Character Testing

- Requires \texttt{cctype} header file

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

Character Case Conversion

- Require \texttt{cctype} header file

- Functions:
  - \texttt{toupper}: if char argument is lowercase letter, return uppercase equivalent; otherwise, return input unchanged
  - \texttt{tolower}: if char argument is uppercase letter, return lowercase equivalent; otherwise, return input unchanged

```cpp
10  char ch1 = 'H';
11  char ch2 = 'e';
12  char ch3 = '!';
13
14  cout << toupper(ch1); // displays 'H'
15  cout << toupper(ch2); // displays 'E'
16  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 '!'
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

- `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

- `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

- `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(river, 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) 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

### The C++ string Class

- Special data type supports working with strings
- `#include <string>`
- Can define string variables in programs:
  ```c
  string firstName, lastName;
  ```
- Can receive values with assignment operator:
  ```c
  firstName = "George";
  lastName = "Washington";
  ```
- Can be displayed via `cout`
  ```c
  cout << firstName << " " << lastName;
  ```

### 10.7

#### More About the C++ string Class

### Using the `string` class in Program 10-15

#### Program 10-15

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

Program Output

My favorite movie is Wheels of Fury

### Input into a string Object

- Use `cin >>` to read an item into a string:
  ```c
  string firstName;
  cout << "Enter your first name: ";
  cin >> firstName;
  ```
Using `cin` and `string` objects in program 10-16

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  int main()
7  {  
8      string name;
9      cin >> name;
10     cout << "Hello " << name << endl;
11     return 0;
12  }
```

Program Output with Example Input Shown in Bold

What is your name? Peggy [Enter]
Hello Peggy

Input into a string Object

- Use `getline` function to put a line of input, possibly including spaces, into a string:
  ```cpp
  string address;
  cin >> address;
  ```

String Comparison

- Can use relational operators directly to compare string objects:
  ```cpp
  string str1 = "George",
  str2 = "Georgia";
  if (str1 < str2)
    cout << str1 << " is less than " << str2;
  ```

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 aname(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

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt;</td>
<td>extracts characters from stream up to whitespace, insert into string</td>
</tr>
<tr>
<td>&lt;=</td>
<td>assigns string on right to string object on left</td>
</tr>
<tr>
<td>+=</td>
<td>appends string on right to end of contents on left</td>
</tr>
<tr>
<td>+</td>
<td>concatenates two strings</td>
</tr>
<tr>
<td>[]</td>
<td>references character in string using array notation</td>
</tr>
<tr>
<td>&gt;, &gt;=, &lt;, &lt;=, ==, !=</td>
<td>relational operators for string comparison. Return true or false</td>
</tr>
</tbody>
</table>

String Operators

```cpp
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, reserve, size
  - substrings: find, front, back, at, substr
  - comparison: compare
- See Table 10-8 for a list of functions.

```cpp
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"
```