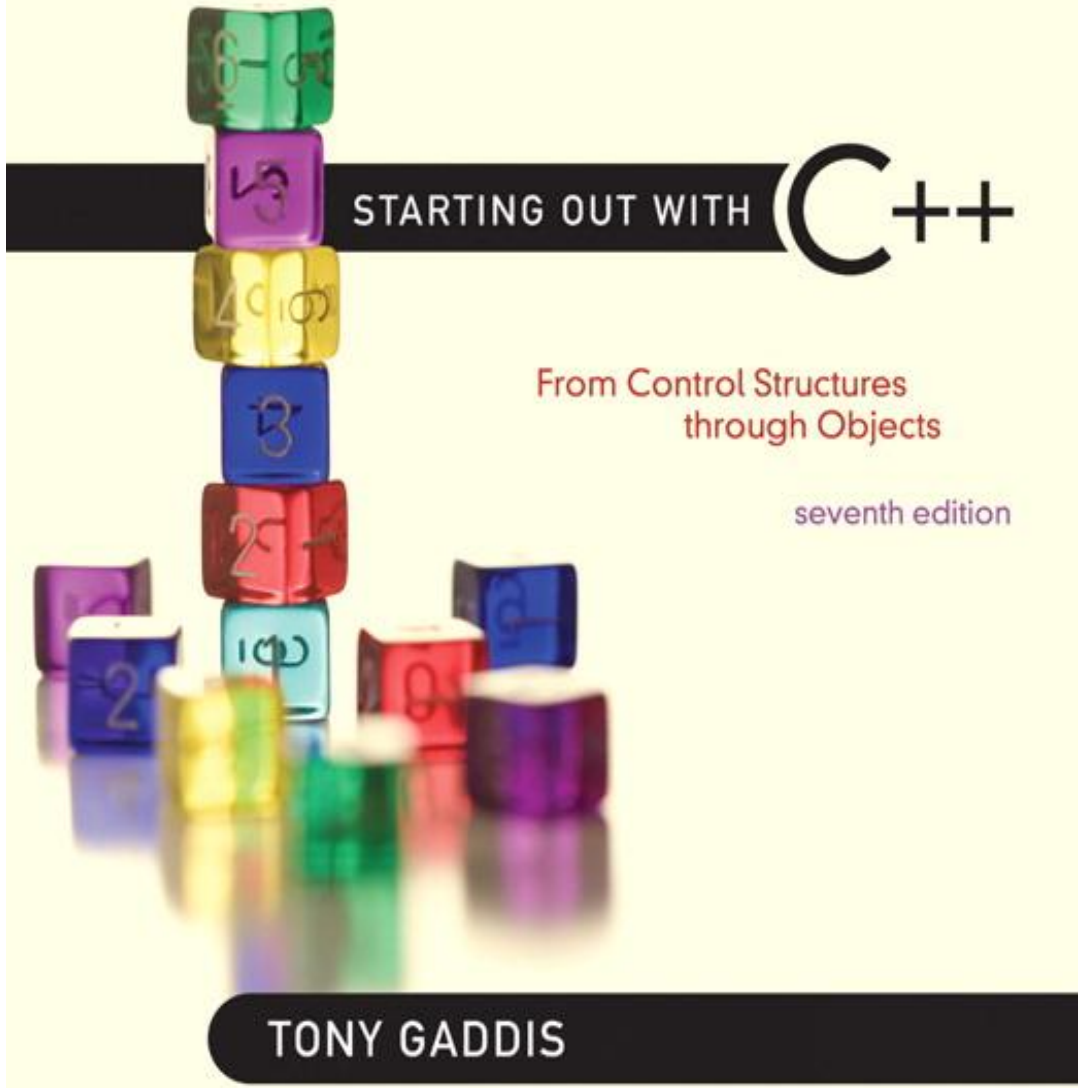
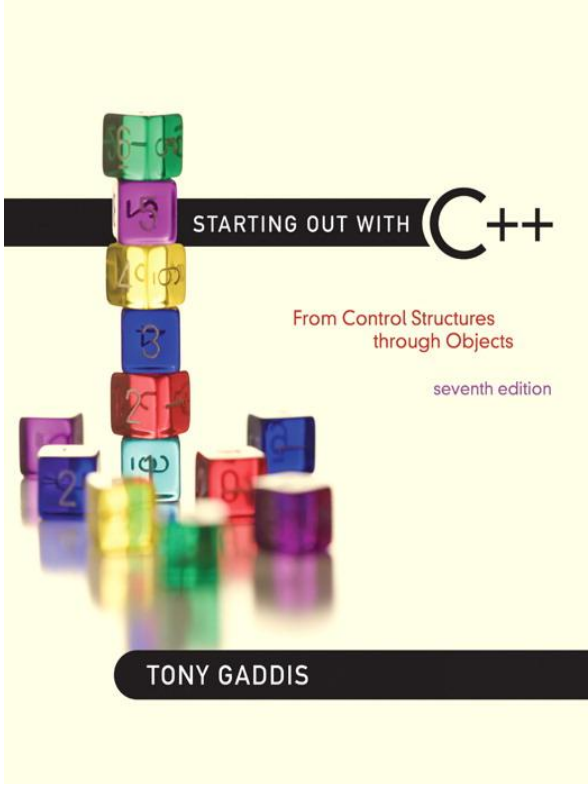
Chapter 10:

Characters, C-Strings, and More About the `string` Class



Addison-Wesley
is an imprint of
PEARSON



10.1

Character Testing

Character Testing

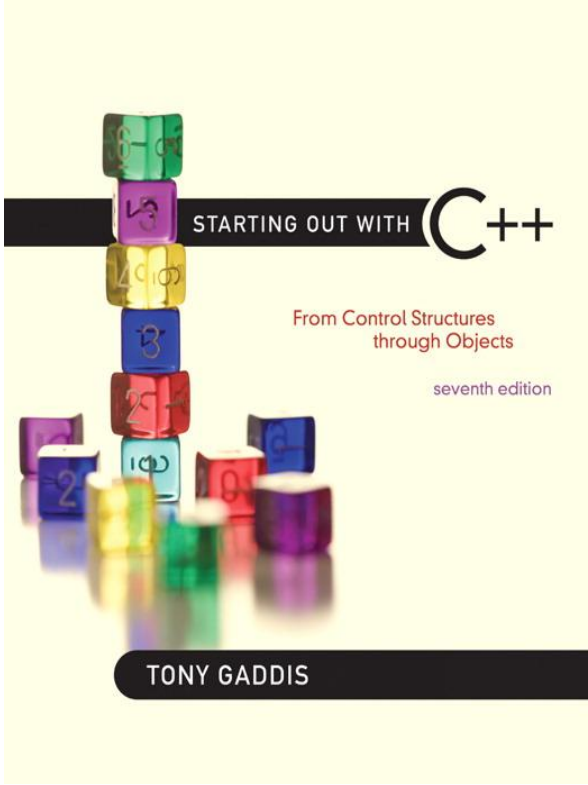
Requires **cctype** header file

```
char j;
```

Function	Meaning
isalpha(j)	returns true if j is a letter
isalnum(j)	returns true if j is a letter or digit
isdigit(j)	returns true if j is a digit 0-9
islower(j)	returns true if j is a lowercase letter
isprint(j)	returns true if j is a printable character
ispunct(j)	returns true if j is a punctuation character
isupper(j)	returns true if j is an uppercase letter
isspace(j)	returns true if j is a whitespace character

From Program 10-1

```
10      cout << "Enter any character: ";
11      cin.get(input);
12      cout << "The character you entered is: " << input << endl;
13      if (isalpha(input))
14         cout << "That's an alphabetic character.\n";
15      if (isdigit(input))
16         cout << "That's a numeric digit.\n";
17      if (islower(input))
18         cout << "The letter you entered is lowercase.\n";
19      if (isupper(input))
20         cout << "The letter you entered is uppercase.\n";
21      if (isspace(input))
22         cout << "That's a whitespace character.\n";
```



10.2

Character Case Conversion

Character Case Conversion

- Requires cctype header file
- Function toupper(ch)
  - if **char** argument is lowercase letter, return uppercase equivalent; otherwise, return input unchanged

```
char ch1 = 'H';
char ch2 = 'e';
char ch3 = '!';
cout << toupper(ch1);  // displays 'H'
cout << toupper(ch2);  // displays 'E'
cout << toupper(ch3);  // displays '!'
```

Character Case Conversion

- Function tolower(ch)
  - if `char` argument is uppercase letter, return lowercase equivalent; otherwise, return input unchanged

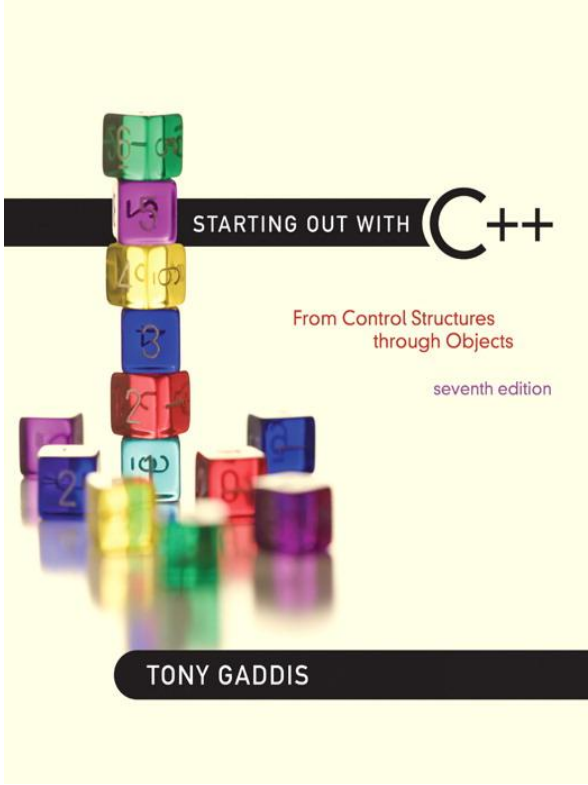
```
char ch1 = 'H';
char ch2 = 'e';
char ch3 = '!';
cout << tolower(ch1);  // displays 'h'
cout << tolower(ch2);  // displays 'e'
cout << tolower(ch3);  // displays '!'
```

Function	Description
toupper	Returns the uppercase equivalent of its argument.
tolower	Returns the lowercase equivalent of its argument.

Function Prototypes for toupper and tolower

```
int  toupper( int );    //notice functions return type int!
int  tolower( int );
```

```
char x = 'j';  x = toupper(x);
```



10.3

C-Strings

Internal Storage of C-strings

- C-string:
  - sequence of characters stored in adjacent memory locations and terminated by **NULL** character
- string literal (string constant):
  - sequence of characters enclosed in double quotes " " : `"Hi there!"`

H	i		t	h	e	r	e	!	\0

Internal Storage of C-Strings

- A `char` array is used to can be used to define storage for a string:

```
char city[21];
```

- Leave room for **`NULL`** character!
- Can enter a value using **`cin >>`**
  - Input is **whitespace-terminated**
  - No check to see if enough space
- For input containing whitespace, and to control amount of input, use

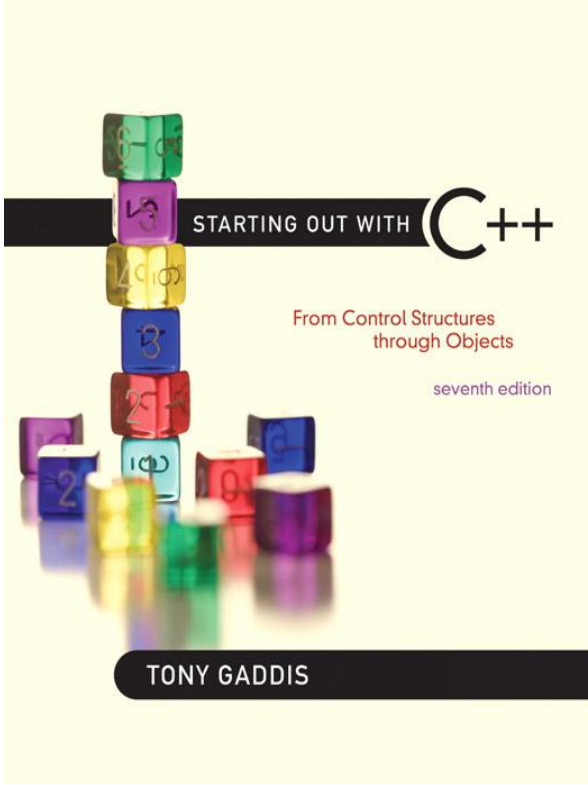
```
cin.getline(array,size);
```

Program 10-5

```
1  // This program displays a string stored in a char array.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     const int SIZE = 80;  // Array size
8     char line[SIZE];      // To hold a line of input
9     int count = 0;        // Loop counter variable
10
11    // Get a line of input.
12    cout << "Enter a sentence of no more than "
13         << (SIZE - 1) << " characters:\n";
14    cin.getline(line, SIZE);
15
16    // Display the input one character at a time.
17    cout << "The sentence you entered is:\n";
18    while (line[count] != '\0')
19    {
20       cout << line[count];
21       count++;
22    }
23    return 0;
24 }
```

Program Output with Example Input Shown in Bold

```
Enter a sentence of no more than 79 characters:
C++ is challenging but fun! [Enter]
The sentence you entered is:
C++ is challenging but fun!
```



10.4

Library Functions for Working with C-Strings

Library Functions for Working with C-Strings

- requires **`cstring`** header file
- functions take one or more C-strings as arguments. A C-string may be a:
  - char array name
  - pointer to char `char *`
  - literal string

Function	Description
strlen	Accepts a C-string or a pointer to a string as an argument. Returns the length of the string (not including the null terminator. Example Usage: len = strlen(name);
strcat	Accepts two C-strings or pointers to two strings as arguments. The function appends the contents of the second string to the first string. (The first string is altered, the second string is left unchanged.) Example Usage: strcat(string1, string2); The first argument must be a character array or a pointer to an element in a character array. Returns a pointer to string1.
strcpy	Accepts two C-strings or pointers to two strings as arguments. The function copies the second string to the first string. The second string is left unchanged. Example Usage: strcpy(string1, string2); The first argument must be a character array or a pointer to an element in a character array. Returns a pointer to string1.
strncpy	Accepts two C-strings or pointers to two strings and an integer argument. The third argument, an integer, indicates how many characters to copy from the second string to the first string. If the *string2* has fewer than n characters, *string1* is padded with '\0' characters. Example Usage: strncpy(string1, string2, n); The first argument must be a character array or a pointer to an element in a character array. Returns a pointer to string1.
strcmp	Accepts two C-strings or pointers to two string arguments. If *string1* and *string2*are the same, this function returns 0. If *string2* is alphabetically greater than *string1*, it returns a negative number. If *string2* is alphabetically less than *string1*, it returns a positive number. Example Usage: if (strcmp(string1, string2))
strstr	Accepts two C-strings or pointers to two C-strings as arguments, searches for the first occurrence of *string2* in *string1*. If an occurrence of *string2* is found, the function returns a pointer to it. Otherwise, it returns a NULL pointer (address 0). Example Usage: cout << strstr(string1, string2);

Library Functions for Working with C-Strings

Functions:

```
strlen(str)   //returns length of C-string str
```

```
char city[20] = "Missoula";
cout << strlen(city); // prints 8
```

```
strcat(str1, str2)   //appends str2 to the end of str1
```

```
char location[20] = "Missoula, ";
char state[3] = "MT";

strcat(location, state);

// location now has "Missoula, MT"
```

Library Functions for working with C-Strings

Function:

```
strcpy(str1, str2) //copies str2 to str1
```

```
char fname[20] = "Maureen", name[20];
```

```
strcpy(name, fname);
```

Note: strcat and strcpy perform no bounds checking to determine if there is enough space in receiving character array to hold the string it is being assigned.

C-string Inside a C-string

Function:

`strstr(str1, str2)`

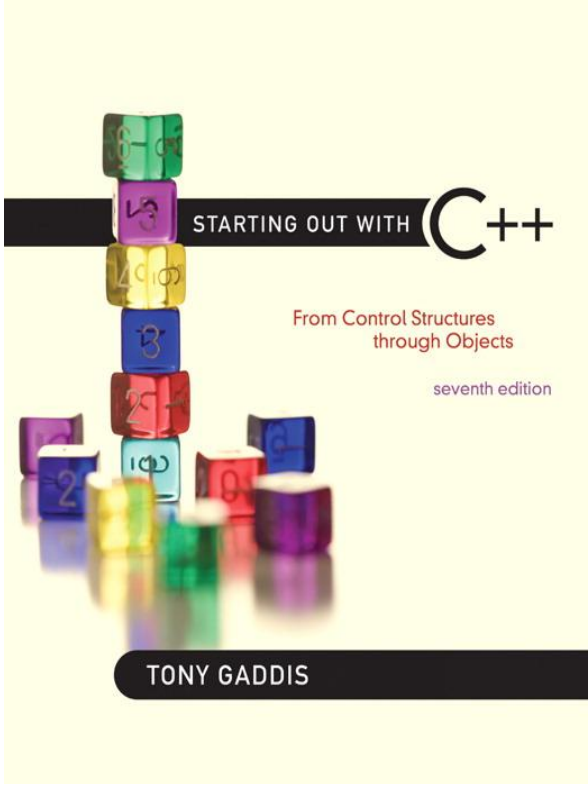
finds the first occurrence of **`str2`** in **`str1`**. Returns a pointer to match, or **`NULL`** if no match.

```
char river[10] = "Wabash";
char word[5] = "aba";

cout << strstr(river, word);

// displays "abash"
```

String/Numeric Conversion Functions

require `cstdlib` header file

Function	Parameter	Action
`atoi`	C-string	converts C-string to an `int` value, returns the value
`atol`	C-string	converts C-string to a `long` value, returns the value
`atof`	C-string	converts C-string to a `double` value, returns the value
`itoa`	**`int`**, C-string, **`int`**	converts 1st `int` parameter to a C-string, stores it in 2nd parameter. 3rd parameter is base of converted value



10.5

C-String/Numeric Conversion Functions

String/Numeric Conversion Functions

require `cstdlib` header file

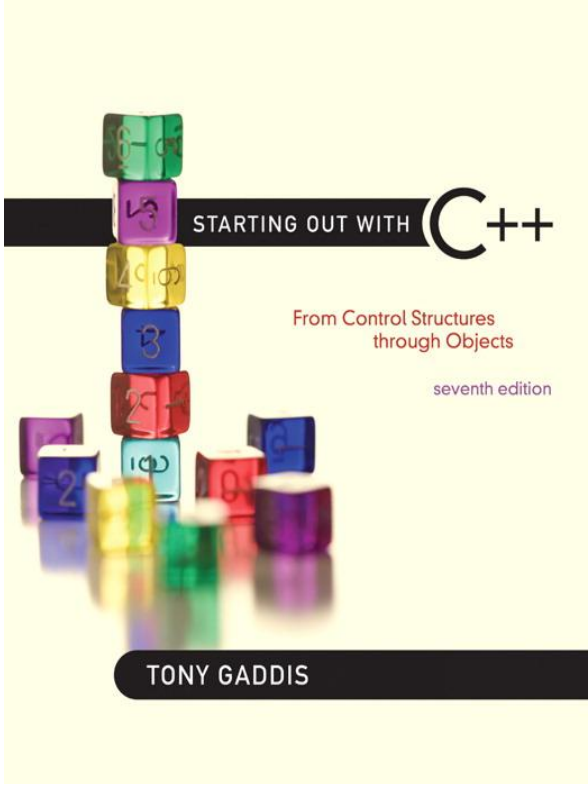
Function	Parameter	Action
`atoi`	C-string	converts C-string to an `int` value, returns the value
`atol`	C-string	converts C-string to a `long` value, returns the value
`atof`	C-string	converts C-string to a `double` value, returns the value
`itoa`	**`int`**, C-string, **`int`**	converts 1st `int` parameter to a C-string, stores it in 2nd parameter. 3rd parameter is base of converted value

String/Numeric Conversion Functions

```
int     i_Num;
long    l_Num;
double d+Num;
char    intChar[10];

i_Num = atoi("1234");  // puts 1234 in i_Num
l_Num = atol("5678");  // puts 5678 in l_Num
d_Num = atof("35.7");  // puts 35.7 in d_Num

itoa(i_Num, intChar, 8);

// puts the string "2322" (base 8 for 1234₁₀) in intChar
```

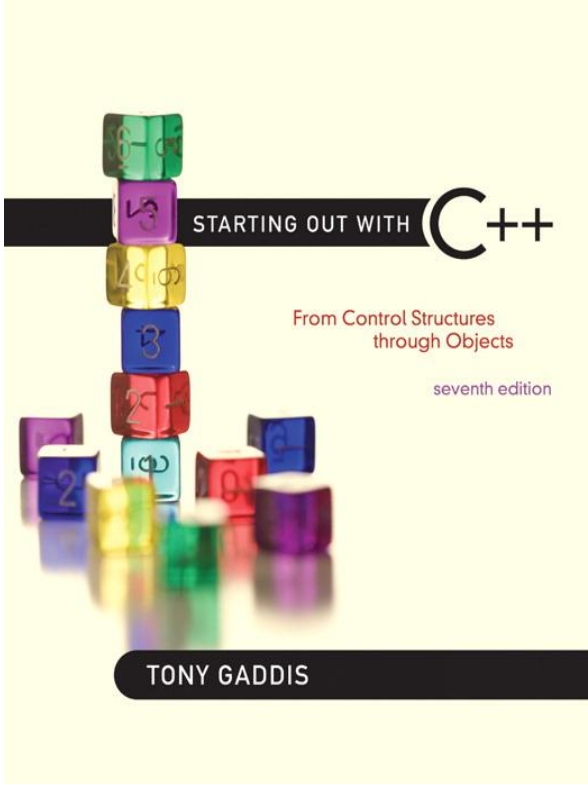
String/Numeric Conversion Functions

- if the C-string contains non-digits, results are undefined:
  - function may return result up to non-digit
  - function may return 0
- itoa does not do bounds checking; make sure there is enough space to store the result



10.6

Writing Your Own C-String Handling Functions



10.6

Writing Your Own C-String Handling Functions

Writing Your OwnC-String Handling Functions

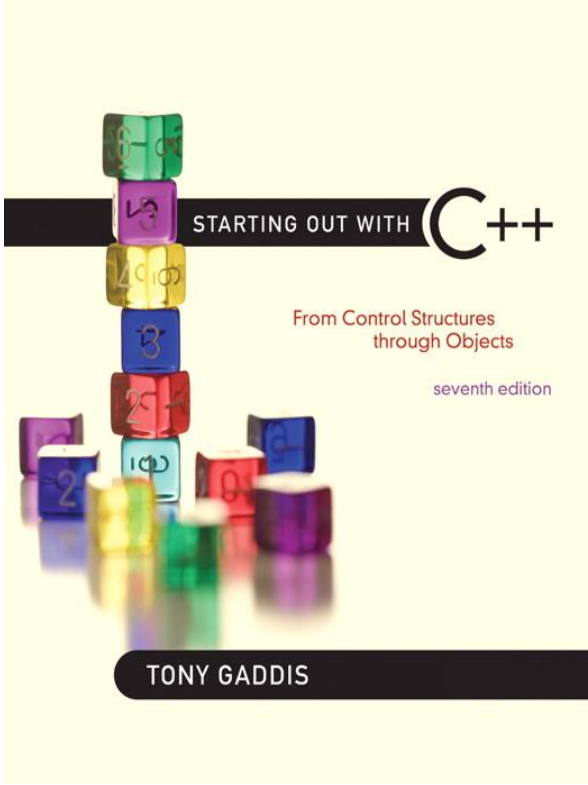
- Designing C-String Handling Functions
  - can pass arrays or pointers to `char` arrays
  - can perform bounds checking to ensure enough space for results
  - can anticipate unexpected user input

From Program 10-9

```
31  void stringCopy(char string1[], char string2[])
32  {
33     int index = 0;  // Loop counter
34
35     // Step through string1, copying each element to
36     // string2. Stop when the null character is encountered.
37     while (string1[index] != '\0')
38     {
39        string2[index] = string1[index];
40        index++;
41     }
42
43     // Place a null character in string2.
44     string2[index] = '\0';
45  }
```

From Program 10-10

```
29  void nameSlice(char userName[])
30  {
31     int count = 0;  // Loop counter
32
33     // Locate the first space, or the null terminator if there
34     // are no spaces.
35     while (userName[count] != ' ' && userName[count] != '\0')
36        count++;
37
38     // If a space was found, replace it with a null terminator.
39     if (userName[count] == ' ')
40        userName[count] = '\0';
41  }
```



10.7

More About the C++ `string` Class

The C++ string Class

- Special datatype supports working with strings
- #include <string> [versus cstring]
- 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;
```

- Strings are mutable in C++ (they are immutable in Java…)

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
```

Input into a **string** Object

- Use getline function to put a line of input, possibly including spaces, into a string:

```
string address;
cout << "Enter your address: ";

getline(cin,address);
```

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` Comparison

- Can use relational operators directly to compare string objects:

```
string str1 = "George",
       str2 = "Georgia";

if ( str1 < str2 )
   cout << str1 << " is less than "
        << str2;
```

- Comparison is performed similar to strcmp function. Result is **`true`** or **`false`**

Program 10-18

```
1  // This program uses relational operators to alphabetically
2  // sort two strings entered by the user.
3  #include <iostream>
4  #include <string>
5  using namespace std;
6
7  int main ()
8  {
9     string name1, name2;
10
11    // Get a name.
12    cout << "Enter a name (last name first): ";
13    getline(cin, name1);
14
15    // Get another name.
16    cout << "Enter another name: ";
17    getline(cin, name2);
18
19    // Display them in alphabetical order.
20    cout << "Here are the names sorted alphabetically:\n";
21    if (name1 < name2)
22       cout << name1 << endl << name2 << endl;
23    else if (name1 > name2)
24       cout << name2 << endl << name1 << endl;
25    else
26       cout << "You entered the same name twice!\n";
27    return 0;
28 }
```

Program Output with Example Input Shown in Bold

```
Enter a name (last name first): Smith, Richard [Enter]
Enter another name: Jones, John [Enter]
Here are the names sorted alphabetically:
Jones, John
Smith, Richard
```

Other Definitions of C++ string's

Definition	Meaning
`string name;`	defines an empty string object
`string myname("Chris");`	defines a string and initializes it
`string yourname(myname);`	defines a string and initializes it
`string aname(myname, 3);`	defines a string and initializes it with first 3 characters of **myname**
`string verb(myname,3,2);`	defines a string and initializes it with 2 characters from **myname** starting at position 3
`string noname('A', 5);`	defines string and initializes it to 5 'A's

string Operators

Operator	Meaning
>>	extracts characters from stream up to whitespace, insert into string
<<	inserts string into stream
=	assigns string on right to string object on left
+=	appends string on right to end of contents on left
+	concatenates two strings
[ ]	references character in string using array notation
>, >=, <, <=, ==, !=	relational operators for string comparison. Return true or false

string Operators

```
string word1, phrase;
string word2 = " Dog";

cin >> word1; // user enters "Hot Tamale"
              // word1 has "Hot"

phrase = word1 + word2; // phrase has "Hot Dog"

phrase += " on a bun";

// displays "Hot Dog on a bun"

for (int i = 0; i < 16; i++)
       cout << phrase[i];
```

Program 10-20

```
1  // This program demonstrates the C++ string class.
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  int main ()
7  {
8     // Define three string objects.
9     string str1, str2, str3;
10
11    // Assign values to all three.
12    str1 = "ABC";
13    str2 = "DEF";
14    str3 = str1 + str2;
15
16    // Display all three.
17    cout << str1 << endl;
18    cout << str2 << endl;
19    cout << str3 << endl;
20
21    // Concatenate a string onto str3 and display it.
22    str3 += "GHI";
23    cout << str3 << endl;
24    return 0;
25 }
```

Program Output

```
ABC
DEF
ABCDEF
ABCDEFGHI
```

string class member functions

s.append(str);	Appends str to theString. str can be a string object or a character array.
s.append(str, x, n);	n number of characters from str, starting at position x, are appended to theString. If theString is too small, the function will copy as many characters as possible.
s.append(str, n);	The first n characters of the character array str are appended to theString.
s.append(n, 'z');	Appends n copies of 'z' to theString.

string class member functions

s.assign(str);	Assigns str to theString. str can be a string object or character array.
s.assign(str, x, n);	n number of characters from str, starting at position x, are assigned to theString. If theString is too small, the function will copy as many characters as possible.
s.assign(str, n);	The first n charactrers of the character array str are assigned to theString.
s.assign(n, 'z');	Assigns n copies of 'z' to theString.

string class member functions

s.at(x);	Returns the character at position x in theString.
s.begin( );	Returns an iterator pointing to the first character in the string.
s.capacity( );	Returns the size of the storage allocated for theString.
s.clear( );	Clears theString by deleting all the characters stored in it.
s.compare (str);	Performs a comparison like the strcmp function with the same return values. str can be a string object or a character array.
s.compare(str, x, n);	Compares theString and str, starting at position x, and continuing for n characters. The return value is like strcmp. str can be a string object or character array.

string class member functions

s.copy(str, x, n);	Copies the character array str to theString, beginning at position x, for n characters. If theString is too small, the function will copy as many characters as possible.
s.data( );	Returns a character array containing a null terminated string, as stored in theString.
s.empty( );	Returns true if theString is empty.
s.end( );	Returns an iterator pointing to the last characters of the string in theString.
s.erase(x, n);	Erases n characters from theString, beginning at position x.

string class member functions

s.find(str, x);	Returns the first position at or beyond position x where the string str is found in theString. str may be either a string object or a character array.
s.find('z', x);	Returns the first position at or beyond position x where 'z' is found in theString.
s.insert(x, str);	Inserts a copy of str into theString, beginning at position x. str may be either a string object or a character array.
s.insert(x, n, 'z');	Inserts 'z' n times into theString at position x.
s.length( );	Returns the length of the string in theString.

string class member functions

s.replace(x, n, str);	Replaces the n characters in theString beginning at position x with the characters in string object str.
s.resize(n, 'z');	changes the size of the allocation in theString to n. If n is less that the current size of the string, theString is truncated to n characters. If n is greater, theString is expanded and 'z' is appended at the end enough times to fill the new positions.
s.size( );	Returns the length of the string in theString.
s.substr(x, n);	Returns a copy of a substring. The substring is n characters long and begins at position x of theString.
s.swap(str)	Swaps the contents of theString with str.

Program 10-21

```
1  // This program demonstrates a string
2  // object's length member function.
3  #include <iostream>
4  #include <string>
5  using namespace std;
6
7  int main ()
8  {
9     string town;
10
11    cout << "Where do you live? ";
12    cin >> town;
13    cout << "Your town's name has " << town.length() ;
14    cout << " characters\n";
15    return 0;
16 }
```

Program Output with Example Input Shown in Bold

```
Where do you live? Jacksonville [Enter]
Your town's name has 12 characters
```