10.1 Character Testing

The C++ library provides several functions for testing characters.
Requires the `cctype` header file
These functions have this signature:
- `int isupper (int c);`
They take a char (or int) argument
They return non-zero for true, 0 for false.

```
char input;
...  
if (isupper(input)) ...
```
10.2 Character Case conversion

- `int toupper (int c)`
  - converts lowercase letters to uppercase
  - otherwise returns c
- `int tolower (int c)`
  - converts uppercase letters to lowercase
  - otherwise returns c
- Does NOT change argument

```
char x = 'A';
char y = tolower(x);
cout << x << " " << y << endl;  // 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
  - stored in consecutive memory locations
  - ALWAYS terminated by a null character (\'\0\')

```
Hi There! \0
```

C-String

- String literals are stored in memory as C-Strings:
  - “Jim Kase”, “A00123456”, “$2.35/lb”
  - they have type char[] (or char*)
- A C-String can be stored in a char array.
  - Make sure it is large enough for the null char
- Do not pass size to functions taking C-strings as arguments
  - Unnecessary, because the null char marks the end.

Operations over C-Strings

- Don’t use = or == on char[] (arrays: doesn’t work)
- input: can use >>
  - input stops at whitespace!
  - copying to memory does NOT stop at end of array
- input: can use `cin.getline(char *s, int n)`
  - input stops at \'\n\' or after n-1 characters have been read
- output: can use `<<`

```
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
```

```
Enter a name: Tom Johnson
```

```
Enter a name: Tom Johnson
You entered: Tom Johns
```
10.4 Library Functions for C-Strings

- Require the cstring header
- Function headers look like this: `func(char *s)`
- argument can be:
  - name of array containing a C-string
  - pointer variable holding address of a C-string
  - a literal string

C-string length

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

```cpp
char cstr[30] = "Economics";
cout << strlen(cstr) << endl; //prints 9
```

C-string concatenation

- `char *strcat (char *destination, char *source);`
- Appends a copy of the source C-string to the destination C-string
  - destination is modified
  - destination must be long enough
  - returns pointer to destination (can ignore)
- example:

```cpp
char string1[13] = "Hello ";
char string2[7] = "World!"
strcat(string1, string2);
cout << string1 << endl;
```

Output: `Hello World!`

C-string copy

- `char *strcpy (char *destination, char *source);`
- Use strcpy to perform assignment for C-strings
- Copies source C-string to destination
  - destination is modified
  - destination must be long enough
  - returns pointer to destination (can ignore)
- example:

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

Output: `World!`
C-string compare

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

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

Output: `Negative`

10.7 More about the C++ string class

- string is a data type provided by the C++ library.
  - Specifically it is a class.
- string requires the `<string>` header file
  - `<iostream>` may work as well
- To define a string variable:
  - `string name1;`
  - name1 is a string object.
- The representation in memory of a string object is hidden from the programmer.

Operations over string objects

- initialization using = with a C-String literal

```cpp
string name1 = "Steve Jobs";
// can do this with char arrays too:
char name2[20] = "Steve Jobs";
```

- assignment using =

```cpp
string name1, name2;
cout << "Enter a name: ";
cin >> name1;
name2 = name1; // can’t do with char arrays
```

- assignment of C-Strings to string objects;

```cpp
string name1;
name1 = "Andre Johnson";
```

- output using <<

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

- input using >>

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

- input using getline

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

Note: not the same one as for c-strings

still stops at first ‘\n’

(input still stops at whitespace!)
Operations over string objects

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

```cpp
string str1, str2;
str1 = "Hello ";
str2 = "World!";
if (str1 < str2)
    cout << "Hello comes before World" << endl;
```

• string objects can be compared to C-strings

```cpp
string str1;
cout << "Enter a word: ";
cin >> str1;
if (str1 == "Hello")
    cout << "You entered Hello." << endl;
```

More operations over string objects

• + concatenation of string objects:

```cpp
string str1 = "Hello ";
string str2 = "World!";
cout << str1 + str2 << endl;
```

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

```cpp
string str1 = "Hello ";
cout << str1[4] << endl;
cout << str1.at(1) << endl;
```

```cpp
string str1[0] = ‘h’; //this works
string str1[6] = ‘s’; //this gets ignored (6>=length)
string str1.at(6) = ‘s’; //this causes an error
```

Output:

```
Hello World!
```

Other ways to initialize string objects

```cpp
string greeting = "Hello!"; //Hello!
string name("william smith"); //william smith
string name1(name); //makes a copy //william smith
char cName[15] = "Grace Hopper";
string name2 (cName, 5); //first 5 chars of C-str //Grace
string stars (20, '*'); //book has args backwards //***************
string partial(name, 0, 4); //start: 0, length: 4 //will
```

String class member functions

• string class has many member functions that operate over the string object

• 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 object or c-string) to the end of theString.

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

• theString.append(n, ‘z’) : appends n copies of char to end of string

```cpp
string theString = "Hello ";
theString.append(2,'z');
cout << theString << endl; //outputs: Hello zz
```

• theString.substr(x,n): returns a new string, copies n chars starting at position x from theString.

```cpp
string string1 = "hello there";
cout << string1.substr(6,3) << endl; //outputs: the
```

Exercise

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

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