CS 122 Lab 4

C++ string

Paul Cao

C++ string is currently the best data structure that we have to process text data. In this lab, we will learn practice the use of it by simulating a telegraph machine. The code for this lab is due before midnight on 2/10/2011.

Pre-Lab

Read the following paragraphs about using C++ strings. You don't have to turn in anything for the pre-lab section.

The C-strings were simply arrays of characters terminated with the null character, `'\0'`. To manipulate these strings you used some functions. In almost all cases, you as the programmer need to somehow keep track of number of elements that you have stored in a C-string. The standard string class is defined in the library <string> and the definitions are placed in the std namespace. The class string allows you to treat string values and string expressions very much like values of a simple type. For example, when `s1`, `s2`, and `s3` are objects of type string, to concatenate string `s2` at the end of string `s1` and to store the resulting string into `s3`, we will use:

```
s3 = s1 + s2;
```

Also, to initialized a string s4, we no longer need strcpy function and we can use:

```
s4 = "Hello World";
```

The string class has a constructor that initializes the string to empty and one that initializes the string to a desired string. Following are the examples for these two:
The first one uses a constructor to create the empty_string as an empty string and the second one initializes the string to "hello". The following two lines are equivalent:

```
string empty_string;
string something("hello");
```

Here is an example in which a string class is used:

```
#include<iostream>
#include<string>
using namespace std;
int main( )
{
    string phrase;
    string adjective("fried"), noun("ants");
    string wish = "Bon appetite!";
    phrase = "I love " + adjective + " " + noun + "!";
    cout << phrase << endl
    << wish << endl;
    return 0;
}
```

As you may have noticed, in this program "+" is overloaded in the string class such that it now does the concatenation of strings.

The standard string class has several functions that make things very easy. A list of these functions appears is given in the textbook and can also be found at http://www.cplusplus.com/reference/string/string/.

The following program illustrates a few examples.

```
#include<iostream>
#include<string>
using namespace std;
int main( )
{
    string phrase,temp1, temp2;
    string adjective("fried"), noun("ants");
    string wish = "Bon appetite!";
    phrase = "I love " + adjective + " " + noun + "!";
    cout << phrase << endl << wish << endl;
    // copy phrase to newphrase
    string newphrase(phrase);
    // append wish to the end of newpharase
    newphrase += wish;
    // separates the first sentence
    temp1 = newphrase.substr(0, 18);
    cout << temp1 << endl;
    // separates the second sentence
    temp2 = newphrase.substr(18, 10);
    cout << temp2 << endl;
    return 0;
}
```

In the above program the statement:
`temp1 = newphrase.substr(0, 18);` and
`temp2 = newphrase.substr(18, 10);`

use the `substr(position, length)` function which returns the substring of the calling object starting at position and having length characters.

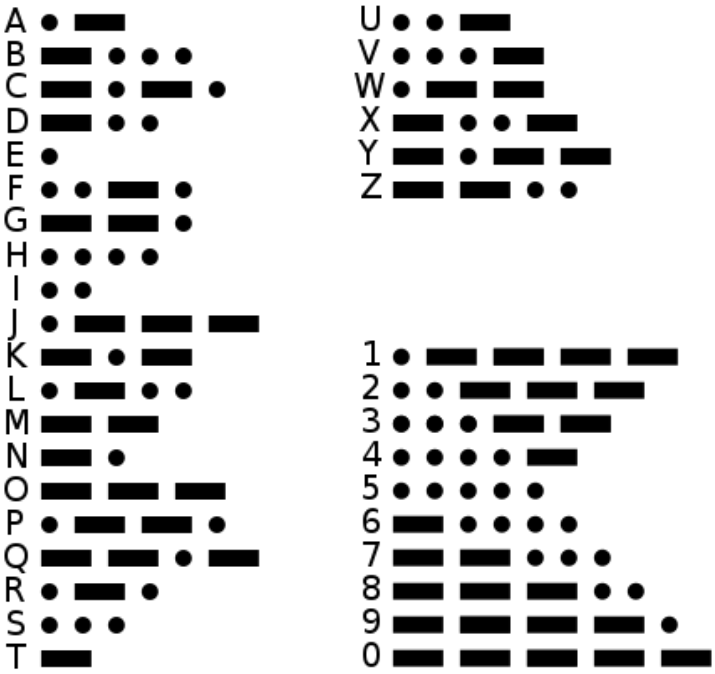
Also when dealing with the input of C++ strings, getline function is usually used when a whole line is needed to be read into a string variable.

In-Lab

In this lab, we will write a code to simulate a Morse code telegraph machine. Morse code is a code where each letter of the English alphabet, each digit, and various punctuation characters are represented by a series of dots and dashes. The following figure shows parts of the code. As you can see, Morse code isn't a case sensitive code thus it is impossible to tell which letter is capitalized at the receiver side.

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.



Write a code that accepts a sentence from the user and convert the sentence into its corresponding Morse Code. Display the Morse code on the console and generate a series of beeps to mimic the Morse code machine. The Morse code is given in a file called morse.txt and it is available in the lab folder. In this file, dot is represented as a period and dash is represented as a dash. There is no space between the . and – for each character while there is exactly one space between the character and its Morse code. In this file, all characters are in their capital form. You can assume that the input sentence won't contain any punctuation or any digit.

In order to generate beeps that simulate the beeping sound, you can use the Beep function available in the Windows API directive. This function requires two integer inputs with the first number for the frequency of the beep and the second integer is for the duration of the beep. The following code will generate a sound at frequency 5000Hz for 300 milliseconds (i.e. .3 second) followed by a beep at 2000 Hz for 900 milliseconds.

```
#include <iostream>
#include <Windows.h>

using namespace std;

int main(){
    Beep(5000, 300);
    Beep(2000, 900);
```

```
    return 0;
}
```

The rule to generate the beep is given in the figure in the previous page. For dots, their frequency should be 5000Hz while 2000Hz for dashes. The duration of a dot should be 300 milliseconds. The duration of a dash should be 3 times that of a dot. For the silence between symbols, you can use Beep function to beep at a frequency that is inaudible for human (anything that is below 20Hz). This will ensure the timing synchronization.

For example, in order to generate the Morse code sound for the following sentence
Attack him

The Morse code should be

A				silence		t		silence		t		silence		a				silence		c								silence	
.		-				-				-				.		-				-		.		-		.			
k						silence						h								silence		i				silence		m	
-		.		-								.		.		.		.				.		.				-	
-																													

Note
- It is easier if you define three functions with one for the display and visualization of dot, one for dash, and one for the silence period. For the silence function, it is better that it is only for one period of 300 milliseconds. For longer silence periods, a loop can be used.
- The display of the Morse code should be synchronized with each character.
- You can assume that the sender will use exactly one space character between two words and wont' start or end the sentence with a space.
- To avoid complicated file operations, it is better that the morse.txt is opened and closed for every search.
- Don't forget that the user may input lower case letters but the morse.txt only has the morse code for letters in the capital form.

I have uploaded the executable exe file on angel so you can download it and see the behavior of the code by yourself. To run this exe file (or any exe file we have built), you need to go the folder containing this file through the command prompt (can be found in the accessories in the start menu). Then simply type the name of the exe file. In this problem, the morse.txt file must be in the same folder of the exe file.

Sample Output

Welcome to the C++ Morse Telegraph Machine!

Please enter a sentence (no punctuation or digits): tell paul this code is fun
t -
e .
l .-..
l .-..

p .--.
a .-

u ..-
l .-..

t -
h ....
i ..
s ...

c -.-.
o ---
d -..
e .

i ..
s ...

f ..-.
u ..-
n -.
Your telegraph is sent. Please pay at Paul Cao's office. Thanks for your business.