

TP n°5

Classes

Note : ces exercices nécessitent la lecture des supports des onze premiers cours que l'on peut retrouver à l'adresse <http://www.liafa.univ-paris-diderot.fr/~yunes/cours/cpp/>

Exercice 1 (Implémentation d'un stack)

La classe **Stack** est définie comme suit :

```
template <class T> class Stack
{
public:
    Stack(int = 10);
    ~Stack() ;
    int push(const T&);
    T pop();
    int isEmpty() const    { return top == -1; }
    int isFull()  const    { return top == size - 1; }
private:
    int size;
    int top;
    T* stackPtr;
};
```

Implémentez les fonctions manquantes et testez les types **Stack<int>** et **Stack<char>**.

Exercice 2 (Trier des personnes)

La fonction **bubbleSort** est définie comme suit :

```
template<class T> void bubbleSort(T *pT, int size)
{
    for(int l1=0; l1 < size-1; l1++)
        for(int l2=l1+1; l2 < size; l2++)
        {
            if(pT[l1] > pT[l2])        // echanger pT[l1] et pT[l2]
            {
                T tmp = pT[l1];
                pT[l1] = pT[l2];
                pT[l2] = tmp;
            }
        }
}
```

1. Utilisez **bubbleSort** pour trier un tableau d'entiers.
2. Ecrivez une classe **Personne** qui a un prénom et un nom. Triez un tableau de **Personne** par ordre alphabétique des noms (si deux personnes portent le même nom prenez leurs prénoms en compte) en utilisant **bubbleSort** (astuce : surcharge d'opérateur).

Exercice 3 (Une liste de fonctions)

Premièrement définissez une classe `Fonction` avec une méthode virtuelle pure `Integration()` qui renvoie la valeur de l'intégrale entre 0 et 1 et une méthode virtuelle `string Fonction::toString()` qui renvoie le nom de la classe comme string. Ecrivez des sous-classes `Constante` (qui représente la fonction $f(x) = c$ pour une constante c), `Lineaire` (qui représente une fonction $f(x) = ax + b$ pour des constantes a et b) et `Sinus` (qui représente la fonction $\sin(x)$) de la classe `Fonction` et implémentez leurs méthodes `Integration()` et `toString()`.

Deuxièmement implémentez une classe `template<class T> class Liste` permettant de stocker une liste d'objets de type `T`. Le constructeur prend la taille maximale de la liste. La classe `Liste` possède les méthodes suivantes :

- une méthode `affiche()` qui affiche la liste des éléments de la liste,
- une méthode `ajoute(T t1)` qui ajoute `t1` dans la liste,
- une méthode `recherche(T & t1)`