10.1 Character Testing

- The C library provides several functions for testing characters.
- Requires the `<cctype>` header file
- They take a char (or int as ascii) argument
- They return true or false and can be used as boolean expressions in if/while/etc.:

```c
char input;
...
if (isupper(input)) ...
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isalpha</code></td>
<td>True for any letter of the alphabet</td>
</tr>
<tr>
<td><code>isdigit</code></td>
<td>True for digit</td>
</tr>
<tr>
<td><code>islower</code></td>
<td>True for lowercase letter</td>
</tr>
<tr>
<td><code>isupper</code></td>
<td>True for uppercase letter</td>
</tr>
<tr>
<td><code>isalnum</code></td>
<td>True for letter or digit</td>
</tr>
<tr>
<td><code>isspace</code></td>
<td>True for space, tab, newline (aka whitespace)</td>
</tr>
<tr>
<td><code>ispunct</code></td>
<td>True for anything not a digit, letter or whitespace</td>
</tr>
</tbody>
</table>
10.2 Character Case conversion

- These take a char (or int), and return an int(!)
  - `toupper(c)`
    - if c is lowercase, returns (the ascii value of) its uppercase version
    - otherwise returns c
  - `tolower(c)`
    - if c is uppercase, returns (the ascii value of) its lowercase version
    - otherwise returns c
- Does NOT change argument (c)

```cpp
char x = 'A';
char y = tolower(x); // converts to char
cout << x << " " << y << endl;
```

Output: A a

10.3 C-Strings

- In any programming language, a “string” is a sequence of characters.
- In C++, a C-String is a certain way of representing a string in memory.
- A C-String is:
  - a sequence of characters (char)
  - stored in consecutive memory locations
  - ALWAYS terminated by a null character (\0, ascii=0)

```
Hi There!
```

- Functions that take a C-string as an argument do NOT require an additional parameter for the size.
  - The size is unnecessary, because the null char marks the end. It’s a sentinel!
  - Use a sentinel-controlled loop:

```cpp
int cstringLength (char cstr[]) {
  int count=0;
  while (cstr[count] != \0) {
    count++;
  }
  return count;
}
```
Operations over C-Strings

- **Cannot** use = or == on char[] (arrays: doesn’t work)
- **output: can** use << (!)
- **input: can** use >> (!)
  - input stops at whitespace (space, tab, newline)!
  - but input does NOT stop at end of array!
  - it puts the ‘\0’ at the end for you
- **input: can** use cin.getline(char s[], int n)
  - input stops at ‘\n’ OR after n-1 characters have been read

```
char cstr[10];
cout << "Enter a name: ";
cin.getline(cstr,10);
cout << "You entered: " << cstr << endl;
```

```
Enter a name: Tom Fox
You entered: Tom Fox
```

```
char cstr[300] = "Economics";
cout << strlen(cstr) << endl; //prints 9
```

```
int strlen (char* str)
```

- **Returns** the number of characters in a C-string (up to but not including the null char).

```
char cstr[300] = "Economics";
cout << strlen(cstr) << endl; //prints 9
```

```
C-string copy
```

- **char* strcpy (char *destination, char *source);**
- Copies source C-string to destination
  - destination is modified (a variable)
  - destination must be long enough
  - ignore returned value (destination is returned)
- Use strcpy to perform assignment for C-strings
- example:

```
char string1[13] = "Hello ";
char string2[7] = "World!";
//simulate: string1 = string2;
strcpy(string1, string2);
cout << string1 << endl;
```

Output:
```
World!
```
C-string compare

- int strcmp (char *str1, char *str2);
- Compares str1 and str2, using ascii values
  - if str1 and str2 are the same, returns 0
  - if str1 comes before str2 alphabetically, returns -1
  - if str1 comes after str2 alphabetically, returns 1
- Use strcmp to perform comparison for C-strings
- example:

```
char string1[13] = "Hello ";
char string2[7] = "World!";
// if (string1<string2)... if (strcmp(string1, string2) < 0)
cout << "Hello comes before World" << endl;
```

10.7 More about the C++ string class

- string is a data type provided by the C++ library.
  - Specifically it is a class (see chapter 13).
- string requires the <string> header file
  - <iostream> may work as well
- To define a string variable:
  - string name1;
  - name1 is called a string object.
  - initialized to the empty string (size is 0!)
- The representation in memory of a string object is hidden from the programmer.

Operations over string objects

- initialization using = with a C-string literal or string

```
string name1 = "Steve Jobs";
string name2 = name1;
```

- assignment using = with C-string literal or string

```
string name1, name2;
name1 = "Andre Johnson";
name2 = name1;
```

Operations over string objects

- output using <<

```
string name1 = "Steve Jobs";
cout << "Name " << name1 << endl;
```

- input using >>

```
string name1;
cout << "Enter your name ";
cin >> name1;
```

- input using getline

```
string name1;
cout << "Enter your name ";
getline (cin, name1);
```

note: not the same one as for c-strings

Empty string literal: \"\"
Operations over string objects

• comparing string objects: < <= > >= == !=  (alphabetical order using ascii values)

```cpp
string string1, string2;
string1 = "Hello ";
string2 = "World!";
if (string1 < string2)
    cout << "Hello comes before World" << endl;
```

• string objects can be compared to C-strings

```cpp
string string1;
cout << "Enter a word: ";
cin >> string1;
if (string1 == "Hello")
    cout << "You entered Hello." << endl;
```

More operations over string objects

• [n] subscript notation, returns char at position n
  or use string.at(n)--performs bounds check

```cpp
string string1 = "Hello ";
cout << string1[4] << endl;
cout << string1.at(1) << endl;
```

string class member functions

• string class has many member functions that operate over the string object (Table 10-7).
  • theString.length() : returns length of string stored in theString (can also use .size()).
    ```cpp
    string theString = "Hello ";
cout << theString.length() << endl; //outputs 5
    ```

• theString.append(str): appends str (string or c-string) to the end of theString
  • It changes theString!! (also changes its length)

```cpp
string theString = "Hello ";
theString.append(" World");
cout << theString << endl; //outputs: Hello World
```

Exercise

• Write a function countDigits that takes a string as an argument and outputs the number of digits it contains.

```cpp
int countDigits (string p) {
    int count = 0;
    for (int i=0; i < p.length(); i++) {
        if (isdigit(p.at(i)))
            count++;
    }
    return count;
}
```

• Now write it for C-strings.
Exercise (watchout)

- Write a function `toLowerCase` that takes a string `p` as an argument and returns a NEW string that is a copy of `p` with all of its uppercase letters converted to lowercase.

What is wrong with this solution?

```cpp
string toLowerCase (string p) {
    string newP;
    for (int i=0; i < p.length(); i++) {
        newP.at(i) = tolower(p.at(i));
    }
    return newP;
}
```

terminate called throwing an exception
Abort trap: 6