1- Introduction

Le pascal est un langage de programmation compilé : cela signifie que le programmeur fournit un fichier texte décrivant son programme et que ce fichier va ensuite être transformé en fichier binaire contenant des instructions compréhensibles par le micro processeur et éventuellement exécutable par le système d'exploitation (fichier EXE pour le DOS). Cette opération se nomme compilation. Turbo Pascal réalise la compilation et permet l'exécution des programmes.

Fichiers nécessaires

Pour fonctionner correctement, Turbo Pascal utilise les fichiers suivants, certains n'étant nécessaires que pour l'accès au mode graphique.

Turbo.exe	exécutable qui fournit un environnement complet de programmation : éditeur de textes, compilateur, aide en ligne, fonction de débogages, ...
Turbo.tpl	contient les librairies standards.
Turbo.tph	contient l'aide en ligne.
Graph.tpu	librairie graphique.
Les fichiers *.BGI	drivers pour différentes cartes vidéo; le fichier EGAVGA.BGI est indispensable pour les cartes EGA et VGA.
Les fichiers *.CHR	fontes de caractères graphiques.

Les versions de Turbo Pascal

Turbo Pascal évolue sans cesse. Ce document concerne et fournit donc des exemples utilisables par les versions 5.5, 6.0 et 7.0 de Turbo Pascal.

Petite bibliographie

Pour connaître l'ensemble des possibilités offertes par les unités standard de Turbo Pascal on peut se référer aux manuels fournis avec le produit ou à l'aide en ligne accessible directement à partir de l'environnement de programmation proposé. Quelques ouvrages complémentaires peuvent rendre des services :

- **"L'indispensable pour Turbo Pascal sur PC"**
  de J. Morleghem. (Livre de poche de la collection Marabout 499 pages)
  Présente l'environnement Turbo Pascal, la syntaxe, la structure d'un programme, les

types de données, les instructions, les notions de procédures et de fonctions, les unités, le contenu des unités standards SYSTEM, CRT, DOS, GRAPH et PRINTER, des tables et contient un index utile.
Peut servir de mini-dictionnaire, permet de retrouver rapidement une information.

- **"Turbo Pascal. Approche de la programmation objets."**
de J. Rivière. (Editions Dunod Informatique 425 pages)
Rappelle les bases de la programmation en Turbo Pascal, puis traite de problèmes spécifiques comme l'utilisation des pointeurs, la récursivité, la manipulation des fichiers, l'accès à l'environnement matériel, la gestion du mode graphique. Les deux derniers chapitres traitent de la programmation orientée objets.
109 exemples de programmes sont proposés. Peut servir de livre d'initiation.
- **"Le guide du Programmeur Turbo Pascal."**
de N.Rubenking. (Editions Dunod Tech 800 pages)
Présente des techniques avancées de programmation comme l'utilisation de l'assembleur intégré, la programmation orientée objets, des exemples de structures de données, etc...
Contient une multitude de trucs et conseils sur tous les sujets.
Une disquette contenant de nombreux exemples accompagne le livre.
Utile pour approfondir ses connaissances.

2- Les concepts de base

Un programme applique des opérations à des informations. Le programmeur doit :

- prévoir de l'espace mémoire pour ces informations
- permettre de recevoir les informations : gestion des entrées
- spécifier les opérations à effectuer
- permettre de fournir des résultats à l'utilisateur : gestion des sorties

Les variables et les types

Les variables sont destinées à contenir des informations modifiables : elles sont typées, ceci permet au compilateur d'une part de prévoir la taille de l'espace mémoire qui leur est nécessaire et d'autre part de définir les opérations qui peuvent les modifier. Les premiers types de variables sont :

- **Valeurs numériques entières**

Byte	taille de 1 octet, permet d'écrire des entiers de 0 à 255
Integer	taille de 2 octets, permet d'écrire des entiers de -32768 à 32767
Word	taille de 2 octets, permet d'écrire des entiers de 0 à 65535
LongInt	taille de 4 octets, permet d'écrire des entiers de -2147483648 à 2147483647

- **Valeurs numériques fractionnaires**

Real	taille de 6 octets; il donne 11 chiffres significatifs et permet d'écrire des nombres en écriture scientifique avec des exposants entre -38 et +38.

  - Note : l'utilisation d'un coprocesseur donne accès à d'autres types.
- **Valeurs booléennes**

Boolean	taille de 1 octet; deux valeurs possibles : true (vrai) et false (faux)

- **Caractères et chaînes de caractères**

Char	taille de 1 octet; 256 valeurs possibles définies par la table ASCII
String	taille de 256 octets; peut contenir des chaînes d'au plus 255 caractères; le premier octet représente en effet la longueur de la chaîne.
String[n]	n représente un entier positif inférieur à 255; taille de n+1 octets; peut contenir des chaînes d'au plus n caractères.

  - Note : il existe d'autres types prédéfinis et le programmeur peut aussi définir des types.

Les constantes

Les constantes sont destinées à recevoir des informations non modifiables.

Constantes numériques

Plusieurs notations sont possibles :

- notation décimale avec signe éventuel et le point pour séparer la partie entière de la partie fractionnaire.
  Exemples : 56.4 ou -7

- notation hexadécimale, le signe $ suivi d'un nombre hexadécimal.
  Exemples : $15 ou $AF00
- notation scientifique, mantisse suivi de E et de l'exposant
  Exemples : 1.6E8 ou 2.3E-5

Caractères

Le caractère est compris entre deux apostrophes ou le symbole # suivi du code ASCII.
Exemples : 'f' ou #123

Chaînes de caractères

La chaîne de caractères est comprise entre deux apostrophes.
Exemples : 'Ceci est une chaîne de caractères.'
Pour inclure une apostrophe dans une chaîne de caractères on écrit deux apostrophes consécutives.
Exemple : 'Voici l''exemple.'

Gestion des entrées/sorties

Entrées

L'instruction *Readln(Variable)* permet à l'utilisateur d'entrer au clavier la valeur associée à la variable qui est de type numérique ou de type chaîne de caractères.

Sorties

L'instruction *Write(Expression1[[,Expression2]....])* permet l'affichage à l'écran du contenu de Variable.
L'instruction Writeln(Expression1[[,Expression2]....]) effectue la même opération que Write avec un retour à la ligne à la fin de l'affichage.
Les expressions sont des variables ou des constantes ou des résultats d'opérations de type numérique ou chaîne de caractères.
Note : il est possible de mettre en forme les résultats numériques en utilisant la notation *write(nombre:c:d)* où c et d représentent le nombre total de chiffres affichés et le nombre de chiffres après la virgule.

Opérations sur les variables et constantes

Opérateur d'affectation

Opère sur tout type de variables. Représenté par := . Il permet d'affecter une valeur à une variable.

Opérateurs arithmétiques

Opèrent sur les variables numériques.

Multiplication	*

```
Division entière         div
Division fractionnaire   /
Modulo                   mod
Addition                 +
Soustraction             -
```

Opérateurs logiques

Opèrent sur les variables booléennes.

```
and     Et logique
or      Ou logique
xor     Ou exclusif
not     Négation
```

Opérateurs relationnels

Opèrent sur toutes les types de variables précédemment définis.

```
>      Supérieur
>=     Supérieur ou égal
<      Inférieur
<=     Inférieur ou égal
=      Egal
<>     Différent
```

Opérateur de chaînes de caractères

L'opérateur + permet de concaténer deux chaînes.

3- Structure d'un programme simple

Un programme est un texte que le compilateur va traduire en exécutable. Il doit avoir une structure particulière pour permettre au compilateur d'effectuer son travail.

Format du texte

Les lignes ont un maximum de 126 caractères.
Note : la largeur de l'écran est de 80 caractères.

Les lignes ou parties de lignes comprises entre les symboles { et } ou (* et *) sont ignorées par le compilateur. Elles permettent au programmeur d'entrer des commentaires.

Les instructions destinées au compilateur se terminent par un point virgule; elles peuvent contenir plusieurs lignes.

Identificateurs

Les entités manipulées par un programme (variables, instructions, ...) sont représentées par des identificateurs. Ceux-ci peuvent être formés d'un maximum de 63 caractères pris parmi les lettres de a à z, les chiffres de 0 à 9 et le caractère souligné _. Le premier caractère ne doit pas être un chiffre. Majuscules et minuscules sont confondues. Note : pas de caractères accentués et pas d'espaces.

Les différentes parties d'un programme

Le texte d'un programme contient au moins trois parties.

L'entête

Ne contient qu'une ligne; commence par le mot réservé Program et donne un nom au programme.

Les déclarations

Permettent de définir les éléments utilisés dans le programme.
Principe général du Pascal : définir avant d'utiliser.
En particulier on devra déclarer les variables utilisées pour permettre au compilateur d'effectuer les réservations de mémoire.

Le corps du programme

Commence par le mot réservé Begin et se termine par le mot réservé End suivi d'un point final. Ce qui suit ce End n'est pas pris en compte par le compilateur.
Entre Begin et End se trouvent les instructions à effectuer par le programme.

Premier exemple

```
{ Ce programme calcule votre âge. }
{ Entête du programme }
Program Calc_age;
{ Déclarations des constantes et des variables }
Const
{ mot réservé permettant de déclarer des constantes }
 cet_annee=1995;
Var
{ mot réservé permettant de déclarer des variables }
 naissance, age : integer;
{ corps du programme }
```

```
Begin
 write('Année de naissance : ');
 readln(naissance);
 age:=cet_annee-naissance;
 writeln('Tu as ',age,' ans.');
 writeln('Au revoir.');
End.
```

4- Contrôle du déroulement d'un programme

Le corps du programme est une suite d'instructions. Il est souvent nécessaire de les organiser en utilisant les notions d'exécution conditionnelle (certaines instructions ne sont exécutées que si certaines conditions sont vérifiées) et de boucle (certaines instructions sont exécutées un certain nombre de fois).

Exécution conditionnelle

L'instruction if...then...else...

Cette instruction prend l'un des 2 aspects suivants :

```
if Condition then Instruction1;
```

ou

```
if Condition then Instruction1
else Instruction2;
```

Condition est une expression booléenne. Instruction1 et Instruction2 sont soit une instruction simple soit un bloc d'instructions délimité par les mots réservés Begin et End. Si Condition est vraie, le programme exécute Instruction1, sinon c'est Instruction2 qui est exécutée lorsqu'elle existe.

L'instruction case...of

Cette instruction prend l'aspect suivant :

```
case Expression of
 valeur1 : Instruction1;
 valeur2 : Instruction2;
 .
 else Instruction par défaut
 end;
```

Expression représente une expression de type entier ou caractère; Valeur1, valeur2, ... représentent des constantes de type entier ou caractère qui sont les valeurs possibles de

Expression; Instruction1, Instruction2, ... sont soit une instruction simple soit un bloc d'instructions délimité par les mots réservés Begin et End.
Le programme n'exécutera que Instruction1 ou Instruction2 ou ... selon la valeur de Expression.
Note : la partie 'else Instruction par défaut' est facultative.

Boucles

La boucle While...do...

Cette instruction prend l'aspect suivant :

```
While Condition do Instruction;
```

Condition est une expression booléenne; Instruction est soit une instruction simple soit un bloc d'instructions délimité par les mots réservés Begin et End.
Le programme exécute Instruction tant que Condition est vraie.

La boucle Repeat...until...

Cette instruction prend l'aspect suivant :

```
Repeat
 Instruction1;
 Instruction2;
 .
Until Condition;
```

Instruction1, Instruction2, ... sont des instructions simples. Condition est une expression booléenne.
Le programme exécute la suite Instruction1, Instruction2, ... jusqu'à ce que Condition soit vraie.

La boucle For...do...

Cette instruction prend l'un des 2 aspects suivants :

```
For Compteur:=liminf to limsup do Instruction
```

ou

```
For Compteur:=limsup downto liminf do Instruction.
```

Compteur est une variable entière; liminf et limsup sont les valeurs limites de Compteur; Instruction est soit une instruction simple soit un bloc d'instructions délimité par les mots réservés Begin et End.
Dans les 2 cas le programme initialise Compteur, puis exécute Instruction et augmente (ou diminue) Compteur d'une unité et recommence jusqu'à ce que Compteur atteigne sa valeur finale.

5- Procédures et fonctions

Des suites d'instructions peuvent être rassemblées en blocs séparés qui peuvent être appelés depuis plusieurs endroits d'un programme. Ceci donne lieu aux notions de procédures et de fonctions.

Procédures

Ce sont des groupes d'instructions qui vont former une nouvelle instruction simple utilisable dans un programme. Comme toujours en Pascal il faut les définir avant de les utiliser. Ceci se fait en utilisant une structure similaire à celle d'un programme.

Entête

```
Procedure Identificateur(Param1:Type1,Param2:Type2,...);
```

Identificateur est le nom de la procédure; Param1, Param2 ... sont des paramètres que le programme fournit à la procédure sous forme de constantes, de variables ou d'expressions; Type1, Type2 ... sont les types de ces paramètres.

Déclarations

déclarations de constantes, types, variables utilisés à l'intérieur de la procédure

Corps de la procédure

```
Begin
 Instruction1;
 Instruction2;
 .
End;
```

Il s'agit des instructions exécutées par le programme à l'appel de la procédure. Une procédure peut appeler d'autres procédures définies avant elle. L'appel d'une procédure se fait en écrivant son nom suivi des paramètres nécessaires entre parenthèses.

Fonctions

Une fonction est une procédure qui devra fournir un résultat de type numérique ou chaîne de caractères. La définition se fait en utilisant une structure similaire à celle de la procédure.

Entête

```
 Function
Identificateur(Param1:Type1,Param2:Type2,...):TypeR;
```

Identificateur est le nom de la procédure; Param1, Param2 ... sont des paramètres que le programme fournit à la fonction sous forme de constantes, de variables ou d'expressions; Type1, Type2 ... sont les types de ces paramètres; TypeR est le type du résultat fourni par la fonction.

Déclarations

déclarations de constantes, types, variables utilisés à l'intérieur de la fonction.

Corps de la fonction

```
Begin
 Instruction1;
 Instruction2;
 .
 Identificateur:=résultat;
 .
End;
```

Il s'agit des instructions exécutées par le programme à l'appel de la fonction. L'une de ces instructions doit fournir le résultat de la fonction en l'affectant au nom de la fonction. L'appel d'une fonction se fait en écrivant son nom suivi des paramètres nécessaires entre parenthèses. Elle représente une expression du type du résultat fourni.

Variables et paramètres d'une procédure ou d'une fonction

Variables locales et variables globales

Les déclarations se trouvant à l'intérieur d'une procédure ou d'une fonction ne sont valables que pour la procédure ou la fonction et sont donc inconnues pour le reste du programme ; les variables ainsi déclarées sont dites locales. Par contre les déclarations faites dans la partie déclaration du programme sont valables pour les procédures et fonctions qui les suivent; les variables ainsi déclarées sont dites globales.

Transmission des paramètres

Les variables transmises en paramètres à une procédure ou une fonction ne sont pas modifiées pour le reste du programme, en effet la procédure ou la fonction qui les utilisent ne disposent que d'une copie : on a une transmission de paramètres par valeur.

Pour permettre à une procédure ou une fonction de modifier le contenu d'une variable passée en paramètre on utilise le mot réservé Var. Ainsi, la procédure déclarée par MaProc(Var r:Real) peut modifier le contenu de la variable r car elle travaille non avec une copie de la valeur de r, mais avec l'adresse mémoire de la variable r : on une transmission de paramètres par adresse.

Bibliothèques de procédures et de fonctions.

Les procédures et fonctions utilisées couramment peuvent être regroupées en bibliothèques appelées unités en Turbo Pascal. Dans ce cas on déclare l'utilisation des bibliothèques dans la partie déclarations du programme en utilisant le mot réservé Uses suivi des noms des unités.

Le programmeur peut créer de nouvelles unités.

Le Turbo Pascal fournit un certain nombre d'unités prêtes à l'emploi:

- **Unité System**
  Elle contient les procédures et fonctions de base comme Write ou Readln. C'est la seule unité qui n'a pas besoin d'être déclarée et qui est toujours utilisable.
- **Unité Crt**
  Permet la gestion de l'écran en mode texte, la gestion du clavier, la gestion du son.
- **Unité Dos**
  Permet la gestion des fonctions fournies par le DOS.
- **Unité Lst**
  Permet la gestion de l'imprimante.
- **Unité Graph**
  Permet la gestion de l'écran en mode graphique.

6- Programmes et unités

Un fichier source *.PAS de Turbo Pascal peut représenter un programme ou une unité (bibliothèque de procédures et de fonctions). Lors de la compilation un programme produira un fichier *.EXE qui sera donc directement exécutable sous Dos alors qu'une unité produira un fichier *.TPU non exécutable mais qui sera utilisé pendant la compilation d'un programme.

Structure générale d'un programme

On retrouve les parties déjà vues précédemment.

Entête et déclarations

- Program NomProg;
- Uses suivi des noms d'unités utilisées
- Const suivi des définitions de constantes
- Type suivi des définitions de types de données
- Var suivi des variables déclarées
- Procedure et function, déclaration des procédures et fonctions utilisées dans le programme.

Corps du programme

```
Begin
```

```
  Instruction1;
  Instruction2;
  .
End.
```

Structure générale d'une unité

La structure d'une unité est différente de celle d'un programme.

Entête

Cette partie ne contient qu'une ligne commençant par le mot réservé Unit suivi du nom de l'unité qui sera aussi le nom du fichier TPU produit.
Par exemple : Unit Outils;

Interface

Cette partie commence par le mot réservé INTERFACE.
Elle contient une série de déclarations d'objets qui seront accessibles aux programmes utilisant l'unité.

On retrouvera :

- Uses suivi des noms d'unités utilisées
- Const suivi des définitions de constantes
- Type suivi des définitions de types de données
- Var suivi des variables déclarées
- les déclarations des procédures et des fonctions partagées, seuls les entêtes apparaissent ici.

Implémentation

Cette partie commence par le mot réservé IMPLEMENTATION.
Elle peut contenir de nouvelles déclarations d'unités, de constantes, de types ou de variables, suivis de procédures et fonctions définies entièrement.
Tout ce qui n'a pas été déclaré dans la partie Interface sera invisible pour un programme utilisant l'unité.

Initialisation de l'unité

C'est la dernière partie de l'unité. Elle commence par un BEGIN et se termine par un END suivi du point final. Cette partie sera exécutée avant le début du programme utilisant l'unité.

7- Procédures et fonctions de l'unité System

L'unité System contient toutes les fonctions et procédures standards et d'entrées/sorties de Turbo Pascal.
Elle contient aussi les déclarations de constantes et variables utiles.

Procédures et fonctions de transfert

Elles transforment des données d'un type en un autre.

CHR	CHR(n):Char;	renvoie le caractère de code ASCII n.
ORD	ORD(ident):LongInt;	renvoie la position de ident dans son type; par exemple, ord('A') renvoie 65 car A est le 65ème caractère de la table ASCII.
ROUND	ROUND(r) : LongInt;	renvoie la valeur arrondie du réel r.
STR	STR(nombre[:i[:j]], chaine);	écrit nombre (entier ou réel) dans chaine de type String avec éventuellement un total de i caractères dont j après la virgule.
TRUNC	TRUNC(r):LongInt;	renvoie la partie entière du réel r.
VAL	Val(chaine,nombre,coderreur);	transforme chaine de type String en nombre (entier ou réel) et place un code dans la variable coderreur de type Integer; coderreur=0 lorsque l'interprétation a réussi.

Fonctions mathématiques

Ces fonctions opèrent sur les nombres du type Real et renvoient des valeurs de type Real.

ABS	ABS(r):Real;	renvoie la valeur absolue de r.
ARCTAN	ARCTAN(r):Real;	renvoie l'arc tangente de r.
COS	COS(r):REAL; renvoie le cosinus de r.	renvoie le cosinus de r.
EXP	EXP(r):Real;	renvoie l'exponentielle de r.
FRAC	FRAC(r):Real;	renvoie la partie fractionnaire de r.
INT	INT(r):Real;	renvoie la partie entière de r.
LN	LN(r):Real;	renvoie le logarithme népérien de r.

PI	PI;	renvoie la valeur du nombre pi.
SIN	SIN(r):Real;	renvoie le sinus de r.
SQR	SQR(r):Real;	renvoie le carré de r.
SQRT	SQRT(r):Real;	renvoie la racine carrée de r.

Procédures et fonctions sur les chaînes de caractères

Elles permettent les manipulations de chaînes de caractères.

CONCAT	CONCAT(chaine1,chaine2,[,...]):String;	équivalente à l'addition des chaines de caractères.
COPY	COPY(chaine,position,nombrecar):String;	renvoie une chaîne formée par les nombrecar caractères situés à partir de position dans chaine.
DELETE	DELETE(chaine,position,nombrecar);	supprime dans chaine nombrecar caractères situés à partir de position.
INSERT	INSERT(chaine1,chaine2,position);	insère chaine1 dans chaine2 à partir de position.
LENGTH	LENGTH(chaine):Byte;	renvoie la longueur de chaine.
POS	POS(chaine1,chaine2):Byte;	renvoie la position de la première occurence de chaine1 dans chaine2 et 0 si chaine1 n'a pas été trouvée.
UPCASE	UPCASE(c):Char;	renvoie la majuscule du caractère c.

Procédures et fonctions diverses

DEC	DEC(nomb1[,nomb2]);	exécute nomb1:=nomb1-nomb2 où nomb1 et nomb2 sont des entiers; la valeur par défaut de nomb2 est 1.
INC	INC(nomb1[,nomb2]);	exécute nomb1:=nomb1+nomb2 où nomb1 et nomb2 sont des entiers; la valeur par défaut de nomb2 est 1.
PARAMCOUNT	PARAMCOUNT:Word;	renvoie le nombre de paramètres spécifiés à l'appel du programme.

PARAMSTR	PARAMSTR(nomb):String;	renvoie le paramètre numéro nomb spécifié à l'appel du programme.
EXIT	EXIT;	permet de sortir d'une procédure ou d'une fonction avant sa fin.
HALT	HALT(n);	permet de quitter le programme en envoyant un ErrorLevel égal à n au DOS.
RANDOM	RANDOM[(nomb)]:Integer ou Real;	renvoie un nombre aléatoire entier entre 0 et nomb; si nomb n'est pas précisé on obtient un nombre aléatoire décimal entre 0 et 1.
RANDOMIZE	RANDOMIZE;	initialise le générateur de nombres aléatoires.