TD/TP04

Exercice 01 :

Créez un patron de fonction qui permet de calculer le maximum entre deux valeurs de type quelconque. Testez ce patron avec différents types de données dans un programme (par exemple int, double et char).

Exercice 02 :

Créez un **patron de fonction** qui accepte un tableau de n'importe quel type numérique (ex. int, float, double) et retourne la **moyenne** des éléments du tableau. Testez avec des tableaux d'entiers et de nombres à virgule flottante.

Exercice 03 :

Créez un **patron de classe** Pair pour stocker deux éléments de même type.

1. Implémentez un constructeur pour initialiser les deux valeurs.
2. Ajoutez une méthode display pour afficher les deux valeurs.
3. Ajoutez une méthode pour échanger les deux valeurs.

Testez avec des types int, double et string.

Exercice 04 :

Créez un **patron de classe** Box qui stocke une valeur de n'importe quel type :

1. Ajoutez un constructeur pour initialiser la valeur.
2. Implémentez une méthode getValue pour renvoyer la valeur.
3. Implémentez une méthode setValue pour changer la valeur.

Testez avec des types int, double et char.

Exercice 05 :

Implémentez un **patron de classe** Array pour gérer un tableau générique. La classe doit inclure :

1. Un constructeur qui initialise un tableau de taille spécifiée.
2. Une méthode pour **remplir** le tableau.
3. Une méthode pour afficher le contenu du tableau.
4. Une méthode pour trouver la **plus grande valeur** dans le tableau.

Testez avec des tableaux de types int et float.

Exercices 06 :

Créer une classe appelée Time, qui a des membres de type int tels que hours, minutes et secondes (rendez-les private) :

- Un constructeur doit initialiser ces données à 0
- Un autre constructeur devrait l'initialiser à des valeurs fixes.
- Une fonction membre devrait l'afficher, au format 11:59:59.
- Une autre fonction pour renvoyer les données de chaque membre nommez les getHours, getMin et getSec
- Une fonction membre doit ajouter deux objets de type Time passé en arguments.
- Rendre la fonction membre appropriée constante.

Votre classe doit fonctionner avec le code ci-dessous et affiche la sortie.

```
Int main(){
      Time t1(4, 45, 59), t2(1, 20, 32) ;
      Time t3 ;
      T1.getTime() ;          //04:45 59
      T2.getTime() ;          //01:20:32
```

TD/TP04

```
        T3.addTime(t1, t2) ;
        T3.getTime() ;          //06:06:31
}
```

Exercice 07 :

Ecrivez un programme qui simule la gestion d'un simple compte bancaire. Le compte est créé avec un solde initial. Il est possible de déposer et de retirer des fonds, d'ajouter des intérêts et de connaître le solde actuel. Cela devrait être implémenté dans une classe nommée Account qui comprend:

- Un constructeur par défaut qui met le solde initial à zéro.
- Un constructeur qui accepte une balance initial comme paramètre.
- Une fonction getBalance qui renvoie le solde actuel.
- Une méthode deposit pour déposer un montant spécifié.
- Une méthode withdraw pour retirer un montant spécifié.
- Une méthode addInterest pour ajouter de l'intérêt au compte.
- La méthode addInterest prend le taux d'intérêt comme paramètre et modifie le solde du compte en solde * (1 + taux d'intérêt).

Votre classe doit fonctionner avec le code ci-dessous et affiche la sortie.

```
int main() {
        Account account1;
        Account account2(3000.0);
        account1.deposit(100);
        account2.withdraw(1000) ;
        account1.addInterest(0.3);
        cout << "Solde de account1 : " << account1.getBalance() << "\n";
        cout << "Solde de account2 : " << account2.getBalance() << "\n";
        return 0;
}
```

Exercice 08 :

Écrivez un programme avec une classe mère A et une classe fille B. Les deux doivent avoir une méthode ***void display()*** qui affiche un message (différent pour la classe mère et la classe fille). Dans la méthode principale créer un objet de la classe fille et appelez la méthode display() sur elle.

Exercice 09 :

Écrivez un programme qui définit une classe appelée Shape avec un constructeur qui donne de la valeur à la largeur(x) et à la hauteur(y). Définir la méthode area() dans les deux sous-classes Triangle et Rectangle, qui calculent l'aire. Dans la méthode principale main, définissez deux variables, un triangle et un rectangle, puis appelez la fonction area() dans ces deux variables.

Notez que l'aire du triangle est = largeur * hauteur / 2 et l'aire du rectangle est = largeur * hauteur.

Votre classe doit fonctionner avec le code ci-dessous et affiche la sortie.

```
int main() {
        Rectangle rectangle(2, 3); // Largeur = 2, Hauteur = 3
        Triangle triangle(2, 3); // Largeur = 2, Hauteur = 3
        cout << rectangle.area() << endl; // Affiche 6
        cout << triangle.area() << endl; // Affiche 3
        return 0;
}
```