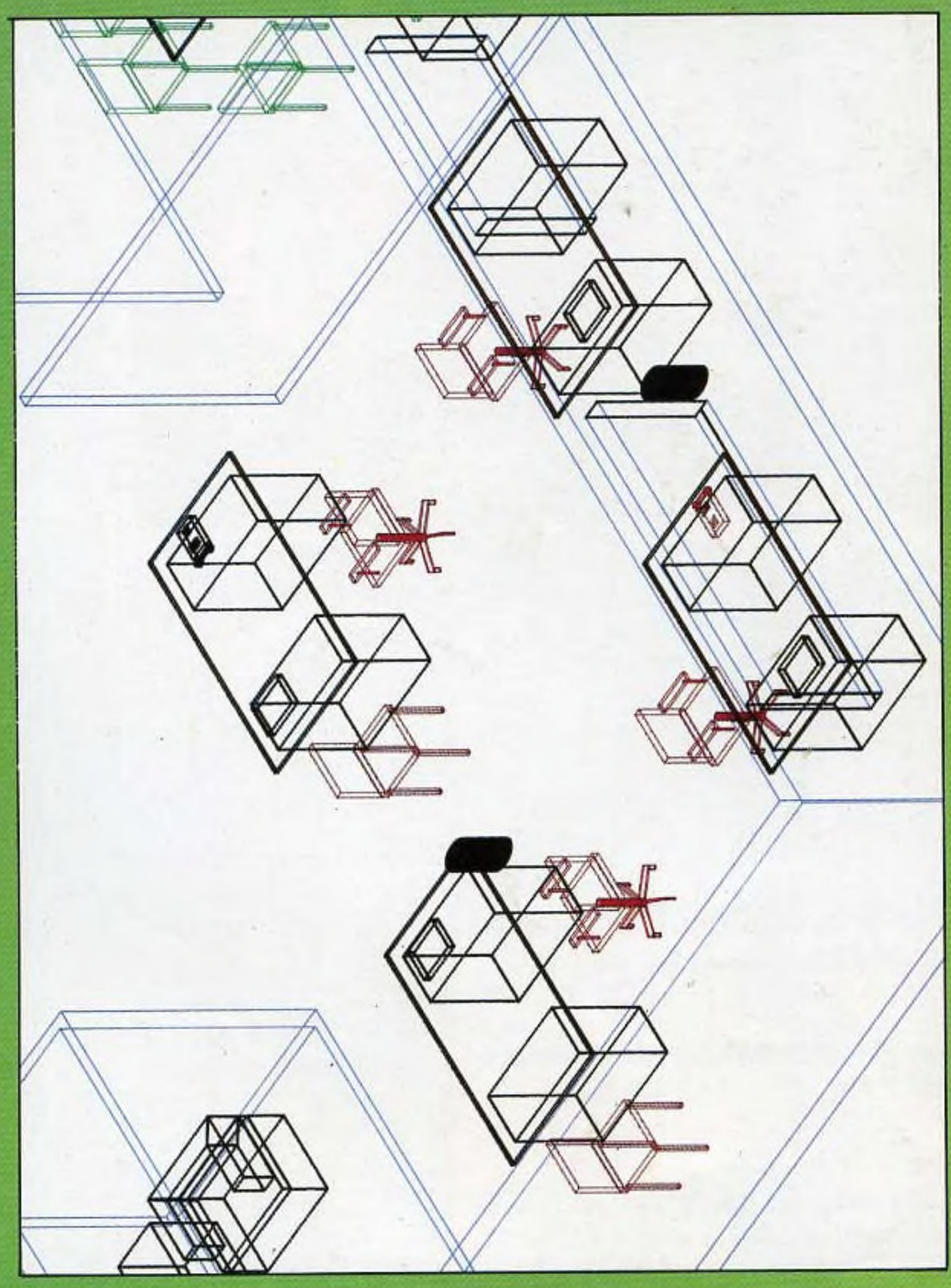
hors série

Led MICRO

LOISIR[S] ET [TECH]NIQUES D'AUJOURD'HUI

PROGRAMMATION
COURS 2ème CYCLE



ISSN 0757-6889

COURS
N°38
Suite
2e cycle
N°18

COURS DE PASCAL
la notion de récursivité

COURS DE PROGRAMMATION APPROFONDIE :
éditeur de texte

AUTOCAD
D.A.O.

M 1988 - 38 - 18,00 F
3791988018008 00380

MENSUEL MARS 1987 - BELGIQUE 139 FB/CANADA 3,75 $/SUISSE 6,75 FS

VOYAGE AU CŒUR DES MICRO-ORDINATEURS

dans la COLLECTION «ETUDES» aux éditions fréquences

une véritable schémathèque

- 128 pages
- 101 schémas
- 34 tableaux

Prix : 150 F

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

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

En vente chez votre libraire et aux Editions Fréquences

BON DE COMMANDE

Je désire recevoir l'ouvrage **L'électronique des micro-ordinateurs»** au prix de **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.

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

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

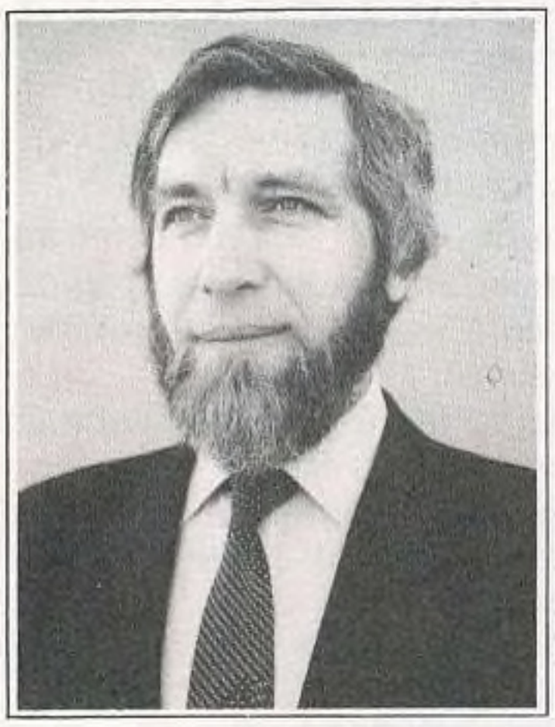
COURS
N°38
Suite
2e cycle
N°18

MARS 87

NOTRE COUVERTURE : AutoCad est un outil tout à fait indiqué pour les architectes d'intérieur, en effet grâce à ses fonctions de zoom et de rotation il est possible d'optimiser l'agencement.

Electronique digitale ?

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



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

En vente chez votre libraire et aux Editions Fréquences

Bon de commande

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

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

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

COURS DE PASCAL

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

**Il est temps d'aborder les sous-programmes et – aussi et surtout – la notion de récursivité qui fait toute la force du PASCAL.
Vous le verrez d'abord de manière théorique puis dans toutes les formes d'applications pratiques.**

COURS Nº 9

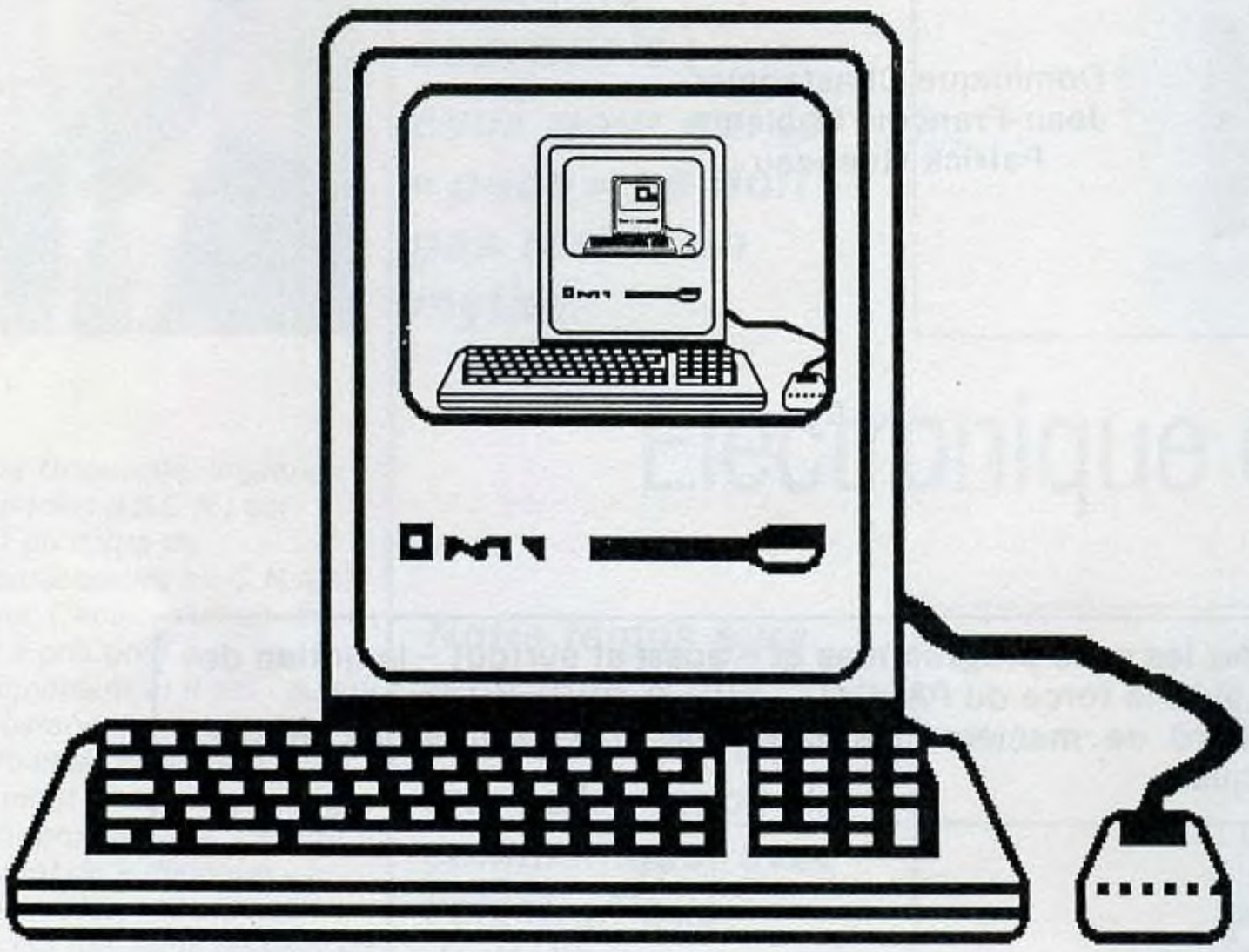
PLAN DU COURS

1. INTRODUCTION

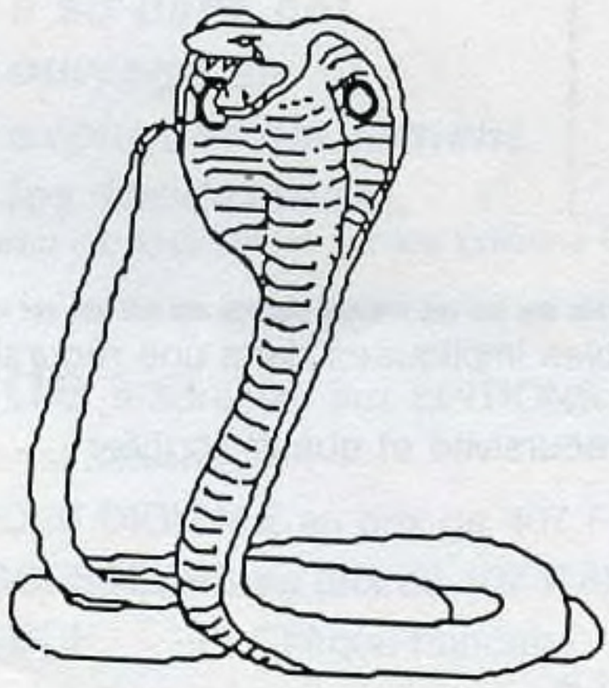
La récursivité est une des notions fondamentales en informatique. Elle donne aux langages qui en bénéficient une puissance inégalable. Mais cela se traduit par une complexité de la gestion interne du programme. Il n'est pas, heureusement, indispensable de maîtriser cette dernière pour pouvoir utiliser la récursivité, le compilateur prenant en charge l'essentiel.

2. DIS, QU'EST-CE QUE C'EST ?

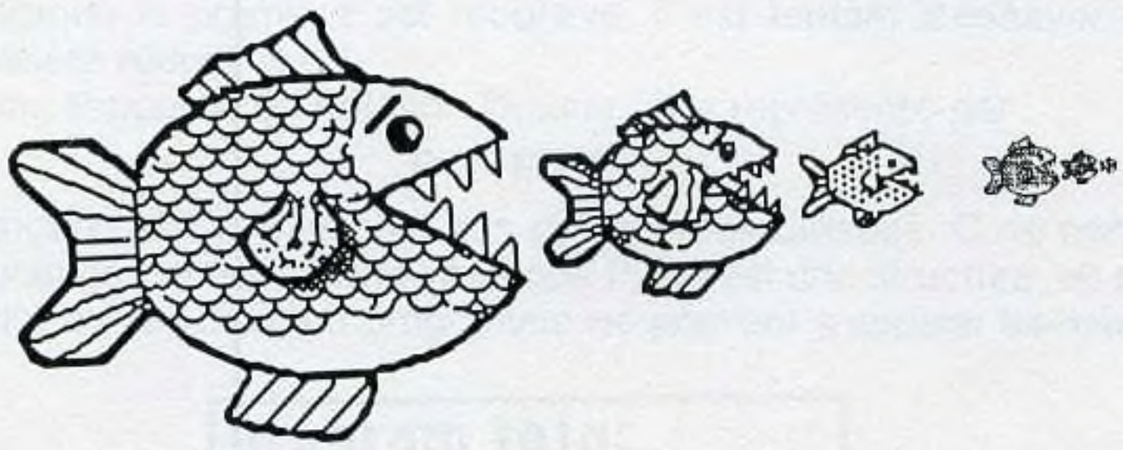
La récursivité, ou récursion, est la possibilité de reprendre un thème à l'intérieur de ce thème lui-même. Comment cela, ce n'est pas clair ? Bon, alors, voici deux dessins pour comprendre plus simplement. Le Macintosh a, sur son écran, un Macintosh qui, lui aussi, sur son écran, a un Macintosh,...



Cela ressemble au serpent qui se mord la queue. En fait, c'est vraiment similaire.



Cela pourrait aussi être vu comme un poisson qui en utilise un autre, qui en utilise un autre,...



3. UNE DEFINITION RIGOUREUSE

Un objet est récursif s'il est défini, partiellement ou en totalité, à partir de lui-même. Il arrive assez fréquemment que ce type d'objet soit créé souvent à des fins publicitaires. Par exemple, un écran télé, qui contient l'écran,...
En mathématiques, la récursion est un outil de première importance, dont voici quelques exemples, que l'on retrouve d'ailleurs dans la structure même de Pascal :

3.1. Les nombres entiers

Est un nombre entier :

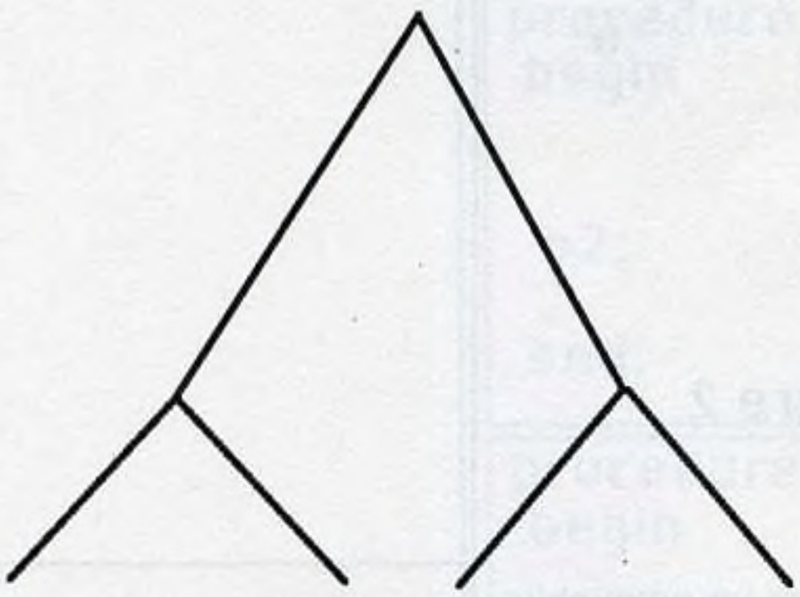
- 0
- le successeur d'un nombre entier (successeur au sens de k + 1).

On voit que tout est défini à partir de 0, puis que l'on remonte la chaîne jusqu'au nombre désiré. Notons que la commande SUCC existe en Pascal et permet exactement ce type de logique. Son inverse PRED permet de redescendre la chaîne.

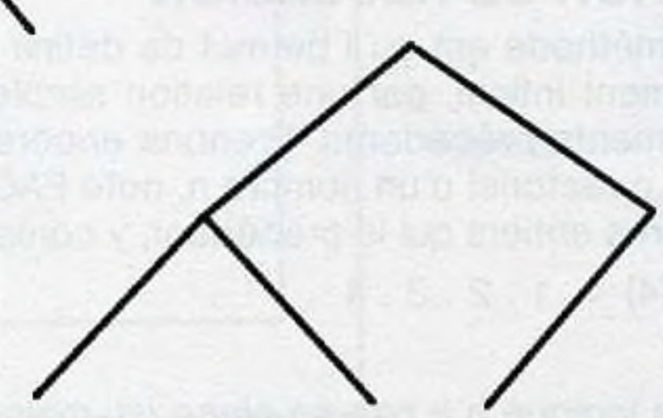
3.2. Les arbres

Un arbre est :

- l'arbre vide
- si t1 et t2 sont des arbres, t1 uni à/ou t2 est un arbre.



arbre 1



arbre 2

L'union peut être définie de manière intuitive par accrochage de l'un des arbres à l'autre (fig. 1) ou par création d'une liaison commune (fig. 2) :

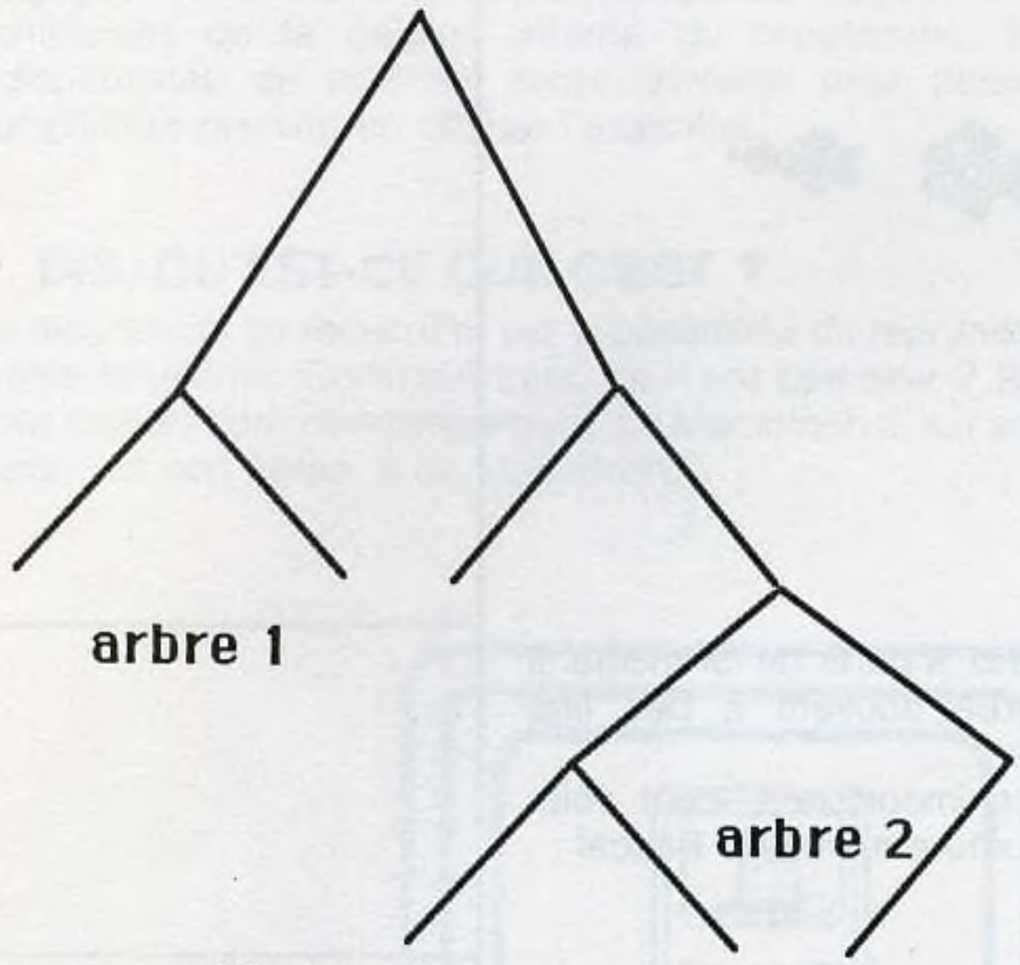


Figure 1

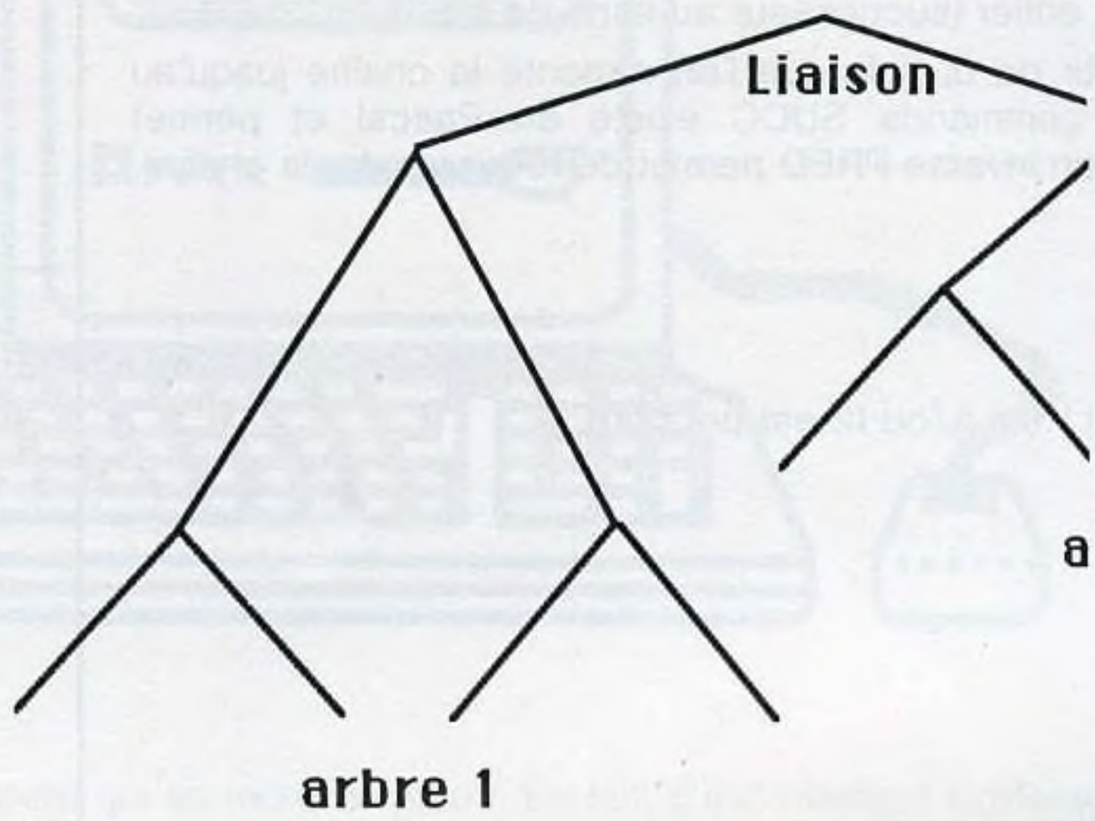


Figure 2

4. GENERALITE SUR LA NOTION DE RECURSION

La grande puissance de ce type de méthode est qu'il permet de définir un ensemble de longueur quelconque, éventuellement infinie, par une relation simple, récurrente, c'est-à-dire définie par le ou les éléments précédents. Prenons encore un exemple mathématique : la fonction factoriel. Le factoriel d'un nombre n, noté FACT(n), est, par définition, le produit de tous les nombres entiers qui le précèdent, y compris lui-même.

FACT(4) = 1 . 2 . 3 . 4

Il est alors logique de le définir par :

- FACT(0) = 1 — La logique n'a pas sa place ici, mais c'est une convention
- FACT(n) = FACT(n − 1) . n

De la même manière, un nombre infini (en théorie) d'exécution d'un langage peut être défini par un nombre fini de commandes, par simplification récursive des répétitions. Notons que ces deux notions, l'algorithme mathématique et le programme, sont très liées, et que lorsque la première est récursive, il est tentant d'essayer de faire la seconde de manière récursive.
Un tel programme P pourra, en notation logique, être représenté par :

$$P = f(C, P)$$

où f est une fonction et C l'ensemble des commandes utilisées. C ne contient donc pas P, cela n'aurait pas de sens. On voit ici que P, qui est une structure, ne pourra être qu'une procédure ou fonction, un programme ne pouvant s'appeler lui-même.

```
program toto;

  procedure p1;
   begin
    .
    .
    p1;
    .
   end;
```

Il peut y avoir récursivité de deux manières différentes :

- directe, comme dans le cas de la figure précédente, où p1 s'appelle,
- indirecte, dans le cas où une procédure p1 appelle p2, qui appelle p1.

```
program toto;

  procedure p1;
   begin
    .
    .
    p2;
    .
   end;

  procedure p2;
   begin
    .
    .
    p1;
    .
   end;
```

Il peut donc arriver que la récursivité soit partiellement cachée, et peu visible.

A l'attention des lecteurs vigilants : la procédure p1 appelle p2, déclarée après p1, ce qui semble en contradiction avec la règle d'or du Pascal, qui dit que toute structure doit être déclarée «avant» d'être utilisée. Or ici, p2 est déclarée après p1. Pour s'en sortir, un artifice de Pascal permet de réaliser cette infraction à la règle. Il s'agit de la commande FORWARD. Cette commande indique qu'une structure, utilisée à un certain niveau, est en fait déclarée entièrement plus loin. On marque ainsi qu'il ne s'agit pas d'une erreur, mais d'un acte volontaire. En effet, dans l'exemple précédent, que ce soit p1 ou p2 qui est déclarée en premier, sans cette possibilité, il y aurait toujours un problème. Donc, le programme correct est :

```
program toto;

procedure p2 : forward;

 procedure p1; .
  begin
   .
   .
   p2;
   .
  end;

 procedure p2;
  begin
   .
   .
   p1;
   .
  end;
```

Si une procédure récursive, ou indirectement récursive, gère des variables locales ou globales, il est bon de savoir comment ces variables sont mémorisées lors d'un appel récursif. Prenons un premier exemple, très simple :

```
program  toto;
 var    a, b, c : integer;

 function p1 (i, j : integer) : integer;
  var k : integer;

begin
 if j = 0 then
  k := i
 else
  k := p1(i, j - 1) + 1;
 p1 := k;
end;
begin
  readln(a, b);
  c := p1(a, b);
  writeln(c);
end.
```

Ce programme réalise la somme de deux entiers. Il y a plus simple pour ce calcul, mais nous pourrons suivre le déroulement du programme.
Supposons que les valeurs fournies soient 2 et 2. Voici le déroulement :

- entrée dans p1, i = 2, j = 2, k = ? (k n'étant pas initialisée, sa valeur est inconnue).
- j#0, donc la partie dans le ELSE est exécutée.
  - i et j sont sauvées pour la récurrence. Cela fait deux piles, une par variable.
  - appel à p1, avec les valeurs i et j − 1, soit 2 et 1 . k est passée avec la valeur inconnue déjà justifiée.
  - entrée dans p1, i = 2, j = 1, k = ? (k n'étant pas initialisée,...)
  - j#0, donc la partie dans le ELSE est exécutée :
    - Empilement de chaque variable sur sa pile,
    - appel à p1, avec les valeurs i et j − 1, soit 2 et 0,
    - entrée dans p1, i = 2, j = 0, k = ? (k n'étant pas initialisée,...),
    - j = 0, donc k : = i, soit k : = 2,
    - il n'y a plus d'appel à p1, donc on a un retour à la structure d'appel précédente.
  - k : = k + 1, donc k : = 3 ;
  - pas d'autres appels, donc retour à la procédure appelante.
- k : = k + 1, donc k : = 4.
- Sortie de p1, retour au programme principal, avec c : = 4.

Ici, les valeurs empilées n'ont pas vraiment servi, i et j étant vraiment locales. Mais il est possible de se servir, à chaque niveau de variables stockées. Voici un exemple, proche du précédent :

```
program  toto;
 var
  a,  b,  c  :  integer;

 function  p1  (i,  j  :  integer)  :  integer;
  var
    k  :  integer;

  begin
   if  j  =  0  then
    k  :=  i
   else
    k  :=  p1(i,  j  -  1)  +  j;
   p1  :=  k;
  end;


begin
  readln(a,  b);
  c  :=  p1(a,  b);
  writeln(c);
end.
```

Ici, au lieu d'ajouter 1 à chaque décrémentation de j, on ajoute j elle-même. Autrement dit, lorsque les valeurs successives de j sont dépilées, elles servent au calcul. Reprenons le détail, comme précédemment :
Supposons que les valeurs fournies soient 2 et 2. Voici le déroulement :

- entrée dans p1, i = 2, j = 2, k = ? (k n'étant pas initialisée, sa valeur est inconnue).
- j#0, donc la partie dans le ELSE est exécutée.
  - i et j sont sauvées également. Cela fait deux piles, une par variable de la récurrence,
  - appel à p1, avec les valeurs i et j − 1, soit 2 et 1. k est passée en adresse avec la valeur inconnue déjà justifiée,
  - entrée dans p1, i = 2, j = 1, k = ? (k n'étant pas initialisée,...),

- j#0, donc la partie dans le ELSE est exécutée.
  - empilement de chaque variable sur sa pile,
  - appel à p1, avec les valeurs i et j – 1, soit 2 et 0,
  - entrée dans p1, i = 2, j = 0, k = ? (k n'étant pas initialisée,...),
  - j = 0, donc k : = i, soit k : = 2,
  - il n'y a plus d'appel à p1, donc on a un retour à la structure d'appel précédente. Dépilement des variables, qui sont aux valeurs i = 2, j = 1.
- k : = k + j, donc k : = 3 ;
- pas d'autres appels, donc retour à la procédure appelante.

- k : = k + j, donc k : = 5, puisque j est dépilée avec la valeur 2.
- sortie de p1, retour au programme principal, avec c : = 5.

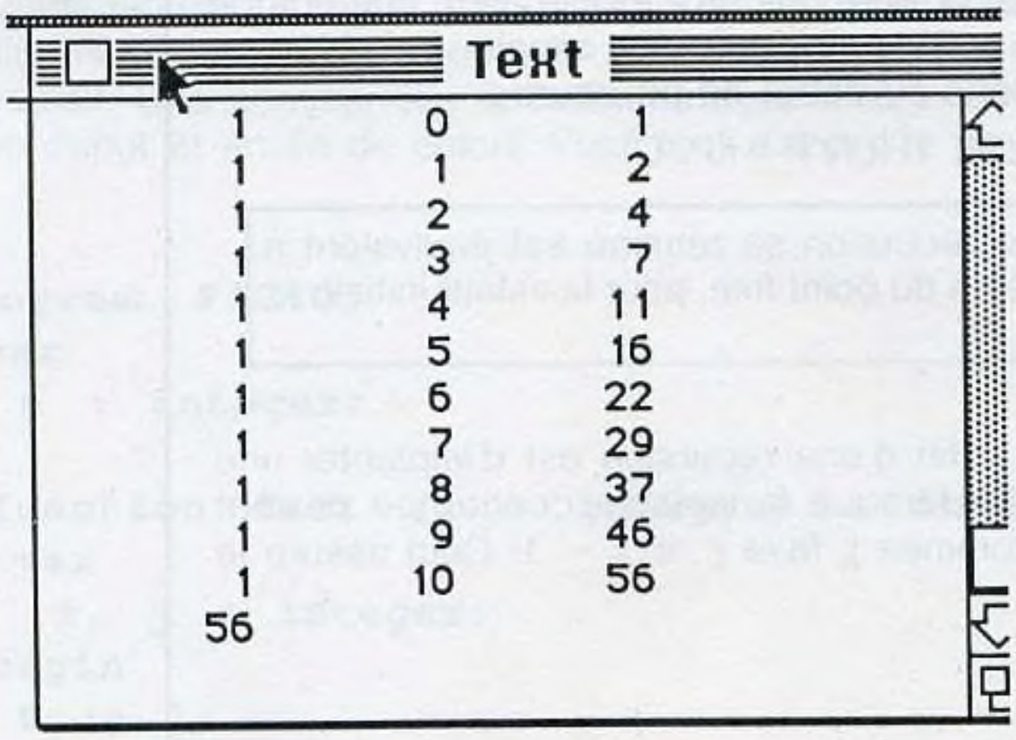
Le programme calcule la somme des b premiers entiers, à laquelle on ajoute une valeur a, d'origine.

5. COMPLEMENT SUR LA GESTION DES VARIABLES IMPLIQUEES DANS UNE RECURSION

A chaque appel récursif d'une procédure, la totalité des variables impliquées dans la récursion, c'est-à-dire celles passées par valeurs seulement sont empilées en vue d'une sauvegarde pour usage ultérieur dans cette procédure. Un jeu complet de variables locales, portant le même nom, est donc créé, permettant au processus de se poursuivre. Seul le champ d'action de ces variables permet de les différencier de celles créées à l'étape précédente, ou à l'étape suivante. Elles ne sont en effet actives que dans la procédure et au niveau d'appel correspondant, géré avec les piles.
Voici, en forçant à peine le programme précédent, le jeu des variables au cours de l'exécution :

```
program  toto;
 var
  a,  b,  c  :  integer;
  t  :  string;

 function  p1  (i,  j  :  integer)  :  integer;
  var
   k  :  integer;
 begin
  if  j  =  0  then
   k  :=  i
  else
   k  :=  p1(i,  j  -  1)  +  j;
  writeln(i,  j,  k);
  p1  :=  k;
 end;

begin
  readln(a,  b);
  c  :=  p1(a,  b);
  writeln(c);
end.
```

L'exécution de ce programme, avec i = 1 et j = 10, donne :



On voit bien l'évolution des variables locales, en fait celle de j, i étant constante.

6. UN PREMIER PROBLEME

L'un des plus gros problèmes de la récursivité est qu'un programme l'utilisant peut ne pas finir, si les variables de contrôle sont mal gérées par le programmeur. Il n'est pas toujours facile de tester ceci, surtout dans le cas de récursivité indirecte. Nous donnons ici un exemple d'un tel cas, en récursivité directe, bien sûr un peu outré. Le programme est le même que précédemment, avec une seule et simple modification qui pourrait être une faute de frappe :

```
program  toto;
 var
  a, b, c : integer;
  t : string;

 function p1 (i, j : integer) : integer;
  var
   k : integer;
 begin
  if j = 0 then
   k := i
  else
   k := p1(i, j - i) + j;
  writeln(i, j, k);
  p1 := k;
 end;

begin
  readln(a, b);
  c := p1(a, b);
  writeln(c);
end.
```

Vous l'avez vu, la seule nuance est que l'appel se fait avec i, et j – i, et non j – 1 comme avant. Si j – i, j – 2i, j – 3i,... j – ni, ne sont pas nuls, le programme boucle à l'infini. Donc **ATTENTION**, ces méthodes demandent une certaine pratique et un grand soin.

En particulier, le test de conclusion de la récursivité doit être étudié avec une grande attention, et éventuellement testé sur un petit cas particulier simple. La structure générale est donc, en reprenant les notations du début de ce cours :

P : IF test de fin FALSE THEN P = f(P, C)

Note pour les amateurs : Prouver qu'une récursion se termine est équivalent à prouver qu'une fonction f(x) vérifie le théorème du point fixe, pour la valeur initiale x_0 fournie.

Une bonne méthode simple pour assurer la fin d'une récursion est d'implanter une variable à décroissance fixe et de s'arrêter lorsque la variable concernée devient négative. Par exemple, si la variable est nommée j, faire j := j − 1. Ceci assure le résultat.

On peut réécrire l'agorithme général :

```
IF j≥0 THEN
    { j:= j-1
    { P:=f(P,C)
```

7. UN AUTRE PROBLEME, QUAND UTILISER LA RECURSIVITE, ET QUAND L'OUBLIER

Il faut garder en mémoire en permanence que la gestion de la récursivité est une perte de temps et de place mémoire, en raison de la gestion des piles et autres mémoires à conserver. Il est donc indispensable de savoir si une autre technique peut être employée avant d'envisager l'utilisation extrême que représente la récursivité. Il semble clair que, pour envisager un programme récursif, il faut au moins que l'on dispose d'un algorithme récursif mais, même dans ce cas, on ne peut garantir que le programme calqué sur l'algorithme sera le meilleur. Prenons le cas classique, s'il en est, de la fonction factorielle déjà définie. On a :

$$FACT(n) = 1 \,.\, 2 \,.\, \ldots\ldots \,.\, (n-1) \,.\, n$$

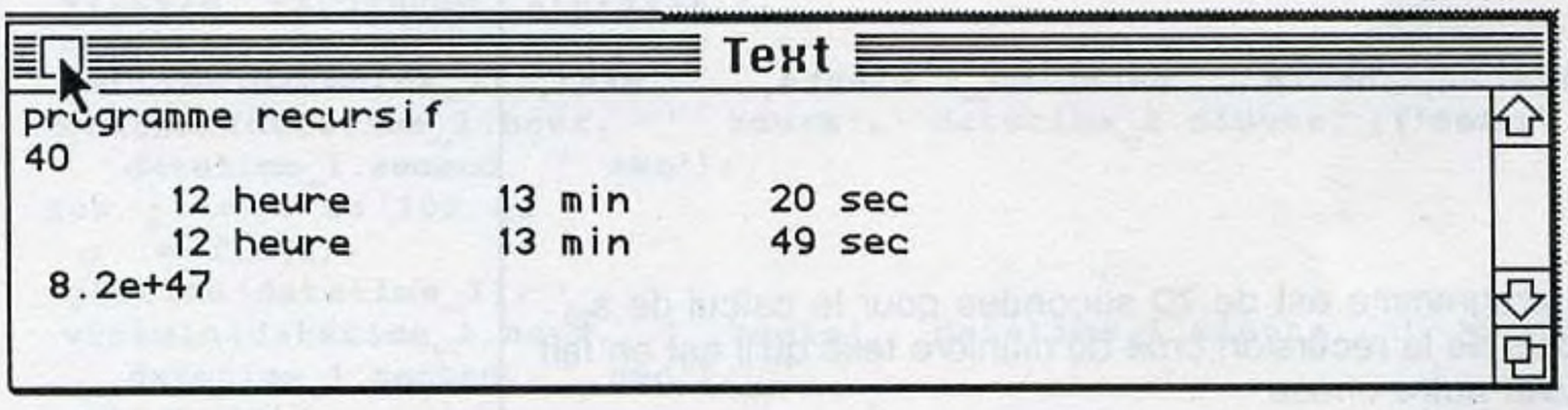
avec FACT(0) = 1 par convention.
On a donc, de manière directe : FACT(n) = FACT(n − 1) * n
Cette relation est récurrente, donc un programme pourra être du type :

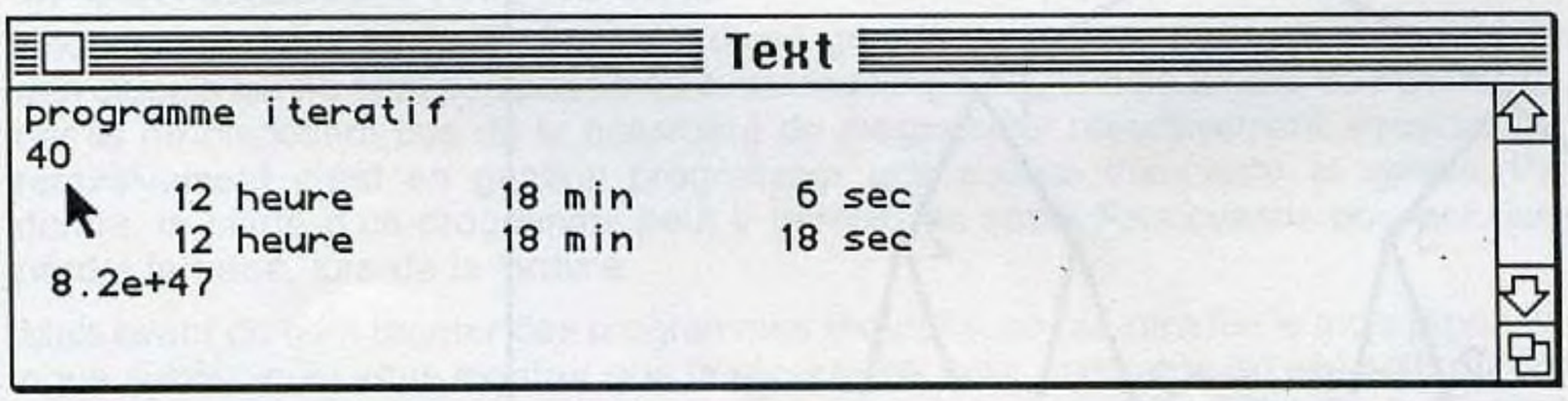
```
program  FACTO;
  var
    a  :  integer;

  function  fact  (i  :  integer)  :  integer;
  begin
   if  i  =  0  then
    fact  :=  1
   else
    fact  :=  i  *  fact(i  -  1);
  end;

begin
  readln(a);
  writeln(fact(a));
end.
```

Ce programme est parfaitement standard, et nous allons le comparer, en utilisation des commandes spécifiques au Macintosh, que nous utilisons pour programmer, avec un programme non récursif. Ces commandes spécifiques sont simplement un appel à l'horloge interne, en début et en fin de calcul. Voici tout d'abord le programme non récursif :

```
program  FACTO;
 var
  a  :  integer;

 function  fact  (i  :  integer)  :  integer;
  var
   f,  j  :  integer;
  begin
   f  :=  1;
   for  j  :=  1  to  i  do
    f  :=  f  *  j;
   fact  :=  f;
  end;

begin
  readln(a);
  writeln(fact(a));
end.
```

Pour un calcul effectué 100 fois, les temps de calculs sont respectivement :
On considère que la première ligne indique le moment où le programme a démarré et la seconde le moment d'arrêt.





Le gain est net et se passe de commentaires. Douze secondes ou vingt-neuf, cela fait une grande différence.

Un exemple différent montre une autre face de la difficulté que peut rencontrer un programmeur lors du développement d'un programme. Cet exemple est, lui aussi,

mathématique mais nous ferons plus loin la part belle à des exemples non mathématiques. Considérons une suite de nombres définis de la manière suivante :

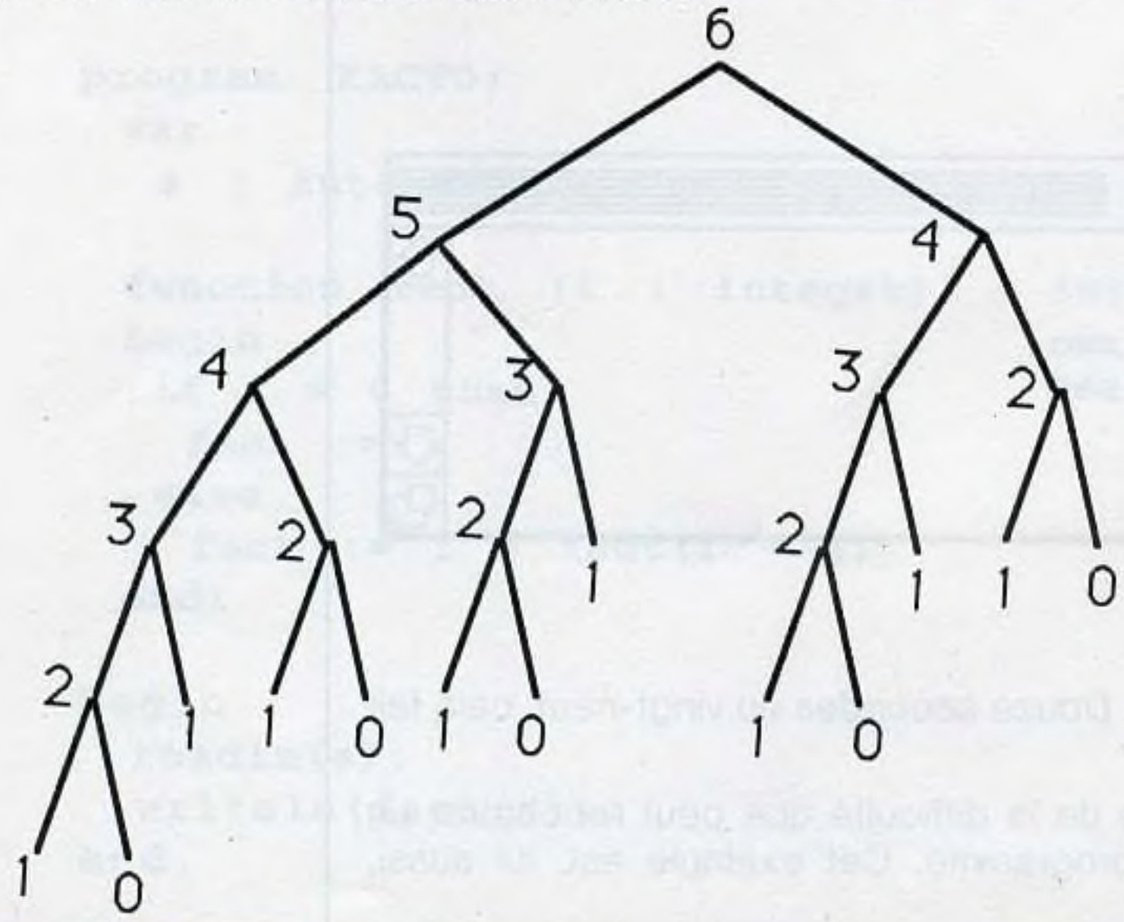
- le premier est nul : $s_0 = 0$
- le second vaut 1 : $s_1 = 1$
- les suivants sont définis par : $s_n = s_{n-1} + s_{n-2}$

Par exemple, $s_2 = 1$, $s_3 = 2$,...

Le problème consiste à calculer s_n , pour un n quelconque, choisi en début de programme. Le programme pourra être :

```
program  FIBONACCI;
 var
  a, j : integer;
  c : extended;
   datetime_1 : datetimerec;

 function fib (i : integer) : integer;
 begin
  if i = 0 then
   fib := 0
          else
  if i = 1 then
   fib := 1
          else
    fib := fib(i - 1) + fib(i - 2);
 end;

begin
 showtext;
 writeln('programme  iteratif');
 readln(a);
 gettime(datetime_1);
 writeln(datetime_1.hour, ' heure', datetime_1.minute, ' min',
    datetime_1.second, ' sec');
 for j := 1 to 100 do
  c := fib(a);
 gettime(datetime_1);
 writeln(datetime_1.hour, ' heure', datetime_1.minute, ' min',
    datetime_1.second, ' sec');
 writeln(c);
end.
```

Le temps d'exécution de ce programme est de 70 secondes pour le calcul de s_{20} . D'autre part, le nombre d'appels de la récursion croît de manière telle qu'il est en fait impraticable. Il faut donc trouver autre chose.



Complexité :

pour s_2 2 additions
pour s_3 3 additions
pour s_4 5 additions
pour s_5 8 additions
pour s_6 13 additions

Pour chaque terme, il faut bien sûr le nombre d'additions cumulé des deux termes précédents.

Une amélioration simple consiste à utiliser, une fois encore, un programme itératif :

```
program  FIBONACCI;
  var
    a,  j  :  integer;
    c  :  extended;
     datetime_1  :  datetimerec;

  function  fib  (i  :  integer)  :  integer;
   var
     x,  y,  z,  j  :  integer;
  begin
   x  :=  1;
   y  :=  0;
   for  j  :=  1  to  i  do
    begin
      z  :=  x;
      x  :=  x  +  y;
      y  :=  z;
    end;
   fib  :=  x;
  end;

begin
  showtext;
  writeln('programme  iteratif');
  readln(a);
  gettime(datetime_1);
  writeln(datetime_1.hour,  '  heure',  datetime_1.minute,  '  min',
     datetime_1.second,  '  sec');
 for  j  :=  1  to  100  do
   c  :=  fib(a);
  gettime(datetime_1);
  writeln(datetime_1.hour,  '  heure',  datetime_1.minute,  '  min',
     datetime_1.second,  '  sec');
  writeln(c);
end.
```

8. CONCLUSION PROVISOIRE

Nous avons décrit ce mois-ci un outil d'une grande puissance, permettant de réaliser des calculs qui ne sont pas possibles avec des langages comme BASIC ou FORTRAN, car ils ne disposent pas de la possibilité de programmer récursivement. Programmer récursivement c'est en général programmer une source compacte et simple. Par contre, la clarté d'un programme peut y perdre, les appels successifs pouvant faire perdre la trace, lors de la lecture.

Mais avant de faire tourner des programmes récursifs, ce qui sera fait le mois prochain, nous avons voulu vous montrer que la récursivité, pour puissante qu'elle soit, ne doit pas faire oublier la règle d'or de tout programme : simplicité et rapidité. Or, l'utilisation de la récursivité, si elle satisfait le premier critère (le plus souvent), ne vérifie presque jamais le second. Il faut donc choisir avec soin les applications où elle est indispensable et l'éviter si c'est possible.

Le mois prochain sera consacré à la description d'applications récursives, en particulier graphiques. Ceux qui ne disposent pas de facilités graphiques sur leur Pascal auront tout de même des programmes spectaculaires, c'est promis !

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Comme promis, voici l'examen détaillé de la majorité des procédures que nous avons évoquées le mois dernier. Cependant, avant de les décrire, nous allons étudier, de la manière la plus complète possible, les routines que l'on peut qualifier de plus bas niveau, c'est-à-dire la gestion du clavier et l'affichage à l'écran, car ces deux entités nous seront nécessaires dans la programmation des autres routines.

COURS Nº 18

PLAN DU COURS

1. Gestion du clavier (routine Litcar)
2. Affichage du buffer
3. Quelques fonctions simples
4. Mouvements du curseur
5. Exercice : le jeu des chiffres et des lettres

1. GESTION DU CLAVIER (ROUTINE LITCAR)

Deux contraintes sont essentielles au bon fonctionnement de notre éditeur :

- la possibilité de séparer lecture du clavier et écho à l'écran,
- l'accès à l'ensemble des codes ASCII sans filtrage (notamment les caractères de contrôle).

Les raisons sont presque évidentes. D'une part il nous faut pouvoir répartir les actions en fonction du code frappé au clavier (c'est le rôle de la routine CORPS, cf. sa description en figure 1), et d'autre part l'utilisateur doit être en mesure de composer directement au clavier tous les codes de contrôle prévus.

```
début_procédure
                                                    LITCAR(c);
                                                    { lecture d'un
caractère c }
                                                    si "c est un
caractère de contrôle" alors
                                                      début_si
                                                       suivant la
valeur de c   { on teste les différentes valeurs de c }

      Insert        : INSERTION;        {on se met en mode insertion }
      Surimp        : SURIMPRESSION;    {on se met en mode remplacement}
      Destruc-car   : EFFACE;           {on efface un caractère}
      Destruc-ligne : EFFACE_LIGNE;     {on efface la ligne}
      Mont          : Monte;            { on monte  }
      ""
      ""
      Termi         : Fin="vrai";       { on sort de la boucle principale}

      Autre         : ERREUR;           { il n'y a pas de fonction
                                          correspondant à c }
   fin_suivant
                                  sinon
   INSERER(c);    {on ajoute c à l'endroit du curseur }
   fin_si
fin_procedure
```

Figure 1 (routine CORPS).

En BASIC, pas de problème, on dispose soit de la procédure **GET** soit de la fonction **INKEY$** qui n'agissent pas sur l'affichage ; en PASCAL c'est un peu plus délicat, tout dépend du type d'implantation du langage. En général, il est possible d'ouvrir un fichier prédéfini **KEYBOARD** de type **TEXT** (par exemple en PASCAL UCSD), qui n'engendre pas d'écho à l'écran contrairement au **READ** standard ; la seconde solution consiste à appeler directement la fonction du système d'exploitation (sous CP/M, MS-DOS, etc.) qui lit directement le clavier sans écho (c'est notamment possible en TURBO-PASCAL). Voici donc une première solution pour ces deux langages.

BASIC

```
10000 REM LITCAR lecture d'un caractère C$ au clavier
10010 C$=INKEY$: REM ou GET C$ suivant les BASICs
10020 IF C$="" THEN 10010 : REM on boucle
10030 RETURN
```

PASCAL

```
function LITCAR : char;
 var c:char;
 begin
      read(keyboard,c); { attention read est synchrone }
      LITCAR:=c;
 end;
```

Améliorations :

Suivant le type de votre ordinateur, il se peut que vous soyez gêné par le clignotement parasite du curseur. Ainsi, le plus souvent, celui-ci reste après le dernier caractère affiché même si entre-temps vous avez effectué un positionnement ailleurs dans l'écran (il est fréquent que ce positionnement ne soit pris en compte qu'au moment de l'affichage d'un nouveau caractère). Voici une solution un peu compliquée mais qui améliore sensiblement la qualité de l'affichage. Elle repose sur le principe simple, en théorie, de la définition de son propre curseur (vous avez sans doute remarqué dans les constantes que nous avons définies le mois dernier que ce curseur avait été prévu, il s'appelle **curseur** en PASCAL et **CU$** en BASIC). Dans la pratique, il peut être nécessaire de remettre en cause l'utilisation des fonctions utilisées en BASIC. En effet, nous avons précisé plus haut que **GET** ou **INKEY$** n'affectaient pas l'affichage à l'écran. Ce n'est pas tout à fait vrai puisque, le plus souvent, elles font réapparaître le curseur. Il faut donc utiliser des routines de plus bas niveau (des fonctions du système d'exploitation ou plus directement des routines personnelles lisant l'adresse du clavier en zone d'entrée/sortie (E/S). Voici deux exemples détaillés en BASIC d'une gestion complète du curseur parallèlement à celle du clavier.

BASIC :

Premier cas pour lequel le positionnement prend effet dès le **GET**.

```
10000 REM LITCAR lecture d'un caractère C$ au clavier
10005 REM et gestion du curseur standard
10010 HTAB X:VTAB Y: REM on se positionne à l'écran en X,Y.
10015 GET C$:REM le curseur standard est bien placé
10020 IF C$="" THEN 10015 : REM on boucle
10030 RETURN
```

Deuxième cas pour lequel on ne peut plus utiliser le GET. Pour l'exemple, on a supposé que l'adresse AD était celle du clavier et que par un PEEK on pouvait lire le code ASCII du caractère tapé, sinon 0. Il faut en plus récupérer le caractère sur lequel se situe le curseur afin d'alterner l'affichage du curseur et l'affichage de ce caractère. Nous rappelons que :

- T$ est le buffer contenant le texte,
- PR est le numéro de la première ligne du buffer à être affichée,
- X, Y sont la position en colonne, ligne du curseur et que le coin en haut et à gauche est 1, 1 (attention si ce n'est pas pour vous 0, 0 !),
- enfin, on ajoute les variables locales C1$ pour le caractère sous le curseur, et 11 pour la gestion du clignotement.

```
10000 REM LITCAR lecture d'un caractère C$ au clavier
10005 REM et gestion du curseur CU$
10008 C1$=MID$(T$(PR+(Y-1)),X,1):II=0
10010 HTAB X:VTAB Y: REM on se positionne à l'écran en X,Y.
10015 C=PEEK(AD)
10020 IF II=0 THEN PRINT CU$;
10028 IF II=25 THEN PRINT C1$;
10030 II=II+1:IF II=51 THEN II=0: REM 50 est à adapter.
10040 IF C=0 THEN 10010 : REM on boucle
10045 HTAB X :VTAB Y : PRINT C1$;:REM attention à remettre le
10050 RETURN:REM                        caractère en place
```

Remarque :
L'amélioration oblige à initialiser le buffer de texte pour le premier positionnement (X = 1, Y = 1 et PR = 1) de telle sorte que C1$ existe dans tous les cas. On insère donc un blanc : T$(1) = " ".

2. AFFICHAGE DU BUFFER

La routine INSERER est chargée de mettre à jour le buffer texte mais aussi de rafraîchir l'affichage. Voici tout d'abord la description de cette routine en pseudo-langage, ainsi que la définition des fonctions utilisées :

Début_procédure

(initialisation des 2 variables locales ligne et taille)

ligne = prem_ligne
 + (y_curseur − 1); (ligne est la ligne du buffer concernée)
taille = *long*(buffer(i)) ; (longueur de la ligne)

(test si ligne pleine en mode insertion)

si (x_curseur = nb_col) *alors*
 début_si
 ERREUR ;
 sinon

(gestion du mode d'écriture à l'aide de la variable locale dep)

si (mode_ins est vrai) *alors*
 début_si
 dep = 0
 sinon
 dep = 1
 fin_si

(modification de la ligne du buffer)

Buffer(ligne) = *sous_chaîne*(buffer(ligne) , 1, x_curseur − 1)
 + c + *sous_chaîne*(buffer(ligne) , x_curseur + dep,
 taille − x_curseur + (1 − dep)) ;
 x_curseur = x_curseur + 1 ; (décalage du curseur)
fin_si

(modification de la ligne à l'écran)

```
position_écran(1, Y)          (on se positionne au bon endroit dans l'écran)
écrit(buffer(ligne) ;         (routine prédéfinie d'affichage, le plus
                               simple est de réécrire toute la ligne (on
                               pourrait tout à fait optimiser)

    fin_procédure
```

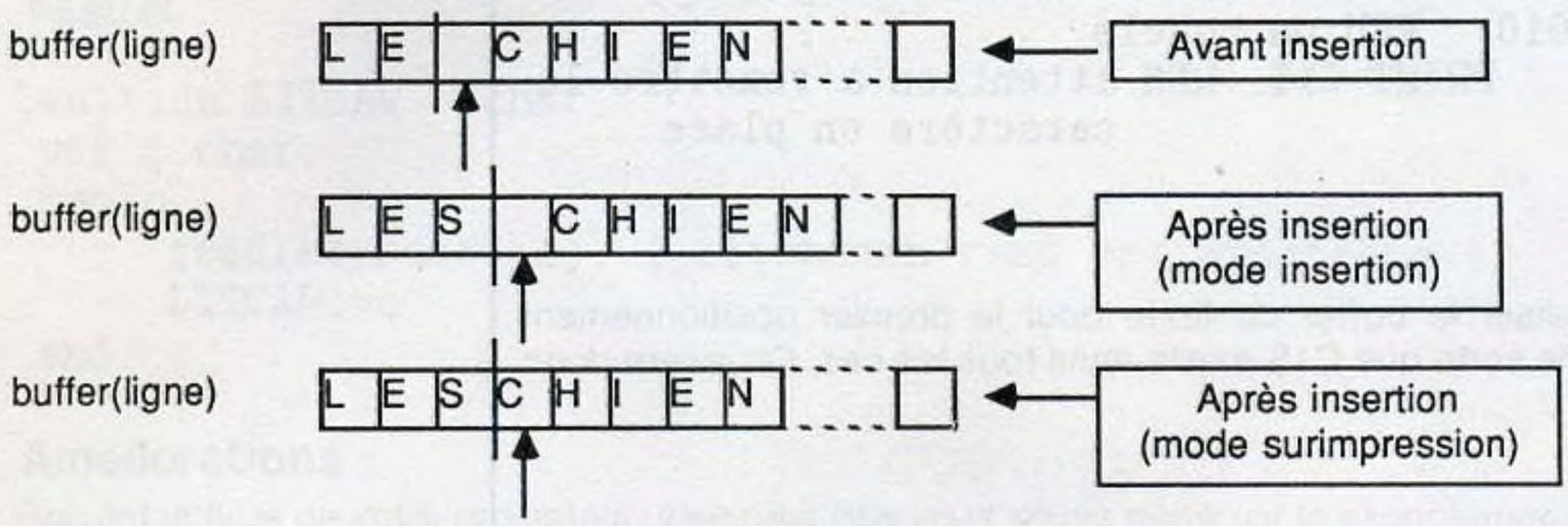


Figure 2 (routine INSERER).

N.B. : description sommaire des fonctions utilisées.

- *sous_chaîne*(**s, p, l**) permet d'extraire de la chaine **s, l** caractères à partir de la position p,
- *long*(**s**) donne la longueur de la chaine **s**,
- *position_écran*(**c, l**) précise la position **c**olonne et **l**igne du curseur,
- *écrit*(**s**) inscrit la chaine **s** à l'endroit du curseur sans retour à la ligne.

A vous de traduire ces fonctions dans votre langage préféré ; cela ne devrait pas poser trop de problème !
D'autre part, la longueur d'une ligne est limitée à 1 de moins que la largeur de l'écran, cette limitation simplifie considérablement la gestion du curseur lorsque celui-ci arrive en fin de ligne. En effet, que ce soit en mode insertion ou en mode surimpression, le curseur restera bloqué sur la dernière colonne de la ligne et chaque caractère affichable introduit produira un BEEP sans effet sur le texte affiché

3. QUELQUES FONCTIONS SIMPLES

Avant d'étudier les mouvements du curseur et les effacements divers, voici les trois routines les plus simples participant au fonctionnement de la procédure CORPS :

ERREUR :

C'est une procédure simple qui produit un Beep.
Voici un exemple en BASIC de cette routine élémentaire :

INSERTION :

```
début_procédure
    mode_ins = vrai :
fin_procédure
```

SURIMPRESSION :

```
début_procédure
    mode_ins = faux :
fin_procédure
```

Si vous envisagez de compliquer quelque peu cet éditeur et, par exemple, d'améliorer l'affichage en l'agrémentant d'une ligne de «status» qui précisera la position du curseur, le mode (insertion ou surimpression), etc., il faudra ajouter la gestion de cette ligne à la fois dans la routine INSERER et dans les deux dernières procédures décrites (INSERTION et SURIMPRESSION).

4. MOUVEMENTS DU CURSEUR

Les choses se compliquent ; il faut en effet, au niveau du déplacement du curseur, prévoir le défilement de l'écran, la gestion des limitations en taille du buffer texte, prévoir la remontée du curseur lorsqu'il arrive en début de ligne et de manière symétrique le passage à la ligne suivante dans le cas contraire. Détaillons au cas par cas.

Déplacement à droite

Une limitation en fin de ligne ; auquel cas on appelle la routine de descente avec comme position du curseur le premier caractère de la ligne en cours.

Déplacement à gauche

Une limitation en début de ligne provoque, symétriquement à la précédente, un appel à la montée d'une ligne avec, comme position du curseur, le dernier caractère de ligne actuelle.

Déplacements vers le haut et le bas

Il y a pour ces deux actions deux limitations.

a. atteinte du bord de l'écran, ce qui impose un défilement,
b. atteinte de la limite du buffer, donc blocage du curseur.

La deuxième contrainte sera gérée par la routine chargée de faire défiler le texte. Nous allons donc définir six routines :

MONTE	: curseur un cran vers le haut,
DESCEND	: curseur un cran vers le bas,
GAUCHE	: curseur un cran vers la gauche,
DROITE	: curseur un cran vers la droite,
DEF_HAUT_BAS	: défilement de haut en bas,
DEF_BAS_HAUT	: défilement de bas en haut.

Remarque :
Nous commencerons, pour simplifier la progression des deux routines de défilement, par effectuer un rafraîchissement complet de l'écran. Il est bien évident que, dans le cas d'un défilement du bas vers le haut, il est possible de faire travailler directement le défilement standard de l'affichage qui sera plus performant et plus agréable à l'œil qu'une réécriture complète. Cette solution sera, par contre, incompatible avec la gestion d'une ligne de «status». On s'aperçoit donc, à cette étape de la programmation, qu'il faut faire un certain nombre de choix dans les caractéristiques du produit final et que certaines options ne peuvent coexister (tout au moins du point de vue de la programmation). Tout ceci ne fait que conforter la position que nous avons adoptée au départ : à savoir la définition d'un logiciel de base modulaire et évolutif.

5. EXERCICE : LE JEU DES CHIFFRES ET DES LETTRES

Le titre pourrait vous induire en erreur. Il s'agit d'un jeu de déduction où l'un des adversaires choisit une combinaison qu'il laisse le soin de découvrir à l'autre. Cette combinaison est constituée de 4 couples comportant chacun un chiffre et une lettre. On a généralement le choix entre 5 lettres et 5 chiffres.
Exemple :

A B E B
5 3 3 4

Ceci est la combinaison choisie pour le premier joueur et, à ce moment-là, son

adversaire se doit de tenter une première solution :

A B C D
1 2 3 4

La réponse à cette première proposition est que 4 pièces sur 8 sont correctement placées, à savoir :

A B
3 4

et la réponse est 4 - 0 - 0. La signification des deux zéros vous sera fournie ultérieurement.
Il est temps d'essayer une nouvelle combinaison :

A E B D
4 2 3 5

Là, nous n'avons plus que deux pièces bien placées :

A
3

E B
4 5

sont des pièces entrant dans la combinaison mais mal placées. Nous n'en tenons pas compte. Par contre, le couple nº 3 :

B
3

existe dans la combinaison mais à une autre place. Il convient de le signaler. Nous avons alors la réponse : 2 - 1 - 0.
Nous tentons donc une nouvelle combinaison :

A B C B
5 3 3 4

Ici, nous avons 7 pièces correctement placées mais 6 d'entre elles constituent des couples complets. Nous avons donc la réponse suivante : 1 - 0 - 3.

Alors maintenant, à vos claviers et à vos crayons. Dans un premier temps, faites un programme qui choisisse une combinaison et qui vous permette de jouer contre l'ordinateur puis, quand vous aurez établi la martingale que vous employez pour réussir le décryptage, essayez-la par ordinateur interposé.
Attention, ce programme nécessite une bonne analyse avant de commencer à programmer.

DIALOGUE AVEC NOS LECTEURS

I. PRESENTATION DES VOLUMES ELEMENTAIRES

1. Le parallélépipède est caractérisé par trois dimensions :

- longueur
- largeur
- hauteur

Son origine est située au centre de la pièce à mi-distance de chacun des paramètres.

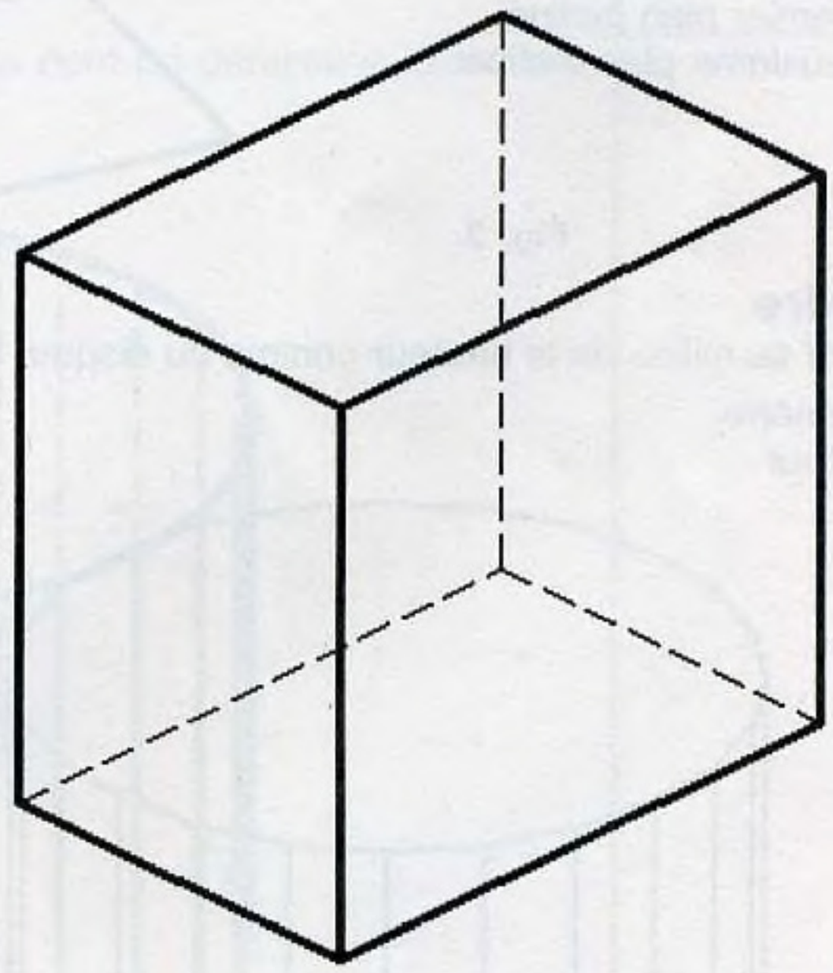


Fig. 1

2. Le coin est un parallélépipède coupé en deux par une diagonale de l'une de ses faces. Son origine est située au milieu d'un plan de jauge arbitraire parallèle au plan xOy. Il nécessite quatre paramètres :

- longueur
- largeur au plan de jauge
- hauteur du plan de jauge par rapport à la base de la pièce
- angle du plan incliné.

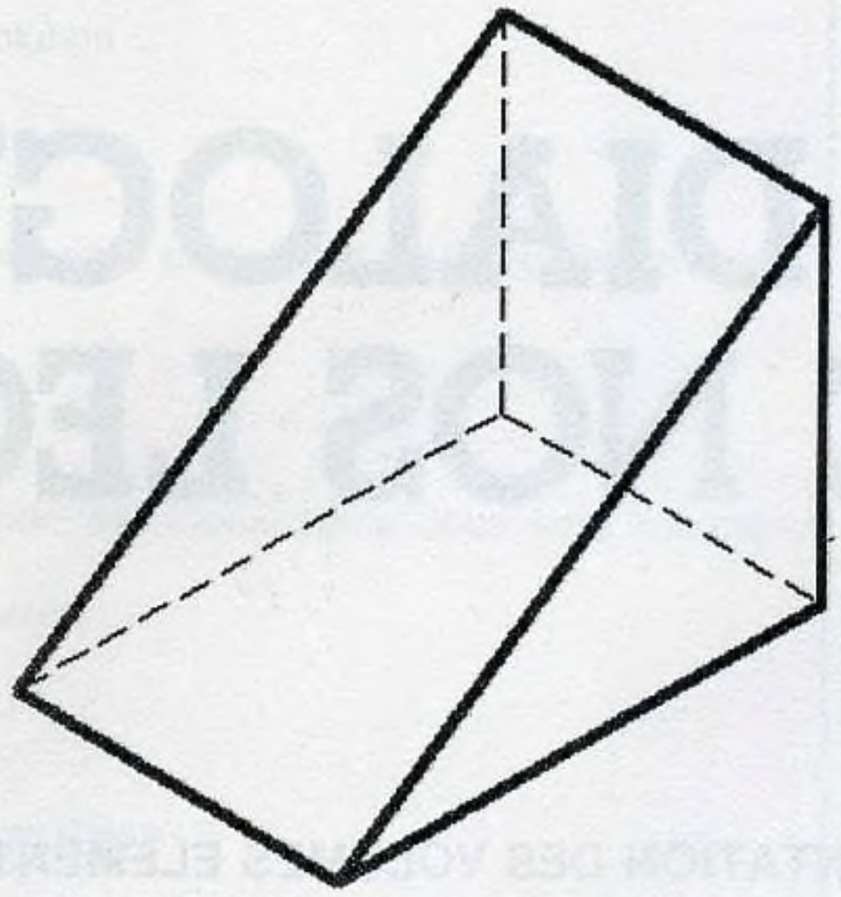


Fig. 2

3. Le double coin

C'est encore un parallélépipède, cette fois-ci doublement biseauté. Son origine est aussi au milieu d'un plan de jauge. Il requiert six paramètres :

- longueur
- largeur au plan de jauge
- hauteur totale
- hauteur au plan de jauge
- angle du premier plan incliné
- angle du deuxième plan incliné

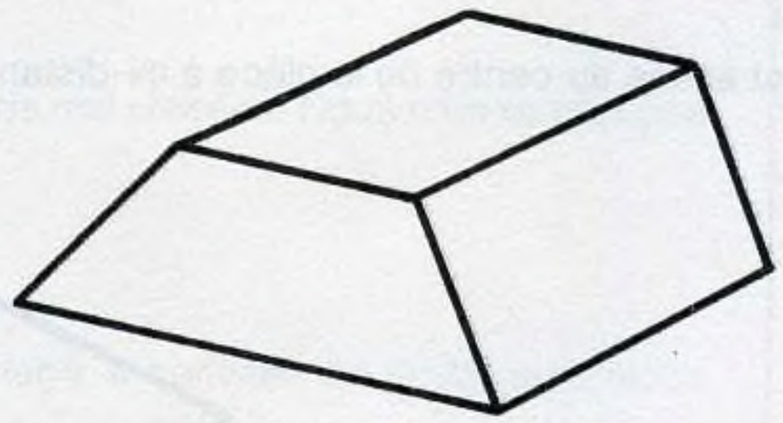


Fig. 3

4. Le cylindre

Son origine est au milieu de la hauteur comme du disque. Il appelle deux paramètres :

- son diamètre
- sa hauteur

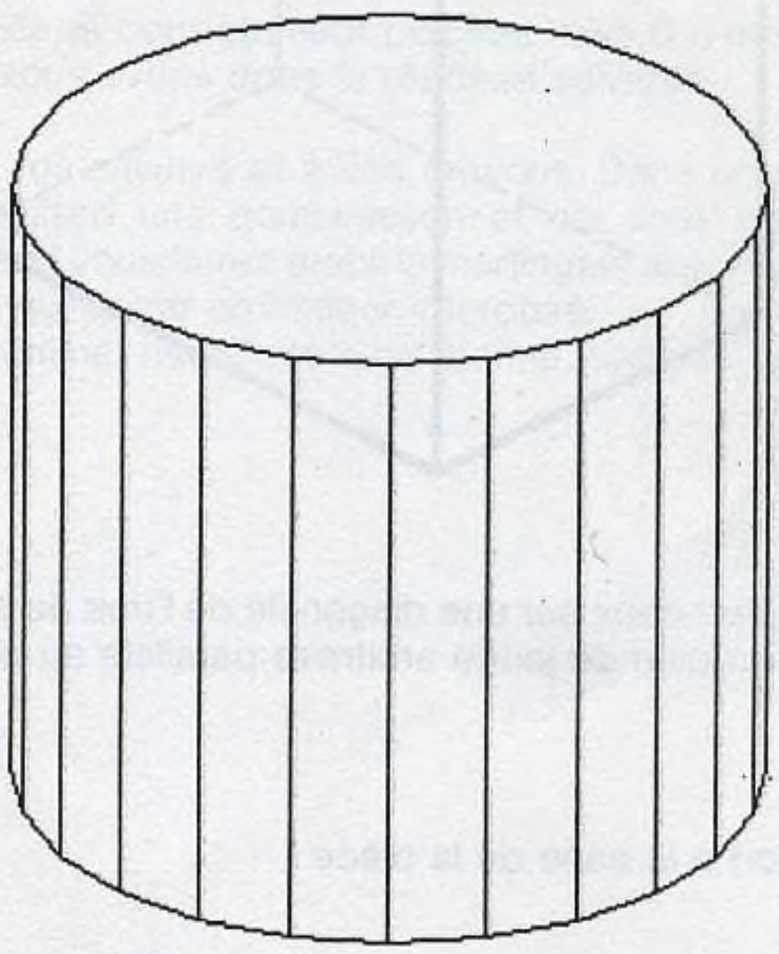


Fig. 4

5. La portion de cylindre

C'est un cylindre coupé verticalement parallèlement à une direction. Il nécessite un paramètre supplémentaire au cylindre :

– la flèche - longueur du cylindre restante.

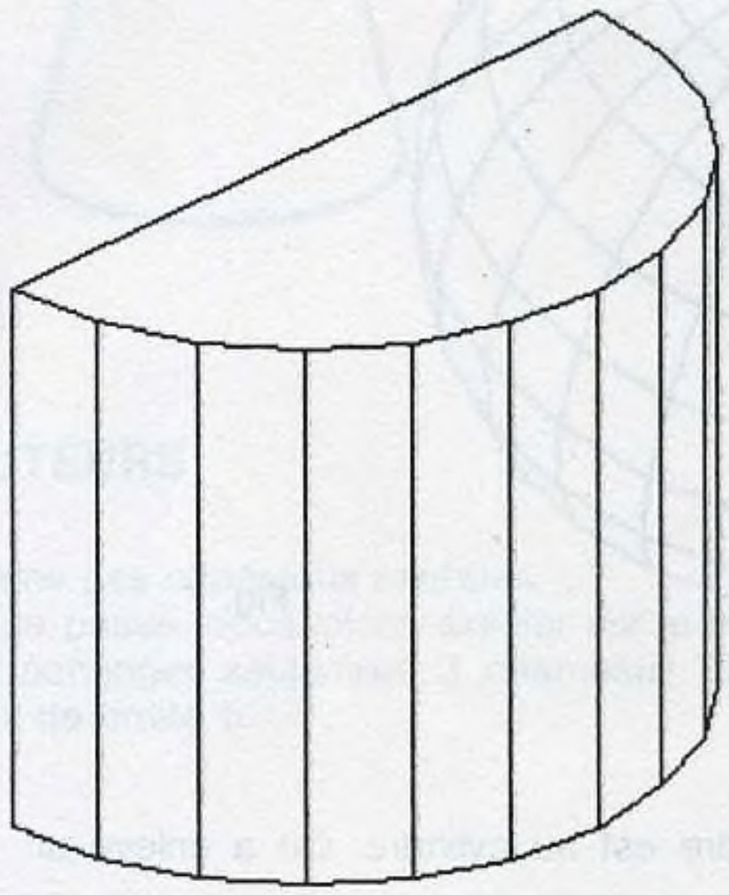


Fig. 5

6. Secteur de cylindre

C'est une portion de «Vache qui rit» mais plus épaisse dont on détermine, outre les caractéristiques du cylindre, l'angle.

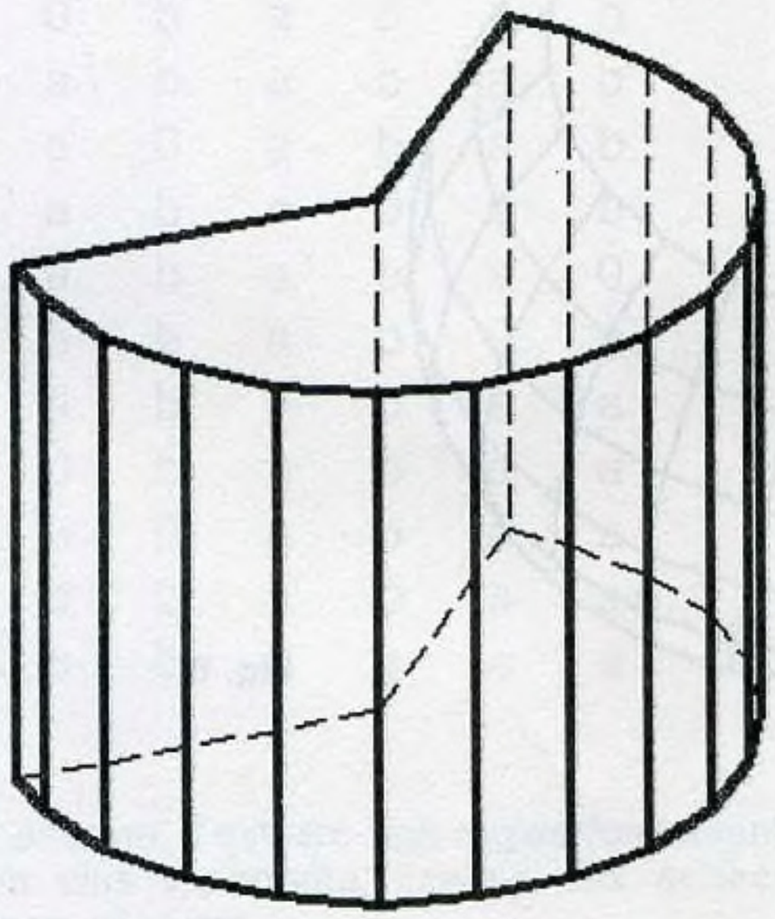


Fig. 6

7. La sphère

Un jour à Toulouse, un enfant demande à son père : «Papa, papa ! Qu'est-ce que c'est qu'une sphère ? - Une sphèrrre ? Mon garçon ! Attends. Eh bien, tu prrrends un ballon de rrrubi et il est rrrond ?!» Une origine, son centre ; un paramètre, son diamètre.

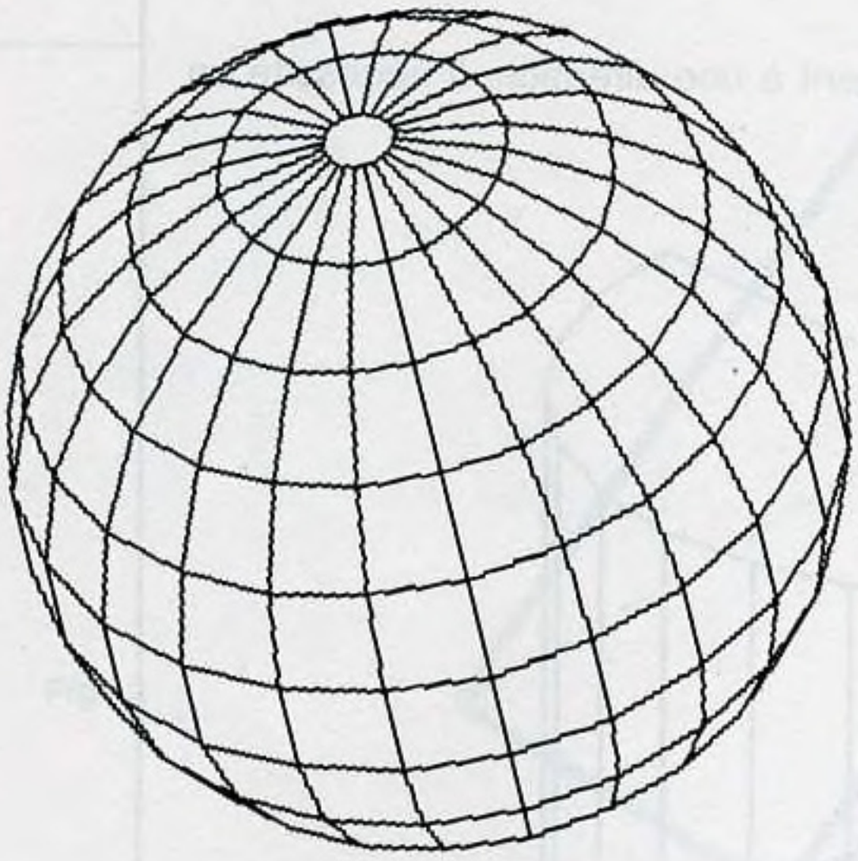


Fig. 7

8. Calotte sphérique

C'est à la sphère ce que la portion de cylindre est au cylindre. On a enlevé un morceau. Il y a donc aussi une flèche.

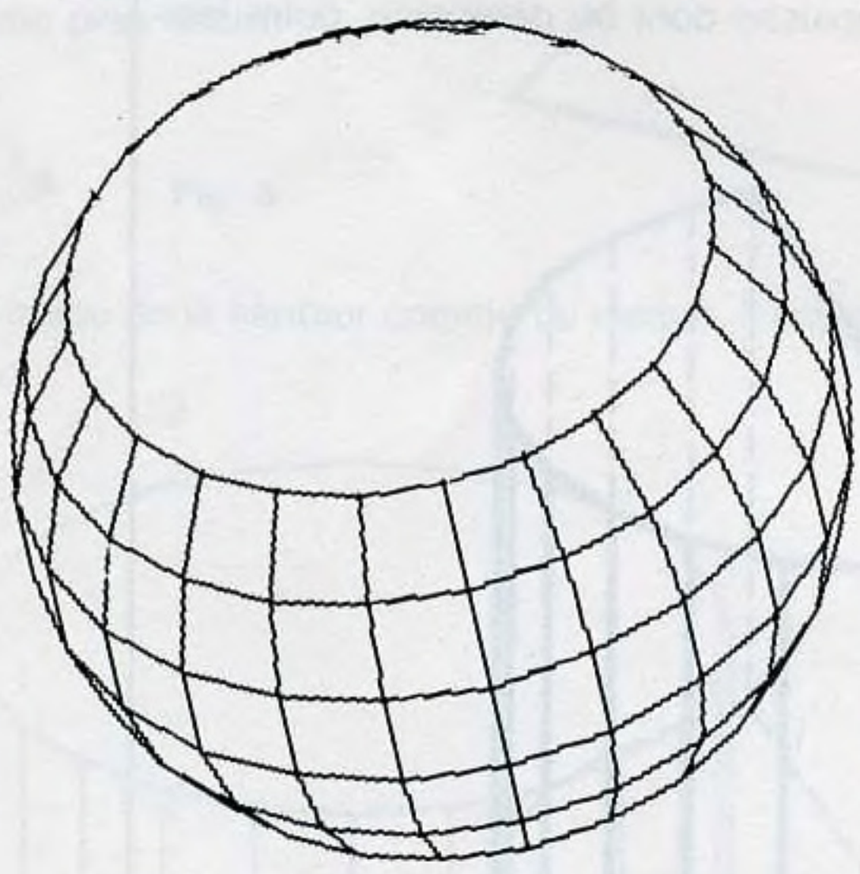


Fig. 8

9. Tronc de cône

C'est un cône tronqué parallèlement à sa base. Son origine se situe sur un plan de jauge. Il a recours à quatre paramètres :

- diamètre au plan de jauge
- hauteur totale
- hauteur au plan de jauge
- angle au plan de jauge.

Fig. 9

II. LES CHAMEAUX SAUTEURS

Passons maintenant au problème des chameaux sauteurs.
Afin de bien observer ce qui se passe, nous allons simuler sur le papier, toutes les opérations nécessaires pour échanger seulement 3 chameaux, ceux de gauche s'appellent a, le vide 0 et ceux de droite b.
Au départ :

	a	a	a	0	b	b	b
1	a	a	0	a	b	b	b
2	a	a	b	a	0	b	b
3	a	a	b	a	b	0	b
4	a	a	b	0	b	a	b
5	a	0	b	a	b	a	b
6	0	a	b	a	b	a	b
7	b	a	0	a	b	a	b
8	b	a	b	a	0	a	b
9	b	a	b	a	b	a	0
10	b	a	b	a	b	0	a
11	b	a	b	0	b	a	a
12	b	0	b	a	b	a	a
13	b	b	0	a	b	a	a
14	b	b	b	a	0	a	a
15	b	b	b	0	a	a	a

De ces déplacements, il faut essayer d'extraire des règles fondamentales et de les hiérarchiser en partant de la plus importante, comme aux échecs où la règle primordiale est de conserver son roi vivant.
Ici, la règle fondamentale est d'effectuer un saut lorsque c'est possible, c'est ce que nous avons fait aux lignes 2, 4, 5, 7, 8, 9, 11, 12 et 14.
Sinon effectuer un déplacement dans la même direction que le dernier saut, c'est le cas des lignes 3 et 6.
Pour toutes les autres (10, 13 et 15), il n'y a qu'un seul choix possible. La ligne 1 étant bien sûr à part, les deux éventualités ayant même valeur.

NOTES PERSONNELLES

```
100 REM  LECTURE DES DONNEES ET CHARGEMENT
110 READ N
120 TC = 2 * N + 1
130 TD = (N + 1) * 2 - 1
140 DIM C$ (TC), D$ (TD)
150 FOR I = 1 TO N
160 C$ (I) = 'A'
170 C$ (N + 1 + I) = 'B'
180 NEXT I
190 C$ (N + 1) = 'V'
200 REM PREMIER MOUVEMENT
210 ND = 1
220 V = N + 1
230 GOTO 750
300 FOR I = 1 TO TC
310 PRINT C$ (I) ;
320 NEXT I
330 PRINT
340 ND = ND + 1
400 IF V + 2 > TC THEN 500
410 IF (C$ (V + 2) < > 'B') OR (C$ (V + 1) < > 'A') THEN 500
420 D$ (ND) = 'BS'
430 C$ (V) = 'B'
440 V = V + 2
450 GO TO 800
500 IF V - 2 <   1 THEN 600
510 IF (C$ (V - 2) < > 'A') OR (C$ (V - 1) < > 'B') THEN 600
520 D$ (ND) = 'AS'
530 C$ (V) = 'A'
540 V = V - 2
550 GOTO 800
600 IF V = TC THEN 750
610 IF (C$ (V + 1) = 'B') AND (D$ (ND - 1) = 'BS') THEN 700
620 IF V = 1 THEN 700
630 IF (C$ (V - 1) = 'A') AND (D$ (ND - 1) = 'AS') THEN 750
640 IF (C$ (V + 1) = 'B') AND (C$ (N - 1) = 'B') THEN 700
650 IF (C$ (V + 1 ) = 'A') AND (C$ (V - 1) = 'A') THEN 750
660 IF V = N + 1 THEN STOP
670 PRINT 'ERREUR, ON NE DOIT JAMAIS ARRIVER ICI'
680 STOP
700 D$ (ND) = 'BG'
710 C$ (V) = 'B'
```

```
720 V = V + 1
730 GOTO 800
750 D$ (ND) = 'AG'
760 C$ (V) = 'A'
770 V = V - 1
800 C$ (V) = 'V'
810 PRINT 'COUP N.' ; ND ; '       ' ; D$ (ND) ;
820 GOTO 300
```

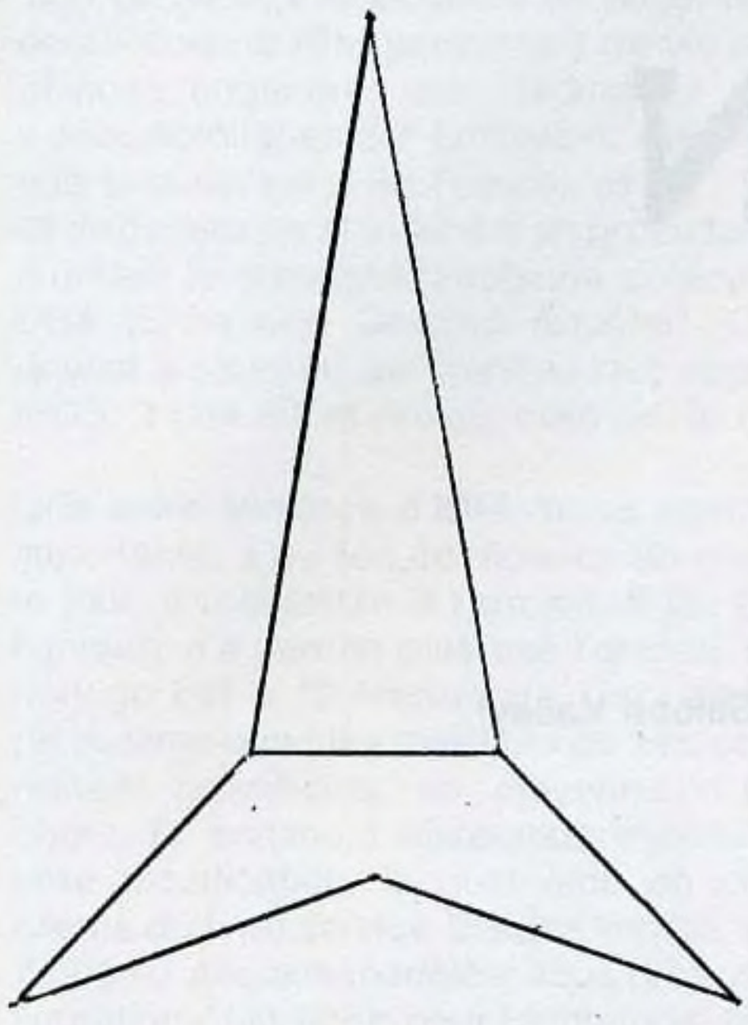
Ci-dessous la version Pascal du même programme :

```
Program Sauts_de_Chameaux ;

Type

  t_ensemble = set of [1..30] ;
  t_deplacement = ( saut_gauche , saut_droit , gauche_glisse , droit_glisse ) ;


Var

  nbre_chameaux , case_vide ,
  limite                    : Integer ;
  gauche , droite ,
  gauche_init , droite_init : t_ensemble ;

Function Deplacement ( var groupe  : t_ensemble ;
                           chameau ,
                           vide    : integer  ) : integer ;


  Begin
    groupe := groupe - [chameau] + [vide] ;
    deplacement := chameau ;
  End ;

Procedure Affichage_deplacement ( var gauche    ,
                                      droite    : t_ensemble ;
                                      vide      : integer    ) : integer ;


  Var
    i : integer ;


  Begin
    For i := 1 to limite do
      if [i] in gauche then write('G')
      else if [i] in droite then write('D')
      else write ('V') ;
    writeln ;
  End ;


Function Mouvement ( var gauche    ,
                         droite    : t_ensemble    ;
                         vide      : integer       ;
                         depl_prec : t_deplacement ) : integer ;

  Var
```

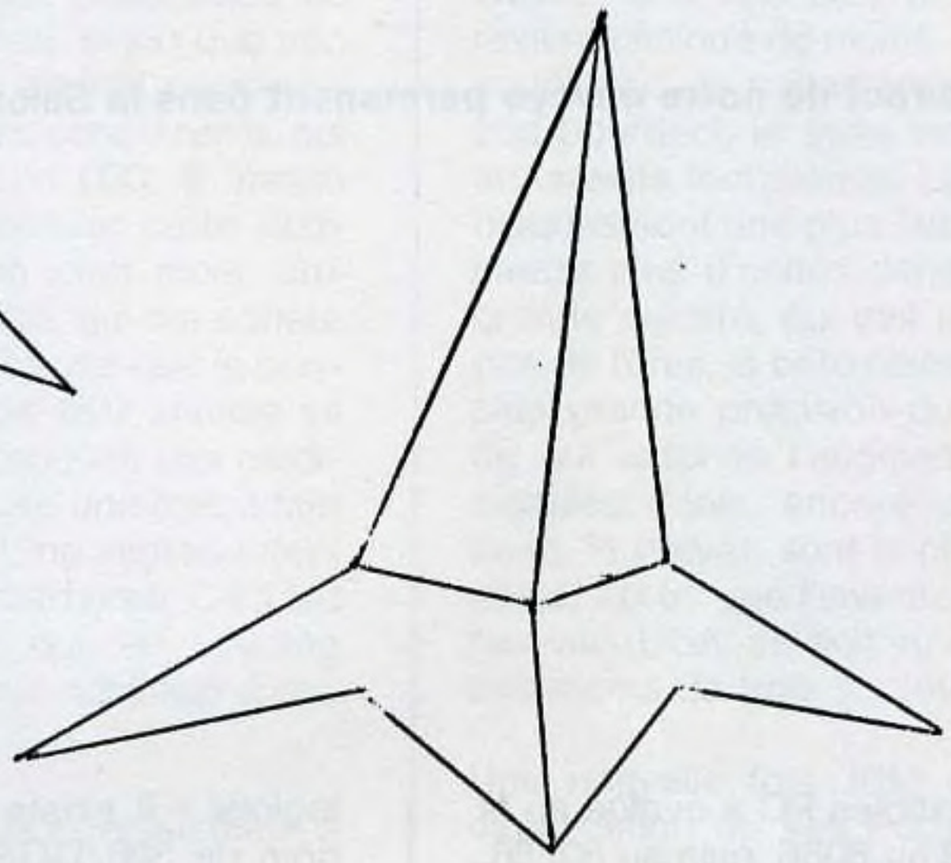
```
Begin
  if ( gauche = droite_init ) and ( droite = gauche_init ) then
     Mouvement := 0
  else
    begin
      if ( [vide - 2] in gauche ) and ( [vide - 1] in droite ) then
          vide := deplacement ( gauche , vide - 2 , vide )
      else
      if ( [vide + 2] in droite ) and ( [vide + 1] in gauche ) then
          vide := deplacement ( droite , vide - 2 , vide )
      else
      case depl_prec of
        saut_droit :

          begin
            if [vide + 1] in droite then
              begin
                vide := deplacement ( droite , vide + 1 , vide ) ;
                depl_prec := droit_glisse ;
              end ;

            else
              begin
                vide := deplacement ( gauche , vide - 1 , vide )
                depl_prec := gauche_glisse ;
              end ;

          end ;
        saut_gauche :
          begin
            if [vide - 1] in gauche then
              begin
                vide := deplacement ( gauche , vide - 1 , vide )
                depl_prec := gauche_glisse ;
              end ;
            else

              begin
                vide := deplacement ( droite , vide + 1 , vide ) ;
                depl_prec := droit_glisse ;
              end ;
          end ;
        otherwise : writeln (' impossible : erreur ');
      end ;
    Mouvement := Mouvement ( gauche , droite , vide , depl_prec ) + 1 ;
    Affichage_deplacement ( gauche , droite , vide ) ;
end ;


Begin
  writeln('quel est le nombre de chameaux de chaque cote ?') ;
  readln (nbre_chameaux) ;
  limite := nbre_chameaux * 2 + 1 ;
  gauche_init := [1..nbre_chameaux] ;
  droite_init := [(nbre_chameaux + 2)..limite] ;
  gauche := gauche_init ;
  droite := droite_init ;
  case_vide := nbre_chameaux ;
  Affichage_deplacement ( gauche , droite , case_vide ) ;
  { premier mouvement Gauche ---> Vide }
  case_vide := Deplacement ( gauche , nbre_chameaux , case_vide ) ;
  Affichage_deplacement ( gauche , droite , case_vide ) ;
  depl_init := gauche_glisse ;
  nbre_deplacement := Mouvement ( gauche , droite , case_vide , depl_init ) + 1 ;
End.
```

III. LA TOUR EIFFEL

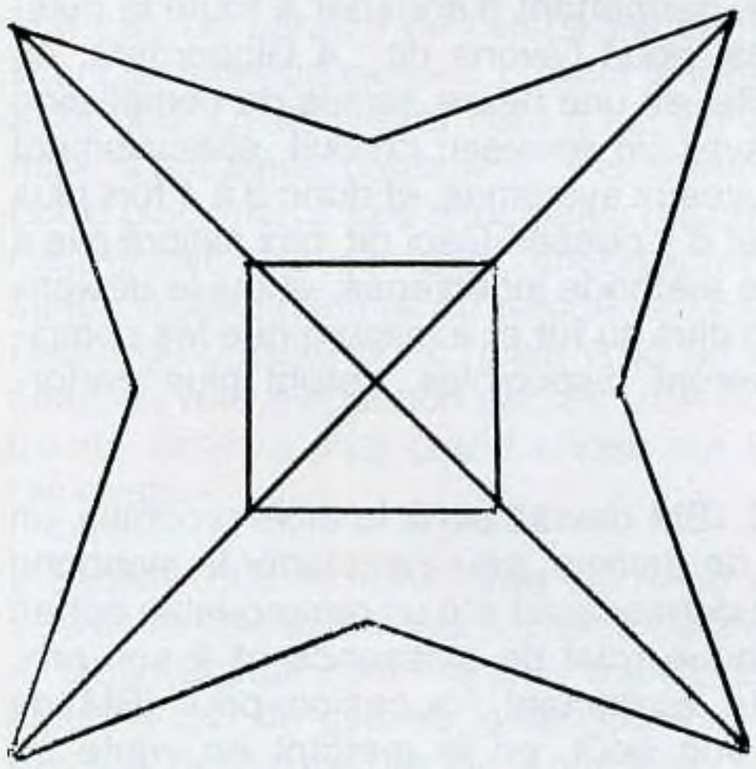
Eh oui, notre dessin à découvrir dans le numéro 36 n'avait qu'une resemblance très lointaine avec la Tour Eiffel. Si d'aucuns ont une meilleure solution, qu'ils nous l'envoient



Vue de face.



Vue en pespective.



Vue de dessus.

C'EST ARRIVE DEMAIN

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

La grande famille des compatibles PC à évolué au fil des temps, passant du 8088 au 8086, puis au 80286. Le dernier en date des processeurs pour ces ordinateurs est le 80386, rapide, performant, et maintenant fiable, il est donc logique qu'il équipe progressivement IBM et ses concurrents. Mais on pouvait se demander s'il ne faudrait pas attendre longtemps des logiciels utilisant leurs possibilités intrinsèques et pas seulement leur compatibilité avec leurs prédécesseurs. Des petits futés ont trouvé comment modifier les programmes existants pour tirer parti des nouvelles instructions et capacités de stockage du 80386-4 Gigaoctets. La méthode est très simple. Prenez un programme bien connu tournant sur un IBM PC ou compatible, écrit en langage évolué. Vous aurez besoin du source, donc seules les sociétés ayant écrit les programmes pourront réaliser la modification. Achetez un des compilateurs de chez Meta-Ware, la seule firme – pour l'instant – proposant un compilateur dédié au nouveau processeur. Compilez le programme à l'aide de ce compilateur, qui utilisera à fond les possibilités du 80386 sur le plan des instructions de son assembleur. Ajoutez un zeste de logiciel – il existe tout prêt chez Phar Lap, sous le nom de 386/DOS Extender – et vous tirez parti du mode protégé, permettant d'accéder à toute la puissance d'accès, nous l'avons dit : 4 Gigaoctets. Au bout du compte, en une heure, temps de compilation inclus, vous avez un nouveau produit, spécialement dédié aux nouveaux systèmes, et donc 3 à 4 fois plus cher. Il suffisait d'y penser. Ceci dit, prix majoré mis à part, c'est une méthode ingénieuse, et qui le deviendra de plus en plus au fur et à mesure que les compilateurs, qui seront disponibles, seront plus performants.

Parlant du PC, IBM devrait sortir le mois prochain, un nouveau bas de gamme, pour remplacer le moribond PC Junior. Ce dernier avait été un remarquable échec dû à son manque total de puissance et à son prix. Cela avait été également l'occasion pour IBM de montrer son bon goût, en le mettant en vente en Europe alors qu'aux USA, la machine avait déjà été retirée pour cause de vente nulle. La palme de l'élégance avait alors été décernée à IBM à cette occasion par de nombreux confrères. Le nouveau serait en

fait le bas de gamme de la nouvelle série IBM, répondant au nom de code «Renegade». Il serait une réplique de l'Apple II GS et à peine plus cher (Apple II GS : 995 $). Les performances semblent proches. En version de base, il disposerait du 8086, de 512 ko, d'un écran couleur IBM permettant de visualiser des graphiques engendrés par l'ordinateur et des images vidéo, simultanément (attention, cela ne signifie pas que cela sera vrai en France), et en 1 024 par 1 024, et disposera en standard d'un processeur permettant d'utiliser le standard graphique couleur IBM, nommé EGA (Enhanced Graphic Adapter). Cette machine devrait s'attaquer au créneau des appareils de type II GS, 1 024 ST et Amiga, mais par le haut.

Une autre annonce d'IBM, moins spectaculaire mais importante, a eu lieu ce mois-ci. Un nouvel AT va voir le jour. Il ressemble à l'ancien, a les possibilités de l'ancien, n'a rien de plus que l'ancien, sinon que son horloge bat à 10 Megahertz. Ceci devrait permettre de rivaliser avec les meilleurs de ses concurrents, qui restent néanmoins, en moyenne, 1 000 $ moins chers. Et comment allez-vous expliquer cette nouvelle modification, la quatrième en cinq mois, aux clients du type service US des impôts, qui ont acheté 7 000 AT anciens modèles, sous prétexte que la configuration était figée pour longtemps. IBM semble se débattre, et à chaque soubresaut, produire une modification de ses appareils. Mais encore une fois, il faut rappeler que les micros, chez IBM, ne représentent que quelques pour-cent du chiffre d'affaires. Ceci fait de plus l'affaire des concurrents qui, tel Leading Edge, fournit du matériel coréen aux administrations US. Du jamais vu en électronique.

A ce sujet, et la faute ne revient pas spécialement à IBM, il semble de plus en plus que la Silicon Valley ne soit pas un paradis de l'électronique. Encore 2 500 licenciements ce mois-ci, portant le total en deux ans à 34 000. Le déclin du hard n'est donc pas récent, mais s'accélère nettement. Par contre, le soft se porte bien, merci. Les principales raisons des problèmes en électronique sont, encore une fois, la puissance d'adaptation et de plagiat de l'Extrême-Orient. Rien de nouveau donc, mais voilà un nouveau créneau en voie d'abandon par les USA ou l'Europe, ce qui ne laissera plus grand chose sur le marché de l'innovation.

Il semble que le futur Mac perturbe de nombreux cadres et décideurs des sociétés concurrentes. SUN et DEC (Digital) sont les plus visées, à travers leurs stations bas de gamme, graphiques et de C.A.O. En effet, il ne se passe de semaines sans que des revues spécialisées pour l'une ou l'autre de ces compagnies ne proposent des articles sur les comparaisons, hypothétiques à ce jour, entre le futur haut de gamme Apple et les bas de gamme des concurrents. En fait, la concurrence ne sera effective qu'à partir de l'hiver prochain, mais chacun affûte d'ores et déjà ses armes. L'enjeu est d'importance car le marché de ce type de stations graphiques est considéré comme le plus susceptible de fort développement dans les années à venir. On comprend que des compagnies comme Sun ou Computervision dont les produits ne couvrent que ce seul réseau, soient inquiètes. Par contre, pour Dec, ce marché n'est que l'un des aspects de sa stratégie qui, même s'il lui semble prometteur, n'est peut-être pas aussi vital que pour les sociétés précédemment citées.

Voilà un progrès qui va peut-être faire grandement évoluer le monde des disques durs. Une société californienne vient d'annoncer la production de disques en verre, au lieu du traditionnel plateau d'aluminium. Gain : deux fois plus de capacité, pour un prix de revient diminué de moitié. Ceci est dû à la plus grande souplesse de traitement du verre. Ne me demandez pas pourquoi, je serai incapable de comprendre les arguments techniques. Les autres avantages de ces disques sont une plus faible épaisseur, permettant de mettre plus d'unités dans le même coffret, une plus grande solidité, qui irait jusqu'à offrir des atterrissages de têtes, la bête noire des disques actuels et une plus grande précision du stockage de l'information, ce qui autorise l'augmentation de la capacité déjà signalée. Mais, encore une fois, la technologie de base, le brevet, sont la propriété d'une société japonaise, après que l'inventeur, Japonais d'origine habitant aux USA, se soit vu claquer la porte au nez par pas moins de trois sociétés américaines.

Une nouvelle fois, IBM doit se protéger contre la modification de ses PC, modification permettant de monter un cristal dont la fréquence supérieure fait tourner l'ordinateur bien plus vite. Le problème, si problème il y a, avait déjà été évoqué l'an dernier, engendrant une réaction un peu épidermique de la part d'IBM, qui avait modifié ses produits pour empêcher cette évolution. La raison, simple au demeurant, en était que cela permettait d'acheter un PC moins cher, un XT par exemple, et de le modifier en AT pour le prix du cristal, soit quelques dollars. Il n'était plus possible, après l'intervention d'IBM, de modifier cette pièce, les ROM vérifiant la vitesse d'horloge du PC. Il semble que certains fabricants de cristaux aient trouvé la parade et proposent de nouveau des modifications pour augmenter la vitesse de tout PC. IBM a réagi différemment cette fois, en publiant un article dans une des revues qu'elle possède, décrivant une méthode pour assurer le même accroissement des performances. En fait, il devenait impossible de mettre encore les bâtons dans les roues des usagers, le nombre de PC compatibles augmentant sans cesse et ceux-ci n'étant pas protégés au niveau de leur cristal. Ceci permettait de doper encore plus des

ordinateurs dont le prix de vente est bien moindre que celui des IBM et rendait la concurrence complètement déséquilibrée.
Les parts de marché que l'Atari 1024 ST s'etaient péniblement taillées se réduisent chaque jour un peu plus. En effet, la concurrence nouvelle de l'Apple II GS se fait sentir et la pression est sans cesse plus forte, surtout du fait des réductions importantes qui sont consenties sur l'ancienne gamme des II, le E et le C. Ils sont maintenant à un prix inférieur à celui de l'Atari et leur bibliothèque de programmes est autrement alléchante. Nous ne parlerons même pas du moribond Amiga dont le potentiel n'a jamais pu devenir un bon argument de vente. C'est d'autant plus dommage que cet ordinateur est de loin le meilleur de sa génération, la génération des machines sorties en 1984-85. Le pari de Commodore semble définitivement perdu, la société étant de plus en plus déficitaire, malgré le prêt important consenti l'an dernier par des banques.

Faites-le vous-même. C'est le pari d'une société de Cuppertino (la ville de la grosse pomme), qui propose une série de kits pour monter vous-même un compatible PC. Ainsi, pour 650 $ vous aurez un XT avec 640 ko et un écran couleur. Pour 200 $ de plus, vous pourrez installer un disque dur interne, si vous savez le câbler. Vous avez un peu peur de vos capacités ? ATD, la société en question, vous montera le tout pour 50 $ supplémentaires. Le disque monté coûte 350 $. Tout cela dans une boutique style supermarché où les pièces sont accessibles directement dans des rayons. Bientôt, acheter un ordinateur sera possible dans toute épicerie, sous emballage plastique. Ce n'est pas forcément un mal si le service après-vente suit l'évolution.

La souris gagne du terrain. Pas de piège nécessaire, il s'agit du sympathique animal qui ne bouge plus du bord du clavier. Une société d'études de marchés estime à 40 000 unités la vente mensuelle de ce périphérique. Selon le PDG de cette petite société, dans moins de deux ans, tout fabricant digne de ce nom proposera une souris en équipement standard et IBM a annoncé qu'elle sera incluse avec la future gamme éducation prévue pour le milieu de cette année. C'est la première fois que cette firme accepte de reconnaître que ce «gadget» peut être un peu plus que cela. Les mentalités évoluent... devant la poussée de la concurrence surtout. Les fabricants de souris se frottent les mains devant cette progression des ventes et espèrent que cela permettra un développement majeur des petits périphériques de confort dans un proche avenir. Parmi ceux-là, notons une intéressante tentative, hélas, avortée. Il s'agit d'une sorte de petit casque Hi-Fi, style Walkman – oh, pardon, baladeur – qui, posé sur la tête permet de suivre les mouvements de la tête et d'aligner le curseur au point désiré. Ceci libère totalement les mains qui peuvent donc rester entièrement au clavier. Formidable en traitement de texte ou en programmation, ce produit a disparu de la circulation, victime de son propre succès, la société qui l'a conçu ne pouvant fournir la trop nombreuse demande.

D'autres analystes ont publié un papier ce mois-ci, indiquant les tendances futures qu'ils estiment vraisemblables pour les micros personnels domestiques. Selon eux, les ordinateurs de ce type ne pourront survivre que dans la mesure où ils colleront de plus en plus aux standards de la micro professionnelle. Cela signifie avant tout que ces ordinateurs n'offriront plus un marché potentiel aux produits originaux, donc à risque. Comme nous le disions le mois dernier, c'est bien dommage. Maintenant, il faut bien voir un point positif majeur. En effet, une (relative) standardisation ne peut faire de mal à un domaine où l'on s'arrache les cheveux pour faire communiquer des machines. On ne peut se plaindre de l'incommunicabilité et de l'uniformité à la fois. Cela reviendrait à crier au feu, puis à assommer les pompiers qui accourent.

Je viens d'obtenir les chiffres de l'année 1986, pour les deux sociétés Commodore et Atari. La première parvient à limiter ses pertes à 31 millions de dollars contre 237 l'an passé. Pour Atari, les pertes de 85 sont devenus des gains, même si cela reste modeste. Ceci est dû au bon comportement de ces deux firmes à l'exportation. En effet, sur le marché US, leurs parts respectives diminuent encore. Pour toutes les deux, le chiffre d'affaires augmente et même sensiblement, passant de 142 à 315 millions de dollars pour Atari et de 800 à presque 1 000 millions pour Commodore. Il faut noter ici que, malgré une santé plus précaire, Commodore est toujours une bien plus grosse société que sa concurrente Atari. En fait, ce qui manque aux produits de ces marques est tout simplement des clients qui croient à leur fiabilité financière. Car comment expliquer autrement que les meilleurs produits du marché, dans leur créneau de prix, ne se vendent pas ? Car, il faut le rappeler, pour moins de 800 $ vous aurez, dans les deux gammes, un ordinateur aussi puissant que le Mac 512, avec la couleur, un bon BASIC et un bon LOGO. Ou bien manquent-ils peut-être d'une vraie diffusion de masse, pourquoi pas en supermarchés, comme a choisi de le faire un fabricant de compatible Apple IIc, qui propose le Laser 128 pour quelque 420 $? Pour ce prix, vous avez dans votre panier à provisions un IIc. Pas mal, non ? En fait, le mieux est que cela marche, car le laser 128 est un bon succès au niveau de la vente. Et en plus, cela a relancé les ventes de produits dédiés au IIc, dont certains proposés par Apple. Tout le monde est content.

Au mois prochain.

LA BIBLIOTHEQUE TECHNIQUE DES EDITIONS FREQUENCES

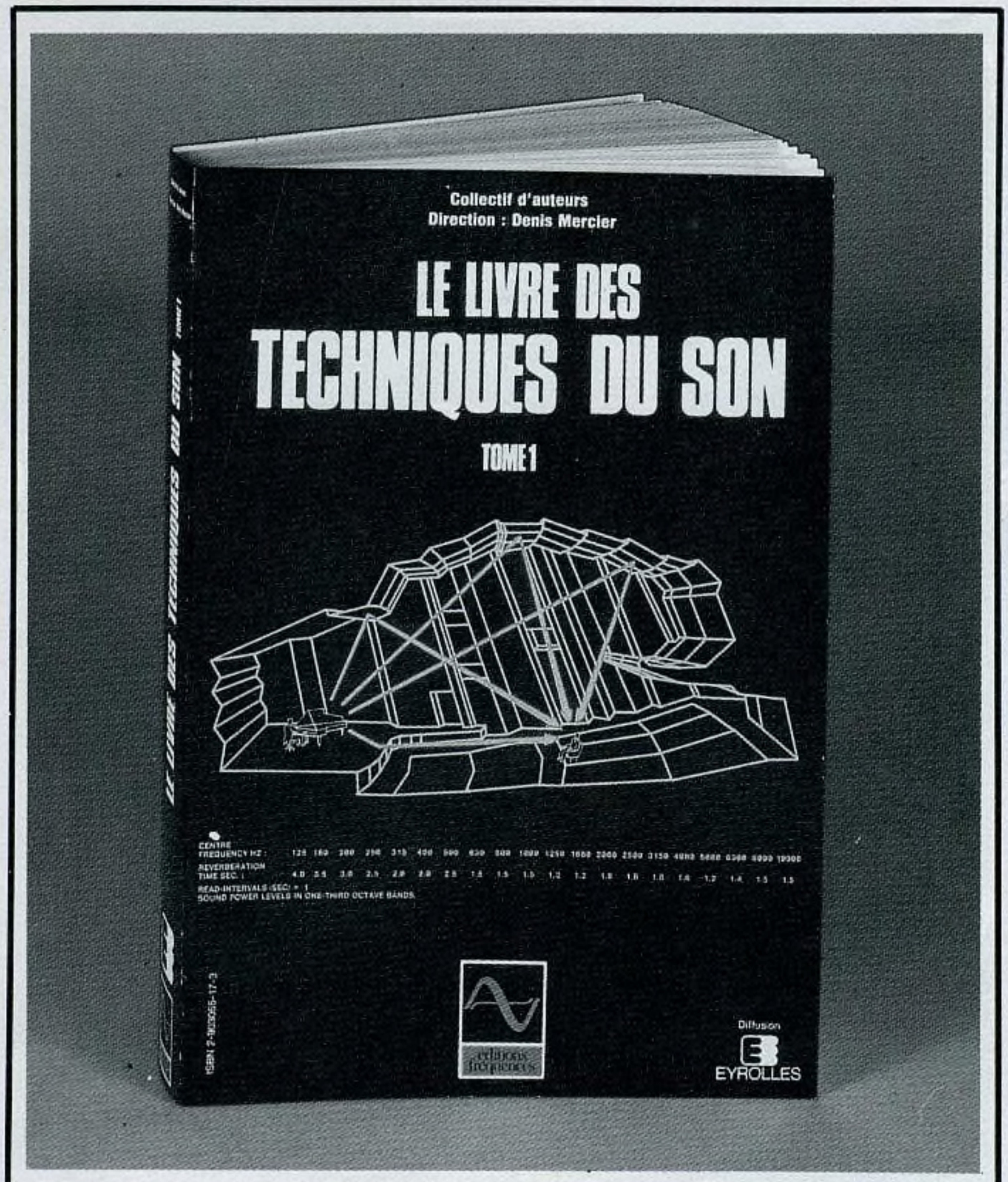
offre des ouvrages techniques **très actuels** rédigés par des auteurs passionnés et impliqués complètement dans le sujet qu'ils traitent.

Vous trouverez soit des études approfondies sur les techniques ou les technologies de votre métier, soit des initiations théoriques et pratiques de techniques ou technologies que vous désirez approcher ou mieux cerner. Vous découvrirez au verso la description des ouvrages récemment parus ainsi que les commentaires sur les additifs d'éventuelles rééditions.

Les titres dont la parution est prochaine sont également mentionnés.

La page suivante comporte la liste complète des titres, leurs codes et leurs prix.

VIENT DE PARAITRE :



- 11 auteurs
- 360 pages
- 300 schémas et illustrations

Prix : 350 F

Il y a bientôt trois ans démarrait ce travail de fond. Plus de vingt auteurs étaient sollicités pour concentrer en trois tomes les techniques du son. Le premier tome vient de paraître. Il traite de l'**Acoustique fondamentale**, des **Sources acoustiques**, de l'**Acoustique architecturale**, de la **Perception auditive**, des **Notions fondamentales de l'Electricité**, de l'**Enregistrement magnétique** ainsi que de la **Technologie audio-numérique.**

L'équipe des plus grands spécialistes actuels a été animée par Denis Mercier. Ensemble, ils ont mis sur pied un ouvrage actuellement unique au monde.

PROCHAINEMENT :

Collection jaune **Etude autour du 6809 (constructions et logiciels)** de Claude Vicidomini
L'image numérique de Jean-Marc Nasr
Le Basic structuré de Jean-François Coblentz
Divertissements en Basic de Franck Brown

Collection noire **La création musicale par ordinateur** de Frédéric Levé
Pratique de l'Amiga de Henri Cohen et François Dress

Collection noire (format 165 × 240)

	Réf.	Prix TTC
LES SYNTHETISEURS, UNE NOUVELLE LUTHERIE de Claude Gendre - 184 p. - 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	E 15	140 F
LES HAUT-PARLEURS de Jean Hiraga - 320 p. - 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 01	165 F
INTRODUCTION A L'AUDIO-NUMERIQUE de Jean-Pierre Picot - 160 p. - C'est le premier ouvrage paru en langue française traitant de l'audio numérique ; écrit par un professionnel, avec rigueur et simplicité, il explique brillamment les bases de cette technique : quantification, conversion, formats, codes d'erreurs	E 05	155 F
L'OPTIMISATION DES HAUT-PARLEURS ET ENCEINTES ACOUSTIQUES de Charles-Henry Delaleu - 240 p. - 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 04	154 F
LES MAGNETOPHONES de Claude Gendre - 160 p. - 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 02	92 F
LES MAGNETOSCOPES ET LA TELEVISION de Claude Gendre - 256 p. - 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 03	155 F
L'ELECTRONIQUE DES MICRO-ORDINATEURS de Philippe Faugeras - 128 p. - 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 : mémoire, clavier, écran, interfaces de toutes sortes	E 06	150 F
PERIPHERIQUES : INTERFACES ET TECHNOLOGIE de Philippe Faugeras - 136 p. - Faisant suite à la parution de «L'électronique des micro-ordinateurs», cet ouvrage s'adresse aux électroniciens désireux de s'initier aux montages périphériques des micro-ordinateurs, interfaces en particulier, qui permettent la communication avec monde extérieur	E 22	150 F
SELECTION DE L'AUDIOPHILE - TOME 1 : L'ELECTRONIQUE 256 p.	E 13	155 F
SELECTION DE L'AUDIOPHILE - TOME 2 : LES TRANSDUCTEURS 256 p. Introuvable aujourd'hui, une sélection des meilleurs articles de la célèbre revue «L'Audiophile». Le tome 1 traite de l'électronique audio à tubes et transistors. Dans un esprit identique, le tome 2 traite du domaine passionnant que constituent les transducteurs en audio.	E 12	165 F
LE MINI STUDIO de Denis Fortier - 160 p. - Le monde de l'audio évolue... Un secteur d'activité entièrement neuf vient d'apparaître : les mini-studios. L'ouvrage de Denis Fortier, ingénieur du son, aborde le sujet de la manière la plus globale. Après les données physiques indispensables, le choix des maillons, la manière d'installer et d'exploiter	E 25	140 F
LES TECHNIQUES DU SON Collectif d'auteurs sous la direction de Denis Mercier - 360 p. - Le Livre des Techniques du Son est le premier ouvrage interdisciplinaire en langue française s'adressant aux professionnels du son.	E 33	350 F

Collection rouge (format 135 × 210)

CONSEILS ET TOURS DE MAIN EN ELECTRONIQUE de Jean Hiraga 160 p. - 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 07	68 F
LES LECTEURS DE COMPACT-DISCS 200 p. - 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 10	130 F
LEXIQUE DE L'ELECTRONIQUE ANGLAIS-FRANÇAIS de Jean Hiraga - 72 p. - 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 09	65 F
FILTRES ACTIFS ET PASSIFS POUR ENCEINTES ACOUSTIQUES de Charles-Henry Delaleu - 160 p. - 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 11	85 F
17 MONTAGES ELECTRONIQUES de Bernard Duval - 128 p. 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 14	95 F
WEEK-END PHOTO de Philippe Folie-Dupart - 208 p. - 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	L 20	130 F

Collection jaune (format 210 × 270)

INITIATION A LA ROBOTIQUE 96 p. - 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 et les fans de cybernétique enfin réunis ! (épuisé)	P 08	115 F
INITIATION A LA MICRO-INFORMATIQUE COURS 1er CYCLE - LE VOLUME 1 de Claude Polgar - 272 p.	P 16	130 F
INITIATION A LA MICRO-INFORMATIQUE COURS 1er CYCLE - LE VOLUME 2 de Claude Polgar - 208 p.	P 17	130 F
INITIATION A LA MICRO-INFORMATIQUE COURS 1er CYCLE - LE VOLUME 3 de Claude Polgar - 250 p. 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	P 27	190
INITIATION A L'ELECTRONIQUE DIGITALE de Philippe Duquesne - 104 p. - Ce cours d'initiation à l'électronique digital 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 19	95 F
INITIATION AUX MICROPROCESSEURS de Philippe Duquesne - 136 p. - 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 18	95 F
INITIATION TV : RECEPTION, PRATIQUE, MESURES, CIRCUITS de Roger-Charles Houzé - 136 p. - 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	P 21	135 F
INITIATION A LA MESURE ELECTRONIQUE de Michel Casabo - 120 p. - 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 23	140 F
INITIATION AUX AMPLIS A TRANSISTORS de Gilles Le Doré - 96 p. - 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 24	130 F
INITIATION AUX AMPLIS A TUBES de Jean Hiraga - 152 p. - 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	P 26	155 F
INITIATION A L'ELECTRICITE ET A L'ELECTROTECHNIQUE de Roger Friederich - 110 p. - Vous trouverez aisément en librairie des ouvrages d'initiation à l'électronique ou aux techniques les plus avancées des circuits intégrés, etc. Mais si vous désirez une initiation aux bases de l'électricité et de l'électrotechnique sans vous en remettre à des ouvrages scolaires, alors vous ne trouverez pas !	P 28	150 F
INITIATION A LA VIDEO LEGERE - THEORIE ET PRATIQUE de Claude Gendre - 72 p. - Choix d'un standard ? Camescopes VHS, VHS-C ou 8 mm ? Connexion ? Compatibilité ? Accessoires ? Montage ? Enfin... comment filmer ? Le nouveau livre de Claude Gendre répond à toutes ces questions. Cet ouvrage essentiellement pratique n'a pas d'équivalent en librairie aujourd'hui	P 29	100 F
LES MONTAGES ELECTRONIQUES de Jean-Pierre Lemoine - 276 p. - Véritable encyclopédie. Plus de 1 000 dessins. 25 montages originaux	P 30	250 F
LE TELEPHONE ET LES RADIOTELEPHONES de Roger-Charles Houzé - 96 p., 73 schémas	P 31	130 F
LES BASES DE L'ELECTRONIQUE de Raymond Breton - 84 p., 162 schémas Vous ne connaissez pas l'électronique : ce livre vous permet d'accéder aux bases nécessaires mais néanmoins d'atteindre un niveau vous permettant d'aborder des constructions de bases	P 32	120 F

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) ouvrages(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 (épuisé) ☐	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 ☐	E 25 ☐	P 26 ☐	P 27 ☐	P 28 ☐	P 29 ☐	P 30 ☐
P 31 ☐	P 32 ☐	E 33 ☐							

Frais de port : + 12 F par livre commandé, soit la somme totale ci-jointe, de Frs par CCP ☐ Chèque bancaire ☐ Mandat-lettre ☐

Nom ..... Prénom .....

Adresse .....

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

AutoCad
D.A.O.

Charles-Henry Delaleu

Nous vous présentons ce mois AUTOCAD, le logiciel de D.A.O. On appelle D.A.O. le dessin assisté par ordinateur ; à ne pas confondre avec C.A.O. Le D.A.O. automatise les tâches effectuées par un dessinateur en dessin industriel ; la C.A.O. évoque un concept plus complet qui associe le D.A.O. avec des fonctions de calcul et de recherche avancées. La C.A.O. permet de mettre au point, au niveau de la recherche, des structures mécanique, électronique, de bâtiment, etc.

Bien qu'AutoCad soit, à l'origine, un programme de D.A.O., son énorme succès commercial a permis de développer plusieurs extensions dont des extensions de C.A.O. Ainsi, AutoCad peut-il devenir très efficace pour les bureaux d'études et les laboratoires.

Il convient de noter les énormes changements intérieurs survenus dans le monde du D.A.O. ces dernières années. En effet, il y a peu de temps encore, Computervision régnait en maître absolu sur le marché du dessin assisté par ordinateur. Il fallait investir entre 1,5 et 3 millions de francs pour un équipement classique.

Actuellement, grâce aux progrès réalisés en micro-informatique et à l'arrivée de progiciels de type AutoCad, il est possible de commencer à travailler pour des budgets inférieurs à 100 000 F.

Le programme AutoCad est un progiciel appliqué au dessin assisté par ordinateur. Les applications du D.A.O. sont très nombreuses. Grâce à l'architecture d'AutoCad, un dessin peut être effectué ou modifié avec une grande facilité. La vitesse d'exécution est très rapide. AutoCad gère au maximum les possibilités des coprocesseurs arithmétiques du type 8087-80287, etc. Il est donc préférable, afin d'utiliser au maximum ce progiciel, d'utiliser un coprocesseur.

Les principales applications d'AutoCad sont :

- dessins d'architecture de toutes sortes
- dessins d'architecture d'intérieur
- schémas de systèmes et de processus
- dessins pour l'électronique, l'électricité
- dessins pour les applications d'ingénierie civile et mécanique
- dessins des pièces mécaniques de toutes sortes
- traces de fonctions mathématiques et scientifiques
- dessins artistiques.

AutoCad existe en standard en plusieurs versions :

- la version de base
- l'extension ADE-1 (cotations, unités, hachures)
- l'extension ADE-2 (positionnement dynamique)
- l'extension ADE-3 (dessins en 3D)

CONFIGURATION

AutoCad devra, de préférence, être implanté sur un XT ou mieux un AT équipé d'un disque dur.
En ce qui concerne les entrées, il est possible d'utiliser :

- le mode clavier
- une souris
- une table à digitaliser.

En mode sortie, une table traçante permettra de dessiner le travail effectué avec AutoCad.
Une des premières choses à faire lors du premier démarrage d'AutoCad concerne la configuration du système. Un nombre impressionnant de possibilités est offert. En effet, la majorité des souris, des écrans graphiques, des digitaliseurs, des tables traçantes, etc., sont directement interfaçables. Un nombre important d'utilitaires est préprogrammé pour auto-configurer l'ensemble. On choisit ses options à l'aide de menus. Cette opération est ensuite sauvegardée. Grâce à cet outil, la préparation ne pose aucun problème. Dans le manuel de service, on trouvera des indications pour paramétrer les swtiches des différentes extensions machines (digitaliseur, table traçante, etc.).

CHARGEMENT D'AUTOCAD

Le chargement d'AutoCad est simple. Il suffit d'appeler le menu principal grâce au fichier ACAD.
Commande du menu principal :

- Choix 0 : Sortir d'AutoCad
- Choix 1 : Commencer un nouveau dessin
- Choix 2 : Editer un dessin existant
- Choix 3 : Sortie traceur du dessin
- Choix 4 : Configuration d'AutoCad
- Choix 5 : Utilitaires des fichiers
- Choix 6 : Compiler des fichiers formes/texte
- Choix 7 : Conversion des anciens fichiers DWG

AutoCad peut être utilisé avec un ou deux écrans. En mode deux écrans, on peut avoir un écran haute définition couleur pour le dessin et un écran noir et blanc pour les commandes.

LES DESSINS

Il est possible de travailler de deux manières sous AutoCad.

1. Les dessins classiques

En dessin classique, plusieurs choix sont offerts à l'utilisateur. On peut dessiner à main levée en utilisant par exemple des routines de dessin du style : arc, cercle, ligne, etc.

2. Les dessins automatiques

Grâce à un système de blocs, il est possible d'entrer en mémoire de masse plusieurs symboles ou dessins élémentaires afin de pouvoir les réutiliser à volonté. De ce fait,

dans le cas de l'électronique par exemple, l'utilisation de blocs est très intéressante. Il suffit de dessiner une fois pour toutes une résistance, un condensateur, un transistor, etc., et de les mettre en mémoire. Lors de la réalisation d'un nouveau dessin, chaque composant est rappelé dès que l'on en a besoin. Ainsi le travail est-il très rapide. Le gain de temps est énorme.

PRINCIPALES COMMANDES D'AUTOCAD

DESSIN

Arc	: Permet de dessiner des arcs de cercle en plusieurs modes
Cercle	: Permet de dessiner des cercles en plusieurs modes
Insérer	: Insérer des blocs à l'écran
Ligne	: Permet de tracer des lignes en automatique
Solide	: Permet de tracer des solides
Texte	: Ecrire du texte sur l'écran
Trace	: Tracer des lignes d'une épaisseur donnée
Point	: Permet de mettre des points sur l'écran (trame, repère, etc.)
Formes	: Reprendre des formes créées avec charges
Mainlev	: Dessin à main levée

ECRAN

Pan	: Déplacement d'un graphique dans tous les sens
Regen	: Régénération de tout le dessin
Redess	: Supprime les points à l'écran
Zoom	: Zoom sur tout ou partie de l'écran
Vues	: Sauve et rappelle une partie du dessin
Raptext	: Remplace le texte par des lignes
Regnauto	: Empêche les régénérations automatiques

BLOCS

Blocs	: Création, mise en mémoire et rappel de blocs de dessin

COTATION

Cotation	: Cotation semi-automatique Lignes d'extension Texte de cotation Tolérances Limites, etc.

EDITION

Réseau	: Répéter les graphismes existants en Circle et en Rectangle
Coupure	: Couper une ligne ou une forme
Changer	: Enlever un morceau de dessin et le mettre ailleurs avec effacement
Copie	: Copier un morceau de dessin sans effacement
Effacer	: Efface une fenêtre ou le dernier graphisme
Raccord	: Raccorder deux lignes avec un rayon
Miroir	: Projection d'un graphisme en X ou en Y
Déplacer	: Déplacer un ou plusieurs éléments du dessin
Repet	: Idem à Réseau

HACHURE

Hachure	: Permet de réaliser tout type de hachures dans des surfaces

RENSEIGN (Renseignement)

Aire	: Permet de préciser des points pour le calcul d'un polygone
Dbliste	: Sortir un listing de points du dessin en X, Y, Z
Distance	: Distance, angle en coordonnées X, Y
Id	: Position d'un point en coordonnées X, Y
Liste	: Liste des positions définies dans une fenêtre
Etat	: Renseignements divers, plans mémoire, plans, etc.

PLANS

Plan : Choix du plan de travail
Typelign : Choix du type de ligne
Echltp : Contrôle le facteur d'échelle, hors écran

MODES

Axes : Mise en place à l'écran des traits d'axes
Coord : Donne à l'écran les coordonnées X et Y
Dragmod : Visualiser dynamiquement la position des éléments dans le dessin avec l'option drag
Grille : Espacement des points de la grille avec F7
Isometr : Choix du plan de travail
Ortho : Actif : tracé horizontal et vertical ; inactif : libre
Objres : Choix du mode d'accrochage des objets, nodal
Resol : Résolution de la grille, actif ou inactif
Tablet : Configuration du digitaliseur

FLOT : Sortie sur table traçante

UTILIT :

Ouverture : Taille de la cible en mode accrochage. Objres
Dxf : Change un fichier en dessin AutoCad. Ex. : DBASE III +
Fichier : Utilitaire de contrôle des fichiers
Aide : Affiche le menu d'aide
Limite : Changement des limites de l'écran
Menu : Mise en place d'un menu existant sur le disque dur
Purge : Pour détruire sélectivement les objets non utilisés de votre dessin
Renomme : Nomme à nouveau un bloc, un plan, etc...
Vuecran : Mvue : création de vues écran
Vuecran : rappel des vues
Style : Création ou modification des styles d'écritures
Unités : Unités mathématiques, décimales, etc.
Sauvgrd : Sauve votre dessin en restant dans l'éditeur (en cas de panne de courant)
Fin : Sauve le dessin et retourne au menu principal
Finsv : Sauve le nouveau dessin et garde celui d'avant
Quitter : Pour quitter AutoCad

EQUIPEMENT REQUIS

L'équipement minimum pour AutoCad consiste en :

- une unité centrale + un clavier
- une carte graphique + un écran graphique
- une table traçante

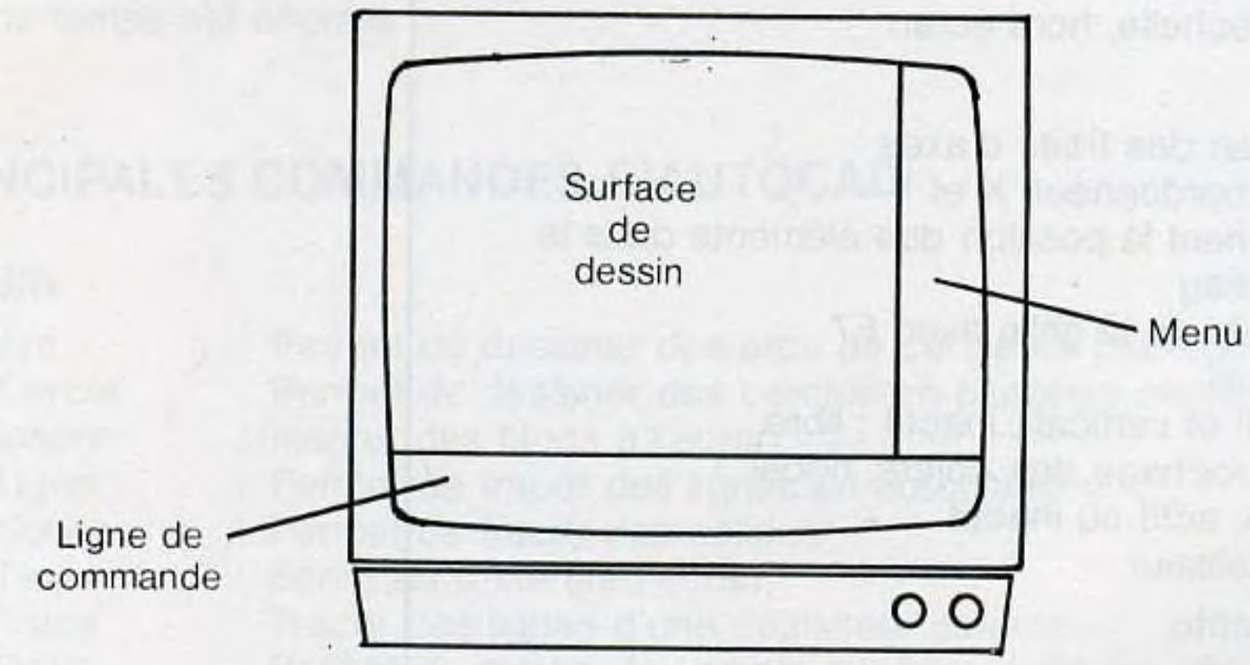
ELEMENTS DE SAISIE

Bien qu'il soit possible de travailler en mode saisie uniquement par le clavier, il est tout à fait souhaitable de disposer d'une souris ou d'une table à digitaliser. Ces périphériques servent à pointer les différentes commandes et, bien sûr, à déplacer le curseur sur l'écran pour dessiner.

L'ECRAN GRAPHIQUE

Afin de réaliser des dessins de qualité, la résolution de l'écran graphique devra être la meilleure possible, le minimum étant de 640 × 350 points. Pour obtenir une meilleure lisibilité, le choix se portera sur un écran couleur.

Visualisation de l'écran



DESSIN AUTOCAD

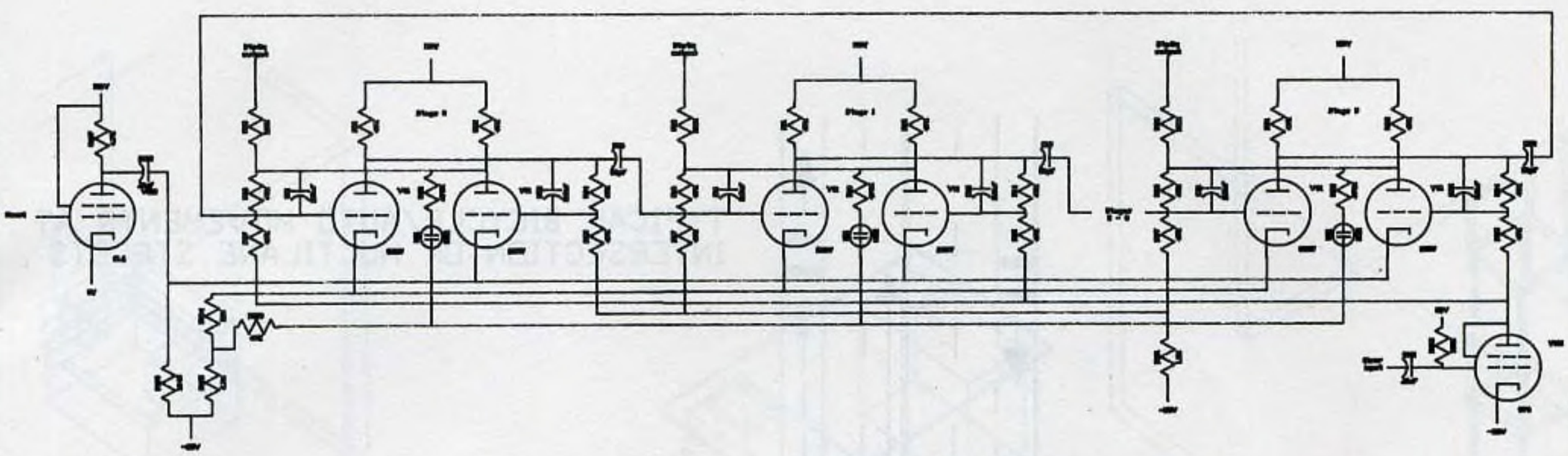
Un dessin sous AutoCad peut avoir n'importe quelles dimensions. L'unité est choisie par l'utilisateur. Un système de coordonnées cartésiennes est utilisé pour définir les points du dessin. Les coordonnées X définissent l'abscisse, les coordonnées Y l'ordonnée. Chaque point est donc défini par ses coordonnées X, Y. Le point (0, 0) est situé par convention en bas à gauche de l'écran. Les limites du dessin sont un point maximum en haut à droite de (30, 18). Les limites de l'écran sont au point (15, 12). Il est donc possible comme pour un tableur de se déplacer sur la surface de travail.

MANIEMENT DU PROGRAMME

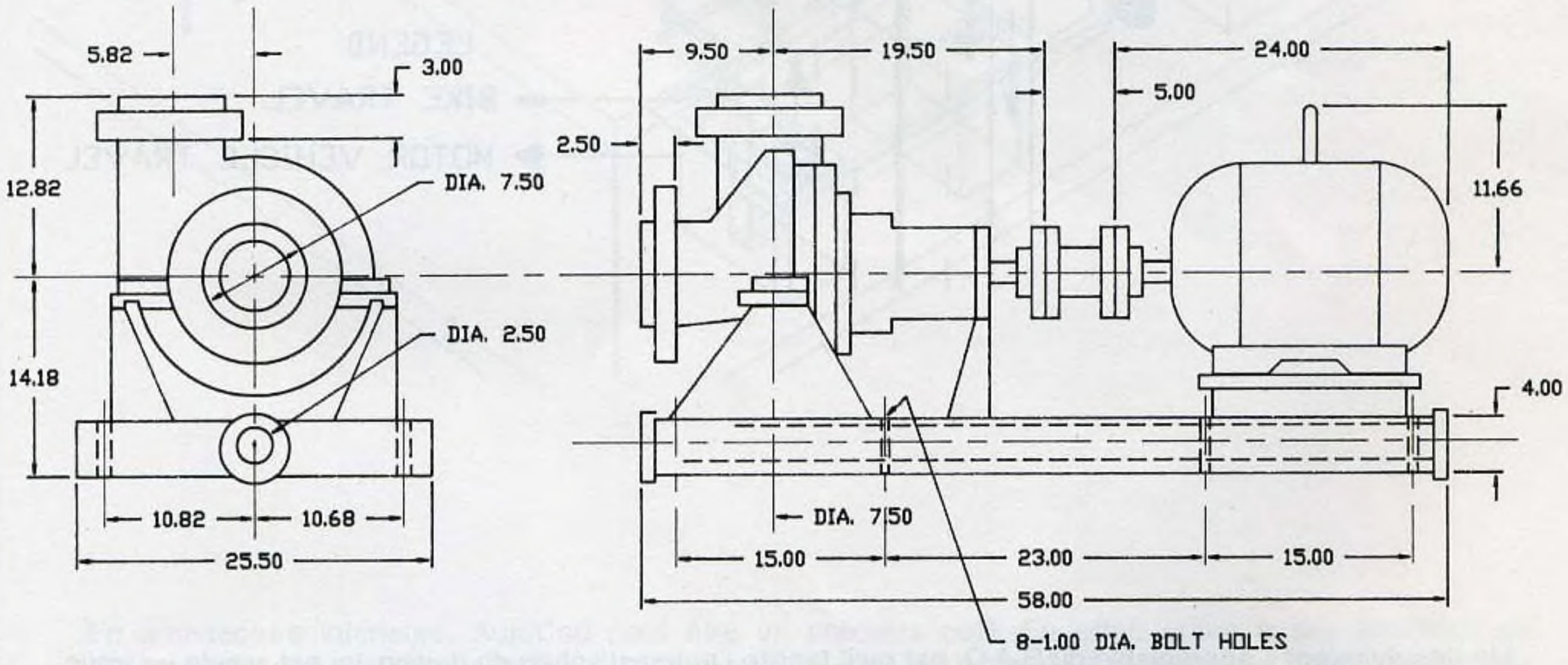
Le maniement du programme est très simple. Toutes les commandes sont accessibles par menu. Un menu principal appelle un menu secondaire, etc.
Une commande AIDE permet d'obtenir à l'écran une explication sur les commandes AutoCad ; ce qui évite de se reporter à la documentation au moindre accroc.

CONCLUSION

Grâce à AutoCad, l'univers de la D.A.O. devient enfin plus accessible aux petites sociétés. C'est, sans aucun doute, un outil efficace et performant. Grâce à ses possibilités, ce peut être un premier pas vers la C.A.O. En effet, beaucoup de S.S.C.I. ont développé de nombreuses interfaces avec ce progiciel, ce qui le transforme en véritable atelier de C.A.O. Grâce à deux ou trois journées de formation, il est possible de l'exploiter au maximum.

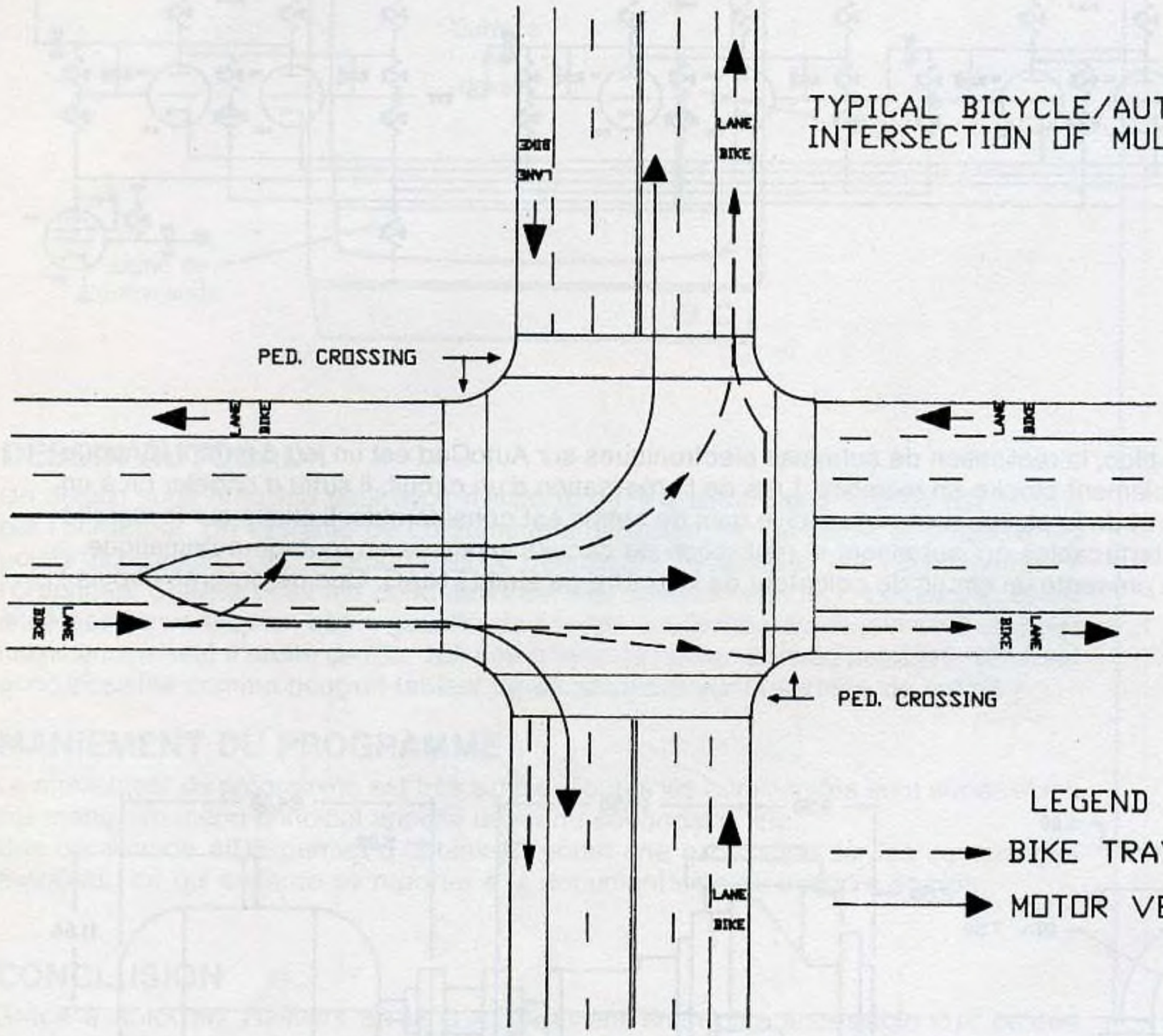


Grâce à la fonction bloc, la réalisation de schémas électroniques sur AutoCad est un jeu d'enfant. Chaque symbole est préalablement stocké en mémoire. Lors de la réalisation d'un circuit, il suffit d'appeler un à un chaque composant et de le placer sur l'écran. Ici le gain de temps est considérable. Il existe sur le marché des programmes interfaçables qui autorisent la réalisation de circuits imprimés en routage automatique.
Le dessin ci-contre présente un circuit de compteur de l'ancêtre qu'était l'ENIAC. Que de progrès depuis !



Motor/Pump Mechanical Assembly

Le dessin industriel : il ne fait aucun doute qu'en dessin industriel, la D.A.O. est un outil décisif.



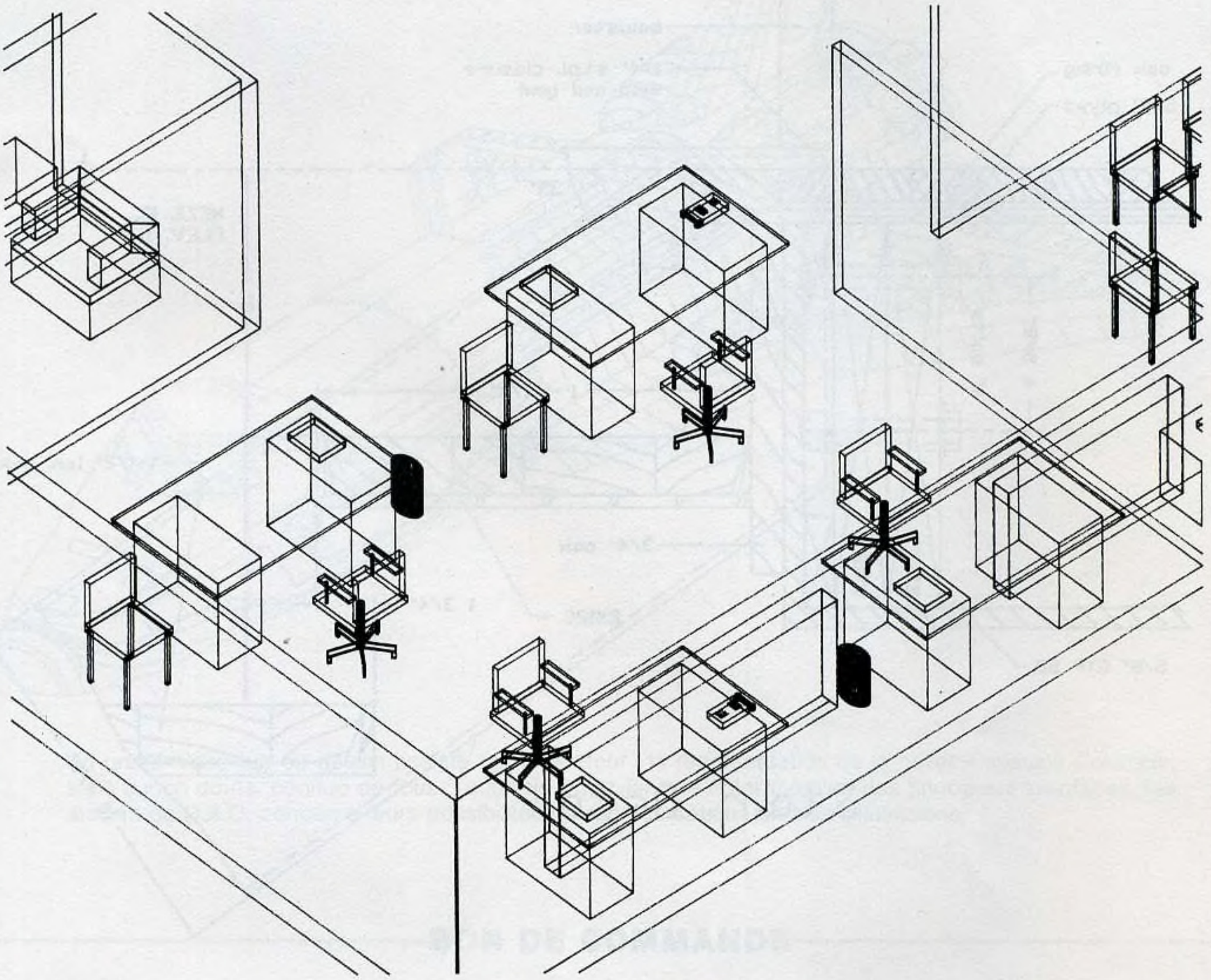
TYPICAL BICYCLE/AUTO MOVEMENTS AT INTERSECTION OF MULTILANE STREETS

LEGEND

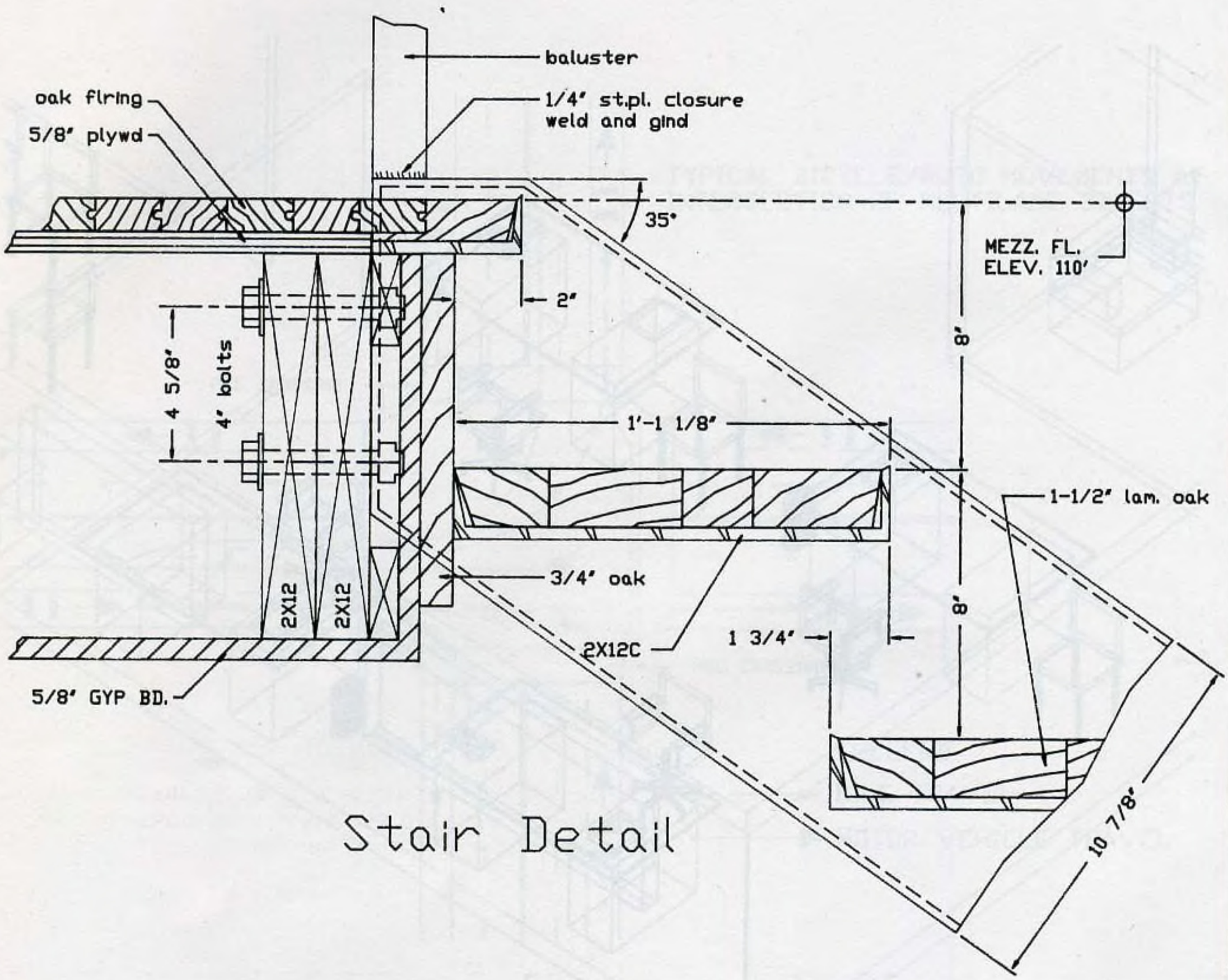
BIKE TRAVEL

MOTOR VEHICLE TRAVEL

Un des avantages appréciable du D.A.O. est qu'il facilite l'automatisation du dessin. Ici est simulé un trafic routier sur une intersection. Ainsi, les responsables de la sécurité routière peuvent-ils créer une trame de base et simuler sur celle-ci différents cas. Les positions des flèches sur la chaussée, les feux tricolores, etc., seront optimisés après l'étude de plusieurs scénarios.

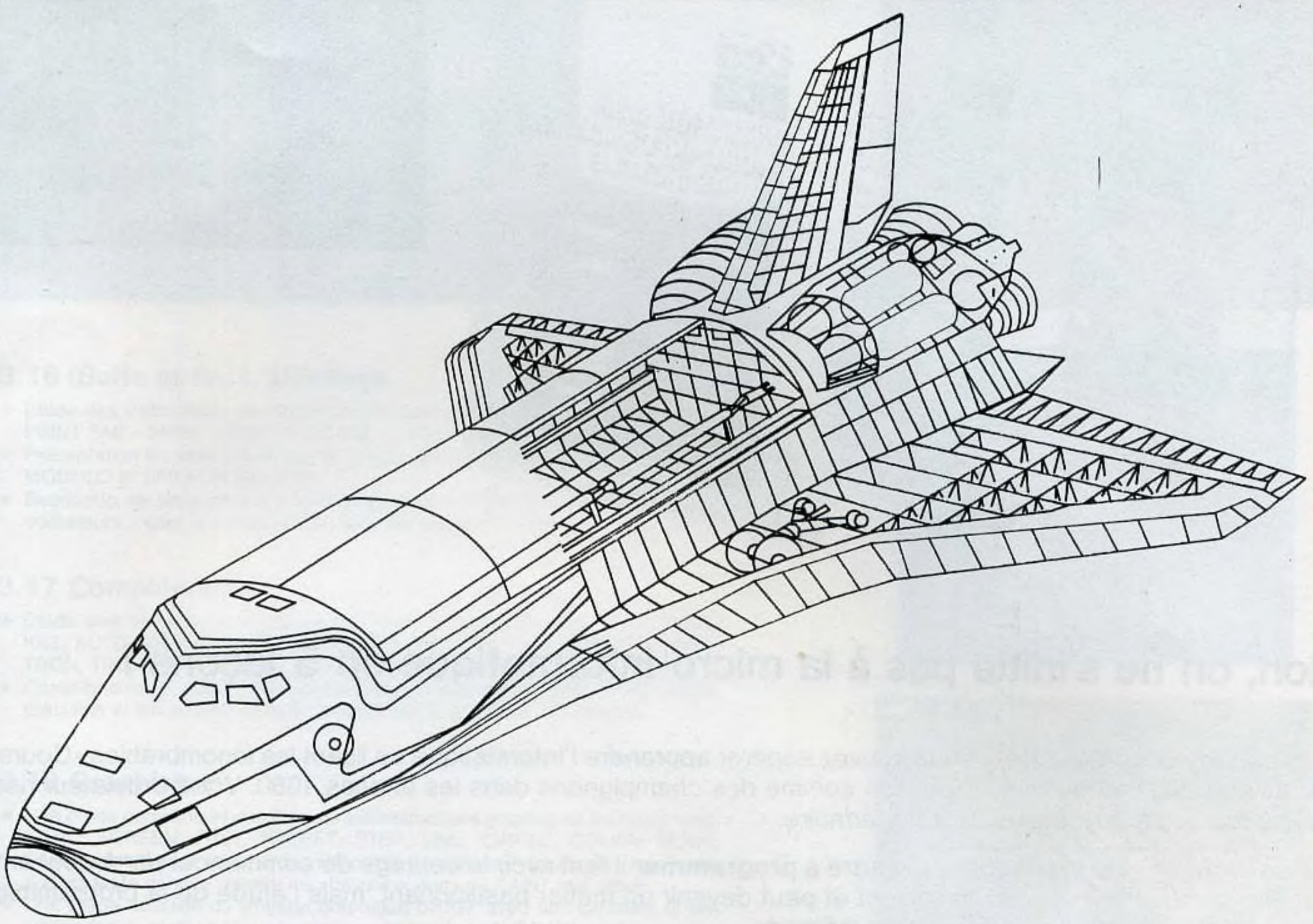


En architecture intérieure, AutoCad peut être un précieux outil. En effet, grâce à ses fonctions de déplacement et à son extension trois dimensions (ADE-3), il est possible d'étudier la meilleure position des meubles dans une pièce. Ici, les bureaux, les chaises, etc., peuvent être déplacés afin de trouver le meilleur remplissage tout en optimisant la fonctionnalité.



Stair Detail

A part la création des plans d'une maison, d'un immeuble, AutoCad autorise de nombreuses autres fonctions en architecture. Les zooms, les coupes, etc., permettent d'étudier chaque détail d'un bâtiment. Il s'agit ici de la coupe d'un escalier.



Un grand classique du dessin assisté par ordinateur : la représentation de la navette spatiale Columbia, sans aucun doute, connue de tous. Toutefois il convient de noter que l'un des principaux avantages des ateliers de D.A.O. concerne leurs possibilités de représentations en trois dimensions.

BON DE COMMANDE

Pour compléter votre collection de Led-Micro

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

Je désire le n° ☐

1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 27 29 32 34 35

(cocher le ou les n^{os} désirés).

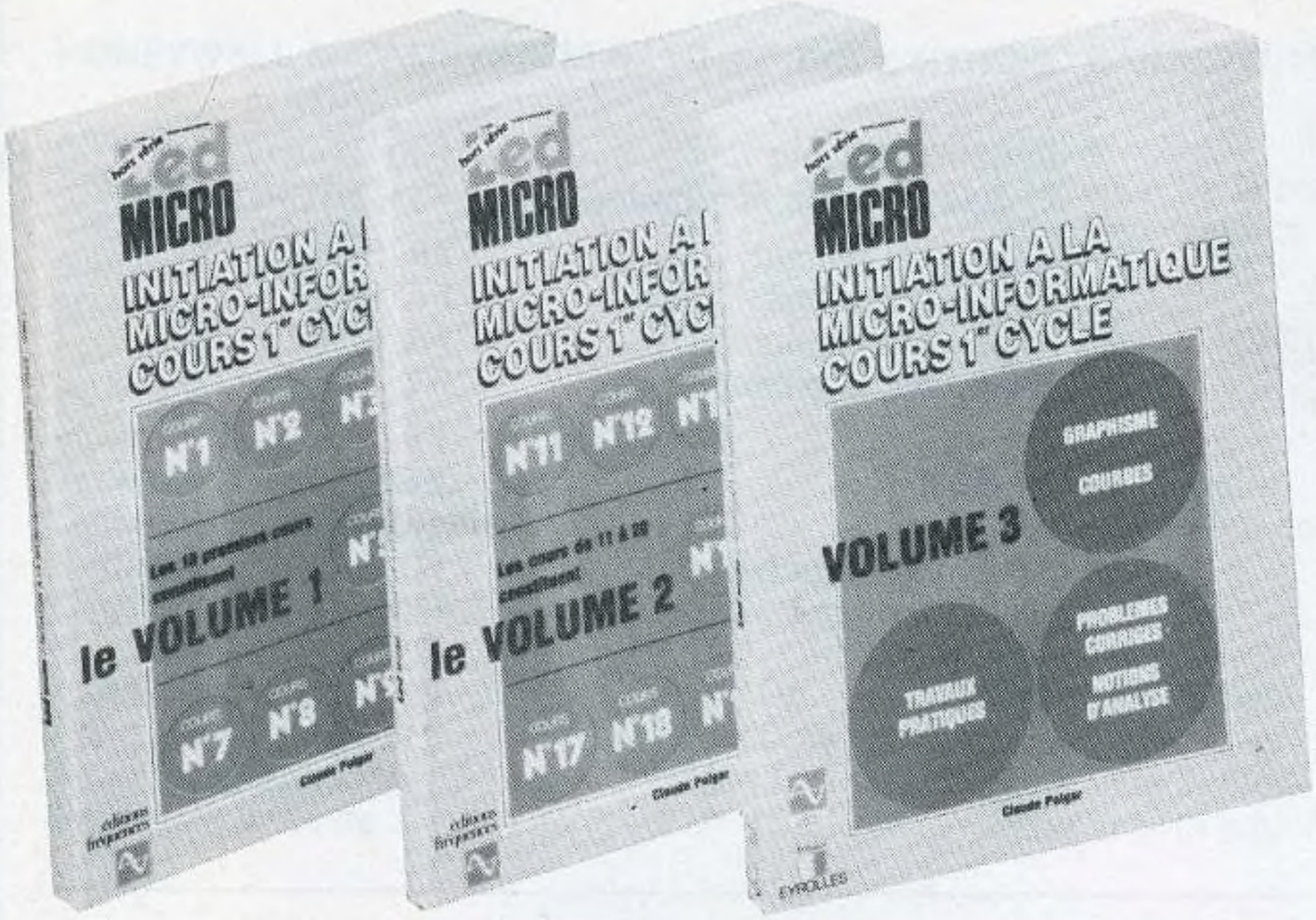
AU PRIX DE 22 F par numéro (port compris).

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

NOM ..... Prénom .....

Adresse .....

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



Le cours d'initiation le plus complet

+ de 700 pages

Non, on ne s'initie pas à la micro-informatique en 5 leçons !

Si vous croyez au Père Noël vous pouvez espérer apprendre l'Informatique en lisant les innombrables «Cours de BASIC pour débutants» qui ont poussé comme des champignons dans les années 1980. Votre ordinateur risque de finir ses jours au-dessus de votre armoire.

Mais si vous voulez vraiment apprendre à programmer il faut avoir le courage de commencer par A pour arriver à Z. Programmer est un loisir intelligent et peut devenir un métier passionnant, mais l'étude de la programmation nécessite un minimum de travail et de méthode.

Etre sérieux – c'est le pari que fit la revue LED-MICRO en publiant à partir de 1985 les 20 premiers cours de C. Polgar. **Plus de 40 000 lecteurs** les ont suivis. Ce succès nous a conduit à demander à C. Polgar de remettre son cours à jour et de le compléter. Le résultat : un ouvrage épais (3 tomes, plus de 700 pages format 21 × 27), permettant d'acquérir agréablement des connaissances solides.

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)
le tome 3 ☐ **200 F** (190 F + 10 F de frais de port)

Ci-joint mon règlement par :

☐ CCP ☐ Chèque bancaire ☐ Mandat

Nom .....

Prénom .....

Adresse .....

.....

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

Une seule
parmi près de 600 lettres
de lecteurs :

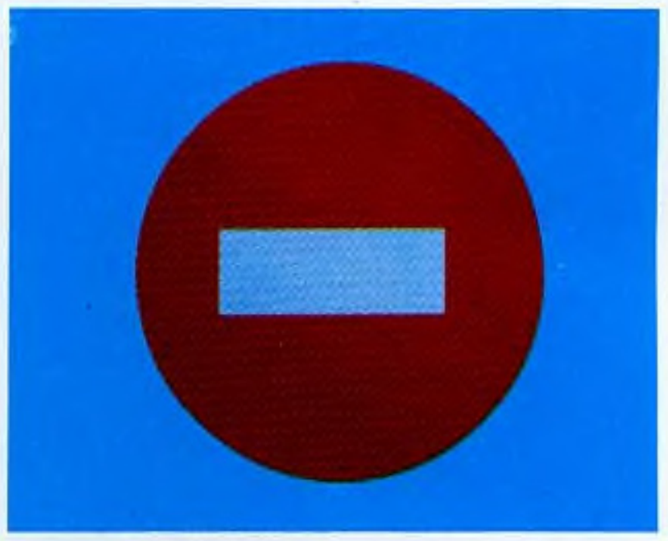
J'enseigne les mathématiques dans une Université de Sciences Humaines et j'ai été amenée, alors que n'avais moi-même reçu aucune formation à la micro-informatique, à initier des étudiants de 1re année de Mathématiques et Sciences Sociales (MASS) à la programmation en S-BASIC (sur Goupil-3), dans le but 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

Initiation à la Micro-Informatique 1er Cycle Tome 3

(enfin paru !)

3.16 (Suite et fin) L'affichage

★ Etude des instructions permettant d'effectuer des présentations «évoluées» : PRINT TAB - PRINT USING - LOCATE - COLOR en mode texte.
★ Présentation en tableaux de toutes sortes grâce à la pratique des opérateurs MODULO et DIVISION ENTIERE.
★ Beaucoup de programmes utilisent des assemblages de ces instructions et opérateurs... dont la combinaison n'est pas toujours facile.

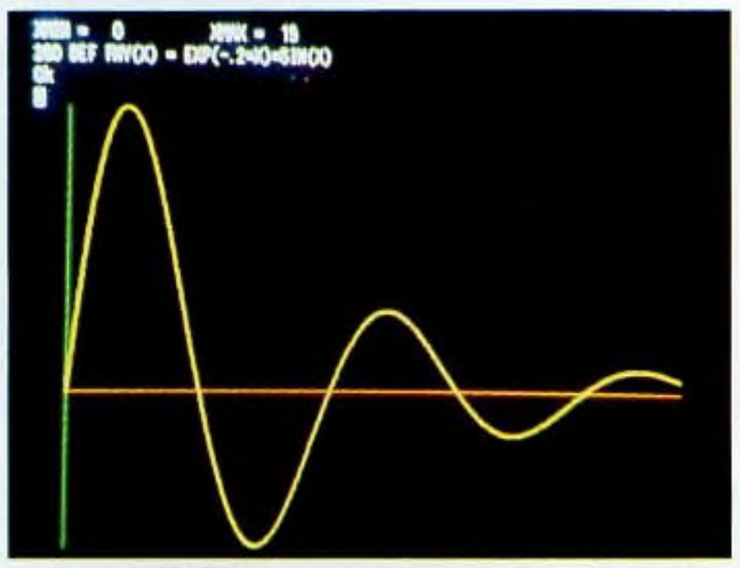


3.17 Compléments

★ Etude des dernières instructions, fonctions et variables du cycle 1 : FILES, KILL, AUTO, ON ERROR GOTO, RESUME, ERR, ERL, DELETE, EDIT, RENUM TRON, TROFF, STOP, CONT, KEY ON, KEY OFF, FIX, BEEP.
★ Compléments de cycle 1 qui sont maintenant accessibles aux éléves : sur la précision et les erreurs dues à l'arrondi, sur la sélection, les boucles.

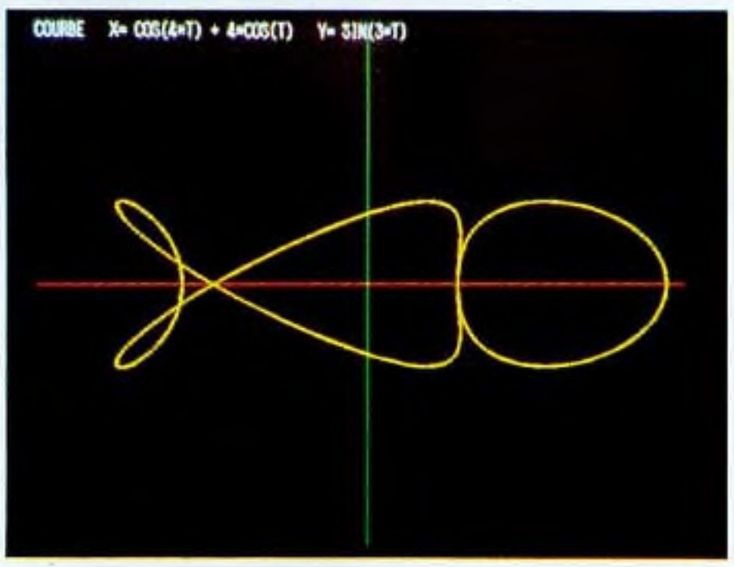
3.18 Graphisme

★ Une étude complète et détaillée sur les instructions graphiques en haute résolution : SCREEN, PSET, PRESET, STEP, LINE, CIRCLE, COLOR, POINT, PAINT, sans éluder aucune des difficultés et «pièges» classiques : l'incrustation de texte dans le dessin, les «bavures» dues au PAINT mal utilisé.
★ Une étude détaillée du langage graphique DRAW, avec ses subtilités et ses pièges (sous-chaînes X, paramètres variables dans le DRAW, etc.).
★ De nombreux exercices avec leurs solutions (80) et leurs illustrations sur des photos d'écran en couleur (48 photos).



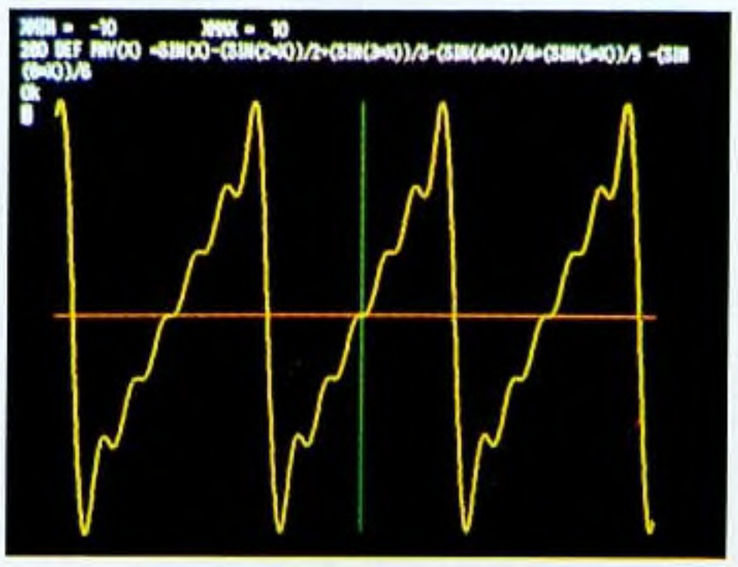
3.19. Dessin des courbes

★ Un chapitre séparé du graphisme général (chapitre 3.18) de façon à ce que les «non matheux» puissent le sauter sans remords : ils ne seront pas punis !
★ Pour les matheux : une excellente révision et illustration des courbes de toutes sortes : Y = f (x), courbes paramétrées, courbes en coordonnées polaires, avec des exemples utiles : courbes d'amortissement, astroïde, cardioïde, décomposition d'une fonction périodique par une série de Fourier.



3.20. Révision générale

★ L'enchaînement des notions selon l'ordre «pédagogique» qui a été utilisé jusqu'ici est bien différent de l'ordre «logique». Autant qu'un cours d'anglais suit un ordre différent de celui (plus logique !) d'une grammaire anglaise.
★ Tout ce qui a été enseigné jusqu'ici résumé en 30 pages. Une référence pour retrouver la notion dont on a besoin à travers le cours et ses exercices. Mais aussi une réflexion sur la structure d'un langage informatique, d'où une préparation à la lecture des cours de PASCAL (par exemple !).



3.21. Techniques de mise au point

★ Les outils de base : étude des éditeurs de texte, connaissance et interprétation des messages d'erreur.
★ Comment rechercher et corriger ses erreurs.
★ La représentation du dialogue homme-machine, pour noter l'expérience que vous acquérez par la pratique.

3.22. Problèmes de synthèse - Notions d'analyse

C'est à la fois la conclusion, la partie la plus originale et la plus utile de ce cours. L'auteur ne se contente pas de fournir une liste de problèmes avec leur solution : il se met à la place du programmeur débutant en essayant de décortiquer le «processus de réflexion» qui fait passer de l'énoncé d'un problème à sa solution: une initiation pratique à l'analyse.

1 livre broché de 248 pages pages 21×27, dont 8 pages en couleur

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