# Chapter 10:

## Characters, C-Strings, and More About the `string` Class
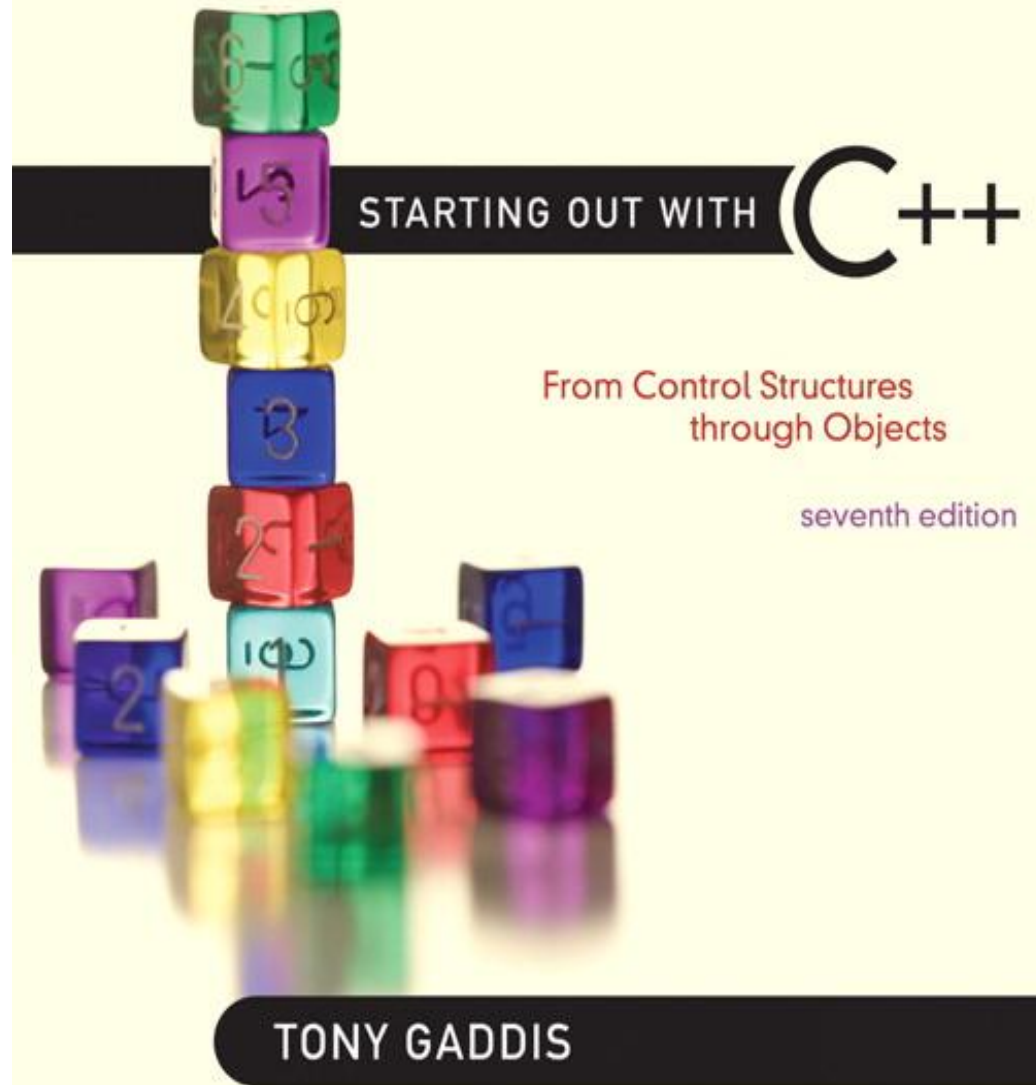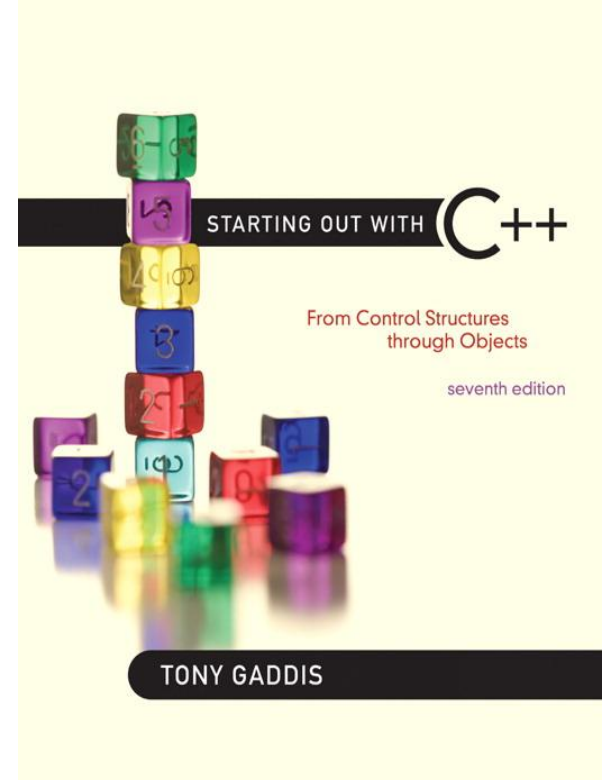
STARTING OUT WITH C++

From Control Structures through Objects

seventh edition

**TONY GADDIS**

# 10.1

## Character Testing

# Character Testing

- require `cctype` header file

| FUNCTION | MEANING |
|----------|---------|
| isalpha | true if arg. is a letter, false otherwise |
| isalnum | true if arg. is a letter or digit, false otherwise |
| isdigit | true if arg. is a digit 0-9, false otherwise |
| islower | true if arg. is lowercase letter, false otherwise |
| isprint | true if arg. is a printable character, false otherwise |
| ispunct | true if arg. is a punctuation character, false otherwise |
| isupper | true if arg. is an uppercase letter, false otherwise |
| isspace | true if arg. is a whitespace character, false otherwise |

# 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
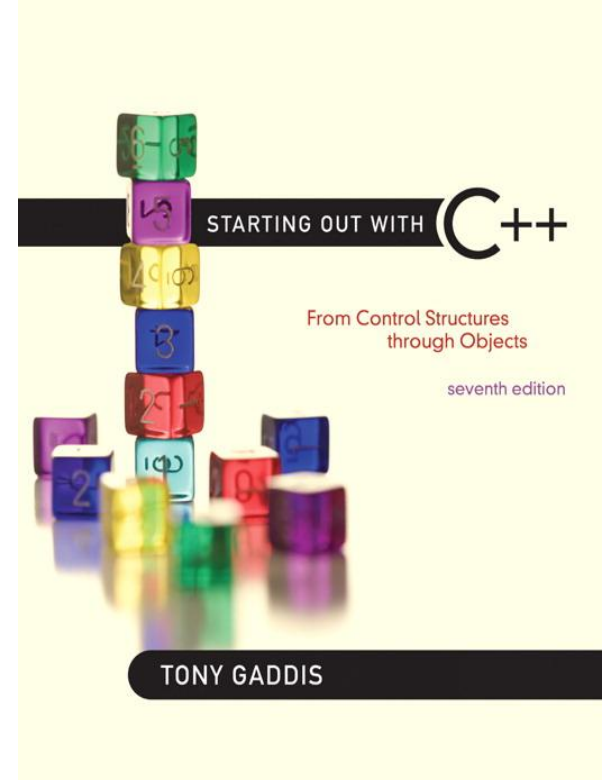
# Character Case Conversion

- Require `cctype` header file
- Functions:

  `toupper`: 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

- Functions:

  `tolower`: 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 '!'
```
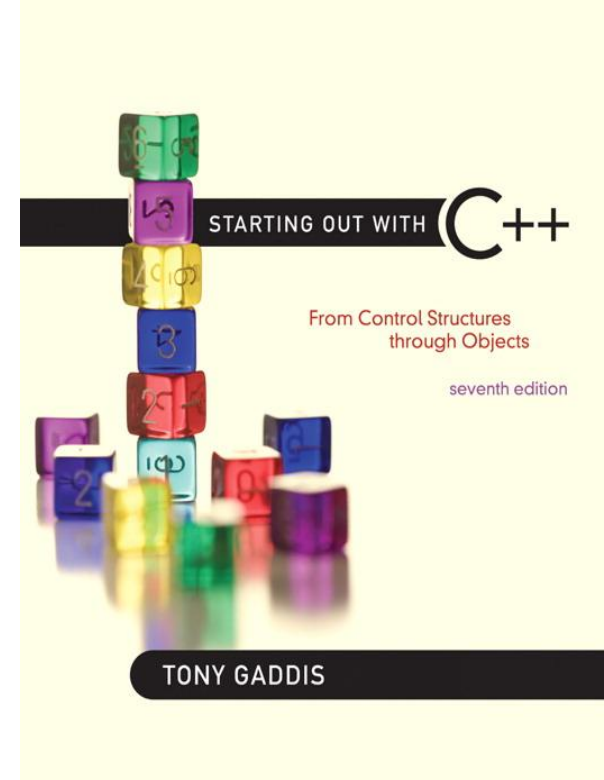
# 10.3

## C-Strings

# C-Strings

- <u>C-string</u>: sequence of characters stored in adjacent memory locations and terminated by `NULL` character

- <u>String literal</u> (<u>string constant</u>): sequence of characters enclosed in double quotes " " :

  `"Hi there!"`

| H | i |   | t | h | e | r | e | ! | \0 |
|---|---|---|---|---|---|---|---|---|----|

# C-Strings

- Array of `char`s can be used to define storage for string:
  ```
  const int SIZE = 20;
  char city[SIZE];
  ```

- Leave room for `NULL` at end
- Can enter a value using `cin` or `>>`
  - Input is whitespace-terminated
  - No check to see if enough space
- For input containing whitespace, and to control amount of input, use `cin.getline()`

**Program 10-5**

```cpp
 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**

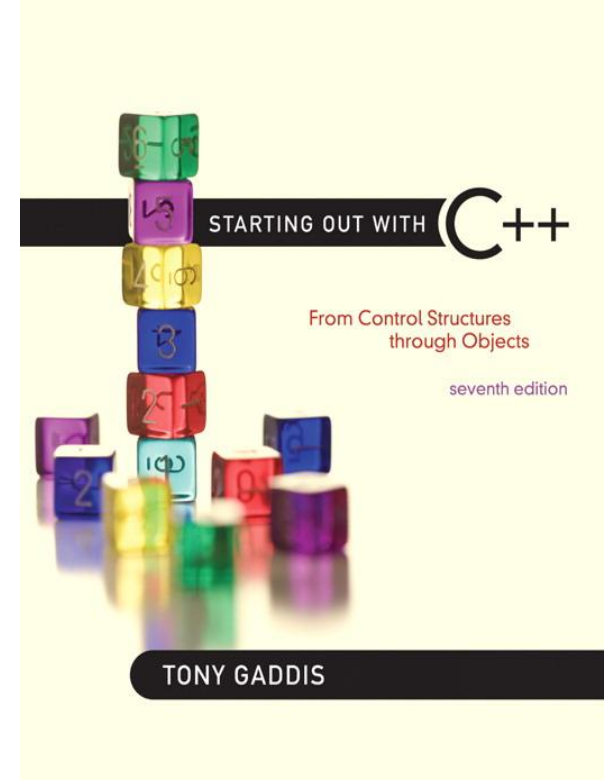Enter a sentence of no more than 79 characters:
**C++ is challenging but fun! [Enter]**
The sentence you entered is:
C++ is challenging but fun!

# 10.4

## Library Functions for Working with C-Strings

# Library Functions for Working with  C-Strings

- Require the `cstring` header file

- Functions take one or more C-strings as arguments.  Can use:
  - C-string name
  - pointer to C-string
  - literal string

# Library Functions for Working with C-Strings

Functions:

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

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

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

  ```
  char location[SIZE] = "Missoula, ";
  char state[3] = "MT";
  strcat(location, state);
  // location now has "Missoula, MT"
  ```

# Library Functions for Working with C-Strings

Functions:

– `strcpy(str1, str2)`: copies `str2` to `str1`

```
const int SIZE = 20;
char fname[SIZE] = "Maureen", name[SIZE];
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.

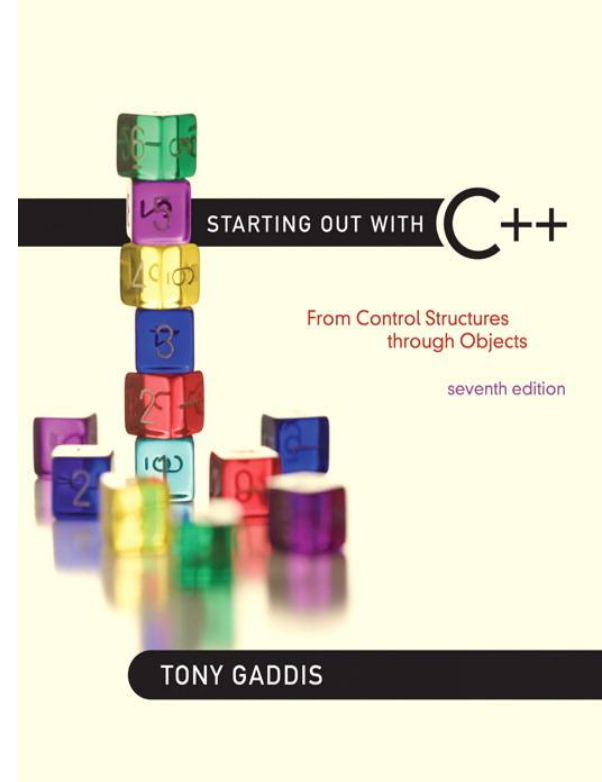# C-string Inside a C-string

Function:

- `strstr(str1, str2)`: finds the first occurrence of `str2` in `str1`. Returns a pointer to match, or `NULL` if no match.

    ```
    char river[] = "Wabash";
    char word[] = "aba";
    cout << strstr(state, word);
    // displays "abash"
    ```

# 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 iNum;
long lNum;
double dNum;
char intChar[10];
iNum = atoi("1234"); // puts 1234 in iNum
lNum = atol("5678"); // puts 5678 in lNum
dNum = atof("35.7"); // puts 35.7 in dNum
itoa(iNum, intChar, 8); // puts the string
   // "2322" (base 8 for 1234₁₀) in intChar
```

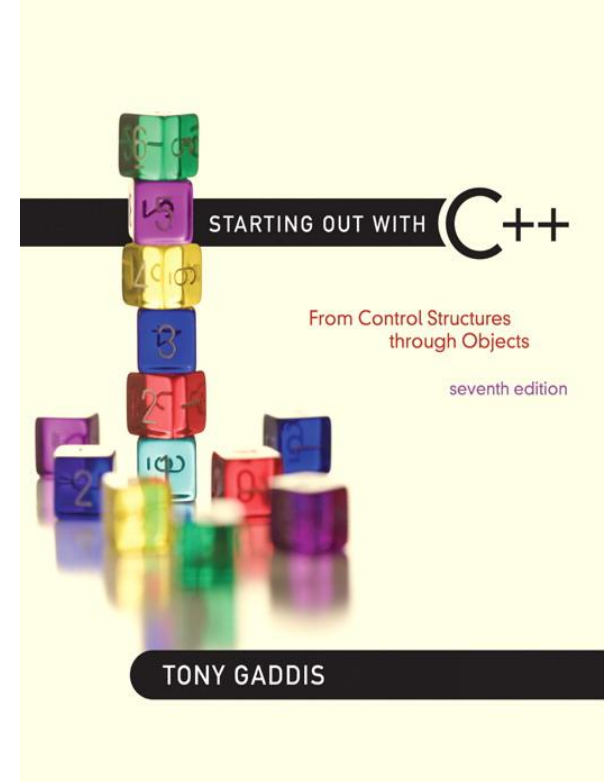# String/Numeric Conversion Functions - Notes

- if C-string contains non-digits, results are undefined
  - function may return result up to non-digit
  - function may return 0

- `itoa` does no 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 Own C-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

```cpp
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 data type supports working with strings
- `#include <string>`
- 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;
  ```

## Program 10-15

```cpp
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 `cin >>` to read an item into a string:

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

**Program 10-16**

```cpp
 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

# 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);
```

# `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**

```cpp
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++ strings

| 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";
for (int i = 0; i < 16; i++)
     cout << phrase[i]; // displays
                 // "Hot Dog on a bun"
```

**Program 10-20**

```cpp
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` Member Functions

- 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

# `string` Member Functions

```
string word1, word2, phrase;
cin >> word1;              // word1 is "Hot"
word2.assign(" Dog");
phrase.append(word1);
phrase.append(word2);   // phrase has "Hot Dog"
phrase.append(" with mustard relish", 13);
         // phrase has "Hot Dog with mustard"
phrase.insert(8, "on a bun ");
cout << phrase << endl; // displays
         // "Hot Dog on a bun with mustard"
```

## 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