Objets, expressions, instructions de base

Dr Khadim DRAME

Département d'Informatique
UFR des Sciences et Technologies
Université Assane Seck de Ziguinchor

27 mai 2021

Plan

Objets

- Un objet est une abstraction qui désigne un élément.
- Un objet possède trois caractéristiques :
  - un **identificateur** : nom qui permet de l'identifier de manière unique ;
  - un **type** qui détermine la nature de l'objet (entier, réel, ...) ;
  - une **valeur** qui peut être fixe (constante) ou changée durant l'exécution (variable).

Objets

Identificateur

- C'est un nom qui désigne et identifie de manière unique un objet.
- Un identificateur est une suite de caractères alphanumériques (lettres de l'alphabet et chiffres) et de trait de soulignement (tiret de la touche 8).
- Il commence obligatoirement par une lettre ou trait de soulignement.
- Il est choisi de préférence, en rapport avec le contenu de l'objet.
- Attention : ***les mots réservés du langage sont interdits !***

Objets

Type

- Le type détermine la nature de l'objet, définit le domaine de ses valeurs et les opérations applicables.
- Les types de bases en Pascal sont :
  - **integer** (entier) ;
  - **real** (réel) ;
  - **boolean** (booléen) ;
  - **char** (caractère) ;
  - **string** (chaîne de caractères).

Types

Type integer

- Le type **integer** désigne l'ensemble des nombres entiers relatifs.

  Exemples : 137, -54, 0

Type real

- Le type **real** désigne l'ensemble des nombres réels signés.
- La **virgule** est représentée par un **point** !

  Exemples : -3.65, 845.4, 5.0

Type boolean

- Le type **boolean** désigne les objets prenant leur valeur dans {true, false}.
- Il est utile pour exprimer les conditions.

Types

Type char

- Le type **char** désigne les caractères imprimables :
  - les 26 lettres de l'alphabet (minuscules et majuscules) ;
  - les 10 chiffres ;
  - les caractères spéciaux (espace, virgule, parenthèse, ...).
- Les valeurs de type char sont mises entre côtes.

  Exemple : 2 représente un chiffre et '2' représente un caractère.

Type string

- Le type **string** désigne une suite de caractères (texte).
- Les valeurs de type string sont mises entre côtes.

  Exemples : 'bonjour', 'au revoir'

Variable/Constante

Variable

- Une variable est un emplacement mémoire réservé pour stocker une valeur.
- Une variable est un objet dont la valeur est modifiable.
- La valeur d'une variable peut être modifiée au cours de l'exécution du programme.

Constante

- Une constante est un objet dont la valeur est fixe et attribuée avant l'exécution du programme.
- La valeur d'une constante ne peut pas être modifiée au cours de l'exécution du programme.

Variable/Constante

Déclaration de variable

- Elle permet de réserver un espace mémoire et de l'associer à une variable.
- Elle indique l'identificateur et le type de la variable.
- Syntaxe

  var <identificateur_variable> : <type> ;

Exemple

```
1 var
2   age : integer;
3   poids, taille : real;
4   nom : string;
```

Variable/Constante

Déclaration de constante

- Elle permet de réserver un espace mémoire et de l'associer à une constante.
- Syntaxe

  const <identificateur_constante> = <valeur> ;

Exemple

```
1 const
2   pi = 3.14;
3   g = 9.8;  {gravitation de la terre}
```

Plan

Opérateurs

Opérateur

- Un **opérateur** permet d'effectuer une opération particulière sur les données.
- La donnée à laquelle s'applique un opérateur est dite **opérande**. Elle peut être une constante, une variable, un appel de fonction ou une expression.
- Selon le nombre d'opérandes, on peut distinguer :
  - les **opérateurs unaires** qui admettent une seule opérande ;
  - les **opérateurs binaires** qui admettent deux opérandes.
- On peut classer les opérateurs en :
  - **opérateurs arithmétiques** ;
  - **opérateurs relationnels** ;
  - **opérateurs logiques**.

Opérateurs

Opérateurs arithmétiques

- Ils permettent des opérations sur les nombres (entiers et réels).

Opérateur	Rôle	Exemples
+	addition	5+3 vaut 8
-	soustraction	3-5 vaut -2
*	multiplication	3*5 vaut 15
/	division réelle	13/5 vaut 2.6
div	**division entière**	13 div 5 vaut 2
mod	**modulo** (reste de la division entière)	13 mod 5 vaut 3

Attention : ***les opérateurs div et mod ne s'appliquent que sur des entiers*** !

Opérateurs

Opérateurs relationnels

- Ils permettent de faire des comparaisons.
- Une opération relationnelle renvoie une valeur booléenne (true, false).

Opérateur	Sémantique	Exemple
=	égal	3=2+1 vaut true
<	strictement inférieur	3<2 vaut false
<=	inférieur ou égal	3<=3 vaut true
>	strictement supérieur	3>3 vaut false
>=	supérieur ou égal	5>=3 vaut true
<>	différent	'a'<>'b' vaut true

Ils s'appliquent sur *les nombres, les caractères et les booléens*.

Opérateurs

Opérateurs logiques

- Ils permettent des opérations sur des objets et expressions booléens.

Opérateur	Rôle
not	négation
and	ET logique
or	OU logique

A	B	A and B	A or B
false	false	false	**false**
false	true	false	true
true	false	false	true
true	true	**true**	true

Opérateurs

Exemples (Opérations booléennes)

not(3>5)

not(3<5)

(3>2) and (3<5)

(3>2) and (3>5)

(3>2) or (3>5)

(3<0) or (3>5)

Opérateurs

Fonctions prédéfinies sur les nombres

- **cos** (cosinus), **sin** (sinus), **tan** (tangente), ... ;
- **abs** (valeur absolue), **sqr** (carré), **sqrt** (racine carrée) ;
- **ln** (logarithme), **exp** (exponentiel) ;
- **trunc** (partie entière), **round** (entier le plus proche).

Exemples

abs(-3) vaut 3

sqr(2+3) vaut 25

sqrt(9) vaut 3.0

trunc(12.7) vaut 12

round(12.7) vaut 13

Expressions

Expression

- Une **expression** est une combinaison d'opérandes (constantes, variables, appels de fonction et/ou expressions) et d'opérateurs.
- Comme un objet, une expression a une valeur et un type.
- Exemples :

  $12 + 13$ vaut **25** et est de type **integer**

  $8 > 5$ vaut **true** et est de type **boolean**

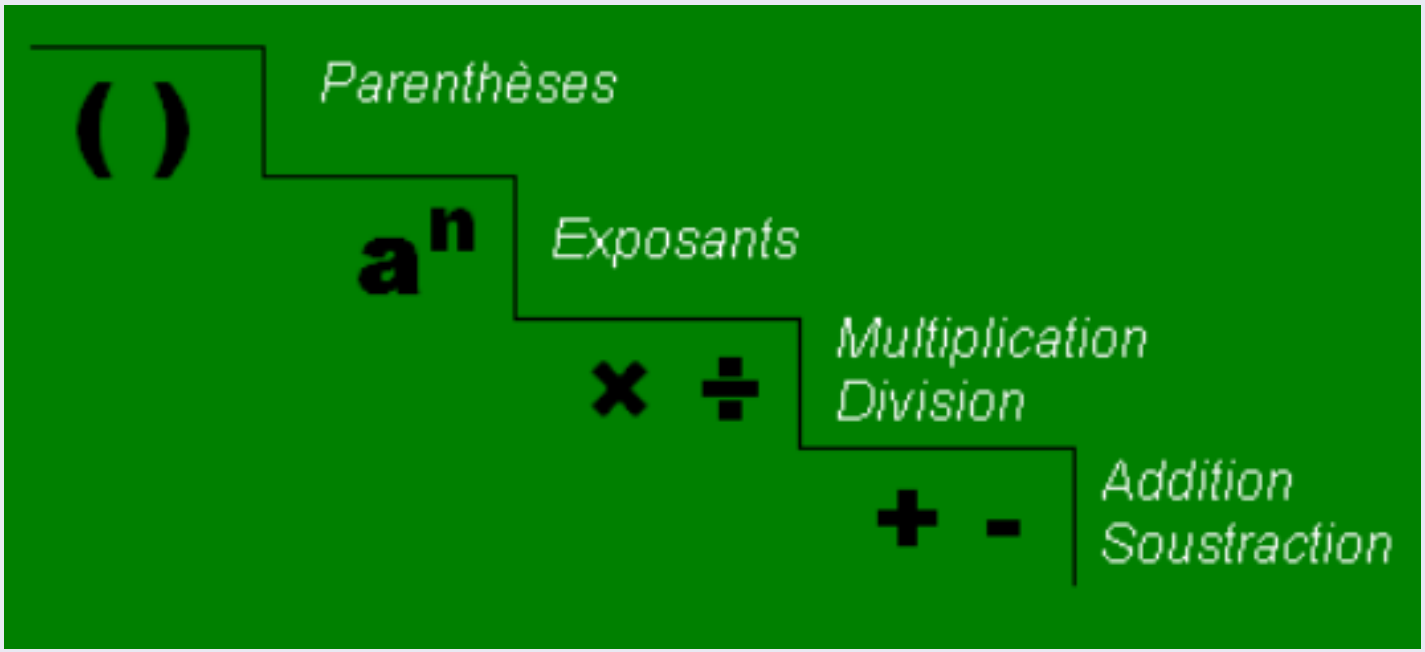
Évaluation d'une expression

- L'évaluation d'une expression ayant plusieurs opérateurs se fait en suivant les **règles de priorité** de ses opérateurs.

Expressions

Règles de priorité

- Les règles de priorité sont les mêmes qu'en Mathématiques.
- Si deux opérateurs ont la même priorité, l'évaluation est effectuée de gauche à droite.
- Règles de priorité des **opérateurs arithmétiques** : **PEMDAS**



Expressions

Règles de priorité

- Règles de priorité entre les opérateurs logiques :
  - **not** est prioritaire sur tous ;
  - **and** est prioritaire sur **or** et **xor**.
- Les opérateurs arithmétiques et logiques sont prioritaires sur les opérateurs relationnels.
- Exemple :

  (3>0) and (3<5)

  true and true

  true

Expressions

Exercice d'application : Donner la valeur et le type des expressions suivantes :

1. 10 * 2 + 3
2. 10 * (2 + 3)
3. 10 - 5 + 6 * 4 + 6
4. 20 div 6
5. 10 = 8 + 2
6. 15 mod 2 = 0
7. (10 < 20) and (10 <= 5)
8. (10 < 20) or (10 <= 5)
9. (10 < 20) and (10 < 15)
10. (10 < 7) and (10 > 15)

Plan

Instructions simples en Pascal

Instructions

- Deux types d'instructions :
  - **Instructions simples** : affectation, lecture et écriture ;
  - **Instructions composées** : séquence, choix et répétition.
- Les instructions simples sont étudiées dans cette partie.
- Les instructions composés seront étudiées dans les prochains chapitres.

Instructions simples

Affectation

- L'**affectation** ou l'**assignation** consiste à allouer une valeur à une variable.
- L'opérateur d'affectation est **:=** (deux points égal).
- Syntaxe :
  <identificateur_variable> := <valeur ou expression> ;
- La valeur de l'expression est calculée et mémorisée dans la variable.
- Le type de l'expression (toujours à droite) doit être compatible avec le type de la variable (toujours à gauche).

Instructions simples

Exemples (affectation)

```
1  age := 45; {age recoit la valeur 45}
2  poids := 57.4; {poids recoit la valeur 57.4}
3  rep := true; {rep recoit la valeur true}
4  a := 12 + 8; {a recoit la valeur 20}
5  b := 2 * a; {b recoit la valeur 40}
6  b := b + 6; {b recoit la valeur 46}
```

Instructions simples

Exercices d'application (affectation)

- Exercice 1

```
1 program exo1;
2 var
3   a, b : integer;
4 begin
5   a := 1 ;
6   b := a + 3 ;
7   a := a + 1 ;
8   b := a - 4 ;
9 end.
```

Quelles seront les valeurs des variables a et b après l'exécution des instructions suivantes ?

Instructions simples

Exercices d'application (affectation)

- Exercice 1

```
1 program exo1;
2 var
3   a, b : integer;
4 begin
5   a := 1 ;
6   b := a + 3 ;
7   a := a + 1 ;
8   b := a - 4 ;
9 end.
```

Quelles seront les valeurs des variables a et b après l'exécution des instructions suivantes ?

Réponse : **a vaut 2** et **b vaut -2**.

Instructions simples

Exercices d'application (affectation)

- Exercice 2

```
1 program exo2;
2 var
3   a, b, c : integer;
4 begin
5   a := 5;
6   c := a + b;
7   c := c - a;
8   b := 4.5;
9 end.
```

Ce programme contient deux erreurs. Identifiez les.

Instructions simples

Exercices d'application (affectation)

- Exercice 2

```
1 program exo2;
2 var
3   a, b, c : integer;
4 begin
5   a := 5;
6   c := a + b;
7   c := c - a;
8   b := 4.5;
9 end.
```

Ce programme contient deux erreurs. Identifiez les.

Réponse :

c := a + b : la variable b non initialisée !

b := 4.5 : une valeur réelle affectée à une variable entière.

Instructions simples

Lecture/Écriture

- Les périphériques d'entrée/sortie permettent à la machine de communiquer avec l'utilisateur à travers :
  - un clavier (périphérique d'entrée) pour entrer les données ;
  - un écran (périphérique de sortie) pour afficher les résultats.
- L'instruction permettant d'entrer les données est dite **lecture**.
- L'instruction permettant d'afficher les résultats est dite **écriture**.

Instructions simples

Écriture

- L'**écriture** consiste à imprimer une suite de caractères imprimables à l'écran.
- Deux types d'informations peuvent être imprimer :
  - les **chaînes de caractères** :
    - Elles sont délimitées par des côtes(') ;
    - elles sont affichées tel qu'elles ont été saisies ;
    - elles sont souvent utilisées pour envoyer des messages à l'utilisateur.
  - les **données** (variable, constante, expression) :
    - elles sont notées sans côtes ;
    - c'est leur valeur qui est affichée à l'exécution.

Instructions simples

Écriture

- Deux procédures pour l'écriture :
  - **write()**qui écrit à l'écran et positionne le curseur après le dernier caractère sur la même ligne ;
  - **writeln()** qui écrit à l'écran et positionne le curseur sur la ligne suivante.
- Syntaxes :

  write(<expression>) ;

  writeln(<expression>) ;

  Où expression peut être une chaine de caractères, une constante, une variable ou une combinaison de ces dernières.

Instructions simples

Exemples (écriture)

```
1  x := 10;
2  y := 20;
3  write('La valeur de x est ');
4  writeln(x);
5  writeln('La valeur de y est ', y);
6  writeln('La valeur de x+y est ', x+y);
7  write('La valeur de ',x,'+',y,' est ',x+y);
```

Instructions simples

Exemples (écriture)

```
1  x := 10;
2  y := 20;
3  write('La valeur de x est ');
4  writeln(x);
5  writeln('La valeur de y est ', y);
6  writeln('La valeur de x+y est ', x+y);
7  write('La valeur de ',x,'+',y,' est ',x+y);
```

Après exécution

La valeur de x est 10
La valeur de y est 20
La valeur de x+y est 30
La valeur de 10+20 est 30

Instructions simples

Lecture

- La **lecture** consiste à récupérer une valeur saisie au clavier et à l'assigner à une **variable**.
- La valeur saisie doit être compatible avec le type de la variable.
- Deux procédures pour la lecture :
  - **read()** qui lit au clavier et positionne le curseur après le dernier caractère sur la même ligne ;
  - **readln()** qui lit au clavier et positionne le curseur sur la ligne suivante.
- Syntaxes :

  read(<liste_variables>) ;

  readln(<liste_variables>) ;

Instructions simples

Exemples (lecture)

```
1  write('Donner la valeur de x : ');
2  readln(x);
3  write('Donner les valeurs de y et z :');
4  readln(y,z);
5  write('La valeur de x est ',x);
```

Instructions simples

Exemples (lecture)

```
1  write('Donner la valeur de x : ');
2  readln(x);
3  write('Donner les valeurs de y et z :');
4  readln(y,z);
5  write('La valeur de x est ',x);
```

Après exécution

Donner la valeur de x : 2
Donner les valeurs de y et z :
4 5
La valeur de x est 2

Instructions simples

Exercices d'application (lecture/écriture)

- Exercice 3
  Écrire un programme qui calcule et affiche la somme de deux nombres entiers entrés par l'utilisateur.

Instructions simples

Exercices d'application (lecture/écriture)

- Exercice 3
  Écrire un programme qui calcule et affiche la somme de deux nombres entiers entrés par l'utilisateur.
  Réponse :

```
1  program somme_entier;
2  var
3    a, b, somme : integer;
4  begin
5    write('Donner un nombre entier : ');
6    readln(a);
7    write('Donner un autre nombre entier : ');
8    readln(b);
9    somme := a + b;
10   write('La somme est : ',somme);
11   readln();
12 end.
```

Synthèse

- La **déclaration** d'une variable permet de **réserver un espace mémoire** et de l'associer à la variable.
- Une variable est caractérisée par :
  - un **identificateur** qui permet de l'identifier ;
  - une **valeur** qui peut varier au cours du programme ;
  - un **type** qui détermine sa taille et les opérations possibles.
- Trois instructions de base :
  - l'**affectation** qui permet d'attribuer une valeur à une variable ;
  - la **lecture** (saisie) qui permet d'assigner à une variable, une valeur saisie au clavier ;
  - l'**écriture** (affichage) qui permet d'imprimer le contenu d'une variable ou d'une expression à l'écran.
- Priorité des opérateurs numériques : **PEMDAS**.