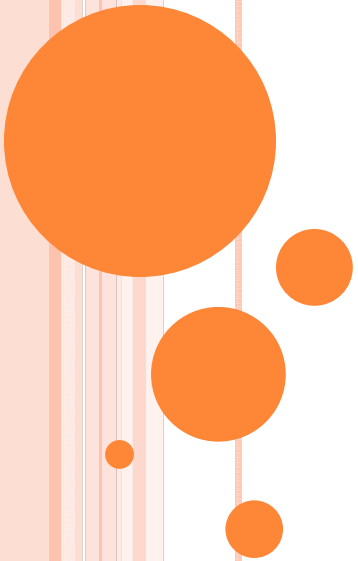

Manipulating Strings

Resources

[1] Object Oriented Programming with C++ (3rd Edition) E Balagurusamy

[2] Teach Yourself C++ (3rd Edition) H Schildt

Introduction

- 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.
- ANSI C++ provides a class called **string**.
- We must include <string> in our program.

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.

Creating String Objects

- string s1, s3; // Using constructor with no arguments.
- string s2("xyz"); // Using one-argument constructor.
- s1 = s2; // Assigning string objects
- s3 = "abc" + s2; // Concatenating strings

- 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 += s1; // s3 = s3 + s1;
- s3 += "abc"; // s3 = s3 + "abc";

Manipulating String Objects

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

- s1.insert(4, s2); // s1 = 1234abcde5

- s1.erase(4, 5); // s1 = 12345

- s2.replace(1, 3, s1); // s2 = a12345e

Manipulating String Objects

- insert()
- erase()
- replace()
- append()

Relational Operations

Operator	Meaning
==	Equality
!=	Inequality
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal

- 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

String Characteristics

```
void display(string &str)
{
   cout << "Size = " << str.size() << endl;
   cout << "Length = " << str.length() << endl;
   cout << "Capacity = " << str.capacity() << endl;
   cout << "Max Size = " << str.max_size() << endl;
   cout << "Empty: " << (str.empty() ? "yes" : "no")
   << endl;
   cout << endl << endl;
}
```

String Characteristics

Function	Task
size()	Number of elements currently stored
length()	Number of elements currently stored
capacity()	Total elements that can be stored
max_size()	Maximum size of a string object that a system can support
emply()	Return true or 1 if the string is empty otherwise returns false or 0
resize()	Used to resize a string object (effects only size and length)

Accessing Characters in Strings

Function	Task
at()	For accessing individual characters
substr()	For retrieving a substring
find()	For finding a specific substring
find_first_of()	For finding the location of first occurrence of the specific character(s)
find_last_of()	For finding the location of first occurrence of the specific character(s)
[] operator	For accessing individual character. Makes the string object to look like an array.

Comparing and Swapping

- There is another overloaded version of compare
- int compare(int start_1, int length_1, string s_2, int start_2, int length_2)
- string s1, s2;
- int x = s1.compare(0, 2, s2, 2, 2);
- s1.swap(s2)
- Exchanges the content of string s1 and s2

Lecture Contents

- [1] Object Oriented Programming with C++ (3rd Edition) E Balagurusamy
  - Chapter 15 (Full)
- [2] Teach Yourself C++ (3rd Edition) H Schildt
  - Examples only
- **Study the examples and exercise from both books carefully**