

Computer Programming

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

Session : C++ Standard Library – The “string” Class

Quick Recap of Relevant Topics



- Object-oriented programming with structures and classes
 - All classes seen so far were custom designed by us
- Template classes and functions

Overview of This Lecture



- C++ Standard Library
 - Collection of very useful classes that come with all C++ distributions
- The “**string**” classes

Acknowledgment



- Much of this lecture is motivated by the treatment in
An Introduction to Programming Through C++
by Abhiram G. Ranade
McGraw Hill Education 2014

C++ Standard Library



- Some classes and functionalities very commonly used across a wide variety of applications
 - string, vector, list, queue, stack, priority_queue, set, map, ...
- Instead of each user defining these separately, C++ provides implementations of these as part of every distribution
- **C++ Standard Library**
 - **Collection of commonly used classes and functions**
 - **Part of C++ ISO Standard**
 - **Uses the best algorithms, are extensively tested, uses good dynamic memory management, ...**

C++ Standard Library



- All features declared within “**std**” namespace
 - Saying “**using namespace std;**” at the beginning of your program takes care of this
- Must include appropriate header file in program to access standard library features
 - **#include <string>**
 - **#include <vector>**
- Complete library fairly large
 - We’ll study only a few features in lectures
 - Handout contains list of other classes in library
 - Strongly encouraged to use Standard Library classes in your programs

C++ Standard Library (To Be Studied)



- **string** class
 - For storing and manipulating strings without worrying about internal representation
- Container classes
 - Holder of a collection of objects of another class (generic type)
 - Usually implemented as template classes
 - **vector** class
 - **map** class
 - **list** class
- Many more interesting classes ... not covered in lectures

The “string” class



- For representing and manipulating strings
- Must use **#include <string>** at start of program
- Large collection of member functions
 - We'll see only a small subset
- Some non-member functions (e.g. **operator+**, **getline**) also defined and very useful

Simple Programming using “string” Class

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Hello ";
    string s2 = "world! ";
    string s3 = s2;
    cout << s1 << " " << s3 << endl;
    return 0;
}
```

Hello world!

Reading Input Strings: **cin**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Please type in your name: ";
    string s2;
    cout << s1;
    cin >> s2;
    cout << "Your name is: " << s2 << endl;
    return 0;
}
```

Ajit Trivedi

Your name is: Ajit

Reading Input Strings: **getline**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Please type in your name: ";
    string s2;
    cout << s1;
    getline(cin, s2); Ajit Trivedi
    cout << "Your name is: " << s2 << endl;
    return 0; Your name is: Ajit Trivedi
}
```

Concatenating strings: **operator+**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Hello ";
    string s2 = " world";
    string s3 = s1 + s2 + "!!!";
    cout << s3 << endl;
    return 0;
}
```

Hello world!!!

Concatenating strings: **append**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s; string s1 = "Hello ";
    string s2 = " my world!!!";
    s.append(s1);      Hello .
    s.append(s2, 4, 5); Hello world
    s.append("!!!");  Hello world!!!
    cout << s << endl;
    return 0;
}
```

Accessing Characters: **operator[]**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Hello world!!!";
    if ((s1[6] == ' ') && (s1[7] == ' ')) {
        s1[6] = 'm'; s1[7] = 'y';
    }
    cout << s1;
    return 0;
}
```

Hello my world!!!

Accessing Characters: **at**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Hello world!!!";
    if ((s1.at(6) == ' ') && (s1.at(7) == ' ')) {
        s1.at(6) = 'm'; s1.at(7) = 'y';
    }
    cout << s1;
    return 0;
}
```

s1[100] vs s1.at(100):
Illegal memory/garbage
access
vs
out_of_range exception

Finding a Substring: **find** and **rfind**

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Hello world Hello!!!";
    int i = s1.find("Hello");
    int j = s1.find("Hi");
    cout << i << ", " << j << endl;
    return 0;
}
```

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s1 = "Hello world Hello!!!";
    int i = s1.rfind("Hello");
    int j = s1.rfind("Hi");
    cout << i << ", " << j << endl;
    return 0;
}
```

string::npos A value that can never be a valid index

Extracting Substrings: **substr**



```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {
```

```
    string s1 = "Hello world Hello!!!";
```

```
    cout << s1.substr(6) << " " << s1.substr(0, 5) << endl;
```

```
    return 0;
```

```
}
```

world Hello!!!. Hello

C++ Iterator



- An object that points to an element in a collection of elements, and can be used to iterate through the elements in the collection
- Like a pointer, but not exactly the same
- Must support ++ (increment) and * (dereference) operations

Iterator Related Functions in “string” Class

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {
```

```
    string s1 = “Hi there!”;
```

```
    for (string::iterator it = s1.begin(); it != s1.end(); it++) { cout <<  
    *it;}
```

```
    return 0;
```

```
}
```

begin(), end()
member functions

Hi . there!

Iterator Related Functions in “string” Class

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int main() {
```

```
    string s1 = “Hi there!”;
```

```
    for (string::reverse_iterator rit= s1.rbegin(); rit != s1.rend(); rit++) {
```

```
        cout << *rit;
```

```
    }
```

```
    return 0;
```

```
}
```

rbegin(), rend()
member functions

!ereht iH

Summary



- C++ Standard Library
- “**string**” class and its usage
 - Many more member functions exist
 - Encouraged to read them up on your own