Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session : Concluding Comments on C++ Standard Library

Quick Recap of Relevant Topics

IIT Bombay

- C++ Standard Library
  - The “string” class
  - The “vector” class
  - The “map” class
  - The “list” class

Overview of This Lecture

- Comments about passing container objects as function parameters/return values
- Use of "typedef" to simplify complex container class definitions
- Overview of some useful C++ Standard Library modules

Acknowledgment

- Some parts of this lecture are motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

Using Container Classes in Function Calls

Passing container objects as parameters by value:

void func1(vector<int> v1, map<string, int> m1)

- Vector v1 and map m1 must be copied to activation record of func1
- Copy constructor of vector<int> and map<string, int>used for copying values
- Default copy constructor copies values of all data members

Can be highly inefficient (both time- and memory-wise) if large container objects passed as parameters

Using Container Classes in Function Calls

Passing container objects as parameters by reference:

void func1(vector<int> &v1, map<string, int> &m1)

No copying of data member values to activation record of func1.

Time and memory overhead independent of size of parameter container object

Using Container Classes in Function Calls

If parameters passed by reference must be prevented from being modified by callee

```
void func1(vector<int> const &v1,
           map<string, int> const &m1)
```

Container objects can have huge memory footprints. Parameter passing mechanism must be chosen with discretion.

Using Container Classes in Function Calls

map<string, double> func2(vector<int> const &v1)

Result map computed by func2 must be copied to activation record of caller when func2 returns

Copy constructor can be inefficient if map is large

void altFunc2(vector<int> const &v1, map<string, double> &res)

No copying of result map needed – much more efficient in practice

Using Simple Type Names in Container Classes

- Declarations like

  vector<map<string, map<double, list<V3> > > > myVec;

  difficult to read, understand, modify

- C++ provides **typedef** to give custom names to types

```
typedef   list<V3>  listOfV3;
typedef   map<double, listOfV3> mapDoubleListOfV3;
typedef   map<string, mapDoubleListOfV3> myType;
typedef   vector<myType> vecOfMyType;
vecOfMyType myVec;
```

C++ Standard Library: Additional Useful Modules

- Several useful container class libraries
  - queue
  - deque
  - stack
  - set
  - forward_list
  - …

C++ Standard Library: Additional Useful Modules

- Several other useful libraries of utilities
  - algorithm
  - complex
  - exception
  - random
  - memory
  - …
- Several excellent online references
  - http://www.cplusplus.com/reference
  - http://en.wikipedia.org/wiki/C++_Standard_Library

Summary

- Use of container classes in function calls
- Use of “typedef“ to simplify complex container class definitions
- High-level view of useful modules in C++ Standard Library
- Strongly encouraged to read more about C++ Standard Library and use it in your programs
  - Extremely well-designed, thoroughly tested and well-documented