hors série

Led MICRO

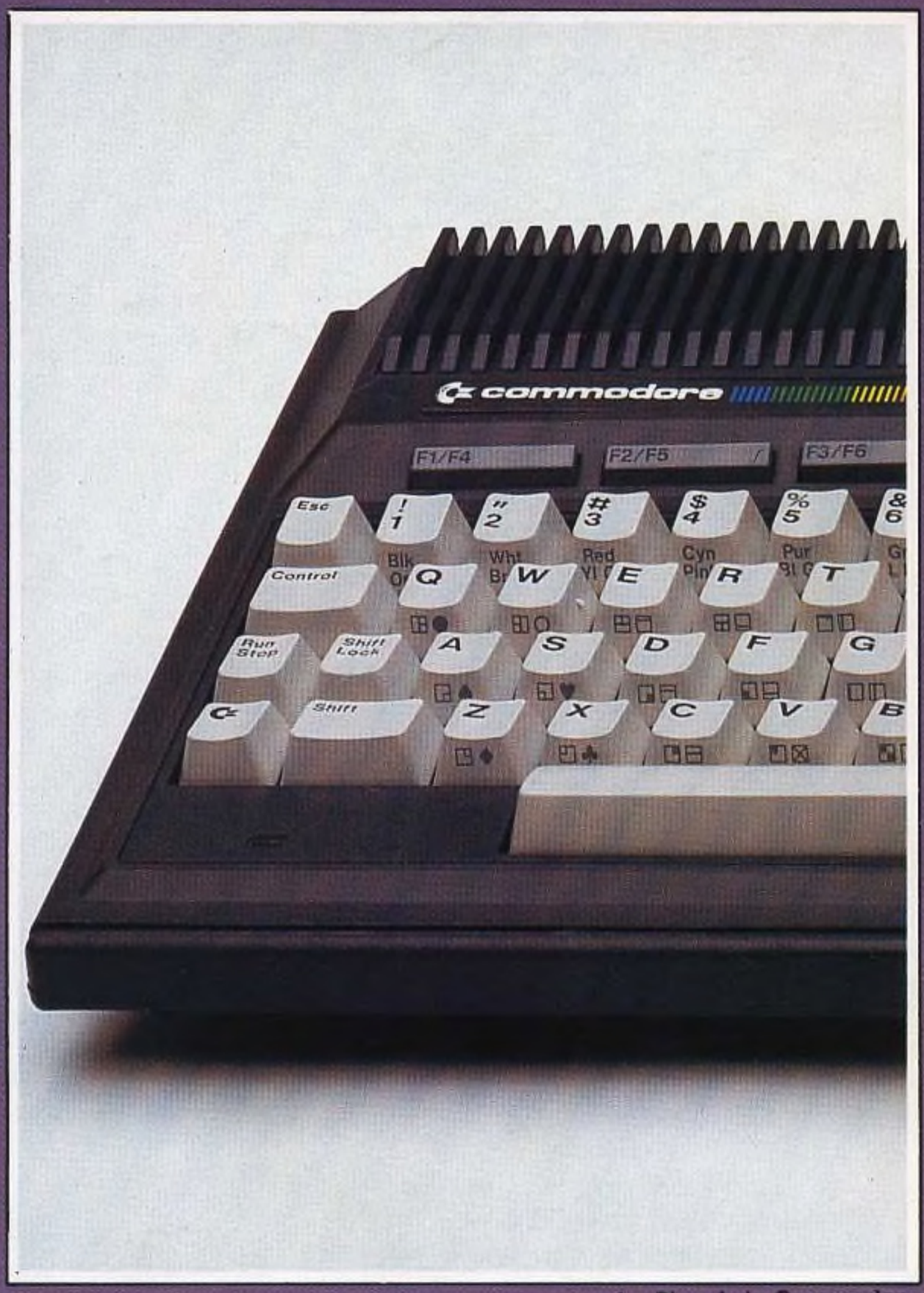
PROGRAMMATION
COURS 2ème CYCLE

COURS
N°24
Suite
2e cycle
N°4

COURS DE BASIC : les chaînes de caractères

COURS DE PROGRAM-MATION APPROFONDIE : les structures de données

COURS DE GENIE LOGICIEL : de la théorie à la pratique



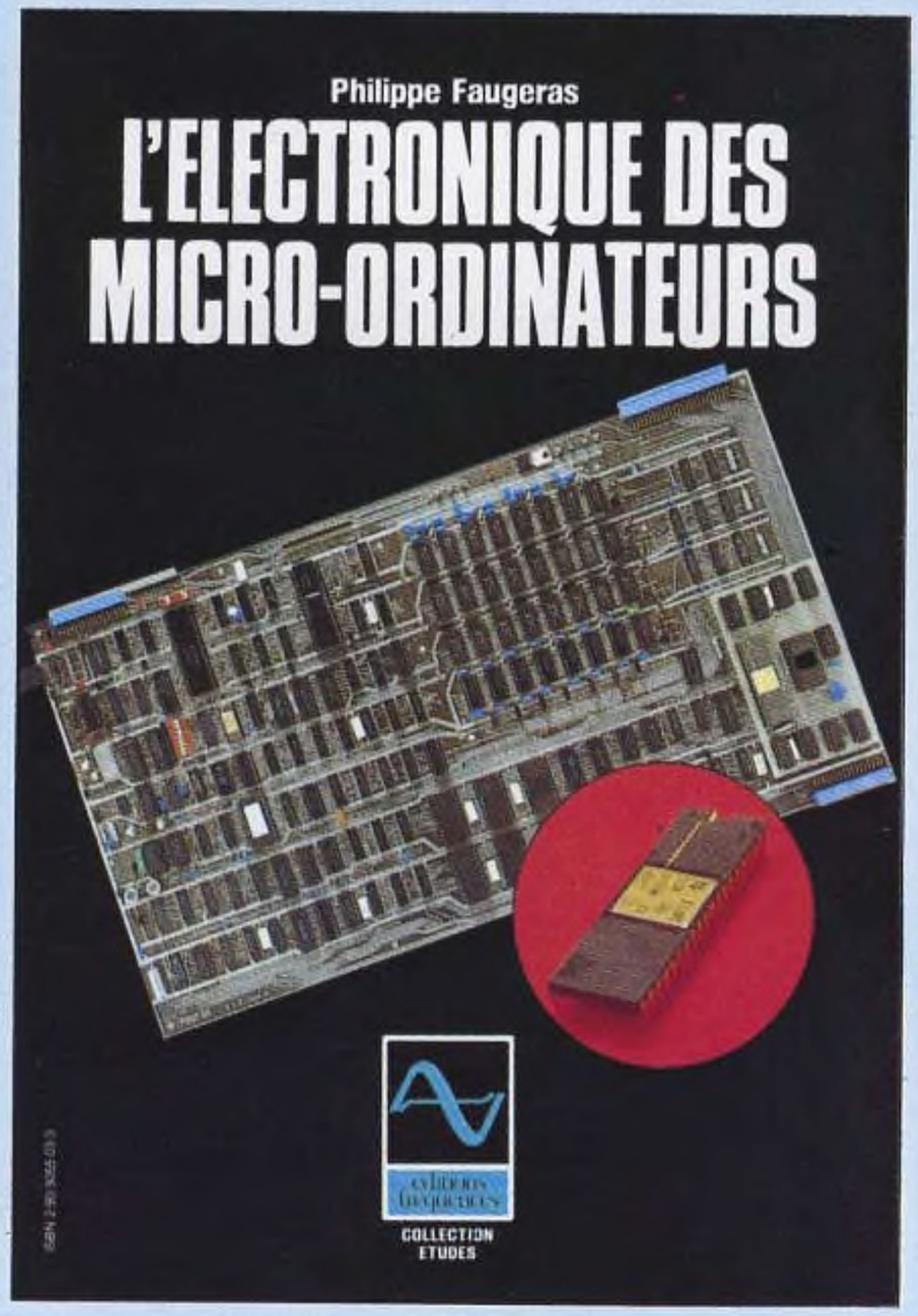
Le Plus 4 de Commodore

ISSN 0757-6889

M 1988 - N°24 (4) - 18 F MENSUEL NOVEMBRE 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

NOVEMBRE 85

Société éditrice :
Editions Fréquences
Siège social :
1, bd Ney, 75018 Paris
Tél. : (1) 46.07.01.97 +
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) 46.07.01.97
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

NOTRE COUVERTURE : Le Plus 4 importé par Commodore France 3, rue du Docteur Lancereau 75008 Paris. Tél. (1) 45.62.01.09.

COURS DE BASIC

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

1. Introduction

Comme convenu, nous allons ce mois-ci nous attaquer à des applications du cours précédent, c'est-à-dire des programmes utilisant extensivement des chaînes de caractères. A cette fin, nous vous proposons un programme de MOT LE PLUS LONG (ou faut-il envoyer les droits d'auteurs ?), une amélioration du programme de répertoire téléphonique débuté avec la série de cours et une simulation de commandes n'existant pas sur toutes les machines (le PRINT USING). D'autres commandes sont absentes sur certaines machines, si vous désirez les programmer, envoyez-nous vos solutions, nous les proposerons aux autres lecteurs.

2. Extension du programme de répertoire

Nous allons mettre au point un module qui permettra de retrouver une référence en ne donnant qu'une partie du nom ou du prénom, pour les pressés, ou ceux qui ont la mémoire un peu courte. Pour cela, un simple module va suffire, ce qui sera une preuve excellente de la supériorité de la programmation structurée.
Voici le listing, les commentaires suivront. Ceci pour que vous commenciez toujours par lire une fois un listing avant de vous attaquer au «bavardage» l'accompagnant afin de vous faire une première idée et de pouvoir aiguiser votre sens critique. En effet, le cheminement intellectuel inverse met le lecteur en position de «receveur universel», sans qu'il lui soit nécessaire de réfléchir, tout lui étant donné. L'expérience prouve que cette deuxième situation limite énormément la qualité de l'apprentissage. (Merci M. Piaget).

```
10 REM  LE VRAI PROGRAMME DEBUTE EN 10000
11 REM ********************************
20 GOTO 10000
30 REM AVANT LA LIGNE 10000, LES SOUS-PROGRAMMES
40 REM ****************************************
```

```
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
160 PRINT
199 RETURN



 300 REM SOUS-PROGRAMME D'AJOUT D'UNE REFERENCE
 301 REM ***************************************
 310 NBINF = NBINF + 1
 320 INPUT " NOM DU CORRESPONDANT      : ";NOM$ : INFO$(NBINF,1)=NOM$
 330 INPUT " PRENOM                    : ";PRENOM$ : INFO$(NBINF,2)=PRENOM$
 340 INPUT " NUMERO  : "; NUMERO$ : INFO$(NBINF,3)=NUMERO$
 399 RETURN


 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$(STOCK(1),1)=" " :
                  NEXT :
                  GOTO 499
 460 IF J>1 THEN INPUT "PRENOM DU CORRESPONDANT  : ";PR$
 470 FOR I=1 TO J :
         IF INFO$(STOCK(I),2)=PR$ THEN
                                FOR K=1 TO 3 :
                       INFO$(STOCK(I),K)=" " :
                NEXT :
     NEXT
 499 RETURN

 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 :
        FOR K= 1 TO 3:
            PRINT INFO$(STOCK(I),K);:
        NEXT K :
        PRINT:
    NEXT I
599 RETURN

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
1035 IF J=0 THEN GOSUB 2000
1099 RETURN
2000 REM RECHERCHE DE SOUS-CHAINE DANS LES NOMS ET LES PRENOMS
2001 REM ************************************************
2010 FOR I=1 TO NBINF
2020 IF LEN(INFO$(I,L))>=LEN(NOM$ THEN
          FOR K=1 TO LEN (INFO$(I,L))-LEN(NOM$)+1:
              IF MID$(INFO$(I,L),K,LEN(NOM$))=NOM$ THEN
                  J=J+1 :
                  STOCK(J)=I :
          NEXT K
2030 NEXT I
2040 IF J=0 AND L=1 THEN L=2: GOTO 2010
2099 RETURN

10000 REM DEBUT DU PROGRAMME "PRINCIPAL"
10010 REM ******************************
10020 DIM INFO$(20,3)
10030 DIM STOCK(10)
10040 NBINF =0

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 10100
20000 END
```

Comme vous pouvez le constater, le programme a un tout petit peu été modifié mais ces modifications ne sont pas nécessaires à l'exclusion de la ligne 470 dont le NEXT I doit passer à la ligne 475 car sinon, lorsque la condition du IF n'est pas vérifiée, le NEXT n'est pas rencontré par le programme. Désolé de ce type d'erreur due à une retranscription manuelle du programme dans le cours et qui ne pourra plus se reproduire à l'avenir, merci M. Apple pour votre MacIntoh (pour les pots-de-vins, nous les envoyer directement à la rédaction qui fera suivre) mais M. Apple, vous ne pourriez pas le faire un peu plus rapide, le MacIntosh ?
Par contre, nous avons ajouté une partie destinée à traquer les sous-chaînes dans les noms et les prénoms et la ligne 1035 qui sert à activer cette recherche lorsqu'elle est nécessaire, c'est-à-dire lorsqu'aucun nom n'a été trouvé par la voie habituelle.
Regardons maintenant comment fonctionne le module débutant en ligne 2000 :
– La variable L est destinée à savoir si la recherche est faite sur les noms ou les prénoms. On commence par les noms, mais on pourrait très bien commencer par les prénoms ou étudier le nom et le prénom d'une même référence avant de passer à la suivante. L = 1 correspond donc à une recherche sur les noms. La suite est donc un parcours de tous les noms en mémoire pour trouver ceux pour qui la chaîne fournie par l'utilisateur pourrait être une sous-chaîne. Si aucun nom n'est trouvé, la recherche repart sur les prénoms (L = 2) de la même façon.
Voici le schéma de cette partie :

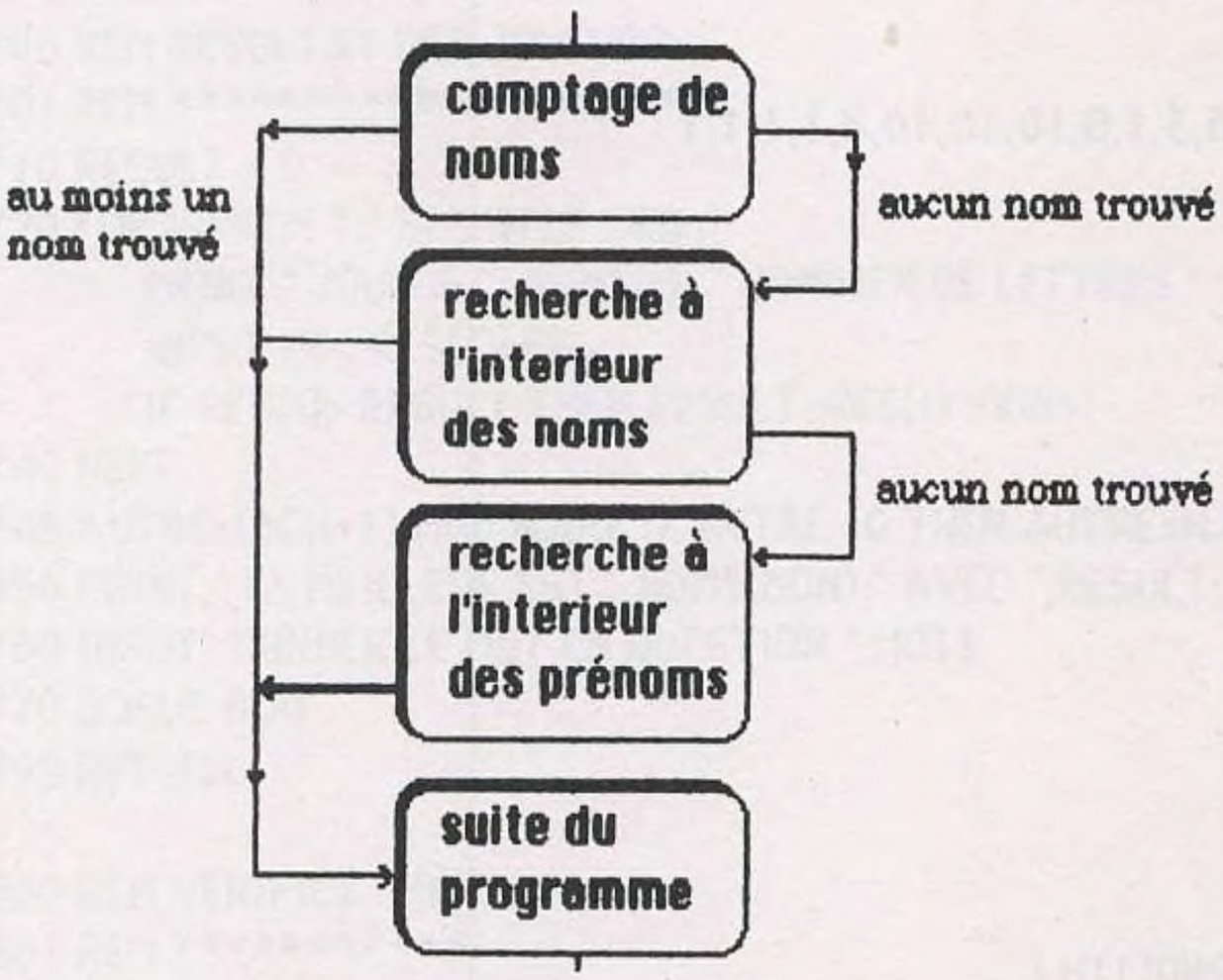


Schéma général de la recherche.

3. Un programme de MOT LE PLUS LONG

Ce programme est destiné à simuler une partie de ce jeu, populaire s'il en est, et très simple à réaliser, si certaines limitations sont acceptées. Dans un premier temps, nous lui ferons organiser le jeu, tirer les lettres et compter les points. Chaque partie du programme sera structurée en module(s) et des améliorations seront aisément ajoutées.

Tout de suite, nous voulons vous prévenir d'un détail : la commande RANDOMIZE TIMER est une commande de mise à 0 de la suite de nombres tirés par la commande RND. Elle n'est pas indispensable et certains systèmes ne l'ont pas. Dans ce cas, vous pouvez l'ignorer.

Voici ce programme, les commentaires suivent :

```
10 REM ************************************
20 REM le programme commence à la ligne10000
30 REM ************************************
40 GOSUB 10000

100 REM initialisation des variables
101 REM **************************
110 RESTORE : K=0
120 FOR I=A TO Z :
        READ LETTRE(I) :
        FOR J=1 TO LETTRE(I) :
          K=K+1:
          CHLETTRE$(K)=CHR$(64+I) :
        NEXT J :
    NEXT I
130 DATA 10,3,5,4,14,3,2,2,11,2,1,8,4,6,6,3,1,8,10,10,10,4,1,1,1,1
199 RETURN

200 REM SAISIE DES NOMS
201 REM ***************
210 DIM RES(NJOU) : DIM NOM$(NJOU)
220 FOR I=1 TO NJOU :
        PRINT " NOM DU JOUEUR "; I :
        INPUT NOM$ : NOM$(I)=NOM$ :
    NEXT
299 RETURN

300 REM TIRAGE DU PREMIER A JOUER
301 REM ************************
310 RANDOMIZE TIMER : PREM=INT( NJOU*RND(1))+1
320 PRINT " LE PREMIER A JOUER SERA " ;NOM$(PREM)
330 SEC=1 : ORD=-1
340 IF PREM=1 THEN SEC=2 : ORD=1
399 RETURN
```

```
400 REM TIRAGE ET AFFICHAGE DES LETTRES
401 REM ******************************
405 BLANC$=".": GOSUB 100
410 FOR I=1 TO TIRMAX :
        RANDOMIZE TIMER
420 TEMPO=INT ((TOTAL-1)*RND(1)+1) :
    TEMPO$=CHLETTRE$(TEMPO) :
    IF TEMPO$=BLANC$ THEN 420
430 TIRLETR$(I)=TEMPO$ : CHLETTRE$(TEMPO)=" "
440 NEXT
450 PRINT " LES LETTRES SONT  :  ";:
    FOR I=1 TO TIRMAX :
      PRINT TIRLETR$(I) ;" "; :
    NEXT :
    PRINT
499 RETURN

600 REM BOUCLE D'ATTENTE
601 REM *****************
610 FOR I=1 TO LONGATTEN : NEXT
699 RETURN

700 REM RESULTAT DES JOUEURS
701 REM ********************
710 RESULT = 0
720 FOR I=PREM TO SEC STEP ORD :
        PRINT " JOUEUR "; NOM$(I); " COMBIEN DE LETTRES " :
        INPUT RE : RES(I)=RE
        IF RES(I)> RESULT THEN RESULT=RES(I) :BON=I
740 NEXT
745 AUTRE=(BON+1) MOD NJOU: IF AUTRE =0 THEN AUTRE=NJOU
750 PRINT " LE MEILLEUR EST ";NOM$(BON);" AVEC ";RESULT;" LETTRES"
760 INPUT" DONNER LE MOT EN QUESTION ";MOT$
770 GOSUB 800
799 RETURN

800 REM VERIFICATION
801 REM ************
810 NL=0: OK=0

820 IF LEN(MOT$) <> RESULT THEN PRINT " ERREUR DE LONGUEUR " :
                                GOTO 899
```

```
830 FOR I=1 TO LEN ( MOT$)
840 FOR J=1 TO TIRMAX
850 IF MID$(MOT$,I,1)=TIRLETR$(J) THEN NL=NL+1:
                                       TIRLETR$(J)=" " : J=TIRMAX
860 NEXT J
870 NEXT I
880 IF NL=RESULT THEN OK=1
899 RETURN

900 REM TRAITEMENT DES SCORES
901 REM **********************
910 IF OK=1 THEN SCORE(BON)=SCORE(BON)+RESULT : GOTO 930
920 IF OK=0 THEN SCORE(AUTRE)=SCORE(AUTRE)+RESULT:
          PRINT" ERREUR DE MOT. LES POINTS VONT A ";NOM$(AUTRE)
930 PRINT " VOICI LES NOUVEAUX SCORES"
940 FOR I=1 TO NJOU :
      PRINT " JOUEUR "; NOM$(I);" : ";SCORE(I) ;" POINTS" :
    NEXT
950 FOR I=1 TO 1000 : NEXT
999 RETURN

1000 REM BOUCLAGE SUR LES PARTIES
1001 REM ************************
1010 PRINT" ENCORE UN TIRAGE ? "
1020 INPUT" OUI/NON";A$
1030 IF LEFT$(A$,1)<>"N" THEN OK=1
1099 RETURN

10000 REM DEBUT DU PROGRAMME PRINCIPAL
10001 REM ******************************
10100 A=1 : Z=26 : TIRMAX=8 : TOTAL=131 : LONGATTEN=10000 : NJOU=2:
      DIM LETTRE(Z) : DIM CHLETTRE$(TOTAL)
10200 GOSUB 200
10300 GOSUB 300
10400 GOSUB 400
10600 GOSUB 600
10700 GOSUB 700
10900 GOSUB 900
11000 GOSUB 1000
11100 IF OK=1 THEN TT=PREM :
                   PREM = SEC :
                   SEC=TT :
                   ORD=-ORD :
```

```
GOTO 10400

11200 END
```

Le programme est relativement simple, et des améliorations seront les bienvenues, aussi, si vous en implantez, envoyez-les, pour en faire profiter les autres lecteurs.
Au début (lignes 100 et suivantes), on trouve les initialisations indispensables et les paramètres de base :
- TIRMAX est le nombre de lettres tirées à chaque fois,
- NJOU est le nombre de joueurs, mis ici à deux pour refléter une vraie partie,
- LETTRE est le tableau où est stocké le nombre de fois que chaque lettre de l'alphabet peut apparaître. C'est le nombre de carte(s) de cette lettre lu en ligne 130,
- CHLETTRE$ est le tableau de toutes ces cartes,
- TIRLETR$ est le tableau des lettres tirées (celles avec lesquelles il faut jouer),
- LONGATTEN permet de régler la durée du temps de réflexion à chaque tirage,
- RES permet de stocker les scores.

Puis il vous sera demandé le nom de chaque joueur pour le bon déroulement de la partie. Les lettres sont ensuite tirées (sans séparation voyelles-consonnes, une possibilité d'amélioration !!) puis avec une boucle d'attente, le nombre de lettres de chacun est demandé et le mot est vérifié en ce qui concerne l'utilisation des lettres car la bonne orthographe du mot ne peut pas être vérifiée de par l'absence de dictionnaire. A ce sujet, voir le commentaire un peu plus loin. Enfin, les scores sont mis à jour et la partie peut se poursuivre.

Un point à souligner en ce qui concerne le premier à jouer. Si le premier est le joueur dont le nom a été donné en premier, pas de problème, pour le premier tour en tout cas. Par contre, dès que l'inverse est rencontré, il faut considérer l'ordre inverse. Pour cela, l'ordre de parcours des tableaux de noms de joueurs et de scores est défini par une variable ORD, dont la valeur est 1 ou − 1 afin de partir du premier nom pour finir par le deuxième (ORD = 1) ou de faire l'inverse (ORD = − 1). En fonction de cela, le premier à jouer est le premier nommé (PREM = 1, SEC = 2) ou le contraire. L'inversion est réalisée en ligne 11100.
Le stockage des lettres est effectué à chaque tour, car lorsqu'une lettre est tirée, sa case est remplie par un blanc, afin d'éviter de la tirer plusieurs fois. Pour cela, la ligne de DATA est lue à chaque fois, d'où la commande RESTORE, qui autorise la relecture d'un même DATA. Le tirage des lettres est réalisé par le biais d'un nombre aléatoire, compris entre 1 et TOTAL, et calculé en ligne 420.
Au sujet de la correction des mots, seul un dictionnaire en mémoire permettrait de vérifier la validité d'une proposition. Comme ceci n'est pas envisageable, il faut procéder par vérification manuelle, hélas. Par contre, ce qui est réalisable, c'est de rentrer dans un fichier (nous verrons comment les mois prochains) un certain nombre de mots de huit lettres (ou plus) de l'alphabet. Ainsi, il suffira de proposer en ordre bouleversé les lettres de ce mot. Ceci augmente le «challenge», car vous saurez qu'un mot de longueur maximale existe.

4. Un programme autorisant le formatage de type PRINT USING

Le PRINT USING est une commande extrêmement performante du BASIC. Hélas, certains BASIC ne le proposent pas. Aussi, il nous a paru intéressant de réaliser un programme vous en donnant la plupart des possibilités.
Voici une brève description de l'utilisation du PRINT USING :
- possibilité de fixer la position du point décimal, afin de présenter des colonnes de chiffres bien alignées,
- possibilité de choisir le nombre de chiffres avant et après la virgule. Ceci implique aussi quelques contraintes à connaître, comme par exemple la troncature de nombres trop grands,

– possibilité de remplir d'éventuels vides par un symbole particulier (cela servait à l'origine à écrire des chèques automatiquement, sans qu'il soit possible de rajouter des chiffres par la suite).

Le programme qui suit permet de simuler ceci.

```
10 REM LE PROGRAMME COMMENCE EN 10000
11 REM **********************************
20 GOTO 1000



100 REM CALCUL DE LA PARTIE ENTIERE (1ERE PARTIE )
101 REM ****************************************
110 IF ENT<=10^A+1 THEN  NENT=ENT :RETURN
120 FOR I=1 TO A :
              ENT=ENT/10 :
              NENT=NENT+(ENT-INT(ENT))*10^(I) :
              ENT=INT(ENT):
    NEXT
149 RETURN



150 REM CALCUL DE LA PARTIE ENTIERE (2EME PARTIE )
151 REM ****************************************
155 NNENT=NENT
160 FOR ESP=1 TO A :


        NNENT=NNENT/10 :
        IF INT(NNENT)=0 THEN 199
170 NEXT
199 RETURN



200 REM CALCUL DE LA PARTIE DECIMALE
201 REM ****************************
210 NDEC=INT(DEC* 10^B)/10^B
220 NDEC=NDEC*10^B
299 RETURN
```

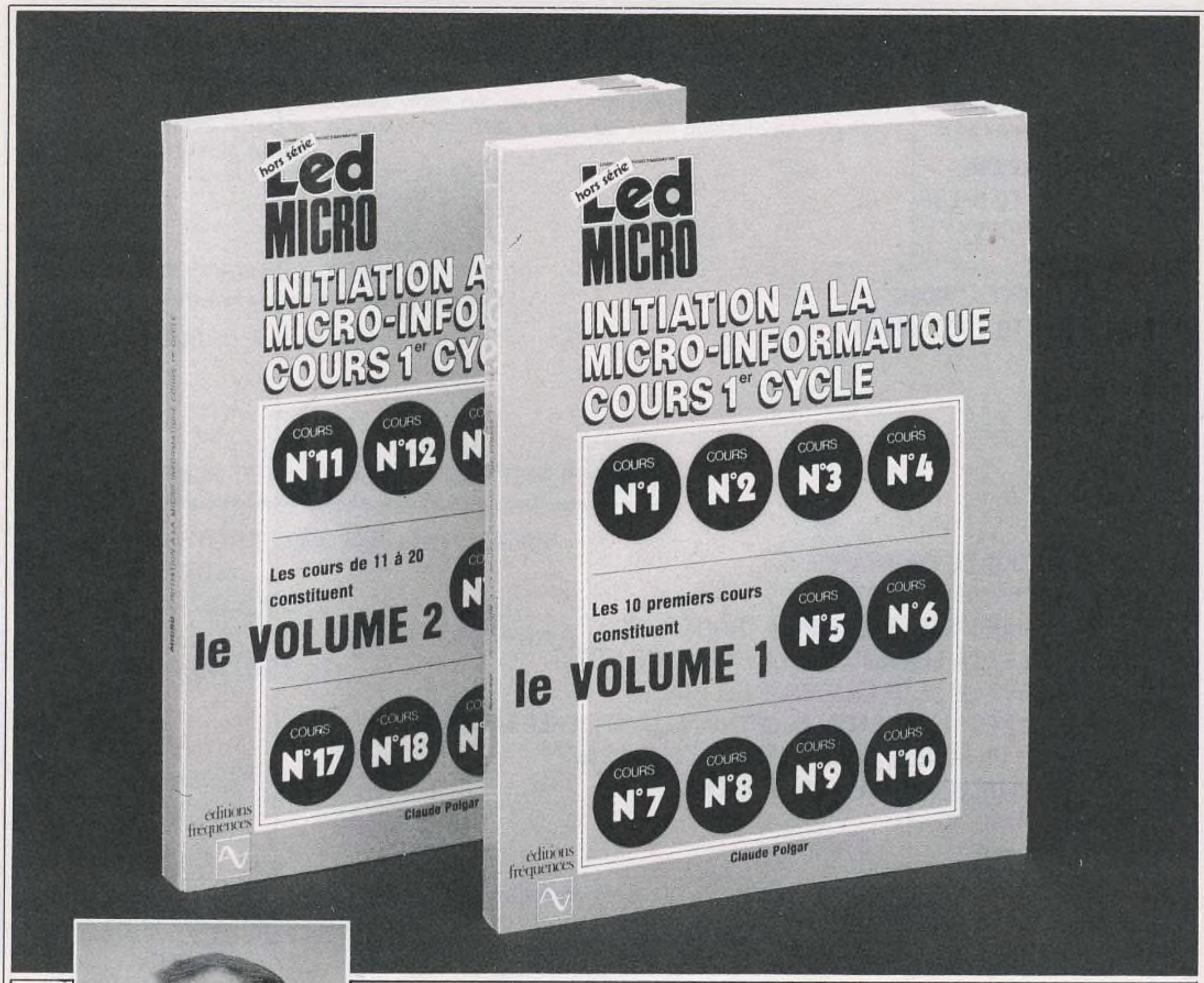
```
300 REM AFFICHAGE
301 REM **********
305 PRINT NOMBRE
310 FOR J=1 TO 5-ESP:
        PRINT SP$;:
    NEXT
320 PRINT NENT;".";NDEC:
330 FOR I=1 TO 10000: NEXT

399 RETURN
```

```
1000 REM DEBUT DU PROGRAMME
1001 REM ********************
1010 REM NOMBRE EST LE NOMBRE A TRAITER
1020 REM A EST LA LONGUEUR DE LA PARTIE ENTIERE
1030 REM B EST CELLE DE LA PARTIE DECIMALE
1040 REM SP$ EST LE CARACTERE OPTIONNEL A AFFICHER EN TETE
1050 INPUT " UN NOMBRE";NOMBRE# : A=5 : B=3: SP$="#"
1060 ENT=INT(NOMBRE) : DEC=NOMBRE-ENT
1070 GOSUB 100
1080 GOSUB 150
1090 GOSUB 200
1100 GOSUB 300
1110 GOTO 1050
```

La première partie permet de définir une éventuelle troncature du nombre, s'il est trop grand par rapport au format proposé. Faites attention à cette troncature qui peut être indésirable. A vous de bien dimensionner les paramètres. La seconde partie permet de remplir avec un symbole particulier si ce symbole a été fourni (à ce sujet, la ligne 1050 n'est qu'un exemple et doit être supprimée pour une exécution réelle). La troisième partie sert à réaliser la troncature sur la partie décimale.

5. Conclusion

Voici un tour d'horizon en deux cours des commandes portant sur les chaînes de caractères. A partir du mois prochain, nous nous intéresserons à la dernière partie du cours de BASIC, les fichiers et les techniques de stockage.
Si un point vous a chagriné, il est temps de nous écrire car nous reprendrons les parties que vous nous aurez demandé de revoir tout de suite après la prochaine série de cours, qui s'étalera sur trois ou quatre numéros.



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 ou 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.

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 abordons ce mois-ci un cours qui s'étalera sur deux mois consécutivement à sa longueur et au contenu particulièrement important pour la suite de vos programmes. Son sujet : les Structures de Données.
Il ne sera exposé ce mois-ci que les plus simples, à savoir, d'une part celles existant en BASIC, et d'autre part celles assez faciles à simuler dans ce langage. Pour ces dernières, nous vous proposerons une manière de les implanter sur votre machine.
Enfin, après ce cours que nous espérons le plus clair possible, nous vous présenterons de nouveaux sujets d'exercices, sans commentaires mais patience, en décembre, nous vous laisserons la parole.

COURS N° 4

Les structures de données - Première partie

PLAN DU COURS

1. PRESENTATION DES STRUCTURES

1.1. Introduction

Même si vous avez une idée du cours qui va suivre, nous pensons que vous ne serez pas mécontents que nous précisions notre propos. Lorsque vous faites un agenda sur votre ordinateur, pour chaque personne que vous souhaitez voir figurer dans ce répertoire, vous rassemblez divers renseignements qui, chacun, respecte des règles somme toute précises. Exemple : le nom, le (ou les) prénom (s), l'adresse (qui comporte un numéro, un nom de rue, un code postal (nombre compris entre 01000 et 99999), une ville au minimum), le numéro de téléphone. Vous souhaitez ensuite retrouver l'ensemble de ces informations simplement en faisant mention du nom.

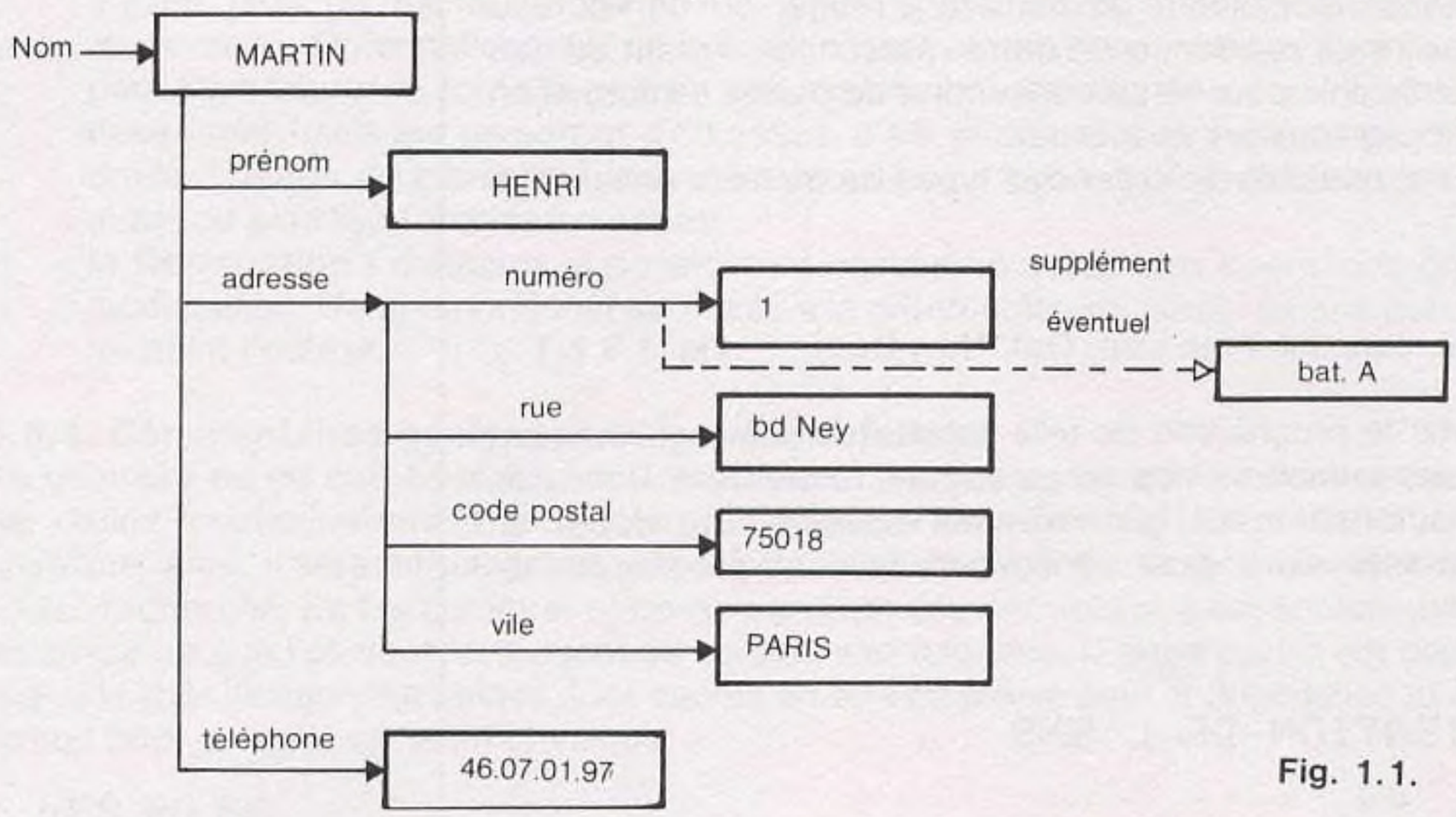


Fig. 1.1.

Cet ensemble d'informations est «structuré» comme nous le voyons ci-dessus du fait de l'existence pour chaque personne de l'agenda de chacun des renseignements énumérés. Il existe des langages où l'élaboration de telles structures est facilitée par l'existence de modèles variables de données à l'intérieur même du langage. Mais ce travail, qui a été fait par les concepteurs de ces langages n'existant pas en BASIC, nous devons le réaliser à partir de ce qui est présent dans le BASIC.

1.2. Ce qui existe en BASIC

En faisant le tour du propriétaire des outils mis à notre disposition par ce langage, nous ne ferons qu'un constat de pauvreté flagrante. Toutefois, le désespoir ne doit point nous envahir car tout peut être créé à partir de ce désert :

- les ENTIERS que l'on peut, si l'on en a besoin, réduire aux positifs,
- les REELS,
- les TEXTES ou variables caractères que le BASIC sait très bien manipuler,
- les LOGIQUES : variables valant en BASIC 0 ou 1 selon qu'elles sont fausses ou vraies.

Il est à noter que l'on peut assimiler les TEXTES à une structure déjà existante de variables manipulant des chaînes de variables élémentaires de type caractère.

1.3. Description des structures de données

Avant de voir dans le détail chacune des structures que nous serons amenés à construire pour la suite, il convient de définir ce que cela représente à plusieurs niveaux.

1.3.1. Représentation physique

Ce niveau est celui qui sera le plus décortiqué pour chacun des types que nous allons concevoir. En effet, c'est à ce stade que l'on réalise l'implantation des structures avec les moyens du bord et que l'on visualise ce qui se passe réellement lorsque l'on procède à une opération de niveau supérieur.

1.3.2. Description logique

Ici est établie la décomposition de toute la structure telle que nous l'avons défrichée lors de l'introduction. Cela peut être plus sophistiqué que le premier exemple ne le laisse supposer, si l'on a recours à des structures conditionnelles. Dans l'agenda, il est possible que certaines adresses nécessitent un numéro d'étage, ou un «bureau distributeur» différent de la localité de résidence de notre «personne». Autant de cas où notre structure se doit d'être flexible pour ne pas engendrer de pertes d'information ou de mémoire (ce qui ne peut pas toujours être évité).
Dans la description logique, il est possible de créer des types de manière exhaustive comme les mois ou les jours.

Mois : (Jan, Fev, Mar, Avr, Mai, Jun, Jul, Aou, Sep, Oct, Nov, Dec) **Fig. 1.3.2.1**

Il conviendra alors de concevoir le programme de telle façon que soit régulièrement vérifiée la concordance entre les variables et les types qu'elles respectent. Dans ces cas-là, la liste des valeurs autorisées est généralement spécifiée en début de programme de la manière suivante avec, pour vérification, le sous-programme ci-dessous.

```
100  REM  INITIALISATION DE L'ENS
     EMBLE
110  DIM MOIS$(12)
120  DATA  JAN,FEV,MAR,AVR,MAI,JU
     N,JUL,AOU,SEP,OCT,NOV,DEC
130  FOR I = 1 TO 12
140  READ MOIS$(I): NEXT I
150  REM   FIN INITIALISATION



200  REM  VERIFICATION
210  REM  LECT$ EST LA VARIABLE A
      VERIFIER
220  FOR I = 1 TO 12
230  IF MOIS$(I) = LECT$ THEN  PRINT
     "VALABLE": GOTO 260
240  NEXT I
250  PRINT "INCONNU DANS L'ENSEMB
     LE"
260  REM  FIN VERIFICATION
```

Fig. 1.3.2.2

1.3.3. Spécification fonctionnelle

Pour manipuler la structure choisie, il sera nécessaire de faire appel à des outils prédéfinis. Ces outils se rangent en trois catégories :

- **la Création :** ce sont les opérations qui ont pour but de créer un nouveau type de structure. Nous trouvons, pour l'exemple que nous décrivons depuis le début, la création de l'agenda lui-même avec l'ensemble des conditions que doivent respecter les divers paramètres de sa structure.
- **la Modification :** ce n'est pas le titre cher à Michel Butor, mais l'ensemble des opérations visant à corriger un élément du type donné, que ce soit à la suite d'une première saisie malheureuse ou pour toute autre raison, ou encore, plus globalement, la création d'un nouvel élément du type de la structure. Pour l'agenda, cela concerne autant la saisie d'un nouvel élément que la modification d'une donnée déjà implantée dans l'ensemble que constitue la structure depuis la renumérotation téléphonique jusqu'au déménagement.
- **l'Accès :** ici, sont rassemblées toutes les fonctions de consultation avec des sorties plus ou moins consistantes. Nous appliquerons à cet endroit les recherches d'informations sur un individu à partir de son nom. A noter que l'on peut faire figurer ici toute la gestion des impressions de données. Il est parfois nécessaire (mais sur un carnet d'adresses, c'est plutôt rare) de faire usage de droits d'accès de manière à limiter les facultés de consultation d'autrui, cela aussi ce sont les fonctions d'accès.
- **la Destruction :** d'aucuns la considèrent comme un avatar des opérations de modification. Nous tenons tout de même à la différencier, ne serait-ce que pour ne point l'oublier.

1.3.4. Commentaires généraux sur les structurations

Au contraire de ce que nous vous avons présenté, il est plus logique de commencer par définir fonctionnellement la structure de données pour finir par la représentation physique. Ainsi, il sera nettement plus facile de choisir le type s'adaptant exactement au but recherché. En l'occurence, et comme en bien des occasions, il est souhaitable de savoir ce que l'on veut faire avant de commencer à le faire. D'autre part, il est clair que si la spécification est unique, il ne saurait en être de même pour la description et a fortiori pour la représentation physique.

2. LES PILES

Chacun d'entre nous connaît cette notion en usage dans plus d'un lieu administratif : la pile. Le dossier se trouvant sur le dessus est le dernier arrivé mais aussi le premier dont on se préoccupe. On imagine facilement qu'en période de surcharge, les premiers arrivés auront bien du mal à refaire surface. En anglais technique, cela s'appelle LIFO (Last In First Out - Dernier Entré Premier Sorti). On conçoit aisément que l'on ne peut traiter que le dossier supérieur sous peine de faire écrouler l'édifice.

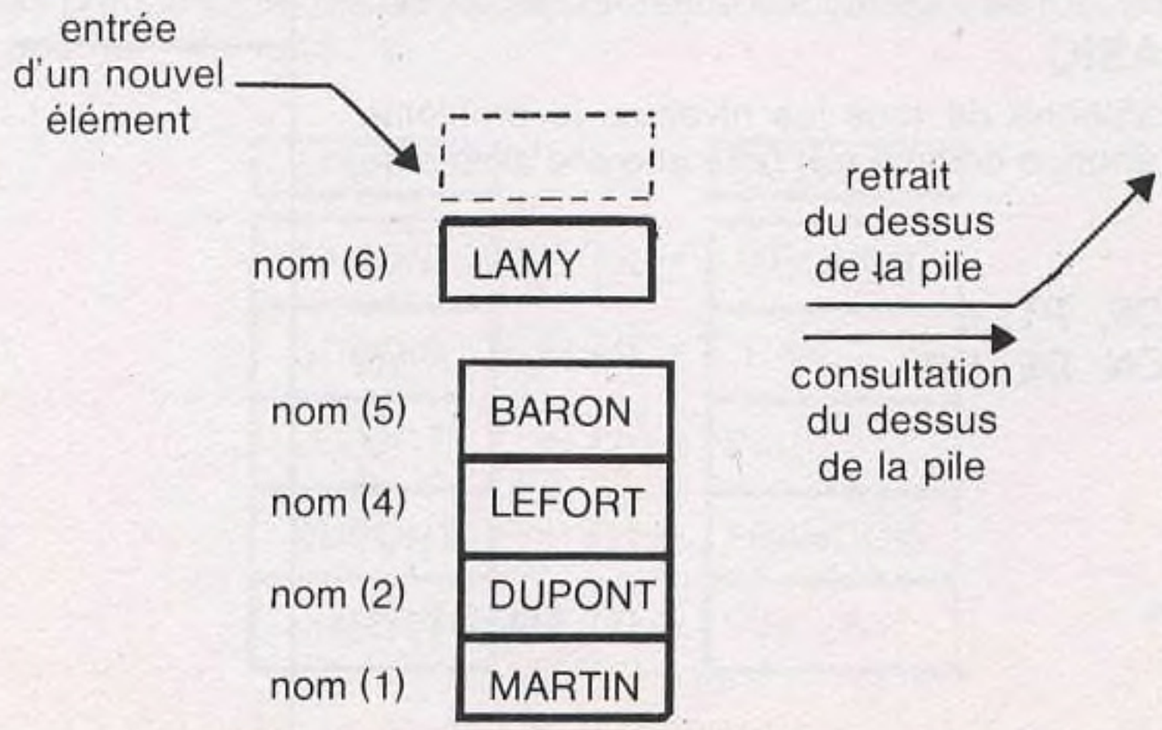


Fig. 2.0

2.1. Elaboration de la Pile à tous les niveaux

2.1.1. Du point de vue de la spécification fonctionnelle, on retrouve les opérations que l'on vient de déterminer comme nécessaires, plus quelques autres de même urgence :

- en création, la création de la pile même qui est initialisée à vide.
- en accès, le test sur l'état de la pile et la consultation du dernier élément entré.
- en modification, l'empilage d'un nouvel élément dans la pile ou au contraire le dépilage.

2.1.2. La description logique de la pile n'amène dans la structure qu'à mettre en exergue le rôle particulier joué par l'élément supérieur ; on vient à considérer la structure sous la juxtaposition de deux parties : le haut de la pile comprenant un seul objet, et le corps de la pile englobant tout le reste.

Nous déduisons à la simple observation du schéma précédent, que cette structure ne demande qu'un seul point de lecture/écriture, à savoir l'élément situé en haut de la pile. Pour obtenir un élément plus «enfoui» il faudra commencer par «dépiler» les objets le recouvrant. A cet égard, on rencontre le problème de fond de pile, car il est indispensable de vérifier s'il existe encore des éléments dans la pile avant de retirer éventuellement l'un d'entre eux.
On obtient ainsi un faisceau de propriétés qu'une pile doit être en mesure d'accomplir :

- ajout d'un nouvel élément,
- vérification de l'état de la pile (vide ou non)
- consultation de l'élément situé sur le dessus
- suppression de cet élément

ces deux dernières étant naturellement dépendantes d'une réponse positive à une vérification préalable de l'état de la pile.

2.1.3. Le langage imposé étant le BASIC, la façon la plus simple qui vient à l'esprit pour simuler cette structure consiste en un tableau prédéfini avec un indice représentant le nombre d'éléments actuellement dans la pile. Les faiblesses d'une telle politique résident dans l'obligation de déterminer à l'avance le nombre maximum d'éléments que la pile sera en mesure d'accueillir ; on risque ainsi deux écueils, soit la taille de la pile sera trop étriquée provoquant, si cela a été prévu, non pas une sortie en erreur mais une perte de l'information, soit la pile sera d'une grandeur démesurée pour l'usage que l'on en fera, réservant abusivement de la mémoire que l'on ne pourra employer par ailleurs. Ici encore, comme partout en informatique, ce n'est pas mauvais de soupçonner ce que va faire votre programme. Sans être un remède miracle, vous pourrez atténuer les effets pervers de cette réservation anticipée en utilisant un dimensionnement variable de votre pile, aussi en cas de malheur, il ne vous restera plus qu'à modifier un paramètre au début de votre listing pour réparer habilement cette interruption.

2.2. Implantation de la Pile en BASIC

Une fois rédigées correctement les descriptions de tous les niveaux, le problème paraît nettement plus limpide, la création s'énonce comme suit pour une pile simple de noms :

```
100  REM  INITIALISATION DE PILE
110 N = 100: REM  DIMENSION DE LA
      PILE
120  DIM PILE$(N)
130 I = 0
140  FOR J = 1 TO N
150 PILE$(J) = "VIDE"
160  NEXT J
```

Fig. 2.2.1

la vérification de l'état de la pile consiste en un test :

```
200  REM  TEST D'ETAT DE LA PILE
210  IF I = 0 THEN  PRINT "PILE V
     IDE": GOTO 230
220  PRINT "PILE REMPLIE"
230  REM  FIN TEST
```

Fig. 2.2.2

la consultation du haut de la pile se résume à :

```
300  REM  CONSULTATION DU SOMMET
310  PRINT PILE$(I)
```

Fig. 2.2.3

l'empilage et son opposé le dépilage se bornent à des expressions pour le moins sommaires :

```
400  REM  EMPILAGE
410 I = I + 1
420  INPUT "NOUVEL ELEMENT ";PILE
     $(I)
```

Fig. 2.2.4

```
500  REM  DEPILAGE
510 PILE$(I) = "VIDE"
520 I = I - 1
```

Fig. 2.2.5

à noter que ces instructions sont directement intéractives avec le clavier ou l'écran, il est certain qu'une utilisation dans un programme compliquerait quelque peu ces procédures mais elles resteraient manipulables sans grande difficulté, on remplace alors les INPUT et PRINT par des assignations.

2.3. Structures parallèles en Pile

Pour une structure de type plus complexe dans ce que doit conserver chaque élément de la pile, on a recours à autant de tableaux ayant pour taille maximale celle spécifiée pour la pile, chaque tableau représentant un des morceaux élémentaires constituant l'élément lui-même ; pour la personne de l'agenda, nous aurions ainsi un tableau des noms, un des prénoms, un des adresses (lui-même décomposé en rue, code postal et ville) et un pour le téléphone.

Fig. 2.3

3. LES FILES

Il n'est pas besoin de grands discours pour vous présenter les files. L'informatisation de la Sécurité Sociale est trop récente pour que vous ayez oublié un de ses charmes les plus célèbres. Toutefois, et cela ne souffre ici aucune exception, il n'y a pas de resquilleur et l'adage Premier Entré Premier Sorti prend toute sa valeur (en anglais FIFO pour First In First Out). Outre votre usage personnel, les files d'attente ont en informatique un rôle propre qui leur est assigné ; en effet, elles ont une utilité intrinsèque en matière de simulation. A chaque file d'attente correspond un service espéré par les membres de la queue, aussi le temps nécessaire à l'accomplissement de ce service ajouté à la fréquence d'arrivée de tout nouveau «client» dans la file permettent d'établir avec précision les comportements de la dite file. Les simulations de tels événements ont autant d'applications que vous pouvez l'imaginer depuis le nombre de caisses à ouvrir dans tout supermarché à la gestion même des gros ordinateurs multitaches, afin d'exploiter au mieux les ressources allouées.
La file se présente sous une forme familière :

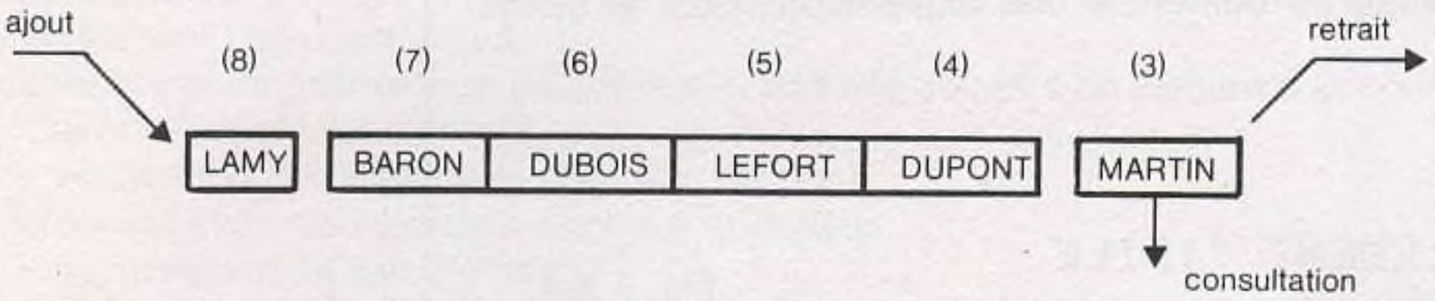


Fig. 3.0

3.1. Elaboration de la File à tous les niveaux

3.1.1. La spécification fonctionnelle de la pile nous a mâché grandement le travail puisqu'on retrouve ici des opérations analogues à celles que la structure précédente se devait de mener à bien. Nous réemployons la création, la vérification de l'état de la file, l'insertion d'un nouvel élément, seules la consultation et la destruction sont modifiées : ce n'est plus le dernier élément entré qui est concerné par ces procédures mais le plus ancien élément encore présent à l'intérieur de la structure.

3.1.2. Encore une fois, la spécification a avancé sensiblement le travail, on entrevoit clairement les trois parties qui vont servir d'ossature à l'édifice, à savoir la tête ou premier élément, le corps, et la queue comprenant uniquement le dernier élément.

3.1.3. Nous abordons maintenant un problème qui, sans être épineux, n'en est pas moins délicat. L'emploi de tableaux avec utilisation de l'indice pour pointer l'élément désiré est encore la solution, avec la difficulté de gérer cette fois-ci deux indices, l'un désignant la tête et l'autre la queue.
Chaque fois que l'on crée un élément, on augmente l'indice de queue, chaque fois que l'on détruit, on incrémente celui de tête et nulle on décrémente l'un ou l'autre des indices. Ainsi, quelle que soit la taille du tableau, inéluctablement on atteindra les limites de celui-ci avec, à l'opposé du cas de la pile de la place certainement libérée par les premiers éléments détruits dans la file. Comment alors récupérer cette place laissée vacante ? Trois solutions plus ou moins faciles à programmer ou astucieuses :

- chaque fois que l'on défile un élément, plutôt que d'incrémenter un indice de tête, redescrendre tous les éléments d'un cran. Il n'est besoin de vous souligner la lourdeur du procédé sur une file de longueur importante.
- on laisse monter la file tant que possible et seulement arrivée à sa saturation en haut, on redescend l'ensemble de l'édifice d'autant de rangs qu'il y a déjà eu d'éléments supprimés. L'économie est appréciable mais ces transferts sont autant d'opérations gratuites.

– lorsque la queue atteint le sommet de l'espace alloué, on continue à remplir en repartant par le bas ce qui donne dans notre esprit une impression différente sur la réelle disposition des éléments dans la file.

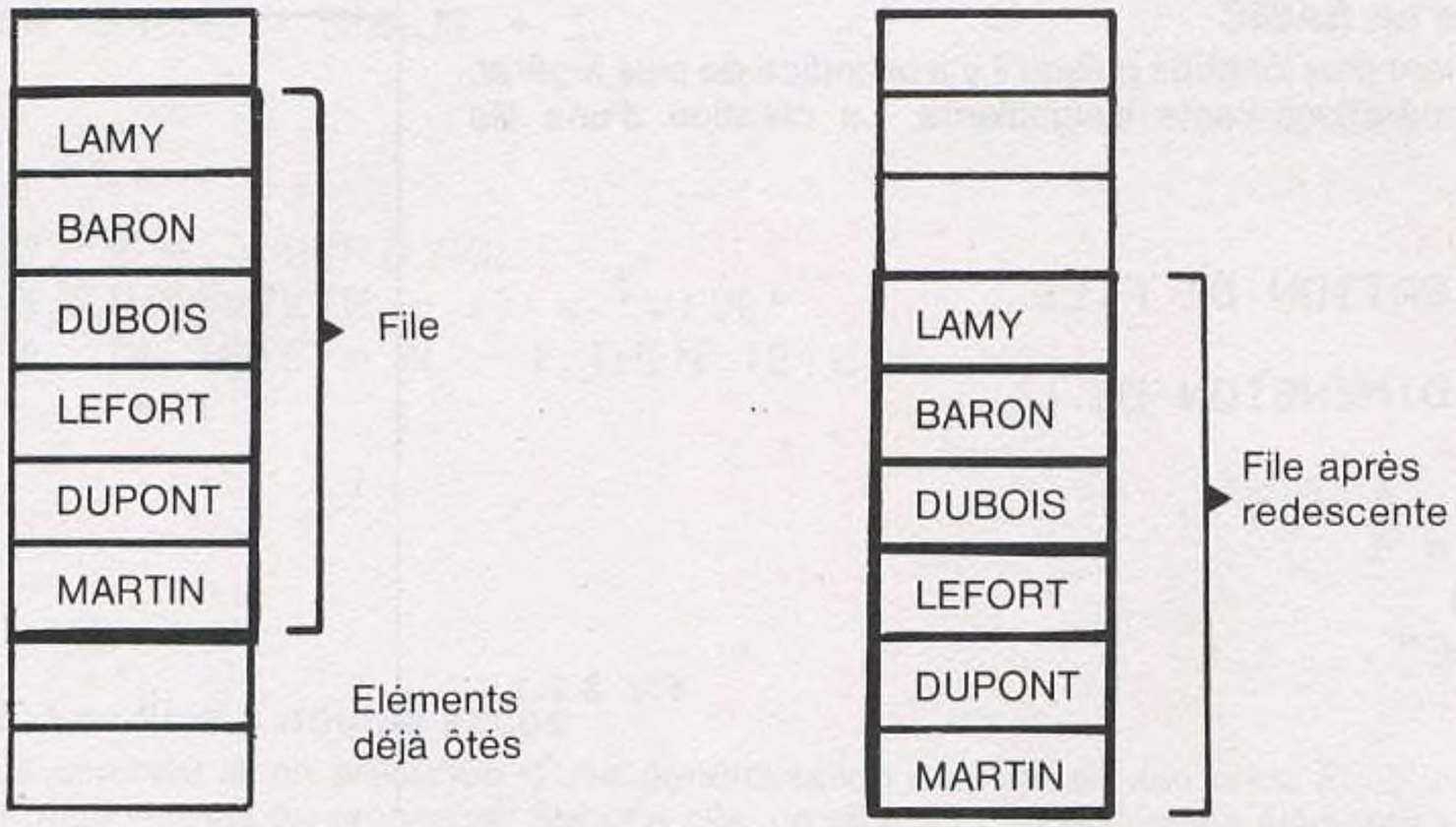


Fig. 3.1.3.1

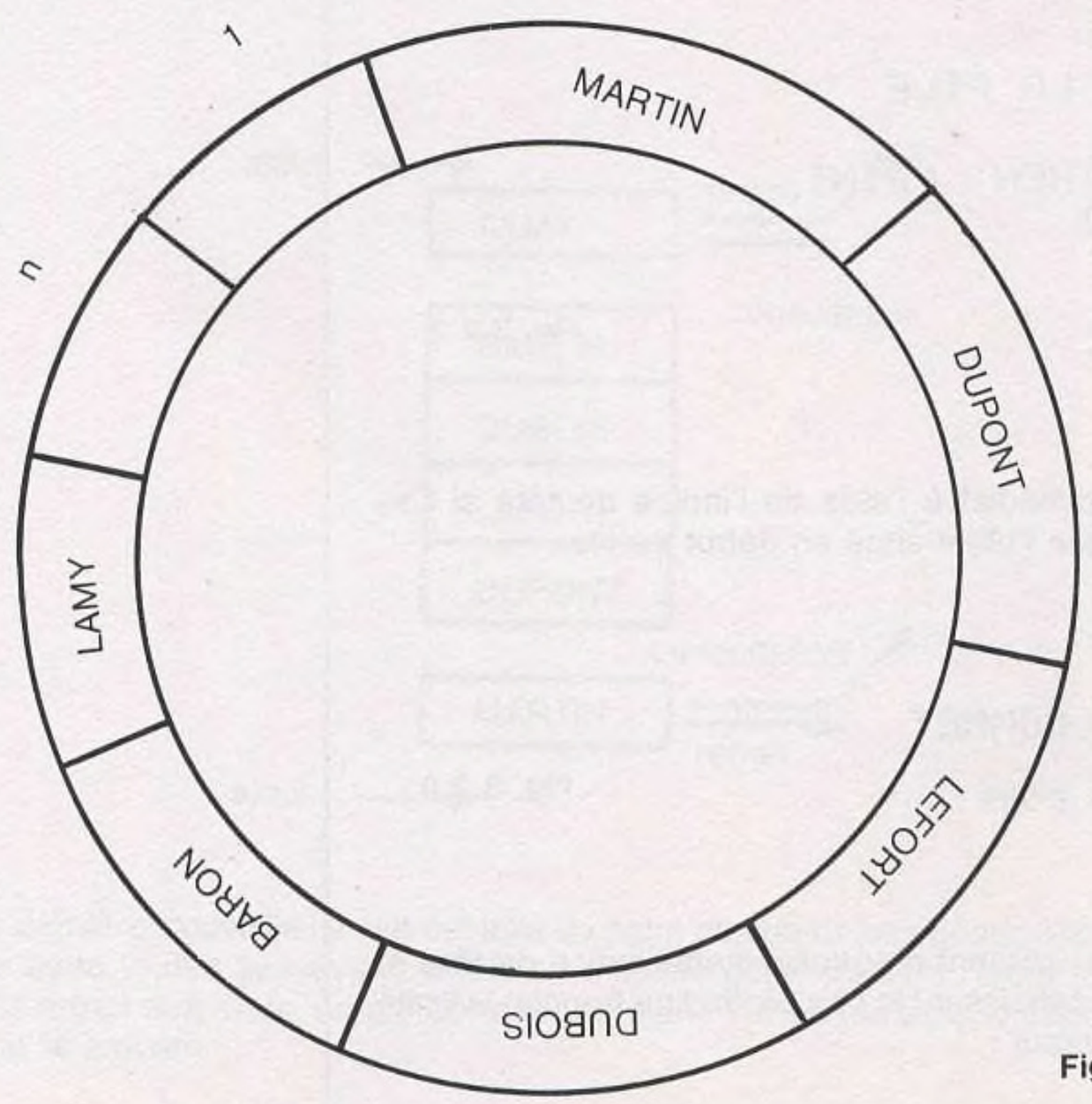


Fig. 3.1.3.2

Il y a dans ce système un danger : si la file est vide, la tête et la queue auront la même valeur, mais si la file est pleine, la tête et la queue auront aussi mêmes valeurs, on autorisera donc une capacité maximale de N – 1 éléments si la file contient N places disponibles. Désormais, le test de remplissage sera vrai lorsque Abs (tête - queue) > 0.

3.2. Implantation de la File en BASIC

Les procédures sont ici légèrement plus longues puisqu'il y a un indice de plus à gérer, toutefois, la complexité des opérations reste insignifiante. La création d'une file s'exprime donc :

```
100   REM    INITIALISATION DE FILE

110 N = 100: REM    DIMENSION DE L
      A FILE
120   DIM FILE$(N)
130 TETE = 0:QUEUE = 1
140   FOR J = 1 TO N
150 FILE$(J) = "VIDE"
160   NEXT J
```

Fig. 3.2.1

La consultation de l'état de la file reste de nature analogue :

```
200   REM    TEST D'ETAT DE LA FILE

210   IF TETE = QUEUE - 1 THEN  PRINT
      "FILE VIDE": GOTO 230
220   PRINT "FILE REMPLIE"
230   REM  FIN TEST
```

Fig. 3.2.2

La visualisation du dernier élément est immédiat à l'aide de l'indice de tête si l'on respecte le fait de ne pouvoir consulter que l'objet situé en début de file :

```
300   REM  CONSULTATION DU SOMMET
310   PRINT FILE$(TETE)
```

Fig. 3.2.3

Les procédures de «défilage» et «enfilage» utilisent respectivement l'indice de tête et celui de queue, avec un test incorporé reconnaissant le cas où l'indice franchit la limite d'espace mémoire pour être replacé au début :

```
400   REM    ENFILAGE
410   IF QUEUE = TETE THEN  PRINT
      "FILE SATUREE": GOTO 450
420   INPUT "NOUVEL ELEMENT ";FILE
      $(QUEUE)
430   IF QUEUE = N THEN QUEUE = 0
440 QUEUE = QUEUE + 1
```

Fig. 3.2.4

```
500   REM    DEFILAGE
510 FILE$(TETE + 1) = "VIDE"
520   IF TETE = N - 1 THEN TETE =
       - 1
```

Fig. 3.2.5

3.3. Les files à double accès

Nous sommes là en présence d'une généralisation astucieuse des deux structures que nous venons de présenter. Sur une pile, on ajoute et on enlève les éléments à la même extrémité, au contraire, sur une file, on ajoute d'un bout et on élimine de l'autre. Vous avez deviné : en combinant, on pourra désormais ajouter et enlever des deux côtés. Pour la création, cela ne change rien, mais pour toutes les autres procédures, il conviendra de préciser à quel bout on souhaite travailler : ceci déterminé, le protocole demeure inchangé :

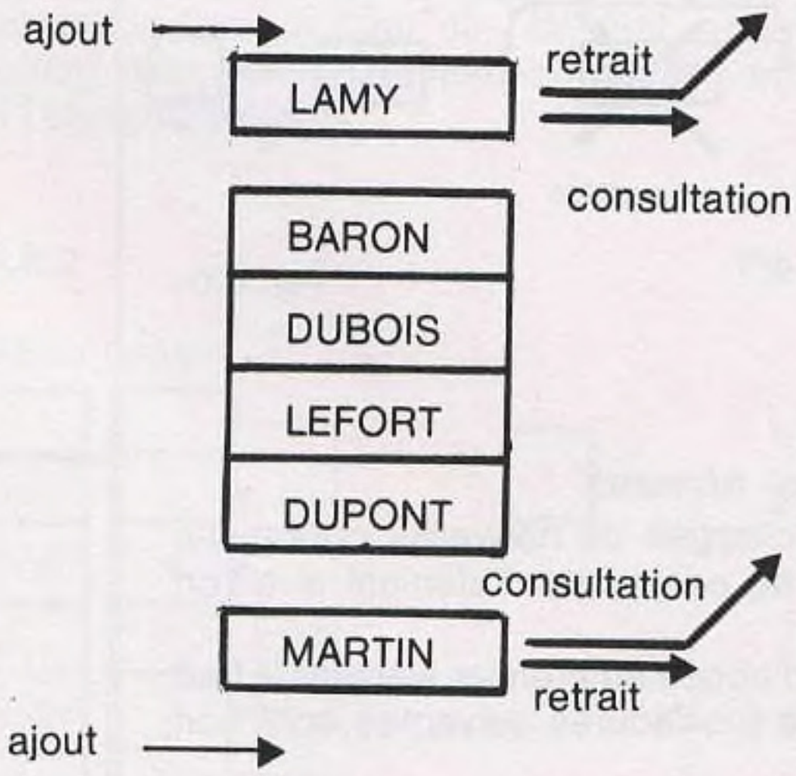
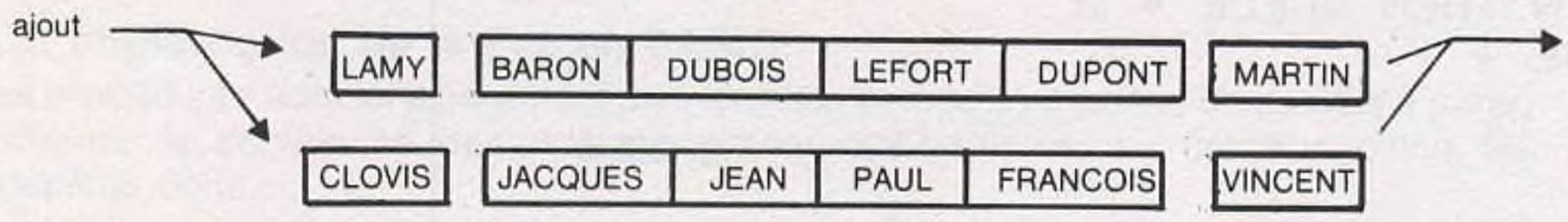


Fig. 3.3.1

Cette dernière trouvaille, si elle est jolie du point de vue de ses possibilités, n'est guère usitée avec toutes ses éventualités, on emploie généralement l'ajout à un seul bout pour le retrait aux deux, ce qui correspond à une file où certains individus sont servis dès qu'ils arrivent.

3.4. Structures parallèles en files

La solution choisie est rigoureusement la même que pour la pile, à savoir des tableaux évoluant avec un indice identique.



4. LES LISTES LINEAIRES

L'adjonction de l'adjectif «linéaire» au terme de liste est expliqué par la différenciation que nous voulons faire des listes «simples» que vous verrez le mois prochain. La liste linéaire est une émanation des structures précédentes. Par rapport aux piles et files, elle possède la faculté de pouvoir accéder directement à chacun de ses éléments sans nécessiter le retrait d'autres objets de l'ensemble. Dans ce même ordre d'idées, il est également possible d'insérer n'importe quel élément à n'importe quel endroit dans la liste linéaire. Vous devinez l'intérêt que peut avoir une telle structure à l'application de l'agenda car nous pouvons ainsi, par programme, choisir d'intercaler une nouvelle personne à son emplacement alphabétique.

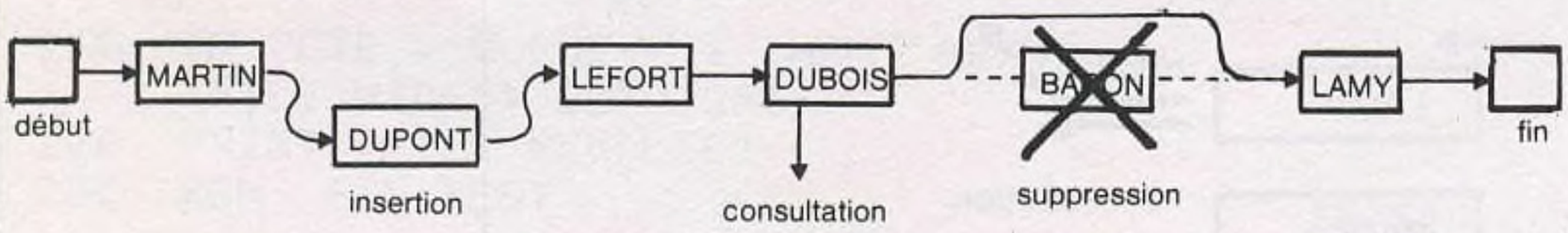


Fig. 4.0

4.1. Elaboration de la liste linéaire à tous les niveaux

4.1.1. La spécification fonctionnelle se trouve surchargée de nouvelles opérations nécessaires à l'exploration de la liste linéaire afin de positionner l'élément que l'on souhaite manipuler.

Outre les fonctions de création, de test sur l'état et d'accès au premier élément (il faut bien savoir d'où l'on part) toujours nécessaires, les procédures suivantes sont soit considérablement modifiées, soit nouvelles :

- en accès, une opération doit être capable d'obtenir l'élément suivant dans la liste linéaire telle qu'on a souhaité l'ordonner ;
- en modification, l'insertion avant l'élément que l'on positionne doit être possible même en fin de liste linéaire ;
- en modification toujours, la destruction d'un quelconque élément doit être possible sans bouleverser irrémédiablement la liste linéaire.

4.1.2. La description logique revient au dépouillement que l'on constatait à la découverte de la pile : une liste linéaire ne comporte qu'un début constitué du premier élément et d'une suite comprenant le reste des objets.

4.1.3. La représentation physique est, par contre, nettement plus élaborée car deux politiques bien distinctes peuvent être mises en œuvre et toutes deux présentent des désagréments qui sont loin d'être négligeables.

La première, de loin la plus simple à comprendre, consiste, à chaque nouvelle insertion, à déplacer tous les éléments situés en aval de l'emplacement choisi d'autant de rangs que nécessaires. De même, à la suppression, il est obligatoire de remonter tous les éléments situés en dessous du «mort» de la hauteur libérée. Il n'est pas besoin de vous développer la lourdeur de cette méthode du point de vue du temps de calcul.

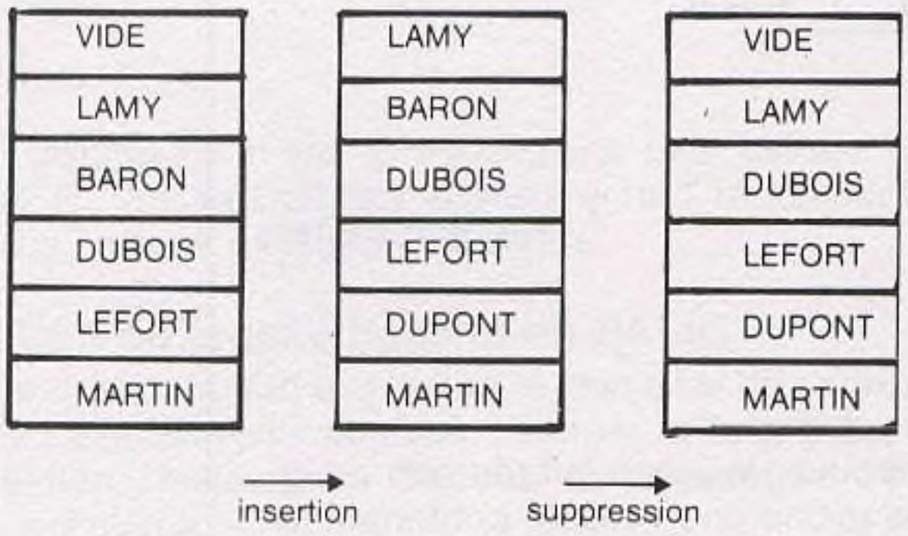


Fig. 4.1.3.1

La seconde a plus les faveurs des programmeurs mais nécessite pour une liste linéaire dénuée de tout artifice le double en taille mémoire puisque l'on requiert l'usage d'un tableau de taille analogue à celle de la liste linéaire. L'idée même ne vous est pas inconnue, car c'est la méthode employée pour le rangement en machine des lignes d'instructions de BASIC.
Nous avons d'une part le tableau où se trouvent rangés les éléments, et juste à côté, un autre tableau où est indiqué le rang de l'élément suivant dans la liste linéaire. Comme cela, l'insertion d'un nouvel élément consiste à changer, dans le tableau associé, le rang de l'élément suivant.

Fig. 4.1.3.2

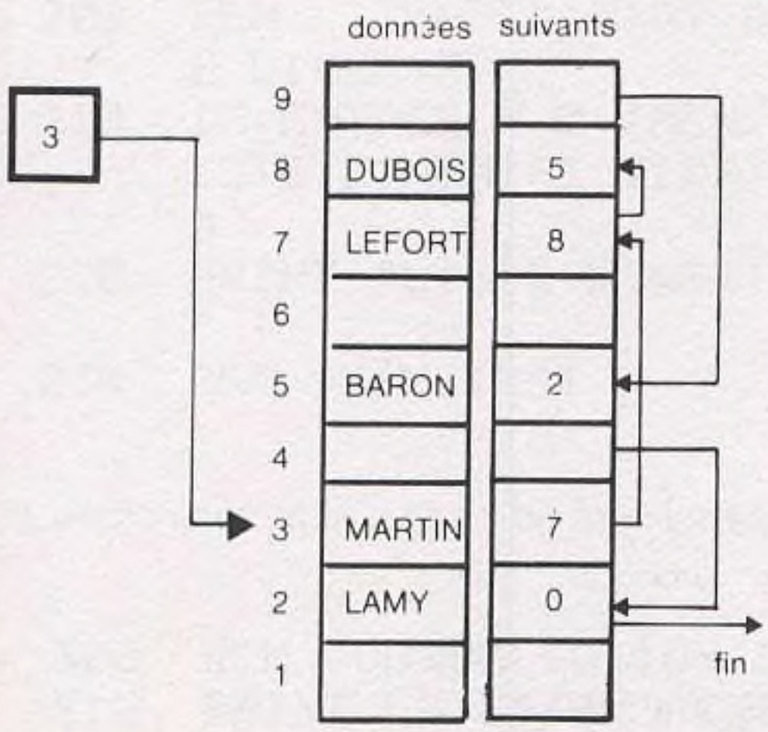
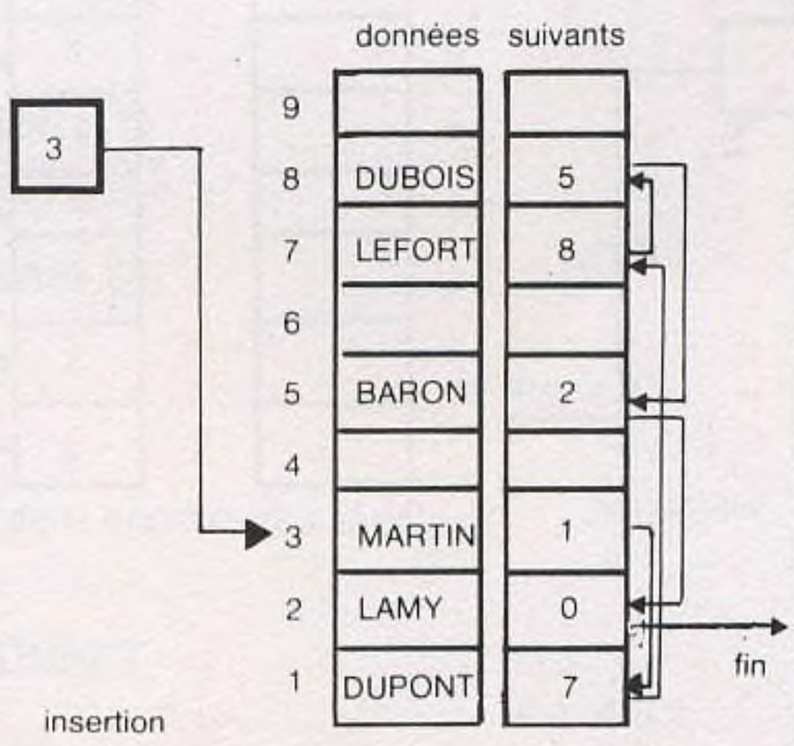


Fig. 4.1.3,2 bis



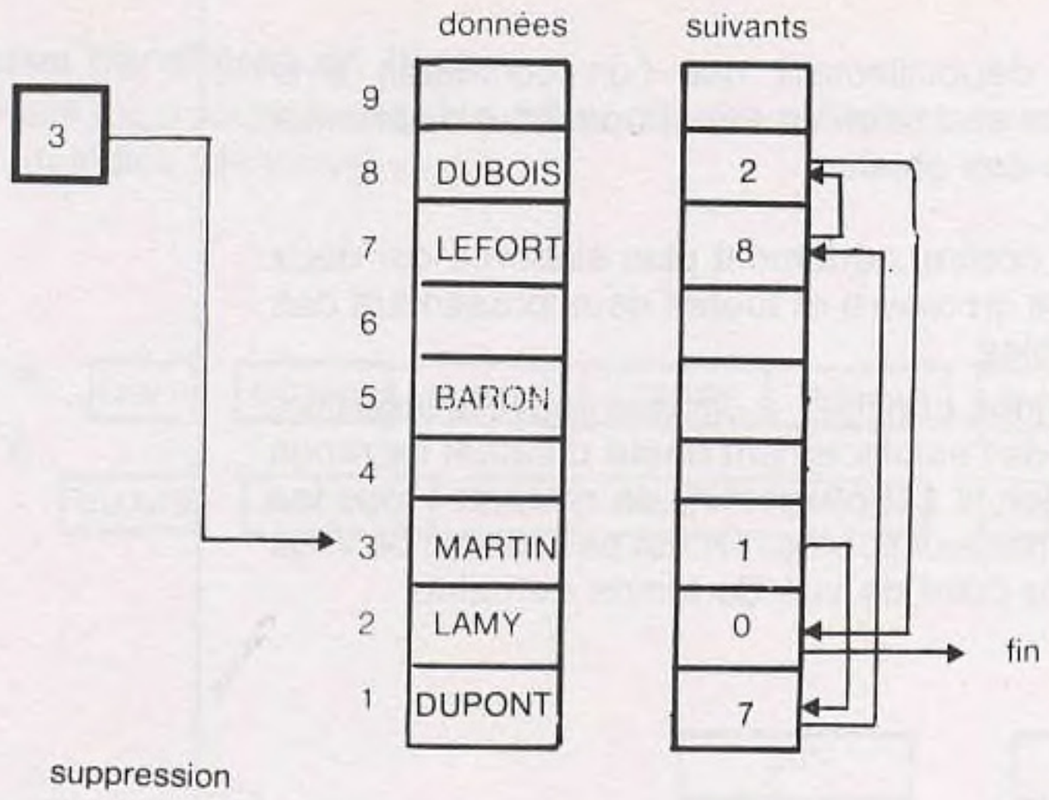


Fig. 4.1.3.2 ter

Mais lorsqu'on détruit un élément, il faut pouvoir retrouver la place libérée afin de la réutiliser ultérieurement. Pour cela, tout comme nous avons une variable contenant le rang de la première donnée de la liste linéaire, nous nous devons de sauvegarder quelque part le rang de la première place libre. A la création de la liste linéaire, on initialise l'ensemble du tableau des rangs en donnant l'ordre du suivant puis en cours de programme, à l'occasion de chaque libération, on remet au rang libéré l'ancien premier rang libre ; nous avons de cette façon une pile des emplacements libres.

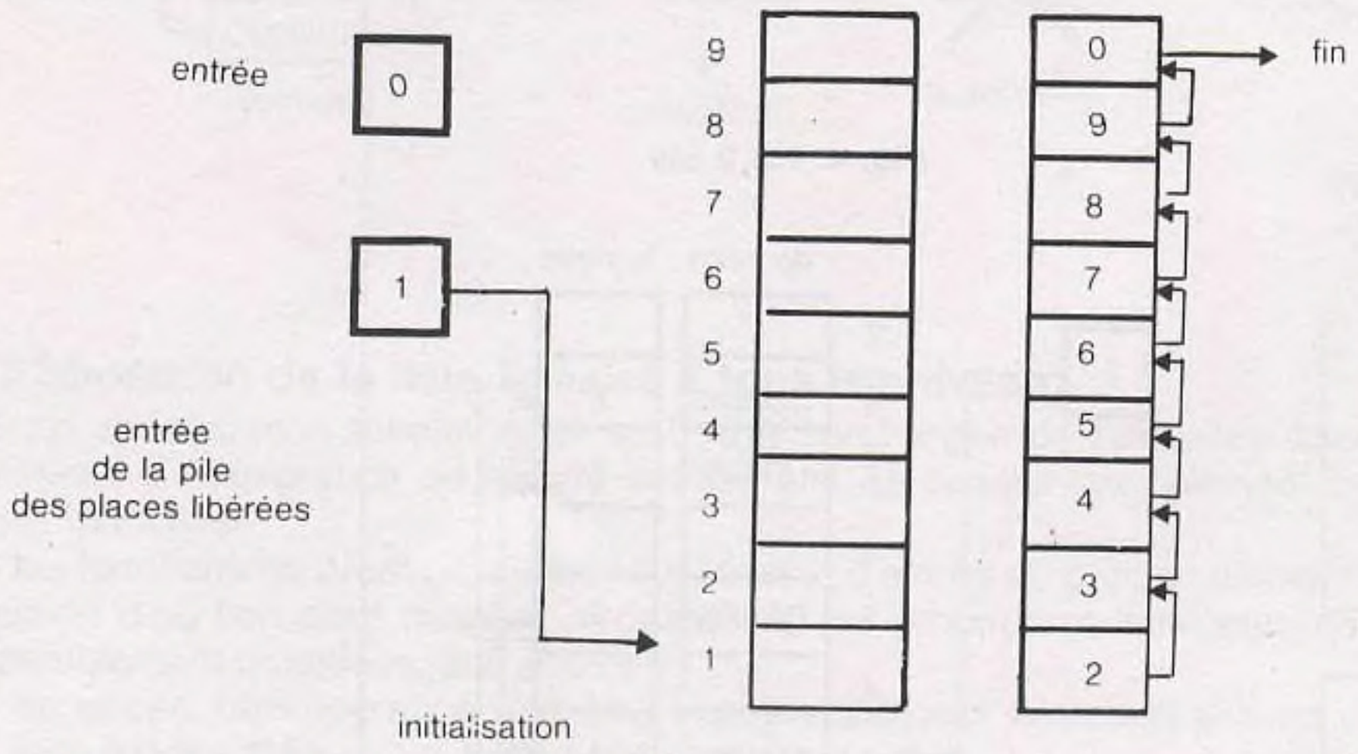


Fig. 4.1.3.4

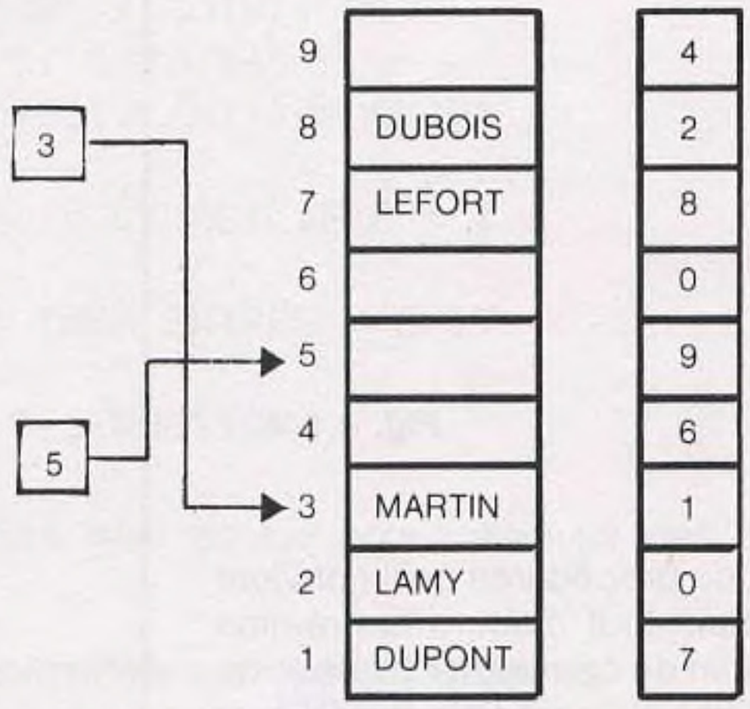


Fig. 4.1.3.5

Cette dernière procédure n'est généralement pas utilisée dans la gestion des programmes BASIC, les ordinateurs préférant tout recompacter à votre demande comme nous vous l'avions expliqué à l'époque.

4.2. Implantation de la liste linéaire en BASIC

Vous vous doutez que nous avons quitté le domaine de mini-procédures réduites à une ligne hormis celle de vérification de l'état de la liste linéaire égale à elle-même. Mais dès la création, l'initialisation de tous les paramètres d'exploitation complique sensiblement le problème :

```
100  REM    INITIALISATION DE LIS
     TE LINEAIRE
110 N = 100: REM    DIMENSION DE
     LA LISTE LINEAIRE
120  DIM LSTE$(N),SUIV%(N)
130 ENTREE = 0:LIBRE = 1
140  FOR J = 1 TO N
150 LSTE$(J) = "VIDE"
160 SUIV%(J) = J + 1
170  NEXT J
180 SUIV%(N) = 0
```

Fig. 4.2.1

```
200  REM   TEST D'ETAT DE LA LIST
     E LINEAIRE
210  IF ENTREE = 0 THEN  PRINT "L
     ISTE LINEAIRE VIDE": GOTO 23
     0
220  PRINT "LISTE LINEAIRE EMPLEE
     "
230  REM  FIN TEST
```

Fig. 4.2.2

L'exploration et la lecture du premier élément sont encore très simples.

```
300  REM  CONSULTATION DU SOMMET
310  PRINT LSTE$(ENTREE)
```

Fig. 4.2.3

```
400  REM    EXPLORATION DE LA LIS
     TE LINEAIRE
410  INPUT "RANG SOUHAITE";RG
420 NO = 1
430 DRL = ENTREE
440  IF NO = RG THEN 470
450 DRL = SUIV%(DRL)
460 NO = NO + 1
470  PRINT LSTE$(DRL)
```

Fig. 4.2.4

Mais avec l'insertion et la destruction nous rencontrons des procédures qu'il convient d'observer deux fois pour maîtriser le mécanisme. Etudions tout d'abord la création d'un nouvel élément dans la liste linéaire : nous avons besoin de connaître l'adresse de l'entrée (ENTREE) dans la liste linéaire ; de celle du premier élément libre (LIBRE), et l'ordre des éléments précédent (NXT) et suivant (DRL) ; les éléments étant rangés dans LST$ et l'adresse de leur successeur dans SUIV$: ces caractéristiques étant communes aux deux sous-programmes : on commence par déterminer l'emplacement où on compte pratiquer les opérations désirées puis on rectifie les liens existant entre les éléments de la liste linéaire.

```
500  REM     CREATION D'UN ELEMEN
     T DE LA LISTE LINEAIRE
510  INPUT "RANG SOUHAITE":RG
520 NO = 1
530 DRL = ENTREE:NXT = DRL
540  IF NO = RG THEN 570
545  IF SUIV%(DRL) = 0 THEN  PRINT
     " LISTE TERMINEE ": GOTO 570

550 NXT = DRL:DRL = SUIV%(DRL)
560 NO = NO + 1: GOTO 540
570  IF LIBRE = 0 THEN  PRINT "LI
     STE PLEINE": GOTO 590
572 TEMP = LIBRE
575 SUIV%(NXT) = TEMP: INPUT LSTE
     $(TEMP)
580 LIBRE = SUIV%(TEMP):SUIV%(TEM
     P) = DRL
585  IF RG = 1 THEN ENTREE = TEMP

590  REM  FIN DE L' INSERTION
```

Fig. 4.2.5

```
600  REM  DESTRUCTION D' UN ELEME
     NT DE LA LISTE LINEAIRE
610  INPUT "RANG SOUHAITE":RG
620 NO = 1
630 DRL = ENTREE:NXT = DRL
640  IF NO = RG THEN 670
645  IF SUIV%(DRL) = 0 THEN  PRINT
     " LISTE TERMINEE ": GOTO 690
```

```
650 NXT = DRL:DRL = SUIV%(DRL)
660 NO = NO + 1: GOTO 640
670 TEMP = DRL:SUIV%(NXT) = SUIV%
     (DRL)
680 SUIV%(TEMP) = LIBRE:LIBRE = T
     EMP
685  IF RG = 1 THEN ENTREE = SUIV
     %(NXT)
690  REM  FIN DESTRUCTION
```

Fig. 4.2.6

Avec cela, vous êtes déjà équipé pour manipuler des listes à l'intérieur de vos programmes.

4.3. Structures parallèles en listes linéaires

Ici, la généralisation à plusieurs tableaux n'est pas plus difficile que pour les structures précédentes ; le bon sens vous dira que l'on n'a besoin que d'un seul tableau adjoint pour obtenir les éléments suivants :

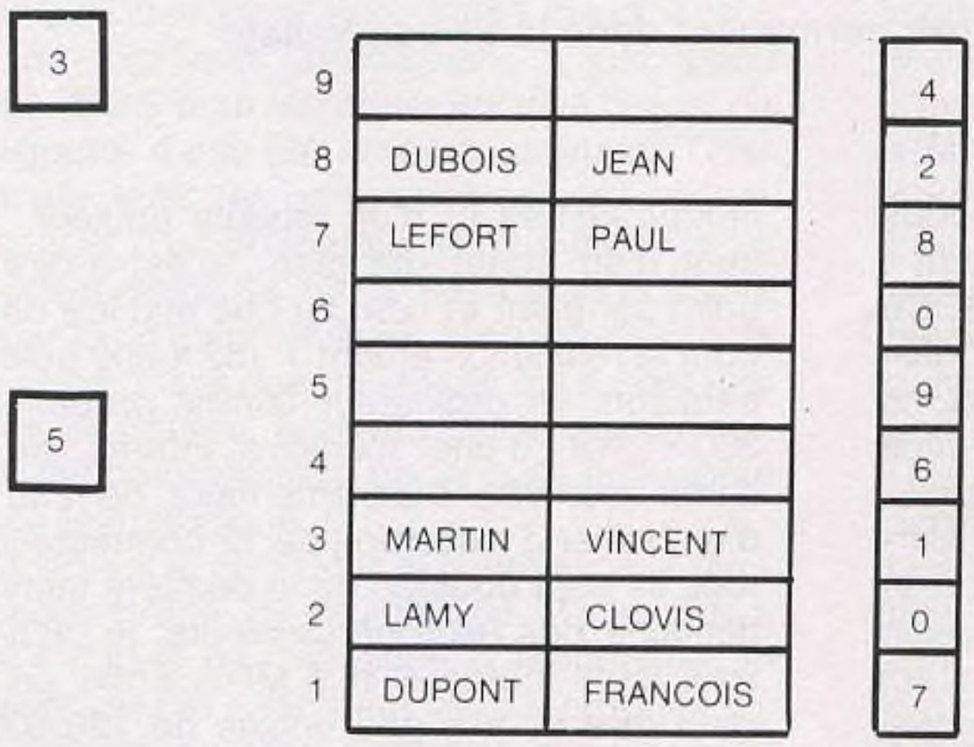


Fig. 4.3.1

Remarque : Nous avons toujours parlé de listes linéaires, faites de même car dans un mois, l'arrivée des vraies listes devra être entrée dans les mœurs.

5. CONSEILS DES ENTRAINEURS A LA MI-TEMPS

Nous avons vu ce mois-ci la première partie de notre cours sur les structures de données. Nous en avons profité pour introduire de façon pernicieuse des notions de niveaux de spécification. Nous espérons que vous avez saisi avec nous l'utilité de ce travail de défrichage avant programmation et qu'il vous a permis de comprendre plus rapidement les procédures associées à chaque structure ; ce fut pour nous particulièrement notable avec les listes linéaires où il convient de bien définir comment cela va se passer avant toute chose car, après coup, on perd des choses, on en oublie, il arrive même qu'on crée ! Alors, assimilez bien tout cela et attendez le mois prochain, nous verrons le reste.

6. EXERCICE D'APPLICATION

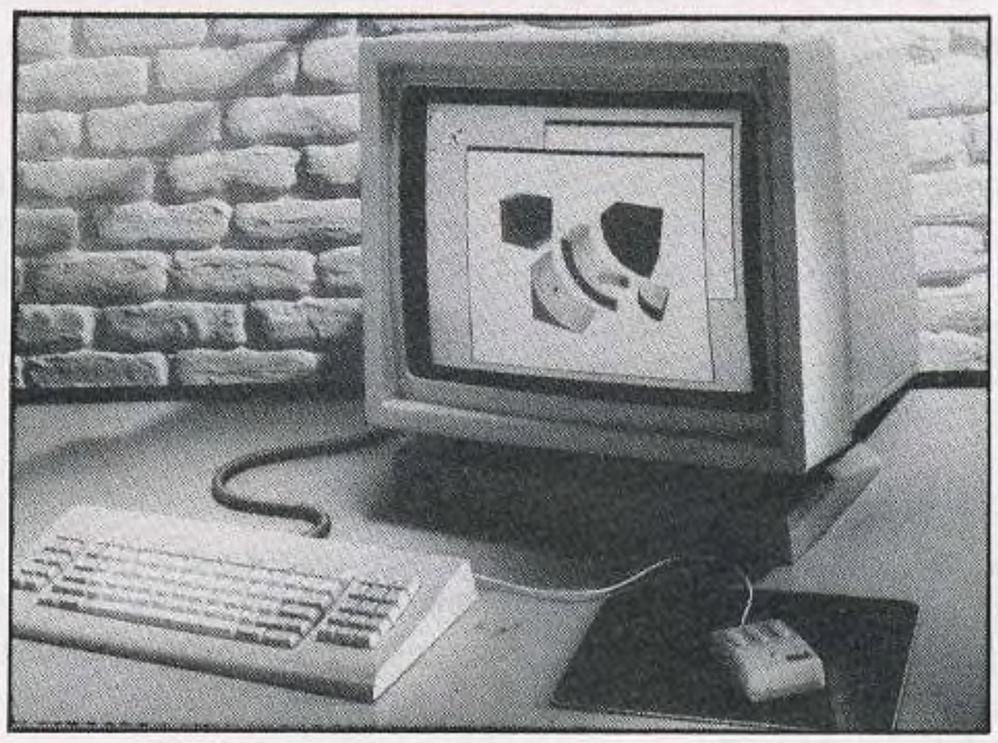
Il vient de lui-même, vous devez être en mesure de simuler un agenda manipulant une pile ; une file et une liste linéaire. Une fois ce problème résolu, c'est alors que vous reprendrez vos Benchmarks afin de mesurer les vertus de chacune des structures que nous venons de vous exposer. Ce mois-ci, le sujet est succinct mais le travail n'est pas léger.

C'EST ARRIVE DEMAIN

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

Peut-être vous souvenez-vous de ce que je vous avais dit au sujet de la publication de livres touchant à l'informatique, les ordinateurs ou les programmes. Ce mois-ci, notre confrère *Popular Computer* s'intéresse à ce problème et à l'évolution des maisons d'éditions spécialisées. Ces sociétés semblent avoir trouvé une parade relative à la désaffection croissante du public pour leurs produits. Il s'agit tout simplement de s'associer avec un fabricant ou un créateur de logiciels à succès, pour assurer la diffusion des livres techniques concernant le produit. Ceci assure une certaine exclusivité et leur permet de prévoir l'évolution du marché, grâce aux données du fabricant. Ainsi, Apple travaille avec Addison-Wesley et Hayden, ATT voit ses titres diffusés par Prentice-Hall, et Lotus (1-2-3 et Symphony) par Addison-Wesley. Il sera intéressant d'étudier l'évolution du contenu des ouvrages, pour savoir s'il s'agit ou non d'une allégeance tournant à la publicité ou si ces éditeurs conserveront une indépendance qui reste indispensable pour l'information du client. En effet, de ce côté de l'Atlantique, les livres mais surtout les magazines, qui sont édités eux aussi par les maisons précitées, sont la principale source d'informations du consommateur.

J'ai été convié, cette semaine, à la présentation mondiale par Sun-Microsystems de sa nouvelle gamme de stations de travail (many thanks to Mary for the invitation. Fin du message personnel). La présentation était vraiment impressionnante, à tel point que, bien qu'on ne puisse qualifier de système personnel un ensemble valant dans les 15 millions de centimes, pour la version de base, j'ai décidé de vous en parler. La gamme Sun-3 est constituée de deux modèles monochromes et d'un modèle couleur. Tous disposent d'un écran «bit map», c'est-à-dire adressable point par point et non par une matrice de caractères, dont la résolution atteint 1 152 × 900 points (par comparaison, un ordinateur familial propose en général 300 × 200), d'une mémoire interne de 2 Moctets, d'une mémoire supplémentaire de 256 Moctets et d'un boîtier d'extension de 12 connecteurs. Mais surtout, ils sont équipés de la dernière merveille dans le domaine des microprocesseurs, le 68020 de Motorola. Ici, il tourne à 16,5 MHz. Enfin, un tel système peut être acquis pour moins de 120 000 francs, en version de base, ce prix pouvant (ou devant) naturellement être augmenté de façon importante avec les diverses possibilités offertes. Si je vous parle de tou-



Le Sun 3/160 C.

tes ces caractéristiques, c'est parce que ce type de système n'est pas plus éloigné de votre ordinateur que ce dernier ne l'est des ordinateurs d'il y a dix ans. Alors un peu de patience, et vous pourrez sans doute disposer de l'équivalent dans peu de temps.

Un phénomène intéressant se développe depuis quelques mois sur le plan éducatif. Un système, appelé l'Université Electronique, vous permet de suivre des enseignements en restant chez vous, dans des domaines aussi différents que l'informatique, les arts et les affaires (ici, on les retrouve toujours en cherchant un peu). L'important est que les inscriptions se font sur plus de 1 700 universités, pour un niveau pouvant aller jusqu'à l'équivalent de la maîtrise en France. Les plus grandes universités sont intégrées dans ce système comme le MIT, Berkeley, Carnegie-Mellon, Cornell, UCLA ou Stanford. Ceci permet à des mères de famille, à des salariés ou à des infirmes d'étudier dans de bonnes conditions à l'obtention de diplômes prestigieux. L'inscription coûte au plus 300 $ pour les cours les plus chers, ce qui est une goutte d'eau par rapport au prix de l'inscription normale dans ces universités et il est nécessaire de disposer d'un ordinateur et d'un modem. La connexion valant moins de 100 $, il est alors possible de s'entretenir avec un professeur, de subir des tests de contrôle et de poser des questions qui auront une réponse sous 24 heures, envoyée par le professeur. Tout ceci se fait naturellement par le biais du modem et d'une ligne téléphonique.

Apple a décidément le chic pour déclencher des tempêtes. Le responsable du programme Macintosh a annoncé qu'il n'était pas question d'abandonner le développement de cet ordinateur et des produits s'y rapportant, alors qu'en effet personne n'envisageait cette éventualité. Cela a jeté un froid dans la communauté informatique locale. (Imaginez que quelqu'un vous dise que, finalement, il n'y a pas de loup en liberté dans Paris, sans que rien ne puisse vous l'avoir fait penser auparavant. Il est probable que vous commenciez à vous demander s'il n'y aurait pas quelques loups dans Paris). Tout ceci est d'autant plus intéressant qu'Apple est coutumier du fait, c'est-à-dire faire ce genre de déclaration avant l'abandon d'un produit, ainsi que cela s'est passé pour le Lisa. Du coup, deux développeurs ont annoncé l'arrêt de tout travail pour le Macinotsh. Il reste une autre hypothèse, qui est qu'Apple en soit réduit à ce type de publicité pour faire parler de la marque, ce qui n'est pas trop rassurant.

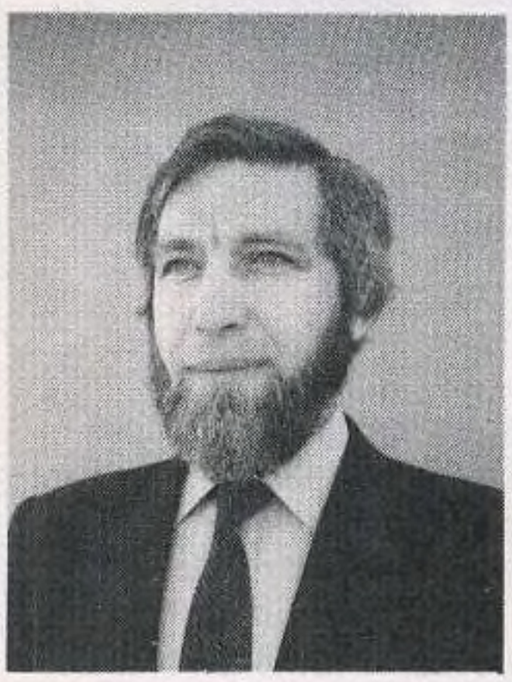
Un événement important cette semaine est l'annonce par ATT de la sortie très prochaine d'un compatible IBM PC/AT, qui est le gros modèle de la série des micro-ordinateurs PC de chez IBM. Cela semble être le signe de la volonté d'ATT de vouloir définitivement prendre la place de second sur le marché des micros,

Un exemple de texte sur micro.

place qui appartient toujours à Apple. Jusqu'à présent, ATT se contentait d'une place somme toute modeste, compte tenu de la dimension de la société. Mais les fabricants de compatibles commencent à s'inquiéter de l'intérêt subit d'ATT pour ce marché. En effet, IBM s'est jusqu'alors contentée d'imposer sa norme, sans trop chercher à pousser à la vente forcée, ce qui a laissé le champ libre aux fabricants de compatibles, développant des produits souvent bien moins chers. Mais la venue d'ATT va créer une concurrence de fait avec IBM, ce qui entraînera peut-être une baisse des prix de cette dernière société.

Une enquête récente montre que les ordinateurs personnels voient leur usage évoluer nettement ces dernier temps. En effet, jusqu'à maintenant, les programmes les plus utilisés étaient les tableurs, comme Visicalc, Multiplan, Supercalc, et ceux des logiciels intégrés comme 1-2-3, Excel ou Appleworks. Mais il semble que les possibilités des nouvelles imprimantes, lasers ou non, incitent les utilisateurs à faire de leur petit système favori une machine à traitement de texte, afin de se passer d'une secrétaire et de pouvoir ordonner ainsi leurs idées en conservant une trace sur papier, mais aussi pour des rapports et des présentations demandant des performances graphiques que les gros systèmes ne proposent pas en général, pas plus que les machines à traitement de textes. On assiste à une explosion du nombre de programmes permettant ces facilités sur micro-ordinateurs. Il en va de même pour les imprimantes, dont les capacités graphiques figurent en tête des publicités. Certaines proposent même des aides à la photocomposition plus ou moins élaborées, pour des mises en page de qualité professionnelle. Quelqu'un a dit, voilà une dizaine d'années que nous étions à l'époque de l'image, et cela se vérifie jusqu'au plus profond des bureaux. L'ordinateur, qui est déjà en train de causer la mort d'un certain nombre de professions, comme dessinateur industriel, semble s'attaquer maintenant aux emplois de bureau.

Microprocesseurs un cours essentiellement pratique !



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.

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.

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.

En vente chez votre libraire et aux Editions Fréquences

Bon de commande à adresser aux EDITIONS FREQUENCES 1, bd Ney 75018 PARIS

Je désire recevoir le(s) ouvrages(s) suivant(s) :

☐ INITIATION A L'ELECTRONIQUE DIGITALE au prix de **105 F** (95 F + 10 F de port).

☐ INITIATION AUX MICROPROCESSEURS au prix de **105 F** (95 F + 10 F de port).

Ci-joint mon règlement par : ☐ CCP ☐ Chèque bancaire ☐ Mandat

Nom ..... Prénom .....

Adresse .....

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

COURS DE GENIE LOGICIEL

De la théorie à la pratique

Charles-Henry Delaleu

LA STRUCTURATION

Devant l'ampleur que l'informatique a prise de nos jours rien ne peut plus être laissé au hasard. La réalisation d'un programme au moment de son écriture est devenue une des plus importantes étapes d'un produit logiciel.
Il est aujourd'hui impossible d'écrire un programme sans un minimum de structuration de sa présentation et de son architecture.
Un programme, outre ses qualités internes, doit pouvoir évoluer dans le temps. De ce fait, il est indispensable de l'écrire avec un minimum de règles. Ces règles seront appliquées quelle que soit l'importance du programme à réaliser.
Hormis une clarté bien plus grande à la lecture du listing, un programme bien structuré sera toujours beaucoup plus fiable. Un nombre impressionnant d'erreurs sera évité. Toutes modifications ou aménagement ultérieurs seront facilités et ne poseront aucun problème majeur.
Enfin, il sera tout à l'honneur d'un programmeur de réaliser un beau travail, preuve de ses compétences et de l'amour du travail bien fait.
Hormis les raisons techniques qui poussent les analystes-programmeurs à structurer leurs applications, il convient de noter que la réalisation d'une ligne de programme coûte de plus en plus cher. Dans les grandes applications, la ligne de programme est facturée aux alentours de 500 F. Chaque programme comptant un nombre non négligeable de lignes, on comprendra aisément que la meilleure solution consiste à réécrire un nombre minimal de lignes pour chaque nouvelle application. Les outils de structuration permettent un net progrès dans ce sens.

STRUCTURATION DES OBJETS

La structuration des objets a pour but d'organiser avec un minimum de soin les objets manipulés et utilisés par les algorithmes. Il convient d'organiser l'espace mémoire dans lequel ils seront stockés, ainsi que leur accès. Ceci peut être réalisé de différentes manières. Nous abordons ici les principes de base. Des précisions seront fournies dans le chapitre consacré aux fichiers. Toutefois, il est nécessaire dès à présent de commencer à se familiariser avec le jargon couramment utilisé par les spécialistes.
L'accès à un objet peut être effectué de plusieurs manières :

ACCES LOGIQUE A UN OBJET DANS UNE COLLECTION :

Un accès logique peut être :
- soit par valeur
- soit relatif, récurrent ou récursif.

Nota : récursif : cherche dans plusieurs sens
récurrent : cherche le suivant.

L'organisation d'un fichier peut être :
Soit inexistante : c'est le cas d'un tas, l'accès ne peut donc se réaliser que par valeur.
Soit graphe : les relations entre articles sont marquées par des pointeurs ou par des clefs. Ici les objets sont présentés sous forme de chaînes.

Une base de données peut être organisée :
Soit en réseau : les relations sont marquées par des pointeurs. L'accès y est relatif. Ici les objets sont présentés de manière hiérarchique.
Soit en relationnel : les relations sont marquées par clefs, l'accès se fait par valeur.

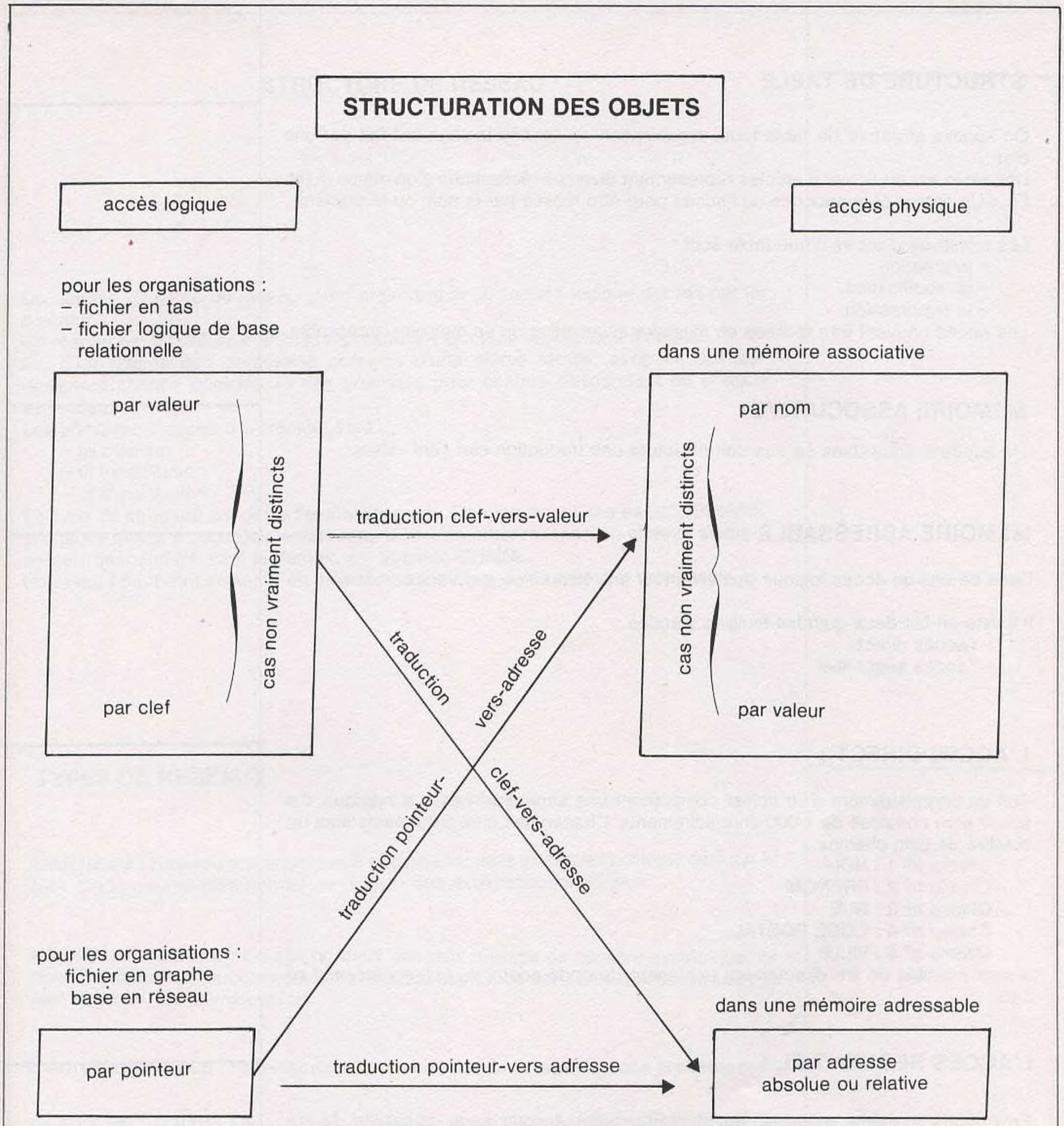
ACCES PHYSIQUE

L'accès physique peut se réaliser de plusieurs manières :
- soit par nom
- soit par valeur
- soit par adresse absolue
- soit par adresse relative.

L'accès physique est possible grâce à la gestion des cases mémoires de l'ordinateur.

Adresse absolue :
Dans ce cas, l'accès à l'objet est très facile. En effet, ce dernier a été placé dans une case mémoire préalablement désignée et nommée.
Il suffit donc d'appeler cette case mémoire pour obtenir l'objet que l'on cherche.

Adresse relative :
Ici, il existe une petite nuance qui est la suivante : l'objet a été placé dans une case mémoire définie par la machine et non par l'utilisateur. Dans ce cas, il va falloir faire appel à une case mémoire qui contient l'adresse où a été réellement stocké l'objet.



Représentation des accès logiques et des accès physiques.

STRUCTURE DE TABLE

On appelle structure de table toute organisation où l'accès logique est fait par une clef.
Une table est un fichier d'articles représentant diverses réalisations d'un même objet.
Ex. : Un fichier de personnes où l'accès peut être réalisé par le nom ou le prénom.

Les primitives d'accès d'une table sont :
- la création
- la modification
- la suppression.

Les accès peuvent être réalisés en mémoire associative ou en mémoire adressable.

MEMOIRE ASSOCIATIVE :

Un accès logique dans ce cas doit effectuer une traduction clef vers valeur.

MEMOIRE ADRESSABLE :

Dans ce cas un accès logique doit effectuer une traduction clef vers adresse.

Il existe en fait deux grandes familles d'accès :
- l'accès direct
- l'accès séquentiel.

L'ACCES DIRECT :

Soit un enregistrement d'un fichier comprenant une série d'adresses d'individus. Ce fichier sera composé de 1 000 enregistrements. Chacun des enregistrements aura un nombre de cinq champs.
Champ n° 1 : NOM
Champ n° 2 : PRENOM
Champ n° 3 : RUE
Champ n° 4 : CODE POSTAL
Champ n° 5 : VILLE
Il sera possible de lire directement le numéro de code postal de l'enregistrement n° 543.

L'ACCES SEQUENTIEL :

En prenant le même exemple que précédemment, il nous sera impossible de lire directement le numéro de code postal de l'enregistrement n° 543. Il sera nécessaire, après s'être placé à l'enregistrement 543, de lire les cinq champs. L'information désirée étant le code postal, nous lirons d'abord le nom, le prénom, la rue, puis enfin le code postal. Il y a chaînage d'objets les uns à la suite des autres.

Nota : Le premier est beaucoup plus rapide, le second permet de limiter l'espace mémoire. (Nous reviendrons sur ces notions).

STRUCTURE DE RESEAU

On appelle structure de réseau toute organisation où l'accès logique est fait par un pointeur.
Un réseau est un fichier d'articles représentant diverses réalisations d'objets.
Ex. : Le réseau des personnes connues d'une même famille, avec un pointeur désignant chaque conjoint, et des pointeurs pour chaque descendant ou chaque ascendant directs.
Les primitives d'accès d'un réseau sont :

- la création
- la modification
- la suppression

Ce type de structure est de forme arborescente. Une arborescence est dite BINAIRE si chaque article a deux descendants au plus. Une arborescence où chaque article a un seul descendant, sauf le dernier, est appelée CHAÎNE.
Un réseau peut être implanté en mémoire associative ou en mémoire adressable.

TYPES DE RESEAUX

L'ANNEAU : l'anneau est analogue à une chaîne, mais le dernier pointeur désigne la tête. Cette organisation permet de réaliser des explorations cycliques.

LA CHAÎNE DOUBLE : il s'agit de deux anneaux réalisés de manière symétrique, ils permettent ainsi un déroulement dans les deux sens. Ces deux anneaux comprennent, bien entendu, les mêmes objets.

L'ANNEAU DOUBLE : l'anneau double combine les deux organisations précédentes.

LA FILE SEQUENTIELLE : c'est une chaîne où les ajouts ne sont effectués qu'à la fin de la chaîne. Il est impossible de réaliser des mises à jour dans une file séquentielle.

LA FILE D'ATTENTE : c'est une chaîne où les ajouts se font comme dans la file séquentielle, mais qui autorise des suppressions à la tête de la file.

LA PILE : c'est une chaîne où les ajouts et les suppressions ne se font que par la tête de la file.

STRUCTURATION DES ACTIONS

Une fois que sera déterminée la structure choisie pour les objets, il sera nécessaire de se pencher sur les actions.
La presque totalité des langages autorisent une bonne structuration des actions. Dans le cas où certains termes n'existeraient pas, il sera nécessaire, tout en augmentant le nombre de lignes du programme, de respecter une bonne structuration des actions. Ceci est primordial afin d'obtenir un programme fiable qui puisse évoluer dans le temps. La structuration des actions sera réalisée à partir d'un nombre limité de concepts qu'il convient de parfaitement maîtriser.

LA PROCEDURE : Une procédure est un ensemble d'ordres éventuellement paramétrés.

UNE TACHE : Une tâche est une procédure qui est exécutée simultanément avec un ou plusieurs programmes.

UN BLOC : Un bloc est un ensemble d'ordres formant un tout.

UNE SEQUENCE : Une séquence est une suite d'ordres exécutés l'un après l'autre invariablement une fois.

LA BOUCLE LOGIQUE : Une boucle logique est une suite d'ordres exécutés l'un après l'autre un certain nombre de fois selon la valeur d'une condition.

LA BOUCLE ITERATIVE : Une boucle itérative est une suite d'ordres exécutés l'un après l'autre un nombre bien défini de fois.

LA SELECTION : C'est une suite de plusieurs ordres exécutée selon la valeur d'autant de conditions qu'il y a de telles suites.

L'INTERROGATIVE : L'interrogative est un ordre comprenant un prédicat.

LA SEQUENCE SYNCHRONISEE : C'est une suite d'ordres exécutés une fois à la survenance d'un événement temporel ou à chaque survenance d'un événement, ou encore seulement jusqu'à la survenance d'un événement.

LA BOUCLE SYNCHRONISEE : La boucle synchronisée est une suite d'ordres exécutés indéfiniment jusqu'à la survenance d'un événement.

LA SELECTION SYNCHRONISEE : La sélection synchronisée correspond à plusieurs suites d'ordres dont une ou plus est exécutée selon la survenance d'un événement ou d'un autre.

L'INTERROGATIVE TEMPORELLE : L'interrogative temporelle correspond à un ordre d'accès portant sur un événement.

CHOIX DE STRUCTURE D'ACTIONS

Le choix de la structure d'actions doit être réalisé avec méthode. Il correspond à une démarche basée sur des règles générales.
Chaque ordre doit être compréhensif et naturel.

ANALYSE ARBORESCENTE :

Il convient d'analyser un algorithme décrit en langage courant en le décomposant en une combinaison de structures définies précédemment.

Cette analyse doit être descendante allant des actions principales aux plus petites. Elle procède par un affinage progressif des objets et des actions. De même elle se concrétise par une décomposition fonctionnelle en modules.
A chaque étape, il convient de respecter les règles de toute bonne analyse :

- Règles de faisabilité
- Règles d'évolutivité et de généralité
- Règles de découpage fonctionnel
- Règles de planification
- Règles de fiabilité
- Règles de stratégie de vérification
- Règles de compréhensibilité.

FAISABILITE : Toute action, toute fonction, doit être décomposée afin d'éviter toute ambiguité.

EVOLUTIVITE : La fonction d'un programme est souvent amenée à évoluer dans le temps. Il sera donc nécessaire de bien garder une grande souplesse dans ses algorithmes et dans l'architecture de son programme.

DECOUPAGE FONCTIONNEL : Le rôle essentiel du découpage fonctionnel est d'assurer l'adaptation des objets et des actions, leur cohérence et leur homogénéité. Chaque module devra donc être testé séparément. Son domaine de validité sera vérifié.
Ceci permet d'obtenir une intégration complète de l'ensemble ainsi qu'une optimisation.
Toutefois, avant de chercher à avoir directement un programme efficace, il est préférable de commencer par un programme qui fonctionne. Les plus seront ajoutés par la suite avec la plus grande prudence.

PLANIFICATION : Chaque module doit être planifié dans sa réalisation et dans son implantation avec le plus grand soin.

FIABILITE : Fiabilité et vérification sont des termes qui vont bien ensemble. En effet, la qualité d'un programme se mesure à tous les niveaux :

- Procédure
- Sous-programme
- Programme principal.

COMPREHENSIBILITE : La fonction d'un programme n'est pas seulement son exploitation, son utilisation et son traitement doivent être représentés clairement et doivent être documentés.

Mise en œuvre des intentions : l'analyse arborescente rend compréhensible le rôle des objets et des actions en raison de son concept.

Mise en évidence du déroulement : tout doit être mis en œuvre pour mettre en évidence le déroulement d'un programme :

- Ecriture du programme
- Autodocumentation
- Découpage en :
  - programme
  - sous-programme
  - procédure
  - fonction
  - tâche

CONCLUSION

Nous venons d'aborder un des chapitres les plus importants dans l'analyse-programmation.
La programmation structurée, l'emploi de modules, l'autodocumentation, sont les nuances qui existent entre un amateur au sens péjoratif du terme et un véritable analyste-programmeur.

En fait, la structuration est un pli à prendre dès le départ. Dans le cas contraire, cela pourra sembler difficile à certains de quitter le bricolage pour la rigueur. En réalité, cette rigueur n'est qu'apparente. Elle autorise une bien meilleure qualité dans le travail qui apportera :

- la simplification de la réalisation
- une sécurité accrue
- une homogénéité
- une fiabilité
- l'évolutivité.

Les fiches techniques suivantes permettent d'aborder la structuration de manière plus concrète, et indiquent les marches à suivre au moment de la réalisation d'un programme.

NOTA : La structuration est un tout. Elle s'applique aux objets et aux actions. Il ne suffit pas d'écrire un programme d'une manière aérée et documentée. Cela va beaucoup plus loin. Chaque notion décrite dans les fiches techniques doit être respectée. Enfin, rien ne sert de bien structurer les actions si les objets sont stockés de manière anarchique, si leur accès n'est pas simplifié, si leurs modifications sont impossibles.
Dans tous les cas, une application réalisée sans un minimum de structuration est amenée à mourir à brève échéance. Seule la structuration permet aux programmes d'évoluer dans le temps et leur procure une bien meilleure fiabilité.

LES BOUCLES

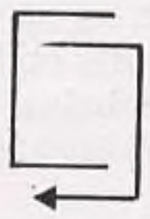
LA BOUCLE : FOR (Faire de x_1 à x_2 telle opération)

La boucle FOR est présente dans la presque totalité des langages
de FOR... NEXT
à FOR... DO.

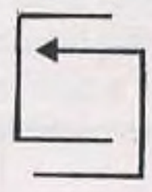
Cette boucle fonctionne de la manière suivante :
Soit x = 25, pour I = 1 à 10, calculer I × x.

MISE EN ŒUVRE DE LA BOUCLE FOR

Branchement vers l'extérieur permis

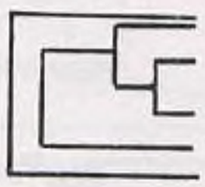


Branchement à l'intérieur interdit



IMBRICATION DE BOUCLES FOR

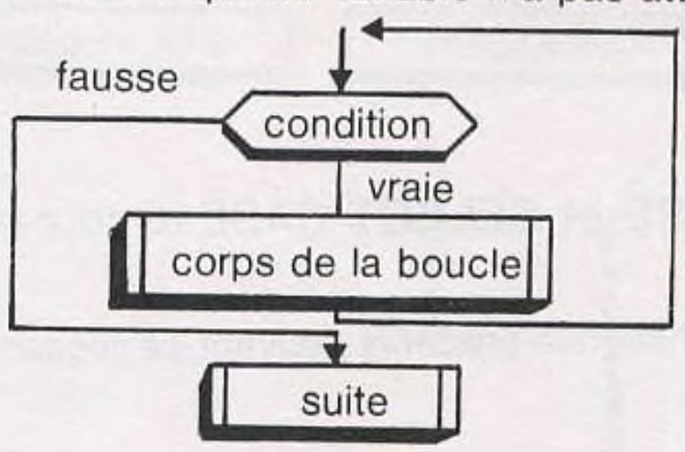
Permis



Interdit

LA BOUCLE : WHILE... DO (Faire tant que...)

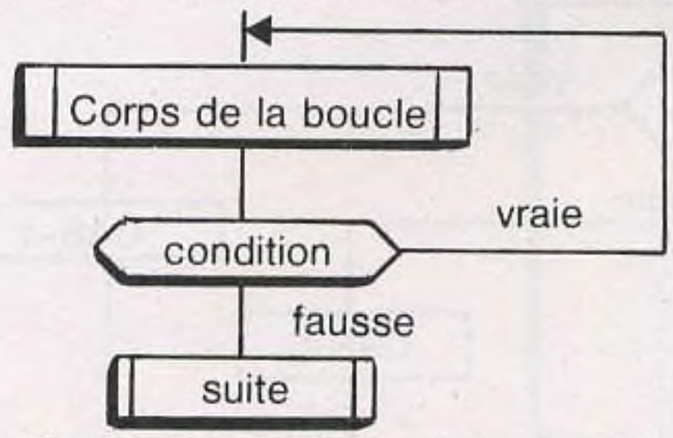
La boucle WHILE... DO est présente dans tous les langages qui offrent une programmation structurée.
Elle revient à dire : faire telle opération tant qu'une variable n'a pas atteint telle valeur.



Test avant le coprs de la boucle.

LA BOUCLE : REPEAT... UNTIL (faire tant que...)

Le principe de base est ici sensiblement différent car le test de conditions est réalisé après le corps de la boucle. Cela revient à dire : faire telle opération tant qu'une variable reste dans un domaine de validité.



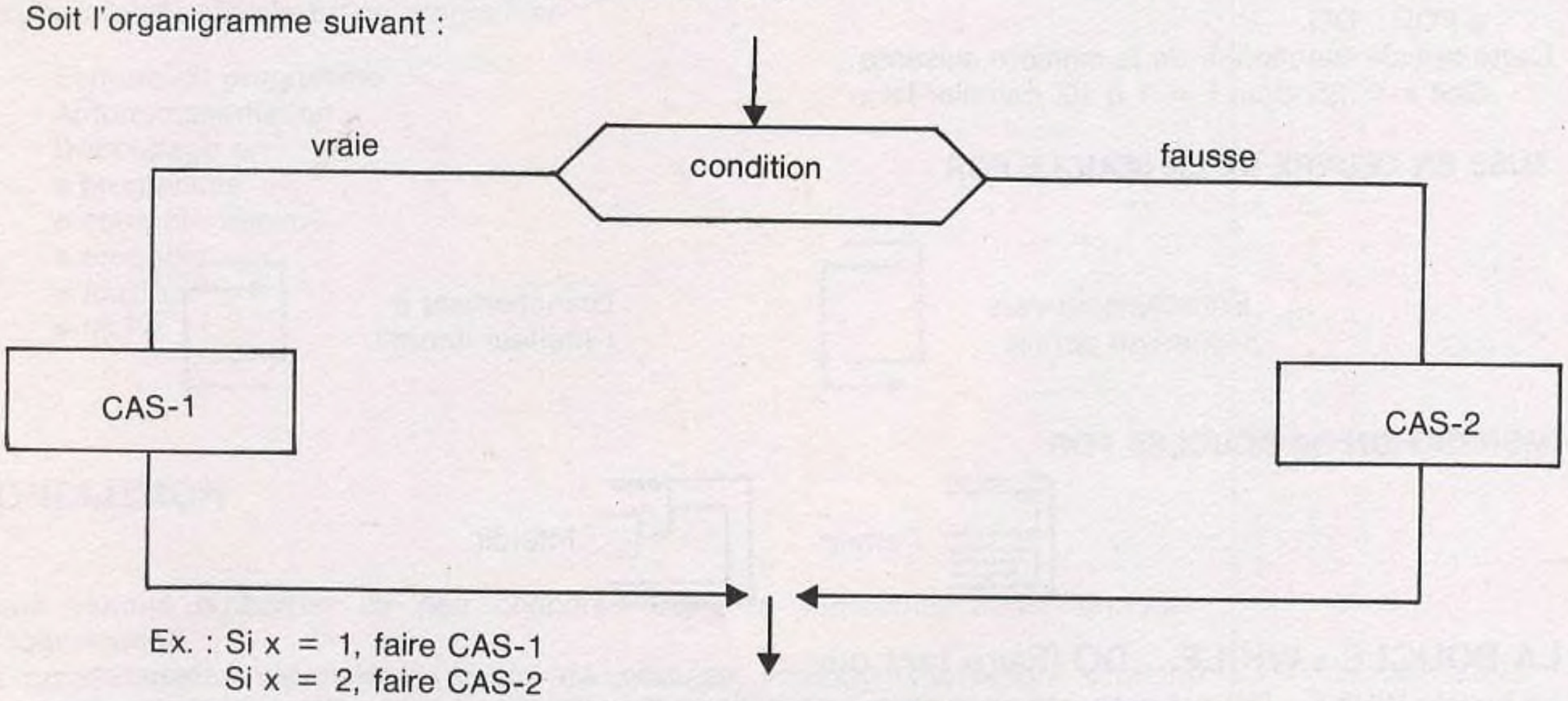
Test après le corps de la boucle.

LES TESTS

Il existe deux types de tests dans la majorité des langages de programmation :
– le test simple où deux cas de figures peuvent se présenter,
– le test en cascade où plusieurs cas de figures (supérieur à 2) peuvent se présenter.

LE TEST SIMPLE : IF... THEN (Si... alors...)

Soit l'organigramme suivant :



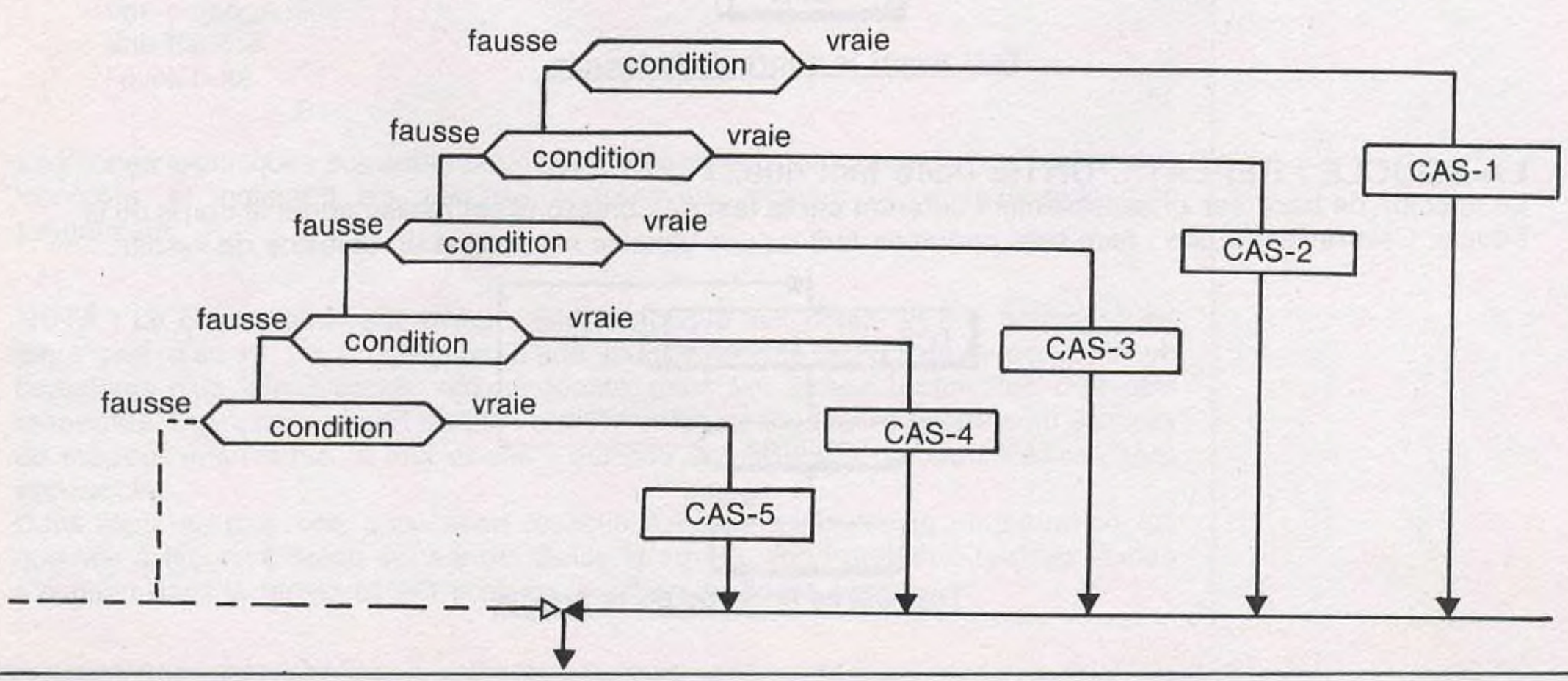
Ex. : Si x = 1, faire CAS-1
Si x = 2, faire CAS-2

Certains tests simples possèdent une option : sinon

Ex : Si x = 1, faire CAS-1
Si x = 2, faire CAS-2
Sinon, faire suite

LE TEST EN CASCADE : CASE et SELECT CASE (Choix en fonction de...)

Soit l'organigramme suivant :
Plusieurs cas appartenant à un ensemble prédéfini peuvent se présenter. Chaque condition est connue à l'avance,



PROGRAMMATION STRUCTUREE

Il existe deux types de programmations :
- la programmation structurée
- la programmation fantaisiste.

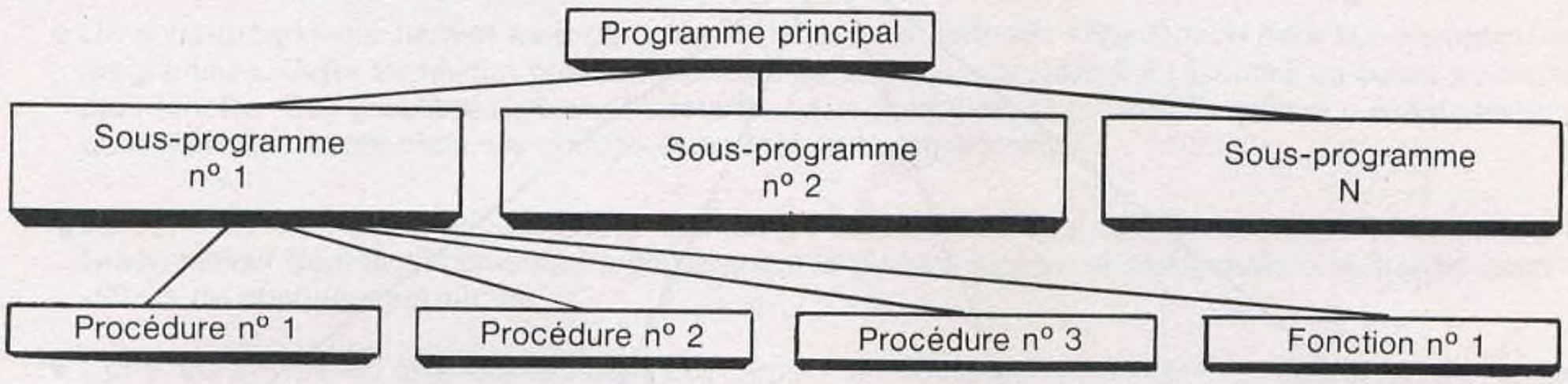
STRUCTURE

La structure d'un programme est un arrangement des différentes instructions qui permettent de mettre en valeur les différents chemins d'exécution possibles.

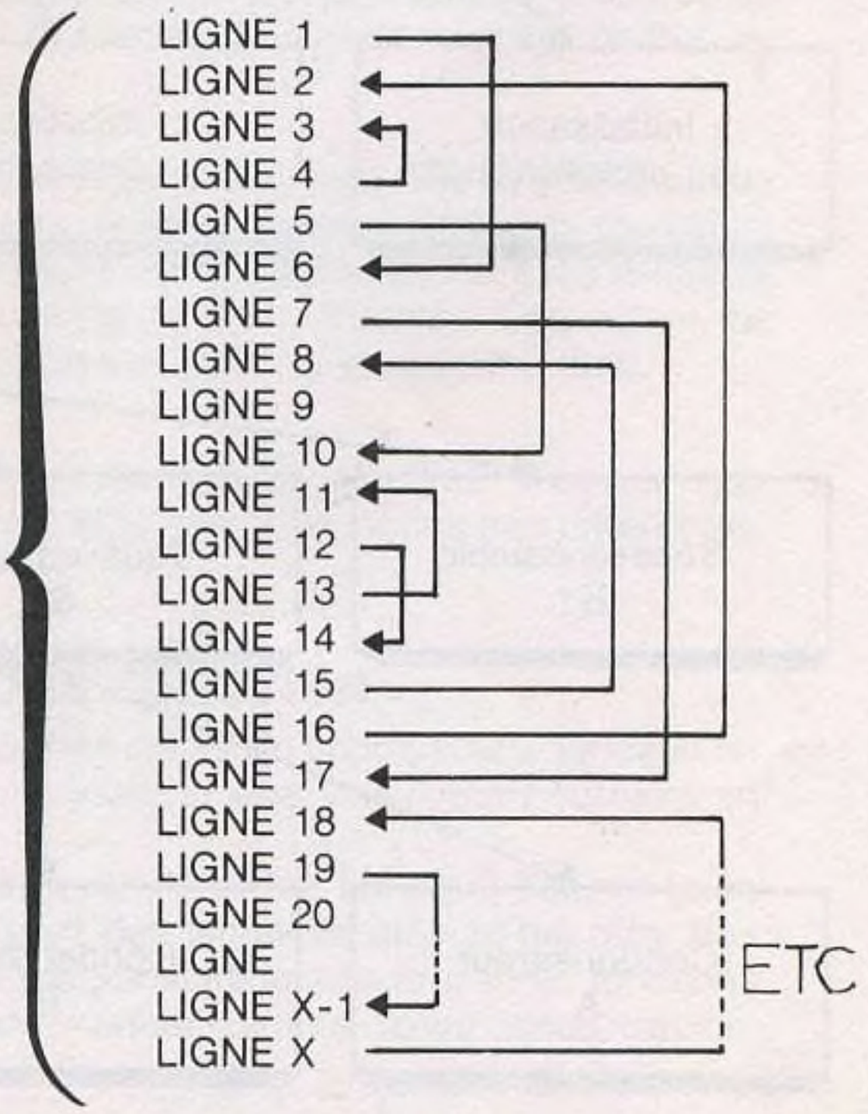
ORIGINE

Dans les premiers temps de la programmation, les applications étaient écrites de manière fantaisiste, et plusieurs choses se déroulaient en même temps sans tenir compte d'une certaine homogénéité. Cela avait pour conclusion une lecture aride des programmes, un nombre non négligeable d'erreurs et une quasi-impossibilité de modifications ultérieures. Le temps a permis de voir arriver sur le marché des langages qui autorisent une structuration claire et précise de chaque application. Le premier de ces langages est sans aucun doute le PL 1. Le plus connu est aujourd'hui le PASCAL qui est devenu la base même de l'apprentissage à la programmation dans les écoles d'informatique. Ce dernier a donné naissance à ADA qui est la référence.

En programmation structurée, nous avons le schéma directeur suivant :



En programmation fantaisiste, nous aurons le schéma suivant :



ARCHITECTURE D'UN PROGRAMME EN ARBRE

Soit un problème de test de circuit imprimé : dans un atelier de fabrication de cartes électroniques, un programme est mis au point afin d'effectuer à l'aide d'un calculateur et d'interfaces, des contrôles en fin de fabrication. Les tests peuvent être réalisés sur plusieurs cartes, différentes mesures sont possibles. Nous aurons donc un programme monté de la manière suivante :

SOUS-PROGRAMME - FONCTION

INTRODUCTION

L'une des constructions les plus puissantes, valable dans tous les langages est le sous-programme (une fonction définie par l'utilisateur n'est qu'un cas particulier de sous-programme). Un sous-programme peut faire tout ce que fait un programme principal mais il doit toujours être «appelé».
Un sous-programme possède un contexte ou état distinct de celui du programme principal. Cela signifie que chaque sous-programme dispose de son propre jeu de variables, de ses touches de fonctions spéciales, de ses propres blocs de données et de ses propres labels de lignes. On peut tirer des bénéfices nombreux d'une bonne utilisation des sous-programmes.

- Un sous-programme permet au programmeur d'utiliser la méthode «Top-Down» pour la conception des programmes. Cette technique provoque le morcellement d'un problème à résoudre en petits problèmes plus simples. Ces problèmes peuvent à leur tour être morcelés, etc. Cette technique a été développée pour aider à la conception, au codage et au test des programmes.

- La séparation en sous-tâches, indépendantes du déroulement du programme, en facilite la lecture. Le programmeur peut ainsi comprendre globalement la tâche à accomplir plutôt que de se perdre dans les détails de chaque sous-tâche.

- L'une des parties les plus longues de la conception de programmes est la recherche d'erreurs et la mise au point du fonctionnement convenable. Ce qui prend le plus de temps est la recherche de la source de l'erreur. L'utilisation de sous-programmes permet d'effectuer des tests de fonctionnement partiel, de découvrir ainsi plus facilement les problèmes et de les corriger.

- Il est courant de rencontrer la même tâche dans plusieurs endroits du programme. Par exemple, si une prise de mesures sur un voltmètre doit être effectuée pour quatre signaux d'entrée différents fournis au circuit à tester, le même sous-programme peut être utilisé pour positionner le voltmètre, faire les mesures, même si certains paramètres doivent être modifiés pour tenir compte du signal d'entrée. Cependant, les sous-programmes peuvent être très utiles dans ce cas pour réduire la taille du programme total.

- Enfin, il est possible de constituer des bibliothèques de sous-programmes d'usage courant. Il se peut que plusieurs utilisateurs aient besoin de sous-programmes identiques dans des applications très différentes.

DIFFRENCES ENTRE UN SOUS-PROGRAMME ET UNE FONCTION

Une fonction est un traitement qui permet d'obtenir un résultat d'une équation. En principe, une fonction ne possède que des paramètres d'entrée. Une fonction sert donc à indiquer une valeur qui peut être un nombre ou une chaîne de caractères.
Une fonction est donc un sous-programme limité à l'obtention d'un résultat. Une fonction, comme toute procédure, doit toujours être définie avant son utilisation. La plupart des langages aujourd'hui offre des fonctions intégrées qui simplifient dans de grandes proportions la programmation. Les premières fonctions pré-programmes ont, sans aucun doute, été les manipulations mathématiques telles sinus, racine, puissance, etc.

SCHEMA SIMPLIFIE D'UN PROGRAMME STRUCTURE

Programme

NOM du programme (liste des noms des fichiers externes)

Déclaration programme
variables,
constantes, etc.

Procédure n° 1

Déclaration procédure n° 1

Début de la procédure

Corps de la procédure n° 1

Fin de la procédure n° 1

Procédure n° 2

Déclaration procédure n° 2

Début de la procédure

Corps de la procédure

Fin de la procédure n° 2

Fonction n° 1

Déclaration fonction n° 1

Début de la fonction

Corps de la fonction

Fin de la fonction

Début du programme

Corps du programme ← Nota : gestion de procédure, calcul, gestion des fonctions, etc.

Fin du programme

LES PROCEDURES

UNE PROCEDURE

Il est possible de concevoir un algorithme particulier, non pas comme un tout, mais plutôt comme un élément d'une construction plus vaste. De ce fait, il sera tentant, pour les problèmes fréquents, de réaliser un module qui pourra servir à de nombreux programmes. Par exemple, au lieu d'écrire un programme qui résoud une équation du second ordre, il sera très intéressant de bâtir un module indépendant utilisable par d'autres programmes. En fait, une procédure est la traduction d'un algorithme précis qui est appelable par une autre procédure.

Ceci autorise la conception et la construction de programmes de façon modulaire, ce qui offre les nombreux avantages suivants :

- Réalisation plus aisée
- Simplificité des modules pris séparément
- Facilité de mise au point
- Modification plus facile
- Contrôles ponctuels
- etc.

PROCEDURES INTERNE ET EXTERNE

La différence essentielle entre une procédure interne et une procédure externe réside dans le fait que la première est réalisée avec le programme principal, la seconde séparément. Dans le deuxième cas, les procédures externes sont compilées séparément. Ceci offre plusieurs avantages.

LES BIBLIOTHEQUES DE MODULES

La réalisation de modules en externe autorise la fabrication de larges bibliothèques de modules qu'il suffira d'aller chercher et de rassembler par un programme principal afin de résoudre une application donnée. Ceci offre d'énormes avantages :

- La fabrication d'un programme est beaucoup plus rapide
- Il est possible de tester séparément chaque module, ce qui permet d'éviter des contrôles fastidieux en aval
- Prix de revient d'une application plus optimisé
- Vitesse de fabrication d'un programme
- Compacité d'une application.

LIENS ENTRE LES PROCEDURES ET LE PROGRAMME

Une procédure est toujours paramétrée avant d'obtenir une parfaite compréhensibilité entre les informations injectées dans la procédure. De même, les résultats obtenus à l'aide de la procédure devront être compris par le reste de l'application. Il s'agit de réaliser une interface entre la procédure et son environnement. Il existe donc trois types de paramètres dans les procédures :

Les paramètres d'entrées : Il s'agit des paramètres figurant dans la définition algorithmique de la procédure

Les paramètres de sorties : Ce sont les résultats fournis par la procédure

Les paramètres intermédiaires : Il s'agit d'informations qui sont utilisables et modifiables par plusieurs procédures, ce sont à la fois des données et des résultats.

Il doit y avoir correspondance absolue entre les arguments et les paramètres en correspondance, soit :

- le même nombre
- les mêmes ensembles de définitions.

Les paramètres d'entrée et de sortie sont connus et paramétrés par avance.

PETITES ANNONCES

Vds Apple IIC + moniteur IIC + souris + logiciels. Matériel sous garantie. Tél. : 45.92.86.46 le soir.

Vds Atari 800 XL + lecteur de disk 1050 + nombreux logiciels sur K7, cartouches et disquettes (jeux et éducatifs) + 4 livres pour 4 000 F. Espinasse 196, rue Beauregard 73000 Chambéry. Tél. : 79.75.04.43.

Vends micro-ordinateur Tandy TRS 80, color 2 Basic étendu 64 k + K7 Tandy CCR 82 + fichier color file. L'ensemble 2 000 F. Tél. : (1) 60.16.72.98.

Vends Micro-Professor MPF1-B avec Basic, RAM 4 k, PIO, 1 100 F. Collection complète Micro-Système : 700 F + port. Nefussy 145 G, ch. de Choulans 69005 Lyon.

A vendre Hector HRX 64 k (Forth + Basic) + disc 2 2 × 800 k (juin 85) + moniteur couleur + imprim. Seikosha GP 100 A + 50 K7 jeux et prog. (Pyrentexte, etc.) + disquettes prog. (Visicalc, etc.) + 15 diskettes + nombreuses doc. Prix : 12 000 F. Tél. : 56.21.86.98 ap. 20 h.

Vends TRS mod. III 48 ko + 2 drives + progs utilitaires (compta., stock, etc.) + jeux : 8 000 FF + en prime imprimante Logabax 132 colonnes, 120 cps à réparer. Vends TRS mod. 1 16 ko avec clavier minuscules accentuées + progs : 2 000 FF. Durr Michel 18, rue Laperouse 31120 Portet/Garonne. Tél. : 16-61.72.23.18.

• Vous avez de bonnes bases en électronique

• Vous savez (et vous aimez) rédiger dans un très bon français

• Vous êtes la femme ou l'homme que recherchent les Editions Fréquences dans la perspective de son développement.

— expérience exigée ? Non

— dynamisme et sens des responsabilités ? Oui

---

ne nous téléphonez pas mais envoyez-nous un bout d'essai (quelques pages) rédigé sur le thème de votre choix (BF, électronique générale, micro-informatique, etc.), ainsi que votre curriculum vitae

---

— place stable et possibilité de carrière.

Editions Fréquences 1, boulevard Ney
75018 Paris

BON DE COMMANDE

Pour compléter votre collection de Led-Micro

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

Je désire le n° ☐ (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 22 23

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 n° .....

Nom ..... Prénom .....

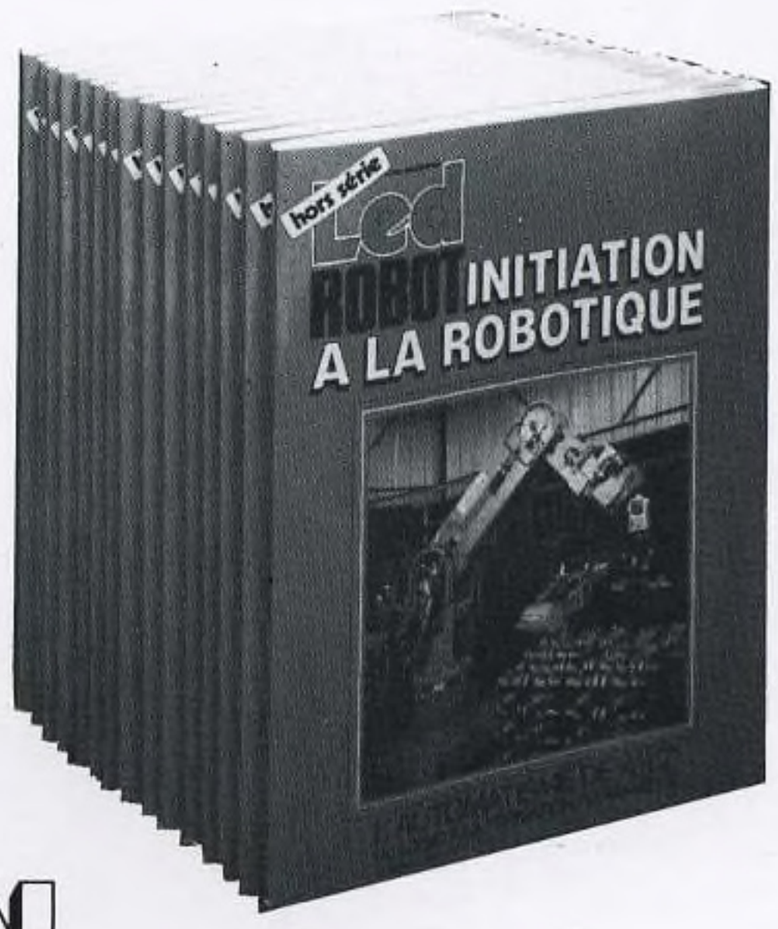
N° ..... Rue .....

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 — 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.

Alice.

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]