College of Computing **Université Mohammed VI Polytechnique**

TP3 - Variables et Constantes en C++

Imad Kissami

03 Février 2025

Instructions

- L'utilisation de ChatGPT est autorisée, mais chaque ligne de code doit être comprise.
- Chaque code doit être compilé via des commandes en ligne.
- Chaque TP doit être placé dans un répertoire nommé **TP[numéro]_Nom_Prénom**.
- Chaque répertoire doit contenir un fichier **README** expliquant comment exécuter les codes.
- Le fichier principal de chaque TP doit être nommé **main.cpp**.
- S'il y a d'autres fichiers en plus du fichier principal, ils doivent être nommés **nomfichier_exo[numéro].[hpp/cpp]**.
- Les codes doivent être commentés.
- Pour ce TP il faut avoir un seul fichier **main.cpp**, qui utilise les directives du préprocesseur pour choisir quel exercice compiler. Par défaut, l'exercice 1 sera compilé, mais vous pouvez définir la macro EXO lors de la compilation pour sélectionner l'exercice souhaité (par exemple, en passant **-DEXO=2** pour l'exercice 2).

Exercice 1 : Analyse de la mémoire d'un tableau

Écrire un programme qui :

- Déclare et initialise un tableau d'entiers de 10 éléments (par exemple, `int tab[10] = {0, 1, 2, ..., 9};`).
- Utilise l'opérateur `sizeof` pour afficher :
  1. La taille totale du tableau en octets.
  2. La taille d'un élément du tableau.
  3. Le nombre d'éléments du tableau (calculé en divisant la taille totale par la taille d'un élément).

Exemple de sortie :

```
Taille totale du tableau : 40 octet(s)
Taille d'un élément : 4 octet(s)
Nombre d'éléments : 10
```

Exercice 2 : Estimation pour le service de nettoyage de tapis

Écrire un programme qui :

- Demande à l'utilisateur le nombre de petites et de grandes pièces à nettoyer.
- Utilise des constantes pour définir :
  - Le tarif pour une petite pièce : $250dh$.
  - Le tarif pour une grande pièce : $350dh$.
  - Le taux de taxe de vente : 6%.
  - La validité de l'estimation : 30 jours.
- Calcule le coût total (coût des pièces, taxe et total).
- Il faut respecter la précision lors de l'affichage des variables.
- Affiche une estimation détaillée selon l'exemple suivant :

```
Estimate for carpet cleaning service
Number of small rooms: 3
Number of large rooms: 1
Price per small room: 250dh
Price per large room: 350dh
Cost : 1100dh
Tax: 66.0dh
==============================
Total estimate: 1166.0dh
This estimate is valid for 30 days
```

Exercice 3 : Calcul de la factorielle à la compilation avec `constexpr`

Écrire un programme qui :
- Déclare une fonction `constexpr` récursive pour calculer la factorielle d'un entier.
- Utilise cette fonction pour définir une constante compile-time.
- Effectue un `static_assert` afin de vérifier que la factorielle calculée est correcte.
- Affiche la factorielle calculée dans la fonction `main`.

Exercice 4 : Variables Globales et Locales (Shadowing)

Écrire un programme qui :
- Déclare une variable globale `int value = 100;`.
- Dans la fonction `main()`, déclare une variable locale avec le même nom `value` initialisée à 50.
- Affiche la valeur de la variable locale.
- Affiche la valeur de la variable globale en utilisant l'opérateur de résolution de portée (`::`).

Exemple de sortie :

```
Valeur locale : 50
Valeur globale : 100
```

Exercice 5 : Utilisation avancée des constantes

Écrire un programme qui :
- Déclare une constante avec le mot-clé `const` pour représenter le nombre de mois dans une année (12).
- Déclare une constante compile-time avec `constexpr` pour représenter le rayon d'un cercle (par exemple, 7.0).
- Utilise une macro (`#define`) pour définir une valeur de π (par exemple, 3.14159).
- Calcule et affiche l'aire d'un cercle en utilisant la formule $\pi \times r^2$.
- Tentez (en commentant la ligne) de modifier l'une de ces constantes pour observer la protection offerte.

Exemple de sortie :

```
Nombre de mois dans une année : 12
Rayon du cercle : 7
Aire du cercle : 153.9381
```

Exercice 6 : Détection d'overflow lors d'une multiplication

Écrire un programme qui :

- Implémente une fonction `safeMultiply` spécifique aux entiers (type `int`) qui multiplie deux nombres.
- Avant de réaliser la multiplication, la fonction doit vérifier, à l'aide de `std::numeric_limits`, que le résultat ne dépasse pas la capacité du type `int` (pour éviter l'overflow).
- En cas de risque d'overflow, la fonction affiche un message d'erreur et retourne une valeur spéciale (par exemple, -1).
- Dans la fonction `main`, teste cette fonction en multipliant a = 30000 par b = 1000. Puis a = 300000 et b = 10000.

Exemple de code de test :

```
int main() {
 int a = 30000;
 int b = 1000;
 int result = safeMultiply(a, b);

 return 0;
}
```