



Cours N°2

Règles Générales d'Écriture d'un Programme Pascal



1. Les Identificateurs

- Pour manipuler différents objets dans un programme, il faut leur donner des noms.
- Les noms utilisés pour les objets manipulés sont des **identificateurs**.

Définition :

L'identificateur est un nom symbolique utilisé pour nommer (identifier) un objet dans un programme informatique.



1. Les Identificateurs (suite)

Les « objets » dans un programme sont des :

- ✓ ***Nom du programme,***
- ✓ ***Constantes,***
- ✓ ***Variables,***
- ✓ ***Types,***
- ✓ ***Procédures,***
- ✓ ***Fonctions.***



1. Les Identificateurs (suite)

❑ Règles d'écriture d'un identificateur :

Les identificateurs sont représentés par une suite de lettres et/ou de chiffres avec les **restrictions** suivantes :

- ✓ le premier caractère doit être alphabétique, donc une lettre obligatoirement ;
- ✓ les caractères suivant le premier peuvent être numériques ;
- ✓ le caractère souligné « _ » est permis;
- ✓ les caractères dits « spéciaux » c'est-à-dire l'espace et les symboles : parenthèses, signe plus (+), signe moins (-), signe égal (=), point-virgule (;) sont **interdits** ;



1. Les Identificateurs (suite)

❑ Règles d'écriture d'un identificateur :

- ✓ l'utilisation des accents sur les lettres est interdite ;
- ✓ l'utilisation des mots clés (réservés) du langage est interdite ;
- ✓ l'utilisation des minuscules ou des majuscules est permise parce que *TURBO-PASCAL ne fait pas la différence entre les minuscules et les majuscules.*

Exemples:

ValeurM, Valeur_A, AB, B7, Nom

Sont des identificateurs corrects

Valeur M, Valeur-A, A/B, 7B, Nom\$

Sont des identificateurs incorrects



1. Les Identificateurs (suite)

❑ Les mots clés (mots réservés) du langage :
**and, array, begin, const, div, do, downto,
else, end, for, function, if, mod, not, of,
or, procedure, program, repeat, then, to,
type, until, var, while :**

sont des mots standards imposés par Turbo-Pascal ; leur signification et leur rôle sont parfaitement définis. On les appelle **mots clés** ou **mots réservés**.

Attention ! Un mot clé n'est **JAMAIS** accepté comme identificateur.



1. Les Identificateurs (suite)

❑ Les identificateurs prédéfinis :

- ✓ Les identificateurs prédéfinis sont des mots qui ont une signification *par défaut* en Turbo-Pascal.
- ✓ Il s'agit de : **abs, arctan, boolean, cos, exp, false, integer, ln, read, readln, real, round, sin, sqr, sqrt, true, trunc, write, writeln.**
- ✓ La différence par rapport aux mots clés est que les identificateurs prédéfinis peuvent être utilisés comme identificateurs ordinaires.
- ✓ Par exemple, on peut appeler une variable par **REAL** sans avoir des erreurs de compilation, ***mais cela est tout à fait déconseillé.***



2. Les Séparateurs

Définition :

Un séparateur est un espace ou un caractère ou une série de caractères, destinés à séparer des identificateurs.

- En Turbo-Pascal, les différents mots du langage sont séparés soit par un espace, soit par un signe particulier ou une fin de ligne.
- Dans un programme, deux identificateurs successifs doivent être séparés soit par un espace, soit par une fin de ligne. Sinon, le compilateur renvoie un message d'erreur.



2. Les Séparateurs (suite)

□ Exemples de séparateurs :

Séparateurs	Définition	Exemple
:	séparateur permettant de préciser le type d'une variable	VAR a : REAL;
;	séparateur fin de ligne	PROGRAM Essai;
,	séparateur virgule pour séparer des variables	VAR a, b : REAL;



3. Structure d'Ensemble de la Partie Déclaration

- La partie déclaration peut contenir différentes sortes de déclaration. Ces dernières sont introduites par des mots clés.
- Les mots clés utilisés dans la partie déclaration sont:

Uses

Const

Var

Label

Type

Function

Procedure



3. Structure d'Ensemble de la Partie Déclaration

Mots Clés	Signification	Exemples
Uses	C'est une bibliothèque utilisée par le compilateur la ou existe les fonctions et les procédures prédéfinis du langage	Uses Wincrt;
Const	Pour la déclaration des <i>constantes</i>	Const coeff=5;
Var	Pour la déclaration des <i>variables</i>	Var i:Integer; x,y:Real;
Label	Pour la déclaration des <i>étiquettes</i>	Label 10,20,nom;



3. Structure d'Ensemble de la Partie Déclaration

Mots Clés	Signification	Exemples
Type	Pour la définition de <i>nouveaux types</i>	Type Tab=Array[1..50] of Real;
Function	Pour la définition des fonctions	Function Max(a,b:Real):Real;
Procedure	Pour la définition des procédures	Procedure Min(a,b:Integer);

Chaque déclaration est séparée de la suivante ou du début du programme (**Begin**) par un point virgule (;)

Exemple

```
Program Essai;  
Uses wincrt;  
Const   Coeff=0.27;  
        Nbr_fois=15.5;  
Var     A,B:Real;  
        i,j:Integer;  
Begin
```



4. Types de Données en Pascal

- La notion de **type** est liée à la notion de **donnée**. C'est l'ensemble des valeurs que peut prendre une donnée.
- Donc le type désigne, pour un langage de programmation, deux ensembles :
 - *un ensemble de valeurs désigné explicitement ou par des valeurs extrêmes,*
 - *un ensemble d'opérations permises par le type.*
- Le langage pascal donne la possibilité d'utiliser :
 - *des types de données **prédéfinis**,*
 - *des types de données **définis par l'utilisateur**.*



4. Types de Données en Pascal (suite)

- Une classification nous permet de saisir trois catégories de types en Pascal :
 - 1) Les **types simples** pour lesquels les valeurs ne sont pas décomposables en constituants plus simples.
 - 2) Les **types structurés** qui couvre quatre catégories de données structurées :
 - Les *tableaux* (le type **ARRAY**)
 - Les *enregistrements* (le type **RECORD**)
 - Les *ensembles* (le type **SET**)
 - Les *fichiers* (le type **FILE**)
 - 3) Le type pointeur



5. Types de Données Simples Standards

Il existe 4 types de données simples standards :

- ✓ Le type **Entier**
- ✓ Le type **Réel**
- ✓ Le type **Caractère**
- ✓ Le type **Booléen**



5. Types de Données Simples Standards (suite)

A. Type Entier (Integer) :

Le type Entier spécifié par l'identificateur standard **INTEGER** correspond à l'ensemble des nombres entiers.

Représentation des valeurs

Les entiers s'écrivent en notation décimale avec éventuellement un signe + ou - devant

Exemple : **+23**
 23
 -6



5. Types de Données Simples Standards (suite)

A. Type Entier (Integer) :

- Les variables associées au type **INTEGER** ne correspondent pas aux ensembles infinis que l'on rencontre en mathématiques.
- L'ensemble des valeurs des entiers pris par une variable de type **INTEGER** est limité et ces limites sont liées à la longueur des mots mémoire utilisés pour représenter ces nombres.



5. Types de Données Simples Standards (suite)

A. Type Entier (Integer) :

Turbo-Pascal permet l'utilisation de cinq types de données prédéfinis pour le domaine des nombres entiers conformément au tableau suivant :

Type	Intervalle	Longueur
Shortint	-128 .. 127	1 octet (8 bits)
Integer	-32 768 .. 32 767	2 octets
Longint	-2 147 483 648 .. 2 147 483 647	4 octets
Byte	0 .. 255	1 octet
Word	0 .. 65 535	2 octet



5. Types de Données Simples Standards (suite)

A. Type Entier (Integer) :

Opérateurs et Fonctions relatifs au type entier :

Type	Notation	Signification	Exemples
Opérateurs de Comparaison	< , > , < > , = , <= , >=		
Opérateurs Arithmétiques	+	Addition	5+8 vaut 13
	-	Soustraction	9-10 vaut -1
	*	Multiplication	2*10 vaut 20
	Div	Division entière	(7)Div(2) vaut 3
	Mod	Reste de la division entière	(7)Mod(2) vaut 1
Fonctions Prédéfinies	Sqr	Élévation au carré	Sqr(4) vaut 16
	Abs	Valeur absolue	Abs(-10) vaut 10
	Succ	Entier suivant	Succ(5) vaut 6
	Pred	Entier précédent	Pred(5) vaut 4



5. Types de Données Simples Standards (suite)

B. Type Réel (Real) :

Le type Réel spécifié par l'identificateur standard **REAL** correspond à l'ensemble des nombres réels.

Représentation des valeurs

Les réels peuvent s'écrire sous forme de notation :

– Décimale (virgule fixe):

Nombre **REAL** avec partie entière et fractionnaire

Exemple : **12.43** **-0.45** **+1.0**

– Exponentielle (virgule flottante):

Nombre **REAL** avec partie entière, fractionnaire et un exposant

Exemple : **20E+2** **0.45e-4** **-1.5E1**



5. Types de Données Simples Standards (suite)

B. Type Réel (Real) :

- Le point fixe d'un nombre réel doit être toujours précédé et suivi d'une valeur. Ainsi **5.** est illégal mais **5.0** est correct.
- Comme dans le cas des nombres entiers, l'ensemble des valeurs des réels pris par une variable de type **REAL** est limité et ces limites sont liées à la longueur des mots mémoire utilisés pour représenter ces nombres.



5. Types de Données Simples Standards (suite)

B. Type Réel (Real) :

Turbo-Pascal permet l'utilisation de quatre types de données prédéfinis pour le domaine des nombres réels conformément au tableau suivant :

Type	Intervalle (en valeur absolue)	Longueur
Real	2.9 E-39 à 1.7 E+38	6 octets
Single	1.5 E-45 à 3.4 E+38	4 octets
Double	5.0 E-324 à 1.7 E+308	8 octets
Extended	3.4 E-4932 à 1.1 E+4932	10 octets

Les types Single, Double et Extended ne peuvent pas être utilisés que si l'ordinateur est équipé d'un coprocesseur mathématique du type 8087, 80287, 80387, etc., ou si on dispose d'un émulateur.



5. Types de Données Simples Standards (suite)

B. Type Réel (Real) :

Opérateurs et Fonctions relatifs au type réel :

Type	Notation	Signification	Exemples
Opérateurs de Comparaison	< , > , <> , = , <= , >=		
Opérateurs Arithmétiques	+	Addition	$x+y$
	-	Soustraction	$x-y$
	*	Multiplication	$x*y$
	/	Division	x/y
Fonctions Prédéfinies	Sqrt	Racine carré	Sqrt (x)
	Sqr	Élévation au carré	Sqr (x)
	Abs	Valeur absolue	Abs (x)
	Frac	Partie fractionnaire	Frac (1.35) vaut 0.35
	Int	Partie entière	Int (1.35) vaut 1



5. Types de Données Simples Standards (suite)

B. Type Réel (Real) :

Opérateurs et Fonctions relatifs au type réel :

Type	Notation	Signification	Exemples
Fonctions Prédéfinies	Round	Arrondi à l'entier le plus proche	Round (2 . 35) vaut 2 Round (2 . 85) vaut 3 Round (2 . 50) vaut 3
	Trunc	Éliminer la partie fractionnaire	Trunc (2 . 35) vaut 2 Trunc (2 . 85) vaut 2
	Sin	Sinus (en radiant)	Sin (x)
	Cos	Cosinus (en radiant)	Cos (x)
	Arctan	Arctg (en radiant)	Arctan (x)
	Ln	Logarithme népérien	Ln (x)
	Exp	Exponentiel	Exp (x)

Remarque : les fonctions Sin, Cos, Arctan, Ln et Exp peuvent être utilisées avec des variables de type entier.



5. Types de Données Simples Standards (suite)

C. Type Caractère (Char) :

Le type **CHAR** correspond à l'ensemble des valeurs qui définit les caractères.

Représentation des valeurs

L'ensemble du type **Char** est formé de plusieurs sous-ensembles :

- Les caractères alphabétiques minuscules ou majuscules :
a, b, c, ..., z, A, B, C, ..., Z
- Les caractères numériques : **0, 1, 2, ..., 9**
- Les caractères spéciaux : **+, -, *, /, =, ?, (,), [,], :, ;, \$, ...**
- L'espace (appelé le blanc)



5. Types de Données Simples Standards (suite)

C. Type Caractère (Char) :

- Turbo-Pascal utilise le code **ASCII** pour représenter les caractères. Les lettres **ASCII** représentent l'abréviation de *American Standard Code for Information Interchange*.
- En conformité avec ce code, un caractère est codé sur un octet, ce qui permet de définir 255 différents caractères.
- Une constante de type **CHAR** s'écrit par ***un caractère*** encadré d'une paire d'apostrophes

Exemple : 'A' , 'm' , ' ' , ... etc

Rem : si le caractère apostrophe doit être écrit sous forme d'une constante, alors il faut le doubler : '' {constante caractère apostrophe}.



5. Types de Données Simples Standards (suite)

C. Type Caractère (Char) :

Fonctions relatifs au type caractère :

Type	Notation	Signification	Exemples
Fonctions de Conversion	Ord	Le résultat de cette fonction est une valeur entière qui représente le code ASCII d'un caractère	Ord ('X') vaut 88 Ord ('A') vaut 65
	Chr	Le résultat de cette fonction retourne le caractère qui correspond au code ASCII d'un entier	Chr (88) vaut X Chr (65) vaut A
Fonctions de Succession	Succ	Caractère suivant	Succ ('B') vaut C
	Pred	Caractère précédent	Pred ('C') vaut B



5. Types de Données Simples Standards (suite)

D. Type Booléen (Boolean) :

Le **type booléen**, dit type logique, est un type symbolique qui ne peut recevoir que deux valeurs logiques : **TRUE** (Vrai) et **FALSE** (Faux).

Représentation des valeurs

- ✓ Si, dans un programme, une variable doit recevoir des valeurs logiques, elle doit être déclarée de type **BOOLEAN**. Alors, les valeurs possibles de cette variable sont **TRUE** et **FALSE**.
- ✓ La déclaration d'une *constante de type* **BOOLEAN** se fait en affectant la valeur constante **TRUE** ou **FALSE** à la constante.



5. Types de Données Simples Standards (suite)

D. Type Booléen (Boolean) :

- Des valeurs logiques ne peuvent pas être entrées au clavier ; elles doivent être affectées à une variable de type booléen pendant le déroulement du programme donc, de façon dynamique.
- Le contenu des variables ou constantes de type **BOOLEAN** peut être affiché, comme pour les autres variables, avec l'instruction **Write** ou **Writeln**.

Exemple :

```
OK:= True;  
WRITE(OK); {affiche TRUE}  
FIN:= False;  
WRITE(FIN); {affiche FALSE}
```



5. Types de Données Simples Standards (suite)

D. Type Booléen (Boolean) :

Opérateurs relatifs au type booléen :

Type	Notation	Signification	Exemples
Opérateurs de Comparaison	=	Égale	
	<	Inférieur	
	>	Supérieur	
	<=	Inférieur ou égale	
	>=	Supérieur ou égale	
	<>	Différent	
Opérateurs Logique	And	Et (Conjonction)	(A < B) And (C < D)
	Or	Ou (Disjonction)	(A < B) Or (C < D)
	Not	Non (Négation)	Not (A < B)

La règle de comparaison suivante est utilisée pour les valeurs logiques:

FALSE < TRUE



6. L'Affectation

Définition :

- ✓ *L'affectation permet d'attribuer une valeur ou une expression à une variable de même type*
- ✓ *L'instruction d'affectation se note par le symbole " : = "*

Exemple: on suppose que x est une variable entière

Avant	Affectation	Après
X ?	$X := 10;$	X 10
X 10	$X := X + 5;$	X 15

Si x est de type réel et y est de type entier, alors:

$x := y;$	Possible
$y := x;$	Impossible



7. Les Commentaires

- Comme tout langage évolué, Pascal permet la présence de commentaires dans un ***programme source***.
- Les commentaires sont des textes explicatifs destinés aux lecteurs du programme et qui ne seront pas lus par la machine.
- Les commentaires sont ignorés par le compilateur et n'influencent pas l'exécution du programme ; ils sont utilisés seulement pour ***documenter le programme***.
- Pour introduire un commentaire dans le programme source, il y a deux possibilités :
 - Utilisation des accolades { }
 - Utilisation des parenthèses (* *)

Exemple

```
{Ceci est commentaire}  
(* ceci est un autre commentaire *)
```




7. Les Commentaires (suite)

- Les deux symboles sont équivalents dans les sens qu'on peut utiliser un ou l'autre pour écrire de commentaires mais avec une restriction :
 - un commentaire ouvert par { doit être absolument fermé par }
 - un commentaire ouvert par (* doit être absolument *fermé* par *)
- Un commentaire peut apparaître dans un programme à n'importe quel endroit où un espace ou une fin de ligne sont permis. Par contre un commentaire ne pourra pas apparaître dans un identificateur ou dans une constante.

Exemple :

```
Program addition;  
(* Programme permettant l'addition de 2 nombre réels *)  
Uses wincrt;  
Var A,B,Somme:Real; { Partie déclaration }  
Begin
```



8. Entrées – Sorties

Définition :

Pour faire fonctionner un programme il faut lui fournir des données et prévoir la possibilité de récupérer les résultats. Ces deux opérations portent le nom générique d'opérations d'entrée/sortie. Donc il s'agit de deux opérations distinctes :

- ✓ *Entrée de données (Lecture des données)*
- ✓ *Sortie de résultats (Écriture des résultats)*

- En Pascal, les opérations d'entrée/sortie sont réalisées par deux procédures système :

READ – procédure standard de lecture

WRITE – procédure standard d'écriture



8. Entées – Sorties (suite)

A. Instructions de Lecture (Read, ReadLn) :

- Les instructions de lecture permettent à l'utilisateur de saisir des valeurs au clavier (ou a partir d'un fichier) pour qu'elles soient utilisées par le programme.
- Dès que le programme rencontre une instruction **READ** ou **READLN** l'exécution s'interrompt attendant la saisi d'une valeur.

Syntaxe des instructions de lecture

Read(Liste_de_variables);

ReadLn(Liste_de_variables);

Lire la valeur de
liste_de_variables et revenir
a la ligne



8. Entrées – Sorties (suite)

A. Instructions de Lecture (Read, ReadLn) :

Exemple

Supposons qu'on veut saisir 4 variables entières dont les valeurs sont :

A=5 B=10 C=2 D=6

```
Read(A, B);  
ReadLn(C);  
Read(D);
```

Exécution

```
5 10 2  
6
```



8. Entées – Sorties (suite)

B. Instructions d'Écriture (**Write**, **WriteLn**) :

- Les instructions d'écriture permettent au programme de communiquer des valeurs (ou des messages) à l'utilisateur en les affichant à l'écran (ou sur un fichier).

Syntaxe des instructions d'écriture

```
Write(Liste_de_variables);
```

```
WriteLn(Liste_de_variables);
```

Écrire la valeur de **liste_de_variables** et revenir à la ligne

```
Write('Ceci est un Message');
```

Écrire le message écrit entre les apostrophes



8. Entées – Sorties (suite)

B. Instructions d'Écriture (Write, WriteLn) :

Exemple

Supposons qu'on a : **A=5** est de type réel

```
Write('La valeur de A est:');  
WriteLn(A);  
Write('Ceci est un message affiché à l''écran');
```

Exécution

```
La valeur de A est: 5.000000000000E+00  
Ceci est un message affiché à l'écran
```



8. Entées – Sorties (suite)

B. Instructions d'Écriture (`Write`, `WriteLn`) :

☐ Instructions d'affichage par défaut :

Pour chaque information mentionnée dans une instruction d'écriture, on peut choisir entre :

- ✓ Laisser le langage Pascal imposer sa présentation :
on parle alors de "***Format d'affichage par défaut***"
- ✓ Imposer notre propre format d'affichage



8. Entées – Sorties (suite)

B. Instructions d'Écriture (Write, WriteLn) :

❑ Exemple d'affichage par défaut :

```
program affichage;
uses wincrt;
var n,p:Integer;
    x,y:Real;
    C1,C2:Char;
    ok:Boolean;
Begin
  write('n= ');readln(n);
  write('p= ');readln(p);
  write('x= ');readln(x);
  write('y= ');readln(y);
  write('C1= ');readln(C1);
  write('C2= ');readln(C2);
  ok:=false;
  writeln('Nombre',n);
  writeln('Nombre ',n);
  writeln(n,    p);
  writeln(n,'  ',p);
  writeln(x,y);
  writeln(C1,C2);
  writeln('Cela est',ok);
End.
```

Exécution

```
n= 3
p= 125
x= -34.5e6
y= 2
C1= A
C2= i
Nombre3
Nombre 3
3 125
-3.4500000000E+07 2.0000000000E+00
Ai
Cela estFALSE
```




8. Entées – Sorties (suite)

B. Instructions d'Écriture (`Write`, `WriteLn`) :

☐ Règles générales de l'affichage par défaut :

Le tableau suivant présente les cases utilisés pour l'affichage par défaut de chaque type de données:

Nature de l'Expression	Cases utilisés pour son affichage
Entière (Integer)	Sa propre longueur
Réelle (Real)	17
Caractère (Char)	1
Booléenne (Boolean)	4 (pour True) ou 5 (pour False)



8. Entées – Sorties (suite)

B. Instructions d'Écriture (**Write**, **WriteLn**) :

❑ Imposition d'un format d'affichage :

- Tous les types d'expressions figurant dans une instruction **Write** (ou **WriteLn**), peuvent se voir imposer un gabarit (nombre de cases) par une indication de la forme (**:g**) où **g** représente une expression entière quelconque.
- De plus, pour les expressions de type réel, on peut imposer un gabarit de la forme (**:g:d**) qui correspond à la forme point fixe avec **d** décimales.



8. Entées – Sorties (suite)

B. Instructions d'Écriture (`Write`, `WriteLn`) :

- ❑ Imposition d'un format d'affichage :

Exemple

Supposons qu'on a : **A=25.35** est de type réel

```
WriteLn('La valeur de A est: ',A);  
WriteLn('La valeur de A est: ',A:10);  
WriteLn('La valeur de A est: ',A:9:3);
```

Exécution

```
La valeur de A est:  2.53500000000E+01  
La valeur de A est:  2.535E+01  
La valeur de A est:  25.350
```