Unit 3 Lesson 5

Strings and the String Class

Mr. Dave Clausen
(modifications from the textbook)

Introduction to Strings and Literals

- **A group of characters put together to create text is called a *string*. Strings are one of the most useful kinds of data in a program.**
- **C++ doesn't have a "built-in" data type for strings.**
- **Values or strings that are hard-coded into the source code are called *literals*. .**
- **A hard-coded numeric value is called a *numeric literal*.**
- **A string of text that is hard-coded is called a *string literal*.**
- **A single character can also be hard-coded. A *character literal* appears in single quotation marks.**
- **A string literal appears in double quotation marks.**

Code List 5-1

```
x = 6.3;                    // 6.3 is a numeric literal
cout << “Hello”;            // “Hello” is a string literal
MyChar = ‘A’;               // ‘A’ is a character literal
```

- **Literals cannot change their values while the program runs.**

Introduction to Classes and Objects

- **C++ is an Object Oriented Programming (OOP) language.**
- **The purpose of a high level programming language is to hide the details and make programming a more rapid and dependable process.**
- **OOP allows programmers to create their own operations, and data types while hiding the details.**
- **You don’t need to know the details of how strings work, just how to use them.**

String Class

- **An object-oriented string data type is referred to as a string class.**
- **A string class is a definition used to create a string object.**
- **It is important to know the distinction between a class and an object.**
  - **Think of a class as a generic definition from which an object is created.**
  - **A dog would be an example of a class, while "Rover" would be an example of an object of that class (the dog class).**
  - **An object is said to be an *instance* of a class.**
  - **Therefore, "Rover" is an instance of the dog class.**

The “oostring” class

- **Our textbook authors have provided us with a string class to use, they have named it the “oostring” class.**
- **The class consists of two parts, a header file and a definition file, or**

oostring.h **oostring.cpp**

oostringh.txt **oostringcpp.txt**

Using the String Class

- **Using the string class is a little more complicated than using the built-in data types.**
- **To use the string class, you must include a header file in your source code.**
- **#include “oostring.cpp” for Borland C++ 5.02**
- **//”oostring.h” for other compilers.**
- **Use “ ” instead of < > since oostring is not precompiled.**
- **The two files (oostring.h and oostring.cpp) have to be in the same directory as your source code.**

Code List 5-2

- **When you declare a string object, you can create an empty string object or you can initialize the object with a string.**

```
#include “oostring.cpp”
int main ( )
{
    oostring MyString1;          // declare an empty string object

    oostring MyString2 (“ABCDEF”);  // initializing while declaring
}
```

Assigning Strings to String Objects

- **You can assign strings to string objects in one of three ways:**
  - **You can assign the contents of one string object to another string object.**
  - **You can assign a string literal to a string object.**
  - **You can assign a character literal to a string object.**

Code List 5-3

```
#include “oostring.cpp”
int main ( )
{
    MyString1 = MyString2;  //object to object
    MyString1 = “string literal”;  //literal to object
    MyString1 = ‘A’;  //character literal to string object
}
```

Printing the Contents of a String Object to the Screen

- **You can use cout to display the contents of a string object:**

```
cout << MyString1 << endl;
```

Code List 5-4

/ / stringex.cpp stringex.txt

```
# include  <iostream.h>
# include  “oostring.cpp”

int main ()
{
     oostring MyString1;
     oostring MyString2 (“ABCDEFGHIJKLMNOPQRSTUVWXYZ”) ;
    MyString1  =  “Hello world!”;

    cout  <<  MyString1  <<  ‘\n’;
    cout  <<  MyString2  <<  ‘\n’;
    return  0;
}
```

String Operations

- **Programs often need to perform a variety of operations on the strings it stores.**
- **For example, to properly center a string, you may need to know the number of characters in the string.**
- **You may also need to add the strings together.**

Messages

- **One of the important concepts behind the use of objects is the idea of *containment* .**
- **This term refers to the way an object hides the details of how data is stored, and how operations work.**
- **The data, and the code required to work with the data, is *contained* or *encapsulated* within the object itself.**
- **To make the object do what we want it to do, we send the object a *message*.**

Obtaining the Length of a String

- **The message used to obtain the length of a string is simply *length*.**

  length = MyString2.length();
- **The period that follows the name of the object is called the *dot operator*.**
- **The dot operator separates the name of the object from the message, in this case length.**
- **The code inside the object that performs the length operation is called a *method*.**
- **Therefore, when you are sending the length message you could say that you are using the length method of the string class.**

Code List 5-5

```
// stringex2.cpp                    stringex2.txt
#include <iostream.h>
#include <conio.h>
#include "oostring.cpp"
int main()
{
  int length1;
  int length2;
  oostring MyString1;
  oostring MyString2("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
  MyString1 = "Hello World!";
  length1 = MyString1.length();
  length2 = MyString2.length();
  cout << MyString1 << endl;
  cout << "Length = " << length1 <<endl;
  cout << MyString2 << endl;
  cout << "Length = " << length2 <<endl;
  getch();
  return 0;
}
```

Shorthand Notation

- **I prefer that we do not use the following shorthand notation in our programs, since this is an introductory class.**

TABLE 5-1
Concatenation examples

SHORTHAND METHOD	LONG METHOD
j += 7;	j = j + 7;
k += n;	k = k + n;

String Concatenation

- ***Concatenation*** **is a big word that describes the operation of adding or appending one string onto the end of another string.**
- **Suppose, for example, that you have one string object holding a first name and another string object holding a last name.**
- **To get both strings together in one string object, you need to concatenate the last name onto the first name.**
- **Actually, you would first concatenate a space onto the end of the first name to insert a space between the first and last names.**

Concatenation Shorthand

- **I prefer that we do not use the following shorthand notation in our programs, since this is an introductory class.**

TABLE 5-2
Concatenation statements

STATEMENT	DESCRIPTION
`MyString1 += MyString2;`	Add MyString2 to the end of MyString1.
`MyString1 += "string literal";`	Add a string literal to the end of MyString1.
`MyString1 += Ch;`	Add a character to the end of MyString1.
`MyString1 += 'A';`	Add a character literal to the end of MyString1.

Code List 5-6

```
MyString1  =  “Tracy”;
MyString2  =  “Stewart”;

MyString1  = MyString1 +  ‘  ‘;          / /  add a space after the first name
MyString1  = MyString1 + MyString2;  / /  add the last name to MyString1
MyString1  = MyString1 +  “  was here.”;  / /  add a string literal to MyString1

cout  <<  MyString1  <<  endl;
```