CSC 161 - Lab 2 – Due Wednesday 2/12 by 11 am

Searching

Background

Recall we can ask our program to generate random numbers with the following lines of code:

#include <math.h> // This library lets us access random numbers

#include <ctime> // This library lets us access the computer's clock

At the top of the main:

int time = clock(); // Stores the computer's time in the variable time

srand(time); // Primes the pump for the random number generator

Then, in the main, whenever we type rand(), we will get some random integer. Since we often want to have some control as to how large the random number will be, we can use the modulus operator as below:

num = rand()%10; //stores a random number between 0 and 9 in num

Lab Assignment

Grab the file search.cpp which contains some setup code (as above) to get you started with this program.

Prompt the user to enter a number, and then declare an array of integers of that size, and fill it with ASCENDING random numbers. One way to do this (and not have the numbers get too large) is to put a random number in the first slot. Then to fill each slot, generate another random number and add it to the number in the previous slot, putting the result in the current slot. If you mod by something smallish (like 10), you’ll never be adding more than 9 each times, so the growing sum won’t get too large.

Once you’ve filled your array, print the entire array to the screen, in a reasonable format.

Then, launch into a menu loop with three options:

1. Linear Search – if the user picks this option, you should print the smallest and largest numbers in your array (which should be easy to do) and then prompt them to enter a number in between to search for. You should perform the search linearly, jumping out as soon as you either find the number or go past where the number would be located. Keep track of how many times your searching loop runs and once it finishes, print a message indicating whether or not the number was found and how many times your loop ran.
2. Binary Search – same as above, except perform the search according to the binary search algorithm.
3. Quit – you can end your program

To begin, spend at least 10 minutes thinking and then write (on paper) pseudo code that details your full plan for coding this task. This should include the variables and data structures you plan on using, as well as a detailed plan for how you’re going to perform these search options. Obviously, you should test your code many, many times to make sure it’s bug free and does what you expect it to do.

Lastly, check to see that your program is well documented (with comments, including one that has your name and program description at the top), and also check to see if it’s as efficient as possible.

BONUS: Do at least 10 trial runs through each of your two searching algorithms, with significantly varying sizes for your array and summarize your results in a chart where you list the size of the array and how many loop runs were needed to find/not find the user’s value in both the linear and binary search cases. Compare these to how many runs you expected to need (n/2 for linear searching, log2(n) for binary searching). Discuss your results and any conclusions. You can do this on paper (maybe in an Excel chart) or upload a file to the Canvas assignment.

SUBMIT to this Canvas assignment two things:

1. Your .cpp file
2. A scan of your pseudo code. Alternately, you can bring this to class on Wednesday and give me a paper copy.