

## CGT 215

### Lab 2

#### Programming Math Problems

##### Learning Objectives:

- Learn how to program mathematical expressions.
- Implement simple basic user input
- Implement simple outputting of variables

##### Description

In this lab, you will be using C++ to solve mathematical expressions based off user inputted values. This will involve using ***cin*** for input and ***cout*** for output.

##### Deliverable:

The .cpp file that you do your implementation of the lab in submitted to brightspace.

## Outline:

- 1) Create a new visual studio C++ console app project in visual studio and connect everything up to github just like in the first lab. This will be the method for starting most labs in this class from now on.
  - 2) Replace the template code with this template code if you would like somewhere to start from:
- 

```
#include <iostream>

using namespace std;

int main()
{
    float A;
    float B;
    float X;

    cout << "Please enter a value for A: ";
    cin >> A;

    cout << "A=" << A << endl;

    B = (A * 4) + 1;
    cout << "And B=" << B << endl;
}
```

---

What does all this mean though? Let's break it down:

```
using namespace std;
```

What this line *does* is far less important than what it lets *you do*. It tells the compiler that you would like to be inside the standard library namespace. This allows you to use the input and output functions ***cin*** and ***cout*** without needing to prepend them like ***std::cin*** or ***std::cout***.

```
float A;
float B;
float X;
```

This declares 3 variables of type float called A, B, and X. Remember float means that they can hold decimal numbers

```
cout << "Please enter a value for A: ";
cin >> A;

cout << "A=" << A << endl;

B = (A * 4) + 1;
cout << "And B=" << B << endl;
```

**cout** is how we print to the terminal in C++ and **cin** is how we get user input from the terminal. I like to think of cout as the screen and cin as the keyboard. Then the arrows point in the direction that the data is moving i.e. if you want to move a string or variable to the string, the arrows point towards **cout** and if you want to get input from the user's keyboard, the arrows point from **cin** to the variable that you are storing the input inside of.

**endl** takes us to the next line on the terminal, you can think of it like pressing the enter key to insert a carriage return. Without it, the text would all output on the same line.

The last two lines deal with doing mathematical operations to A and storing that value into B then outputting B just as we output A before.

- 3) Run the template code and play around with it modifying it until you have a good grasp on how the syntax for input and output works. It's always okay to try things and see what happens, you won't break anything!

- 4) Solving a linear equation:

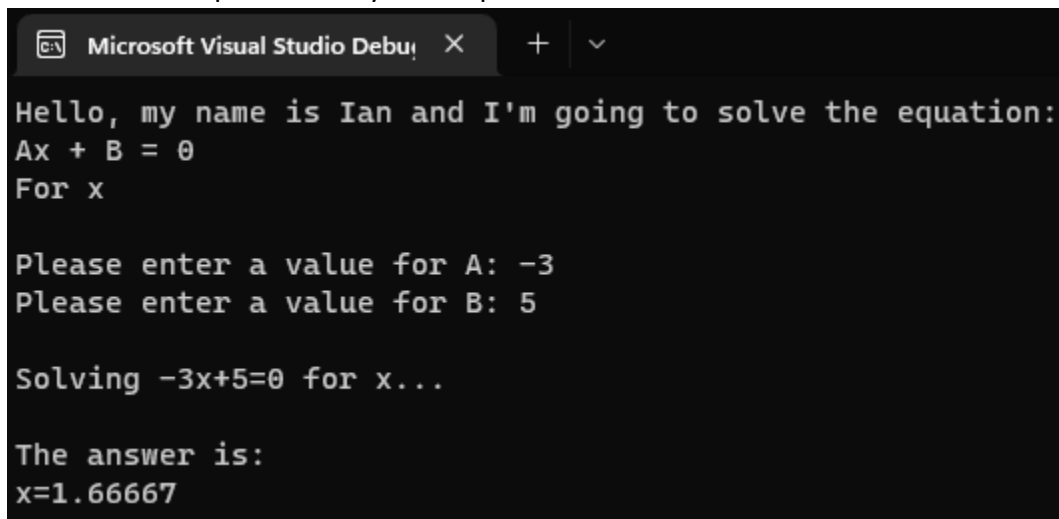
We ultimately want to solve the linear equation

$$Ax + B = 0$$

For x given user input values for A and B.

Your real task for this lab is to come up with a program that will ask the user for A and B then output the equation it is solving followed by the answer.

Here is an example of what your output should look like when done:



```
Microsoft Visual Studio Debug Console
Hello, my name is Ian and I'm going to solve the equation:
Ax + B = 0
For x

Please enter a value for A: -3
Please enter a value for B: 5

Solving -3x+5=0 for x...

The answer is:
x=1.66667
```

- 5) Submit your .cpp file from brightspace. You can open the file explorer to the project directory by right clicking inside the solution explorer and selecting "Open Folder in File Explorer"

## Grading:

Of the correct output portion of your grade, this is the breakdown:

Outputs a greeting showing the equation that will be solved: 25%

Prompts the user for input before getting each variable: 25%

Shows the equation to be solved with the user input substituted in: 25%

Correctly calculates and outputs the right answer: 25%

## Closing Thoughts

It often helps to play around with things you aren't familiar with to see how they work. Knowing precisely what a program will do before you run it is key to writing good code, and the best way to know is to learn by doing.

If you get stuck, know that searching online for syntax and assistance in programming is an invaluable way to learn. As long as you implement things yourself, looking up the syntax or ideas of how to solve a problem isn't going to count against you. Copying someone else's solution code to a problem however constitutes academic dishonesty in programming.