hors série

Led MICRO

PROGRAMMATION

COURS 2ème CYCLE

ISSN 0757-6889

COURS
N° 40
Suite
2e cycle
N° 20

COURS DE PASCAL
l'allocation dynamique

COURS DE PROGRAM-MATION APPROFONDIE :
éditeur de texte

dB Adresse

Nathalie 2

M 1988 - 40 - 18,00 F
3791988018008 00400

MENSUEL MAI 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

ISSN 0757-6889

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 A la revue
Tél. : 46.07.01.97
Secrétaire responsable
Annie Perbal

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

Réalisation
Composition-Photogravure
Edi Systèmes

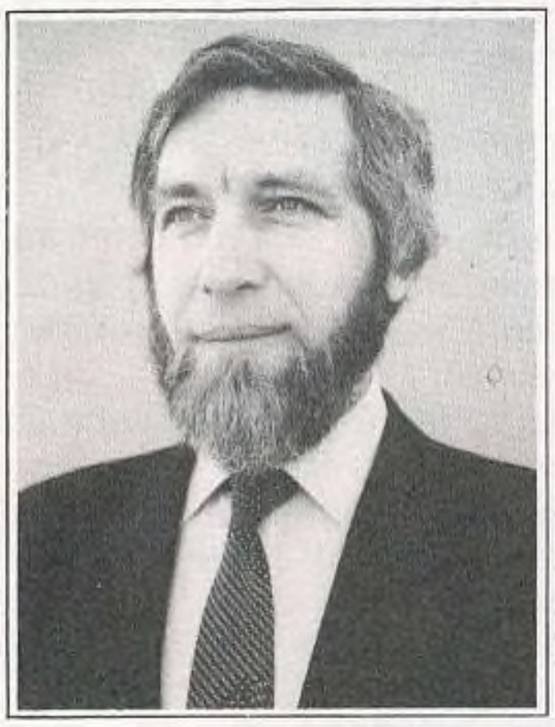
Impression
Berger-Levrault - Nancy

MAI 1987

NOTRE COUVERTURE : Le PAC 286 de Tandon a été l'une des nouveautés les plus marquantes du dernier Sicob. Ses disques durs (30 Mo, bientôt 100 Mo) sont amovibles, ce qui procure une très grande souplesse d'emploi lors des sauvegardes. Le prix est assez attractif puisque Tandon annonce 23 000 F pour une machine équipée de 2 ADD-PAC.

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

Ce mois-ci, nous pénétrons plus avant dans les recoins les plus intéressants du Pascal, à savoir l'allocation dynamique de mémoire. Cette formulation un peu barbare et rebutante cache un trésor de possibilités qui font une grande partie de la puissance de notre langage préféré.

COURS Nº 11

PLAN DU COURS

1. Notion d'allocation dynamique
   - Pourquoi une nouvelle, et plus complexe, structure ?
   - Concept de base
2. Pointeurs
   - Logique du stockage
3. Mise en pratique
   - Déclaration d'un pointeur
   - Allocation dynamique
   - Un exemple simple, qui montre déjà des possibilités intéressantes
   - Description des commandes liées aux pointeurs
   - Comment connaître le contenu d'une liste ?
4. Notion de listes et de files
   - Qu'est-ce qu'une file ?
5. Les arbres binaires
   - Pourquoi introduire cette notion maintenant ?
   - Une idée plus précise des arbres
   - Balayage d'un arbre
   - Utilité d'un arbre, première évidence
   - La structure pour gérer un arbre
   - Modification d'un arbre par programme
   - Lister le contenu d'un arbre
6. Conclusion
7. Exercices

Annexe

1. NOTION D'ALLOCATION DYNAMIQUE

1.1. Pourquoi une nouvelle, et plus complexe, structure ?

Nous avons présenté une série de structures fort importantes au cours des mois précédents. Parmi celles-ci, citons les tableaux, les ensembles, les enregistrements (ou Record). Ces structures constituent l'architecture permettant la fabrication de données complexes adaptées aux besoins du programme. Elles ont en commun une propriété primordiale : une fois créées, ces données sont figées, non sur le plan de la valeur, mais sur un plan, disons, géographique, ou d'encombrement. En effet, leur structure en mémoire n'est plus accessible. Ceci justifie le nom que nous leur donnerons maintenant, *variables statiques.* Il arrive souvent dans les programmes complexes que ce manque de liberté soit un vrai handicap pour la réalisation d'algorithmes. Pour pallier ce problème, il existe une structure, un type de données appelées *dynamiques* car elles sont accessibles et modifiables en cours de programmes. Il ne s'agit pas de faire n'importe quoi à tout moment mais, moyennant quelques précautions et avertissements (au programme), il est possible de faire de grandes choses. Précautions disions-nous : il est en effet évident qu'à un certain niveau, ces variables sont tout de même statiques, car de l'un des types déjà décrits. Nous allons voir comment comprendre, construire et utiliser ces structures étranges, venues d'aill..., venues de Suisse, bien sûr (S.V.P., ne pas confondre avec une publicité pour une boisson gazeuse que nous ne nommerons pas, naturellement).

1.2. Concept de base

L'idée est la suivante : peut-on créer une structure qui soit l'équivalent d'une procédure, donc d'un ensemble de commandes, et ayant la plupart de ses propriétés. Tout se passe bien tant que l'on utilise les propriétés les plus simples et au plus, un Record permet de s'en sortir. Mais, ne l'oublions pas, une procédure peut être récursive et là, il semble que cela se complique. Cela revient donc à avoir quelque part dans le programme une déclaration qui ressemble à :

```
var noeud : Record
        aa : integer;
        noeud_recur : toto;
        ploum_ploum : plouf_plouf;
    end;
```

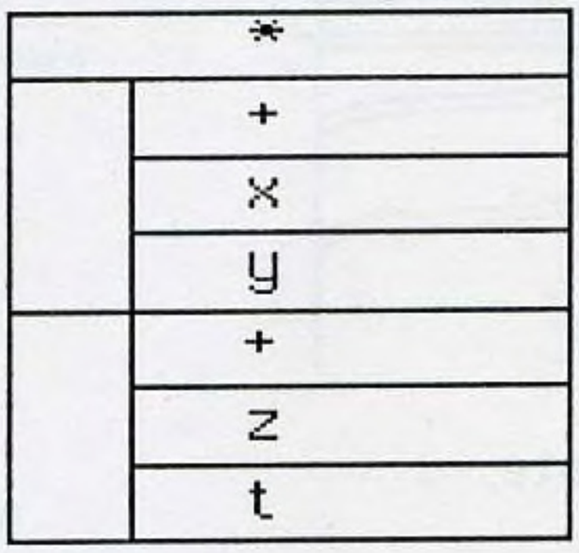
De plus, si dans le cas précédent la récursivité est directe, cela peut ne pas être le cas, et nous aurions alors quelque chose comme :

```
type noeud_1 = Record
        aa : integer;
        noeud_recur_1 : toto_2;
        ploum_ploum : plouf_plouf;
    end;

type noeud_2 = Record
        aa : integer;
        case t: boolean of
            true : noeud_recur_2 : toto_1;
            false : ploum_ploum : plouf_plouf;
        end;
    end;
```

Voilà une série de problèmes intéressants !!!
La notion la plus simple de récursivité est celle que l'on trouve en arithmétique. Par exemple : l'expression suivante l'utilise et en fait toute expression avec des parenthèses (qui peuvent ne pas être présentes mais seulement suggérées) fait de même :

$$(x + y) * (z + t)$$

Lorsque le programme qui doit lire cette expression rencontre la première parenthèse, il stocke le résultat de la première évaluation en attendant de pouvoir s'en servir avec le second. Une représentation de ce travail peut être vue de la manière suivante :



ou encore :

$$*(+(X,Y),+(z,t)$$

Si nous parvenons à ce type de notions, les déclarations ci-dessus sont possibles.
Un dernier exemple, pour montrer que tout ceci ne sort pas uniquement du cerveau d'un maniaque dangereux. Supposons que vous fassiez un programme, très simple où vous stockerez votre généalogie. Il sera possible de décrire une personne par un enregistrement comme celui-ci :

```
var un_tel : Record
             nom : string;
             prénom : string;
             date_naissance : date;
             mort :boolean;
             date_décès : date;
             case ancetres_connus : boolean of
               true :(mère , père : un_tel);
               false:();
             end;
```

Une telle définition est compacte, claire et simple, tout au moins à lire. Or, c'est là le plus important.
Dans le dernier cas proposé, l'analogie avec une procédure est encore plus frappante puisque nous avons incorporé des conditions. Notons qu'une telle condition est indispensable, sinon la structure ne se termine pas et les résultats sont à la mesure des espérances que vous pouvez avoir. Essayez, pour voir !!!

Les premiers exemples donnés sont dans ce cas, ils ne marchent pas !!!

Il s'agissait seulement d'être clair et simple.

2. POINTEURS

2.1. Introduction

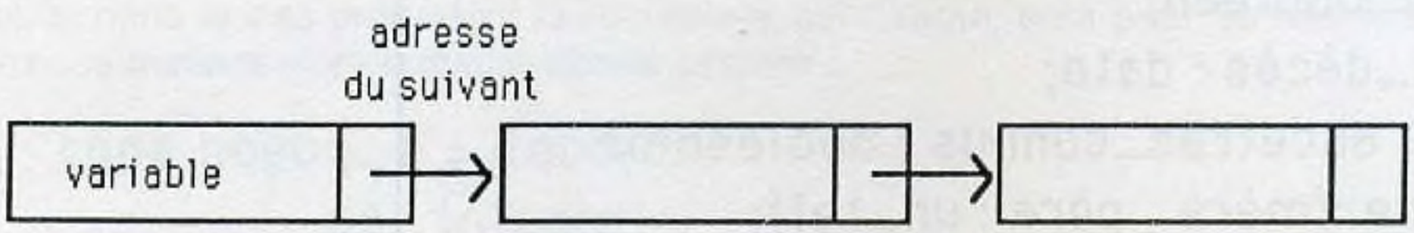
Nous avons donc vu que nous pouvons imaginer des structures récursives, dont la caractéristique principale est leur taille variable, ou plutôt qui peut varier. Il n'est donc pas possible de leur assigner une place en mémoire lors de la compilation, comme cela est fait pour les variables statiques. Il faut une allocation de mémoire qui soit effectuée au moment où l'on se sert de ces variables, ou **allocations dynamiques**. Le seul travail réalisé par le compilateur est de garder une place, non pour la variable, nous l'avons dit, mais pour l'adresse mémoire de cette variable, et c'est tout.

2.2. Exemple

Reprenons la structure généalogique déjà vue :

```
var un_tel : Record
             nom : string;
             prénom : string;
             date_naissance : date;
             mort :boolean;
             date_décès : date;
             case ancetres_connus : boolean of
               true :(mère , père : un_tel);
               false:();
             end;
```

Nous pouvons donner une interprétation graphique du lien existant entre les niveaux. On voit donc que le lien est réalisé physiquement par des flèches, partant d'une case où se trouve stockée l'adresse du bout de la flèche. Une liste d'éléments chaînés par allocation dynamique est en général représentée par un schéma du type suivant, pour expliquer la structure :

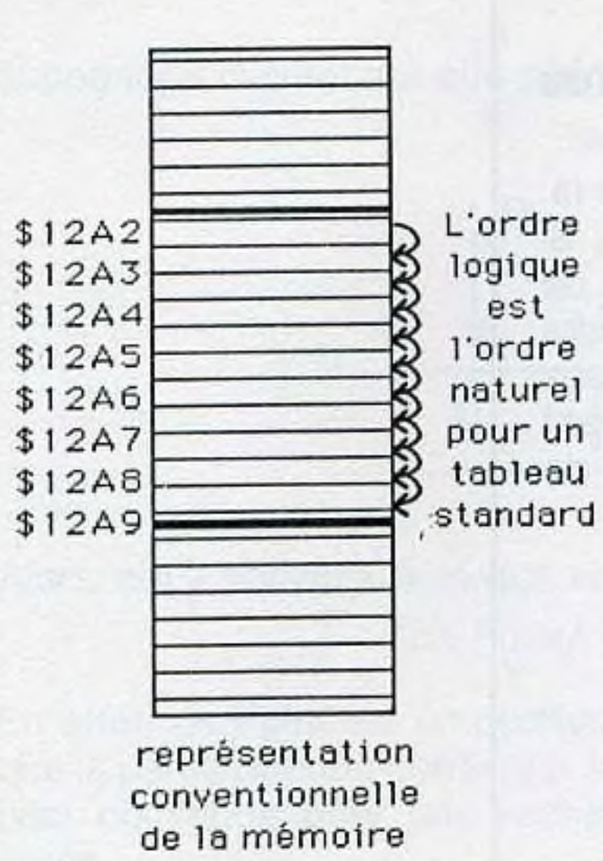


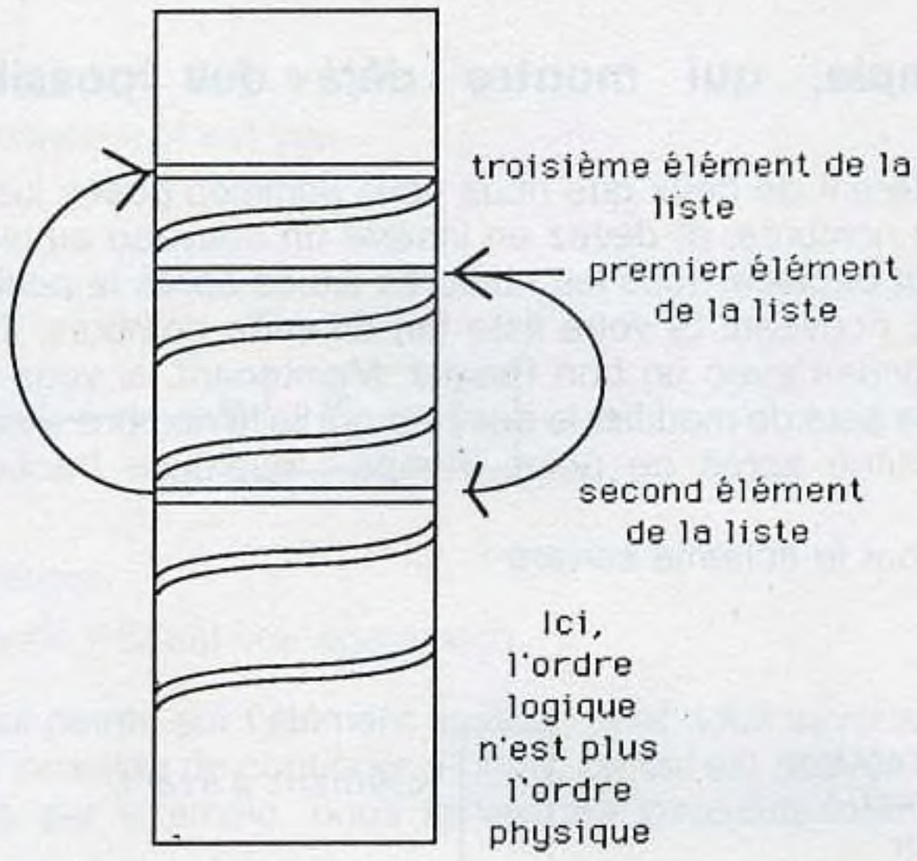
2.3. Définition

Ces flèches en machine sont appelées POINTEURS, ou POINTERS en anglais. Elles pointent en effet sur la case suivante.
Notons qu'il ne s'agit que d'une justification de leur existence, car dans ce type de structure, le programmeur ne voit pas, dans son programme, de différence.

2.4. Logique du stockage

Avec une allocation dynamique, il ne peut plus être question de penser à un tableau comme à une liste d'éléments rangés les uns derrière les autres. D'ailleurs, tout ce que vous utilisez comme information est que l'élément n + 1 est stocké (d'un point de vue logique) après le $n^{ème}$. Il en va de même ici. Nous ne nous occupons plus de l'ordre réel de rangement, mais de son ordre logique qui est celui mémorisé par le système en cours de programme.





3. MISE EN PRATIQUE

3.1. Déclaration d'un pointeur

En Pascal, la déclaration d'un pointeur se fait comme toute déclaration, mais il faut ajouter un petit symbole Λ pour préciser le pointeur. Par exemple :

```
var
   P = ^integer;
```

Nous avons donc P, qui est un pointeur vers un entier. P contiendra l'adresse où se trouvera cet entier, si on l'utilise.
Plus loin nous verrons que l'objet pointé par un pointeur est appelé PΛ. Ici PΛ est l'entier pointé par P. C'est peut-être un peu sujet à confusion, mais en fait l'habitude vient très vite.

3.2. Allocation dynamique

Pour utiliser effectivement une structure dynamique, il faut utiliser la séquence type suivante où nous déclarons un pointeur vers un enregistrement :

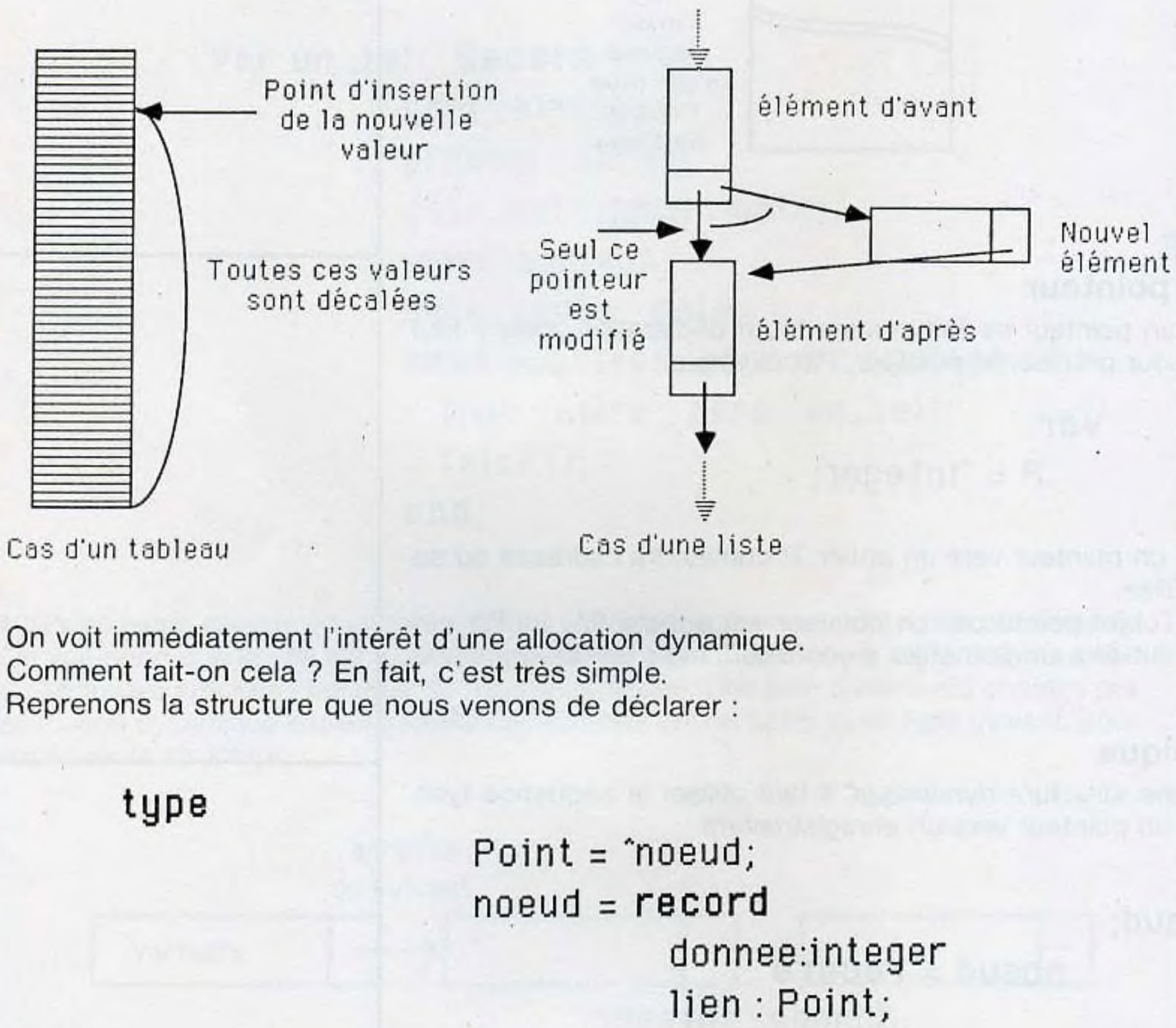
```
type
   Point = ^noeud;
                  noeud = record
                          donnee : integer;
                          lien : Point;
                          end;
var
   p,q : Point;
```

Rappelons que, dans cette déclaration, p et q sont des pointeurs, des flèches, vers des enregistrements de type toto.

Une remarque, tout de suite, sur un point qui semble fort choquant. **Contrairement à tout ce que nous avons écrit sur la philosophie du Pascal, Point est déclaré comme un pointeur vers nœud, mais nœud n'est pas encore déclaré à ce moment.** Disons-le tout net, nous ne sommes pas fous, et il n'y a pas d'erreur. Il s'agit là du seul cas où il est possible de le faire. Pourquoi ? Nous y reviendrons implicitement plus loin.

3.3. Un exemple simple, qui montre déjà des possibilités intéressantes

Voilà le problème, un peu différent de ceux que nous nous sommes posés jusque là. Vous disposez d'une liste de nombres, et devez en insérer un nouveau au milieu. Si vous utilisez un tableau, il faut déplacer tous les nombres situés après la position de celui à insérer, puis placer le nouveau. Si votre liste fait dix-mille nombres, il faudra approximativement trois secondes avec un bon Pascal. Maintenant, si vous utilisez une liste, la seule chose à faire sera de modifier le pointeur qui lie le nombre situé avant le point d'insertion à celui situé après ce point. Temps : quelques fractions de secondes.
Nous pouvons résumer cela par le schéma suivant :



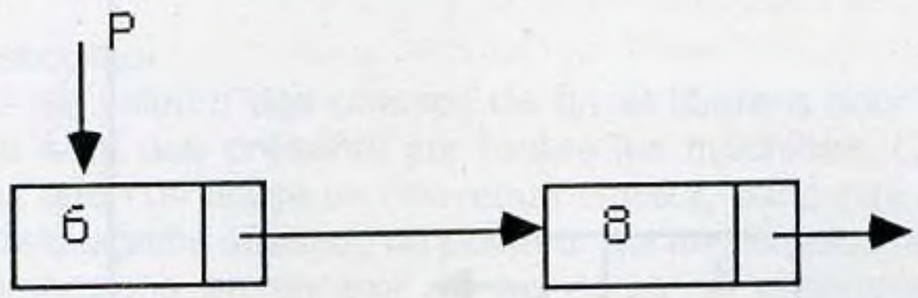
On voit immédiatement l'intérêt d'une allocation dynamique.
Comment fait-on cela ? En fait, c'est très simple.
Reprenons la structure que nous venons de déclarer :

```
type
        Point = ^noeud;
        noeud = record
                    donnee:integer
                    lien : Point;
                end;

var
    p,q : Point;
```

Supposons que cette structure contienne dans sa partie donnée (donc dans la partie au format entier) des nombres, ordonnés, et en particulier les nombres 6 et 8, sans qu'il y ait de 7.

Supposons maintenant que p pointe sur le 6. Cela signifie :

(pΛ.donnée = 6) est vrai



Alors, nous sommes sûrs que nous avons :

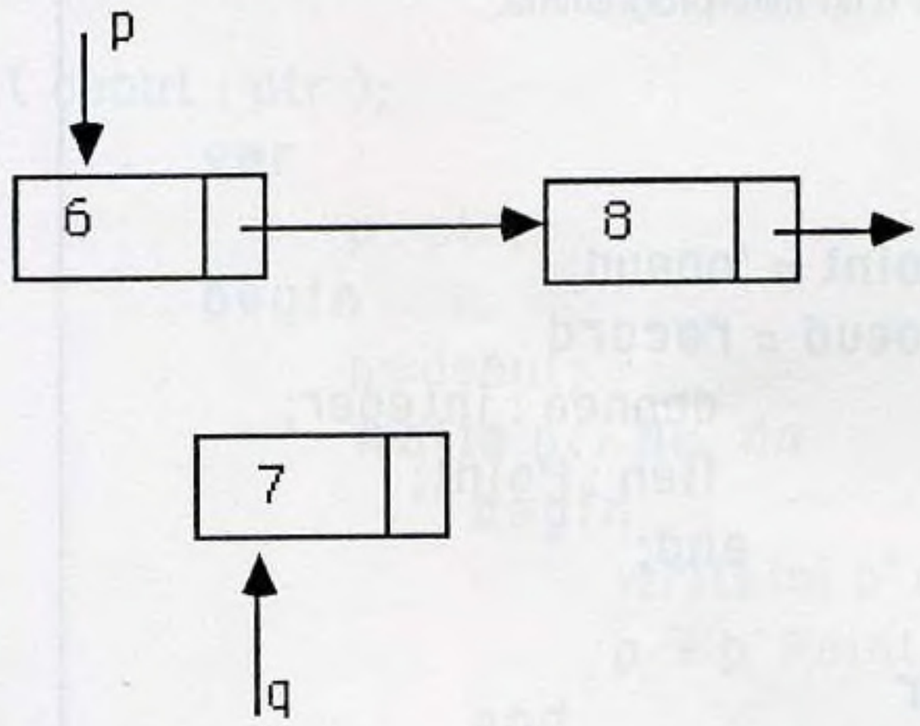
(pΛ.PointΛ.donnée = 8) est vrai également.

En effet, pΛ.Point est un pointeur, qui pointe sur l'élément suivant, dont nous savons que la partie donnée contient 8. Il est possible de continuer à l'infini, ce qui est souvent bien commode pour une recherche par exemple, nous le verrons immédiatement après.

Maintenant, il reste à insérer le 7. Pour cela, il faut créer un nouveau pointeur, puisqu'un nouvel élément est inséré. Comment le faire ? Par la commande NEW.

Ici, ce sera :

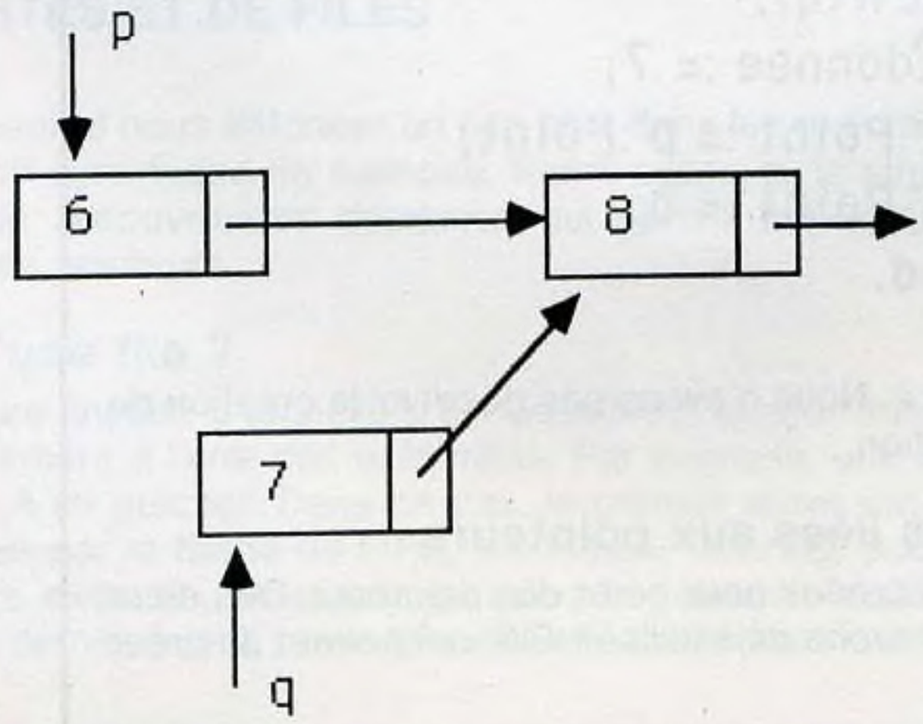
```
NEW(q);
qΛdonnée := 7;
```



Rappelons que q est déclaré dans la structure. Sinon, il faut le rajouter, puisque tout doit être déclaré en Pascal.

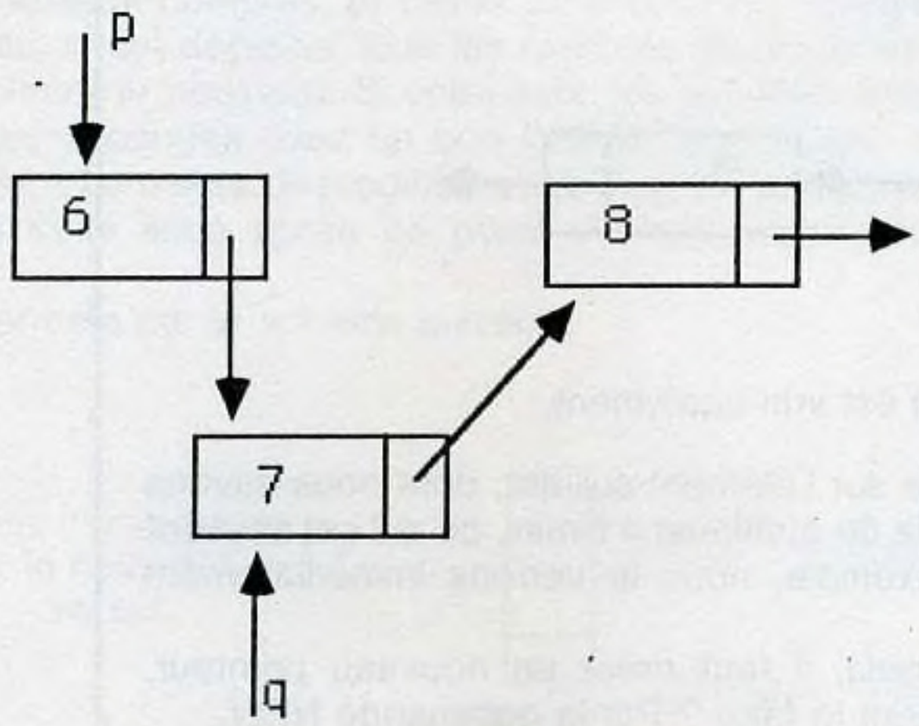
Maintenant, c'est très simple. On récupère le lien entre 6 et 8 par :

qΛ.Point := pΛ.Point;



Enfin, il reste à effectuer le dernier lien :

pΛ.Point := q;



C'est fini !

Reprenons toutes les étapes, sous forme d'un mini-programme.

```
Program insert;
     type
                    Point = ^noeud;
                    noeud = record
                              donnee : integer;
                              lien : Point;
                            end;

                  var
                    p,q : Point;

                  begin
                    NEW(q);
                    q^.donnee := 7;
                    q^.Point := p^.Point;
                    p^.Point := q;
                  end.
```

C'est simple, comme nous l'avions annoncé. Nous n'avons pas décrit ici la création de la liste elle-même, seulement sa modification.

3.4. Description des commandes liées aux pointeurs

Il existe deux commandes spécifiquement créées pour gérer des pointeurs. Ces deux commandes sont NEW et DISPOSE. Nous avons déjà utilisé NEW, qui permet de créer

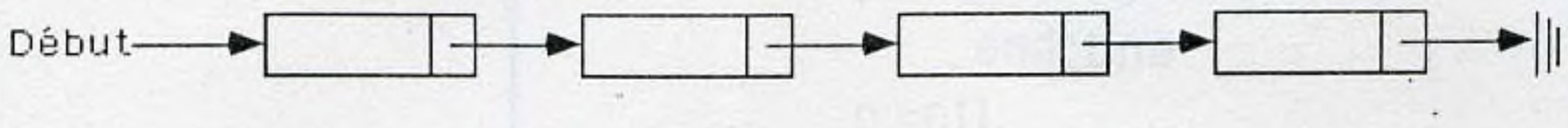
physiquement le pointeur, avant de s'en servir. Il est possible de faire le contraire pour libérer la mémoire d'un pointeur qui ne sera plus utilisé et la commande DISPOSE est là pour ça :

Dispose(q)

efface de la mémoire les valeurs des champs de qΛ et libère q pour usage ultérieur. NEW et DISPOSE ne sont pas présents sur toutes les machines. Cela signifie que NEW est implicite dès que l'on utilise un nouveau pointeur, donc pas de problème de ce côté. Par contre, cela signifie aussi qu'un pointeur qui ne sert plus reste en mémoire malgré tout. Nous proposons en annexe de ce cours un programme qui teste si DISPOSE est présent.

3.5. Comment connaître le contenu d'une liste ?

Considérons une liste chaînée comme nous en avons vues quelques-unes :



Début est un pointeur, vers le premier élément de la liste. Le dernier élément de cette liste possède un pointeur dont la valeur est NIL, valeur signifiant qu'il ne pointe vers rien. La procédure suivante réalise la sortie de toutes les valeurs d'une liste dont les éléments sont de type toto, comme décrit précédemment.

```
Procedure ecrit ( debut : ptr );
                    var
                        p : ptr;
                    begin
                        p:=debut;
                        while p<> NIL do
                            begin
                                writeln( p^.donnee );
                                p := p^.Point;
                            end;
                    end.
```

La manière de travailler est évidente, donc nous ne reviendrons pas sur ce programme.

4. NOTION DE LISTES ET DE FILES

4.1. Introduction

Nous allons dans ce chapitre nous enfoncer un peu plus dans les notions importantes qui vont avec l'allocation dynamique de mémoire. Parmi celles-ci, la structure de file est primordiale, car elle recouvre des domaines qui seront traités avec plus de simplicité en utilisant des pointeurs.

4.2. Qu'est-ce qu'une file ?

Une file est une structure linéaire à laquelle il est possible d'ajouter des éléments et d'en enlever, mais seulement à l'une des extrémités. Par exemple, une file peut être vue comme une queue à un guichet. Dans ce cas, le premier arrivé est le premier à repartir (ce qui se traduit par le terme de FIFO, hautement français, puisqu'il signifie First In, First Out). Autre exemple, lorsque vous lavez la vaisselle, vous empilez les assiettes et reprenez la dernière de la pile la fois suivante (la file est alors FILO, First In. Last Out).

Nous sommes désolés, mais la défense de la langue française ne peut aller jusqu'à donner des termes autres que ceux utilisés par les spécialistes français eux-mêmes.
Comment créer, gérer et utiliser de telles listes linéaires.
Pour créer une liste, il suffit de créer un pointeur vers une structure, qui pointe vers une autre, etc. C'est ce que nous avons déjà fait. Reprenons une dernière fois la structure toto :

```
type
        Point = ^noeud;
        noeud = record
                    donnee : integer;
                    lien : Point;
                end;

        var
          p,q : Point;
```

Cette structure simple va nous permettre bien des exemples pour illustrer notre propos. Tout d'abord, ajouter un élément à notre liste.
Premier cas, on le rajoute en fin de liste :

```
Procedure ajout_fin;
begin
   new(q);                    (* creation du nouveau pointeur *)
   readln(q^.donnee);         (* valeur du champ donnee *)
   q^.Point:=nil;             (* on ajoute en fin de liste *)
   dernier^.Point:=q;         (* l'ancien dernier pointe sur le nouveau dernier *)
end;
```

Dernier représente le dernier élément d'une liste. Il faut éventuellement le chercher. Cela sera aisément par une boucle gérée par WHILE, comme nous l'avons déjà fait plusieurs fois.
Maintenant, un ajout en tête de liste. Ceci a déjà été traité par la procédure INSERT pour une insertion en milieu de liste. Ici, nous voulons le faire en tête de liste :

```
procedure ajout_tete;
                    begin
                        NEW(q);
                        readln(q.donnee);
                        q^.Point := tete^.Point;
                        tete^.Point := q;
                    end;
```

Tete désigne le premier pointeur de la liste.
Un retrait en fin de liste :

```
procedure retrait_fin;
                    begin
                        q:=tete;
                        while q <> nil do
                            begin
                                p:=q;
                                q:=q^.Point;
                            end;
                        p:=nil;
                        dispose(q);
                    end;
```

Un retrait en tête de liste :

```
procedure retrait_tete;
                        begin
                            tete:=tete^.Point;
                        end;
```

Simple, non ? En fait, tete est maintenant à la valeur qu'avait le second pointeur de liste, avant la modification.
L'insertion en milieu de liste a déjà été vue. Un retrait en milieu de liste est similaire.

4.3. Conclusion

Vous savez maintenant gérer toute liste linéaire. A vous de jouer, les possibilités sont grandes et vous pouvez reprendre des programmes déjà faits pour augmenter vitesse et optimisation de la place mémoire.

5. LES ARBRES BINAIRES

5.1. Introduction

Encore une notion barbare ! C'est fini, je ne veux plus voir cette revue débilitante où l'on ne parle que de trucs bizarres et compliqués.
Nostra culpa.
Nous avouons. Plus de tortures. Assez !
Mais restez encore un peu avec nous, on se sentira moins seuls.
Surtout que ce qui vient est carrément passionnant. Si, si.
Bon, en voilà assez pour l'introduction et la publicité (ou le désespoir si vous partez).

5.2. C'est quoi ça ???

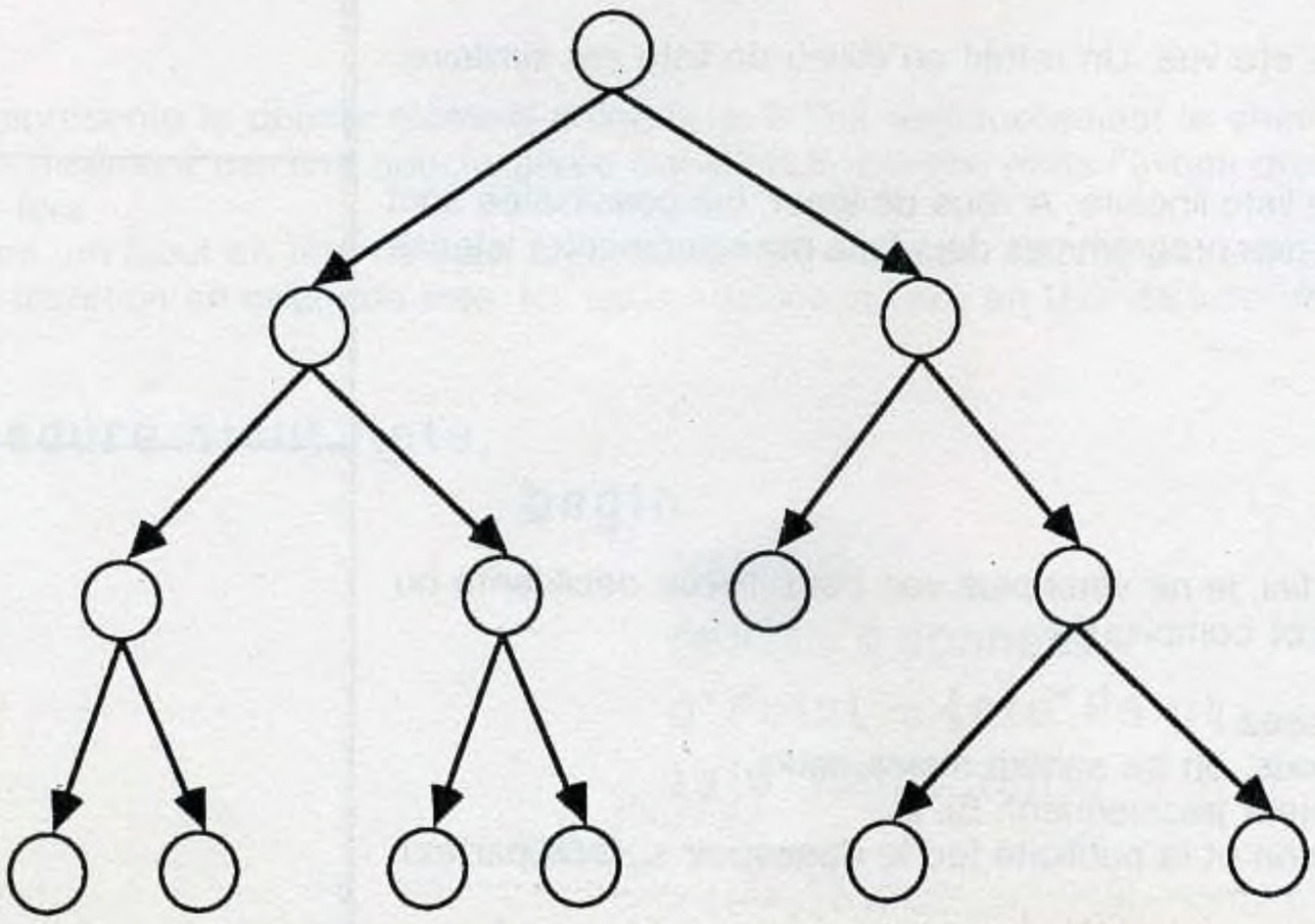
Nous pouvons déjà vous dire que ce n'est pas cela,
Mais alors, pas du tout.

Ce n'est pas cela non plus.

Alors, c'est quoi, dites ?

Cela ressemble un peu à cela,
mais en moins écologique.

Allons, il est temps d'être plus précis.
Un arbre binaire est une liste bidimensionelle. Au lieu d'une liste linéaire, nous nous attaquons aux listes planes, dont voici un exemple :

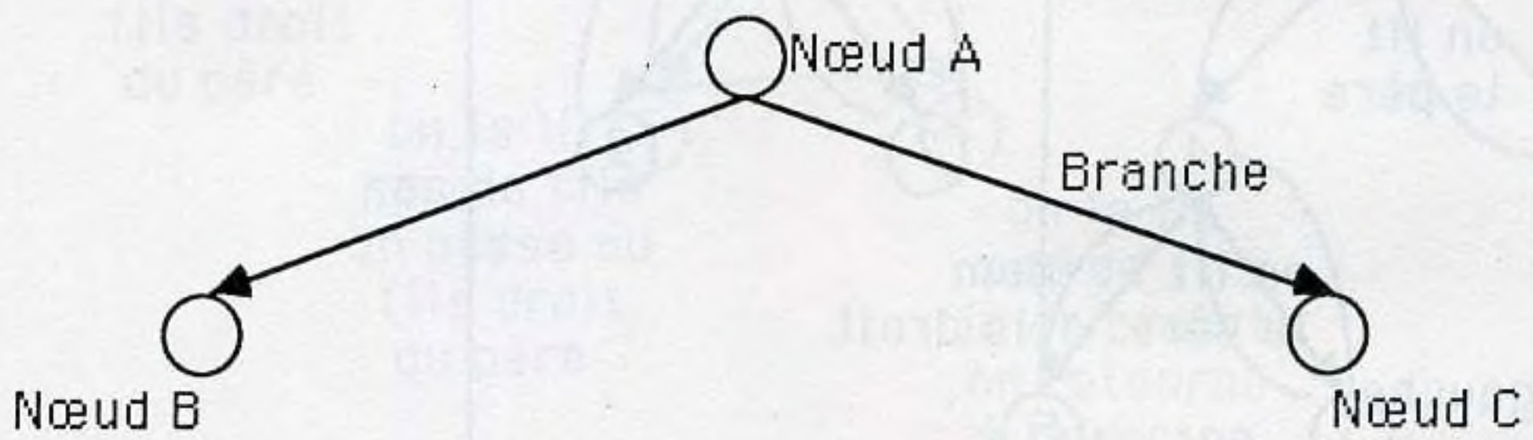


5.3. Pourquoi introduire cette notion maintenant ?

Il s'avère que des arbres sont très facilement gérés par récursivité. Il est facile et efficace d'écrire des procédures récursives pour travailler avec des arbres. De plus, ces procédures sont en général de très bonnes démonstrations de l'utilité et de la puissance de la récursivité. Donc, les avantages se cumulent et il n'est pas imaginable de ne pas parler d'arbre binaire en programmation en Pascal.
Parmi les programmes qui font un usage absolu des arbres, le tri est certainement le plus représentatif puisque faire un tri par arbre est une méthode vraiment puissante et simple, nous le verrons. Un tri par arbre permet de gagner des minutes dans un programme, ce qui est vraiment un avantage primordial.

5.4. Une idée plus précise des arbres

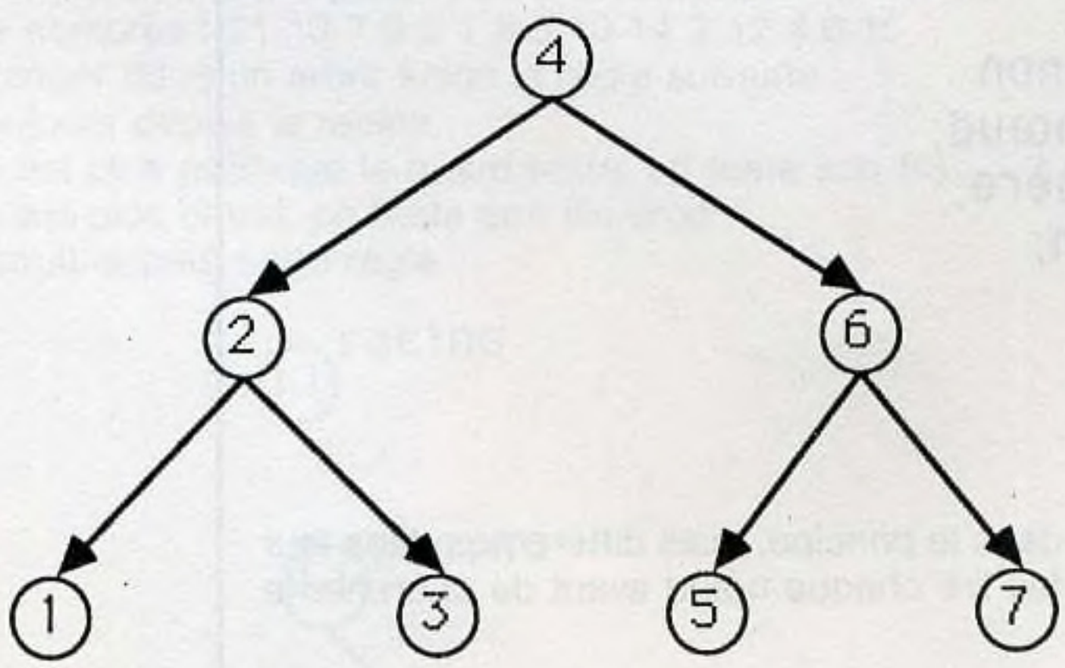
Un arbre est formé de nœuds et de branches, comme un arbre réel, d'où son nom. Les nœuds sont les points de réunion des branches et seront pour nous l'équivalent des éléments valués d'une liste. Les branches sont les pointeurs d'un nœud à un autre.



Sur ce dessin, A est la racine et B, C et D sont les descendants de A. A est le seul nœud qui ne possède pas de parent. Chacun des autres a exactement un unique parent.
Enfin, un arbre est binaire si chaque parent a, au plus, deux descendants.

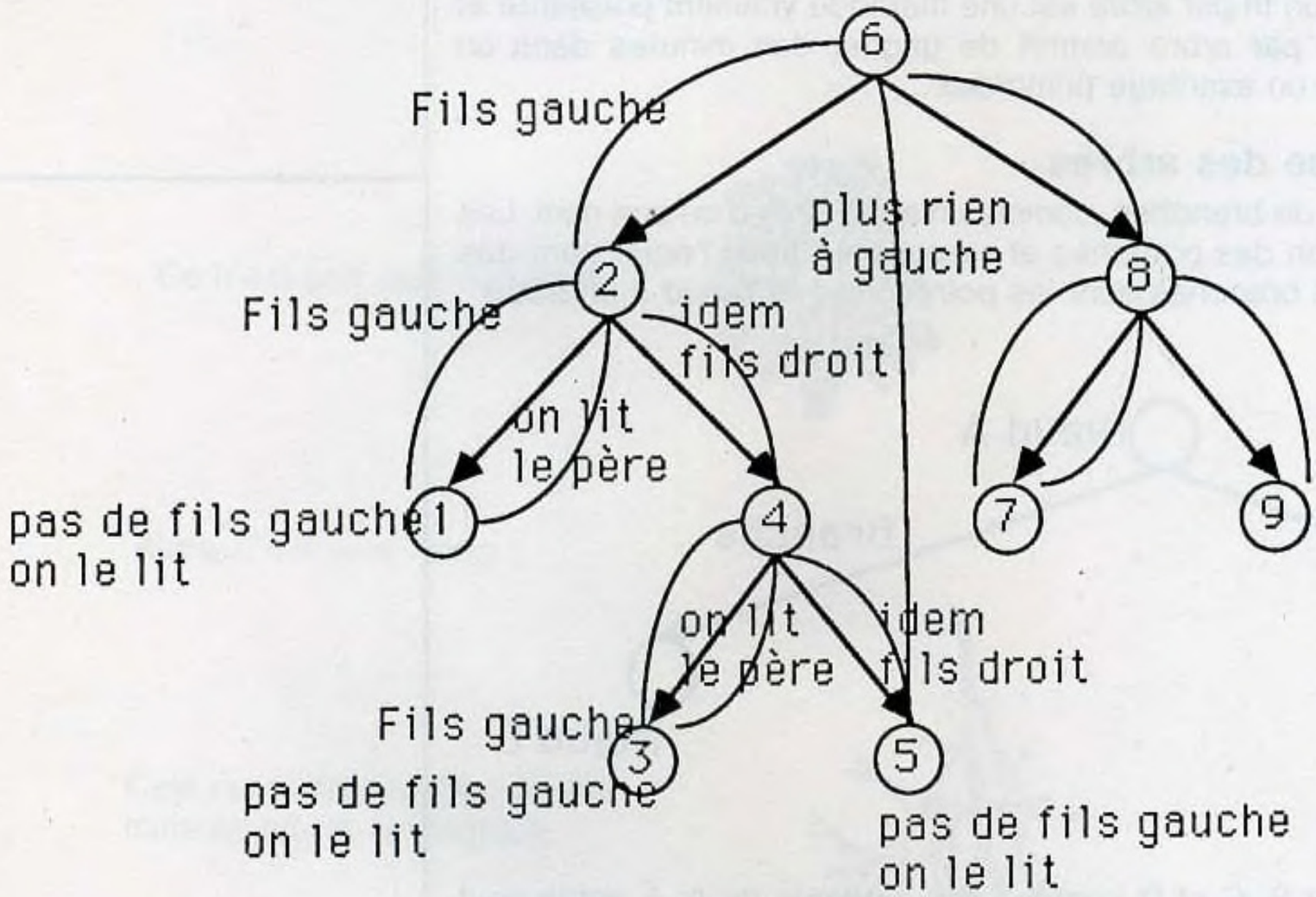
5.5. Balayage d'un arbre

Ne partez pas !!!
Balayer un arbre signifie simplement lire toutes les données qui sont liées à l'arbre, nœud par nœud. Et pour cela, il faut se donner un sens de balayage.
Plusieurs méthodes existent, dont les principales sont illustrées ci-dessous :
Tout d'abord, le balayage en profondeur, Gauche Racine Droit :



GRD (Gauche-Racine-Droit)

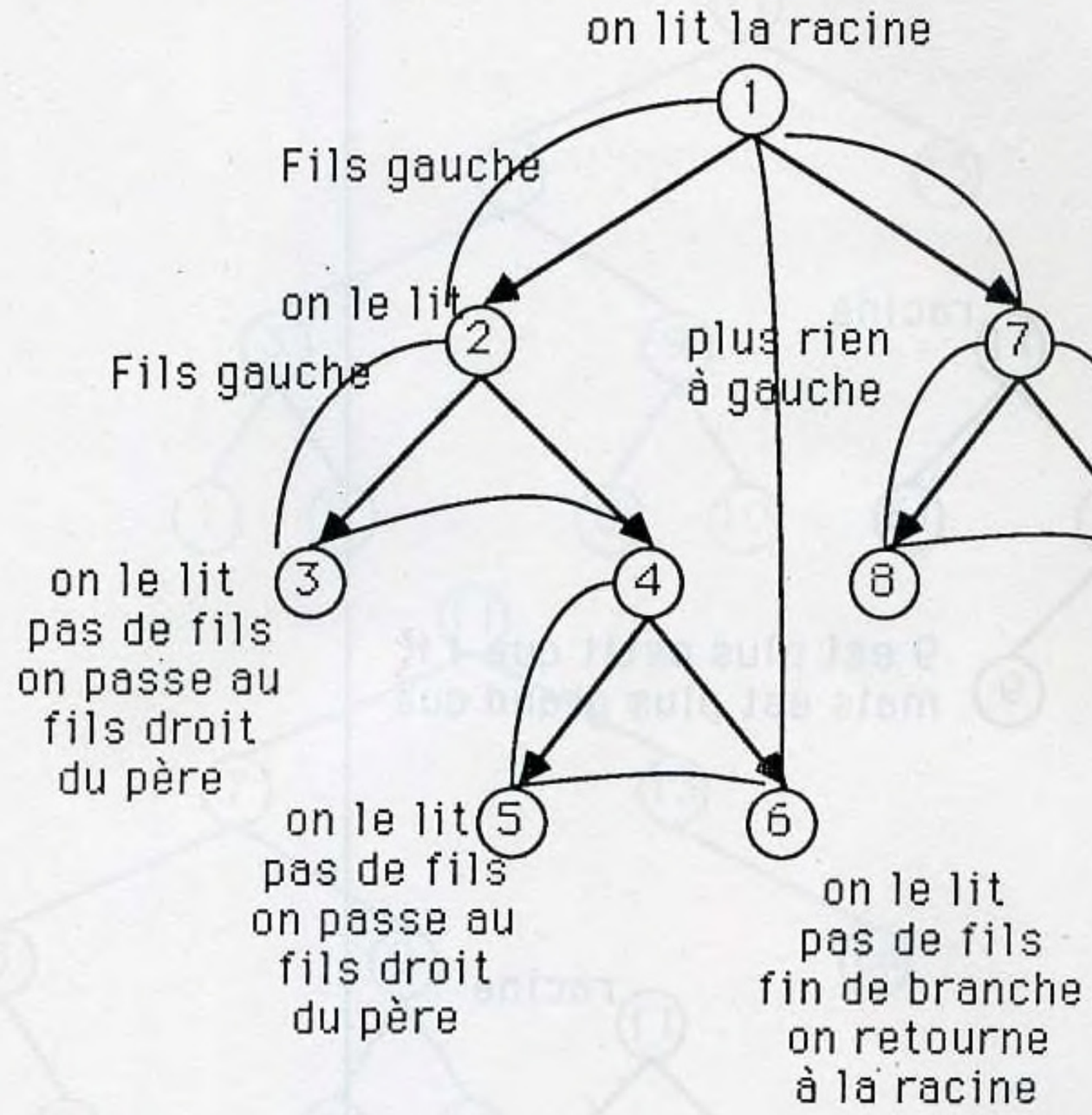
L'ordre des nœuds est celui de la lecture des valeurs. On cherche le fils gauche de la racine puis celui du nœud trouvé... tant qu'il y en a un. Puis, lorsqu'il n'y en a plus, on lit la valeur du nœud, celle de son ascendant et on recommence la recherche sur le fils droit, en cherchant son fils gauche... Voici le détail sous forme graphique :



On voit bien ici la récursivité. Il est possible de la formaliser par une routine du type :

```
procedure lit_GRD(nœud);
 debut
  si fils_gauche
   alors lit_GRD(fils_gauche)
   sinon
    debut_sinon
     lire le nœud;
     lire le père;
    fin_sinon;
fin;
```

Les autres méthodes sont assez proches dans le principe, mais différentes dans leur application. Il existe la méthode RGD, qui fait lire chaque nœud avant de chercher le fils gauche :



On le voit, la différence consiste à lire le nœud où l'on se trouve avant de passer à son éventuel fils gauche et non le contraire comme avant.
Nous ne multiplierons pas les exemples à l'infini mais sachez que toute méthode de lecture est valide. Simplement, la plus utilisée est GRD, la première que nous avons montrée. Celle-ci peut se résumer à la séquence suivante :

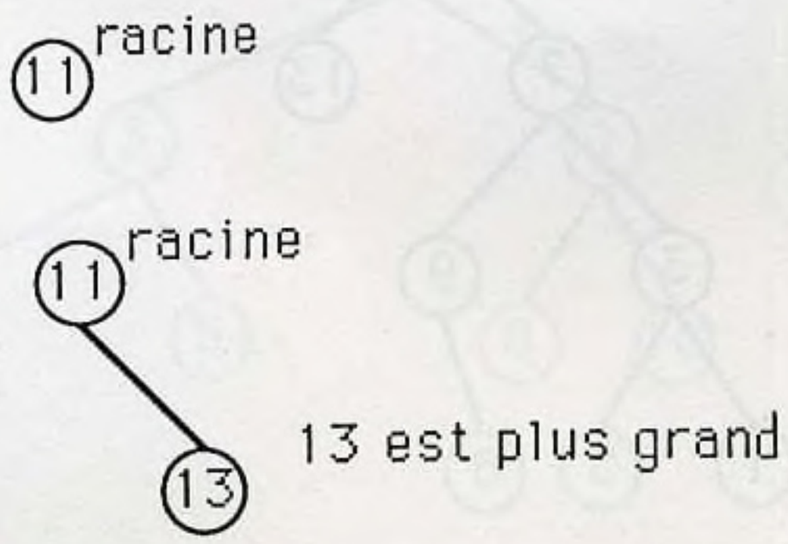
```
à un nœud, vous lisez le sous arbre gauche,
vous lisez le nœud lui-même, puis,
vous lisez le sous-arbre droit.
```

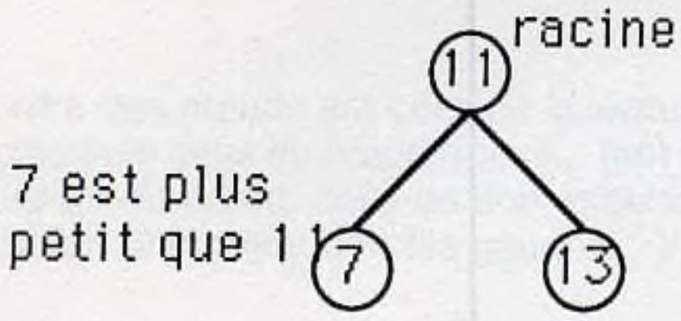
5.6. Utilité d'un arbre, première évidence

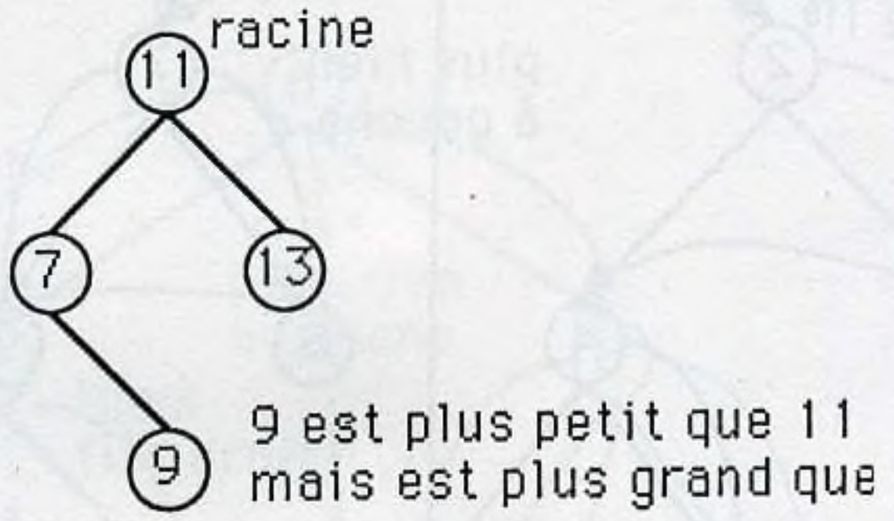
Nous allons construire un arbre à partir d'un critère simple.
Voici une liste de nombres : 11 13 7 9 3 1 8 5 10 14 2 12 4 6 15
Nous allons les ranger dans un arbre selon la règle suivante :

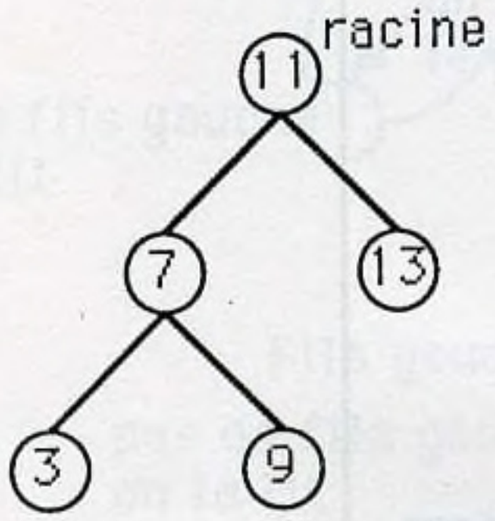
on balaye toujours depuis la racine,
si le nombre est plus petit que le nœud testé, on teste son fils
si le nombre est plus grand, on teste son fils droit.

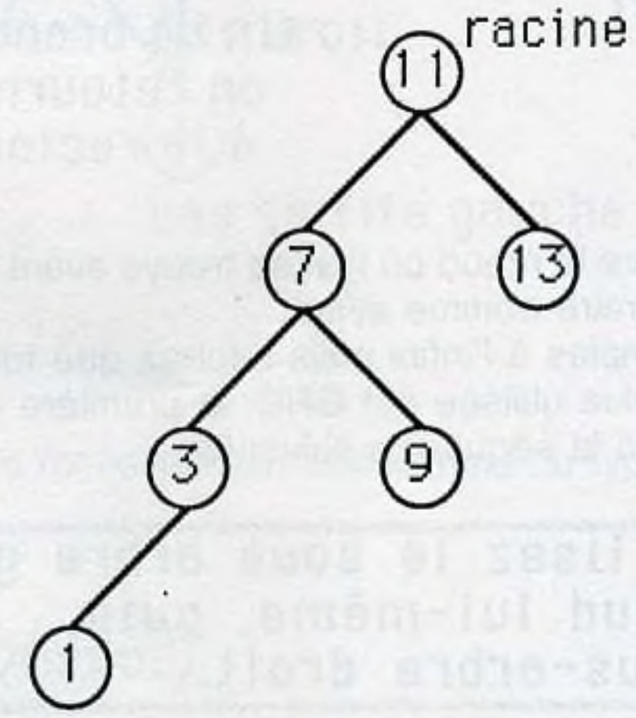
Voici l'arbre construit depuis cette règle :

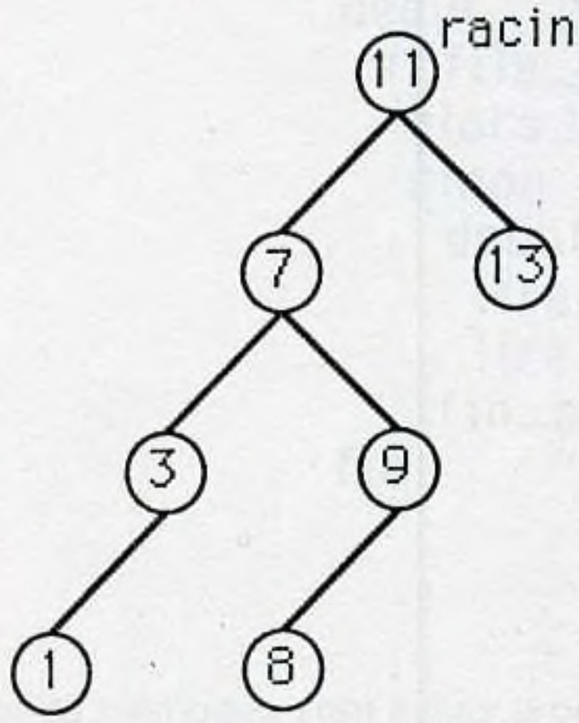


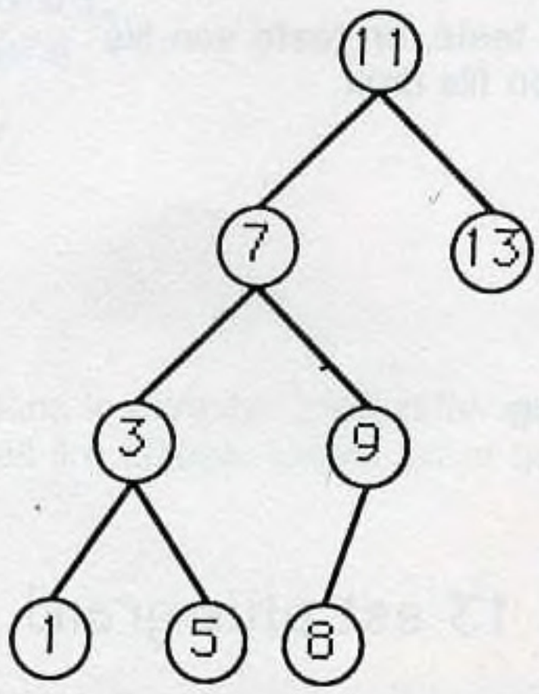

racine
11
7 est plus
petit que 11
7
13

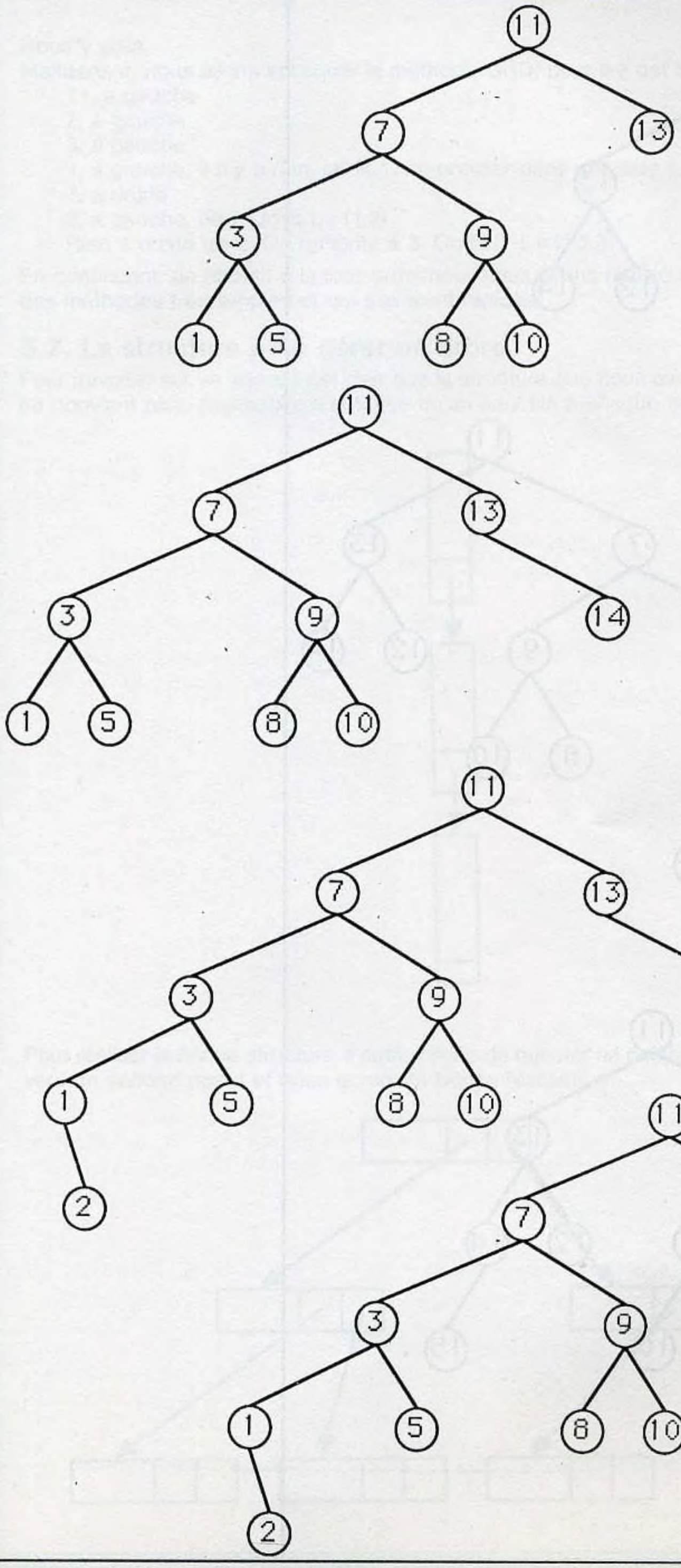

racine
11
7
13
9
9 est plus petit que 11
mais est plus grand que


racine
11
7
13
3
9

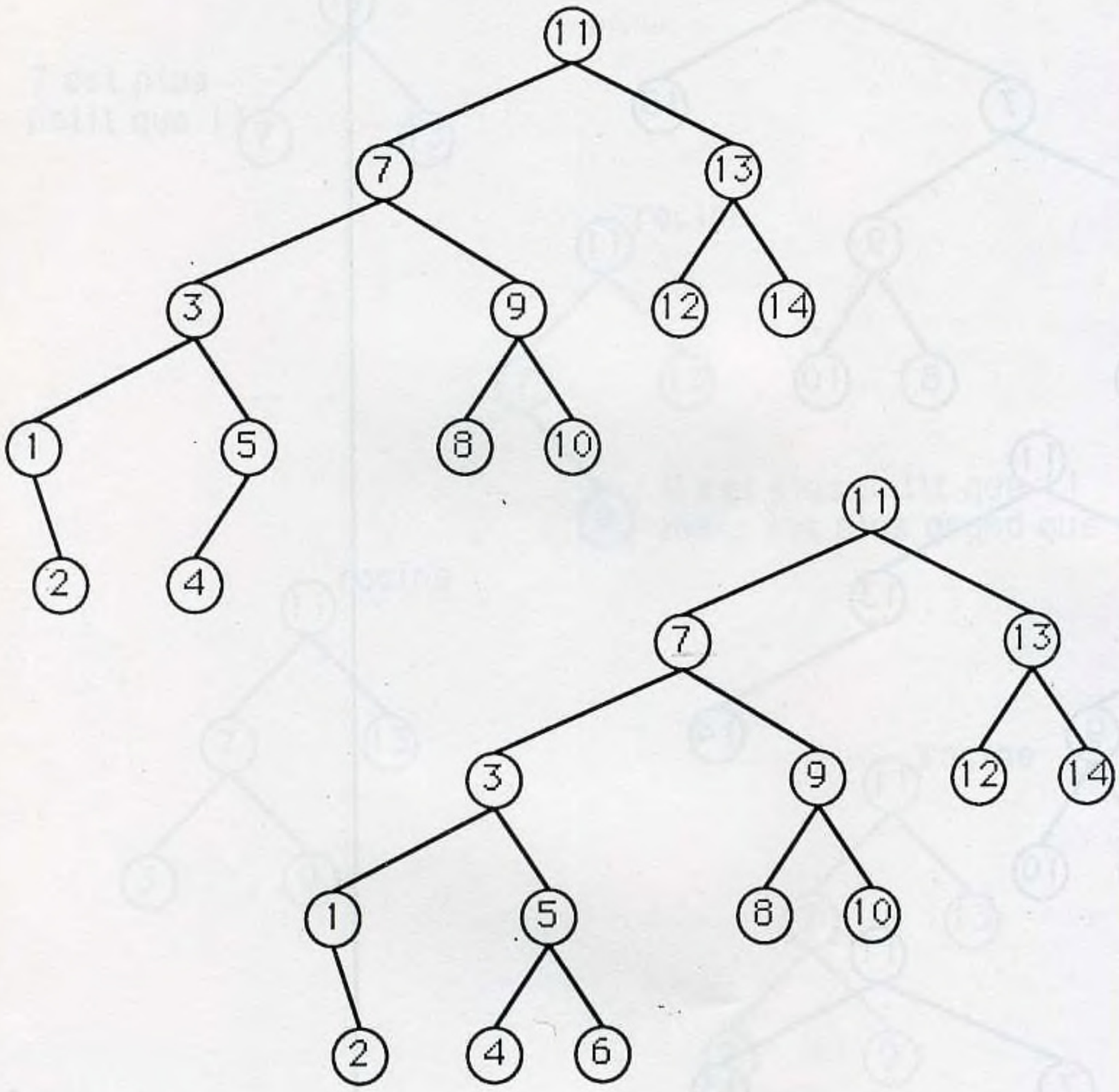

racine
11
7
13
3
9
1


racine
11
7
13
3
9
1
8

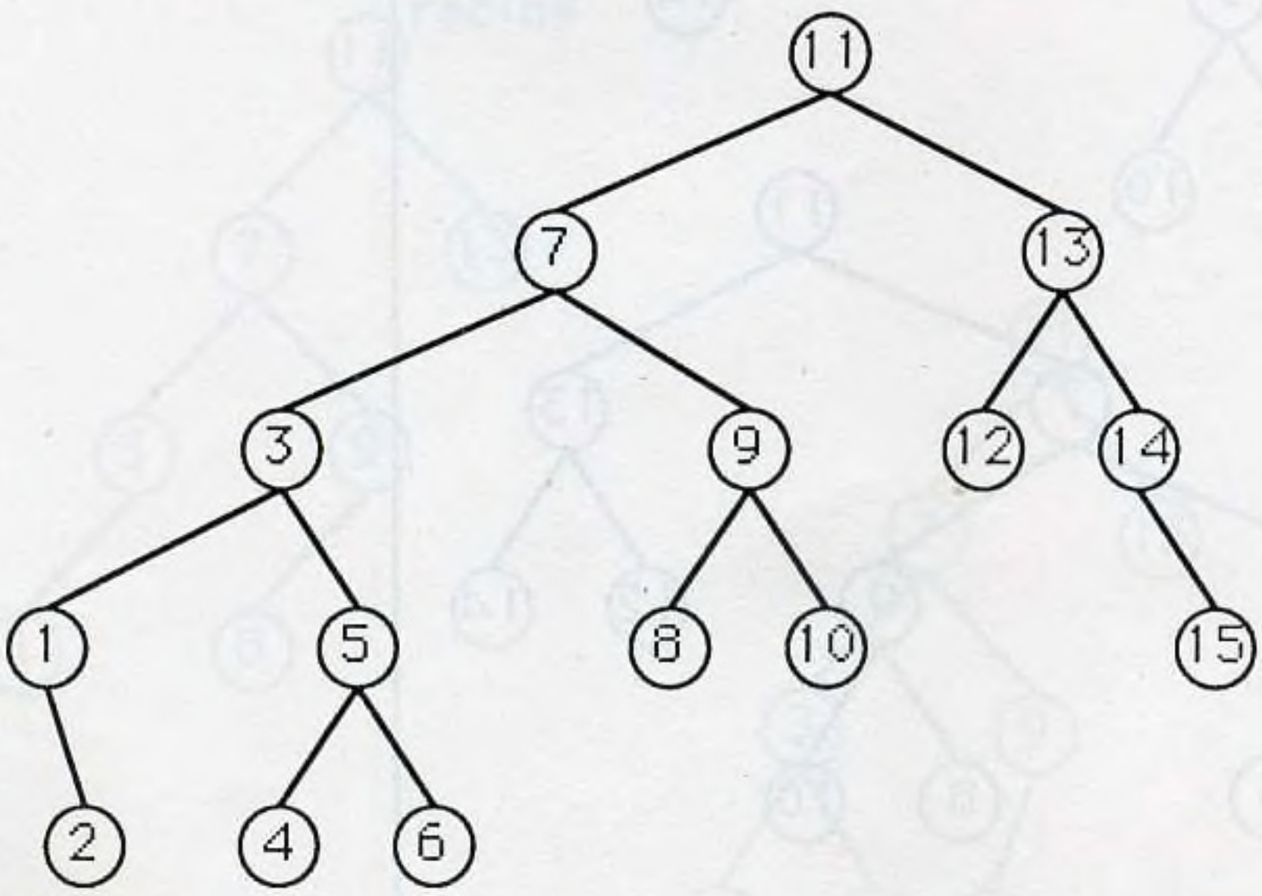

11
7
13
3
9
1
5
8


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

NOTES PERSONNELLES



Encore un minuscule effort.



Nous y voilà.
Maintenant, nous allons appliquer la méthode GRD, pour lire cet arbre :

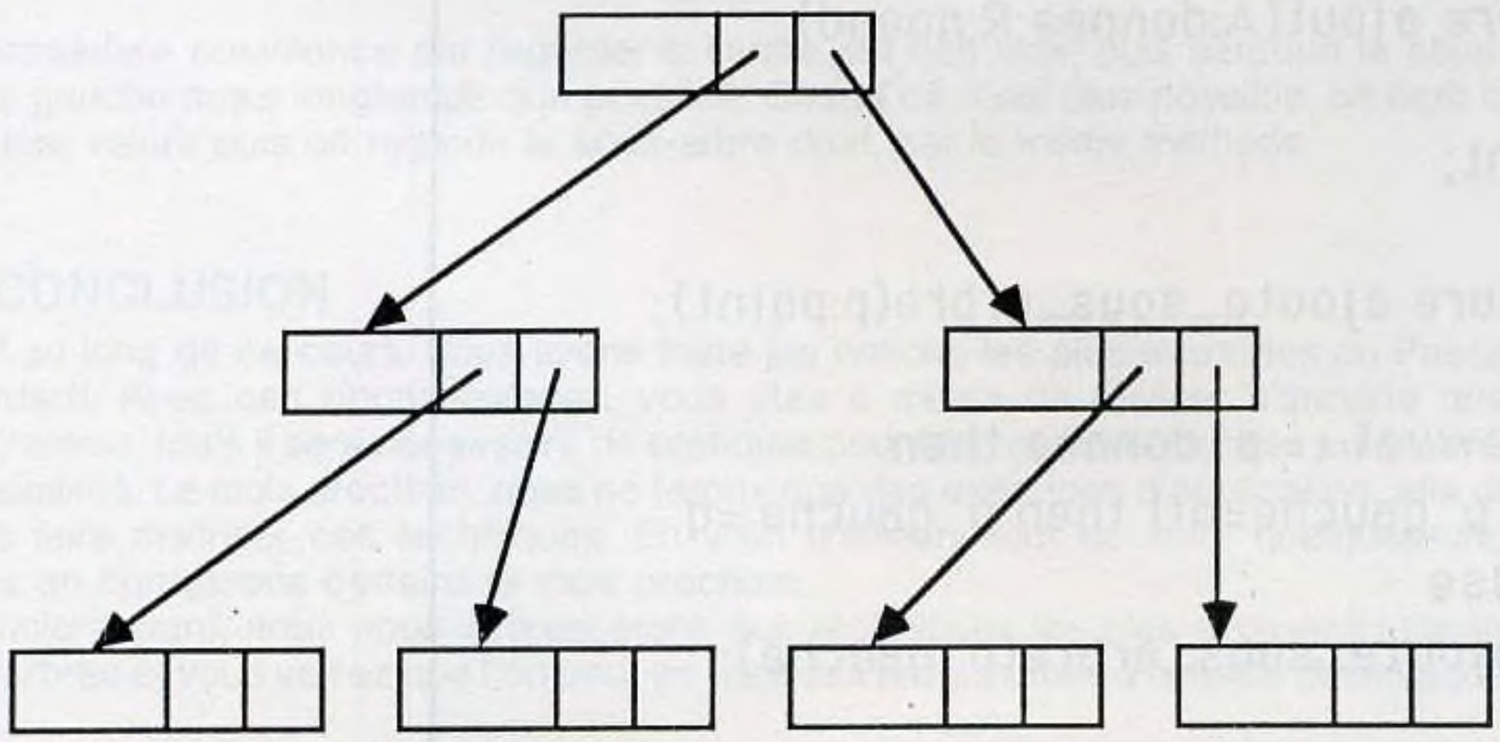
11, à gauche
7, à gauche
3, à gauche
1, à gauche, il n'y a rien, on lit 1, en premier dans une liste L=(1)
1, à droite
2, à gauche, rien. On lit L=(1,2)
Rien à droite de 2. On remonte à 3. On lit 3. L=(1,2,3)

En continuant, on aboutit à la liste ordonnée. Nous avons réalisé un tri par arbre, par des méthodes très simples et qui s'avèrent rapides.

5.7. La structure pour gérer un arbre

Pour travailler sur un arbre, il est clair que la structure que nous avions précédemment ne convient plus, puisqu'elle n'autorise qu'un seul fils à chaque nœud :

Pour réaliser la bonne structure, il suffira donc de rajouter un pointeur supplémentaire vers un second nœud et nous aurons la bonne réalisation.



Pour cela, nous utiliserons la structure (très proche de la précédente) :

```
type
    Point = ^noeud;
                    noeud = record
                        donnee : integer;
                        lien1,lien2 : Point;
                    end;
```

Comme annoncé, la modification est très limitée. Nous avons maintenant de quoi gérer un arbre binaire. De plus, si l'on veut que l'arbre soit ternaire ou plus, il suffit de rajouter des pointeurs.

5.8. Modification d'un arbre par programme

Comment faire pour ajouter un élément à un arbre ? C'est simple si l'on se réfère à l'exemple du tri. Le principe d'action pour insérer une donnée A est le suivant :
Soit R la racine de l'arbre.
Procédure AJOUT(A,R)

```
Si A<=R alors
 Si R a un fils_gauche alors
  AJOUT(A,fils_gauche(R))
 sinon
  A est le fils_gauche de R
sinon
 Si R a un fils_droit alors
  AJOUT(A,fils_droit(R))
 sinon
  A est le fils_droit de R
```

Sous forme d'un programme, cela donne un code plus long, car il faut créer les pointeurs nécessaires :

```
Procédure ajout(A:donnee;R:noeud);
 var
  q:point;

 procedure ajoute_sous_arbre(p:point);
  begin .
   if newval <= p^.donnee then
     if p^.gauche=nil then p^.gauche:=q
      else
       ajoute_sous_arbre(p^.gauche);
```

```
    else
     if p^.droit=nil then p^.droit:=q
     else
      ajoute_sous_arbre(p^.droit);
end;

   begin
    new(q);
    q^.gauche:=nil;
    q^.droit:=nil;
    q^.donnee:=A;
    if R=nil then R:=q
     else ajoute_sous_arbre(R);
end;
```

5.9. Lister le contenu d'un arbre

Comme pour insérer une nouvelle valeur, le listage des valeurs d'un arbre est simple et rapide. La procédure que voilà est très proche de la précédente.

```
Procédure liste(p:point);
 begin
  if p<>nil then
   begin
    liste(p^.gauche);
    writeln(p^.donnee);
    liste(p^.droit);
   end;
end;
```

La procédure commence par regarder si l'arbre est non vide, puis parcourt le sous-arbre gauche aussi longtemps que possible. Quand ce n'est plus possible, on écrit la dernière valeur puis on regarde le sous-arbre droit, par la même méthode.

6. CONCLUSION

Tout au long de ce cours, nous avons traité les notions les plus avancées du Pascal standard. Avec ces connaissances, vous êtes à même de réaliser n'importe quel programme. Mais il sera nécessaire de pratiquer pour bien comprendre ses puissantes possibilités. Le mois prochain, nous ne ferons que des exercices d'application, afin de vous faire maîtriser ces techniques. En voici d'ailleurs tout de suite quelques-uns. Nous en corrigerons certains le mois prochain.
Le mois suivant, nous nous intéresserons aux techniques les plus évoluées utilisant des arbres et vous verrez que l'on peut en faire des programmes à la limite du magique.

7. EXERCICES

7.1. Faire le programme complet pour le tri par arbre et faire tourner pour la liste proposée en cours.

7.2. Modifier ce programme pour le faire travailler sur des noms propres.

7.3. Modifier le programme du 7.1. pour lui faire lister l'arbre selon l'ordre DRG (droit-racine-gauche). Etudier le résultat obtenu.

7.4. Faire un programme de gestion d'un carnet d'adresses. Le plus complet sera le mieux.

7.5. Ecrire un programme qui efface un nœud d'un arbre. Attention, ce n'est plus aussi simple en raison du nombre de cas particuliers. Comme application, reprendre le 7.4. et en faire un vrai utilitaire, souple et efficace.

7.6. Comparer les vitesses des programmes de tri déjà vus en cours et celui développé cette fois-ci.

ANNEXE

Avez-vous DISPOSE ? Pour le savoir, essayez ce programme :

```
Program essai_dispose;
              type
                   ptr = ^Point;
                   Point = record
                        donnee : array [1..100] of integer;
                        lien : ptr;
                    end;

              var
                   q : ptr;
                   i ; integer;

         begin
              for i:=1 to 100000 do
                   begin
                        new(q);
                        q^.donnee[87]:=1;
                        dispose(q);
                   end;
              writeln('ok');
              end.
```

Si ce programme plante, c'est que DISPOSE ne fonctionne pas. Dans ce cas, il faut le créer. Pour ce faire, il faut faire attention à ne pas détruire n'importe quoi et une pile pourra être utilisée pour gérer les pointeurs inutilisés.

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chastagnier
Jean-François Coblentz
Patrick Gueneau

Dernier volet du descriptif des procédures et fonctions de l'application éditeur de texte. Il ne nous restait qu'à préciser trois routines complémentaires mais néanmoins indispensables au bon fonctionnement de l'éditeur. Nous avons reporté au mois suivant la vérification de la cohérence «inter-module» afin de réserver l'ensemble de l'article à cette tâche plus ardue qu'elle n'y paraît à première vue.

COURS N° 20

PLAN DU COURS

5. Routines d'effacements
   - Destruction d'un caractère
   - Destruction d'une ligne
6. Routine d'insertion

Conclusion

5. ROUTINES D'EFFACEMENTS

Nous n'envisagerons que deux types d'effacements :

- la suppression d'un caractère,
- la suppression d'une ligne.

Parmi les extensions possibles, il serait notamment intéressant d'ajouter des fonctions comme l'effacement du curseur jusqu'en fin de ligne, ou jusqu'en début de ligne, la suppression par bloc (c'est-à-dire par mot), etc.

Destruction d'un caractère

Il n'y a guère de difficulté, il suffit de couper le caractère sous le curseur (ou, au choix, à gauche du curseur) et de recoller les deux parties gauche et droite de la ligne. Attention cependant aux trois cas particuliers en début et fin de ligne, et surtout si l'on est après le dernier caractère on ne peut rien supprimer. Enfin, si la ligne ne contient plus de caractère, on la supprime, en appelant la routine de destruction de ligne.

```
début_procédure
si (x_curseur > ll) alors
    début_si
    ERREUR ;
                sinon
    l := prem_ligne + y_curseur – 1 ; (l est la ligne du curseur)
    ll := long(buffer(l)) ;                (ll est la longueur de la ligne courante)
                        (test des 3 exceptions)

si(x_curseur = 1) et (ll > 1) alors
    début_si
    buffer(l) := sous_chaîne(buffer(l), 2, ll – 1) ;
                sinon

    si(x_curseur = ll) alors
    début_si
    buffer(l) := sous_chaîne(buffer(l), 1, ll – 1) ; (si l l = 1 la chaîne est vide)
                sinon
    buffer(l) := sous_chaîne(buffer(l), 1, x_curseur – 1)
            + sous_chaîne(buffer(l), x_curseur + 1, ll – x_curseur) ;
    fin_si

fin_si
si(long(buffer(l)) = 0  alors
    début_si
        DETRUIT_LIGNE ; (on supprime la ligne de l'écran et du buffer)
                sinon
                (il ne reste plus qu'à réécrire la ligne modifiée)

    position_écran(1, y_curseur)
    écrit(buffer( l)) ;
    fin_si
fin_procédure
```

Destruction d'une ligne

Il y a un peu plus de travail à effectuer, puisqu'il faut réordonner le buffer de texte, mettre à jour quelques variables globales et raffraîchir toute la page.

```
début_procédure
si(nb_occupé = 0)  alors
    début_si
                                (le tableau est vide)
        ERREUR ;
                sinon
        l := prem_ligne + y_curseur – 1 ; (ligne du buffer concernée)

            (on décale l'ensemble du texte au niveau du tableau)
            (à partir de la ligne où se trouve le curseur)

        pour i allant de l à (nb_occupé – 1)
        début_pour
            buffer(i) := buffer(i + 1) ; (on décale d'un cran)
            fin_pour
        buffer(nb_occupé) := '' ; (la chaîne vide)
        nb_occupé := nb_occupé – 1 ;
                (on raffraîchit l'écran seulement pour la partie à réécrire)

        pour i allant de y_curseur à nb_ligne
            début_pour
                position_écran(1, i) ;
                écrit(buffer(i + prem_ligne – 1) ;
            fin_pour
        fin_si
fin_procédure
```

6. ROUTINE D'INSERTION

Elle correspond à la frappe du retour chariot (< CR > ou < RETURN >) qui provoque la coupure de la ligne en cours et la formation d'une nouvelle ligne. Nous avons choisi de couper la ligne à gauche du curseur afin de rester en accord avec les traitements similaires (par exemple pour la destruction de caractères).

```
début_procédure
si(nb_occupe = max_buffer)  alors
    début_si

                        (on est en dépassement de capacité)

        ERREUR ;
                                sinon
        l := prem_ligne + y_curseur – 1 ;
        ll := long(buffer(l) ;
        buffer(l) := sous_chaîne(buffer(l), 1, x_curseur – 1) ;
                                (on met à jour le tableau)
```

```
pour i allant_de l + 1 à nb_occupé
    début_pour
        buffer(i + 1) := buffer(i) ;
    fin_pour

                (on insère avant le curseur d'où la séparation)

  buffer(l + 1) := sous_chaîne(buffer(l), x_curseur, ll - x_curseur + 1) ;

nb_occupé := nb_occupé - 1 ;

                (on raffraîchit la seconde partie de l'écran)

pour i allant_de y_curseur à nb_ligne
    début_pour
        position_écran(1, i) ;
        écrit(buffer(i + prem_ligne - 1) ;
        fin_pour
    fin_pour
fin_procédure
```

CONCLUSION

A part l'écriture dans un vrai langage des routines décrites, il faut assembler tous les modules et vérifier la cohérence de l'ensemble. Il est en effet fréquent que l'étude morcelée d'un problème entraîne quelques incohérences dans la mise en place du projet complet. Avant de terminer notre étude, nous allons donc récapituler les principes et liens essentiels communs aux différents modules. Nous essaierons de fournir un listing de programme complet mais il risque d'être long et nous serons obligés de le réduire pour qu'il ne prenne pas la place de tout l'article (la loupe ne sera pas fournie).

DIALOGUE AVEC NOS LECTEURS

I. CHAMEAUX

M. Marsaly de Rochefort-sur-Mer nous a fait remarquer que notre programme de calcul de chameaux n'était pas valable si la consommation au kilomètre était supérieure à l'unité. Nous nous sommes alors rendu compte que nous avions omis de la faire varier lors de la constitution de notre jeu d'essais. Le calcul des aller-retour devenait donc faux. Il suffisait d'écrire :

Aller-retour := chargement DIV charge-max ;

Si l'on prend une valeur de 1 000 pour la charge maximum et de 500 pour le chargement, le nombre d'aller-retour est nul. Or, jusqu'à 1 000, il doit être de 1 et à partir de 1 001 de 2. On décide d'ajouter 1 systématiquement mais comme à 1 000 juste, il passe à 2, il faut enlever 1 à chargement. M. Marsaly l'avait remarqué mais sa solution faisait intervenir un test et la fonction MOD, opérations nettement plus coûteuses en temps.
Voici donc le programme définitif :

```
Program Chameau ;

Var
  Charge_maximale         ,
  Consommation            ,
  Distance_a_parcourir    ,
  Distance_deja_parcourue ,
  Chargement_total        ,
  Bananes_restantes       : INTEGER ;

Function Voyage(Chargement     ,
                Charge_max     ,
                Consommation   ,
                Dist_parcourue ,
                Distance_reste : INTEGER ) : INTEGER ;
```

```
Var

  Allers_retours       ,
  Consommat_kilometrique ,
  Chargement_restant   ,
  Kilometrage          : INTEGER ;

Begin
  Allers_retours := ( Chargement - 1 ) DIV Charge_max + 1 ;
  Consommat_kilometrique := ( 2 * Allers_retours - 1 ) * Consommation ;
  Chargement_restant := ( ( Allers_retours - 1 ) * Charge_max ) ;
  Kilometrage :=  ( Chargement - Chargement_restant ) DIV Consommat_kilometrique ;
  Dist_parcourue := Dist_parcourue + Kilometrage ;
  if Kilometrage >= Distance_reste then
      Voyage := ( Kilometrage - Distance_reste ) * Consommat_kilometrique
                + Chargement_restant
  else
      if Chargement_restant <= 0 then
        begin
          writeln (' Distance maximale possible ',Dist_parcourue) ;
          Voyage := 0 ;
        end
      else
        begin
          writeln (' Etape suivante au KM :',Dist_parcourue) ;
          Voyage := Voyage ( Chargement_restant , Charge_max ,
                             Consommation , Dist_parcourue ,
                             Distance_reste - Kilometrage ) ;
        end ;
End ;


  {------------------ Programme Principal ----------------------}

Begin
  writeln('donnez le nombre de bananes du chargement :') ;
  readln(Chargement_total) ;
  writeln('donnez le nombre de bananes consommees par kilometre :') ;
  readln(Consommation) ;
  writeln('donnez la charge maximale en bananes acceptee par le chameau :') ;
  readln(Charge_maximale) ;
  writeln('donnez la distance a parcourir :') ;
  readln(Distance_a_parcourir) ;
  Distance_deja_parcourue := 0 ;
  Bananes_restantes := Voyage( Chargement_total , Charge_maximale ,
                               Consommation , Distance_deja_parcourue ,
                               Distance_a_parcourir ) ;
  writeln (' il reste au bout du voyage ',Bananes_restantes,' bananes ');
End
```

II. LES CARRES MAGIQUES

M. Gueron semble réellement enthousiasmé par les carrés magiques ; il nous envoie une solution fournissant 3 600 carrés magiques d'ordre 5 (comme il exerce la noble profession de professeur de mathématiques, nous devons pouvoir lui faire confiance).

Ces carrés ont la propriété d'avoir toutes leurs diagonales égales à 65 lorsqu'on les prolonge.

2	10	23	11	19
13	16	4	7	25
9	22	15	18	1
20	3	6	24	12
21	14	17	5	8

2	10	23	11	19
13	16	4	7	25
9	22	15	18	1
20	3	6	24	12
21	14	17	5	8

2	10	23	11	19
13	16	4	7	25
9	22	15	18	1
20	3	6	24	12
21	14	17	5	8

Nous pouvons remarquer que ces carrés étaient ceux obtenus par la solution présentée en cours de Pascal.

```
Program Carres_Magiques ;

Type

  tab       = array [1..5] of INTEGER ;
  tab_long = array [1..10] of INTEGER ;
  carre     = array [1..5,1..5] of INTEGER ;

Var
  Nombre_de_Carres : INTEGER ;


  {------------------------------------------------------------}

Function Carre_Magique (      t , s     : tab      ;
                          VAR compteur : INTEGER ) : INTEGER ;


  Var
    Carres   : carre    ;
    t_long   ,
    s_long   : tab_long ;
    i , test : INTEGER  ;
```

```
Begin
  test := s[1] + t[1] ;
  if ( test < s[5] + t[5] ) then
    begin
      for i := 1 to 5 do
        begin
          t_long[i] := t[i] ;
          t_long[i + 5] := t[i] ;
          s_long[i] := s[i] ;
          s_long[i + 5] := s[i] ;
        end ;
      if (      ( test < s[4] + t[4] ) and ( test < s[3] + t[3] )
           and ( s[2] + t[2] < s[4] + t[3] )                      ) then
        begin
          for i := 1 to 5 do

          begin
            Carres[1,i] := s[i] + t[i] ;
            Carres[2,i] := s_long[i + 3] + t_long[i + 2] ;
            Carres[3,i] := s_long[i + 1] + t_long[i + 4] ;
            Carres[4,i] := s_long[i + 4] + t_long[i + 1] ;
            Carres[5,i] := s_long[i + 2] + t_long[i + 3] ;
          end ;
        compteur := compteur + 1 ;
      end ;
    if (      ( test < s[4] + t[3] ) and ( test < s[3] + t[2] )
         and ( s[2] + t[2] < s[3] + t[4] )                      ) then
      begin
        for i := 1 to 5 do
          begin
            Carres[1,i] := s[i] + t[i] ;
            Carres[2,i] := s_long[i + 2] + t_long[i + 3] ;
            Carres[3,i] := s_long[i + 4] + t_long[i + 1] ;
            Carres[4,i] := s_long[i + 1] + t_long[i + 4] ;
            Carres[5,i] := s_long[i + 3] + t_long[i + 2] ;
          end ;
        compteur := compteur + 1 ;
      end ;
  end ;
  Carre_magique := compteur ;
End ;

{-------------------------------------------------------------------------}

Function Petite_boucle (     t         : tab     ;
                         VAR compteur : INTEGER ) : INTEGER ;

  Var
    anc , s : tab ;
    i,j,k,l,m : INTEGER ;

  Begin
    compteur := 0 ;
    for i := 1 to 5 do
      begin
        s[i] := i ;          {   de 1 a 5      }
        anc[i] := s[i] ;
      end ;
    for i := 1 to 5 do
```

```
begin
  anc[1] := s[1] ;
  s[1] := s[i] ;
  s[i] := anc[1] ;
  for j := 2 to 5 do
    begin
      anc[2] := s[2] ;
      s[2] := s[j] ;
      s[j] := anc[2] ;
      for k := 3 to 5 do
    begin
      anc[3] := s[3] ;
      s[3] := s[k] ;
      s[k] := anc[3] ;
      compteur := Carre_Magique(t,s,compteur) ;
      anc[4] := s[4] ;
      s[4] := s[5] ;
      s[5] := anc[4] ;
      compteur := Carre_Magique(t,s,compteur) ;
      anc[4] := s[4] ;
      s[4] := s[5] ;
      s[5] := anc[4] ;
    end ;
  anc[3] := s[3] ;
  for l := 4 to 5 do s[l - 1] := s[l] ;
  s[5] := anc[3] ;
end ;
anc[2] := s[2] ;
for k := 3 to 5 do s[k - 1] := s[k] ;
s[5] := anc[2] ;
end ;
Petite_boucle := compteur ;
End ;

{-------------------------------------------------------------------}

Function Grande_boucle : INTEGER ;

  Var
    anc , s : tab ;
    compteur ,
    i,j,k,l,m : INTEGER ;

  Begin
    compteur := 0 ;
    for i := 1 to 5 do
      begin
        s[i] := 5 * ( i - 1 ) ;    {   de 0 a 20 , de 5 en 5       }
        anc[i] := s[i] ;

      end ;
   for i := 1 to 5 do
     begin
       anc[1] := s[1] ;
       s[1] := s[i] ;
       s[i] := anc[1] ;
       for j := 2 to 5 do
         begin
           anc[2] := s[2] ;
           s[2] := s[j] ;
           s[j] := anc[2] ;
           for k := 3 to 5 do
```

```
        begin
          anc[3] := s[3] ;
          s[3] := s[k] ;
          s[k] := anc[3] ;
          compteur := Petite_boucle(s,compteur) ;
          anc[4] := s[4] ;
          s[4] := s[l] ;
          s[l] := anc[4] ;
          compteur := Petite_boucle(s,compteur) ;
          anc[4] := s[4] ;
          s[4] := s[5] ;
          s[5] := anc[4] ;
          end ;
        anc[3] := s[3] ;
        for l := 4 to 5 do s[l - 1] := s[l] ;
        s[5] := anc[3] ;
      end ;
    anc[2] := s[2] ;
    for k := 3 to 5 do s[k - 1] := s[k] ;
    s[5] := anc[2] ;
  end ;
  Grande_boucle := compteur ;
End ;
{------------------ Programme Principal ----------------------}

Begin
  Nombre_de_Carres := Grande_boucle ;
End
.
```

Nous avons quelque peu modifié la structure du programme BASIC proposé. En effet, celui-ci effectue 5 boucles imbriquées en testant à chaque fois l'élément de la nouvelle boucle qui doit être différent de tous les précédents : ce qui provoque un nombre de tests assez conséquent. La boucle externe s'effectue 5 fois (ligne 110). La ligne 120 s'exécute elle aussi 5 fois, comprenant un test, passant à la ligne suivante 4 fois sur 5 de proche en proche, on en conclut pour les 5 boucles 110 à 150 :

$5\times(5\times1+4\times(5\times2+3\times(5\times3+2\times(5\times4))))$ tests, ce qui donne 3 525 tests pour 120 cas possibles.

Comme nous avons deux boucles de ce type qui sont imbriquées, nous obtenons :

$120\times3\,525$ tests $+$ 3 525 $=$ 426 525 tests pour 14 400 cas possibles.

boucle interne

Notre suggestion substitue les assignations (remplacement du contenu d'une variable par le contenu d'une autre variable) aux tests (ce qui, en temps de calcul, est déjà moins coûteux) et de plus en réduit le nombre. Dans une boucle, il y en a en effet :

$5\times(3+4\times(3+3\times(9)+4)+5)$, ce qui donne 720 assignations,

donc pour deux boucles 87 120 opérations, environ 5 fois moins.

Mais comme le programme de notre lecteur tourne déjà, il serait beaucoup trop coûteux de le refondre complètement, aussi nous contenterons-nous de l'améliorer par une propriété, à savoir : $A+B+C+D+E=50$ quelles que soient les valeurs prises, donc plutôt que de faire une boucle sur E, il suffirait de faire :

150 $E=50-A-B-C-D$

de même $X+4+2+T+U=15$ d'où

200 $U=15-X-4-2-T$

Le nombre de tests descend alors :

$5\times(5+4\times(10+3\times(15)))=1\,125$ tests

et pour deux boucles : 136 125, cela divise déjà par 3.

LA BIBLIOTHEQUE TECHNIQUE DES EDITIONS FREQUENCES

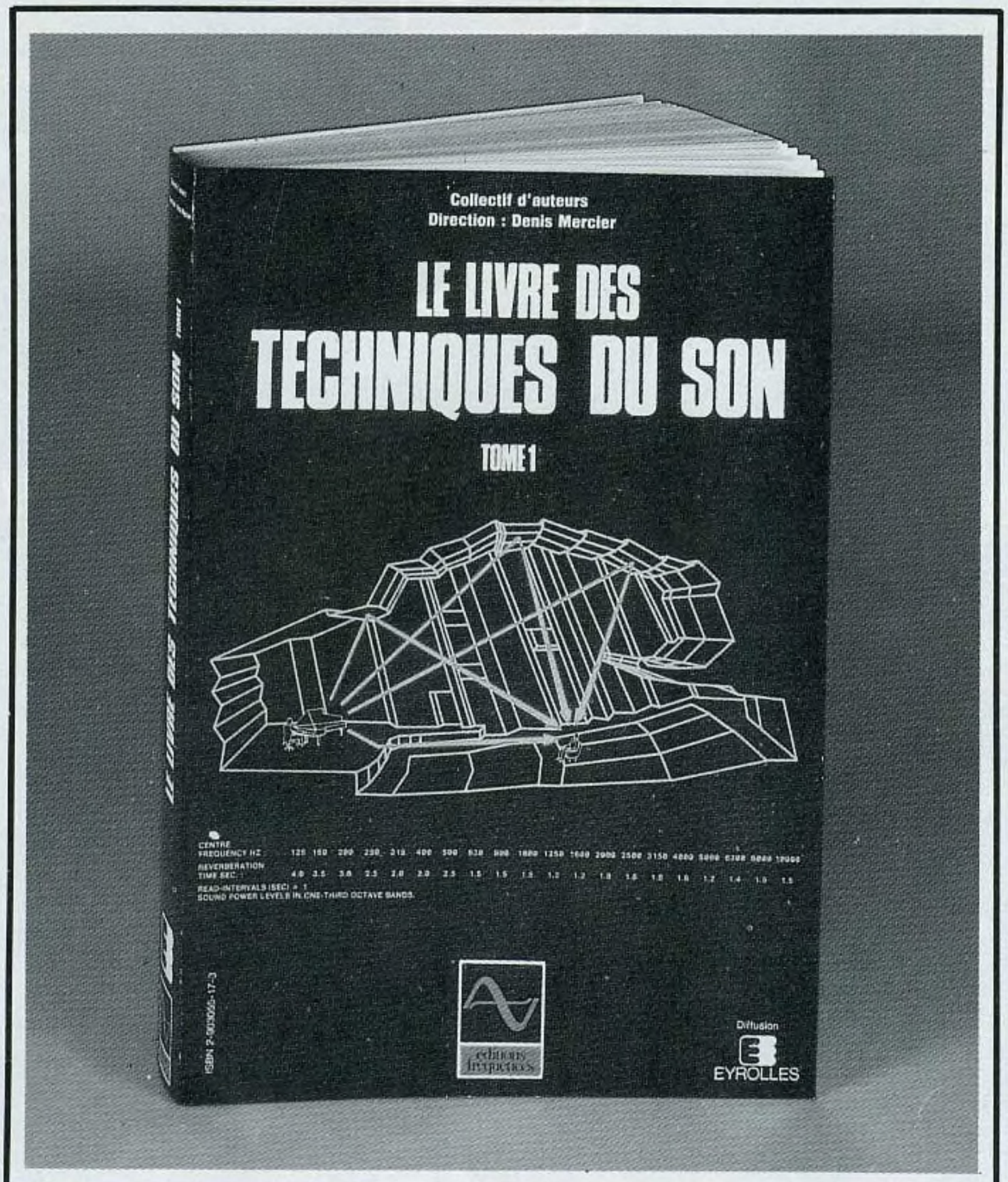
offre des ouvrages techniques **très actuels** rédigés par des auteurs passionnés et impliqués complètement dans le sujet qu'ils traitent.

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 :

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

LE SICOB

Pour la première fois, cette année, le SICOB ne s'est pas tenu au CNIT mais au Parc des Expositions de Villepinte Paris Nord.
Une des grandes nouvelles concerne la réorganisation de la gamme PC d'IBM. En effet, Big Blue annonce une nouvelle série de Personal Systems. Il ne faudra plus dire PC, mais PS. Il existera huit modèles. La disquette trois pouces et demi remplace désormais la cinq pouces un quart. La capacité de ces dernières est de 720 Ko ou 1,4 Mo. La carte graphique EGA est montée en standard et les outils de communication sont intégrés. Les disques durs iront de 20 à 125 Mo. Il existera un port souris IBM. Mais attention, la grande nouveauté réside dans l'arrivée d'un nouveau bus orienté multitâche spécialisé pour les 286 et autres 386. Un nouveau BIOS qui portera le nom de ABIOS. En fait, seul le premier modèle sera encore capable d'accepter les anciens formats de cartes additionnelles. Les PS accepteront des nouveaux systèmes d'exploitation, les DOS 3.3, OS/2 et l'AIX. L'AIX est présenté comme le système d'exploitation UNIX d'IBM.
Depuis plusieurs mois, les spécialistes en informatique annonçaient l'arrivée d'un CLONE-KILLER à 5 000 F ; en fait, la nouvelle politique est beaucoup plus ambitieuse. Elle consiste à se protéger pour l'avenir sans subir les attaques du Sud-Est asiatique. Avec un nouveau BIOS compatible avec l'ancien mais protégé par un microcode, des composants spécialisés, une redéfinition de la connectique, les copieurs vont souffrir. Il est évident que les grands éditeurs de progiciels vont dès à présent annoncer des versions compatibles de leurs produits. 1988 s'annonce très intéressante car il y aura pas mal de remous à la suite de cette importante information.
Parmi les nouveautés du salon, notons l'apparition du ADD-PAC (disque dur amovible) chez TANDON. La première version est composée de deux lecteurs 30 méga, mais il existera une version 100 méga pour la fin de l'année. Les prix sont très attractifs.
Il semble que XENIX de Microsoft et UNIX de ATT, qui seront fusionnés en une seule version pour les machines équipées d'un 386, pourraient bien être la solution, comme système d'exploitation pour les grosses machines de la gamme des micro-ordinateurs, finissant d'ailleurs par ne plus avoir grand chose d'un micro.
Apple craque, les nouveaux MACINTOSH SE et II seront capables d'exécuter des logiciels pour PC.
La P.A.O. devient accessible. En effet, la publication assistée par ordinateur n'est plus réservée aux grandes sociétés d'édition. Une imprimante laser associée à un scanner transforment, grâce à un logiciel spécialisé, votre micro en véritable atelier d'édition.

dB-Adresse

Charles-Henry Delaleu

dB-Adresse, progiciel de gestion d'adresses, est écrit en langage dBase III compilé. Il s'agit d'une application simple mais complète d'une base d'adresse. Outre les rubriques habituelles, des champs libres ont été ajoutés afin de disposer d'éventuelles extensions. Il semble – mais nous n'en avons pas la confirmation – que dB Outils ait été utilisé à la réalisation des menus et des masques. dB-Adresse est un outil bien fait qui a l'avantage d'être très facile à assimiler. En fait, quelques minutes suffisent à la maîtrise des commandes.

dB-Adresse engendre des états de sortie à partir d'une sélection multicritère. Un des grands avantages de dB-Adresse est d'être directement utilisable avec la fonction fusion (mailing) du traitement de texte NATHALIE-2. De ce fait, il est possible d'effectuer un tri et/ou une sélection afin d'obtenir des mailings spécialisés ou personnalisés.

Les quatre sous-menus de dB-Adresse sont :

1. **Saisie**
   Ce sous-menu permet de créer de nouvelles adresses.
2. **Mise à jour**
   Ce sous-menu permet la mise à jour des adresses, qui comprend :
   - la modification
   - la suppression.
3. **Edition d'adresses**
   Ce module est divisé en deux sous-modules :
   - la sélection d'enregistrements à l'aide d'opérateurs relationnels ;
   - l'édition des fiches sélectionnées vers un fichier (pour opérer un mailing avec un traitement de texte), à l'écran ou vers l'imprimante.

4. Fin de programme
Retour au DOS.

dB-Adresse fonctionne sur PC, XT et AT IBM ou compatibles version DOS 2.0 ou 3.0. Pour lancer l'application, il suffit de taper DBA. Dans le cas d'une installation sur disque dur, il faut créer un sous-répertoire portant le nom de DBA. Le transfert de dB-Adresse est alors très simple. La commande est la suivante :

COPY A : * . * C :

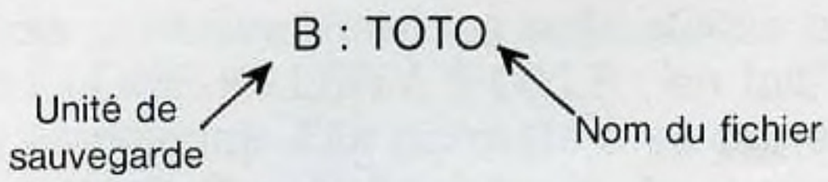
Le chargement effectué, le menu principal apparaît à l'écran.

```
                    ADRESSE

   1    Changement de fichier adresse.
   2    Catalogue du disque.
   3    Saisie d'une adresse.
   4    Mise à jour.
   5    Edition d'adresses.
   6    Quitter le programme courant.

  Pressez la touche de votre choix:
```

LES COMMANDES

Le chargement du fichier adresse

Le chargement du fichier adresse se fait de la manière suivante :



Attention : Il est impossible d'éditer un état à partir de deux fichiers ouverts en même temps. Si l'on désire éditer un état à partir d'une sélection, on doit préalablement construire un même fichier reprenant les informations de base.

La saisie d'une adresse

La saisie d'une adresse se fera en sélectionnant **3** sur le menu principal. Dès lors apparaîtra le masque de saisie des adresses à l'écran :

```
N° de code:

Titre:                          Nom:
                          Cplt Nom:

      Adresse:
Cplt adresse:

Code Postal:                 Ville:

         N° de téléphone:

Commentaires:

Code n°1:     Code n°2:       Code n°3:

(C)rée (A)n (S)uiv (P)réc (D)épt (L)ot (Q)
```

Les commandes de ce masque se sélectionnent en tapant la lettre qui correspond à la fonction choisie :

C → créer : Cette fonction crée un enregistrement. Il est possible pour corriger des erreurs d'utiliser les flèches et la touche INS.
NOTA : Un enregistrement est validé si le champ CODE ou NOM est non vide.

A → annuler : Efface la fiche commande.

S → suivant : Donne la fiche suivante.

P → précédente : Donne la fiche précédente.

D → déplacement : Saute n enregistrements.

L → lot de création : Idem que création, mais pour la saisie de plusieurs fiches.

Q → quitter

La mise à jour

Cette fonction reprend un écran similaire à l'écran de création. Les commandes sont :

M → mise à jour

A → (Ann) : Annule la fiche courante.

S → (Suiv) : Donne la fiche suivante.

P → (Préc) : Donne la fiche précédente.

D → (Déplct) : Déplacement.

Q → quitter

C → (Crit) : Cette fonction permet de rechercher une fiche sur un critère simple.

L'édition d'adresses

Accessible par la commande n° **5** du menu principal, l'édition d'adresses autorise :

- la sélection de fiches
- l'édition de rapports
- l'édition d'étiquettes.

```
EDITION

1  Sélection de fiches.
2  Edition de rapport.
3  Edition d'étiquettes.
4  Fin de sélection de fiches.
5  Quitter le menu EDITION.
```

La sélection de fiches est effectuée par un menu.

N C	Criteres définis par l'utilisateur
1	LISTE DES DEPARTEMENTS >= 75000
2	LISTE DES DEPARTEMENTS INFERIEURS A 76000
3	LISTE DES HABITANTS DE LA REGION PARISIENNE
4	LISTE DES PERSONNES N'HABITANT PAS LA R P
5	LISTE DES HABITANTS DE LA REGION RHONE-ALPE
6	LISTE DES X NE S'APPELANT PAS MART ET NON F

NOM	CODE	CODE1	CODE2	CODE3
CPTNOM			CP	

Créer Un Int Mém Suiv Préc Recher Eff Opter Dpt Q

Il est possible d'utiliser les opérateurs relationnels suivants :

- `>` supérieur à
- `<` inférieur à
- `=` égal à
- `< >` différent de
- `<=` inférieur ou égal à
- `> =` supérieur ou égal à

L'édition de rapports est réalisée dans le style de dBase III. L'édition d'étiquettes est effectuée par un retour au DOS et l'appel des fichiers LBLCOL.COM (écran couleur) ou LBLMONO.COM (écran N/B). Ce générateur d'étiquettes doit être utilisé pour créer un fichier de type XXXX.LBL qui contient tous les paramètres pour éditer une étiquette.

- Format de la feuille (40, 80, ..., 132 col)
- Format de l'étiquette.

CONCLUSION

dB-Adresse est un programme simple, facile à utiliser. Il peut être interfacé à de nombreux traitements de texte. Il correspond parfaitement à son cahier des charges. Le premier avantage est qu'il offre un rapport qualité-prix très attractif. Enfin un progiciel à un prix très raisonnable !

Nathalie 2

Charles-Henry Delaleu

Nathalie 2 est un progiciel de traitement de texte intéressant à plus d'un titre. Outre ses commandes très utiles, son prix est très attractif (990 F HT à ce jour). Ce programme est équipé d'un correcteur orthographique, d'une police de caractères spéciaux étrangers, d'une police de caractères spéciaux scientifiques et bien sûr de toutes les commandes habituellement rencontrées sur un programme de traitement de texte. Bien que la documentation qui l'accompagne soit abondante, on peut déplorer l'absence d'illustrations.
Nathalie 2 fonctionne sur PC ou compatible. Il utilise le système d'exploitation MS-DOS.
L'installation du logiciel ne pose aucun problème ; après le masque de départ le menu principal apparaît en appuyant sur la touche ESC.

MENU PRINCIPAL

F 1 : Sytème/aide
F 2 : Fenêtre/règle
F 3 : Copier
F 4 : Effacer
F 5 : Sup-marque
F 6 : Déplacer
F 7 : Paragraphe
F 8 : Min/Maj
F 9 : Rechercher
F10 : Remplacer

Sur ce menu existe en F1 une commande d'aide. La fonction aide sur Nathalie 2 est bien illustrée. Elle reprend chacune des commandes et ses diverses applications.

ENSEMBLE DES COMMANDES REPRISES PAR LA FONCTION AIDE

```
Esc:Fin de l'aide  F1:reprendre édition   Flèches:Sélection d'un sujet:
Bases            Commandes DOS   Annotations      Fusion: variab  Index/TdM
N° auto. notes   Com-point I     Lignes commands  Fusion: canevas Réglettes
Reformat auto    Com-point II    Hauts/bas pages  Fusion: étapes  Déplacement
Caracs étrang    Com-point III   Emplacement      Divers          Disponible
Caracs maths     Typos           Reformat manuel  Sauts de pages  Césures
Copie/bloc       Entrer texte    Marges /tabs     Mise en page    Systeme/fich
Mouvmt curseur   Gestion fichier Marquer texte    Impression      Fenêtres
Effacer texte    Recher/Remplt   Fusion           Macros          Glossaire
Justification    Filets/graphes  Dictionnaire     Disponible      Disponible
```

Le système d'aide fonctionne à partir de menus déroulants. Il est possible d'accéder au DOS directement de Nathalie 2. Ceci peut, dans bien des cas, rendre de grands services.

COMMANDES DU DOS

Voici certaines commandes du DOS. Nathalie possède des touches pour la plupart des opérations sur fichiers. «A>» est la sollicitation.

```
♦FORMAT♦
 A>FORMAT B:
    formatte disquette sur lecteur B

♦DIR♦
 A:DIR B:
    Liste les fichiers du lecteur B

♦TYPE♦
 A:TYPE nonfich
   montre le contenu de "nomfich"
```

```
 ♦RENAME
A>RENAME  anciennom nouveaunom
  renomme l'"ancien" en"nouveau"

 ♦COPY
A>COPY "fichoriqin" "fichdesti"
  copie  d'un fichier vers fichier

 ♦DEL
A>DEL nomfich
  efface le fichier "nomfich"
```

Parmi les commandes, notons :

– les déplacements.

a. Dans une portion de texte

DEPLACEMENT

L'écran est une fenêtre qui vous permet de voir une portion d'un fichier. Vous pouvez aller dans toutes les directions et même sauter directement à l'endroit voulu.

♦VERS LE HAUT♦	♦HORIZONTALEMENT♦
par ligne PgUp	vers la droite Flèche droite
par parag. Ctl PgUp	vers la gauche Flèche gauche
fenêtre Shf PgUp	
	♦SAUT♦
♦VERS LE BAS♦	début du fich. Alt + ou Shf Gris+
par ligne PgDn	fin du fichier Alt - ou Shf Gris-
par parag. Ctl PgDn	page précédente Shf Ctl PgUp
fenêtre Shf PgDn	page suivante Shf Ctl PgDn

b. Dans un fichier complet

DEPLACEMENT

Se déplacer dans un fichier

Vous pouvez sauter à n'importe quel endroit dans un fichier. Vous pouvez également laisser un repère à un endroit voulu.

♦EMPLACEMENT CURSEUR ♦	♦REPERES♦
choisir Shf F9	placer premier repère Ctl Home
sauter à ligne Alt F9 F7	sauter au premier repère Ctl End
sauter à colonne Alt F9 F8	placer 2ème repère Shf Ctl Home
seuter ligne/page Alt F9 F9 F10	sauter au 2ème repère Shf Ctl End

Lorsque vous faites Shift F9, la ligne de statut dit :

Ligne x/xx dans fich. Colonnx/xx. Ligne x/xx sur pages/xx.

Lorsque vous faites Alt F9, la ligne de statut dit :

F7 : Ligne xx/xx dans fich. F8 : Colonnexx/xx. F9 : Lignexx/xx sur F10 : Page xx/xx

Une des particularités de Nathalie 2 concerne la possibilité d'utiliser des fenêtres.

FENETRES

Les fenêtres sont créées avec la réglette On peut aussi «séparer» l'écran en deux fenêtres. Vous pouvez recevoir deux fichiers différents sur l'écran.

♦CREATION ♦	♦DEPLACEMENT ♦
1. Le curseur au milieu de l'écran. 2. Tapez F2 pour la réglette. 3. Tapez Flèches pour déplacement dans une fenêtre. 4S Tapez F1 puis F6 pour donner le nom du 2ème fichier si désiré.	1. Faites F2 pour aller ds réglette 2. Faites flèches pour vous déplacer dans l'autre fenêtre.

Il ne fait aucun doute qu'en fonction de son prix de vente, le premier avantage de Nathalie 2 est son correcteur orthographique.

CORRECTEUR ORTHOGRAPHIQUE

Dictionnaire

Le dictionnaire orthographique WORDS.MAG est chargé lors de la mise en route de la fonction Dictionnaire. Vous pouvez aussi charger une liste auxiliaire appelée WORDS.USE que vous pouvez créer avec Nathalie.

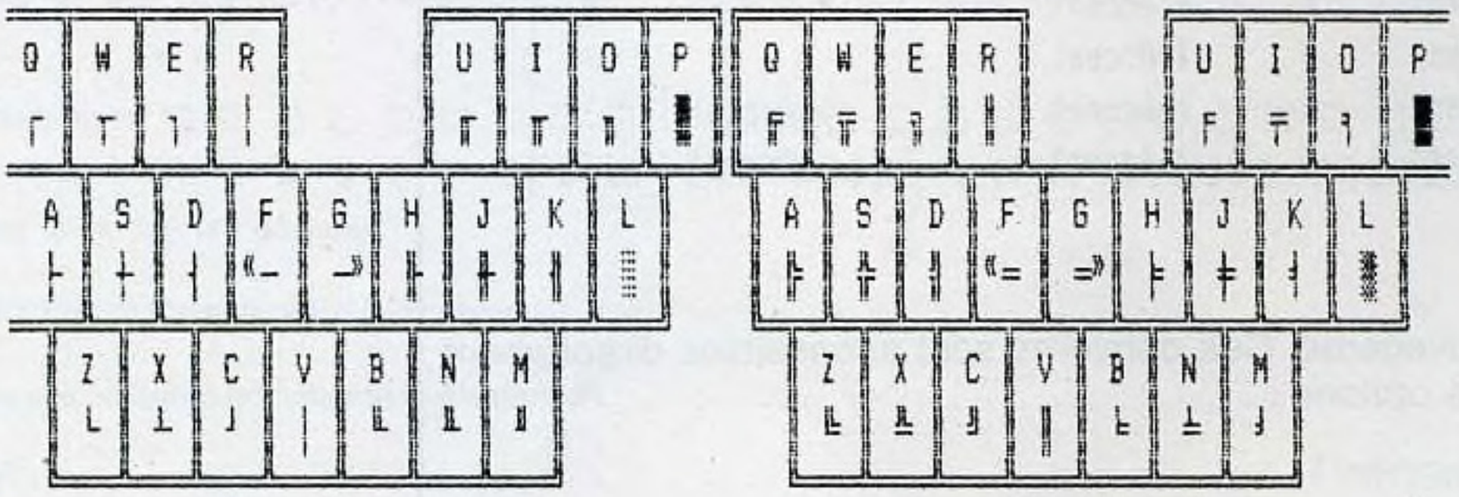
♦VERIFICATION♦		♦RECHERCHE HOMONYME♦	
dernier mot tapé	Alt F2, F2	sur un mot incorrect	Alt F2, F3
auto (temps réel)	Alt F2, F7		
		♦DICTIONNAIRE AUXILIAIRE♦	
♦BALAYAGE D'UN TEXTE♦		charge en mémoire	Alt F2, F5
en avant	Alt F2, Gris+	ajoute un mot	Alt F2, F4
en arrière	Alt F2, Gris-	sauvegarde sur disque	Alt F2, F6

LE PSEUDO-GRAPHIQUE

Il peut être intéressant afin de mieux présenter certains documents d'utiliser des caractères pseudo-graphiques. Ils aident à la présentation de tableaux par exemple :

FILETS ET ENCADREMENTS

Appendice



Avertissement : Certains PC dits «compatibles» ont du mal à faire ces graphiques.

LES CARACTERES SPECIAUX

Un autre avantage de Nathalie concerne l'intégration en standard de police de caractères spéciaux.

a. Caractères étrangers

CARACTERES SPECIAUX : ETRANGERS

Tapez la première touche, puis la touche ACCENT, puis la deuxième touche. Attention ! la touche Accent n'est disponible que sur certains claviers.

a Accent " ä	a Accent ` à	a Accent ^ â	o Accent a å
A Accent " Ä	e Accent ` è	e Accent ^ ê	A Accent o Å
e Accent " ë	i Accent ` ì	i Accent ^ î	
i Accent " ï	o Accent ` ò	o Accent ^ ô	a Accent _ ª
o Accent " ö	u Accent ` ù	u Accent ^ û	o Accent _ º
O Accent " Ö			
u Accent " ü	a Accent ' á	c Accent , ç	
U Accent " Ü	E Accent ' é	C Accent , Ç	c Accent / ¢
y Accent " ÿ	e Accent ' é	n Accent ~ ñ	- Accent L £
	i Accent ' í	N Accent ~ Ñ	= Accent Y ¥
a Accent e æ	o Accent ' ó	? Accent ? ¿	t Accent P ₧
A Accent E Æ	u Accent ' ú	! Accent ! ¡	- Accent f ƒ

b. Caractères scientifiques

CARACTERES SPECIAUX : SCIENTIFIQUES

Tapez la première touche, puis la touche ACCENT, puis la deuxième touche. Attention ! la touche Accent n'est disponible que sur certains claviers.

1 Accent `	a Accent / α	f Accent / ϕ	1 Accent 2 ½
2 Accent `	b Accent / β	e Accent / ε	1 Accent 4 ¼
3 Accent `	g Accent / Γ		: Accent ` ÷
4 Accent `	p Accent / π	! Accent ` ∏	/ Accent ` √
5 Accent `	S Accent / Σ	# Accent ` ∎	+ Accent ` ±
6 Accent `	s Accent / σ	$ Accent ` ⌐	. Accent ` •
7 Accent `	m Accent / µ	% Accent ` °	
8 Accent `	t Accent / τ	^ Accent ` ∩	= Accent ` ≡
	F Accent / Φ	& Accent ` ∘	~ Accent ` ≈
@ Accent `	h Accent / θ	* Accent ` ²	
[Accent ` ⌐	O Accent / Ω	(Accent ` ⌠	> Accent ` ≥
] Accent ` ¬	d Accent / δ	) Accent ` ⌡	< Accent ` ≤

LES SAUVEGARDES

Il existe plusieurs types de sauvegarde. Ces dernières sont accessibles directement par le menu principal. Liste des options :

LES SAUVEGARDES

ESC : Annule le menu de commande → retour à l'éditeur

F1 = AIDE : Affiche le menu AIDE (notice d'emploi).

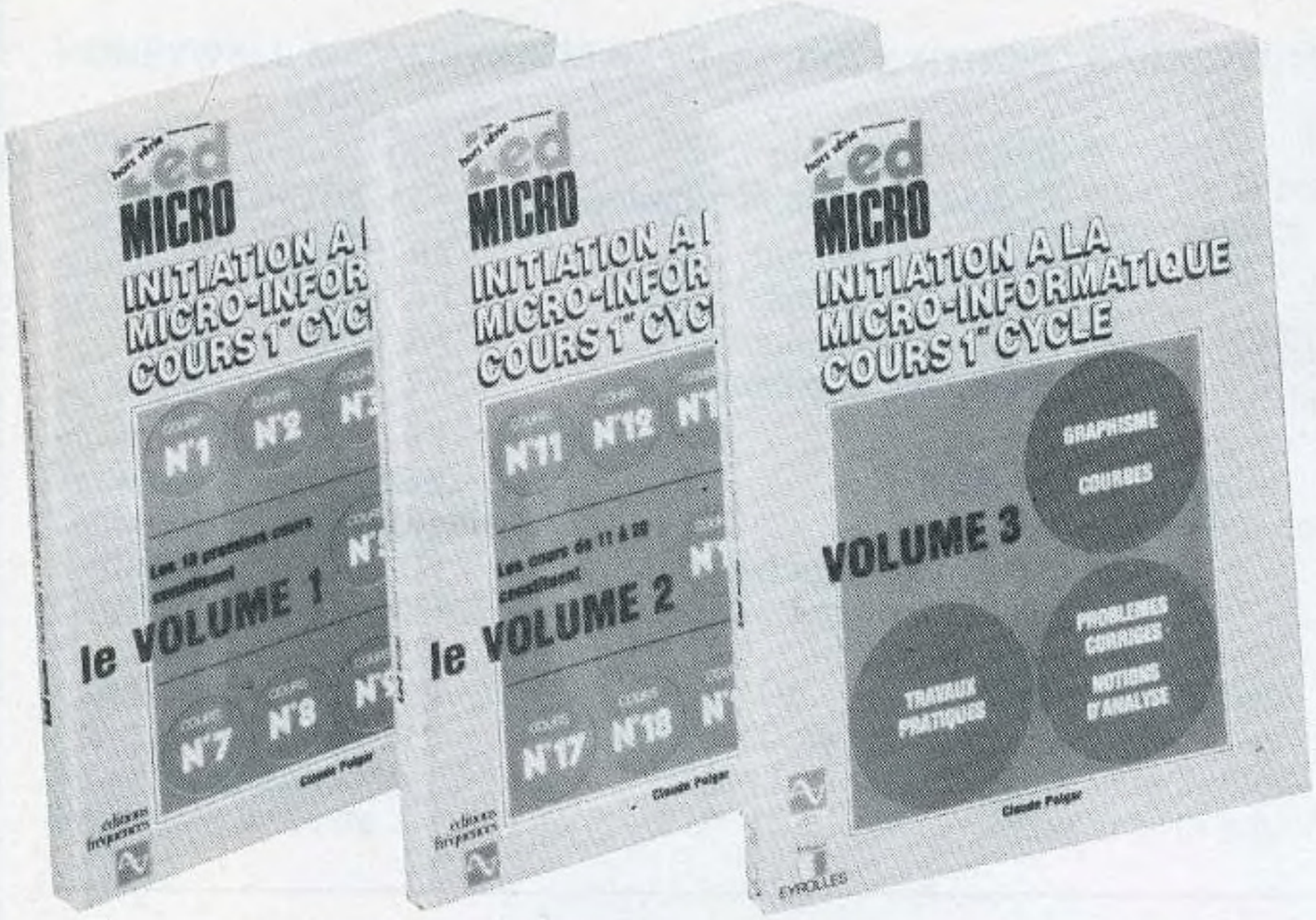
F2 = Sort : Sauvegarde le texte s'il a été modifié depuis le retour au DOS.
F3 = Sauv : Sauvegarde le texte même s'il n'a pas été modifié, puis continue l'édition.
F4 = Command : Sauvegarde le texte s'il a été modifié puis reste résident, mais exécute une commande DOS.
F5 = NOM : Permet de sauvegarder un fichier sur disque sous un nouveau nom.
F6 = Fich : Sauvegarde un texte s'il a été modifié puis charge ou crée un autre fichier.
F7 = Impri : Sauvegarde un texte s'il a été modifié puis imprime le texte.
F8 = Dir : Catalogue des fichiers contenus sur le disque.
Nonsauv : Annule les actions de sauvegarde (F2, F4, F6, F7).

CONCLUSION

Nous venons de présenter très brièvement le traitement de texte Nathalie 2. Il est impossible de décrire en si peu de pages l'ensemble de ses possibilités. Toutefois, nous avons décrit les plus intéressantes, c'est-à-dire celles que l'on ne trouve pas dans tous les traitements de texte. Il est bon de rappeler les commandes généralement implantées dans un tel progiciel :

- Fusion, mailing, étiquettes
- Déplacement, copie, insertion
- Justification, césures
- Police de caractères.

Bien sûr, Nathalie 2 les possède toutes. Notons aussi que le progiciel dB Adresse du même éditeur (EBP) s'interface sans problème. Ainsi, il sera possible d'effectuer à moindre coût des mailings personnalisés de qualité.
La seule conclusion que l'on puisse formuler sur ce progiciel : enfin un traitement de texte performant, efficace, complet, à un prix très très étudié.



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 :

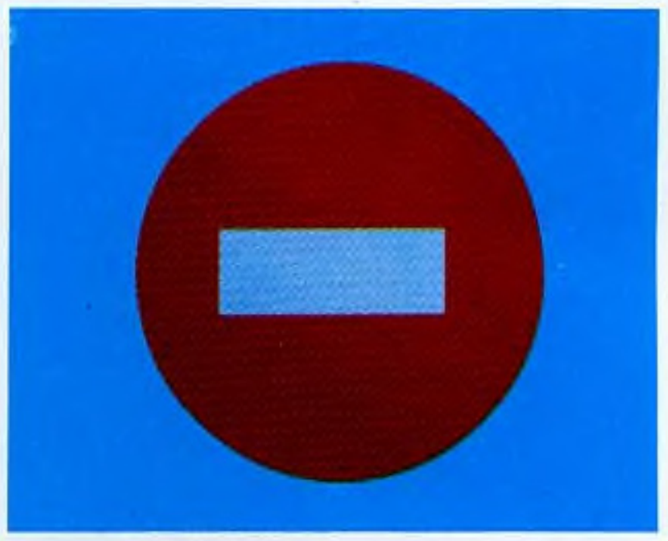
J'enseigne les mathématiques dans une Université de Sciences Humaines et j'ai été amenée, alors que n'avais moi-même reçu aucune formation à la micro-informatique, à initier des étudiants de 1re année de Mathématiques et Sciences Sociales (MASS) à la programmation en S-BASIC (sur Goupil-3), dans le but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

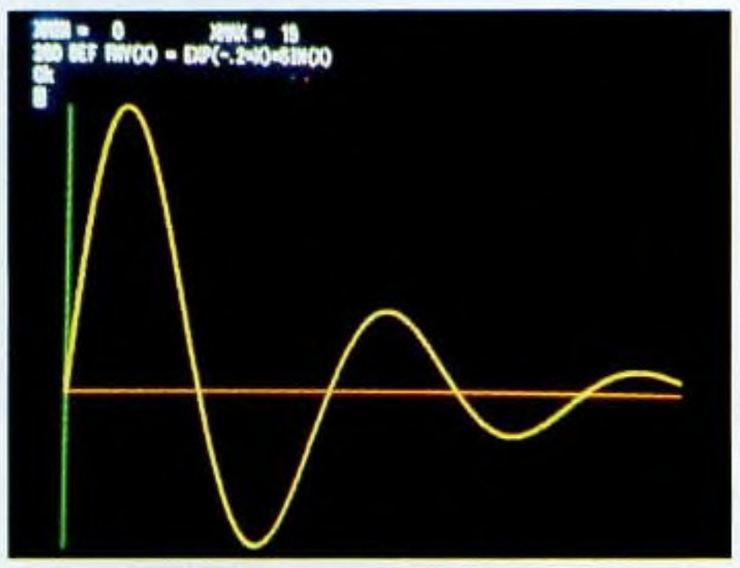


3.17 Compléments

★ 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.

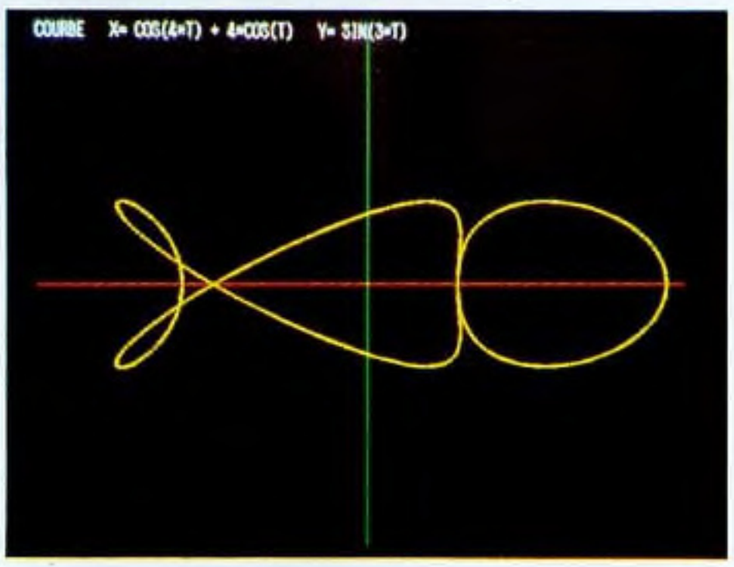
3.18 Graphisme

★ 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).



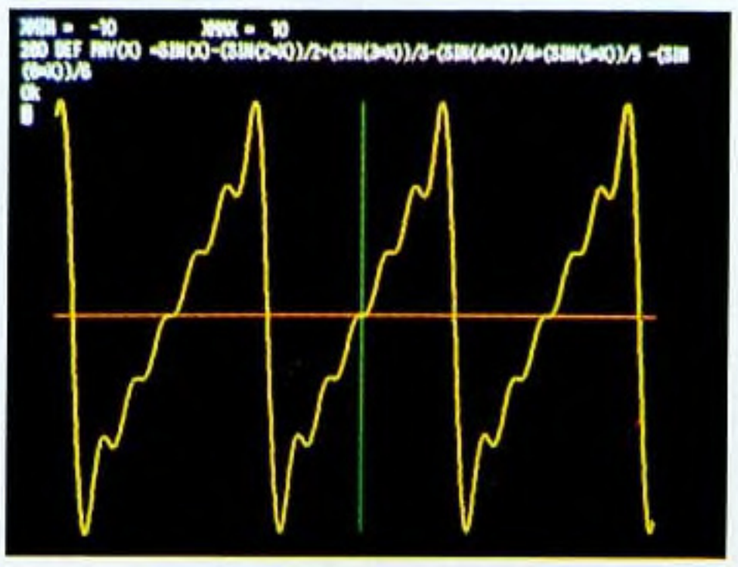
3.19. Dessin des courbes

★ 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