Strings – For your chatty programs

Chapter 4

Strings C and C++ style

Consider the following programs

- Create a program to store a customers first and last name
- Create a program that reads in a sentence and counts the number of vowels
- Create a program that allows for word processing
- Create a program that reads in a equation from a user
- These are require the storage of sentences

Strings

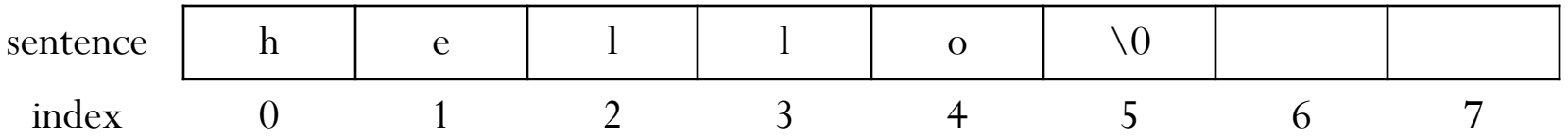
- A sentence in C++ is referred to as a string

- Definition of String
  - A sequence of characters

- So how do we store and represent a string?
  - Arrays
  - char myString[100];
  - Here a sentence of 99 characters can be accepted (99?)

Anatomy of a String

- char sentence[8] = “hello”;



- Each character takes a spot in the array
- Then end of the string is denoted by \0
  - Backlash zero
  - Note: without this it is not a string
  - Always consumes a position, thus a string of size n must have an array of minimal size n + 1
- Often positions are not used

Console IO and Strings

```
char word[100] ;
cout << “Type a word: “;
cin >> word;
cout << word << endl;
```

- Code reads in one word (must be less then 100 characters) and prints it back to console
- Any whitespace typed into the console breaks apart data
  - Thus “more c++ headacahes” must be stored into three separate strings
- Strings are the only type of array that does not require a loop to print its contents

Ways to initialize a string

```
char word[7] = "hello";   // string notation

char word[7] = {'h', 'e', 'l', 'l', 'o', '\0'};  // array notation

char *word = "hello";  // Using pointer notation
```

Strings can be a whole sentence

- char sentence[7] = “To be?”;

Sentence	T	o		b	e	?	\0
Index	0	1	2	3	4	5	6

Guided Example 7.1

Often raw data in a file will be a stream of numbers delimited by a character i.e. a file may look like the following:

2434:342

where the data is 2434, and 342

let

char numbers[20] = “2434:342”;

Parse out the two numbers and store them as ints. Assume there will be only two number, separated by a colon, but not always 2434 and 342.

Introduce New Functions

int atoi (const char * str);

- Given a valid string input that contains only digits, the corresponding integer that the string represents will be returned

Example 7.1 Breakdown

char numbers[20] = “2434:342”;

1. Split the string in half
   1. Need colon position
   2. Need size of string
2. Extract substring of first number
3. Extract substring of second number

Unguided Example 7.2

Write a program that accepts a sentence as an input from the user.

Do the following with the sentence.

- Count the number of vowels
- Change all upper case letters to a lower case

String class – C++ Style Strings

- Character arrays previously mentioned are c style strings
- C++ Style has the following advantages
  - Sizing of the string dynamically meets the size of the text. (less wasted memory)
  - Array overflow is not a concern
  - Large number of standard functions to perform string manipulation
    - Can use stringstream library
  - Overall, C++ strings are more user friendly than C style and less prone to errors
  - Disadvantage: dynamic nature makes their initial creation slower than c style

Example of the String Class

```
#include<string>

…

string word;
cout << "Enter a word: ";
cin >> word;
cout << word << endl;
```

- The above code outputs a word typed in by the user
- string is a class contained within the string library
  - treat string as a datatype much like int or float, though we will see there will be some key differences

Retrieving Sentence from Console

```
string sentence;
cout << "Enter a sentence: ";
getline(cin, sentence);
cout << sentence << endl;
```

- Reads until a newline is encountered (user presses return)
- cin is considered a data stream
- We can pass any data stream to getline which will be used later when we cover file IO

Important Methods of the String

- Define Method: a function associated to a class.
  - Use dot operator to access methods

- append
  - adds the passed string to the current string
  - Ex. word.append(“ added part”);
- length
  - returns size of string
- find
  - returns back the location of the first occurrence of a sequence of characters
- substr
  - returns back a substring based on a start position and size
  - Example

    string word = “programming”;
    word.substr(3, 4); yields “gram”

Methods of the String Cont.

- compare
  - compares two strings to determine which is greater based on dictionary order
  - Example

    string word = "put“, word2= "get";

    word.compare(word2); returns 1 since *p* comes after *g*
- c_str
  - converts string class to c-style string (char array)
  - some standard functions only work with c style strings, thus this conversion is useful
- testing equivalence is still ==
- For complete detailed list:
  - http://www.cplusplus.com/reference/string/string/

Anatomy of a C++ String

```
Data:
char *buff;     // Character String
int buffSize;       // size of character string

Functions:
c_str();
length();
compare();
….
```

string class

- string class is like a container for the data and functions associated to the string
- Each container represents one string
- Importantly, underneath all the magic is a c-style string array, just all the book keeping is done for you

Strings and array notation

```
unsigned int i;
string word = "even this string is really just an array";
for(i = 0; i < word.size(); i++)
    cout << word[i]; // Prints string one char at a time
cout << endl;
```

- like c-style, strings can be accessed using array notation
- this allows direct manipulation to the string
- notice we use the method size to determine how large the string is
  - **YOU CANNOT USE .size() ON NORMAL ARRAYS!!!!**

Guided Example 7.3

Often raw data in a file will be a stream of numbers delimited by a character i.e. a file may look like the following:

2434:342

where the data is 2434, and 342

let

string numbers = “2434:342”;

Parse out the two numbers and store them as ints. Assume there will be only two number, separated by a colon, but not always 2434 and 342

Example 7.3 Breakdown

- Here we are using the string class which we will leverage to our advantage.
- Locating colon can be achieved through find()
- String can easily be broken apart using substr()
- Using a similar algorithm to 7.1 we can use atoi() to convert the string for us

Unguided Example 7.4

Have the user enter two words into two separate strings. Take the two strings and combine the two words as a sentence and store them into a single string

Maintain a space between the two words

Place a period at the end of the string

Capitalize the first letter

Ensure all other letters are lower case

You may use the functions tolower() and toupper()