TP6 : Templates en C++

Objectifs du TP

À la fin de ce TP, l'étudiant devra être capable de :

- Comprendre le principe de la **programmation générique**
- Définir et utiliser des **fonctions templates**
- Définir et utiliser des **classes templates**
- Utiliser plusieurs paramètres de type
- Comprendre la **spécialisation de templates**
- Manipuler des templates avec la STL

Contexte

On souhaite développer une petite bibliothèque générique capable de manipuler différents types de données (int, double, char, string, etc.) sans dupliquer le code.

Partie 1 : Fonctions Templates

Exercice 1 : Fonction maximum générique

Écrire une fonction template maximum qui :

1. Prend deux paramètres du même type.
2. Retourne la valeur maximale.
3. Fonctionne avec :
    - int
    - double
    - char
    - string

Test attendu :

```
maximum(5, 10);
maximum(3.14, 7.8);
maximum('a', 'z');
maximum(string("Ali"), string("Zara"));
```

Exercice 2 : Fonction d'échange

Écrire une fonction template echanger qui échange deux variables de n'importe quel type.

Contraintes :

- Utiliser le passage par référence.
- Tester avec plusieurs types.

Exercice 3 : Template avec deux types

Créer une fonction template addition qui :

1. Prend deux paramètres de types différents.
2. Retourne leur somme.
3. Utilise le mot-clé auto pour le type de retour.

Exemple :

```
addition(5, 2.5);
```

Partie 2 : Classe Template

Exercice 4 : Classe Boite<T>

Créer une classe template Boite contenant :
Attribut :

- Une variable de type générique T

Méthodes :

- Constructeur
- getValeur()
- setValeur()
- afficher()

Test :

Créer :

- Boite<int>
- Boite<double>
- Boite<string>

Exercice 5 : Classe Pile<T>

Créer une classe template Pile qui :
Attributs :

- T* data
- int sommet
- int capacite

Méthodes :

- Constructeur
- push(T valeur)
- pop()
- isEmpty()
- isFull()
- afficher()

Contraintes :

- Utiliser allocation dynamique
- Gérer les erreurs (pile vide / pile pleine)

Partie 3 : Spécialisation de Template

Exercice 6 : Spécialisation pour le type char*

Modifier la classe Boite<T> pour créer une spécialisation lorsque T = char*.
Objectif :

- Copier correctement la chaîne
- Éviter la copie superficielle

Partie 4 : Templates et STL

Exercice 7 : Comparaison avec vector

1. Créer un vector<int>
2. Créer un vector<double>
3. Expliquer pourquoi vector est une classe template.