CS 101: Computer Programming and Utilization

Jan-Apr 2017

Sharat
(piazza.com/iitb.ac.in/summer2017/cs101iitb/home)

Lecture 9A: String

About These Slides

- Based on Chapter 3 of the book
  An Introduction to Programming Through C++
  by Abhiram Ranade (Tata McGraw Hill, 2014)

- Original slides by Abhiram Ranade
  –First update by Varsha Apte
  –Second update by Uday Khedker
  –Third update by Sunita Sarawagi

Announcements

1. Individual inlab June 16
2. Final Exam: Monday June 26 2pm
3. Viva slots: On demand

The Standard Library

- Comes with every C++ distribution
- Contains many functions and classes that you are likely to need in day to day programming
- The classes have been optimized and debugged thoroughly
- If you use them, you may be able to write programs with very little work
- Highly recommended that you use functions and classes form the standard library whenever possible
- Today the “string” object
- You will need: http://www.cplusplus.com/reference/string/

Outline

- Motivation:
  - Recall that the “cin” constructs eats white spaces
  - More important, we want to deal with a complete sequence of characters
  - For example, genes are made by the sequence of characters from the alphabet ‘ACTG’
- Goal
  - Read “Hello world” (possibly mistyped as “hello.world”) and output as Hello World

The String Class: Reading

- `#include <string>`
- `string p`
- `getline(cin, p)`
- This will read a complete line (including leading spaces, and any spaces in between up to the newline)
- But first we will consider cases when we do not have space
- A string object is a sequence of characters, so there is a good relation between a string object and a char object

The String Class: Manipulating

- Suppose the user types in ‘virAt’ and we want to output “Virat”
  - How would we access the content
- Once a string object is read or made available
  - We access the members of the object using the square bracket syntax
  - p[0] is the first character, it’s not p[1]
  - Programmers start counting from zero not 1
- We manipulate just as we would manipulate a character

Examples

```
#include <string>      // Needed to use the string class
string v = “abcdab”;   // constructor
string w(v);           // another constructor. w = v
v[2] = v[3];   // indexing allowed. v becomes “abddab”
```

Valid Characters in Name

- Suppose the user types in ‘vir+t’ and we want to output “Virat”
  - How would we check that it is not a valid name?
- The isalpha() construct in the package cctype

Demo

Valid Characters: Alternative

- Suppose the user types in ‘vir+t’ and we want to output “Virat”
  - How would we check that it is not a valid name?
- We are going to use the powerful “find” syntax

Examples

```
#include <string>      // Needed to use the string class
string v = “abcdab”;   // constructor
int i = v.find(“ab”);      // find occurrence of “ab” in v
                           // and return index
int j = v.find(“ab”,1);   // find from index 1
cout << i << “, “ << j << endl; // will print out 0, 4.
v.find_first_of(“cd”)
// find the first occurrence of any character of “cd” in v
// substring operation
```

Remarks

- If the find member function does not find the argument in the receiver, then it returns a constant string::npos, which is a value which cannot be a valid index
  - You can determine whether the argument was found by checking whether the returned index equals string::npos
- string s; s.size() or s.length() returns a value of type size_t (a redefinition of unsigned int)
-

Demo

Examples

```
#include <string>      // Needed to use the string class
string v = “abcdab”;   // constructor
cout << v.substr(2)  << v.substr(1,3)  << endl;
          // substring starting at v[2] (“cdab”)
          // Substring starting at v[1] of length 3 (“bcd”)
string longerString = v + ‘ ‘ + World // the + operator
```