CHAPITRE 3

LES ÉLÉMENTS DE BASE DU LANGAGE PASCAL

OBJECTIFS

- EXPLICITER LES PRINCIPALES CARACTÉRISTIQUES DU LANGAGE PASCAL.
- PRÉCISER LA STRUCTURE D'UN PROGRAMME ÉCRIT DANS CE LANGAGE.
- DÉFINIR LA SYNTAXE DU LANGAGE EN METTANT L'ACCENT SUR LES FORMALISMES DE REPRÉSENTATION.
- DISTINGUER LES INSTRUCTIONS QUI SONT EXÉCUTABLES DE CELLES QUI NE LE SONT PAS DANS UN PROGRAMME ÉCRIT EN PASCAL.

Conçu au début des années 70 par N. Wirth, *PASCAL* est un langage de programmation évolué qui s'est d'abord développé dans les universités, avant de connaître une expansion fulgurante dans les entreprises en général; de plus, il existe plusieurs implantations de ce langage. En effet, s'inspirant du PASCAL standard qui suit à la lettre les spécifications originelles de Wirth, l'Université de Californie à San Diego (UCSD) a développé la version qui porte son nom, le *PASCAL UCSD*. En fait, il s'agit essentiellement d'une extension par laquelle certains types de données ont été ajoutés au PASCAL standard. Avec l'avènement des micro-ordinateurs, on a vu apparaître certaines versions de ce langage qui leur sont spécifiquement dédiées. Les deux plus populaires d'entre elles demeurent le *PASCAL* et le *DELPHI* réalisés par la firme Borland. Dans ce chapitre, nous ne considérerons que les fondements du langage PASCAL pour DELPHI. Plus précisément, nous aborderons les caractéristiques du langage, la structure des programmes, la syntaxe du langage et les formalismes de représentation généralement utilisés pour le décrire.

3.1 LES CARACTÉRISTIQUES DU LANGAGE PASCAL

Le langage de programmation PASCAL possède quatre caractéristiques fondamentales : il est un langage à la fois évolué, universel, structuré et procédural. Dans les sections qui vont suivre, nous allons expliciter ces différentes caractéristiques.

3.1.1 Un langage évolué

En tant que langage *évolué* ou *de haut niveau*, PASCAL peut être implanté sur les ordinateurs les plus divers. En effet, contrairement aux langages d'assemblage, sa définition est indépendante de l'organisation propre à chaque type d'ordinateur. Cette différence dans l'organisation est plutôt prise en compte par le compilateur qui assure essentiellement la fonction de traduction des programmes écrits en PASCAL en des programmes objets correspondants, exécutables par l'ordinateur en question.

3.1.2 Un langage universel

PASCAL est aussi un langage *universel*, en ce sens qu'il est recommandé pour la programmation d'applications scientifiques, aussi bien que pour celles qui sont liées à la gestion. En fait, même s'il dérive du langage *ALGOL* qui est, de par sa nature, un langage de programmation scientifique, le langage PASCAL intègre certains types de données et de structures de fichiers non disponibles dans ALGOL. Ainsi muni de toute ces facilités, PASCAL apparaît comme un langage complet, qui peut être utilisé dans des domaines aussi variés que les calculs scientifiques, la gestion, la bureautique, le contrôle de processus, etc.

3.1.3 Un langage structuré

En tant que langage *structuré*, PASCAL donne lieu à des programmes organisés en blocs. Comme nous allons le voir dans la section suivante, ces programmes comprennent un en-tête, une partie déclarative et le programme à proprement parler, lequel est constitué d'une série d'instructions exécutables.

Un *bloc* n'est rien d'autre qu'une série d'instructions commençant par un énoncé de *début* et se terminant par un énoncé de *fin*. De plus, cette série d'instructions est traitée logiquement comme un tout. Cette structure est fondamentale en PASCAL.

Comme le montre la figure 3.1, plusieurs blocs peuvent être imbriqués pour constituer un seul bloc. Ainsi, le bloc B est constitué des deux blocs imbriqués B1 et B2 qui, à leur tour, résultent eux aussi d'une imbrication de blocs : le bloc B1 contient les blocs B11, B12 et B13, alors que le bloc B2 contient les blocs B21 et B22. Au dernier niveau, on retrouve les blocs B121 et B122 qui sont imbriqués à l'intérieur de B12, ainsi que les blocs B221 et B222 qui sont contenus dans le bloc B22.

Par ailleurs, comme l'illustre la figure 3.2, il est possible de représenter cette imbrication de blocs par un arbre qui en indique la structure hiérarchique :

– le bloc B se situe au niveau 1;
– les blocs B1 et B2, au niveau 2;
– les blocs B11, B12, B13, B21 et B22, au niveau 3;
– les blocs B121, B122, B221 et B222, au niveau 4.

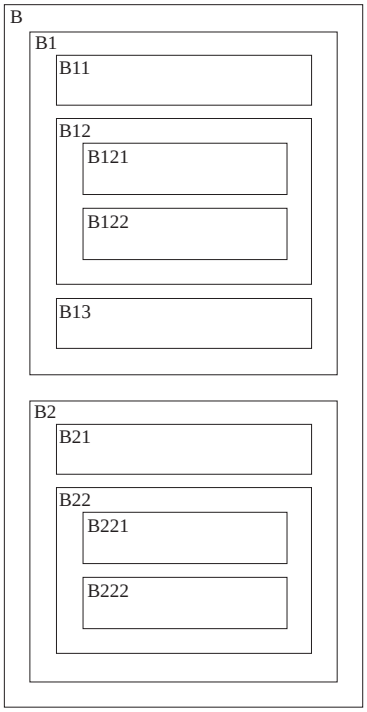


FIGURE 3.1

IMBRICATION DE BLOCS EN PASCAL.

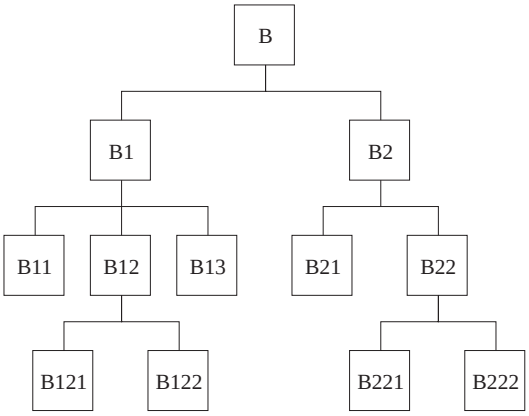
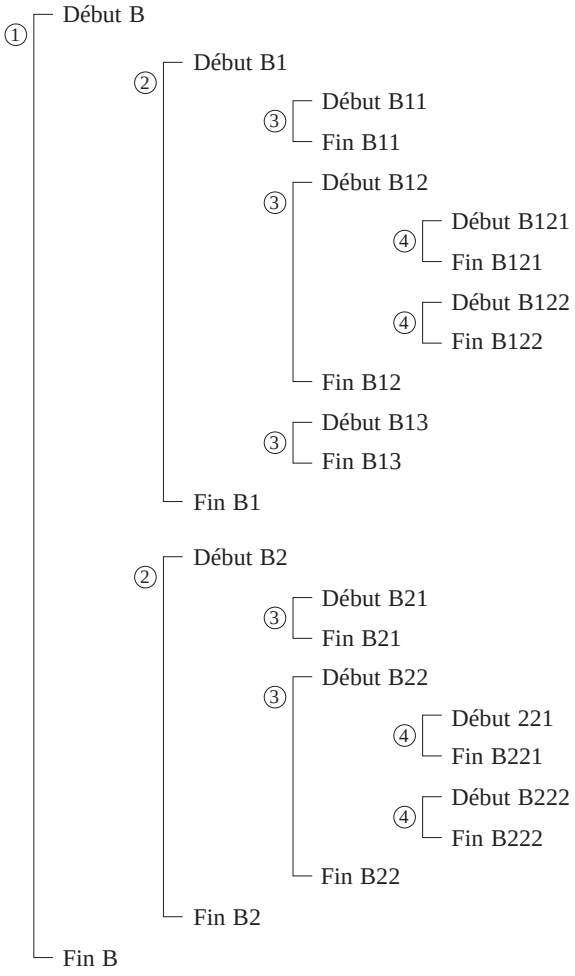


FIGURE 3.2

HIÉRARCHIE DE BLOCS IMBRIQUÉS.

Dans le programme, cette structure de blocs, pour des raisons de commodité, se traduit par des marges décalées ou *indentations* exprimant les différents niveaux d'imbrication. La figure 3.3, dans laquelle les chiffres encerclés indiquent les niveaux des blocs, montre le schéma d'indentation correspondant à la hiérarchie de la figure 3.2.



(les chiffres encerclés indiquent le niveau d'imbrication)

FIGURE 3.3
STRUCTURE INDENTÉE DES DIFFÉRENTS NIVEAUX D'IMBRICATION.

L'aspect structuré du langage PASCAL constitue l'une de ses caractéristiques les plus intéressantes. Aussi est-il particulièrement mis à contribution lorsque vient le moment, dans un contexte de programmation modulaire, d'implanter un algorithme dont les diverses étapes ont été clairement définies au préalable.

3.1.4 Un langage procédural

PASCAL est un langage avec lequel on peut décrire, de façon très naturelle, une démarche de résolution de problème. Cette démarche de résolution, une fois déterminée pour un problème donné, peut être mise sous la forme d'un algorithme. Ainsi, en utilisant le langage PASCAL, on peut écrire le programme correspondant qui pourra par la suite être compilé, c'est-à-dire traduit en programme objet, puis exécuté par l'ordinateur. C'est dans ce sens que l'on classe PASCAL parmi les langages procéduraux.

3.2 LA STRUCTURE D'UN PROGRAMME PASCAL

En général, un programme écrit en PASCAL comprend six sections qui doivent normalement apparaître dans l'ordre suivant :

- la section de déclaration des *étiquettes*;
- la section de déclaration des *constantes*;
- la section de déclaration des *types*;
- la section de déclaration des *variables*;
- la section de déclaration des *procédures* et des *fonctions*;
- la section des *instructions exécutables*.

Il convient de mentionner qu'il n'est pas obligatoire qu'un programme PASCAL comprenne les six sections. En effet, un programme qui ne traite ni d'étiquettes ni de constantes sera amputé des deux premières sections relatives à la déclaration des étiquettes et des constantes. De même, si le programme ne contient aucune procédure ni fonction, la section de déclaration des procédures et des fonctions sera omise. La seule section dont on ne peut absolument se passer est celle des instructions exécutables. Par *instruction exécutable*, on entend toute instruction utilisée pour décrire une action que doit accomplir l'ordinateur dans un processus de résolution de problème.

Par ailleurs, les procédures et les fonctions constituent en elles-mêmes des programmes. À ce titre, elles doivent respecter la structure d'un programme PASCAL. En conséquence, toute fonction ou procédure peut, elle aussi, contenir les six sections, le cas échéant.

Certaines implantations de PASCAL ne respectent pas rigoureusement l'ordre d'apparition des sections. En effet, on trouve sur le marché des compilateurs PASCAL qui acceptent des programmes dans lesquels l'ordre des sections est interverti. Ce n'est toutefois pas une raison, pour le programmeur rigoureux et soucieux des principes, d'écrire des programmes qui ne se conforment pas à la règle générale.

Du point de vue structurel, un programme PASCAL comprend trois parties :

– un *en-tête,* qui indique le nom du programme, ainsi que les fichiers d'entrée et de sortie sur lesquels le programme interagit;

– une *partie déclarative,* réservée, comme son nom l'indique, à la déclaration des étiquettes, des constantes, des types, des variables, des procédures et des fonctions, s'il y a lieu;

– un *corps,* qui regroupe essentiellement l'ensemble des instructions exécutables du programme, en rapport avec l'algorithme de résolution du problème envisagé.

La figure 3.4 montre un programme PASCAL qui permet d'affecter des notes littérales (A, B, C, D ou E) à des étudiants ayant obtenu une note numérique sur 100. Dans la partie en-tête, on trouve le nom du programme *Calcul*. La partie déclarative est constituée d'une seule section, la déclaration de variables. Quant au corps du programme, il débute au premier BEGIN et prend fin avec le dernier END suivi d'un point (.).

3.3 LA SYNTAXE DU LANGAGE PASCAL

On désigne par *syntaxe* d'un langage l'ensemble des règles qui régissent la composition des phrases de ce langage. Dans le langage PASCAL, les phrases sont construites à partir d'un certain alphabet permettant de former des mots et intègrent éventuellement un ou plusieurs opérateurs. De plus, elles donnent lieu à des instructions pouvant être exécutables ou non.

```
Ⓐ PROGRAM Calcul;
  USES FORMS;

Ⓑ VAR
      Notenum : Real;

Ⓒ BEGIN
      WRITELN ('Entrer la note numerique de l''etudiant :');
      READ (Notenum);
      IF (Notenum >= 0) and (Notenum <= 100) then
        BEGIN
          IF (Notenum >= 90) then
              WRITELN ('Note de l''etudiant : A')
          ELSE
              IF (Notenum >= 80) then
                  WRITELN ('Note de l''etudiant : B')
              ELSE
                  IF (Notenum >= 70) then
                      WRITELN ('Note de l''etudiant : C')
                  ELSE
                      IF (Notenum >= 60) then
                          WRITELN ('Note de l''etudiant : D')
                      ELSE
                          WRITELN ('Note de l''etudiant : E')
        END
  END.
```

Ⓐ : en-tête
Ⓑ : partie déclarative
Ⓒ : corps

FIGURE 3.4
EXEMPLE DE PROGRAMME PASCAL.

3.3.1 L'alphabet

L'alphabet du langage PASCAL comprend les éléments suivants :

– les vingt-six lettres majuscules (A à Z);
– les vingt-six lettres minuscules (*a* à *z*);
– les dix chiffres décimaux allant de 0 à 9;
– le symbole d'affectation (:=);
– la flèche verticale (^) ou symbole de pointeur;
– les séparateurs tels que les parenthèses ( ), les crochets [ ] et les accolades { };
– le caractère « blanc » (ou espace);
– les signes de ponctuation tels que la virgule (,), le point (.), le point-virgule (;), les deux-points (:) et l'apostrophe (').

Ces caractères ou symboles servent à définir les mots du langage. Ces derniers sont des suites de caractères de l'alphabet, chacune étant encadrée par le caractère « blanc » ou un signe de ponctuation. Leur construction obéit à un certain nombre de règles dites *lexicales*, sans lesquelles il serait impossible de dire si une suite de caractères d'un alphabet donné constitue ou non un mot du langage.

3.3.2 Les opérateurs

On appelle *opérateur* un mot ou un symbole qui prend une signification précise dans un langage de programmation donné et qui revêt un caractère opératoire. Dans le cas du langage PASCAL, on distingue trois types d'opérateurs : les opérateurs arithmétiques, les opérateurs relationnels et les opérateurs logiques.

Par *opérateur arithmétique*, nous entendons les symboles :

– d'addition (+),
– de soustraction (–),
– de multiplication (*),
– de division réelle (/),
– de division entière (DIV),
– de reste d'une division entière (MOD).

Notons au passage qu'en PASCAL il existe deux types de division : la *division entière*, dans laquelle dividende, diviseur et quotient sont des entiers, et la *division réelle*, où les trois éléments sont des nombres réels. Quant au dernier opérateur MOD, il permet d'obtenir le reste de la division d'un nombre entier par un autre.

Exemple 3.1

```
5 DIV 2 = 2
5.0 / 2.0 = 2.5
5 MOD 2 = 1
```

Les *opérateurs relationnels* ou *de comparaison* sont au nombre de sept et correspondent aux symboles suivants :

– inférieur (<),
– inférieur ou égal (<=),
– supérieur (>),
– supérieur ou égal (>=),
– égal (=),
– différent (<>),
– membre de (IN).

De tels opérateurs servent à former des expressions logiques du langage PASCAL, c'est-à-dire des expressions dans lesquelles on établit des comparaisons entre des objets de même nature et dont l'évaluation entraîne deux valeurs possibles : *vrai* ou *faux*.

Exemple 3.2

Considérons quatre nombres entiers A, B, C et D ayant pour valeur :

```
A := 40
B := 50
C := 50
D := 10
```

Les expressions suivantes sont toutes vraies :

A < B
A > D
B <= C
(A + D) = B

Les expressions suivantes sont toutes fausses :

A > B
A = C
B <= D
(A + D) < B

Quant aux *opérateurs logiques*, ils sont définis dans le calcul propositionnel et sont au nombre de trois. Ce sont :

– la négation (NOT),
– la conjonction (AND),
– la disjonction (OR).

La *négation* est une opération « unaire », dans le sens qu'elle met en jeu une seule expression logique ou proposition. Elle permet d'inverser la valeur de vérité associée à une expression donnée. Ainsi, si l'expression :

[(A + D) < B]

est fausse, sa négation :

NOT [(A + D) < B]

est vraie.

La *conjonction* est une opération binaire, c'est-à-dire qui porte sur deux expressions logiques. En effet, on peut combiner deux expressions logiques P et Q par l'opérateur AND, conformément à la table de vérité de la figure 3.5.

P	Q	P AND Q
V	V	V
V	F	F
F	V	F
F	F	F

FIGURE 3.5
TABLE DE VÉRITÉ DE LA CONJONCTION.

Si on considère les données de l'exemple 3.2, l'expression :

$$[(A + D) = B] \text{ AND } [(A + D) < B]$$

est fausse puisque la première expression est vraie, alors que la seconde est fausse.

L'interprétation de la table de vérité de la conjonction, telle que cette table est présentée à la figure 3.5, permet d'énoncer ce qui suit :

> *Le seul cas où la conjonction de deux expressions logiques P et Q est vraie est celui où les deux expressions sont simultanément vraies.*

Par ailleurs, la *disjonction* est également une opération binaire qui permet de combiner deux expressions logiques. Comme il est indiqué à la table de vérité de la figure 3.6, une proposition (P OR Q) obtenue par disjonction des propositions P et Q est vraie si et seulement si l'une ou l'autre de ces deux propositions est vraie. Autrement dit, pour qu'une disjonction de propositions soit fausse, il faut et il suffit que toutes les propositions qui la composent soient indistinctement fausses.

P	Q	P OR Q
V	V	V
V	F	V
F	V	V
F	F	F

FIGURE 3.6
TABLE DE VÉRITÉ DE LA DISJONCTION.

3.3.3 Les mots du langage

Dans tout langage de programmation, il existe des règles lexicales qui permettent de construire des mots à partir de l'alphabet du langage. Dans ce contexte, il convient de préciser que toutes les suites de caractères d'un alphabet ne donnent pas lieu nécessairement à un mot. Dans le cas du langage PASCAL, on distingue quatre types de mots :

– les nombres,
– les identificateurs,
– les mots clés,
– les chaînes de caractères.

En ce qui a trait aux *nombres*, on en distingue deux types : les *nombres entiers* et les *nombres réels*. Notons que cette distinction est fondée sur des considérations liées à la représentation interne des nombres en mémoire d'ordinateur, aspect qui sera développé plus loin au chapitre 4. Pour l'instant, retenons que la notion d'entier fait référence à l'ensemble des entiers relatifs défini en algèbre; ces derniers peuvent être positifs, négatifs ou nuls. Quant aux réels, ils regroupent aussi bien les nombres fractionnaires que les nombres décimaux. À titre d'exemple, 3 et – 4 sont des entiers, alors que -1.32×10^{-15} (ou – 1.32E–15) est un réel.

On désigne par *identificateur* une suite de caractères alphanumériques d'une certaine longueur, commençant obligatoirement par une lettre. Par exemple, *Compteur* est un identificateur valide en PASCAL, alors que *2bon* n'en est pas un puisqu'il ne commence pas par une lettre.

En PASCAL standard, la longueur maximale d'un identificateur est de huit caractères. Au-delà de ce nombre, il devient impossible de différencier deux identificateurs. Ainsi, même si *Compteurdemot* et *Compteurdeson* sont deux identificateurs valides en PASCAL, ils ne sont pas recommandés aux programmeurs. En effet, étant donné que les dix premiers caractères sont identiques, ces deux identificateurs se révèlent indifférenciables pour un compilateur qui ne reconnaît que les huit premiers caractères.

On appelle *mots clés* ou *mots réservés* d'un langage de programmation tous ceux qui ont une signification précise dans ce langage. Notons que les mots clés ne doivent absolument pas être utilisés comme des identificateurs. Le tableau 3.1 donne une liste des principaux mots clés du PASCAL standard.

TABLEAU 3.1

LES MOTS CLÉS DU PASCAL STANDARD

Mots clés (PASCAL)	**Termes français correspondants**
AND	et
ARRAY	tableau
BEGIN	début
CASE OF	cas parmi
CONST	const(ante)
DIV	div(iser par)
DOWNTO	descendant
END	fin
FILE	fichier
FOR... TO	pour... à
FUNCTION	fonction
GOTO	aller à
IF... THEN... ELSE	si... alors... sinon
IN	dans
INPUT	entrée
LABEL	étiquette
MOD	mod(ulo)
NIL	nul
NOT	non
OF	de
OR	ou
OUTPUT	sortie
PACKED	compacte
PROCEDURE	procédure
PROGRAM	programme
RECORD	enregistrement
REPEAT... UNTIL	répéter... jusqu'à
SET	ensemble
TYPE	type
VAR	var(iable)
WHILE... DO	tant que... faire
WITH	avec

On désigne par *chaîne de caractères* une suite de caractères de l'alphabet du langage, laquelle suite est encadrée par des apostrophes. Par exemple,

'LE LANGAGE PASCAL EST UNIVERSEL'

est une chaîne de caractères de longueur 31.

Il convient de mentionner que la chaîne de caractères peut contenir n'importe quel caractères de l'alphabet. Toutefois, si le caractère apostrophe (') y figure, il doit être répété au même endroit dans le texte, comme c'est le cas dans la chaîne suivante :

'LE LANGAGE PASCAL, C''EST UN LANGAGE UNIVERSEL'

3.4 LES FORMALISMES DE REPRÉSENTATION SYNTAXIQUE

En plus de posséder un ensemble de règles lexicales qui précisent le mécanisme de formation des mots, le langage PASCAL possède également une *syntaxe* qui regroupe les règles de composition des phrases. Il existe deux manières pour décrire la syntaxe du langage PASCAL : les *formes normales de Backus-Naur* (BNF) et les *diagrammes syntaxiques* de Conway.

3.4.1 Les formes normales de Backus-Naur

On appelle *forme normale de Backus-Naur* (BNF) un système de notation qui permet de définir aussi bien les mots que les phrases (ou instructions) du langage PASCAL. Ce système fait appel à des éléments dits *terminaux* et à d'autres dits *non terminaux* du langage. Par éléments terminaux, on entend des éléments déjà définis dans le langage comme les caractères de l'alphabet. Ce système intègre également un certain nombre de symboles tels que :

- le symbole d'affectation (:=), qui sert à introduire une définition;
- la barre verticale (/), qui sert à indiquer un choix entre plusieurs éléments;
- les signes (< >), qui servent à encadrer les éléments non terminaux utilisés dans une définition;
- les accolades ({ }), qui servent à encadrer les éléments qui font l'objet d'une répétition;
- l'astérisque (*) placé en exposant, qui sert à indiquer une répétition.

En notation BNF, un chiffre peut être défini de la manière suivante :

<chiffre> := 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9

Autrement dit, un chiffre c'est n'importe quel caractère numérique. Notons au passage que ces caractères appartiennent tous à l'alphabet du langage. En conséquence, *chiffre* est défini comme un des éléments terminaux.

La notion de chiffre étant définie, il est maintenant possible d'utiliser celui-ci pour définir un nombre entier :

<nombre entier> := <chiffre> { <chiffre> }*

Donc, un nombre entier n'est rien d'autre qu'un chiffre suivi éventuellement d'un ou de plusieurs autres chiffres.

On peut également définir un identificateur par la notation suivante :

<identificateur> := <lettre> { <chiffre / lettre> }*

Ainsi, par identificateur, on entend une suite de caractères alphanumériques dont le premier caractère est obligatoirement une lettre. Et c'est exactement ce que traduit cette notation : le premier caractère est une lettre, alors que les caractères suivants peuvent tout aussi bien être des chiffres que des lettres. Il ne faut toutefois pas oublier que le nombre maximum de caractères qui définissent un identificateur, pour des fins de différenciation, est déterminé par l'implantation de PASCAL concrétisée par le compilateur.

3.4.2 Les diagrammes syntaxiques

On appelle *diagramme syntaxique* une représentation graphique qui décrit le mécanisme de construction d'un mot ou d'une phrase d'un langage. Ainsi, si on considère les chiffres et les lettres comme des éléments terminaux du langage PASCAL, on peut définir un nombre entier par le diagramme syntaxique de la figure 3.7, et un identificateur par celui de la figure 3.8.

FIGURE 3.7
DIAGRAMME SYNTAXIQUE D'UN NOMBRE ENTIER.

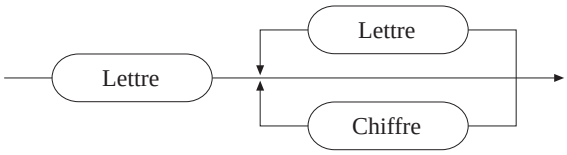


FIGURE 3.8
DIAGRAMME SYNTAXIQUE D'UN IDENTIFICATEUR.

De même, la figure 3.9 présente le diagramme syntaxique d'une constante numérique. D'après ce diagramme, une constante numérique peut prendre l'une des quatre formes suivantes :

(forme 1) un ou plusieurs chiffres représentant une partie entière;

(forme 2) un ou plusieurs chiffres représentant une partie entière, suivi d'un symbole d'exposant (E), suivi d'un signe (+ ou –), suivi d'un ou de plusieurs chiffres représentant l'exposant;

(forme 3) un ou plusieurs chiffres représentant une partie entière, suivi du point décimal, suivi d'un ou de plusieurs chiffres constituant la partie décimale;

(forme 4) un ou plusieurs chiffres représentant une partie entière, suivi du point décimal, suivi d'un ou de plusieurs chiffres constituant la partie décimale, suivi d'un symbole d'exposant (E), suivi d'un signe (+ ou –), suivi d'un ou de plusieurs chiffres représentant l'exposant.

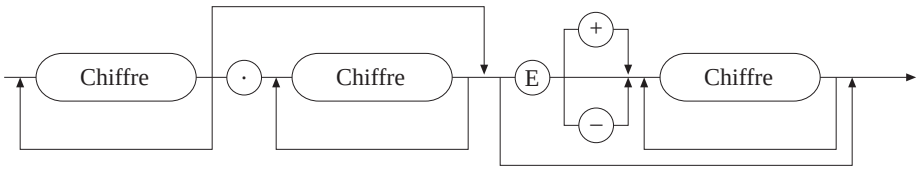


FIGURE 3.9
DIAGRAMME SYNTAXIQUE D'UNE CONSTANTE NUMÉRIQUE.

Exemple 3.3

234 est une constante numérique selon la forme 1;
234E–12 (ou 234 x 10^{-12}) est une constante numérique selon la forme 2;
234.45 est une constante numérique selon la forme 3;
234.45E–12 (ou 234.45 x 10^{-12}) est une constante numérique selon la forme 4.

Par ailleurs, la figure 3.10 donne le diagramme syntaxique d'un programme PASCAL. Après analyse, on peut caractérisé ce dernier de la manière suivante :

– il commence toujours par le mot clé PROGRAM;

– le mot clé PROGRAM est toujours suivi d'un identificateur (qui est le nom du programme);

– le nom du programme peut être suivi ou non d'un ou de plusieurs identificateurs (nom de fichier) séparés deux à deux par une virgule. La liste de ces identificateurs, s'il y a lieu, doit être placée entre parenthèses;

– un point-virgule indique la fin de l'en-tête du programme et annonce le bloc de programme;

– un point indique la fin du programme.

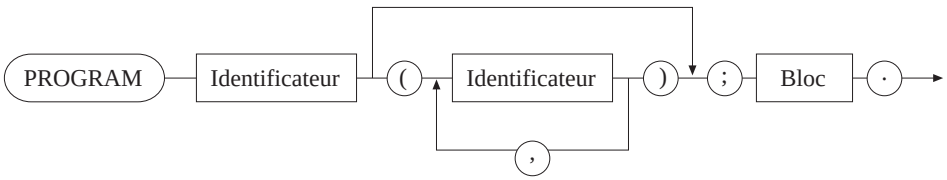


FIGURE 3.10
DIAGRAMME SYNTAXIQUE D'UN PROGRAMME.

Revenons à la figure 3.4 pour examiner la partie en-tête d'un programme PASCAL : le nom de ce programme est *Calcul* et les fichiers traités sont INPUT et OUTPUT, deux mots clés ou identificateurs standards du langage. On notera la présence du point-virgule à la fin de la première ligne du programme représentant l'en-tête (A). Quant au

bloc de programme, il regroupe la partie déclarative et le corps du programme, soit les parties B et C de la figure 3.4.

De manière plus formelle, on peut définir un bloc de programme par le diagramme syntaxique de la figure 3.11. On y distingue les différentes sections possibles de la partie déclarative, suivies du corps du programme commençant par un indicateur de DEBUT et se terminant par un indicateur de FIN.

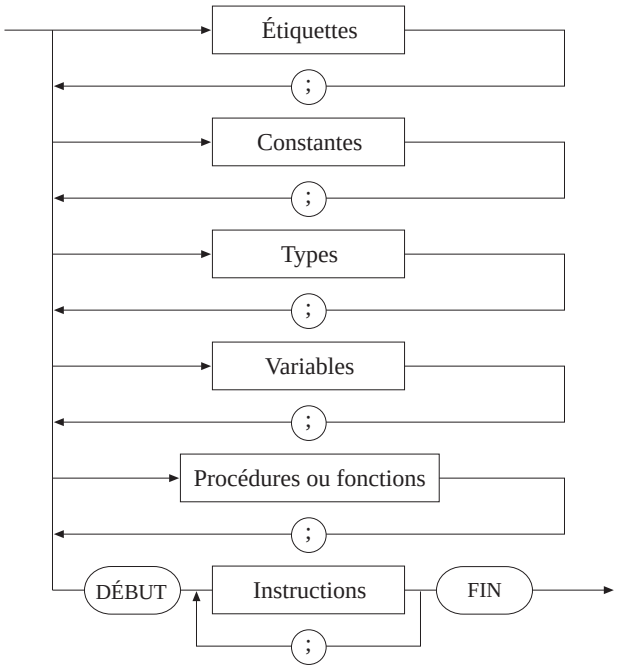


FIGURE 3.11
DIAGRAMME SYNTAXIQUE D'UN BLOC DE PROGRAMME.

3.5 LES INSTRUCTIONS EXÉCUTABLES

Dans tout programme il existe des instructions qui sont exécutables et d'autres qui ne le sont pas. Les instructions exécutables correspondent à des actions que le programmeur demande à l'ordinateur d'exécuter en vue de la résolution d'un problème. Dans

ce sens, elles sont étroitement liées à l'algorithme qui a été préalablement conçu pour résoudre le problème en question. C'est le cas de la plupart des instructions qui composent le corps du programme.

Les instructions non exécutables ou *pseudo-instructions* peuvent être considérées comme des directives qui s'adressent au compilateur de langage et ne concernent pas vraiment la démarche de résolution du problème. C'est le cas des instructions qui constituent la partie déclarative des programmes PASCAL. De telles instructions servent, pour la plupart, à réserver de l'espace en mémoire en vue du traitement ultérieur des instructions exécutables du programme.