ELEN2004: Software Development I

Laboratory Assignment No 5

Due Date: Monday April 20th, 2015, By 07:59Hrs

Outcome

After the assignment is completed we should understand better the following:

1. Use of String functions.
2. Use of for-loops/while-loops, etc.
3. The use of a simple Makefile to organise our programs
4. The use of clock() function to estimate the execution time of your coded algorithm.

The Work Schedule

Given a string **y** of length n and a substring **x** of length m, the task is to write a program in C++, naturally, to count the number of occurrences of the substring **x** in **y** and the start positions, within the string **y** that these occur, relative to the first character of string **y**. In the course of counting and detecting these positions, you are required to estimate the time interval that your algorithm takes to run. The program is to be nicely organised and run with help of a simply **makefile**.

Detailed Description of the Task

You are given two files for this assignment:

1. The first is called "ecoliSmall.txt" and holds the string **y**
2. The second is called "testStrings.txt" and holds a number of the substrings, one per line.

The actual length of the string **y** and each of the substrings should be determined by your program. You are to write the following functions and a main program. Lets call the main program "searchStrings{}:

int myBFSearch(int startpos, string &x, int m, string &y, int n);
myBFSearch() uses a brute force search algorithm, as discussed in class, to return the position of an occurrence of **x** in **y** with the search starting at startpos. If no occurrence is found it returns -1.

int CStringSearch(int startpos, string &x, int m, string &y, int n);
This does the same as in myBFSearch(), except that it uses the string functions in C++. Some of these functions are mentioned in the textbook for the course.

int main():

This is the main program that invokes myBFSearch() and CStringSearch() algorithm to determine the occurrences of the substrings **x** in **y**.

makefile:

The program compilations should be driven by a "makefile" Each of the above functions should be coded in an independent file. All header files should be put in a single .h file.

For each of the substrings **x**, the number of occurrences in the string **y** should be determined independently by myBFSearch() and also by CStringSearch(), as well as the time each function took to find all the occurrences.

The output can be tabulated in a nice layout that looks like (but not exactly) what is shown below. It may be captured in an output file or sent to cout with the result piped into a text file.

Substring	No Found	Time of	No Found	Time of CStringSearch
	myBFSearch	myBFSearch	CStringSearch	CStringSearch
substring1	...	...	...	...
substring2	...	...	...	...
...	...	...	...	...

The Deliverables

- Submit a zipped or a tarred file that includes the files for the .h file, your function files, the file of your your main program and the output text of your result. Please do not send a set of independent .cpp files.

- You DO NOT need to include the text of the pseudo-code developed for your functions.

Submission should be under the name of the **first student** of the pair of students that worked together.

Sample Code to Time a Segment of Your Code

Here is a sample program that does some timing of a segment of code that does nothing useful.

```
#include <iostream>
#include <iomanip>
#include <ctime>
#include <cstdlib>
using namespace std;
int main()
{
    double x;
    srand(time(NULL));

    clock_t startTime = clock(); //First Clocked

    // This is just to compute something consume time
    for (int i = 0; i < 1000000; i++)
        x = ((double) rand())/RAND_MAX;

    clock_t finishTime = clock(); // Second Clocked

    clock_t duration = finishTime-startTime;
    std::cout << "time interval (proc time) = " << duration << std::endl;
    std::cout << "time interval (milliseconds) = "
        << ((double)(duration)/CLOCKS_PER_SEC)*1000 << std::endl;
    return 0;
}
```