

05: The C++ string class

Programming Technique II
(SCSJ1023)

Adapted from Tony Gaddis and Barret Krupnow (2016), Starting out with C++: From Control Structures through Objects

(Last update: Oct. 7, 2018)

The C++ string Class

✿ The `string` class offers several advantages over C-style strings

- ◆ large body of member functions
- ◆ overloaded operators to simplify expressions

✿ Special data type supports working strings

```
#include <string>
```

String output - Example

Program 10-15

```
1 // This program demonstrates the string class.
2 #include <iostream>
3 #include <string> // Required for the string class.
4 using namespace std;
5
6 int main()
7 {
8     string movieTitle;
9
10    movieTitle = "Wheels of Fury";
11    cout << "My favorite movie is " << movieTitle << endl;
12    return 0;
13 }
```

Program Output

My favorite movie is Wheels of Fury

The C++ string Class

✿ Can define string variables in programs:

```
string firstName, lastName;
```

✿ Can receive values with assignment operator:

```
firstName = "George";  
lastName = "Washington";
```

✿ Can be displayed via cout

```
cout << firstName << " " << lastName;
```

The C++ string Class

❁ Objects of type string can be declared and initialized in several ways:

```
string s1;  
string s2 = "New York";  
string s3(60, '*');  
string s4 = s3;  
string s5 = 'N'; //Error. s5 must hold a string, not a character  
string s6 =78; //Error.
```

❁ If the string is not initialized, it represents the empty string.

Other ways to define C++ strings

Definition	Meaning
<code>string name;</code>	defines an empty string object
<code>string myName("Abu Bakar");</code>	defines a string and initializes it with a literal. myName: "Abu Bakar"
<code>string yourName(myName);</code>	defines a string and initializes it with values from other string. yourName: "Abu Bakar"
<code>string aName(myName, 5);</code>	defines a string and initializes it with string from myName starting at position 5 (i.e., at 6th char.) aName: "akar"
<code>string newName(myname, 0, 3);</code>	defines a string and initializes it with 2 characters from myname starting at position 3 aName: "Abu"
<code>string noName(5, 'A');</code>	defines string and initializes it to 5 'A's noName: "AAAAA"

Input into a `string` Object

✿ Use `cin >>` to read an item into a string:

```
string firstName;  
cout << "Enter your first name : ";  
cin >> firstName;
```

String Input - example

Program 10-16

```
1 // This program demonstrates how cin can read a string into
2 // a string class object.
3 #include <iostream>
4 #include <string>
5 using namespace std;
6
7 int main()
8 {
9     string name;
10
11     cout << "What is your name? ";
12     cin >> name;
13     cout << "Good morning " << name << endl;
14     return 0;
15 }
```

Program Output with Example Input Shown in Bold

What is your name? **Peggy** [Enter]
Good morning **Peggy**

String input : cin

- ✿ When the input stream `cin` is used, white space characters (space, tab, newline etc.) are used
 - ◆ Separators – read next data
 - ◆ Terminators – terminate reading a value

String input getline()

✿ Use `getline ()` to read a line of input (with spaces) into a string object

✿ The `getline ()` reads from the current position in the input stream until a newline character is found

✿ The `getline ()` takes 2 parameters:

- ◆ specifies an input stream eg. `cin` or file name
- ◆ specifies a string variable

✿ Example :

```
getline (cin, name)
```

String input : `getline()`

🌸 The `getline()` with 3 parameters

- ◆ 1st and 2nd as the `getline()` with 2 parameters
- ◆ 3rd specifies the terminator character i.e. the character at which `getline()` will stop reading from the input stream

🌸 Example :

```
iFile.open("String1.dat");  
getline(iFile, name, '\\t');
```

string operator

```
string s3, s4;  
string s1("Hi");  
string s2(" Dad");
```

```
s3 = s1 + s2;           // string concatenation  
s4 = "Hello Mom!";    // string assignment
```

Overloaded string Operators

OPERATOR	MEANING
>>	reads whitespace-delimited strings into string object
<<	outputs string object to a stream
=	assigns string on right to string object on left
+=	appends string on right to end of contents of string on left

continued ...

OPERATOR	MEANING
+	concatenates two strings
[]	references character in string using array notation
>, >=, <, <=, ==, !=	relational operators for string comparison. Return true or false

continued ...

```
string word1, phrase;  
string word2 = " Dog";  
cin >> word1; // user enters "Hot", word1 has "Hot"  
phrase = word1 + word2; // phrase has "Hot Dog"  
  
phrase += " on a bun";  
  
cout << phrase <<endl; // displays "Hot Dog on a bun"
```

string Methods

✿ Are behind many overloaded operators

✿ Categories

- ◆ assignment : `assign`, `copy`, `data`
- ◆ modification : `append`, `clear`, `erase`, `insert`, `replace`, `swap`
- ◆ space management : `capacity`, `empty`, `length`, `resize`, `size`
- ◆ substrings : `find`, `substr`
- ◆ comparison : `compare`

✿ See Table 10-7 for A list of functions

Table 10-7

Member Function Example	Description
<code>theString.append(str);</code>	Appends <code>str</code> to <code>theString</code> . <code>str</code> can be a <code>string</code> object or character array.
<code>theString.append(str, x, n);</code>	<code>n</code> number of characters from <code>str</code> , starting at position <code>x</code> , are appended to <code>theString</code> . If <code>theString</code> is too small, the function will copy as many characters as possible.
<code>theString.append(str, n);</code>	The first <code>n</code> characters of the character array <code>str</code> are appended to <code>theString</code> .
<code>theString.append(n, 'z');</code>	Appends <code>n</code> copies of 'z' to <code>theString</code> .
<code>theString.assign(str);</code>	Assigns <code>str</code> to <code>theString</code> . <code>str</code> can be a <code>string</code> object or character array.
<code>theString.assign(str, x, n);</code>	<code>n</code> number of characters from <code>str</code> , starting at position <code>x</code> , are assigned to <code>theString</code> . If <code>theString</code> is too small, the function will copy as many characters as possible.
<code>theString.assign(str, n);</code>	The first <code>n</code> characters of the character array <code>str</code> are assigned to <code>theString</code> .
<code>theString.assign(n, 'z');</code>	Assigns <code>n</code> copies of 'z' to <code>theString</code> .
<code>theString.at(x);</code>	Returns the character at position <code>x</code> in the <code>string</code> .
<code>theString.begin();</code>	Returns an iterator pointing to the first character in the <code>string</code> . (For more information on iterators, see Chapter 15.)
<code>theString.capacity();</code>	Returns the size of the storage allocated for the <code>string</code> .
<code>theString.clear();</code>	Clears the <code>string</code> by deleting all the characters stored in it.

(table continues)

EXAMPLE	REMARKS
Constructors	
string str;	Default constructor; creates empty string object str.
string str("string");	Creates a string object with data "string".
string str(aString);	Creates a string object str that is a copy of aString. aString is an object of the class string.
Element access	
str[i]	Returns read/write reference to character in str at index i.
str.at(i)	Returns read/write reference to character in str at index i.
str.substr(position, length)	Returns the substring of the calling object starting at position and having length characters.
Assignment/Modifiers	
str1 = str2;	Allocates space and initializes it to str2's data, releases memory allocated for str1, and sets str1's size to that of str2.
str1 += str2;	Character data of str2 is concatenated to the end of str1; the size is set appropriately.
str.empty()	Returns true if str is an empty string; returns false otherwise.
str1 + str2	Returns a string that has str2's data concatenated to the end of str1's data. The size is set appropriately.
str.insert(pos, str2)	Inserts str2 into str beginning at position pos.
str.remove(pos, length)	Removes substring of size length, starting at position pos.
Comparisons	
str1 == str2 str1 != str2	Compare for equality or inequality; returns a Boolean value.
str1 < str2 str1 > str2	Four comparisons. All are lexicographical comparisons.
str1 <= str2 str1 >= str2	
str.find(str1)	Returns index of the first occurrence of str1 in str.
str.find(str1, pos)	Returns index of the first occurrence of string str1 in str; the search starts at position pos.
str.find_first_of(str1, pos)	Returns the index of the first instance in str of any character in str1, starting the search at position pos.
str.find_first_not_of(str1, pos)	Returns the index of the first instance in str of any character not in str1, starting search at position pos.

<code>theString.compare(str);</code>	Performs a comparison like the <code>strcmp</code> function (see Chapter 4), with the same return values. <code>str</code> can be a <code>string</code> object or a character array.
<code>theString.compare(x, n, str);</code>	Compares <code>theString</code> and <code>str</code> , starting at position <code>x</code> , and continuing for <code>n</code> characters. The return value is like <code>strcmp</code> . <code>str</code> can be a <code>string</code> object or character array.
<code>theString.copy(str, x, n);</code>	Copies the character array <code>str</code> to <code>theString</code> , beginning at position <code>x</code> , for <code>n</code> characters. If <code>theString</code> is too small, the function will copy as many characters as possible.
<code>theString.data();</code>	Returns a character array containing a null terminated string, as stored in <code>theString</code> .
<code>theString.empty();</code>	Returns true if <code>theString</code> is empty.
<code>theString.end();</code>	Returns an iterator pointing to the last character of the string in <code>theString</code> . (For more information on iterators, see Chapter 15.)
<code>theString.erase(x, n);</code>	Erases <code>n</code> characters from <code>theString</code> , beginning at position <code>x</code> .
<code>theString.find(str, x);</code>	Returns the first position at or beyond position <code>x</code> where the string <code>str</code> is found in <code>theString</code> . <code>str</code> may be either a <code>string</code> object or a character array.
<code>theString.find('z', x);</code>	Returns the first position at or beyond position <code>x</code> where 'z' is found in <code>theString</code> .
<code>theString.insert(x, str);</code>	Inserts a copy of <code>str</code> into <code>theString</code> , beginning at position <code>x</code> , <code>str</code> may be either a <code>string</code> object or a character array.
<code>theString.insert(x, n, 'z');</code>	Inserts 'z' <code>n</code> times into <code>theString</code> at position <code>x</code> .
<code>theString.length();</code>	Returns the length of the string in <code>theString</code> .
<code>theString.replace(x, n, str);</code>	Replaces the <code>n</code> characters in <code>theString</code> beginning at position <code>x</code> with the characters in string object <code>str</code> .
<code>theString.resize(n, 'z');</code>	Changes the size of the allocation in <code>theString</code> to <code>n</code> . If <code>n</code> is less than the current size of the string, the string is truncated to <code>n</code> characters. If <code>n</code> is greater, the string is expanded and 'z' is appended at the end enough times to fill the new spaces.
<code>theString.size();</code>	Returns the length of the string in <code>theString</code> .
<code>theString.substr(x, n);</code>	Returns a copy of a substring. The substring is <code>n</code> characters long and begins at position <code>x</code> of <code>theString</code> .
<code>theString.swap(str);</code>	Swaps the contents of <code>theString</code> with <code>str</code> .

Length of a String

✿ The number of characters it contains including whitespace characters, if any.

✿ Example:

```
string s1 = "UTM Skudai";  
int len = s1.length(); // len is 10
```

Modification of string objects

✿ Appending strings (to the end of a string)

```
string str1 = "UTM ";  
string str2 = "Skudai ";
```

```
str1.append(str2); // str1 becomes "UTM Skudai "  
str2.append(str1); // str2 becomes "Skudai UTM Skudai"
```

✿ Inserting strings (within a string)

```
string str3 = "Skudai ";  
string str4("UTM Johor"); // str4 has "UTM Johor"  
// insert str3 into str4 starting at position 4  
str4.insert(4, str3); // str4 becomes "UTM Skudai Johor"
```

Substrings

✿ `substr` method returns a substring

✿ syntax is :

```
s.substr(position, length) ;
```

✿ Example:

```
string name = "objects" ;
```

```
// copy a sub string from name starting at position 2 and taking 3 characters
```

```
string sub = name.substr(2, 3) ; // sub has "jec"
```

```
// copy a sub string from name starting at position 3
```

```
sub = name.substr(3, 15) ; // sub has "ects"
```