LA PROGRAMMATION EN TURBO PASCAL

I- INTRODUCTION

Le PASCAL est un langage de programmation évolué car sa définition ne dépend pas de l'ordinateur sur lequel le programme est écrit. Il a été conçu par N. WIRTH. Depuis 1970, l'utilisation de ce langage s'est développé dans les universités et dans la communauté scientifique. C'est un langage très pédagogique.

1. CARACTERISTIQUES GENERALES DU LANGAGE

Le PASCAL est un langage structuré (utilisant des programmes par module logique). Un programme en PASCAL comporte trois parties :

- En-tête : elle contient le nom du programme et ses paramètres ;
- Partie déclarative : tous les objets utilisés doivent être prédéfinis ;
- Corps principal du programme : cette partie contient toutes les instructions nécessaires à l'exécution du programme.

2. CONCEPTION D'UN PROGRAMME EN PASCAL

La conception d'un programme en PASCAL passe par cinq étapes :

- l'analyse du problème,
- l'édition des lignes du programme,
- la compilation du fichier source,
- la sauvegarde du fichier correctement compilé,
- l'exécution du programme.

1- Analyse du problème

C'est l'étape la plus importante car les solutions du problème en dépendent. L'analyse du problème peut être fait en disséquant les difficultés de façon arborescente. D'une manière générale, il faut répondre aux questions suivantes :

- que doit faire le programme ?(il s'agit là de répertorier les résultats escomptés)
- quelles sont les informations dont je dispose pour résoudre le problème ?
- quelle méthode de résolution vais-je utiliser compte tenu des informations dont je dispose ?

2- Edition du fichier source

La saisie du texte de programme se fait tout comme celle d'un texte normal, mais elle doit respecter les règles de syntaxe et de lexique imposées par Turbo Pascal. Il est conseillé d'aérer suffisamment le texte afin de faciliter la lecture du programme et la correction des erreurs.

3- Compilation

Le texte source tel qu'il vient d'être édité ne peut pas être directement exécuté par la machine. Il faut donc le convertir dans un code compréhensible par la machine : c'est la **compilation**. Il s'agira de vérifier l'orthographe correcte des instructions (vérification lexicale) et leur bonne syntaxe (vérification syntaxique). Au cours de cette opération, une erreur rencontrée est signalée par un message. Il faut la corriger avant de continuer l'opération. Le succès d'une compilation est signalée par une fenêtre d'information qui apparaît à l'écran ; appuyer sur une touche pour la désactiver.

4- Sauvegarde du fichier source

Il s'agit essentiellement de garder une version du programme sur disque dur ou sur disquette.

5- Exécution du programme

Elle consiste à demander à la machine de suivre les instructions du programme pour produire les résultats. Cette opération n'est possible qu'après une compilation avec succès.

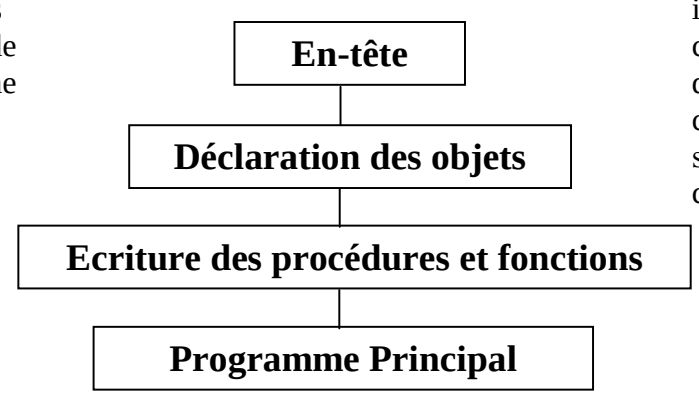
6- Principales touches de fonction

F1 : Pour avoir de l'aide contextuelle
F2 : Pour sauvegarder le texte source sur disque ou sur disquette
F3 : Ouvre un programme existant, ceci s'effectue dans une fenêtre des fichiers programme situés dans le répertoire TP.
F10 ou **ALT :** Pour activer la barre de menu
ALT + F9 : Pour lancer la compilation ou l'option **COMPILE** du **menu COMPILE**
CTRL + F9 : Pour lancer le programme ou sélectionner l'option **RUN** du **menu RUN**

2. INSTRUCTIONS EN TURBO PASCAL

Les instructions en TP sont construites autour de certains mots dits **réservés**. Les mots réservés ne doivent en aucun cas être utilisés n'importe comment.

La structure d'un programme en Turbo Pascal comprend :



1- En-tête

Tout programme en PASCAL débute par l'en-tête ; il commence comme suit :

PROGRAM <nom du programme> **;**

Exemple : **PROGRAM** Somme **;**

2- Partie déclarative

2.1- Déclaration des unités

Une unité est un programme spécial comportant des fonctions qui peuvent être directement utilisées dans le programme. TP dispose de certaines unités : SYSTEM, DOS, CRT , GRAPH, etc.

Pour déclarer une unité on commence par **USES,** ensuite on met le nom de l'unité :

USES
<nom de l'unité> ;

Exemple : **USES**
CRT ;

2.2- Déclarations des étiquettes

Une étiquette est un repère désignant une instruction ou un groupe d'instructions .
La déclaration des étiquettes se fait de la façon suivante :

Label
<noms des étiquettes séparées par des virgules> **;**

Exemple : **Label**
Fin, début ;

2.3- Déclaration des constantes

Une constantes en programmation est une valeur qui n'est modifiée par aucune instruction du programme. La déclaration se fait avec le mot réservé **CONST** :

CONST
<identificateur de la constante> = <valeur constante> ;

Exemple : **CONST**

Ville = 'Lomé' ;
Pi = 3.14 ;

2.4- Déclaration des variables

Une variable est une unité destinée à recevoir une valeur qui peut être utilisée suivant les instructions du programme.

Pour déclarer les variables dans un programme on commence par le mot **VAR**. D'une façon générale on donne le nom de l'identificateur de la variable, suivi de **:** et ensuite du type des données contenues dans la variable.

Type de variable en Turbo Pascal

Integer : Pour les entiers positifs ou négatifs.
REAL : Pour les réels.
BOOLEAN : Pour les variables ne recevant que les valeurs « True » ou « False ».
CHAR : Pour les variables ne recevant qu'un seul caractère du jeu ASCII.
SRTING : Pour les chaînes de caractère.
RECORD : Pour définir les enregistrements.
ARRAY : Pour définir les tableaux (matrices).

2.5- Les actions

Le programme principal commence par « **BEGIN** » et se termine par « **END.** ».

2.5.1- Affectations de valeurs

L'affectation est une opération qui permet d'attribuer une valeur à une variable. Elle est représentée par le signe : « **:=** ».

Exemple : nombre : = 5 ; la variable nombre prend la valeur 5.

2.5.2- Opérateurs arithmétiques

Opérateurs	Opération	Type des opérandes	Type du résultat
+	addition	entier	entier
		réel	réel
-	soustraction	entier	entier
		réel	réel
*	multiplication	entier	entier
		réel	réel
/	division	entier	réel
		réel	réel
div	division entière	entier	entier
mod	modulo	entier	entier

2.5.2- Expressions relationnelles

=	égalité	<>	différent
<	inférieur	<=	inférieur ou égal
>	supérieur	>=	supérieur ou égal

2.5.3- Opérations de lecture et d'écriture

READ : permet de lire une valeur tapée au clavier et de l'affecter à une variable.
Syntaxe : **READ** (<nom de variable>) ;

READLN : lit une variable et positionne le curseur sur la ligne suivante.

WRITE : permet d'envoyer un message à l'écran (affichage). On distingue deux cas :

- si le message est un commentaire destiné à éclairer l'utilisateur du programme alors on tape : **WRITE** ('<message>') ;

Exemple : **WRITE** ('Bonjour') ;

- lorsqu'il s'agit du résultat d'une opération effectuée par la machine on tape :
  WRITE (<nom de la variable contenant le résultat>) ;

Exemple : **WRITE** (somme) ;

- **WRITELN** : joue le même rôle que WRITE, à la seule différence qu'il repositionne le curseur sur la ligne suivante.

2.5.4- Actions conditionnelles : IF …THEN … ELSE

```
IF <condition> THEN
  Begin
      <action 1> ;
          .
      <action n> ;
  end
ELSE
  Begin
      <action a> ;
          .
      <action z> ;
  end ;
```

Exemple : IF nombre > 0 Then writeln ('Bravo') ;
Else Writeln('Dommage') ;

2.5.5- Les boucles

Le Turbo Pascal dispose de trois schémas d'itération :

- **La boucle « Pour » : FOR …TO … DO**

Syntaxe :

FOR <compteur> := <valeur min > **TO** <valeur max> **DO**
Begin
<action 1> ;
.
<action n> ;
End ;

Exemple : Faire la somme des 10 premiers entiers naturels.

```
Somme :=0 ;
For i := 1 to 9 Do somme := somme + i ;
```

- **La boucle « Tant que » : WHILE ..… DO**

Syntaxe :

WHILE <condition> **DO**
Begin
<action 1> ;
.
<action n> ;
End ;

Exemple : Faire la somme des 10 premiers entiers naturels.

```
i :=1 ;    Somme :=0 ;
While i < 10 Do
  begin
      somme := somme + i ;
      i := i + 1 ;
  end ;
```

- **La boucle « Répéter » : REPEAT ..… UNTIL**

Syntaxe :

REPEAT
<action 1> ;
.
<action n> ;
UNTIL <condition>;

Exemple : Faire la somme des 10 premiers entiers naturels.

```
Somme :=0 ;  i := 1 ;
Repeat
      somme := somme + i ;
      i :=i+1 ;
Until i = 10 ;
```

2.5.6- Présentation des résultats à l'écran

- **Effacer l'écran** : commande **CLRSCR** ;
- **Positionnement du curseur Instruction** : **GOTOXY**(colonne , ligne) ;

Exemple : GOTOXY(25, 10) ; le curseur se met à la colonne 20 et la ligne 10.

- **Commentaire dans le texte**

Les commentaires dans le programme sont mis dans des accolades.

Exemple : BEGIN **{**Début du corps de programme principal**}**
Au moment de la compilation, ce qui se trouve entre accolades n'est pas considéré.

- **Tracé de cadre**

Pour tracer un cadre, on commence par tracer les segments de droite qui le composent. Pour tracer une droite à l'écran on positionne le curseur puis on écrit le caractère approprié dans l'itération.

Exemple : Tracer une ligne de (15 , 5) à (45 , 5)

```
For i := 1 to 31 do
  Begin
      Gotoxy (14 + i , 5) ; write ('-') ;
  End ;
```

Pour le tracé on utilise souvent le code ASCII.

1. PROCEDURES ET FONCTIONS

Les procédures et fonctions permettent d'insérer des sous-blocs d'instructions dans le bloc principal du programme. Chaque déclaration de procédure ou de fonction est composée d'un en-tête puis d'un bloc.

Une procédure est activée par une instruction procédure.

Une fonction est activée par l'évaluation d'une expression contenant l'activation de la fonction ; la fonction se termine par le renvoi d'une valeur à cette expression.

1- Déclaration de procédure

Une déclaration de procédure associe un identificateur à un bloc d'instructions. La procédure peut ensuite être activée (appelée) par une instruction procédure.

Exemple :

```
PROCEDURE Numchaine (N : Integer ; VAR S : String) ;
VAR
   V : Integer ;
BEGIN
   V := Abs(N) ;
   S :='';
   REPEAT
         S := Chr (N MOD 10 + Ord('0')) + S;
         N := N DIV 10;
   UNTIL N=0;
   IF N < 0 THEN  S := '-' + S;
END;
```

2- Déclaration de fonction

Une déclaration de fonction définit un sous-programme qui effectue un certain traitement avant de renvoyer une valeur. L'en-tête de fonction comporte un mot réservé **FUNCTION**, l'identificateur de la fonction, les paramètres formels éventuels et le type de la valeur renvoyée.

Exemple : **FUNCTION** Max (a : Vecteur ; n : **Integer**) : **Extended** ;

```
VAR
    x : Extended ;
    i : Integer ;
Begin
    x:= a[ 1 ];
    For i:=2 to n DO
        If x < a[ i ] Then   x: = a[ i ] ;
    Max := x ;
End;
```

3- Quelques fonctions et procédures usuelles

Procédures ou fonctions	Effet
Chr	Renvoi du caractère de rang spécifié
Round	Arrondi d'une valeur de type réel en un entier long
Trunc	Ecrêtage d'une valeur de type réel en un entier long
Abs	Valeur absolue
Arctan	Arc tangente
Cos	Cosinus
Exp	Exponentielle
Frac	Partie fractionnaire
Int	Partie entière d'un nombre
Ln	Logarithme naturel
Sin	Sinus
Sqr	Carré d'un nombre
Sqrt	Racine carré d'un nombre
Dec	Décrémentation d'une variable scalaire
Inc	Incrémentation d'une variable scalaire
Odd	Test de non parité de l'argument
Pred	Renvoi de l valeur précédente de l'argument
Succ	Renvoi de la valeur suivante de l'argument
Str	Conversion d'une valeur numérique en chaîne de caractère
Val	Conversion d'une chaîne de caractère en valeur numérique