# Structures de données et algorithmes – TP3

Andreea Geamanu

# Objectifs pour aujourd'hui

| |
|---|
| Fonctions de type “template” |
| Classes de type “template” |
| Piles (stack) – definition |
| Utilisation des headers |
| Applications avec piles |
| Info: Méthodes de tri |

# Support théorique

# Templates

- En français, template = modèle, patron
- Programmation générique
- Permettent d’écrire du code sans connaître exactement le type des données

# A. Fonctions “template”

- Une fonction de type template supporte n’importe quel type (T) pour ses arguments
- Syntaxe:

```
template <typename T>  T nomFonction (T a,
T b)
{
        // le code
}
```

- typename peut être remplacé avec class (même effet)

# A. Fonctions “template”

- Ex:

```
template <typename T>  T maxim(T a, T
 b) {
     return a > b ? a : b; //si a>b on
      retourne a, sinon, b
// meme que: if (a>b) return a; else
 return b;
 }
```

- Appel:
  - maxim (10, 15);
  - maxim (1.3, 2.4);  ou:
  - maxim<int> (10, 15);

➢ Ex. fonctions

template (std):

- std::min, std::max
- std::count, std::sort etc.

# B. Classes “template”

- Une classe générique, qui supporte n’importe quel type (T) pour ses membres
- Syntaxe:

```
template <typename T>  class
nomClasse {
        // le code
};
```

- Création objets:
  - nomClasse <type> nomObjet (…);

# B. Classes “template”

```
template <class T>
class mypair {
        T values [2];  public:
                mypair (T first, T
                second)
                {
                        values[0]=first;
                        values[1]=second;
                }
};
```

On crée des objets:

- mypair <int> objet (25, 13);

# B. Classes “template”

- Exemple avec plusieurs champs:

```
template<typename Type1, typename Type2>
class KeyValue
{
public:
    int key;
    Type1 value1;  Type2 value2;
    //constructeur avec les 2
   champs
};
```

- On crée des objets:
  - KeyValue <int, char> obj
    (12, ‘c’);

# Exercice 1.

- Modifier la classe Point du TP précédent en utilisant les templates (les coordonnées x et y étant du type T). Testez la nouvelle classe en utilisant des paramètres float et int, respectivement .

# Info – Les méthodes de tri

1. Tri a bulles

Principe: Comparer deux valeurs adjacentes et inverser  leur position si elles sont mal placées.

```
bool ok = false;  int n = tab.size();  while(!ok)
{
   ok = true;
   for(int i=0 ; i < n-1; i++)
   {
      if(tab[i] > tab[i+1])
      {
        // swap(tab[i],tab[i+1]);  ok = false;
      }
   }
   n--;
}
```

6 5 3 1 8 7 2 4

# Info – Les méthodes de tri

## 2. Tri par sélection

Principe: rechercher le plus petit élément, le placer en  début du tableau, recommencer avec le second plus  petit, le placer en seconde position et ainsi de suite  jusqu'à avoir parcouru la totalité du tableau.

```
for (int i=0; i<n-1; i++)
{
        int minIndex = i;  for(int j=i+1; j<n; j++)
        {
                if(tab[j]<tab[minIndex])
                {
                        minIndex= j;
                }
        }
        if (minIndex != i)
                //swap (tab[i], tab[minIndex]);
}
```

| 8 | 5 | 2 | 6 | 9 | 3 | 1 | 4 | 0 | 7 |
|---|---|---|---|---|---|---|---|---|---|

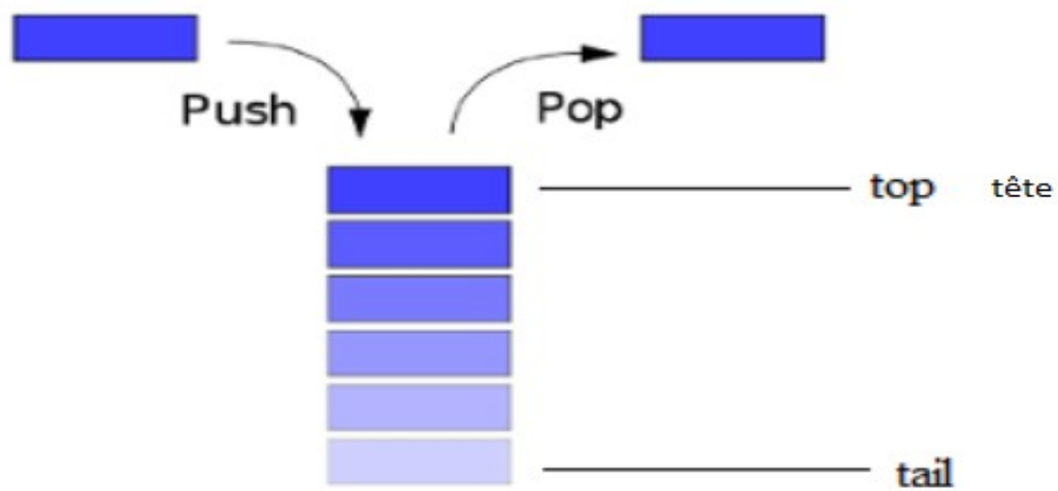
# Exercice 2.

Ecrire une fonction template pour trier un tableau de 5 éléments. Ecrire une autre fonction template pour le « swap » (changement des deux valeurs du tableau). Lire les valeurs du tableau du clavier.

Rappel déclaration tableau: int a[5];

# Pile (stack)

- Instance d’un type de données abstraites  (ADT)
- Une collection d’éléments,  basée sur le  modèle LIFO (last in, first out)



# Opérations de base

- push(x): ajoute l'élément x à la tête de la pile
- pop(): supprime l'élément qui se trouve en haut de la pile et l'affiche; renvoie une erreur si la pile est vide
- peek(): renvoie (mais ne supprime pas) l'élément de la tête de la pile
- isEmpty(): renvoie 1 si la pile est vide et 0 sinon

A voir l'implémentation des fichiers source! (moodle)

# Exercice 3.

a) Testez l'implémentation avec tableau du stack (moodle) Appelez les différents fonctions (pop, push, peek).

b) Définissez la classe Stack dans un fichier de type header (e.g. mystack.h) et testez-le dans un autre fichier qui contient le main et inclue le fichier header:

```
#include "mystack.h"
```

- Ecrivez un programme qui utilise l' implémentation du stack, lit du clavier un nombre n, ensuite n nombres de type double et les affiche dans l'ordre inverse de la lecture.
- Attention! On ne peut pas modifier la structure du stack (le header), on doit obtenir cette tache dans le programme principal.

# Exercice 4.

- Mettez en oeuvre une classe appelée LargeStack qui peut stocker des valeurs arbitraires d'un type T (utiliser class template pour le type). La classe a deux variables internes (attributs):

  Stack<T> Smain, Saux;

- Smain est la pile principale qui permet de stocker les valeurs ajoutées dans LargeStack. Saux est une pile auxiliaire qui doit être vide avant et après l'appel d'une fonction de la classe LargeStack. (Saux utilisée seulement pour des opérations internes)
- La classe LargeStack a les fonctions suivantes:
  - void push(T x): ajoute l'élément x en haut de la pile Smain.
  - T pop() : supprime et retourne l'élément à partir du haut de la pile Smain
  - void swap(int i, int j): échange les valeurs des niveaux i et j de la pile Smain (Les niveaux sont numérotés à partir de 0 (niveau 0 est le niveau le plus bas)).

HINT: Vous pouvez utiliser la pile auxiliaire Saux pour stocker temporairement les éléments du Smain. Vous pouvez utiliser toutes les methodes de la classe Stack (pop,push,isEmpty, etc) pour les variables Smain et Saux.

# Exercices extra

L’utilisateur introduit une suite de caractères du clavier. A l’aide de la pile, on doit vérifier si la suite est palindrome ou non. (e.g.: rever, tot)

Attention aux espaces, pax example
“a santa at nasa”
and
“a nut for a jar of tuna”
sont palindromes.

# Exercices extra

(exo 2 suite) HINT: (comment on crée un tableau de caractères)

```
#include <iostream.h>  #include
<string.h>
using namespace std;  int main(){
        string s ="Hello";
        char suite[10];
        for(int i=0; i<s.length(); i++){
                suite[i]=s[i];  cout<<suite[i]<<" ";
        }
}
```