Ministère de l'enseignement supérieur et de la recherche scientifique
Université Djillali BOUNAAMA - Khemis Miliana (UDBKM)
Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Chapitre 2

Programmation impérative

MI-GLSD-M1-UEM213 : Paradigmes de langages de Programmation

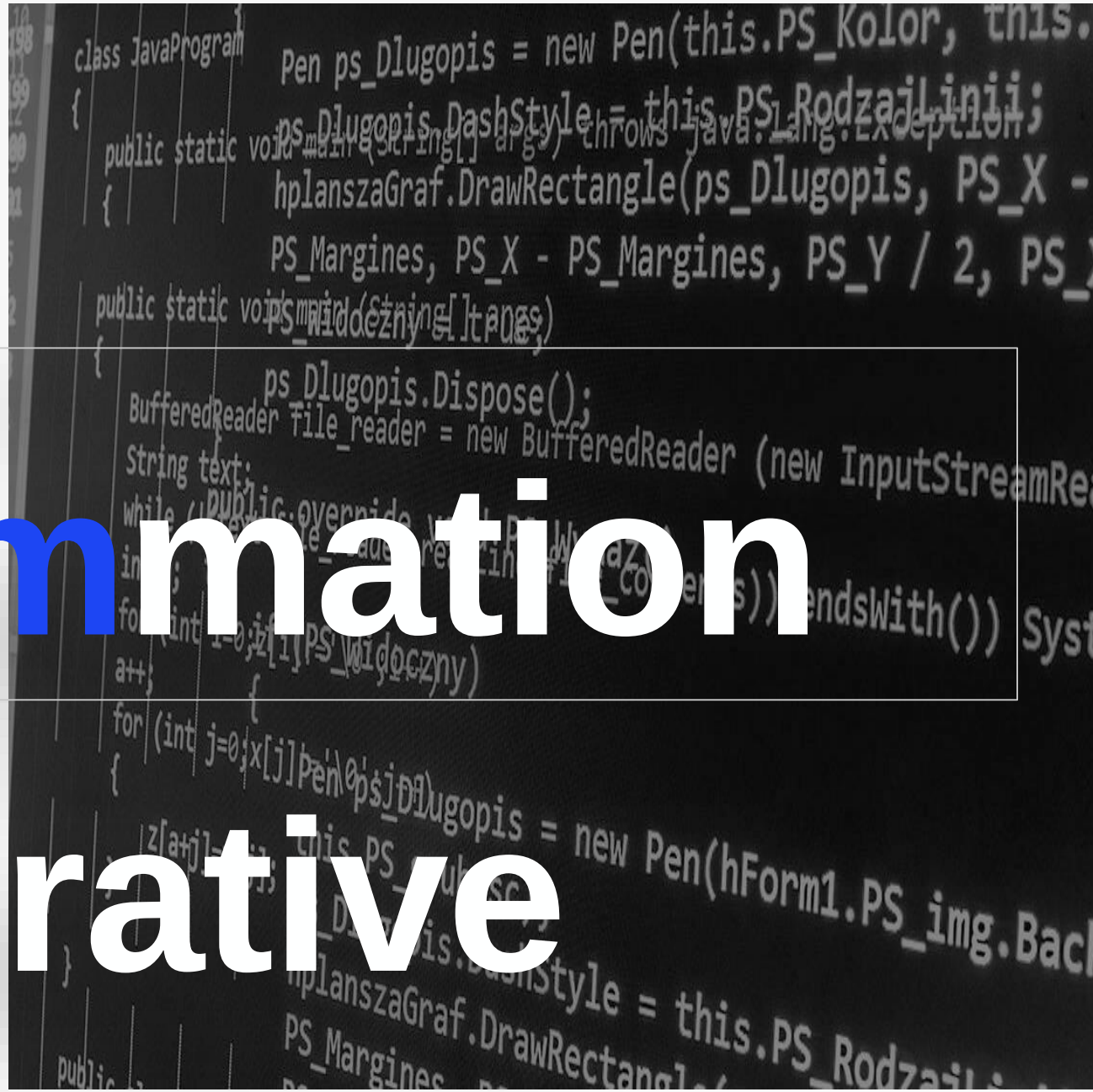
Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz

Plan du Cours

1. Programmation Impérative
2. Concepts de Programmation

Programmation Impérative



Définition et Principe

- ✓ la programmation impérative est un style de programmation qui **décrit les opérations** en termes **d'états** de programme et **des séquences** d'instructions exécutées par l'ordinateur pour modifier l'état du programme dans le but d'obtenir un résultat.
- ✓ Elle est axée sur l'**affectation**.
- ✓ L'implémentation de la quasi-totalité des microprocesseurs qui équipent les ordinateurs est de nature impérative.
- ✓ L'état du programme en un instant donné est défini par le contenu de la mémoire centrale à cet instant , et le programme est lui-même écrit en style impératif en langage machine ou une traduction d'un langage lisible (langage assembleur).

Caractéristiques

- ✓ Les programmes utilisent des structures de contrôle explicite (si, sinon, while,..) pour spécifier précisément l'ordre dans lequel les instructions doivent être exécutées.
- ✓ La raison fondamentale de l'importance de ce paradigme est reliée à la nature de la programmation.
- ✓ Enfin, ce paradigme est très important, il est dominant jusqu'à l'année 90.

Exemples

- ✓ La grande majorité des langages de programmation de haut niveau sont impératifs. Parmi ceux-ci on distingue :
  - ❑ Des langages destinés aux applications de gestion dont **COBOL**
  - ❑ Des langages destinés aux calcul scientifique dont **FORTRAN**
  - ❑ Des langages d'analyse générale : **PASCAL, ADA, C**
  - ❑ Des langages pour applications interactives : **JAVASCRIPT**

Historique

- ✓ **1951** : langage **AO** « langage machine » Le premier compilateur.
- ✓ **1954 : fortran** développé par JOHN BACKUS chez IBM.
- ✓ **1958** : **ALGOL** , père de tout langage impératif.
- ✓ **1960** : **COBOL** pour les applications de gestion.
- ✓ **1964** : **BASIC** pour un but éducatif.
- ✓ **1970** : **PASCAL** pour la programmation structurée et modulaire
- ✓ **1972** : **C** par DENUS RITCHUE pour développer les SE UNIX.
- ✓ **1985** : **C++** une extension de C par STRANSTRUP(laboratoire Bell)
- ✓ **1990** : **PYTHON** (GUIDO VIN ROSSUM)
- ✓ **1996** : **PHP** et **JAVA** (Sun MICRO-SYSTEMS)

Concepts de Programmation

Entité

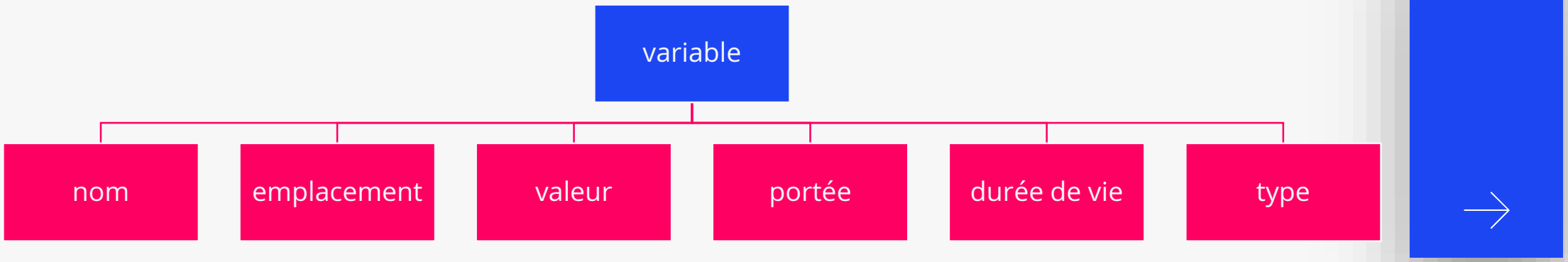
- ✓ Tout programme manipule des **entités** telles que les variables, sous programmes. Une **entité** est définie par ses **propriétés** qui sont les **attributs**.
- ✓ **Exemple** :
  - ❑ les attributs d'une **variable** : nom, location mémoire, type….
  - ❑ les attributs d'une **fonction** : nom, liste des paramètres formels,…
- ✓ La nature d'un attribut est spécifiée par une action appelée **liaison**.
  - ❑ Lors de la **définition du langage**. « statique »
  - ❑ Lors de la **compilation**.
  - ❑ Lors de l'exécution « **dynamique** »

Principe

- ✓ Les langages introduisent le concept de **variable** comme une abstraction d'une **cellule mémoire** et l'énoncé d'**affectation** comme une abstraction de la **modification de son contenu**.

Définition

- ✓ une variable est défini par :



Déclaration d'une variable

PASCAL

```
VAR longueur, largeur : real ;
    Prix, taxe, total : integer ;
    Symbole : char ;
```

C, C++, JAVA

```
float longueur, largeur ;
Int prix, taxe, total ;
Char symbole
```

FORTRAN

```
REAL longueur , largeur
INTEGER prix , taxe , total
CHARACTER symbole
```

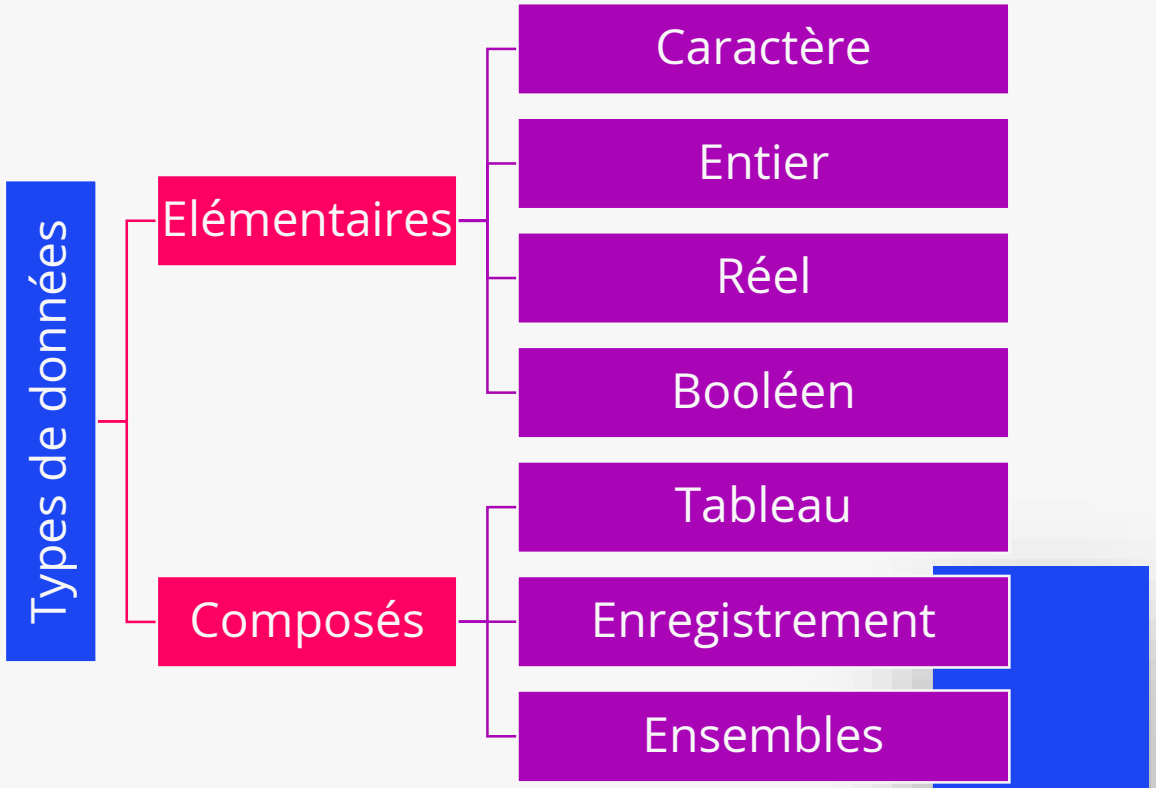
- ✓ toute variable doit être déclarée avant son utilisation avec également le type de la donnée pour définir ainsi la façon de coder la donnée et les opérations peuvent l'utiliser.

Concepts de Variable

1.Nom	• sert à identifier la variable et à s'y référer. • Se rapprocher du problème décrit par le biais du langage.
2.Emplacement	• s'exprime par une adresse où toute entité peut la localisée • L'adresse est créée lorsque l'entité est créée.
3.Valeur	• c'est le contenu de la cellule mémoire associée à la variable. • le type permet d'interpréter ce contenu et de déduire la valeur.
4.Portée	• l'étendue de l'énoncé ou la variable est connue. • l'ensemble des endroits où référence à la variable peut exister • Une variable est visible dans sa portée et invisible à l'extérieur.
5.Durée de vie	• l'intervalle de temps pour lequel un emplacement mémoire de la variable existe
6.Type	• le type d'une variable est l'abstraction qui définit l'ensemble des valeurs et des opérations associées à cette variable.

Principe

- ✓ Les langages offrent toujours un ensemble de types élémentaires et de mécanismes pour lesquels il est possible de construire des types de plus en plus complexes à partir des types élémentaires.



Types Composés

✓ Un tableau

- ❑ il est formé d'éléments homogènes ayant le même type.
- ❑ Les éléments sont identifiés par un nom dont chaque élément est accessible individuellement par un mécanisme de sélection

✓ Un enregistrement

- ❑ un agrégat de données homogènes ou hétérogènes, formé de champs identifiables par un nom.
- ❑ La sélection d'un champ particulier se fait en désignant l'enregistrement et le nom du champ.

✓ Une ensemble

- ❑ collection exhaustive des ensembles qui peuvent être formés à partir d'un type de base.

Structures de contrôle

- ✓ Les structures de contrôle d'un langage impératif sont les énoncées exécutés qui modifient l'état des variables d'un programme.
- ✓ Une phrase de contrôle altère la séquence d'exécution du programme.
- ✓ La lisibilité des programmes dépend en bonne partie du choix des structures de contrôles
- ✓ la programmation structurée est une méthodologie de programmation qui utilise une certaine famille de structures de contrôle à fin de faciliter la lisibilité et la preuve des programmes.

Structures de contrôle

✓ 1.Sélection

- ❑ Quand une série d'alternatives doit être écrites, la majorité des langages fournissent un énoncé particulier à cet effet.
- ❑ **Exemple** : if B then S1 else S2

✓ 2. Itération

- ❑ La variable de boucle forne peut pas être modifiée à l'intérieur de la boucle.
- ❑ **Exemple** : While B do S1

  Repeat S1 until B

Définition

- ✓ un sous programme est un bloc d'énoncés ayant un nom
- ✓ il s'agit encore d'un mécanisme d'abstraction car on se préoccupe de ce que fait ou réalise le sous programme plutôt du comment s'est accomplit l'opération.
- ✓ Un sous programme permet donc de traiter une opération complexe et la sémantique doit lui permettre de :
  - ❑ être appelé.
  - ❑ donner un moyen de spécifier et de recevoir des paramètres.
  - ❑ définir des variables locales.
  - ❑ définir des opérations à exécuter.
  - ❑ retourner des résultats.
  - ❑ retourner à l'appelant.

Définition

- ✓ **Sous programme**= (nom, liste des paramètres, séquence d'énoncés, environnement)
- ✓ Environnement : un ensemble des variables définies à l'exécution du sous programme mais visibles pour le sous programme qui peut les modifier lors de l'exécution.
- ✓ Il y a deux types de sous programmes :
  - ❑ **Fonctions** : sous programmes qui retournent un résultat. Equivalents à une opération.
  - ❑ **Procédures** : des sous programmes qui sont équivalents à des énoncés.