Quelques éléments de syntaxe du langage Pascal

Éric Wegrzynowski

5 septembre 2005

1 Introduction

Ce document a pour vocation de rassembler l'essentiel de la syntaxe à connaître pour écrire un programme en Pascal. Ce document n'est certainement pas complet (), et ne donne aucune méthode pour concevoir un programme (ce n'est pas parce que vous connaissez l'orthographe et la grammaire que vous savez exprimer vos idées).

Dans tout ce document, les présentations d'éléments de syntaxe sont sur fond de

```
couleur abricot,
```

et les exemples d'éléments de syntaxe sur fond de

```
couleur jaune.
```

2 Structure d'un programme

Le listing 1 montre la structure d'un programme en Pascal.

Listing 1 – Structure d'un programme

```
program <idProgramme> ;

  (* déclarations *)

begin

  (* séquence d'instructions*)

end.
```

Dans un programme en Pascal, on trouve
- le *mot réservé* **program** suivi d'un *identificateur* le nommant, suivi d'un point-virgule,
- les différentes déclarations dont il a besoin (types, constantes, variables, fonctions et procédures),
- le bloc d'instructions le définissant encadré par les *mots réservés* **begin** et **end**,
- et finalement un point.

Dans le listing ci-dessus, les deux lignes entre *(∗* et *∗)* sont des *commentaires* destinés à visualiser l'emplacement des déclarations et instructions du programme.

Un exemple complet de programme est donné à la fin de ce document.

3 Commentaires

Les commentaires sont des éléments du texte d'un programme qui sont ignorés par le compilateur. Ils servent à apporter des renseignements sur le programme.

En Pascal, il y a trois façons d'écrire des commentaires

1. commentaires sur une seule ligne : ils débutent par //

```
// commentaire sur une seule ligne
```

2. commentaires sur plusieurs lignes : encadrés par { et }

```
{ commentaire sur
  plusieurs
  lignes }
```

3. commentaires sur plusieurs lignes : encadrés par (∗ et ∗)

```
(* commentaires
   sur plusieurs
   lignes *)
```

4 Identificateurs

Les *identificateurs* sont des noms permettant de faire référence à des variables, des constantes, des procédures, des fonctions. Les programmes possèdent aussi un identificateur.

En PASCAL un identificateur doit débuter par une lettre et peut être poursuivi par n'importe quelle combinaison de lettres, chiffres et blancs soulignés (_).

Le langage PASCAL ne fait aucune distinction entre lettre minuscule et lettre majuscule. Ainsi les identificateurs `Ma_Variable` et `ma_variable` sont considérés comme équivalents.

Certains identificateurs sont des *mots réservés* du langage, et ne peuvent donc pas être utilisés pour désigner des variables ou autres entités. Les mots réservés varient d'un compilateur à l'autre (FREE PASCAL, GnuPascal, DELPHI, ...). En voici une liste (non exhaustive) pour le PASCAL de FREE PASCAL.

and	**array**	**begin**	**case**	**const**	**div**	**do**
downto	**else**	**end**	**exit**	**false**	**file**	**for**
function	**if**	**implementation**	**in**	**interface**	**mod**	**new**
not	**of**	**or**	**procedure**	**program**	**record**	**repeat**
then	**to**	**true**	**type**	**until**	**uses**	**var**
while	**with**					

TAB. 1 – Mots réservés du PASCAL

5 Constantes

Les *constantes* sont déclarées dans la partie de déclarations d'un programme, d'une procédure ou d'une fonction, après le mot réservé **const**. On utilise le symbole = dans les déclarations de constantes.

Listing 2 – Déclarations de constantes

```
const
  <id> = <valeur>;
```

Il est possible de déclarer plusieurs constantes,

```
const
  LARG = 640;
  HAUT = 480;
  NB_PIX = LARG*HAUT;
```

6 Variables

Les *variables* sont déclarées dans la partie de déclarations d'un programme, d'une procédure ou d'une fonction, après le mot réservé **var**.

Listing 3 – Déclarations de variables

```
var
  <id> : <type>;
```

Il est possible de déclarer plusieurs variables

```
var
  n : CARDINAL;
  x : REAL;
  s, t : STRING;
```

7 Instruction

Une *instruction simple* peut être
- une affectation,
- un appel de procédure,
- une structure de contrôle : conditionnelle ou itérative.

Une *séquence d'instructions* est une suite finie d'instructions séparées par des points-virgule (;).

Listing 4 – Séquence d'instructions

```
<instruction 1> ;
<instruction 2> ;
...
```

Un *bloc d'instructions* est une séquence d'instructions encadrées par les deux mots réservés **begin** et **end**.

Listing 5 – Bloc d'instructions

```
begin

  (* séquence d'instructions *)

end
```

Le but d'une instruction est de modifier l'environnement d'exécution du programme.

8 Affectation

L'*affectation* est une instruction qui permet de modifier la valeur d'une variable. En Pascal, c'est le symbole := qui est utilisé pour écrire une affectation. À gauche de ce symbole figure l'identificateur d'une variable, et à droite une expression.

Listing 6 – L'affectation

```
  <id> := <expression>;
```

9 Instruction conditionnelle

Listing 7 – Instruction conditionnelle simple

```
if <condition> then
   (* bloc d'instructions *)
```

Listing 8 – Instruction conditionnelle complète

```
if <condition> then
  (* bloc d'instructions *)
else
  (* bloc d'instructions *)
```

10 Instruction itérative

Les instructions itératives sont aussi parfois appelées *boucles*.

10.1 Boucle Tant que

Listing 9 – Boucle **Tant que**

```
while <condition> do
  (* bloc d'instructions *)
```

10.2 Boucle Pour

Listing 10 – Boucle **Pour**

```
for <indice> := <debut> to <fin>  do
  (* bloc d'instructions *)
```

11 Fonctions

Listing 11 – Structure d'une fonction

```
function <idFonction> (<paramètres>) : <type_résultat>;

  (* déclarations *)

begin

  (* séquence d'instructions*)

end;
```

12 Procédures

Listing 12 – Structure d'une procédure

```
procedure <idProcedure> (<paramètres>);

  (* déclarations *)

begin
```

```
  (* séquence d'instructions*)

end;
```

Une procédure a une structure syntaxique proche de celle d'un programme. Notez cependant qu'elle se termine par

13 Un exemple de programme

Listing 13 – Un exemple de programme

```
1  // auteur : EW
2  // date : août 2005
3  // objet : calcul du pgcd de deux entiers
4  program calcul_pgcd ;
5
6
7  // pgcd(a,b) = plus grand commun diviseur des entiers
       a et b
8  // calcul selon l'algorithme d'Euclide
9  function pgcd(a,b : CARDINAL) : CARDINAL;
10 var
11    m,n,r : CARDINAL;
12 begin
13    m := a;
14    n := b;
15    while n<>0 do
16       {propriété invariante : pgcd(a,b)=pgcd(m,n)}
17    begin
18       r := m mod n;
19       m := n;
20       n := r;
21    end {while};
22    pgcd := m;
23 end {pgcd};
24
25 var
26    a, b : CARDINAL ;
27
28 begin
29    write('a = ');
30    readln(a);
31    write('b = ');
32    readln(b);
33
34    writeln('pgcd(',a,',',b,') = ', pgcd(a,b));
35 end.
```

- ligne 4 : le nom `calcul_pgcd` est attribué au programme
- lignes 5-27 : déclarations du programme
  - de la ligne 9 à 23 : déclaration d'une fonction
  - lignes 25-26 : déclaration de deux variables
- lignes 28-35 : le bloc d'instructions du programme.