Data structures and algorithms- Lab 3

Iulia-Cristina Stanica
iulia.stanica@gmail.com

Roadmap

- Function templates
- Class templates
- Sorting algorithms

Templates

- Generic programming
- Allows a function or class to work on many different data types without being rewritten for each one

A. Function templates

- A function template allows any type of data (T) for its arguments
- Syntax:

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

- **typename** can be replaced with **class**

A. Function templates

- Ex:

```
template <typename T>
 T maxim(T a, T b) {
      return a > b ? a : b; //if a>b return a, else, b
// equivalent to: if (a>b) return a; else return b;
 }
```

- Function call:
  - maxim (10, 15);
  - maxim (1.3, 2.4);

  or:
  - maxim<int> (10, 15);

➢ Ex. Templates (std):
- std::min, std::max
- std::count, std::sort etc.

B. Class templates

- A generic class, which allows any type (T) for its members
- Syntax:

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

- Create objects:
  - className <type> object (…);

B. Class templates

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

Create objects:

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

B. Class templates

- Example with 2 fields:

```
template<typename Type1, typename Type2>
class KeyValue
{
public:
    int key;
    Type1 value1;
   Type2 value2;
   //constructor with 2 fields
};
```

- We create objects:
  - KeyValue <int, char> obj (12, ‘c’);

Sorting methods

1. ### Bubble sort

Method: Compare each two adjacent items and swap their positions if they are in the wrong order.

```
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

Sorting methods

2. Selection sort

Method: search for the smallest element, place it on the first position of the array, restart with the second smallest element, place it on the second position etc., until we run through the whole array.

```
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]);
}
```

6   5   3   1   8   7   2   4

Exercises

1. Write a function template to sort an array of 5 elements. Write a different function template for swapping 2 values of the array. Read the values from keyboard.

Array declaration: int a[5];

2. Create a template class to represent a matrix with elements of type T.

Implement the functions:

```
      void printMatrix();
      void setElement(int row, int col, T value); //set
an element of the matrix
      void addMatrix(Matrix m);
```

Homework

1. Change the class Point from the previous homework by using templates (the x and y coordinates are of type T).

2. Search for a new sorting method (not bubble sort or selection sort) and implement it in C++.

By using the class Point from the previous homework, create a function which sorts 5 points using the previously chosen sorting method. You should take as a sorting criteria the distance of the point from the origin.