hors série

Led MICRO

PROGRAMMATION
COURS 2ème CYCLE

ISSN 0757-6889

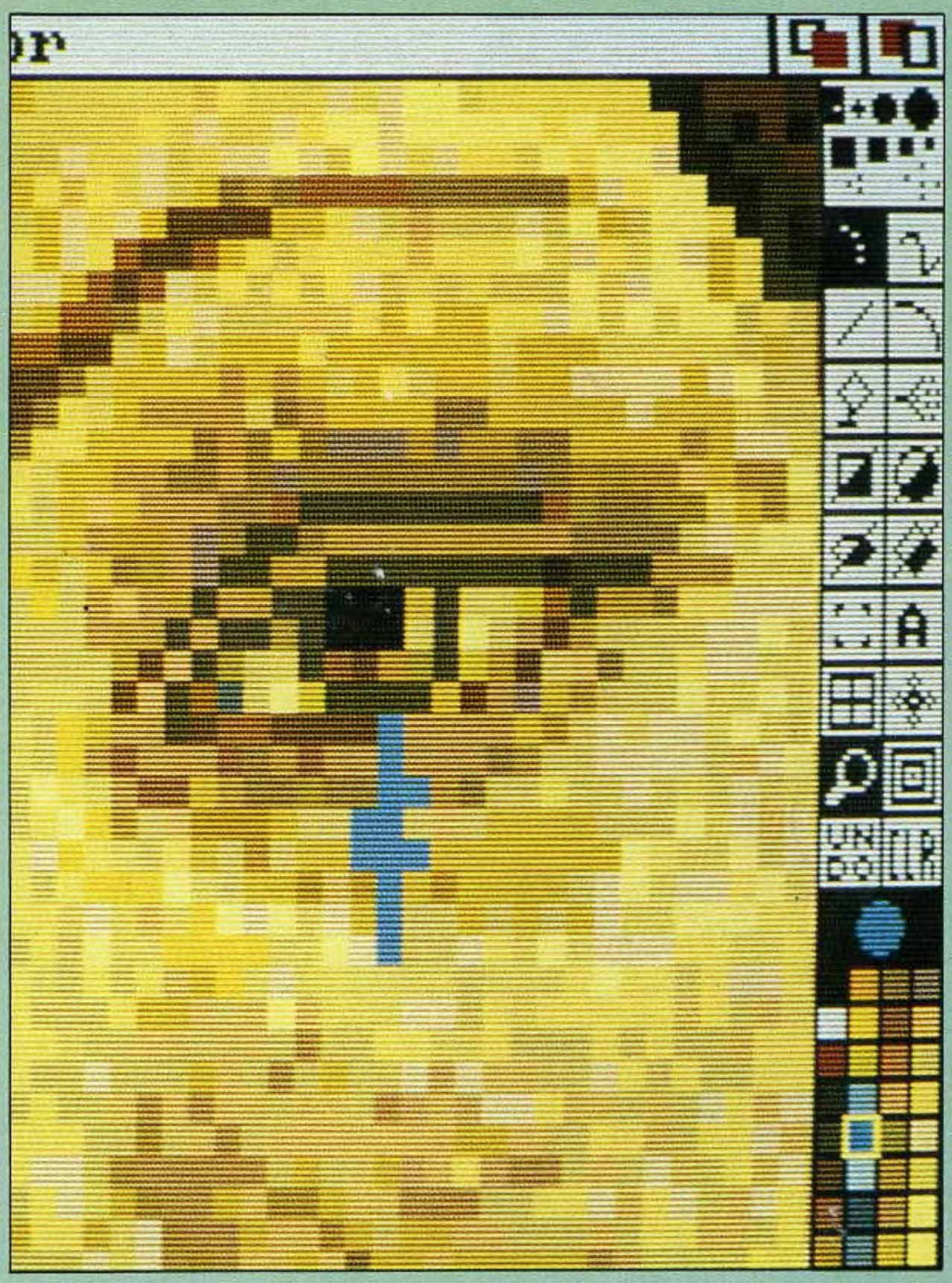
COURS
N°31
Suite
2e cycle

N°11

COURS DE PASCAL
la syntaxe pascalienne

COURS DE PROGRAM-MATION APPROFONDIE :
mise au point d'un programme

COURS DE GENIE LOGICIEL :
de la théorie à la pratique



M1988 -N°31 (11)-18 F MENSUEL JUIN/JUILLET 1986 - BELGIQUE 139 FB/CANADA 3,75 $/SUISSE 6,75 FS

nouveau !

- *exploiter toutes les possibilités des systèmes MIDI*
- *réaliser vous-mêmes un clip vidéo*
- *tirer le maximum de vos synthétiseurs*
- *installer chez vous votre studio d'enregistrement*
- *tout savoir sur les nouveautés musique et vidéo créatives*

Tout cela chaque mois dans Music Vidéo Systèmes

une publication des Editions Fréquences chez votre marchand de journaux

Editions Fréquences 1, boulevard Ney 75018 Paris - Tél. 46.07.01.97

hors série
Led MICRO

PROGRAMMATION COURS 2e CYCLE

Société éditrice :
Editions Fréquences
Siège social :
1, bd Ney, 75018 Paris
Tél. : (1) 46.07.01.97 +
SA au capital de 1 000 000 F
Président-Directeur Général :
Edouard Pastor

JUIN/JUILLET 86

NOTRE COUVERTURE : Essai d'une incrustation d'une larme par outil graphique sur l'œil droit de la Venus de Botticelli sur l'ordinateur Amiga de Commodore.

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

Services **Rédaction-Publicité-Abonnements :**
1, bd Ney, 75018 Paris
Tél. : (1) 46.07.01.97
Lignes groupées

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

Secrétaire de Rédaction
Chantal Cauchois

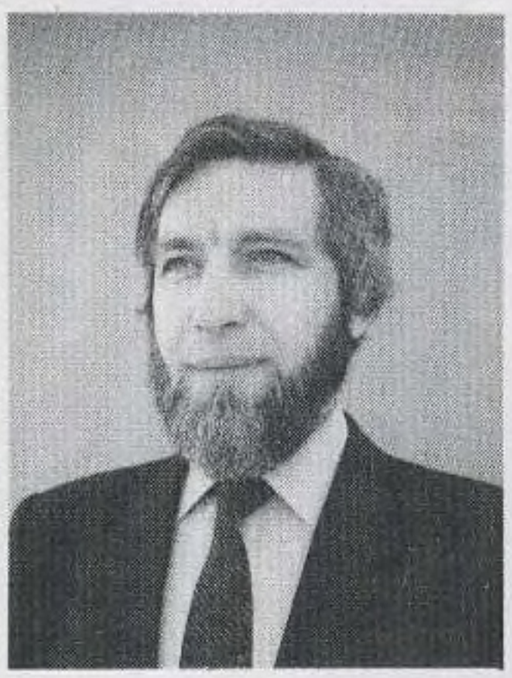
Publicité, à la revue
Tél. : 607.01.97
Secrétaire responsable
Annie Perbal

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

Réalisation
Composition-Photogravure
Edi'Systèmes

Impression
Berger-Levrault - Nancy

Microprocesseurs un cours essentiellement pratique !



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

Pour ceux qui veulent aborder la micro-informatique en désirant en connaître les éléments essentiels ;

ceux pour qui la « puce » ne doit pas rester un mythe.

Electronique digitale ?

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

Diffusion auprès des libraires assurée exclusivement par les Editions Eyrolles.

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

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

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

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

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

Nom ..... Prénom .....

Adresse .....

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

COURS DE PASCAL

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Chose promise, chose due. Aujourd'hui nous allons nous enfoncer dans la syntaxe Pascalienne, pour décrire un certain nombre d'outils indispensables à l'utilisation du langage. Le mois dernier, nous avons vu qu'il est fait de structures imbriquées, dans lesquelles on peut discerner deux grands blocs, les déclarations et les instructions. Dans les premières se trouvent les déclarations de constantes, de types, de variables (dans cet ordre), de procédures et de fonctions. Ainsi toute procédure ou fonction que le programme utilise doit se trouver décrite en partie déclaration. De plus, cet édifice se retrouve à l'intérieur des procédures et des fonctions, c'est-à-dire que ces dernières sont faites de deux blocs, le premier de déclarations...

COURS N°2

PLAN DU COURS

Arrêtons là car nous pourrions continuer longtemps. Disons simplement que c'est le principe des photos où vous voyez quelqu'un tenir la même photo, avec sur celle-ci la même personne tenant la même photo...
Passons aux nouveautés.
Note : Nous avons placé en Annexe la liste des mots réservés du Pascal Standard, et du Pascal UCSD qui sont les deux Pascal les plus courants. Vous aurez ainsi, avant que nous ne vous les décrivions, la liste de toutes les commandes.

1. LE BLOC D'INSTRUCTIONS

1.1 Introduction

Nous avons décrit très en détail le bloc de déclarations le mois dernier. Ce mois-ci, nos investigations seront particulièrement orientées vers le bloc d'instructions. Il suit directement la partie déclarative, et se sert de cette partie pour tout ce qui est utilisation de paramètres et de sous-programmes.
Cela signifie que toute variable, constante, sous-programme et fonction doivent être annoncés dans la première partie.

1.2 Forme générale de ce bloc. Les instructions BEGIN et END

Une grande constante des blocs d'instructions en Pascal est qu'ils sont «coincés» dans des structures qui sont annoncées au système par les mots réservés BEGIN et END, situées en début et fin de bloc. Rappelons que BEGIN signifie commencer ou encore début, et END fin. Toute série d'instructions possède donc ces deux mots.
A l'intérieur d'un tel bloc, il est possible qu'une série autonome d'instructions possède ses propres BEGIN et END.
Ainsi, il est fréquent de trouver :

```
program toto;

    Partie déclarations


begin
   for i:=1 to 10 do
     begin
         if i=5 then
            begin
               write('i');
               writeln(i)
            end
         else
            writeln('non')
     end
end.
```

Naturellement, cela peut être bien plus complexe !!! Mais dans ce cas, généralement, il est préférable de scinder le programme en sous-programmes, pour clarifier la situation.

Notions importantes sur les commandes BEGIN et END :
Il y a un certain nombre de choses à savoir concernant les BEGIN et END.
– Il doit y avoir un END pour chaque BEGIN, sinon lors de la compilation, le système vous dira des grossièretés, du genre **NO END MATCHING,** ou tout autre message fort désagréable comme **ERROR 108,** ce qui est franchement indécent, non ?
– Il peut y avoir des END sans BEGIN, nous le verrons plus tard, mais pour l'instant oubliez cela, car cette situation est assez caractéristique de certaines commandes

que nous verrons par la suite.
– Entre un BEGIN et un END, l'ensemble des commandes est considéré comme une seule et unique instruction. Ainsi un programme est, pour le système une seule instruction, qu'il soit long de mille pages, ou simple comme le programme qui suit. C'est une constante des langages structurés de ne travailler que par instructions élémentaires, formant des instructions de niveaux supérieurs.

```
program toto;
 begin
   writeln('coucou c'est moi');
   writeln('il fait beau, non ?');
   writeln('comment allez-vous ?');
end.
```

Les instructions comprises entre le BEGIN et le END sont donc en représentation interne une seule instruction, que les anglais appellent «compound», ce que nous traduirons librement par «composé» ou encore «bloc». Toutes réclamations au sujet de la traduction sont à envoyer aux Editions Fréquences.

– Les blocs d'instructions peuvent être inclus les uns dans les autres, comme nous l'avons déjà vu, par exemple dans le premier programme de ce cours. Dans ce cas, il y avait donc un bloc, qui en contenait d'autres. Regardons un exemple précis pour compter les instructions distinctes :

```
program toto;

    Partie déclarations


begin
   for i:=1 to 10 do
     begin
        if i=5 then
           begin
              write('i');
              writeln(i)
           end
        else
           writeln('non')
     end
end.
```

Nous avons ici une seule instruction, qui est faite de la boucle FOR, qui contient une seule instruction, le bloc IF THEN ELSE, ce bloc étant composé de deux instructions, une pour le THEN, l'autre traitant le ELSE. Enfin, il y a deux instructions dans la partie THEN, et une dans le ELSE.

1.3 Les points-virgules (;)

Il est possible de considérer dans un premier temps que chaque instruction Pascal est séparée de la suivante par un signe cabalistique, le point-virgule. Dans un bloc, les diverses instructions sont ainsi terminées par ce point-virgule, pour le séparer de l'instruction d'après.
Cette règle comporte des entorses, comme toujours dans la science soit-disant exacte qu'est l'informatique. Elles se justifient néanmoins fort bien, et nous verrons le mois prochain, dans un cours hautement philosophique, le pourquoi de cette attitude.

Nous traiterons dans ce cours de la partie théorique de la syntaxe du Pascal, et de ses représentations graphiques. Les exceptions sont principalement :
– Avant un END ou tout mot réservé du langage, le point-virgule est optionnel, mais certains systèmes le traitent comme une erreur, et il faut tester le votre pour savoir si ce cas est accepté ou prohibé.
– Avant un ELSE, le point-virgule est interdit. Donc pas de :

Attention erreur

```
if x=1 then write(x);
   else writeln('manqué');
```

– Le dernier END est suivi d'un point. C'est le symbole de fin de programme. Cette remarque ne s'applique donc pas aux fins de sous-programmes (procédures ou fonctions).

1.4 Le cas de l'indentation des commandes

Vous avez pu remarquer que dès les premiers exemples de programmes que nous avons proposés, les lignes étaient décalées pour souligner la structuration du langage. Chaque bloc possède ainsi une unité visuelle. Ce décalage est appelé indentation et est sans effet sur le compilateur, qui se contente de l'ignorer. Ceci permet un confort visuel très grand, comme le prouvent les programmes qui suivent. Alors imaginez le cas de programmes de 100 pages.

```
program toto;
var
     n,i: integer;
begin
     for i:=1 to n do
          begin
               if i mod 4 = 0 then
                    begin
                         writeln(i,'est un multiple de 4');
                         writeln('le programme continue')
                    end
               else
                    begin
                         write('reste de la division :');
                         writeln(i mod 4)
                    end
          end
end.
```

Le suivant est moins lisible, ne croyez-vous pas ? Et pourtant, aussi surprenant que cela puisse sembler, il s'agit du même programme, et du même langage.
Et qui plus est, d'un langage évolué.

```
program toto;
var n,i: integer;
```

```
begin for i:=1 to n do begin if i mod 4 = 0 then  begin
writeln(i,'est un multiple de 4');
writeln('le programme continue') end else begin
write('reste de la division :'); writeln(i mod 4) end end end.
```

Il est sûr que le résultat est plus compact. Mais lequel préférez-vous ? En fait, les retours à la ligne que nous avons mis sont par pure raison de présentation, car aucune n'est recquise. Vous pouvez écrire tout programme sur une seule ligne. Qui se propose de faire une telle chose ?
Restons sérieux. En général, le mieux est d'aligner le BEGIN et le END correspondant, avec le contenu du bloc inclus entre ces deux commandes décalé vers la droite, pour rendre bien visible les limites. En bref, prenez la structure du premier programme ci-dessus comme modèle simple, même si vous ne décalez pas autant.
N'oubliez pas que le seul objectif est de réaliser un programme aussi lisible que possible. Donc, abusez des décalages, et rappelez-vous qu'il est également possible de sauter autant de lignes que vous le voulez, à tout moment.

1.5 Les commentaires (Rappel)
Comme la plupart des langages, Pascal autorise l'insertion de commentaires. Pour cela, pour prévenir le programme qu'il s'agit de commentaires, nous disposons en Pascal comme en BASIC de symboles cabalistiques. Ici, il s'agit des caractères (* et *) à la fin. Par exemple, un commentaire pourra être :

```
(* ceci est un commentaire *)
```

Ceci est la commande standard, mais certains Pascal proposent, en plus, la commande { }, qui a le même effet. Il ne s'agit pas d'un autre standard, simplement d'une commande supplémentaire. Attention, à la suite d'une erreur d'impression, les accolades ont été remplacées par des parenthèses dans le numéro précédent, Led n°10, page 8.

```
{ ceci est aussi un commentaire }
```

Il est possible de placer des commentaires absolument n'importe où sauf s'il coupe un mot du langage en deux. Soyons juste, ceci constitue rarement une limitation de toute façon.
Plus vous mettez de commentaires, plus le programme est clair. Comme le langage est compilé, les commentaires sont ignorés à la compilation, donc ils ne ralentissent pas le programme lors de l'éxecution. Cela n'a donc que des avantages.

1.6 Les affectations en Pascal
En Pascal, contrairement au BASIC, l'affectation ne se fait pas avec le symbole =, mais avec les symboles := sans espace entre les deux. De plus, une affectation ne peut se faire qu'entre paramètres ou variables de même type. Prenons des exemples.

```
program essai;
var
        a,b : integer;
        x,y : real;
        lettre, autre_lettre : char;

begin
        a:=1;
```

```
        a:=a+1;
        b:=2;
        a:=a+b;
        x:=1.2;
        y:=x;
        b:=-104;
        x:=x-y;
        lettre:='a';
        autre-lettre:='x'
end.
```

Vous pouvez constater que nous n'avons jamais pu réaliser des affectations de type : **a: = x + y,** qui de toute manière ne voudrait rien dire. Par contre, il peut être utile de faire x: = a pour utiliser des entiers avec des réels. Nous vous montrerons comment faire cela.
Le symbole = est utilisé en Pascal pour étudier l'égalité de deux quantités sans affectations. Ainsi, dans un test, vous aurez :

```
if x=y then ...
```

Cela permet la dissociation de deux utilisations foncièrement différentes du symbole égal. C'est là une des grandes forces du langage, même si, à première vue il semble que ce soit plutôt une entrave ou une complication.
Précisons enfin que les autres langages (Fortran, C, etc) utilisent eux aussi deux symboles différents pour distinguer l'affectation du test d'égalité.

2. LES OPERATEURS ARITHMETIQUES

Introduction
Nous allons décrire dans cette partie un certain nombre d'opérateurs mais pas la totalité. En effet, certains sont liés à des structures ou à des types (au sens du Pascal) que nous n'avons pas encore décrits. Il viendrait fort mal à propos de les présenter ici.

2.1 Les opérateurs de base

Vous disposez des quatre opérations (si, si, c'est carrément le luxe) via les symboles habituels, c'est-à-dire +, −, *, /
En plus de ces opérateurs, un certain nombre d'autres sont standards.

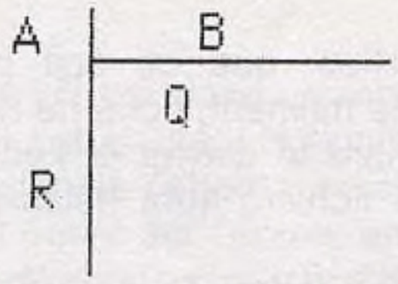
2.2 Les instructions MOD et DIV

Parmi ces opérateurs se trouvent MOD et DIV. Ils sont complémentaires l'un de l'autre. DIV permet le calcul de la division entière de deux nombres. Qu'est-ce que cette opération barbare ? Tout simplement, lors d'une division, vous vous arrêtez avant d'aller au-delà de la virgule. Ainsi, la division entière de 3 par 2 est 1.

3 DIV 2 = 1 10 DIV 3 = 3

La théorie mathématique, qui réussit à rendre tout logique, à défaut d'être simple vous dit : considérant deux nombres A et B, il existe deux autres nombres, et deux seulement, disons Q et R, tel que l'on ait :

$A = BQ + R$ et $R < B$



Ici, Q est le résultat de la divison entière.

Q = A DIV B

MOD est le résultat complémentaire, c'est-à-dire R dans notre exemple, MOD signifie MODULO, qui, en mathématique, est le reste de la divison entière que nous venons d'évoquer. D'où le choix de R, qui signifie RESTE.

2.3 D'autres fonctions mathématiques

En Pascal standard, vous avez les commandes :

- ABS valeur absolue
- EXP passage à l'exponentielle
- LN logarithme en base e
- SQR carré d'un nombre
- SQRT racine carrée d'un nombre
- ROUND arrondi d'un nombre à l'entier le plus proche
- TRUNC tronque un nombre en l'entier correspondant

Au sujet de ces deux dernières commandes, il est sans doute nécessaire de revenir sur la notion de partie entière. Pour ceux qui ne la connaissent pas, la fonction partie entière renvoie le plus grand nombre entier inférieur au nombre d'entrée. Ainsi, en notant ENT cette fonction, vous aurez :

ENT(1.2) = 1
ENT(− 1.2) = − 2 car − 1 est plus grand que − 1.2

Nous voyons donc que ni ROUND ni TRUNC ne correspondent à cette définition. Ne l'oubliez pas, car l'usage erroné de ces fonctions est fréquent.

2.4 Les opérateurs relationnels

Vous les connaissez pour la plupart. Il s'agit de commandes permettant de comparer deux quantités pour décider de la suite. Voici une liste de ces opérateurs :

< =	égalité	
< =	supérieur ou égal	Pas de blanc entre les deux
> =	inférieur ou égal	Pas de blanc entre les deux
<	supérieur	
>	inférieur	
<>	différent	Pas de blanc entre les deux

Il n'est sans doute pas nécessaire de s'apesantir sur ces commandes, similaires à celles que nous avons rencontrées en BASIC, que le plus grand nombre d'entre vous a déjà manipulé. Disons simplement que dans le test suivant, c'est la seconde partie qui est exécutée, car la condition est fausse, donc le programme passe au ELSE sans exécuter les instructions dans le THEN :

```
x:=3.5;
a:=2;
if a>=x then writeln(' a est plus grand')
else writeln(' x est plus grand');
```

3. LES ENTREES-SORTIES

Pascal dispose de possibilités importantes d'entrées-sorties, que ce soit pour contrôler le clavier, l'écran, des fichiers ou tout autre. Pour le moment, nous ne nous intéresserons qu'aux possibilités plus interactives, concernant le clavier et l'écran. Disons simplement que les entrées sorties de gestion de fichiers sont hélas peu standards. Air connu...

Vous avez certainement noté que nous avions utilisé les commandes WRITE, WRITELN, READ, READLN, lors de programmes d'exemples. Ceux d'entres vous qui ont quelques familiarités avec l'anglais ont pu comprendre l'utilité de ces commandes, en gros tout au moins. Pour bien saisir les nuances entre les commandes sans LN et avec, tentons l'expérience avec un petit programme.

```
program essai(input, output);
var
      x : real;
      i : integer;
      a : char;

begin
      write('donnez un nombre quelconque : ');
      readln(x);
      write('le nombre est : ');
      writeln(x);
      write('donnez maintenant un nombre entier : ');
      readln(i);
      writeln('le nombre est : ');
      writeln(i);
      write('donnez maintenant une lettre : ');
      read(a);
      write('la lettre est : ');
      writeln(a)
end.
```

Le résultat de ce programme est :

```
donnez un nombre quelconque :   12.7
le nombre est : 12.7
donnez maintenant un nombre entier : 4
le nombre est :
 4
donnez maintenant une lettre :  h  la lettre est :  h
```

La différence entre avec et sans LN est donc un renvoi à la ligne suivante, après la commande, lorsque le suffixe LN est ajouté. LN signifie «line», ligne en anglais.
Vous avez aussi vu que READ signifie l'attente au clavier d'une donnée, en entrée donc («input»), et que WRITE fait écrire sur l'écran, en sortie («output»).
L'un des principaux problèmes du Pascal est que la différence entre READ et READLN est en général mal implémentée sur certains systèmes, et que READ occasionne des plantages, ou des effets indésirables. Il est donc préférable d'utiliser READLN, ce qui ne pose pas de problèmes pratiques, le plus souvent.

Comment entrer des données à partir d'une seule instruction READLN. Il suffit de mettre en paramètres les données que vous souhaitez demander. Prenons un exemple très court, la somme de trois nombres.

```
program somme(input,output);

var
      x,y,z : real;

begin
      write('donnez trois nombres : ');
      readln(x,y,z);
      writeln('leur somme fait : ',x+y+z);
end.
```

Simple, non ? Ce programme est équivalent au suivant, mais avec une petite différence. La trouverez-vous ? Sinon essayez les tous les deux.

```
program somme_deux(input,output);

var
      x,y,z : real;

begin
      write('donnez trois nombres : ');
      readln(x);
      readln(y);
      readln(z);
      writeln('leur somme fait : ',x+y+z);
end.
```

Et que fera donc celui-ci ?

```
program somme_deux(input,output);

var
     x,y,z : real;

begin
     write('donnez trois nombres : ');
     read(x);
     read(y);
     readln(z);
     writeln('leur somme fait : ',x+y+z);
end.
```

Vous voyez que de multiples options sont disponibles pour réaliser une même chose.

Remarque : Nous avons volontairement ajouté les paramètres (INPUT, OUTPUT) à la déclaration. Il est possible que votre environnement PASCAL vous exige cette déclaration. Dans le cas contraire, rien ne gène à garder cette déclaration.

4. CONCLUSION

Nous avons vu ce mois-ci comment était constitué le bloc d'instructions, avec sa structure, sa cohésion très forte, rigide même. L'utilisation, et la présentation des commandes BEGIN et END ont été décrites. Puis nous avons parlé des commandes et opérateurs arithmétiques, mais vous verrez un peu plus tard que cette liste n'est pas complète, et ne le sera que lorsque nous parlerons des structures et types d'ensembles. Nous avons décrits les symboles d'égalité et d'affectation, du bon usage du point-virgule, enfin des commandes d'entrées-sorties.

5. DES EXERCICES

1) Construire un programme qui écrive la somme des 100 premiers entiers.
2) Construire un programme qui écrive votre nom, en décalant d'une lettre à droite à chaque fois, puis lorsque vous êtes au bout de la ligne en fasse autant dans le sens inverse (voir exemple).

```
EDITIONS FREQUENCES
 EDITIONS FREQUENCES
  EDITIONS FREQUENCES
   EDITIONS FREQUENCES
    EDITIONS FREQUENCES
     EDITIONS FREQUENCES
      EDITIONS FREQUENCES
     EDITIONS FREQUENCES
    EDITIONS FREQUENCES
   EDITIONS FREQUENCES
  EDITIONS FREQUENCES
 EDITIONS FREQUENCES
EDITIONS FREQUENCES
```

3) Ecrire un programme écrivant tous les mots de deux lettres imaginables. Pour cela, sachez que les lettres sont, en Pascal, un ensemble ordonné, de «A» à «Z».
4) Ecrire un programme qui calcule tous les produits de deux entiers entre 1 et 100.
5) Ecrire un programme qui après avoir lu, disons cinq chiffres, fasse leur somme, leur produit, et toute opération qui vous intéresse (moyenne, variance, ...).

ANNEXE : LES MOTS RESERVES DU LANGAGE

Nous vous proposons cette liste pour que vous connaissiez la liste des mots réservés, et que vous ne tentiez pas de donner à une variable d'un de vos programmes l'un de ces noms. De plus, ceci vous permettra de connaître ces mots, et de disposer d'une référence. La liste est en deux grandes parties. La seconde est la liste supplémentaire des mots du Pascal UCSD. Cette implémentation un peu différente est principalement celle que l'on trouve sur les ordinateurs de la marque Apple. Les possibilités additionnelles sont surtout dans le domaine du traitement des chaines de caractères, et des commandes graphiques, points sur lesquels le Pascal standard possède des lacunes certaines.

1) Les commandes

AND	ARRAY	BEGIN	CASE	CONST	DIV
DO	DOWNTO	ELSE	END	EXTERN	FILE
FOR	FUNCTION	GOTO	IF	IN	LABEL
MOD	NIL	NOT	OF	OR	PACKED
PROCEDURE	PROGRAM	RECORD	REPEAT	SET	THEN
TO	TYPE	UNTIL	VAR	WHILE	WITH

2) Les fonctions intégrées

ABS	ARCTAN	CHR	COS	EOLN	EOF
EXP	LN	ODD	ORD	PRED	ROUND
SIN	SQR	SQRT	SUCC	TRUNC	

3) Les opérateurs

+	–	*	/	DIV	MOD
AND	OR	NOT	<	>	=
<=	>=	<>	IN	TRUE	FALSE
MAXINT	NIL	↑			

4) Les types prédéfinis

ARRAY	BOOLEAN	CHAR	INTEGER	REAL	RECORD

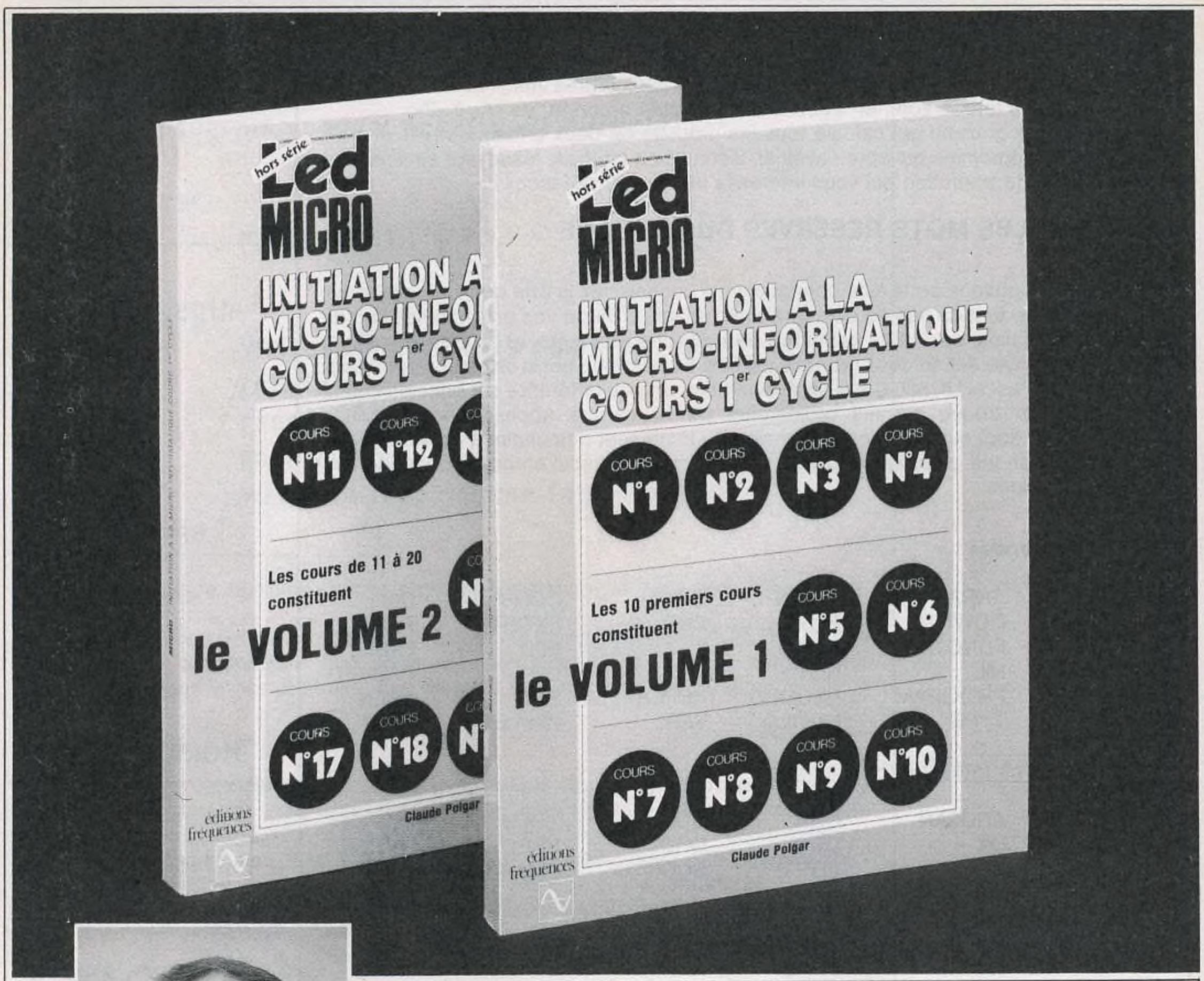
Attention ! Certains ordinateurs recherchent les mots réservés très profondémment provoquant par là-même des erreurs particulièrement exaspérantes. Si vous appelez une variable BING par exemple comme booléen et que vous fassiez :

IF BING THEN

Il est capable de comprendre

IF B **IN** THEN

et de vous rétorquer que vous n'avez pas plus défini B que G dans les variables. Avouez que ce n'est pas très malin ! Heureusement, tous ne le font pas.



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

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*

Diffusion auprès des libraires assurée exclusivement par les Editions Eyrolles.

Initiation à la micro-informatique C. Polgar

Bon de commande à retourner aux Editions Fréquences 1, boulevard Ney 75018 Paris.

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

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

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

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Comme pour la structuration des données, la résolution des erreurs nous a beaucoup plus entraîné que prévu et sera encore le sujet de ce cours. Nous vous présentons les méthodes limitant les risques d'erreur ou les rendant facilement corrigibles.
Nous vous présentons encore de nouveaux exercices. Celui concernant les joueurs de tennis devrait normalement être rapidement résolu par le programme que vous avez fait pour la C.E.E. en tous cas, bonnes vacances.

COURS N° 11

La mise au point d'un programme

PLAN DU COURS

1. Analyse de l'application
2. La programmation
   - Quelques idées pour faciliter la mise au point
   - Conclusion
   - Le tournoi de tennis

LA MISE AU POINT D'UN PROGRAMME

Le mois dernier nous vous avons présenté un premier aperçu des problèmes que vous pouvez rencontrer dans la mise au point de vos programmes. Nous revenons en détail ce mois-ci sur tous les éléments susceptibles de vous aider à «debugger» vos programmes, c'est-à-dire : trouver puis supprimer les erreurs de toutes sortes qui l'empêchent de «tourner rond».

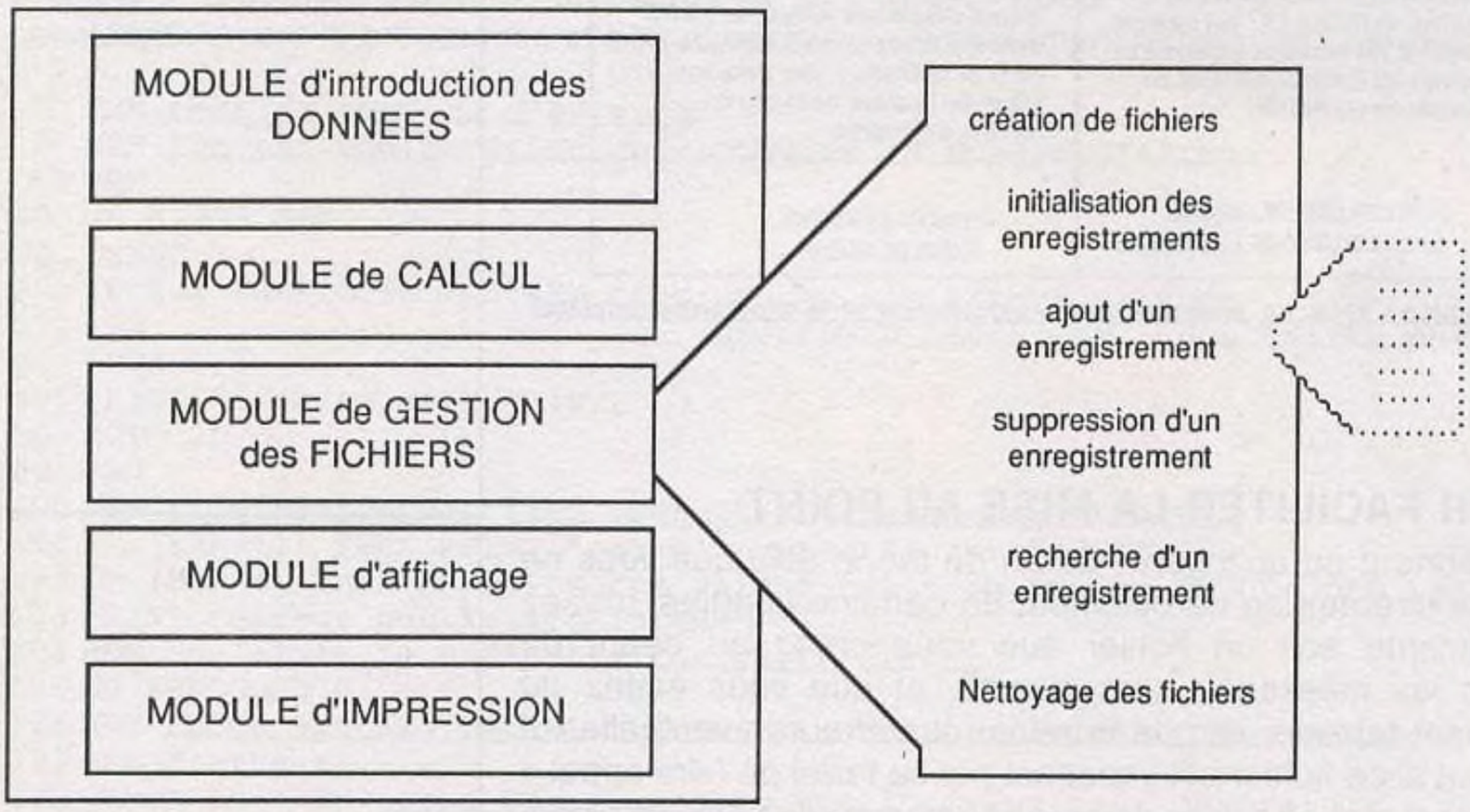
Afin de bien cerner le sujet, nous allons aborder les choix à opérer en fonction des phases suivantes :
- la phase d'analyse,
- la phase de programmation,
- la mise au point du programme,
- la finition et les dernières retouches,
- l'amélioration des performances.

Une dernière précision avant d'entrer dans le vif du sujet, cet article devrait aider tout programmeur quel que soit son langage préféré, mais les exemples feront essentiellement référence au BASIC, que vous connaissez bien ou au PASCAL que vous découvrez.

I. ANALYSE DE L'APPLICATION

Ce ne sera pas une surprise (tout du moins pour nos fidèles lecteurs) de nous entendre dire haut et fort que cette phase est essentielle ; vous connaissez maintenant notre point de vue sur la programmation structurée, et nous ne reviendrons pas sur ce sujet. Il faut néanmoins souligner que les avantages de la programmation structurée sont importants pour :
- permettre de développer de grosses applications en les scindant en modules indépendants (le plus possible) les uns des autres,
- faciliter la maintenance de ces applications, ou plus simplement la relecture de ces modules,

Ainsi, si une erreur vient perturber le bon fonctionnement de votre programme, il suffira



EXEMPLE d'une analyse d'une application de type GESTION

de reprendre ce module responsable et de l'analyser plus en détail. Cette opération sera d'autant plus aisée que vous aurez pris soin de bien commenter chaque partie complexe voire équivoque du module en question.
Toutefois, l'analyse d'une application, c'est avant toute chose la détermination des grandes lignes de votre programme, le choix des algorithmes et les limites que vous (ou un cahier des charges) précisez. Aussi, pensez à choisir de temps en temps les algorithmes les plus simples (à programmer) et non pas forcément les plus performants, et réservez éventuellement certaines analyses plus fines au moment de la programmation, voire de la phase de test.

II. LA PROGRAMMATION

Bien que l'on espère ne pas avoir d'erreurs, il est préférable d'inclure directement des messages renseignant sur le déroulement du programme ainsi que sur le contenu de variables importantes. Nous insistons sur cet aspect important car la plupart d'entre nous travaillons en BASIC qui ne dispose ni de variables locales ni d'obligation de déclarations. Les trois exemples ci-dessous montrent bien l'importance d'une grande rigueur de programmation dans un environnement structuré où l'on utilise des sous-programmes indépendants (qui ont peut être été conçus indépendamment) et des passages fictifs de paramètres (par la réservation de noms de variables déterminées).

```
10 FOR I=1 TO 10
20 GOSUB 1000
30 NEXT
40 END
   ::::::::
   ::::::::
1000 REM I est une variable
1010 REM que l'on croit sûre
1020 I=I+1
1030 PRINT I
1040 RETURN
```

Du fait d'une double utilisation de la variable I,ce petit programme affichera à son exécution le résultat suivant (test effectué sur APPLE 2):

```
]RUN
1
3
5
7
9
]
```

Erreur de Nom de variable
Utilisation d'une même variable

```
10 GOSUB 1000
20 GOSUB 2000:REM calcul
30 PRINT RESULTAT
   ::::::::
1000 REM saisie d'une valeur
   ::::::::
1040 INPUT ENTRE
   ::::::::
1200 RETURN
   ::::::::

2000 REM calcul
   ::::::::
2100 RESUL=ENTRE*10.25+U0
   ::::::::
2200 RETURN
```

Le programme ci-dessus utilise simultanément deux variables RESUL et RESULTAT qui peuvent être soit identiques soit différentes suivant les BASICs (nombres de caractères significatifs).

Erreur de Nom de variable
Confusion de nom (ou de type)

```
10 DIM T(100)
20 REM initialisation
25 INPUT TAILLE
30 FOR I=1 TO TAILLE
40 INPUT T(I)
50 NEXT I
   ::::::::
200 NB=TAILLE:GOSUB 1000
   ::::::::
1000 REM sous-prog général de
1010 REM moyennage
1020 for II=0 to NB
1030 M=M+T(II)
1040 NEXT
1050 M=M/(NB+1)
1060 RETURN
```

Dans cet exemple, il y a confusion au niveau de l'indice d'origine du tableau. le sous-programme compte de 0 à NB alors que le tableau n'est gèré qu'à partir de l'élément 1. Une autre confusion est possible au niveau du nombre d'éléments.

Erreur de paramètre
Indice de tableau

EXEMPLES d'erreurs de passage de paramètres entre un sous-programme indépendant et le programme principal

QUELQUES IDEES POUR FACILITER LA MISE AU POINT

A. Si votre programme est important ou qu'il gère l'écran de façon telle que vous ne puissiez pas afficher de traces directes de déroulement de certaines parties, utilisez alors soit directement l'imprimante soit un fichier que vous créez au début du programme, dans lequel tous les messages sont inscrits, et que vous éditez ou imprimez une fois le déroulement terminé, afin de retrouver les erreurs éventuelles. Il est même alors possible (surtout si ce fichier est imposant par sa taille) de faire appel à un autre programme qui se chargera d'effectuer des recherches ou des statistiques à partir du fichier lui-même.
Un exemple simple est la recherche d'une valeur anormale dans une liste de valeurs et de déduire l'origine possible de cette erreur :

```
100 FOR N=1 TO NB
110 GOSUB 1000:REM calcul suspect à vérifier
120 GOSUB 2000:REM routine espionne qui génère un fichier
    ::::::::::::
    ::::::::::::
200 NEXT
210 LPRINT "fin de la boucle ligne 100-200":LPRINT

2000 REM impression des valeurs des variables les plus importantes
2010 LPRINT "itération ";N;"/ X,Y,Z=";X,Y,Z
2020 RETURN
```

On obtient avec ce sous-programme une liste de la forme:

```
itération     1 / X,Y,Z=    10.5       0.5        0.7
itération     2 / X,Y,Z=    12.35      -26.01     0.7
.....................................................
.....................................................
.....................................................
itération   27 / X,Y,Z=-99999E9        0.0        0.00014   ERREUR !
.....................................................
.....................................................
.....................................................
itération   54 / X,Y,Z= 99999E9        1.0        0.0001    ERREUR !
.....................................................
.....................................................
.....................................................
fin de la boucle ligne 100-200
.....................................................
.....................................................
```

B. Utilisez avec discernement les fonctions d'aide au «Debugging» comme les fonctions TRACE, NOTRACE, (ou TRON, TROFF, suivant les systèmes) ou encore les ON ERR... GOTO.
Il faut bien reconnaître que les premières ne sont guères utiles, si ce n'est à ralentir considérablement le programme et seront donc avantageusement remplacées par des

```
 5  REM test de gestion d'erreurs
 7  REM ERR est une fonction qui retourne le numéro d'erreur
 8  REM
10  ON ERROR GOTO 100
20  INPUT Z
25  IF Z=1 THEN GOTO 200:REM on provoque une erreur de type
27  REM                              UNDEFINED LABEL car 200 n'existe pas
30  INVZ=1/Z
40  PRINT "inverse de Z:";INVZ
50  GOTO 20
60  END
100 REM si division par zero (ERR=11), dépassement (ERR=6)
110 IF (ERR=11) THEN PRINT "pas d'inverse de zero":GOTO 130
120 IF (ERR=6) THEN PRINT "votre valeur est trop proche de 0":GOTO 130
123 PRINT "erreur non traitée:";ERR
124 REM on laisse le BASIC traiter cette erreur
125 ON ERR GOTO 0:
126 REM cette syntaxe rétablit le traitement par le BASIC des messages
127 REM d'erreurs.
127 ERROR ERR:REM cette fonction provoque l'erreur pour générer le
128 REM            message (ici si on entre pour valeur 1, on
129 REM            provoque le message UNDEFINED LABEL.
130 RESUME 60:REM fin du programme
```

PRINTs judicieusement placés aux endroits clés. Il faut savoir en outre que le ON ERR GOTO n'est attrayant que lorsque votre programme tourne et que vous êtes capable de prendre en charge l'ensemble des erreurs d'exécution qui pourraient se produire. Ces erreurs ont souvent pour origine des problèmes dans la gestion des entrées/sorties (plus de place sur la disquette, plus de disquette dans le lecteur désigné, fichier inexistant, imprimante non prête ou non branchée, etc...) ou de la mémoire disponible. Mais attention, si une erreur (de syntaxe par exemple), jusqu'alors bien cachée dans votre programme, voit le jour, il est très possible que la façon dont vous traitez les erreurs, la fasse passer complètement inaperçue... A partir de cet instant, votre programme fera tout et n'importe quoi !

C. Par contre, si vous disposez d'une fonction de TRACE vous montrant toutes les opértions que vous effectuez sur les fichiers (comme le MON, C, I,O de l'Applesoft), profitez-en, c'est la seule façon de s'assurer que l'on écrit ou lit réellement quelque chose. Malheureusement, la plupart des BASICs n'ont pas cette fontion et la seule solution est alors d'examiner directement les fichiers à partir d'utilitaires (dans le genre DUMP sous CP/M, MS-DOS ou similaire), à condition bien entendu de connaître quelque peu la structure interne. Vous trouverez ci-dessous l'exemple d'un programme BASIC et du fichier généré par ce programme relu à l'aide de l'utilitaire DUMP sous CP/M.

```
10 REM creation d'un fichier
20 REM relu ensuite grace a l'utilitaire DUMP (sous CP/M)
30 OPEN "O",#1,"FICHE.DAT"
40 INPUT "nom:";N$
45 IF N$="" THEN GOTO 100
50 INPUT "prenom:";P$
60 INPUT "tel:";T$
70 INPUT "adresse:";AD$
80 GOSUB 1000
90 GOTO 40
100 CLOSE 1
110 END
1000 REM ecriture sur le fichier
1010 PRINT #1,N$;P$;T$;AD$
1020 RETURN
```

```
      SECTOR #   00 00
0000  4D 41 52 54 45 4C 43 48 41 52 4C 45 53 28 41 55 MARTELCHARLES(AU
0010  43 55 4E 29 31 20 41 56 45 4E 55 45 20 44 45 53 CUN)1 AVENUE DES
0020  20 43 45 52 49 53 45 52 53 20 50 55 54 45 41 55  CERISERS PUTEAU
0030  58 0D 0A 4D 41 52 54 49 4E 4A 45 41 4E 2D 46 52 X..MARTINJEAN-FR
0040  41 4E 43 4F 49 53 28 31 36 20 31 29 20 34 34 2E ANCOIS(16 1) 44.
0050  34 34 2E 30 30 2E 30 30 31 32 20 52 55 45 20 4C 44.00.0012 RUE L
0060  41 46 41 59 45 54 54 45 0D 0A 1A 43 3A CD 64 1C AFAYETTE...C:Md.
0070  E8 22 EA 0B EB C3 B4 5E 00 01 2B 2A 76 0A 2B 22 k"j.kC4↑..+*v.+"
```

Return + Line Feed
Fin de Ligne

Marque de Fin de Fichier
(CTRL-Z

D. Pour simplifier l'entrée de données, surtout si elles sont nombreuses, il est préférable :
– d'utiliser la seule commande INPUT dont les caractéristiques sont bien standardisées plutôt que des fonctions comme les GET ou INKEY$,
– de se limiter à une variable par INPUT,
– et enfin de ne lire que des variables de types chaines de caractères. Il est ensuite facile de contrôler la validité de la réponse par des conversions adaptées au cas traité (à l'aide de la fonction VAL par exemple) et si nécessaire de boucler sur cet INPUT tant que toutes ces conditions ne sont pas remplies. L'exemple ci-dessous illustre cette remarque :

```
10 REM Exemple de conversion de chaines:
15 REM la souplesse d'un telle solution permet notamment
17 REM de tester plusieurs cas très différents les uns des autres
18 REM ainsi la fin du programme.
19 REM Dans ce cas on ne souhaite pas avoir un nombre nul !
20 INPUT Z$:IF Z$="" THEN GOTO 20
25 IF (Z$="fin") OR (Z$="FIN") THEN GOTO 70
30 Z=VAL(Z$)
40 IF Z=0 THEN GOTO 20:REM la chaine ne correspond pas à un nombre,
45 REM                     ou le nombre est nul
50 PRINT "Z=";Z
60 GOTO 20
70 END
```

CONCLUSION

Nous ne sommes pas au bout de nos peines, et malgré toutes ces précautions, un gros programme «tourne» rarement du premier coup. Aussi, aborderons-nous à la rentrée de septembre les phases douloureuses de mise au point, de finition, mais aussi d'optimisation. Nous en profiterons pour vous donner un aperçu de méthodes générales de développement avec notamment l'utilisation de bibliothèques de sous-programmes (procédures pour les pascaliens), et de structures standardisées (constantes, variables, données, fichiers, etc.) pour faciliter l'interfaçage entre différentes routines au sein d'une même application.

LE TOURNOI DE TENNIS

Six couples participent à un tournoi de tennis. Ils s'appellent Ambert, Bellot, Charon, Dupont, Evrard, Flamand. Les épouses se prénomment, Laurence, Martine, Nicole, Ophélie, Patricia et Roselyne. Chacune d'entre elles est originaire d'une ville différente, Grenoble, Honfleur, Sedan, Toulouse, Valence et Paris. Elles ont toutes une chevelure de couleur différente, noire, brune, grise, rousse, auburn et blonde.
Les parties suivantes se déroulèrent :
- Ambert et Bellot contre Patricia et Martine
- Charon et Dupont contre Nicole et Martine
- Les femmes aux cheveux noirs et bruns jouèrent d'abord contre Ambert et Charon puis contre Dupont et Bellot.

Furent joués en simple :
- Patricia contre Charon
- Dupont contre Evrad
- les femmes aux cheveux gris contre Laurence
- Ophélie contre Roselyne
- la Parisienne contre Laurence
- Nicole contre Ophélie
- Laurence contre la blonde et celle aux cheveux auburns,
- l'originaire de Honfleur contre Ambert et Charon
- Bellot contre Nicole et Roselyne,
- celle de Sedan contre la rousse et celle aux cheveux noirs
- Charon contre Ophélie et Roselyne
- la Toulousaine contre les cheveux gris
- Flamand joue contre celle aux cheveux noirs
- celle aux cheveux auburns contre Ophélie
- Evard contre la Parisienne
- la Sedanaise contre Nicole
- la Parisienne contre Roselyne
- Dupont contre la Sedanaise
- la Grenobloise contre la rousse

Enfin, dernier renseignement, aucun mari n'a joué dans le même match que son épouse.
Quels sont les noms de famille de Patricia, Martine, Nicole, Laurence, Ophélie et Roselyne ?

DIALOGUE AVEC NOS LECTEURS

1. Les Tours de Hanoï

Dernier exemple d'application des sous-programmes récursifs, avec cette programmation des célèbres tours de Hanoï ! Les principes sont donc très proches des deux exemples des mois précédents. On peut les résumer de la façon suivante :
1) Utilisation d'une pile sous forme d'un tableau unidimensionnel pour sauvegarder, puis restituer le contexte,
2) Appel récursif du sous-programme, c'est-à-dire appel à lui-même dans sa définition.

2. Remarque préliminaire sur l'utilisation des piles

Nous avons cette fois-ci utilisé une seule pile plutôt que quatre, pour la sauvegarde des paramètres utilisés dans le sous-programme récursif. En effet, il n'y a aucun intérêt à effectuer quatre sauvegardes indépendantes, contrairement à l'application sur les arbres équilibrés (AVL) qui avaient besoin de deux piles indépendantes utilisées par deux procédures différentes.

3. Principe de transfert des disques

Comme nous en avons maintenant l'habitude, il suffit pour résoudre ce problème de déplacement d'étudier la situation à un moment donné, en examinant l'ensemble des cas possibles.

Supposons que nous ayons à bouger N disques ; deux cas sont possibles : soit N est nul dans ce cas il n'y a rien à faire ; soit N est non nul et alors :

a) On déplace (N – 1) disques sur la tour intermédiaire (IN dans le programme) ; il suffit pour ce faire d'un appel récursif à la routine elle-même à la seule condition d'avoir inversé la tour de destination et la tour intermédiaire (d'où l'échange de IN et de DE).

b) Comme il ne reste plus qu'un disque sur la tour de départ, on le déplace vers la tour d'arrivée, grâce au sous-programme des lignes 5 000 à 5 050.

c) Enfin, il nous faut à nouveau déplacer les (N – 1) disques qui sont sur la tour intermédiaire au-dessus du Nième disque de la tour de destination, on appelle donc encore une fois la routine récursivement avec, dans ce cas, un échange entre tour intermédiaire et tour d'origine, ... et c'est tout !

L'appel pour le mouvement de l'ensemble d'une tour revient à appeler la routine (débutant en 3000) après avoir initialisé N à taille, et les trois variables ORI IN et DE en fonction du transfert désiré. A noter, l'utilisation de IN qui simplifie énormément l'échange entre tours. Comme pour les applications traitant des arbres binaires, la sauvegarde n'est nécessaire que pour le premier appel récursif, car on utilise après celui-ci les variables sauvegardées (ORI, IN, DE, IN), alors que ce n'est pas le cas après le second appel.

4. Les autres variables du programme

Essentiellement, nous avons besoin :
– de stocker la valeur de chaque tour (c'est le tableau SOMMET),
– de connaître la valeur de chaque niveau des trois tours (tableau T),
– d'une pile assez grosse, (le choix des 20* taille n'est pas du tout la valeur optimale mais si vous avez des idées sur la question ou surtout si vous êtes limités en taille mémoire, tenez-en compte dans votre déclaration),
– enfin d'un compteur du nombre de déplacements. Le calcul de ce nombre se fait directement et facilement (pour les matheux !), et on obtient dans une taille N de la tour de départ :

Nbre de déplacements = 2^{N-1}, cela peut vous aider dans la mise au point du programme.

5. L'affichage des trois tours

Cette partie fait appel à des fonctions d'affichages classiques, sauf peut-être SPACE $ (nbre_blancs) qui, comme son nom l'indique, permet d'afficher nbre_blancs à l'endroit du curseur. Elle sera très facilement remplacée par un sous-programme si vous n'en disposez pas. Rien ne vous empêche d'agrémenter cette sortie par des tracés graphiques en haute résolution qui donneraient l'illusion du mouvement.

6. La fonction SWAP

Cette commande permet d'échanger deux variables :

SWAP x, y, équivaut à : temps = x : x = y : y = temps.

Remarque
La routine récursive des lignes 3000 à 3080 peut être utilisée à n'importe quel moment pour effectuer un transfert partiel d'une partie d'une pile vers une autre pile, grâce au paramétrage par ORI, IN, DE et N. Dans le cas où vous désirez programmer une aide à la résolution de ce problème des tours de HANOÏ, il est possible de l'appliquer pour, par exemple, donner la solution en cours de manipulation, ou encore montrer la, ou plutôt une, méthode de résolution. Voilà autant d'idées de programmes qu'il vous appartient de développer !

7. Le listing du programme

```
100  REM  ***************************
102  REM  *      TOUR DE HANOI      .*
103  REM  ***************************
105  DEFINT A-Z
110  INPUT "nbre de disques:";TAILLE
120  GOSUB 1000:REM initialisation
125  FOR i=1 TO TAILLE
```

```
128  T(1,i)=TAILLE-i+1
129  NEXT i
130  REM au départ, la 1ère TOUR est pleine et les 2 autres
135  REM vides. On choisit la 3ième TOUR comme point d'arrivée.
140  SOMMET(1)=TAILLE:SOMMET(2)=0:SOMMET(3)=0
145  REM désignation des variables:
146  REM  N   : est le nombre de disques à transporter,
147  REM  ORI : est le numéro de la Tour de départ concernée par
                le déplacement,
148  REM  DE  : est la Tour de destination qui doit recevoir les
                disques à déplacer,
149  REM  IN  : est la tour intermédiaire servant aux déplacements
155  REM  num : comptabilise le nombre de déplacements.
160  DE=3:IN=2:ORI=1:N=TAILLE:num=1
170  GOSUB 3000:REM transport de tous les disques de 1->3
180  END

1000  REM déclaration des tableaux
1005  REM SOMMET indique la hauteur de chaque TOUR
1007  REM T est le tableau des 3 TOURS
1010  DIM T(3,TAILLE),SOMMET(3)
1020  DIM PILE(20*TAILLE)
1030  PPILE=0:D$="":REM D$ sert à la partie d'affichage
1035  FOR i=1 TO TAILLE:D$=D$+"*":NEXT i
1040  RETURN

2000  REM affichage des 3 piles T(1,.),T(2,.) et T(3,.)
2020  FOR i=TAILLE TO 1 STEP -1
2010  FOR j=1 TO 3
2030  PRINT" ";:GOSUB 2500:PRINT " ";
2040  NEXT j
2045  PRINT:NEXT i
2048  FOR i=1 TO 6*TAILLE+12:PRINT"=";:NEXT:REM le socle
2049  PRINT"    ";num:REM le numéro du déplacment
2050  num=num+1:RETURN

2500  REM affichage d'un disque
2510  L=T(j,i)
2520  PRINT SPACE$(2*(TAILLE-L));LEFT$(D$,L);" ! ";LEFT$(D$,L);
      SPACE$(2*(TAILLE-L));
2530  RETURN

3000  REM tranfert de N disques de ORI vers DE (IN est la tour
```

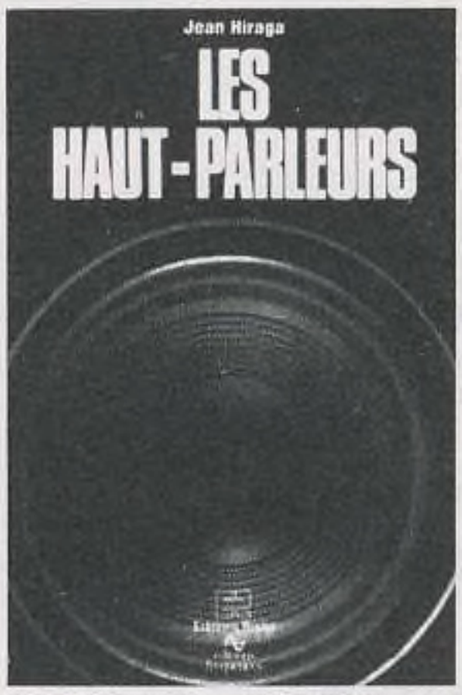
```
3005  REM intermdéiaire).
3010  IF (N=0) THEN RETURN:REM on ne fait rien
3020  GOSUB 4000:REM on empile.
3025  REM on commence par transférer les (N-1) premiers disques sur
3027  REM la tour intermédiaire.
3030  N=N-1:SWAP DE,IN:GOSUB 3010
3040  GOSUB 4500:REM on dépile.
3045  REM On pose le dernier disque sur la TOUR d'arrivée.
3050  OO=ORI:DD=DE:GOSUB 5000
3055  REM on transfert les (N-1) premiers disques sur la tour
3057  REM d'arrivée au-dessus du disque qui vient d'être posé.
3060  N=N-1:SWAP ORI,IN:GOSUB 3010
3080  RETURN

4000  REM empilage par saut de 4 éléments de tableau.
4010  PILE(PPILE+1)=N
4020  PILE(PPILE+2)=DE
4030  PILE(PPILE+3)=IN
4040  PILE(PPILE+4)=ORI
4080  PPILE=PPILE+4
4090  RETURN
4500  REM dépilage, attention à bien respecter l'ordre!
4510  PPILE=PPILE-4
4520  N=PILE(PPILE+1)
4530  DE=PILE(PPILE+2)
4540  IN=PILE(PPILE+3)
4550  ORI=PILE(PPILE+4)
4560  RETURN

5000  REM transfert du sommet de la TOUR OO vers la TOUR DD.
5010  SD=SOMMET(DD):SO=SOMMET(OO)
5020  T(DD,SD+1)=T(OO,SO):T(OO,SO)=0
5030  SOMMET(DD)=SD+1:SOMMET(OO)=SO-1
5040  GOSUB 2000:RETURN:REM à chaque appel à cette routine, on
5050  REM                   affiche la nouvelle situation.
```

Nous tenons à féciliter Monsieur MAUDOT de Calluire dont nous ne pouvons publier le programme. En effet, ce lecteur nous a envoyé un exemple très développé de gestion de budget, comprenant des procédures de comparaisons entre les dépenses de chaque mois, entre autres.
Mais plus encore que le programme, nous avons reçu avec celui-ci une documentation fort bien fournie, Monsieur MAUDOT ayant poussé la courtoisie jusqu'à photographier ces résultats affichés sur son Canon X 07 ainsi qu'une notice explicative des instructions BASIC de la machine. En informatique, c'est un devoir de bien expliquer un programme et celui-ci en est un parfait exemple. Bravo.

BIBLIOTHEQUE TECHNIQUE

Collection études (format 165 × 240)

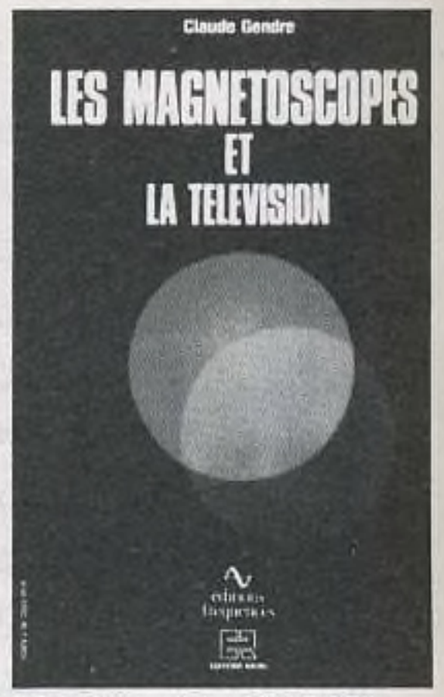
E 15. 184 p. Prix : **140 F TTC**
Face au développement spectaculaire des synthétiseurs, grâce à l'électronique numérique, le besoin d'un ouvrage complet, accessible, et surtout bien informé des dernières ou futures techniques, se faisait ressentir. Le vœu est comblé, en 180 pages... à dévorer.



E 01. 320 p. Prix : **165 F TTC**
Un gros volume qui connaît un succès constant : bien plus qu'un traité, il s'agit d'une véritable encyclopédie, alliant théorie et pratique, histoire, en une mine inépuisable d'informations, reconnue dans le monde entier !

E 05. 160 p. Prix : **155 F TTC**
C'est le premier ouvrage paru en langue française traitant de l'audionumérique ; écrit par un professionnel, avec rigueur, simplicité, il explique brillamment les bases de cette technique : quantification, conversion, formats, codes d'erreurs.

E 04. 240 p. Prix : **154 F TTC**
Seconde édition améliorée d'un ouvrage fort attendu des passionnés d'électroacoustique. Ce livre permet aux amateurs et aux professionnels de se familiariser avec les rigoureuses techniques de modélisation des haut-parleurs et enceintes acoustiques et d'en mener à bien la réalisation.



E 03. 256 p. Prix : **155 F TTC**
Complément direct des «Magnétophones», les «Magnétoscopes et la Télévision» débute par un bel historique de la télévision et la description des premiers magnétoscopes. La théorie et la pratique de la capture et de l'enregistrement moderne des images vidéo en sont la teneur essentielle.

E 22. 136 p. Prix : **150 F TTC**
Faisant suite à la parution de «L'électronique des micro-ordinateurs», cet ouvrage s'adresse aux électroniciens qui désirent s'initier aux montages périphériques des micro-ordinateurs, interfaces en particulier, qui permettent la communication avec le monde extérieur.

E 06. 128 p. Prix : **150 F TTC**
Cet ouvrage est destiné aux électroniciens désireux d'aborder l'étude du «hard» des micro-ordinateurs. Cette étude s'articule autour du microprocesseur Z-80, très répandu, et en décrit les éléments périphériques : mémoires, clavier, écran, interfaces de toutes sortes.

E 02. 160 p. Prix : **92 F TTC**
Pour tout savoir sur le magnétophone, depuis l'avènement de cette mémoire des temps modernes, jusqu'aux enregistreurs numériques en passant par la cassette. «Les magnétophones» est un ouvrage pratique, complet, indispensable à l'amateur d'enregistrement magnétique.

E 13. 256 p. Prix : **165 F TTC**
Une sélection des meilleurs articles de la célèbre revue «l'Audiophile» choisis parmi les plus significatifs des quinze premiers numéros, introuvables aujourd'hui. Le tome 1 traite de l'électronique audio, à tubes et à transistors.

E 12. 256 p. Prix : **155 F TTC**
Dans un esprit identique, le tome 2 traite du domaine passionnant que constituent les transducteurs en audio ; on y aborde la modélisation théorique des enceintes, la conception géométrique des tables de lecture, le réglage des cellules et des bras.

Collection loisirs (format 135 × 210)

L 07. 160 p. Prix : **68 F TTC**
Le «dernier coup de patte» apporté à un montage, celui qui fait la différence entre la réalisation approximative et le kit bien fini, ce savoir-faire s'acquiert au fil des ans.. ou en parcourant «Conseils et tours de main en électronique».

L 10. 200 p. Prix : **130 F TTC**
Tout beau, tout nouveau, le lecteur laser. Qu'en est-il réellement ? Pour en savoir plus, un livre traitant du sujet s'imposait. «Les lecteurs de compact-discs» permet de faire son choix parmi 37 modèles testés, analysés, examinés et écoutés.

L 09. 72 p. Prix : **65 F TTC**
Pour la première fois en électronique, un lexique anglais-français est présenté sous une forme pratique avec en plus des explications techniques, succinctes mais précises. Ce sont plus de 1 500 mots ou termes anglais qui n'auront plus de secret pour vous.

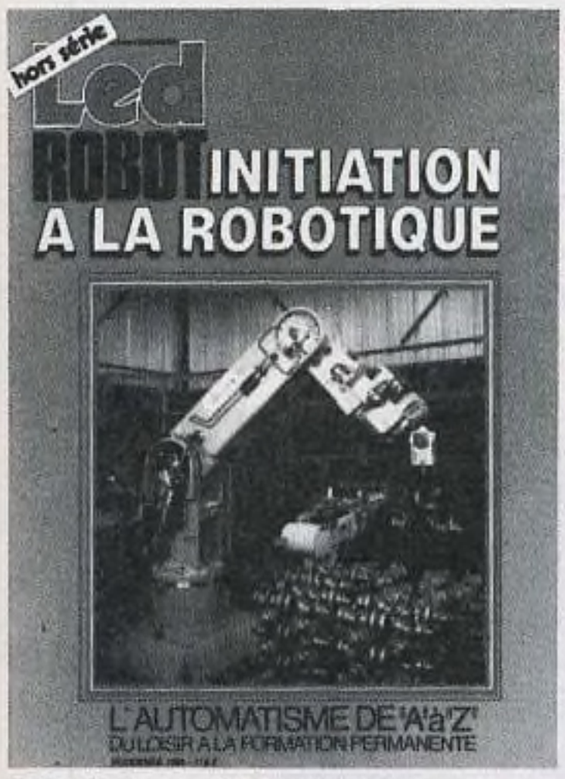
L 11. 160 p. Prix : **85 F TTC**
Finis les calculs fastidieux et erronés ! Grâce à cet ouvrage, les concepteurs d'enceintes acoustiques gagneront un temps appréciable durant la phase d'étude et de mise au point : 120 abaques et tableaux pour tous types de filtres et d'impédances de HP !

L 14. 128 p. Prix : **95 F TTC**
Voici enfin réunies dans un même ouvrage, dix-sept descriptions complètes et précises de montages électroniques simples. Il s'agit de réalisations à la portée de tous, dont bon nombre d'exemplaires fonctionnent régulièrement. Les schémas d'implantation et de circuits imprimés sont systématiquement publiés.

L 20. 208 p. Prix : **130 F TTC**
Accessible à tous, «Week-end photo» permet de découvrir de façon simple les différents aspects de la photographie actuelle. Vous y trouverez les bases indispensables pour vous perfectionner, un guide de choix des appareils 24×36 et des illustrations abondamment commentées.

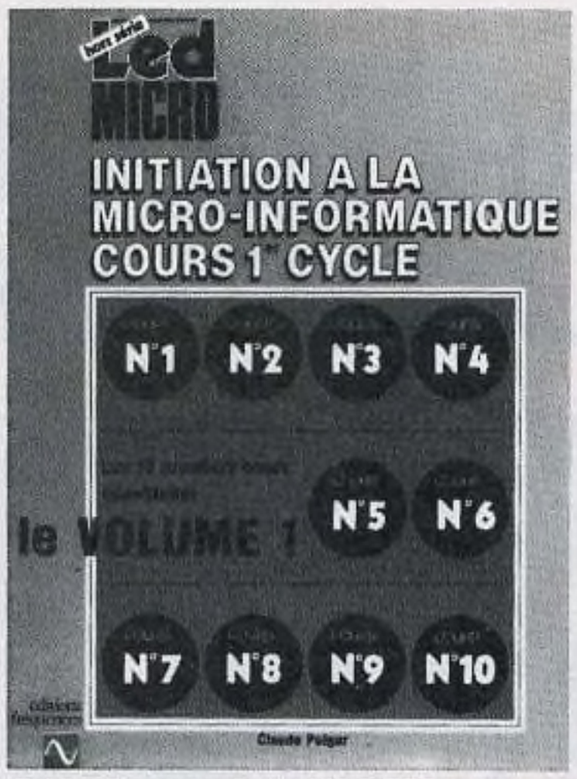
DES EDITIONS FREQUENCES

Collection initiation (format 210×270)



P 08. 96 pages. Prix : **115 F TTC**

Cet ouvrage eut un succès retentissant dès sa sortie. Bien plus qu'un cours d'initiation, il s'agit aussi du premier recueil d'informations données par les concepteurs, les utilisateurs de robots et les fans de cybernétique, enfin réunis !

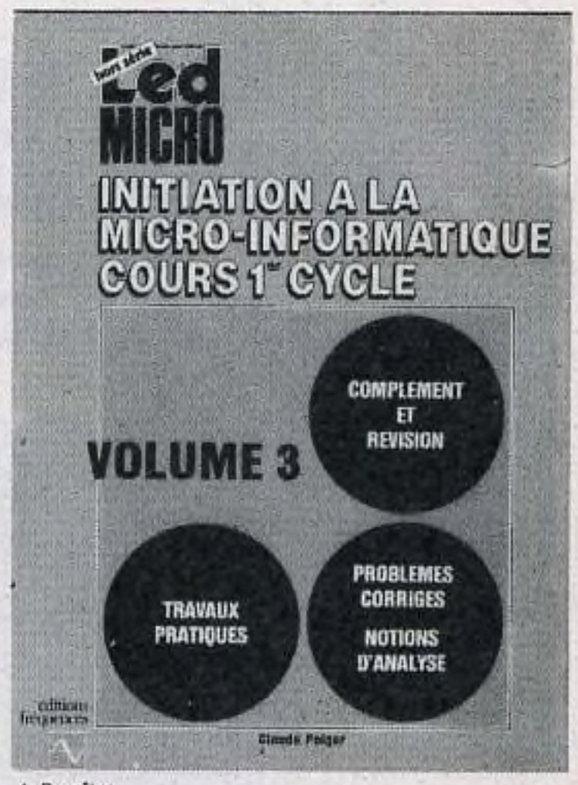


P 16. 272 pages. Prix : **130 F TTC**

Passé les premiers remous de la révolution que fut l'avènement de la micro-informatique, il fallut bien tenter d'en réunir les enseignements. Une lacune apparut : celle d'un ouvrage d'initiation à la programmation, universel et complet. En voici le premier tome.

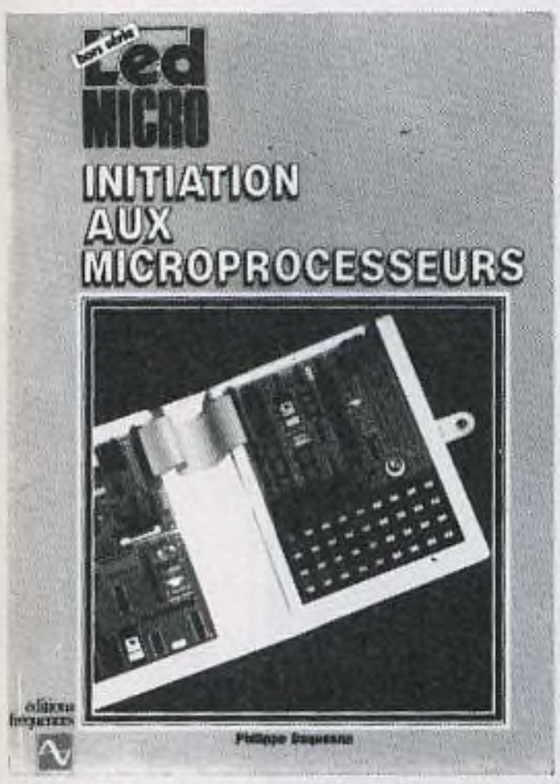
P 17. 208 pages. Prix : **130 F TTC**

Le tome 2 est la suite du tome 1 : l'esprit puissamment didactique de l'auteur s'y retrouve, le contenu du livre permettra d'acquérir un niveau suffisant pour exercer l'analyse, la programmation, la gestion, l'automatisme, la simulation et d'autres choses encore !



A Paraître

Le troisième volume du cours de Programmation, dû à Cl. Polgar, pédagogue apprécié de tous. Il continue dans la lignée d'un réel souci didactique, de haut niveau, maintenant, mais en conservant l'aspect progressif qui fit son succès initial.

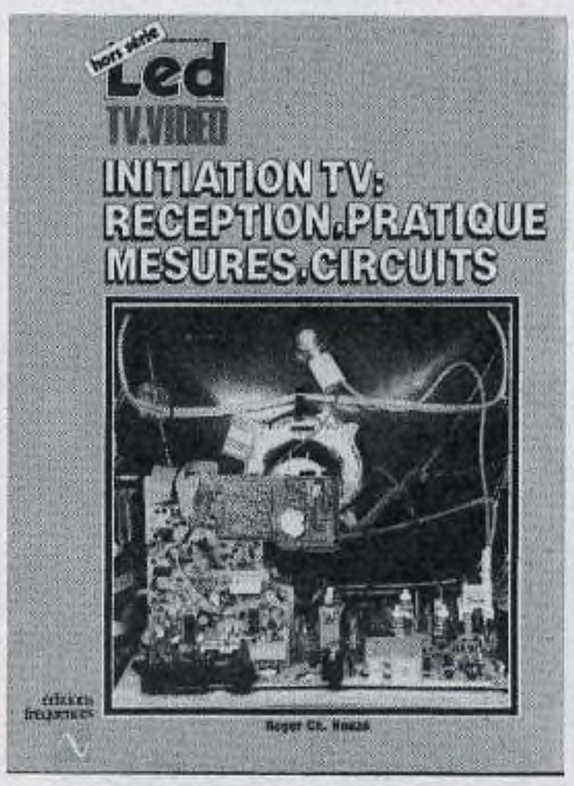


P 18. 136 pages. Prix : **95 F TTC**

Du même auteur, Ph. Duquesne, on nous propose cette fois-ci, de pénétrer au cœur même de l'ordinateur, de comprendre le fonctionnement de l'élément vital qu'est le microprocesseur et enfin de maîtriser l'assembleur, langage du microprocesseur.

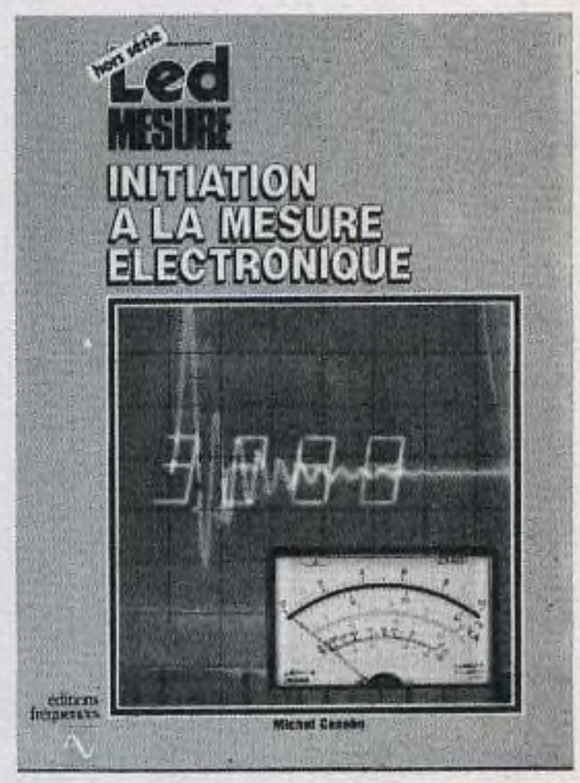
P 19. 104 pages. Prix : **95 F TTC**

Ce cours d'Initiation à l'Electronique Digitale est dû à Ph. Duquesne, chargé de cours de microprocesseurs au CNAM. L'objet de cet ouvrage est de présenter les opérateurs logiques et leurs associations. La technologie est évoquée, brièvement, elle aussi.



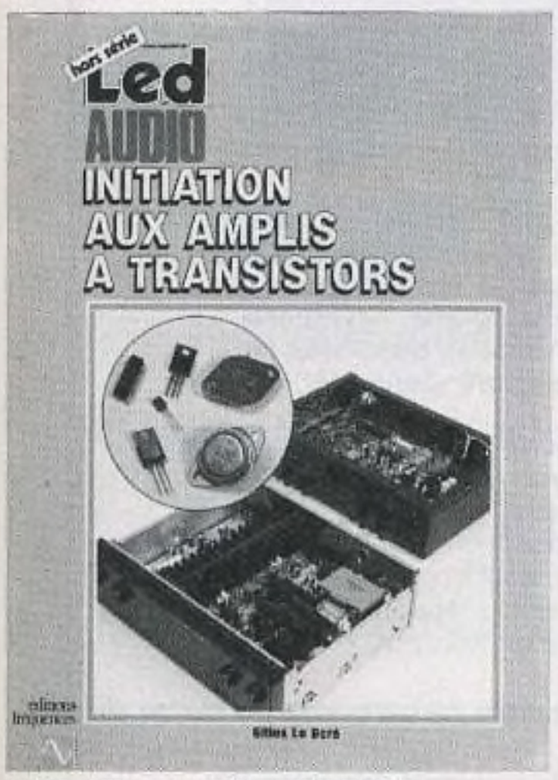
P 21. 136 pages. Prix : **135 F TTC**

Issu d'un cours régulièrement remis à jour, ce livre permet à l'amateur comme au professionnel de se tenir au courant de l'état actuel de la technologie en télévision. De nombreux schémas explicatifs illustrent le contenu du livre.



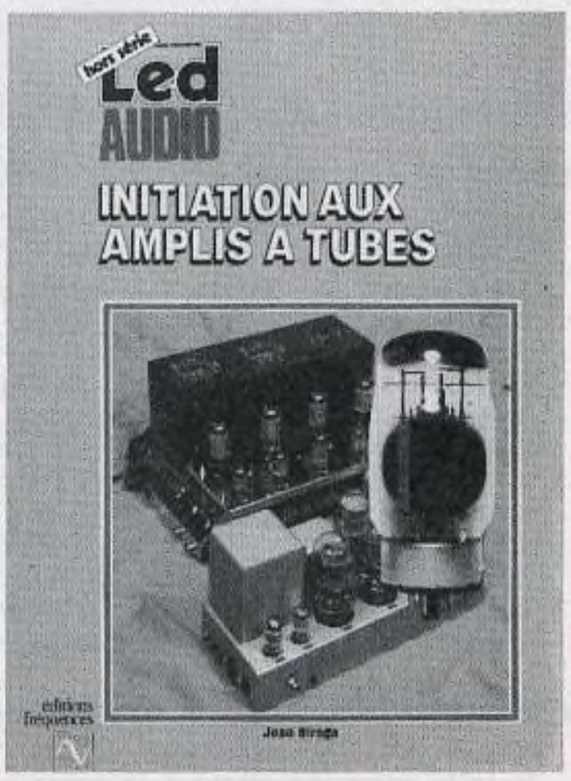
P23. 120 pages. Prix : **140 F TTC**

Il n'existait pas, jusqu'à présent, un ouvrage couvrant de manière générale mais précise, l'ensemble des problèmes relatifs à l'instrumentation et à la méthodologie du laboratoire électronique. C'est chose faite aujourd'hui avec ce volume récemment paru.



P 24. 96 pages. Prix : **130 F TTC**

Après un bref historique du transistor, cet ouvrage traite essentiellement de la conception des amplificateurs modernes à transistors. La théorie est décrite de manière simple et abordable, illustrée d'exemples de réalisations commerciales. Le but du livre est de donner à chacun la possibilité de réaliser soi-même son amplificateur...



P 26. 152 pages. Prix : **155 F TTC**

Complémentaires des «Amplis à transistors», les «Amplis à tubes» sera certainement une petite encyclopédie sur ce sujet : historique, mais aussi polémique, puisque les tubes sont encore d'actualité et parce que les arguments en faveur de cette technique et ses défenseurs sont encore nombreux.

Diffusion auprès des libraires assurée exclusivement par les Editions Eyrolles.

Bon de commande à retourner aux Editions Fréquences
1, boulevard Ney 75018 Paris

Je désire recevoir le(s) ouvrage(s) ci-dessous référencé(s) que je coche d'une croix :

E 01 ☐	E 02 ☐	E 03 ☐	E 04 ☐	E 05 ☐
E 06 ☐	L 07 ☐	P 08 ☐	L 09 ☐	L 10 ☐
L 11 ☐	E 12 ☐	E 13 ☐	L 14 ☐	E 15 ☐
P 16 ☐	P 17 ☐	P 18 ☐	P 19 ☐	L 20 ☐
P 21 ☐	E 22 ☐	P 23 ☐	P 24 ☐	P 26 ☐

Frais de port : + 10 F par livre commandé,
soit la somme totale ci-jointe, de Frs par

CCP ☐ Chèque bancaire ☐ Mandat-lettre ☐

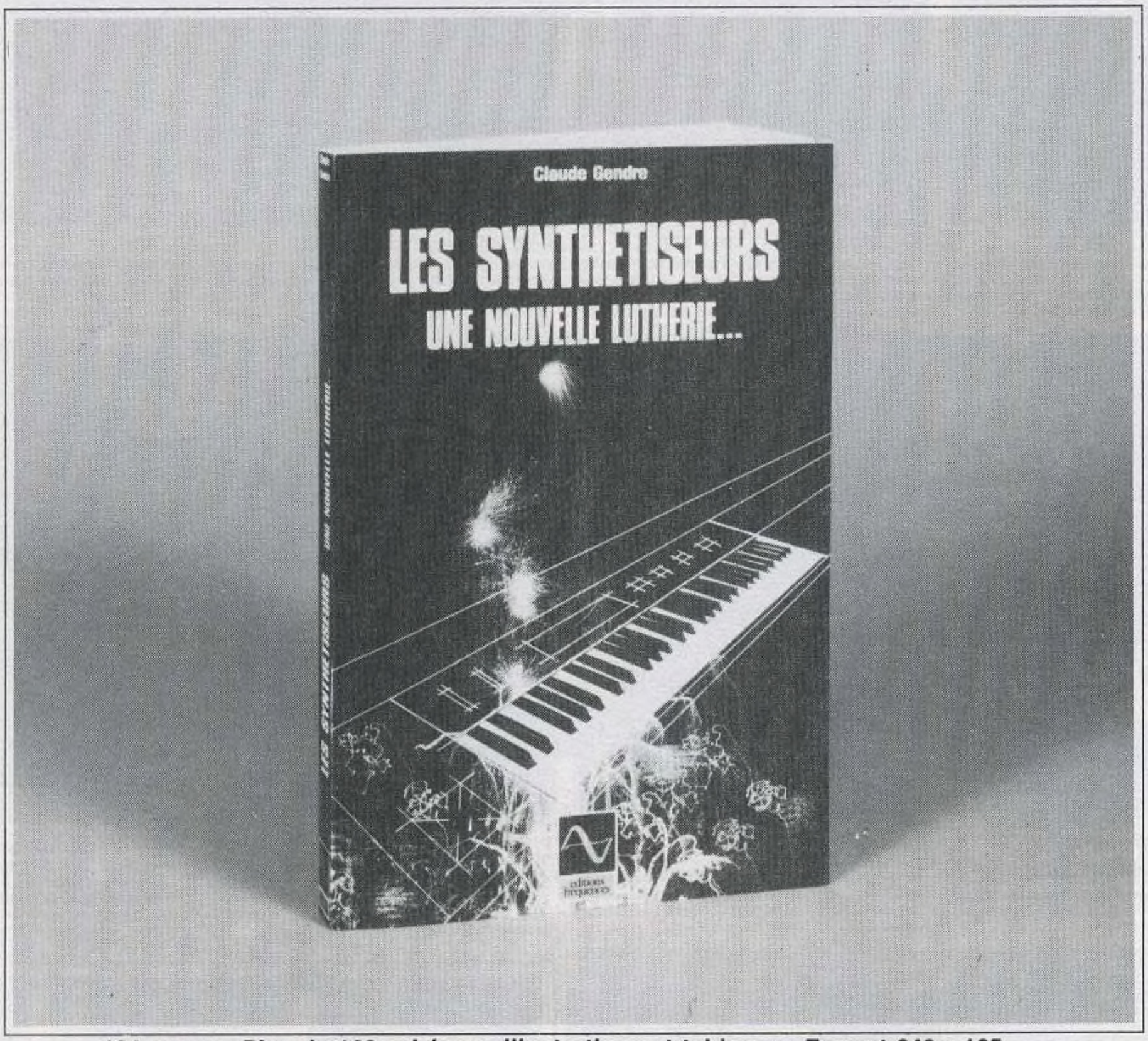
Nom ..... Prénom .....

Adresse .....

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

collection "études"

LES SYNTHÉTISEURS UNE NOUVELLE LUTHERIE...



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

Diffusion auprès des libraires assurée exclusivement par les Editions Eyrolles.

Je désire recevoir l'ouvrage « Les Synthétiseurs» au prix de **150 F** (140 F + 10 F frais de port).

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

Nom ..... Prénom .....

Adresse .....

Code postal ..... Localité .....

C'EST ARRIVE DEMAIN

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

Pour commencer ce mois-ci, une nouvelle qui peut avoir une grande importance. Une société de développement de logiciels graphiques vient de rendre publique une proposition de standard graphique pour micros. Si ceci est suivi d'effets, cela signifierait la première étape vers une normalisation de la micro-informatique, et une diminution de l'anarchie qui préside à la compatibilité (je devrais plutôt dire la non-compatibilité) de fait actuellement. Cette norme consiste en un jeu de routines graphiques, assez complet, qui serait implémenté en ROM, sous forme d'un composant enfichable sur les cartes mères. Quelques développeurs, et non les moindres ont déclaré être intéressés, et Borland a annoncé qu'ils développaient déjà un produit qui suit cette norme. Alors, il est possible que ceci soit sérieusement un bon début pour une refonte plus intelligente du marché. 1986 est une grande année.

L'industrie Sud-Coréenne est-elle en passe de supplanter son homologue japonaise en Occident ? C'est la conclusion d'une analyse de consultants industriels américains. Ils proposent d'illustrer le futur par un simple rapport de coûts :si un produit coûte un dollar à monter aux USA, il coûte 60 cents au Japon, et 10 cents en Corée. Vous me direz donc que la production coréenne ne peut être concurrencée. Mais ce que vous ignorez à ce niveau, c'est que les Coréens sont encore plus malins. Ils ont repris l'équation, et l'ont complétée vers le bas. Au Nigéria, Bangladesh, Ethiopie et autres, le même produit sera monté pour moins de 1 cent. Donc, ils se sont implantés à ces endroits, accueillis à bras ouverts par les gouvernements locaux. La rentabilité est au plus haut pour eux, et la concurrence est impossible. Mais de plus, ils font transiter leurs produits par la Corée pour éviter le label «monté en...» pensant à juste titre que la provenance réelle n'inciterait pas le client potentiel à acheter. Leur stratégie commerciale est bien au point. De plus, un peu comme au Japon, il y a peu de sociétés en Corée, dont cinq vraiment grandes. Ce sont : Lucky-Goldstar, Samsung, Daewoo, Hyundaï et Go-Bei. A elles cinq, elles représentent la quasi-totalité

du potentiel économique du pays, comme Mitsui, Mitsushita, Mitsubishi,... représente la majeure partie du potentiel japonais. Une telle concentration est difficile à battre, car une branche peut perdre de l'argent pendant des années, sans affecter le résultat global du groupe.

Puisque nous venons de parler de Corée, voici une nouvelle qui fera sourire, je l'espère. Le Leading Edge, un ordinateur dessiné en Corée, est livré avec un disque dur de 20 MégaBytes. Or, un revendeur New-Yorkais, étonné de l'interdiction formelle du fabricant de reformatter le disque à quelque moment que ce soit, est passé outre. Le résultat est un splendide disque de... 30 MégaBytes, pour le même prix. L'explication semble être que Leading Edge, manquant de disques 20 MB, et devant la demande en hausse rapide du marché, a dû se fournir en disques 30 MB, sans en avertir les clients, d'où les interdictions de reformater. Les clients, après reformatage pourront savoir si ils sont les heureux possesseurs de tels suppléments gratuits.

Pour tenter de se remettre en piste malgré ses difficultés financières, Commodore a conçu un nouvel Amiga, qui n'a de commun avec celui existant que le nom. Sous le nom de code Ranger (mais le premier nom de code était Rambo, naturellement) se cache un monstre tout simplement. Le processeur est un 68020, le plus puissant de la série des 68000 qui équipent les Amiga, Macintosh et 520ST. C'est un vrai 32 bits, avec une mémoire cache intégrée. Le reste des performances fait aussi le poids. La mémoire sera de 512 Ko en version de base, avec possibilité de l'étendre à 4Mo. L'écran enfin sera un écran bit-map, ayant une résolution de... 1024 par 780 (un peu plus de deux fois la résolution du Mac). La première présentation, pour les intimes de Commodore, sera cette semaine, et l'annonce officielle pour fin Juin, ou début Juillet. Les premières ventes publiques seraient alors pour la fin de l'année. Quant au prix, il atteindra des sommets pour cette société distribuant habituellement des produits grand public. Comptez sur 4000 $, soit en France dans les 60.000 Francs. C'est beaucoup pour des particuliers, mais Commodore veut semble-t-il s'attaquer aux petites sociétés. Apple a mis dix ans pour à peine commencer à les pénétrer, alors bon courage. Espérons que le pari, osé pour une société très mal en point, soit réussi, car cette gamme commence à être très prometteuse. Mais pour commencer sa percée, Commodore annonce que le prix de l'Amiga est baissé de 500 $, ce qui le met à 1200 $. Bien joué, car il devient alors le moins cher de la catégorie des 68000.
Dans un créneau assez proche, Apple semble avoir touché le Jackpot avec le Mac Plus. Alors que les ventes de Mac baissaient de 500 unités par mois, et se situaient vers 10.000 mensuelles, depuis l'apparition du Plus, les ventes sont à plus de 30.000 mensuelles, et montent encore. Ceci est inattendu, car les problèmes ne manquent pas avec le Plus. Les ROMs sont encore buggées, et des problèmes de compatibilité se font jour. Ainsi, le nouveau système d'exploitation, connu sous le nom de HFS (hierarchical filing system), n'est pas totalement compatible, et le logiciel du système semble planter dans certains cas. Ce dernier problème est officiellement résolu sur les nouveaux Plus sortis. Mais le plus gros problème vient de HFS, le nouveau gestionnaire système, qui régit l'ensemble des fichiers et dossiers sous forme de hiérarchie que l'on trouve habituellement sur les plus gros systèmes uniquement. Apple considère qu'il lui faudra au moins trois mois pour rendre HFS compatible entièrement. Par exemple, des programmes Apple, comme Mac Paint et Mac Draw, ne fonctionnent pas parfaitement. Quelle ironie que les programmes développés par Apple ne tournent pas sur les ordinateurs Apple. Enfin, dans le système, version 3.1, des problèmes de démarrage ont été notés. Ces problèmes sont manifestes sur des programmes comme Word, Multiplan, PageMaker, FileMaker, qui marchent mal ou pas du tout. La version 3.2 est censée éliminer ces problèmes, et sera diffusée sous peu, mais ici. Le temps nécessaire pour franciser un système chez Apple est relativement aléatoire, mais toujours long.

Kodak vient de présenter un disque souple de taille 5''1/4, capable de stocker 10 Mo. Les meilleurs actuels de cette taille font actuellement moins de 1 Mo. Le progrès est de taille... mémoire. Le prix est bien sûr un peu élevé, mais sans excès pour une nouveauté, puisque de 1.500 $ approximativement. C'est le prix d'un disque dur de 20 Mo, et il baissera dès que le produit sera diffusé, et il est vraisemblable que le prix descendra alors jusqu'à 700 ou 800 $. C'est alors très compétitif, puisque c'est à peine plus que le prix de lecteurs de capacité habituelle.

Comme vous pouvez le constater dans ces colonnes, les modifications de programmes, révisions, nouvelles versions, sont des événements courants. Pour permettre de suivre le train d'enfer des mises à jour, une société vient de voir le jour, qui propose une liste des numéros de versions les plus récentes pour la plupart des ordinateurs du marché. Cette liste est accessible à tout moment, via un modem, ou par courrier, après acquittement d'une cotisation annuelle de 20 $. Il semble qu'il suffise d'avoir des idées pour faire de l'argent aux USA, car cela marche très bien.

Un sport à la mode ici est de se connecter sur un BBS. Kézaco ?? Un BBS est un Bulletin Board Service. Non, attendez, j'explique. Il s'agit d'un service de courrier électronique, généralement gratuit, à la communication téléphonique près. Vous pouvez vous connecter, lire, envoyer, noter des informations en tout genre. La plupart sont spécialisés, et vous y trou-

verez les cours de la bourse, comment éviter un bug sur tel programme sur tel ordinateur, des jeux... C'est un service. Mais, si vous parvenez à entrer dans les profondeurs de ces messageries, il est possible de ne laisser des informations que pour des utilisateurs très particuliers, comme les fameux pirates de gros systèmes. Ainsi, dans le Texas, un tel service de messageries s'est révélé être truffé d'informations permettant d'accéder les systèmes des banques locales. Pour le découvrir, les services de la police locale ont crée une telle messagerie, il y a plusieurs années. Ils ont ainsi pisté les amoureux de vol électronique. L'affaire fait jaser, car malgré les autorisations dont la police s'est protégée, certains inculpés, via leurs avocats, parlent d'incitations à la malhonnêteté de la part des policiers qui leur ont laissé pénétrer le BBS. Mais c'est la première fois que les BBS sont la cible de la justice américaine.

Le plus beau flop des derniers mois avait été le programme JAZZ, de Lotus. Annoncé à grand renfort de trompettes, le programme s'est révélé être sans intérêt, lent, cher, et peu performant. Fermez le ban. De façon étonnante, il ne s'est pas vendu. (A noter que c'est officiellement la copie pour Macintosh du fameux 1-2-3, qui fait fureur sur IBM PC. Les utilisateurs de ces derniers semblent donc se contenter de peu) Lotus vient d'annoncer une nouvelle version de ce programme. Il devient beaucoup plus rapide, et inclus des performances devenues honnêtes. Le premier coûtait 600 $, donc le suivant aurait dû coûter beaucoup plus cher, disons 1000 $. En fait, il coûte 400 $, du fait de la concurrence du programme EXCEL, à peu près au même prix. Les lois de la concurrence sont bien comprises ici.
Une autre invention récente est le programme télévisé informatique. Il s'agit de discussions, spécialisées ou non, sur des sujets concernant la micro-informatique. Cela va de la vulgarisation aux sujets extrêmement techniques. Même dans la Silicon Valley, le nombre de ces programmes est resté très bas, le succès semblant bouder cette formule ingénieuse. Mais le problème est que, pour les auditeurs potentiels, elle semble surtout ingénieuse pour les invités des émissions plutôt que pour les gens soucieux d'apprendre. En conclusion, le taux d'écoute est très bas, donc la publicité ne suit pas. C'est tout simple.

Depuis quelques jours, la banque à laquelle j'ai la chance de confier mon argent, permet le paiement de factures, le relevé de compte, et le transfert de tout type de sommes directement de chez soi, par utilisation d'une ligne spécialisée, et d'un modem, depuis un ordinateur personnel. C'est très commode, rapide, mais payant, comme tout chèque aux USA.

La presse américaine spécialisée s'est faite l'écho cette semaine des résultats de SMT GOUPIL, société française de production de micros. Il semble que cette firme soit enfin bénéficiaire, et représente 16 % du marché français. Voilà, vue d'ici, une bonne nouvelle. Goupil envisage d'augmenter son capital de 25 %, et son introduction sur le second marché boursier, disent également les échos que j'ai trouvés ici. Souhaitons-lui le succès. Cela nous change de trompettes de Bull, qui perd de l'argent (des contribuables), en vendant du matériel «français», fabriqué au Texas, compatible PC, et dont Bull se contente de repeindre les boites, pour les livrer aux administrations, puisque ces dernières n'ont pas le droit d'acheter des produits non français. Mais la ficelle est tout de même grosse.

Un lecteur de Infoworld, magazine très diffusé, a réagi à un article concernant les versions successives de programmes. Le sujet de l'article, très développé, expliquait que le plus souvent, lorsque vous achetez un programme et qu'une nouvelle version sort, si vous souhaitez avoir cette dernière, il faut l'acheter, et généralement au prix fort. Certains fabricants proposent dans ce cas des prix réduits, mais le plus souvent ce n'est pas vrai. Ce lecteur venait d'acheter Reflex de Borland. Borland est, vous le savez sans doute la société de Philippe Kahn. Peu de temps après, il reçoit une autre disquette de Borland, sans avoir demandé quoi que ce soit. Simplement, Borland lui envoyait la nouvelle version, tout naturellement, sans attendre que ce client ne la lui demande.
Hardi, Philippe.

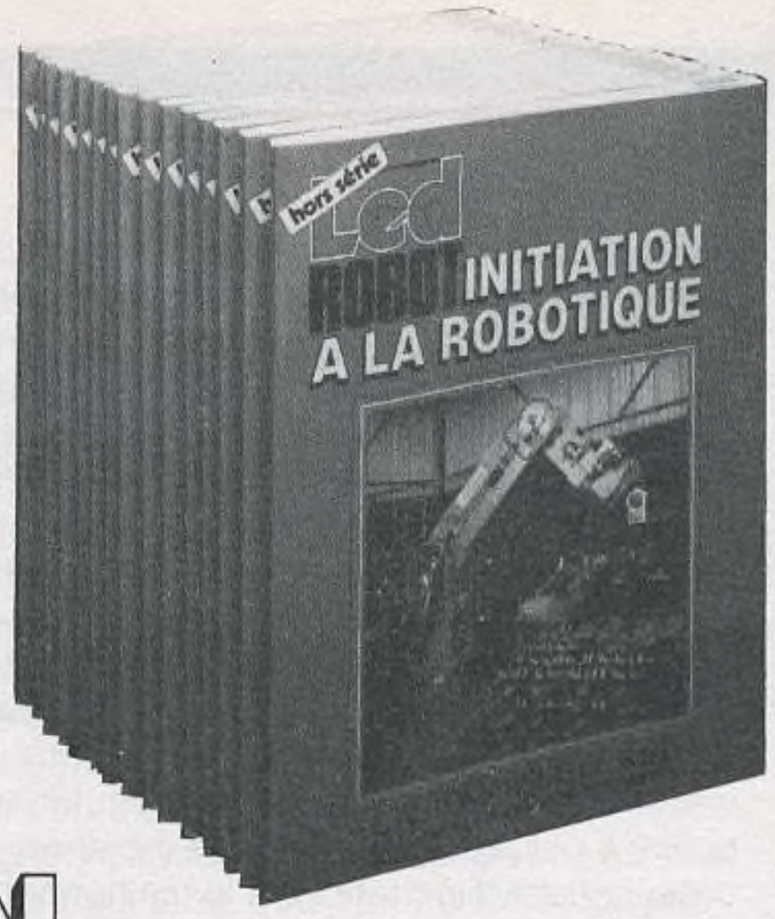
Je viens de recevoir une invitation pour une présentation d'un téléphone un peu particulier. Il s'agit d'un téléphone permettant de transmettre des images stockées sur diapositives, au cours de la conversation. Il semble que ce soit le premier pas vers un téléphone visuel, en temps réel, et non plus comme ici, sur images fixes uniquement. Mais le gros avantage de ce procédé est qu'il utilise des lignes standards, et non des lignes informatiques spécialisées. En revanche, les visiophones risquent de ne pas pouvoir passer sur des lignes à petit débit, et à faible bande passante. Je vous dirai le mois prochain ce que j'en aurai vu.

Une revue mensuelle grand public vient de publier les résultats d'un sondage auprès de ses lecteurs (1 million de lecteurs). 60 % ont un matériel IBM ou compatible, 30 % Apple, 10 % d'autres marques. 30 % ont un disque dur, 30 % ont un modem, 80 % ont une imprimante. 90 % ont un ordinateur chez eux. Il s'agit une fois encore d'une revue grand public. Cela met l'équipement moyen à plus de 5000 $, soit environ 37 000 Francs au cours du jour. Les particuliers ont les reins solides par ici.

Au mois prochain.

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
...IQUES D'AUJOURD'HUI
Led ROBOT

INITIATION A LA ROBOTIQUE

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

Le sommaire : une somme !

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

BON DE COMMANDE

Diffusion auprès des libraires assurée exclusivement par les Editions Eyrolles.

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.

COURS DE GENIE LOGICIEL

De la théorie à la pratique

Charles-Henry Delaleu

CONCLUSION

Le temps passe vite ! Voici arrivé le dernier chapitre de notre cours de génie logiciel (programmation approfondie). Ce cours sera donc celui de la conclusion. De ce fait, nous n'insisterons jamais assez sur la programmation structurée, ainsi que sur les réflexes à maintenir dans l'art d'une programmation efficace et fiable. De même, la production de logiciels fait référence à des règles qu'il ne faut jamais oublier. Un programme n'est jamais complètement fini. Il doit pouvoir évoluer dans le temps, être optimisé, perfectionné. Il n'y a pas de méthode-miracle. Le responsable d'un projet doit, avant tout, avoir une bonne vision d'ensemble. Dans un second temps, il conviendra de simplifier les tâches à l'extrême. Enfin, l'application devra toujours rester ouverte.

Il existe aujourd'hui de nombreux langages permettant de telles règles. D'autres plus anciens ont, dans la majorité des cas, été dotés d'extension de langages leur ouvrant de larges horizons. Dans ce cas, il faudra oublier les raccourcis (GOTO) et faire appel aux ordres plus puissants, plus structurés (CALL, DO).

Lorsqu'on commence un programme, il ne faut pas commencer sur le clavier mais avec un crayon et une feuille de papier. De même, un calcul possède, dans la plupart des cas, un domaine de validité. Une variable doit toujours être déclarée, initialisée, vérifiée, puis validée. Enfin, l'unité centrale ne fonctionne jamais seule, il convient de gérer les périphériques. L'utilisateur doit connaître une anomalie (fil non branché, etc.) avant le déroulement d'un programme, et non pas lors de l'impression des résultats. La première chose à faire : vérifier par auto-tests les entrées-sorties et l'espace-mémoire.

L'art de l'analyse et de la programmation est un sujet à la mode de nos jours. C'est assez logique et prouve que cette dernière atteint l'âge adulte. Si les premiers ordinateurs dignes de cette appellation ont vu le jour grâce aux efforts technologiques faits pendant la dernière guerre mondiale, il convient de noter les négligences qui ont suivi dans le domaine de la programmation. Le véritable essor de l'informatique commence en 1953 avec l'IBM 701. En 1956, le Fortran est créé. Cette même année, M. Perret invente le mot «ordinateur». En 1959, le gouvernement américain, en collaboration avec les utilisateurs, annonce les bases du Cobol. Le premier calculateur à mémoire virtuelle date de 1962... Pendant vingt ans, les programmes seront réalisés avec plus ou moins de rigueur. Les informaticiens seuls détenteurs du «savoir» seront, pendant cette période, verrouillés dans leurs connaissances et leurs systèmes. Ainsi, ils pourront vivre dans un monde protégé et obtenir des avantages substantiels. Les problèmes seront toujours cachés et les excuses seront nombreuses. Il faudra attendre les gros problèmes (ex. : une fusée détruite au décollage pendant la phase ascensionnelle à la N.A.S.A.) pour que certains dirigeants désirent mettre un peu de rigueur dans le beau monde. Vers la fin des années 70, plusieurs rapports et conférences, associés à la naissance de langages de plus en plus structurés et puissants, iront vers une certaine sécurité en programmation. Enfin, dès le début des années 80, le message passe et le sujet est enseigné dans les écoles spécialisées.

La programmation pose de nombreux problèmes :
- le manque de précision dans les cahiers des charges,
- les différences entre les concepts humains et les langages informatiques,
- le peu de résistance des systèmes aux erreurs humaines,
- la difficulté à maîtriser de longs programmes,
- les problèmes liés au mariage langage-machine,
- les problèmes liés à la sécurité et à la protection des informations,
- les problèmes de partage de ressources.

Ecrire un bon programme impose plusieurs données :
- la connaissance du langage utilisé,
- la connaissance du système utilisé,
- l'expérience,
- la maîtrise du sujet à traiter,
- l'application de certaines règles de bases.

L'évaluation des qualités d'un produit logiciel n'est pas évidente. Elle met en œuvre toutes sortes de concepts partant de l'application technique aux problèmes humains en passant par la machine. La validité d'un programme est généralement subjective. Sa clarté est parfois une affaire de goût. Bref, la programmation n'est pas un jeu.

La programmation est, sans aucun doute, l'étape la plus longue et la plus difficile dans un processus d'informatisation. Elle en est aussi la moins aimée. L'enseignement de l'informatique ne doit en aucun cas rester un simple apprentissage d'un ou plusieurs langages. L'essentiel doit être consacré à la logique, à l'architecture, au déroulement des programmes.

L'informatique est un domaine où les progrès techniques sont assez rapides. De plus, le rapport qualité-prix-performances est de plus en plus intéressant. Dès lors, les espoirs sont nombreux et les rêves passent. Qu'il s'agisse de traduction automatique ou d'intelligence artificielle, ces deux sujets sont à la mode depuis fort longtemps maintenant, toutefois leur large démocratisation sur toutes les machines n'est pas encore pour tout de suite. La tendance actuelle serait plutôt de s'attaquer à des modules simples, de les optimiser et d'avancer à petits pas, mais de manière plus sûre.

Il y eut crise du logiciel vers les années 60-65. Cette crise avait pour origine le coût de plus en plus important de la programmation et l'apparition d'échecs spectaculaires dans certains très grands projets. Or, le logiciel est devenu une industrie qu'il faut rentabiliser comme les autres. L'énormité des coûts de l'informatique à cette époque avait de quoi faire réfléchir. Aujourd'hui, le luxe a été balayé par l'immense prolifération de sites informatiques. Le coût de la machine est dépassé par celui du logiciel associé au personnel qui doit en assurer le fonctionnement. Il y a quelques années encore, le coût d'une machine était très nettement supérieur à celui du logiciel. Aujourd'hui la tendance est nettement inversée. Les fabricants de matériels ont très bien su gagner

des points en productivité. Les fabricants de logiciels doivent suivre le même chemin. C'est ainsi que sont apparus sur le marché des ateliers de génie logiciel tels que MAESTRO de PHILIPS ou MULTIPRO de CAP-SOGETI. Ces systèmes, appelés systèmes interactifs de développement de programmes, sont amenés à se développer dans un futur proche. Une ligne d'instruction doit baisser en prix et gagner en fiabilité et en performances.

Combien de programmes ont été modifiés et remodifiés pour devenir des monstres illisibles, dangereux et peu fiables. Ces logiciels deviennent dès lors très onéreux car leur maintenance ne peut être assurée rapidement. C'est en 1972 que Dahl Dijkstra publia son livre : «La programmation structurée». Grâce à ce livre et au langage PL/1 qui donnera naissance à de nombreux langages structurés les choses allaient commencer à bouger. Les industriels et les universitaires commencèrent à emboîter le pas. Ce n'était que le début et il fallait attendre encore quelques années pour que les concepts de base fussent connus d'un plus large auditoire.

LES QUALITES D'UN PROGRAMME

En mécanique, si l'on désire mesurer une pièce, cela ne pose généralement aucun problème. En informatique, il est très difficile de dire si un programme est valide ou non. Le programme satisfait-il exactement le but recherché ? Il est donc nécessaire de démontrer, au sens mathématique du terme, les propriétés d'un programme. Un programme doit être impérativement testé. Ces tests doivent avoir pour but de démontrer la validité des données, des calculs, des résultats. Mais ceci n'est pas suffisant. Il faut aussi effectuer des contre-mesures, vérifier les domaines de validité, contrôler les résistances à l'erreur. Des jeux d'essais doivent être réalisés afin de contrôler, sonder, vérifier chaque programme, chaque sous-programme, chaque bloc ou procédure. Il faut à tout instant se rappeler qu'une série de tests peut servir à montrer la présence d'erreurs, mais en aucun cas leur absence. On comprend ici que la démonstration de validité d'un programme prend toute sa signification. La notion de test doit être prise en compte dès l'écriture du programme. Les tests seront effectués sur chaque étape et chaque stade de la programmation. Ils iront du plus petit bloc d'instructions à l'ensemble de l'application. Ces tests seront faits en tenant compte de l'environnement : tester la validité d'une variable dans une procédure ne sert à rien si cette dernière est en mesure de provoquer des troubles au niveau du sous-programme ou du programme principal. Chaque étage sera donc validé et testé. Les tests devront, dans la mesure du possible, être réalisés par des personnes différentes à celles qui ont programmé le sujet : c'est une obligation pour les grandes applications. Dans le cas des erreurs rencontrées lors de l'exécution d'un programme, il faudra gérer leur détection, leur contrôle et leurs remèdes. Dans le cas d'erreurs à la mise au point d'un programme, il faudra connaître leur type et leur emplacement dans le listing. Cette trace est nécessaire à l'amélioration du programme.

La simple lecture du listing d'un programme permet de se faire une opinion sur sa fiabilité. Certains critères ne trompent pas :

- lisibilité
- expression
- style
- commentaire
- documentation.

Lisibilité

Un programme difficile à comprendre est sûrement truffé d'erreurs. Les tâches ne sont pas découpées, l'architecture est lourde, il y a danger.

Expression

Un programme rempli de GOTO est, sans aucun doute, un programme non modifiable, qui posera de gros problèmes à sa maintenance et à son optimisation.

Style

Un programme très bien structuré, simplifié à l'extrême, sera, sauf erreur, performant.

Commentaire

Un programme comprenant de nombreux commentaires dans son listing sera facile à

lire et à comprendre. Un programme pouvant afficher de nombreux commentaires à l'intention de son utilisateur sera mieux utilisé et les risques d'erreur seront limités.

Documentation

Un programme sans documentation est voué à une mort rapide.

LA SECURITE

La sécurité revêt plusieurs aspects en informatique, il y a :

- les erreurs de données,
- les erreurs de programmation,
- les erreurs machine,
- les cas limites,
- la propriété des informations,
- la résistance aux piratages (réseaux).

LA SOUPLESSE

Un bon programme doit posséder deux caractéristiques très importantes. Il doit pouvoir évoluer dans le temps et pouvoir être implanté sur d'autres machines. Plus un programme aura été réussi, plus il y a de chance pour qu'il puisse être optimisé et qu'il devienne portable d'un calculateur à un autre. Cette notion de portabilité est cruciale pour deux raisons essentielles :

- économie (plus grande diffusion),
- Sécurité (peut être opérationnel sur d'autres machines en cas de panne).

CONCLUSION

Le génie logiciel et les méthodes de programmation sont amenés à se démocratiser à brève échéance. Il y a deux raisons à cela : les chefs de projet, les programmeurs ont donné d'eux pendant longtemps une image galvaudée. Le dumping est fini et les compétences se reconnaissent. Un informaticien n'est pas le détenteur d'un ensemble de trucs et d'astuces. Les combines ne sont pas profitables à long terme. Il y a peu de chances pour que certains responsables informaticiens puissent se présenter (comme dans le passé) comme des spécialistes, alors qu'ils savent à peine bouter un système et changer des disques durs.

Les utilisateurs en informatique se familiarisent avec le jargon informatique. Certains concepts commencent à se démocratiser.

L'évolution de la programmation ne dépend plus que du seul programmeur. La technique du logiciel devient une technique scientifique respectable. Pour cela, le chef de projet, l'analyste ou le programmeur ne doivent plus être de simples individus mais les détenteurs d'un savoir et d'une technique reposant sur des bases solides : le génie logiciel.

GENIE INFORMATIQUE

Les différentes phases de génie informatique sont :

1re étude : — Faisabilité
- Etude d'opportunité
- Faisabilité technique
- Faisabilité économique
- Impact économique et social

2e étude : — Cahier des charges
- Cahier des charges
- Spécifications techniques

3e étude : — Choix
- Etude technique
- Etude prix/performances
- Choix du matériel
- Architecture du système

Elaboration du projet : — Planning
- Devis
- Calendrier
- Organisation

Spécifications à suivre : — Spécifications
- Spécifications techniques
  A. Systèmes
  b. Périphériques c. Interfaces
- Spécifications logiciel
  a. Programmes b. Sous-programmes
  c. Procédures d. Blocs

Achat du matériel : — Achats
- Conception
- Réalisation
- Sous-traitance

Achats des programmes :
- Système d'exploitation
- Utilitaire
- Programmes
  a. Fabriquer (logiciels) b. Acheter (progiciels)

Préparation logiciel — Préparation 1er degré
- Test, vérification
- Jeux d'essais

Préparation matériel :
- Test, vérification
- Jeux d'essais
- Compatibilité machine/logiciel

Intégration : — Préparation 2e degré
- Interprétation des éléments
- Mise au point

Préparation :
- Préparation du site
- Formation des utilisateurs

Recette du système : — Lancement
- Lancement des applications
- Vérification

Maintenance : — Maintenance
- Maintenance machine
- Maintenance logiciel
- Maintenance préventive M et L

Evolutions : — Avenir
- Adjonction ou changement de matériel
- Adjonction ou changement de logiciel

LES CRITERES D'EVALUATION D'UN SYSTEME INFORMATIQUE

Les critères d'évaluation d'un système informatique sont :

La faisabilité en personnel :
- Faut-il embaucher du personnel ?
- Faut-il former les utilisateurs (délais)

Le matériel
- Types de périphériques
- Unités de stockage
- Unités de traitement
- Communication (modem, réseaux, etc.)

Les locaux
- Equipement des salles
- Types des salles

Le système transitoire
- Locaux
- Fichiers manuels
- Modifications de logiciels

Coût d'exploitation et de maintenance
- Matériel en location
- Maintenance du matériel
- Maintenance du logiciel
- Saisie et traitement à façon
- Entretien des locaux
- Coût du personnel d'exploitation et de saisie
- Coût du personnel utilisateur
- Consommables

Retombées financières
- Economie de gestion
- Economie en personnel
- Opportunités économiques
- Diminution des temps de traitement
- Simplification des dossiers

Conditions de travail
- Responsabilité des personnes
- Délégation des décisions
- Autonomie de travail

Critère de marketing
- Meilleur image de marque de la société

Management
- Gestion en temps réel
- Simulation de cas
- Aide à la décision

PROGRAMMATION STRUCTUREE

Le programme-maître

- Test de la machine
- Test des périphériques
- Configuration de base
- Mise à zéro des variables

Menu principal
appel des sous-programmes

- Divers
- Retour au système

} Corps du programme

Sous-programme nº 1

- Ensemble des déclarations
- Mise à zéro des variables

Corps du sous-programme nº 1

Appel des procédures

Retour au programme-maître

Sous-programme nº 2

- Ensemble des déclarations
- Mise à zéro des variables

Corps du sous-programme nº 2

Appel des procédures

Retour au programme-maître

Sous-programme nº n

- Ensemble des déclarations
- Mise à zéro des variables

Corps du sous-programme nº n

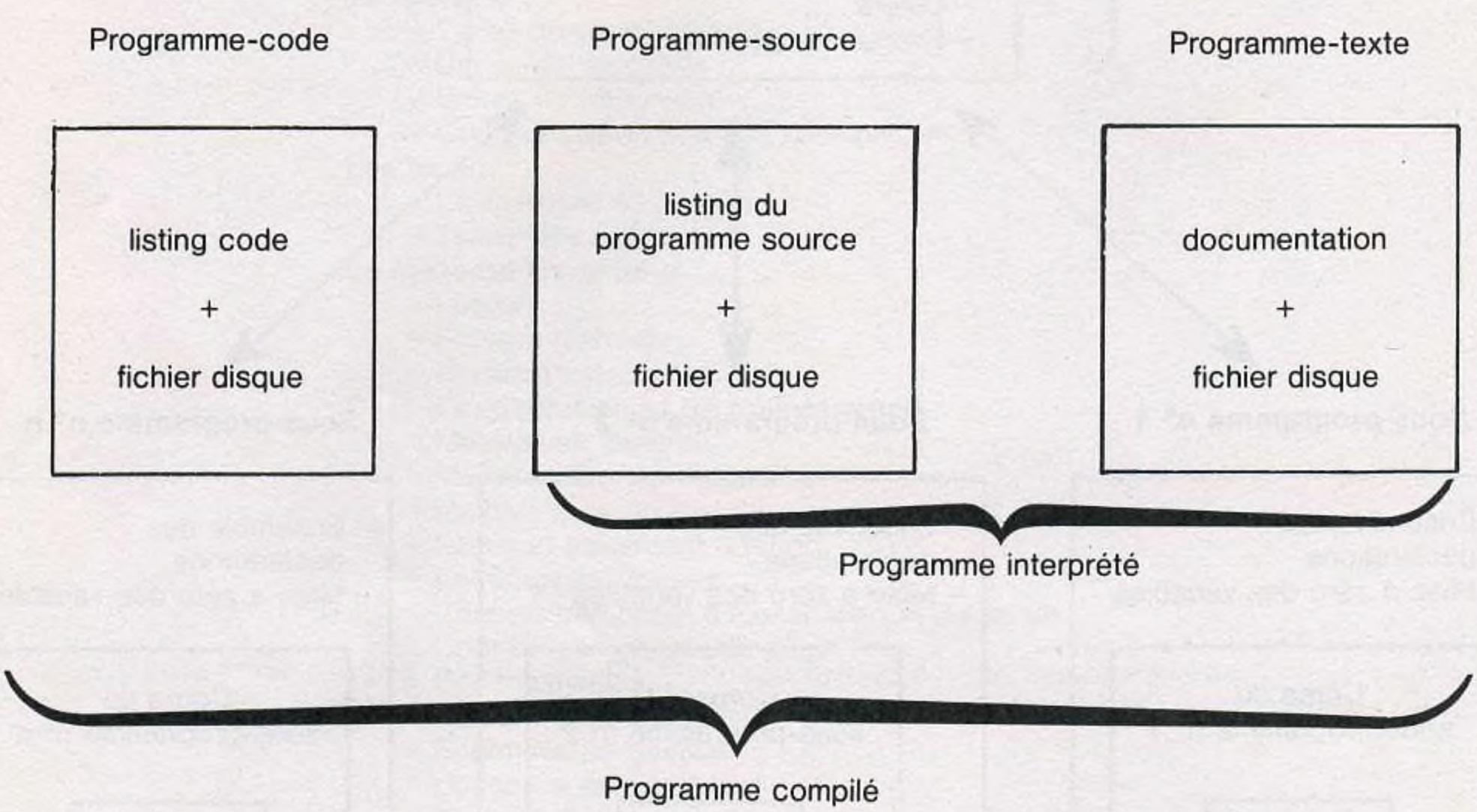
Appel des procédures

Retour au programme-maître

Banque ou bibliothèques
des procédures

PROGRAMMATION STRUCTUREE

Pour tous programmes, sous-programmes, procédures, il est nécessaire de réaliser une documentation très complète. Cette documentation sera double. Elle sera réalisée sous forme de fichiers en mémoire de masse dans le système informatique et couchée sur papier classé.



Le programme-source :
Il s'agit du programme tel qu'il est écrit de manière conventionnelle. Chaque partie sera précédée d'un titre écrit sous forme de remarque qui n'apparaîtra pas à l'écran.

Le programme-texte :
Il s'agit de la documentation concernant les algorithmes du programme ainsi que son architecture. Il sera réalisé dans le même ordre que le programme-source. Il est très documenté, très détaillé.

Le programme-code :
Certains compilateurs autorisent un listing du programme-code sous forme d'assembleur. Dans ce cas, il faudra insérer dans la documentation un listing de ce programme-code.

LES MENUS

LES MENUS :
L'utilisation de menus présente de nombreux avantages :

- Programme structuré
- Facilité d'utilisation
- Maintenance aisée
- Lecture facile.

PROGRAMMATION STRUCTUREE
Un logiciel bien structuré sera réalisé à l'aide d'un programme et de sous-programmes. Chacun commencera par un ensemble de déclarations, de tests. Ceci sera suivi par un menu.

FACILITE D'UTILISATION
Pour l'utilisateur, le menu offre une compréhension aisée du déroulement du programme. Il pourra être réalisé soit par clés de fonctions, soit par souris, soit par données simples.

MAINTENANCE AISEE
L'emploi de menus amène généralement un découpage des tâches. C'est un premier pas vers la structuration. Si les blocs d'instructions sont bien agencés, la maintenance n'en sera que plus facile.

LECTURE FACILE
Si la lecture du programme est facilitée par l'utilisation de menus, elle n'en est pas moins obligatoire pour l'utilisateur du programme. De ce fait, le concepteur comme l'utilisateur auront tout à gagner des menus.

ALGORITHME POUR UN MENU : ex. la comptabilité.

- Menu général
  Afficher ligne 5 colonne 10 «Menu général de comptabilité»
  Afficher ligne 10 colonne 12 «Comptabilité générale → 1»
  Afficher ligne 12 colonne 12 «Journaux → 2»
  Afficher ligne 14 colonne 12 «Lettrage des comptes → 3»
  Afficher ligne 16 colonne 12 «Autres fonctions → ENTER»
- ENTER CHOIX
  CHOIX
    CHOIX = «1»
    Appeler sous-programme compta. générale
    CHOIX = «2»
    Appeler sous-programme journaux
    CHOIX = «3»
    Appeler sous-programme lettrage
    CHOIX = « »
    FIN
  FIN DE CHOIX

LES PROCEDURES

Bien que nous ayons déjà abordé les procédures, il est bon d'en approfondir le principe. Une procédure est un ensemble d'instructions ayant pour but de réaliser un algorithme, une tâche, une opération déterminée. Le concept de base consiste à réaliser un ensemble de procédures qui pourra être utilisé comme un Méccano afin de construire, avec les mêmes modules de base, différentes applications. Il y a quelques années, un nouveau programme était toujours réalisé de manière linéaire et le concepteur créait le tout. Aujourdhui pour des gains de temps, mais aussi pour des problèmes de qualité, il n'est plus question d'écrire un programme de A jusqu'à Z. Chaque nouvelle application doit générer des procédures qui viendront enrichir les bibliothèques de procédures pour être ultérieurement employée par d'autres programmes. De ce fait, chaque nouveau programme est en fait composé d'une architecture nouvelle élaborée autour d'une majorité de procédures déjà réalisées et fiables.
Toutes ces procédures seront intégrées dans des bibliothèques de procédures :

- bibliothèque de procédures de tris
- bibliothèque de procédures de dessins
- bibliothèque de procédures de calculs
- bibliothèque de procédures d'entrées-sorties, etc.

Pour un programme donné, le concepteur n'aura qu'à réaliser un ensemble de procédures formant un tout à partir des bases déjà existantes. Dès lors, le programme sera très facile à écrire et à entretenir. Il n'est plus question de réaliser un programme sous forme de listing, mais plutôt sous la forme d'une suite réduite d'appels de procédure.
Exemple d'un programme utilisant des procédures :

```
Sous-programme MACHIN
   Déclaration des constantes, des variables
   Appel de la bibliothèque des procédures TRUC
      Faire procédure MASQUE de SAISIE
      Faire procédure CALCUL N° 1
      Faire procédure IMPRESSION
   Fermer la bibliothèque de procédure TRUC.
FIN du sous-programme MACHIN (retour au programme principal).
```

NOTA : En cas de modification du sous-programme MACHIN, il n'y a rien de plus facile. Si on désire modifier le calcul, il suffit de remplacer la ligne :

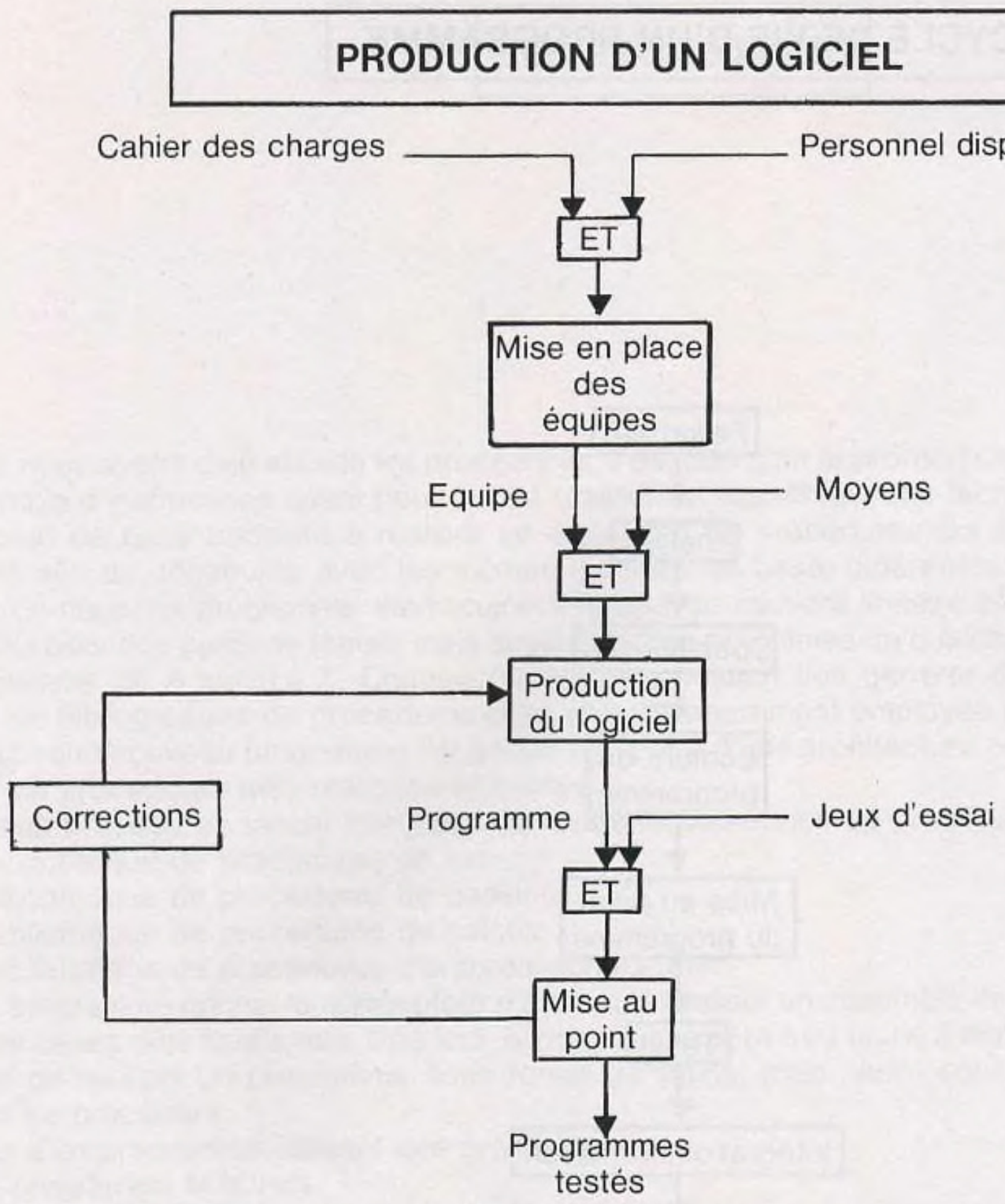
```
   Faire procédure CALCUL N° 1
```

par :

```
   Faire procédure CALCUL N° 2.
```

CYCLE DE VIE D'UN PROGRAMME

Le cycle de vie d'un programme est un ensemble d'étapes dont il ne faut sauter aucune phase. Il ne faut jamais oublier que la phase d'écriture du programme doit générer une plus value dans les bibliothèques d'algorithmes, de procédure, de tâches, etc. Un programme nouveau doit toujours être réalisé à partir d'outils (programmes) existant dans leur majorité, l'essentiel du travail étant d'agencer le tout.



La production d'un logiciel fait appel à trois grands axes :

LE TRAITEMENT

La qualité des traitements dépend :

1. de la politique de sécurité
   1.1. Point de reprise
   1.2. Contrôle
   1.3. Accès sélectifs
2. de la présentation des résultats
   2.1. Masque
   2.2. Formulaire, etc.
3. de la gestion des erreurs
4. de la précision des résultats
5. des algorithmes choisis
6. des tests et de la contre-mesure
7. des jeux d'essais en vitesse
8. des jeux d'essais en capacité
9. de l'arborescence et de la structure, etc.

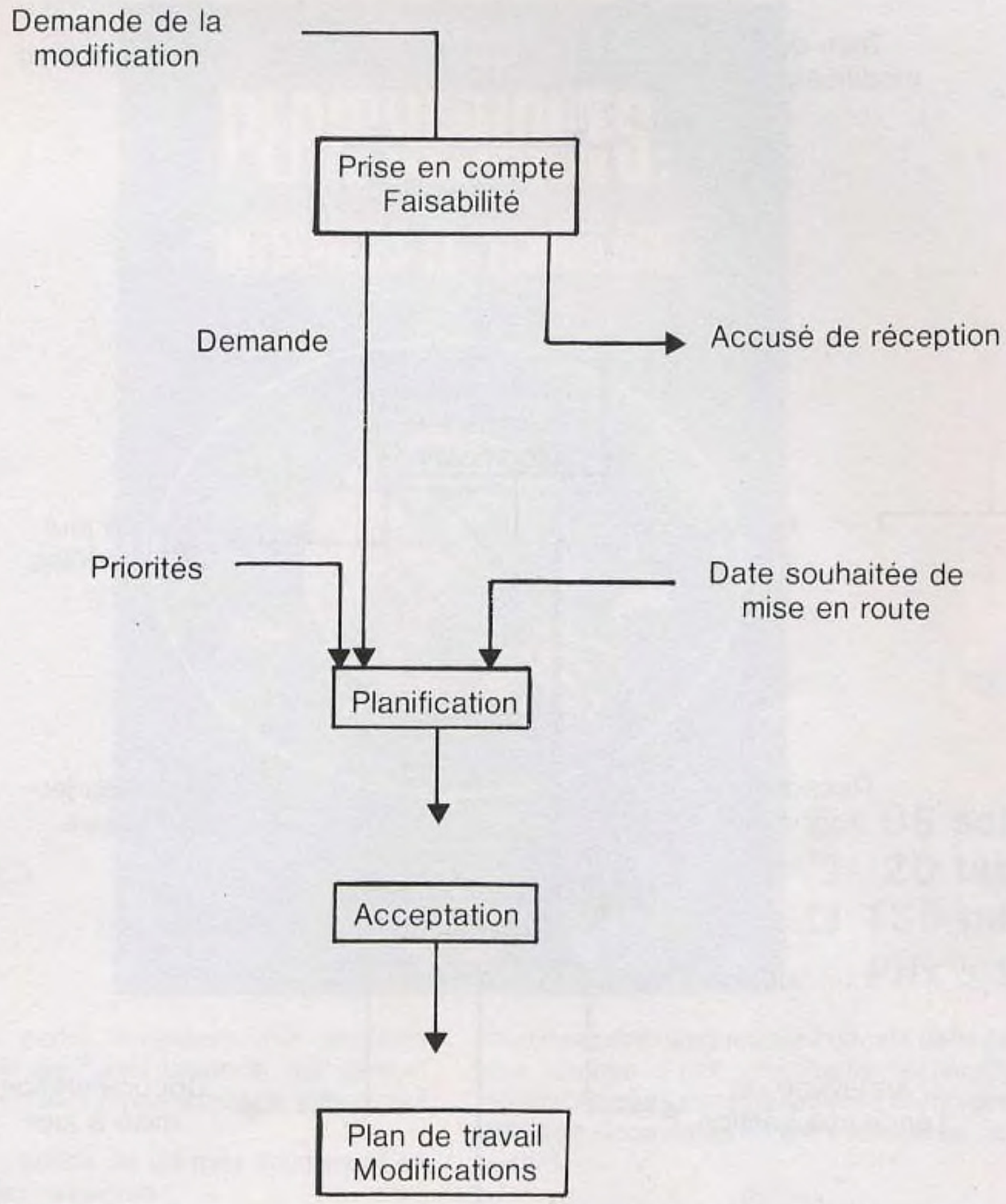
LES DONNEES

Les données seront déclarées, typées et classées.

LA DOCUMENTATION

La documentation devra être fournie, précise et mise à jour. Elle partira du cahier des charges à la réalisation finale.

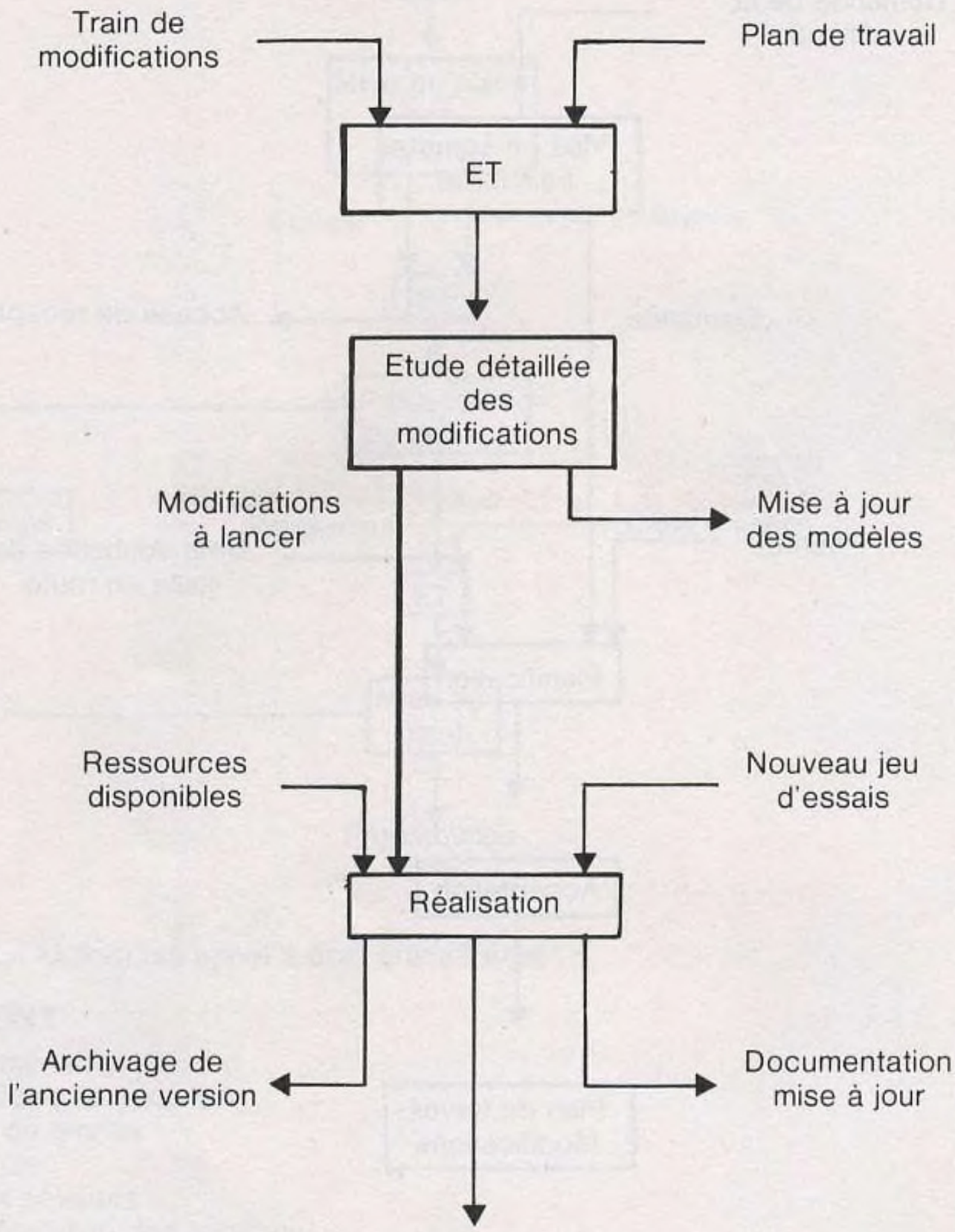
MODIFICATION D'UN LOGICIEL DEMANDE



Les modifications sont toujours nécessaires. Elles sont le reflet d'un produit vivant dont les caractéristiques sont, sans cesse, optimisées. Toutefois, il est nécessaire de les maîtriser. Elles ne doivent en aucun cas mettre en danger le programme.
Les choix portant sur les modifications reposent sur les critères suivants :

- Faisabilité (est-ce possible, réalisable ?)
- Priorités (nécessité)
- Coût (en temps, en matériel, en personnel).

MODIFICATION D'UN LOGICIEL
REALISATION



La mise en œuvre de modifications en logiciel revient à répéter une grande partie des phases de la réalisation du programme d'origine. Dans certains cas des modifications importantes peuvent entraîner la fabrication d'un nouveau programme. Il est préférable de bâtir du neuf, plutôt que de vouloir faire du neuf avec de l'ancien.

TOUT SUR LES PÉRIPHÉRIQUES

- ☐ 85 schémas
- ☐ 20 tableaux
- ☐ 136 pages

Prix : 150 F

Les périphériques font partie intégrante d'un système informatique. En parallèle de l'unité centrale, qui gère et synchronise l'ensemble, ils sont reponsables de différentes fonctions comme :

- la mémoire de masse : unités de disques souples et de disques durs, lecteur de cassettes ;
- le dialogue avec l'utilisateur : clavier, écran vidéo, imprimante ;
- les télécommunications : modem.

Tous ces périphériques sont décrits dans cet ouvrage avec, pour chacun d'eux, une partie technologie (principe de fonctionnement, caractéristiques techniques) et une partie interface (coupleurs d'entrées-sorties, connecteurs de liaison).

Dans chaque grande catégorie (mémoire, imprimante), une analyse comparative des différents produits existants est effectuée.

Philippe Faugeras, docteur ingénieur en électronique, est responsable matériel dans une entreprise d'informatique traitant des réseaux de P.C. Au préalable, il a acquis son expérience en travaillant sur des sujets comme les automatismes et les télécommunications dans deux grandes sociétés françaises (Bull, CGE). Philippe Faugeras est l'auteur d'un premier ouvrage «L'électronique des micro-ordinateurs» paru aux Editions Fréquences.

BON DE COMMANDE

Diffusion auprès des libraires assurée exclusivement par les Editions Eyrolles.

Je désire recevoir l'ouvrage **«Périphériques interfaces et technologie»** au prix de **160 F** (150 F + 10 F de port).

Nom .....

Adresse .....

.....

Règlement ci-joint :
Par chèque bancaire ☐ par chèque postal ☐ par mandat ☐

PETITES ANNONCES

Cherche possesseur de MSX pour échanger des programmes, j'en possède + de 180, vends aussi un MSX (PMC 28 S 32 K) + ext. 64 K + joystick + une centaine de programmes + manuels. Faire offre à Stéphane Le Coq 16, rue Fromont 37000 Tours. Tél. (16) 47.37.50.99 après 18 h 45.

Vends ZX-81 + 16 K + clavier ABS + livre + Mg k7 : 900 F. TV jeu N60 avec écran N.B. incorporé + 2 manettes + 1 cassettes jeu, 1 cassette programmation avec notice et schémas : 950 F. Tél. (1) 43.57.42.33 le soir.

Achète pour TRS-80 modèle 1, prgs Visicalc, Scripsit, Accel 3/4, New-Dos 8* version 2. Tél. (1) 43.70.06.61 aprèS 18 heures.

Vends livres sur TI et revues. List nos 1, 2, 3, 4, 5, 6, 8, 9. Votre ordinateur : nos 2 à 14. SVM : nos 1 « 24. Hebdo : nos 12, 13, 15 à 17, 57 à 59, 61, 64, 66 à 69, 77 et 78, 80, 84, 94 à 97. M. Laval Daniel rue de la Ressence 06220 Vallauris. Tél. (16) 93.64.26.12 après 20 h ou (16) 93.62.26.66 h.b.

Echangerai amicalement tous programmes pour Amstrad CPC 6128 (disquette). Tél. 35.71.92.97 Rouen.

Vends Micro XT 512 K + 1 disque dur 20 Mo + 2 lecteurs + 1 carte multifonctions + 1 carte vidéo N/B + 1 moniteur ambre : 18 000 F. Erwane Hoecke : 42.62.59.43.

Vends cartouches standard MSX Hyper Olympic 2 : 100 F, Hyper Sport 1 : 100 F, joystick MSX Canon : 100 F, moniteur Philips vert interface son intégré : 700 F. Tél. 42.20.48.08 le soir.

Vds Led-Micro hors série nº 1A 22 : 130 F, livres IBM PC (3) : 100 F, trucs ZX81 : 20 F, assembleur du Spectrum : 20 F, 20 diskettes SF/DD 5'' 1/4 (Fuji/Flexette) : 120 F. M. Fis. Tél. 38.76.94.20.

Achète ou échange : mémoire 64 k pour C.B.M. 8032. Fantino Marcel 9, rue du Chateau 40800 Ain s/Adour.

Vends Apple 2C 128 k, 80 col. + moniteur écran vert avec stand + lecteur externe + carte Chat Mauve + joystick + souris + sac de transport, 8 mois de garantie, + nombreux logiciels (100) : 13 000 F. Perret Gérard 14, rue du Champ de Manœuvre 67200 Strasbourg. Tél. 88.28.24.38.

Le Parc studios 8, 16, 24 pistes locaux de répétition, location matériel-instruments, réparation, maintenance, plateau vidéo enregistrement son-image synchro. Tél. 45.21.45.07.

Vends Ti-99 + manettes de jeux + cartouches (minimémoire, alligator-Mix. Tombstone City, addition-soustraction) + livres de programmes. Le tout : 1 500 F. Vends cartouche Logo pour M05 avec livres d'accompagnement : 500 F. Tél. 38.53.56.59 (Loiret).

BON DE COMMANDE

Pour compléter votre collection de Led-Micro

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

Je désire le nº ☐

1 2 3 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

(cocher le ou les nos désirés).

AU PRIX DE 20 F par numéro (port compris).

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

NOM ..... Prénom .....

Adresse : .....

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

Bulletin d'Abonnement

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

Nom ..... Prénom .....

Nº ..... Rue .....

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

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

VOYAGE AU CŒUR DES MICRO-ORDINATEURS

dans la COLLECTION «ETUDES» aux éditions fréquences

une véritable schémathèque

- 128 pages
- 101 schémas
- 34 tableaux

Prix : 150 F

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

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

En vente chez votre libraire et aux Editions Fréquences

BON DE COMMANDE

Je désire recevoir l'ouvrage **L'électronique des micro-ordinateurs»** au prix de **160 F** (150 F + 10 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]