Présentation générale du langage PASCAL

Eléments de base

Un programme est une suite d'instructions, certaines étant des mots clés.
Ce programme affiche la chaîne de caractères < Bonjour > à l'écran :

```
PROGRAM bonjour;
BEGIN
writeln ('Bonjour');
END.
```

Le compilateur est un logiciel qui lit (analyse) un programme et le traduit en code machine, directement exécutable par le processeur de l'ordinateur.

1 Vocabulaire

C'est l'ensemble

- des identificateurs (nom des variables et constantes),
- mots clés (BEGIN, END, WHILE, ...),
- séparateurs ('.', ';', ',',...),
- opérateurs ('+', '-', '/',...),
- chaînes de caractères.

2 Objets

Les objets sont répartis en deux classes : les constantes et les variables. Les objets peuvent être de type : entier (INTEGER) , réel (REAL), caractère (CHAR), chaine de caractères (STRING), booléen (BOOLEAN)

Les constantes sont définies par :
CONST *Identificateur = valeur*

Les variables sont déclarées par *:*
VAR *Identificateur1, Identificateur2,.... : Type*

Une variable est une zone dans la mémoire vive de l'ordinateur, dotée d'un nom et d'un type. Le nom de la variable permet d'accéder au contenu de la zone mémoire ; le type spécifie la nature de ce qui peut être stocké dans la zone mémoire (entier, réel, caractère, etc).

Identificateur

Sert à donner un nom à un objet.
Syntaxe

- On appelle lettre un caractère de 'a'..'z' ou 'A'..'Z' ou '_'.
- On appelle numérique un caractère de '0'..'9'.
- Un identificateur Pascal est une suite de lettres ou de numérique accolés, commençant par une lettre.

Exemples
x, y1, jour, mois, annee, NbCouleurs, longueur_ligne.

Remarques

- Il n'y a pas de différence entre minuscules et majuscules.
- On n'a pas le droit de mettre d'accents, ni de caractères de ponctuation.
- Un identificateur doit être différent des mots clés (begin, write, real, . . .)

On se sert des identificateurs pour : le nom du programme, les noms de variables, les noms de constantes, les noms de types.

3 Expressions

Les expressions peuvent être arithmétiques (+, -, *, /, Div, ...), logiques (Not, And, OR,...) ou relationnelles (<, <=, >, >=, <>).

4 Instructions

Une instruction PASCAL peut être simple ou composée.
Dans ce qui suit, Instruction désigne une instruction simple ou composée, Variable une variable et Expression une expression.

Instructions simples

Donnons dans un premier temps les instructions simples suivantes :
Affectation : *Variable := Expression*
Boucle While : *WHILE Expression DO instruction*
Conditionnelle : *IF Cond THEN Instruction*
Alternative : *IF Cond THEN Instruction ELSE Instruction*
Lecture : *READ(Variable)*
Ecriture : *WRITE (Expression)*

Instruction composée

Une instruction composée a la forme suivante :
BEGIN Instruction1; Instruction2; .....END

Exemples

Affectation : *A := (A + B) / C*
Boucle While : *WHILE X > 5 DO X := X + 5*

```
WHILE X+Y >= 1000 DO
BEGIN
X := X + 10 ;
Y := Y - 2;
END
```

Conditionnelle : `IF A > B THEN READ(X, Y)`

Alternative : `IF X < Y THEN WRITE(X) ELSE WRITE(Y)`

Lecture : `READ(X, Y, Z)`

Ecriture : `WRITE ( ' Résultats : ', Res)`

Commentaires dans un programme

On place un *{commentaire}* dans un programme au-dessus ou à côté d'une instruction.
Le commentaire n'est pas pris en compte à la compilation. Il sert à rendre le programme plus clair à la lecture, à noter des remarques, etc :

```
{ Kamel École Prépa, Bejaia - 21/01/2019}
PROGRAM bonjour;
BEGIN
{ Affiche Bonjour à l'écran }
writeln ('Bonjour');
END.
```

5 Structure d'un programme

Un programme PASCAL a la forme suivante :

```
PROGRAM Nom;
définition des constantes ( Const .... )
définition des variables (Var .... )
BEGIN
Corps du programme
END.
```

La structure d'un programme est composé en 3 parties : le nom du programme, la partie déclarations, et le corps du programme, qui est une suite d'instructions.
La partie déclaration crée les variables (les boîtes) ; leur contenu est indéterminé (on met un ' ?' dans chaque boîte). La taille de la zone mémoire de chaque variable est adaptée au type (par exemple 1 octet pour un caractère, 4 octets pour un entier, etc).

6 Exemples

Reprenons nos exemples et donnons les programmes PASCAL correspondants.

Equation du second degré

```
PROGRAM Equation;
VAR A, B, C, Delta : REAL;
BEGIN
READ(A B, C);
Delta := B *B - 4 A*C;
IF Delta > 0 THEN
      BEGIN
      WRITE( 'la première racine est', - B+RAC(Ardoise)/4A*C);

      WRITE( 'la deuxième racine est', -B- RAC(Ardoise)/4 A*C)
      END
ELSE
      IF Delta = 0 THEN
            WRITE( ' Une racine double', - B / 2*A )
      ELSE
            WRITE( ' Pas de racine réelle' )
END.
```

Racine cubique

```
PROGRAM Racine_cubique;
VAR A, X0, Mu, Ancien, Nouveau, Difference : REAL;
BEGIN
READ(A, X0, Mu);
Ancien := X0;
Nouveau := ( 2* Ancien + A/ (Ancien*Ancien) ) / 3 ;
Difference := Abs ( Nouveau - Ancien ) ;
WHILE Difference < Mu DO
      BEGIN
      Ancien := Nouveau;
      Nouveau := ( 2*Ancien + A/ Ancien*Ancien ) / 3;
      Difference := Abs ( Nouveau - Ancien )
      END;
WRITE (' Racine carrée de', A , " est ", Nouveau )
END.
```

Entrées/Sorties PASCAL

1 Lecture

Toute introduction de données se fait par l'ordre :

```
READ[LN]( V1, V2, ...Vn)
```

[ ] désigne une partie facultative.

Vi est une variable de type INTEGER, REAL ou CHAR.

Cette instruction provoque la lecture de n données à partir de l'écran. Pour les nombres, les données sont séparées par des blancs. Pour les caractères, la lecture se fait toujours caractère/caractère.

Si l'option LN est présente, il y a positionnement à la ligne suivante après la lecture des données.

2 Ecriture

Un restitution de résultats se fait par l'ordre

```
WRITE[LN] (P1, P2, ....., Pn)
```

P1, P2, ..., Pn sont des expressions suivies éventuellement du mode d'écriture.

On peut avoir les 3 formes :

- E
- E : E1
- R : E1 : E2

avec E, E1, E2, R des expressions.

E : expression de type INTEGER, CHAR, REAL, BOOLEAN

R : expression de type REAL

E1, E2 de type INTEGER indique une largeur de champ (E1 : nombre total de caractères de la valeur écrite, E2 : nombre de chiffres après le point décimal)

Première forme : E

Si E1 n'est pas spécifiée, une valeur (pour E1) par défaut est assurée dépendant du type c'est à dire :

```
CHAR : 1
INTEGER : 11
BOOLEAN : 8
REAL : 20
```

Ces valeurs peuvent changer selon la version PASCAL considérée.

Deuxième forme : E : E1

E1 désigne le nombre total de caractères à écrire. Si E1 ne suffit pas, le nombre ne sera pas tronqué sinon cadré à gauche par des blancs. Si E est réelle, le nombre est écrit en virgule flottante.

Troisième forme : R : E1 : E2

La partie fractionnaire de R doit comporter E2 caractères. Le nombre est écrit en format virgule fixe sans exposant.

Exemple

```
PROGRAM Exemple;
CONST Tab = 5;
VAR I, I1, I2 : INTEGER;
R1 : REAL;
B : BOOLEAN;
BEGIN
B := TRUE ;

I1 := -3 ;
R1 := 4.5;
I2 := 6 ;
WRITELN(' Exemple de sortie');
WRITELN; { saut de ligne }
WRITELN(' ':Tab, 'I1=',I1:I2,'R1=',R1:13);
WRITELN(I1:2, I1, 'B=', B : 3 );
WRITELN( R1:8, R1:4:1)
END.
```

Ce programme donne en sortie :
I1= -3R1= 4.500000E+00
-3-3B=TRUE
4.5E+00 4.5

3 Les fichiers TEXT

Au lieu de lire les données à partir de l'écran, ce qui peut être fastidieux lors de la mise au point des programmes, il est préférable et même très avantageux de lire les données à partir d'un fichier TEXT construit préalablement par un éditeur de texte.
La déclaration d'un tel fichier se fait comme suit :

```
VAR Fe : TEXT
```

où `Fe` désigne le nom logique du fichier.
Tout fichier déclaré de la sorte doit être lié à un fichier physique. Ce lien est établit grâce à l'instruction `ASSIGN` définie comme suit :

```
ASSIGN ( Nom logique, Nom physique)
```

De plus, le fichier doit être ouvert par l'opération

```
RESET(Nom logique)
```

Les opérations de lectures sont faites par :

```
READ[LN] ( Nom logique, V1, V2, ......Vn)
```

De même, il est conseillé de récupérer les résultats sur un fichier TEXT, puis d'aller vers un éditeur de texte et d'exploiter calmement les résultats de votre programme. Il faudra donc déclarer le fichier et le lier avec le fichier physique comme précédemment. Par contre le fichier doit être ouvert par :

```
REWRITE( Nom logique )
```

et les opérations d'écriture se font par

```
WRITE[LN] ( Nom logique, P1, P2, ....Pn)
```

Il faut toujours rajouter l'instruction `CLOSE( Nom logique )` afin de ne pas perdre les dernières écritures sur le fichier.

Exemple

Le programme qui suit effectue la somme des données lues sur le fichier Entree.pas et écrit les sommes temporaires sur le fichier Sortie.pas.

```
PROGRAM SOMME;
VAR
Fe, Fs : TEXT;
I, S, Nombre, Val : INTEGER ;
BEGIN
ASSIGN(Fe, 'Entree.pas');
ASSIGN(Fs, 'Sortie.pas');
RESET(Fe); REWRITE(Fs);
READLN(Fe, Nombre);
S := 0;
FOR I:= 1 TO Nombre DO
      BEGIN
      READLN(Fe, Val);
      S := S + Val;
      WRITELN(Fs, 'Somme temporaire = ', S);
      END;
WRITELN(Fs, '> Somme = ', S);
CLOSE(Fs)
END.
```

Contenu du fichier Entree.pas :

12 Nombre d'éléments
34
65
87
34
23
64
93
88
12
54
34
33

Contenu du fichier Sortie.pas :

Somme temporaire = 34
Somme temporaire = 99
Somme temporaire = 186

Somme temporaire = 220
Somme temporaire = 243
Somme temporaire = 307
Somme temporaire = 400
Somme temporaire = 488
Somme temporaire = 500
Somme temporaire = 554
Somme temporaire = 588
Somme temporaire = 621
> Somme = 621

Proverbes de programmation

- Définissez les problèmes complètement.

- Réfléchissez d'abord, vous programmerez plus tard.

- Choisissez bien vos identificateurs.

- Evitez les astuces.

- Employez les commentaires.

- Soignez la présentation.

- Fournissez une bonne documentation.

- Testez le programme à la main avant de l'exécuter.

- Ne vous occuper pas d'une belle présentation des résultats avant que le programme soit correct.

- Quand le programme tourne soignez la présentation des résultats.

- N'ayez pas peur de tout recommencer.

- Divisez pour régner.

- Ne supposez jamais que l'ordinateur suppose quelque chose.