CHAPITRE 4

LES TYPES DE DONNÉES DU LANGAGE PASCAL

OBJECTIFS

- PRÉSENTER LES NOTIONS D'ÉTIQUETTE, DE CONSTANTE ET DE VARIABLE DANS LE CONTEXTE DU LANGAGE PASCAL.
- DÉFINIR LES TYPES DE DONNÉES STANDARDS ET NON STANDARDS DE CE LANGAGE.
- RÉPERTORIER LES FONCTIONS STANDARDS INTRINSÈQUES AU LANGAGE PASCAL.

Ce chapitre traite des différents types de données dans le langage PASCAL. Dans ce contexte, nous commencerons par préciser le sens de certaines notions de base telles que *étiquette*, *constante* et *variable*. Par la suite, nous distinguerons les types de données standards et non standards, ce qui nous amènera finalement aux fonctions standards du langage PASCAL.

4.1 LA NOTION D'ÉTIQUETTE

Dans le langage PASCAL, on désigne par *étiquette* un nombre entier, suivi des *deux-points* (:), placé au début d'une instruction exécutable et qui identifie de manière unique cette instruction.

Exemple 4.1

Association d'une étiquette à une instruction :

```
3 : WRITELN('C"EST LA TROISIEME ETIQUETTE DU PROGRAMME.');
```

Dans cet exemple, le chiffre 3 placé au début de l'instruction constitue l'étiquette. De cette façon, il est possible de se référer à cette instruction simplement en spécifiant l'étiquette 3. Le mot réservé correspondant à *étiquette* en PASCAL est LABEL.

Les déclarations d'étiquettes s'effectuent conformément au diagramme syntaxique de la figure 4.1. Selon ce diagramme, pour déclarer une étiquette, on place le nombre entier sans signe représentant cette étiquette entre le mot clé LABEL et un point-virgule. Pour déclarer plusieurs étiquettes, on place d'abord le mot clé LABEL, suivi de la liste des nombres entiers sans signe représentant ces étiquettes, ceux-ci étant séparés l'un de l'autre par une virgule, le tout se terminant par un point-virgule.

Exemple 4.2

a) Déclaration d'une étiquette :

LABEL 1;

b) Déclaration de trois étiquettes :

LABEL 4, 8, 10;

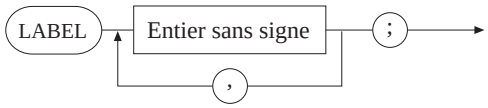

FIGURE 4.1
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION D'ÉTIQUETTES.

4.2 LES CONSTANTES ET LES VARIABLES

D'une manière générale, les données sont manipulées dans l'ordinateur soit comme des constantes, soit comme des variables. Il importe donc de spécifier dans les programmes qui manipulent ces données – et ce dès le départ – s'il s'agit d'une constante ou d'une variable. C'est ce qu'on appelle une *déclaration de constante* ou de variable.

4.2.1 La notion de constante

Une *constante* est une donnée qui a une valeur fixe tout au long du programme dans lequel elle est utilisée. En PASCAL, on en distingue deux types : les *constantes numériques* et les *constantes chaînes de caractères*.

Un diagramme syntaxique de constante numérique a été présenté au chapitre précédent à la figure 3.9. Il est toutefois possible de généraliser cette définition en introduisant les notions déjà connues de nombre sans signe et d'identificateur. Ainsi, selon la figure 4.2,

on peut obtenir une constante numérique en affectant d'un signe (+ ou –) un nombre sans signe ou un identificateur.

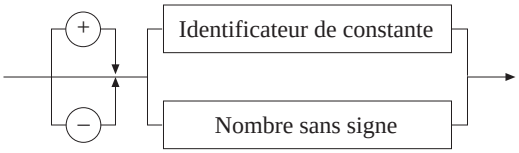

FIGURE 4.2
DIAGRAMME SYNTAXIQUE D'UNE CONSTANTE NUMÉRIQUE.

Quant à la constante chaîne de caractères, aussi appelée *littérale*, elle correspond à une suite de caractères encadrée par deux apostrophes. La figure 4.3 en donne le diagramme syntaxique.

FIGURE 4.3
DIAGRAMME SYNTAXIQUE D'UNE CONSTANTE CHAÎNE DE CARACTÈRES.

Pour associer un identificateur à une constante numérique ou chaîne de caractères, il faut faire une déclaration de constante. Ainsi, conformément à la figure 4.4, pour déclarer une seule constante, on fait suivre le mot clé CONST du nom de la constante (identificateur de constante), suivi du symbole d'égalité, suivi de la constante, suivi finalement d'un point-virgule. Dans le cas d'une déclaration de plusieurs constantes, on répète l'opération précédente pour chaque nouvelle constante à déclarer, sans toutefois répéter le mot clé CONST.

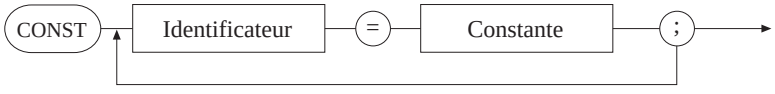

FIGURE 4.4
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION DE CONSTANTE.

Exemple 4.3

a) Déclaration d'une constante numérique :

```
CONST   huit = 8;
```

b) Déclaration de deux constantes numériques et d'une constante chaîne de caractères :

```
CONST   huit = 8;
        tps = 0.07;
        manuel = 'INTRODUCTION A LA PROGRAMMATION';
```

Notons qu'on ne peut pas modifier la valeur d'un identificateur de constante à l'intérieur d'un programme.

4.2.2 La notion de variable

La mémoire de l'ordinateur se compose d'une série de cellules répondant chacune à une adresse en mémoire. Ces cellules, dont la taille dépend du type d'ordinateur, contiennent des données accessibles par l'intermédiaire de noms symboliques appelés *variables*. Une variable est donc un nom donné à une adresse en mémoire, permettant ainsi de consulter ou de modifier le contenu de cette dernière**.**

Le contenu d'une adresse est constitué d'un ensemble d'éléments binaires qui, pris isolément, n'ont aucune signification précise. Ce n'est que le contexte qui déterminera le type de données qui est représenté. Par exemple, dans la mémoire d'un ordinateur, la suite binaire

1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 1

peut aussi bien représenter un nombre entier, un nombre réel, une chaîne de caractères ou un tout autre type de données. C'est le type de la variable, telle que celle-ci a été déclarée dans le programme, qui nous renseignera.

Toute variable utilisée dans un programme PASCAL doit être déclarée à la section de déclaration des variables. La syntaxe de cette déclaration est donnée à la figure 4.5. Selon ce diagramme syntaxique, pour déclarer une variable, on fait suivre le mot clé VAR du nom (identificateur) de la variable, suivi des *deux-points* (:), suivi du type de la variable, suivi finalement d'un point-virgule. Dans le cas où la déclaration porte sur plusieurs variables de même type, la liste des variables séparées deux à deux par une virgule remplace la variable unique de l'opération précédente. Lorsque les variables à déclarer sont de types différents, on les regroupe selon leur type. Cette notion de *type* sert à préciser l'ensemble des valeurs que peut prendre une donnée. À ce titre, il convient de mentionner qu'il existe deux catégories de types de données en PASCAL : les *types standards* et les *types non standards*, lesquels font l'objet des sections qui vont suivre.

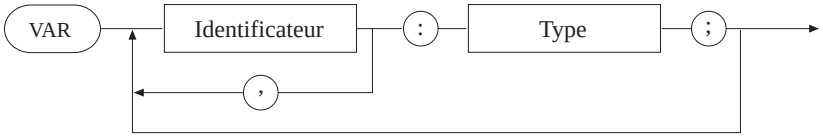

FIGURE 4.5
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION DE VARIABLES.

Exemple 4.4

a) Déclaration d'une variable :

```
VAR   age : type;
```

b) Déclaration de quatre variables de même type :

```
VAR   age, annee, mois, jour : type;
```

c) Déclaration de plusieurs variables de types différents :

```
VAR   age, annee, mois, jour : type1;
      nom, prenom : type2;
      salairbrut, salairnet : type3;
```

4.3 LES TYPES DE DONNÉES STANDARDS

Les types de données standards ne sont autres que des types de base, prédéfinis dans le langage et qui, par conséquent, peuvent être utilisés dans les programmes sans une définition préalable. En PASCAL, on compte quatre types de données standards :

– les entiers qui correspondent au mot clé INTEGER,
– les réels (REAL),
– les caractères (CHAR),
– les booléens (BOOLEAN).

4.3.1 Les entiers

On désigne par *constante entière*, ou tout simplement *entier*, tout nombre entier positif, négatif ou nul. Un nombre entier positif ou nul est désigné, en PASCAL, par le terme *entier sans signe* dont le diagramme syntaxique a été présenté au chapitre précédent et fait l'objet de la figure 3.7. De même, un nombre entier négatif est considéré comme un entier sans signe auquel on affecte un signe négatif (–). Ainsi, comme le montre la figure 4.6, une constante entière n'est rien d'autre qu'un entier sans signe affecté d'un signe positif ou négatif. Ce qui peut encore s'écrire, en notation BNF :

<signe> := + / –
<entier> := <entier sans signe> / <signe> <entier sans signe>

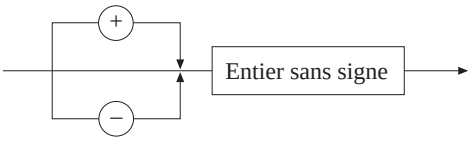

FIGURE 4.6
DIAGRAMME SYNTAXIQUE D'UN ENTIER OU CONSTANTE ENTIÈRE.

La représentation en mémoire des nombres entiers est assujettie à des contraintes matérielles qui limitent la taille des mots ou des registres utilisés par les ordinateurs. En effet, pour un ordinateur à mots de *n* bits, le plus grand et le plus petit entiers que l'on peut représenter sont respectivement :

$$N_{max} = + 2^{n-1} - 1$$
$$N_{min} = - 2^{n-1}$$

Ainsi, pour un micro-ordinateur à mots de 16 bits (n = 16), on a :

$$N_{max} = + 2^{n-1} - 1 = + 2^{16-1} - 1 = + 32\ 767$$
$$N_{min} = - 2^{n-1} = - 2^{16-1} = - 32\ 768$$

En PASCAL, la valeur N_{max} prend une signification particulière : c'est le plus grand entier qu'un ordinateur peut traiter. Aussi, l'appelle-t-on MAXINT, mot clé résultant de la contraction des mots « MAXimum INTeger » et dont la valeur est prédéfinie. En PASCAL, il existe le type LONGINT (pour LONG INTeger) pour désigner des entiers plus grands que MAXINT.

Exemple 4.5

a) Déclaration d'une variable entière :

```
VAR   age : INTEGER;
```

b) Déclaration de quatre variables entières :

```
VAR   age, annee, mois, jour : INTEGER;
```

ou

```
VAR   age : INTEGER;
      annee, mois : INTEGER;
      jour : INTEGER;
```

4.3.2 Les réels

Tout nombre réel N peut être écrit sous la forme :

$$N = 0.f \times 10^e$$

où f désigne un nombre entier représentant la partie fractionnaire ou mantisse, et e un nombre entier positif ou négatif représentant une puissance de 10.

Cette expression est donc la représentation en point flottant du nombre réel N.

Exemple 4.6

a) Représentation en point flottant du réel 3.1416 :

$3.1416 = 3.1416 \times 10^0 = 0.31416 \times 10^1$

b) Représentation en point flottant du réel 0.000005 :

$0.000005 = 0.000005 \times 10^0 = 0.5 \times 10^{-5}$

c) Représentation en point flottant du réel 1945 :

$1945 = 1945 \times 10^0 = 0.1945 \times 10^4$

Il est donc clair qu'on peut représenter tout nombre réel par deux entiers dont l'un serait la mantisse et l'autre l'exposant. Il suffit, pour cela, de choisir un format approprié pour les identifier dans la chaîne de bits représentant le nombre réel en question. Ainsi, le nombre maximal qu'on peut représenter en point flottant dépend du nombre d'éléments binaires composant chacune de ces parties dans la conception du système. Par exemple, si on choisit un format selon lequel 5 bits sont réservés à la mantisse et 3 bits à l'exposant, pour un mot de mémoire de 8 bits, on a ce qui suit :

valeur maximale de mantisse : $2^5 - 1 = 32 - 1 = 31$
valeur maximale de l'exposant : $2^3 - 1 = 8 - 1 = 7$.

Donc, le plus grand réel que l'on peut représenter dans un tel format est :

$$0.31 \times 10^7 = 3{,}100{,}000.00$$

Évidemment, plus l'exposant est élevé, plus le nombre représenté sera considérable; et plus la mantisse est grande, plus la précision est appréciable.

Étant donné que mantisse et exposant peuvent, à leur tour, être positifs ou négatifs, il importe de retenir, dans chaque cas, un bit pour indiquer le signe. Si on convient d'utiliser le bit 1 pour le signe négatif et 0 pour le signe positif, un nombre réel peut être représenté selon le format de la figure 4.7. Notons que, dans une telle représentation, le signe de la mantisse est également celui du nombre réel représenté.

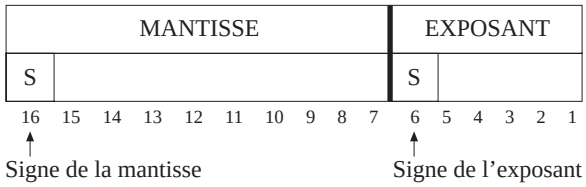

FIGURE 4.7
FORMAT DE REPRÉSENTATION D'UN NOMBRE RÉEL EN POINT FLOTTANT.

Exemple 4.7

Considérons un ordinateur à mots de 16 bits. Un réel y est représenté selon la convention et le format de la figure 4.7, avec 10 bits de mantisse, 1 bit de signe de mantisse, 4 bits d'exposant et 1 bit de signe d'exposant. Pour un tel ordinateur, que vaut la suite binaire :

1 0100110010 0 1011

interprétée comme un réel? Examinons cette suite :

– Le bit de signe de la mantisse étant 1, il s'agit là d'un nombre négatif.
– La mantisse étant 0100110010, elle correspond au nombre décimal (base 10) 306.
– Le bit de signe de l'exposant étant 0, l'exposant est donc positif.
– L'exposant étant 1011, il correspond au nombre décimal 11.

Donc, cette suite binaire équivaut à :

$-0.306 \times 10^{11} = -30{,}600{,}000{,}000.00$

On peut constater que, même si les réels sont eux aussi sujets à des limites en PASCAL, ils permettent malgré tout de représenter des nombres plus grands que ceux que permettent les entiers. Aussi, est-il possible de traiter les nombres entiers qui dépassent MAXINT tout simplement en les considérant comme des réels (Exemple 4.6c).

La déclaration des variables réelles s'effectue conformément au diagramme syntaxique de la figure 4.5.

Exemple 4.8

a) Déclaration d'une variable réelle :

```
VAR   salairbrut : REAL;
```

b) Déclaration de deux variables réelles :

```
VAR   salairbrut, salairnet : REAL;
```

ou

```
VAR   salairbrut : REAL;
      salairnet : REAL;
```

4.3.3 Les caractères

Comme nous l'avons mentionné au chapitre précédent, tout langage de programmation se définit à partir d'un ensemble de symboles constituant son alphabet. En PASCAL, ces symboles correspondent au type CHAR, mot clé servant à désigner les caractères pouvant être affichés à l'écran ou imprimés.

La déclaration des variables caractères s'effectue conformément au diagramme syntaxique de la figure 4.5.

Exemple 4.9

a) Déclaration d'une variable de type caractère :

```
VAR   sexe : CHAR;
```

b) Déclaration de deux variables de type caractère :

```
VAR   sexe, reponse : CHAR;
```

ou

```
VAR   sexe : CHAR;
      reponse : CHAR;
```

4.3.4 Les booléens

Il arrive souvent que l'on veut savoir si un énoncé est *vrai* ou *faux*. Cette situation logique ou booléenne nécessite deux possibilités qui peuvent être représentées, dans la mémoire de l'ordinateur, par un bit. Ce bit équivaut à 0 pour la valeur vraie et à 1 pour la valeur fausse, ou vice versa suivant la convention adoptée. En PASCAL, une variable est dite de type *booléen* (BOOLEAN) lorsqu'elle ne peut prendre que l'une des deux valeurs suivantes : TRUE (vraie) ou FALSE (faux). Soulignons que BOOLEAN, TRUE et FALSE sont trois mots clés du langage PASCAL.

La déclaration des variables booléennes s'effectue conformément au diagramme syntaxique de la figure 4.5.

Exemple 4.10

a) Déclaration d'une variable de type booléen :

```
VAR   fini : BOOLEAN;
```

b) Déclaration de deux variables de type booléen :

```
VAR   fini, trace : BOOLEAN;
```

ou

```
VAR   fini : BOOLEAN;
      trace : BOOLEAN;
```

4.4 LES TYPES DE DONNÉES NON STANDARDS

Le langage PASCAL permet au programmeur de définir, à partir des quatre types standards que nous venons de voir, des types propres à un problème donné. De tels types sont dits *non standards*; c'est le cas des tableaux, des chaînes de caractères, des énumérations, des intervalles, des ensembles, des enregistrements, des fichiers et des pointeurs. Avant de les utiliser dans un programme PASCAL, il faut d'abord les

déclarer. Les tableaux, les ensembles, les enregistrements et les fichiers constituent des données structurées, et les quatre autres des types de base. La figure 4.8 présente le diagramme syntaxique d'une déclaration de type.

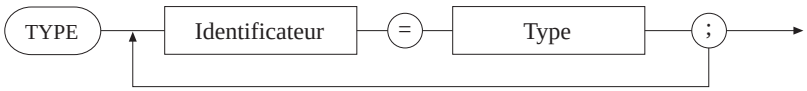

FIGURE 4.8
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION DE TYPE.

4.4.1 Les tableaux

Un *tableau* est une structure comprenant un nombre fixe d'éléments de même type, répondant à un nom collectif. Ces éléments peuvent être de type *entier*, *réel*, *caractère*, *booléen* ou autre. Dans ce dernier cas, il est de la responsabilité du programmeur de s'assurer que tout type non standard a bel et bien été défini au préalable avant d'y faire référence.

Un paramètre très important d'un tableau, c'est sa *taille*, qui n'est autre que le nombre d'éléments qu'il peut contenir. La taille d'un tableau est limitée par la capacité de mémoire de l'ordinateur considéré.

On serait porté à croire que les éléments d'un tableau sont consécutifs dans cette mémoire, ce qui n'est pas toujours le cas. Intuitivement, un tableau n'est qu'un ensemble de valeurs repérées chacune par un jeu d'indices. Deux opérations fondamentales touchent les tableaux : l'emmagasinage des valeurs dans le tableau et la restitution de ces valeurs au besoin. On accède à un élément du tableau en spécifiant le nom de celui-ci et un ou plusieurs indices qui se réfèrent à cet élément. Le nombre d'indices à spécifier dépend du nombre de dimensions du tableau.

Les tableaux à une dimension

Un *tableau à une dimension* est une structure linéaire dans laquelle chaque élément est repéré par un seul indice. On l'appelle aussi un *vecteur*. D'où le mot clé ARRAY utilisé en PASCAL pour le déclarer.

Exemple 4.11

a) Déclaration d'un type tableau à une dimension :

```
TYPE  vecteur = ARRAY [1..10] of INTEGER;
```

b) Déclaration subséquente de deux variables de type vecteur :

```
VAR   vect1, vect2 : vecteur;
```

ou

```
VAR   vect1 : vecteur;
      vect2 : vecteur;
```

Dans une déclaration de tableau, l'expression entre les crochets sert à spécifier le domaine des valeurs possibles des indices.

Les tableaux à deux dimensions

Le cas le plus simple de tableau à plusieurs dimensions est le *tableau à deux dimensions*. C'est une structure de données dans laquelle chaque élément est repéré par deux indices : un indice indiquant le numéro de la ligne et un autre indiquant le numéro de la colonne. C'est donc un vecteur dont chacun des éléments ou composantes est lui-même un vecteur. D'où leur appellation équivalente de *matrice* dans laquelle l'élément auquel on se réfère se trouve à l'intersection de la ligne et de la colonne spécifiées par les indices.

Exemple 4.12

Soit la déclaration suivante :

```
TYPE  matrice = ARRAY[1..5, 1..4] OF REAL;
VAR   chemin1, chemin2 : matrice;
```

Elle permet de définir deux tableaux ayant chacun cinq lignes et quatre colonnes. De tels tableaux sont capables de contenir vingt (5×4) nombres réels. Le troisième élément de la quatrième ligne de *chemin2* est identifié par *chemin2[4, 3]*.

Notons que le type *matrice*, un vecteur de vecteur, peut également se définir comme suit :

```
TYPE matrice = ARRAY[1..5] of ARRAY[1..4] OF REAL;
```

Exemple 4.13

Soit la déclaration suivante :

```
TYPE  matrice2 = ARRAY[4..7, 2..3] OF REAL;
VAR   mat1 : matrice2;
```

Elle permet de définir un tableau à deux dimensions *mat1* ayant quatre ($7 - 4 + 1$) lignes et deux ($3 - 2 + 1$) colonnes pouvant contenir huit (4×2) nombres réels.

Les tableaux à trois dimensions

Il existe également en PASCAL des tableaux à trois dimensions, considérés comme des vecteurs dont chacune des composantes ou cellules correspond à une matrice. Cette structure nécessite donc trois indices dont le premier se réfère à une cellule-matrice ou tranche, alors que les deux autres spécifient la ligne et la colonne dans la tranche en question.

Exemple 4.14

Voici trois déclarations équivalentes d'un même type *tab3*, un tableau à trois dimensions de trois tranches, trois lignes et trois colonnes en PASCAL :

```
TYPE
tab3 = ARRAY[1..3] OF ARRAY[2..4] OF ARRAY [1..3] OF INTEGER;
```

ou `tab3 = ARRAY[1..3] OF ARRAY[2..4, 1..3] OF INTEGER;`

ou `tab3 = ARRAY[1..3, 2..4, 1..3] OF INTEGER;`

4.4.2 Les chaînes de caractères

En PASCAL standard, le type *chaîne de caractères* n'existe pas. Cependant, dans certaines versions dérivées, le type STRING a été introduit pour désigner une chaîne de caractères. Dans ce contexte, STRING serait un mot clé désignant une chaîne de 255 caractères et serait équivalent à STRING[255]. Une chaîne de 20 caractères serait définie par STRING[20]. En Pascal, le type STRING désigne une chaîne pouvant contenir jusqu'à 255 caractères.

Exemple 4.15

a) Déclaration d'un type chaîne de caractères de longueur 255 :

```
TYPE  chaine1 = STRING;
```

b) Déclaration d'un type chaîne de caractères de longueur 20 :

```
TYPE  chaine2 = STRING[20];
```

En PASCAL standard, le type chaîne de caractères est implanté comme un tableau « compacté » (PACKED ARRAY) de caractères.

4.4.3 Les énumérations

Le type *énumération* est défini par une suite de valeurs ou d'identificateurs constituant un ensemble ordonné. Sa déclaration est conforme au diagramme syntaxique de la figure 4.8.

Exemple 4.16

a) Déclaration d'un type *énumération* identifié par *jour* :

```
TYPE  jour = (dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi);
```

b) Déclaration d'un type *énumération* identifié par *weekend* :

```
TYPE  weekend = (samedi, dimanche);
```

4.4.4 Les intervalles

Le type *intervalle*, comme son nom l'indique, permet de définir un ensemble ordonné de valeurs ou d'identificateurs se situant entre une borne inférieure et une borne supérieure, toutes deux des constantes en PASCAL. Sa déclaration est donnée par le diagramme syntaxique de la figure 4.9.

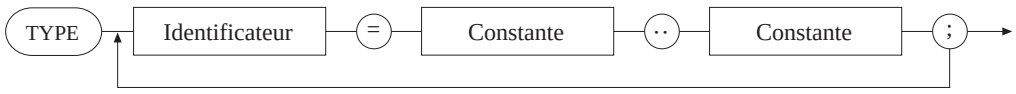

FIGURE 4.9
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION DE TYPE INTERVALLE.

Exemple 4.17

a) Déclaration d'un type *intervalle* identifié par *mois* :

```
TYPE  mois = 1..12;
```

b) Déclaration d'un type *intervalle* identifié par *majuscule* :

```
TYPE  majuscule = 'A'..'Z';
```

c) Déclaration d'un type *intervalle* identifié par *joursemaine* :

```
TYPE  jour = (dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi);
      joursemaine = lundi..vendredi;
```

Dans cet exemple, le type *mois* est défini à partir des bornes 1 et 12, alors que le type *majuscule* utilise les caractères « A » et « Z » comme bornes inférieure et supérieure respectivement. À ce sujet, il convient de noter deux choses :

– les bornes sont toutes des constantes bien définies dans le langage PASCAL, correspondant aux types standards INTEGER et CHAR;

– ces intervalles correspondent à des sous-ensembles finis et ordonnés d'ensembles prédéfinis en PASCAL.

Dans le cas du type *joursemaine* dont les bornes ne sont pas des constantes PASCAL, il est nécessaire de définir au préalable le type *jour*, constituant ainsi un ensemble ordonné dont *joursemaine* est un sous-ensemble. La définition préalable du type *jour* fait de *lundi* et *vendredi* des constantes pouvant servir de bornes au type *joursemaine* donnant la liste des cinq jours de la semaine.

On notera par ailleurs que les ensembles de base qui servent à définir le type intervalle sont des ensembles finis d'éléments non réels. C'est pourquoi les types énumération et intervalle sont considérés comme des types *scalaires*. De plus, le choix des bornes inférieure et supérieure doit respecter l'ordre des éléments dans l'ensemble de base. Ainsi, en se référant à la figure 4.9, la constante qui précède les deux points (..) doit être de rang inférieur à celle qui les suit.

4.4.5 Les ensembles

Dans le langage PASCAL, le type *ensemble* (SET) fait référence à un sous-ensemble d'un type de base préalablement défini et rentrant dans l'une des catégories suivantes : standard, énumération ou intervalle. La figure 4.10 présente le diagramme syntaxique de la déclaration d'un type ensemble, à partir d'un type de base préalablement défini.

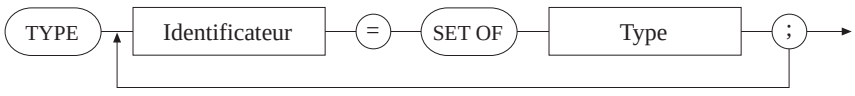

FIGURE 4.10
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION DE TYPE ENSEMBLE.

Exemple 4.18

Déclaration d'une variable *jourferie* de type ensemble à partir des types *joursemaine* et *jour* :

```
TYPE  jour = (dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi);
      joursemaine = SET OF jour;
VAR   jourferie : joursemaine;
```

Dans cet exemple, *jour* est le type de base, *joursemaine* le type ensemble et *jourferie* une variable de type *joursemaine*.

Les éléments d'un ensemble sont des valeurs ou identificateurs de l'ensemble de base généralement placés entre crochets. Un ensemble vide ne contient aucun élément à l'intérieur des crochets.

Tout comme en mathématique, il est possible d'effectuer en PASCAL un certain nombre d'opérations de base sur les ensembles. Décrivons ces opérations.

– *La réunion de deux ensembles* (+) : la réunion de deux ensembles A et B est noté A + B; c'est l'ensemble de tous les éléments appartenant soit à A soit à B.

– *L'intersection de deux ensembles* (*) : leur intersection est notée A * B et correspond à l'ensemble de tous les éléments appartenant à la fois à A et à B.

– *La différence de deux ensembles* (–) : la différence entre A et B est notée A – B et correspond à l'ensemble des éléments présents dans A et absents dans B.

– *L'égalité de deux ensembles* (=) : l'égalité de deux ensembles A et B se note A = B et exprime le fait que ces deux ensembles possèdent les mêmes éléments.

– *L'inégalité de deux ensembles* (< >) : l'inégalité (≠) de deux ensembles A et B se note en PASCAL A < > B et exprime le fait que ces deux ensembles ne contiennent pas exactement les mêmes éléments .

– *L'inclusion d'un ensemble dans un autre* (=<) : lorsque tous les éléments d'un ensemble A sont aussi éléments d'un ensemble B, on dit que A est inclus dans B, ce qui se traduit en PASCAL par la notation A =< B.

– *L'appartenance à un ensemble* (IN) : enfin, pour exprimer le fait qu'un élément *a* appartient à un ensemble A, en PASCAL, on écrit : *a* IN A.

4.4.6 Les enregistrements

Un *enregistrement* (RECORD) ou article est une structure de données composée d'un ensemble d'éléments qui ne sont pas nécessairement du même type, appelés *champs*, et qui sont manipulés comme un tout. Les champs peuvent être de type entier, réel, caractère, booléen, tableau, chaîne de caractères, intervalle, énumération, ensemble ou autre. Ils sont désignés par des noms ou identificateurs distincts; cependant, deux enregistrements distincts peuvent avoir des noms de champs identiques. La figure 4.11 présente le diagramme syntaxique d'une déclaration de type enregistrement.

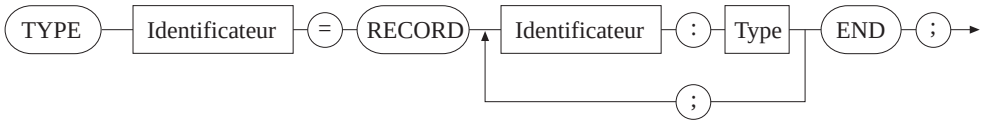

FIGURE 4.11
DIAGRAMME HIÉRARCHIQUE D'UNE DÉCLARATION DE TYPE ENREGISTREMENT.

Exemple 4.19

a) Déclaration d'un type enregistrement identifié par *personne* :

```
TYPE  personne = RECORD
                     nom, prenom : STRING[20];
                     age : INTEGER;
                 END;
```

b) Déclaration d'un type enregistrement identifié par *classe*, à partir d'un type enregistrement *personne* :

```
TYPE  personne = RECORD
                     nom, prenom : STRING[20];
                     age : INTEGER;
                 END;

      classe =   RECORD
                     nomprof : STRING[30];
                     eleve : ARRAY[1..40] OF personne;
                     annee : INTEGER;
                 END;
```

On accède à l'information contenue dans un enregistrement en juxtaposant un point entre le nom de l'enregistrement et le nom du champ auquel on veut accéder. Dans le cas du traitement de plusieurs champs d'un même enregistrement, on peut éviter la répétition du nom de cet enregistrement en utilisant l'instruction WITH dont le format est le suivant :

```
WITH  enregistrement DO
BEGIN
        expression(champ_1);
        .....
        expression(champ_n);
END;
```

Dans ce bloc, *enregistrement* désigne le nom de l'enregistrement dont les champs font l'objet de répétition, tandis que *expression(champ_i)* représente une expression ou une instruction mettant *champ_i* à contribution, *champ_i* étant un champ de *enregistrement*.

Exemple 4.20

Considérons les déclarations suivantes :

```
TYPE  personne = RECORD
                    nom, prenom : STRING[20];
                    age : INTEGER;
                 END;
      classe =   RECORD
                    nomprof : STRING[30];
                    eleve : ARRAY[1..40] OF personne;
                    annee : INTEGER;
                 END;
VAR   etudiant : personne;
      infoclasse : classe;
```

On peut écrire :

```
etudiant.nom = 'JACQUES'
etudiant.age = 35
infoclasse.nomprof = 'Rosemont'
infoclasse.annee = 1992
```

4.4.7 Les fichiers

On désigne par *fichier* (FILE) une structure de données regroupant une série d'éléments de même type, généralement maintenus en mémoire secondaire, sur disque ou sur bande magnétique. L'accès à ces éléments obéit à un certain nombre de principes qui permettent de distinguer les différentes sortes de fichiers : séquentiel, à accès direct, à accès aléatoire. La figure 4.12 présente le diagramme syntaxique d'une déclaration de type fichier.

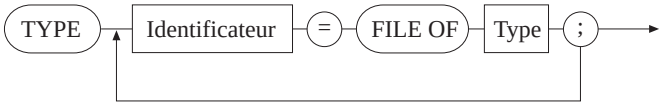

FIGURE 4.12
DIAGRAMME SYNTAXIQUE DE DÉCLARATION DE TYPE FICHIER.

Exemple 4.21

Déclaration d'un fichier identifié par *dossier*, à partir d'un type *personne* préalablement défini :

```
TYPE  personne = RECORD
                    nom, prenom : STRING[20];
                    age : INTEGER;
                 END;

VAR   dossier : FILE OF personne;
```

4.4.8 Les pointeurs

Toute information traitée par l'ordinateur réside à une certaine adresse dans la mémoire. Un *pointeur* fait référence à l'adresse de la mémoire où cette information est stockée. C'est donc une variable dynamique dont le contenu est lui-même une adresse. Un pointeur P qui ne renvoie à aucune adresse prend la valeur nulle (NIL); on dit dans ce cas que P ne pointe sur aucun élément.

La déclaration d'un type *pointeur*, en PASCAL, est donnée par le diagramme syntaxique de la figure 4.13.

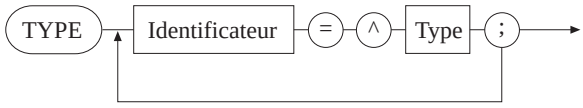

FIGURE 4.13
DIAGRAMME SYNTAXIQUE D'UNE DÉCLARATION DE TYPE POINTEUR.

Exemple 4.22

Déclaration d'un type pointeur identifié par *avance*, à partir d'un type *personne* préalablement défini :

```
TYPE  personne = RECORD
                    nom, prenom : STRING[20];
                    age : INTEGER;
                 END;

      avance = ^personne;
```

4.5 LES FONCTIONS STANDARDS

Il existe en PASCAL un certain nombre de fonctions qui n'ont pas besoin d'être définies par le programmeur et qui sont intrinsèques au langage : on les appelle les *fonctions standards* du PASCAL. En voici la liste :

ABS(X)	cette fonction retourne la valeur absolue du nombre X.
ARCTAN(X)	cette fonction retourne la valeur de l'arc tangente du nombre X.
CHR(X)	cette fonction retourne le caractère dont le numéro d'ordre est égal à l'entier X.
COS(X)	cette fonction retourne le cosinus du nombre X.
EXP(X)	cette fonction retourne l'exponentielle du nombre X.
LN(X)	cette fonction retourne le logarithme népérien du nombre X.
ORD(X)	cette fonction retourne le numéro d'ordre de X dans l'ensemble des valeurs du même type.
PRED(X)	cette fonction retourne la valeur précédente de X dans l'ensemble type associé.
ROUND(X)	cette fonction retourne l'entier le plus proche du réel X : ROUND(2.3) = 2, alors que ROUND(2.7) = 3.
SIN(X)	cette fonction retourne le sinus du nombre X.
SQR(X)	cette fonction retourne le carré du nombre X.
SQRT(X)	cette fonction retourne la racine carrée du nombre positif ou nul X.
SUCC(X)	cette fonction retourne la valeur suivante de l'élément non réel X dans l'ensemble type associé.
TRUNC(X)	cette fonction retourne la partie entière du nombre réel X.

Exemple 4.23

a) Considérons un nombre réel X de valeur – 324.6. Pour cette valeur de X, on a :

ABS(X) = 324.6
TRUNC(X) = – 324
ROUND(X) = – 325

b) Considérons un nombre entier X de valeur 4. Pour cette valeur de X, on a :

SQR(X) = 16
SQRT(X) = 2
CHR(X) = 'D' (en considerant l'ensemble des lettres majuscules)

c) Considérons le type *jour* de l'exemple 4.17. Pour X = mardi, on a :

ORD(X) = 3
SUCC(X) = mercredi
PRED(X) = lundi