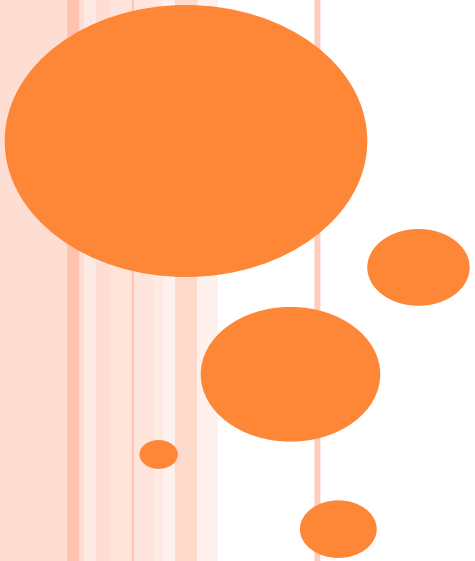


CSE 107: Object Oriented Programming Language

Dr. Tanzima Hashem
Professor
CSE, BUET

String

- A string is a sequence of character

- We have used null terminated <char> arrays (C-strings or C-style strings) to store and manipulate strings
  - We include <cstring> to use C-style string functions, such as strlen, strcpy etc

- C++ provides a class called **string**
  - We include <string> in our program to create objects of string class

String

- Reason for including string class in C++ library
  - Consistency
    - A string now defines a data type
  - Convenience
    - Use of standard C++ operator
  - Safety
    - Array boundaries will not be overrun

Available Operations

- Creating string objects
- Reading string objects from keyboard
- Displaying string objects to the screen
- Finding a substring from a string
- Modifying string objects
- Adding string objects
- Accessing characters in a string
- Obtaining the size of string
- And many more

Commonly Used String Constructors

- String();
  // For creating an empty string.
- String(const char *str);
  // For creating a string object from a null-terminated string
- String(const string &str);
  // For creating a string object from other string object

Operator Overloaded in string Class

Operator	Meaning
=	Assignment
+	Concatenation
+=	Concatenation assignment
==	Equality
!=	Inequality
<	Less than
<=	Less than equal

Operator Overloaded in string Class

Operator	Meaning
>	Greater than
>=	Greater than equal
[ ]	Subscripting
<<	Output
>>	Input

Creating String Objects

- string s1, s3; // Using constructor with no arguments
- string s2("xyz"); // Using one-argument constructor
- string s4(s1) // Create a string object with another string object
- s1 = s2; // Assigning string objects
- cin >> s1; // Reading from keyboard (one word)
- cout << s2; // Display the content of s2
- getline(cin, s1) // Reading from keyboard a line of text
- s3 = "abc" + s2; // Concatenating strings
- s3 += s1; // s3 = s3 + s1;
- s3 += "abc"; // s3 = s3 + "abc";

Manipulating String Objects

- assign
  - string &assign(const string &strob, size_type start, size_type num)
    - string str;
    - string base="The quick brown fox jumps over a lazy dog.";
    - str.assign(base,10,9);
    - "brown fox"
  - string &assign(const char *str, size_type num)
    - string str;
    - str.assign("pangrams are cool",7);
    - "pangram"

Manipulating String Objects

- append()
  - string &append(const string &strob, size_type start, size_type num)
    - string str;
    - str3="print 10 and then 5 more“
    - str.append(str3,6,3);
    - "10 "
  - string &append(const char *str, size_type num)
    - string str;
    - str.append("dots are cool",5);
    - "dots "

Manipulating String Objects

- insert()
  - string &insert(size_type, start, const string &t b)
    - string str="to be question";
    - string str2="the ";
    - str.insert(6,str2);
    - “to be the question”
  - string &insert(size_type, start, const string &strob, size_type inStart, size_type num)
    - string str="to be question";
    - str3="or not to be";
    - str.insert(6,str3,3,4);
    - “to be not question”

Manipulating String Objects

- replace()
  - string &replace(size_type start, size_type num, const string &strob)
    - string str="this is a test string.";
    - string str2="n example";
    - str.replace(9,5,str2);
    - "this is an example string."
  - string &replace(size_type start, size_type num, const string &strob, size_type replaceStrat, size_type replaceNum)
    - string str="this is an example string.";
    - string str3="sample phrase";
    - str.replace(19,6,str3,7,6);
    - "this is an example phrase."

Manipulating String Objects

- erase()
  - string &erase(size_type start=0, size_type num=npos) //npos= -1
  - string str ("This is an example sentence.");
  - str.erase (10,8);
  - "This is an sentence."

Manipulating String Objects

- string s1("12345");
- string s2("abcde");

// s1 = 1234abcde5

- s1.insert(4, s2);

// s1 = 12345

- s1.erase(4, 5);

// s2 = a12345e

- s2.replace(1, 3, s1);

Relational Operations

- string s1("ABC"); string s2("XYZ");

- int x = s1.**compare**(s2);
  - x == 0 if s1 == s2
  - x > 0 if s1 > s2
  - x < 0 if s1 < s2

More functions…

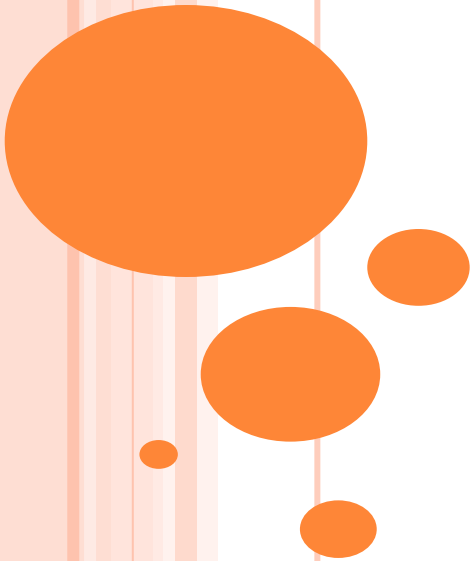
- size_type find(const string &strob, size_type start=0) const
  - Beginning at start, searches the invoking string for the first occurrence of the string contained in strobe
- size_type rfind(const string &strob, size_type start=npos) const
  - Beginning at start, searches the invoking string in the reverse direction for the first occurrence of the string contained in strobe
- int compare(size_type start, size_type num, const string &strob) const
  - Compare num characters in strobe, beginning at start against the invoking string
- const char *c_str() const
  - Returns c-string contained in the invoking string object
  - You might use a string object to construct a filename but when you open a file you will need to specify a pointer to a standard, null-terminated string
- void swap(string & strob)
  - Exchanges the content of the invoking stringby the content of strobe

String Characteristics

```
#include <iostream>
#include <string>

int main ()
{
  std::string str ("Test string");
  std::cout << "size: " << str.size() << "\n";
  std::cout << "length: " << str.length() << "\n";
\\ size and length function return the same value
  std::cout << "capacity: " << str.capacity() << "\n";
  std::cout << "max_size: " << str.max_size() << "\n";
  std::cout << “Empty: ” << (str.empty() ? “yes” : “no”) << endl;
  return 0;
}
```

Acknowledgement
http://faizulbari.buet.ac.bd/Courses.html
http://mhkabir.buet.ac.bd/cse201/index.html



The End

Topic Covered: Sections 14.7 (Except Example 3)