CSCI 136 Assignment – C++ Starter
April 12th, 2023
Due Date: Tuesday, 4/18/2023 11:59PM
Points: 30
Topic: Getting Started with C++ (and Visual Studio)

This assignment will be challenge for several reasons. Even though the programs you are tasked with are relatively simple, you will need to get up to speed with Visual Studio and C++. Don't underestimate what it takes to do this. By design, this assignment is set up so that you will have to "figure out" some of this. Knowing what to look for and where to seek help are key. Your Python experience will help you along with understanding that C++ is different and Visual Studio is different than Spyder or simple editing with Notepad++. Further guidance will be given on Friday, and you will better appreciate that guidance after getting a head start on the lab. Do your best to make progress while in the lab. Quite a bit of help is given in the Helpful Hints section.

In this assignment, you will gain experience working with C++ on several small programs. Each was chosen to give you practice working with different C++ constructs. You must use Microsoft Visual C++ on this assignment. Follow examples shown in class to create a new project (and solution) for each program. While there are multiple files in a Visual Studio C++ project, your focus is the C++ source file for each program. These source files will have the extension .cpp. Name your new projects to match the first part (without the .cpp) of the target file name and the correctly-name .cpp file will be created as part of the project.

After following all instructions below, please submit your C++ source files in Moodle. Only submit the .cpp files, not the project files or executables. Upload these as individual files – do not zip them. Be sure the source files have header sections that include the file name, your name, credits, a description of the file, and other useful information. Make sure your code is commented in general and thoroughly.

- `Distance.cpp`
- `GuessUntil.cpp`
- `GuessThree.cpp`

Grading:

This assignment is worth 30 points total with each program worth 10 points. You will be graded according to the following criteria:

Grade Item	Points
Distance.cpp Compiles	2
Distance.cpp Runs	2
Distance.cpp Comments are Correct and Complete	2
Distance.cpp Runs Correctly	4
GuessUntil.cpp Compiles	2
GuessUntil.cpp Runs	2
GuessUntil.cpp Comments are Correct and Complete	2
GuessUntil.cpp Runs Correctly	4
GuessThree.cpp Compiles	2
GuessThree.cpp Runs	2
GuessThree.cpp Comments are Correct and Complete	2
GuessThree.cpp Runs Correctly	4
Total	**30**

Distance.cpp

Calculates and prints the distance between two points whose coordinates are provided as command line arguments.

The straight-line distance between two points (x1, y1) and (x2, y2) in a Cartesian coordinate plane is given by the equation:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Your program will take in the values of the x1, y1, x2, y2 points as integers from the command line in that order, separated by spaces. It must then calculate the (floating point) distance between the two points. For example, if the command line arguments are `0 0 3 4`, your program will calculate and output the distance between (0, 0) and (3, 4) which is 5.0.

In C++, to take the square root of a number use the `sqrt()` function.

The output from your program should be formatted exactly as below, with the integer coordinates and the calculated distance at the end:

```
The distance between (0, 0) and (3, 4) is 5.0
```

Name this program Distance.cpp.

GuessUntil.cpp

Generates a random number between 1 and 100 and asks the user to guess the number, and continue guessing until they get it right.

Write a program that generates a random number between 1 and 100. The user can enter guesses until they get it right. To do this, you need a way to get interactive user input into your program. The user should know what they are expected to enter, so you should print out a line (a prompt) that tell the user what is expected.

```
Enter a number between 1 and 100:
```

Your program should use some form of a while loop to continue getting user guesses until they finally guess the number that was randomly generated. You can decide whether a while loop or a do...while loop is more appropriate. At each user guess, you should let the user know if their guess was too high or too low. When the user finally guesses the correct number, print out:

```
You got it!
```

Name this program GuessUntil.cpp.

GuessThree.cpp

Generates a random number between 1 and 10 and asks the user to guess the number in no more than three guesses.

This time the user only gets three guesses. Since the number is between 1 and 10, chances are reasonable that they will guess the number, but it isn't a sure thing. Use a for loop in this program. Each time the user guesses wrong, you should print out whether the guess was too high or too low. If they get it right you should print out "You got it!"

If the user guesses correctly before their three guesses are used up, you should exit the loop without using the break command or similar.

Here is an example output when the user guessed on the second try:

```
Enter a number between 1 and 10: 5
Your guess was too low.
Enter a number between 1 and 10: 7
You got it!
```

Name this program GuessThree.cpp.

Helpful Hints:

Do I need to follow the given specifications? Yes. You must implement the individual programs as specified.

Do I need to declare variables and their types? Yes. C++ is strongly typed. You must declare variables and the declaration includes the type.

What variable types will I need to use? At the very least, you will use `int` and `float`. You will also use strings, although you might not have to directly declare them. Note the mention and use of `char *` below. You might also want to use `bool`.

How do I concatenate strings in C++? The concatenation operator for strings is `+`. Realize that `int` or `float` variables may need to be first converted to strings before concatenating them with strings.

What should I do when I get compilation errors? Look at the first error and try to fix it. Often fixing the first error will eliminate subsequent errors. Also, the compiler may give you the line and column number of where the error occurred, and Visual Studio will highlight errors. Pay attention to this. If you can't find the error on the corresponding line, look at lines preceding it.

How do I get command line arguments into my program? You will need to modify the definition of the main function as follows:

```
int main(int argc, char *argv[])
```

Similar to Python, the `int` argument `argc` will be the count of the arguments read from the command line, including the path of the program, and `argv` will be an array of (string) arguments. (Technically `argv` is an array of pointers to characters which form the strings.) If these command line arguments are to be treated as numeric values, you will need to convert them from strings to numbers. See below.

How do I pass command line arguments to the executable produced by compiling my program? There are several ways to do this. In the Solution Explorer in Visual Studio, right click on the project, which is one level beneath the solution. Select Properties and then Debugging beneath Configuration Properties. You can then provide command line arguments via the Command Arguments property, and these arguments will be passed to the executable when Visual Studio runs it. Another way is to run the executable built by your project in a command window and provide it directly with the command line arguments. Doing this is ultimately how command line arguments would be used with your program, but requires you to bounce back and forth between Visual Studio and a separate command window when developing and debugging your program.

How do I convert a string or `char *` to an `int`? The `stoi()` function from the standard library (`std` namespace) can be used to convert a `char *` string to an `int`.

How do I convert an `int` to a string? To convert an `int` to a string, use `to_string()`, which is available via `#include <string>`.

What should I include at the top of my programs? Some combination of the following will suffice for these programs.

```
#include <iostream>
#include <string>
using namespace std;
```

You will also need the following to set the seed for the random number generator. See below.

```
#include <time.h>
```

Don't forget to include a header comment with important information about the file at the top of your .cpp source files.

How do I generate random numbers in C++? The `rand()` function is in the standard library and generates a pseudo-random integer between 0 and `RAND_MAX`, a "large" integer. The `srand()` function is also in the standard library and sets the seed for the pseudo-random number generator. If you set the seed to the same value every time, you will get the same sequence of random numbers. Using the system epoch time to set the seed is a common way to help ensure that you get a different sequence of random numbers in subsequent runs. To get a random number in the range of 1-100, for example, use the code below:

```
#include <time.h>
using namespace std;
...
// Seed the random number generator with the current time.
srand(time(NULL));
// Generate a random integer between 1 and RAND_MAX, mod by 100
// and add 1 to get get an integer in the range 1-100.
int number = rand() % 100 + 1;
```

Setting the seed to a set value can be helpful in debugging in order to get replicable results. Use the approach above to ensure different numbers on different runs.

How do I get user console input in the form of an integer? The following block of code will prompt the user to enter a number and will store the input they provide in an integer variable. (This is not the same as reading command line arguments. See above.)

```
cout << "Enter a number between 1 and 100: ";
int guess;
cin >> guess;
```

Do I need to do exception handling in my programs for this lab? Exception handling is above and beyond what is expected in this lab. If you can't check for potential errors via simple if statements, then you aren't expected to handle them. One example where a simple if statement could be used is to check if the number of arguments passed to your program matches what is expected. An example of something you aren't expected to do is check whether a command line argument is an integer – for this lab, assume what is provided is correct.

Do I need to include math.h to use sqrt()? Using the std namespace suffices for this as well as for rand() and srand() with our defaults in Visual Studio. In other implementations, this could depend on the version of C++ as well as compiler flags.

How do I make a for loop exit based on a condition other than or in addition to checking for the end of a range of values? for loops in C++ are very flexible. The condition part of a for loop that governs continued operation or exiting from the for loop is a boolean expression that can consist of multiple clauses joined by logic operators such as && and ||. In addition to the check for the end of a range, you could check to see if some other condition is satisfied (or not).

Submit Your Work

After following all instructions above, please submit your C++ source files in Moodle. Only submit the .cpp files, not the project files or resulting executables. Upload these as individual files – do not zip them. Be sure the source files have header sections that include the file name, your name, credits, a description of the file, and other useful information. Make sure the code is thoroughly commented.

- `Distance.cpp`
- `GuessUntil.cpp`
- `GuessThree.cpp`

EXTENSIONS

You will not submit the answers to these extensions as part of your assignment submission. You should work through and understand them to learn more and practice. You may be held responsible for the content within, so you really should work through them. With that said, this are outside the scope of what will be submitted for this lab. Do not change your original submittal to address these extensions.

1. Teach yourself more about C++. Think about what is different about C++ from Python.

2. Teach yourself more about Visual Studio. What more can it do to help develop C++ projects? What other projects and language types does it support?