Chapter 10:
Characters, C-Strings, and More About the string Class

10.1 Character Testing

• Requires cctype header file

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>isalpha</td>
<td>true if arg. is a letter, false otherwise</td>
</tr>
<tr>
<td>isalnum</td>
<td>true if arg. is a letter or digit, false otherwise</td>
</tr>
<tr>
<td>isdigit</td>
<td>true if arg. is a digit 0-9, false otherwise</td>
</tr>
<tr>
<td>islower</td>
<td>true if arg. is lowercase letter, false otherwise</td>
</tr>
<tr>
<td>isprint</td>
<td>true if arg. is a printable character, false otherwise</td>
</tr>
<tr>
<td>ispunct</td>
<td>true if arg. is a punctuation character, false otherwise</td>
</tr>
<tr>
<td>isupper</td>
<td>true if arg. is an uppercase letter, false otherwise</td>
</tr>
<tr>
<td>isspace</td>
<td>true if arg. is a whitespace character, false otherwise</td>
</tr>
</tbody>
</table>

From Program 10-1

```cpp
10 cout << "Enter any character: ");
11 cin.get(input);
12 if (isalpha(input))
13 cout << "The character you entered is: " << input << endl;
14 else 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

• 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 '!' 
  ```

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!"

Array of chars 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()`

Using C-Strings in Program 10-5

```
Program 10-5
// This program displays a string stored in a char array.
// Define size of array using constant.
// Using character array:
char names[] = "John Doe"; // array name
// Access characters in array using
// array name followed by index.
// Use cin to read string.
```

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
  ```c
  char city[SIZE] = "Missoula";
  cout << strlen(city); // prints 8
  ```
- `strcat(str1, str2)`: appends str2 to the end of str1
  ```c
  char location[SIZE] = "Missoula, ";
  char state[3] = "MT";
  strcat(location, state);
  // location now has "Missoula, MT"
  ```

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

String/Numeric Conversion Functions

- Requires cstdlib header file

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>PARAMETER</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>atoi</code></td>
<td>C-string</td>
<td>converts C-string to an int value, returns the value</td>
</tr>
<tr>
<td><code>atol</code></td>
<td>C-string</td>
<td>converts C-string to a long value, returns the value</td>
</tr>
<tr>
<td><code>atof</code></td>
<td>C-string</td>
<td>converts C-string to a double value, returns the value</td>
</tr>
<tr>
<td><code>itoa</code></td>
<td>int, C-string, int</td>
<td>converts 1st int parameter to a C-string, stores it in 2nd parameter. 3rd parameter is base of converted value</td>
</tr>
</tbody>
</table>
String/Numeric Conversion Functions

```c
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 1234_{10}) 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

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

- 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

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

From Program 10-10

```c
29 void nameSlice(char userName[]) {  
30     int count = 0; // Loop counter
31  
32     // Locate the first space, or the null terminator if there
33     // are no spaces.
34     while (userName[count] != ' ' && userName[count] != '\0') {
35         count++;
36     }
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:
  ```cpp
  string firstName, lastName;
  ```
- Can receive values with assignment operator:
  ```cpp
  firstName = "George";
  lastName = "Washington";
  ```
- Can be displayed via `cout`
  ```cpp
  cout << firstName << " " << lastName;
  ```

Using the string class in Program 10-15

Program 10-15
```
// This program demonstrates the string class.
#include <string>
#include <iostream>
using namespace std;

int main()
{
  string movieTitle;
  10.  movieTitle = "Wheels of Fury";
  11.  cout << "My favorite movie is " << movieTitle << endl;
  12.  return 0;
}
```

Program Output
```
My favorite movie is Wheels of Fury
```    

Input into a string Object

- Use `cin >>` to read an item into a string:
  ```cpp
  string firstName;
  cout << "Enter your first name: ";
  cin >> firstName;
  ```

Using cin and string objects in program 10-16

Program 10-16
```
// This program demonstrates how cin can read a string into
// a string class object.
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string name;
  11.  cout << "What is your name? ";
  12.  cin >> name;
  13.  cout << "Good morning " << name << endl;
  14.  return 0;
}
```

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

Other Definitions of C++ strings

<table>
<thead>
<tr>
<th>Definition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>string name;</td>
<td>defines an empty string object</td>
</tr>
<tr>
<td>string myname(&quot;Chris&quot;);</td>
<td>defines a string and initializes it</td>
</tr>
<tr>
<td>string yourname(myname);</td>
<td>defines a string and initializes it</td>
</tr>
<tr>
<td>string name(myname, 3);</td>
<td>defines a string and initializes it with first 3 characters of myname</td>
</tr>
<tr>
<td>string verb(myname,3,2);</td>
<td>defines a string and initializes it with 2 characters from myname starting at position 3</td>
</tr>
<tr>
<td>string noname('A', 5);</td>
<td>defines string and initializes it to 5 'A's</td>
</tr>
</tbody>
</table>

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"
```
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, front, back, at, substr
  - comparison: compare
- See Table 10-8 for a list of 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"

string Member Functions in Program 10-21

```
// This program demonstrates a string
// object's length member function.
#include <iostream>
#include <string>

int main ()
{
   string town;
   cout << "Where do you live? ":
   cin >> town;
   cout << "Your town's name has " << town.length() << " characters.
   return 0;
}
```

Program Output with Example Input Shown in Bold

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