hors série

Led MICRO

PROGRAMMATION
COURS 2ème CYCLE

ISSN 0757-6889

COURS
N° 21
Suite
2e cycle

N° 1

Le HX-10 de Toshiba

M 1988 - N° 21 - (1) - 18 F

MENSUEL JUIN 1985 - BELGIQUE 139 FB/CANADA 3,75 $/SUISSE 6,75 FS

UN PREMIER LEXIQUE ANGLAIS-FRANÇAIS VRAIMENT PRATIQUE ET TRÈS COMPLET

+ de 1500 termes !

- Index français-anglais
- Lexique des termes anglais et américains avec explication en français
- Tables de conversion

JEAN HIRAGA

lexique de l'électronique anglais-français

Pour la première fois en électronique, un lexique anglais-français présenté sous forme pratique avec en plus des explications techniques succinctes mais précises.

112 pages
PRIX : 65 F

En vente chez votre libraire et aux Editions Fréquences

BON DE COMMANDE

Je désire recevoir le livre «le lexique de l'électronique anglais-français» au prix de **72 F** (65 F + 7 F de port). Adresser ce bon aux EDITIONS FREQUENCES 1, bd Ney, 75018 Paris.

Nom .....

Prénom .....

Adresse .....

.....

Code postal .....

Règlement effectué
☐ par CCP ☐ par chèque bancaire
☐ par mandat

éditions fréquences
COLLECTION Led LOISIRS

DÉJA PARUS
DANS LA MÊME COLLECTION

«Les lecteurs de compact-discs» au prix de **130 F** + 10 F de port.	«17 montages électroniques» au prix de **95 F** + 10 F de port.
«Conseils et tour de main en électronique» au prix de **68 F** + 7 F de port.	«Filtres actifs et passifs pour enceintes acoustiques» au prix de **85 F** + 7 F de port.

hors série
Led MICRO

Nous signalons à nos lecteurs que les Editions Fréquences seront fermées du 1er au 31 juillet 85 pour congés annuels.

PROGRAMMATION COURS 2e CYCLE

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

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

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

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

Secrétaire de Rédaction
Chantal Cauchois

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

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

Réalisation
Composition-Photogravure
Edi'Systèmes

Impression
Berger-Levrault - Nancy

JUIN 85

COURS
N° 21
Suite
2e cycle
N° 1

NOTRE COUVERTURE : Le micro-ordinateur familial MSX de Toshiba : le HX-10 de 64 K.

LE DEUXIEME SOUFFLE

Voici deux ans paraissait le premier numéro de Led Micro. Cette publication avait pour vocation d'être un supplément bimestriel de micro-informatique à notre revue Led (Loisirs Electroniques D'aujourd'hui).
Le succès de la formule dû au talent de nos rédacteurs Claude Polgar et Philippe Duquesne fut instantané !

Des centaines de lettres nous incitaient dans les jours qui suivirent la première parution à devenir mensuel. Nous acceptions la gageure et de «supplément», Led Micro devenait une revue à part entière que plus de 30 000 lecteurs suivaient avec enthousiasme.
Ces lecteurs avaient découvert un cours vivant où chaque étape était pesée et non brûlée...
Le «basic en trois semaines» ou «en cinq leçons» n'était pas le chemin que suivait Led Micro, nous avions raison. Ce 21e numéro porte en surcharge le numéro 1 du cours de programmation 2e cycle.
Nous sommes persuadés que cette série confiée à Dominique Chastagnier, Jean-François Coblentz Charles-Henry Delaleu et Patrick Gueneau aura le même succès auprès de ceux qui ont été formés par les 20 premiers numéros. Ces 20 premiers numéros font l'objet actuellement d'une édition en 2 tomes pour le cours d'initiation à la programmation et au basic de Claude Polgar et en 2 tomes également pour l'initiation à l'électronique digitale et au microprocesseur de Philippe Duquesne.

Un troisième tome est actuellement en préparation par Claude Polgar, nous en prévoyons la sortie fin octobre. Dans ce troisième tome une fois de plus seront passées au peigne fin **toutes les indispensables étapes du très difficile niveau 1.**
Notre ami Claude Polgar s'en explique d'ailleurs dans l'éditorial ci-contre.
Philippe Duquesne, quant à lui, prépare un troisième tome également sur l'application des microprocesseurs, ce troisième tome étant une suite logique à ses deux cours précédents sur l'électronique digitale et sur les microprocesseurs.

L'éditeur
Edouard Pastor

CONTINUITE ET CHANGEMENT

Voici déjà deux ans que nous nous sommes lancés dans l'aventure Led Micro : «Enseigner l'informatique en ayant le courage de commencer par A pour être sûrs d'arriver à Z». Que les choses ont changé en deux ans !
A cette époque, la micro-informatique était l'Eldorado, et ce fut la ruée vers ce nouveau pactole.
Ne parlons pas des heurs et malheurs des constructeurs de matériels et des créateurs de logiciels : cordonnier, pas plus haut que la chaussure ! Contentons-nous de regarder ce qui s'est passé dans notre métier à nous, l'édition.

On a vu se créer des quantités de revues de micro-informatique familiale : actuellement si vous achetez toutes les publications d'informatique des kiosques, il vous faudra dépenser 1 100 francs par mois. Et pour quoi faire ? Pour relire 20 fois le compte rendu d'une même exposition ou le banc d'essai du dernier micro.

Beaucoup de ces nouvelles revues sont bien faites, flattent le lecteur par leurs quadrichromies et leur optimisme délirant : «l'Informatique c'est facile». Un exemple : une de ces revues a commencé un cours pour débutants. A la fin du premier cours (de 10 pages), l'auteur a osé écrire : «Maintenant que nous savons programmer nous allons, dans le prochain numéro, apprendre à manipuler les fichiers».
Lorsqu'on sème l'illusion, il faut s'attendre à récolter la désillusion. Elle est venue, brusquement, en février 85. Les ventes des revues de micro-informatique se sont effondrées. Le mythe de l'informatique de loisir facile s'est dégonflé. Parallèlement, les annonceurs ont préféré placer le budget de publicité de leurs matériels grand public dans les revues grand public. Moins de lecteurs, moins de pub, c'est la déroute ! On peut prévoir la disparition d'une grande partie de ces revues dans les mois qui viennent. Un «assainissement du marché», comme on dit.

Led Micro subit, comme tout le monde, le contrecoup de la fin de cet engouement assez irrationnel. Mais il résiste beaucoup mieux pour deux raisons :
– d'une part il est resté le seul sur son créneau «l'enseignement sérieux» ;
– d'autre part il a toujours largement privilégié la partie rédactionnelle sur la partie publicitaire.
Combien de revues peuvent, comme nous, équilibrer leur budget sans courir après la publicité ?
L'ère des 20 revues de micro-informatique traitant toutes de tous les sujets, et paraissant copiées les unes sur les autres s'achève. Il y a de la place en France pour un maximum de 2 à 3 revues «tous azimuts». Les autres revues devront se spécialiser pour satisfaire le mieux possible les besoins précis d'un public bien «ciblé». Choisir un créneau et un seul.

Led Micro veut donner l'exemple.
Dès la rentrée, il «cultivera sa différence» et restera sur son créneau : la pédagogie.

Et même, nous allons plus loin dans la spécialisation. Le courrier des lecteurs nous a montré que Led Micro touchait deux publics :
– un public de «débutants»
– un public qui ne l'est plus.
C'est ainsi que beaucoup de nos lecteurs motivés (peut-être grâce à nous !) ont voulu aller plus vite et, par des moyens divers (clubs, stages, travail personnel...) m'ont dépassé. Je m'épuisai à satisfaire les exigences contradictoires de mes deux lectorats : «plus vite ! pas si vite !»
Il a fallu trancher. C'est ce que fit notre éditeur de la façon suivante :

D'une part, à partir du présent numéro 21, Led Micro passe à la vitesse supérieure, sous la direction technique d'une nouvelle équipe de haut niveau.

D'autre part, je continue mon petit bonhomme de chemin avec mes fidèles de «niveau 1» pour les amener comme prévu et **au même rythme** à une bonne maîtrise du basic (vers juin 1986) puis (vraisemblablement) à la pratique des fichiers et du graphisme à partir de septembre 86... puis à la pratique des tableurs... Mais tout ceci dans un cadre différent : des volumes d'environ 80 pages à parution trimestrielle. Mes cours des numéros 1 à 20 de Led Micro vont être regroupés en 2 volumes, le 3e volume paraîtra fin octobre 1985.

Différents sujets «de niveau 1» que j'avais envisagé de publier dans Led Micro vont devoir être repensés pour, vraisemblablement ressortir sous une forme ou sous une autre. Je pense en particulier au cours de Pascal, au cours de Logo, à la série «Traductions dans les autres Basics» et à la rubrique «Réseaux et banques de données amateur».
Des exercices de récapitulation similaires à ceux que vous connaissez vont illustrer le 3e volume. Les solutions paraîtront dans le tome 4. En cas d'urgence, je ferai des communiqués dans le Led Micro de la nouvelle équipe.
Continuez à m'envoyer solutions, suggestions, critiques et/ou encouragements : j'en tiendrai le plus grand compte.

Claude Polgar

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

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

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

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

COURS DE BASIC

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Bien que la plupart des lecteurs soient désormais aptes à «accrocher» les cours d'un niveau supérieur, nous avons jugé utile (et ce, sur quelques numéros) de publier un cours complémentaire d'initiation afin que tout ce qui concerne les instructions du basic soit complètement et définitivement étudié.
Ce mois-ci, nous vous présentons la structure fondamentale de tableaux, que certains connaissent déjà, mais ce sont ceux qui avaient été plus vite que le cours de Claude Polgar.

LES TABLEAUX EN BASIC

Introduction

Le BASIC tel que vous le connaissez actuellement ne vous permet pas de gérer des grandes quantités de données de même nature, car vous ne savez pas les stocker, ou des données en nombre variable, car vous ne pouvez envisager tous les cas, un par un, dans votre programme. Il est facile de vous donner des exemples. Tout d'abord dans le cas des tableurs, on retrouve une structure de tableau, telle que nous allons vous la présenter. Ensuite, dans le cas où vous souhaitez programmer un jeu, il vous faut demander le nombre de joueurs, et la suite du jeu en dépend, c'est-à-dire que sans les tableaux, il vous faudrait programmer tous les cas. Programmer tous les cas, cela signifie en programmer un certain nombre, donc être obligé de le modifier lorsque le nombre de joueurs est imprévu. Un tableau est une structure qui présente des rapports très étroits avec les boucles, que vous étudiez maintenant depuis plusieurs mois.

1. La structure, en BASIC

1.1. Tableaux uni-dimensionnels

Un tableau est une boîte disposant de cases en aussi grand nombre que vous le désirez, puisque ce nombre est fixé par votre programme. Pour introduire le plus clairement possible la notion de tableau, il est indispensable de comprendre qu'un tableau est un ensemble de cases, chaque case étant une variable tout ce qu'il y a de banale. La seule différence est l'étape de déclaration qui se fait de la manière suivante :

```
10 DIM TABLEAU (10)
```

et qui correspond à la déclaration d'une boîte contenant dix cases. Le stockage de données dans le tableau, se fait comme pour une variable simple :

```
100 TABLEAU(6) = 12
```

qui place la valeur 12 dans la sixième case de tableau.

Maintenant, observez l'adaptation du début d'un jeu. Ce jeu a un nombre variable de joueurs.

Tout d'abord, vous stockez le nombre de joueurs, puis leur nom. Sans tableau vous devez en passer par une solution du type :

```
7   REM*******************************
8   REM  ENTREE DU NOMBRE DE JOUEURS
9   REM*******************************
10  INPUT " NOMBRE DE JOUEURS : ";NJ
20  FOR I=1 TO NJ
30  PRINT " NOM DU JOUEUR N° ";I
35  INPUT A$
40  ON I GOTO 110,120,130,140,150
99   REM ************************
100 REM STOCKAGE DES NOMS
101  REM ************************
110  A1$=A$
115  GOTO 200
120  A2$=A$
125  GOTO 200
130  A3$=A$
135  GOTO 200
140  A4$=A$
145  GOTO 200
150  A5$=A$
200  NEXT
300  suite ...
```

Vous voyez donc que pour jouer à 6, vous serez obligés de modifier votre programme. Si vous vous retrouvez un jour à dix, cela devient long à modifier. Avec un tableau, vous auriez cela :

```
7   REM*******************************
8   REM  ENTREE DU NOMBRE DE JOUEURS
9   REM*******************************
10  INPUT " NOMBRE DE JOUEURS : ";NJ
15  DIM NOM$(NJ)
20  FOR I=1 TO NJ
30  PRINT " NOM DU JOUEUR N° ";I
35  INPUT NOM$(I)
40  NEXT
```

C'est plus simple, plus court et il n'y a pas d'adaptation à réaliser, donc il serait dommage de s'en priver. C'est pourquoi il est nécessaire de bien comprendre le fonctionnement et l'utilisation des tableaux.
Dans le programme précédent, le nombre de joueurs est connu en ligne 10. Cela signifie que vous avez, à partir de cette ligne une information sur la quantité d'informations à conserver. Pour chaque joueur, vous devez par exemple connaître son nom, son prénom, son score... A la ligne 15, un tableau est donc créé, possédant le nombre de cases exact nécessaire à la partie. Il est évident que vous auriez pu le dimensionner, à l'avance, par une valeur fixe de votre choix, mais ici, le programme s'adapte vraiment à vos besoins. Puis vous voyez qu'avec quatre lignes de programme, vous pouvez stocker tous les noms des joueurs. On utilise ici une boucle, notion très étroitement liée à celle de tableau, car il vous faut connaître les boucles

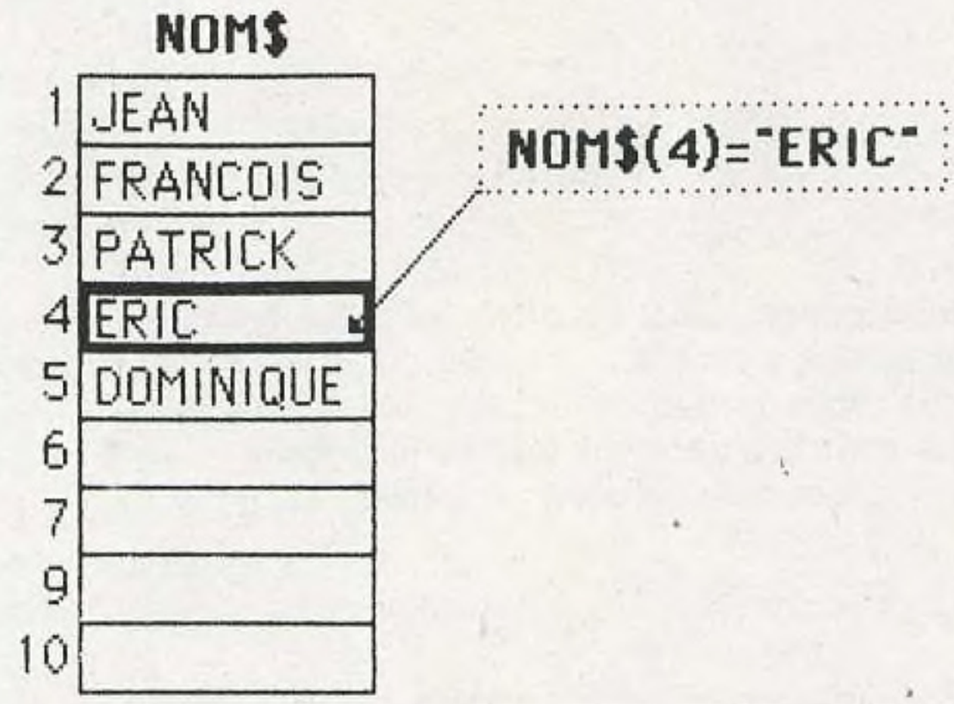


Figure 1

pour utiliser correctement les tableaux, comme l'exemple précédent l'illustre, pour un cas très simple. Ici, I prenant successivement les valeurs 1, 2..., jusqu'à I = NJ, le nombre de joueurs, le nom des joueurs se trouve stocké l'un après l'autre, comme le montre la figure 1.
Un tableau est donc un ensemble de cases dans la mémoire, dans lequel vous pouvez mettre des informations, informations que vous pourrez récupérer à votre guise. En effet, pour récupérer une donnée, il vous suffit de reprendre la case du tableau qui vous intéresse.
Maintenant reprenons les étapes en détail. Tout d'abord, créer le tableau qui va s'appeler ici ESSAI et qui aura 31 cases (pourquoi pas !!!).

```
10 DIM ESSAI(31)
```

Voilà qui est fait, dans les règles de l'art. Maintenant, il faut mettre des informations dans ce tableau, ce que nous allons faire immédiatement :

```
20 FOR I = 1 TO 31
30 ESSAI(I) = I
40 NEXT
```

Nous voilà donc avec un tableau, dont le nom est ESSAI et dont la première case contient la valeur 1, la deuxième case contient la valeur 2 et ainsi de suite jusqu'à la case 31 qui contient, vous l'aurez deviné, la valeur 31. Vous voyez que chaque case d'un tableau se manipule comme une variable simple.

REMARQUES IMPORTANTES :

1. Un tableau, comme une variable, peut contenir des réels, des entiers, des chaînes de caractères. Le nom de ce tableau doit en tenir compte, comme pour des variables, c'est-à-dire suivre les règles traditionnelles, qui sont, rappelons-le l'utilisation des suffixes % et $ pour les entiers et les chaînes de caractères respectivement.
2. Un tableau ne peut être dimensionné qu'une seule fois dans un programme, sous peine de message d'erreurs à l'exécution, car le programme s'arrêtera.
3. La longueur du nom d'un tableau suit les mêmes règles que celles appliquées aux variables simples et dépend donc du BASIC que vous utilisez, ce qui signifie par exemple sur APPLE II que les tableaux TABLEAU et TAPIS sont un seul et même tableau TA aux yeux du programme. Cela entraîne donc, par exemple, une erreur de redimensionnement du tableau TA. Mais TABLEAU et TAPIS$ ne représentent pas la même structure aux yeux du progamme (voir la remarque 4, qui précise ce point).

4. A titre de renseignement général, les entités suivantes sont toutes différentes :
   AA variable réelle
   AA% variable entière
   AA$ chaîne de caractères
   AA(12) 12e case d'un tableau de réels
   AA%(12) 12e case d'un tableau d'entiers
   AA$(12) 12e case d'un tableau de chaînes

1.2. Cas multi-dimensionnel

Les tableaux peuvent vous permettre d'en faire encore plus. En effet, telle une boîte, pour reprendre l'image du début, qui peut avoir des cases les unes en dessous des autres, mais aussi les unes à côté des autres, vous pouvez avoir des tableaux plus compliqués que ceux sur lesquels nous nous sommes penchés depuis le début.
Nous vous proposons plus loin un programme d'exemple utilisant un tableau du type suivant:

INFO$

	1	2	3
1	DURAND	JACQUES	(1) 777.77.77
2	DUPONT	PIERRE	(3) 983.00.00
3	DUGOMMIER	PAUL	(86) 34.11.11
4	MARTIN	HENRI	(75) 36.20.00
5	...	..	...
6	...	...	...
7	...	...	...
8	...	...	...
9	...	...	...
10	...	...	...

INFO$(3,2)="PAUL"

Figure 2

Voyez donc qu'un tableau peut avoir plusieurs «indices», ces nombres qui permettent de numéroter les cases. Vous avez ici un exemple de tableau à deux dimensions, sur lequel une case est repérée par deux nombres, l'un pour sa position horizontale, 2 dans l'exemple, l'autre pour sa position verticale, sa ligne, 3 dans l'exemple. Le nombre maximal de dimensions est largement suffisant, variant de 8 à 88 (!!!) selon les machines.
Pour l'exemple précédent, la déclaration se fera par l'instruction suivante :

```
10 DIM INFO$(10,3)
```

signifiant que ce tableau possède 30 cases, réparties en trois colonnes de 10 cases.
Un élément de ce tableau sera INFO$(8,2) par exemple, et sur le dessin 2, PAUL correspond à l'élément dont le nom est INFO$(3,2), on dit encore que l'indice de ligne est 3 et l'indice de colonne est 2.
La deuxième partie du cours se base sur un programme d'application que nous vous proposons, qui vous permet d'avoir un petit répertoire téléphonique, que nous ferons évoluer au fil des cours, mais que vous-même pourrez facilement améliorer. N'hésitez pas à nous envoyer vos idées, vos améliorations.

2. Application : un répertoire

2.1. Le programme lui-même

```
10 dim info$(20,3)
20 dim stock(10)
30 nbinf = 0
```

```
90 rem ***************
91 rem choix des options
92 rem***************
100 print "type de travail à effectuer"
110 print "    1: ajout d'une référence"
120 print "    2: effacement d'une référence"
130 print "    3: recherche d'un numéro"
140 input "              votre choix";choix
200 on choix goto 1000,2000,3000
990 rem ********************
991 rem ajout d'une référence
992 rem ******************
1000 input "nom à ajouter";nom$
1010 input "prénom";prénom$
1020 input "numéro de téléphone correspondant";numéro$
1030 nbinf = nbinf + 1
1040 info$(nbinf,1) = nom$
1050 info$(nbinf,2) = prénom$
1060 info$(nbinf,3) = numéro$
1990 rem ***********************
1991 rem retrait d'une référence
1992 rem ***********************
2000 input "nom à éliminer";nom$
2010 j=0
2100 rem ************************************************
2101 rem existence de ce nom, et nombre de fois où sa présence est
      détectée ( comptée dans j , les indices correspondants sont
      stockés dans le tableau appelé stock)
2102 rem ************************************************
2110 for i = 1 to nbinf
2120 if info$(i)=nom$ then j=j+1 : stock(j) = i
2130 next i
2200 rem ************************************************
2201 rem traitement en fonction du nombre de référence détectées
2202 rem ************************************************
2210 if j = 1 then 2300
2220 if j=0 then print "pas de référence à ce nom " goto 100
2230 if j > 1 then input " précisez le prénom" prénom$
2240 for i = 1 to j
2250  if info$(stock(j),2) = prénom$ then for k = 1 to 3 :
      info$(stock(j),k) = " " : next k
2260 next i
2270 goto 100
2300 if j = 1 then for k = 1 to 3 : info$(stock(j),k) = " " next k
2990 rem ***********************************
2991 rem recherche d'un numéro téléphonique
2292 rem ***********************************
3000 input " nom à rechercher "; nom$
3010 j=0
3100 rem ************************************************
3101 rem existence de ce nom, et nombre de fois où sa présence est
      détectée ( comptée dans j , les indices correspondants sont
      stockés dans le tableau appelé stock)
3102 rem ************************************************
3110 for i = 1 to nbinf
3120 if info$(i)=nom$ then j=j+1 : stock(j) = i
```

```
3130 next i
3200 rem ************************************************
3201 rem traitement en fonction du nombre de référence détectées
3202 rem ************************************************
3210 if j = 1 then 3300
3220 if j=0 then print "pas de référence à ce nom ": goto 100
3230 if j > 1 then input "  précisez le prénom", prénom$
3240 for i = 1 to j
3250 if info$(stock(j),2) = prénom$ then print info$(stock(j),3)
3260 next i
3270 goto 100
3300 if j = 1 then print info$(stock(j),3)
4000 end
```

2.2. Description du programme

Les premières lignes servent à déclarer le tableau principal INFO$, qui contiendra tous les renseignements, ainsi que le tableau STOCK, qui sert localement à connaître dans quelles lignes de INFO$ se trouvent les noms rcherchés, et NBINF qui est le nombre de noms stockés. Les lignes 100 à 200 servent à définir le type de travail à effectuer, la ligne 200 servant à orienter la suite en fonction du choix retenu. Les lignes 1000 et suivantes servent à ajouter une référence, les lignes 2000 et suivantes à retirer une référence, sans optimiser le programme, c'est-à-dire sans utiliser la place devenue vacante et les lignes 3000 et suivantes, à interroger les données, pour obtenir un numéro. Dans ces deux dernier ças il est nécessaire de connaître le prénom si le nom apparaît plusieurs fois, afin de ne pas effacer une mauvaise référence, ou de ne pas fournir un numéro erroné. Ce travail est effectué aux lignes 2100 à 2160 et 3100 et 3160, respectivement, le parallélisme servant à vous montrer l'identité des traitements, mais nous en reparlerons dès le prochain cours. Une dernière remarque, concernant la typographie ; le programme a été tapé en minuscules. Généralement vous n'en disposez pas sur vos micro. Ce programme a été réalisé sur un APPLE II, équipé d'un clavier modifié, qui les autorise mais, pour vous, rien ne se trouve modifié, donc pas d'inquiétudes particulières, il vous suffit de taper tout en majuscules.

2.3. Les commentaires indispensables

Ce programme est uniquement une ébauche, et n'a pas d'autre ambition que d'être un exemple, et éventuellement une base pour un travail sérieux. C'est pourquoi il est très commenté, structuré sans excès pour rester lisible. En bref, il a les avantages et les inconvénients de tous programmes de ce type, c'est un compromis.
Ce programme sera suivi dans la suite de nos cours, afin de l'améliorer, de permettre un stockage définitif, et non comme ici, un stockage qui s'effacera lorsque vous éteindrez votre appareil, et autres améliorations que nous prévoyons ou que vous pourrez nous proposer. Une amélioration notable est par exemple de décaler les données lorsque vous en effacez une au milieu du tableau, laissant une ligne vide et inoccupée, ce qui est une perte de place. Une autre sera de trier les noms dans le tableau, de manière à limiter la recherche d'un numéro. Toutes vos suggestions seront analysées et insérées dans les prochains cours, si vous nous envoyez la modification car c'est à vous de programmer pour vous entraîner.
Ce programme est suffisamment général pour s'adapter à toute machine. Les extensions ne pourront pas toujours l'être, et c'est pourquoi, pour la première étape, il est resté simple, et incomplet. En particulier, une modification sur l'affichage ne sera généralement pas directement écrite pour toutes les machines.

3. Prévisions pour les prochains cours

Le prochain cours traitera de la structuration des programmes, avec la notion de sous-programmes et la façon de s'en servir avec efficacité. Le cours suivant servira à réaliser des exercices d'applications sur les notions traitées jusqu'alors. Par la suite, nous aborderons l'utilisation des fichiers, pour reprendre l'excellent cours paru dans le coin des fortiches et dû à Bruno Lilamand. Cela servira à conclure le cours sur la description des commandes de bases du BASIC.

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.

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

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).

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

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

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

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

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Voici le premier cours de programmation approfondie. Dans cette série, nous vous donnerons les éléments indispensables à la rédaction d'un programme efficace. Le premier cours a surtout pour but de vous sensibiliser à l'un des gros problèmes de la micro-informatique, la taille réduite de la mémoire disponible.

COURS N° 1

Entiers et réels, description de l'implantation en machine

PLAN DU COURS

1. Codage des nombres
   1.1. Codage des entiers
      1.1.1. Introduction
      1.1.2. Le codage des entiers non signés
      1.1.3. Le codage des entiers signés
         1.1.3.1. Cas des entiers positifs
         1.1.3.2. Cas des entiers négatifs
   1.2. Codage des réels
      1.2.1. Introduction
      1.2.2. Format de codage
      1.2.3. Limites du format défini
2. Optimisation des programmes
   2.1. Optimisation de la place mémoire
   2.2. Amélioration du temps d'exécution
3. Exemples d'applications
   3.1. Programme de codage d'un entier positif
   3.2. Programme de codage d'un entier négatif
   3.3. Codage d'un entier quelconque
   3.4. Utilisation de VARPTR pour connaître le codage des réels
4. Problèmes liés aux codages présentés
   4.1. Exemples pratiques
   4.2. Codages différents
5. Chaînes de caractères
6. Conclusion générale sur les codages

1. CODAGE DES NOMBRES

1.1. Codage des entiers

1.1.1. Introduction

Les entiers sont déclarés en BASIC par un symbole particulier, placé en suffixe du nom déclaratif de variable. Ce suffixe est le signe % (figure 1.1.1.1).

Pour déclarer une variable entière A, ce suffixe sera donc placé de la façon suivante : A % (comme la plupart des suffixes).
Dans tout programme BASIC, la déclaration est du même type, et ce quel que soit votre ordinateur. Il s'agit d'une norme unanimement respectée, ce qui prouve qu'il en existe !

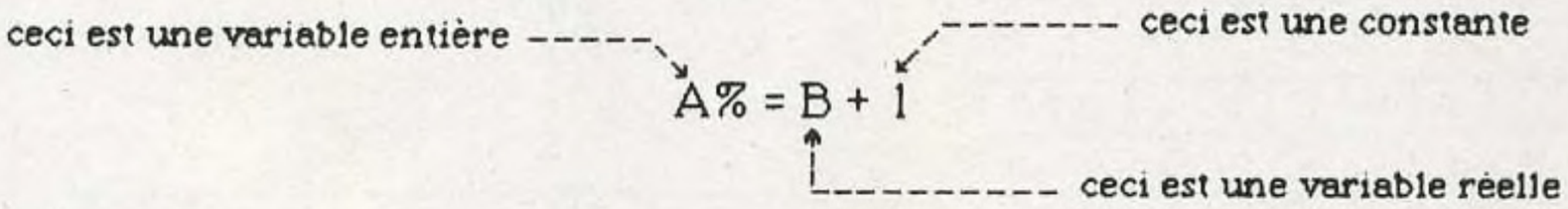


Figure 1.1.1.1

on peut alors envisager le programme suivant

```
10  B = 0.5
20  A% = B + 1
30  PRINT A%
```

vous aurez alors la sortie suivante sur votre écran

```
 1
```

en effet, A% est un entier, donc l'ordinateur coupe la partie décimale pour ne garder que la partie entière

1.1.2. Le codage des entiers non signés

En BASIC, les entiers sont codés sur 2 octets (figure 1.1.2.1), ce qui permet de coder 2^{16} nombres. Comme il est indispensable d'avoir le 0 (zéro), ces nombres peuvent aller de 0 à $2^{16}-1$ (soit 65535, vous pouvez nous croire sur parole). Avec ce type de méthode, il sera possible d'avoir un codage dont des exemples sont fournis en figure 1.1.2.2.

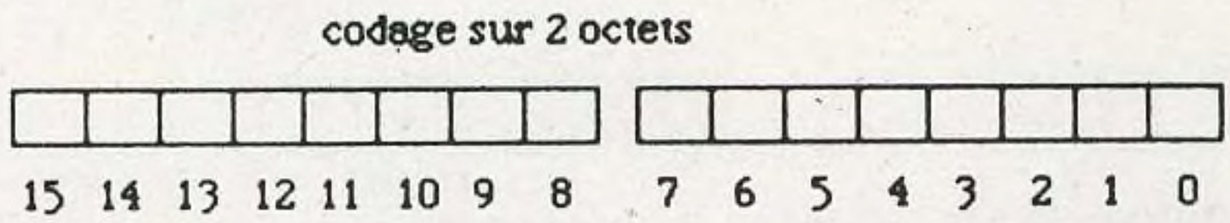


Figure 1.1.2.1

Hélas, trois fois hélas, il est parfois nécessaire d'utiliser des nombres entiers négatifs (des éléments de Z^-). En conséquence, un autre codage est adopté.

exemples de codage

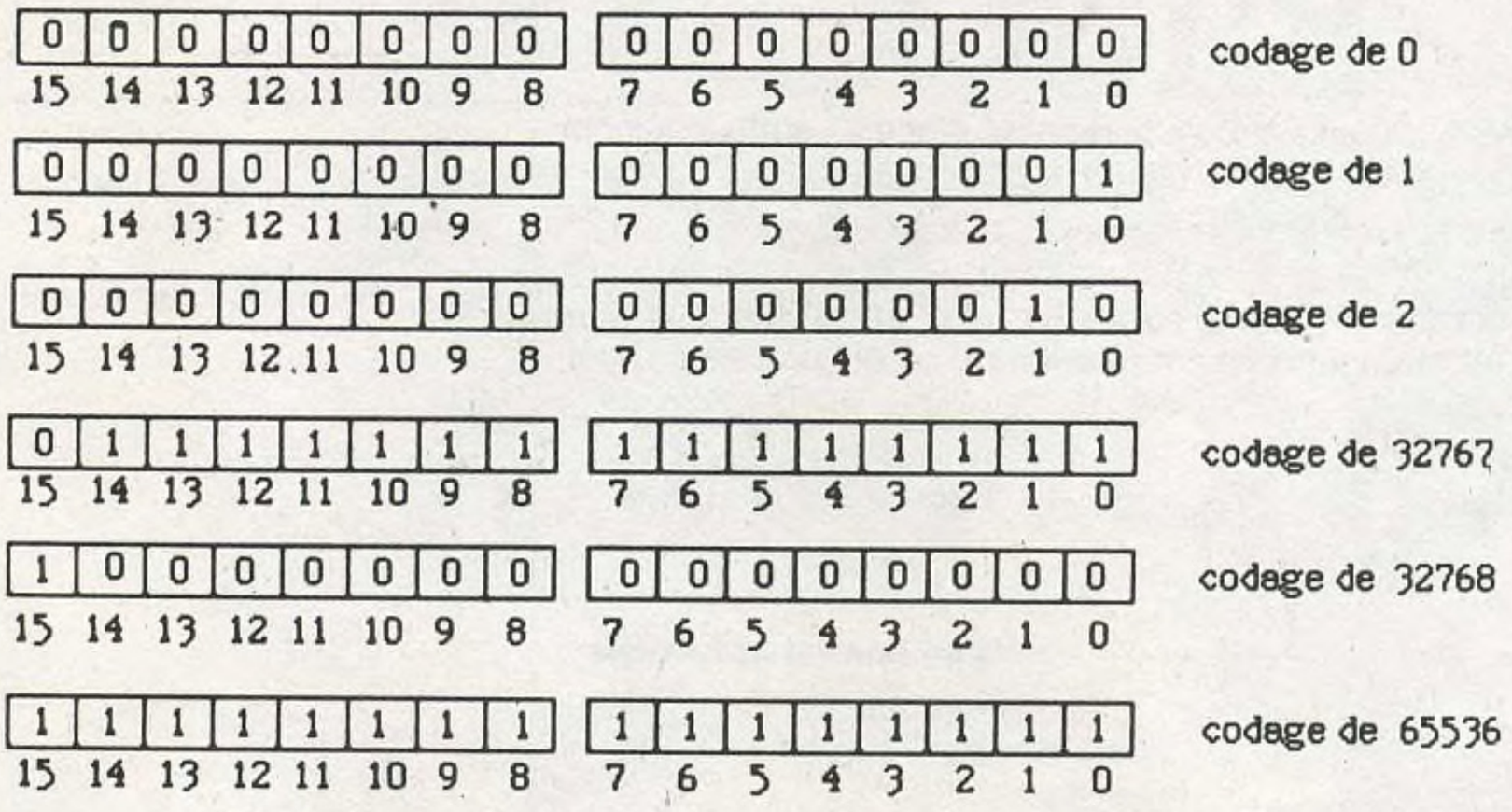


Figure 1.1.2.2

1.1.3. Le codage des entiers signés

1.1.3.1. Le codage des entiers positifs

Pour différencier les entiers positifs des entiers négatifs, le bit n° 15 (appelé bit de poids fort ou bit le plus significatif), qui est donc le bit le plus à gauche, sert à déterminer le signe de l'entier (figure 1.1.3.1).

Avec cette méthode, il ne reste plus que 15 bits pour coder les nombres, soit la possibilité de coder 2^{15} entiers, dans l'intervalle $[0, 2^{15}-1]$ ($2^{15}-1=32767$).

Ce codage est celui exposé pour les entiers non signés. Ainsi 0, 1, 2, 32767 auront le codage décrit au paragraphe précédent. Pour les nombres positifs inférieurs à 32767, il n'y a pas le moindre changement par rapport à ce qui a été dit.

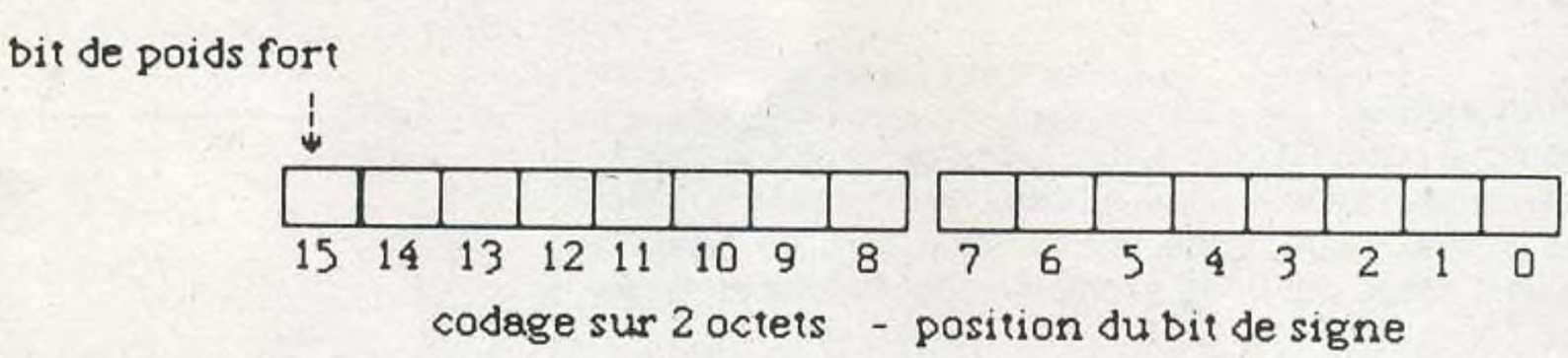


Figure 1.1.3.1

1.1.3.2. Le codage des entiers négatifs

Pour des raisons d'opérations machines qui ne seront pas décrites ici, le codage des entiers est sensiblement différent. Considérons le codage d'un tel nombre (– 49 par exemple) :

– on code + 49
– on inverse la valeur de chaque bit autre que le bit de signe
– on rajoute 1

Les opérations décrites correspondent aux figures 1.1.3.2.1, 1.1.3.2.2 et 1.1.3.2.3.

Ceci nous permet de donner une «formule» de codage de ces entiers :

code (ent. nég) = code (65535 +ent. nég +1)
(codage signé) (codage non signé)

et on voit que cette formule revient à coder des entiers non signés, dans l'intervalle $[0, 2^{16} - 1]$.

Un exemple : codage de - 49

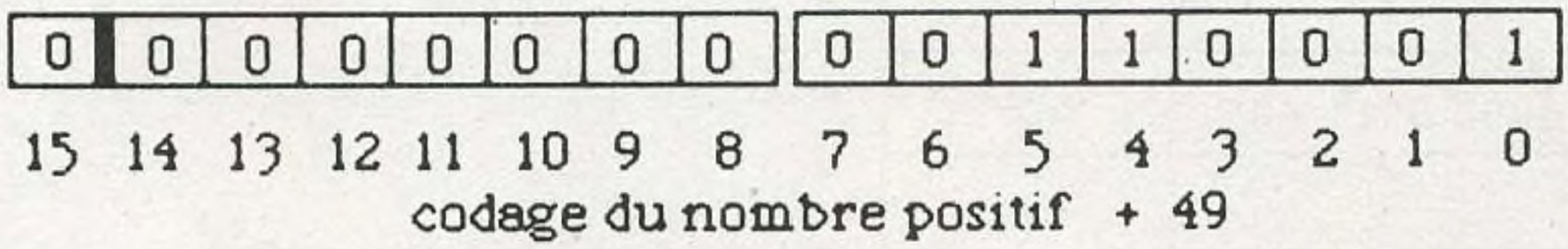


Figure 1.1.3.2.1

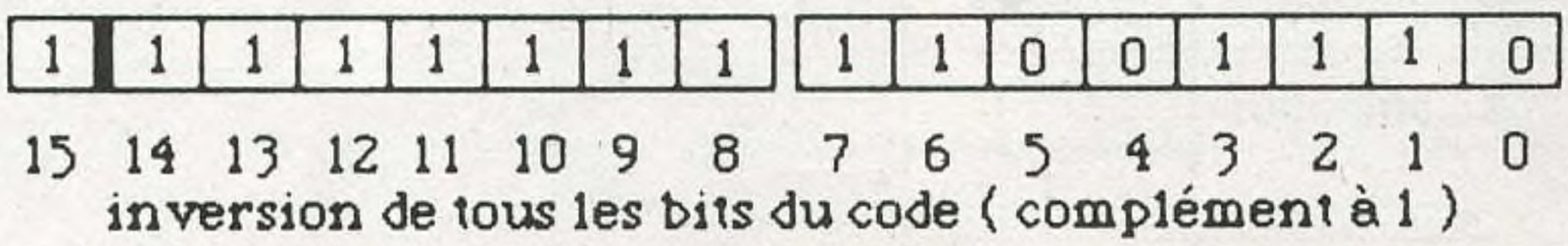


Figure 1.1.3.2.2

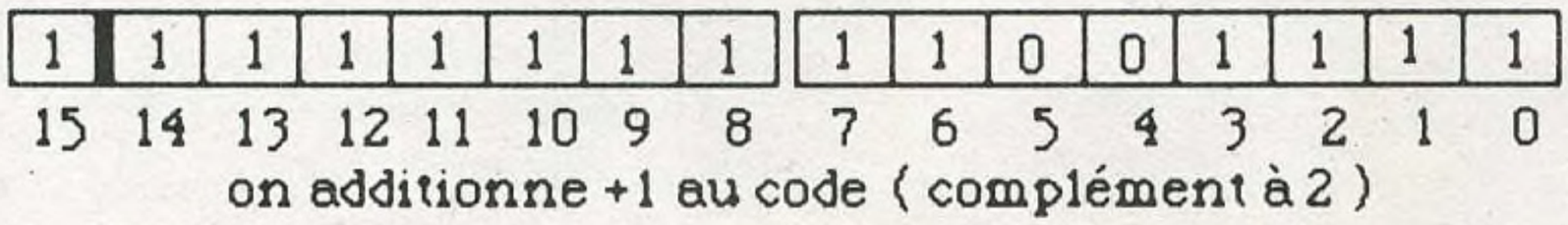
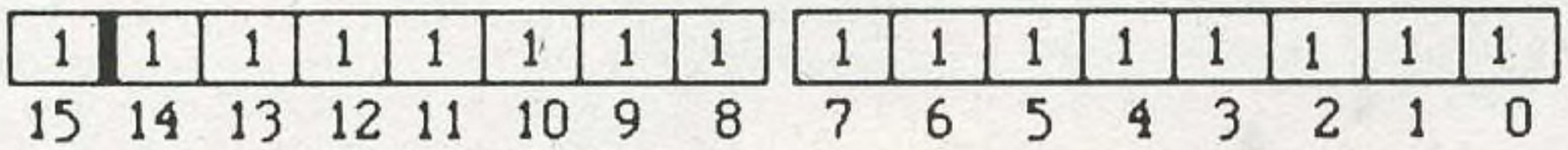


Figure 1.1.3.2.3

L'étape d'inversion des bits (figure 1.1.3.2.2) s'appelle faire le complément à 1, puis, lorsque la valeur 1 été ajoutée, on dit que l'on a effectué un complément à 2 (complément à 1, plus 1. figure 1.1.3.2.3).
Il reste un point intéressant à soulever. Par cette méthode, nous avons obtenu une incertitude concernant le nombre entier correspondant au code binaire 1000 0000 0000 0000. Par la correspondance de la formule indiquée précédemment, on obtient le nombre – 32768 ce qui permet un gain de l'intervalle et reste très logique (voir figure 1.1.3.2.4.1).
Nous obtenons alors le tableau de codage de la figure 1.1.3.2.4.2. Faites bien attention à l'ordre des codes, en remarquant que l'ordre des négatifs est inversé (pour vous en souvenir, faites le rapprochement avec le négatif d'une photographie).

codage de - 32767



codage de - 32768

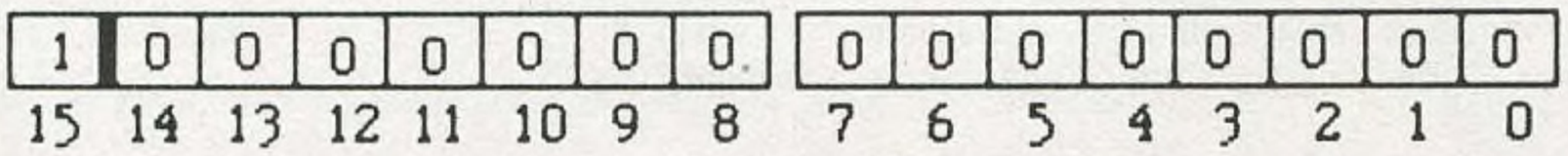


Figure 1.1.3.2.4.1

nombre à coder	code binaire machine	traduction décimale
0	0000 0000 0000 0000	0
1	0000 0000 0000 0001	1
⋮	⋮	⋮
49	0000 0000 0011 0001	49
⋮	⋮	⋮
32767	0111 1111 1111 1111	32767
-32768	1000 0000 0000 0000	32768
-32767	1000 0000 0000 0001	
⋮	⋮	⋮
-1	1111 1111 1111 1111	65535

Figure 1.1.3.2.4.2

1.2. Codage des réels

1.2.1. Introduction

La déclaration de variables réelles suit un schéma plus souple que celui de déclaration des variables entières. En effet, toute variable ne se terminant pas par un code particulier déclarant une autre structure est considérée comme une variable réelle. Ces codes particuliers sont :

- % pour les variables entières
- $ pour les variables alpha-numériques (ou chaînes de caractères)
- # pour les réels en double précision.

Tout nom de variable ne se terminant pas par un de ces suffixes indique par défaut une variable réelle.

1.2.2. Format de codage

Les variables réelles sont codées sur 4 octets (figure 1.2.2.1) avec la structure décrite à la figure 1.2.2.2, où «a» représente la mantisse et «b» l'exposant, car les mathématiques montrent que tout nombre x peut s'écrire selon le schéma suivant :

$$x = a \cdot k^b$$

où k prend généralement les valeurs 10 ou 2.

Pour un ordinateur, k prend la valeur 2, pour la représentation interne, et 10 pour la représentation à l'écran, et a (la mantisse) est l'élément de l'intervalle numérique [1, 2]. b (l'exposant) est un entier tel que 2^b soit la plus grande puissance de 2 inférieure à x (la figure 1.2.2.3 donne des exemples).

Comme a est toujours dans l'intervalle [1, 2], il est possible de ne coder que la partie décimale du nombre, ce qui limite la place utilisée par le code.

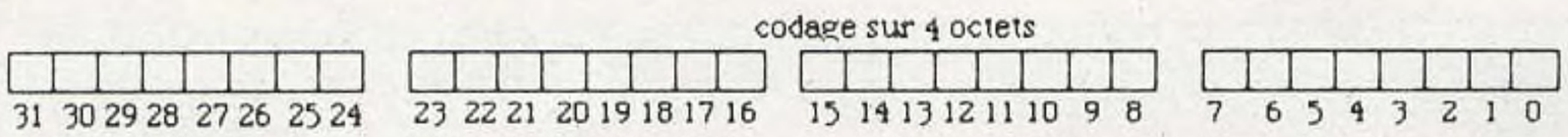


Figure 1.2.2.1

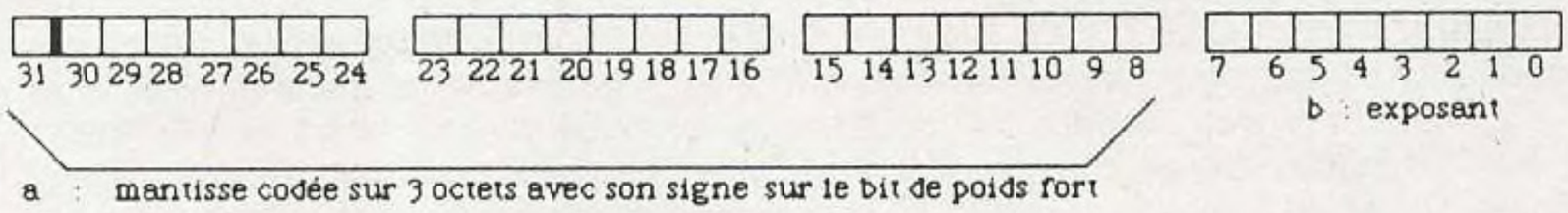
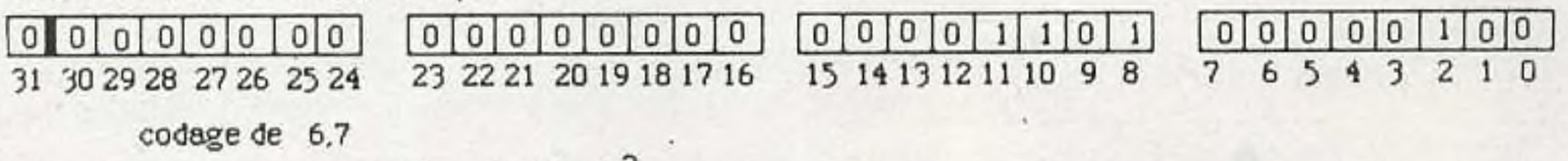


Figure 1.2.2.2

deux exemples de codage de nombres réels

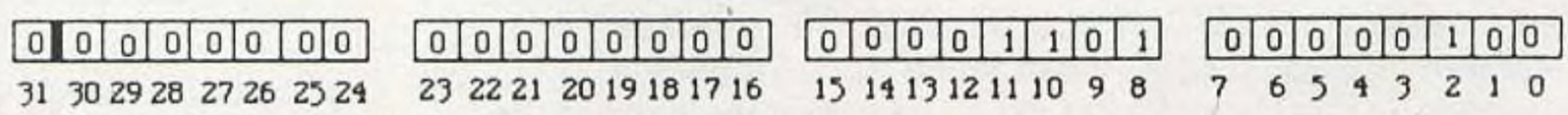


codage de 6,7

$6.7 = 1,675 * 2^2$

on code donc : a = 0.675 = 1/2 + 1/8 + 1/16

et : b = 2



codage de 6.700001

$6.700001 = 1.6750003 * 2^2$

on code : a = 0.6750003 = 1/2 + 1/8 + 1/16 + 1/ 4096

et : b = 2

donc les codes sont égaux

1.2.3. Les limites du format

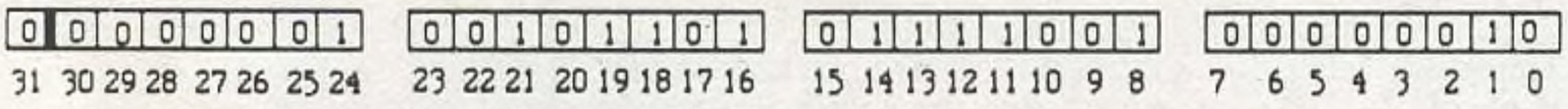
Le format ainsi défini présente hélas des limites très ennuyeuses au niveau de la précision en machine des nombres réels traités. Vous avez tous entendu parler de ces programmes qui peuvent donner la 1 000 000e décimale de PI (qui ne vaut pas du tout 3,14 ni 3,1415926... puisqu'il ne peut être défini de manière décimale). Ces programmes jonglent avec le codage des nombres pour parvenir à ce résultat. Dans un prochain article (dans deux mois exactement), nous vous proposerons un tel programme, car c'est certainement la meilleure approche possible pour bien maîtriser le problème du codage des réels. La figure 1.2.3.1 montre le codage de PI en machine.

Nous vous avons indiqué le format de codage des réels. La mantisse, a, est donc un nombre signé, codé sur 3 octets, signe compris, codé sur le bit 31 (figure 1.2.3.2). Ce codage sur 23 bits de la mantisse permet une précision de sept chiffres significatifs. Ceci se vérifie de la façon suivante :

sur 23 bits, on peut coder 2^{23} nombres. Or 2^{10} est peu différent de 10^3,

d'où le résultat.

Nous allons maintenant nous occuper de la partie exposant, b.
Il est codé sur l'octet restant, en codage signé, ou non signé, selon le processeur. On voit que ce codage est peu optimisé, tout au moins pour l'occupation mémoire de b, puisque seul un bit de son codage est utilisé.



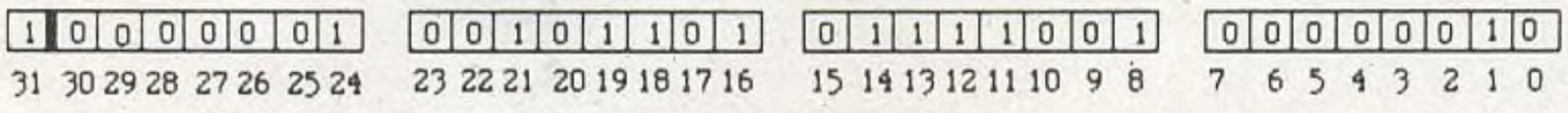
codage de PI = 3.141592628

PI = 2 * 1.5707961

on code donc : a = 0.5707961 = 1/2 + 1/16 + 1/32 +1/64
+ 1/128 + 1/256 + 1/512 + ...

et : b = 2^1

Figure 1.2.3.1



codage de -PI = 3.141592628

PI = 2 * 1.5707961

on code donc a = 0.5707961 = 1/2 + 1/16 + 1/32 +1/64
+ 1/128 + 1/256 + 1/512 + ...

et : b = 2^1

Figure 1.2.3.2

1.2.4. Autres codages

Malgré ce type de problèmes, le codage décrit est presque universel. Nous avons bien écrit presque, car en effet quelques exceptions doivent être décrites. Par exemple l'APPLE II dont le codage est légèrement meilleur, tout au moint en ce qui concerne la précision. En effet, cet ordinateur code les réels sur cinq (5) octets, quatre étant utilisés par la mantisse (voir figure 1.2.4.1.). Donc, ce qui a été dit pour l'exposant ne change pas, mais la précision sur la mantisse gagne deux ou trois chiffres, ce qui peut être très précieux. La plupart des ordinateurs à base de 6502 utilisent ce même type de codage, mais comme toujours, vérifiez si vous ne possédez pas une des très nombreuses exceptions.

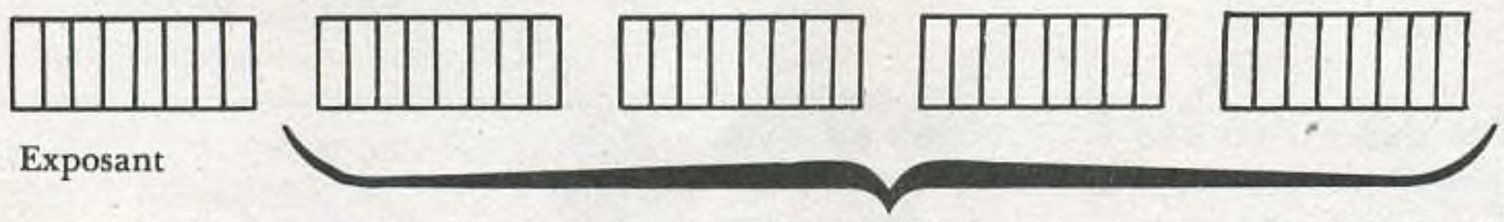


Figure 1.2.4.1

2. OPTIMISATION DES PROGRAMMES

2.1. Optimisation de la place mémoire

Les chapitres précédents ont montré qu'il est plus économique d'utiliser des variables entières. En manipulant des variables entières dans vos programmes, vous gagnerez la moitié de la place occupée par des variables réelles. Rajouter le symbole « % » n'est pas contraignant. D'un autre côté, il peut vous sembler plus facile d'écrire des variables réelles. Dans ce cas, il sera toujours temps d'ajouter ce symbole, si des problèmes apparaissent. Attention, ce qui a été dit est vrai avec la plupart des ordinateurs disponibles. Hélas, sur l'APPLE II, ce n'est pas du tout ce qui se passe. En effet, cet ordinateur code bien les entiers sur deux octets, mais leur laisse néanmoins cinq octets systématiquement, donc trois sont inutilisés. L'occupation mémoire est équivalente à celle des réels, qui est, elle aussi, particulière (voir paragraphe 4.2.). Sur cette machine, l'intérêt des entiers n'est donc pas là.

2.2. Optimisation du temps d'exécution

Travailler sur l'optimisation d'un programme est un travail très ingrat, tout au moins dans les premiers temps, lorsque tout se fait de manière empirique. En ce qui concerne l'amélioration du temps d'exécution, chaque processeur ayant ses méthodes propres, il nous est difficile de vous donner des recettes absolues. Ainsi, sous Z 80 et 8088, il est préférable, en ce qui concerne les variables, d'utiliser des entiers, car le travail sur des variables réelles est beaucoup plus long. Mais sur le 6502 de l'APPLE II, à chaque traitement, lorsque l'unité centrale rencontre une variable entière, avant toute autre chose, elle la transforme au format réel, exécute l'instruction, puis retransforme au format entier, ce qui prend un temps considérable. Donc, sur des ordinateurs travaillant de cette manière, il faudra composer entre le gain de place et le gain de temps, selon chaque programme. Il faut néanmoins noter que la plupart du temps, c'est la place qui pose problème. Enfin, si vous ne connaissez pas la façon dont travaille votre ordinateur, essayez les programmes 2.2.1. et 2.2.2. Ce sont des boucles très simples, pour lesquelles nous vous fournissons le temps de travail sur l'APPLE II, en premier, et sur ce même ordinateur, en utilisant une carte Z 80, en second. Vous pouvez ainsi constater que pour un même ordinateur, selon le processeur utilisé, la façon d'aborder le problème sera différent. Alors, d'un ordinateur à l'autre...

Vous avez pu remarquer que les ordinateurs utilisant le même processeur que l'APPLE II ne semblent privilégier les entiers ni pour la place mémoire, ni pour le temps d'exécution. Dans ces conditions, il est permis de se poser la question de connaître leur intérêt. En fait, un privilège immédiat des entiers est de travailler sur des nombres ayant une précision parfaite, car leur codage, nous l'avons vu, n'est pas une approximation.

Programme 2.2.1

étude du temps d'éxécution d'une boucle
cas de variables réelles

```
10 FOR I = 1 TO 10000
20 A = A + 1
30 NEXT I
```

temps d'éxécution sur APPLE II : 40,7 sec

temps d'éxécution sur APPLE II sous Z 80 : 55 sec.

Programme 2.2.2

étude du temps d'éxécution d'une boucle
cas de variables entières

```
10 FOR I = 1 TO 10000
20 A% = A% + 1
30 NEXT I
```

temps d'éxécution sur APPLE II : 45,9 sec.

temps d'éxécution sur APPLE II sous Z 80 : 49 sec.

3. EXEMPLES D'APPLICATION

3.1. Programme de codage d'un entier positif

3.1.1. Introduction

Ce programme a pour but de réaliser la visualisation à l'écran du codage d'un entier positif, selon le format utilisé en BASIC, et décrit au paragraphe 1.1.3.1. C'est le programme 3.1.1.

3.1.2. Description du programme

On charge un nombre entier dans la variable A %. Comme le nombre est positif, son bit de signe est à 0 (ligne 40). Il reste à tester, bit à bit, la valeur du code, ce qui est fait ici en comparant A % avec toutes les puissances successives de 2, par ordre décroissant, pour des raisons d'ordre d'affichage à l'écran.

3.1.3. Analyse des méthodes employées

Pour la lisibilité du programme, les valeurs de chaque bit sont écrites sans passer par des variables. Nous étudierons dans le prochain article, la différence entre variables déclarées et constantes. Pour le reste, nous travaillons avec des variables entières. La plus grande valeur à tester est 16384, puisqu'un entier est au plus égal à 2. 16384 – 1. Il faut noter que la variable C % est explicitement chargée avec la valeur 16384, car le calcul de 2^{14} est long et coûteux, et que de plus, pour des problèmes de précision, il est toujours meilleur de procéder ainsi.

Programme 3.1.1

représentation du code d'un entier positif

```
10  INPUT A%
40  PRINT 0
45  C% = 16384
50  FOR I = 1 TO 15
60  IF A% >= C% THEN PRINT 1; A% = A% - C%
                ELSE PRINT 0;
70  C% = C% DIV 2
80  NEXT I
90  PRINT
```

Enfin vous pouvez remarquer que la division par 2 effectuée à la ligne 70 est un peu particulière. Sur des variables entières, la division par deux se traduit par un simple décalage des bits, puisque les puissances de deux sont elles-mêmes décalées (figure 3.1.3.1). Or la division normale ne prend pas en compte cet état de fait, alors que la division entière (DIV) le fait, n'exécutant que le décalage, d'où gain de temps. Essayez ceci sur votre machine, pour voir si cela marche aussi dessus, si votre ordinateur vous propose cette instruction, car elle n'est pas véritablement universelle.
Enfin, vous pouvez voir que nous avons utilisé une boucle, et non un test suivi d'un GOTO, car cette deuxième méthode est beaucoup plus longue. En effet, elle nécessite un test qui n'est pas un test micro-programmé par la machine, mais que vous lui imposez, suivi d'une recherche de ligne, alors que dans le cas d'une boucle FOR... NEXT, l'ordinateur aura stocké le numéro de ligne de l'instruction FOR... dans un registre spécial de sa mémoire, appelé pile d'exécution, d'où un accès très rapide.
Il faut noter par la même occasion que certains BASIC n'imposent pas la forme NEXT I, mais que l'on peut écrire simplement NEXT, lorsque cela ne crée pas d'ambiguités. Dans cette hypothèse, n'hésitez pas à utiliser cette simplification, car vous y gagnerez beaucoup de temps. En effet, le programme n'effectue pas dans ce cas la vérification de l'homogénéité de la variable index entre le début et la fin de la boucle. Le temps gagné par cette méthode est considérable, mais vous devrez faire attention vous-même aux problèmes d'homogénéité. Le travail supplémentaire est vraiment

récompensé par le gain de temps, qui peut être de plusieurs secondes sur une boucle un peu longue (de 1 à 10 000 par exemple est une boucle bien assez longue pour tester la différence).

Utilisation du DIV pour les divisions par 2
DIV effectue la division entière, c'est-à-dire que DIV peut être obtenu par l'algorithme suivant :

$$x \text{ DIV } y = E(x/y)$$

où E représente l'opérateur de partie entière.
Mais, pour la division par 2, DIV travaille différemment. En effet, diviser par 2 un nombre entier revient à décaler son codage binaire, même si ce nombre est impair. Vérifiez-le ! Cette opération représente un gain de temps considérable, car le décalage est une opération de base des microprocesseurs.

3.2. Programme de codage d'un entier négatif

3.2.1. Introduction

Comme le précédent, ce programme (programme 3.2.1) a pour but de visualiser le codage d'une variable entière, mais cette fois-ci, dans le cas où elle est négative.

3.2.2. Description du programme

Ce programme travaille à peu de choses près comme le précédent, en utilisant la méthode de codage suivante :
Le nombre est transformé en sa valeur absolue à laquelle on soustrait 1, autrement dit, on effectue le complément à deux en inversant les étapes décrites au paragraphe 1.1.3.2. Il reste donc à réaliser l'inversion des bits, ce qui est fait en même temps que le test de dépassement des puissances successives de 2. En effet, contrairement à ce qui avait été fait pour le premier programme, il sera affiché la valeur 0 dans le bit numéro x si le nombre testé à cette étape est plus grand que 2^x. Ceci effectue donc automatiquement le complément à 1.

Programme 3.2.1

représentation du code d'un entier négatif

```
10  INPUT A%
20  A% = A% - 1
40  PRINT 1
45  C% = 16384
50  FOR I = 1 TO 15
60  IF A% >= C% THEN PRINT 0; A% = A% - C%
                ELSE PRINT 1 ;
70  C% = C% DIV 2
80  NEXT I
90  PRINT
```

La ligne 20 effectue la deuxième partie du complément à deux, ce qui limite les opérations postérieures à l'inversion des bits par rapport au programme précédent (3.3.1), ce qui est fait dans le traitement du test de la ligne 60.

3.3. Programme de codage d'un entier quelconque

3.3.1. Présentation du programme

Ce programme (programme 3.3.1) est la synthèse des deux précédents, mais nous avons préféré scinder l'explication en deux parties, afin de la rendre plus claire.

3.3.2. Description du programme

La différence majeure avec les programmes précédents est l'apparition d'une variable B %, qui prend la valeur 1 si A % est positive, 0 dans le cas contraire. On reconnaît là la valeur des bits selon le signe de A %, qui rend ou ne rend pas nécessaire le complément à 1 (voir pour plus de précisions le paragraphe 3.2.2).

Programme 3.3.1

représentation du code d'un entier quelconque

```
10  INPUT A%
20  B% = 1
30  IF A% < 0 THEN B% = 0 : A% = A% - 1
40  PRINT 1 - B%
45  C% = 16384
50  FOR I = 1 TO 15
60  IF A% >= C% THEN PRINT B% ; A% = A% - C%
                ELSE PRINT 1 - B% ;
70  C% = C% DIV 2
80  NEXT I
90  PRINT
```

Ce programme est donc un condensé des deux précédents. Nous vous le proposons car il nous semble représenter l'exemple type de la réunion de deux cas nécessitant des traitements peu différents. Donc, suivant le signe, on fait ou non le complément à 2. Ce complément est fait comme pour le programme 3.1.2.
Mais pour concaténer les deux programmes, il apparaît une nouvelle variable, B %, dont la valeur varie avec le signe de A %.
Pour ceux qui ont de l'avance, B % joue le rôle d'une variable booléenne ; structure dont nous reparlerons dans le prochain cours.

3.4. Utilisation de VARPTR pour connaître le codage des réels

3.4.1. Description de la commande VARPTR

VARPTR est une commande spécifique du BASIC MICROSOFT, que ce soit le MS-BASIC, le M-BASIC ou le G-BASIC. Cette commande a pour but de donner l'adresse mémoire d'une variable. Ainsi, la simple instruction 3.4.1 donne l'adresse du premier octet de la variable A (si la valeur obtenue est négative, il faut ajouter 65535).

3.4.2. Description du programme

Ce programme très simple permet de récupérer le codage machine d'une variable réelle (pour une variable entière il suffit de lire 2 octets). Au début, une variable est chargée avec l'adresse renvoyée par VARPTR puis les quatre octets sont successivement lus et affichés immédiatement, ceci pour ne pas créer de nouvelles et inutiles variables. Les octets sont donc lus en commençant par l'octet de poids fort et en terminant par l'octet de poids faible. Un exemple, sur APPLE II, en

utilisant le M-BASIC avec la carte Z 80, est donné avec le programme 3.4.1 et les sorties obtenues se trouvent en figure 3.4.1. Le codage obtenu est propre au MBASIC MICROSOFT.

Programme 3.4.1

```
10 INPUT A
20 LPRINT A
30 FOR I = 0 TO 3
40 LPRINT PEEK ( VARPTR ( A ) + I)
50 NEXT
60 GOTO 10
```

Figure 3.4.1

Sorties obtenues avec MBASIC

A=

1	-1	2	0	-2	100	100.1
0	0	0	0	0	0	51
0	0	0	0	0	0	51
0	128	0	0	128	72	72
129	129	130	0	130	135	135

A=

1.1	-1.1	-1.75	.25	-.25	.5	-100.1
205	205	0	0	0	0	51
204	204	0	0	0	0	51
12	140	224	0	128	0	200
129	129	129	127	127	128	135

4. PROBLEMES LIES AUX CODAGES DECRITS

4.1. Exemple pratique simple

Nous l'avons dit, la précision peut poser des problèmes en raison du codage des réels tel qu'il est effectué. Le programme suivant le montre bien :

```
10 FOR I = 1 TO 100
20 A = A + .1
30 PRINT A ;"   ";
40 IF ( I MOD 10 ) = 0 THEN PRINT
50 NEXT
```

Les sorties à l'écran sont (sur un APPLE II, sous BASIC MICROSOFT Z 80) :

```
.1   .2   .3   .4   .5   .6   .7   .8   .9  1.0
1.1 1.2 1.3 1.4 1;5 1.6  1.7 1.8 1.9 2.0
2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0
3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0
etc
7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.79999 7.89999 7.99999
8.09999 8.2 etc
```

Vous voyez que pour ce programme simpliste, pour lequel aucune calculatrice à 100 francs ne se tromperait, l'ordinateur est totalement incompétent. Imaginez ce qui se passe dans des calculs complexes !

4.2. Codages différents

Pour vos programmes, la précision de votre ordinateur est, le plus souvent, suffisante, mais pour des grosses applications professionnelles, ce n'est plus du tout vrai. Une banque, ou la sécurité sociale, ont besoin de bonnes précisions, de même qu'un technicien travaillant sur des mesures de résistances, etc. Pour ce type d'applications d'autres codages existent, beaucoup plus complexes ou tout simplement sur un nombre d'octets plus important.

4.2.1. La double précision

Sur votre micro-ordinateur, vous avez peut-être la «double précision». Ces réels sont codés sur le double d'octets (soit 8), dont la répartition est la suivante : 7 octets pour la mantisse, ce qui permet d'obtenir jusqu'à 16 chiffres de précision, tout en restant dans le même intervalle de nombre accessibles. C'est mieux, mais il arrive fréquemment d'avoir besoin, ou envie, d'une précision plus grande, dans des programmes scientifiques, ou ludiques. De toute façon, nous aborderons très bientôt ce problème. Pour être complets, nous devons signaler enfin que d'autres BASIC permettent plus de place pour l'exposant, autorisant des nombres jusqu'à $10^{\pm 96}$.

5. CHAINES DE CARACTERES

5.1. Introduction

Les chaînes de caractères représentent un outil très puissant, et très pratique, pour gérer des entrées-sorties entre le programme, et l'utilisateur, entre le programmeur et son client. En effet, à une époque où l'informatique devient «conviviale», «interactive» voire même «intelligente», les programmeurs doivent se plier à cette évolution, et, peu à peu, s'orienter vers une utilisation la plus proche de ce qu'on appelle «le langage naturel», autrement dit le sport que vous pratiquez tous les jours, en toute connaissance de cause, contrairement à Monsieur Jourdain.
Ce problème crucial pour rendre un programme agréable à reprendre, à améliorer, est d'une actualité brûlante, de nos jours où il n'est question que d'interface homme-machine, ou du traitement du langage naturel. Nous ne considerons que les parties intéressantes pour un utilisateur d'ordinateur personnel, avec pour ce premier cours, l'implantation machine.

Le chaînes de caractères sont donc un élément important du confort d'utilisation d'un ordinateur, ou d'un langage. Il est en effet beaucoup plus agréable, lors de l'exécution d'un programme, de voir sous ses yeux un écran tel que celui de la figure 5.1.1. plutôt que celui de la figure 5.1.2.

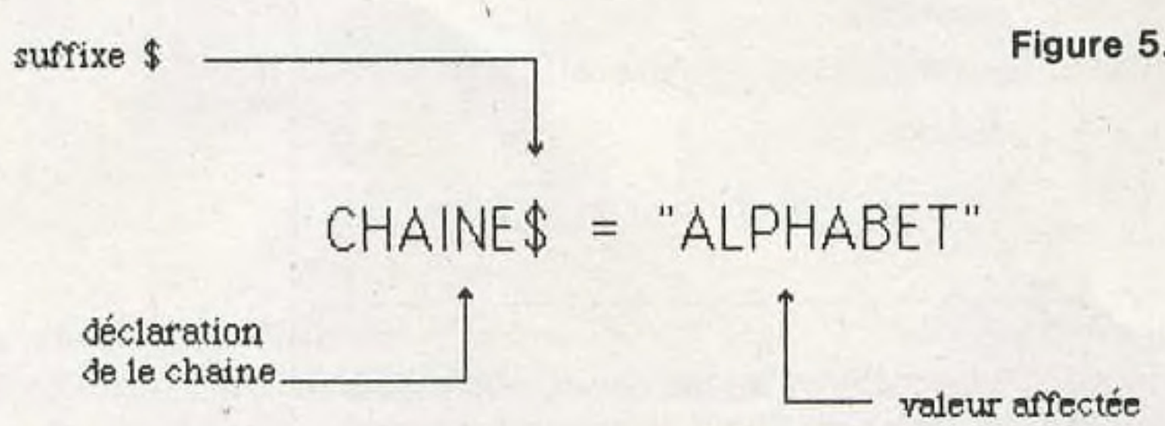
Figure 5.1.1.

A ce sujet, le BASIC est un des langages les plus agréable qui soit, en raison de la facilité à utiliser, et à gérer, les mots. Il dispose pour cela d'un ensemble d'instructions complet, et d'une structure bien définie.

Figure 5.1.2.

Une variable chaîne de caractères (on dit aussi alphanumérique, car elle peut contenir aussi bien des caractères alpha-bétiques que numériques) est déclarée par le suffixe $ comme illustré par la figure 5.1.3. Toute variable dont le dernier caractère du nom est ce $ est une variable alphanumérique, et ce à l'exclusion de tout autre suffixe.

Figure 5.1.3.



5.2. Les codages

5.2.1. Introduction

On peut distinguer deux principaux types de codages, codage avec dimensionnement ou codage dynamique.

5.2.2. Le codage avec dimensionnement

Avec ce codage, très fréquent, il est nécessaire d'annoncer la longueur de la chaîne de caractères. Naturellement, il n'est pas toujours facile, ni possible, de connaître la longueur de cette chaîne, par définition variable. Donc, le plus souvent, il vous faudra déclarer une chaîne de longueur supérieure, parfois nettement plus, afin de ne pas occasionner d'erreurs d'exécution. Ceci est un évident problème pour les petits ordinateurs, pour lesquels la place est comptée, car les chaînes de caractères sont très gourmandes en place mémoire.

La déclaration de la chaîne se fait de la manière indiquée ci-dessous, pour une chaine de longueur 36 (c'est-à-dire comprenant au plus 36 caractères). Il s'agit donc d'un «dimensionnement». 1000 DIM A$ (36)

Lorsque vous dimensionnez une chaîne de caractères, vous demandez à l'ordinateur de vous réserver un nombre de cases mémoires équivalent au nombre fourni entre parenthèses. Chaque terme de la chaîne est codé selon son code ASCII, en général. De plus, un octet est utilisé pour rappeler la longueur de la variable (pour avoir la table des codes ASCII, reprendre le n° 1 de LED-MICRO).
Autrement dit, une chaîne déclarée comme précédemment réserve 36 octets de la mémoire centrale, plus l'en-tête, en général moins de dix octets, même si vous n'utilisez pas cette variable par la suite. En conséquence, faites très attention, et ne soyez pas trop prodigue de ce type de facilités.
Un codage de ce type se rencontre entre autres, en INTEGER BASIC, sur les premières versions de l'APPLE II, mais reste assez fréquent, sur les ordinateurs familiaux.

5.2.3. Le codage non dimensionné

Avec ce codage, l'utilisateur n'est plus obligé de définir la longueur de la variable, de la déclarer. Lorsque, pendant une exécution, le programme rencontre une variable, il crée une zone réservée pour cette variable en fonction de la chaîne qui lui a été affectée (figure 5.2.3.1.). Ceci permet une occupation mémoire optimisée, tout au moins en apparence (voir à ce sujet le paragraphe 5.4.). En effet, il n'y a pas de place arbitrairement réservée, et qui peut ne jamais servir. Pour cette raison, ce codage est dit dynamique, car le programme réserve la place en mémoire lorsque le besoin se fait sentir, méthode appelée «dynamique» en informatique, et que l'on retrouvera en étudiant le langage PASCAL, dans quelques mois.

Figure 5.2.3.1.

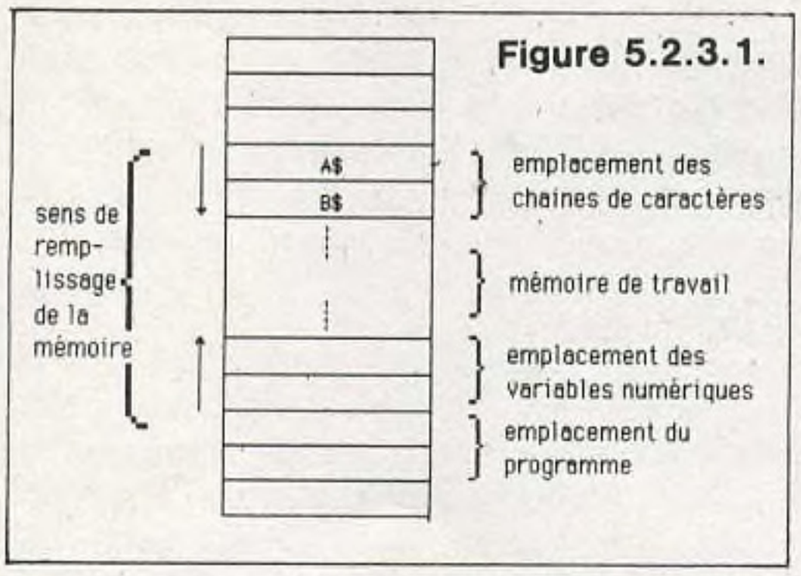


Figure 5.2.3.2.

exemple de chaine :

A$ = "COUCOU"

codes ASCII : c : 67
o : 79
u : 85

décalage : 12

codage effectif : 79 91 97 79 91 97

Les chaînes de ce type admettent, en général, 255 caractères au maximum, y compris les blancs, ceci parce que la longueur est codée sur un octet (voir le début du cours, pour le codage des entiers, et les capacités de codage d'un ou de plusieurs octets). Les caractères de la chaîne sont codés séquentiellement (à la suite les uns des autres), presque toujours par leur code ASCII. Il arrive que ce code ASCII soit appliqué avec un déplacement, ou décalage, qui soit propre à chaque ordinateur (la figure 5.2.3.2. illustre le cas d'un BASIC utilisant un décalage).
Dans ce dernier cas, il vous faudra trouver ce décalage pour retrouver le code, ce qui est compliqué car il s'agit toujours d'une addition, ou d'une soustraction.

CONCLUSION

Notre but dans ce premier cours était de montrer que si, dans la théorie, les choses ne sont pas trop complexes, il est nécessaire de prêter la plus grande attention à la moindre application, et ce d'autant plus que votre ordinateur aura une faible capacité mémoire. Donc, il semble indispensable de s'exercer à connaître les limites de son appareil mais, cette condition une fois remplie, vous connaîtrez alors la joie sans partage de maîtriser parfaitement votre application informatique.

C'EST ARRIVE DEMAIN

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

Le marché de la micro-informatique est ici un vrai bouillon de culture. Tout s'agite, se fait et se défait, se dit puis se contredit, en quelques minutes. C'est encore une ambiance de ruée vers l'or, mais entre gens de «bonne compagnie», tous très comme il faut. La rumeur suit l'inspiration et se fut tout particulièrement vrai lors de la dernière exposition, le COMDEX/SPRING qui a eu lieu à Atlanta, la semaine dernière. Le plus beau roman de l'exposition a même été l'annonce de la faillite d'APPLE. Ceci mis à part, il est possible de tirer quelques enseignements prudents de la situation actuelle. La tendance est à la concentration autour des marques phares du marché, au grand dam des petites sociétés créatrices de très beaux produits, dont les clients potentiels se détournent, par prudence excessive, et préfèrent s'orienter vers des produits plus banalisés, mais portant la griffe IBM (ou d'un autre géant de ce type), même si ce n'en est qu'une pâle copie. Comme ces petites sociétés ne parviennent pas toujours à se renouveler, leur situation est souvent critique (c'est le cas de la société VISICORP, qui a totalement disparu, après avoir inventé le tableur avec VISICALC). Maintenant,

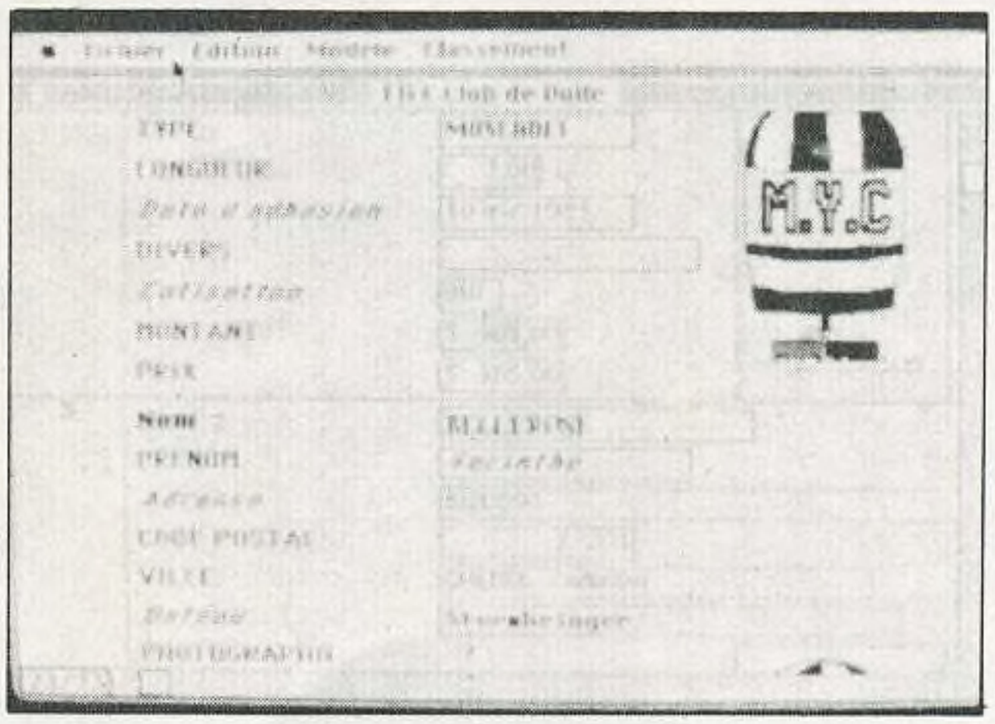
Encore des logiciels de très grande qualité pour le Mac.

le client joue la carte du «standard sans risques». On obtient alors les résultats suivants, au plan de la vente d'équipements nouveaux : IBM 76 % et APPLE 2,8 %. Le déséquilibre est risible, mais il faut noter que la marque APPLE passe à 31 %, sur le marché européen.

Un phénomène très intéressant se développe depuis quelques mois sur le plan logiciels grand public. De plus en plus, des sociétés de création de logiciels proposent des produits d'excellente qualité, et à des prix très bas. Le pionnier en la manière est la société BORLAND, devenue fameuse en proposant un langage Pascal, TURBO PASCAL, dont toute la presse s'attache à dire depuis sa sortie qu'il est le meilleur sur le marché, rien moins !! Ce langage s'adresse aux IBM PC, compatibles, et aux ordinateurs travaillant sous CP/M, ou MS-DOS. Ses performances sont époustouflantes, sur le plan rapidité et facilité de mise au point. Un dernier détail, il est vendu 400 F. Intéressant, non ? C'est beaucoup moins qu'un jeu. D'autres sociétés ont suivi cette voie, produisant des logiciels performants (très performants) à des prix défiant l'imagination, et décourageant complètement le piratage. C'est pourquoi la loi anti-piratage votée en France fait sourire ici, car la politique commerciale française est surrannée. Elle consiste à produire des logiciels très pointus, donc potentiellement peu vendables, ou des jeux, et à vouloir les amortir très vite, donc à vendre très cher. Ce prix encourage la copie à grande échelle.

Parmi les grands espoirs du futur proche, figure le compact-disque. En effet, des sociétés aussi crédibles que Philips ou Denon et Sony, viennent de dévoiler, pour la première fois hors du Japon, de tels lecteurs dont la capacité est actuellement de 600 méga-octets, sur un support dont la taille est celle d'une disquette. Le problème reste toutefois entier en ce qui concerne le problème de l'effacement et de la réécriture, car pour l'instant, il s'agit uniquement de mémoire morte (ROM ou MEM selon le pays où l'on s'exprime). Des sociétés américaines proposent des supports proches du compact-disque, mais permettant d'écrire une fois dessus, ce qui peut être intéressant pour une société de diffusion de logiciels, car il est ainsi possible, pour les acheteurs de ces logiciels, de stocker les données fixes qui leur sont propres, et cela empêche la copie, l'effacement accidentel, etc. Tout le monde y gagne.

Pour en revenir à la micro-informatique, le sujet du moment de ce côté de l'Atlantique est de savoir si Jack Tramiel va gagner son pari. Jack Tramiel est le patron d'Atari, autrefois réputée pour ses ordinateurs et programmes de jeux, et qui depuis quelques années coulait gentiment, dans l'indifférence quasi-générale. A son arrivée à la tête de la société, Tramiel décida de changer tout cela. Le premier résultat de cette reprise en main fut l'annonce, voilà quelques semaines, d'une nouvelle machine, l'ordinateur 520 ST, dont les spécifications sont les suivantes : une sorte de MACINTOSH, avec la couleur en plus, un lecteur de disquettes 3,5 pouces, un pavé numérique d'origine, des touches de fonctions, le tout pour moins de 10 000 F (en France), avec un écran noir et blanc. Cela signifie que cet ordinateur, qui devrait être très proche du MACINTOSH, sera proposé pour le tiers du prix de son glorieux ainé, et moins cher que l'APPLE II en configuration «équivalente». De plus, les périphériques dédiés sont annoncés dans des gam-



Un écran type de nouveaux ordinateurs «conviviaux».

mes de prix correspondantes, soit 100 $ le lecteur supplémentaire de 500 Ko, et un disque dur de 10 Mo pour moins de 400 $. Faites le compte, c'est époustouflant. De plus la gamme devrait être étoffée vers le bas par un familial, possédant 128 Ko en standard, et ce pour moins de 1 800 F. Voilà un pari, et même une série de paris, assez intéressants, et dont la réalisation mérite d'être suivie.

Un domaine où la France ne fait pas rire du tout les Américains, est celui de l'intelligence artificielle, et de ses applications. On le sait, notre pays est en avance pour le développement des systèmes experts, tarte à la crème des années 80, mais aussi dans la recherche concernant la robotique, et la synthèse vocale et musicale. Hélas, la diffusion ne suit pas, eu égard au fossé qui semble infranchissable, quoique un peu moins ces derniers temps, entre les

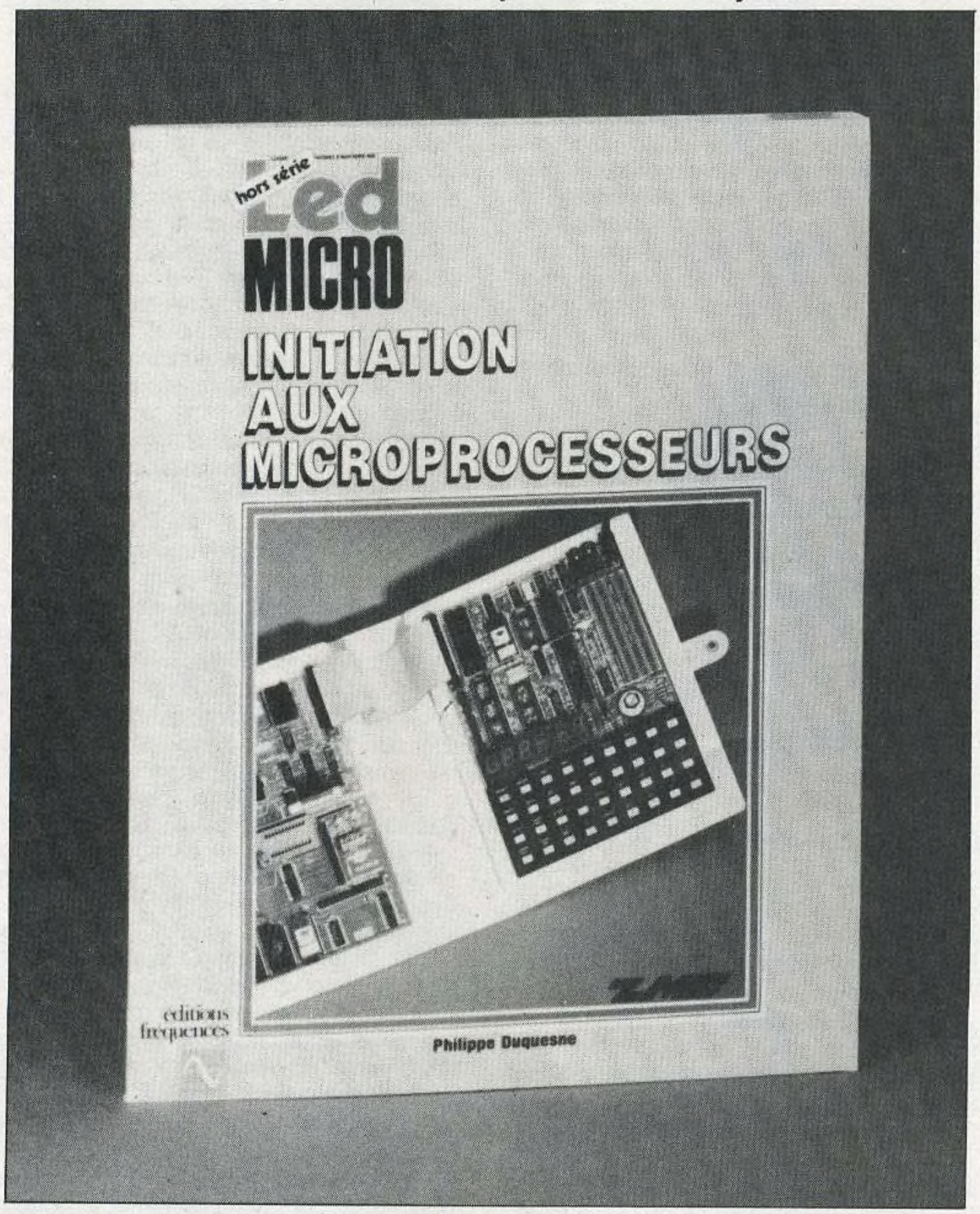
laboratoires, universitaires ou non, et l'industrie, pour tout ce qui ne semble pas immédiatement source de profits. Malgré cela, de nombreux chercheurs et industriels d'outre-Atlantique ont fait le déplacement pour le spécial Sicob, dans le but de visiter l'exposition de machines parlantes qui étaient proposées sur le parvis, dans le cadre du carrefour de la communication, et sont revenus impressionnés par la qualité des réalisations qu'ils ont pu apercevoir. On s'approche à grands pas, des deux côtés de l'Atlantique, à la commande vocale complexe de machines, même pour des produits grands publics. Il est satisfaisant de voir que la France semble être parmi les pays de pointe dans ces domaines, très prometteurs, et sources d'investissements importants, donc de développements pour les industries concernées.

Le nouvel Atari... Macari ou Jackintosh ?

Un autre domaine en grande expansion est celui des logiciels éducatifs pour les jeunes. Ce marché semble prendre ici une direction enfin intéressante, délaissant les jeux débilitants pour créer des programmes que l'on pourra vraiment qualifier d'éveil. Ce nouveau départ des soft pour jeunes est très nouveau, mais le développement se fait à une vitesse tout particulièrement remarquable. A l'origine, l'essor a été voulu par les institutions pédagogiques, universités, écoles et autres. C'est pourquoi les premiers logiciels disponibles ont été pour la plupart mathématiques. Mais il s'agissait des balbutiements et la diversification n'a pas tardé. Ce qu'il faut bien comprendre, c'est que les problèmes ont évolué au fil des années. Dans les années 70, le gros problème était matériel, dû aux limitations des supports de stockage (cassettes) et au prix des mémoires (en général les premiers ordinateurs étaient livrés avec 4 ou 8 koctets). Maintenant, les limitations sont plutôt logicielles et ici, l'intelligence artificielle a actuellement la part belle, car les nouveaux algorithmes développés dans ce domaine sont immédiatement applicables, par exemple l'alpha-béta pour les programmes qui ont à effectuer des recherches parmi des possibilités quantifiées (aux échecs par exemple).
Toujours est-il que les parents ou les éducateurs n'ont plus à s'arracher les cheveux de désespoir devant la pauvreté des logiciels disponibles. Maintenant, chaque mois, sortent plusieurs programmes de grandes qualités, qui arriveront sans doute très bientôt en France. Il est d'ailleurs très symptomatique que les magasins spécialisés laissent en démonstration permanente tous ces programmes, alors qu'auparavant, ils étaient soigneusement emballés, ce qui permettait de n'être déçu qu'une fois le produit acheté. Le plus amusant est que ces logiciels en démonstration sont testés non seulement par les parents mais aussi par les enfants, ce qui prouve que ces programmes sont à la fois attrayants et pédagogiques. Les programmes varient naturellement selon les âges visés. Pour les plus petits, les programmes proposés sont à la fois très beaux sur le plan graphique, et sont également une bonne aide au développement psycho-moteur du jeune enfant. Les choses vraiment sérieuses commencent pour les catégories d'âges à partir de 6 ans. Les programmes sont en général des aides à l'enseignement (vous savez, l'EAO), sous forme de répétiteurs, jugeant, corrigeant les résultats, et orientant la suite du travail. Vous me direz que ce type de programmes existe depuis plusieurs années, mais les critères d'évaluations étaient généralement très frustres, alors que l'on voit maintenant des produits de très belle qualité, capables d'un véritable suivi intelligent de la leçon. Cela est encourageant. D'autres programmes, dit de simulations, permettent d'enseigner à la fois des idées et des concepts (un programme de ce type fait beaucoup parler de lui en ce moment : il s'agit pour des étudiants en biologie, de lutter contre la malaria et donc d'apprendre le plus possible sur les moustiques, vecteurs de la maladie, les pesticides et les traitements). Il existe aussi des programmes de documentation, remarquables par la grande quantité de données qu'ils proposent. Tous les domaines sont visés, des sciences exactes aux sciences humaines. Ensuite sont venus des programmes de type répétiteurs, proposant par exemple des listes de mots dont un seul est correctement orthographié. Viennent enfin les programmes plus banalisés, destinés à apprendre la programmation, ce qui est un comble mais aussi une nécessité, car le plus souvent les possesseurs d'ordinateurs ne programment pas, mais se contentent d'utiliser des programmes du commerce. Espérons que ces programmes aideront vraiment les gens à se lancer dans la programmation.
Au mois prochain.

Microprocesseurs
un cours essentiellement pratique !

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



En vente chez votre libraire et aux Editions Fréquences

Bon de commande

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

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

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

COURS DE GENIE LOGICIEL

De la théorie à la pratique

Charles-Henry Delaleu

Notre histoire commence il y a un peu plus de vingt ans maintenant. La fusée est prête à partir désormais, le compte à rebours est déclenché, il n'y a rien à signaler.
Finalement, depuis quelques tirs, la fusée américaine est devenue fiable et tout permet de penser que ce lancement se passera aussi bien que les autres. 5, 4, 3, 2, 1, 0, mise à feu, la fusée s'élance normalement et commence à monter sous le soleil de Floride.
Panique au centre de contrôle, la fusée vient de dévier de sa trajectoire, elle s'écarte de son axe, elle quitte sa route, elle devient folle, elle va se jeter dans la mer des Caraïbes.
La fusée vient de sombrer et pourtant aucune anomalie n'a été constatée sur les systèmes de contrôle. Non vraiment rien, tout est absolument normal.
Dès lors, la longue enquête des ingénieurs va commencer. Tout sera passé au crible, chaque phase du processus sera ré-étudiée. Finalement, une première analyse indiquera qu'aucune panne n'a eu lieu pendant les préparatifs du vol ainsi que pendant la phase de décollage.
Ce n'est qu'après plusieurs semaines qu'un ingénieur s'apercevra en listant le programme source écrit en fortran qui gérait la commande à distance de l'envol de la fusée qu'un signe sur une équation avait été inversé.
Cette erreur, très grave de conséquences, allait remettre en question les techniques de programmation utilisées. La perte de la fusée avait coûté très cher et il convenait de trouver un remède afin d'éviter tout incident de ce genre. Les Américains allaient alors inventer le Software Engineering.
Les Français baptiseront cette technique le Génie Logiciel. Elle sera très peu utilisée au départ, mais aujourd'hui le Génie Logiciel est couramment employé dans tous les projets informatiques.
Le Génie Logiciel est une science qui se rapporte à l'ensemble des outils et des

NOTES PERSONNELLES

procédures relatifs aux phases de l'élaboration d'un logiciel. Il s'agit en fait des différentes phases qui entrent dans la vie d'un produit logiciel et des moyens qui s'y rapportent :
- Etude de faisabilité
- Spécifications
- Cahier des charges
- Fabrication du produit logiciel
- Production
- Test
- Mise au point
- Qualification
- Recette
- Formation
- Maintenance
- Documentation
- Cycles de vie, etc...

Le cours «de la théorie à la pratique» sera en fait un cours de Génie Logiciel élaboré de manière à être compréhensible pour des non informaticiens de métier. De même, il sera orienté vers la micro-informatique.
En fait, le but de ce cours est de permettre d'obtenir un minimum de rigueur et de ne pas être bloqué par un travail qui a été fait et qui, par le temps, est devenu incompréhensible.

Première partie

QUELQUES NOTIONS DE DEPART

On appelle logiciel un programme qui tourne sur un ordinateur, en anglais cela se nomme Software. Il convient de noter que dans le monde des informaticiens, les termes anglais sont les préférés (il paraît que cela fait plus pro).
Les progiciels sont en fait des logiciels copiés en grande quantité pour être vendus comme un produit commercial classique.
Le terme de génie en Génie Logiciel ne signifie pas que l'on a en face de soi un génie de l'informatique mais plus simplement l'ensemble des techniques qui gravite autour du logiciel.
Au début de l'informatique, le matériel et le logiciel étaient considérés comme un seul et unique produit. Depuis, chaque partie est vendue séparemment. Chez les fabricants, ces deux ensembles sont traités par des équipes différentes.
Il devient très rare de trouver des ingénieurs spécialisés en matériels et en logiciels. Il est courant de rencontrer des individus pour qui le fonctionnement d'une machine n'a pas de secret, qui ne connaissent aucun langage de programmation et inversement.
En 1960, soixante quinze pour cent du coût d'un système informatique était occupé par le matériel. En 1970, le matériel et le logiciel se partagaient l'enveloppe financière. En 1985, pour les sociétés qui réalisent elles-mêmes leurs programmes sans acheter de progiciels (ex : les banques, certaines grosses industries), le coût du matériel est passé à dix pour cent et le coût des programmes a grimpé à quatre vingt dix pour cent de l'investissement total. Il convient de noter que deux choix se profilent à l'horizon :
1. L'utilisateur d'un système informatique désire lui-même réaliser ses propres programmes pour des raisons diverses :
- Sécurité en regard de l'extérieur
- Manque d'application équivalente sur le marché
- Stratégie (piratage, etc...)
- Politique personnelle

Le coût d'une telle décision implique des dépenses très importantes en réalisations des logiciels. Dès lors, une technique rigoureuse doit être appliquée afin d'éviter tout problème inutile.
2. L'utilisateur préfère acheter des progiciels, dans ce cas, les dépenses seront bien inférieures. Mais attention, certaines règles sont à respecter car il ne viendrait à l'esprit

de personne d'acheter une paire de chaussures sans l'avoir essayée. Une démarche équivalente doit être suivie en informatique et certaines règles de Génie Logiciel sont à appliquer :
– Test de fiabilité
– Test de conformité avec le cahier des charges
– Test de vitesse
– Test de portabilité, etc...
On appelle portabilité le fait qu'un programme qui tourne sur un certain type de machine puisse être placé sur un autre calculateur.
Même dans le cas d'achat de programmes clés en main, il est absolument obligatoire de respecter certaines règles.

UN PEU DE VOCABULAIRE

La semantique : la semantique est la science de la signification, c'est le concept.
La linguistique : c'est la science qui exprime le concept.
En bref :
Semantique = contenu
Linguistique = exprime le contenu
La linguiste nous amène directement aux langages. En programmation, ces dernières vont profondément modeler les applications.
Approche des langages :
1. Alphabet
2. Vocabulaire
3. Syntax — Comparaison entre les langages
4. Données
5. Règles spécifiques, traitement

Les grandes règles :
A) Identifier : les objets, les entités, les concepts de base
B) Attribut → fonction
C) Mesure des attributs correspondant à l'objet
Liens logiques + liens chronologiques

Avant d'entrer de manière plus précise dans l'univers des langages informatiques, il convient de se rappeler la construction d'un langage tel que le français et de ses possibilités afin d'imaginer la difficulté qui existe dans la mise en œuvre d'un langage de programmation. Comment avec un vocabulaire très pauvre et des possibilités très restreintes fabriquer des programmes qui donnent entière satisfaction ?

METHODE DE PROGRAMMATION :

Avant toute chose, il convient de noter que la programmation doit être structurée
Objectif :
– Choix du langage de programmation
– Structure de contrôle :
  – Répétition des tâches
  – Choix à déterminer
  – Enchainement
  – Sous-programmes
– Structure des données :
  – Piles
  – Listes
  – Files (enregistrements sur fichiers)
  – Arbres (tris)
  – Tableaux
  – Graphes
– Recursivité
– Algorithmique

NOTES PERSONNELLES

SIGNIFICATION DES DERNIERS TERMES UTILISES :

Pile : une pile est une file d'attente gérée selon la méthode du dernier entré-premier sorti. En d'autres termes, l'accès aux informations se fait dans l'ordre inverse ou celles-ci ont été rangées.

Liste : A la différence d'un langage interprété, un langage compilé donne naissance à un ensemble imprimé comportant le texte d'un programme source, sa traduction en langage objet, les messages d'erreur et les résultats de l'exécution. Ceci est une liste. Mais une liste c'est aussi (c'est le cas qui nous intéresse) un ensemble de données qui comporte toutes une partie lien. Ce lien peut être une adresse, un pointeur, etc... Il permet de réaliser la recherche des informations désirées.

File : Une file d'attente est un rangement par ordre chronologique des événements associés à plusieurs processus. Il existe deux types de file d'attente :
Fifo : l'ordre de traitement est identique à l'ordre d'arrivée
Lifo : l'ordre de traitement est inverse à l'ordre d'arrivée

Arbre : Un arbre est une structure hiérarchisée entre différents éléments d'informations. Il permet d'atteindre tous les éléments à la suite d'une énumération (cette notion sera reprise en détail dans la partie concernant les tris).

Tableau : Un tableau est un mode de structuration des données qui autorise un rassemblement des variables (ou tout autre donnée) de même type sous un seul nom. L'exemple le plus connu est DIM en Basic où l'accès à chaque information se fait par un ou plusieurs indices. Un indice pour un tableau à une dimension donne ceci :
DIM A (50) → crée un tableau de 50 éléments
B = A (25) → met dans B la variable à ranger dans la 25e case
Pour un tableau à deux dimensions, nous avons :
DIM A (50, 20) → crée un tableau de 1 000 éléments, 20 colonnes de 50 éléments
B = A (25,3) → met dans B la variable à ranger dans la colonne 3, à la 25e case

Graphe : La théorie des graphes serait assez longue à expliquer. Il existe toute une série de graphes qui ont chacun leur particularité. Nous ne traiterons ici que rapidement leurs utilisations.
Les graphes servent à gérer le flux des informations à traiter. Ils servent aussi à l'ordonnancement et à la décomposition des tâches à effectuer.

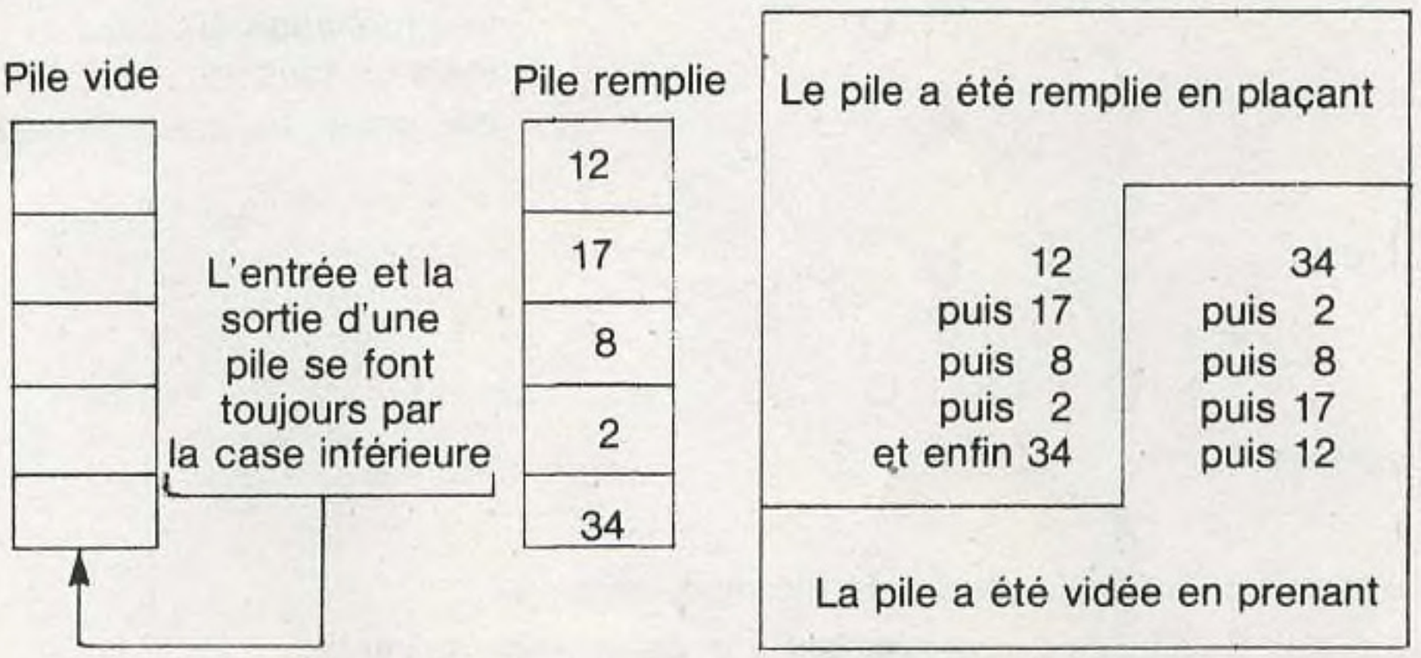
Recursivité : Une procédure est recursive lorsqu'elle doit s'appeler elle-même, ou être appelée par une procédure qu'elle a elle-même appelée. La cursivité est très utilisée en tri rapide.
Pour simplifier les choses, notons pour l'instant qu'une procédure est une partie de programme qui effectue un travail précis.

Algorithme : Lorsque l'on désire réaliser un programme, il est nécessaire de réfléchir à la manière dont va s'articuler cet ensemble : c'est de l'algorithme.
En d'autres termes, l'algorithme est une description schématique et/ou logique d'un programme ou d'une partie de programme. Nous consacrerons un chapitre entier aux algorithmes.

LA PILE

Soit cinq chiffres pris dans le désordre :

12 17 8 2 34

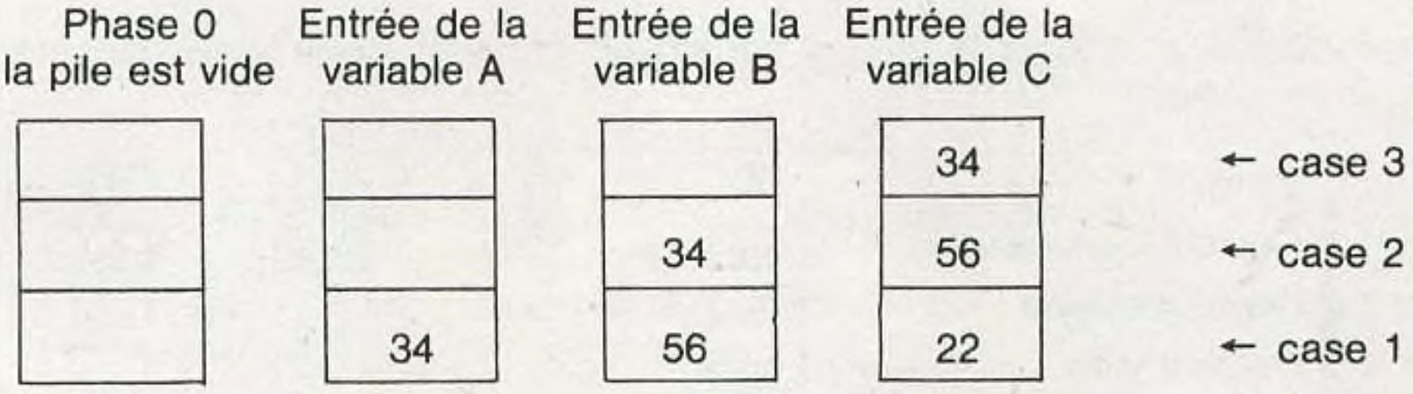


Soit 3 variables

A = 34 B = 56 C = 22

Fonctionnement d'une pile

Pour réaliser le calcul A + B + C à l'aide d'une pile, nous faisons :



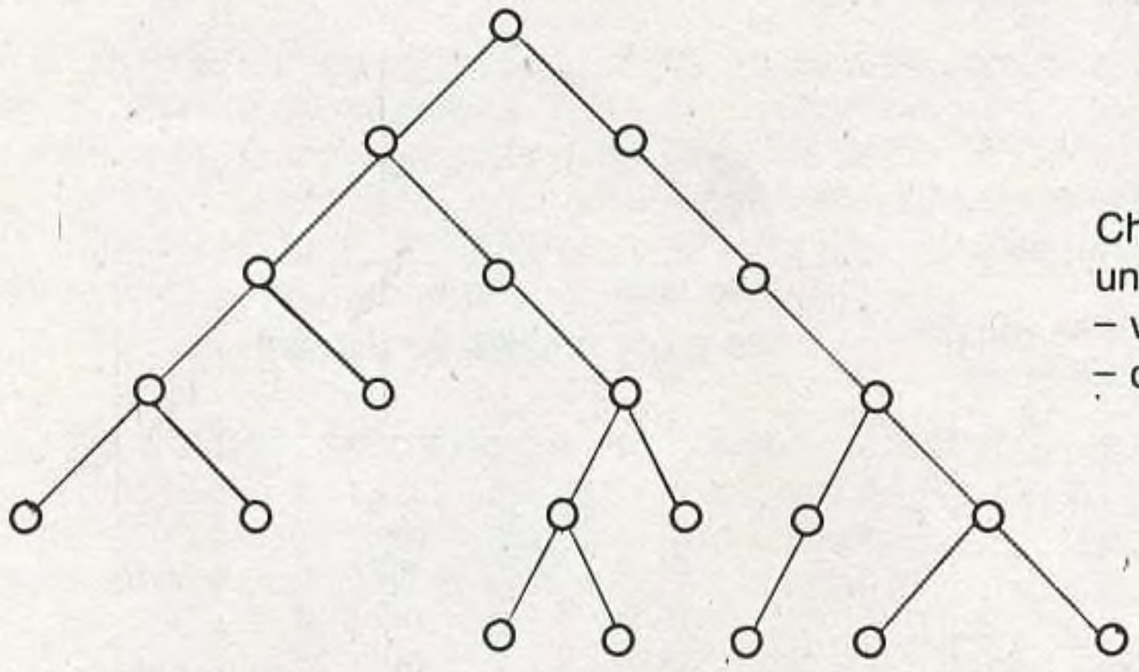
Addition avec la pile.

L'additionneur prend 22 dans la case 1
dès lors 56 passe en case 1 et 34 en case 2
L'additionneur prend 56 dans la case 1
dès lors 34 passe en case 1
L'additionneur prend 34 dans la case 1.

Conclusion : Le dernier chiffre entré dans la pile est le premier sorti
Le premier chiffre entré dans la pile est le dernier sorti.

ARBRE BINAIRE

Soit la présentation simplifiée d'un arbre.



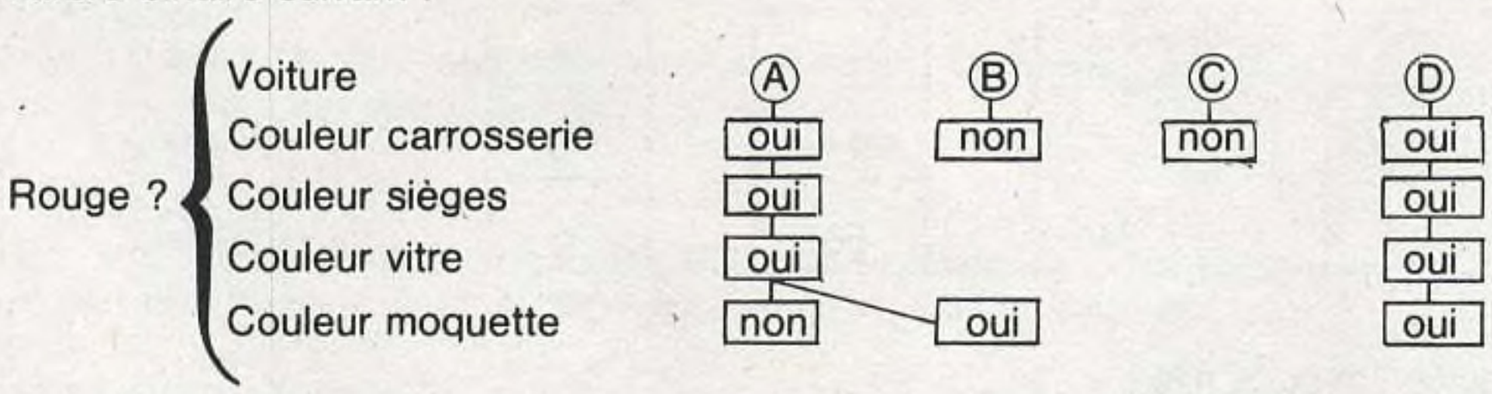
Chaque petit rond représente une information qui est :
– vraie ou fausse
– qui existe, qui n'existe pas

Ainsi, prenons l'exemple d'un petit trio multi-critères :

Nous avons 5 voitures correspondant à la description suivante :

Nom de la voiture	Couleur carrosserie	Couleur sièges	Couleur vitres (teintes)	Couleur moquette
A	rouge	rouge	rouge	bleu
B	vert	bleu	vert	rouge
C	bleu	vert	bleu	blanche
D	rouge	rouge	rouge	rouge

Trier les voitures et trouver la voiture qui est entièrement rouge nous donnerait l'arbre binaire suivant :



Soit : un seul arbre descend verticalement. Dès lors les couleurs des carrosseries B et C sont éjectées (couleur carrosserie rouge → NON). La voiture A satisfait aux tests 1, 2 et 3. Au 4e test, la couleur de la moquette rouge n'est plus satisfaite, par contre elle est conforme en B.

ATTENTION C'EST UN ARBRE TRES SIMPLIFIE.

TABLEAUX

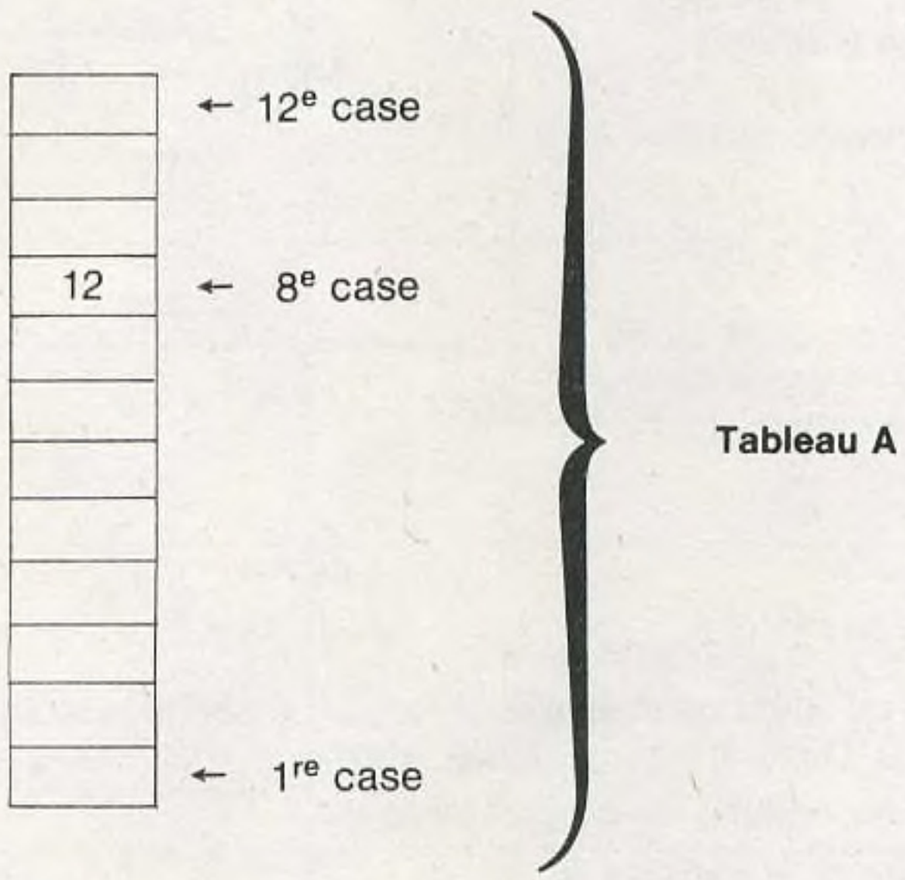
SOIT 12 NOMBRES ENTIERS

1 5 24 76 26 12 8 3 56 37 14 75

Nous pouvons les ranger dans un tableau A composé de 12 cases. Il est possible de placer les dix nombres dans n'importe quelle case. Si nous voulons mettre 12 dans la 8e case, nous dirons :

A(8) = 12

Ceci revient à faire



SOIT NOS 12 NOMBRES PLACES dans un tableau à deux dimensions. Par exemple, deux lignes de 6 colonnes :

	1	2	3	4	5	6
1	1	5	24	3	37	14
2	76	8	26	75	12	56

1 est placé en cas A(1,1) 56 est placé en A(2,6)

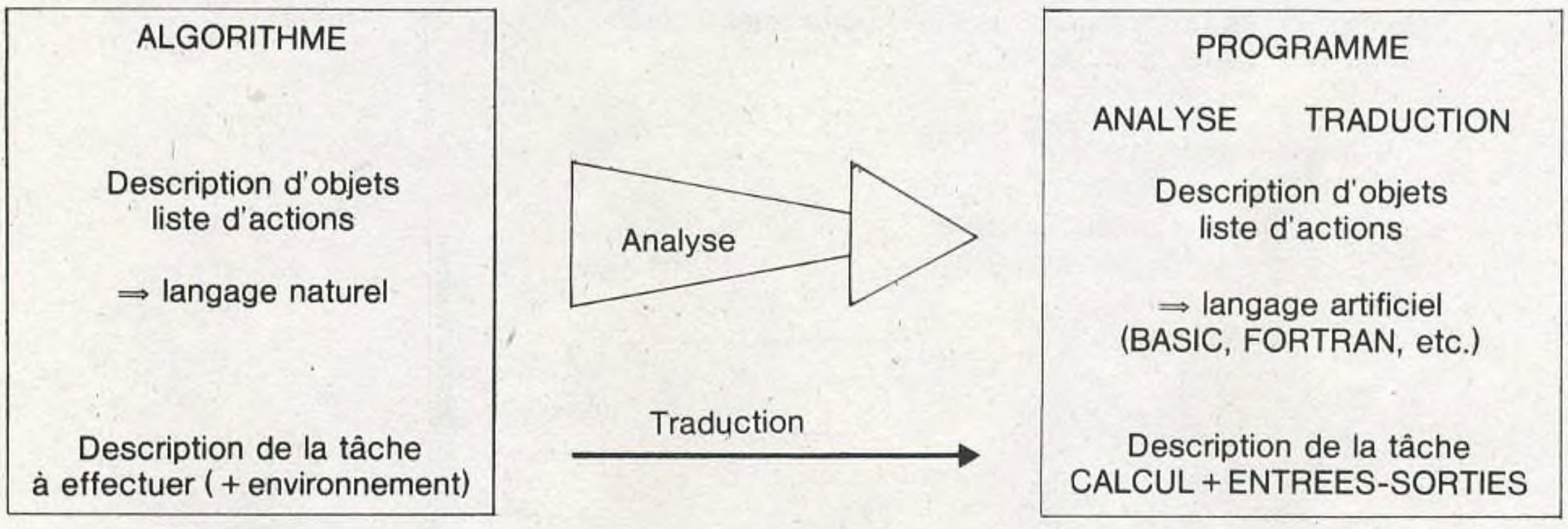
ATTENTION : Dans certains langages, on commence par l'ordre de la ligne puis de la colonne, dans d'autres c'est le contraire.

NOTA : Il est possible de réaliser des tableaux complexes qui permettent de stocker des informations denses. (Exemple : le dessin en trois dimensions en couleur).

Chaque point utilisable donne naissance à cinq informations de bases :
- point allumé ou éteint (0 ou 1)
- couleur du point (0, 1, 2, 3,...)
- position dans l'axe des x
- position dans l'axe des y
- position dans l'axe des z

Ces tableaux sont de type à trois dimensions architecturés en matrice.

L'ALGORITHME



Soit un problème posé :

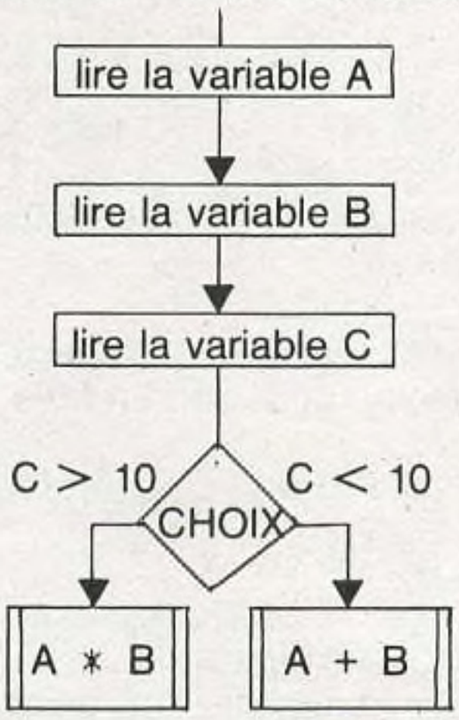
Il convient de multiplier la variable A par la variable B, si la variable C est inférieure à 10. Dans le cas contraire, additionner les variables A et B.

L'algorithme de ce problème est :

1) Lire la variable A
2) Lire la variable B
3) Lire la variable C
4) Si C > 10, alors multiplier A par B
5) Si C < 10, alors additionner A + B

(les variables peuvent être préalablement déclarées en ENTIERS ou REELS)

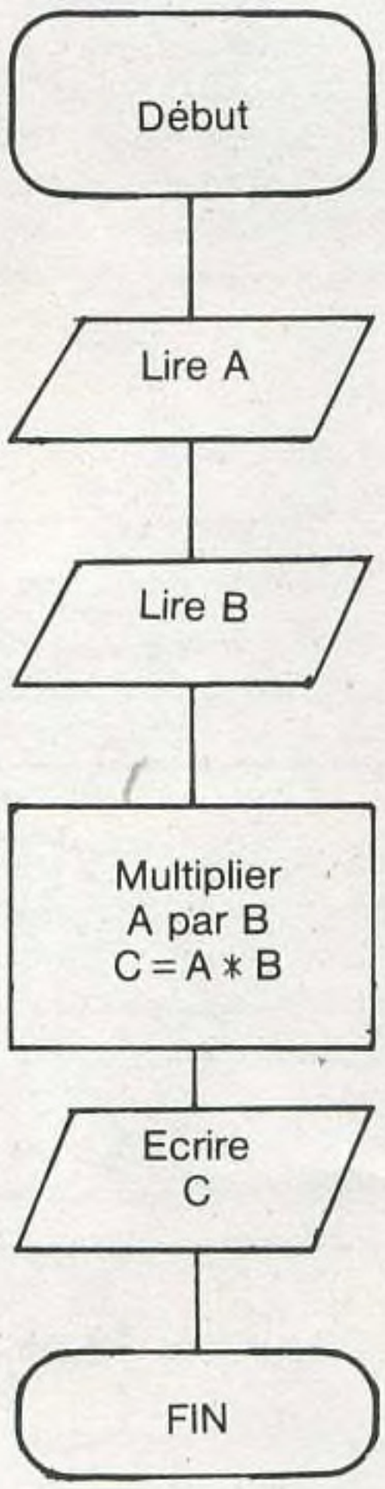
En fait un algorithme n'est en réalité qu'une description chronologique de la résolution d'un problème posé.
Cette dernière peut être réalisée de manière schématique par un organigramme.



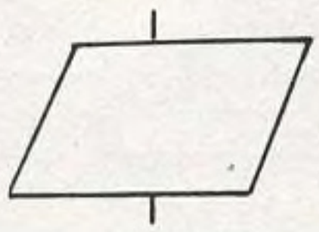
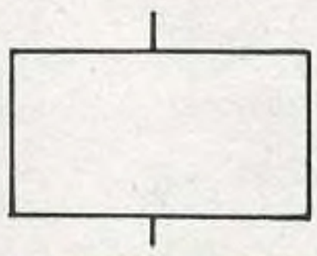
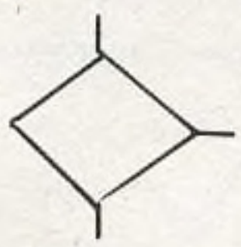
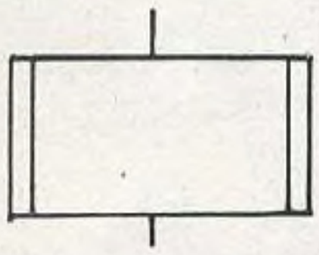
NOTA : Le terme d'algorithme ne vient pas du mot rythme, mais de l'origine traduite du nom d'un personnage persan très connu pour ses travaux sur la logique.

ORGANIGRAMME

Les symboles de base



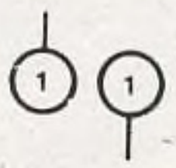
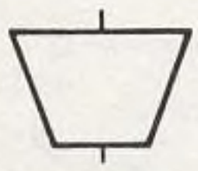
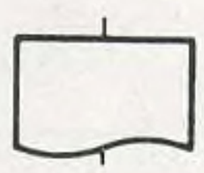
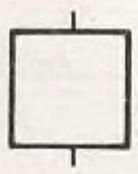
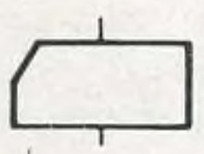
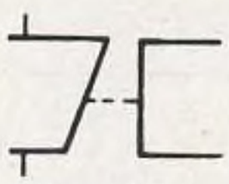
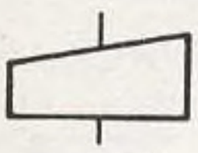
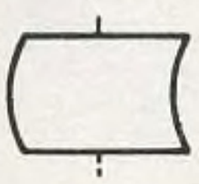
Soit lire A et B, multiplier A par B et donner le résultat C de la multiplication

Symboles de programmation	
Symboles	Signification
	Début Fin Interruption ①
	Entrée/sortie ②
	Traitement ③
	Décision ④
	Sous-programme ⑤

1) Début, fin, interruption : Ce symbole équivaut aux ordres BASIC suivant : END, STOP, INTR, etc.
2) Entrée/sortie. Ce symbole équivaut aux ordres BASIC suivant : INPUT, PRINT.
3) Traitement : Ce symbole correspond aux opérations diverses effectuées par le programme tel que :
   – Calculs
   – Opérations sur chaînes de caractères, etc.
4) Décision : Ce symbole équivaut à l'ordre BASIC suivant : IF... THEN... ELSE...
5) Sous-programme : Il s'agit de tout sous-programme ou macro-instructions utilisés par le programme.

ORGANIGRAMME

Les extensions aux symboles de base

1 	6 	11 
2	7 	12 
3 	8 	13 
4 	9 	14 
5 	10 	15 

1 : Connecteur
2 : Liaison
3 : Explication : ce symbole équivaut en BASIC à REM
4 : Préparation
5 : Communication directe homme-machine : visualisation écran
6 : Opération manuelle
7 : Traitement additionnel
8 : Communication directe homme-machine : imprimante
9 : Mémoire externe
10 : Mémoire non connectée : ficher non ouvert
11 : Document imprimé : listing, etc.
12 : Carte perforée
13 : Bande perforée
14 : Bande magnétique
15 : Disques magnétiques

INTRODUCTION VERS L'ANALYSE

Classes de problèmes : spécification

Quels que soient les problèmes que l'homme doit résoudre, il apparaît que ceux-ci sont généralement posés en **termes flous.** Il n'est pas rare – si ce n'est général – que la solution à ces problèmes fasse appel à des interprétations, des expressions plus ou moins explicites faisant référence à l'intuition et à l'imagination.
En bref, les solutions apportées sont diverses et échappent à des critères précis d'appréciation. Il convient de noter que les problèmes professionnels appartiennent à cette catégorie.

Exemple :
Un épicier doit passer commande de bouteilles d'huile, il doit éviter à la fois les ruptures de stocks et leurs gonflements excessifs. Il ignore la consommation en terme de statistique, il ne connaît pas le coût exact d'une rupture de stock ni d'un sur-stockage. Il a un problème flou à résoudre.

Il arrive que d'autres problèmes soient posés en **termes précis,** mais aucune règle connue n'est applicable directement. Leur résolution implique donc des choix plus ou moins arbitraires. L'efficacité de la solution n'apparaît qu'une fois le problème résolu.

Exemple :
Un chauffeur de taxi a une course à réaliser. Il désire minimiser le plus possible sa consommation d'essence. Il a un problème précis à résoudre : choisir le chemin le plus court ou le chemin qui risque d'être le moins embouteillé. il se fiera à son intuition et non à une évaluation de toutes les solutions possibles.

Il existe enfin des problèmes posés en termes précis, dont la solution est connue de manière acceptable. Cette solution peut être traitée mécaniquement sans erreur. Ces problèmes sont **solubles automatiquement.** Les travaux répétitifs et simples appartiennent à cette troisième catégorie.

Exemple :
La tenue des livres comptables.

Il est possible de traiter ces problèmes par une machine.

N.B. : La machine peut aussi servir à titre expérimental afin de résoudre des problèmes précis sans règle applicable de résolution. Il s'agit de simulation.

Exemple :
A la construction d'un ascenseur, on peut établir qu'il sera manœuvré, demande par demande, dans le cas d'un immeuble de petit taille. Par contre, pour un grand bâtiment, il sera nécessaire de tenir compte de tous les appels (voire des mouvements des autres ascenseurs). Ce choix, ainsi que celui du nombre d'ascenseurs, peut être guidé par une simulation.

La **spécification** d'un problème à résoudre est l'exposé de celui-ci. Or, écrite en langue naturelle, elle est infiniment plus difficile à rédiger qu'il n'y paraît. L'habitude de s'expliquer verbalement entraîne automatiquement les problèmes suivants :

- sous-spécification, oubli d'un détail
- ambiguïté et imprécision des termes utilisés
- contradiction, incohérence dûe à un manque de réflexion
- sur-spécification de certains détails.

NOTES PERSONNELLES

Retour vers l'algorithme

L'ensemble des règles menant à la solution d'un problème soluble automatiquement doit décrire précisément les entités indissociables suivantes :

- les objets
- les actions.

Les objets :

L'existence des objets est fondamentale et utile pour appliquer des règles.

Les actions :

Les actions vont s'exercer sur les objets.

Cet ensemble s'appelle algorithme ⟹ RESOLUTION DES PROBLÈMES

NOTA : Pour être performant, un algorithme se doit d'être décrit en un nombre limité d'objets et d'actions. Les actions doivent être traitées dans un temps minimal.

LA MACHINE

Nous avons vu qu'un problème posé en termes précis pouvait être traité grâce à une machine. La mise en œuvre d'un algorithme pouvant résoudre ce problème implique qu'il soit posé en tenant compte des possibilités du calculateur sur lequel il sera implanté.

- Il convient de vérifier la conformité des objets décrits dans l'algorithme avec la machine.
- Les actions prévues peuvent-elles être exécutées par le calculateur ?

NOTA : On appelle action primitive, toute action pouvant être traitée directement par un processeur, ou par un «effecteur» (périphérique, etc.)

L'avènement de l'électronique puis des ordinateurs implique que chacune des caractéristiques des objets soit traitée et fournie par des niveaux de tension ou de courant. Grâce à ces niveaux, les objets sont ramenés à un ensemble de valeurs discontinues.

Il est possible de conserver dans des mémoires de la machine, les informations désirées.

Rappelons qu'il existe deux types de mémoire :

Les mémoires internes (ROM, RAM, etc.)

- Rapides d'accès
- Faibles capacités.

Les mémoires externes (disques, bandes magnétiques) :

- Lentes
- Grandes, voire très grandes capacités.

L'ANALYSE

Nous avons vu qu'un algorithme est écrit en langage naturel. Or, le langage naturel est destiné à un interlocuteur humain. La plupart des langues sont riches et ambiguës. Enfin, dans une conversation, on fait appel à l'intelligence, aux connaissances et à l'imagination.

Il est impossible dans ces conditions de prévoir un algorithme.

Il convient de ramener l'algorithme à une description simple et précise d'objets et d'actions. Il est nécessaire d'utiliser un **langage artificiel** sans ambiguïté : c'est un langage pauvre.

Il est obligatoire :

- d'attribuer aux objets certaines caractéristiques parmi un ensemble prédéfini de types ;
- d'exécuter des ordres simples, éventuellement circonstanciels, mais indiscutables parmi un ensemble prédéfini d'actions.

En d'autres termes, il faut :
- utiliser un alphabet figé
- utiliser un vocabulaire limité
- utiliser une syntaxe rigoureuse
- utiliser des données pré-définies
- utiliser des règles spécifiques de traitement.

Ce langage artificiel donne naissance à un **langage de programmation.** Ce langage de programmation donne naissance à une description de l'algorithme qui s'appelle le **programme** :

Le passage entre l'algorithme et le programme est : l'ANALYSE.

Attention : L'analyse est une activité complexe, elle doit faire appel à une grande rigueur. Il convient d'appliquer certaines **méthodes** :
Il faut :
- examiner en détails les objets
- examiner les actions
- établir une bonne analyse.

Il est courant de dire : C'EST LA FAUTE DE L'ORDINATEUR

⇒ IL N'Y A PAS DE MAUVAIS PROGRAMMES, IL N'Y A QUE DE MAUVAISES ANALYSES.

IL N'Y A PAS DE MAUVAISES MACHINES, IL N'Y A QUE DE MAUVAIS ANALYSTES.

L'ORDINATEUR

Un ordinateur est une machine réalisée à partir d'une ou plusieurs cartes électroniques. Il se résume à :
- Une mémoire centrale
- Une unité de traitement.

Associé à ces deux sous-ensembles, il est possible de rencontrer autour d'un ordinateur toutes sortes de périphériques :
- Clavier
- Ecran
- Unité de stokage de mémoire de masse
- Lecteur en tout genre
- Effecteurs divers, etc.

La mémoire centrale :
Il s'agit de la mémoire résidant en machine, elle est composée de mémoire morte (ROM) et de mémoire vive (RAM). Elle permet de ranger temporairement (RAM), ou non (ROM), les informations manipulées par l'ordinateur.

L'unité centrale :
L'unité centrale est le cœur du système. Elle est capable de réaliser un certain nombre d'opérations élémentaires, telles que :
- l'addition de deux nombres
- la comparaison de deux nombres
- la multiplication simple, etc.

C'est l'unité de traitement.

PROBLEME

Soit un problème posé en termes flous

Il fait face à des

- Interprétations
- Préférences
- Imaginations
- Intuitions

Il ne peut se résoudre que d'une manière HUMAINE

Soit un problème posé en termes précis

Il fait face à des

– SPECIFICATIONS

Il peut être résolu soit par une MACHINE soit par un HOMME

Il n'existe pas de règles connues de réalisation

Le problème peut entraîner plusieurs choix

SOLUTION

Dans ce cas, la ou les solutions seront traitées par l'homme. Toutefois, il est possible de recourir à une simulation par machine.

Il existe une ou plusieurs règles connues de réalisation

Le problème peut être traité par une solution automatique

SOLUTION AUTOMATIQUE

ORDINATEUR

↕

GENIE LOGICIEL

NOTES PERSONNELLES

Généralement, directement associée à l'unité de traitement, on rencontrera l'unité de commande qui autorise la gestion des échanges entre la mémoire centrale et l'unité de traitement. En fait, cette unité de commande peut être étendue au contrôle des périphériques et des entrées-sorties.

Le microprocesseur

Le microprocesseur est en général une unité centrale réalisée en un seul boîtier. Ce dernier, de plus en plus performant, renferme désormais toute une série de circuits annexes. Ce circuit intégré de grande complexité, est devenu aujourd'hui, à lui seul, un véritable ordinateur. En 1972, il pouvait traiter des mots de 4 bits, en 1974 des mots de 8 bits, en 1979 des mots de 16 bits et en début 1980 des mots de 32 bits. Rappelons que ces formats étaient, il y a encore très peu de temps, réservés aux grosses machines. Notons que pendant le même temps, la vitesse de traitement a pu être augmentée dans de notables proportions (1→20).

Récemment sont apparus sur le marché les microprocesseurs en tranches. Ces derniers sont en fait une série de sous-ensembles de microprocesseur permettant de réaliser des microprocesseurs spécialisés.

CONCLUSION

Nous venons d'énumérer toute une série de mots qui, souvent, sont nouveaux pour le lecteur. Il convient de s'arrêter ici pour l'instant.
Les trois prochains cours seront :

Cours n° 2 : L'ALGORITHME ET L'ANALYSE
Cours n° 3 : LES CRITÈRES DE QUALITÉ d'une analyse
Cours n° 4 : STRUCTURATION.

Il n'est pas souhaitable d'aller trop vite. Il est nécessaire que le lecteur fasse un bout de chemin plus en avant dans ses concepts appris dans les 20 premiers numéros.
Le cours de programmation avancé devrait d'ailleurs faciliter la digestion de ce cours de génie logiciel qui peut paraître un peu trop ardu pour certains.
Il ne faut pas se cacher les yeux, car le plus souvent le «nouveau programmeur» est tout étonné de faire tourner son premier programme sur sa machine nouvellement acquise. Avec un peu d'humour, il se croit presque prêt à entamer de grosses applications : il n'en est rien, il y a danger !
En effet, il est facile de comparer le nouveau chauffeur avec le nouvel analyste-programmeur. Une fois les balbutiements passés, il se voit déjà en train de gagner un grand prix de Formule 1, il est en fait un danger. Bien sûr, notre «informaticien» a bien peu de chance de tuer quelques personnes sur la route. Par contre, il est certain que sans méthode, il réalisera des applications non fiables, non performantes, mais que surtout il sera incapable de les suivre dans le temps.

PETITES ANNONCES

Vds cause double emploi, Alphatronic PC (Triumph-Adler), 1 an, unité centrale 64 K, 1 drive 320 K, écran monochrome, CPM, PC, DOS Basic étendu, jeux, voir descriptif Led Micro nº 7 page 80, prix 8 000 F. Tél : (53) 87.58.44.

Vds Apple III 256 K et moniteur III avec documentation et logiciels : 15 800 F. Tél : 16 (43) 89.01.12. Mr Marie Ph.

Vds Prof 80 complet (compat. TRS 80) : 4 500 F ; doubleur : 500 F ; drive SF : 1 300 F ; DF : 1 700 F ; 96 TPI : 2 200 F + doc. Tél : (soir même tard ou W.E.) : (1) 335.18.44.

Vds Atari 800 XL (fév. 85), model Pal/Secam : 1 500 F + drive DOS 3 : 2 600 F. Mr Camier Pascal. Tél : (6) 003.01.03.

Urgent, recherche tous livres, cours, résumés, photocopies, sur divers sujets : les exposants et les poids, le ruban perforé, le feuillet et le disque magnétiques, les supports originaux, les caractères magnétiques E 13 B et C M C 7, les documents graphités et les documents photolisibles, package, assemblage, compilation. Règlement immédiat. J.P. Clavel, 7 lot. de Bellesagne 48000 Mende.

Vds Commodore 64 Secam + lecteur de disquette + lecteur de cassette + Joysticks + une centaine de programmes sur disquette (jeux et utilitaires), le tout pour 6 000 F + 4 livres. Mr Guerin, 12 rue de la Banque 75002 Paris. Tél : 703.31.27.

Vds TRS 80 Mod 1 16 K minuscules, extension LNW 32 K, sortie imp. série et parall. contrôleur floppy, 2 lect. 40 pistes Pertec, moniteur et K7, nombreux programmes possibles : 7 500 F. LNW 80 en boîtier métal, 2 fois 16 K, hte résol. graph., couleur possi. dito TRS 80 : 3 500 F. Interf. Macrotronics RTTY/CW Em/Rx 45/50/75/110 Baudot/Accii, écho et boucle de courant, logiciel K7/disq. : 1 100 F. Lecteur Pertec 40 pistes : 850 F, 2×35 pistes : 1 000 F, Olivetti FD 502 : 1 600 F. Collection Trace + 4 disquettes : 400 F. 80 Micro (5 ans) : 600 F. Livres et manuels divers. Recherche Rotor KR 400 ainsi qu'une cage. Prat Irenee F6 GAL, 5 bis rue Thirard 94240 L'Hay les Roses. Tél : 16 (1) 664.79.36.

Vds cartouches standard MSX : Mr Chin, Circus Charlie : 300 F ; duplicateur manuel à alcool Rally 376-C : 400 F ; oscilloscope Dumont 401-A : 500 F. Tél : (42) 20.48.08 le soir h.r.

Vds HP 41 CV + manuels au prix de 1 600 F pour cause 41 CX. Tél : (35) 32.32.07.

Vds Oric Atmos 48 K Peritel + cassettes + magnéto, état neuf, prix à débattre. Tél : (3) 993.62.43.

Vds 2×81 + 16 K : 5 500 F ; 2 manettes jeu : 200 F ; 4 K7 jeux + utilitaire : 150 F ; lect. K7 : 250 F ; imprimante Seikosma GP 50S (6 mois) : 1 000 F. Pauvert, 4 Le Toupin 76170 Lillebonne. Tél : (35) 38.03.15.

Vds ensemble ou séparé : Prof 80 (circuit imprimé + tous composants montés) : 2 000 F ; clavier : 500 F ; doubleur : 800 F ; carte CPM : 400 F ; drives DF/DD (à réviser) : 1 000 F les deux ; FD 1771 : 150 F ; nombreuses doc. G. Kocher, 5 allée A. Gide 45100 Orléans. Tél : (38) 63.43.83.

Vds Sinclair ZX 81 + computer Program Rubycom + 1 cassette jeu + adaptateur + cours de programmation Basic, le tout jamais servi : 1 000 F. Tél : 919.27.22 après 18 h.

BON DE COMMANDE

Pour compléter votre collection de Led-Micro

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

Je désire le nº ☐ (cocher le ou les nºs désirés)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

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

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

Nom : ..... Prénom : .....

Adresse : .....

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

Bulletin d'Abonnement

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

Nom ..... Prénom .....

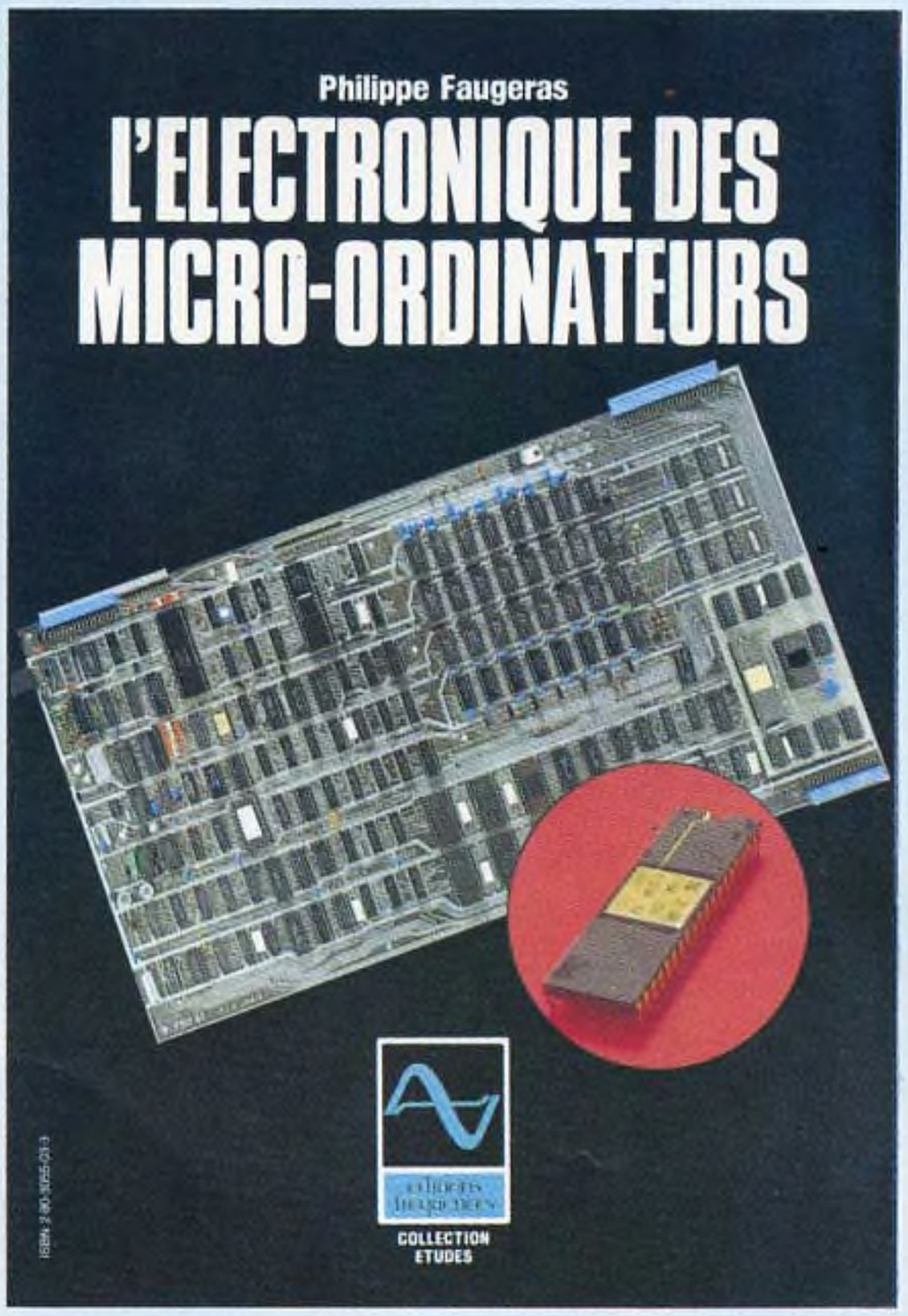
Nº ..... Rue .....

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

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

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.

MSX: le premier réel standard pour ordinateur personnel. Développé par Microsoft sur un ordinateur Spectravideo. Adapté et utilisé par un nombre croissant de constructeurs.

MSX: résidé en un mot, compatibilité du hard et du software. Le Spectravideo SV 728 MSX répond entièrement à ce standard. Sous un aspect modeste, il dissimule une puissance énorme. D'aspect extérieur discret, sobre et fonctionnel, il s'intègre aisément au cadre de vie journalier. Son clavier type professionel de 90 touches avec 'pavé numérique' séparé, comporte une touche spéciale pour la frappe des accents.

Le MSX Basic intégré avec plus de 140 commandes et instructions, complète le potentiel de cet ordinateur, qui sans difficultés peut être utilisé en gestion commerciale.

L'unité de disquette de 5 1/4" permet l'utilisation de programmes MSX-DOS et CP/M 2.2

Avec le Spectravideo SV 728 MSX prenez le bon départ pour l'avenir.

Caractéristiques	
Microprocesseur	Z80A
Horloge	3,6 MHz
Mémoire	80K octets RAM (64K octets utilisables + 16K octets vidéo pour le graphisme) 32K octets
	ROM
Logiciel	Basic MSX intégré avec plus de 140 commandes et instructions. 10 touches de fonctions
	programmables.
	Compatibilité aux systèmes MSX-DOS et CP/M.
Clavier	mécanique de 90 touches, inclus les fonctions spéciales et 'pavé numérique'
Affichage	maximum de 256*192 points en résolution graphique 40 colonnes x 24 lignes en mode texte
	32 sprites (lutins) indépendants et programmables
	16 couleurs.
Son	3 voies avec 8 octaves par voie.

Documentation complète sur demande.

SPECTRAVIDEO MSX

SV 728

SVI™

Importateur
Serepe Sarl. 103/115, Rue Charles Michels, 93200 Zac de Saint-Denis. Tél. (01) 2433622. Télex 620310 lectron f.