LAB BOOK

HENRY PAUL - 632666

Contents

632666 Lab Book

Week 1 – Lab A

Date: 4th Feb 2022

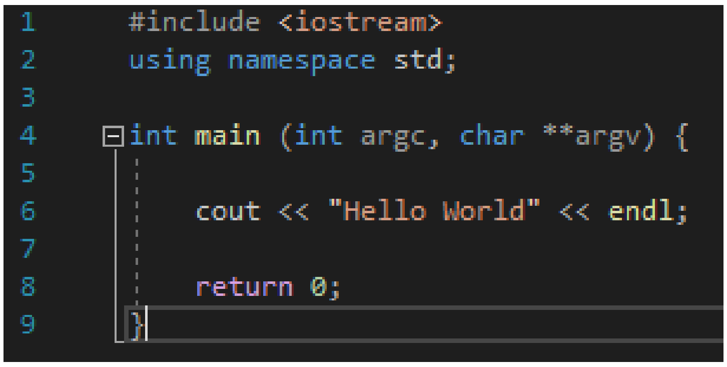
Q1. Hello World

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:



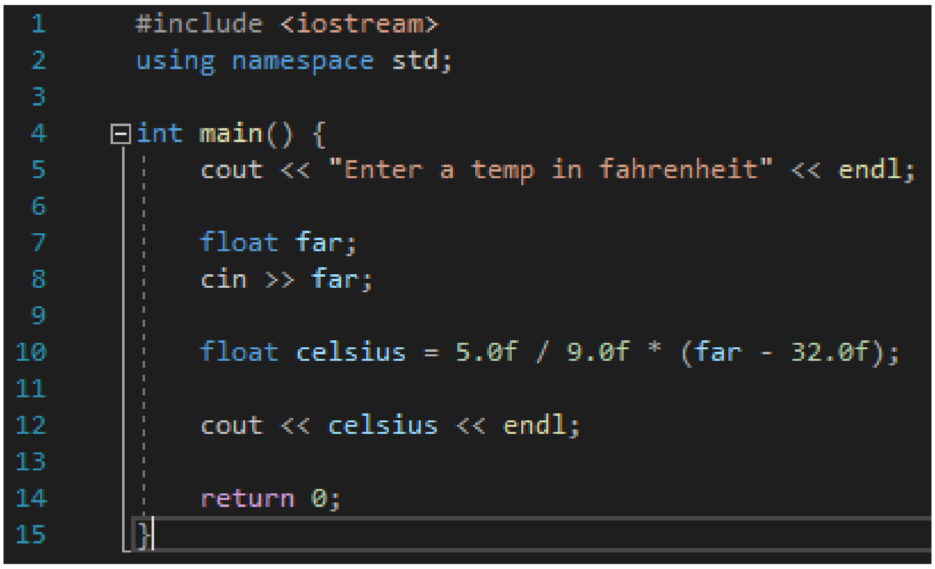
```
#include <iostream>
using namespace std;

int main (int argc, char **argv) {

    cout << "Hello World" << endl;

    return 0;
}
```

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:



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

int main() {
    cout << "Enter a temp in fahrenheit" << endl;

    float far;
    cin >> far;

    float celsius = 5.0f / 9.0f * (far - 32.0f);

    cout << celsius << endl;

    return 0;
}
```

Test data:

Input 1: 50

Input 2: 100

Sample output:

Output 1: 10

Output 2: 37.7778

Reflection:

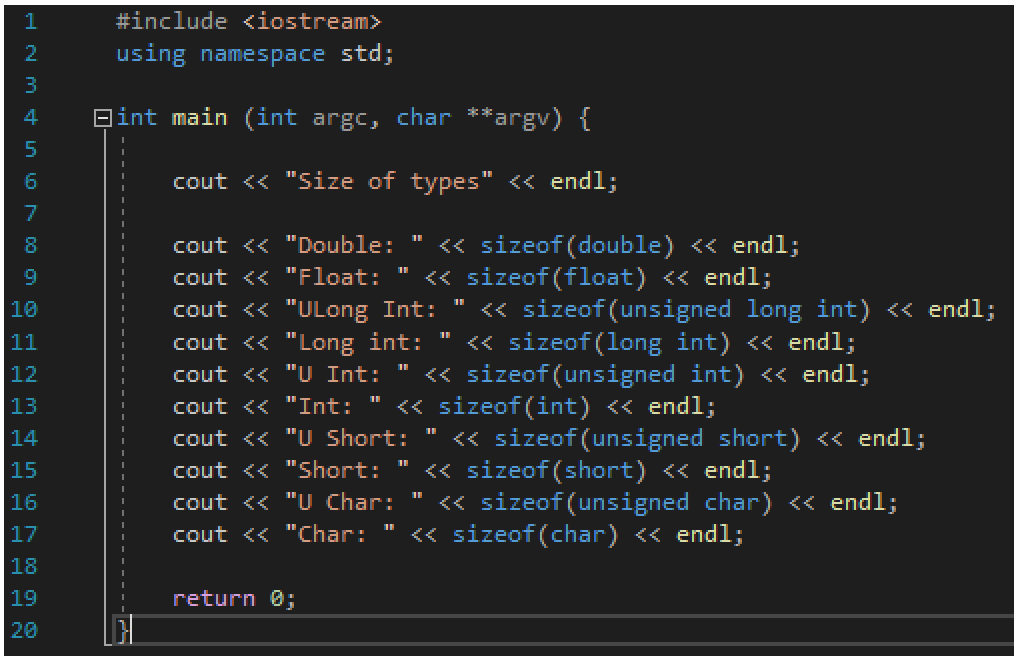
Initially, the code was written using integer types and this caused the output to always be 0. This is because 5 divided by 9 is 0.5 which can't be stored fully in an integer so it would be rounded down to 0. To overcome this, I used floats to make sure that all values retain their precision.

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:



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

int main (int argc, char **argv) {

    cout << "Size of types" << endl;

    cout << "Double: " << sizeof(double) << endl;
    cout << "Float: " << sizeof(float) << endl;
    cout << "ULong Int: " << sizeof(unsigned long int) << endl;
    cout << "Long int: " << sizeof(long int) << endl;
    cout << "U Int: " << sizeof(unsigned int) << endl;
    cout << "Int: " << sizeof(int) << endl;
    cout << "U Short: " << sizeof(unsigned short) << endl;
    cout << "Short: " << sizeof(short) << endl;
    cout << "U Char: " << sizeof(unsigned char) << endl;
    cout << "Char: " << sizeof(char) << endl;

    return 0;
}
```

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:

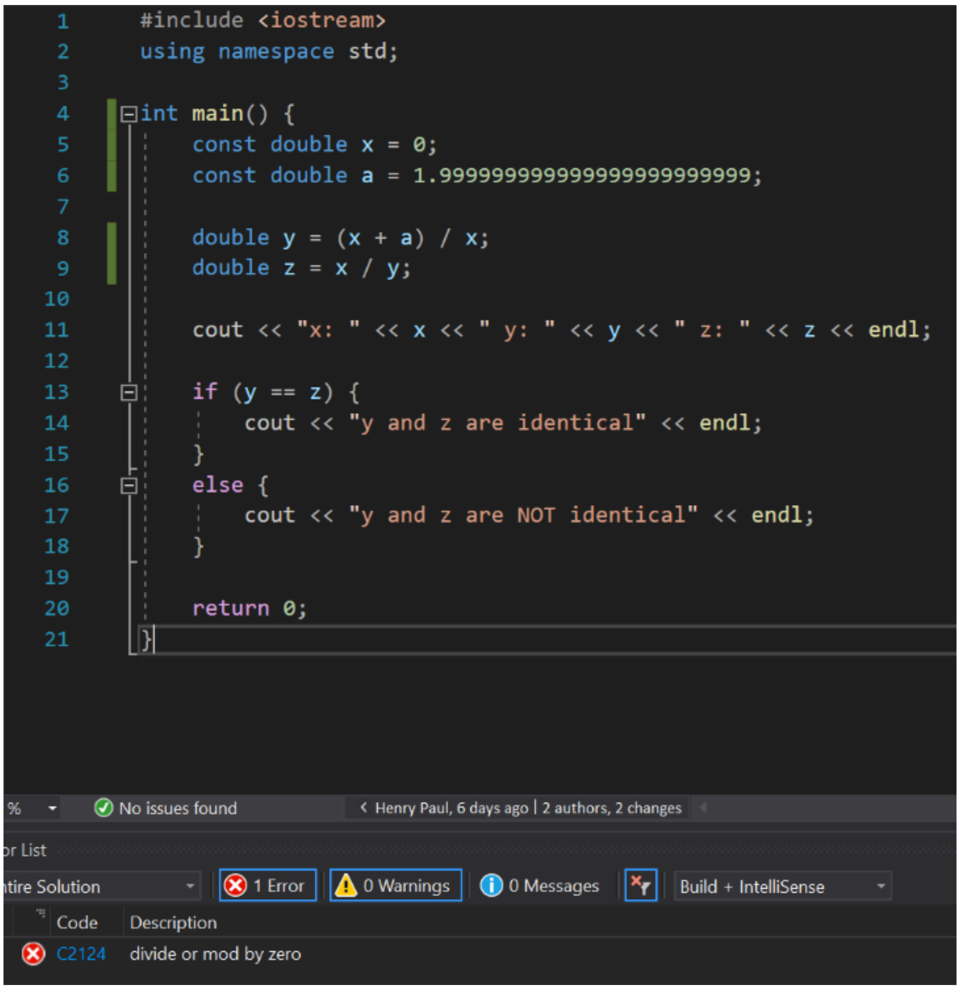
Using “sizeof()” will return the size of any data type in bytes so this is a good way to record how much memory you are using and ways you can reduce the amount of memory you’re using. For example, a float uses half as much memory as a double so if you need to store a decimal number with a relatively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



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

int main() {
    const double x = 0;
    const double a = 1.99999999999999999999999;

    double y = (x + a) / x;
    double z = x / y;

    cout << "x: " << x << " y: " << y << " z: " << z << endl;

    if (y == z) {
        cout << "y and z are identical" << endl;
    }
    else {
        cout << "y and z are NOT identical" << endl;
    }

    return 0;
}
```

```
%    No issues found    Henry Paul, 6 days ago | 2 authors, 2 changes
or List
ntire Solution    1 Error    0 Warnings    0 Messages    Build + IntelliSense
    Code     Description
    C2124    divide or mod by zero
```

Test data:

N/A

Sample output:

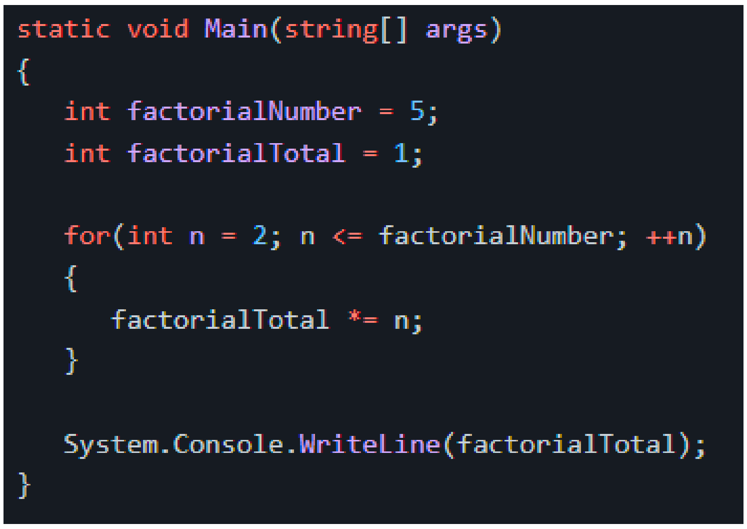
“divide or mod by zero” error

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)

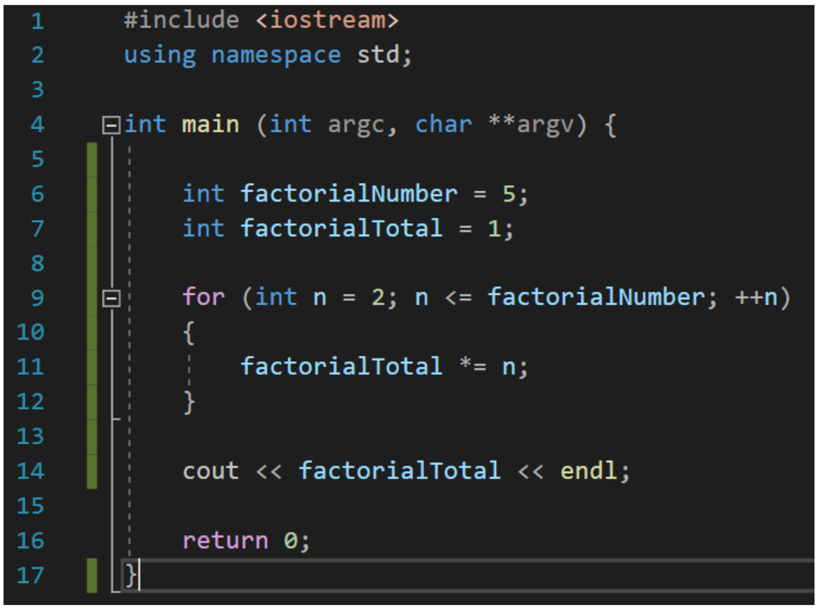
Question:



```csharp
static void Main(string[] args)
{
    int factorialNumber = 5;
    int factorialTotal = 1;

    for(int n = 2; n <= factorialNumber; ++n)
    {
        factorialTotal *= n;
    }

    System.Console.WriteLine(factorialTotal);
}
```

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

Solution:



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

int main (int argc, char **argv) {

    int factorialNumber = 5;
    int factorialTotal = 1;

    for (int n = 2; n <= factorialNumber; ++n)
    {
        factorialTotal *= n;
    }

    cout << factorialTotal << endl;

    return 0;
}
```

Test data:

factorialNumber 1: 4

factorialNumber 2: 5

Sample output:

Output 1: 24

Output 2: 120

Reflection:

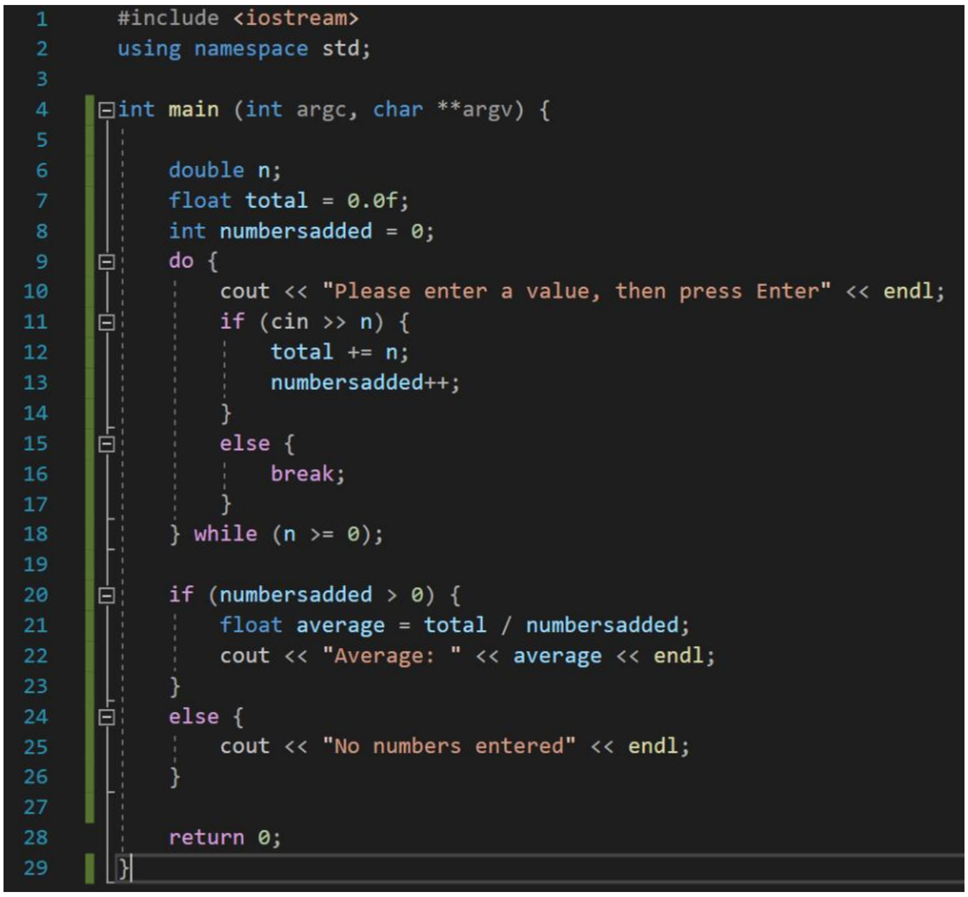
The syntax for a for loop in C++ is identical to C# so no changes needed to be made here. The only change I needed to make was the "Console.WriteLine()" to "cout".

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:



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

int main (int argc, char **argv) {

    double n;
    float total = 0.0f;
    int numbersadded = 0;
    do {
        cout << "Please enter a value, then press Enter" << endl;
        if (cin >> n) {
            total += n;
            numbersadded++;
        }
        else {
            break;
        }
    } while (n >= 0);

    if (numbersadded > 0) {
        float average = total / numbersadded;
        cout << "Average: " << average << endl;
    }
    else {
        cout << "No numbers entered" << endl;
    }

    return 0;
}
```

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

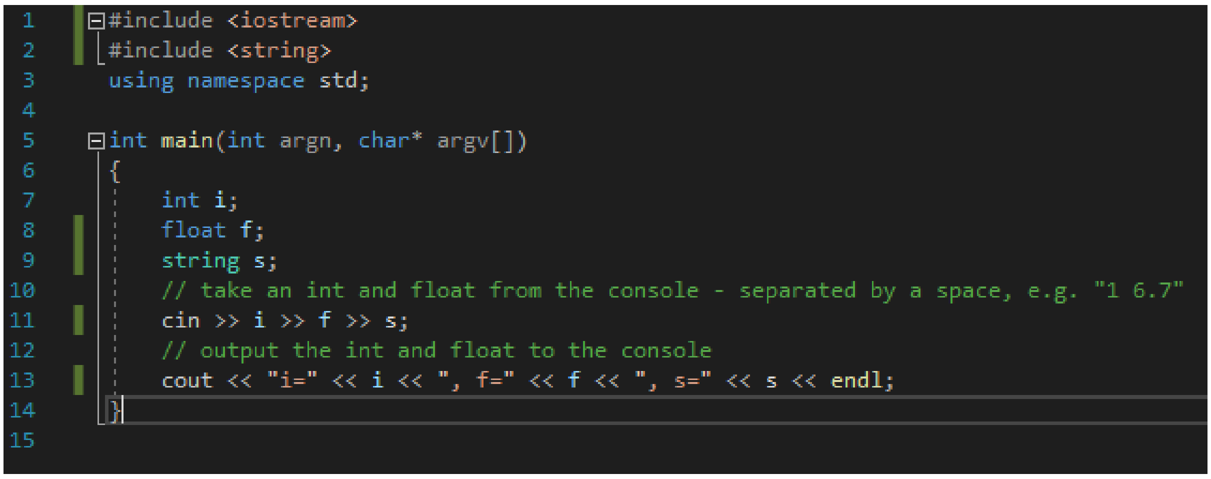
Q1. The good >> Streaming Operator

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:



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

int main(int argn, char* argv[])
{
    int i;
    float f;
    string s;
    // take an int and float from the console - separated by a space, e.g. "1 6.7"
    cin >> i >> f >> s;
    // output the int and float to the console
    cout << "i=" << i << ", f=" << f << ", s=" << s << endl;
}
```

Test data:

Input: 23 4.586 Hello

Sample output:

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

Reflection:

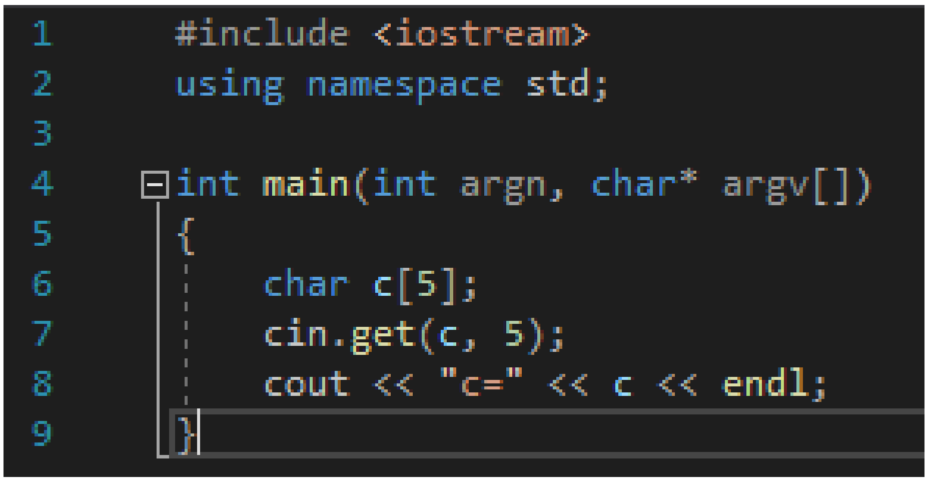
It is easy to get multiple inputs from a user in a single line in C++ using “cin” and the >> operator. The operator can be used to split the input into multiple variables.

Q2. The Bad >> Streaming Operator

Question:

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

Solution:



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

int main(int argn, char* argv[])
{
    char c[5];
    cin.get(c, 5);
    cout << "c=" << c << endl;
}
```

Test data:

Input: 123456789

Sample output:

Output: c=1234

Reflection:

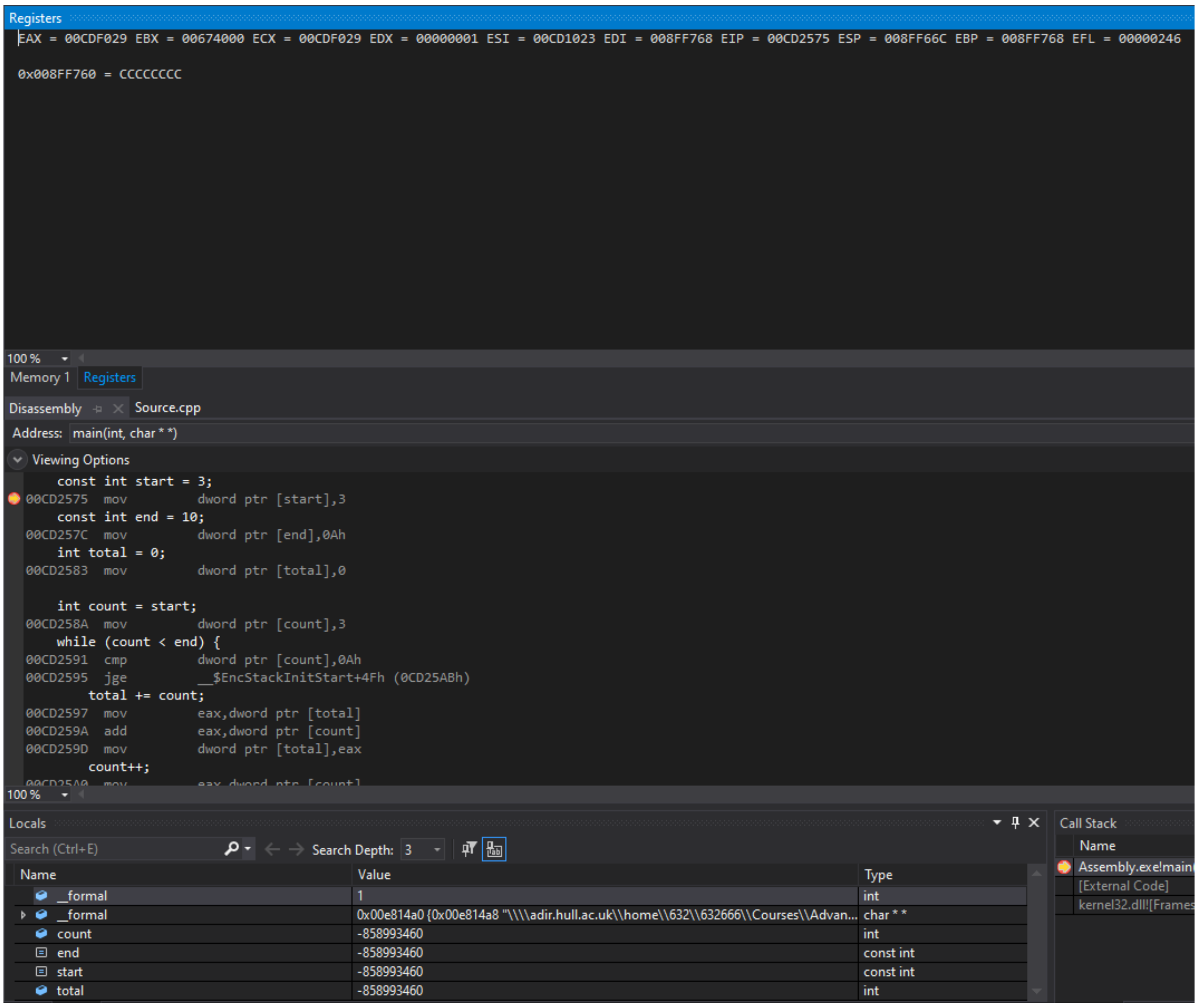
Arrays have an “end of array” character stored in memory to determine the end of the array. If too many things are inputted into an array then this character will be too far along in the memory and cause an error. This problem can be caused when using the >> operator to put user inputs into an array. 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:



Registers

```
EAX = 00CDF029 EBX = 00674000 ECX = 00CDF029 EDX = 00000001 ESI = 00CD1023 EDI = 008FF768 EIP = 00CD2575 ESP = 008FF66C EBP = 008FF768 EFL = 00000246

0x008FF760 = CCCCCCCC
```

Address: main(int, char * *)

```
    const int start = 3;
00CD2575  mov         dword ptr [start],3
    const int end = 10;
00CD257C  mov         dword ptr [end],0Ah
    int total = 0;
00CD2583  mov         dword ptr [total],0

    int count = start;
00CD258A  mov         dword ptr [count],3
    while (count < end) {
00CD2591  cmp         dword ptr [count],0Ah
00CD2595  jge         __$EncStackInitStart+4Fh (0CD25ABh)
        total += count;
00CD2597  mov         eax,dword ptr [total]
00CD259A  add         eax,dword ptr [count]
00CD259D  mov         dword ptr [total],eax
        count++;
```

Locals

Name	Value	Type
_formal	1	int
_formal	0x00e814a0 {0x00e814a8 "\\\\adir.hull.ac.uk\\home\\632\\632666\\Courses\\Advan...	char * *
count	-858993460	int
end	-858993460	const int
start	-858993460	const int
total	-858993460	int

Call Stack

```
Assembly.exe!main
[External Code]
kernel32.dll![Frames
```

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

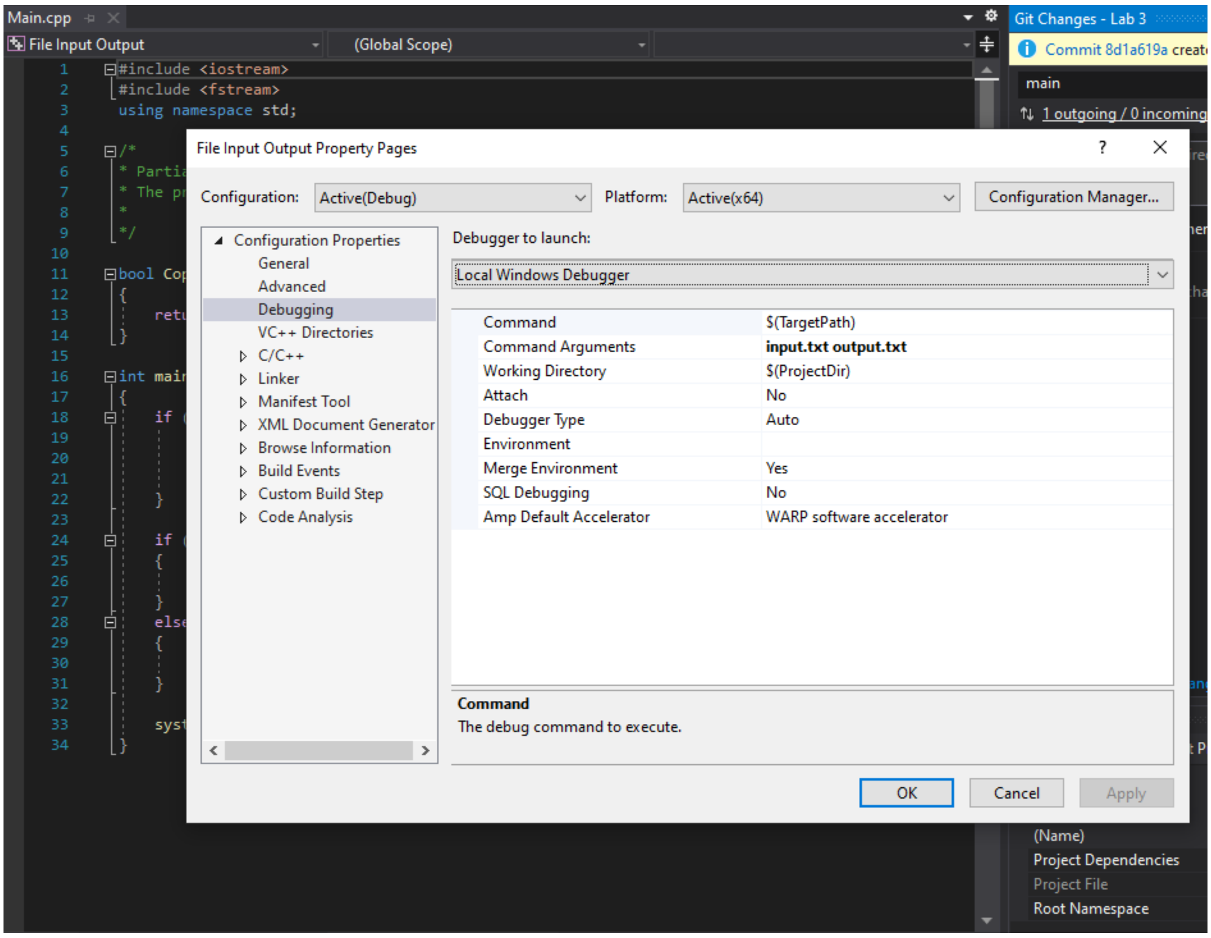
Date: 18th Feb 2022

Q1. Passing Command Arguments

Question:

Record the steps required to enter arguments into Visual Studio

Solution:



File Input Output Property Pages

Configuration: Active(Debug) Platform: Active(x64)

Debugger to launch: Local Windows Debugger

Command	$(TargetPath)
Command Arguments	**input.txt output.txt**
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Command
The debug command to execute.

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

Test data:

N/A

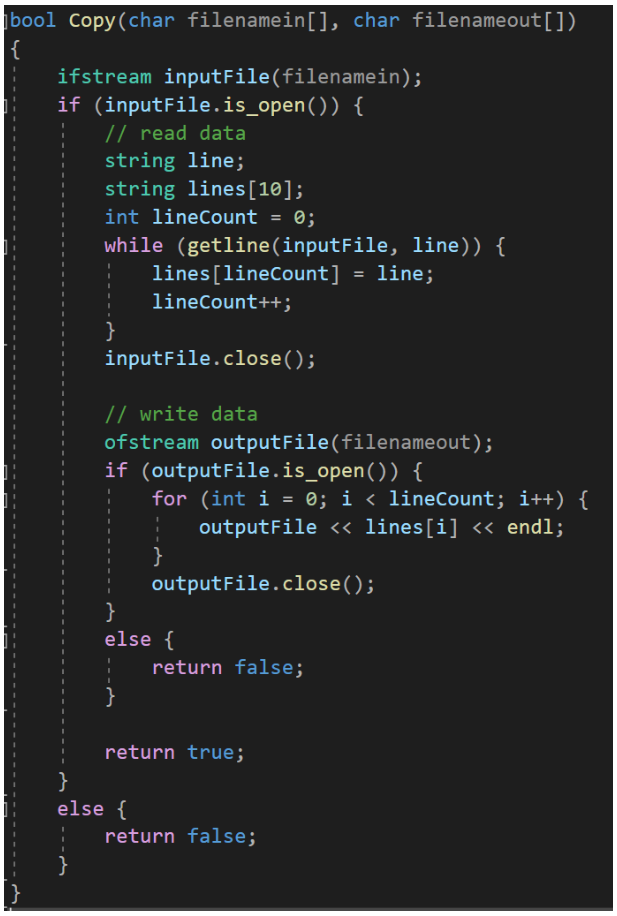
Sample output:

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:


```
bool Copy(char filenamein[], char filenameout[])
{
    ifstream inputFile(filenamein);
    if (inputFile.is_open()) {
        // read data
        string line;
        string lines[10];
        int lineCount = 0;
        while (getline(inputFile, line)) {
            lines[lineCount] = line;
            lineCount++;
        }
        inputFile.close();

        // write data
        ofstream outputFile(filenameout);
        if (outputFile.is_open()) {
            for (int i = 0; i < lineCount; i++) {
                outputFile << lines[i] << endl;
            }
            outputFile.close();
        }
        else {
            return false;
        }

        return true;
    }
    else {
        return false;
    }
}
```

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:

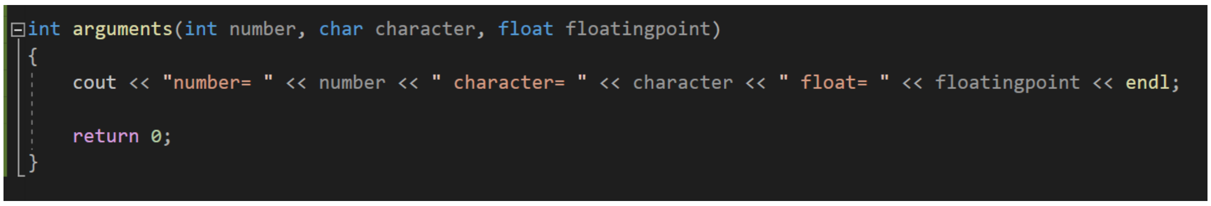
The ofstream class creates a file with the name you give it if the file does not already exist. This means that changing the name of the output file in the command arguments field didn’t cause an error. However, ifstream relies on the name being correct as it has to find a file with that name to open.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

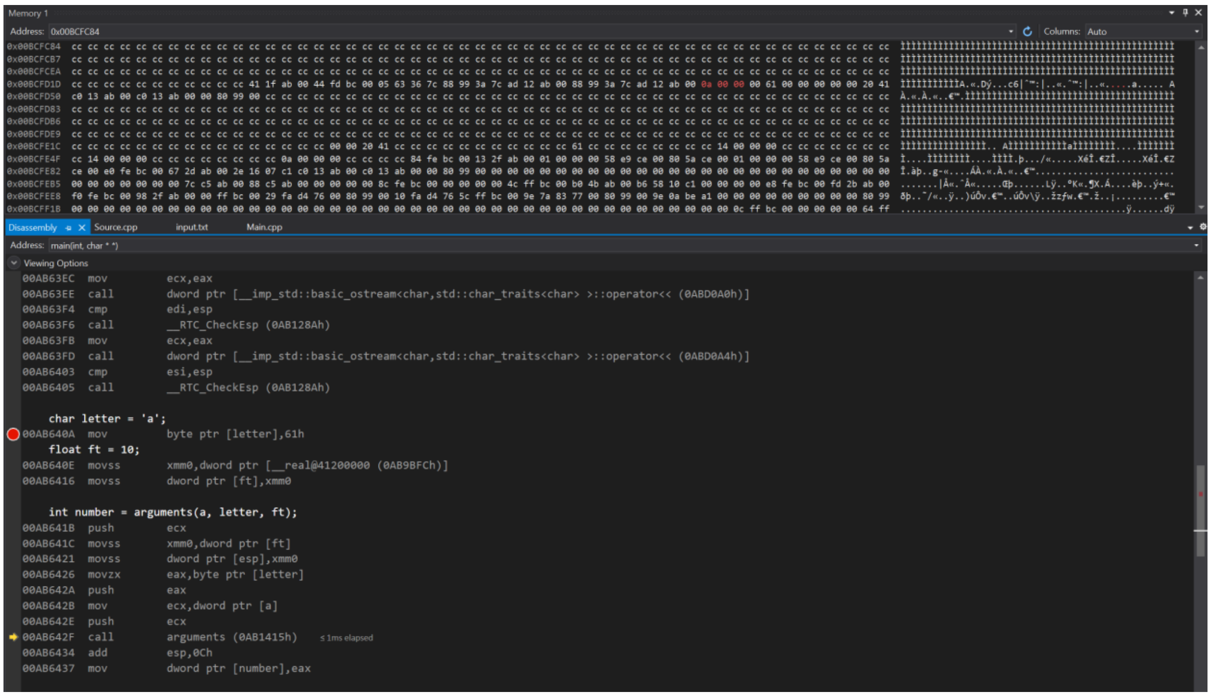
Question:

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

Solution:



```cpp
int arguments(int number, char character, float floatingpoint)
{
    cout << "number= " << number << " character= " << character << " float= " << floatingpoint << endl;

    return 0;
}
```



Memory 1

Address: 0x00BCFC84 Columns: Auto

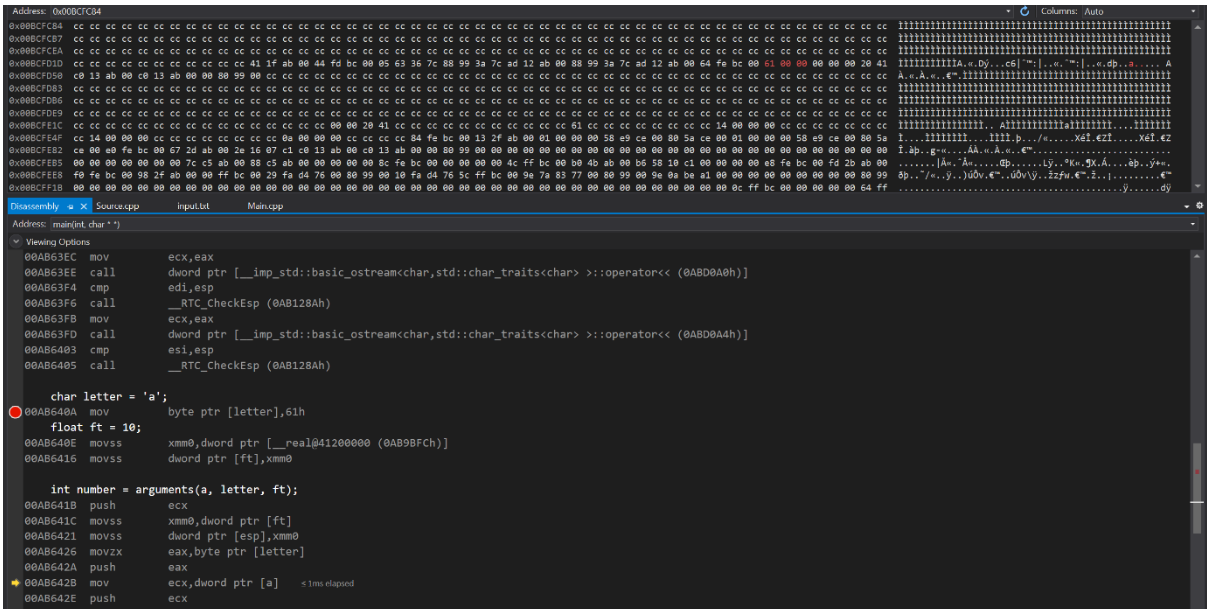
```
0x00BCFC84  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFCB7  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFCEA  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFD1D  cc cc cc cc cc cc cc cc cc cc cc 41 1f ab 00 44 fd bc 00 05 63 36 7c 88 99 3a 7c ad 12 ab 00 88 99 3a 7c ad 12 ab 00 0a 00 00 00 61 00 00 00 00 00 20 41  ÌÌÌÌÌÌÌÌÌÌÌA.«.Dý...c6|ˆ™:|..«.ˆ™:|..«....a..... A
0x00BCFD50  c0 13 ab 00 c0 13 ab 00 00 80 99 00 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  À.«.À.«..€™.ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFD83  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFDB6  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFDE9  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFE1C  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc 00 00 20 41 cc cc cc cc cc cc cc cc cc cc cc cc 61 cc cc cc cc cc cc cc cc 14 00 00 00 cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ.. AÌÌÌÌÌÌÌÌÌÌÌÌaÌÌÌÌÌÌÌÌ....ÌÌÌÌÌÌÌ
0x00BCFE4F  cc 14 00 00 00 cc cc cc cc cc cc cc cc 0a 00 00 00 cc cc cc cc 84 fe bc 00 13 2f ab 00 01 00 00 00 58 e9 ce 00 80 5a ce 00 01 00 00 00 58 e9 ce 00 80 5a  Ì....ÌÌÌÌÌÌÌÌ....ÌÌÌÌ.þ.../«.....XéÎ.€ZÎ.....XéÎ.€Z
0x00BCFE82  ce 00 e0 fe bc 00 67 2d ab 00 2e 16 07 c1 c0 13 ab 00 c0 13 ab 00 00 80 99 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  Î.àþ..g-«....ÁÀ.«.À.«..€™..........................
0x00BCFEB5  00 00 00 00 00 00 00 7c c5 ab 00 88 c5 ab 00 00 00 00 00 8c fe bc 00 00 00 00 00 4c ff bc 00 b0 4b ab 00 b6 58 10 c1 00 00 00 00 e8 fe bc 00 fd 2b ab 00  .......|Å«.ˆÅ«.....Œþ......Lÿ..°K«.¶X.Á....èþ..ý+«.
0x00BCFEE8  f0 fe bc 00 98 2f ab 00 00 ff bc 00 29 fa d4 76 00 80 99 00 10 fa d4 76 5c ff bc 00 9e 7a 83 77 00 80 99 00 9e 0a be a1 00 00 00 00 00 00 00 00 00 80 99  ðþ..˜/«..ÿ..)úÔv.€™..úÔv\ÿ..žzƒw.€™.ž..¡.........€™
0x00BCFF1B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0c ff bc 00 00 00 00 00 64 ff  ..........................................ÿ......dÿ
```

Disassembly | Source.cpp | input.txt | Main.cpp

Address: main(int, char * *)

Viewing Options

```
00AB63EC  mov         ecx,eax
00AB63EE  call        dword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::operator<< (0ABD0A0h)]
00AB63F4  cmp         edi,esp
00AB63F6  call        __RTC_CheckEsp (0AB128Ah)
00AB63FB  mov         ecx,eax
00AB63FD  call        dword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::operator<< (0ABD0A4h)]
00AB6403  cmp         esi,esp
00AB6405  call        __RTC_CheckEsp (0AB128Ah)

    char letter = 'a';
00AB640A  mov         byte ptr [letter],61h
    float ft = 10;
00AB640E  movss       xmm0,dword ptr [__real@41200000 (0AB9BFCh)]
00AB6416  movss       dword ptr [ft],xmm0

    int number = arguments(a, letter, ft);
00AB641B  push        ecx
00AB641C  movss       xmm0,dword ptr [ft]
00AB6421  movss       dword ptr [esp],xmm0
00AB6426  movzx       eax,byte ptr [letter]
00AB642A  push        eax
00AB642B  mov         ecx,dword ptr [a]
00AB642E  push        ecx
00AB642F  call        arguments (0AB1415h)  ≤ 1ms elapsed
00AB6434  add         esp,0Ch
00AB6437  mov         dword ptr [number],eax
```



```
Address: 0x00BCFC84                                                                                   Columns: Auto
0x00BCFC84  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFCB7  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFCEA  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFD1D  cc cc cc cc cc cc cc cc cc cc cc 41 1f ab 00 44 fd bc 00 05 63 36 7c 88 99 3a 7c ad 12 ab 00 88 99 3a 7c ad 12 ab 00 64 fe bc 00 61 00 00 00 00 00 20 41  ÌÌÌÌÌÌÌÌÌÌÌA.«.Dý...c6|ˆ™:|..«.ˆ™:|..«.dþ..a..... A
0x00BCFD50  c0 13 ab 00 c0 13 ab 00 00 80 99 00 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  À.«.À.«..€™.ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFD83  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFDB6  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFDE9  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ
0x00BCFE1C  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc 00 00 20 41 cc cc cc cc cc cc cc cc cc cc cc 61 cc cc cc cc cc cc cc cc 14 00 00 00 cc cc cc cc cc cc cc  ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ.. AÌÌÌÌÌÌÌÌÌÌÌaÌÌÌÌÌÌÌÌ....ÌÌÌÌÌÌÌ
0x00BCFE4F  cc 14 00 00 00 cc cc cc cc cc cc cc cc 0a 00 00 00 cc cc cc cc 84 fe bc 00 13 2f ab 00 01 00 00 00 58 e9 ce 00 80 5a ce 00 01 00 00 00 58 e9 ce 00 80 5a  Ì....ÌÌÌÌÌÌÌÌ....ÌÌÌÌ.þ.../«.....XéÎ.€ZÎ.....XéÎ.€Z
0x00BCFE82  ce 00 e0 fe bc 00 67 2d ab 00 2e 16 07 c1 c0 13 ab 00 c0 13 ab 00 00 80 99 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  Î.àþ..g-«....ÁÀ.«.À.«..€™.........................
0x00BCFEB5  00 00 00 00 00 00 00 7c c5 ab 00 88 c5 ab 00 00 00 00 00 8c fe bc 00 00 00 00 00 4c ff bc 00 b0 4b ab 00 b6 58 10 c1 00 00 00 00 e8 fe bc 00 fd 2b ab 00  .......|Å«.ˆÅ«.....Œþ......Lÿ..°K«.¶X.Á....èþ..ý+«.
0x00BCFEE8  f0 fe bc 00 98 2f ab 00 00 ff bc 00 29 fa d4 76 00 80 99 00 10 fa d4 76 5c ff bc 00 9e 7a 83 77 00 80 99 00 9e 0a be a1 00 00 00 00 00 00 00 00 00 80 99  ðþ..˜/«..ÿ..)úÔv.€™..úÔv\ÿ..žzƒw.€™.ž..¡.........€™
0x00BCFF1B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0c ff bc 00 00 00 00 00 64 ff  ..........................................ÿ......dÿ
```

```
Address: main(int, char * *)

00AB63EC  mov         ecx,eax
00AB63EE  call        dword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::operator<< (0ABD0A0h)]
00AB63F4  cmp         edi,esp
00AB63F6  call        __RTC_CheckEsp (0AB128Ah)
00AB63FB  mov         ecx,eax
00AB63FD  call        dword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::operator<< (0ABD0A4h)]
00AB6403  cmp         esi,esp
00AB6405  call        __RTC_CheckEsp (0AB128Ah)

    char letter = 'a';
00AB640A  mov         byte ptr [letter],61h
    float ft = 10;
00AB640E  movss       xmm0,dword ptr [__real@41200000 (0AB9BFCh)]
00AB6416  movss       dword ptr [ft],xmm0

    int number = arguments(a, letter, ft);
00AB641B  push        ecx
00AB641C  movss       xmm0,dword ptr [ft]
00AB6421  movss       dword ptr [esp],xmm0
00AB6426  movzx       eax,byte ptr [letter]
00AB642A  push        eax
00AB642B  mov         ecx,dword ptr [a]    ≤ 1ms elapsed
00AB642E  push        ecx
```

Test data:

N/A

Sample output:

N/A

Reflection:

A uses 3 bytes

Letter uses 3 bytes

Week 4 – Lab D

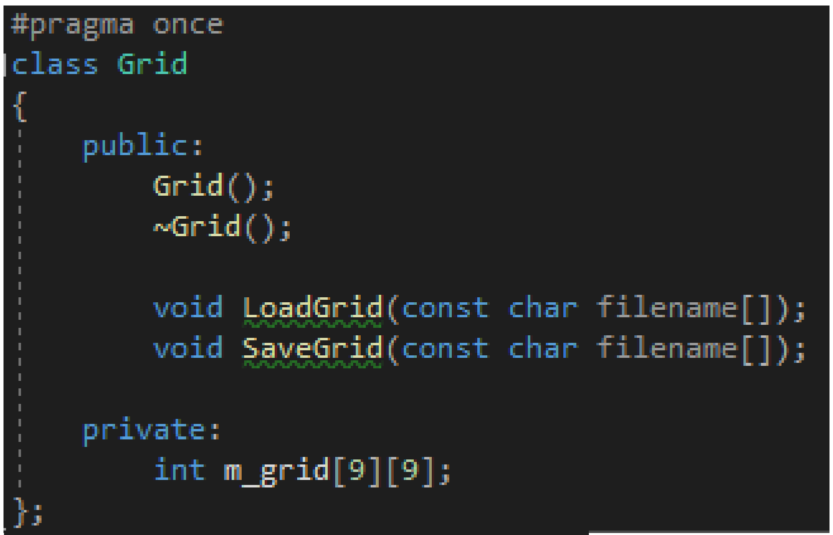
Date: 25th Feb 2022

Q1. Linker Errors

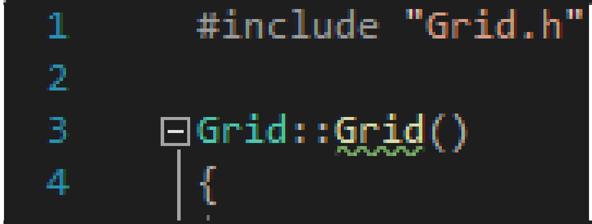
Question:

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

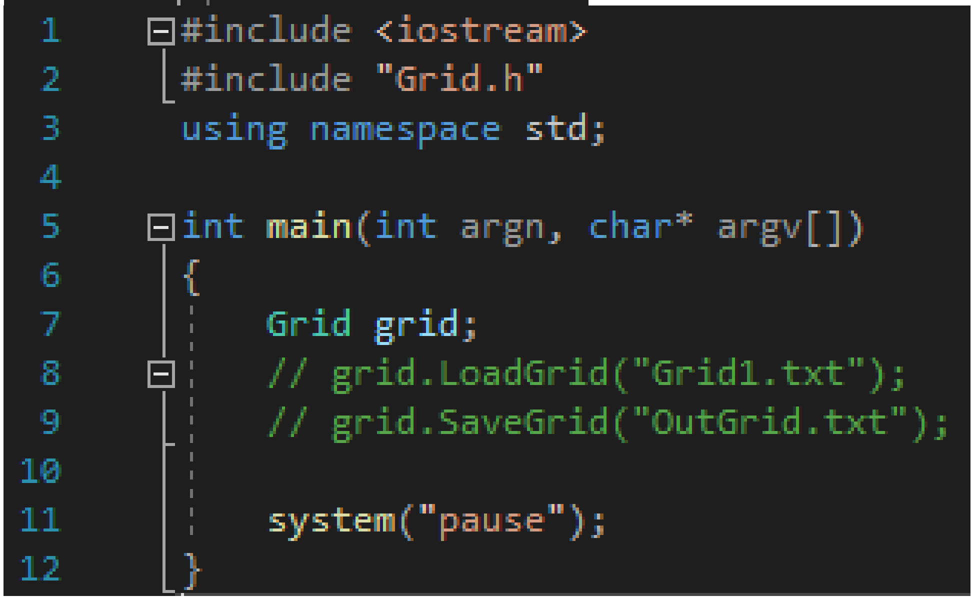
Solution:



```
#pragma once
class Grid
{
    public:
        Grid();
        ~Grid();

        void LoadGrid(const char filename[]);
        void SaveGrid(const char filename[]);

    private:
        int m_grid[9][9];
};
```



```
#include "Grid.h"

Grid::Grid()
{
```



```
#include <iostream>
#include "Grid.h"
using namespace std;

int main(int argn, char* argv[])
{
    Grid grid;
    // grid.LoadGrid("Grid1.txt");
    // grid.SaveGrid("OutGrid.txt");

    system("pause");
}
```

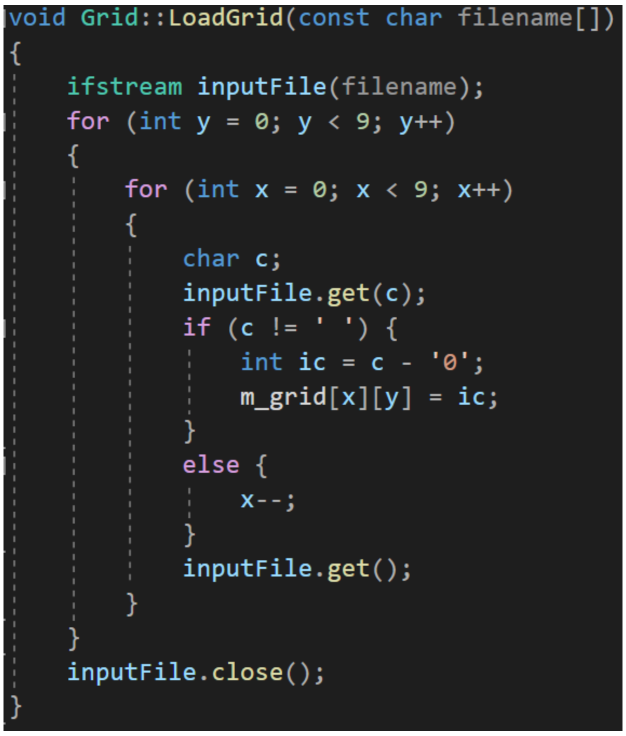
Test data:
N/A

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.

Solution:

```cpp
void Grid::LoadGrid(const char filename[])
{
    ifstream inputFile(filename);
    for (int y = 0; y < 9; y++)
    {
        for (int x = 0; x < 9; x++)
        {
            char c;
            inputFile.get(c);
            if (c != ' ') {
                int ic = c - '0';
                m_grid[x][y] = ic;
            }
            else {
                x--;
            }
            inputFile.get();
        }
    }
    inputFile.close();
}
```

Test data:
N/A

Sample output:
N/A

Reflection:

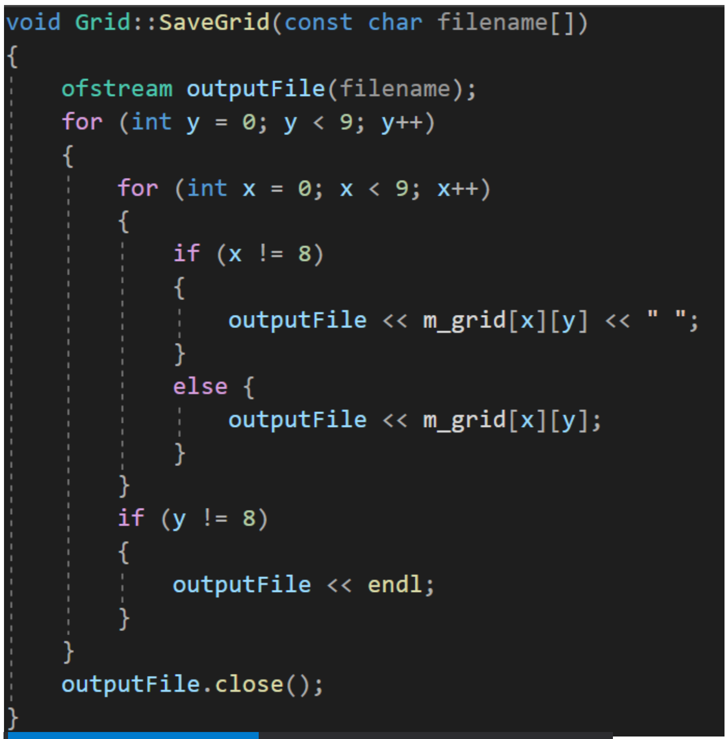
Using a nested for loop, I could step through each character in the file and assign each character to a space in the 2d integer array. At first, I discovered a problem where it was reading a character from the file, but the value that was being stored in the array as an ASCII value instead due to how 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

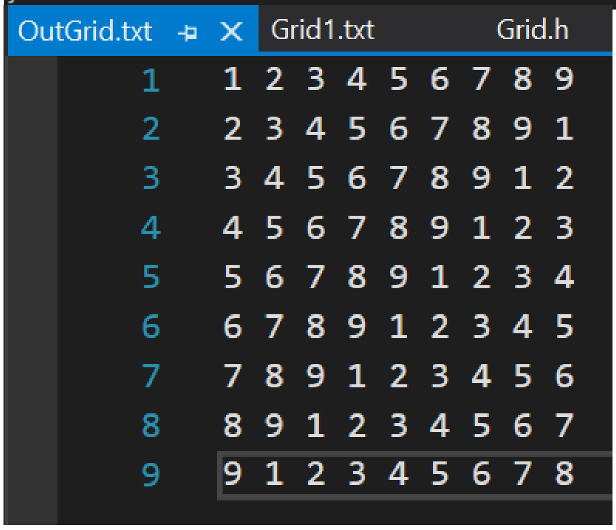
Question:

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

Solution:



```
void Grid::SaveGrid(const char filename[])
{
    ofstream outputFile(filename);
    for (int y = 0; y < 9; y++)
    {
        for (int x = 0; x < 9; x++)
        {
            if (x != 8)
            {
                outputFile << m_grid[x][y] << " ";
            }
            else {
                outputFile << m_grid[x][y];
            }
        }
        if (y != 8)
        {
            outputFile << endl;
        }
    }
    outputFile.close();
}
```



OutGrid.txt | Grid1.txt | Grid.h

```
1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 1
3 4 5 6 7 8 9 1 2
4 5 6 7 8 9 1 2 3
5 6 7 8 9 1 2 3 4
6 7 8 9 1 2 3 4 5
7 8 9 1 2 3 4 5 6
8 9 1 2 3 4 5 6 7
9 1 2 3 4 5 6 7 8
```

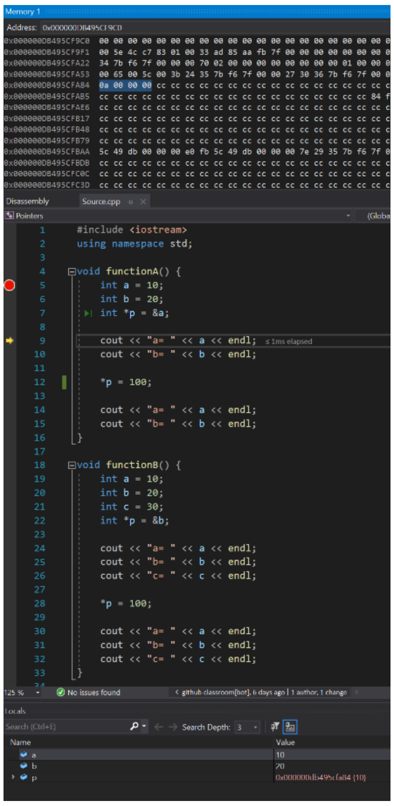
Test data:
N/A

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.

Solution:


Address: 0x000000DB495CF9CD

```
0x000000DB495CF9C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
0x000000DB495CF9F1  00 5e 4c c7 83 01 00 33 ad 85 aa fb 7f 00 00 00 00 00 00 00 0
0x000000DB495CFA22  34 7b f6 7f 00 00 00 70 02 00 00 00 00 00 00 00 01 00 00 0
0x000000DB495CFA53  00 65 00 5c 00 3b 24 35 7b f6 7f 00 00 27 30 36 7b f6 7f 00 0
0x000000DB495CFA84  0a 00 00 00 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFAB5  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc 84 f
0x000000DB495CFAE6  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFB17  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFB48  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFB79  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFBAA  5c 49 db 00 00 00 e0 fb 5c 49 db 00 00 00 7e 29 35 7b f6 7f 0
0x000000DB495CFBDB  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFC0C  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
0x000000DB495CFC3D  cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc c
```

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

void functionA() {
    int a = 10;
    int b = 20;
    int *p = &a;

    cout << "a= " << a << endl;  ≤ 1ms elapsed
    cout << "b= " << b << endl;

    *p = 100;

    cout << "a= " << a << endl;
    cout << "b= " << b << endl;
}

void functionB() {
    int a = 10;
    int b = 20;
    int c = 30;
    int *p = &b;

    cout << "a= " << a << endl;
    cout << "b= " << b << endl;
    cout << "c= " << c << endl;

    *p = 100;

    cout << "a= " << a << endl;
    cout << "b= " << b << endl;
    cout << "c= " << c << endl;
}
```

Name	Value
a	10
b	20
p	0x000000db495cfa84 {10}

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

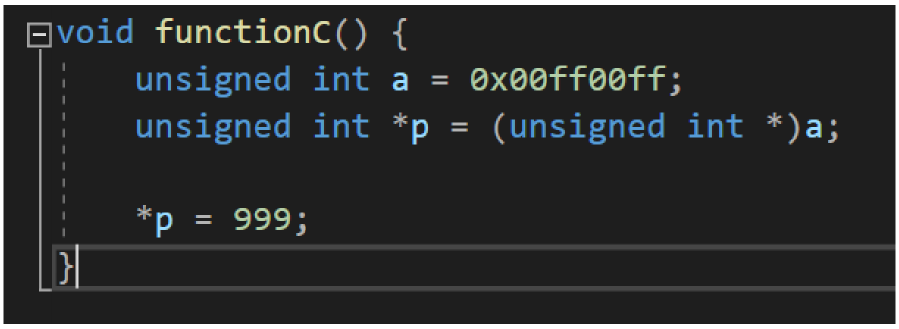
Sample output:
N/A

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:



```
void functionC() {
    unsigned int a = 0x00ff00ff;
    unsigned int *p = (unsigned int *)a;

    *p = 999;
}
```

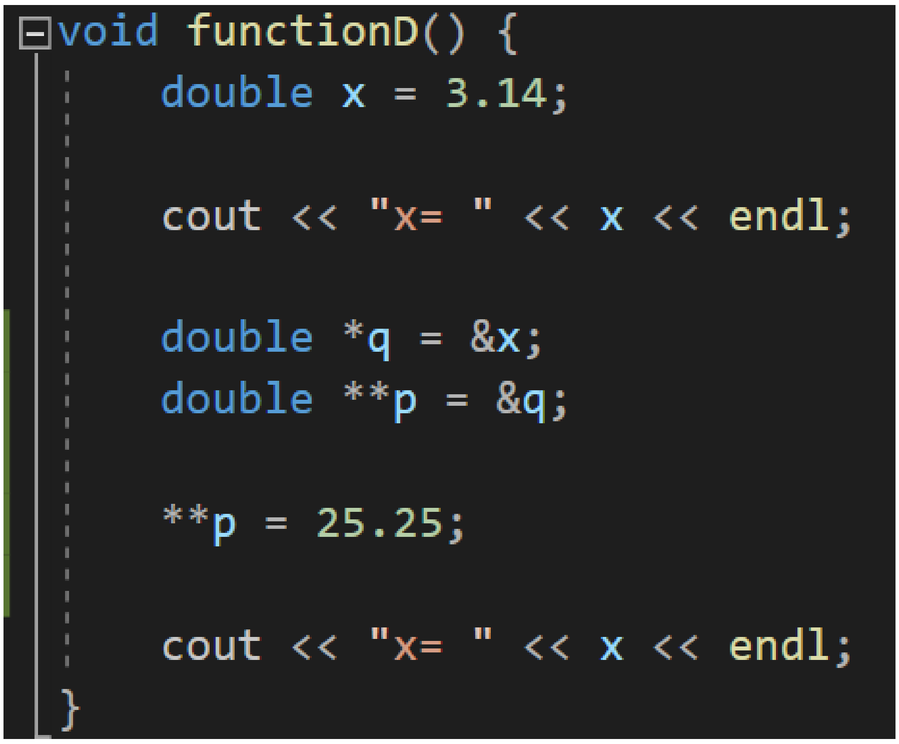
Test data:
N/A

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)

Solution:


```
void functionD() {
    double x = 3.14;

    cout << "x= " << x << endl;

    double *q = &x;
    double **p = &q;

    **p = 25.25;

    cout << "x= " << x << endl;
}
```

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

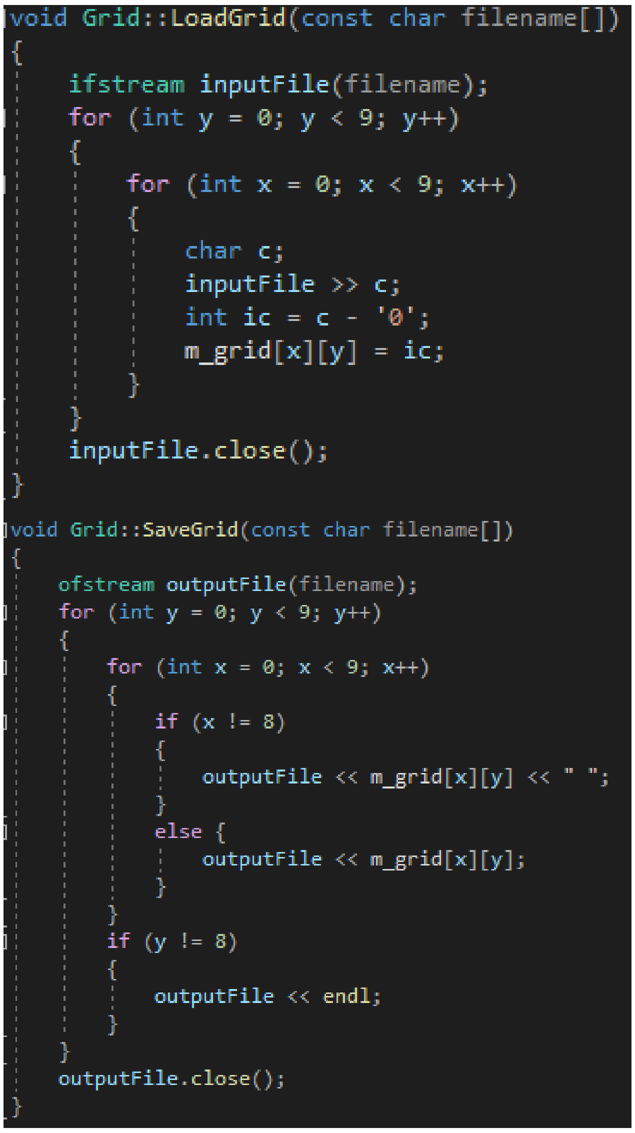
Question:

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:



```cpp
void Grid::LoadGrid(const char filename[])
{
    ifstream inputFile(filename);
    for (int y = 0; y < 9; y++)
    {
        for (int x = 0; x < 9; x++)
        {
            char c;
            inputFile >> c;
            int ic = c - '0';
            m_grid[x][y] = ic;
        }
    }
    inputFile.close();
}
void Grid::SaveGrid(const char filename[])
{
    ofstream outputFile(filename);
    for (int y = 0; y < 9; y++)
    {
        for (int x = 0; x < 9; x++)
        {
            if (x != 8)
            {
                outputFile << m_grid[x][y] << " ";
            }
            else {
                outputFile << m_grid[x][y];
            }
        }
        if (y != 8)
        {
            outputFile << endl;
        }
    }
    outputFile.close();
}
```

Test data:

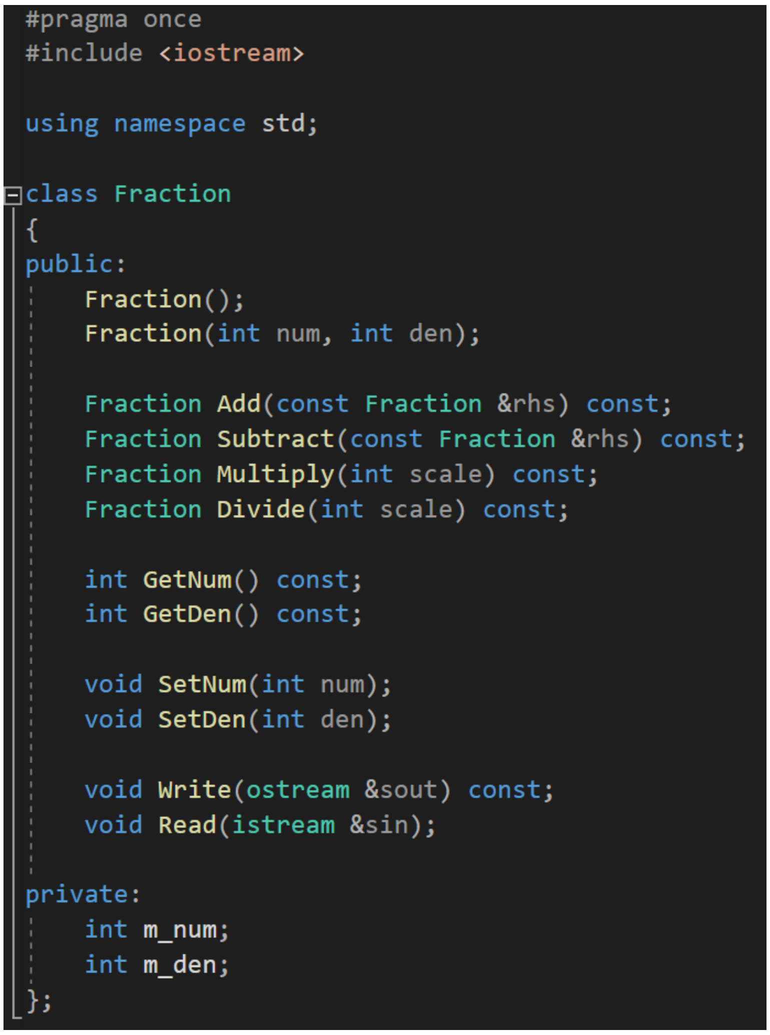
N/A

Sample output:
N/A

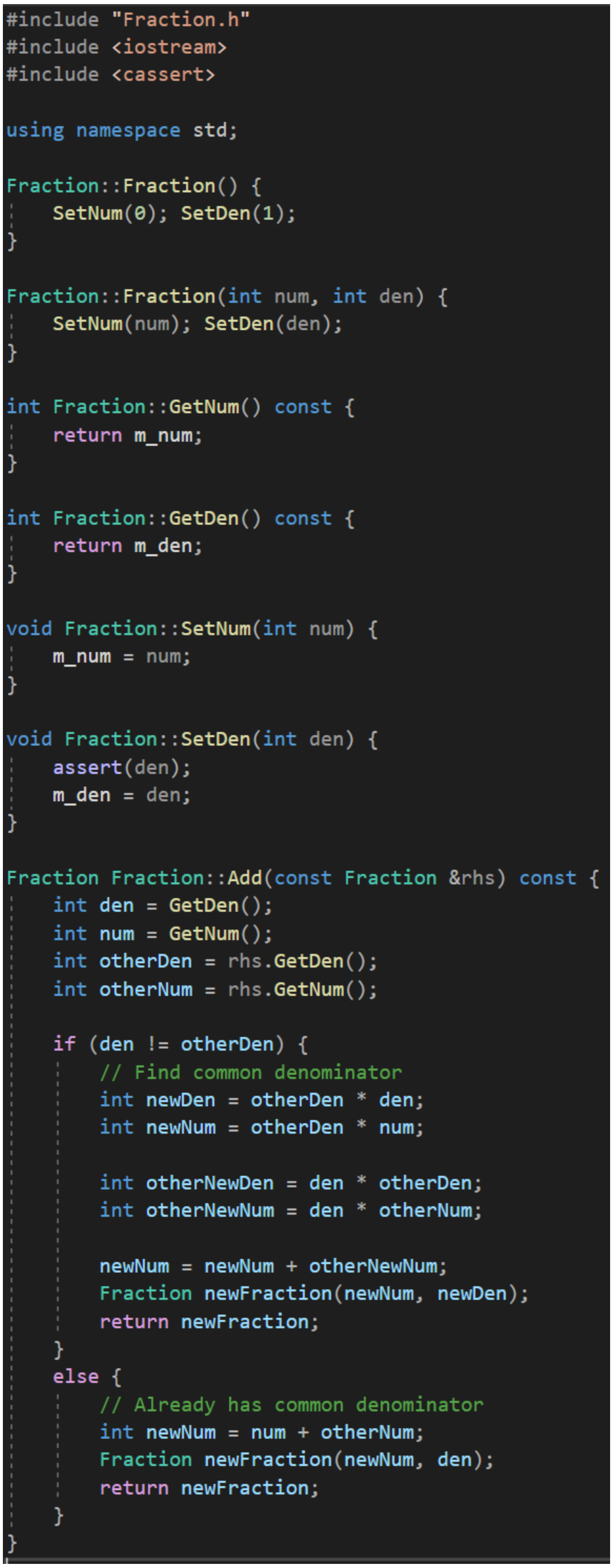
Reflection:
I originally used the << operator for writing the grid to a file so no changes needed to be made here. However, for reading from the grid, I originally used “inputFile.get()”. By doing so, I also had to use an if statement to check if the character it read was a space or not. A better and 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

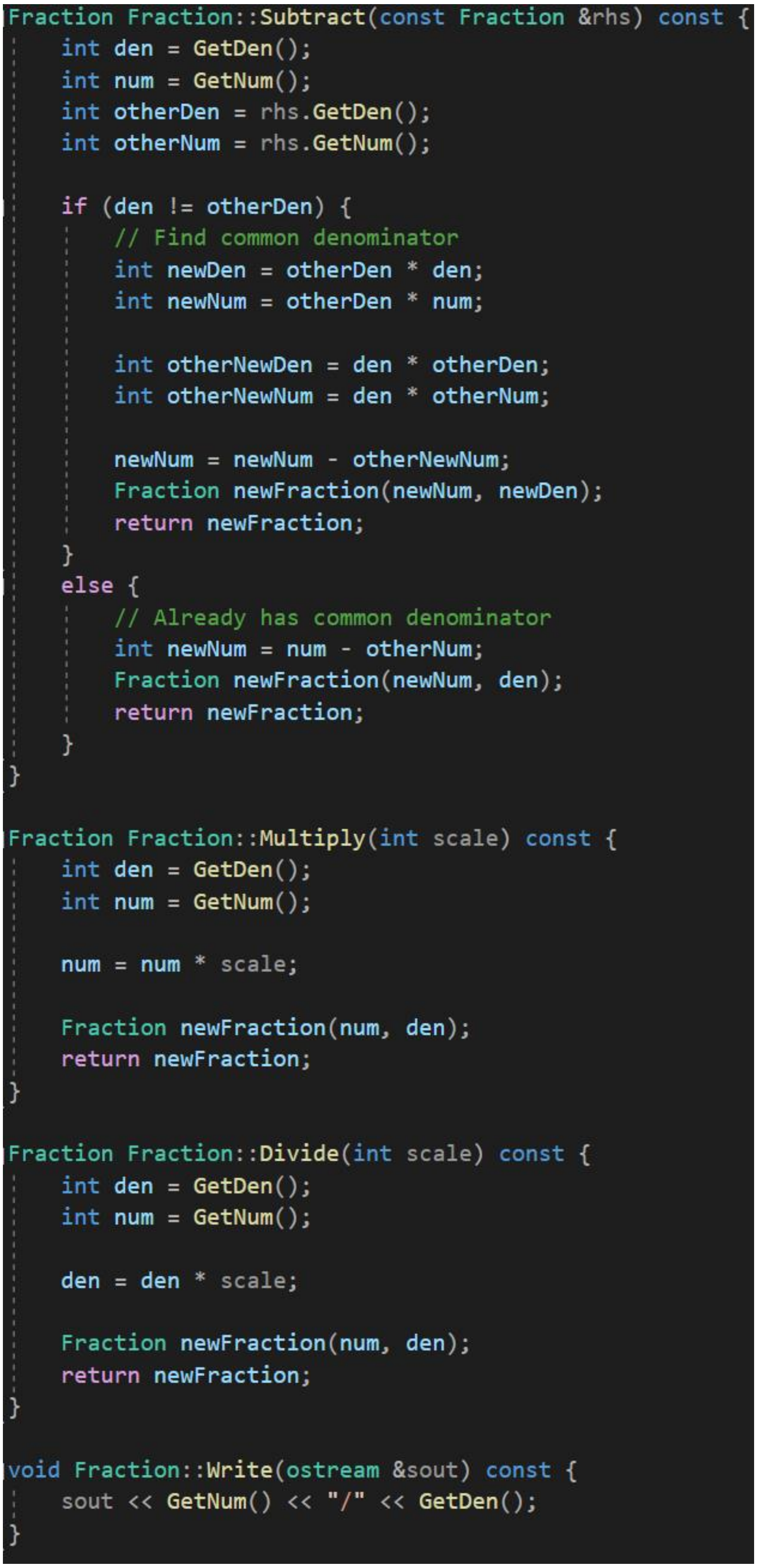
Question:
Implement the Fraction class that you have seen in lectures. Use the header file example that was presented in lectures to define your class, methods, member variables etc

Solution:


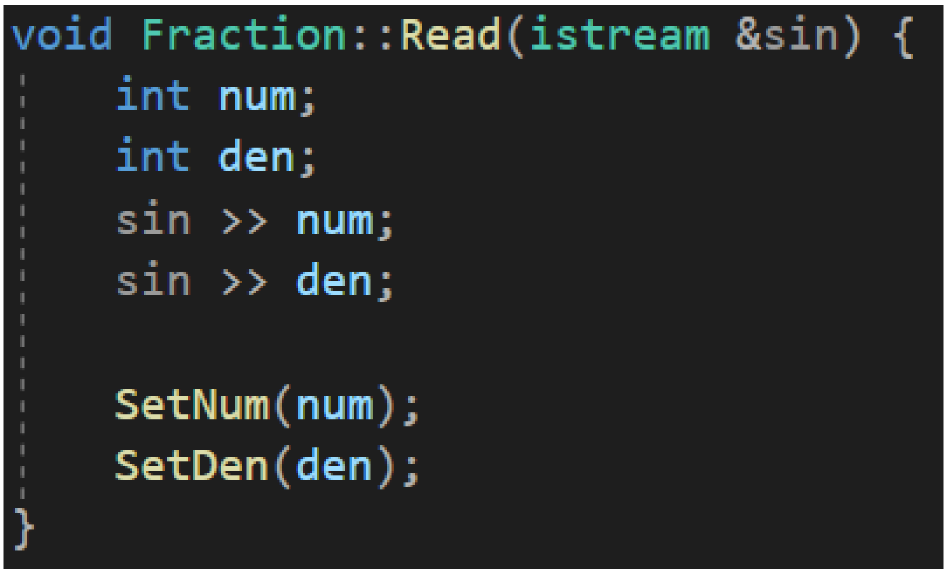
```
#pragma once
#include <iostream>

using namespace std;

class Fraction
{
public:
    Fraction();
    Fraction(int num, int den);

    Fraction Add(const Fraction &rhs) const;
    Fraction Subtract(const Fraction &rhs) const;
    Fraction Multiply(int scale) const;
    Fraction Divide(int scale) const;

    int GetNum() const;
    int GetDen() const;

    void SetNum(int num);
    void SetDen(int den);

    void Write(ostream &sout) const;
    void Read(istream &sin);

private:
    int m_num;
    int m_den;
};
```



```cpp
#include "Fraction.h"
#include <iostream>
#include <cassert>

using namespace std;

Fraction::Fraction() {
    SetNum(0); SetDen(1);
}

Fraction::Fraction(int num, int den) {
    SetNum(num); SetDen(den);
}

int Fraction::GetNum() const {
    return m_num;
}

int Fraction::GetDen() const {
    return m_den;
}

void Fraction::SetNum(int num) {
    m_num = num;
}

void Fraction::SetDen(int den) {
    assert(den);
    m_den = den;
}

Fraction Fraction::Add(const Fraction &rhs) const {
    int den = GetDen();
    int num = GetNum();
    int otherDen = rhs.GetDen();
    int otherNum = rhs.GetNum();

    if (den != otherDen) {
        // Find common denominator
        int newDen = otherDen * den;
        int newNum = otherDen * num;

        int otherNewDen = den * otherDen;
        int otherNewNum = den * otherNum;

        newNum = newNum + otherNewNum;
        Fraction newFraction(newNum, newDen);
        return newFraction;
    }
    else {
        // Already has common denominator
        int newNum = num + otherNum;
        Fraction newFraction(newNum, den);
        return newFraction;
    }
}
```



```cpp
Fraction Fraction::Subtract(const Fraction &rhs) const {
    int den = GetDen();
    int num = GetNum();
    int otherDen = rhs.GetDen();
    int otherNum = rhs.GetNum();

    if (den != otherDen) {
        // Find common denominator
        int newDen = otherDen * den;
        int newNum = otherDen * num;

        int otherNewDen = den * otherDen;
        int otherNewNum = den * otherNum;

        newNum = newNum - otherNewNum;
        Fraction newFraction(newNum, newDen);
        return newFraction;
    }
    else {
        // Already has common denominator
        int newNum = num - otherNum;
        Fraction newFraction(newNum, den);
        return newFraction;
    }
}

Fraction Fraction::Multiply(int scale) const {
    int den = GetDen();
    int num = GetNum();

    num = num * scale;

    Fraction newFraction(num, den);
    return newFraction;
}

Fraction Fraction::Divide(int scale) const {
    int den = GetDen();
    int num = GetNum();

    den = den * scale;

    Fraction newFraction(num, den);
    return newFraction;
}

void Fraction::Write(ostream &sout) const {
    sout << GetNum() << "/" << GetDen();
}
```



```
void Fraction::Read(istream &sin) {
    int num;
    int den;
    sin >> num;
    sin >> den;

    SetNum(num);
    SetDen(den);
}
```

Test data:
Input: 3 10

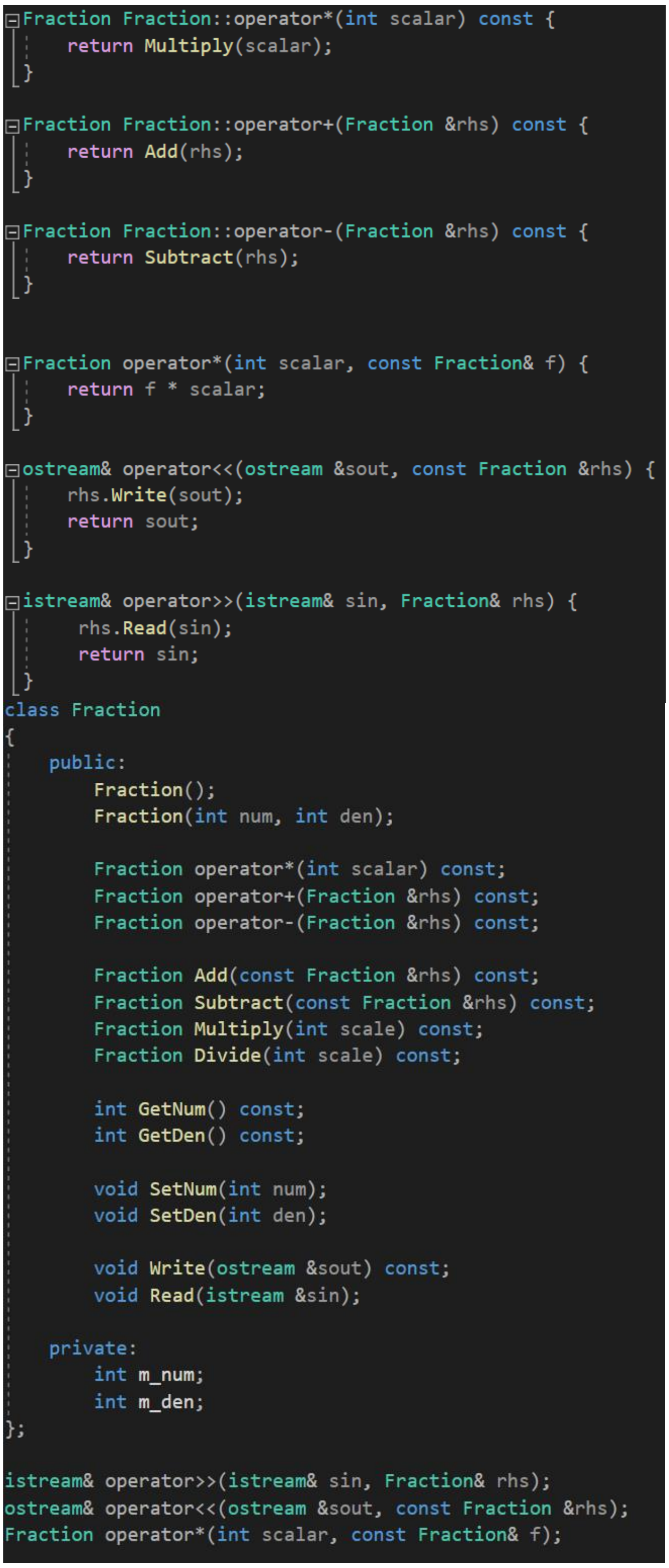
Sample output:
```
1/2 + 3/4 = 10/8

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.

Solution:



```
Fraction Fraction::operator*(int scalar) const {
    return Multiply(scalar);
}

Fraction Fraction::operator+(Fraction &rhs) const {
    return Add(rhs);
}

Fraction Fraction::operator-(Fraction &rhs) const {
    return Subtract(rhs);
}


Fraction operator*(int scalar, const Fraction& f) {
    return f * scalar;
}

ostream& operator<<(ostream &sout, const Fraction &rhs) {
    rhs.Write(sout);
    return sout;
}

istream& operator>>(istream& sin, Fraction& rhs) {
     rhs.Read(sin);
     return sin;
}
class Fraction
{
    public:
        Fraction();
        Fraction(int num, int den);

        Fraction operator*(int scalar) const;
        Fraction operator+(Fraction &rhs) const;
        Fraction operator-(Fraction &rhs) const;

        Fraction Add(const Fraction &rhs) const;
        Fraction Subtract(const Fraction &rhs) const;
        Fraction Multiply(int scale) const;
        Fraction Divide(int scale) const;

        int GetNum() const;
        int GetDen() const;

        void SetNum(int num);
        void SetDen(int den);

        void Write(ostream &sout) const;
        void Read(istream &sin);

    private:
        int m_num;
        int m_den;
};

istream& operator>>(istream& sin, Fraction& rhs);
ostream& operator<<(ostream &sout, const Fraction &rhs);
Fraction operator*(int scalar, const Fraction& f);
```

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:

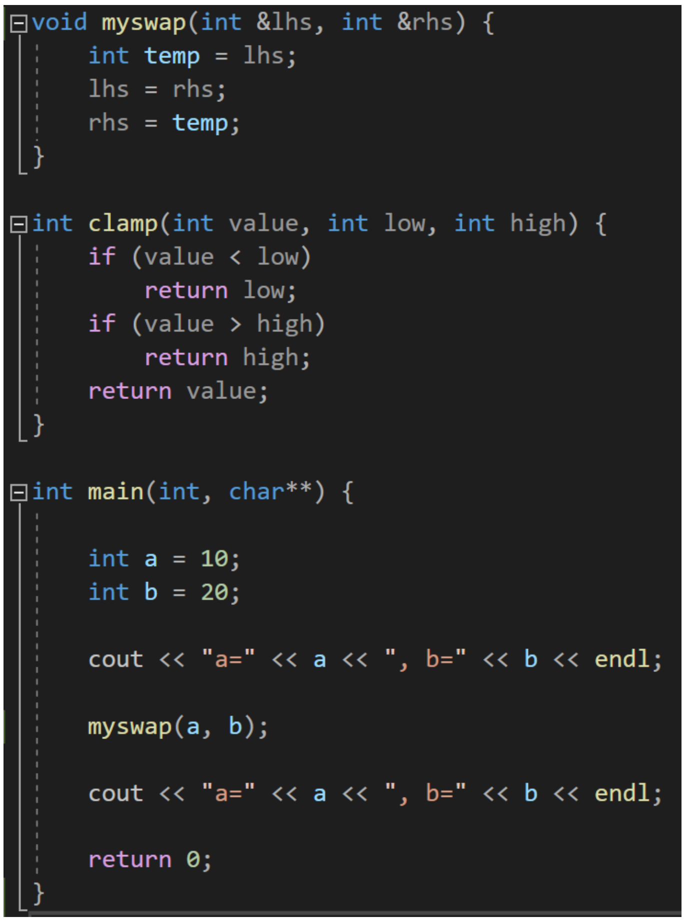
Operator member methods are written inside the class and auxiliary methods are written outside the class. Overloading an operator as a member method of a class can be used to replace code like Fraction.multiply(number) and instead be written as Fraction * number. On the left of the operator is an object 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.

Solution:



```cpp
void myswap(int &lhs, int &rhs) {
    int temp = lhs;
    lhs = rhs;
    rhs = temp;
}

int clamp(int value, int low, int high) {
    if (value < low)
        return low;
    if (value > high)
        return high;
    return value;
}

int main(int, char**) {

    int a = 10;
    int b = 20;

    cout << "a=" << a << ", b=" << b << endl;

    myswap(a, b);

    cout << "a=" << a << ", b=" << b << endl;

    return 0;
}
```

Test data:

N/A

Sample output:

N/A

Reflection:

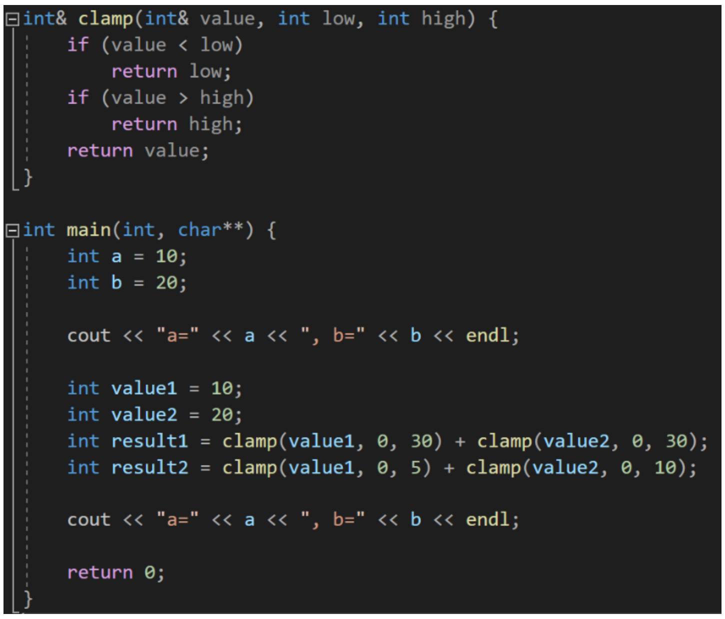
Pass by value essentially creates a copy of a variable. If an integer with the value of 10 is passed into a method, then the method will create a new integer with the value of 10. Any changes made to the variable within the method will only apply to the newly created integer and not the original one. 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.

Solution:



```
int& clamp(int& value, int low, int high) {
    if (value < low)
        return low;
    if (value > high)
        return high;
    return value;
}

int main(int, char**) {
    int a = 10;
    int b = 20;

    cout << "a=" << a << ", b=" << b << endl;

    int value1 = 10;
    int value2 = 20;
    int result1 = clamp(value1, 0, 30) + clamp(value2, 0, 30);
    int result2 = clamp(value1, 0, 5) + clamp(value2, 0, 10);

    cout << "a=" << a << ", b=" << b << endl;

    return 0;
}
```

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

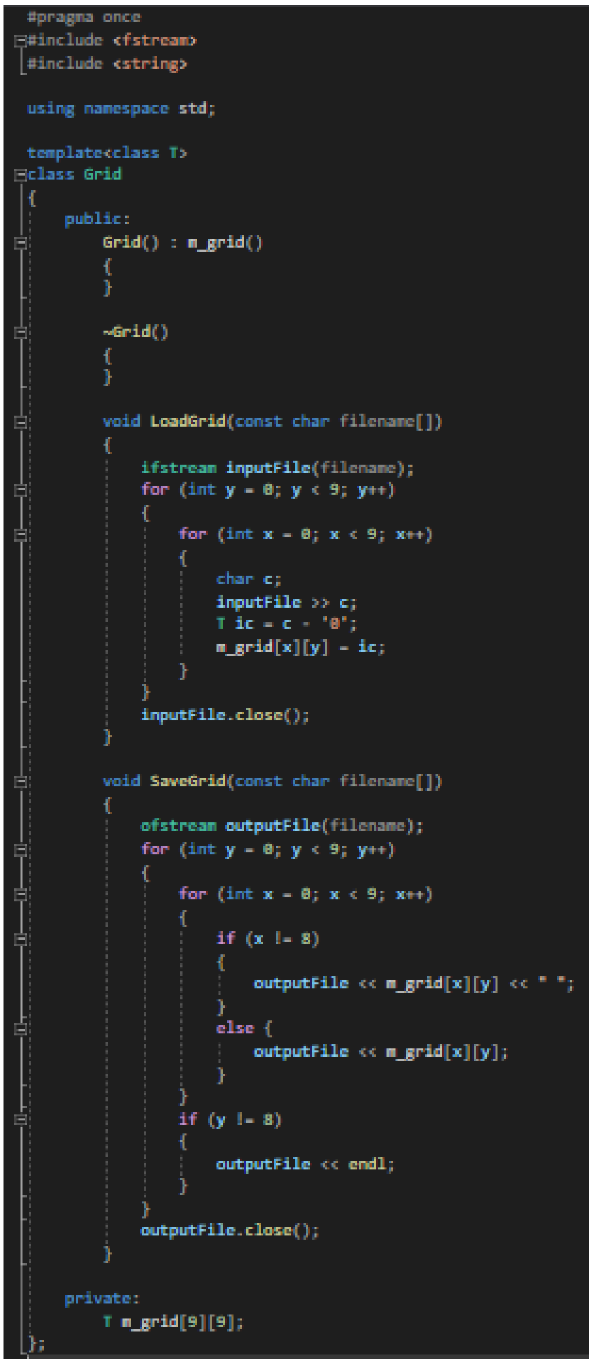
Date: 11th March 2022

Q1. Template Grid

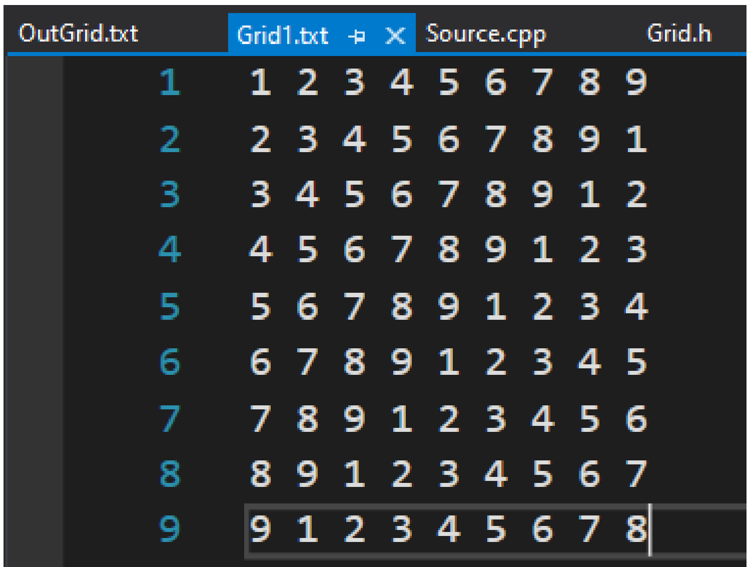
Question:

Add your Grid.h code

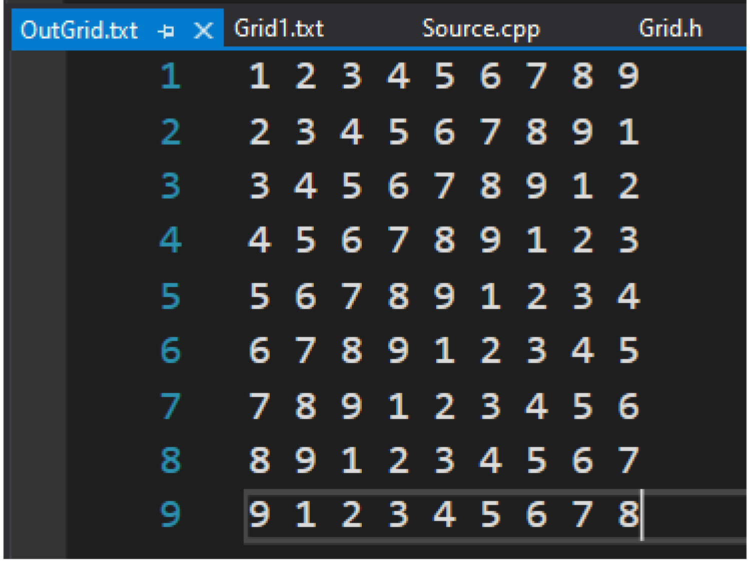
Solution:



```cpp
#pragma once
#include <fstream>
#include <string>

using namespace std;

template<class T>
class Grid
{
    public:
        Grid() : m_grid()
        {
        }

        ~Grid()
        {
        }

        void LoadGrid(const char filename[])
        {
            ifstream inputFile(filename);
            for (int y = 0; y < 9; y++)
            {
                for (int x = 0; x < 9; x++)
                {
                    char c;
                    inputFile >> c;
                    T ic = c - '0';
                    m_grid[x][y] = ic;
                }
            }
            inputFile.close();
        }

        void SaveGrid(const char filename[])
        {
            ofstream outputFile(filename);
            for (int y = 0; y < 9; y++)
            {
                for (int x = 0; x < 9; x++)
                {
                    if (x != 8)
                    {
                        outputFile << m_grid[x][y] << " ";
                    }
                    else {
                        outputFile << m_grid[x][y];
                    }
                }
                if (y != 8)
                {
                    outputFile << endl;
                }
            }
            outputFile.close();
        }

    private:
        T m_grid[9][9];
};
```


OutGrid.txt Grid1.txt Source.cpp Grid.h

```
1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 1
3 4 5 6 7 8 9 1 2
4 5 6 7 8 9 1 2 3
5 6 7 8 9 1 2 3 4
6 7 8 9 1 2 3 4 5
7 8 9 1 2 3 4 5 6
8 9 1 2 3 4 5 6 7
9 1 2 3 4 5 6 7 8
```


OutGrid.txt Grid1.txt Source.cpp Grid.h

```
1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 1
3 4 5 6 7 8 9 1 2
4 5 6 7 8 9 1 2 3
5 6 7 8 9 1 2 3 4
6 7 8 9 1 2 3 4 5
7 8 9 1 2 3 4 5 6
8 9 1 2 3 4 5 6 7
9 1 2 3 4 5 6 7 8
```

Test data:
N/A

Sample output:
N/A

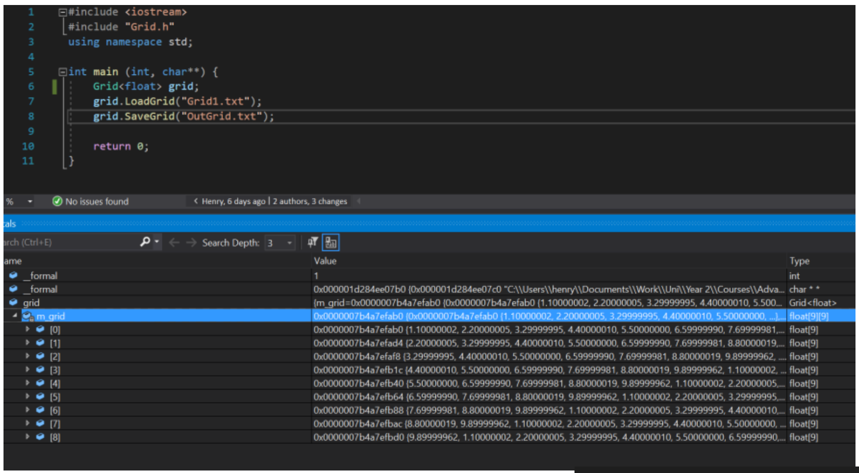
Reflection:
A template class can be used to change what type of objects are stored in the class. In this example, the class originally used an integer array, however, using a template class allows you to specify a different type of number like a float or a double when creating an instance of the class.

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.

Solution:



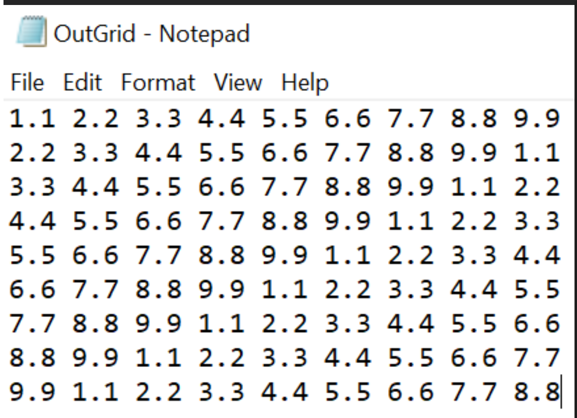
```cpp
#include <iostream>
#include "Grid.h"
using namespace std;

int main (int, char**) {
    Grid<float> grid;
    grid.LoadGrid("Grid1.txt");
    grid.SaveGrid("OutGrid.txt");

    return 0;
}
```

% No issues found   Henry, 6 days ago | 2 authors, 3 changes

als

rch (Ctrl+E)   Search Depth: 3

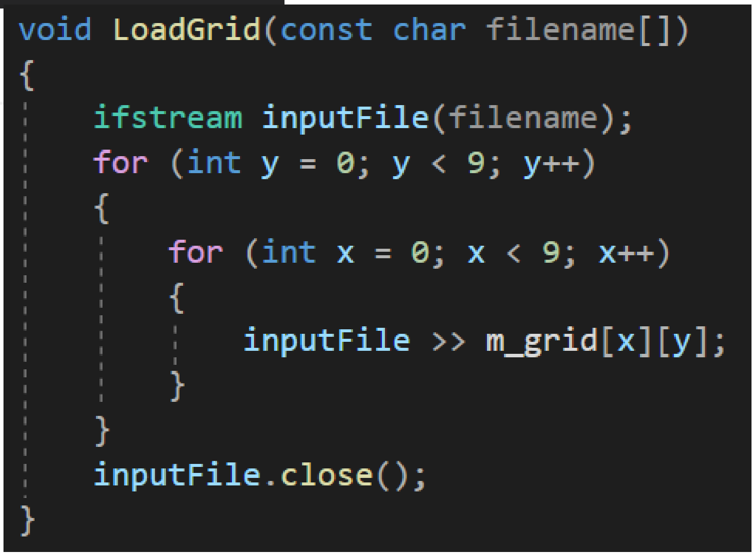
ame	Value	Type
_formal	1	int
_formal	0x000001d284ee07b0 {0x000001d284ee07c0 "C:\\Users\\henry\\Documents\\Work\\Uni\\Year 2\\Courses\\Adva...	char * *
grid	{m_grid=0x0000007b4a7efab0 {0x0000007b4a7efab0 {1.10000002, 2.20000005, 3.29999995, 4.40000010, 5.500...	Grid<float>
m_grid	0x0000007b4a7efab0 {0x0000007b4a7efab0 {1.10000002, 2.20000005, 3.29999995, 4.40000010, 5.50000000, ...},...	float[9][9]
[0]	0x0000007b4a7efab0 {1.10000002, 2.20000005, 3.29999995, 4.40000010, 5.50000000, 6.59999990, 7.69999981,...	float[9]
[1]	0x0000007b4a7efad4 {2.20000005, 3.29999995, 4.40000010, 5.50000000, 6.59999990, 7.69999981, 8.80000019,...	float[9]
[2]	0x0000007b4a7efaf8 {3.29999995, 4.40000010, 5.50000000, 6.59999990, 7.69999981, 8.80000019, 9.89999962, ...	float[9]
[3]	0x0000007b4a7efb1c {4.40000010, 5.50000000, 6.59999990, 7.69999981, 8.80000019, 9.89999962, 1.10000002, ...	float[9]
[4]	0x0000007b4a7efb40 {5.50000000, 6.59999990, 7.69999981, 8.80000019, 9.89999962, 1.10000002, 2.20000005,...	float[9]
[5]	0x0000007b4a7efb64 {6.59999990, 7.69999981, 8.80000019, 9.89999962, 1.10000002, 2.20000005, 3.29999995,...	float[9]
[6]	0x0000007b4a7efb88 {7.69999981, 8.80000019, 9.89999962, 1.10000002, 2.20000005, 3.29999995, 4.40000010,...	float[9]
[7]	0x0000007b4a7efbac {8.80000019, 9.89999962, 1.10000002, 2.20000005, 3.29999995, 4.40000010, 5.50000000, ...	float[9]
[8]	0x0000007b4a7efbd0 {9.89999962, 1.10000002, 2.20000005, 3.29999995, 4.40000010, 5.50000000, 6.59999990,...	float[9]



OutGrid - Notepad

File Edit Format View Help

```
1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9
2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9 1.1
3.3 4.4 5.5 6.6 7.7 8.8 9.9 1.1 2.2
4.4 5.5 6.6 7.7 8.8 9.9 1.1 2.2 3.3
5.5 6.6 7.7 8.8 9.9 1.1 2.2 3.3 4.4
6.6 7.7 8.8 9.9 1.1 2.2 3.3 4.4 5.5
7.7 8.8 9.9 1.1 2.2 3.3 4.4 5.5 6.6
8.8 9.9 1.1 2.2 3.3 4.4 5.5 6.6 7.7
9.9 1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8
```



```cpp
void LoadGrid(const char filename[])
{
    ifstream inputFile(filename);
    for (int y = 0; y < 9; y++)
    {
        for (int x = 0; x < 9; x++)
        {
            inputFile >> m_grid[x][y];
        }
    }
    inputFile.close();
}
```

Test data:

N/A

Sample output:

N/A

Reflection:

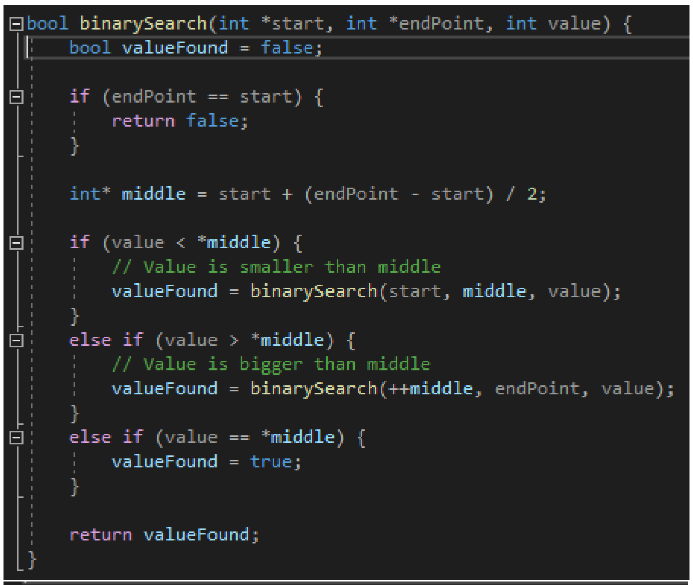
There is a much simpler way of doing what I was trying to do. Replacing all the code in load grid from the last task with a simple “inputFile >> m_grid[x][y]” would allow me to read from the file better. Previously, it would go character by character, and this is a problem when reading a floating point 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

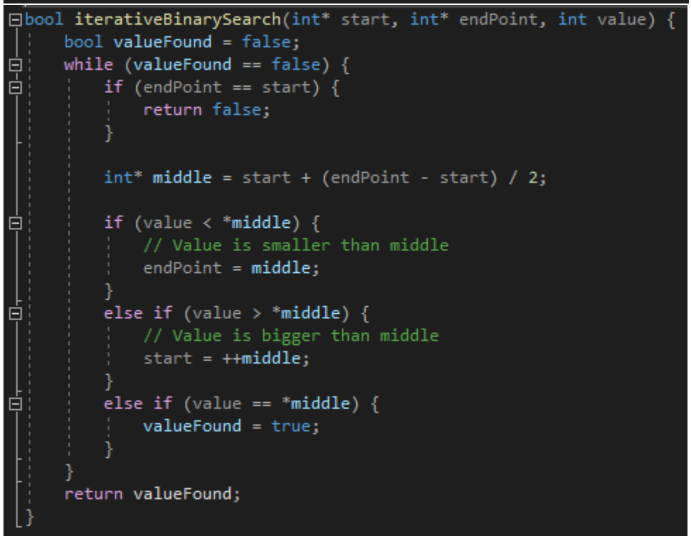
Question:

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

Solution:



```
bool binarySearch(int *start, int *endPoint, int value) {
    bool valueFound = false;

    if (endPoint == start) {
        return false;
    }

    int* middle = start + (endPoint - start) / 2;

    if (value < *middle) {
        // Value is smaller than middle
        valueFound = binarySearch(start, middle, value);
    }
    else if (value > *middle) {
        // Value is bigger than middle
        valueFound = binarySearch(++middle, endPoint, value);
    }
    else if (value == *middle) {
        valueFound = true;
    }

    return valueFound;
}
```



```
bool iterativeBinarySearch(int* start, int* endPoint, int value) {
    bool valueFound = false;
    while (valueFound == false) {
        if (endPoint == start) {
            return false;
        }

        int* middle = start + (endPoint - start) / 2;

        if (value < *middle) {
            // Value is smaller than middle
            endPoint = middle;
        }
        else if (value > *middle) {
            // Value is bigger than middle
            start = ++middle;
        }
        else if (value == *middle) {
            valueFound = true;
        }
    }
    return valueFound;
}
```

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

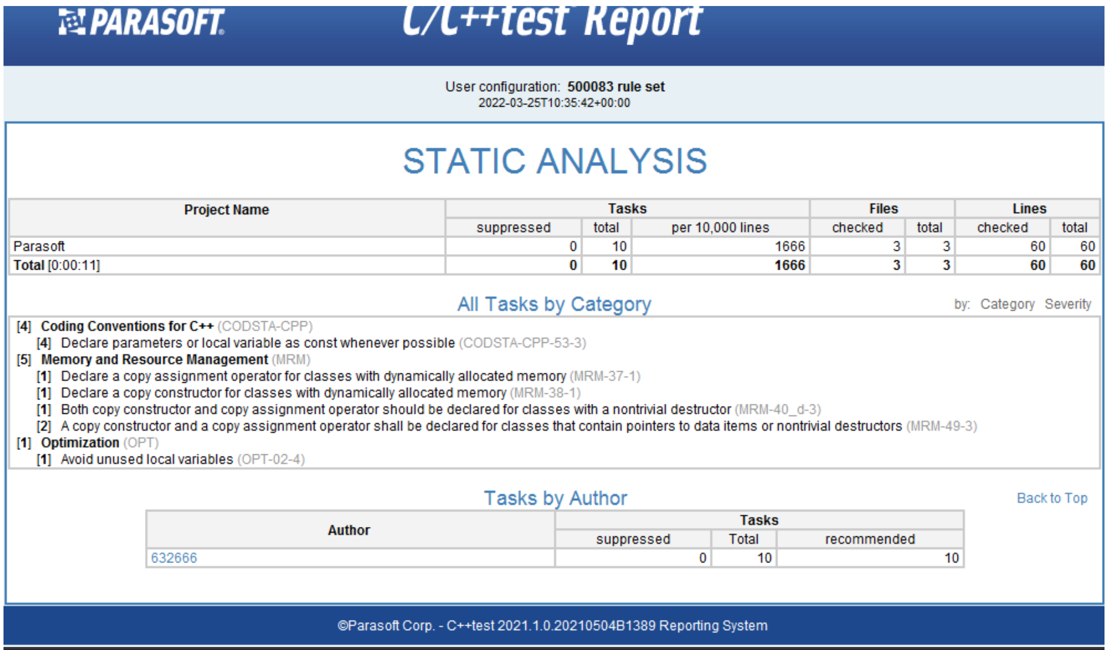
Date: 18th March 2022

Q1. Parasoft

Question:

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

Solution:


PARASOFT C/C++test Report

User configuration: **500083 rule set**
2022-03-25T10:35:42+00:00

STATIC ANALYSIS

Project Name	Tasks			Files		Lines	
	suppressed	total	per 10,000 lines	checked	total	checked	total
Parasoft	0	10	1666	3	3	60	60
Total [0:00:11]	**0**	**10**	**1666**	**3**	**3**	**60**	**60**

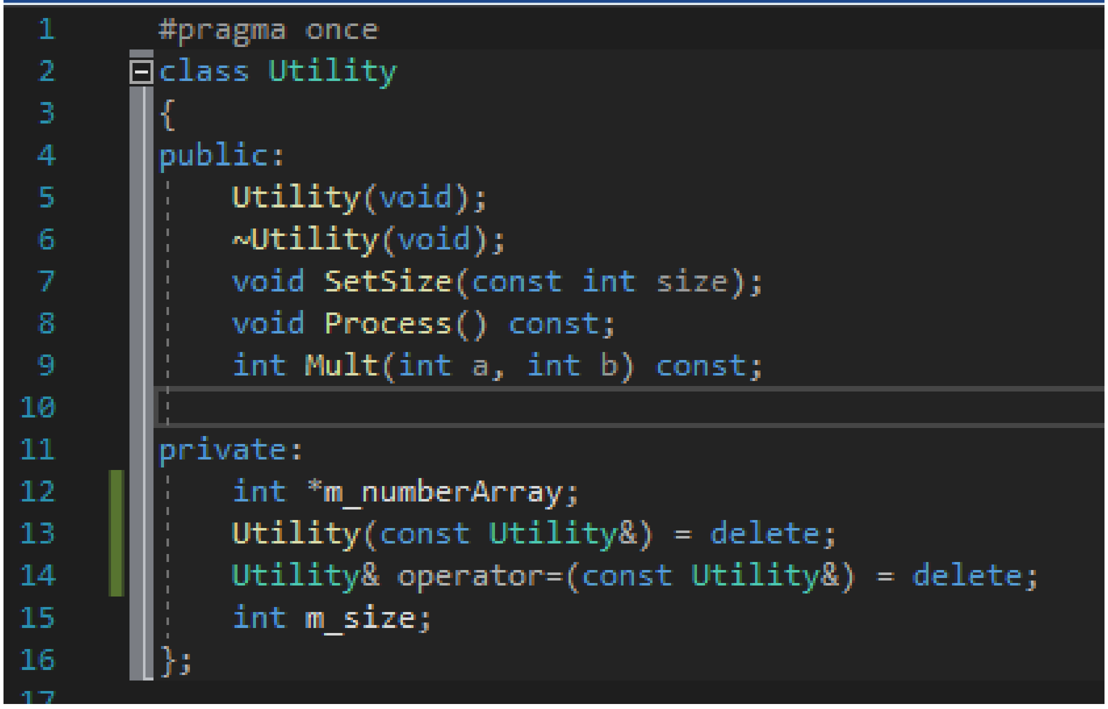
All Tasks by Category — by: Category Severity

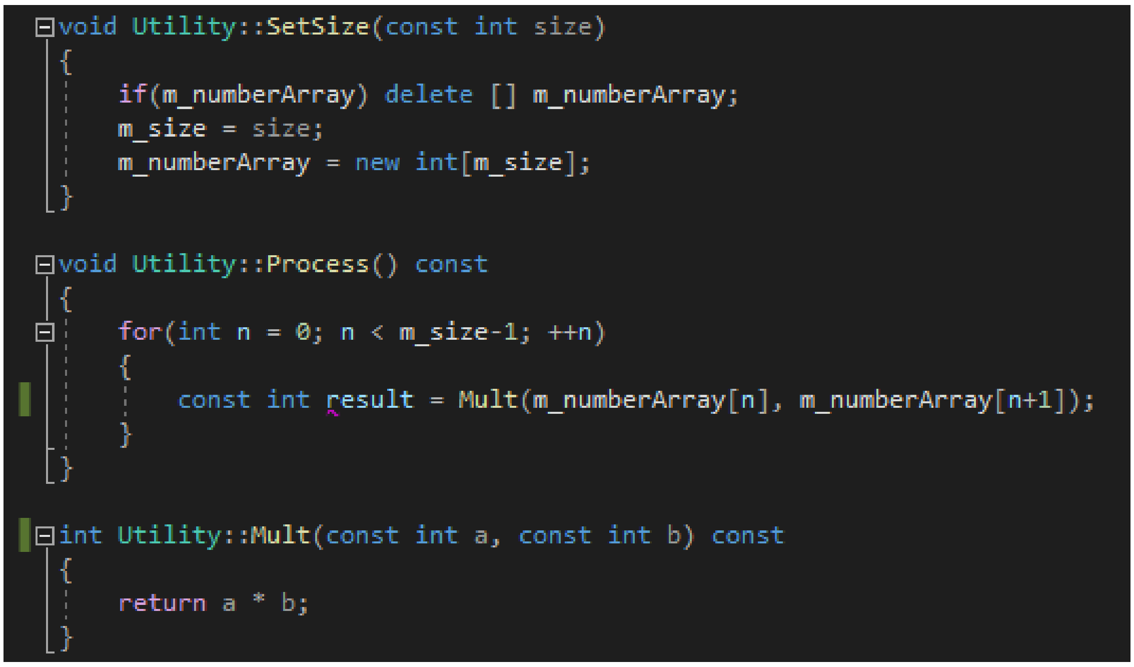
- [4] **Coding Conventions for C++** (CODSTA-CPP)
  - [4] Declare parameters or local variable as const whenever possible (CODSTA-CPP-53-3)
- [5] **Memory and Resource Management** (MRM)
  - [1] Declare a copy assignment operator for classes with dynamically allocated memory (MRM-37-1)
  - [1] Declare a copy constructor for classes with dynamically allocated memory (MRM-38-1)
  - [1] Both copy constructor and copy assignment operator should be declared for classes with a nontrivial destructor (MRM-40_d-3)
  - [2] A copy constructor and a copy assignment operator shall be declared for classes that contain pointers to data items or nontrivial destructors (MRM-49-3)
- [1] **Optimization** (OPT)
  - [1] Avoid unused local variables (OPT-02-4)

Tasks by Author — Back to Top

Author	Tasks		
	suppressed	Total	recommended
632666	0	10	10

©Parasoft Corp. - C++test 2021.1.0.20210504B1389 Reporting System


```cpp
#pragma once
class Utility
{
public:
    Utility(void);
    ~Utility(void);
    void SetSize(const int size);
    void Process() const;
    int Mult(int a, int b) const;

private:
    int *m_numberArray;
    Utility(const Utility&) = delete;
    Utility& operator=(const Utility&) = delete;
    int m_size;
};
```



```
void Utility::SetSize(const int size)
{
    if(m_numberArray) delete [] m_numberArray;
    m_size = size;
    m_numberArray = new int[m_size];
}

void Utility::Process() const
{
    for(int n = 0; n < m_size-1; ++n)
    {
        const int result = Mult(m_numberArray[n], m_numberArray[n+1]);
    }
}

int Utility::Mult(const int a, const int b) const
{
    return a * b;
}
```

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

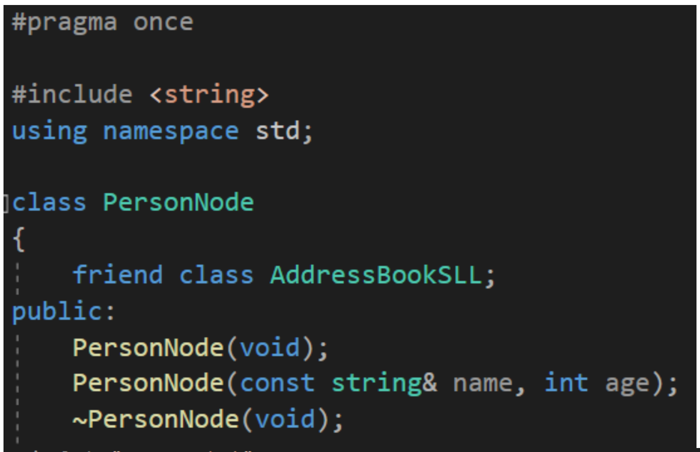
Date: 25th March 2022

Q1. PersonNode

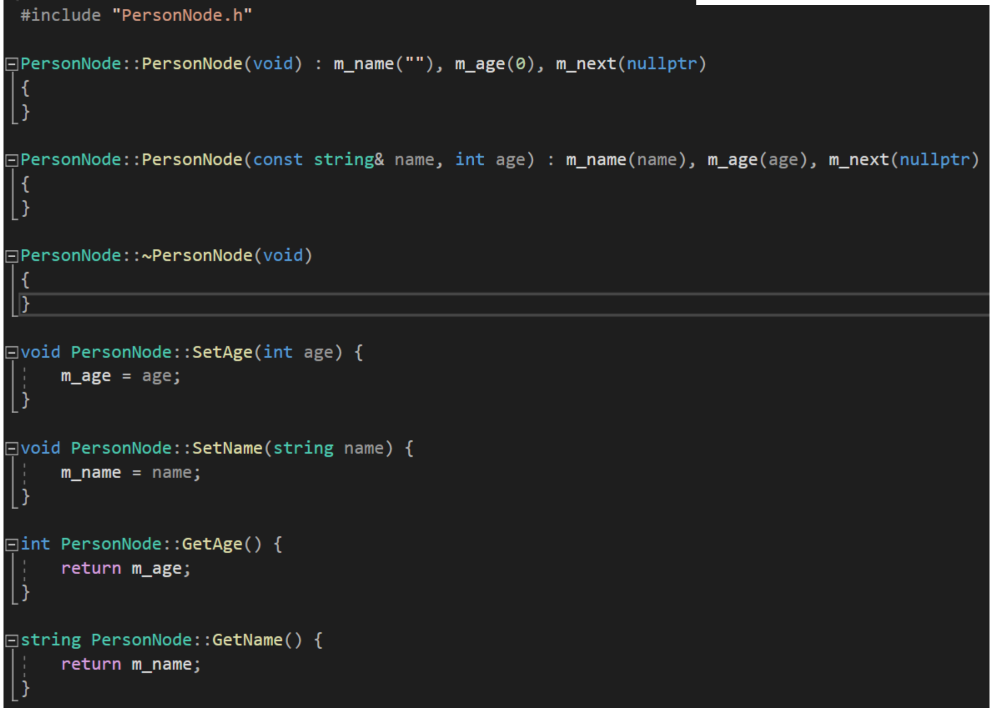
Question:

Add your code and describe what you have done.

Solution:



```cpp
#pragma once

#include <string>
using namespace std;

class PersonNode
{
    friend class AddressBookSLL;
public:
    PersonNode(void);
    PersonNode(const string& name, int age);
    ~PersonNode(void);
```



```cpp
#include "PersonNode.h"

PersonNode::PersonNode(void) : m_name(""), m_age(0), m_next(nullptr)
{
}

PersonNode::PersonNode(const string& name, int age) : m_name(name), m_age(age), m_next(nullptr)
{
}

PersonNode::~PersonNode(void)
{
}

void PersonNode::SetAge(int age) {
    m_age = age;
}

void PersonNode::SetName(string name) {
    m_name = name;
}

int PersonNode::GetAge() {
    return m_age;
}

string PersonNode::GetName() {
    return m_name;
}
```

Test data:

N/A

Sample output:

N/A

Reflection:

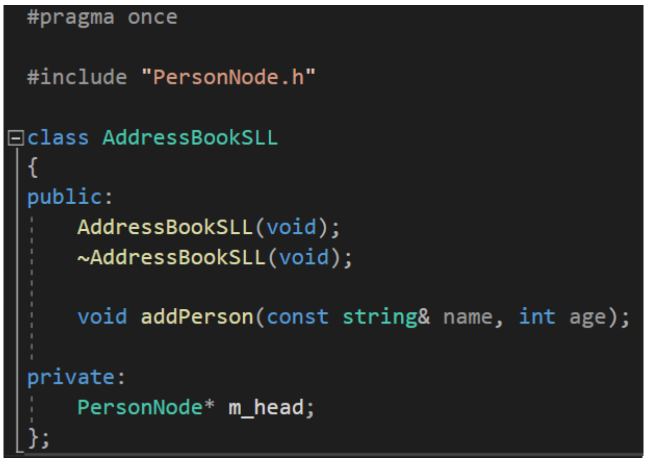
I added public get and set methods for the name and age member variables in the person node class and also added the AddressBookSLL class as a friend so that the person node class can directly access private values and methods inside of the address book class.

Q2. AddressBookSLL

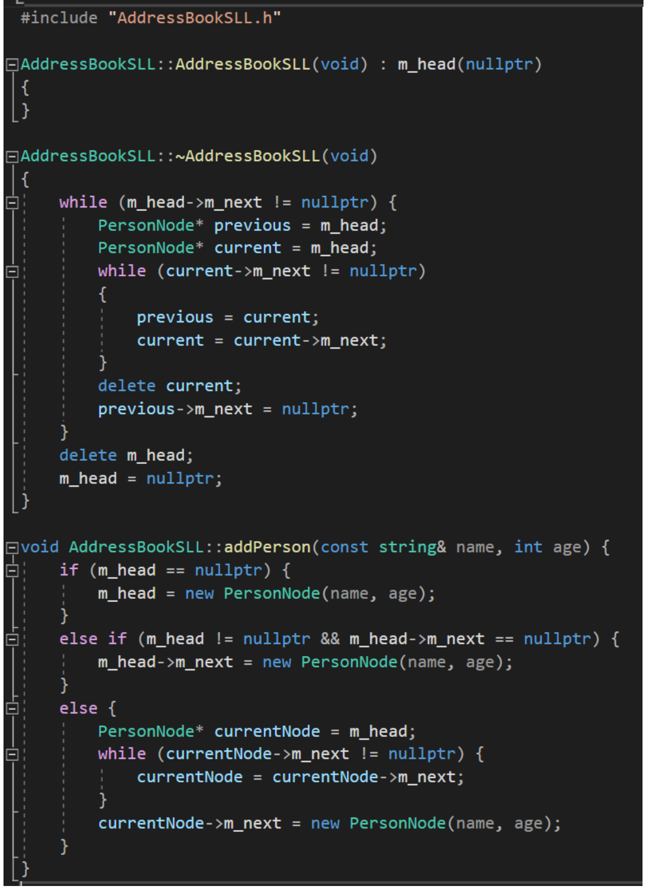
Question:

Add your code and describe what you have done

Solution:



```cpp
#pragma once

#include "PersonNode.h"

class AddressBookSLL
{
public:
    AddressBookSLL(void);
    ~AddressBookSLL(void);

    void addPerson(const string& name, int age);

private:
    PersonNode* m_head;
};
```



```cpp
#include "AddressBookSLL.h"

AddressBookSLL::AddressBookSLL(void) : m_head(nullptr)
{
}

AddressBookSLL::~AddressBookSLL(void)
{
    while (m_head->m_next != nullptr) {
        PersonNode* previous = m_head;
        PersonNode* current = m_head;
        while (current->m_next != nullptr)
        {
            previous = current;
            current = current->m_next;
        }
        delete current;
        previous->m_next = nullptr;
    }
    delete m_head;
    m_head = nullptr;
}

void AddressBookSLL::addPerson(const string& name, int age) {
    if (m_head == nullptr) {
        m_head = new PersonNode(name, age);
    }
    else if (m_head != nullptr && m_head->m_next == nullptr) {
        m_head->m_next = new PersonNode(name, age);
    }
    else {
        PersonNode* currentNode = m_head;
        while (currentNode->m_next != nullptr) {
            currentNode = currentNode->m_next;
        }
        currentNode->m_next = new PersonNode(name, age);
    }
}
```

Test data:
N/A

Sample output:
N/A

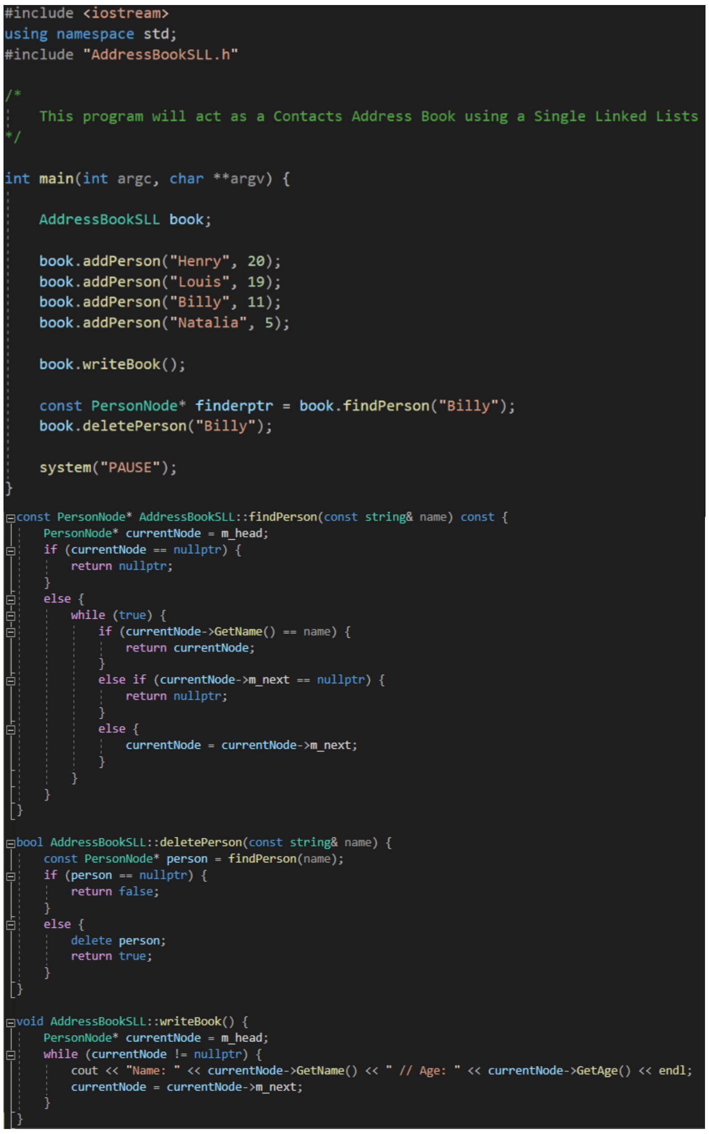
Reflection:
The add person method in the address book class has an if statement inside it that determines the address of the new person to be added. If the head pointer is null (there are no people in the address book) then the person to be added will take the place of the head. If the head pointer is 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

Solution:



```cpp
#include <iostream>
using namespace std;
#include "AddressBookSLL.h"

/*
    This program will act as a Contacts Address Book using a Single Linked Lists
*/

int main(int argc, char **argv) {

    AddressBookSLL book;

    book.addPerson("Henry", 20);
    book.addPerson("Louis", 19);
    book.addPerson("Billy", 11);
    book.addPerson("Natalia", 5);

    book.writeBook();

    const PersonNode* finderptr = book.findPerson("Billy");
    book.deletePerson("Billy");

    system("PAUSE");
}
const PersonNode* AddressBookSLL::findPerson(const string& name) const {
    PersonNode* currentNode = m_head;
    if (currentNode == nullptr) {
        return nullptr;
    }
    else {
        while (true) {
            if (currentNode->GetName() == name) {
                return currentNode;
            }
            else if (currentNode->m_next == nullptr) {
                return nullptr;
            }
            else {
                currentNode = currentNode->m_next;
            }
        }
    }
}

bool AddressBookSLL::deletePerson(const string& name) {
    const PersonNode* person = findPerson(name);
    if (person == nullptr) {
        return false;
    }
    else {
        delete person;
        return true;
    }
}

void AddressBookSLL::writeBook() {
    PersonNode* currentNode = m_head;
    while (currentNode != nullptr) {
        cout << "Name: " << currentNode->GetName() << " // Age: " << currentNode->GetAge() << endl;
        currentNode = currentNode->m_next;
    }
}
```

Test data:

N/A

Sample output:

```
Name: Henry // Age: 20

Name: Louis // Age: 19

Name: Billy // Age: 11

Name: Natalia // Age: 5
```

Reflection:

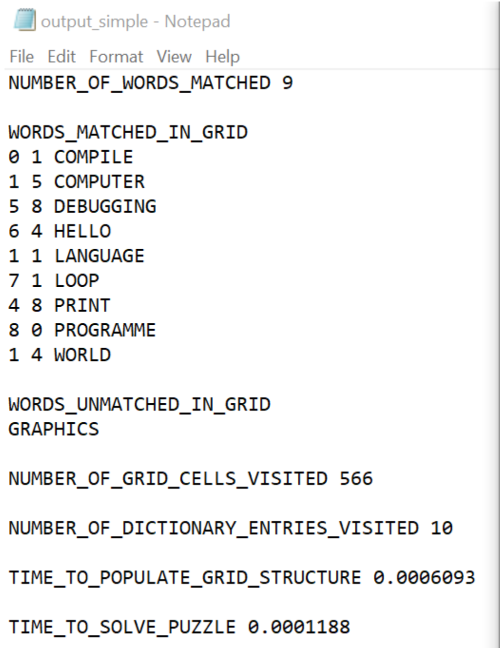
The find person method works by looping through each node in the list and checking the name associated with it against the name provided. If it is found, it returns the pointer to that node. If the list doesn’t contain a node with that name, a null pointer will be returned instead. The delete person 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.

Week 9 – Lab I

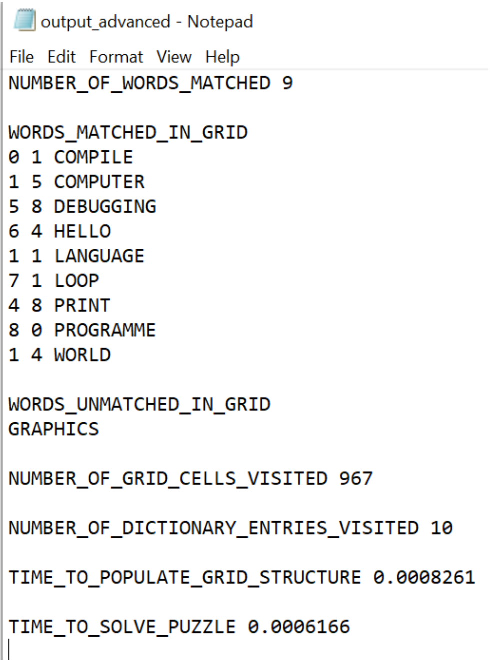
Date: 1st April 2022

Timing and activity results

Example puzzle 1

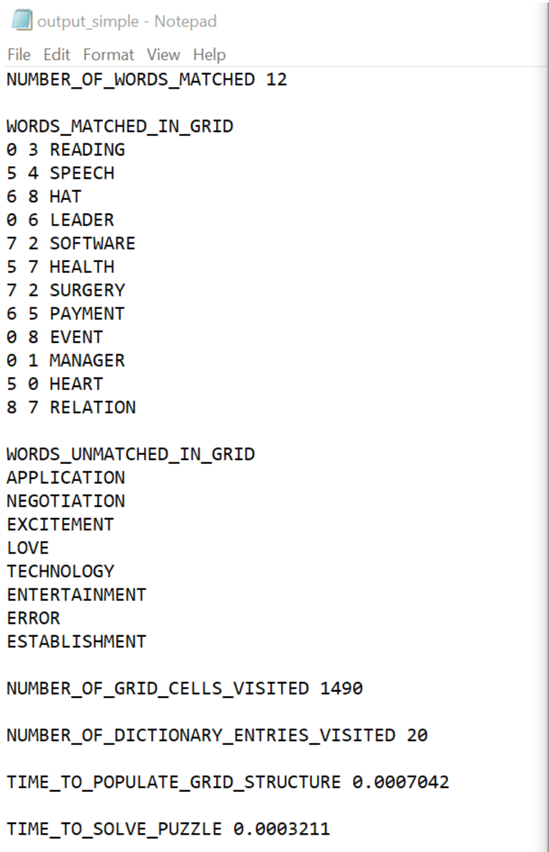


```
NUMBER_OF_WORDS_MATCHED 9

WORDS_MATCHED_IN_GRID
0 1 COMPILE
1 5 COMPUTER
5 8 DEBUGGING
6 4 HELLO
1 1 LANGUAGE
7 1 LOOP
4 8 PRINT
8 0 PROGRAMME
1 4 WORLD

WORDS_UNMATCHED_IN_GRID
GRAPHICS

NUMBER_OF_GRID_CELLS_VISITED 566

NUMBER_OF_DICTIONARY_ENTRIES_VISITED 10

TIME_TO_POPULATE_GRID_STRUCTURE 0.0006093

TIME_TO_SOLVE_PUZZLE 0.0001188
```

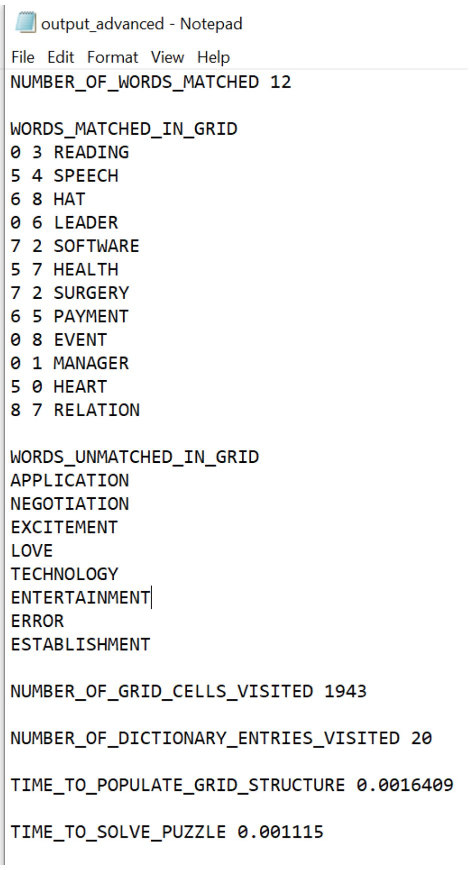


```
NUMBER_OF_WORDS_MATCHED 9

WORDS_MATCHED_IN_GRID
0 1 COMPILE
1 5 COMPUTER
5 8 DEBUGGING
6 4 HELLO
1 1 LANGUAGE
7 1 LOOP
4 8 PRINT
8 0 PROGRAMME
1 4 WORLD

WORDS_UNMATCHED_IN_GRID
GRAPHICS

NUMBER_OF_GRID_CELLS_VISITED 967

NUMBER_OF_DICTIONARY_ENTRIES_VISITED 10

TIME_TO_POPULATE_GRID_STRUCTURE 0.0008261

TIME_TO_SOLVE_PUZZLE 0.0006166
```

Example puzzle 2



```
NUMBER_OF_WORDS_MATCHED 12

WORDS_MATCHED_IN_GRID
0 3 READING
5 4 SPEECH
6 8 HAT
0 6 LEADER
7 2 SOFTWARE
5 7 HEALTH
7 2 SURGERY
6 5 PAYMENT
0 8 EVENT
0 1 MANAGER
5 0 HEART
8 7 RELATION

WORDS_UNMATCHED_IN_GRID
APPLICATION
NEGOTIATION
EXCITEMENT
LOVE
TECHNOLOGY
ENTERTAINMENT
ERROR
ESTABLISHMENT

NUMBER_OF_GRID_CELLS_VISITED 1490

NUMBER_OF_DICTIONARY_ENTRIES_VISITED 20

TIME_TO_POPULATE_GRID_STRUCTURE 0.0007042

TIME_TO_SOLVE_PUZZLE 0.0003211
```

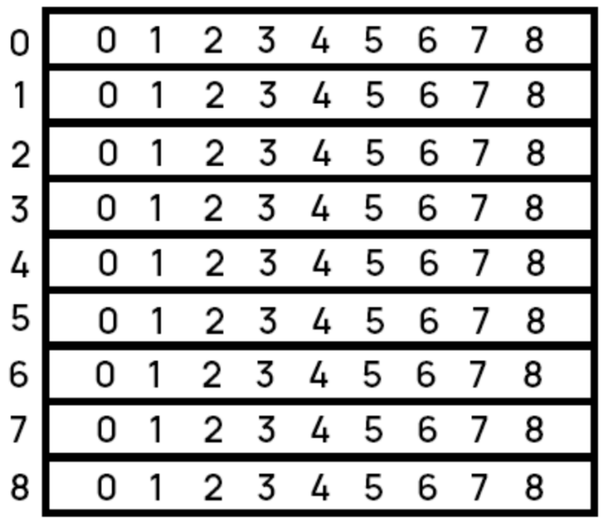


```
NUMBER_OF_WORDS_MATCHED 12

WORDS_MATCHED_IN_GRID
0 3 READING
5 4 SPEECH
6 8 HAT
0 6 LEADER
7 2 SOFTWARE
5 7 HEALTH
7 2 SURGERY
6 5 PAYMENT
0 8 EVENT
0 1 MANAGER
5 0 HEART
8 7 RELATION

WORDS_UNMATCHED_IN_GRID
APPLICATION
NEGOTIATION
EXCITEMENT
LOVE
TECHNOLOGY
ENTERTAINMENT
ERROR
ESTABLISHMENT

NUMBER_OF_GRID_CELLS_VISITED 1943

NUMBER_OF_DICTIONARY_ENTRIES_VISITED 20

TIME_TO_POPULATE_GRID_STRUCTURE 0.0016409

TIME_TO_SOLVE_PUZZLE 0.001115
```

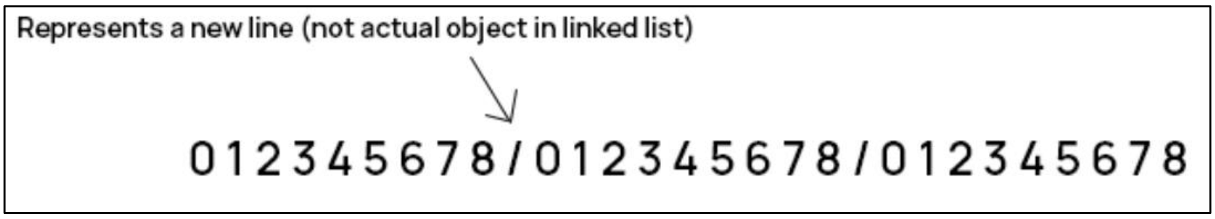
Results reflection

I decided to implement both the simple dictionary and puzzle structure as well as the advanced puzzle structure. In the advanced scenario, the simple dictionary structure will be used in conjunction with the advanced puzzle structure.

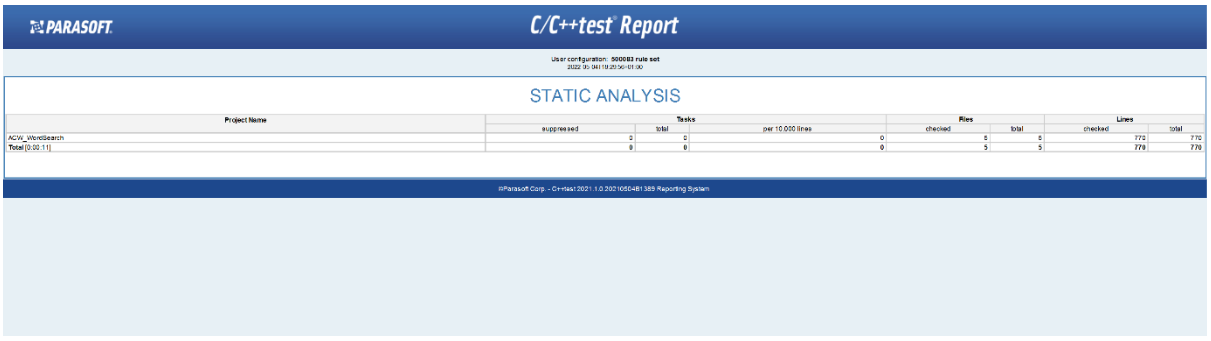
The simple puzzle uses a 2d array of characters to represent the grid. When it is time for the puzzle to be solved, a word will be taken from the dictionary and the grid will be searched for the first letter of the currently selected word. If the letter is found, each of the surrounding letters will be checked against the second letter in the selected word. If a new letter is found, the direction it was found in will be locked in until it either finds the complete word or finds a letter that doesn't belong to the word. At which point it will continue checking the original surrounding letters. If the correct sequence of letters hasn't been found once the end of the grid is reached, the word will be labelled as "missing".



The advanced puzzle has a similar solve method with a few notable differences. Instead of making use of an array, a double linked list is used. Each letter is its own object that has multiple variables associated with it. Importantly, the value of the letter in that position is stored, as well as the x and y position in the grid. Furthermore, two pointers to other grid cells are stored (left and right). When visualised, the list will appear as a long line of objects, all hold together by links. Each object has no knowledge of any part of the list other than the two objects sitting directly next to it. In the simple structure, I mentioned that, when the first letter is found, all surrounding letters are checked. This could simply be done by looking at the array. For example, if I wanted to check the position above I would simply keep the x position the same and subtract the y position by 1. However, with a linked list, this wouldn't be possible. If we know the dimensions of the theoretical grid, we can use that as a guide to determine where we need to go. For example, if a letter was found in position 0, 1 in a 9x9 grid and we wanted to check the letter above it, we would have to move left 9 times. From here, the same method of solving the grid is used.



As seen in the timing results, the simple structure was faster when it came to both populating the grid and solving the grid. Furthermore, less grid cells were visited in the simple results. The population time difference is relatively small though compared to a substantial difference in solve time. This is because of the difference in how the grid is traversed in each of the structures. When a letter is found in the grid and surrounding letters are checked, the advanced structure has to call a lot more methods than the simple structure and jump around to a lot more locations in memory whereas the 2d array has functionality to directly reference any point of itself without the need of all this extra code. Furthermore, the reason it took slightly longer to populate the grid is because it is creating multiple objects with multiple different types stored inside of them, as well as establishing pointer links for each of them.



PARASOFT. C/C++test® Report

User configuration: 500083 rule set
2022 05 04T18:29:56+01:00

STATIC ANALYSIS

Project Name	Tasks			Files		Lines	
	suppressed	total	per 10,000 lines	checked	total	checked	total
ACW_WordSearch	0	0	0	5	5	770	770
Total [0:00:11]	**0**	**0**	**0**	**5**	**5**	**770**	**770**

©Parasoft Corp. - C++test 2021.1.0.20210504B1389 Reporting System