hors série

Led MICRO

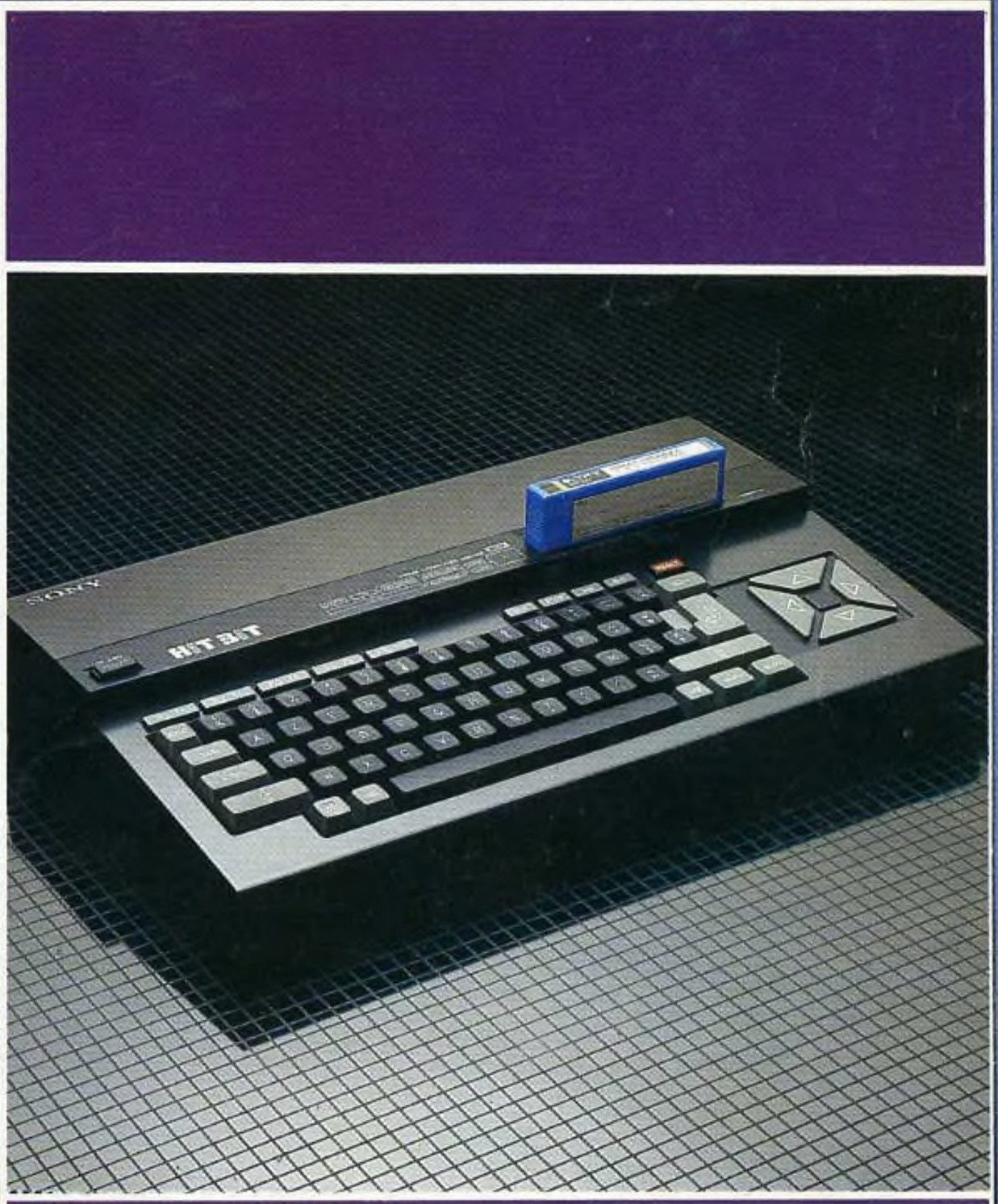
PROGRAMMATION
COURS 2ème CYCLE

COURS
N°23
Suite
2e cycle
N°3

COURS DE BASIC : les chaînes de caractères

COURS DE PROGRAM-MATION APPROFONDIE : la précision

COURS DE GENIE LOGICIEL : de la théorie à la pratique



Le Hit Bit de Sony

ISSN 0757-6889

M 1988 - N°23 (3) - 18 F MENSUEL OCTOBRE 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
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

OCTOBRE 85

COURS N°23 Suite 2e cycle N°3

COURS DE BASIC
Révision sur les chaînes de caractères
de la page 5 à la page 11
Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

COURS DE PROGRAMMATION APPROFONDIE
La précision
de la page 14 à la page 28

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

VU AU SICOB
de la page 31 à la page 33

COURS DE GENIE LOGICIEL
De la théorie à la pratique
de la page 35 à la page 49

Charles-Henry Delaleu

NOTRE COUVERTURE : Le Hit Bit de Sony, un MSX avec clavier Azerty et une mémoire de 64 ko RAM. Egalement disponibles, des périphériques et plus de 20 logiciels.

Electronique digitale ?

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

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

En vente chez votre libraire et aux Editions Fréquences

Bon de commande

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

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

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

COURS DE BASIC

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

REVISION SUR LES CHAINES DE CARACTERES

Introduction

Dans ce cours, nous allons faire un rapide tour d'horizon, de révision, des commandes permettant de travailler avec des chaînes de caractères et revenir sur certains points importants.
Les fonctions permettant de travailler sur les chaînes de caractères sont des fonctions standard, dans leur grande majorité, que l'on retrouve sur chaque BASIC. Ceci est commode et suffisamment peu fréquent pour être souligné.

1. Les commandes BASIC

1.1. Les fonctions de gestion

1.1.1. Des fonctions numériques pour les chaînes de caractères

Cela peut sembler anachronique, mais il est possible de traiter les chaînes comme des nombres, et des nombres comme des chaînes de caractères.

1.1.1.1. La fonction LEN

La fonction LEN sert à déterminer la longueur (length en anglais) de la chaîne de caractères qui lui est proposée en paramètre (figure 1.1.1), c'est-à-dire le nombre de caractères qui la compose, y compris les blancs éventuels. Le résultat est donc un nombre que vous pouvez utiliser dans des calculs, pour cadrer un texte par exemple (figure 1.1.2).

```
90   A$ = " ABCDEF "
100 PRINT  LEN ( A$ )
```

vous donnera le résultat ===> 6

Fig. 1.1.1

```
110 HTAB ( ( X-LEN ( A$ ) ) / 2 )
120 PRINT A$
```

où x est le nombre de caractères par ligne d'écran

Fig. 1.1.2

1.1.1.2. La fonction VAL

La fonction VAL joue un rôle important en BASIC. Une fois un paramètre fourni en entrée, VAL tente de l'exprimer comme un nombre. Si c'est possible, VAL renvoie la valeur de ce nombre, sinon c'est la valeur 0 qui est fournie. Pour des détails, voir le programme 1.1.2 et les réponses correspondantes. Les effets de cette fonction doivent être étudiés sur chaque ordinateur, car ils sont parfois surprenants.

```
100  A$= " 503451"
110  B$ = " DADDY 10"
120  C$ = " HELLO"
130  PRINT VAL(A$),VAL(B$),VAL(C$)
```

Programme 1.1.2

donne les résultats suivants :

```
5     car 5 est le premier nombre trouvé
1     car 1 est le premier nombre trouvé
0     car aucun nombre n'a pu être trouvé
```

1.1.1.3. La fonction CHR$

CHR$ donne le caractère ASCII du nombre fourni

```
100  PRINT CHR$ (90 )        vous affichera    Z
```

Ce nombre doit être compris dans l'intervalle [0, 255], et les nombres réels sont transformés en entiers. Attention, certaines valeurs de CHR$ sont des instructions particulières pour les ordinateurs, donc leur utilisation vous surprendra, mais il s'agit toujours de codes ASCII de caractères non-alphanumériques, par exemple celui du RETURN ou du BEEP qui est 7, est appelé aussi CONTROLE-G (figure 1.1.3.2).

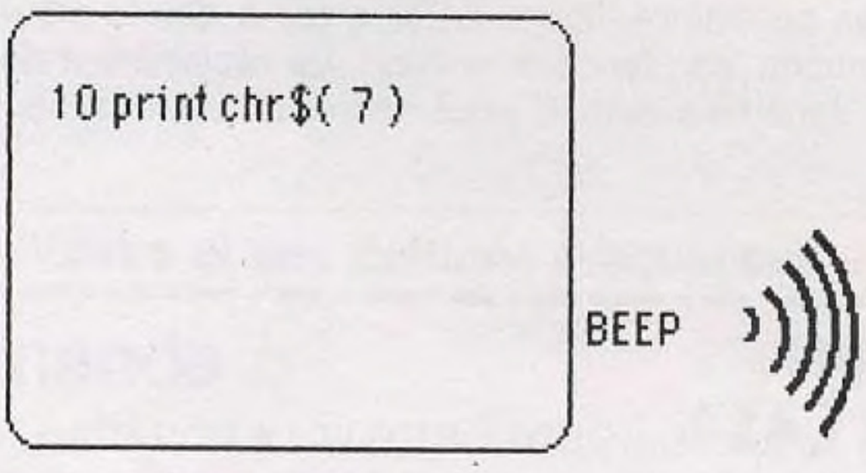


Fig. 1.1.3.2

1.1.1.4. La fonction ASC
Cette fonction est la réciproque de la précédente. Il lui est proposé une chaîne de caractères, et elle donne le code ASCII du premier caractère rencontré (le premier de la chaîne). La chaîne ne peut être vide, et doit être mise entre guillemets, ou dans une variable.

1.1.1.5. La fonction STR$
Cette fonction transforme une expression numérique en une chaîne de caractères.

2. Traitement des chaînes de caractères

2.1. Les instructions LEFT$ et RIGHT$

Ces deux instructions permettent de ne prendre qu'une partie de la chaîne sur laquelle vous travaillez. Ainsi (figure 2.1.1) LEFT$ permet de prendre les X premiers caractères de gauche de la chaîne se trouvant dans la variable A$. RIGHT$ permet de faire exactement la même chose, mais en partant de la droite de la chaîne (c'est-à-dire la fin de la chaîne, en France tout au moins !!)

```
100 A$ = " ABCDEFGHIJ"
110 PRINT LEFT$(A$,5)
120 PRINT RIGHT$(A$,5)
```

vous rendra en sortie le résultat :

```
ABCDE
FGHIJ
```

Fig. 2.1.1

Ces deux fonctions sont très utiles, car ce sont les seules, avec MID$, à permettre de travailler sur une partie des chaînes de caractères. Le résultat est stocké sous forme d'une variable alpha-numérique (figure 2.1.2), ce qui en permet une utilisation simplifiée.

```
100 A$ = " ABCDEFGHIJ"
130 B$ = LEFT$( A$,3)
140  PRINT B$
```

affichera :

```
ABC
```

Fig. 2.1.2

2.2. L'instruction MID$

Cette instruction permet de récupérer une «sous-chaîne» d'une chaîne de caractères, ce qui signifie un morceau, un bout, de la chaîne. Il est nécessaire de donner comme paramètres la longueur de la sous-chaîne que l'on attend, ainsi que l'indice du premier caractère souhaité (figure 2.2.1). Il est possible, à partir de MID$, de retrouver les deux instructions LEFT$ et RIGHT$, comme il est montré à la figure 2.2.2.

```
100 A$ = " AZERTYUIOP "
110 PRINT MID$(A$,5,3)
```

vous proposera la sortie suivante :

```
ERTYU
```

Fig. 2.2.1

```
100 A$ = " AZERTYUIOPQSDFGHJ "
110 B$ = MID$(A$,X,1)
120 C$ = MID$(A$,X,LEN(A$)-X)
```

où X est la longueur de la chaine souhaitée.

Alors, B$ est un LEFT$, et C$ est un RIGHT$

Fig. 2.2.2

2.3. La concaténation de deux chaînes

Jusqu'à présent, les traitements que nous avons détaillés ne sont que diverses méthodes pour réduire la longueur, le nombre de caractères d'une chaîne. Il est très souvent indispensable de réunir (concaténer) deux chaînes. Ceci se fait à l'aide du symbole arithmétique + (figure 2.3.1), de la manière la plus naturelle qui soit.

```
100 A$ = " BONJOUR "
110 B$ = " COMMENT ALLEZ VOUS ?"
120 C$ = A$+B$
130 PRINT C$
```

vous aurez alors :

```
BONJOUR  COMMENT ALLEZ VOUS ?
```

Fig. 2.3.1

3. Les instructions non-standard

Il existe quelques instructions que l'on ne trouve que sur quelques machines. Ces instructions servent au confort d'utilisation et sont, en général, aisément programmables sur les machines qui ne les proposent pas, à l'aide des fonctions disponibles. Donc, si votre BASIC présente des lacunes, vous pourrez en profiter pour vous exercer à les programmer.

3.1. SPACE$

SPACE$ permet de générer des blancs dans une chaîne de caractères. La commande est simplement :

```
SPACE$(6)
```
pour obtenir six blancs

Un exemple qui permet de justifier du texte à gauche :

```
10 A$= "BONJOUR"

20 B$= SPACE$(x-2*LEN(A$))+A$
```

ici, x est le nombre de caractères par ligne dont vous disposez.
ou bien, ajouter un écho à gauche :

```
10 A$= "BONJOUR"

20 B$ = A$ + SPACE$(x-2*LEN(A$))+ A$
```

Superbe, non !!!

En général, cette commande peut être remplacée par des tabulations. Sinon, elle est très simple à simuler, de la façon suivante :

```
10 FOR I   1 TO N : SP$ = SP$+ " " : NEXT
```

où N est la longueur de blancs désirée.

3.2. La commande PRINT USING

Il s'agit d'une des commandes les plus utiles, et pourtant, elle n'existe par sur tous les systèmes. Elle permet de préparer un texte pour une mise en page. Le mois prochain, nous étudierons un programme capable de simuler un PRINT USING.

3.3. Les commandes BIN$, OCT$ et HEX$

Conversion d'un nombre en une chaîne de caractères contenant sa décomposition en base 2 (BIN$), 8 (OCT$), ou 16 (HEX$). Par exemple :

```
10 PRINT 17; OCT$(17)      donnera la sortie    17      21
20 PRINT 17; HEX$(17)      donnera la sortie    17      11
20 PRINT 17; BIN$(17)      donnera la sortie    17  10001
```

3.4. La commande INKEY$

Elle permet de lire un unique caractère au clavier, lors du passage sur cette commande, sans affichage de ?, comme avec INPUT, et sans interruption, même si aucun caractère n'a été frappé, auquel cas, INKEY$ est la chaîne vide.

3.5. La commande INSTR

Recherche de la position d'une sous-chaîne dans une chaîne. Si la sous-chaîne n'apparaît pas, la réponse est 0, sinon c'est la position du premier caractère de la sous-chaîne dans la chaîne.

3.6. Les commandes DEFSTR et DIM

DEFSTR permet, sur certains systèmes, de définir explicitement des variables chaînes de caractères, avec ou sans $. DIM permet le dimensionnement des chaînes, mais avec $ impérativement. Certains systèmes imposent la déclaration avec DIM.

10 DEFSTR A(12) est alors une chaîne d'au plus 12 caractères.

4. Les problèmes de place mémoire avec les chaînes de caractères

Nous avons déjà évoqué ce problème dans le cours sur les codages, du numéro 21. Les chaînes de caractères sont des structures utilisant une place très importante en mémoire. Selon le type de déclaration, l'optimisation se conçoit différemment.

4.1. Codage déclaratif

Ici, les variables chaînes sont déclarées dans le programme, donc la place maximale qui leur est nécessaire est connue du programme, avant toute utilisation éventuelle. En déclarant de telles variables, vous pourrez tester immédiatement si le programme peut en contenir autant. Dans le cas contraire, il vous faudra vous limiter. Mais ici, une méthode peut être envisagée, qui consiste à ne prendre que le nombre de variables strictement nécessaire, et à réemployer les mêmes chaînes le plus souvent possible, voire même une seule – ce qui est parfois possible – ce qui fait que le programme est réécrit toujours dans les mêmes cases mémoires, et votre seul problème consiste à déterminer la longueur de la plus grande chaîne que vous utiliserez avec chaque variable.

4.2. Codage dynamique

Si votre BASIC vous permet la gestion automatique des variables alphanumériques, alors ces variables sont gérées entièrement de façon dynamique, c'est-à-dire au coup par coup, contrairement aux autres variables. En effet, l'occupation mémoire dépend de leur longueur qui peut évoluer avec le programme.

Pour cela, de nombreux problèmes peuvent apparaître. Prenons un exemple très simple : la variable A$ est créée et donc stockée sur une certaine taille mémoire. La variable B$, créée par la suite, est stockée après A$. Si vous modifiez A$ et que cette modification allonge A$, la place précédemment réservée à A$ ne suffit plus, donc A$ est stockée ailleurs, l'ancienne place étant inutilisée, donc perdue. Ceci entraîne la création de «trous» dans la mémoire, et sur les ordinateurs familiaux, il n'existe pas toujours de commande permettant de récupérer cette place perdue. Donc, c'est à vous de faire très attention à ce phénomène qui peut, à lui seul, occasionner la saturation de la mémoire, et de toute façon la réduit de manière catastrophique. Par exemple, une chaîne de 50 caractères, donc très moyenne, occupe environ, et selon les codages, 60 octets. Si vous la modifiez une dizaine de fois, en l'allongeant, vous aurez perdu 9 × 60 octets, soit 540 octets. Si vous disposez de 8 Ko, votre problème est tout de même aigu.

4.3. Cas des fonctions de traitement des chaînes de caractères
Ces instructions aussi doivent être utilisées avec circonspection. En effet, tronquer une chaîne est une opération courante. Si vous utilisez une instruction du type :

```
10 B$=LEFT$(A$,5)
```

l'opération aura pour effet de créer B$, donc de stocker une nouvelle variable. Pour certains ordinateurs, et si vous n'avez plus besoin par la suite de A$, il est préférable de choisir une instruction du type :

```
10 A$=LEFT$(A$,5)
```

qui réutilise la place occupée par A$. Nous avons bien dit pour certains ordinateurs, car d'autres, même dans le deuxième cas, recréent une nouvelle variable A$, à un nouvel emplacement, sans s'occuper de son existence préalable, donc deux inconvénients se trouvent cumulés : perte du premier A$, car bien qu'il soit toujours dans la mémoire, vous ne pourrez le récuperer, et perte de place. Pour bien voir les précautions à prendre avec ce type de méthode, si vous écrivez :

```
10 A$=A$
```

vous dupliquerez dans la mémoire le codage de A$, sans pouvoir en tirer le moindre avantage. Bien sûr, cet exemple est tiré par les cheveux, mais une instruction plus naturelle est :

```
10 B$=A$
```

qui permet de travailler sur le contenu de A$, sans pour autant perdre sa valeur initiale. L'utiliser revient à dupliquer la zone mémoire où se trouve stockée A$.

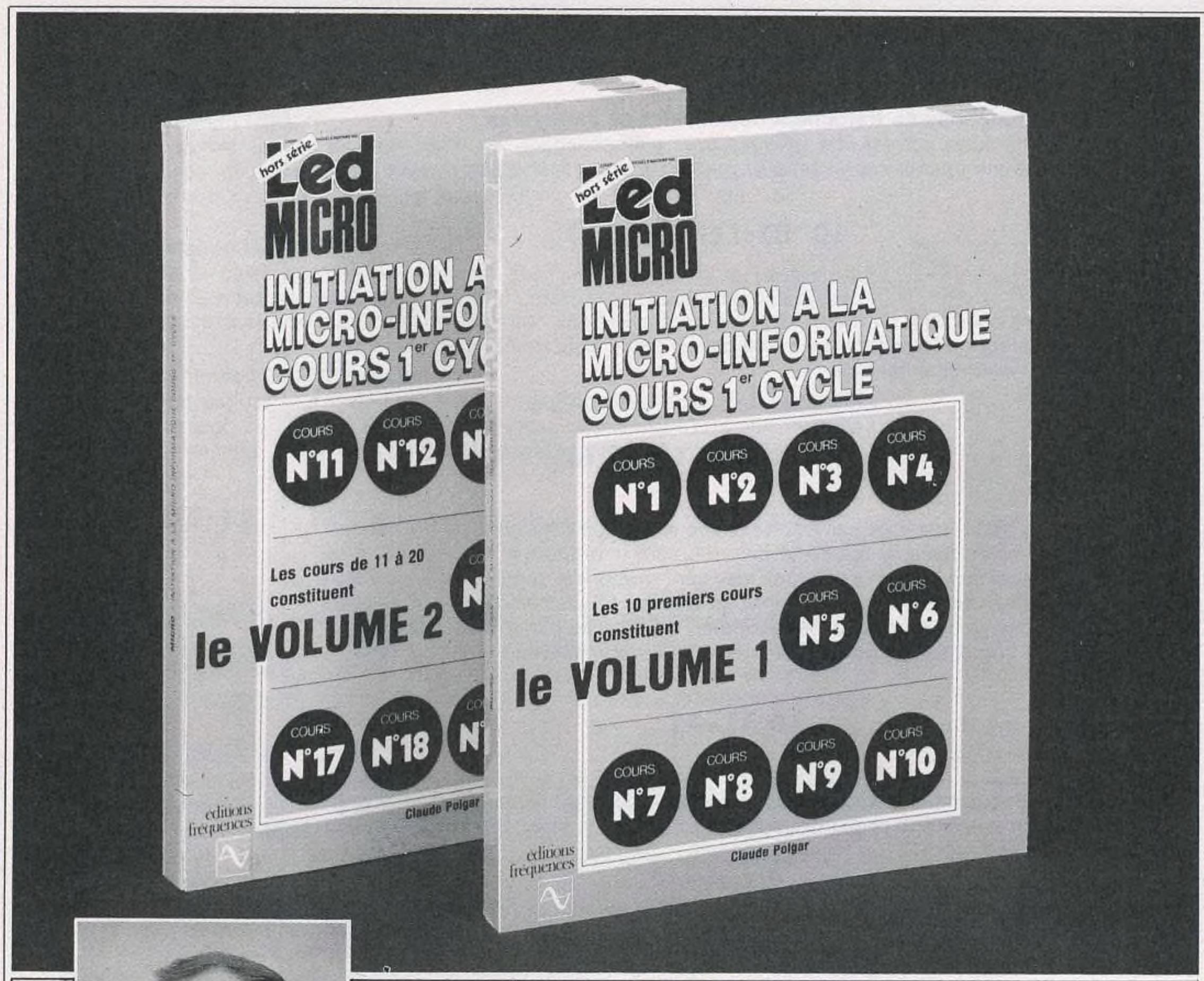
4.4. Des cas particuliers
En APPLESOFT, si vous écrivez une ligne du type :

```
1020 A$=" impôts sur le revenu "
```

A$ est stockée directement dans le programme, ce qui permet de ne pas emplir inutilement la mémoire. Naturellement, à la première modification de A$, ceci n'est plus vrai. Malgré tout, ce peut être très intéressant, en place mémoire, mais aussi en temps d'exécution car il n'est pas nécessaire d'aller chercher la valeur de la variable à l'autre bout de la mémoire.

5. Conclusion

Nous avons fait là un tour d'horizon complet des commandes.
Le mois prochain, nous ferons des exercices sur ces commandes, avec un programme pour créer un PRINT USING, une recherche de sous-chaînes dans le programme de répertoire téléphonique, et un jeu de MOT LE PLUS LONG.



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

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

Un 3e volume en préparation prévu fin octobre 85

DE NOMBREUX ADDITIFS

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

C. POLGAR

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

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

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

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

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

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

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

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

Initiation à la micro-informatique C. Polgar

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

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

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

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

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Maintenant que vos programmes ont gagné en efficacité, il est temps d'améliorer les résultats fournis par l'ordinateur. Pour cela, nous allons vous exposer une des grandes faiblesses de tous les ordinateurs : la précision des résultats.
Après quoi, nous commencerons une rubrique qui attend beaucoup de vous : les exercices d'application.
Toutefois, ne vous alarmez pas, les exercices ne doivent pas tous être faits dans le mois. Choisissez celui qui correspond le plus à votre niveau.
Bon courage !

COURS N° 3

La précision

PLAN DU COURS

1. Bien définir et bien ordonner ses formules de calculs
   1.1. Un exemple plutôt qu'un long discours
   1.2. Quelques règles utiles

2. Les erreurs d'approximation

3. Application aux domaines de gestion

4. Exercices d'application
   4.1. Le calendrier
   4.2. Le jeu de Marienbad
   4.3. Les carrés magiques
   4.4. Calcul de N décimales de PI
5. Correction des cours précédents

NOTES PERSONNELLES

1. BIEN DEFINIR ET BIEN ORDONNER SES FORMULES DE CALCULS

1.1. Un exemple plutôt qu'un long discours

Nous allons voir, à l'aide d'un exemple simple, que les limitations dues au codage des variables (cf. le cours de programmation approfondie du mois de juin) entraînent des pertes de précision qui peuvent jouer de vilains tours. Ainsi, s'il est facile (pour un mathématicien) de résoudre une équation de la forme,

$$R = \frac{X+Y-X}{Y} \qquad \text{qui donne } R = 1$$

il en est autrement pour un calcul effectué sur ordinateur avec des valeurs particulières de X et Y. On écrira la formule sous la forme usuelle en BASIC :

$$R = (X+Y-X)/Y$$

qui fournira le bon résultat pour la plupart des valeurs de X et Y. Cependant voici deux applications numériques bien surprenantes :

$X = 1$ et $Y = 10^{-5}$ donnent $R = 1{,}00000761$ (sur APPLE II)
$X = 10^4$, $Y = 10^{-5}$ donnent $R = 0$.

Par contre, si le calcul avait eu pour forme $R = (X - X + Y)/Y$, il fournirait pour les mêmes valeurs des résultats exacts. L'ordinateur effectuant les calculs de gauche à droite, il calcule de la façon suivante :

$(X+Y-X)/Y$

1re opération	R1 = X+Y	(R1, R2 et R3 sont des variables
2e opération	R2 = R1−X	internes à l'ordinateur,
3e opération	R3 = R2/Y	R prendra la valeur de R3).

$(X-X+Y)/Y$

1re opération	R1 = X−X	(qui donnera toujours 0).
2e opération	R2 = R1+Y	(donc R2 = Y)
3e opération	R3 = R2/Y	(donc R3 = 1).

Cet exemple peut paraître quelque peu trivial, cependant il doit nous mettre en garde contre les pièges de la «traduction informatique» d'un calcul parfaitement défini au niveau mathématique.

1.2. Quelques règles utiles

L'idée essentielle à retenir est que, contrairement aux théories exactes des mathématiques, l'informatique apporte pour chaque formulation un résultat différent. Il est donc nécessaire de choisir en fonction du contexte la formule adaptée (en estimant l'ordre de grandeur des différentes variables calculées). Le plus souvent les erreurs proviennent de l'utilisation de deux valeurs d'ordres de grandeur très différents, le résultat étant de ce fait entaché d'erreurs.

Voici quelques «conseils» et expériences :

a) Répartissez équitablement les opérations à effectuer afin d'éviter des dépassements de capacités et/ou des pertes de précision, voire de générer des résultats aberrants. Par exemple, pour calculer

$$1/X - 1/(X+1)$$

avec X très grand il vaut mieux utiliser une autre forme de calcul c'est-à-dire :

$$1/(X * (X+1)).$$

Faites l'essai, vous comprendrez très vite pourquoi.

b) Si vous n'avez pas besoin de toutes les décimales, il est plus sage de tronquer les valeurs calculées et éviter ainsi de traîner des restes inutiles. Utilisez pour cela la fonction INT.

Exemple : Arrondi à 2 chiffres après la virgule

$X = INT(X * 100 + 5)/100$
si $X = 1{,}123$, il devient 1,12

si X = 1,127, il devient 1,13
si X = 1,125, il devient 1,13.

c) Méfiez-vous des tests de nullité d'une variable calculée. Choisissez un intervalle – autour de 0 – tel que vous pourrez considérer que tout nombre à l'intérieur de cet intervalle est nul.

Exemple : 100 IF (abs(X) < 0.0001) THEN X = 0 :.....

si X appartient à $[-10^{-4}, 10^{-4}]$ on l'assimile à zéro.

d) Essayez les fonctions standards disponibles sur votre machine, elles peuvent vous réserver des surprises. Voici un exemple simple mais très surprenant car la plupart des valeurs vérifient le test J< >I !

```
10 INPUT "Nombres de valeurs:";N
20 FOR I=1 to N
30 J=SQR(I*I):REM normalement J = I
40 IF (I<>J) THEN PRINT I
50 NEXT
60 END
```

e) Enfin, soyons raisonnables et ne demandons pas trop à notre micro-ordinateur. Rechercher 12 à 13 chiffres significatifs a rarement de l'intérêt. Laissons ces performances aux gros ordinateurs. Si, malgré tout, vous voulez absolument calculer (ou plutôt recalculer) les 100 ou 200 premières décimales de PI, vous devrez écrire une application spécifique et redéfinir les calculs élémentaires (+ , – , * , /...).
Voilà un bel exercice !

2. LES ERREURS D'APPROXIMATION

De par son codage, un nombre réel a en moyenne 7 chiffres significatifs (en simple précision). Dès que ce nombre est supérieur à 10^8 ou inférieur à 10^{-8}, il y a un risque notable de perdre la dernière décimale.
Lors de l'écriture d'un algorithme itératif (on boucle tant que le résultat souhaité n'est pas obtenu), la difficulté majeure une fois l'algorithme mis au point, est de savoir comment s'arrêter et surtout à quel moment. Examinons deux exemples :

A) Calcul de PI à l'aide d'une suite simple

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \frac{1}{6^2} \text{ .....à l'infini a pour limite } \frac{\pi^2}{6}$$

Comment tester la convergence, c'est-à-dire, quand doit-on arrêter le calcul ? La solution que nous proposons, calcule l'écart entre deux valeurs consécutives de la somme. Si cet écart (la variable DELTA) est inférieur à la limite fixée au début du programme, on sort de la boucle. En effet, plus on s'approche de la limite (si elle existe !), moins la somme varie.

```
 5 REM calcul d'une série de terme 1/(n*n)
10 INPUT "précision souhaitée:";P
20 SOMME=0
30 ANCIENNE=0
```

```
40 I=1:REM I est la variable de boucle
50 SOMME=SOMME+1/(I*I)
60 DELTA=ABS(SOMME-ANCIENNE)
70 IF (DELTA >= P) THEN ANCIENNE=SOMME:I=I+1:GOTO 50
80 PI=SQR(6*SOMME):PRINT PI
```

Estimation du nombre de termes à calculer
Pour $P = 10^{-5}$, à la 317e itération (I = 317), 1/(I * I) vaudra à peu près $9.952\ 10^{-6}$, donc l'écart entre la somme à l'étape 316 et celle à l'étape suivante sera inférieure à P : on arrête le programme (Sur APPLE II, pour $P = 10^{-9}$, la valeur calculée est proche de 3,14156).

Attention, P ne représente pas la précision du «résultat». Cette méthode permet seulement d'introduire un point d'arrêt dans le programme, elle peut ne pas être efficace du tout. L'exemple qui suit le montre bien.

B) Estimation d'une deuxième suite :

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} \ldots\ldots$$

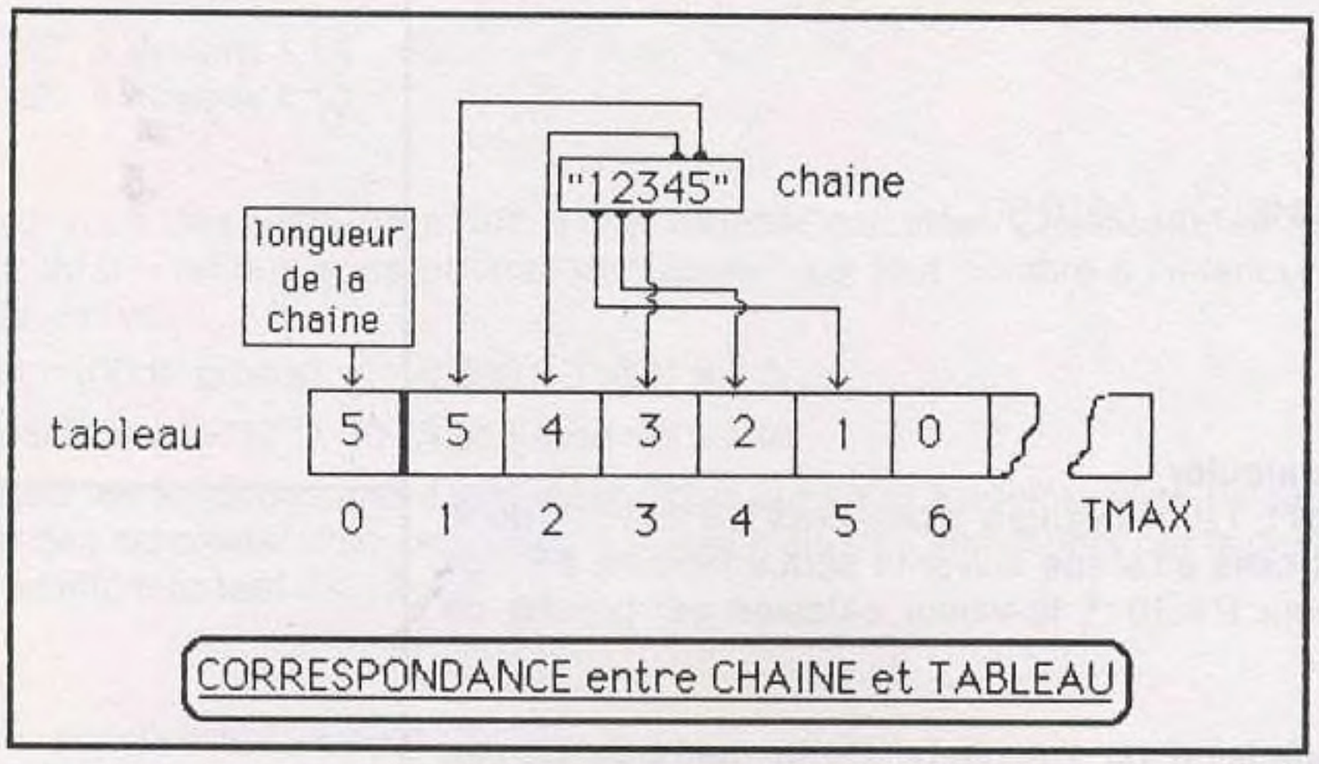
Reprenons le principe de test utilisé dans le programme du premier exemple. Pour une précision de 10^{-5}, il suffit de réitérer 10^5 fois. Le programme stoppe et nous offre un résultat... complètement faux, car cette suite (on peut le démontrer) ne tend pas vers une valeur finie. L'ordinateur se bloquera au moment où on atteindra ses limites de précision, la fraction 1/i sera arrondie automatique à zéro.

3. APPLICATIONS AUX DOMAINES DE GESTION

La gestion des comptes en banque, des facturations, de fiches de paie implique des calculs exacts sur les sommes manipulées. Les erreurs d'arrondis et/ou la grandeur des valeurs peuvent rendre la vie difficile à l'informaticien chargé de traiter ces données.
Nous avons déjà constaté qu'en général, sur un micro-ordinateur, les nombres entiers permettaient de représenter une valeur entre −32768 et 32767. Ils sont donc le plus souvent inadaptés aux applications de gestion (sauf peut-être pour vérifier le contenu de sa tirelire). Les variables réelles, même si elles autorisent le codage de très grands nombres, limitent le nombre de décimales significatives comme on vient de le constater. Ainsi, une valeur comme 1 234 567,15 F ne peut être codée
- ni en entier (nombre trop grand),
- ni en réel (pas assez de chiffres significatifs),
  on perdrait les centimes.

Comment manipuler des grands nombres :
Pour tous ceux qui n'ont ni DOUBLE PRECISION, ni ENTIER LONG (4 octets), nous présentons une idée de codage d'entiers de taille quelconque puisqu'ils sont rangés dans des tableaux, ainsi que des méthodes pour effectuer les calculs élémentaires sur ces tableaux :

Principe :
- Le codage s'effectue très simplement, 1 élément = 1 chiffre, la limite est fixée lors de la déclaration des tableaux



CORRESPONDANCE entre CHAINE et TABLEAU

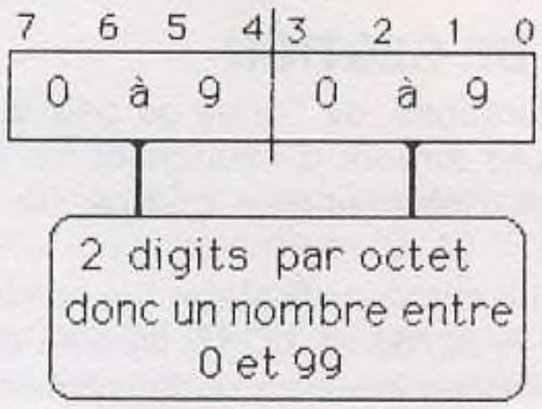
- La taille de l'entier, c'est-à-dire le nombre de chiffres nécessaires à sa représentation est contenu dans l'élément zéro du tableau.
- Les opérations font appel à des sous-programmes. Cette contrainte nous oblige à déclarer trois tableaux qui sont utilisés comme des variables internes. Dans le cas d'une application appelant ces routines, vous devez, avant chaque GOSUB, transférer vos tableaux dans T1 % et T2 %, puis au retour récupérer le résultat dans T3 %. Cette limitation est liée au BASIC qui ne permet pas la déclaration de variables locales au sein d'un sous-programme (sauf pour certaines versions très évoluées).

tableau des codes

Code binaire	digit
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
: :	:
1 0 0 1	9
1 0 1 0	N.A.
: :	
1 1 1 1	

(N.A. : non applicable)

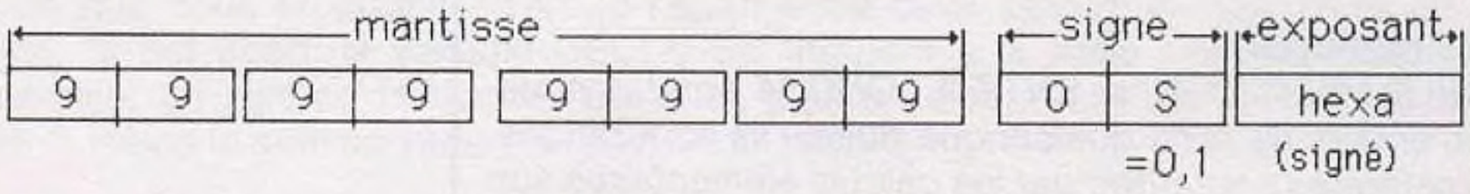
1 OCTET



EXEMPLE: 1234 devient sur 2 octets 0001|0010 0011|0100

(le format est très variable suivant les langages)

FORMAT d'un NOMBRE REEL en D.C.B.



limites : en valeur absolu, de 0.1 E-127 à 0.99999999 E+127

PRINCIPE du CODAGE D.C.B.

- L'addition et la soustraction suivent la méthode classique.
- Au contraire, le sous-programme de multiplication effectue l'addition des sous-totaux au fur et à mesure de leur calcul.

Remarque :

- Cette méthode de stockage est loin d'être optimale, on peut facilement l'améliorer en prenant deux chiffres par élément.
- Elle reprend le principe d'un mode appelé D.C.B. (Décimal Codé Binaire), de plus en plus utilisé sous BASIC (CBASIC de Digital Research, MS-BASIC sur MACINTOSH) ainsi que sous d'autres langages, pour la gestion des variables réels (pour ce qui nous concerne, nous nous limitons aux entiers). Son intérêt principal est de garantir que les conversions des constantes (qui sont des chaînes ASCII, cf. Led-Micro du mois de septembre) en codage binaire s'effectuent sans erreur d'arrondi. (La figure ci-dessous explique le format interne en mode D.C.B.).
- Il n'y a pas de contrôle des erreurs ni de possibilités d'utiliser des valeurs négatives, enfin il manque la division entière. Notamment, il est primordial pour une application complète de tester les dépassements de capacités au niveau de la déclaration des tableaux (sans quoi le BASIC vous répondra «Bad Subscript error» lors d'un accès à un élément non défini du tableau) mais aussi d'être capable de gérer des valeurs négatives (ne serait-ce que pour des découverts bancaires !).
- A vous d'enrichir le programme, ou de proposer d'autres solutions).

```
10 REM PRINCIPE DE CALCULS SUR DES ENTIERS DE GRANDE TAILLE
20 TMAX=20:REM on choisit sa taille désirée
30 DIM T1%(TMAX),T2%(TMAX),T3%(TMAX)
40 REM il faut connaitre toutes les variables nécessaires au programme
50 REM pour les déclarer par l'ordre DIM
60 REM
70 REM On lit les 2 valeurs de départ
80 INPUT"Entrez les deux nombres (n1,n2) :";T1$,T2$
90 REM on convertit les 2 chaines en tableau
95 L=LEN(T1$)
97  REM attention à l'inversion entre la lecture d'une chaine
98  REM et la lecture d'un nombre.
100 FOR i= 1 TO L
110  T1%(L-i+1)=ASC(MID$(T1$,i,1))-48:REM 48=code ascii de '0'
120 NEXT i
122 FOR i=L+1 TO TMAX
124 T1%(i)=0
126 NEXT i
130 T1%(0)=L
128 L=LEN(T2$)
140 FOR i=1 TO L
150  T2%(L-i+1)=ASC(MID$(T2$,i,1))-48
160 NEXT i
165 FOR i=L+1 TO TMAX
```

```
167 T2%(i)=0
168 NEXT
170 T2%(0)=L
180 GOSUB 1000:REM on effectue T1%+T2%
185 PRINT "addition:";:GOSUB 500:REM sortie du résultat
190 GOSUB 2000:REM  ..      ...      T1%-T2%
195 PRINT "soustraction:";:GOSUB 500
200 GOSUB 3000:REM  ..      ...      T1%*T2%
205 PRINT "multiplication:";: GOSUB 500
210 END
```

```
500 REM affichage de T3%
510 FOR L=T3%(0) TO 1 STEP -1 ..... faire attention à l'ordre
520 PRINT CHR$(T3%(L)+48);          d'écriture.
530 NEXT .....
```

```
1000 REM sous programme d'addition
1010 T=T1%(0):IF T< T2%(0) THEN T=T2%(0) ◄.....{Calcul du max
1015 retenue%=0
1020 FOR ii=1 TO T .....
1030 Temp%=T1%(ii)+T2%(ii)+retenue%          Boucle principale propageant
1040 retenue%=INT(Temp%                      la retenue (=0 ou 1);Le test
 /10):T3%(ii)=Temp%-10*retenue%              de sa valeur est implicite
1050 NEXT .....                              dans les calculs
1060 T3%(0)=T+retenue%:T3%(T+1)=retenue%
1070 FOR ii=T+2 TO TMAX .....
1080 T3%(ii)=0                               mise à 0 du reste du
1090 NEXT.....                               tableau
1100 RETURN
```

```
2000 REM sous programme de soustraction
2010 T=T1%(0):IF T< T2%(0) THEN T=T2%(0)◄.....{On prend le max des 2 longueurs
2015 retenue%=0
2020 FOR ii=1 TO T .....
2030 T3%(ii)=T1%(ii)-T2%(ii)-retenue%        {boucle principale propageant
2040 IF T3%(ii)<0 THEN                       {la retenue (= 0 ou 1)
retenue%=1:T3%(ii)=T3%(ii)+10
2050 NEXT .....
2060 ii=T                                    .....{On élimine les 0 en trop
2065 WHILE ( T3%(ii)=0 ):ii=ii-1:WEND ◄.....
2067  T3%(0)=ii:IF ii=0 THEN T3%(ii)=1 ◄.....{si le résultat est nul, il faut tout
                                             {de même garder un 0 pour l'affichage
2070 FOR ii=T+2 TO TMAX .....
2080 T3%(ii)=0                               mise à 0 des éléments restant
2090 NEXT .....
2100 RETURN
```

```
3000 REM multiplication
3005 FOR i=0 TO TMAX:T3%(i)=0:NEXT   <-- initialisation du tableau résultat
3010 FOR i=1 TO T2%(0)  ..... Boucle sur chaque terme on gère à la fois la multiplication et l'addition.
3020 retenue%=0
3030 FOR j=1 TO T1%(0)  ..... Calcul élémentaire chiffre par chiffre
3040 temp%=T3%(i+j-1)+T1%(j)*T2%(i)+retenue%
3050 retenue%=INT(temp%/10)
3060 T3%(i+j-1)=temp%-10*retenue%
3070 NEXT j  .....
3075 T3%(i+T1%(0))=retenue%
3078 NEXT i  .....
3080 L=T1%(0)+T2%(0)-1:T3%(0)=L
3090 IF retenue%<>0 THEN
  T3%(0)=L+1:T3%(L+1)=retenue%  ..... propagation de la retenue
3100 FOR i=T3%(0) TO TMAX :T3%(i)=0:NEXT
3110 RETURN
```

4. EXERCICES D'APPLICATION

Après avoir vu pendant trois cours des sources d'amélioration de vos programmes, il serait temps de les mettre en application. Dans cette optique, nous allons vous proposer une succession d'exercices plus ou moins détaillés dans leur analyse que vous aurez à programmer. Nous ne vous fournirons pas de solution idéale dans nos cours prochains, cependant nous souhaiterions recevoir les résultats de vos travaux personnels. En effet, cela nous permettra de voir ce qui a été assimilé sans problème et de définir les points sur lesquels des précisions seraient utiles : qui plus est, nous pourrons alors mieux adapter le rythme de nos cours à votre vitesse. Enfin, nous publierons les réponses les plus remarquables dans toute l'acception du terme, que ce soient les plus originales, les plus améliorables par le biais de nos préceptes ou les plus orthodoxes.

4.1. Exercice nº 1 : le calendrier

Après les pompiers, le facteur, les éboueurs et autres corps de métier, tout aussi sympathiques, nous vous offrons notre calendrier pour la nouvelle année. Toutefois, nous sommes légèrement moins courageux (c'est pour cela que nous avons choisi l'informatique : le meilleur informaticien est celui qui en fait le moins possible pour résoudre le problème posé) que nos devanciers, aussi vous ferez vous-même le calendrier.

4.1.1. Programme principal

Nous allons pour ce programme vous expliciter une suite de règles que nous pensons souhaitables de respecter lorsque l'on rédige le programme. Celles-ci risquent de vous paraître lourdes à mettre en œuvre ou à manipuler dans la mesure où ce qui est demandé paraît simple : un jour pourtant, vous passerez à des programmes plus étoffés et alors les automatismes acquis précédemment allègeront considérablement le travail à fournir.

Ce que nous appelons le «programme principal» consiste généralement en une simple suite d'appels de «procédures» ou sous-programmes. Il contient parfois la réception des choix entre différentes procédures. Par exemple, si un programme doit faire les quatre opérations, c'est dans le programme principal que vous choisiriez celle qu'il doit exécuter.

Il y a deux façons de ranger programme principal et procédures, l'une et l'autre issues de deux langages ou les sous-programmes sont «l'âme même» du programme. La première est de type FORTRAN.

```
10 GOSUB 1000
20 GOSUB 2000
30 GOSUB 3000
40 END
1000 REM ....................
   ..... INITIALISATION ....
   ..... DES PARAMETRES ..

1999 RETURN
2000 REM ....................
   ..... ETABLISSEMENT .....
   .....DU SEMAINIER...
2999 RETURN
3000 REM ....................
   ..... IMPRESSION .........
   .....DU RESULTAT...
3999 RETURN
```

La seconde s'apparente au PL/1 et à tous les langages provenant de la même philosophie (PASCAL, C, ADA, etc.).

```
10 GOTO 10000
1000 REM ....................
   ..... INITIALISATION ....
   ..... DES PARAMETRES ..



1999 RETURN
2000 REM ....................
   ..... ETABLISSEMENT .....
   .....DU SEMAINIER...
2999 RETURN
3000 REM ....................
   ..... IMPRESSION .........
   .....DU RESULTAT...
3999 RETURN
10000 REM ..PROGRAMME PRINCIPAL
   ....................
10010 GOSUB 1000
10020 GOSUB 2000
10030 GOSUB 3000
10040 END
```

4.1.2. Initialisation des paramètres

Le premier «module» qu'il convient de faire, doit servir à déterminer les valeurs initiales nécessaires à la bonne marche du programme. C'est aussi à cet endroit que l'on

interroge l'usager pour obtenir les renseignements propres à son utilisation personnelle du programme ; dans le cas qui nous intéresse, il s'agit essentiellement de l'année dont on veut obtenir le calendrier.

4.1.3. Etablissement du semainier

Ce programme étant le premier sera exagérément décortiqué ; on peut même dire que nous faisons pour vous l'essentiel de l'analyse en vous proposant la solution théoriquement la plus logique. Cela ne vous empêche pas de découvrir des astuces personnelles se substituant avantageusement à nos suggestions et nous serions alors très heureux que vous nous en faisiez part.

Ce module cherche à déterminer quel jour de la semaine «tombe» le 1er janvier de l'année qui nous intéresse. Pour cela, nous aurons dans le module précédent enregistré que le 1er janvier 1900 était un lundi. Nous utilisons le fait que le 1er janvier tombe le jour suivant de l'année précédente, hormis les années bissextiles ou, alors, la différence est de 2 jours. Le premier outil dont nous nous servons est la fonction «modulo» (elle est implantée dans certains BASICs, à vous de le vérifier). A quoi est-elle utile ?

Cette fonction calcule le reste entier de la division d'un entier par un autre entier.

27 modulo 4 = 3 car 27 = 4 * 6 + 3
35 modulo 7 = 0 35 = 7 * 5 + 0

En l'occurrence, c'est le modulo 7 qui nous importe, le seul détail auquel il faille prêter attention étant que le dimanche ne «vaut» pas 7 mais 0 selon les conventions de cette opération. Si cette fonction n'existe pas dans l'environnement BASIC où vous évoluez, on peut très bien la définir par l'instruction suivante :

```
1000 REM  X MODULO Y
1010 I% = X:J% = Y
1030 MOD% = I% - ((I% / J%) * J%)

1050 RETURN
```

On peut inclure dans ce module ou en faire un module indépendant, le calcul du nombre de jours assigné à chacun des mois de l'année. Si la paresse vous gagne à ce moment, est-il besoin de vous rappeler que vous avez vu le mini-programme que constitue à lui seul ce problème (comme d'ailleurs tout module) entièrement écrit dans le numéro précédent.

4.1.4. Impression du résultat

Dans ce sous-programme se trouveront tous les ordres d'impressions mais aussi les tests indispensables pour faire des retours à la ligne judicieux (à la fin d'une semaine) ou des sauts de ligne de la même espèce (à la fin du mois). Voici pour l'exemple un mois de janvier très sommaire auprès duquel toute amélioration est la bienvenue.

LUN	MAR	MER	JEU	VEN	SAM	DIM
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Et maintenant, place à la programmation, vous avez jusqu'au 31 décembre 1985. Après, il devra être au point.

4.2. Le jeu de Marienbad

Avant d'envisager la programmation, il est pour le moins nécessaire de connaître les règles de ce jeu.
Le principe est le suivant : vous êtes en présence de 15 allumettes ou autres objets à votre convenance et chacun votre tour, votre adversaire et vous-même devez prendre entre une et trois allumettes sans possibilité de passer votre tour. Le perdant est le joueur tenu de prendre la dernière allumette.

4.2.1. La stratégie

Nous n'allons pas nous contenter dans ce jeu d'utiliser l'ordinateur comme un simple terrain de jeu où il ne serait qu'un spectateur privilégié à qui l'on permettrait de déplacer les pièces à notre place. Nous pouvons lui permettre de jouer ; pour cela, celui-ci a besoin d'une stratégie que l'on concevra la plus efficace possible, afin qu'il puisse, dans la mesure du possible, remporter la victoire. Dans le cas qui nous intéresse, nous partirons de ce qui entraîne immanquablement le gain de la partie pour revenir aux coups initiaux et décomposer ainsi le processus menant à la réussite.
Il faut que ce soit à votre adversaire de jouer lorsqu'il ne reste plus qu'une seule allumette : c'est l'évidence même. Observons ce qui se passe s'il reste cinq allumettes et que c'est également à votre adversaire de jouer :

- Il prend une seule allumette. Il en reste donc 4 et la main est à vous. Si vous en prenez 3 – ce qui vous est permis – il en restera une et la victoire vous sera acquise.
- Il s'empare de 2 allumettes. Le reste est donc de 3. Prenons-en deux et c'est bon.
- Il enlève 3 allumettes du tas. Sur les 2 restantes, vous n'en prélevez qu'une seule.

Vous constatez bien que «5 allumettes et à votre adversaire de jouer» est une position tout aussi sûre que 1 seule allumette. Vous observeriez de même que 9 est imparable ; en conséquence, on entrevoit un nouvel usage de la fonction MODULO présentée au paragraphe précédent. Le seul et unique objet de préoccupation sera le reste de la division par 4 du nombre d'allumettes demeurant sur le tapis. Toutes les sommes vérifiant cette propriété sont autant de positions inattaquables.

4.3. Les carrés magiques

Le sujet est en lui-même très simple : il s'agit de construire un carré d'un type particulier. En effet, il sera constitué des N premiers entiers tels que la somme de chaque ligne, de chaque colonne et des deux diagonales soit la même. Voici des solutions possibles pour les carrés 3×3 et 4×4. Dans le premier cas, la somme est toujours 15, dans le second, nous trouvons 34.

2	9	4
7	5	3
6	1	8

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

Ce sujet-là est donné brutalement et se trouve être d'une difficulté non négligeable, il convient notamment de réfléchir à substituer aux boucles imbriquées des solutions plus subtiles gagnant du temps dans des proportions importantes.

4.4. Calcul de N décimales de PI

Comme nous vous l'avions laissé entendre, il y a deux numéros, un de nos sujets de «méditation» va être le calcul d'autant de décimales de PI que souhaité. Cela nécessite quelques connaissances mathématiques dont nous ne retiendrons que l'aspect pratique utile à notre problème (tout comme le conducteur n'a sur la thermodynamique que la notion de dilatation des gaz lors de l'explosion). La méthode employée fait appel aux développements limites et plus particulièrement à celui d'arc-tangente. C'est une fonction qui existe dans la plupart des ordinateurs du commerce, malheureusement pas avec la précision désirée, ce qui, vous le comprendrez bientôt, serait impossible. Le développement limite s'énonce :

$$\text{Arctg}(X) = X - 1/3 * (X)^3 + 1/5 * (X)^5 - 1/7 * (X)^7 + \ldots$$

Ceci étant vrai lorsque X < 1. Vous remarquez alors que si l'on pousse la puissance de X assez loin, X sera très proche de 0 et donc inférieur à la précision recherchée pour Arctg quelle que soit l'erreur tolérée. C'est d'ailleurs comme cela que procède votre ordinateur pour établir Arctangente lorsque cette fonction fait partie de votre BASIC. Découvrons donc la formule de John Machin que nous nous garderons bien de démontrer :

$$= 16 * \text{Arctg}(1/5) - 4 * \text{Arctg}(1/239)$$

Vous notez que les deux Arctangentes à calculer sont des nombres relativement proches de 0 dont les puissances seront très vite infinitésimales.

4.4.2. Les réels codes en tableaux d'entiers

La principale astuce qui nous autorise à calculer Pi avec n'importe quelle précision est de substituer à une variable réelle un tableau d'entiers où chaque élément ne représentera qu'une infime partie (en général 3 chiffres) du nombre ainsi codé. Par exemple, 1/3 s'exprime respectivement :

en réel 0,333333333333 dans les meilleurs cas
en tableau 0 333 333 333 333 333 333 333 333 333 et plus si l'on veut

quant à Pi, comparez 3,14159265358 et
3 141 592 589 793 462 643 383 279, arrêtons-là

Maintenant, le principal problème est de savoir effectuer des opérations sur ces tableaux. La panacée porte un nom que certains d'entre vous auront deviné : la boucle FOR, mais encore faut-il l'employer à bon escient. En effet, les simples quatre opérations sont déjà sources de mille maux. L'enfance de l'art que constitue l'addition présente l'écueil de ne pas égarer la retenue en cours de route.

```
1000 REM  ADDITION C=A+B
1005 R% = 0
1010  FOR I = N TO 1 STEP  - 1
1020 J = A(I) + B(I) + R%:R% = J /
   1000:C(I) = JMOD1000
1030  NEXT I
```

Nous signalons à votre attention que la boucle est décroissante. Pour la soustraction, le problème ne soulève pas de nouvelles difficultés majeures.

```
1000   REM   SOUSTRACTION C=A-B
1005 R% = 0
1010   IF A(1) < B(1) THEN  PRINT
      " IMPOSSIBLE"
1020   FOR I = 1 TO N
1030   IF A(I) > B(I) THEN 1060
1040 C(I - 1) = C(I - 1) - 1:A(I)
       = A(I) + 1000
1060 C(I) = A(I) - B(I)
1070   NEXT I
```

La multiplication et la division par un entier peut se résoudre aisément à l'aide d'une ou deux variables auxiliaires contenant soit la retenue soit la division par un entier peut se résoudre aisément à l'aide d'une ou deux variables auxiliaire contenant soit la retenue soit le reste.

Programme multiplication

```
1005 R% = 0
1010   FOR I = N TO 1 STEP  - 1
1020 J = A(I) * M + R%
       :R% = J / 1000
       :C(I) = JMOD1000
1030   NEXT I
```

Programme division

```
1020   FOR I = 1 TO N
1030 A(I) = A(I) + R% * 1000
1050 S% = A(I) / D
      :R% = A(I) - S% * D
      :C(I) = S%
1070   NEXT I
```

Mais la multiplication d'un tableau par un autre est excessivement lourde et fort longue, quant à la division, elle n'est pas envisageable directement.

Nous allons maintenant adapter les outils à l'application pratique. Pour établir l'Arctangente, nous aurons besoin de deux tableaux du type explicite : le premier contiendra les résultats des premiers éléments de la somme constituant le développement limite d'Arctg, le second représentera la puissance nième de X. En décomposant les opérations nécessaires pour ajouter un nouveau terme au développement limite, nous maîtriserons mieux le mécanisme.

Avant cette opération, nous avons :
– A (i) le développement limite des N – 1 premiers termes
– X (i) la valeur de X à la puissance 2 * N – 3.
ceci étant les deux tableaux.
– une variable L où se trouve l'entier impair par lequel on divise X.

Nous devons donc faire le produit de X (i) par le carré de X, implanter le résultat dans X (i) diviser le tout par L et additionner ou soustraire l'ensemble selon le signe de la fois précédente.

Il convient de remarquer que très rapidement un nombre sans cesse croissant d'éléments du tableau X (i) représentera des 0. Une modification judicieuse des bornes de la boucle pourra éviter des calculs aux solutions aussi connues qu'inutiles.

Nous ne vous cacherons pas que ce programme est pour le moins ardu, il ne sera pas réussi par tout le monde d'ici le mois prochain et les heureux y auront transpiré avant de parvenir à leurs fins. Aussi, toute solution, même partielle, nous intéresse afin de préciser la «foultitude» de questions qui auront pu vous bloquer. Bon courage !

A titre indicatif, voici ce que tout Parisien peut aller vérifier au Palais de la Découverte, pour nos amis provinciaux :

Pi#3,14159,26535,89793,23846,26433,83279,50288,41971,69399,37510,58209,74944

5. CORRECTION DES COURS PRECEDENTS

Nous espérons que la majeure partie d'entre vous n'a pas été trop désorientée par les coquilles disséminées de manière maligne dans les deux précédents articles (et rien ne nous assure que nous n'ayons encore succombé à notre péché mignon, ce mois-ci !). Néanmoins, nous nous devons de rectifier ces «bavures» et faire en sorte qu'elles disparaissent définitivement. Mais comme dit le milliardaire de «Some live it hot» : «Nobody's Perfect».

5.1.1. Cours de BASIC du numéro 21 (1)

Page 8 : comme vous avez pu le remarquer, il y a 4 fois l'instruction 115. Eh bien non ! dès la seconde fois, il s'agit de 125 GOTO 200 puis 135 idem... et 145.

Page 11 : ici, c'est un tableau bidimensionnel que l'on manipule avec un seul paramètre, ce qui est pour le moins fâcheux, surtout à deux reprises (en réalité, c'est la même ligne qui a été recopiée). Il convient donc d'écrire en lieux et places :

```
2120 if info$(i,1) = nom$ then j=j+1 : stock(j)=i
3120 if info$(i,j) = nom$ then j=j+1 : stock(j)=i
```

5.1.2. Cours de BASIC du numéro 22

Pages 9, 10, 11 : ce listing est, à lui seul, un petit chef-d'œuvre !

ligne 20 : un GOSUB dont on ne verra jamais le retour, il fallait 20 GOTO 10000

ligne 450 : il manque un deux points « : » entre NEXT et GOTO 480

ligne 470 : un seul NEXT pour deux boucles, c'est mesquin ! donc NEXT K et 475 NEXT I

ligne 500 : pour les perfectionnistes (dont notre cher monsieur Bourdes de Suresnes qui n'avait pas tout corrigé cependant) ce n'est pas numéro mais référence.

ligne 10330 : il ne faut pas retourner en 10000 ce qui aurait des conséquences fâcheuses sur le dimensionnement mais plutôt en 10100.

Page 11 : connaissez-vous la célèbre instruction «GITI», avouez que non ; pourtant, il suffisait d'observer votre clavier et à côté du «I», que voyez-vous ? Un «O». Nous retrouvons alors du connu et même archi-connu, le «GOTO». Elémentaire, n'est-ce pas ?

5.2.1. Correction du Cours Approfondi du numéro 21

Page 19 : pauvres de vous, tout fiers d'avoir compris les arcanes du codage binaire des entiers négatifs, vous voilà face à −32767 et horreur et stupéfaction, cela ne correspond nullement à ce que votre bon sens lié à vos nouvelles connaissances vous laissait deviner. Pour les indulgents sûrs de leur coup, c'était une erreur des auteurs, mais pour les plus confiants dans notre «infaillibilité» que d'inquiétude. Pour la dissiper, il suffisait de tourner la page, mais quelle décision difficile à prendre quand on pense ne pas avoir compris. Tout rentrait alors dans l'ordre avec la figure 1.1.2.2.

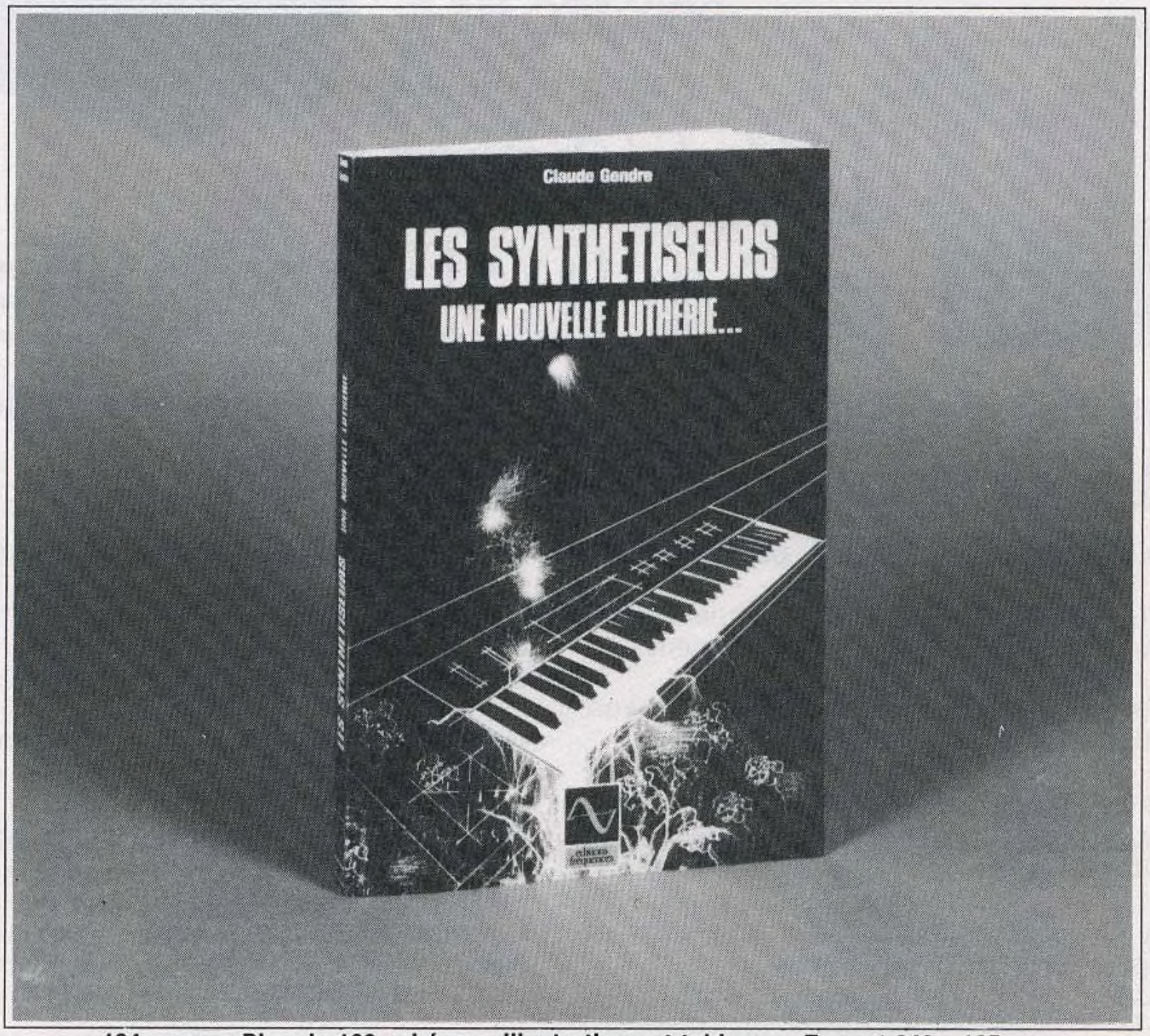
5.2.2. Correction du Cours Approfondi du numéro 22

Tout arrive et à ce jour, nous n'avons ni vous chers lecteurs ni nous-mêmes découvert d'erreur dans ce cours. Etonnant, non !

vient de paraître

collection "études"

LES SYNTHÉTISEURS UNE NOUVELLE LUTHERIE...



184 pages - Plus de 160 schémas, illustrations et tableaux - Format 240 x 165.

Le synthétiseur est certainement un appareil très critiqué, très mal connu et pourtant très admiré par les jeunes (et les moins jeunes...) passionnés de musique. Instrument privilégié du 20e siècle, il existe peu de littérature le concernant.

«Les synthétiseurs, une nouvelle lutherie...» de Claude Gendre, troisième volume de cet auteur paru dans la collection «Etudes», est le premier livre de cette importance qui lui est consacré. Il est donc indispensable à tous ceux qui veulent connaître et bien utiliser cet instrument, qu'ils soient étudiants, formateurs, amateurs de techniques nouvelles, revendeurs de matériel ou, bien sûr, mélomanes !

Accessible sans connaissances scientifiques particulières, cet ouvrage débute par l'histoire de l'orgue et des instruments pour se terminer par l'amplification des claviers en passant par la formation des sons et les différentes techniques actuelles : synthèse analogique, synthèse numérique, modulation de fréquence (Yamaha), distorsion de phase (Casio), système MEG (Multiple Event Generator) du français Christian Deforeit (Hohner).
On trouvera, en particulier, les caractéristiques du futur synthétiseur Hohner 8 D dont le prototype n'a pas encore été présenté mais qui préfigure l'avenir. Enfin, des renseignements pratiques, un lexique des termes spécialisés et les adresses des principaux fabricants et importateurs de matériel complètent cette véritable encyclopédie dont il n'existe pas, actuellement, d'équivalent en librairie.

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

Je désire recevoir l'ouvrage «Les Synthétiseurs» ☐ au prix de **155 F** (140 F + 15 F frais de port).

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

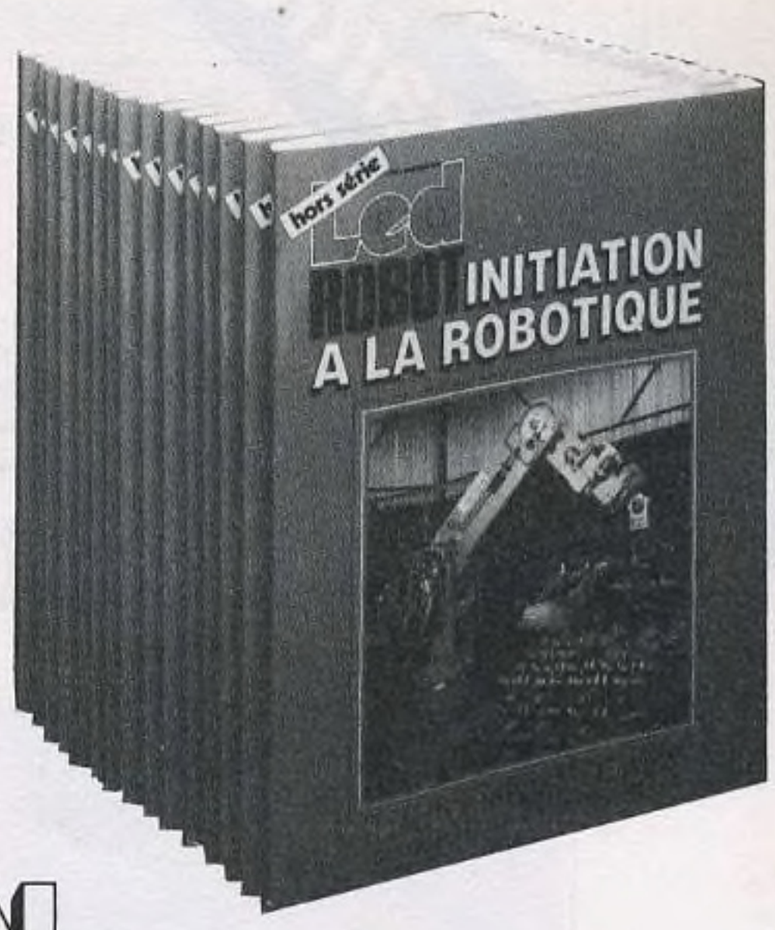
Nom ..... Prénom .....

Adresse .....

Code postal ..... Localité .....

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.

VU AU SICOB

Ce trente-sixième Sicob laisse une impression confuse. Est-ce à cause de l'éloignement du Sicob Boutique, de quelques absences remarquées, ou encore l'éparpillement des stands informatiques dans les différents étages ? Néanmoins, il y eut quelques suprises, et surtout beaucoup de promesses.

TOUT IRA MIEUX DEMAIN !

Cette année fut une année de transition, de l'aveu même des organisateurs. Première année du Sicob Boutique au Palais des Congrès qui n'avait plus rien de commun avec l'ancienne boutique, mais aussi dernière année de présence du mobilier de bureau au CNIT. Malheureusement, cette année encore, le manque de place a posé de sérieux problèmes aux organisateurs. Difficile d'obtenir un stand à la dimension de ses ambitions, ou plus modestement une place bien située, aussi nombreux furent les stands coincés près des issues de secours ou installés devant les baies vitrées.

C'est pourquoi, par exemple, Digital, le deuxième mondial derrière IBM, a préféré organiser sa propre manifestation à Cannes quinze jours avant le Sicob. Parallèlement, on a pu assiter à la multiplication de présentations, de cocktails, et autres réunions de ce style pendant la période du Sicob. On peut aussi citer le cas de Hewlett-Packard qui a profité d'une place importante dans le centre commercial de La Défense.

BEAUCOUP DE PROMESSES...

Nous sommes désormais habitués aux affirmations des commerciaux : «la nouvelle imprimante XXX est annoncée pour la fin de l'année...» ou encore «le dernier ordinateur de YYY sortira en France début 86...» qui finissent par nuire à la bonne image de certains

constructeurs. Le plus souvent, le problème de «francisation» de leurs produits est à l'origine des retards. Presque tous les constructeurs de micros bas de gamme sont concernés (Atari, Commodore, Sinclair). Ce n'est pas une critique délibérée, mais une amère constatation ! Que penser d'un produit qui, enfin commercialisé en France, est rendu obsolète par l'apparition sur le marché américain d'un nouveau modèle. (Citons Atari et son modèle 800 vite remplacé par le 800XL, Commodore et le C64 qui prit la place du Vic 20, et enfin le Sinclair QL annoncé et vendu en Angleterre un an avant son apparition en France).

Mais revenons au Sicob 85 et aux nouveautés du moment : on a pu voir le nouvel Atari. Il ne fallait pas demander trop de précisions ni être exigeant sur la qualité des démonstrations (les logiciels n'étaient pas encore au point), le matériel est annoncé, disponible sur commande, la liste des logiciels prévus est imposante mais n'apporte aucune garantie sur les délais de disponibilité. Enfin, Atari ne compterait-il pas sur les premiers acheteurs pour «debugger» sa machine ?).

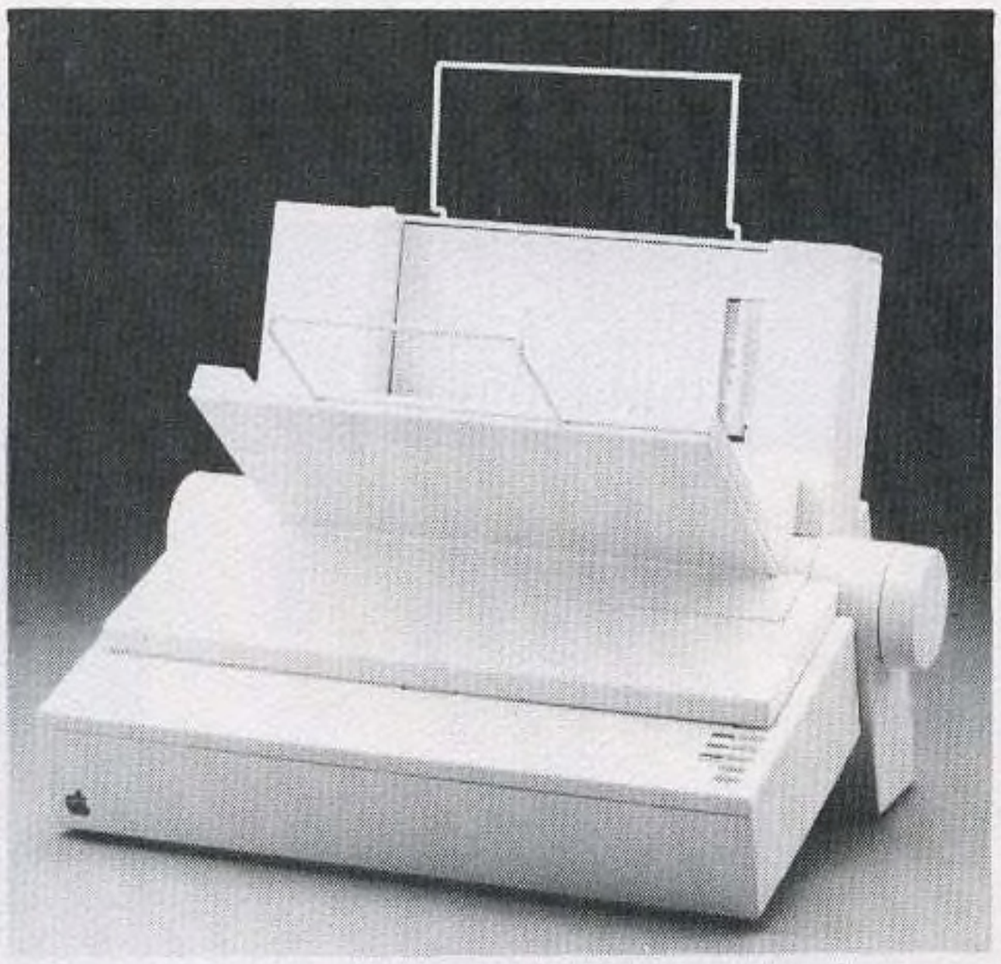
La vedette du stand Commodore était le modèle C 128 plus performant que son aîné tout en restant compatible. Malheureusement, la vraie nouveauté du constructeur (le modèle Amiga, un puissant ordinateur dans la lignée Macintosh) n'était pas exposé. Il est resté caché durant tout le Sicob, à l'abri des regards indiscrets et seuls quelques journalistes ont pu assister à une démonstration de ses possibilités graphiques. Par contre, pas de date officielle de commercialisation ni de prix (aux U.S.A. on parle de 1 200 dollars avec 256 K de mémoire et un moniteur couleur). Tout le problème est de savoir s'il ne restera pas un exercice de style, tant il est difficile de situer le marché qu'il vise !

Du côté français, Thomson a présenté au public le nouvel ordinateur TO9 haut de gamme de la série MO5, TO7/70. La firme française n'a toutefois pas attendu le Sicob, préférant organiser une soirée spéciale au Palais de la Découverte. Tout y était : la foule (manifestement plus nombreuse que prévue !), les médias (France Inter) et le mystère... (Impossible d'entrevoir les TO9 couverts de draps jusqu'au dernier moment !) Que dire de plus, sinon que la montagne a accouché d'une souris (même si elle est à la mode). Le TO9 étant compatible avec ses aînés, il hérite d'une bibliothèque de programmes très riche, cette compatibilité est en grande partie due à l'utilisation du même processeur 8 bits Motorola 6809 qui explique la performance modeste des logiciels présentés par rapport à ceux disponibles sur un Macintosh par exemple. Ne faut-il pas en conclure, quoi qu'en disent les responsables de Thomson Micro-Informatique, que le TO9 sera peu compétitif sur le marché de l'informatique semi-professionnelle ?

L'essoufflement du marché de la micro-informatique familiale explique en grande partie l'apparition de cette nouvelle vague de micro-ordinateurs plus puissants. Le standard MSX tente de se relancer avec le nouveau MSX2 par l'amélioration des possibilités techniques au niveau du graphisme et l'augmentation de la mémoire disponible. De plus en plus, les microprocesseurs 16 bits remplacent les 8 bits et les capacités mémoire sont multipliées par 4 à 8, sans que le prix de revient des composants change. Il faut voir dans cette évolution du matériel plus que le simple progrès technologique. Pour éviter la saturation du marché, la micro-informatique cherche à vendre un produit plus performant, utilisable aussi bien à la maison que dans des applications professionnelles. Ce créneau, qui est déjà une réalité aux Etats-unis (avec notamment l'IBM PC ou compatible ou encore le Macintosh), reste encore à l'état embryonnaire en Europe et particulièrement en France. Au-delà des différences de mentalité et même si les logiciels sont de plus en plus nombreux, performants et de mieux en mieux adaptés aux problèmes posés, les ordinateurs comme le Macintosh sont encore trop chers (entre 25 et 30 000 F pour une version de base), pour prendre la place de la calculatrice dans le travail quotidien du cadre moyen, de l'ingénieur ou a fortiori pour remplacer la machine à écrire de la secrétaire (il ne faut pas croire qu'elles disposent toutes d'une machine à traitement de texte !)

... ET QUELQUES SURPRISES !

Tout d'abord chez Apple qui annonce un disque dur 20M dont le prix serait tout à fait compétitif (une fois n'est pas coutume), une nouvelle imprimante imagewriter II plus performante que le premier modèle. Pour



l'Apple 2, deux nouveautés importantes avec un lecteur 3'' 5 double face de 800 K (le comble alors qu'on attend toujours les lecteurs double face pour le Mac !) et une carte d'extension jusqu'à 1 Moctet.
Mais la grande vague de la micro professionelle est celle de l'IBM AT et des compatibles présentés ou annoncés par de nombreux constructeurs. Il faut noter que Goupil annonce son compatible pour la rentrée : avec le nouveau G4 (compatible PC mais beaucoup plus puissant) cette société semble bien partie pour obtenir une part du gateau (même si certains bruits rapportent que seule l'étiquette est française). On pourrait citer d'autres petites sociétés françaises qui présentent des produits compatibles et bon marché.
Enfin, bien que la grosse informatique ne soit pas à notre portée, la C.A.O. (Conception Assistée par Ordinateur) et essentiellement ses applications graphiques ont été le pôle d'attraction principal de ce Sicob. C'est la grande bagarre entre les grands constructeurs et les grands de la C.A.O. tant sur le plan des logiciels que sur celui des postes de travail. Les domaines d'applications sont vastes (de la conception de séquences de films d'animations à la commande numérique pour la fabrication de pièces) et la puissance des postes peut atteindre celle d'un gros mini (beaucoup de ces machines utilisent un micro de la famille 68000, 68010, voire 68020). Parallèlement, la C.A.O. envahit le marché de la micro dans presque tous les domaines (beaucoup d'applications sur les PC et AT).

L'EVOLUTION DU PERIPHERIQUE

Enfin les périphériques sont à la hauteur des performances des micro-ordinateurs. Ce marché, si longtemps peu dynamique, est aujourd'hui en pleine expansion. Parlons en premier lieu des périphériques lourds, c'est-à-dire des mémoires de masse ; on peut noter l'avènement des disques durs (Atari proposera pour le 520 ST un disque dur 10 Mo pour environ 7 000 F), le développement des disques optiques dont l'interfaçage avec les micros est réalisé (Atari) ou à l'étude chez de nombreux constructeurs (Apple, Digital, Sony, Thomson...) tant pour une utilisation en mémoire morte dans des applications traitant des bases documentaires (dans le format du disque laser audio, 600 Mo), que dans des archivages d'images (disques à lecture et écriture utilisant différentes technologies sous de nombreux formats), la systématisation de l'emploi des disques 3''5 pour les micro-ordinateurs (qui est devenu le standard de facto).
Mais il ne faudrait surtout pas oublier le marché des périphériques d'impression qui est marqué par une nette amélioration du rapport qualité + performance + vitesse/prix : l'impression laser devient abordable pour les petites sociétés (de 30 à 70 000 F en moyenne dans le cas des imprimantes laser type Canon qui débitent deux à huit pages/minute). Les imprimantes à aiguilles cumulent dorénavant trois qualités d'impression : qualité brouillon à haute vitesse (250 à 350 caractères par seconde), qualité standard ou graphique (100 à 200 cps) et enfin une qualité courrier (50 à 80 cps) leur permettant de rivaliser avec les imprimantes à marguerite.
Il n'est pas rare qu'en prime, elles offrent la couleur et possèdent un tampon mémoire soulageant l'ordinateur de l'attente de fin d'impression.
Terminons en précisant que les amoureux de camemberts, hystogrammes et autres tracés de courbes ont maintenant à leur disposition des traceurs petits formats plus souples d'emploi que leurs aînés et dont les prix concurrencent les imprimantes classiques (on pouvait admirer chez Rotring une unité autonome possédant un bras articulé et un clavier autorisant l'impression de caractères et de graphiques sur n'importe quel papier.)

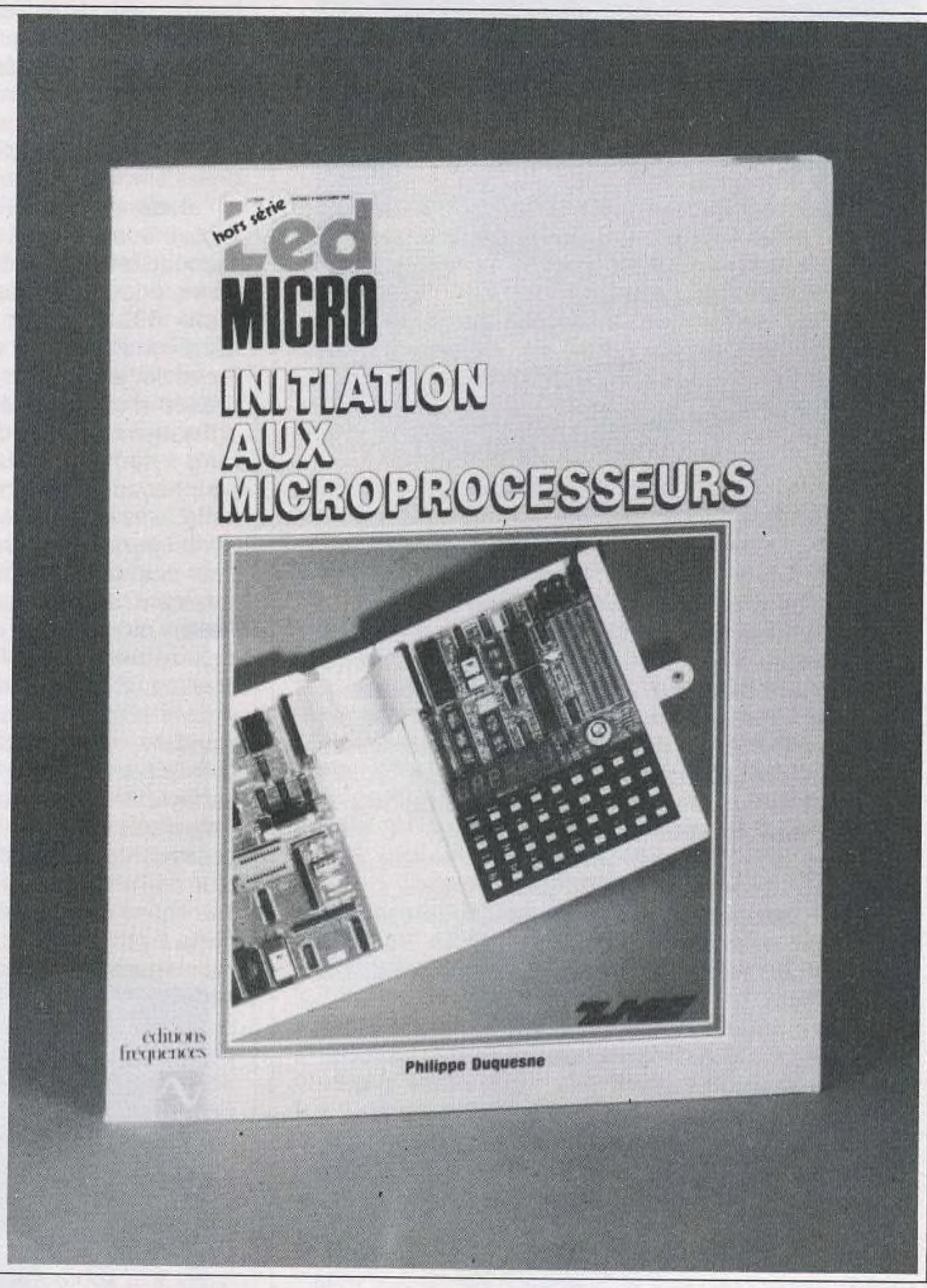
L'ERE DU CONVIVIAL EST NEE !

L'utilisateur non spécialisé a enfin à sa disposition des logiciels faciles d'emploi grâce aux menus déroulants, à la souris, aux fenêtres et aux icônes. Que ce soit sur le PC ou l'Atari 520 ST (grâce à GEM de Digital Research), le Mac, l'Amiga, le TO9 ou même l'Apple 2, nombreuses sont ou seront les applications usant et abusant de ces nouvelles facilités que l'environnement Macintosh nous a fait découvrir, tout cela pour le confort de l'utilisateur et le plaisir de l'œil.

Microprocesseurs
un cours essentiellement pratique !

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

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



En vente chez votre libraire et aux Editions Fréquences

Bon de commande

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

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

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

COURS DE GENIE LOGICIEL

De la théorie à la pratique

Charles-Henry Delaleu

LES CRITERES DE QUALITE

Les critères de qualité sont nombreux. Il est possible de les diviser en deux grandes familles :

- les critères de qualité propres à l'application
- les critères de qualité propres au déroulement du programme

Il est impossible de parler de critères de qualité sans mentionner les termes de :

- test
- contre-mesure
- évaluation
- sécurité

Dans la première partie, nous avons abordé le Génie Logiciel d'une manière générale, dans la seconde, nous avons étudié l'algorithme et l'analyse. Dans ce chapitre, nous ferons la connaissance des techniques liées aux critères de qualité et à leur environnement.

Il est impossible de concevoir un programme sans avoir un regard extrêmement vigilant à ce genre de problème. Aucune personne n'accepte aujourd'hui de conduire une automobile peu sûre. Alors, que penser d'un programme qui donne des résultats peu fiables et qui s'arrête au moindre incident technique ou humain. Que penser d'un logiciel qui bloque subitement son exécution après un long travail parce que l'imprimante n'est pas branchée, ou, plus simplement, par la saisie d'une information alphanumérique à la place d'une information numérique.

Vu de l'utilisateur, le comportement d'un programme aux aléas et anomalies est vital. Vu du concepteur, chaque mauvaise utilisation, chaque panne possible doit être prévue. Lorsque le programmeur réalise une nouvelle application, il doit penser qu'il ne sera pas l'utilisateur final. Il doit anticiper un comportement de la machine et de l'utilisateur en cas de négligence, de troubles.

CRITERES DE QUALITE

Les critères de qualité peuvent être divisés en deux grandes familles, soit :

- Les critères de qualité d'une analyse
- Les critères de qualité d'un programme

La nuance entre ces deux concepts vient du fait que les premiers critères sont liés à l'étude de l'application que l'on désire créer. La seconde se rapporte au résultat de l'application et au résultat de son application.
Les critères de qualité d'un programme sont développés dans les différentes fiches techniques qui suivent. Nous ne détaillerons ici que les critères de qualité d'une analyse.

LES CRITERES DE QUALITE D'UNE ANALYSE

Les principaux critères de qualité sont :

- Règle de faisabilité
- Règle d'évolutivité
- Règle de découpage fonctionnel
- Règle de généralité
- Règle de fiabilité
- Règle de planification
- Règle de compréhensibilité
- Règle de sécurité

Règle de faisabilité

L'une des phrases les plus utilisées dans le jargon de l'informatique est :

«Il n'y a qu'à»

A force d'employer ce terme, on fini par oublier qu'un calculateur a des limites. Que la moindre étude, la moindre écriture, le moindre test prend du temps et coûte souvent des frais divers.
Il va sans dire que le premier critère sera que l'analyse soit faisable. Un algorithme de résolution applicable doit répondre à un problème posé en termes précis.
Lorsqu'une nouvelle application est mise en route au niveau de l'étude, il convient d'être vigilant au fait que la plupart des problèmes à résoudre évolue avec le temps. De même, de nouvelles applications sont toujours demandées par les utilisateurs.

Règle d'évolutivité

Nous venons d'exprimer la volonté des utilisateurs à vouloir toujours de nouvelles applications à un programme qui leur est fourni. Il est donc très important qu'un programme puisse évoluer dans le temps. Pour cela, toute modification devra être possible, toute adjonction de sous-programmes autorisée. Chaque modification ne devra pas mettre le programme en danger, de même cette dernière devra toujours être réalisable.
Il convient donc :

- De bien analyser chaque problème
- De bien organiser son application
- De bien structurer son programme

NOTES PERSONNELLES

- De bien séparer chaque fonction
- De bien rester cohérent
- De bien rester homogène aux étapes de l'analyse.

Règle de découpage fonctionnel
Afin de bien répondre aux règles ci-dessus nommées, l'analyse doit procéder par découpage fonctionnel de fonction en sous-fonction.

Règle de fiabilité
La fiabilité de l'analyse doit être vérifiée pour chacune des fonctions. Chaque vérification doit avoir lieu le plus tôt possible pour chaque étape. Une erreur est beaucoup plus facile à détecter si elle est prise rapidement.

Règle de planification
Toute analyse doit être planifiée. Cela doit être pris au sens large du terme. En effet, il convient de :

- Planifier son travail dans le temps
- Planifier la réalisation des tâches les unes en fonction des autres

Il est absolument nécessaire de suivre un ordre chronologique dans l'étude et la réalisation d'un programme. Le programme principal se fait avant les sous-programmes. Les menus sont construits avant les tâches, etc...

Règle de compréhensibilité
L'analyse doit être naturelle, il n'est pas question de réaliser des acrobaties diverses afin d'arriver au résultat désiré. De plus, l'auto-documentation prend ici toute sa signification. En effet, chaque sous-ensemble doit être très documenté. Cette documentation doit se faire à deux niveaux :

- au niveau listing du programme
- au niveau de l'utilisateur du programme

Listing programme : une documentation abondante doit être insérée au listing du programme source afin que l'architecture et le développement de ce dernier soient compréhensibles par des tiers.
Utilisateur : ici la documentation sera double. Elle sera automatiquement insérée dans le déroulement du programme à chaque fois que l'utilisateur aura à intervenir sur la machine (clavier et autres entrées-sorties). Enfin, l'utilisateur devra pouvoir obtenir des informations complémentaires à l'aide de routines documentaires appelées par menus.

Règle de sécurité
L'analyse-programmation, dans sa conduite et ses résultats, doit faire l'objet de mesures de sécurité. Ces mesures auront un double aspect :

- Intégrité et fiabilité de l'application
- Sécurité vis-à-vis de l'environnement

La sécurité vis-à-vis de l'environnement est très importante et s'applique à de nombreux domaines :

- Propriétés des données
- Sauvegarde
- Piratage
- Accès contrôlés
- Cohérence des données
- Manipulations, etc...

CRITERES TECHNIQUES DE QUALITE

Bien que la plus grande partie du travail concerne les critères d'analyse programmation, les critères techniques de qualité doivent être suivis de près.
Deux questions techniques se posent souvent :

- sommes-nous certains d'arriver au résultat ?
- quel va être le coût de réalisation ?

1er : La réponse à la première question peut être résolue en faisant le bon choix en ce qui concerne les algorithmes retenus. Toutefois, il sera nécessaire de vérifier que le matériel peut les supporter.
2e : Le coût se divise en trois postes bien définis :

- coût de l'analyse-programmation
- coût de la configuration nécessaire
- coût du temps d'exécution

COÛT DE L'ANALYSE-PROGRAMMATION

Il dépend de nombreux facteurs :

- du choix des algorithmes (vitesses, précision, etc.)
- du respect des règles vues précédemment
- du degré de détail final à atteindre

En fait, l'un des principaux facteurs de coût sera donné par le choix du langage de programmation retenu.

COÛT DE LA PROGRAMMATION

Le coût de la programmation est lui-même fonction de la richesse du langage et de la syntaxe utilisés.
Ici, le coût tiendra compte :

- des tests et contrôles
- du traitement des erreurs
- des vérifications

COÛT DE LA CONFIGURATION NECESSAIRE

Le coût de la configuration nécessaire sera fonction :

- du processeur
- des récepteurs
- des effecteurs

Un soin particulier sera porté aux mémoires et à leur gestion. Il convient de bien choisir le type de mémoires de masse qui sera retenu. Ce choix pourra être dicté par l'organisation des collections d'objets et leur accès. De même, le rapport coût-performance sera observé.

CRITERES GENERAUX DE QUALITE

CRITERES GENERAUX DE QUALITE	REGLES
algorithme connu, applicable ?	faisabilité
variations vraisemblables ?	évolutivité → généralité
étapes cohérentes, homogènes ?	décomposition fonctionnelle → généralité
solution du problème posé ?	fiabilité → stratégie de vérification
avancement maîtrisé ?	planification
démarche et résultats communicables ?	compréhensibilité
travail sauvegardé ?	sécurité

algorithme

ANALYSE

critères de qualité

généraux
techniques

d'où **règles de conduite**

objets — actions

CRITERES TECHNIQUES DE QUALITE

résultat assuré ?	pratiquement toujours
A quel coût total ?	**somme des coûts :** d'analyse - programmation, de configuration, de temps d'exécution

UTILISATION OPTIMALE DES RESSOURCES

Chaque ordinateur est doté de caractéristiques particulières qui peuvent le rendre plus performant lorsque le programmeur les connaît et sait comment en tirer profit.

STOCKAGE DES DONNEES

En mémoire vive

Il convient de bien affecter chaque variable, d'utiliser les termes entier et réel, de nommer chaque tableau, de bien le dimensionner, et de bien en connaître les possibilités.

En mémoire de masse

Il sera obligatoire de parfaitement maîtriser la gestion des données en mémoire de masse. Le fonctionnement ne doit provoquer aucun mystère aux programmeurs.

COMMENTAIRES ET IDENTIFICATEURS MULTICARACTERES

L'auto-documentation d'un programme tels les commentaires, les variables multicaractères et les labels de ligne, est très utile pour le développement, le test, le déverminage et la maintenance d'un programme. Ces moyens d'auto-documentation prennent de l'espace en mémoire, mais ne devraient pas poser de problèmes dans la majorité des cas.

INITIALISATION DE VARIABLES ET DE TABLEAUX

Il est fondamental d'initialiser toutes les variables avant de les utiliser dans une expression, même si cela n'est pas toujours nécessaire (c'est le cas de nombreux langages interprétés).

EVALUATION DES PERFORMANCES

Il peut être très intéressant de tester la vitesse d'exécution d'un programme, d'un sous-programme, d'une fonction. Dans une opération simple, le besoin n'est pas évident. Par contre, dans des boucles ou des tris, cela devient une nécessité. Il est nécessaire de connaître la vitesse d'exécution des diverses opérations utilisées. Dans le cas des tris, cela peut même avoir des conséquences fâcheuses, car un mauvais algorithme peut se traduire par un temps de calcul prohibitif apportant un temps d'exécution supérieur à un travail manuel.

LES NOMBRES REELS

Il convient de connaître les valeurs minimales et maximales pouvant être stockées en machine ainsi que leur conversion.

LES NOMBRES ENTIERS

Mêmes remarques.

LES CONSTANTES

Si le résultat d'un calcul défini doit être utilisé plusieurs fois dans un programme, il convient d'en effectuer le calcul une seule fois, le résultat sera alors stocké afin de pouvoir l'utiliser dès que nécessaire sans pour cela refaire des calculs qui entraînent des pertes de temps.

LES ALGORITHMES UTILISES

Quel que soit le but recherché, chaque possibilité concernant le choix d'un algorithme sera évaluée. La préférence ira toujours à l'algorithme qui autorise

- le temps d'exécution le plus rapide
- la place mémoire la plus réduite
- la portabilité maximum.

LES TESTS

Les tests font partie intégrante de la réalisation d'un programme. Il s'agit sans aucun doute d'une des phases les plus importantes.
Ce n'est pas le déverminage ni la présence de défauts.
Il s'agit de faire apparaître les défauts par les effets.

- C'est la preuve du bon fonctionnement des programmes
- Ce n'est pas démontrer l'absence d'erreurs
- Ce n'est pas démontrer la finalité du programme
- Ce n'est pas démontrer la conformité du programme.

C'est le contrôle qualité en programmation comparable à la contre-mesure.

Les moyens

Correction → interne : programmes
Fiabilité → externe : utilisation.

Difficulté

a) Taille des programmes
Il n'est pas souhaitable de tester individuellement des paquets d'instructions dépassant 1 koctet
b) Temps nécessaire aux tests
c) Environnement, moyens utilisés.

Coût du test

Dans les grandes applications, le coût des tests peut représenter de 30 à 60 % du coût total de développement.

Efficacité

De bons tests doivent être effectués à tout niveau :

- tests au niveau composants
- tests au niveau systèmes.

Fondements

Les fondements des tests doivent se faire de la manière suivante :

- Modèles des algorithmes
- Complexité de l'application
  a) qualification
  b) quantification
- Fiabilité : résistance à l'échec
- Couverture des tests : pourcentage du logiciel réellement testé
- Preuve des programmes
- Transformation des programmes en cas d'erreurs.

Règles de mise en œuvre

- Changement de programmeur
- Changement d'utilisateur
- Exécution en groupe
- Jeu de tests
- Niveau des tests
- Stratégie des tests → ascendant descendant

Conclusion

- Pourcentage de couverture des tests
- Quantification des catastrophes.

ASSURANCE ET CONTROLE DE LA QUALITE

Les produits logiciels deviennent de jour en jour des produits industriels, il faut donc :
- une qualification du produit
- une quantification du service rendu (facteur de qualité).

Comme tout produit, un programme possède un cycle de vie et un coût de fabrication.

Assurance
Il s'agit de l'assurance des performances, d'un bon fonctionnement et d'une fabrication correcte.

Contrôle
Voir la fiche sur les tests.

Facteur de qualité
Les facteurs de qualité font appel à :
- moyen d'évaluation
- mise en œuvre des moyens d'évaluation.

Moyens d'évaluation
- programme de tests
- règles de tests

Normes
Il existe de nombreuses normes, de nos jours, qui permettent une certaine assurance et un certain contrôle de la qualité. Notons parmi ces normes : DOD - FAA - NSI - NRC - IEEE, etc. La majorité de ces normes est issue d'instances militaires comme le DOD qui est la défense américaine et qui a beaucoup œuvré dans le domaine de la qualité des logiciels. Notons aussi l'IEEE qui est un syndicat professionnel très influent en électronique et informatique.

Mesures
Pour un logiciel, il convient de mesurer :
- le qualitatif
- le quantitatif.

On considère qu'un module ne doit jamais dépasser 50 instructions (module = bloc, voir cours précédent). Pour ce qui concerne l'auto-documentation, une bonne norme semble être un minimum de 28 %, soit pour 100 lignes de programmes, 28 lignes d'informations. En fait, il n'y a jamais trop d'auto-documentation.
Les mesures sont de types :
- Mesure de temps :
  a) vitesse d'exécution
  b) période d'une tâche
- Mesure absolue
  a) capacité de traitement
  b) capacité de stockage
- Logique (mesure à l'aide de tests booléens)

Evaluation
- Nombre d'instructions par bloc
- Nombre de modules par sous-programme
- Taille des modules et sous-programme
- Complexité des programmes
- Durée du projet, nombre de programmes.

MISE AU POINT - DEVERMINAGE

La mise au point, ou le déverminage de programme, vient avant le traitement des erreurs. Le traitement des erreurs consiste à anticiper et à rattraper les erreurs éventuelles.
Si nous pouvons écrire que le meilleur moyen de déverminer un programme consiste en premier lieu à l'écrire correctement, cela paraîtra utopique à de nombreuses personnes. En effet, s'il est assez facile d'écrire un tout petit programme fiable dès le premier essai, cela se complique dès que le listing de l'application s'allonge.
Certaines techniques permettent d'aider le programmeur à déverminer son travail. La plus simple consiste à écrire ses programmes d'une manière très auto-documentée. Mais la plus efficace est sans aucun doute l'utilisation d'une conception descendante du programme.
Certains ordinateurs autorisent une assistance à la mise au point.

UTILISATION D'UN CLAVIER INTERACTIF

Cette solution qui est sans doute la plus simple, consiste à taper au clavier les ordres compris dans le programme les uns après les autres. Il va sans dire que cette technique ne peut s'utiliser qu'avec les petites applications. Mais cela pourra s'avérer très utile sur les machines qui peuvent offrir cette technique avec ou sans programme en cours de fonctionnement. Inversement, cette option ne peut être effectuée que sur interpréteur.

EXECUTION PAS A PAS

Toujours sur interpréteur, l'un des outils les plus puissants pour la mise au point des programmes est la possibilité d'exécuter un programme ligne par ligne. Ce procédé que l'on rencontre sur certaines machines, permet d'examiner les valeurs des variables et la séquence du programme pour chaque instruction.

TRACE DU PROGRAMME

L'exécution pas à pas est assez lente, ce procédé bien qu'il soit très pratique, se révèle surtout très utile lorsque le programmeur sait où se trouve l'erreur. Inversement cela pourra être fastidieux lorsqu'il s'agit de dérouler de longs programmes. Un autre moyen d'examiner le contenu des variables au fur et à mesure des traitements et de contrôler le chemin suivi, est d'utiliser son imprimante en test. On y affichera toutes les informations qui pourront nous aider.

LE DEBUGGER

Certains compilateurs possèdent un débugger. Cet outil très puissant mais qui, mal utilisé, peut être dangereux, permet une exécution pas à pas d'un programme, l'examen et la modification des registres et des mémoires.
Les bons debuggers autorisent :

- l'affichage des variables
- l'exécution d'un certain nombre de lignes
- l'utilisation de points d'arrêt
- le défilement pas à pas
- le formatage des valeurs de sortie des variables
- l'adressage de cases mémoires
- le suivi des lieux statique et dynamique.

TRAITEMENT D'ERREURS

Nous appellerons traitement d'erreurs, le traitement des erreurs qui apparaissent au moment de l'exécution du programme. Ces erreurs peuvent avoir deux causes principales :

- Elles sont dues au programme
- Elles sont issues des données introduites dans la machine.

Que faut-il faire ?
Dans le premier cas, il convient bien entendu de les éviter au moment de la conception. Dans le second cas, il est nécessaire de les rattraper au moment où elles apparaissent. Parfois il peut s'avérer préférable de ne rien faire et de laisser le programme se dérouler et s'arrêter. Toutefois dans ce dernier cas, il sera souhaitable d'avertir l'opérateur de l'existence de l'erreur.
Certains interpréteurs affichent le numéro de type d'erreurs réalisé. Ceci autorise une adaptation rapide. Si un programmeur connaît exactement ce qu'il est censé faire et quel type d'entrée il utilise, il arrive parfois qu'il ne connaisse pas les limites logiques du programme qu'il réalise. Enfin, il est assez courant que l'utilisateur du programme entre des données erronées qui provoquent soit un arrêt de la machine, soit apportent des résultats non fiables. Dans tous les cas, le programmeur devrait faire un effort pour éliminer tout risque d'erreur.

Il convient de noter :

- Les conditions limites du contexte utilisé
- La détection des erreurs par logiciel
- Le rattrapage des erreurs par logiciel.

DETECTION ET RATTRAPAGE DES ERREURS

Certains langages autorisent une gestion complète des erreurs. Dans ce cas, la meilleure solution consiste à utiliser grandement les ordres spécialisés dans ce domaine. Ils permettent de dérouter le déroulement du programme vers des sous-programme de gestion d'erreurs.
Ces sous-programmes serviront soit :

- à corriger la ou les erreurs
- à modifier le processus en cours
- à prévenir l'opérateur de l'erreur
- à reprendre le processus afin d'en corriger les variables.

Les erreurs les plus classiques

- Algorithme non respecté
- Erreur à la saisie
- Sous-dimensionnement des tableaux et fichiers
- Mauvaise attribution de variables.

LES REMEDES

Lorsqu'une erreur de programme est détectée, que faut-il faire ?

S'il s'agit d'un problème mineur :

- Vérifier les entrées (saisie d'information)
- Vérifier les types (voir cours précédents)
- Vérifier la syntaxe du programme (ligne où s'est produite l'erreur)
- Vérifier les équations s'il y a calcul
- Vérifier le contexte

Si l'erreur est importante

- Vérifier l'algorithme
- Vérifier l'architecture du programme, sa structuration.

LES FICHIERS

Les fichiers posent deux problèmes de base :
- Stockage en mémoire de masse
- Validité des données écrites et lues.

STOCKAGE EN MEMOIRE DE MASSE

Il existe trois grandes familles de fichiers :
- les fichiers avec enregistrement physique
- les fichiers avec enregistrement défini
- les fichiers avec enregistrement logique

Les fichiers avec enregistrement physique

Généralement ces fichiers sont les programmes qu'on utilise :
- la longueur physique de l'enregistrement est définie lors de l'initialisation
- l'enregistrement physique est l'information minimale transférée entre l'operating system et l'unité de stockage de masse
- la longueur des enregistrements est fixe.

Les fichiers avec enregistrement défini

Il s'agit ici de fichiers contenant des informations, utilisées par un gestionnaire de données par exemple.
- La longueur d'un enregistrement défini est décidée par l'utilisateur lors de la création d'un fichier .
- La longueur doit être adaptée aux besoins d'accès du bloc de données.
- La longueur des enregistrements est fixée.

Les fichiers avec enregistrement logique

La finalité est la même que pour les fichiers avec enregistrement défini. Mais l'architecture est très différente :
- La longueur d'un enregistrement logique est fonction du format et du contenu des données enregistrées.
- La longueur d'un enregistrement logique peut varier d'un enregistrement à l'autre.
- La longueur des enregistrements est variable.

FICHIERS ET TESTS

Il convient de bien connaître le type de fichier qu'on désire utiliser, l'occupation sur support de mémoire de masse est, elle aussi, très intéressante à connaître. Elle influe sur les vitesses de traitement ainsi que sur l'espace mémoire occupé.

La majorité des fichiers est utilisée en gestion de fichiers classiques. Dans ce cas, on utilise des fichiers avec enregistrement défini. Dès lors, il convient de réaliser les deux opérations suivantes :
- Il est fortement souhaitable de réserver sur le support de mémoire de masse la place nécessaire au fichier une fois complet.
- Il est tout aussi conseillé de créer tous les enregistrements et de mettre les champs à une valeur prédéfinie. Cette information permettra de gérer de manière beaucoup plus simple l'ensemble du fichier (comptage d'enregistrements occupés, tests, tris, etc.)

Les formats

Il est nécessaire d'écrire ses fichiers dans un format qui soit transportable d'une machine à une autre, ex. : code ASCII ou EBCDIC.

Les mises à jour

Pour des raisons de sécurité il sera, dans certains cas, préférable de réaliser ses mises à jour à partir de deux fichiers distincts. Ceci évite toute catastrophe irrémédiable en cas d'incident majeur (coupure de courant, de câble, panne, etc.).

LES ENTREES/SORTIES ET LES ERREURS

Dans un système informatique, l'utilisateur fait appel aux périphériques reliés à l'unité centrale. Ces périphériques sont gérés par les entrées-sorties appelées aussi interfaces. De nombreuses machines permettent, grâce à leur langage, la gestion de ces cartes, mais aussi des erreurs y afférent.
Les périphériques rencontrés dans un ensemble automatique pourront être

- des imprimantes
- des unités de stockage de mémoire de masse
- des lecteurs divers (carte code barre, convertisseur CAD-CDA)
- des écrans
- des claviers, etc.

Il n'est pas rare dans un programme complexe de faire appel à un nombre non négligeable de périphériques. Le moindre problème, la simple anomalie de fonctionnement doivent alors être contrôlés avec efficacité. Il n'est pas souhaitable de bloquer le déroulement d'un programme à cause d'une imprimante par exemple. Cette dernière peut ne pas fonctionner pour différentes raisons :

- l'appareil n'est pas branché sur le secteur
- le sélecteur «prêt à fonctionner» n'est pas opérationnel
- il manque du papier dans le chargeur
- le câble de liaison interface est déficient, etc.

Dans ces différents cas de figure, il y a de fortes chances que l'échange d'informations entre le calculateur et l'imprimante ne se fasse pas. Il est donc souhaitable et fortement conseillé que cette anomalie n'arrête pas le déroulement du programme et entraîne son arrêt. Pour ce faire, cette éventuelle panne devra être détectée et gérée. Le programmeur devra donc pour chaque utilisation d'un périphérique en prévoir les éventuelles anomalies de fonctionnement. Deux voies sont alors possibles : la gestion en temps des protocoles d'échanges et plus simplement si cela est possible l'utilisation de messages d'erreurs. Bien trop de programmes professionnels, soit-disant éprouvés, se bloquent subitement pour un simple problème d'échanges de données. Dans tous les cas de figure, une anomalie d'entrées-sorties ne doit en aucun cas provoquer l'arrêt du programme. L'utilisateur devra toujours être prévenu. Un sous-programme de gestion d'erreurs d'entrées-sorties devra toujours autoriser la reprise du déroulement du programme principal.
Les erreurs le plus souvent rencontrées en entrées-sorties sont :

- bourrage papier sur imprimante
- alimentation papier non effectuée sur imprimante
- mise hors tension
- volet non fermé sur lecteurs de mémoires de masse.

Les erreurs machine

La téléinformatique et les problèmes liés à l'interfaçage sont complexes. Ils font appel à l'électronique et au langage machine.
Au niveau électronique et langage machine, les erreurs le plus souvent rencontrées sont :

- Erreur de format
- Erreur de parité
- Codage
- Horloge
- Protocole
- Statut
- Interruption
- Identification
- Brochage des connections

DETECTION D'ERREUR SIMPLE

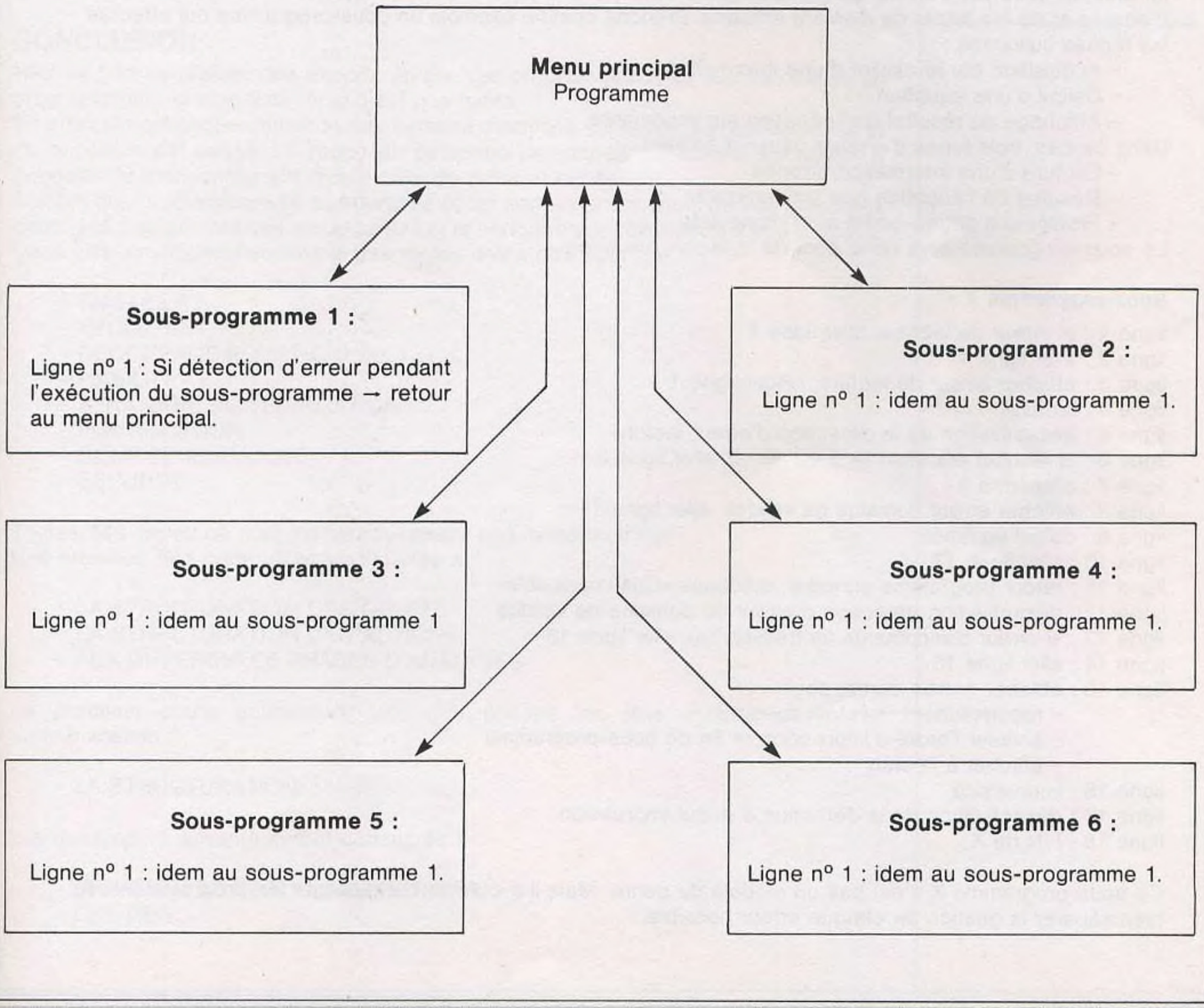
La détection d'une ou de plusieurs erreurs peut se faire pendant l'exécution du programme de deux manières :

- détection d'erreur simple
- détection d'erreur spécialisée avec ou sans correction.

La détection d'erreur simple ne peut être réalisée de façon efficace que sur des programmes bien architecturés faisant appel à de nombreux sous-programmes.
Son but est d'éviter que lorsqu'une erreur grave se manifeste, elle n'entraîne un arrêt général du problème.
Ces erreurs pourront être :

- Problème d'entrées/sorties
- Mauvaise acquisition
- Faute de syntaxe, etc.

Dans ce cas, le seul remède est, lorsque le problème est détecté, de renvoyer le déroulement du programme au menu principal de l'application.



DETECTION D'ERREURS SPECIALISEES

La détection d'erreurs spécialisées permet d'éviter un arrêt d'exécution du programme. En fait, il s'agit d'une série de détections d'erreurs spécialisées pour chacune d'entre elles sur un type d'erreur particulière. La majorité des interpréteurs et des compilateurs-debuggers autorise une gestion spécialisée des erreurs et une série d'ordres est prévue dans ce but. La détection d'erreur peut avoir pour conséquence :

- Retour au menu principal
- Mise en garde de l'utilisateur de programme
- Correction de l'erreur
- Déviation du sous-programme X en sous-programme Y.

ARCHITECTURE DE LA DETECTION

Ici chaque sous-programme gère ses propres détections d'erreurs. Il s'agit de bien évaluer les risques d'erreurs et de les traiter de manière efficace. Prenons comme exemple un sous-programme qui effectue les tâches suivantes :

- Acquisition par le clavier d'une information
- Calcul d'une équation
- Affichage du résultat de l'équation sur imprimante.

Dans ce cas, trois types d'erreurs peuvent arriver :

- Lecture d'une information erronée
- Résultat de l'équation peu vraisemblable
- Problème d'entrée-sortie sur l'imprimante.

Le sous-programme sera donc écrit de la façon suivante :

Sous-programme X :

ligne 1 : si erreur de lecture, aller ligne 3
ligne 2 : aller ligne 4
ligne 3 : afficher erreur de lecture, retour ligne 1
ligne 4 : lecture
ligne 5 : désactivation de la détection d'erreur lecture
ligne 6 : si résultat équation > 5 ou < 0, aller ligne 8
ligne 7 : aller ligne 9
ligne 8 : afficher erreur domaine de validité, aller ligne 11
ligne 9 : calcul équation
ligne 10 : aller ligne 12
ligne 11 : retour programme principal, affichage «Cas impossible»
ligne 12 : désactivation détection d'erreur du domaine de validité
ligne 13 : si erreur d'imprimante (entrée-sortie), aller ligne 15
ligne 14 : aller ligne 16
ligne 15 : afficher entrée-sortie, choix
  – recommencer → retour ligne 13
  – annuler l'ordre d'impression → fin de sous-programme
  – afficher à l'écran
ligne 16 : impression
ligne 17 : désactivation de la détection d'erreur impression
ligne 18 : FIN de X

Ce sous-programme X n'est pas un modèle du genre. Mais il a comme but d'obliger les programmeurs à bien séparer la gestion de chaque erreur possible.

NOTES PERSONNELLES

CONCLUSION

Rien ne sert de réaliser des exploits en analyse-programmation ou des acrobaties en programmation si le résultat final n'est pas fiable.
En effet, l'informatique demande une certaine discipline. Le temps de l'amateurisme et du bricolage est passé. Le règne de certaines personnes qui étaient les seules à connaître le fonctionnement d'un clavier de terminal est fini.
Aujourd'hui, l'informatique se démocratise et fait son apparition dans de très nombreux domaines. Les prix ont beaucoup baissé et la concurrence commence à être efficace.
Dans ces conditions, l'ensemble des règles devra être appliqué :

- FAISABILITE
- EVOLUTION
- DECOUPAGE FONCTIONNEL
- FIABILITE
- STRATEGIE DE VERIFICATION
- PLANIFICATION
- COMPREHENSIBILITE
- SECURITE

Toutes ces règles ne sont malheureusement pas indépendantes.
Une attention très particulière sera portée à :

- LA STRUCTURATION DES OBJETS
- LA STRUCTURATION DES ACTIONS
- AUX DIFFERENTES PHASES D'ANALYSES

Le prochain cours concernera une des phases les plus importantes de la programmation :

- LA STRUCTURATION

Les deux cours suivants seront consacrés à :

- CREATION ET GESTION DE FICHIERS
- LES TRIS.

PETITES ANNONCES

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

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

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

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

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

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

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

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

Vends micro-ordinateur Epson HX 20, 32 Ko, micro-imprimante, microcassette, écran LCD 4 × 20 caractères, mémoire permanente (batterie), logiciels + documentation : 3 500 F à débattre. Imprimante EP-44 : 1 500 F à débattre. Tél : (31) 84.28.74 le soir.

Vends vidéo Geni 16 K + extension 48 K + moniteur : 3 600 F. Tél : 346.85.67.

Vends 20 mensuels Led Micro de 1 à 20 : 14 F le n°, total : 280 F + 28 mensuels Led de 1 à 28 : 14 F le n°, total : 392 F. 94450 Limeil-Brevannes. Tél : 569.47.69 après 19 h.

Vds carte mère Apple + boîtier : 1 500 F, carte controleur disk : 200 F. Tél : (93) 43.11.62.

Vends Oric 48 K + alim + Péritel + magnéto aquarius + livres + K7 : 2 000 F. Mr Chopard. 77 Nandy. Tél : 063.77.20.

Vds Sinclair QL cause double emploi + 4 logiciels francisés + Forth + ass. + Pascal + Lisp + docs + ... sous garantie, valeur 10 000 F, vendu le tt 6 800 F. D. Dagot. Aéroport, 39500 Tavaux. Tél : (84) 72.18.53.

Vds TRS 80 Model 100 portatif, écran C.L 8l × 40 car., 16 K, 4 logiciels intégrés + magnéto cassette Tandy + manuel d'utilisation. Valeur 6 500 F, vendu 4 500 F. Tél : 16 (1) 262.99.96.

Vends Hector HRX 64 K (nov. 84) Forth Magneto Residents + cart. Basic et images pour Sprites + assembleur + K7 + livres + prog. + Joy : 2 500 F (val. 7 200 F). Doukhan. Tél : (91) 44.91.49 Marseille.

BON DE COMMANDE

Pour compléter votre collection de Led-Micro

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

Je désire le n° ☐

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

(cocher le ou les nos désirés)

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

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

Nom : ..... Prénom : .....

Adresse : .....

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

Bulletin d'Abonnement

Je désire m'abonner à Led Micro (10 numéros). France : 160 F - Etranger : 240 F, à partir du n° .....

Nom ..... Prénom .....

N° ..... Rue .....

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

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

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 micro-processeur 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.

Alice.

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

Le Victor PC 15 ne coûte que 24.900 F*.

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

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

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

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

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

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

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

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

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

VICTOR

Comme [redacted] moins cher qu'[redacted]