632666 Lab Book

# **Week 1 – Lab A**

Date: 4th Feb 2022

## **Q1. Hello World**

### **Question:**

Locate the **Solution Explorer** within Visual Studio and select the **Hello World** project. Right click on this project and select **Build**. This should compile and link the project. Now run the Hello World program.

Change between **Debug** and **Release** mode. Compile again and rerun the program.

### **Solution:**

## **Q2. Creating a new project**

### **Question:**

Write a program to input a Fahrenheit measurement, convert it and output a Celsius value. Also, what happens if you divide two integers?

### **Solution:**

### **Test data:**

Input 1: 50

Input 2: 100

### **Sample output:**

Output 1: 10

Output 2: 37.7778

### **Reflection:**

Initially, the code was written using integer types and this caused the output to always be 0. This is because 5 divided by 9 is 0.5 which can’t be stored fully in an integer so it would be rounded down to 0. To overcome this, I used floats to make sure that all values retain their precision.

## **Q3. Types**

### **Question:**

Using the “Hello World” program as a starting point, write a program that prints out the size in bytes of each of the fundamental data types in C++.

### **Solution:**

### **Test data:**

N/A

### **Sample output:**

Size of types

Double: 8

Float: 4

ULong Int: 4

Long int: 4

U Int: 4

Int: 4

U Short: 2

Short: 2

U Char: 1

Char: 1

### **Reflection:**

Using “sizeof()” will return the size of any data type in bytes so this is a good way to record how much memory you are using and ways you can reduce the amount of memory you’re using. For example, a float uses half as much memory as a double so if you need to store a decimal number with a relatively small number of a decimal points then a float is the more memory efficient option.

## **Q4. Floating point precision**

### **Question:**

How small does y have to be before you get a “divide by zero” error? Does the value of x affect the result?

### **Solution:**

### **Test data:**

N/A

### **Sample output:**

“divide or mod by zero” error

### **Reflection:**

Y and Z will only be identical if using an epsilon value to compare them. Comparing them as they are originally shows that technically they are different because of a slight difference in decimal values. Using an epsilon value allows you to compare the variables to a certain decimal place. Increasing the x value made the y value smaller and giving x the value of 0 created a divide by zero error.

## **Q5. C#/C++ Iteration Comparison (for loop)**

### Text Description automatically generated**Question:**

Port the above C# code in to C++ using the provided Main.cpp file

### Text Description automatically generated**Solution:**

### **Test data:**

factorialNumber 1: 4

factorialNumber 2: 5

### **Sample output:**

Output 1: 24

Output 2: 120

### **Reflection:**

The syntax for a for loop in C++ is identical to C# so no changes needed to be made here. The only change I needed to make was the “Console.WriteLine()” to “cout”.

## **Q6. Calculate an average using iteration (while loop)**

### **Question:**

Using a while loop (or do-while loop), calculate the average value of values provided by the user from the console (cin). You should calculate the average after the user either enters a negative number or the user enters a non-number value (e.g. a letter).

### **Solution:**

### **Test data:**

Test 1: 1, 1

Test 2: 1.5, 1.5

### **Sample output:**

Output 1: 1

Output 2: 1.5

### **Reflection:**

I learned that I could put the user input line into an if statement to check if the user is inputting the correct type, if the input fails then the loop will break.

# **Week 2 – Lab B**

Date: 11th Feb 2022

## **Q1. The good >> Streaming Operator**

### **Question:**

Take a string value from the user after you take the float value (the user can enter something like 23 4.586 Hello into the console window).

Then output the string value to the console after you output the float value.

### **Solution:**

### **Test data:**

Input: 23 4.586 Hello

### **Sample output:**

Output: i=23, f=4.586, s=Hello

### **Reflection:**

It is easy to get multiple inputs from a user in a single line in C++ using “cin” and the >> operator. The operator can be used to split the input into multiple variables.

## **Q2. The Bad >> Streaming Operator**

### **Question:**

Reflect on the nature and uses of the >> streaming operator and the get() function.

### **Solution:**

### **Test data:**

Input: 123456789

### **Sample output:**

Output: c=1234

### **Reflection:**

Arrays have an “end of array” character stored in memory to determine the end of the array. If too many things are inputted into an array then this character will be too far along in the memory and cause an error. This problem can be caused when using the >> operator to put user inputs into an array. To overcome this, “cin.get()” should be used instead. This will make sure that the correct number of inputs are put into the array instead of allowing to overflow.

## **Q3. Assembly Language**

### **Question:**

N/A

### **Solution:**

### **Test data :**

N/A

### **Sample output:**

N/A

### **Reflection:**

The disassembly window can be used to view the file in assembly. Furthermore, the registers and memory windows can be used to view how each of the memory locations and registers are being used as you step through the code.

# **Week 3 – Lab C**

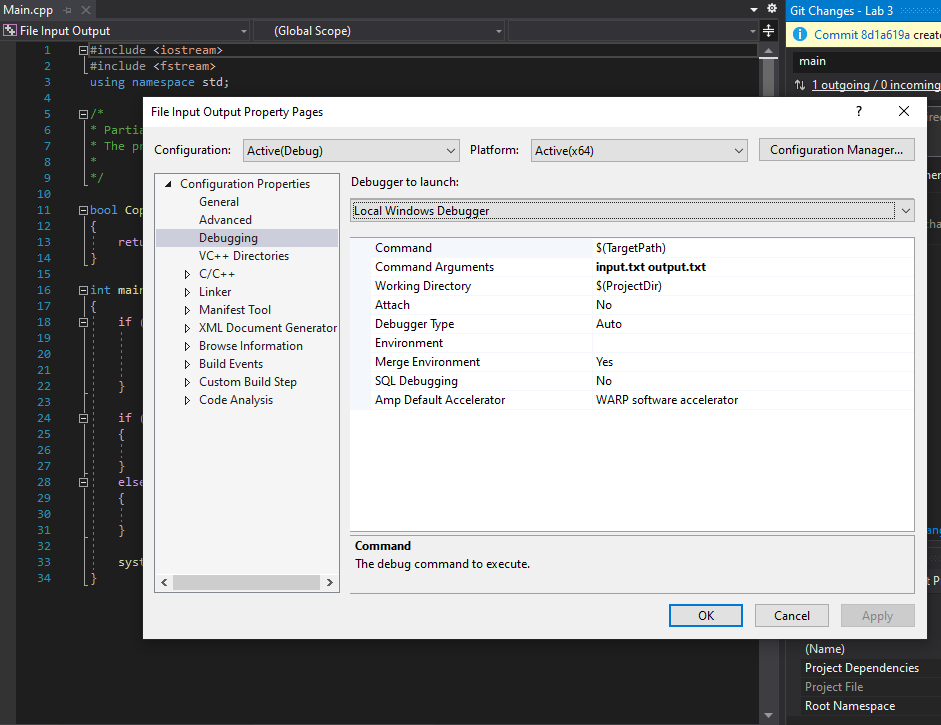
Date: 18th Feb 2022

## **Q1. Passing Command Arguments**

### **Question:**

Record the steps required to enter arguments into Visual Studio

### **Solution:**



### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

Arguments can be passed to Visual Studio by opening the Project tab, selecting the project properties window, and entering the arguments into the “Command Arguments” field. Argument 0 will be the file path of the program and cannot be changed. Any other arguments added to the field will be argument 1 and onwards.

## **Q2. Copying a Text File**

### **Question:**

Complete the functionality inside of the “Copy(char filename[], char filenameout[])” function.

### **Solution:**

### **Test data:**

Input 1: input.txt output.txt

Input 2: input.txt differentoutput.txt

Input 3: wronginput.txt output.txt

### **Sample output:**

Output 1: Copy successful

Output 2: Copy successful

Output 3: Copy error

### **Reflection:**

The ofstream class creates a file with the name you give it if the file does not already exist. This means that changing the name of the output file in the command arguments field didn’t cause an error. However, ifstream relies on the name being correct as it has to find a file with that name to open. If I change the input file argument to a file that doesn’t exist then I get a copy error because it failed to open a file with that name.

## **Q3. Function call mechanism**

### **Question:**

Investigate how the C++ parameter passing mechanism deals with these new parameters.

### **Solution:**

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

A uses 3 bytes

Letter uses 3 bytes

# **Week 4 – Lab D**

Date: 25th Feb 2022

## **Q1. Linker Errors**

### **Question:**

Describe what is required in the “.h” and “.cpp” files of a class so that you can define a constructor method.

### **Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

The “.h” file is a header file that holds declarations of variables and methods so that they can be imported into the “.cpp” class file. The header file must include any methods or variables that are used/written in the main class file.

## **Q2. Reading into Grid Class**

### **Question:**

Add your code for method “Grid::LoadGrid” and describe how you implemented it.

### Text Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

Using a nested for loop, I could step through each character in the file and assign each character to a space in the 2d integer array. At first, I discovered a problem where it was reading a character from the file, but the value that was being stored in the array as an ASCII value instead due to how it was being converted into an integer. To overcome this, I deducted the ASCII value of ‘0’ from the character read from the file and put the result of that in the array.

## **Q3. Saving the Grid**

### **Question:**

Add your code for method “Grid::SaveGrid” and describe how you implemented it.

### A screenshot of a computer Description automatically generated with medium confidenceText Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

This was similar to a previous question for copying a file. This time, I looped through the array that I previously loaded and wrote each character to the output file with a space after each character. If it has reached the end of the line, a space will not be included, and it will move to the next line. Once it has reached the end of the file, a new line will not be created.

## **Q4. Pointers – Basics**

### **Question:**

Open a memory window. Copy the value of “p” into the address field of the memory window and confirm that you are looking at variable “a” in memory.

### A screenshot of a computer Description automatically generated with medium confidence**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

A pointer can be used to point to a specific place in memory. In this case, a pointer is pointing to the memory location of the variable “a” which is currently set to 10. Pasting the value of “p” into the memory window takes you to the memory location of “a” which currently holds the hexadecimal value “0a” (10).

## **Q5. Pointers – False assumptions**

### **Question:**

N/A

### **Solution:**

N/A

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

Incrementing a pointer variable will move the pointer 4 bytes in memory. In this question, 3 variables were created back to back and incrementing the pointer was intended to move the pointer to the next variable. However, this didn’t work because it isn’t gauranteed that those variables have back to back locations in memory. Therefore, a better way of doing this would be to use an array. An array has a set amount of memory it uses when it is created which means that every value inside it will be placed sequentially in memory, allowing us to move through the memory locations with this method.

## **Q6. Pointers – The crash**

### **Question:**

N/A

### **Solution:**

Graphical user interface, text, application

Description automatically generated

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

An exception can be created when trying to access a memory location outside the bounds of the permitted allocation of the current application. This is to avoid any unwanted damage or corruption to data in other locations of the system.

## **Q7. Pointers – Pointer to pointers**

### **Question:**

Add code to change the value of x using only pointer p (p > q > x)

### Text Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

x= 3.14

x= 25.25

### **Reflection:**

Pointers can be setup to point to other pointers to change the value of what the original pointer is pointing to.

# **Week 5 – Lab E**

Date: 4th March 2022

## **Q1. Operators in Grid**

### **Question:**

Add the following functionality to your program:

The ability to write the Grid to an ostream using the auxiliary operator<<

The ability to read in the values from an istream into the Grid using the auxiliary operator>>

### **Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

I originally used the << operator for writing the grid to a file so no changes needed to be made here. However, for reading from the grid, I originally used “inputFile.get()”. By doing so, I also had to use an if statement to check if the character it read was a space or not. A better and more efficient way to do this would be to just use the >> operator, as this function will ignore whitespace, meaning I would no longer have to use the if statement.

## **Q2. Fractions**

### **Question:**

Implement the Fraction class that you have seen in lectures. Use the header file example that was presented in lectures to define your class, methods, member variables etc

### Text Description automatically generated**Solution:**

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

### Text Description automatically generated

### **Test data:**

Input: 3 10

### **Sample output:**

1/2 + 3/4 = 10/8

3/4 - 1/2 = 2/8

3/4 \* 3 = 9/4

3

10

Read = 3/10

### **Reflection:**

N/A

## **Q3. Operators in Fraction**

### **Question:**

Copy your code for your operators into your lab book. Reflect on the difference between class operators and auxiliary operators.

### Text Description automatically generatedText Description automatically generated**Solution:**

### **Test data:**

Input 1: 3 10

Input 2: 3 10

### **Sample output:**

1/2 + 3/4 = 10/8

3/4 - 1/2 = 2/8

3/4 \* 3 = 9/4

3

10

Read = 3/10

1/2 + 3/4 = 10/8

3/4 - 1/2 = 2/8

3/4 \* 3 = 9/4

3 \* 3/4 = 9/4

3

10

Read = 3/10

### **Reflection:**

Operator member methods are written inside the class and auxiliary methods are written outside the class. Overloading an operator as a member method of a class can be used to replace code like Fraction.multiply(number) and instead be written as Fraction \* number. On the left of the operator is an object and on the right of the operator is whatever value is being used for the calculation. For example, with Fraction(3/4) \* 3, the compiler will see that there is a fraction type (of which has an operator overload method inside) to the left of the operator and will call the associated operator method and pass it the argument of 3. However, this would not work for a streaming operator as “cout” or “cin” would always have to be on the left of the operator. To overcome this, an auxiliary operator can be used. These are defined outside of the class and the order that values surrounding it have to be in need to be specified in the method. For example, overloading the operator with two parameters “(int number, Fraction f)” would mean that the compiler will find any instances of the operator having an integer on the left and a fraction type on the right.

## **Q4. Parameters**

### **Question:**

Copy the code for pass-by-value and pass-by-ref into your lab book. Reflect on the difference between them.

### Text Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

Pass by value essentially creates a copy of a variable. If an integer with the value of 10 is passed into a method, then the method will create a new integer with the value of 10. Any changes made to the variable within the method will only apply to the newly created integer and not the original one. So, the changes can only be seen within the scope of the method. However, pass by reference allows the method to directly access and change the variable. This means that any changes made to the variable can be seen outside the scope of the method.

## **Q5. Return by value**

### **Question:**

Copy the code for return-by-value and return-by-ref into your lab book. Reflect on the difference between them.

### Text Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

Result1 = 30

Result2 = 20

### **Reflection:**

Return by value copies the value onto the stack and then copied into a variable, then keeps the original value within the scope of the method. Return by reference copies the memory address of the value. If you are using a large data type and returning it by value, then the whole thing will be copied into the stack which could potentially be very large, passing by referencing will save on all this wasted space by only copying the memory address instead.

# **Week 6 – Lab F**

Date: 11th March 2022

## **Q1. Template Grid**

### **Question:**

Add your Grid.h code

### **Solution:**

### 

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

A template class can be used to change what type of objects are stored in the class. In this example, the class originally used an integer array, however, using a template class allows you to specify a different type of number like a float or a double when creating an instance of the class.

## **Q2. Template Grid (floats)**

### **Question:**

Add your code from your main function. Add the output of OutGrid.txt. Add a screenshot of the debugger showing the contents of the grid instance in main after it has successfully loaded the values from the file.

### Text Description automatically generatedA picture containing text Description automatically generatedGraphical user interface Description automatically generated with medium confidence**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

There is a much simpler way of doing what I was trying to do. Replacing all the code in load grid from the last task with a simple “inputFile >> m\_grid[x][y]” would allow me to read from the file better. Previously, it would go character by character, and this is a problem when reading a floating point number as it would need to read at least 3 characters for one float, but was instead reading 1 character for one float. Instead I could just stream directly into the grid array (the type of which could change depending on how the class is created in main) and would appropriately convert the input into whatever type it needed.

## **Q3. Binary Search**

### **Question:**

Implement two versions of binary search, one using iteration and one using a recursive function.

### **Solution:**

### **Test data:**

Recursive:

Input: 3 (found)

Input: 2 (not found)

Input: 99 (found)

Input: 101 (not found)

Iterative:

Input: 3 (found)

Input: 2 (not found)

Input: 99 (found)

Input: 101 (not found)

### **Sample output:**

N/A

### **Reflection:**

Using a while loop can iterate through a memory location if the value is not found. Using a recursive function can do a similar thing by calling itself and modifying the parameters if a value is not found. In my opinion the recursive function is easier to read and understand.

**Week 7 – Lab G**

Date: 18th March 2022

## **Q1. Parasoft**

### **Question:**

Add screenshots of the Parasoft violation output. Then describe the corrections you made to remove the Parasoft Severity 3 violations.

### **Solution:**

### 

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

Parasoft can be used to perform tests on a selection of code that can reveal problems such as bugs or simple naming convention mistakes. Parasoft will display the results as a series of violations with specific severities. Double clicking on the violations will jump to that line of code and a detailed documentation of each error can be viewed by right clicking on the violation and selecting “View Rule Documentation” which can show how the issues are created and solved. In this case, I had to declare some parameters and variables as constant as they are never changed after being declared and added a copy constructor and assignment operator to the utility class.

# **Week 8 – Lab H**

Date: 25th March 2022

## **Q1. PersonNode**

### **Question:**

Add your code and describe what you have done.

### Text Description automatically generatedText Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

I added public get and set methods for the name and age member variables in the person node class and also added the AddressBookSLL class as a friend so that the person node class can directly access private values and methods inside of the address book class.

## **Q2. AddressBookSLL**

### **Question:**

Add your code and describe what you have done

### Text Description automatically generatedText Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

N/A

### **Reflection:**

The add person method in the address book class has an if statement inside it that determines the address of the new person to be added. If the head pointer is null (there are no people in the address book) then the person to be added will take the place of the head. If the head pointer is not null and the next pointer within that person node is null then the new person will take the place of the next pointer associated with the head pointer. Finally, if none of these conditions are met, a while loop will begin. This will loop through each node in the address book until it finds the end (a node with a next pointer that is equal to null). Once this is done, a new person node will be created at the next pointer associated with the final node that was previously discovered.

## **Q3. Find, Delete, Output**

### **Question:**

Add your code and describe what you have done

### Text Description automatically generatedText Description automatically generated**Solution:**

### **Test data:**

N/A

### **Sample output:**

Name: Henry // Age: 20

Name: Louis // Age: 19

Name: Billy // Age: 11

Name: Natalia // Age: 5

### **Reflection:**

The find person method works by looping through each node in the list and checking the name associated with it against the name provided. If it is found, it returns the pointer to that node. If the list doesn’t contain a node with that name, a null pointer will be returned instead. The delete person method works by using the previous method to find a person with a specific name and then delete that pointer in memory. If the name is not found, the value false will be returned. Finally, the write book method works by, again, looping through all the nodes and writing each of the values associated with a node to the console until no more nodes are left.