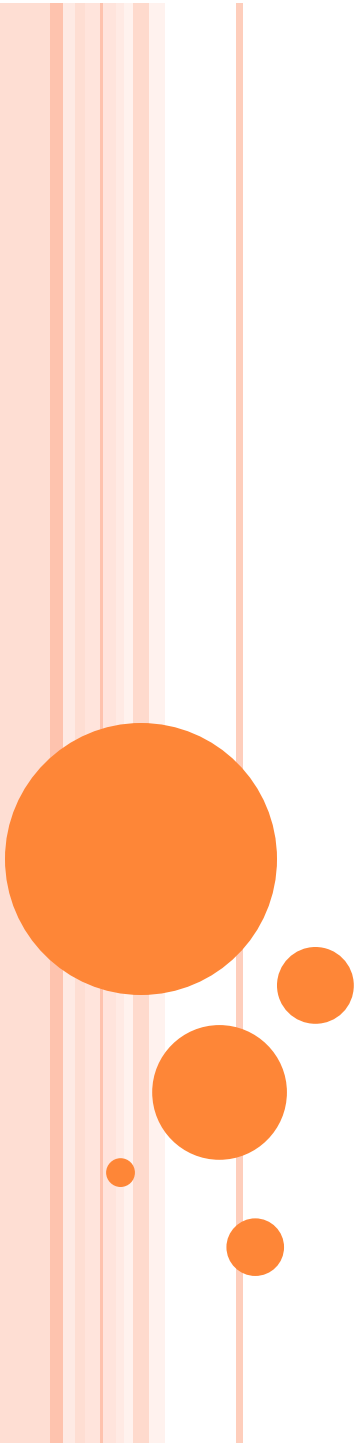

The Standard Template Library (STL – Part 2)

CS 250 – C++ Programming 2

More Containers

- This time we will look at:
  - Class **pair**
    - Used by **map** and **multimap**
  - **Associative containers**
    - **set** and **multiset**
    - **map** and **multimap**

CLASS pair

- The class **pair** combines **two values** in a single unit
- Every object of type pair has **two member variables**:
  - **first** and **second**
  - Can be **different types**
  - Both **member variables** are **public**

> - This means that **first** and **second** can be accessed **without** using an accessor function.

- Need to include `<utility>`

CLASS pair (cont.)

- The class pair has 3 constructors:
  - The **default constructor**:

```
pair<T1,T2> pairObj;
```

  - An **overloaded constructor** with two parameters:

```
pair<T1,T2> pairObj(T1,T2);
```

  - A **copy constructor**:

```
pair<T1,T2> pairObj(otherPairObj);
```

CLASS pair (cont.)

- Note that the class **pair** can have **different types** for the pair.

```
#include <utility>
...
void someFunction()
{
    pair<int,double> pair1(3, 5.4);

    cout << pair1.first << " " << pair1.second << endl;
}
```

```
OUTPUT: 3 5.4
```

CLASS pair (cont.)

- Another example

```
class MyClass
{
public:
        …
private:
        string str;
        double d;
};
```

```
#include <utility>
...
void someFunction()
{
    pair<int,MyClass> pair1;  // default values

    cout << pair1.first << " " << pair1.second << endl;
}
```

The program will crash. Why?

CLASS pair (cont.)

- Another example

```
class MyClass
{
public:
        …
private:
        string str;
        double d;
};
```

```
#include <utility>
...
void someFunction()
{
    pair<int,MyClass> pair1;  // default values

    cout << pair1.first << " " << pair1.second << endl;
}
```

Because the **insertion operator** needs to be **overloaded** in the class **MyClass** to print an object of that class.

CLASS pair – ANOTHER EXAMPLE

```
#include <utility>
...
pair<int,double> p1;
p1.first = 3;
p1.second = 4.0;

pair<int,double> p2(13, 45.9);
pair<string,int> student("Bob", 1234);

cout << student.first;    //Output: Bob
cout << student.second;   //Output: 1234
```

values can be assigned

member variables are public

Associative Containers

- **Associative containers**
  - ***Automatically* sorted**
  - Default **ordering criterion**
    - The relational operator **<** (less than)
    - **Ascending** order
    - Can be changed to other criteria
- STL **associative containers**:
  - **Sets** and **multisets**
  - **Maps** and **multimaps**

Sets and Multisets

- The STL **set** and **multiset** classes ***automatically*** **sort** their elements according to some criteria
  - By **default**, the sorting is done in **ascending order**
  - But it can also be specified according to a different sorting criterion

- The only difference between **sets** and **multisets** is that **multiset** allows **duplicates**

Maps and Multimaps

- The STL **map** and **multimap** classes manage their elements in the form **key/value** (a given ordered pair)
  - The elements are ***automatically*** **sorted** according to some sort criteria applied on the **key**
  - By **default**, the sorting is done in **ascending order**
    - But it can also be specified according to a different sorting criterion

- The only difference between **maps** and **multimaps** is that a **multimap** allows **duplicates**

How to Initialize a Map

- Using **insert** and **make_pair**

```
map<int,int> intMap;

for (int i = 1; i < 10; ++i)   // insert integers
      intMap.insert( make_pair(i, (100 / i)) );

map<int,int>::const_iterator it = intMap.cbegin();
map<int,int>::const_iterator itEnd = intMap.cend();

for (it; it != itEnd; ++ it)
      cout << it->first << " " << it->second << endl;
```

The **map** will contain the following elements:

{ (1, 100), (2, 50), … (8, 12), (9, 11) }

How to Initialize a Map (cont.)

- Using an **initializer list**

```
map<int, string> aMap = {
        {1, "one"},
        {2, "two"},
        {3, "three"},
        {4, "four"}
};

map<int, string>::const_iterator it = aMap.cbegin();
map<int, string>::const_iterator itEnd = aMap.cend();

for (it; it != itEnd; ++ it)
      cout << it->first << " " << it->second << endl;
```

How to Initialize a Map (cont.)

- A pair of elements at a time

```
map<char, int> aMap;

aMap['A'] = 90;
aMap['b'] = 80;
aMap['C'] = 70;
aMap['D'] = 60;

map<char, int>::const_iterator it = aMap.cbegin();
map<char, int>::const_iterator itEnd = aMap.cend();

for (it; it != itEnd; ++ it)
      cout << it->first << " " << it->second << endl;
```

Constructors

- Project: 10_stl_2_files

STL 2 (END)