hors série

Led MICRO

PROGRAMMATION
COURS 2ème CYCLE

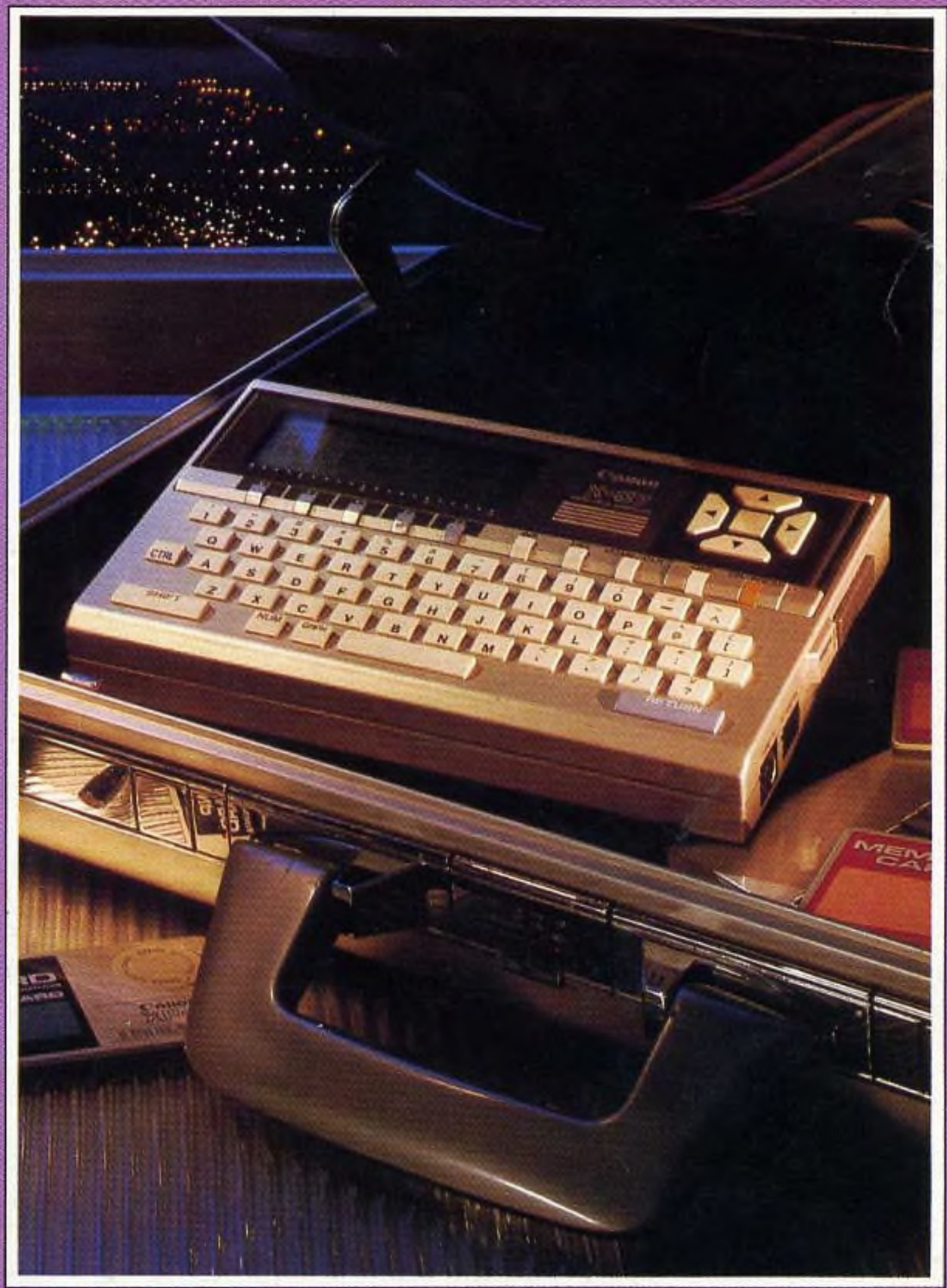
COURS
N°22
Suite
2e cycle
N°2

COURS DE BASIC : programmation structurée

COURS DE PROGRAM-MATION APPROFONDIE : les benchmarks

COURS DE GENIE LOGICIEL : de la théorie à la pratique

C'EST ARRIVE DEMAIN !



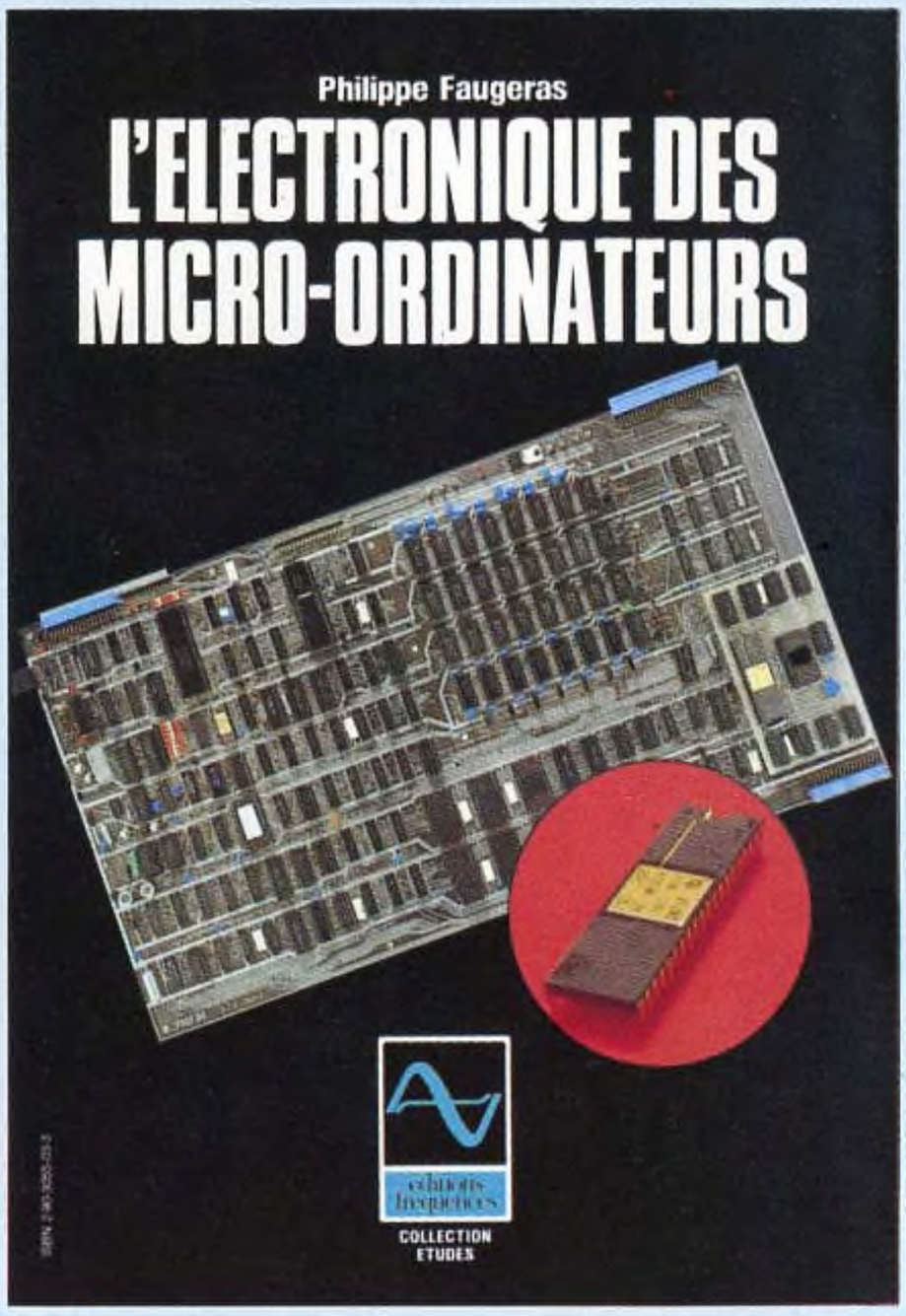
Le X-07 de Canon

ISSN 0757-6889

M 1988 - N° 22 - (2) - 18 F MENSUEL SEPTEMBRE 1985 - BELGIQUE 139 FB/CANADA 3,75 $/SUISSE 6,75 FS

VOYAGE AU CŒUR DES MICRO-ORDINATEURS

dans la COLLECTION «ETUDES» aux éditions fréquences



une véritable schémathèque

- 128 pages
- 101 schémas
- 34 tableaux

Prix : 150 F

Que ce soit pour concevoir des interfaces ou optimiser un programme (utilisation des périphériques, encombrement mémoire...) «un micro-informaticien performant» doit posséder une bonne connaissance de son matériel.

Ce livre s'adresse donc à tous les électroniciens qui désirent découvrir les différents composants constituant un micro-ordinateur. Articulé autour du microprocesseur Z80, cet ouvrage contient de nombreux schémas (plan mémoire, interfaces série et parallèle, interface clavier, interface vidéo, CAN, CNA...) qui pourraient être le thème... de nouvelles extensions.

En vente chez votre libraire et aux Editions Fréquences

BON DE COMMANDE

Je désire recevoir l'ouvrage **«l'électronique des micro-ordinateurs»** au prix de **165 F** (150 F + 15 F de port).

Nom . . . . . .

Adresse . . . . . .

. . . . . .

A adresser aux EDITIONS FREQUENCES 1 boulevard Ney, 75018 Paris

Règlement ci-joint :

Par chèque bancaire ☐ par chèque postal ☐ par mandat ☐

Philippe Faugeras, Docteur-ingénieur en électronique a acquis son expérience dans de grandes entreprises françaises où pendant cinq ans, il a travaillé sur des systèmes d'automatismes à base de microprocesseurs. Philippe Faugeras est responsable de la rubrique «Raconte-moi la micro-informatique» dans la revue LED.

hors série
Led MICRO
PROGRAMMATION COURS 2e CYCLE

Société éditrice :
Editions Fréquences
Siège social :
1, bd Ney, 75018 Paris
Tél. : (1) 607.01.97 +
SA au capital de 1 000 000 F
Président-Directeur Général :
Edouard Pastor

LED MICRO
(cours 2e cycle)
Mensuel : 18 F
Commission paritaire : 64949
Directeur de la publication :
Edouard Pastor

Services **Rédaction-Publicité-Abonnements :**
1, bd Ney, 75018 Paris
Tél. : (1) 607.01.97
Lignes groupées

Comité de rédaction :
Dominique Chastagnier
Jean-François Coblentz
Charles-Henry Delaleu
Patrick Gueneau

Secrétaire de Rédaction
Chantal Cauchois

Publicité, à la revue
Tél. : 607.01.97
Secrétaire responsable
Annie Perbal

Abonnements
10 numéros par an
France : 160 F
Etranger : 240 F

Réalisation
Composition-Photogravure
Edi'Systèmes

Impression
Berger-Levrault - Nancy

SEPTEMBRE 85

COURS N°22 Suite 2e cycle N°2

COURS DE BASIC
Programmation structurée
de la page 5 à la page 13
Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

COURS DE PROGRAMMATION APPROFONDIE
Les benchmarks
de la page 16 à la page 32

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

C'EST ARRIVÉ DEMAIN
de la page 33 à la page 35

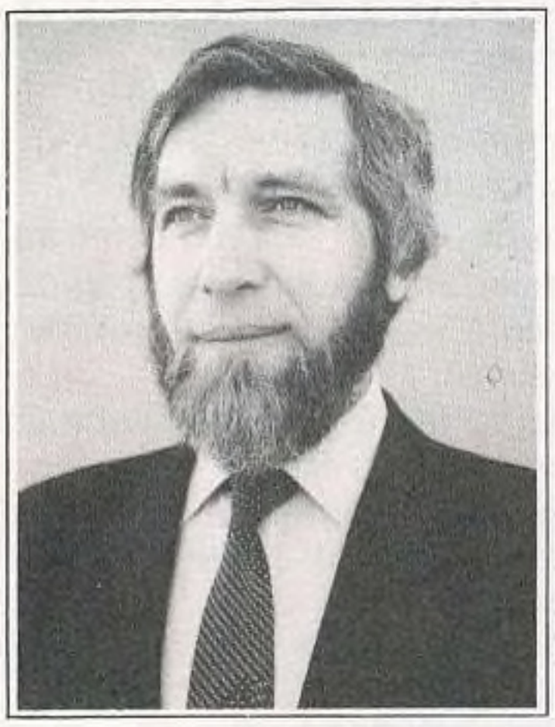
COURS DE GENIE LOGICIEL
De la théorie à la pratique
de la page 37 à la page 49

Charles-Henry Delaleu

NOTRE COUVERTURE : Le Canon X-07, un portable ; à emporter partout. Canon France 93154 Le Blanc-Mesnil Cedex - Tél. : (1) 865.42.23.

Electronique digitale ?

Notre temps aura témoigné d'une nouvelle technique, une autre façon de communiquer avec l'électronique digitale.
Philippe Duquesne, professeur chargé de cours au CNAM a su dans cet ouvrage en expliquer clairement les fondements.



Philippe Duquesne, ingénieur électronicien (I.S.E.N.) est chargé du cours de microprocesseurs au C.N.A.M. de Paris. Depuis plus de dix ans, il a pris goût à l'enseignement et il est l'auteur d'un ouvrage didactique sur l'électronique digitale et notamment d'un cours pratique de microprocesseurs. Fervent pratiquant du « dialogue» école/industrie, après avoir exercé les fonctions de chef de département électronique chez Burroughs, second constructeur mondial en informatique, il est actuellement chef du service Etudes Electroniques au sein de la direction technique chez Messier Hispano Bugatti (groupe SNECMA) avec, pour principal objectif l'introduction des microprocesseurs dans les trains d'atterrissage.

En vente chez votre libraire et aux Editions Fréquences

Bon de commande

Je désire recevoir le livre : INITIATION A L'ELECTRONIQUE DIGITALE au prix de 105 F (95 F + 10 F de port).
Adresser ce bon aux EDITIONS FREQUENCES 1, bd Ney, 75018 PARIS

Nom ..... Prénom .....
Adresse .....
..... Code postal .....

Règlement effectué : ☐ par C.C.P.
☐ par chèque bancaire
☐ par mandat

COURS DE BASIC

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

PROGRAMMATION STRUCTUREE

1. Introduction

Dans ce cours, nous abordons les commandes permettant de réaliser des programmes correctement écrits, lisibles et efficaces. Pour cela, nous allons évoquer en permanence la programmation structurée. Pas de panique, il s'agit là encore, et comme souvent d'un bien grand mot pour évoquer tout simplement une programmation plus stricte et rigoureuse.
La plupart d'entre vous programment déjà ainsi, car c'est la meilleure façon de rédiger des programmes un peu longs, sans s'emmêler dans des lignes qui en appellent d'autres, ces dernières en appelant d'autres encore.

```
10 DIM A$(10)
20 FOR I=1 TO 10
30 INPUT " NOM ";A$(I)
40 IF A$(I) = " " THEN GOTO 1000
50 NEXT
100 PRINT "IL Y A DONC ";I;" JOUEURS"
110 PRINT " PAS D'ERREURS ?"
120 INPUT " TAPER E POUR ERREUR";E$
130 IF E$='E' THEN GOTO 2000
200 REM DEBUT DU JEU
    ........................
1000 REM TRAITEMENT DES JOUEURS
1010 FOR J=1 TO I
1020 PRINT " NOM DU JOUEUR "; A$(I)
1030 INPUT " TAPER E POUR ERREUR ";E$
1040 IF E$=E THEN 2000
2000 REM TRAITEMENT DES ERREURS
2010 ........................
```

Etc.

NOTES PERSONNELLES

En résumé, il s'agit d'éviter le capharnaüm dans votre programme préféré. En effet, programmer en structurant, cela signifie simplement réfléchir puis programmer d'une façon logique et efficace, en sachant parfaitement où l'on va. Pourquoi cela ? Pour pouvoir réaliser des programmes qui tournent le plus vite possible, et ne pas être tributaire de longues périodes de mises au point.
Programmer structuré, cela implique de réaliser des petits modules qui, réunis, vous donneront la forme définitive, la structure de votre programme. Prenons un exemple. Si vous avez besoin de réaliser des sorties sur imprimantes de résultats de votre programme, il serait dommage de réécrire les commandes nécessaires à chaque fois, alors que quelques commandes dans un coin du programme vous permettent de le faire une fois pour toutes. Il en va de même si, au cours de votre programme, vous devez stocker des données sur un support externe (bande, disquette, etc.). Un petit module des commandes adéquates fera très bien l'affaire.
Donc programmez à l'économie, pour programmer bien, rapide et efficace. En faire moins pour faire les choses mieux, voilà ce qui est sympathique, n'est-ce-pas ?

2. Description des commandes Basic

Les commandes permettant de réaliser cet alléchant programme sont en petit nombre, donc vite connues, et très simples d'emploi, ce qui ne gâte rien. Ce sont les commandes :

- GOTO
- ON GOTO
- GOSUB
- ON GOSUB
- RETURN
- POP

Vous connaissez tous déjà les deux premières. Elles vous permettent de vous déplacer à votre guise dans un programme. Mais attention, **DANGER**. Chaque GOTO est un handicap dans un programme, et ce de plusieurs points de vue. Tout d'abord, sur le plan lisibilité de ce programme : à chaque fois que vous, lecteur d'un programme, vous rencontrez un GOTO, il vous faut chercher la ligne d'arrivée, puis lorsque c'est fait, retourner au point de départ, là où vous en étiez pour reprendre le fil de vos pensées et vous remémorer la valeur des variables du programme, et enfin poursuivre le programme jusqu'au prochain déroutement. Cela engendre des «promenades» dans tout le programme, ce qui, convenez-en n'est pas très rationnel.
Ensuite, si vous devez trouver l'endroit du programme où se trouve la ligne appelée, l'ordinateur lors de l'exécution doit en faire autant, ce qui représente une perte de temps qui peut être importante (de l'ordre de la seconde parfois), ce qui constitue un handicap important (figure 1) pour des programmes longs, ou demandant des performances élevées, comme des programmes graphiques d'animation, par exemple.

```
100 GOTO 10000
200 :::::::
    :::::::
10000 :::::::
```

Figure 1

Dans ce cas, pour un programme bien rempli, plus d'une seconde est perdue par rapport au même appel, mais en ligne 1. Si vous utilisez des GOTO, mettez les lignes

appelées parmi les premières du programme, ce qui va hélas à l'encontre d'une meilleure clarté du programme.
Enfin, gardez toujours en mémoire que cette commande est pratiquement la seule qui puisse engendrer des bouclages de programmes, c'est-à-dire que par une succession de GOTO, vous exécutez toujours les mêmes lignes, sans en sortir. Donc votre programme ne se termine jamais, il boucle.
Il faut savoir qu'en Basic, un GOTO est parfois inévitable, mais sachez en user avec modération, comme il est dit sur les paquets de cigarettes ou les bouteilles d'alcool.
Sous certaines conditions, GOTO peut être omis, comme vous l'avez déjà vu. Dans ce cas, on dit que la commande GOTO est implicite, l'interpréteur Basic sachant parfaitement comment traiter ce manque, qui est appelé commande par défaut, ce qui signifie simplement que si la machine rencontre

IF ... THEN xx,

elle sait qu'elle doit le considérer comme la commande I

F ... THEN GOTO xx (figure 2).

```
20 IF X=Y THEN 200          est équivalent à

20 IF X=Y THEN GOTO 200
```

Figure 2

La commande ON GOTO est proche de GOTO dans son principe, mais elle représente une manière très élégante de répartir les tâches dans un programme, après menu de possibilités par exemple (figure 3). Donc, son utilisation, qui est peu fréquente représente, au contraire de GOTO, un apport de simplicité dans un programme, en général tout au moins. Attention, en ce qui concerne le temps d'exécution, les mêmes remarques que pour GOTO peuvent être émises.

```
100 PRINT "FAITES VOTRE CHOIX"
200 PRINT " 1 : DESSIN D'UN MOUTON"
300 PRINT " 2 : DESSIN D'UN CHEVAL"
400 PRINT " 3 : DESSIN D'UNE MAISON"
500 PRINT " 4 : FIN DES DESSINS"
600 INPUT " CHOIX : "; I
1000 ON I GOTO 2000, 3000, 4000, 5000
```

Figure 3

Nous abordons maintenant les réelles nouveautés.
GOSUB vous permet d'entrer dans un «sous-programme». Un sous-programme, c'est un bloc d'instructions que vous pouvez «appeler» quand bon vous semble, et qui se termine par le mot RETURN (figure 4), de façon impérative.
GOSUB s'utilise exactement comme GOTO (figure 4), c'est-à-dire en indiquant un numéro de ligne, qui devient virtuellement le début du sous-programme. Mais attention, votre machine, elle, a bien fait la différence, et elle a noté que vous êtes dans un sous-programme, et non dans le programme dit «principal».
Pour sortir du sous-programme, la solution la plus courante est le mot RETURN, qui donne l'ordre au programme de retourner à la commande qui suit immédiatement après le GOSUB (figure 4). On reprend alors le fil du programme principal.

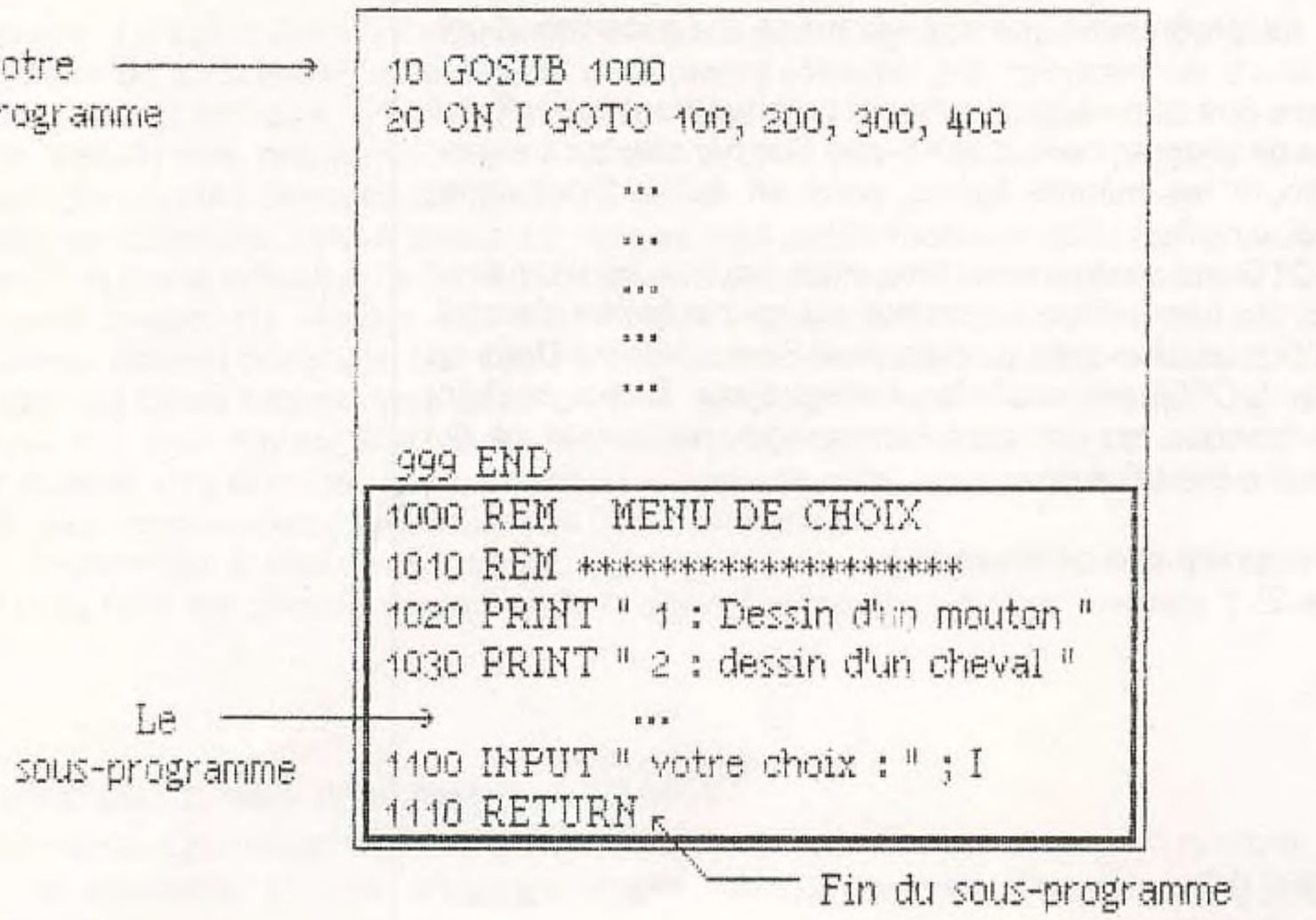


Figure 4

Il faut noter que lorsque le programme rencontre le mot RETURN, il n'a pas besoin du numéro de ligne (figure 4), car il l'a stocké en mémoire, dans ce qui s'appelle une pile, notion que vous verrez en détail dans le cours de C.H. Delaleu. Si vous utilisez un sous-programme alors que vous êtes déjà dans un sous-programme, pas de panique, il ne se passe rien de différent. Sachez que vous pouvez ainsi enfoncer de plusieurs niveaux, en général six, dans des sous-programmes. Mais il faudra en ressortir.
Un des intérêts majeurs de la commande RETURN est justement de ne pas être contraint de préciser la ligne de retour, ce qui permet d'appeler un sous-programme de n'importe quelle partie du programme et, lorsque ce sous-programme est achevé, RETURN vous ramène à l'endroit où vous vous trouviez précédemment, ce qui permet d'utiliser un sous-programme de multiples façons et de multiples endroits (figure 5).

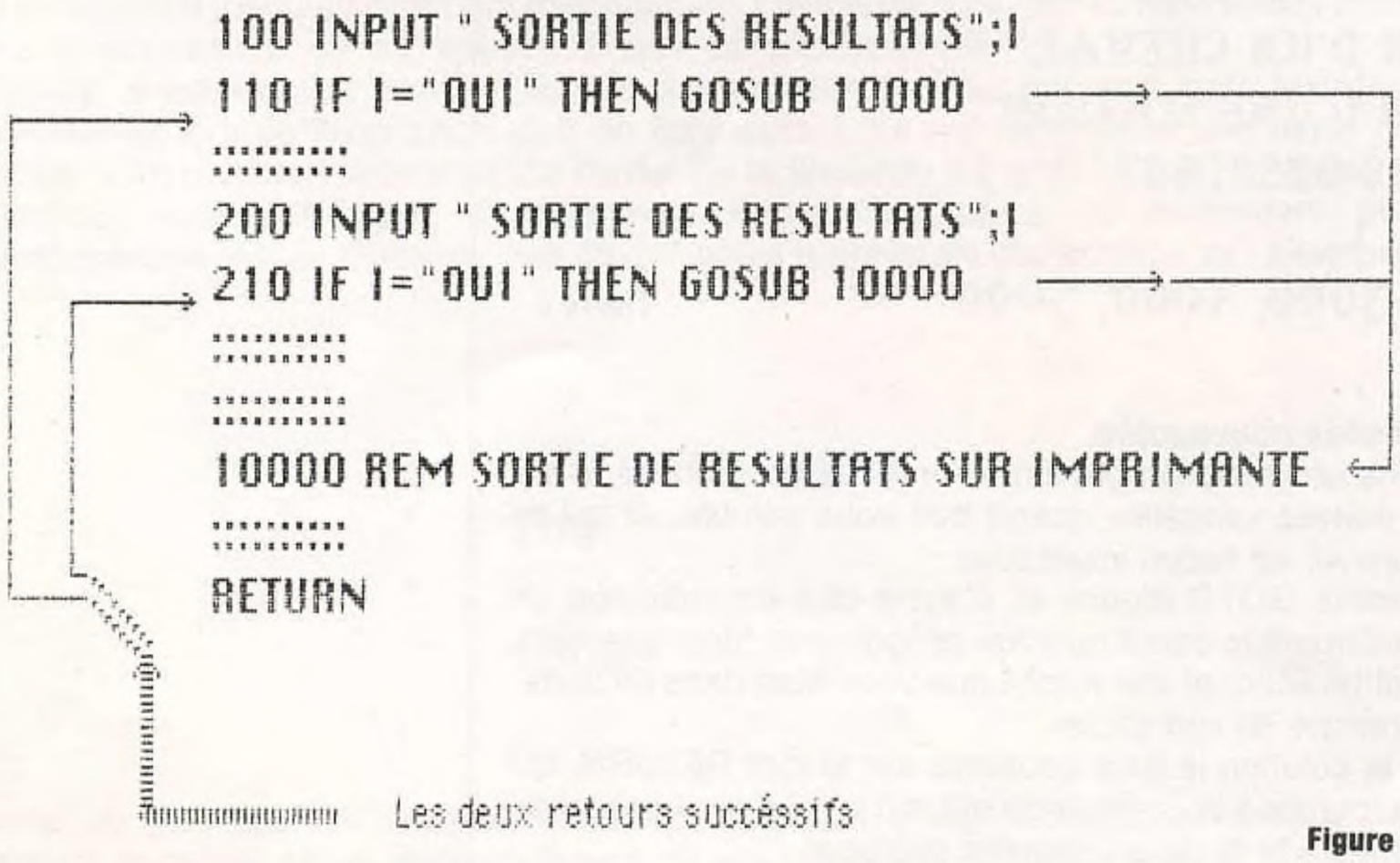


Figure 5

La commande POP est d'utilisation assez rare. Elle permet de sortir d'un sous-programme, sans utiliser de RETURN, donc sans retourner à la ligne appelante. Si le programme rencontre POP, il considère qu'il est sorti du sous-programme, et continue comme si de rien n'était, en séquence. Cela sert à traiter des cas très particuliers dans un sous programme (figure 6). Le plus souvent, cette commande, qui ne fait rien pour clarifier un programme, peut être évitée. Cela demande parfois de travailler sur le programme, mais au bout du compte, représente toujours un gain.

```
10 GOSUB 100
100 REM DEBUT DE SOUS-PROGRAMME
110 REM **************************
120 REM CE SOUS-PROGRAMME TRAITE
130 REM LES SORTIES SUR IMPRIMANTE
140 REM **************************
150 ..........
160 IF PB$="OUI" THEN POP: GOTO 1000
170 ..........
200 RETURN
.................
1000 REM TRAITEMENT DES PROBLEMES EN TOUS GENRES
.................
```

Figure 6

3. Comparaisons entre les commandes GOTO et GOSUB

On peut dire que la commande GOTO permet d'installer dans le programme un sympathique désordre, alors que la création de sous-programme va à l'encontre de ce désordre, donnant une unité à certaines parties du programme. Ces sous-programmes peuvent ainsi être utilisés aussi souvent que nécessaire, à la demande. Mais certains inconvénients se retrouvent, comme la recherche de la ligne du début du sous-programme, ce qui consomme du temps machine (comme nous l'avons vu avec la figure 1).

4. Application

Il nous a semblé intéressant de poursuivre l'étude du programme débuté avec le cours précédent ; il sera amélioré petit à petit, mais selon les idées que nous pourrons avoir. Aussi, pour que vos idées apparaissent également, n'hesitez pas à nous écrire. Toutes vos suggestions seront étudiées et intégrées, car nous ne détenons pas le monopole des bonnes idées, hélas, la concurrence est rude !!!
Reprenons donc le programme répertoire vu la dernière fois, vous allez voir que nous allons le modifier nettement.

```
10 REM  LE VRAI PROGRAMME DEBUTE EN 10000
20 GOSUB 10000 GOTO 10000
30 REM AVANT LA LIGNE 10000, LES SOUS-PROGRAMMES
100 REM SOUS-PROGRAMME DE MENU PRINCIPAL
101 REM ********************************
110 PRINT "TYPE DE TRAVAIL A EFFECTUER"
120 PRINT "      1: AJOUT D'UNE REFERENCE"
130 PRINT "      2: EFFACEMENT D'UNE REFERENCE"
```

```
140 PRINT "     3: RECHERCHE D'UNE REFERENCE"
150 INPUT "  -------> VOTRE CHOIX : ";CHOIX
200 RETURN
210 REM FIN DU SOUS-PROGRAMME
220 REM ************************
300 REM SOUS-PROGRAMME D'AJOUT D'UNE REFERENCE
310 REM ************************************
320 INPUT " NOM DU CORRESPONDANT      : ";INFO$(NBINF,1)
330 INPUT " PRENOM DU CORRESPONDANT   : ";INFO$(NBINF,2)
340 INPUT " NUMERO TELEPHONIQUE       : ";INFO$(NBINF,3)
350 NBINF = NBINF + 1
360 REM FIN DE CE SOUS-PROGRAMME
370 REM ************************
400 REM RETRAIT D'UNE REFERENCE
410 REM ************************
420 INPUT "NOM DU CORRESPONDANT  : "; NOM$
430 GOSUB 1000

440 IF J=0 THEN PRINT "NOM SANS REFERENCE ": GOTO 420
450 IF J=1 THEN FOR I=1 TO 3 : INFO$(STOCK(1),1)=" ":NEXT:GOTO 480
460 IF J>1 THEN INPUT "PRENOM DU CORRESPONDANT  : ";PR$
470 FOR I=1 TO J : IF INFO$(STOCK(I),2)=PR$ THEN
        FOR K=1 TO 3 : INFO$(STOCK(I),K)=" " : NEXT K : Next I
480 RETURN
490 REM FIN DE CE SOUS-PROGRAMME
491 REM ************************
500 REM RECHERCHE D'UN NUMERO
510 REM ********************
520 INPUT "NOM DU CORRESPONDANT  : "; NOM$
530 GOSUB 1000
540 PRINT " NOMBRE DE REFERENCES OBTENUES  : ";J
550 FOR I=1 TO J
560 FOR K= 1 TO 3: PRINT INFO$(STOCK(I),K);: PRINT: NEXT K
570 NEXT I
580 RETURN
590 REM FIN DE CE SOUS-PROGRAMME
600 REM ************************
1000 REM SOUS-PROGRAMME DE COMPTAGE
1010 REM ***************************
1030 J=0: FOR I=1 TO NBINF: IF INFO$(I,1)=NOM$ THEN J=J+1:
        STOCK(J)=I : NEXT
1040 RETURN
1050 REM FIN DE CE SOUS-PROGRAMME
1060 REM ************************
10000 REM DEBUT DU PROGRAMME "PRINCIPAL"
10010 REM ******************************
```

```
10020 DIM INFO$(20,3)
10030 DIM STOCK(10)
10040 NBINF =01
10100 REM CHOIX DU TRAVAIL
10110 REM ****************
10120 GOSUB 100
10200 REM ENVOI A L'ENDROIT SOUHAITE
10210 REM ************************
10220 ON CHOIX GOSUB 300, 400, 500
10300 PRINT "FIN DU TRAITEMENT ? "
10310 INPUT " OUI / NON : "; ENCORE$
10320 IF ENCORE$ =" NON " THEN END
10330 GOTO 10000 10100
```

Ainsi que vous pouvez le constater, ce programme qui est exactement le même que celui présenté le mois dernier, est bien plus agréable à lire, et à modifier. Par contre, nous n'avons pas fait évoluer le contenu du programme lui-même, mais ce sera chose faite dans les mois qui viennent.

5. Courrier

Un lecteur nous demande de parler des passages de paramètres dans des sous-programmes en Basic. Il faut savoir que très peu de Basic le permettent, mais que ces possibilités sont offertes par exemple sur le Macintosh ou le Sinclair QL. Dans la suite, il sera fait principalement référence au Macintosh, sur lequel tourne le Basic Microsoft 2.0, équivalent au Basic Microsoft 5.0 des ordinateurs compatibles avec l'IBM-PC. Ne vous étonnez pas dans la suite, ces Basic autorisent la non-numérotation des lignes, sauf si c'est indispensable, pour savoir où un GITI doit aller par exemple. Et même dans ce cas, plutôt qu'un numéro, vous pouvez donner un nom à cette ligne, ce qui améliore la lisibilité du programme. Sur ce type de Basic, vous pouvez créer des sous-programmes, qui forment des entités parfaitement définies, débutant par la commande :

```
SUB "nom du sous-programme"
```

et se terminant avec la commande

```
END SUB
```

Cela signifie que si le programme rencontre le mot SUB, il recherche la première ligne se trouvant après la commande END SUB suivant automatiquement (figure c.1).

```
.............
.............
SUB SOUSPROG

     .....
     .....
END SUB
.............
.............
```

Figure c.1

```
A=10
B=12
CALL SOUSPROG(B)
```

```
SUB SOUSPROG(X)
    Y=1
    PRINT X,Y                    donne à l'écran la sortie 12   1
END SUB
```

Figure c.2

De plus, une importante nouveauté est à souligner. Si vous le souhaitez, vous pouvez déclarer des variables avec l'en-tête de votre sous-programme, et seules les variables que vous passerez au moment de l'appel seront modifiées (figure c.2). Cela implique que si vous utilisez une variable I (par exemple) dans le programme, et également I dans le sous-programme, sans que cette variable I soit dans l'en-tête de déclaration, pour l'ordinateur, il s'agit d'une autre variable, sans relation avec la première (figure c.3).

```
I=10
PRINT I
                                 donne à l'écran la sortie 10
CALL SOUSPROG1(A,B)
CALL SOUSPROG2

PRINT I                          donne à l'écran la sortie 10

SUB SOUSPROG1(N,J)
    PRINT I                      donne à l'écran la sortie 0
    I=8
END SUB
SUB SOUSPROG2
    PRINT I                      donne à l'écran la sortie 0
    I=5
END SUB
```

Figure c.3

C'est très commode car, pour les boucles par exemple, cela permet d'utiliser toujours les mêmes indices I, J, K, etc. Naturellement, ce qui est vrai entre le programme et le sous-programme est vrai entre deux sous-programmes (figure c.3). L'appel d'un tel sous-programme se fait par :

```
CALL "nom du sous-programme " [liste des paramètres à passer]
```

Les crochets signifient que cette ligne de paramètre peut être vide, car elle est optionnelle (figure c.4).

```
CALL SOUSPROG (A,B,TABLEAU)
::::
::::
SUB SOUSPROG (X,Y,T(2,3))
    X=Y
    Y=T(1,1)
END SUB
```

Figure c.4

Lorsque les variables que vous passez ainsi sont modifiées par le sous-programme, elles reviennent avec leurs nouvelles valeurs dans le programme appelant, sauf si vous les entourez de parenthèses lors de l'appel (figure c.5). Dans ce cas, A sera restituée avec sa valeur originale, même si le sous-programme modifie sa valeur. Ceci est très performant pour ne pas multiplier les variables à l'infini. A vous d'en tirer profit.

```
CALL SOUSPROG ((A),B,TABLEAU)

      Ici, A reviendra avec son ancienne valeur,
      même si son contenu a été modifié
      par le sous-programme, contrairement à B
      et au tableau TABLEAU.
```

Figure c.5

Une autre commande très importante : SHARED qui permet de récupérer dans un sous-programme des variables du programme, ou d'autres sous-programmes. Ceci autorise le passage de variables de la même manière que par la liste de la figure c.4, mais évite de réécrire (figure c.6) la liste pour chaque instruction d'appel de la procédure (sous-programme). Si vous rajoutez STATIC, les variables modifiées dans le sous-programme seront restituées avec les valeurs d'origine (figure c.7).

```
SUB SOUSPROG
    SHARED A,B,TABLEAU()
    ....
END SUB
```

et l'appel se fera par :

```
CALL SOUSPROG
```

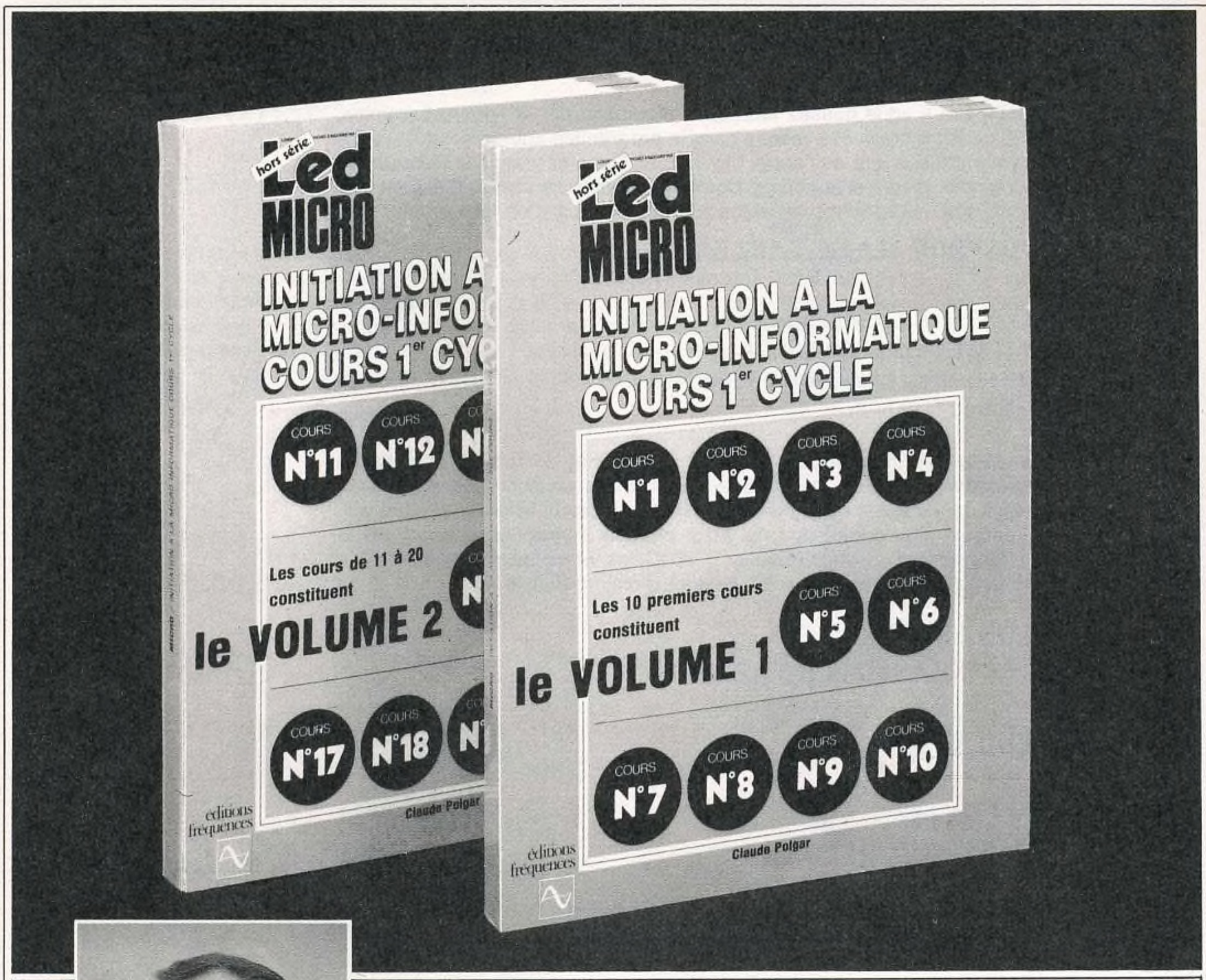
Figure c.6

```
SUB SOUSPROG  STATIC
    SHARED A,B,TABLEAU()
    ....
END SUB
```

Figure c.7

Naturellement, tout ce qui a été dit pour les autres Basic est également vrai ici (GOSUB, ON GOSUB, RETURN, POP sont disponibles), donc un usage courant peut en être fait. Mais il est tellement plus commode de nommer un sous-programme CERCLE si il sert à tracer des cercles.
Nous n'envisageons pas de répondre toujours aussi longuement dans le cadre du courrier des lecteurs.
Néanmoins, lorsque le sujet sera aussi intéressant que celui traité ici, il fera en général l'objet d'un cours complet, ce qui permettra des développements détaillés. En effet, ce lecteur nous aura permis de traiter en détail les nuances des nouveaux Basic disponibles sur le marché, dans le domaine des sous-programmes qui étaient notre sujet du jour.
Cela nous a donc permis un cours vraiment complet, sur un point qui est rarement détaillé dans la littérature française.

Le mois prochain, nous vous proposerons une étude exhaustive des commandes Basic permettant de travailler avec des chaînes de caractères, et la façon de recréer celles qui n'existent pas sur tous les Basic.
Ceci permet de réaliser des programmes très beaux sur le plan présentation, et d'écrire des «jeux de mots» qui peuvent être vraiment distrayants.



Claude Polgar est né en 1926 à Paris. Ingénieur de l'Ecole Centrale de Paris, il fut ingénieur d'études chez Kodak-Pathé, chez Renault-Machine-Outils et aux machines Bull puis chef de département aux engins Matra. Parallèlement à cette carrière classique d'ingénieur, Claude Polgar a poursuivi des recherches personnelles en créant en 1954 le matériel Prototypia (qui fut le premier «Meccano» de micro-robotique) et en 1982 le logiciel d'habillement Alamod (qui permet de réaliser des patrons personnalisés). Claude Polgar se consacre actuellement à l'enseignement des techniques modernes.
Les Editions Fréquences ont publié son cours de programmation dans la revue Led-Micro.

2 volumes (près de 500 pages - format 21 × 27) représentant le récapitulatif de 2 ans des cours progressifs de Claude Polgar

Un 3e volume en préparation prévu fin octobre 85

DE NOMBREUX ADDITIFS

Que de changements depuis la sortie du numéro 1 de LED-MICRO !
Il n'est plus possible d'ignorer :
— le MS-DOS (le système d'exploitation de l'IBM PC)
— les Mémoires à Bulles
— le Compact-Disc
— le développement du Minitel et des réseaux de télématique amateur
— les notions de base de l'Intelligence artificielle (ce qu'est PROLOG etc...)
— l'emploi des calculettes aux examens.
J'ai profité de cette réédition pour ajouter des exercices, mieux présenter certains thèmes, donner aux professeurs le moyen de préparer des disquettes autochargeables.
Que voulez-vous ? C'est ma nature !
C. POLGAR

le cours d'initiation à la micro-informatique le plus complet

non, on ne s'initie pas à la micro-informatique et au basic en 5 leçons ou en 3 semaines !

Le mythe de l'informatique loisir facile s'est envolé , accéder à la programmation relève d'une pédagogie sérieuse et progressive, c'est le pari gagné que fit Led-Micro à une époque où fleurissait chaque jour un nouvel ouvrage-miracle.

Parmi les centaines de lettres reçues, nous nous permettons de citer 3 d'entre elles, elles permettent de situer comment, en général, a été perçu et apprécié ce cours.

J'enseigne les mathématiques dans une Université de Sciences Humaines et j'ai été amenée, alors que je n'avais moi-même reçu aucune formation à la micro-informatique, à initier des étudiants de 1re année de Mathématiques et Sciences Sociales (MASS) à la programmation en S-BASIC (sur Goupil-3), dans le but de faire avec eux de l'analyse numérique élémentaire. Ce que j'ai fait, tant bien que mal, cette année, en collaboration avec deux autres collègues. Nous sommes conscientes d'avoir commis un certain nombre d'erreurs pédagogiques et nous souhaitons tenter d'y remédier l'an prochain.
J'ai découvert votre revue tout récemment, alors que j'arrivais quasiment au bout de mon enseignement. J'ai été très sensible à votre démarche pédagogique et je me sens personnellement tout à fait en accord avec votre manière de procéder. Je me suis procurée l'ensemble des nos de la revue et me permettrai de puiser dans votre cours certains exemples ou certaines façons de présenter les choses l'an prochain. Donc merci à vous... *C.L. St Cloud, le 22/5/85*

J'ai déjà essayé, à deux reprises au moins, antérieurement, de me familiariser vraiment avec le BASIC sans grand résultat, je l'avoue.
La méthode que vous mettez en œuvre dans «Led-Micro» — me conduira-t-elle au but recherché, je n'en sais rien encore — a du moins le mérite d'être sympathique et agréable à suivre. Ma seule ambition étant d'utiliser les micros comme distraction intellectuelle (je suis retraité), j'espère ainsi y parvenir.
Merci, donc, de votre aide et continuez à nous faire avancer progressivement et sûrement.
Docteur Y.C. Sees, le 19/2/84

Je viens de découvrir votre magazine ce matin dans un kiosque, cet après-midi je vous commande les 18 premiers numéros.
Je suis très emballé par vos cours, que je trouve très bien faits.
Je suis un «vrai» débutant, je possède un ZX81 que j'ai du mal à faire tourner, par manque d'information, grâce à vos cours je pense que j'y arriverais. Je possède pas mal de bouquins sur la question mais aucun n'explique aussi clairement que vous. *A.A. Marseille, le 17/4/85*

en vente chez votre libraire ou aux Editions Fréquences (collection pédagogique).

Initiation à la micro-informatique C. Polgar

En vente chez votre libraire ou aux Editions Fréquences 1, bd Ney 75018 Paris. Tél. : (1) 607.01.97

Je désire recevoir le tome 1 ☐ **140 F** (130 F + 10 F de frais de port)
le tome 2 ☐ **140 F** (130 F + 10 F de frais de port)
les deux tomes ☐ **280 F** (260 F + 20 F de frais de port)

Je joins mon règlement à la commande :
chèque bancaire ☐ mandat ☐ C.C.P. ☐

Nom ..... Prénom .....
Adresse .....
Code postal ..... Localité .....

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Nous avons étudié dans le précédent article les codages internes des différentes variables sous BASIC. Nous avons ainsi pu mieux comprendre l'efficacité de chaque type de variable en fonction des besoins. L'idée originale qui nous a conduit à écrire cet article était de vous apporter les moyens d'améliorer l'efficacité de vos programmes en profitant d'une part des remarques faites pour tout ce qui concerne l'utilisation des variables et d'autre part en utilisant des astuces vous permettant d'accélérer les temps d'exécution.
Malheureusement ou heureusement, à vous de juger, nous nous sommes rendus compte qu'il fallait tout vous dire !
Plus précisément, pour que vous puissiez dans un premier temps comprendre nos exemples et les astuces que nous vous présentons, et par la suite en profiter dans le développement de vos programmes en les adaptant à vos machines, il nous a fallu aller un peu plus loin que prévu. Rassurez-vous tout de même, ce n'est pas bien méchant, en tout cas sûrement moins que si l'on vous avait caché les raisons profondes des optimisations que l'on vous présente.

COURS N° 2

Les Benchmarks

PLAN DU COURS

1. Bien programmer
   1.1. Faire de beaux programmes
   1.2. Faire des programmes efficaces
2. Savoir tester les performances de son ordinateur
   2.1. Présentation des Benchmarks
   2.2. Les Benchmarks à l'usage
3. Optimisation sous BASIC
   3.1. Etude simplifiée du fonctionnement d'un BASIC interprété
   3.2. Amélioration des temps d'exécution
   3.3. Les périphériques sous BASIC
4. Comment aller plus vite
   4.1. Solutions logicielles
   4.2. Solutions matérielles

NOTES PERSONNELLES

1. BIEN PROGRAMMER

La «belle programmation» est une vue de l'esprit qui peut exister sur une machine utopique dont les caractéristiques fondamentales (mémoire, temps de calcul, etc) seraient optimales. Nous allons donc mettre en parallèle ce qui est beau et ce qui est bien.

1.1. Faire de beaux programmes

Si l'on voulait faire quelque chose de très beau, il faudrait prendre à la lettre tous les préceptes suivants :

– Le rapport entre les lignes de commentaires (REM...) et les lignes d'instructions doit être de 3 pour 1. En effet, c'est la longueur moyenne des explications nécessaires pour quelqu'un ne connaissant rien en informatique. Toute personne ayant frappé plus de 10 lignes de programme s'empressera d'oublier cette règle presque volontairement.

– La programmation doit être très structurée. Qu'est-ce qu'on entend par là ? Eh bien, pour chaque «module» du programme, on fait appel à un sous-programme. Oui, mais qu'est-ce qu'un module ? Un module est une partie du programme homogène dont l'ensemble des instructions permet d'atteindre un but (impression, lecture, rangement, calcul d'un groupe de valeurs, d'un tableau) ou plus simplement tout groupe d'instructions qui pourrait être un mini-programme (10 instructions en moyenne) à lui tout seul.

– Maintenant que l'on connait ces modules, il ne reste plus dans le «programme principal» qu'une suite de commentaires et d'appels de sous-programmes (GOSUB...). Mais lorsqu'on appelle ces modules, il convient de conserver intactes certaines variables communes à plusieurs modules, aussi, puisque l'on n'est pas limité en mémoire, on commence tout sous-programme par la création de nouvelles variables (que l'on qualifie de locales) dans lesquelles on implante les valeurs importées par les variables communes.

– Il convient de rendre les programmes «souples» c'est-à-dire que si un de nos programmes établit par exemple, le PGCD de 57 et 21, qu'il soit capable plus tard de calculer aussi celui de deux autres valeurs quelconques. Mais, si vous faites de la programmation, vous avez assurément compris que c'est le nerf de l'informatique.

– Le programme doit être le plus transportable possible. Nous entendons par là que le programme peut être implanté sur un autre ordinateur avec un minimum de modifications. Mais ceci est un des serpents de mer les plus vieux de l'histoire pourtant courte de l'informatique, aussi bienvenue au MSX.

– Un bon programme ne doit pas avoir de «verrues». Une verrue est une partie du programme qui de très courte, à force de modifications arrive à être la majeure partie du programme.

– C'est une très bonne précaution de prévoir les fautes de frappes et autres déviations à l'intérieur de boucles en faisant des tests vérifiant si le caractère est un chiffre ou si la variable n'est pas négative (si elle représente des kilomètres c'est toujours gênant) quand elle ne doit pas l'être. En résumé, qu'il y ait cohérence entre les valeurs des variables et ce qu'elles représentent.

1.2. Faire des programmes efficaces

Vous avez vu de vous-mêmes que certaines règles resteront dans tous les programmes au stade de «vœux pieux», aussi si dans la mesure du possible il convient de les respecter, voici quelques règles plus pragmatiques.

– Quand vous avez plusieurs tests, commencez toujours par celui qui vous paraît comme devant être vérifié le plus souvent : dans le programme suivant, on cherche à déterminer le nombre de jours d'un mois donné :

- La variable B détermine si l'année est bissextile ou non

```
10  INPUT "MOIS ";M;" ANNEE ";A%
20 B = A% - (A% / 4) * 4
30  IF (M = 2) AND (B = 0) THEN J = 29: GOTO 100
40  IF (M = 2) AND (B <  > 0) THEN J = 28: GOTO 100
50  IF (M = 4) OR (M = 6) OR (M = 9) OR (M = 11) THEN J = 30: GOTO 100
60 J = 31
100  END
```

Fig. 1.2.1.

- Dans le premier programme en moyenne
  1 fois sur 48 il effectue 1 test
  3 fois sur 48 il effectue 2 tests
  44 fois sur 48 il effectue 3 tests
  ce qui donne une moyenne de 2,896 tests.

```
10  INPUT "MOIS ";M;" ANNEE ";A%
20  IF (M = 1) OR (M = 3) OR (M = 5) OR (M = 7) OR (M = 8) OR (M = 10) OR
     (M = 12) THEN J = 31: GOTO 100
30  IF (M = 4) OR (M = 6) OR (M = 9) OR (M = 11) THEN J = 30: GOTO 100
40 B = A% - (A% / 4) * 4
50  IF (M = 2) AND (B <  > 0) THEN J = 28: GOTO 100
60 J = 29
100  END
```

Fig. 1.2.2.

- Dans le second programme
  28 fois sur 48 il effectue 1 test
  16 fois sur 48 il effectue 2 tests
  4 fois sur 48 il effectue 3 tests
  ce qui donne une moyenne de 1,5 tests
- Mais le 1er test comporte 7 éléments alors que sa négation n'en comporte que 5, cela nous conduit au 3e programme.
  De même, la négation du 2e test ne comprend qu'un élément au lieu de 4.

```
10  INPUT "MOIS ";M;" ANNEE ";A%
20  IF (M = 2) OR (M = 4) OR (M = 6))R(M = 9) OR (M = 11) THEN 30
21 J = 31: GOTO 100
30  IF M = 2 THEN 40
31 J = 30: GOTO 100
40 B = A% - (A% / 4) * 4
50  IF B = 0 THEN 60
51 J = 28: GOTO 100
60 J = 29
100  END
```

Fig. 1.2.3.

- Le 4e programme correspond à ce qu'il est meilleur encore de faire comme nous pensons que beaucoup de nos lecteurs nous l'auraient proposé.

```
10  INPUT "MOIS ";M;" ANNEE ";A%
20  ON M GOTO 21,40,21,31,21,31,21,31,31,21,31,21,31
21 J = 31: GOTO 100
31 J = 30: GOTO 100
40 B = A% - (A% / 4) * 4
50  IF B = 0 THEN 60
51 J = 28: GOTO 100
60 J = 29
100  END
```

Fig. 1.2.4.

- Cette réorganisation nous permet dès le 2e programme de vérifier si l'année est bissextile uniquement lorsque cela est nécessaire.

– Pour récupérer de la place mémoire puisque nous ne sommes plus dans une machine utopique, il est aisé de modérer l'adjonction des commentaires en les limitant au strict minimum.
Dans la même optique, il existe un exercice particulièrement périlleux : déterminer la durée de vie d'une variable. La durée de vie est la longueur du programme pendant laquelle on se sert de cette variable. Au-delà, plutôt que de définir une autre variable qui coûterait un emplacement mémoire de plus, on assigne à l'ancienne variable la valeur de la «nouvelle», une sorte de «réincarnation». Rien que la description de l'opération doit vous faire dresser les cheveux sur la tête mais pour qui maîtrise son programme, rien n'est plus agréable, voire distrayant.

– Voici la règle des 30/70 (certains l'appellent des 20/80, vous allez voir pourquoi). L'expérience prouve que dans 30 % du programme en «taille-instructions» se déroule 70 % du temps d'exécution (les optimistes ou les pessimistes, au choix, disent donc 20/80). Une fois cette constatation faite, on peut en déduire quelques préceptes : on fera porter l'essentiel d'effort sur l'optimisation des 30 % en question.

– Cette règle est récursive, c'est-à-dire que dans ces 30 %, il y a 30 % (donc 9 % au total) qui emploient 70 % (donc 49 % du temps global).

– Il convient donc d'éliminer de l'intérieur des boucles toutes les instructions qui n'y sont pas modifiées (très souvent les tests n'en font pas partie) et de bien vérifier que les bornes des boucles sont les plus subtiles. Voici un exemple de boucles subtiles pour une table de Pythagore 100 × 100. Plutôt que le 1er programme, utilisons la symétrie et calculons uniquement la moitié des éléments :

```
5  DIM T(100,100)
10  PRINT "DEBUT"
20  FOR I = 1 TO 100
30  FOR J = 1 TO 100
40 T(I,J) = I * J
50  NEXT J: NEXT I
60  PRINT "FIN"
```

Fig. 1.2.5.

```
5  DIM T(100,100)
10  PRINT "DEBUT"
20  FOR I = 1 TO 100
30  FOR J = I TO 100
40 T(I,J) = I * J
50  NEXT J: NEXT I
60  PRINT "FIN"
```

Fig. 1.2.6.

– après avoir développé toutes ces «astuces» permettant d'améliorer un programme, il faut néanmoins pondérer ceci par la remarque suivante qui est une forme de crédo :

«Le temps gagné n'est-il pas perdu ?»

En effet, il faut établir la péréquation entre le temps dépensé par le programmeur pour améliorer son programme et le gain de temps à l'exécution ou en mémoire par le programme lui-même. Généralement, la constatation obtenue vient renforcer l'explication selon laquelle les grands programmes ne sont que rarement beaux. Aussi, il est bien plus utile de savoir ce qui est rapide avant de commencer à programmer plutôt que de le découvrir en cours.

2. SAVOIR TESTER LES PERFORMANCES DE SON ORDINATEUR

2.1. Présentation des Benchmarks

Que signifie ce mot barbare ? C'est tout simplement la traduction du jeu d'essais. En effet, il est utile de savoir quel est le temps nécessaire à votre ordinateur pour exécuter telle ou telle instruction. Pour cela, on utilise ce que le bon sens nous souffle immédiatement. Nous ne sommes pas capables de mesurer le temps mis pour une seule instruction, cela va trop vite, alors mesurons le temps nécessaire pour faire 10 000 fois cette même instruction. Si votre machine refuse pour certains des jeux que nous allons vous proposer, reprenez tout avec 2 000 ou 1 000.

2.2. Les Benchmarks à l'usage

2.2.1. Comme pour une pesée, il faut d'abord mettre la balance en équilibre à vide. Donc établissons le temps nécessaire pour la seule boucle. Déjà, nous retrouvons ce que nous avions vu la fois précédente, il y a plusieurs boucles. En voici une liste, la plus exhaustive possible.

```
10  PRINT "DEBUT"
20  FOR I% = 1 TO 10000
30  REM  OPERATION SUR I%
40  NEXT I%
60  PRINT "FIN"
```
Fig. 2.2.1.1.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  REM   OPERATION SUR I
40  NEXT
60  PRINT "FIN"
```
Fig. 2.2.1.2.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 100
30  FOR J = 1 TO 100
40  REM  OPERATION SUR (I,J)
50  NEXT J: NEXT I
60  PRINT "FIN"
```
Fig. 2.2.1.3.

```
10  PRINT "DEBUT"
20 I = 1
30  REM  OPERATION SUR I
40 I = I + 1
50  IF I < 10000 THEN 20
60  PRINT "FIN"
```
Fig. 2.2.1.4.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 9999 STEP 2
30  REM  OPERATION SUR I
40  REM  OPERATION SUR I+1
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.1.5.

Vous comprenez aisément que le temps à mesurer est entre l'apparition de «début» et celle de «fin». Il convient de mettre maintenant des instructions à l'intérieur pour calculer le temps nécessaire pour les exécuter.

2.2.2. Les opérations arithmétiques

Commençons par comparer l'addition et la multiplication, aucun résultat n'est définitif pour l'ensemble des ordinateurs, aussi nous nous contenterons de vous donner les programmes.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30 A = I + I
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.2.1.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30 A = 2 * I
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.2.2.

Maintenant, établissons un parallèle entre la puissance et la multiplication en considérant que certains interprètent la puissance d'entiers par la multiplication n fois.

```
10  PRINT "DÉBUT"
20  FOR I = 1 TO 10000
30  REM  OPERATION SUR I
40 A = I * I * I
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.2.3.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  REM  OPERATION SUR I
40 A = I ^ 3
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.2.4.

Ici on détermine la différence entre les opérations sur des variables et celles sur des valeurs numériques.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  REM  OPERATION SUR I
40 A = I / I * I + I - I
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.2.5.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  REM  OPERATION SUR I
40 A = I / 2 * 3 + 4 - 5
50  NEXT I
60  PRINT "FIN"
```
Fig. 2.2.2.6.

Cet exemple est plus sophistiqué, la première programmation vient immédiatement à l'esprit, la seconde est plus fine si l'on n'est point limité en place mémoire, mais même dans ce cas, il n'est pas joué d'avance que l'on gagne du temps.

```
10  PRINT "DEBUT"
15 A = 10:B = 3:C = 2:D = 1:E = 4
     :F = 3
20  FOR I = 1 TO 10000
30 Z = ((A - 2 * B) * C + D) / ((
     A - 2 * B) * E + F)
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.2.7.

```
10  PRINT "DEBUT"
15 A = 10:B = 3:C = 2:D = 1:E = 4
     :F = 3
20  FOR I = 1 TO 10000
30 X = A - B - B
40 Z = (X * C + D) / (X * E + F)
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.2.8.

2.2.3. Les sauts et tests

Comparons maintenant les sauts directs et les appels de sous-programmes.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  GOSUB 200
50  NEXT I
60  PRINT "FIN"
200 A = I + I
210  RETURN
```

Fig. 2.2.3.1.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  GOTO 35
35 A = I + I
40  GOTO 50
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.3.2.

```
10  FOR A% = 1 TO 400
11  FOR M = 1 TO 12
20  ON M GOTO 21,40,21,31,21,31,21,31,31,21,31,21,31
21 J = 31: GOTO 100
31 J = 30: GOTO 100
40 B = A% - (A% / 4) * 4
50  IF B = 0 THEN 60
51 J = 28: GOTO 100
60 J = 29
100  NEXT M: NEXT A%
110  END
```

Fig. 2.2.3.3.

La comparaison entre les tests peut être menée avec le programme de jours du début du cours en remplaçant l'instruction 10 et en ajoutant une instruction à la fin. Voici le programme pour le dernier exemple (fig. 1.2.4.), pour les autres il suffit de les intégrer à leurs adresses entre 20 et 70 sans changer les adresses.

2.2.4. Les entrées/sorties

Les 3 programmes présents vous donnent l'occasion de comparer quelques politiques d'impression mais les entrées/sorties sont généralement imposées par les résultats que vous devez obtenir.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  PRINT I
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.4.1.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  PRINT I,
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.4.2.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30  PRINT I;" ";
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.4.3.

2.2.5. Les fonctions prédéfinies

Nous vous donnons des exemples pour les plus célèbres mais essayez avec toutes celles que vous avez à votre disposition.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30 X =  SIN (I)
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.5.1.

```
10  PRINT "DEBUT"
20  FOR I = 1 TO 10000
30 X =  LOG (I)
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.5.2.

```
10  PRINT "DEBUT"
15  DEF  FN S(I) = I + I
20  FOR I = 1 TO 10000
30 A =  FN S(I)
50  NEXT I
60  PRINT "FIN"
```

Fig. 2.2.5.3.

Le 3ème programme est également à mettre en parallèle avec le programme simple A = I + I

2.3. Le double intérêt des Benchmarks

2.3.1. Connaissance des performances

Avec la batterie de mini-programmes que nous vous avons exposée, vous êtes en mesure d'effectuer un faisceau de tests adapté à vos besoins.
D'une part, vous pouvez aisément en déduire d'autres tests plus élaborés en suivant la philosophie suivante, ajoutons à chaque fois une instruction et mesurons le temps consécutif à cette adjonction. D'autre part, vous avez ainsi des critères de comparaison vous permettant aisément d'établir une hiérarchie entre différents matériels en utilisant le jeu d'essais que vous aurez vous-mêmes choisi.

2.3.2. Compréhension de l'interpréteur

Les résultats comparés de programmes ayant le même but, vous autorisent à avoir des soupçons vis-à-vis de la façon dont votre BASIC est interprété par votre machine. Suivant les résultats, vous déterminerez les points forts et les points faibles de votre ordinateur et donc vous pourrez procéder à des choix «plus pointus» sur la façon de traduire des algorithmes dans le BASIC de votre machine.

3. OPTIMISATION SOUS BASIC

3.1. Etude simplifiée du fonctionnement d'un BASIC interprété

Rassurez-vous, nous n'allons pas vous assommer de précisions techniques et autres détails incompréhensibles (des amateurs que nous sommes). Nous croyons simplement que connaître quelque peu les rouages de votre BASIC ne peut que vous aider dans le choix des solutions que vous adopterez face à des problèmes de programmation.

3.1.1. Principe d'un interpréteur

Bien qu'il ait été déjà décrit dans les premiers cours de Claude Polgar, il est intéressant, ne serait-ce que pour les nouveaux lecteurs d'expliquer ce qu'est un interpréteur, d'autant plus que nous allons d'une part nous limiter au BASIC, et d'autre part garder un esprit pratique.
Votre BASIC se compose de deux parties :
– Une partie Editeur de texte/Analyseur syntaxique
– Une partie Interprétation : du programme en mémoire ou de la demande directe

a) L'éditeur de texte/analyseur syntaxique :
Il gère l'introduction des lignes du programme ou des commandes à exécuter, et effectue la distinction entre les deux. Suivant les basics, l'analyse syntaxique est plus ou moins poussée : elle va d'une simple gestion des numéros de ligne jusqu'à une reconnaissance presque complète des mots réservés, donc détecte les fautes de frappes.

b) L'interpréteur :
Il est chargé de digérer le programme en mémoire et de traduire chaque exécution d'une commande, d'un calcul etc. dans un code directement utilisable par la machine. Cette traduction s'effectue ligne par ligne. S'il lui faut exécuter plusieurs fois la même ligne, il la retraduira systématiquement.

3.1.2. Déroulement et codage d'un programme

Comment l'interpréteur passe d'une ligne à une autre ?

L'interpréteur connaît à partir du codage en mémoire la position de la ligne suivante, il peut donc passer d'une ligne à l'autre.

ATTENTION : il ne connaît pas à l'avance le numéro de la ligne suivante, et pire encore, il ne connait que l'emplacement de la ligne suivante.

Conséquences :
Pour effectuer un GOTO, il doit, à partir de la ligne en cours d'exécution, monter ou descendre dans le programme ligne par ligne. Il doit comparer à chaque fois le numéro de la ligne courante avec celle du GOTO, jusqu'à ce qu'il la trouve ou jusqu'à ce qu'il rencontre un numéro plus grand (ou plus petit suivant le sens du parcours). Donc, plus on effectue des sauts ou des appels à des sous-programmes éloignés de la ligne actuelle, plus on se pénalise en temps de recherche du numéro de ligne.

Lorsqu'il rencontre une structure FOR... NEXT, il réserve une certaine place en mémoire pour décrire l'ensemble des paramètes de la boucle. Il en résulte :
– contrairement au GOTO, il sait où débute la boucle au moment où il rencontre le NEXT, ce qui explique la différence de performance rencontrée entre des boucles structurées et des constructions GOTO + IF.
– la place réservée n'étant pas illimitée, on ne peut imbriquer indéfiniment des boucles, et on obtient un «OUT OF MEMORY ERROR» ou similaire.

Lorsqu'il reconnaît une structure du type DEF FN...(X) = ..., comme pour une structure de boucle, il réserve une place mémoire pour indiquer l'emplacement de la fonction lorsque l'on y fait appel. Là encore, l'accès à la fonction est beaucoup plus rapide qu'un simple GOSUB.

3.1.3. Comment une ligne est-elle codée ?

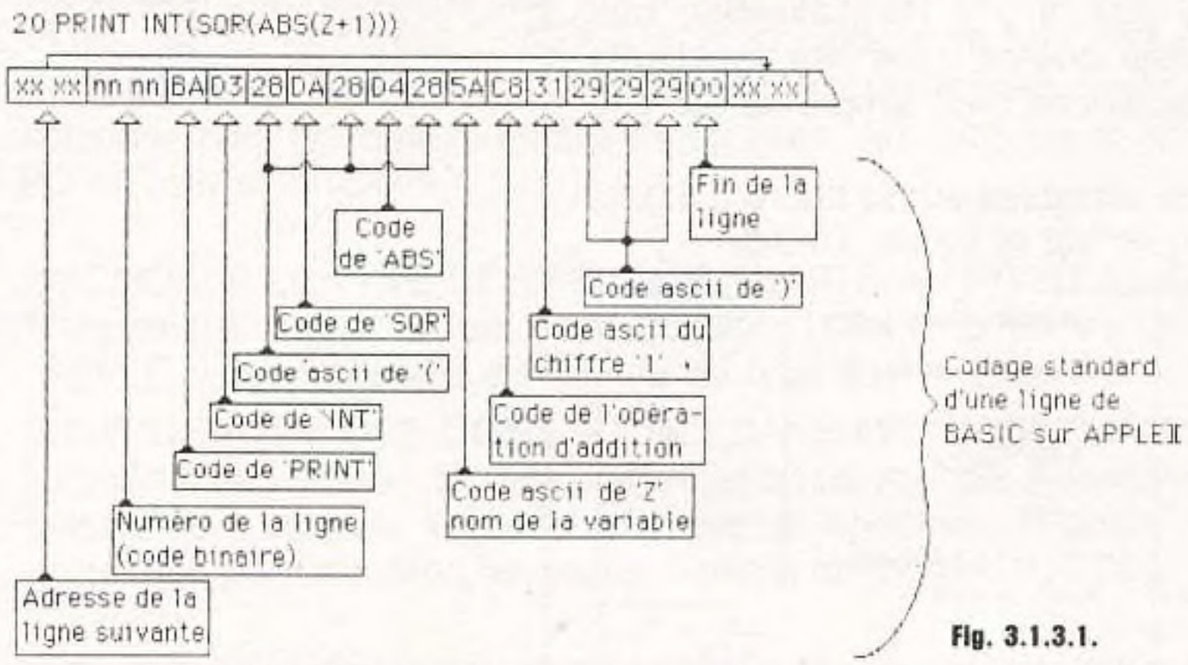
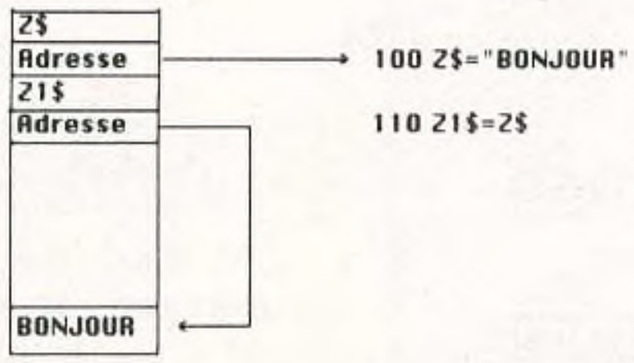


Fig. 3.1.3.1.

MOTS RESERVES : On s'aperçoit qu'à chaque mot réservé, on associe un code unique que l'on appelle en jargon un TOKEN. Ainsi, un mot réservé de plusieurs lettres se résume à un seul octet. Ce codage s'effectue à l'introduction de la ligne (par l'éditeur/analyseur) et facilite le travail de l'interpréteur.

CONSTANTES : Elles sont codées différemment suivant les basics. Dans certains cas, elles restent sous forme ASCII. Dans les autres cas, elles sont optimisées suivant le type et la valeur dans les constantes.

CONSTANTE DE TYPE CHAINES DE CARACTERES : Elles sont codées sous une forme ASCII standard. Sous certains BASICs, on peut gagner un peu de place en mémoire, car le pointeur de la variable adresse directement la zone du programme où se situe la constante.



Implantation mémoire d'une constante chaine de caractères

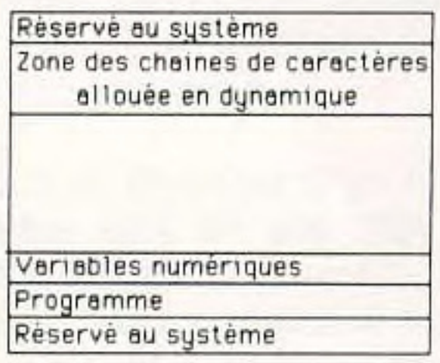
Fig. 3.1.3.2.

Remarque : En fonction du type de BASIC que vous utilisez, vous aurez plutôt intérêt soit à coder directement les constantes, soit à passer par l'intermédiaire d'une autre variable, notamment pour des initialisations de tableaux (pour plus de précision se référer à la fig. 3.1.3.3.).

		Temps d'éxécution	
		Apple II	MAC
Programme 1	10 FOR I= 1 TO 10000 20 ZZ=1234 30 NEXT	57"	14"
Programme 2	10 Z=1234 20 FOR I = 1 TO 10000 30 ZZ=Z 40 NEXT	23"5	12"
Programme 3	10 FOR I=1 TO 10000 20 ZZ=1.0 30 NEXT	57"	11"
Programme 4	10 Z%=1 20 FOR I=1 TO 10000 30 ZZ=Z% 40 NEXT	27"	15"

Fig. 3.1.3.3.

VARIABLES : Il est d'abord nécessaire de préciser leur emplacement en mémoire : en général, elles sont implantées juste derrière le programme, ce qui a pour conséquence principale la destruction du contenu de vos variables après une modification du programme ou ajout d'une ligne. En effet, la taille du programme va changer et entraîne un décalage des positions des variables. Dans ce cas, il y a réinitialisation des variables comme au lancement du «RUN». Il faut préciser que ce n'est pas applicable aux variables de type chaînes de caractère, qu'elles soient de type simple ou de type tableaux.
Vous avez un aperçu d'une implantation classique des variables sur la figure 3.1.3.4., ainsi qu'un exemple de décalage après ajout d'une ligne sur la figure 3.1.3.5.



Implantation mémoire d'un programme et des variables associées

Fig. 3.1.3.4.

Fig. 3.1.3.5.

Le nom de la variable est codé sous forme ASCII standard comme pour une constante de type CHAINE DE CARACTERES. Vous pouvez le constater sur l'exemple de codage fourni à la figure 3.1.3.1.

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Fig. 3.1.3.6. : Tableau hexadécimal des codes ASCII.

CODAGE DES REM : Les commentaires sont codés de façon logique par le «TOKEN» de «REM» suivi du commentaire code en ASCII. Ces lignes prennent donc beaucoup de place et sont néanmoins traitées comme les autres lignes.

INSTRUCTIONS MULTIPLES : Lorsque plusieurs instructions se suivent sur une même ligne grâce à l'utilisation du '':'', l'interpréteur passe directement d'une instruction à l'autre. Par contre, si l'interpréteur doit changer de ligne, il doit effectuer différents calculs concernant la position de la nouvelle ligne, sa longueur, son numéro.

3.2. Amélioration des temps d'exécution

Nous allons résumer les différents moyens d'accélérer le temps d'exécution des programmes. Ces «astuces» sont les conséquences directes de l'étude du fonctionnement de l'interpréteur ainsi que des constatations faites dans l'article du mois de juin concernant le codage interne des variables.

(0) CHOISIR DE BONNES METHODES DE PROGRAMMATION :
C'est une évidence qui ne découle d'aucune étude mais correspond à ce qu'en théorie une bonne programmation impose. Nous avons déjà abordé le sujet au début de l'article.

(1) UTILISATION DES STRUCTURES PREDEFINIES :
Nous avons pu observer que le gain de temps dans l'utilisation de structures du type FOR... NEXT et WHILE... WEND était considérable, et correspondait de plus à une programmation plus rigoureuse. Vous pourrez remarquer que le gain est nettement plus sensible avec les petites machines qu'avec des gros micro-ordinateurs comme l'IBM PC ou le MacINTOSH.

(2) CHOISIR LE TYPE DE VARIABLE ADAPTE AU PROBLEME :
Notamment, utiliser si cela est possible (désolé pour les utilisateurs du BASIC de l'APPLE !) des variables de boucle de type entier (avec l'extension ' % ').

(3) UTILISATION DES CONSTANTES DANS LES PROGRAMMES :
Se méfier des a priori du style, l'affectation de variable à variable est plus rapide que de constante à variable. C'est le cas pour de nombreux BASICs (nous avons pu aussi le constater pour d'autres langages comme le PASCAL UCSD).

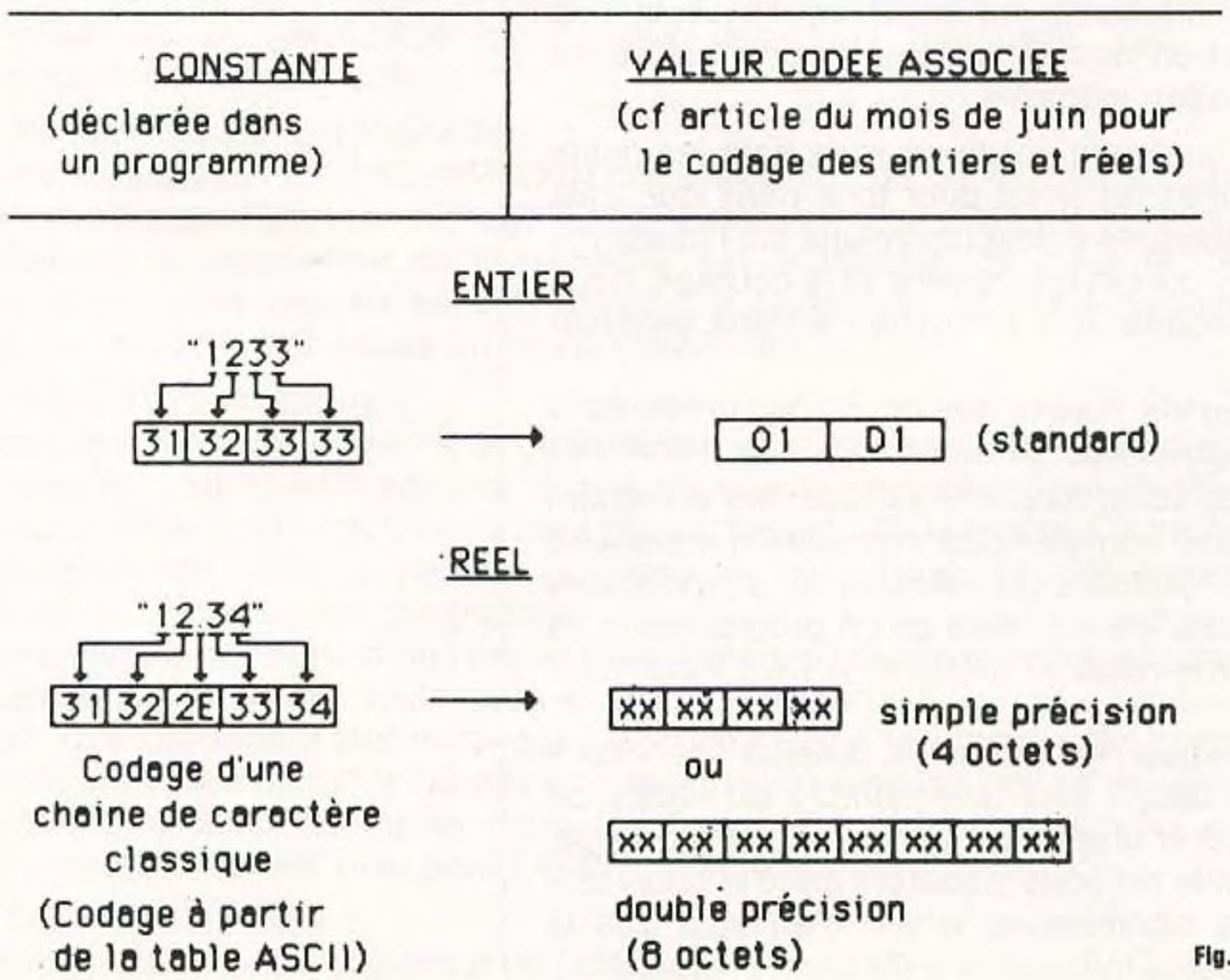


Fig. 3.2.1.

(4) CONVERSIONS DE TYPES :
Les éviter au maximum, surtout lorsqu'elles sont inutiles, par exemple Z = 1.0 est préférable à Z = 1 (cf figure 3.2.2.).

EXEMPLES	CONVERSIONS NECESSAIRES
Z=12.0	ASCII → REEL
Z=12	ASCII → ENTIER → REEL
Z%=1.1	ASCII → REEL → ENTIER
PRINT Z%	ENTIER → ASCII
PRINT Z	REEL → ASCII

Fig. 3.2.2.

(5) DEFINITION DE FONCTIONS DE CALCULS :
Préférer les définitions de fonctions à l'aide de DEF FNA(X) = .. à un simple appel à un sous-programme par GOSUB, on économise la recherche de traitement des numéros de lignes.

(6) COMPACITE DU PROGRAMME :
Supprimer tous les REMs et essayer de diminuer au maximum le nombre de ligne, en n'hésitant pas à mettre plusieurs instructions sur une même ligne. On y gagne à tous les niveaux, d'abord en taille mémoire (on aura ainsi plus de place pour les variables), et sur disque (ou sur cassette), ensuite en vitesse d'exécution et enfin en temps de chargement.

(7) NETTOYAGE DE LA ZONE MEMOIRE :
Il est préférable de forcer régulièrement l'interpréteur BASIC à effectuer ce que l'on appelle en jargon informatique un « GARBAGE COLLECTION ». Cette opération effectue une réorganisation de la zone mémoire réservée à la gestion dynamique des chaînes de caractères qui laisse traîner des bouts de chaînes inutilisées et inutilisables. Le plus simple consiste alors à provoquer le nettoyage par la commande du type Z = FRE(0) ou Z = FRE("") A UN MOMENT NON CRITIQUE !
Précisons que suivant le BASIC dont vous disposez, ce nettoyage se fera soit automatiquement, mais alors pas forcément au bon moment, ou bien pas du tout ce qui risque de poser un problème sérieux d'occupation mémoire.

Cette première liste vous apportera sûrement quelques solutions, mais pour les petits malins qui usent et abusent déjà de ces astuces (et aussi pour tous ceux que cela intéresse), nous allons maintenant présenter quelques éclaircissements sur l'utilisation des périphériques sous BASIC. Enfin pour ceux qui ont les moyens et le courage, nous envisagerons quelques possibilités pour transformer leur ordinateur en une véritable formule 1 du BASIC.

REMARQUE : Nous tenons à vous prévenir que les ficelles que nous vous présentons dans ce numéro ne sont que des solutions générales en réponse à des problèmes classiques, nous ne prétendons nullement que votre machine ira plus vite en tenant compte de chacune de nos suggestions. Aussi, écrivez-nous vos remarques et vos constatations, nous les publierons. Il existe tellement de BASICs et d'ordinateurs différents qu'il est impossible d'envisager toutes les subtilités qu'un programmeur va rencontrer sur sa machine : alors faites comme nous et tester la votre ! Voici deux conseils qui peuvent servir :
– Méfiez-vous des chronomètres tatillons. Lorsque l'on mesure le dixième seconde à partir d'un écho à l'écran signalant la fin du test, il faut tenir compte du temps de réaction du chronométreur qui risque d'engendrer une erreur encore plus importante. Le but est plutôt d'obtenir des résultats qualitatifs qui vous guideront dans le choix des bonnes solutions. Si la différence n'est pas significative, le jeu n'en vaut pas la chandelle.

– En essayant de vous aider à mieux comprendre la mécanique interne de votre BASIC, nous espérons que vous serez très vite capables de maitriser votre machine. C'est à vous de prendre le taureau par les cornes.

3.3. Les périphériques sous Basic

Nous voilà arrivés dans un domaine qui cause bien des soucis aux programmeurs que nous sommes parce que nous le maitrisons peu, voire pas du tout : ils nous apparaissent souvent comme des boîtes qu'il faut relier par des câbles plus ou moins bien détrompés : quelle n'est pas notre surprise lorsque cela marche du premier coup (malheureusement, ce n'est pas souvent le cas !). Nous vous accordons bien volontiers que si nous exprimons un rejet quasi systématique, ce n'est pas par hasard, le «HARDWARE» est d'un abord délicat. Ne soyez pas inquiet, nous ne voulons pas faire concurrence à LED-ROBOT, mais simplement vous apporter là encore quelques notions utiles pour mieux programmer et éviter les erreurs grossières. Nous allons prendre beaucoup de précautions pour d'une part rester dans le domaine du BASIC et d'autre part pour ne pas vous assommer de détails.

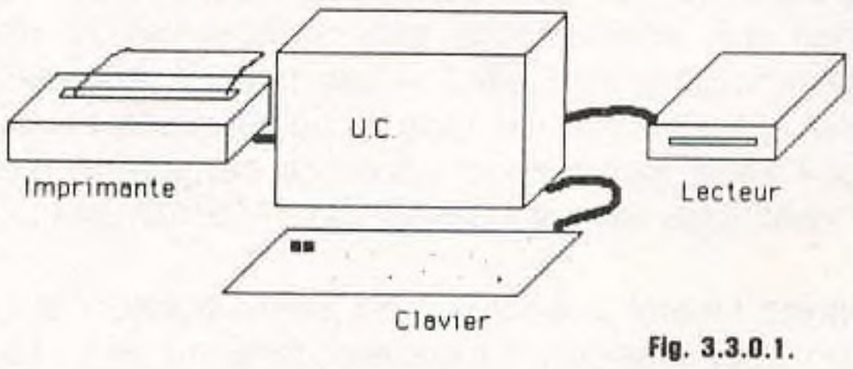


Fig. 3.3.0.1.

Sans détailler les types de liaisons ni les vitesses de transferts, nous allons étudier les quatre organes principaux :
– L'écran
– Le clavier
– L'imprimante
– Les mémoires de masse (cassettes ou disquettes).

3.3.1. L'écran

Suivant le type de matériel utilisé (c'est en règle générale lié au prix de celui-ci), la gestion de la génération de l'image sera soit matérielle soit logicielle, ou éventuellement intermédiaire.

a) Solution totalement logicielle :
C'est le cas pour le SINCLAIR ZX81 qui utilise plus de 50 % des ressources de son processeur pour gérer l'affichage (i.e. créer l'image vidéo). Il en résulte que plus de la moitié de la puissance du processeur est inutilisable. Si c'est le cas de votre ordinateur, vous pouvez espérer doubler la puissance de votre machine pour des calculs importants (il existe un mode «FAST» sur le ZX81).

b) Solution intermédiaire :
C'est plus fréquent qu'on ne le croit ; Il suffit par exemple d'avoir un clignotement du curseur qui soit généré par une routine qui alternativement affiche et efface un pavé. En désactivant cette routine, on peut gagner beaucoup de temps. Ce peut être aussi le défilement du texte à l'écran qui dans le cas d'une solution logicielle ralentit considérablement votre programme.
Par contre, la génération de l'image vidéo proprement dite provient de circuits intégrés spécialisés qui évitent toute surcharge au processeur.
Pour être complet, il faut noter que l'adoption d'une mémoire vidéo commune oblige l'unité centrale à partager l'accès à cette mémoire avec les circuits de génération de la vidéo. On ralentit du même coup la vitesse d'accès à la mémoire contenant le programme (il en est ainsi pour l'APPLE II et le MACINTOSH).

c) Solution matérielle :
Dans ce dernier cas, la gestion de l'affichage est presque totalement indépendante du

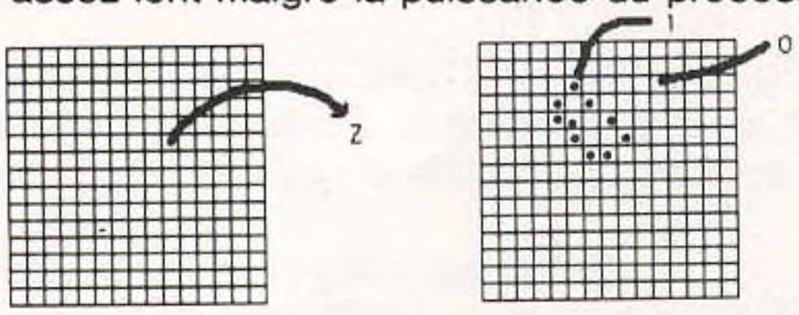
processeur, on peut aller jusqu'à l'utilisation d'un terminal. On gagne du temps car le processeur est soulagé de cette gestion, cependant on perd en souplesse.

Les conclusions sont de toutes façons applicables à tous les types de gestion d'écran et prennent une importance particulière si votre ordinateur souffre d'un affichage peu performant :

L'AFFICHAGE A L'ECRAN RALENTIT TOUJOURS UN PROGRAMME :

Il faut au moins fournir le code ASCII du caractère à afficher et dans le cas d'une chaîne de caractères, il faut aller chercher sa position puis appeler la routine d'écriture pour chacun des codes.

Si l'affichage est du style «BITMAP», c'est-à-dire que l'on travaille en haute résolution que ce soit pour du texte ou du graphique, cette routine doit aller chercher l'image du caractère à afficher dans ce que l'on appelle un «FONT» (qui contient le codage du caractère) et ensuite doit l'afficher bit par bit. Les heureux possesseurs d'un MACINTOSH ont du s'en rendre compte, l'affichage caractère par caractère est assez lent malgré la puissance du processeur qui est aux commandes.



Mémoire vidéo classique
1 case = 1 caractère

Mémoire type BIT MAP
1 case= 1 bit= 1 point d'écran

Fig. 3.3.1.1.

3.3.2. L'imprimante :

Les conclusions précédentes sont à reprendre avec cependant une nuance de taille : une imprimante est de 10 à 100 fois plus lente dans la frappe d'un caractère qu'un écran dans l'affichage du même caractère

Donc, réservez les sorties à l'imprimante pour la fin du programme en sauvegardant au fur et à mesure de leur calcul les résultats que vous souhaitez imprimer.

Nous verrons dans la dernière partie quelles sont les solutions pour s'affranchir des problèmes de sortie de résultats ou de listings sans pour cela devoir acheter une imprimante LAZER.

3.3.3. Le clavier :

Il est difficile de diminuer la charge imposée par la structuration du clavier du fait qu'en BASIC, certaines touches sont toujours actives : ce sont les touches d'interruption du programme (le fameux break ou < ctrl > -C pour la plupart des BASICs) et celles d'arrêt et redémarrage du défilement (en général < ctrl > -S et < ctrl > -Q). Comme il est impossible de supprimer ce contrôle, on ne peut rien gagner dans ce domaine.

Vous allez nous trouvez un peu méchants à force de vanter les avantages des gros micro-ordinateurs (si vous possédez un micro familial), mais sur ces ordinateurs-là, le clavier envoie ce que l'on appelle une interruption qui, comme son nom l'indique, interrompt votre programme pour exécuter une routine spécifique qui est chargée de tester la touche qui a été enfoncée. Si ce n'est pas le cas du votre, l'interpréteur doit à ce moment-là prendre en charge cette tache, oui une de plus !

Si votre application utilise beaucoup les ressources du clavier et a besoin d'un faible temps de réaction – essentiellement pour des jeux –, il est souhaitable d'y accéder le plus directement possible. Tout dépend alors du type de liaison entre l'ordinateur et le clavier. Il est impossible d'expliciter tous les cas. On peut cependant distinguer deux catégories :

– soit votre clavier est facilement accessible par l'intermédiaire d'une case mémoire donnée ; dans ce cas, il est facile de récupérer directement le code tape.

– soit le code de la touche enfoncée provient d'une scrutation complexe des lignes et colonnes du clavier et dans ce cas le plus simple reste d'utiliser les routines BASIC usuelles ('GET' ou 'INKEY').

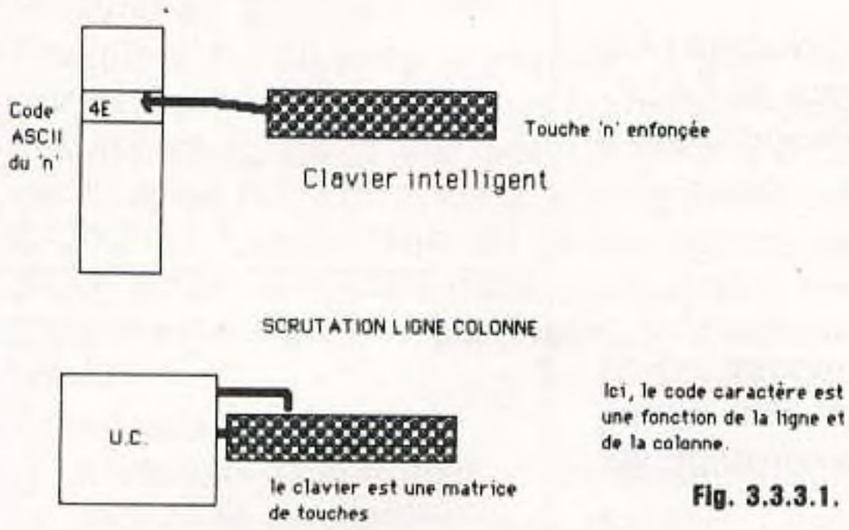


Fig. 3.3.3.1.

3.3.4. Les unités de sauvegarde

Nous allons parler essentiellement des lecteurs de disquettes : la raison est simple, il est inutile de s'intéresser aux cassettes qui sont à la fois trop lentes et trop souvent peu fiables (vous le savez pour les avoir pratiquées ou pour les utiliser encore !).
Comme nous n'avons pas abordé l'utilisation des fichiers sous BASIC, nous ne nous occuperons que de la sauvegarde des programmes. Le bon sens recommande simplement de rechercher à optimiser la taille des programmes et des fichiers : ils seront chargés d'autant plus vite qu'ils sont plus compacts. Ainsi, utilisez plutôt une sauvegarde de vos programmes en mode compresse (sans l'option A pour tous les utilisateurs du BASIC MICROSOFT disponibles sur les machines type IBM ou tournant sous CP/M).
Ce mode est en fait le codage utilisé en mémoire. L'intérêt est double : on diminue le temps de chargement (les programmes sont nettement plus courts) et on se passe d'une conversion qui est nécessaire pour une sauvegarde classique sous forme texte.
Enfin, si vous devez développer une grosse application qui vous oblige à chaîner des programmes, le passage d'un programme à l'autre sera plus rapide.

Résumé :

Voici les dernières conclusions qui découlent de notre brève étude des périphériques :

(8) LIMITER LES SORTIES SUR ECRAN :
Elles ralentiraient votre programme pendant les phases critiques. Cette remarque s'applique a fortiori aux sorties sur imprimante.

(9) IL Y A PEU A GAGNER AU NIVEAU DU CLAVIER :
A part pour des applications ludiques, il est difficile et peu déterminant de gagner sur la gestion du clavier.

(10) SAUVEGARDE DES PROGRAMMES :
En combinant les conclusions formulées au point (6) avec une sauvegarde en mode compresse du programme, on gagne énormément sur le temps de chargement. Plus généralement, nous verrons dans quelques mois qu'il y a beaucoup de pièges à éviter dans la gestion des fichiers sur disques.
Nous n'avons pas voulu être exhaustifs sur le vaste domaine des périphériques mais plutôt apporter des renseignements et des mises au point utiles si vous n'êtes pas un spécialiste. Là aussi, vos expériences sont intéressantes, écrivez-nous, nous reviendrons sûrement plus en détail sur les points précis qui vous intéressent.

4. COMMENT ALLER PLUS VITE

Cette partie ne s'adresse pas spécialement aux programmeurs chevronnés, mais bien au contraire, elle devrait vous permettre de découvrir d'autres aspects de la programmation d'un ordinateur sous BASIC :
Nous allons passer en revue quelques options logicielles ou matérielles qui, tout en vous permettant de continuer à utiliser le BASIC, apportent soit une amélioration des performances, soit une facilité d'utilisation dans des domaines délicats comme les graphismes et autres gestions des périphériques, soit une facilité accrue dans la programmation sous BASIC.

4.1. Les solutions purement logicielles

Deux types d'évolution sont possibles : les outils de programmation qui apportent plus de souplesse et de performances ET qui quelquefois comblent des lacunes du BASIC standard ; les langages plus évolués qui pour ce qui nous intéresse correspondent à des BASICs plus performants.

4.1.1. Les outils de programmation

Ils sont utiles dans les domaines suivants :
– gestion des programmes BASIC : on peut trouver des utilitaires qui permettent de compacter les programmes en supprimant les REMs, en regroupant les lignes susceptibles de l'être, et donc par voie de conséquence, en renumérotant les nouvelles lignes créées.

```
110 I=0
115 IF T(0)=0 THEN GOTO 160
120 FOR J=1 TO 10
130 PRINT T(I)
140 T(I)=0:REM ON NETTOIE T
150 NEXT
160 RETURN
```

Après compactage il deviendra :

```
110 I=0
111 IF T(0)=0 THEN 113
112 FOR J=1 TO 10:PRINT T(I):T(I)=0:NEXT
113 RETURN
```

Il existe aussi, mais c'est déjà plus facile à faire soi-même, des utilitaires qui sortent les références croisées de vos programmes (i.e. pour chaque variable les lignes où elle apparaît).

– utilitaires graphiques : si votre BASIC est peu performant ou (et) peu facile d'emploi dans ce domaine, ils pourront vous apporter des nettes améliorations tant dans la qualité que dans la vitesse de vos programmes.

ATTENTION : Ces utilitaires ne sont pas toujours faciles à incorporer aux programmes, et sont quelquefois incompatibles l'un envers l'autre. Soyez vigilant, et préférez les solutions du constructeur si elles existent. Voici quelques critères pour bien choisir vos utilitaires :
1) la facilité d'interfaçage avec le BASIC
2) la performance du système
3) la souplesse et la richesse des fonctionnalités.

REMARQUE : Nous essaierons dans quelques temps de vous aider à réaliser des utilitaires de ce genre et de publier ce que vous enverrez. Si vous possédez un micro familial, l'utilisation de ces utilitaires est pour ainsi dire la seule possibilité d'améliorer votre BASIC. Il est en effet assez rare de voir plusieurs BASICs de haut niveau sur ce type de machines.

4.1.2. Les langages :

Vous savez qu'il existe de nombreux langages dont les particularités les rendent plus spécifiquement adaptés à certains problèmes. Le BASIC est plutôt un langage passe-partout, aussi a-t-il beaucoup de lacunes. Si vous avez la possibilité de choisir parmi plusieurs BASICs, retenez celui qui est le plus adapté à vos objectifs. Il existe de nombreux BASICs sous CP/M80 et CP/M86 ainsi que sous MSDOS.
Il serait ridicule d'essayer de tout vous dire sur les BASICs, aussi nous orienterons cette dernière partie sur la COMPILATION sous BASIC.

La compilation

C'est finalement la solution miracle pour de nombreuses applications qui sont tout à fait au point mais dont le temps d'exécution est insuffisant. On multiplie la vitesse par un facteur de deux à cinq un peu plus ou un peu moins suivant les applications.

Principe de la compilation :
Chaque ligne du programme est traduite dans le code de la machine (pour les vrais compilateurs car il existe des compilateurs qui génèrent un code intermédiaire qui est ensuite interprété d'une façon analogue à celle du BASIC). Cette phase s'effectue à partir de ce que l'on appelle le programme source qui peut être un ancien programme BASIC qui tournait déjà, ou un programme saisie sous éditeur de texte. Il s'agit donc d'une étape supplémentaire relativement lente qui n'est intéressante que pour des programmes importants, ou lorsque la vitesse d'exécution est une condition sine qua non.

Avantages :
a) La vitesse d'exécution et la compacité du programme :
La traduction ne s'effectue qu'une fois. Le code généré n'est plus surchargé par les REMs, les accès aux variables sont directs ainsi que les sauts de type GOTO ou GOSUB. On peut supprimer la gestion du «BREAK», ce qui simplifie le code et augmente la vitesse du programme.

b) La gestion des erreurs à la compilation :
Lorsque le compilateur traduit les lignes BASIC, il effectue une analyse syntaxique et repère les erreurs éventuelles. Il repère aussi des erreurs moins évidentes concernant la logique du programme.
Voici un exemple typique :

```
10 IF Z=0 THEN GOTO 70:PRINT "erreur Z=0"

erreur ligne 10
```

Le compilateur stipule que ce qui suit le goto ne sera pas exécuté. Avec un interpréteur, il n'y aura aucun avertissement.

c) Listing de compilation :
Il est possible d'obtenir un listing du programme après la compilation : on récupère la liste complète des erreurs de compilation précisées, ainsi que les références croisées des variables avec les lignes.

d) Compatibilité avec votre BASIC :
C'est peut-être la possibilité la plus intéressante, si l'on veut reprendre un programme précédent. Sans aucun effort, on récupère une version plus performante.

e) Optimisation des calculs :
Si votre compilateur est très évolué (dans ce cas la notice saura le vanter), les phases de calculs seront optimisées. En général, les BASICs interprétés ne sont pas très performants dans ce domaine, aussi les compilateurs intègrent une bibliothèque de routines de calculs plus rapides, plus précises, mais aussi optimisent les formules de source, par exemple :
1010 Z = U + V
1020 T (U + V) = A (10) THEN RETURN
le compilateur fera en sorte que le calcul U + V ne soit fait qu'une fois.

Inconvénient :
a) Temps de compilation :
Ce temps ralentit la mise au point d'un programme par rapport à une version interprétée. Aussi, si votre compilateur est compatible avec votre BASIC interprété, il est préférable de mettre au point avec l'interpréteur et de ne compiler que lorsque tout tourne correctement.

b) Erreurs d'exécution :
Tout dépend de votre compilateur : cependant, dans le pire cas, une erreur d'exécution «plantera» votre ordinateur sans que vous puissiez savoir d'où vient l'erreur.

c) Limitations :
Il est fréquent que le compilateur ne reconnaisse pas certaines commandes graphiques ou autres instructions très particulières (pour la génération de musique, etc.). Surtout ne vous attendez pas à ce qu'il soit compatible avec des utilitaires étudiés dans la précédente partie.

d) Coût d'un tel logiciel : c'est cher !

4.2. Solutions matérielles :

Voici quelques matériels susceptibles de réduire l'influence des périphériques sur la performance globale de votre ordinateur.
a) Pour l'unité centrale :
Il est possible en général d'augmenter la taille de la MEV (Mémoire Vive) afin d'avoir plus de place pour les variables et éviter de stocker les gros tableaux dans des fichiers.
Sur certains ordinateurs (il faut qu'ils aient des possibilités d'extension), il est possible d'ajouter des cartes graphiques plus performantes, des processeurs plus puissants restant néanmoins compatibles avec votre BASIC.
b) Pour les périphériques :
Vous pouvez ajouter un buffer pour imprimante qui peut accepter jusqu'à 64 K octets de mémoire et décharge le processeur central de l'attente (vous pouvez alors travailler pendant l'impression).
Si vous êtes riches et que votre machine se le permet, offrez-lui un disque dur, elle se fera un plaisir de vous charger vos applications 5 à 10 fois plus vite. Dans le cas contraire, il vous reste la possibilité d'utiliser ce que l'on appelle un pseudo-disque qui trompe votre système d'exploitation et simule une unité de disque. En fait, ce disque est en mémoire (pour les micros qui ont de la place) ou installé dans une mémoire annexe (pour ceux qui en ont moins). Les accès sont grâce à la vitesse des mémoires vives encore plus rapides que ceux d'un disque dur ; il faut faire attention à sauvegarder les fichiers créés ou modifiés dans ce disque sur un support non volatile (disquette ou disque dur).
Remarque : Il faut noter que ces modifications matérielles ne sont réellement utilisables qu'avec un minimum de logiciels qui assurent la transparence et donc la compatibilité avec la configuration existante.

Précisions sur le cours n° 1 pour le Thomson T07/70

M. J. B., 15100 St Flour, nous fait part du problème suivant : lorsqu'il utilise le programme 3.4.1. du cours précédent, il n'obtient pas ce que nous prédisions pour le codage des valeurs réelles en virgule flottante.
Le T07/70 code un réel sous la forme $x = a.k^b$ avec a compris entre 0,5 et 1 et k vaut 2. Les autres différences proviennent du fait que le premier octet représente l'exposant b et que sur celui-ci, le 1er bit est mis systématiquement à 1. La mantisse est codée sur les 3 octets suivants par puissances décroissantes.
Tout cela, notre lecteur l'avait discerné ; ce qui avait échappé à sa perspicacité se trouve uniquement dans le premier bit du deuxième octet. En effet, comme a est toujours compris entre 0,5 et 1, il devrait toujours y avoir le 1er bit à 1 représentant 1/2 dans la somme formant a. En lieu et place, se trouve le signe de la variable réelle, la logique reprenant son cours dès le 2e bit avec la valeur de $2^{-2} = 1/4$.
Notre lecteur nous avait exposé son problème pour le codage de 10.0, reprenons-le, forts des explications précédentes.

$x = 10.0 = 16 . 0.625$
$= 2^4 . (1/2 + 1/8)$

Exposant

1	0 0 0 0 1 0 0	0	0 1 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
↑ non significatif		↑ signe	↑ 1/8		

Pour l'exemple du cours dernier 6.7, nous obtiendrons

1	0 0 0 0 0 1 1	0 1 0 1 0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0

C'EST ARRIVE DEMAIN

(en direct de notre envoyé permanent dans la Silicon Valley)

Dans un milieu où tout doit être pris au conditionnel, un fait semble néanmoins établi. Le standard MSX n'a pas rencontré ici le succès espéré par ses créateurs et diffuseurs japonais. Le système MSX est pourtant alléchant dans son principe. Il s'agit, ni plus ni moins d'un vrai standard en micro-informatique, sans doute le premier aussi complet, permettant d'acheter une machine «les yeux fermés» comme dans un certain grand magasin parisien très coté ici, et de pouvoir acquérir, avant même de rouvrir ces yeux, n'importe quel logiciel destiné à ce standard, développé par une grande partie de l'industrie niponne. Le phénomène résultant est assez étrange. Aux USA, le procédé n'a tout simplement pas séduit les acheteurs de systèmes familiaux, aux profits des grands fournisseurs traditionnels, qui se portent bien merci !!! La vente de système à base de Z80 et de CP/M, ou de 6502 avec compatibilité APPLE II ne baisse toujours pas, malgré un archaïsme que personne ne nie plus.

Mais ces systèmes inspirent confiance, et leur bibliothèque de programme reste inégalée, et le restera sans doute encore longtemps. D'autant plus qu'une foule de ces programmes sont gratuits, car dans le «domaine public», et il vous suffit de vous rendre avec une disquette vierge chez le distributeur, en général un club, pour le copier. La France, terre de bien des contrastes de ce type, fait par contre, sinon ses choux gras des machines MSX, tout au moins un score plus honorable. Il est d'ailleurs frappant que notre pays soit le lieu privilégié de certaines machines qui échouent en partie de ce côté de l'Atlantique.

C'est, en effet, également le cas du Macintosh, qui se vend bien en France, et si mal ici. Pour y parvenir, certains distributeurs n'hésitent pas à le proposer au prix de 1 995 $, en 512 Ko, avec une imprimante et un lecteur externe. Cela fait à peu près 17 000 F, soit moins que le 128 Ko, sans lecteur en France. Désolé,

mais je dois avouer que je n'ai pas hésité à en profiter, et que mes articles sortent de cette machine. Pour le moment, je baigne dans un océan de félicité, en attendant les premiers problèmes, qui arrivent en général dès que l'on veut aller voir comment tourne la machine. De ce point de vue, cet appareil est un modèle de système fermé, et APPLE aura fait tout son possible pour décourager le particulier curieux, ne décrivant même pas en détail les possibilités des programmes d'origine de la machine (vous savez bien, les fameuses ROM, qui peuvent être utilisées, mais non modifiées ROM = read only memories, mémoires ne pouvant qu'être lues). Par contre, l'utilisation potentielle, ou celle qu'en ont faite les grosses firmes de logiciels, est très bien exploitée publicitairement, merci. Le Macintosh est un système fermé à en pleurer pour un particulier sans connaissance, et qui voudrait aller voir. Si au moins le système d'exploitation était un modèle d'efficacité et de rapidité. Mais on est loin du compte.

Revenons à autre chose. La communauté informatique américaine est persuadée que l'avenir de la micro-informatique est dans les microprocesseurs Intel tels que le fameux 8086, plus que dans la famille des 68000, même le 68020, dont tout le monde pense qu'il est «trop» performant pour la technique actuelle des entrées-sorties et, plus généralement, pour les temps de réponse du reste des systèmes, en particulier les mémoires. Et pourtant, c'est de loin le meilleur des processeurs du marché. En clair, Motorola aura une fois de plus fait preuve de génie dans la réalisation d'un processeur, mais l'équipe développement Motorola est encore très inférieure à celle des autres constructeurs. Pour reprendre la comparaison avec le 8086 et consorts, si tous s'accordent à dire que le 68020 est le meilleur, et qu'il est totalement compatible avec le 68000, tous préfèrent acheter et développer autour de la famille 8086. Il y a donc un truc !
Il semble bien que son principal vrai défaut est que le seul véritable exemple de réalisation autour de Motorola soit le Macintosh/lisa qui, nous l'avons vu, sent un peu le soufre de ce côté de l'Atlantique, ce qui nuit au processeur lui-même. Un comble.
Un autre problème est que Unix, si populaire aujourd'hui dans la micro-informatique (et on peut vraiment se demander pourquoi), est disponible sur de nombreuses machines, dont très peu à base de 68000. Or Unix est devenu un argument de vente (nota : sur Mac, vous pouvez l'avoir pour la «modique» somme de 50 000 F HT).

L'annonce vient d'être faite de la sortie d'un nouveau compilateur Modula-II (sur langage du Pascal et créé également par N. Wirth. Jusqu'à présent, le seul compilateur disponible était celui de SofTech, et inspiré de la philosophie UCSD (University of California - San Diego), c'est-à-dire avec la machine derrière. Volition, dont les patrons viennent tous de UCSD, ont donc annoncé un produit concurrent, basé (origine oblige) sur ce même environnement UCSD. Il s'agit d'une très belle réalisation et il devient par là même possible de développer avec Modula-II, ce qui auparavant n'était faisable qu'avec la légendaire Lilith-machine de N. Wirth, micro-ordinateur de rêve, conçu autour et pour Modula-II, et facturé aux alentours de 20 millions de centimes. C'est cher, même pour une sorte de perfection. La documentation du compilateur de Volition est exhaustive et, même si ce ne peut être considéré comme un livre pour débutant, elle est somptueuse dès que l'on connait un peu ce langage.

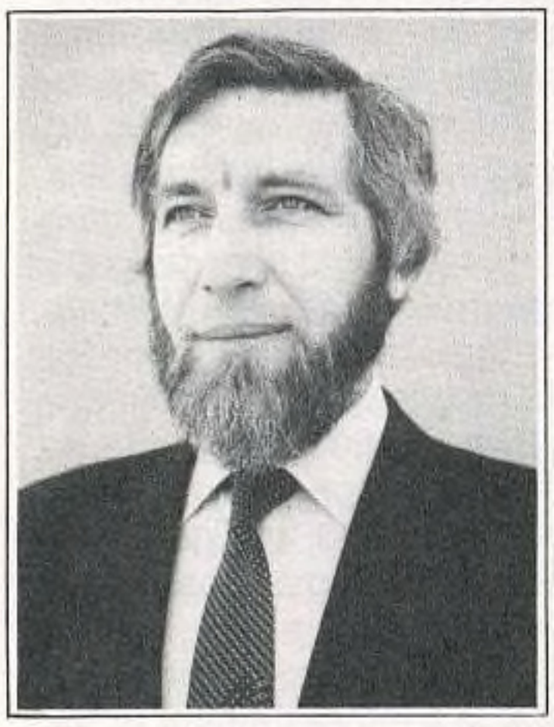
Ceci nous amène à parler de deux phénomènes intéressants. Tout d'abord l'amélioration des documentations, ensuite le problème de l'édition informatique. Il y a matière à disserter, et ce des deux côtés de la grande bleue.
La documentation est enfin devenue aux USA ce qu'elle aurait due être depuis le début, un argument de vente. En conséquence, on assiste à la floraison de notices de 1 000 pages, illustrées d'exemples complets, d'exercices, et accompagnées de disquettes de démonstrations, permettant d'avoir un aperçu d'utilisations auxquelles l'acquéreur moyen ne pense pas toujours. Vous pouvez ainsi tirer profit de votre logiciel, et économisez parfois l'achat d'un autre produit, qui devient inutile grâce à la bonne connaissance des ruses disponibles. Il est indiscutable qu'en Europe non anglophone, la situation est nettement plus frustrante, en raison de documentations misérables, parfois directement à base de photocopies de listings. Une horreur ! Surtout quand on pense que la micro-informatique est censée s'ouvrir aux professions libérales, aux artisans, et autres non-initiés qui achètent sur des conseils fumeux, et se retrouvent devant un système inutilisé en raison de la totale incompréhension entre l'ordinateur et son possesseur. Alors si les programmes contribuent à enfoncer ce dernier, rien ne va plus, et d'ailleurs, les jeux seraient vite faits, après une période d'euphorie, car les attrapes-nigauds ne durent jamais. En conséquence, la tendance semble vouloir évoluer favorablement, et il faut en remercier la concurrence des produits français et germaniques, qui ont contraint les produits anglo-américains à se mettre au niveau.

Le second point est celui de la publication de livres spécialisés. Dans ce domaine, le marasme que connaissent les USA est révélateur. Pour une fois, notre bel hexagone avait de l'avance, la chute de l'édition spécialisée ayant commencé vers le milieu de l'année 1984, alors qu'ici, le sujet devient depuis peu d'actualité. Le nombre de revues ou de livres publiés et ne présentant aucun intérêt est tel que la question était sur toutes les lèvres depuis quelques mois. Comment ces revues vivent-elles ? La réponse évidente avait été tout de suite fournie, elles vivent à crédit. De même pour les livres. Alors, la seconde question est : pourquoi existent-elles ? Tout simplement pour occuper le créneau dans l'espoir d'un second souffle des ventes dans le grand public, et pouvoir être là au bon moment, et non avec six mois de retard sur les concurrents. Il faut donc vivre sur un grand pied, pour cacher le précipice au client, et ne voir que de modestes rentrées. Le phénomène est intéressant, car en France, depuis plus d'un an, les revues spécialisées, à part deux ou trois d'entre elles (dont celle que vous lisez actuellement, merci à vous), perdent de l'argent à un rythme que je qualifierai volontiers d'étonnant. Une locomotive de l'édition s'en tire honorablement, et les autres revues ne s'en sortent pas du tout. En ce qui concerne les livres spécialisés, après une période de vente forcée en raison du manque de titres, la profusion actuelle devrait permettre au lecteur de ne retenir que les plus remarquables. Hélas, la lecture des critiques ne permet pas de discerner le bon grain de l'ivraie, et même si la tendance s'inverse petit à petit, de très mauvais livres, ou pire encore, de très mauvaises traductions continuent de voir le jour. Comment faire un choix quand par définition ces publications sont censées apporter au lecteur une connaissance qu'il ne possède pas. Renseignez-vous auprès de vos amis, auprès de club, ou écrivez-nous, nous vous répondrons toujours dans les colonnes de cette revue. A noter que les américains sont tellement sensibilisés au problème que de grandes campagnes sont en cours pour attirer l'attention du public, mais le fossé reste grand entre les revues grand public, qui sont rarement de bonne qualité, et les livres plus pointus, plus techniques, qui sont souvent de très haute qualité, pour ne pas dire excellents. Ce type de livre est en voie d'apparition en France, mais la présence de traductions des véritables bibles que sont certains livres US est un frein aux productions locales, car faire mieux relève de l'exploit, et demande un travail que peu d'auteurs français peuvent se permettre.

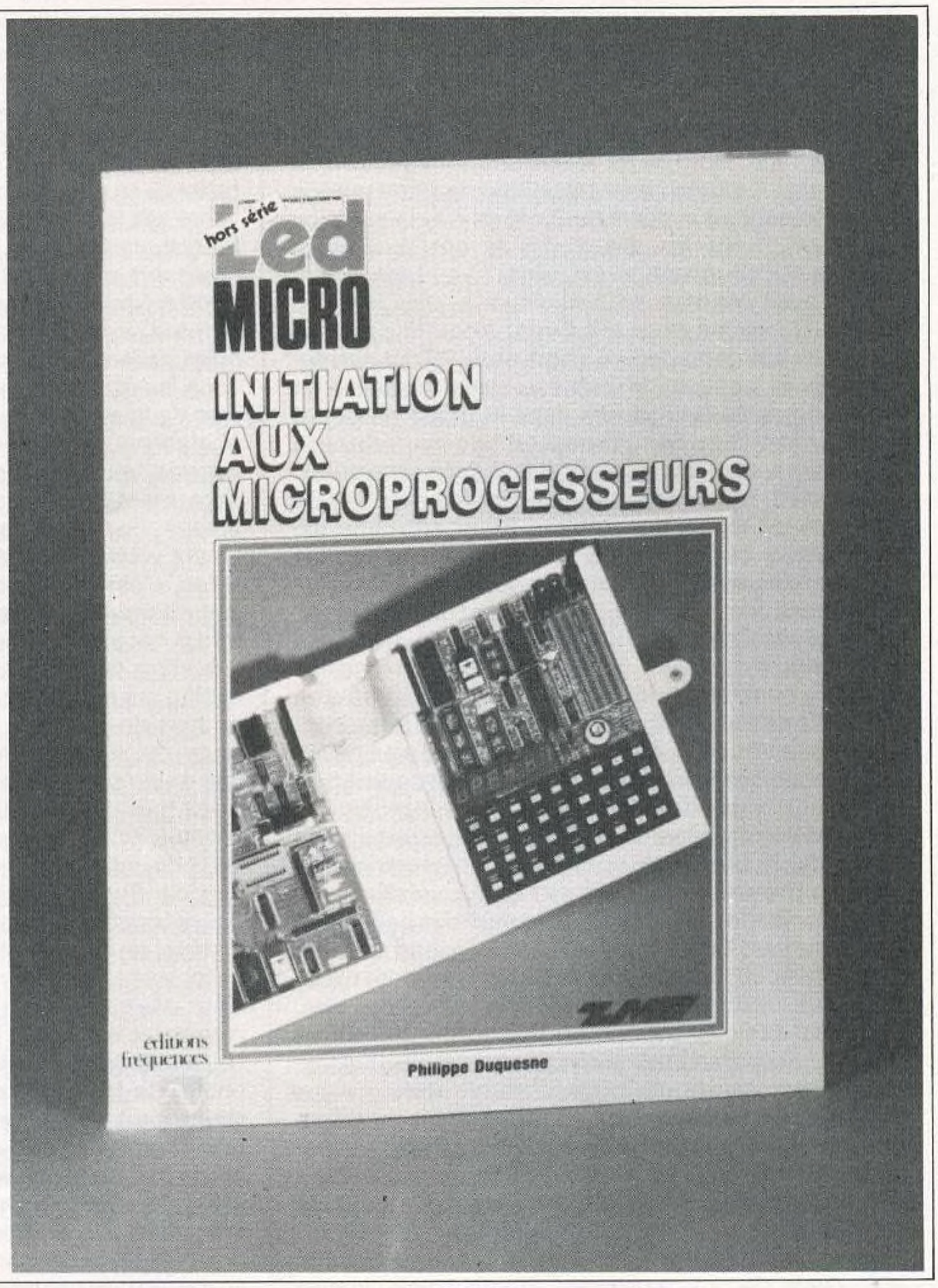
Un lecteur, Monsieur Maigrot, nous a écrit pour nous raconter ses mésaventures avec le service après-vente. Dans son cas, c'est la société distribuant l'ordinateur Amstrad qui est incriminée. Pour une panne de lecteur de cassette, l'appareil a été retourné au SAV Amstrad, et un mois plus tard, après bien des péripéties, l'appareil était enfin restitué, mais toujours en panne. Il faut savoir que le service après-vente est le point noir de l'informatique mondiale. En France, comme aux USA ou au Japon, la politique des fabricants est la vente à tout prix, même au prix d'un effort moindre au niveau de la qualité du service fourni après la vente, et parfois avant la vente. En effet, un point remarquable est l'incompétence des vendeurs dans les magasins français, quelle que soit la dimension de la surface de vente, la rotation des vendeurs et des nouveaux produits étant trop rapide. Vous pourrez vous entendre déclamer les qualités d'un appareil, surtout si ces qualités sont décrites sur un prospectus, mais dans la plupart des cas, on ne pourra vous en dire plus. En ce qui concerne la réparation, c'est pire encore, car là il y a nettement tromperie dans un nombre incalculable de situations. Le retour sans réelle réparation est un fait fréquent, la formation des réparateurs étant inexistante et donc baclée sur le tas. Votre serviteur ayant eu à affronter ce type de problème, il m'est possible d'en parler en toute quiétude et détachement. Dans ce domaine, une seule solution existe, pour les plus chanceux. Il s'agit tout simplement de pouvoir faire une confiance aveugle en son revendeur, et donc discuter avec lui longuement avant l'achat, pour tester ses connaissances. Eventuellement descendre dans les antres de sa boutique pour évaluer la place prise par le matériel en SAV, la pile étant inversement proportionnelle à ses compétences en la matière. Mais, Monsieur Maigrot, vous avez été victime du mal du moment, c'est-à-dire la totale ignorance de vendeurs qui, avant de placer des ordinateurs, s'occupaient de photo, de Hi-Fi, ou même encore, le cas existe, de machines à laver. Alors il ne faut pas trop attendre de tels «professionnels» de l'informatique, qui en savent moins que vous qui aviez probablement lu des essais avant de vous inquiéter de tel appareil. Tant que les fabricants parviendront à vendre sans améliorer ce point, ils le feront sans hésiter, la déontologie de la profession semblant depuis toujours être élastique. Mais le mouvement inverse ne pourra s'amorcer que si les particuliers font le nécessaire, ce qui signifie comparer, s'informer, en un mot, accroître leur connaissance du marché, avant d'acheter. En clair, les associations de consommateurs ont un bel avenir dans ce domaine.
Au mois prochain.

Microprocesseurs
un cours essentiellement pratique !

Pour ceux qui veulent aborder la micro-informatique en désirant en connaître les éléments essentiels ; ceux pour qui la « puce » ne doit pas rester un mythe.



Philippe Duquesne, ingénieur électronicien (I.S.E.N.) est chargé du cours de microprocesseurs au C.N.A.M. de Paris. Depuis plus de dix ans, il a pris goût à l'enseignement et il est l'auteur d'un ouvrage didactique sur l'électronique digitale et notamment d'un cours pratique de microprocesseurs. Fervent pratiquant du « dialogue» école/industrie, après avoir exercé les fonctions de chef de département électronique chez Burroughs, second constructeur mondial en informatique, il est actuellement chef du service Etudes Electroniques au sein de la direction technique chez Messier Hispano Bugatti (groupe SNECMA) avec, pour principal objectif l'introduction des microprocesseurs dans les trains d'atterrissage.



En vente chez votre libraire et aux Editions Fréquences

Bon de commande

Je désire recevoir le livre : INITIATION AUX MICROPROCESSEURS au prix de 105 F (95 F + 10 F de port).
Adresser ce bon aux EDITIONS FREQUENCES 1, bd Ney, 75018 PARIS

Nom ..... Prénom .....
Adresse .....
..... Code postal .....

Règlement effectué : ☐ par C.C.P.
☐ par chèque bancaire
☐ par mandat

COURS DE GENIE LOGICIEL

De la théorie à la pratique

Charles-Henry Delaleu

Dans la première partie de ce cours, nous avons donné les raisons d'être du génie logiciel ainsi que rappelé certains termes dont il est bon de se souvenir. Ce n'est pas par hasard que de très nombreuses personnes se sont penchées sur les problèmes que pose la réalisation d'un programme. La majorité des analystes est en réalité formée aux techniques de l'informatique. De ce fait, ils ont à traiter, lors de nouvelles applications, des données qu'ils ne maîtrisent pas toujours. Il est donc obligatoire de connaître dès le départ de l'étude les différentes variables. Pour ce faire, le cahier des charges devra être le plus précis possible et une étude de faisabilité sera nécessaire.
La première chose à faire est d'observer avec la plus grande précision l'application non-informatisée qu'il convient d'automatiser. Chaque étape, chaque fonction sera décortiquée jusqu'à arriver à des opérations de bases absolument maîtrisées.
Il ne faut jamais avoir peur de passer pour un ignorant à cette étape car le résultat sera toujours le même : une perte énorme de temps à la réalisation de l'étude finale. Dans le cas où il y a facturation de l'application informatisée, ceci viendra absorber le bénéfice escompté ou, pire encore, l'étude se traduira par un déficit qui ne fait jamais plaisir.

LES DIFFERENTES ANALYSES EN GENIE LOGICIEL

L'analyse dans un projet peut se diviser en plusieurs étapes bien définies. De même, le processus à utiliser peut varier suivant les méthodes préférées par l'utilisateur.
Mais avant d'entrer plus dans les détails, il est bon de se rappeler des notions généralement utilisées dans le jargon informatique :
- Analyse descendante
- Analyse ascendante
- Analyse fonctionnelle
- Analyse organique

D'une manière peu précise, il est possible de dire que l'analyse est une étape de la programmation ayant pour objet de décomposer un problème en un ensemble d'opérations de base visant à permettre la conception et la réalisation d'un algorithme qui donnera naissance à un programme d'application. C'est la phase qui doit aboutir à une solution acceptable.

ANALYSE DESCENDANTE

On appelle analyse descendante une analyse dans laquelle l'élaboration du programme final se fait en décomposant le problème global en sous-programmes, qui eux-mêmes se décomposent en actions élémentaires.

ANALYSE ASCENDANTE

Ici le problème est pris à l'envers, en effet, on part des outils donnés par le langage de programmation pour remonter progressivement au niveau de l'application en créant des couches de logiciel de plus en plus élevées.
En d'autres termes, on commence par créer des procédures, puis des sous-programmes contenant ces procédures. Puis enfin, un programme principal qui gère les sous-programmes.
Cette technique est généralement toujours utilisée chez les débutants. Elle est plus lourde à gérer que la précédente analyse, de plus, elle conduit le plus souvent à un résultat moins bien architecturé.

ANALYSE FONCTIONNELLE

Il s'agit d'un processus d'analyse qui consiste lors de la mise en œuvre d'une application à décomposer le traitement en diverses phases auxquelles on associe des modules dont on décrit :
- Les données
- Le fonctionnement
- Le ou les résultats.

ANALYSE ORGANIQUE

L'analyse organique consiste à formuler les conclusions de l'analyse fonctionnelle en fonction du langage de programmation retenu, mais aussi en tenant compte du matériel utilisé. C'est le lien entre la phase intellectuelle et la réalisation physique de la programmation.

NOTES PERSONNELLES

ANALYSE FONCTIONNELLE

L'analyse fonctionnelle peut se diviser en 4 concepts bien définis. Ils permettent d'aborder l'ensemble de cette analyse tout en étant certain de n'oublier aucune fonction. Il s'agit en fait d'une sorte de méthodologie bien rodée.

1. LA METHODOLOGIE

Il s'agit de bien définir les entrées et les sorties du système à savoir :

Entrées → Traitement → Sorties

Les entrées : il convient de déterminer quels types d'entrées seront utilisés en fonction du cahier des charges et des spécifications. Pour une utilisation de bureau, on utilisera un terminal avec écran et clavier, dans l'atelier ce sera un lecteur de code barre par exemple.

Le traitement : il faudra mettre au point des algorithmes chargés de remplir des fonctions.

Les sorties : Quels sont les types de sorties à utiliser ? Les résultats du traitement doivent être présentés sous quelles formes :
- Ecrans
- Tables traçantes
- Actionneurs, etc...

2. APPROCHE METHODIQUE

A. Données stockées : types des données stockées
B. Fichiers : où sont stockées les données → fichiers, bases...
C. Saisies : comment se font les saisies
D. Sécurité : Contrôle de validité des saisies
Domaine de validité du traitement
Sécurité des sorties émises

3. CIRCULATION DES INFORMATIONS

La circulation des informations peut se faire de différentes manières. Chez IBM, l'information progresse à travers des terminaux informatiques classiques du style écran ou imprimante. A la société Bell System, la majorité des informations circule à travers les télécommunications. Chez Sony, la vidéo communication est reine. Bref, même en micro-informatique, la téléinformatique peut être un outil très intéressant (Modem etc).

4. L'EQUIPE

Qui va effectuer l'analyse : des mathématiciens spécialistes de logique, des experts de C.A.O., F.A.O, ou I.A.O ? D'où viennent les connaissances ?

ANALYSE ORGANIQUE

CONTENU

Le contenu de l'analyse organique concerne les données codées. Il faut organiser les informations de base. Chaque donnée doit être codifiée, saisie, classée (fichier, tableau...).

Codifée

- Mise en forme
- Vérification du domaine de validité

Saisie

- Type de récepteur
- Traduction éventuelle

Classée

- Vérification du classement
- Accès aux informations.

IL CONVIENT D'ORDONNER LES TRAITEMENTS

Chaque opération doit être intégrée dans un module. Un ensemble de module sera rassemblé dans un sous-programme. Les sous-programmes seront accessibles par un programme principal.

Nota : Il convient d'être très vigilant à cette étape : le changement d'un module ne doit jamais remettre en cause le programme et l'application générale.

LA PRESENTATION DES RESULTATS

Il s'agit là d'une opération très importante. Rien ne sert d'obtenir une grande quantité d'informations si les résultats ne sont pas correctement présentés. Il est souvent préférable d'avoir un petit graphisme plutôt qu'un impressionnant listing.

Ceci est typique dans les applications de gestion et d'analyse financière où les destinataires des informations sont submergés par des centaines de pages habdomadaires et sont incapables de réaliser une synthèse rapide et fiable des documents qu'ils utilisent.

LES CONTROLES

1. Dans un premier temps, les contrôles se feront sur les premières approches du programme
a) Documentation claire
b) Facilité d'accès
c) Facilité d'utilisation
Un programme qui ne répond pas à ces trois critères a de fortes chances d'être peu fiable.
2. Contrôle quantitatif : le volume d'information à traiter est-il compatible avec l'application ?
3. Contrôle qualitatif : caractéristiques techniques à vérifier.

Nota : Les contrôles sont toujours insuffisants. Il convient de consacrer à cette étape le plus grand soin.

NOTES PERSONNELLES

ANALYSE DES VALEURS

Si les études et les fabrications d'ordinateurs sont aujourd'hui bien contrôlées en termes financiers et commerciaux, le marché du progiciel est en pleine mutation. En effet, il s'agit là d'un phénomène assez récent, mais les plus belles réussites en termes de création d'entreprises et de réalisation de bénéfices se sont opérées dans l'étude et la fabrication de programme.
Dès lors, le logiciel qui devient progiciel se doit de se soumettre aux lois des produits de consommation . Parmi ces dernières, notons l'analyse des valeurs qui, dans la majorité des cas, permet de réaliser un produit beaucoup plus ciblé.
L'analyse des valeurs a été créée en 1947 par Monsieur Miles de General Electric.
Elle correspond à :
- Ce que l'on reçoit pour son argent
- Ce que l'on donne à quelqu'un pour une somme d'argent.

DANS L'INDUSTRIE NOUS AURONS :

Produit manufacturé
La phase négociation → Il existe plusieurs techniques pour résoudre un problème.
Les produits existants → Quelles sont leurs caractéristiques
a) caractéristiques techniques
b) prix
c) service rendu...
Les produits nouveaux → Comparaison avec les existants
Spécification-besoin → Ce que le client attend

Produit de transformation
Les procédés de fabrication
Analyse des valeurs des procédés
Investissement

HORS INDUSTRIE NOUS AURONS :

L'analyse des valeurs des procédés

METHODE DE L'ANALYSE DES VALEURS

Produit / procédé
1. Extérieur au produit → Sa fonction
2. Intérieur → Coût, prix de revient

Fonction
1. Composant élémentaire de l'utilité
2. Nomenclature d'utilité
3. Service rendu

LE PRIX

En ce qui concerne l'informatique, il est possible de définir le prix d'un progiciel de la manière suivante :
1. Usage (est-ce un produit courant, un produit rare ?)
2. Estime (est-ce un produit de grande réputation ?)
3. Le coût de l'étude, de la fabrication.

ANALYSE DES VALEURS

PHASE — **PRODUIT A ETUDIER**

	ECONOMIQUE	Environnement + objectif + invariable	TECHNIQUE
ORIENTATION De quoi s'agit-il ?	Informations économiques Prix de revient Quantités...		Informations techniques Spécifications - Gammes Plans - Nomenclatures...

ANALYSE FONCTIONNELLE/ECONOMIQUE

ANALYSE

A quoi cela sert-il ?
Combien cela coûte-t-il ?

- Analyse fine et comparative des coûts → Elimination des coûts inutiles
- Analyse des fonctions et de leurs performances → Elimination fonctions performances inutiles
- → Redimensionnement économique-technique

SYNTHESE RECHERCHE

Comment faire autrement ?

PRODUIT DEFINI PAR SES FONCTIONS
JUSTE NECESSAIRES
ETALONNAGE DE LA VALEUR

COMMENT REALISER AUTREMENT LES FONCTIONS NECESSAIRES ?

Recherche imaginative } internes
Recherche des informations } externes

EVALUATION

A satisfaction égale, quelle est la meilleure valeur ?

RECONCEPTION

Tri du meilleur projet
- Coûts ?
- Validité technique ?

MISE EN FORME DECISION

FEUILLE DE SUGGESTION

- Application immédiate
- Mise en mémoire
- Rejet

LES OBJETS

Nous avons vu que les règles menant à la solution d'un problème soluble automatiquement doivent décrire précisément les entités indissociables suivantes :

- Les OBJETS
- Les ACTIONS

Un OBJET est une valeur appelée INFORMATION, cette valeur appartient à un DOMAINE de VALIDITE.

Exemple : Dans une facture, nous allons introduire un taux de T.V.A.. Ce taux est égal à :

- Articles de luxe → T.V.A. = 33,33 %
- Articles courant → T.V.A. = 18,60 %
- Livres → T.V.A = 7,00 %
- Presse → T.V.A. = 4,00 %

Si nous faisons notre facture, le taux de T.V.A. ne pourra être que l'un des taux précités. Il s'agit du domaine de validité.

Une information peut être :

- Une INFORMATION ELEMENTAIRE
- Une INFORMATION COMPOSITE
- Une COLLECTION D'INFORMATIONS

INFORMATION ELEMENTAIRE

Une information élémentaire est un objet qui ne peut pas être fragmenté par les actions. Elle est insécable.

Ex : LED-MICRO

Lorsque l'on écrit LED-MICRO, cela correspond à une revue spécialisée. Par contre, si le nom est modifié pour donner L'AIDE MICRO, cela change complètement le sens de l'information. LED-MICRO est donc un objet qui correspond à une INFORMATION ELEMENTAIRE. Le fait de manipuler l'orthographe de ce nom lui enlève son sens.

Une information élémentaire est composée d'un triplet indissociable :

- Sa VALEUR
- Son TYPE
- Son NOM

LA VALEUR

Dans notre exemple de facture, si nous réalisons la vente d'une voiture, la valeur de notre taux de T.V.A. sera 33,33 %.

Si nous lisons la tension d'une pile de 4,5 V lue par un convertisseur possédant une gamme allant de 0 à 10 V, la valeur de la tension de la pile est de 4,5 V. Il s'agit d'un niveau de tension, parmi les niveaux que peut lire le convertisseur.

LE TYPE

Dans notre dernier exemple, la tension à mesurer est du type numérique. Il existe différents TYPES :

- Le TYPE NUMERIQUE
- Le TYPE CHAINE DE CARACTERES
- Le TYPE LOGIQUE
- Le TYPE ENUMERATION

Le TYPE NUMERIQUE : C'est un sous-ensemble de nombres entiers ou fractionnaire.

Ex : 1024 - 3,14116 - 23/75 - 7

Le TYPE CHAINE DE CARACTERES : C'est une suite de lettres, de chiffres, ou d'autres signes.

Ex : Le nom de votre revue de micro-informatique est une CHAINE DE CARACTERES.
Le numéro d'immatriculation de votre véhicule est une CHAINE DE CARACTERES.

Le TYPE LOGIQUE : Il s'agit d'un ensemble de deux éléments :
- VRAI
- FAUX

Ces deux éléments logiques seront repris en détail dans le chapitre consacré aux TRIS.

Le TYPE ENUMERATION : C'est un ensemble de valeurs quelconques données par une LISTE.

Il convient de noter deux autres informations élémentaires :
- LE TEMPS
- UN EVENEMENT

Le temps peut être donné par un processeur spécialisé sous forme numérique au programme qui est en cours d'exécution.

Un événement peut modifier le développement d'une séquence, d'un sous-programme, ex : une interruption.

INFORMATION COMPOSITE

Le meilleur exemple que l'on puisse fournir pour une INFORMATION composite est sans aucun doute le numéro d'identification national. En effet, sous la forme d'une information (il désigne une personne), il rassemble un ensemble d'informations (date de naissance, lieu, etc.).

Chacune des informations élémentaires est appelée CHAMP

Nota : un CHAMP peut être lui même un groupe.

Les groupes peuvent être :
- FIXES
- VARIABLES PAR RECURRENCE
- VARIABLES PAR RECURSIVITE
- VARIANTS

FIXES : Dans le numéro d'identification national, chaque champ est FIXE et invariable pour un individu donné.

VARIABLES PAR RECURRENCE : Dans ce cas, il y a répétition d'un champ. Un champ de type invariable est répété un nombre quelconque de fois.

VARIABLES PAR RECURSIVITE : Ici nous trouverons une subdivision d'un champ de type invariable en éléments comprenant un champ de même type.

VARIANTS : Un groupe de type variant est formé d'informations de type et nombre variables.

COLLECTION D'INFORMATIONS

Les informations ayant un rapport de nature ou de fonction sont très souvent réunies en collections. Dans ce cas, un algorithme sera chargé de gérer les différentes informations contenues. Ici, on appelera une information élémentaire un ARTICLE.

Une collection formée uniquement d'articles de même nature est appelée FICHIER.

Une collection formée d'articles de nature différente mais pouvant être rassemblés par un algorithme est appelée BASE DE DONNEES.

L'organisation et l'accès des fichiers et des bases de données sont assurés par des algorithmes. Ceux-ci peuvent être réalisés de manière :
- LOGIQUE
- PAR VALEUR
- ABSOLU... etc.

Certains fichiers ou base peuvent arriver à une complexité très avancée. Si l'organisation est souvent très poussée, l'accès devra dans tous les cas de figures avoir les caractéristiques suivantes :
- CREATION (création d'un article)

- MODIFICATION (mise à jour d'une valeur)
- INTERROGATION D'EXISTENCE (réponse «VRAI» ou «FAUX»)
- INTERROGATION DE VALEUR (valeur d'un article)
- SUPPRESSION (supprimer l'existence et la valeur d'un article)

Nota : la création, la modification, la suppression sont des primitives d'accès.

LES ACTIONS

Comme pour les objets, nous allons trouver trois grandes familles d'actions. Le processeur traitera des **actions** élaborées grâce à une analyse constituée à partir d'un algorithme.

Les trois groupes d'**actions** sont :
- l'ORDRE ELEMENTAIRE
- l'ORDRE COMPOSITE
- la COLLECTION D'ORDRES.

L'ORDRE ELEMENTAIRE

Un ordre élémentaire correspond à une phrase où ne figure qu'un verbe. Il permet de créer, de donner une valeur, un nom, un type, à une information élémentaire. Il décrit les groupes de champs pour les informations composites. Il définit les articles, leur organisation et leur accès pour les collections d'informations.

Il existe deux structures pour exprimer un ordre élémentaire :

- VERBE + COMPLEMENT D'OBJET DIRECT + PREPOSITION + COMPLEMENT
- VERBE + COMPLEMENT D'ATTRIBUTION + COMPLEMENT D'OBJET

Exemples :

Dans le premier cas, nous aurons : Lire le nombre X
Dans le second cas : Donner à X le type de nombre entier.

Le vocabulaire de ces ordres est composé de :

- VERBES
- COMPLEMENTS
- PREPOSITIONS ET LOCUTIONS

LES VERBES : Ils portent sur :
LES INFORMATIONS : ex. créer, donner une valeur à, lire, recevoir...
LE TEMPS : ex. attendre
LES PROGRAMMES : ex. exécuter, arrêter l'exécution
LES RECEPTEURS : ex. lire
LES EFECTUEURS : ex. écrire.

Nota : Un récepteur pourra être un clavier ou un lecteur de codes barres, un effectueur sera un écran, une imprimante...

LES COMPLEMENTS : Ils indiquent :
OBJETS PARTICULIERS (n° d'imprimante par exemple)
DES TYPES OU DES ATTRIBUTS (type numérique, logique, format).

LES PREPOSITIONS OU LOCUTIONS : Ex. à, de, avant, après, jusqu'à ce que, avec... Nous aurons aussi dans le vocabulaire des ordres élémentaires :
DES PRONOMS INTERROGATIFS (qu'est-ce qui, qu'est-ce que ?...)
DES ADJECTIFS INTERROGATIFS (Quel ?)
DES ADVERBES INTERROGATIFS (combien, quand, depuis quand ?)

LES ORDRES COMPOSITES

On appellera ORDRE COMPOSITE un élémentaire auquel une condition a été ajoutée.
Ex. : Ecrire A + A seulement si A est positif.
La condition logique sera :

SI (Si A = 4 écrire A)
SAUF SI (Ecrire A sauf si A <100)
TANT QUE (Tant que A > 100 faire A + A)
JUSQU'A CE QUE (Faire A + A jusqu'à ce que A > B)
SELON QUE (Selon que A = B ou C, faire...)
SINON (Si A > 100 faire A/C sinon faire A + A)

Outre la condition logique, il est possible d'exprimer une condition temporelle...

SIMULTANEITE (Quand, lorsque, pendant que...)
REPETITION (Chaque fois que)
ANTERIORITE (Dès que, après que)
POSTERIORITE (Jusqu'au moment où).

LES COLLECTIONS D'ORDRES

On appellera COLLECTION D'ORDRES un ensemble d'ordres élémentaires et/ou d'ordres composites. Cet ensemble devra former un tout qui prendra le nom de BLOC. Plusieurs ordres élémentaires et/ou composites seront enchaînés et s'exécuteront de la manière définie par l'algorithme.
Il existe trois genres de collections d'ordres :

LES PROCEDURES
LES SOUS-PROGRAMMES
LES PROGRAMMES.

LES PROCEDURES : Une procédure est un ensemble d'objets et d'actions qui remplissent un rôle défini.
Ex. : Un tel ensemble servira à calculer une fonction alors qu'un autre dessinera les résultats déterminés précédemment.

N.B.

1. Une procédure qui se déroule simultanément avec un autre programme sera appelée TACHE.
2. Dans de nombreux langages de programmation, il est possible de donner un nom à une procédure.

LES SOUS-PROGRAMMES : Un ensemble de procédures sera appelé sous-programme. Si ce sous-programme a un but unique de calcul, il sera appelé FONCTION.
Ex. : Dans un programme de paies, nous aurons différents sous-programmes :

- un sous-programme : menu principal-appel des différentes applications
- un sous-programme de gestion du fichier personnel
- une fonction calcul des paies
- un sous-programme édition des paies...

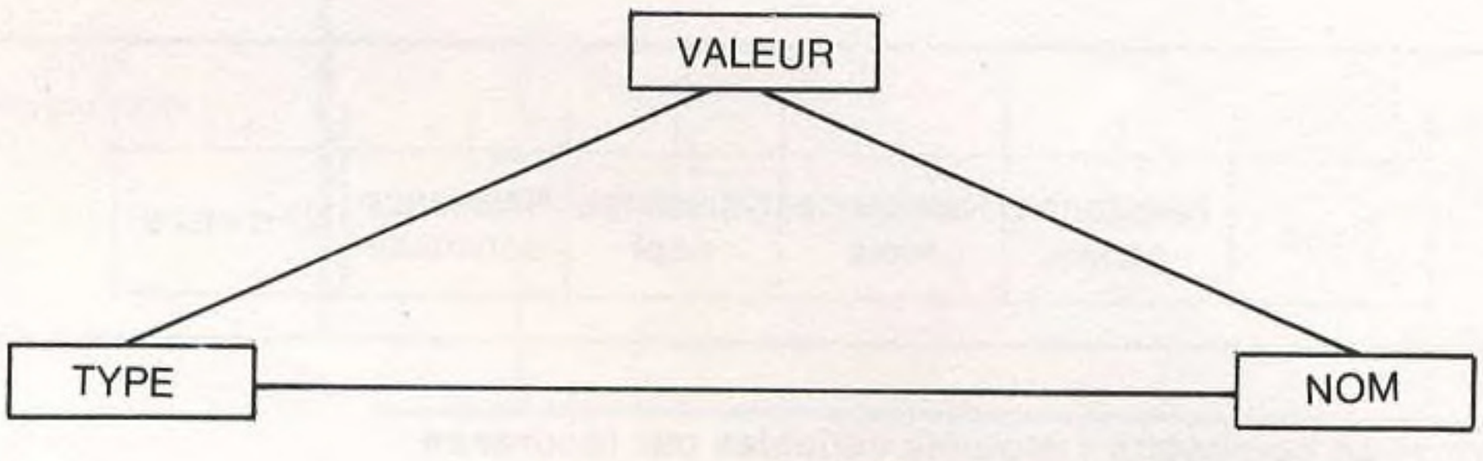
Des sous-programmes pouvant servir dans des applications diverses seront nommés UTILITAIRES. Ex. : un utilitaire de présentation graphique de résultat.

Il convient de noter :

1. Un bon analyste doit toujours être en possession d'un nombre important d'utilitaires en tout genre.
2. Un nouveau programme doit toujours être réalisé de telle sorte qu'il permette d'augmenter sa bibliothèque d'utilitaires.

LES INFORMATIONS

Informations élémentaires

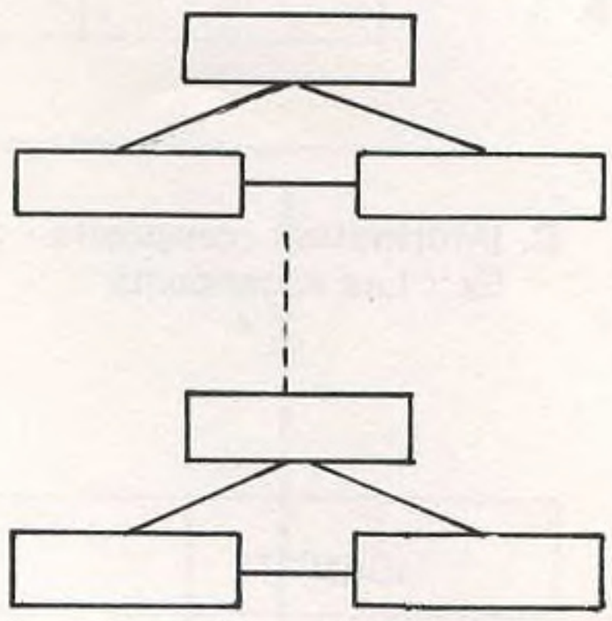


Informations composites

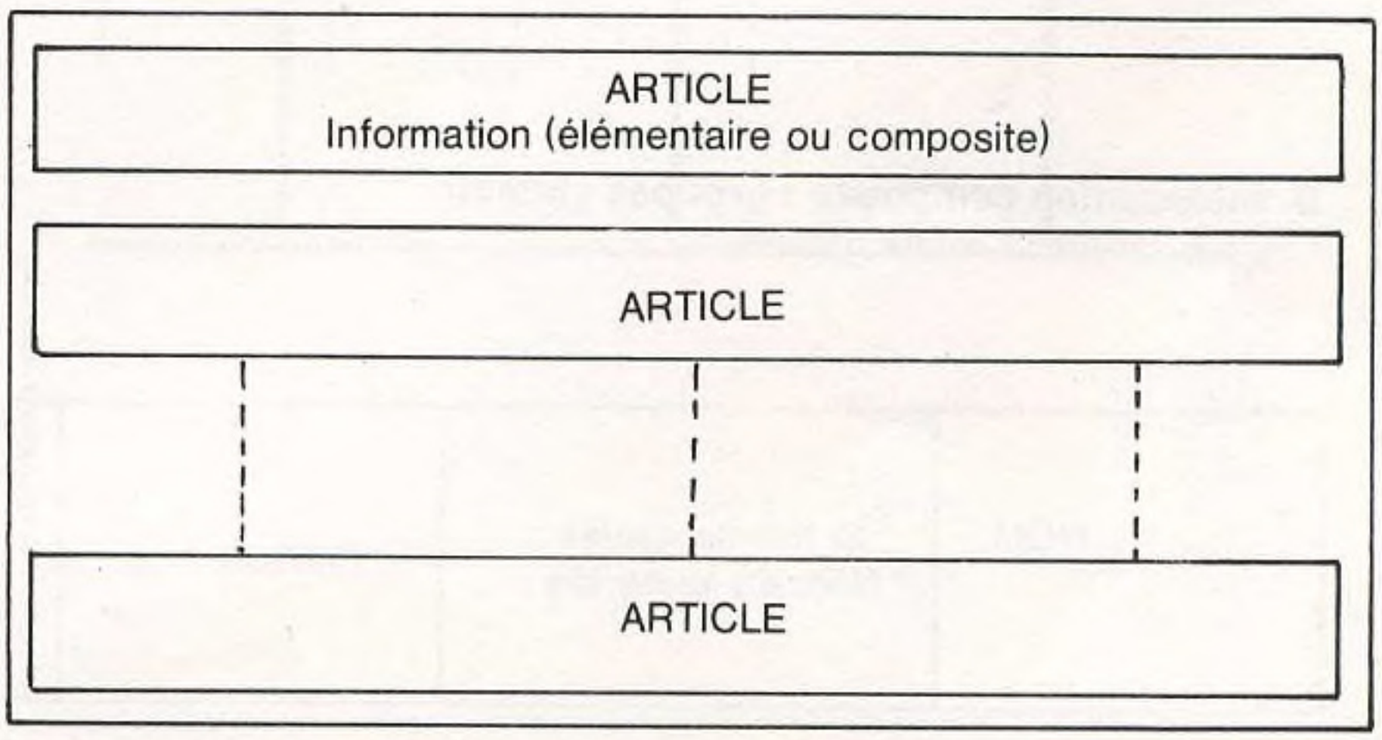
Information composite

Information élémentaire

Information élémentaire



Collection d'informations



INFORMATIONS COMPOSITES

A. Information composite : groupes fixes
Ex. : le numéro d'identification national.

Sexe	Naissance année	Naissance mois	Naissance dépt	Naissance commune	N° d'ordre

B. Information composite : groupes variables par récurence
Ex. : Les prénoms

NOM	Prénom 1	Prénom 2	Prénom 3

C. Information composite : groupes variables par récursivité
Ex. : Les ascendants.

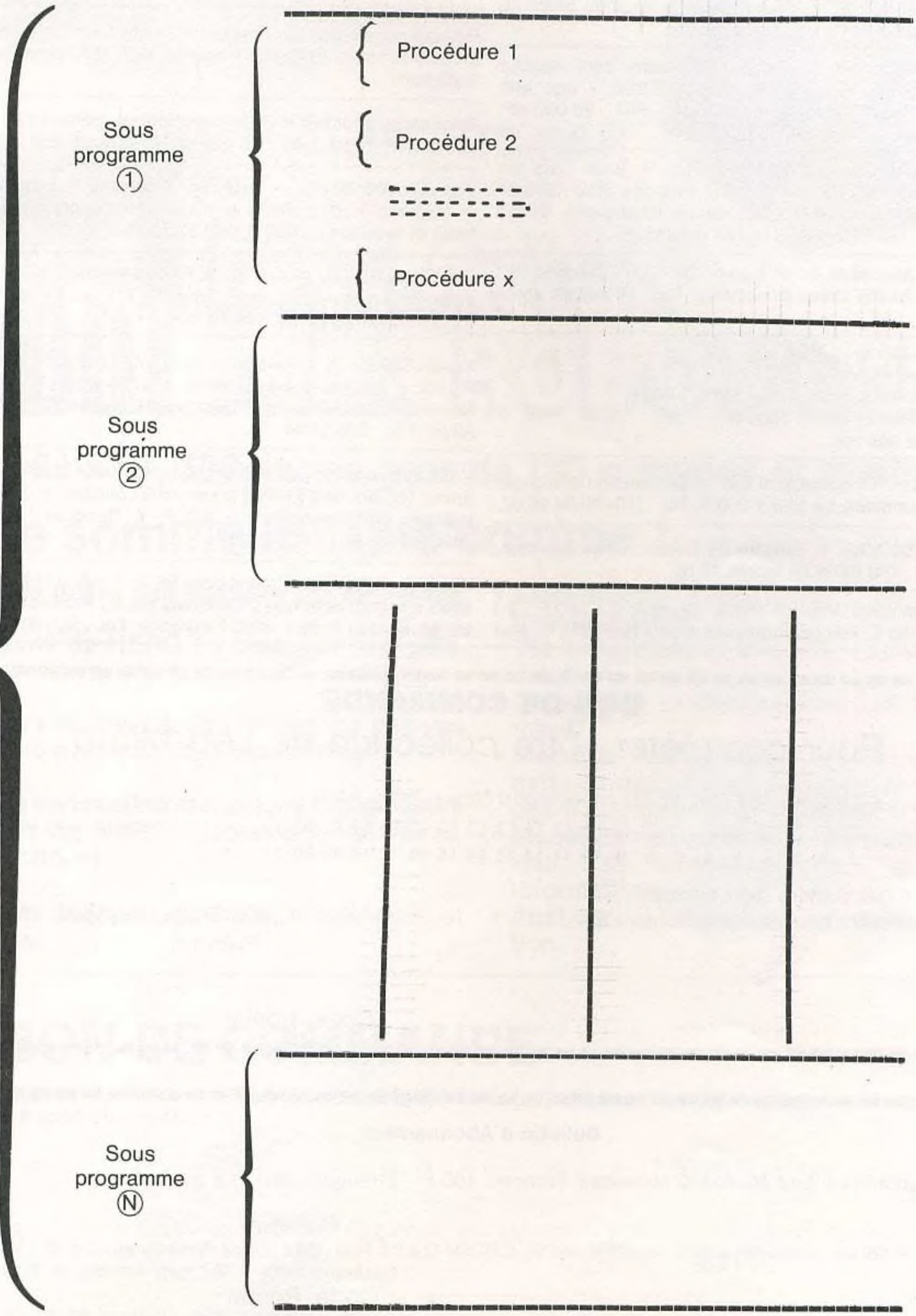
IDENTITE	PERE			MERE		
	Identité	Père	Mère	Identité	Père	Mère

D. Information composite : groupes variants
Ex. : Identité forme usuelle.

NOM	Si femme mariée Nom de jeune fille	Prénom 1	Prénom 2

LES ACTIONS

Les différents niveaux



PETITES ANNONCES

A vendre, cause sans ressources, le tout état neuf : Led-Micro nos 1 à 10 ; 12 ; 14 ; 15 ; 19 à 21 + Sciences et Techniques nos 1 à 7 ; 1 hors série mai 84. Micro et Robots nos 1 à 8 ; 10 à 14 (400 F). M. Berahal Abel, tél. 899.18.06. Donnez rendez-vous ou écrire au 2, rue de la Croix des Craies 94000 Créteil.

Vds U.C. IBM PC/XT 512 K + disque dur amovible demi-hauteur 5 Mo + carte monochrome ou couleur + DOS 2.00 + doc. IBM + softs divers + jeux + 2 cartouches 5 Mo. Prix : 29 000 NF. M. Chemin, tél. : 994.47.34 après 20 h 30.

Vends TRS 80 Mod. 1 niv. 2, 16 Ko + son + jeux. Liste sur demande. Prix de l'ordinateur : 6 000 FF. Prix des jeux : 200 FF pièce. M. Marsallon Jean-Michel 32, rte de Champigny, 94350 Villiers-sur-Marne. Tél. : 303.66.35 (après vacances).

Vends Led-Micro hors série du nº 1 au nº 20 - 200 F. Led du nº 1 au nº 28 - 280 F. 94450 Limeil-Brevannes. Tél. : 569.47.69 après 19 h.

Vends cartouches standard MSX :
1º) Mr Chip - Circus - Charlie : 300 F
2º) Duplicateur manuel à alcool Rally 376 C : 400 F
3º) Oscilloscope Dumont 401-A : 500 F
Tél. (42) 20.48.08 le soir HR.

Vds ZX 81 + 16 K + TV portable N & B + alimentation et manuel + nombreux programmes. Le tout 1 500 F. Tél. : (16-74) 94.25.92.

Vends Casio FP 200 8 Ko + mallette de transport très bon état. Prix : 2 400 F. Tél. : (74) 65.26.94 (après 17 h).

Vds cartouche standard MSX 1) Hyper Olympic 2 : 150 F ; 2) Hyper Sports 1 : 150 F. Vds polycopieuse alcool Rally 376 C, état neuf, peu servi : 350 F. Boîte de couplage d'antenne 2 MHz à 5,5 MHz, possibilité étendre la bande : 150 F. M. Diatta Pierre. Le Montaiguet 1 8, rue des Frères Vallon, 13090 Aix-en-Provence.

Possesseur Sinclair QL résidant Nouvelle Calédonie cherche contacts Métropole. Ecrire M. Chevron, B.P. 382, Nouméa, Nouvelle-Calédonie.

Recherche pour PROF 80 la possibilité de générer des minuscules accentées. Higel J.-M. : 16 (89) 42.70.20 poste 383.

Vds VG 5000, 80 inst. + extensions mémoire + imprimante 40 col. + magnéto + joysticks + logiciels jeux et apprentissage Basic + livres et manuels : 3 900 F. (22) 26.10.38.

Vds Casio PB 700 16 K RAM + FA 10 imp. table traçante + CM 1 magnéto à micro-cassettes + malette et fournitures. Prix à débattre. Tél. hres repas : (65) 45.49.15.

Vends TI 99/4 A + cordon + magnéto + 5 K7 Basic étendu + Péritel + nombreux programmes. Etat neuf : 1 700 F à débattre. Royer Stephan 60, avenue Georges Clemenceau, 94700 Maisons-Alfort. Tél. : 376.07.04.

Vds micro-ord. Sanyo 555 (01.85), 192 Ko, AZERTY, MS-DOS, 2 drives 180 Ko, 640 par 200 pixels en 8 couleurs + div. langages et logiciels professionnels : 9 500 F. Y. Bacquet 17 prom. Marty 34200 Sète. (67) 74.38.81 HR.

Vends logiciels pour Commodore 64, 10 F pièce (150 des meilleurs titres existant) ainsi que 2 consoles de jeu Hanimex. Prix à débattre. M. Barrieu Patrick 3500 Fleurance. Tél. : (62) 06.03.07.

BON DE COMMANDE

Pour compléter votre collection de Led-Micro

A retourner aux EDITIONS FRÉQUENCES 1, boulevard Ney - 75018 Paris

Je désire le nº ☐ (cocher le ou les nos désirés)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

au prix de 18 F par numéro (port compris).

Je joins à la présente commande le montant de ..... F par CCP ☐ ch. bancaire ☐ mandat ☐

Nom : ..... Prénom : .....

Adresse : .....

Ville ..... Code postal .....

Bulletin d'Abonnement

Je désire m abonner à Led Micro (10 numéros). France : 160 F - Etranger : 240 F, à partir du nº .....

Nom ..... Prénom .....

Nº ..... Rue .....

Ville ..... Code Postal .....

Envoyez ce bon accompagné du règlement à l'ordre des Editions Fréquences à :
EDITIONS FREQUENCES 1, boulevard Ney, 75018 PARIS
MODE DE PAIEMENT : CCP ☐ - Chèque bancaire ☐ - Mandat ☐

VOICI ENFIN LA PREMIÈRE PIERRE D'UN DOMAINE ENCORE INEXPLORÉ...

L'ouverture au monde passionnant de la robotique, dans un style simple et direct, travail d'un collectif de spécialistes animé par Claude Polgar.

PRIX TTC 115 F

hors série
...NIQUES D'AUJOURD'HUI
Led
ROBOT INITIATION A LA ROBOTIQUE

Format 21 × 27, 100 pages, plus de 130 schémas et illustrations.

Le sommaire : une somme !

- **La grande relève des hommes par les robots**
- **L'anatomie de HERO 1 :** bras, jambes, ouïe, vue, télémétrie, détection de mouvements.
- **Inventeurs et inventions :** ne confiez pas vos inventions avant de vous être protégé.
- **Cours de conception mécanique :** vocabulaire et notion de base - Ajustement, tolérance, excentricité, etc.
- **Cours de logique générale :** schémas et symboles.
- **Electronique industrielle :** du circuit au démultiplexeur.
- **Vie industrielle :** la CAO, assistante de la création.
- **Conception et construction :** de la tortue au robot.
- **Modules fonctionnels :** construction de la carte de départ pour commander les moteurs pas à pas à partir de votre micro.
- **Maquettes et modélisme :** le modélisme ferroviaire se renouvelle grâce à la micro-informatique.
- **Analyses et méthodes :** les rosaces d'évaluation.

BON DE COMMANDE

Je désire recevoir Led-Robot «INITIATION A LA ROBOTIQUE» (attention, cet ouvrage n'est pas vendu en kiosque) au prix de **125 F** (port compris).

Nom : ..... Prénom : .....

Adresse : .....

.....

ATTENTION : Si je suis abonné soit à LED, soit à LED-MICRO, je bénéficierai d'une réduction de 20 % sur le prix de l'ouvrage et je ne paierai que **100 F** (port compris).

Je vous note, dans le cadre, mon numéro d'abonné : ☐

Ci-joint un chèque bancaire ☐ chèque postal ☐ mandat ☐.

Adressez votre commande et votre règlement aux EDITIONS FRÉQUENCES 1, boulevard Ney, 75018 Paris.

Le Victor PC ne coûte que 24.900 F n'en déplaise à [redacted].

Le Victor PC 15 ne coûte que 24.900 F*.

Certains d'entre vous penseront peut-être – *et nous en connaissons qui aimeraient bien que* ce *soit vrai* – qu'à 24.900 F*, il ne peut s'agir que d'un PC "bradé". Une telle réaction est d'ailleurs compréhensible quand on songe aux prix pratiqués sur le marché, en matière de PC. Prenons par exemple [redacted]. Son PC coûte 50 % plus cher que le Victor PC 15.

Et pourtant, les performances du Victor PC 15 sont équivalentes, *voire supérieures,* à celles de l'[redacted] PC. La preuve, la voici :

Alors que la plupart des micro-ordinateurs propose une capacité de stockage de 10 Mo, le Victor PC 15, lui, offre une capacité de 15 Mo ! De plus, l'utilisateur du Victor PC 15 bénéficie, grâce à un moniteur de 14 pouces, de 30 % de surface écran supplémentaires (la quasi-totalité du matériel concurrent étant équipée d'un moniteur 12 pouces).

Et ce n'est pas tout ! Le Victor VU – l'interface utilisateur – permet un gain de temps appréciable en guidant dans son travail l'usager, par de simples messages organisés comme des menus. Finie, désormais, la consultation fastidieuse et peu pratique du manuel du système d'exploitation !

Et l'on pourrait parler des 5 emplacements d'extensions disponibles pour accroître les possibilités du PC...

Non décidément, [redacted] devra se faire une raison et s'accommoder de la présence sur le marché du Victor PC 15 ! Un PC compatible avec les standards du marché, aussi performant que celui que fabrique [redacted] et à un prix bien plus séduisant que celui affiché par [redacted].

Car au risque de le répéter et de déplaire à [redacted], ces 50 % sont difficilement justifiables. D'ailleurs les vendeurs d'[redacted] doivent déjà en savoir quelque chose...

Lesquels vendeurs d'[redacted] ne vont sans doute guère apprécier que nous vous donnions nos coordonnées - et que vous puissiez nous contacter à Victor Technologies - Tour Horizon, 52, quai de Dion-Bouton, 92800 Puteaux (tél. : 778.14.50) ; ou encore à Lyon : (7) 234.12.45 ; Montpellier : (67) 64.71.72 ; Nantes : (40) 89.24.28. Mais l'on ne peut contenter tout le monde et [redacted] !

** Configuration complète avec clavier et écran monochrome. Prix H.T. au 1/9/85. (Possibilité de location financière : 700 F par mois sur 48 mois - CEGEDATA.).*

VICTOR

Comme [redacted] moins cher qu'[redacted]

Alice.