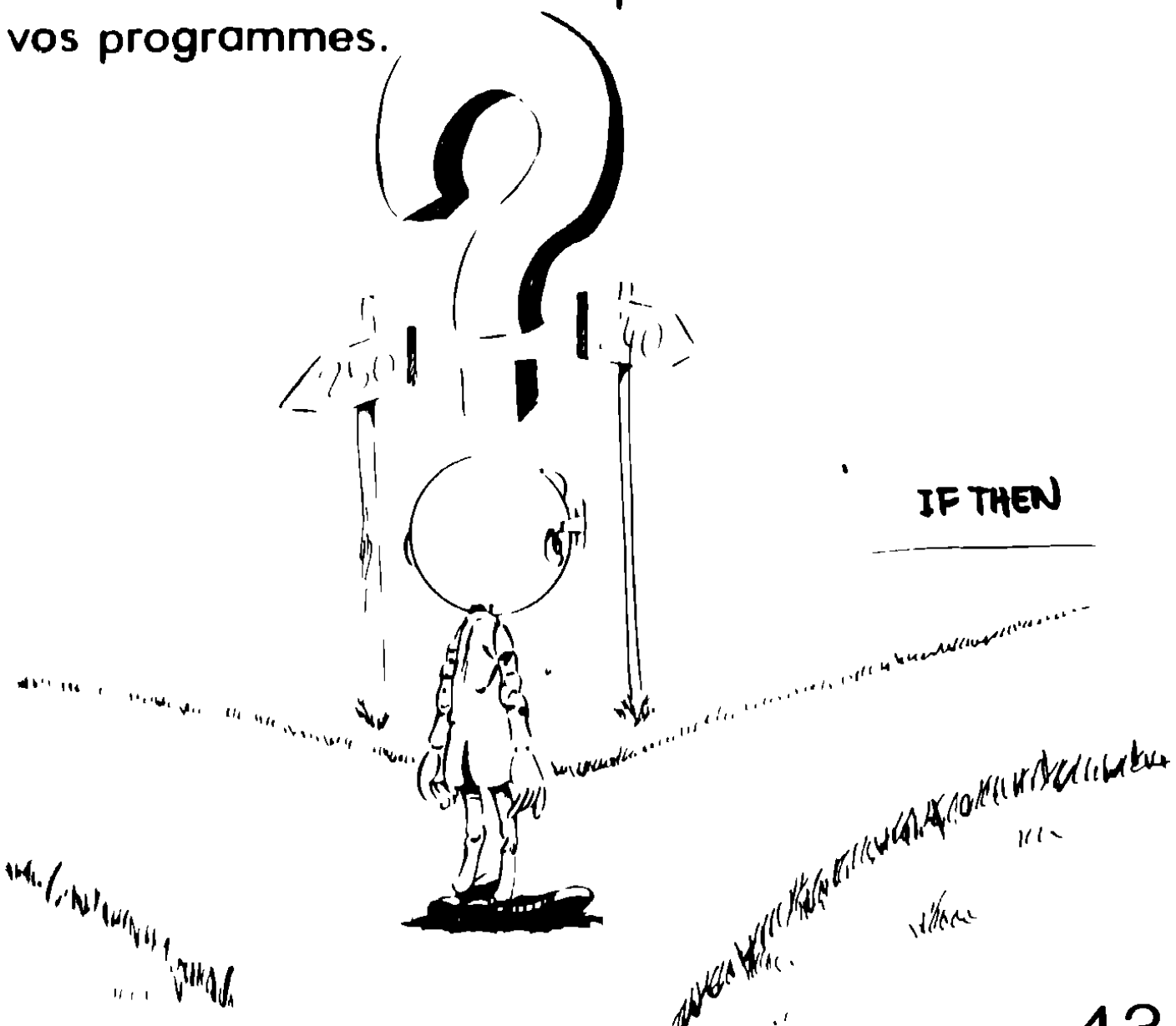
xxxx représente l'adresse à laquelle doit se brancher l'ordinateur si la condition est vérifiée. Si elle ne l'est pas l'ordinateur exécute l'instruction suivant IF... THEN...

On peut utiliser l'instruction sous sa forme :
IF (condition) THEN (action)

On ne se contente pas alors d'indiquer à l'ordinateur où se rendre mais on lui demande

d'agir immédiatement. Par exemple :
IF X = 4 THEN PRINT "BONJOUR"

Dans un premier temps nous vous conseillons d'utiliser cette instruction dans sa première forme afin de faciliter la mise au point et la clarté de vos programmes.

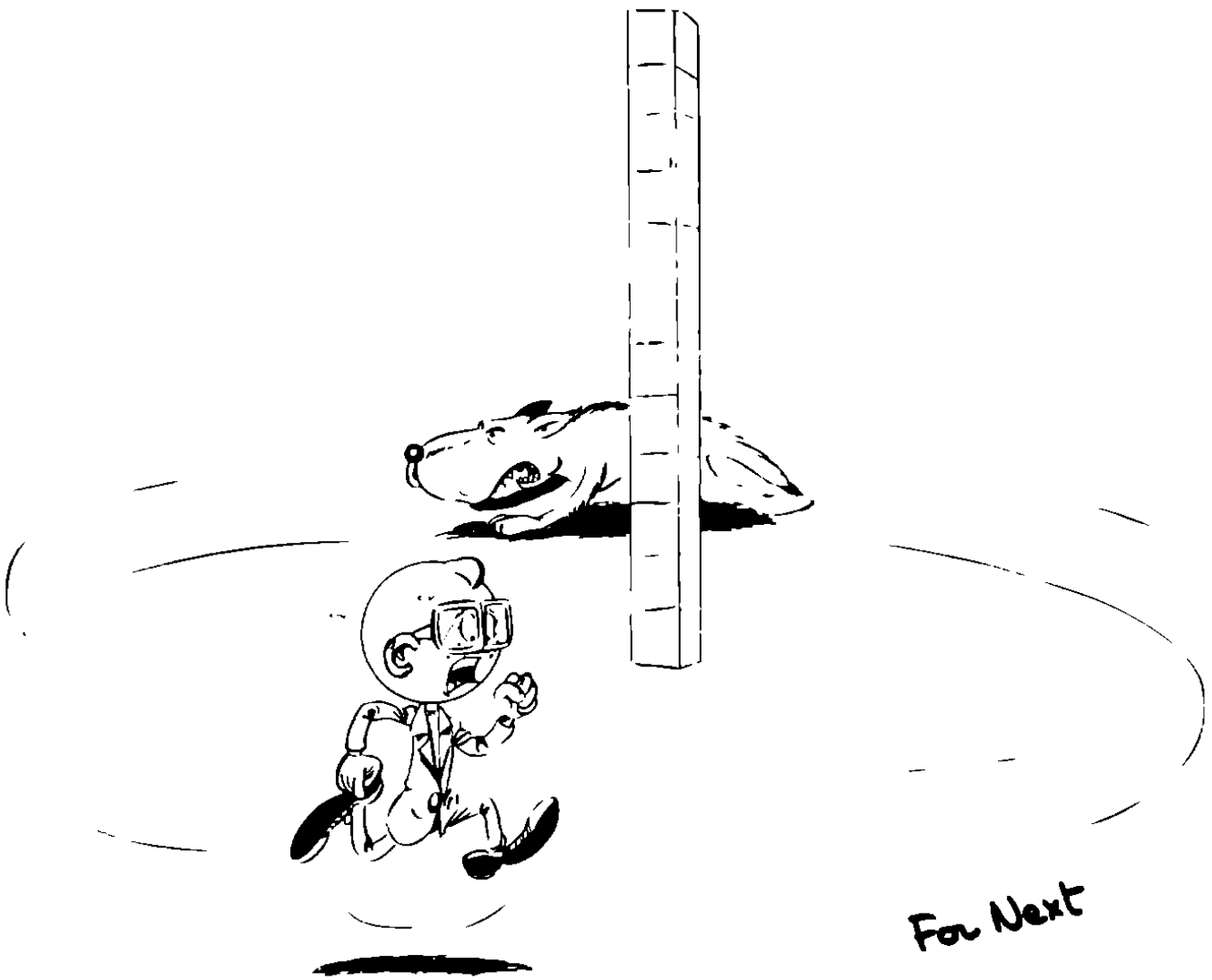


## L'ordre de répétition **FOR... NEXT...**

C'est peut-être l'instruction originelle de l'informatique !!!
Celle qui permet de faire répéter une action un certain nombre de fois avant de passer à l'exécution du reste du programme.
Sa syntaxe est :

```
FOR I = 1 TO 5
Action
NEXT I
```

Elle se décompose en 2 lignes qui donnent le nombre de répétitions à exécuter. Dans notre exemple 5 fois, pour une action donnée.
I est l'indice de la boucle, il représente l'indicateur qui mesure le nombre de fois que l'instruction a été reproduite.
Chaque fois que l'ordinateur rencontre la deuxième partie de l'instruction NEXT I, il augmente la valeur de I de 1 et exécute une nouvelle fois l'instruction l'action requise à condition que la deuxième valeur limite stipulée dans FOR I=1 TO N ne soit pas atteinte sinon il sort de la boucle.


For Next

## Les ordres **DATA** et **READ**

Ils s'utilisent conjointement pour stocker des données dans un programme. Ces données peuvent être soit des nombres, soit des éléments de texte. Les syntaxes sont :

```
DATA 1789,REVOLUTION
FRANCAISE,1515,MARIGNAN,1944,LE
DEBARQUEMENT
READ A,B$,C,D$,E,F$
```

Après exécution de l'instruction de l'ordre DATA, l'ordinateur a rangé en mémoire les renseignements que nous voulions mémoriser et sait où ils sont. L'instruction READ (lire) permet d'aller chercher ces renseignements. Si on donne l'instruction :

```
PRINT A,B$,C,D$,E,F$
```

on obtiendra :

```
1789        REVOLUTION FRANÇAISE
1515        MARIGNAN
1944        LE DEBARQUEMENT
```

Les virgules situées entre les variables les séparent et permettent ainsi à l'ordinateur de les reconnaître. Une instruction DATA peut être

placée à n'importe quel endroit du programme, mais l'ordinateur saura la retrouver en temps utile.

Il existe encore un grand nombre d'instructions du BASIC plus ou moins importantes dont l'emploi sera expliqué tout au long du cube. Par exemple **CLS** qui efface l'écran ou **PLAY "DORE"** qui jouera les deux notes de musique DO et RE.
Nous vous conseillons de consulter le manuel d'utilisation de votre micro-ordinateur qui vous fournira une liste complète des instructions utilisables.

la musique

# *EXERCICES ET JEUX*

*Nous vous proposons plusieurs exercices pour clore ce chapitre. Essayez d'analyser quelques problèmes en rédigeant les ordinogrammes qui leur correspondent :*

*— Vous êtes garde-barrière. Comment gérer le mouvement de votre barrière.*

*— Vous jouez avec un dé. En trois lancers vous devez totaliser au moins douze points mais pas plus. Vous pouvez vous arrêter quand vous le désirez.*

*— Vous vendez des tomates à 6,78 F le kilo. Quel est le programme vous permettant d'obtenir le prix à payer par le client en fonction du poids de marchandise vendue.*

# EXECUTION N° □
# D'UN PROGRAMME
**1 programme**

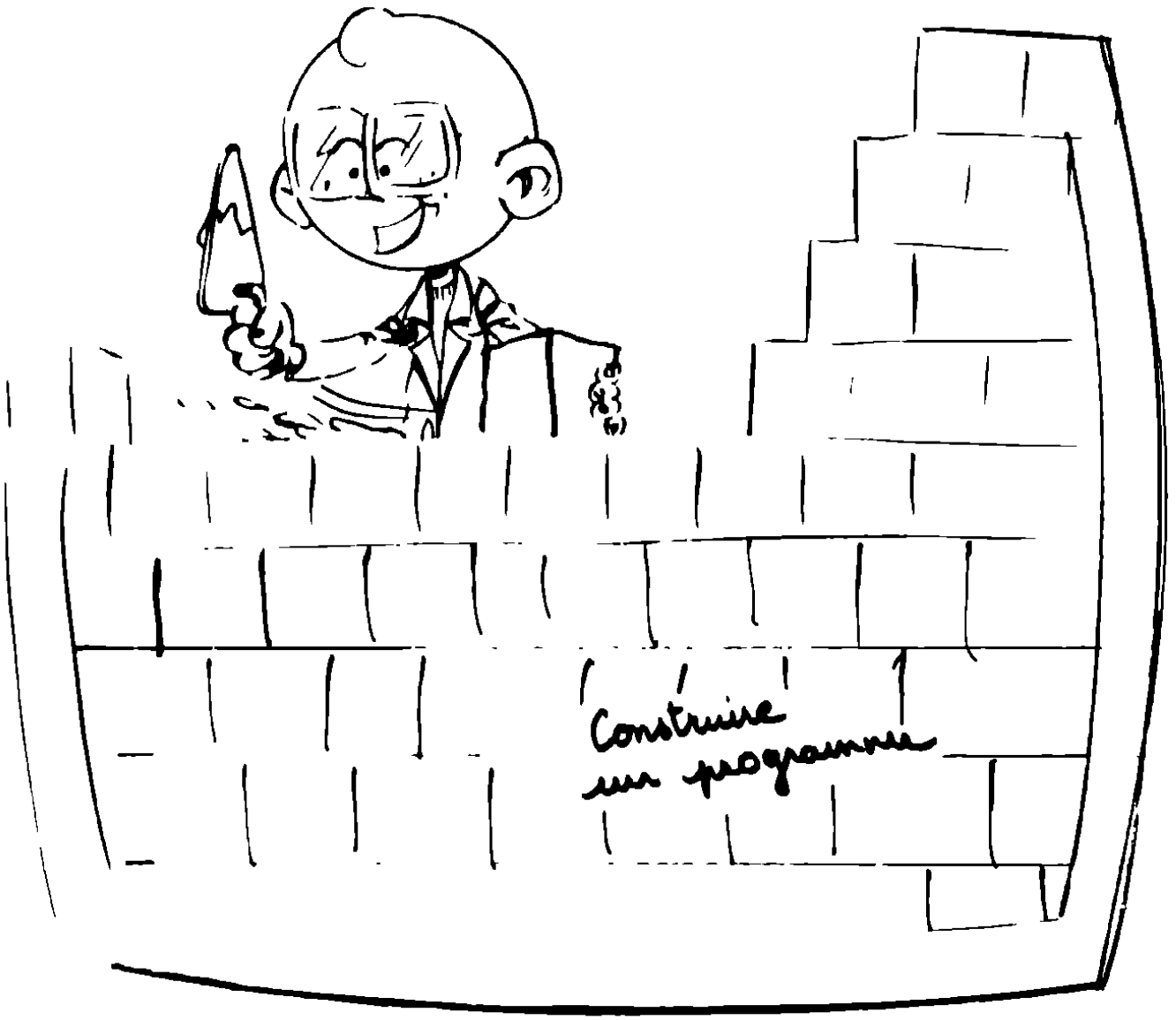
**DEROULEMENT D'UN PROGRAMME**
**SIMULATION DE SON EXECUTION**
**INTERNE ET EXTERNE**
**ASSOCIATION INSTRUCTION-EFFET**

**MOTS-CLEFS**

LISTING, ANALYSE, SIMULATION

# QUELQUES DÉTAILS

Regardez attentivement cette démonstration, elle vous propose simultanément un ordinogramme, un listing et l'exécution d'un programme (d'un lancer de balle) par votre ordinateur. Pas à pas vous pouvez constater facilement les correspondances entre ces trois étapes d'un programme. Remarquez bien que les liaisons entre les différents symboles de l'ordinogramme correspondent bien à l'ordre d'exécution du programme. Les instructions de celui-ci sont rangées les unes derrières les autres par ordre croissant d'adresse.
Un certain nombre d'instructions sont présentées "à l'avance" pour ajouter à la démonstration. Leur syntaxe peut être modifiée, n'en tenez pas compte. Nous reverrons plus tard.
Nous avons volontairement détaillé toutes les phases, sans omettre les répétitions par exemple :
Après cette démonstration nous commencerons l'étude de la rédaction des programmes.


Construire
un programme

# VARIABLES INDICÉES N° □

**2 programmes**

## VARIABLES INDICÉES
## DIMENSIONNEMENT DE VARIABLES

## MOTS-CLEFS

INDICES, DIMENSION, TABLEAU

## QUELQUES DÉTAILS

La manipulation d'un grand nombre de variables impose un certain nombre de précautions : la mémoire de l'ordinateur n'est pas illimitée. Pour cela, il faut les classer dans des tableaux sous les formes de variables dites indicées.

On peut donner des indices aux variables lorsqu'elles ont une homogénéité de sens ou de

fonction (tous les habitants d'une rue, toutes les payes d'une entreprise...). On repèrera donc ainsi les variables non plus suivant leur nom mais le numéro d'indice.

Il faut seulement prendre soin de réserver la place de ces variables en mémoire par DIM (dimension) la syntaxe est la suivante pour les variables à 1 dimension.

DIM A$ (XXX) où XXX est un nombre indiquant le nombre de "places réservées".

Pour les variables à plusieurs dimensions, DIM A$ (XXX, YYY, ZZZ) réserve de la même manière de la place. Prenez garde à ne pas trop en réserver car il n'en resterait plus pour le programme.

## EXERCICES ET JEUX

Exercez vous à reconnaître les places des différentes mémoires du tableau ! Ce n'est pas si facile

Cherchez des exemples de données que vous pouvez traiter sur 1 , 2, 3, 4 ... 255 dimensions.

# QUELQUES DÉTAILS

Un certain nombre d'instructions permettent de modifier des chaînes de caractères (les ajouter, les mesurer...). Il est intéressant de bien les connaître pour l'élaboration de certains programmes.
Leur syntaxe est détaillée dans le manuel de votre micro.

Des exemples vous expliquent leur utilité. Ces fonctions souvent oubliées rendent de nombreux services et évitent souvent des erreurs (confusion numérique/alphanumérique...).

Il faut remarquer que ce traitement de données est semblable à celui qu'exécute l'ordinateur en mémoire (rappeler-vous qu'il tranche, ajoute, décompte... des données élémentaires).

Vous pourrez revoir ce chapitre d'information lors de l'analyse de certains problèmes pour les exemples qu'il propose.

# GRAPHISME N° □

**2 programmes**

## PRESENTATION DES INSTRUCTIONS GRAPHIQUES
## MODE TEXTE/MODE GRAPHIQUE

## MOTS-CLEFS

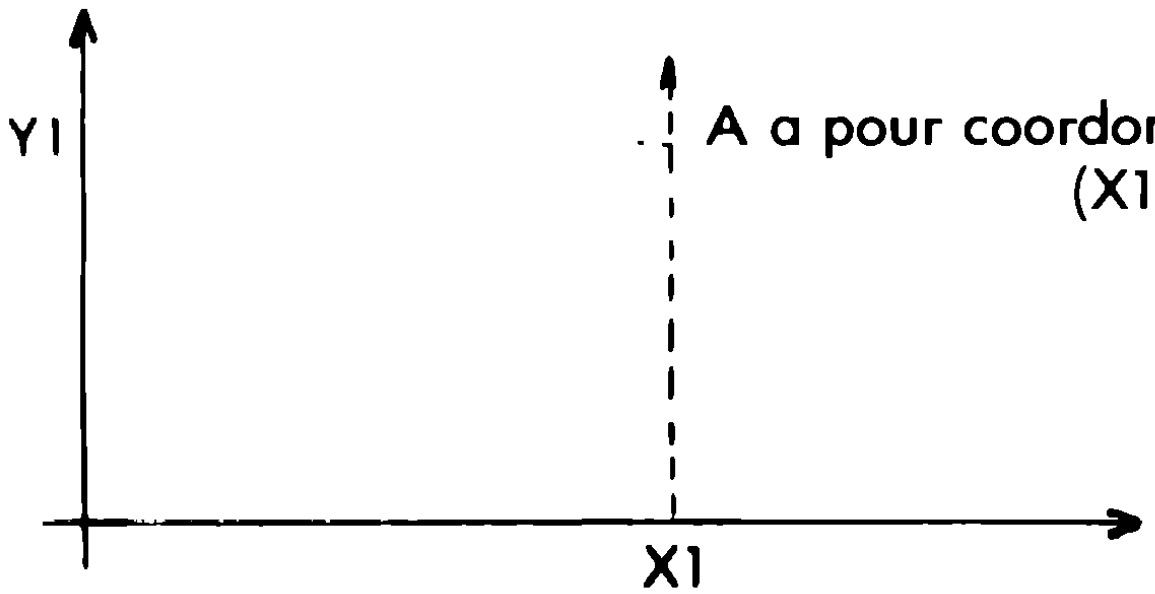
GRAPHISME, DESSIN, MODE

# QUELQUES DÉTAILS

Nous commençons cette cassette par une petite récréation qui nous permet de dresser l'inventaire de tous les outils dont vous disposez pour dessiner sur l'écran. La présentation défile automatiquement en présentant la syntaxe de chaque instruction et un exemple d'application. Si vous trouvez que le rythme est trop rapide vous pouvez appuyer sur STOP, regarder et relancer l'exécution par n'importe quelle touche.

Vous découvrirez la palette de couleur qu'offre votre micro et les différentes manières d'écrire sur l'écran en mode texte.

En mode graphique, la définition de votre micro ordinateur est meilleure qu'en mode texte : une image sur un écran cathodique est composée d'une série de points juxtaposés qui constitue la trame de l'image. Pour écrire ou dessiner, nous ne faisons qu'éclairer et éteindre ces points avec en plus des couleurs. Les couleurs sur l'écran sont repérées par leurs coordonnées, le point A a pour coordonnées X1 et Y1.




Pour réaliser vos dessins ou mises en page, nous vous conseillons d'utiliser une matrice de papier millimétré sur laquelle vous pourrez préparer votre travail. Cela évite les surprises de fautes de coordonnées.

## EXERCICES ET JEUX

*Une série d'exercices sur les couleurs et les positions vous entraînera à l'utilisation du graphisme :*

- *Recherche du code couleur*
  *Repérage sur l'écran*
  *Manipulation de différentes fonctions.*

# STRUCTURE N° □
# DE PROGRAMMATION

**2 programmes**

## STRUCTURE D'UN PROGRAMME
## ORGANISATION DES INSTRUCTIONS
## BOUCLE DE SAISIE

## MOTS-CLEFS

ANALYSE, ORDINOGRAMME,
DONNEES, SAISIE

# QUELQUES DÉTAILS

## PROGRAMME 1

C'est la partie cruciale de votre initiation.

Après avoir recensé tous les outils dont nous disposons avec le langage BASIC et la nature des traitements informatiques, nous allons commencer la rédaction de programmes.

Une première partie explique comment s'organise l'exécution d'un programme en fonction des ordres donnés, sans syntaxe. Dans la deuxième partie, la syntaxe précise sera prise en compte.

Lors de l'exécution de ce programme imaginaire, vous découvrirez une nouvelle fonction, jamais vue **GOSUB...RETURN**, c'est un **GOTO** (instruction de saut) qui entraîne l'exécution d'une action avec un retour automatique : cette action est appelée sous-programme. Ainsi c'est l'ordre d'afficher qui est le sous-programme dans notre exemple. Cette notion difficile pour les débutants doit être connue car c'est le nerf des programmes complexes qui avant de se décomposer en tâches

élémentaires comme pour être analysé en termes de sous-programmes moins importants.
Ces sous-programmes sont eux-mêmes décomposés en tâches élémentaires.

Le programme sur Henri IV est évolutif. Nous allons le compliquer au fur et à mesure pour le rendre plus difficile, plus riche. Commençant par un objectif simple, nous finirons avec un programme riche exploitant presque toute la batterie d'ordres BASIC ! Vous devez chercher à analyser chaque ligne d'instruction pour comprendre sa place et sa fonction. N'oubliez jamais que votre micro vous répétera aussi longtemps que nécessaire une explication si vous butez sur un point précis.

**1re ETAPE :**

Nous posons une question, attendons la réponse RE$ que nous testons. Tant que la réponse n'est pas exacte, nous reposons la question. Si la réponse est juste, nous affichons "c'est cela".

Construisez l'ordinogramme de ce problème et regardez son exécution.

**2e ETAPE :**

Nous ne testons plus une question mais dix. Nous pourrions les tester dix fois comme précédemment mais il est interressant de noter l'utilisation d'une boucle FOR...NEXT combinée à un READ...DATA.

Soit le programme :

```
10 FOR I=1 TO 5
20 READ A
30 PRINT A
40 NEXT I
50 DATA 10, 20, 30, 40, 50
```

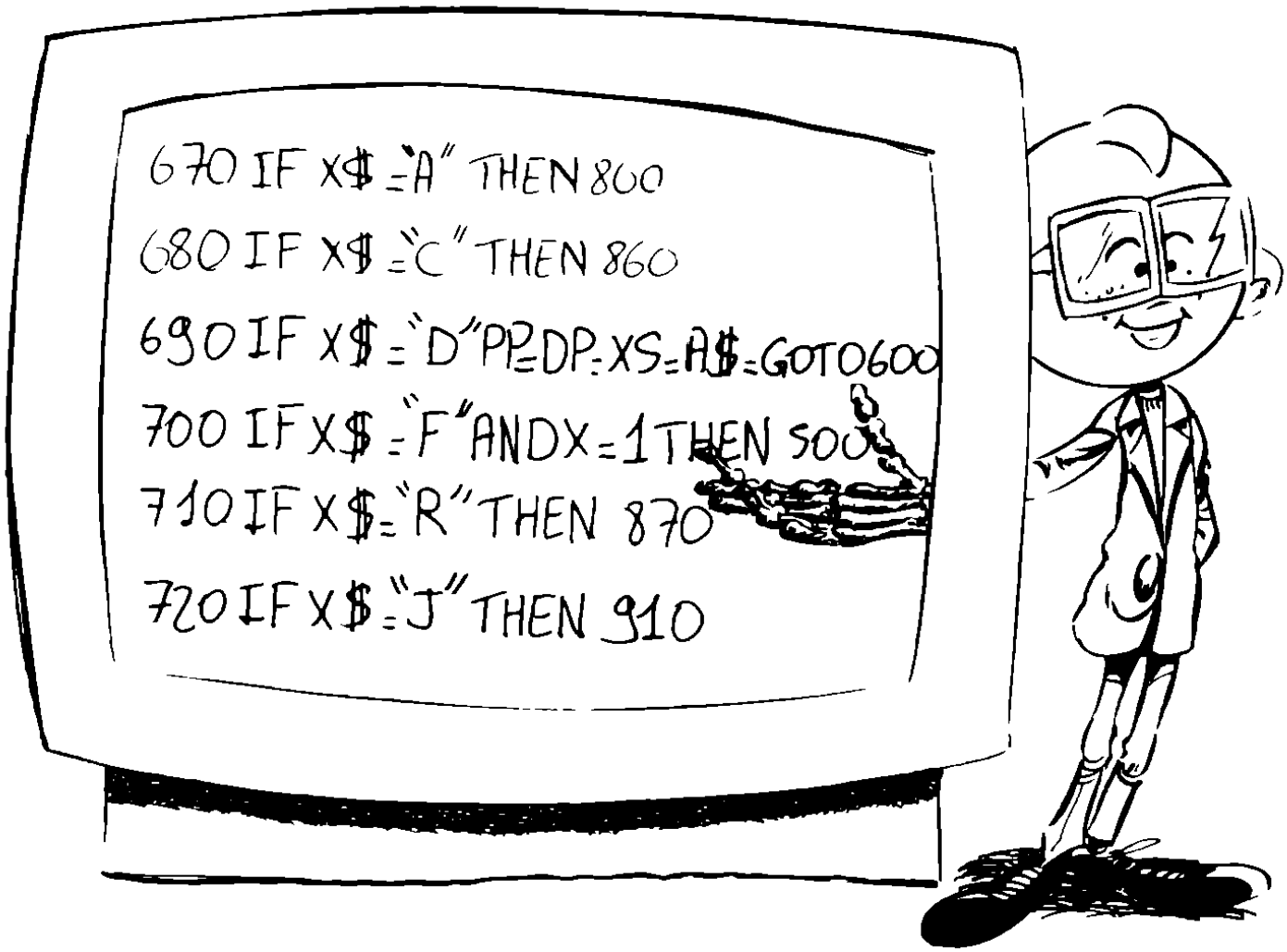
Son exécution donne l'affichage à l'écran de 10 20 30 40 50
Que c'est-il passé ?
READ A fait lire la première donnée non utilisée dans DATA. Ainsi au premier tour de la boucle A -- 10 , au deuxième 20...

Cette simulation vous fait toucher du doigt un fait important de la programmation : les possibilités de faire évoluer un programme. Le jeu de questions aux Shadoks IV est constitué d'un "noyau" qui est l'interrogation d'une réponse et le test sur son exactitude. Tout le reste n'est que présentation et facilité. Il faut toujours chercher à résoudre d'abord l'essentiel d'un problème avant de se lancer dans l'écriture irréfléchie de lignes.

# EXERCICES ET JEUX

## PROGRAMME 2

*Afin de participer un peu, l'ordinateur vous a préparé un exercice d'élaboration d'un répertoire où vous devez taper les instructions qui correspondent aux ordres de l'ordinogramme...*
*BONNE CHANCE.*




execution d'un programme

# ECRIRE UN PROGRAMME N° □

## ECRITURE D'UN PROGRAMME BASIC
## SIMULATION D'EXECUTION

## MOTS-CLEFS

ORDINOGRAMME, SYNTAXE, LISTING.

# QUELQUES DÉTAILS

L'exercice qui vous est proposé va vous permettre de suivre un ordinogramme imposé en tapant vous-même les ordres à transmettre. Les erreurs éventuelles seront analysées et commentées et vous aurez droit à autant d'essais que nécessaire. Si vous désespérez, la touche ACC vous donne la solution.

Nous vous conseillons de recopier l'ordinogramme car le programme est en deux parties et vous risquez de perdre la page... de votre répertoire.

Après l'avoir tapé, vous verrez l'exécution de votre programme. Vous pourrez également le modifier à loisir, l'étoffer. Pour cela repérer bien sur la cassette l'endroit à partir duquel démarre l'exécution du répertoire. Une fois le programme chargé, tapez CNT+C, vous arrêtez l'exécution et vous taper LIST, vous constaterez que le listing est le même que celui que vous avez tapé. Vous pouvez alors avoir les valeurs des couleurs, la forme des lettres... et même la logique du programme !

N'oubliez pas de sauvegarder votre programme modifié sur une cassette vierge (ordre SAVE).

Ainsi, depuis le branchement de votre micro, nous avons découvert comment fonctionnaient les mémoires, la manipulation, les données, l'organisation, les syntaxes et enfin, la construction d'un programme. Tout semble acquis... et pourtant, le plus dur reste à faire. L'informatique demande de l'entraînement et de la manipulation régulière. Nous n'avons pas d'autres prétentions que de vous initier.

La dernière partie de notre présentation va concerner un aspect fondammental de l'informatique : les fichiers.

les fichiers

# FICHIER N° □

## FICHIER EXTERNE

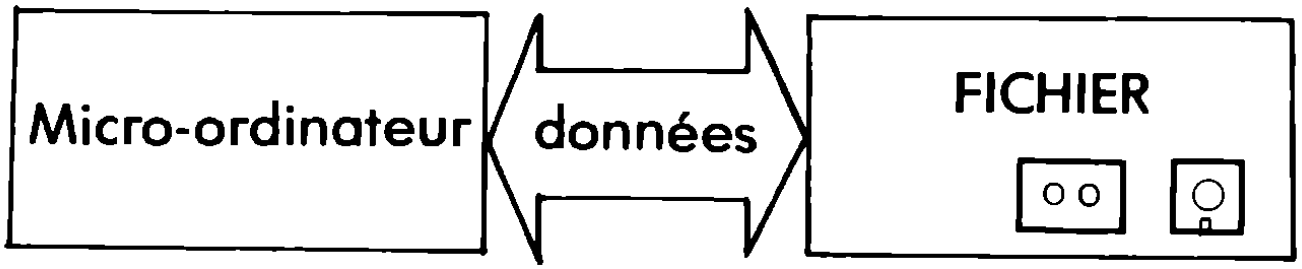
## MOTS-CLEFS

FICHIER

## QUELQUES DÉTAILS

Jusqu'à présent nous avons utilisé un magnétophone à cassettes pour lire des PROGRAMMES. On peut également envisager de pouvoir stocker des DONNEES sur cassette , entendez sans les intégrer dans un programme: c'est un fichier externe de données.

On peut résumer la situation par le schéma suivant :




Un programme en mémoire peut lire ou écrire des résultats sur une bande magnétique. Ce fichier ne prend alors plus de place en mémoire mais l'accès à l'information est plus long. En effet, il faut ouvrir l'accès au fichier, trouver la donnée, l'acquérir, fermer le fichier : tout ceci est bien plus long qu'une recherche en mémoire centrale.

**Quelle est la procédure pour ouvrir ce fichier ?**

1/ Autoriser une liaison externe, soit en écriture (**OUTPUT**) soit en lecture (**INPUT**) par **OPEN.**

2/Utiliser des ordres spéciaux pour écrire ou lire :
PRINT # 1 le # 1 caractérise le fichier et oriente
INPUT # 1 l'écriture vers la bande ou la lecture à partir de la bande.

3/ Fermer la liaison par un **CLOSE**

Ces trois ordres vont toujours ensemble. Sur une bande magnétique, ils signifient :

OPEN "O", # 1, "VINS" = ouvre un fichier appelé vins en écriture (0). Son numéro est 1.

**PRINT # 1, "XXX" = écrit XXX dans le fichier numéro 1.**

CLOSE # 1 = ferme le fichier numéro 1.

Si on veut lire sur ce même fichier, il faut l'ouvrir en lecture (I).OPEN "I", 1, "VINS" ; car la liaison magnétophone/unité centrale est toujours unilatérale. On ne peut lire et écrire en même temps sur une bande.

Si on ouvre un fichier en lecture ou en écriture, il faut prendre soin d'enclencher le magnétophone, la bande démarre et les données sont rangées comme des signaux de programmes.

Ces fichiers ont un avantage supplémentaire ; ils peuvent être exploités par plusieurs programmes. Imaginons que vous créez un fichier de nom et

## EXERCICES ET JEUX

*Voici quelques thèmes de travail pour créer des fichiers :*

*- Gestion de votre automobile. Constituez un fichier de votre consommation par rapport aux kilomètres parcourus. Vous utiliserez ce fichier pour éditer vos moyennes et les rapporter à la moyenne annuelle.*

*- Fichier du jeu sur Henri IV vu précédemment.*

# BILAN

Au terme de cette initiation, résumons les principes fondamentaux qui régissent la création de tout programme :

**ANALYSER LE PROBLEME**

Nous avons vu que l'ordinateur ne sait finalement pas grand-chose ! Il se contente de reconnaître des 0 et des 1. Et pourtant avec des ordinateurs, on dépouille des élections, on calcule des trajectoires de fusées... Pourquoi ? Parce que tout travail à exécuter peut se décomposer en tâches élémentaires s'il est correctement analysé.

**CHERCHER A ETRE SIMPLE ET CLAIR**

Les ordres principaux de gestion des informations sont simples, aussi faut-il ne pas compliquer les problèmes. Boucles, sauts, acquisition... restent les tâches élémentaires qu'il faut exploiter à fond.

## COMMENCER PAR L'ESSENTIEL

Ne vous noyez pas dans les programmes de présentation, de couleurs... Choisissez en premier lieu, le "noyau dur" de votre problème, testez-le puis alors seulement occupez-vous de la présentation.

## FAITES UN ORDINOGRAMME

C'est un conseil de vieux routier. Sans plan, on se perd dans la forêt même si elle paraît petite.

## MAIS UTILISEZ-LE

Ne vous contentez pas de lui pour l'analyse, suivez-le pour la rédaction du programme. Trop souvent les débutants griffonnent quelques flèches sur un brouillon qu'ils oublient pour le plaisir de taper sur le clavier. Les deux actions sont pourtant complémentaires ! Vous avez toujours la possibilité de titrer vos paragraphes de listing par l'instruction REM qui disparaîtra ensuite dans le programme final. En effet REM

introduit une phrase qui n'est pas prise en compte dans l'exécution d'un programme :

Exemple

10 REM Recalcul Hauteur B = A + 4. 20 B = A+4
Faire B = A + 4
20 B = A + 4

REM ne fait que présenter l'action suivante, il n'y a pas affichage sur l'écran. Lorsque tout fonctionne, vous effacerez les lignes REM.

## RESPECTER LES SYNTAXES

Même si dans la langue française, il existe des homonymes dont on sait distinguer le sens, pour un ordinateur, un mot a une seule signification et une seule écriture. Aussi ne ragez pas s'il ne comprend pas vos ordres tronqués ou vos abréviations.

## PREVOIR UNE SORTIE

Une astuce qui évite de perdre le contrôle des programmes : prévoyez d'en sortir !!! Soit par END, soit par un choix de touche "Voulez-vous continuer oui/non ?".

## SAUVEGARDER REGULIEREMENT

Au fur et à mesure de l'écriture de votre programme, prenez soin de le sauvegarder, ainsi, en cas de problème, il vous restera une partie importante de travail qu'il vous sera facile de relire. Ceci évite les longues heures de recopiage de listing.

Cette série de conseils doit être en permanence votre règle de conduite pour vos premiers programmes.

Naturellement, au fil du temps, vous leur préférerez peut-être d'autres méthodes de travail mais à ce moment-là... vous n'aurez plus besoin de conseils.

# ET SI
# ÇA NE MARCHE PAS

Confortablement installé à l'écran, vous avez enfin tapé votre programme de prévision de vacances, vous pressez RUN et... ERROR s'affiche à l'écran. Que faire ?

La première vérification est de détecter le type d'erreur par le numéro du message à l'écran. Une liste des messages d'erreur en annexe du manuel d'utilisation de votre micro ordinateur vous permettra de l'identifier.

Mais vous aviez peut-être déjà fait une erreur grave ! ! ! ! **AVEZ-VOUS SAUVEGARDÉ VOTRE PROGRAMME ?** En effet, il est possible que vous ayez perdu toutes vos données et que dès lors, l'erreur soit irrémédiable. Vous devez retaper tout le listing. Dommage ! Avant l'exécution de tout le programme, nous vous conseillons de le sauvegarder sur cassette.

Le blocage de l'exécution de votre programme peut avoir trois origines principales :

- une faute de logique
- une faute de syntaxe
- une erreur d'utilisation d'instruction

Dans le premier cas, en fait, vous n'aurez pas forcément d'erreur mais des résultats erronés. Par exemple, vous voulez connaître le prix d'un objet et l'ordinateur vous communique 20 kilogrammes. Reprenez alors votre ordinogramme et votre listing, un mauvais branchement, un nom de variable utilisé deux fois... la solution n'est pas forcément simple et vous devez procéder pas à pas.

La faute de syntaxe est la plus banale. Généralement elle est traitée dès la saisie du programme où le numéro de la ligne fausse est précisé. Corrigez simplement votre vocabulaire !

L'erreur d'utilisation d'instruction mèle à la fois les fautes de logiques et de syntaxe. Ce peut être par exemple un READ sans DATA, un FOR sans NEXT ou bien un READ sans assez de

données dans DATA, un mauvais dimensionnement de variables par DIM...

La méthodologie est toujours la même :

**VERIFICATION** du listing par **LIST** puis **CORRECTION** des problèmes un par un. Une fois corrigé, n'oubliez pas de sauver à nouveau votre programme avant de le lancer ou d'éteindre votre micro. Il ne vous resterait que l'ancienne version et vous auriez travaillé pour recommencer.

# INDEX DES MOTS-CLEFS