

500083 Lab Book

Week 1 – Lab A

Date: 4th Feb 2021

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>

int main(int, char**) {
    std::cout << "Hello World" << std::endl;
    return 0;
}
```

Test data:

n/a

Sample output:

n/a

Reflection:

This is programming 101

Metadata:

Hello World

Further information:

argv contains any parameters that were passed to the program.

argc holds the number of parameters passed to the program when it was started.

Q2. Console Window

Question:

Create a new Empty C++ Console project called Temperature and write a program to input a Fahrenheit measurement, convert it and output a Celsius value.

Solution:

```
cout << "Enter a temperature in fahrenheit" << endl;
double fahrenheit;
cin >> fahrenheit;

const double celsius = 5.0 / 9.0 * (fahrenheit - 32.0);

cout << fahrenheit << " degrees fahrenheit is " << celsius << " degrees celsius" << endl;

return 0;
```

Test data:

100 degrees F

Sample output:

37.7778 degrees C

Reflection:

C++ doesn't like to divide integers

Metadata:

Temperature conversion

Further information:

n/a

Q3. Types

Question:

Write a program that prints out the size in bytes of each of the fundamental data types in C++.

Solution:

```
cout << "double: " << sizeof(double) << " bytes" << endl;
cout << "float: " << sizeof(float) << " bytes" << endl;
cout << "int: " << sizeof(int) << " bytes" << endl;
cout << "signed int: " << sizeof(signed int) << " bytes" << endl;
cout << "unsigned int: " << sizeof(unsigned int) << " bytes" << endl;
cout << "boolean: " << sizeof(bool) << " bytes" << endl;
cout << "char: " << sizeof(char) << " bytes" << endl;
cout << "signed char: " << sizeof(signed char) << " bytes" << endl;
cout << "unsigned char: " << sizeof(unsigned char) << " bytes" << endl;
```

Test data:

n/a

Sample output:

double: 8 bytes

float: 4 bytes

int: 4 bytes

signed int: 4 bytes

unsigned int: 4 bytes

boolean: 1 byte

char: 1 byte

signed char: 1 byte

unsigned char: 1 byte

Reflection:

There is not a difference in byte size of signed and unsigned data types

Metadata:

Data type sizes

Further information:

n/a

Q4. Floating point precision

Question:

Create a program to test the precision of floating-point numbers

Solution:

[illegible]

Test data:

Used lots of different numbers and calculations as test data

Sample output:

X and Y are identical

Y and z are identical

Reflection:

You need to have a really small number to have a divide by zero error

It feels like I was supposed to run into more errors than I did

Metadata:

n/a

Further information:

Commented out lines are used for other tests or questions

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

Question:

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

Solution:

```
int factorialNumber = 5;
int factorialTotal = 1;

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

cout << factorialTotal << endl;
```

Test data:

n/a

Sample output:

120

Reflection:

Only really had to change how the factorialTotal is displayed onto the console

Metadata:

n/a

Further information:

n/a

Q6. Calculate 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

Solution:

```
int n = 0;
int count = 0;
double total = 0;
do
{
    cout << "Please enter an int value, then press Enter" << endl;
    cin >> n;

    if (cin.fail() || n < 0)
    {
        break;
    }

    total += n;
    count++;
} while (true);

double average = total / count;

cout << "Average: " << average << endl;

return 0;
```

Test data:

1 1 1 1

5 5

12 3

Sample output:

1

5

7.5

Reflection:

If cin can't assign the input, due to non-matching type or other reasons, I can use cin.fail() to check

Metadata:

While loop

Further information:

n/a

Week 2 – Lab B

Date: 11th Feb 2021

Q1. The Good >> Streaming Operator

Question:

Add 'using namespace std;', a third variable that is a string type & add code to take a string value from the user after you take the float value then output the string value to the console after you output the float value.

Solution:

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

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

Test data:

1 1.1 hello

23 4.586 Hello

Sample output:

i=1, f=1.1, j=hello

i=23, f=4.586, j=Hello

Reflection:

If I want to use a string in need to include it

Warren mentioned its good practice to use std:: instead of using namespace std

Metadata:

Standard library & string

Further information:

n/a

Q2. The Bad >> Streaming Operator

Question:

Place a break point on the cout line of the code. Run the above program. In the console, enter the values 123 before selecting the return key. Run the program again, but this time enter the values 123456789 before selecting the return key.

Solution:

```
char c[5];  
//cin >> c;  
cin.get(c, 5);  
cout << "c=" << c << endl;
```

Test data:

123

123456789

Sample output:

n/a

Reflection:

cin.get() is better for storing multiple characters in an array than cin >>

Metadata:

Storing in arrays

Further information:

n/a

Q3. Assembly Language

Question:

Compile and run the program. Then place a breakpoint on line 5. Use F10 to step through the program and watch the local variables window. Do the same but with F11. Disassemble the program. View the registers and step through the program make notes of the changes. Open the memory window and copy the address of the stack from the stack pointer within the Register window then paste it into the edit field. Make note of the changes. Recompile and run in release mode.

Solution:

```

Viewing Options
--- C:\Users\DBate\Documents\UniWork\602854\500083\500083-lab-b-DanielJBates\Assembly\Source.cpp
#include <iostream>
using namespace std;

int main(int, char**) {
00007FF72AC52490 mov     qword ptr [rsp+10h],rdx
00007FF72AC52495 mov     dword ptr [rsp+8],ecx
00007FF72AC52499 push    rbp
00007FF72AC5249A push    rdi
00007FF72AC5249B sub     rsp,168h
00007FF72AC524A2 lea     rbp,[rsp+20h]
00007FF72AC524A7 mov     rdi,rsr
00007FF72AC524AA mov     ecx,5Ah
00007FF72AC524AF mov     eax,0CCCCCCCCh
00007FF72AC524B4 rep     stos     dword ptr [rdi]
00007FF72AC524B6 mov     ecx,dword ptr [rsp+188h]
00007FF72AC524BD lea     rcx,[_79FCFEDF_Source@cpp (07FF72AC63029h)]
00007FF72AC524C4 call    __CheckForDebuggerJustMyCode (07FF72AC513E3h)
    const int start = 3;
00007FF72AC524C9 mov     dword ptr [start],3
    const int end = 10;
00007FF72AC524D0 mov     dword ptr [end],0Ah
    int total = 0;
00007FF72AC524D7 mov     dword ptr [total],0

    int count = start;
00007FF72AC524DE mov     dword ptr [count],3
    while (count < end) {
00007FF72AC524E5 cmp     dword ptr [count],0Ah
00007FF72AC524E9 jge     main+72h (07FF72AC52502h)
        total += count;
00007FF72AC524EB mov     eax,dword ptr [count]
00007FF72AC524EE mov     ecx,dword ptr [total]
00007FF72AC524F1 add     ecx,eax
00007FF72AC524F3 mov     eax,ecx
00007FF72AC524F5 mov     dword ptr [total],eax
        count++;
00007FF72AC524F8 mov     eax,dword ptr [count]
00007FF72AC524FB inc     eax
00007FF72AC524FE jmp     main+72h (07FF72AC52502h)
    }
}

```

Registers

RAX = 0000000000000001 RBX = 0000000000000000 RCX = 00007FF72AC63029 RDX = 000002AD0B539E90
RSI = 0000000000000000 RDI = 000000AFEC53FC68 R8 = 000002AD0B535DE0 R9 = 000000AFEC53FB78
R10 = 0000000000000012 R11 = 000000AFEC53FC20 R12 = 0000000000000000 R13 = 0000000000000000
R14 = 0000000000000000 R15 = 0000000000000000 RIP = 00007FF72AC524C9 RSP = 000000AFEC53FB00
RBP = 000000AFEC53FB20 EFL = 00000200

0x000000AFEC53FB24 = CCCCCCCC

100 %

```

Viewing Options
00007FF7CEB40FFC add     byte ptr [rax],al
00007FF7CEB40FFE add     byte ptr [rax],al
--- C:\Users\DBate\Documents\UniWork\602854\500083\500083-lab-b-DanielJBates\Assembly\Source.cpp
#include <iostream>
using namespace std;

int main(int, char**) {
00007FF7CEB41000 sub     rsp,28h
    const int start = 3;
    const int end = 10;
    int total = 0;

    int count = start;
    while (count < end) {
        total += count;
        count++;
    }

    cout << "Total= " << total << endl;
00007FF7CEB41004 mov     rcx,qword ptr [__imp_std::cout (07FF7CEB430D0h)]
00007FF7CEB41008 call    std::operator<<<std::char_traits<char> > (07FF7CEB410B0h)
00007FF7CEB41010 mov     rcx,rax
00007FF7CEB41013 mov     edx,2Ah
00007FF7CEB41018 call    qword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::operator<< (07FF7CEB43080h)]
00007FF7CEB4101E mov     rcx,rax
00007FF7CEB41021 lea     rdx,[std::endl<char,std::char_traits<char> > (07FF7CEB41270h)]
00007FF7CEB41028 call    qword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::operator<< (07FF7CEB43088h)]

    return 0;
00007FF7CEB4102E xor     eax,eax
}
00007FF7CEB41030 add     rsp,28h
00007FF7CEB41034 ret
--- No source file ---
00007FF7CEB41035 int     3
00007FF7CEB41036 int     3
00007FF7CEB41037 int     3
00007FF7CEB41038 int     3
00007FF7CEB41039 int     3
00%
```

Test data:

n/a

Sample output:

n/a

Reflection:

This is a bit daunting and I don't really understand this but the code has been optimised.

Metadata:

Assembly language

Further information:

I'm disappointed there was not a question called the ugly streaming operator.

Week 3 – Lab C

Date: 15th Feb 2021

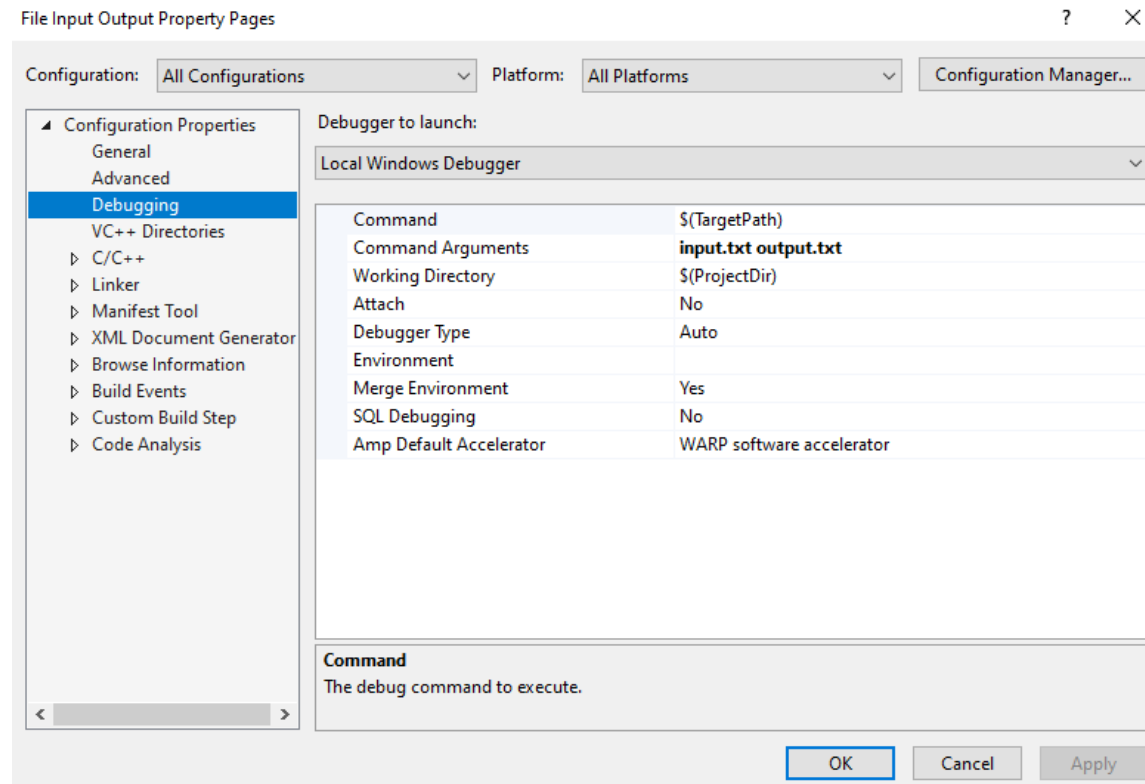
Q1. Passing Command Arguments

Question:

Enter input.txt output.txt into the Command Arguments.

Solution:

Project > Properties > Debugging > Command arguments



Test data:

n/a

Sample output:

n/a

Reflection:

Argv[1] will now contain input.txt

Argv[2] will now contain output.txt

Metadata:

Command arguments

Further information:

n/a

Q2. Copying a text file

Question:

Complete the functionality inside of the Copy(char filenamein[], char filenameout[]) function.

Solution:

```
bool Copy(char filenamein[], char filenameout[])
{
    ifstream fin(filenamein);
    ofstream fout(filenameout);
    if (fin.is_open())
    {
        char ch[40];
        fin.read(ch, 40);
        fin.close();

        fout.write(ch, fin.gcount());
        fout.close();
        return true;
    }
    else
    {
        cout << "Open file error" << endl;
        return false;
    }
}
```

Test data:

n/a

Sample output:

Copy Successful

Reflection:

This is pretty much the same as in c#.

I can use fin.get() and a while loop to work with any size file

Metadata:

Copy

Further information:

n/a

Q3. Function call mechanism

Question:

Compile and run the program. Place a breakpoint on line 14 and disassemble the code, then open the registers. Step through and execute the following line; **00413204 call mymax (040124Eh)**. Take note of the changes to memory location. Continue and execute; **0041322B call std::operator<<<std::char_traits<char> > (0401663h)** and look how the command prompt executes the function. Now execute the line after; **0041320C mov dword ptr [max],eax** make note of the EAX register, copy and paste this into the memory address box and then in the memory window scroll up 4 lines. Execute the next 2 line and make note of the changes. Now modify the code and add in your own function that takes 3 parameters each of a different type.

Solution:

```
int a = 10;
003F25F8 mov     dword ptr [a],0Ah
int b = 20;
003F25FF mov     dword ptr [b],14h

int max = mymax(a, b);
003F2606 mov     eax,dword ptr [b]
003F2609 push     eax
003F260A mov     ecx,dword ptr [a]
003F260D push     ecx
003F260E call     mymax (03F11D6h)  ≤1ms elapsed
003F2613 add     esp,8
003F2616 mov     dword ptr [max],eax
```

Viewing Options

```
std::basic_ostream<char,std::char_traits<char> >::_Sentry_base::_Sentry_base:
003F11BD jmp     std::basic_ostream<char,std::char_traits<char> >::_Sentry_base::_Sentry_base (03F1F10h)
__QueryPerformanceCounter@4:
003F11C2 jmp     __QueryPerformanceCounter@4 (03F5E67h)
__srt_uninitialize_crt:
003F11C7 jmp     __srt_uninitialize_crt (03F42E0h)
_RTC_AllocFailure:
003F11CC jmp     _RTC_AllocFailure (03F2FE0h)  ▶
_MultiByteToWideChar@24:
003F11D1 jmp     _MultiByteToWideChar@24 (03F5E3Dh)
mymax:
▶ 003F11D6 jmp     mymax (03F2550h)  ≤1ms elapsed
__acrt_thread_detach:
003F11DB jmp     __srt_stub_for_acrt_thread_detach (03F5EE0h)
_RTC_SetErrorFunc:
003F11E0 jmp     _RTC_SetErrorFunc (03F3BF0h)
_RTC_Terminate:
003F11E5 jmp     _RTC_Terminate (03F4B00h)
_GetProcAddress@8:
003F11EA jmp     _GetProcAddress@8 (03F5EAFh)
__srt_is_ucrt_dll_in_use:
003F11EF jmp     __srt_is_ucrt_dll_in_use (03F5CF0h)
__isa_available_init:
003F11F4 jmp     __isa_available_init (03F58F0h)
__acrt_uninitialize_critical:
003F11F9 jmp     __srt_stub_for_acrt_uninitialize_critical (03F5F00h)
__report_securityfailureEx:
003F11FE jmp     __report_securityfailureEx (03F3A30h)
_VirtualQuery@12:
003F1203 jmp     _VirtualQuery@12 (03F5EA3h)
```

100 %

Reflection:

If the solutions platform is set to x64 instead of x86 then the assembly will look a lot different.

```
int a = 10;
00007FF6647E2549 mov     dword ptr [a],0Ah
int b = 20;
00007FF6647E2550 mov     dword ptr [b],14h

int max = mymax(a, b);
00007FF6647E2557 mov     edx,dword ptr [b]
00007FF6647E255A mov     ecx,dword ptr [a]
00007FF6647E255D call    mymax (07FF6647E11FEh)
00007FF6647E2562 mov     dword ptr [max],eax
```

Viewing Options

```
_get_startup_commit_mode:
00007FF6647E11DB jmp     _get_startup_commit_mode (07FF6647E4A90h)
GetCurrentProcessId:
00007FF6647E11E0 jmp     GetCurrentProcessId (07FF6647E665Ch)
__acrt_thread_attach:
00007FF6647E11E5 jmp     __scrt_stub_for_acrt_thread_attach (07FF6647E66C0h)
__initterm_e:
00007FF6647E11EA jmp     __initterm_e (07FF6647E6572h)
FreeLibrary:
00007FF6647E11EF jmp     FreeLibrary (07FF6647E6698h)
at_quick_exit:
00007FF6647E11F4 jmp     at_quick_exit (07FF6647E4840h)
__should_initialize_environment:
00007FF6647E11F9 jmp     __should_initialize_environment (07FF6647E4B10h)
mymax:
00007FF6647E11FE jmp     mymax (07FF6647E2480h)    ≤1ms elapsed
__security_check_cookie:
00007FF6647E1203 jmp     __security_check_cookie (07FF6647E2AB0h)
_crt_at_quick_exit:
00007FF6647E1208 jmp     _crt_at_quick_exit (07FF6647E65EAh)
_RTC_Failure:
00007FF6647E120D jmp     _RTC_Failure (07FF6647E31B0h)
HeapFree:
00007FF6647E1212 jmp     HeapFree (07FF6647E6686h)
HeapAlloc:
00007FF6647E1217 jmp     HeapAlloc (07FF6647E6680h)
std::basic_ostream<char,std::char_traits<char> >::operator<<:
00007FF6647E121C jmp     std::basic_ostream<char,std::char_traits<char> >::operator<< (07FF6647E266Eh)
__castguard_slow_path_check_fastfail:
00007FF6647E1221 jmp     castguard_slow_path_check_fastfail (07FF6647E5350h)
```

Metadata:

Functions in assembly language

Further information:

n/a

Week 4 – Lab D

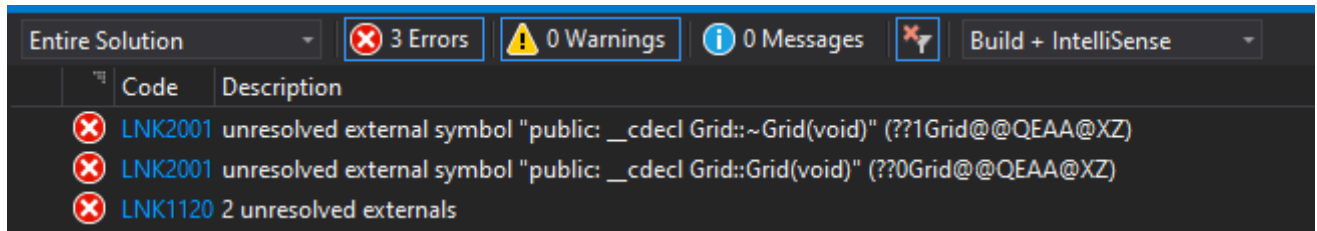
Date: 22/02/2021

Q1. Linker errors

Question:

Create a class named Grid and add in the provided code. Now add a new item this can be called Main.cpp and add the provided code. Compile the code and take a look at the linker errors. Add the provided code to fix these errors.

Solution:



Test data:

n/a

Sample output:

n/a

Reflection:

A .h needs to have a prototype for the constructor and destructor. The class .cpp file needs to contain the source code for a constructor and destructor.

Metadata:

Linkers

Further information:

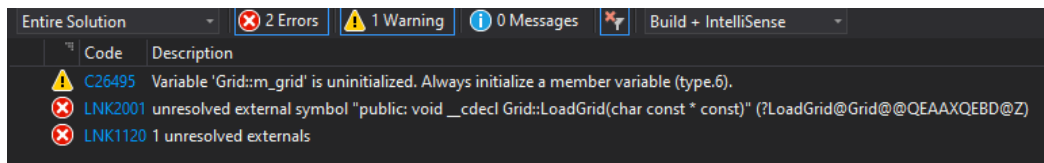
n/a

Q2. Reading into grid class

Question:

Remove the comments on the line containing the LoadGrid method and try to compile the code.

Solution:



```
void Grid::LoadGrid(const char filename[])
{
    std::ifstream fin(filename);
    if (fin.is_open())
    {
        for(int y = 0; y < 9; y++)
        {
            for (int x = 0; x < 9; x++)
            {
                fin >> m_grid[x][y];
            }
        }
    }
}
```

Test data:

Grid1.txt

Sample output:

m_grid is the same as the text file

Reflection:

Its pretty similar to c#

I need to find out how I can get the length of a 2d array in c++

Metadata:

Reading into a grid

Further information:

n/a

Q3. Saving the grid

Question:

Implement the SaveGrid method. Do not overwrite Grid1.txt

Solution:

```
void Grid::SaveGrid(const char filename[])
{
    std::ofstream fout(filename);

    for (int y = 0; y < 9; y++)
    {
        if (y != 0)
        {
            fout << std::endl;
        }
        for (int x = 0; x < 9; x++)
        {
            fout << " " << m_grid[x][y];
        }
    }
}
```

Test data:

m_grid

Sample output:

OutGrid.txt is the same as m_grid

Reflection:

This is pretty much the same as copying just using fout instead of fin.

endl must be done before the second loop if its not the first time going through to get the correct number of lines in the .txt

Metadata:

Save a grid

Further information:

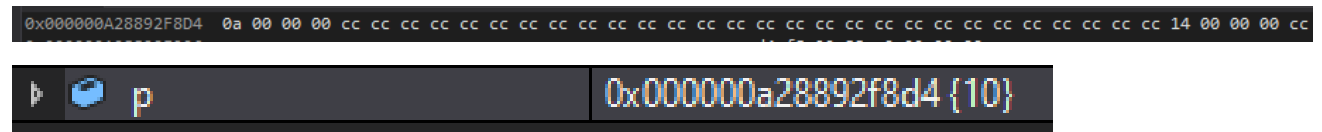
n/a

Q4. Pointers - Basics

Question:

In FunctionA set a to 100 using the pointer. Set a breakpoint on "int a = 10"

Solution:



Test data:

Sample output:

Reflection:

*p is set the a hexadecimal value which is the memory address of a

Metadata:

Pointers

Further information:

n/a

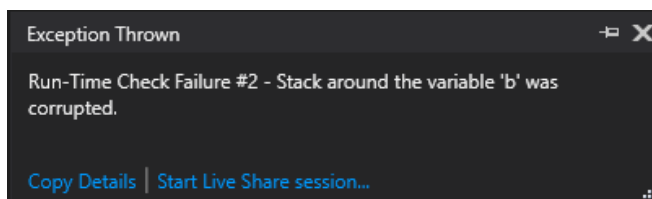
Q5. Pointers – False assumptions

Question:

Comment out FunctionA and uncomment FunctionB. Compile and run then observe the results.
After the line `*p = 100;` add `p++; *p = 200;`.

Solution:

```
a= 10  
b= 20  
c= 30  
a= 10  
b= 100  
c= 30
```



Test data:

n/a

Sample output:

n/a

Reflection:

Just because we list variables sequentially in our programme, does not guarantee that the compiler places them contiguously in memory.

Arrays can be used to guarantee memory layout

Metadata:

Pointer errors

Further information:

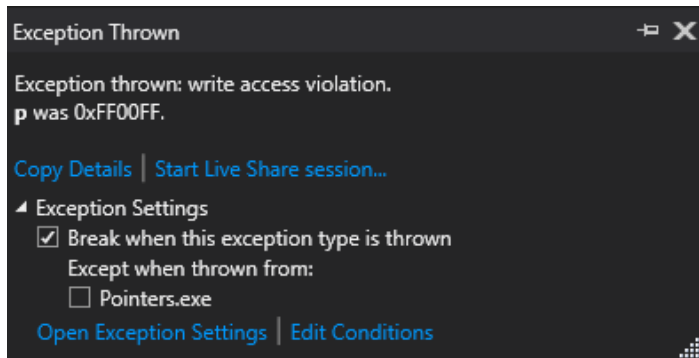
n/a

Q6. Pointers – The crash

Question:

Comment out FunctionB and uncomment FunctionC. Compile and run. Set a breakpoint and find the reason for the crash.

Solution:



Test data:

n/a

Sample output:

n/a

Reflection:

Windows prevents applications from accessing memory outside of its permitted memory footprint

Metadata:

Crash

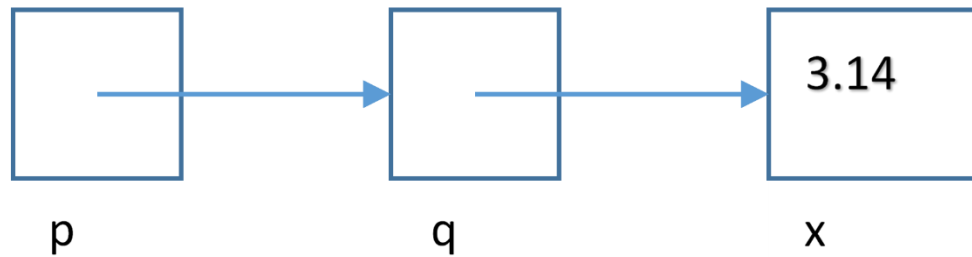
Further information:

n/a

Q7. Pointers – Pointers to pointers

Question:

Comment out FunctionC and uncomment FunctioD. Recreate the pointer chain in the diagram provided.



Solution:

```
void functionD() {  
    double x = 3.14;  
  
    cout << "x= " << x << endl;  
  
    double* q = &x;  
    double** p = &q;  
    **p = 4.2069;  
  
    cout << "x= " << x << endl;  
}
```

Test data:

n/a

Sample output:

X = 3.14

X = 4.2069

Reflection:

If you are going to point to a pointer you need to use **

Metadata:

Pointer chain

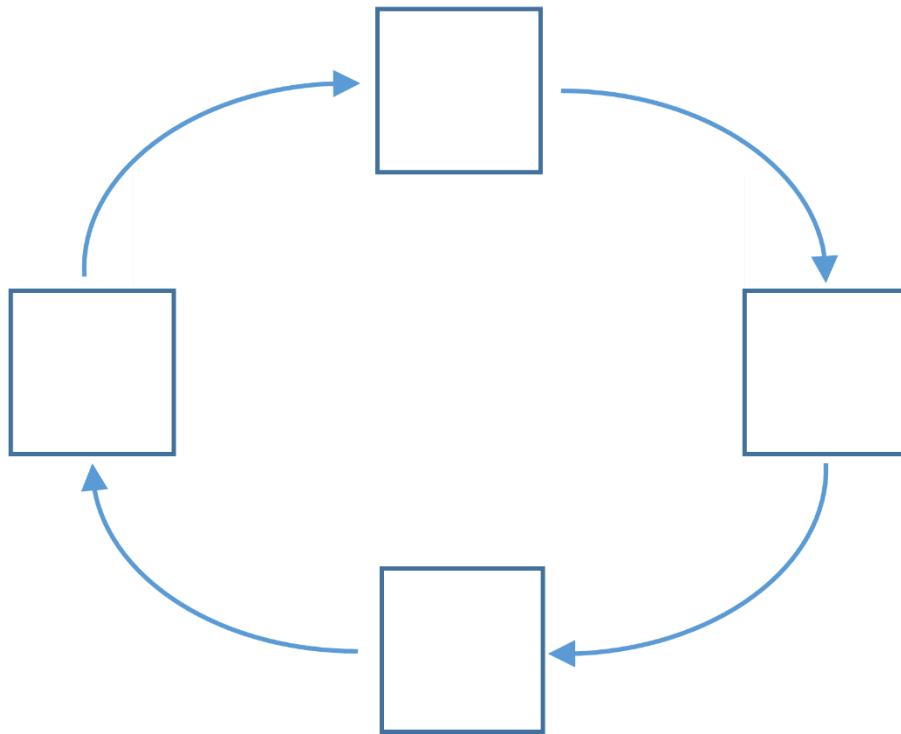
Further information:

n/a

Q8. Pointers – Pointer chain (Optional)

Question:

Implement a pointer chain loop like the diagram provided



Solution:

```
Double ***c;
```

```
Double ****z = &c;
```

```
Double *p = (double *)&z;
```

```
Double **q = &p;
```

```
C=&q;
```

```
Void *
```

Test data:

Sample output:

Reflection:

This would be pointless. It would be like a while(true) loop will no break

Metadata:

Pointer chain loop

Further information:

n/a

Week 5 – Lab E

Date: 01/03/2021

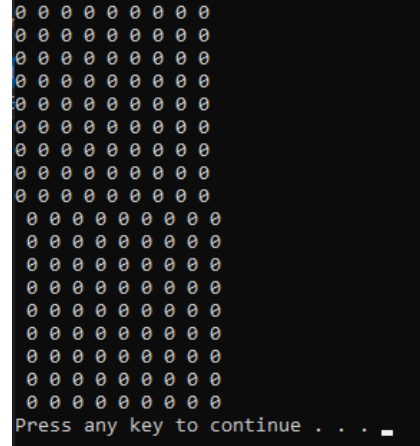
Q1. Operators in Grid

Question:

Extend your code from Q2 and Q3 in Lab D. add the functionality to write the Grid to an ostream using the auxiliary operator<< and the functionality to read in the values from an istream into the Grid using the auxiliary operator>>

Solution:

```
0
7 void Grid::LoadGrid(std::istream& in)
8 {
9     for (int y = 0; y < 9; y++)
10     {
11         for (int x = 0; x < 9; x++)
12         {
13             in >> m_grid[x][y];
14         }
15     }
16 }
17
18 void Grid::SaveGrid(std::ostream& out)
19 {
20     for (int y = 0; y < 9; y++)
21     {
22         if (y != 0)
23         {
24             out << std::endl;
25         }
26         for (int x = 0; x < 9; x++)
27         {
28             out << " " << m_grid[x][y];
29         }
30         out << std::endl;
31     }
32 }
33
34 void operator<<(std::ostream& out, Grid& grid)
35 {
36     grid.SaveGrid(out);
37 }
38
39 void operator>>(std::istream& in, Grid& grid)
40 {
41     grid.LoadGrid(in);
42 }
43
```




```
cin >> grid;  
cout << grid;
```

```
#pragma once  
#include<iostream>  
class Grid  
{  
public:  
    Grid();  
    ~Grid();  
  
    void LoadGrid(const char filename[]);  
    void SaveGrid(const char filename[]);  
    void LoadGrid(std::istream& in);  
    void SaveGrid(std::ostream& out);  
  
private:  
    int m_grid[9][9];  
};  
  
void operator<<(std::ostream& out, Grid& grid);  
void operator>>(std::istream& in, Grid& grid);
```

Test data:

```
000000000  
000000000  
000000000  
000000000  
000000000  
000000000  
000000000  
000000000  
000000000
```

Sample output:

```
000000000  
000000000  
000000000  
000000000  
000000000  
000000000  
000000000  
000000000  
000000000
```

Reflection:

This is simpler than I was thinking it to be. I have to adapt the class functions to accommodate the new streaming operator.

Metadata:

Streaming operators

Further information:

n/a

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:

```

Fraction Fraction::Add(const Fraction& f) const
{
    int resultNum = (this->Num() * f.Den()) + (f.Num() * this->Den());
    int resultDen = (this->Den()) * f.Den();

    Fraction result(resultNum, resultDen);
    return result;
}

Fraction Fraction::Subtract(const Fraction& f) const
{
    int resultNum = (this->Num() * f.Den()) - (f.Num() * this->Den());
    int resultDen = this->Den() * f.Den();

    Fraction result(resultNum, resultDen);
    return result;
}

Fraction Fraction::Multiply(int i) const
{
    int resultNum = this->Num() * i;

    Fraction result(resultNum, this->Den());
    return result;
}

Fraction Fraction::Divide(int i) const
{
    int resultNum = this->Num() / i;
    int resultDen = this->Den() / i;

    Fraction result(resultNum, resultDen);
    return result;
}

```

```

#pragma once
#include<iostream>
class Fraction
{
public:
    Fraction();
    Fraction(int num, int den);

    int Num() const;
    int Den() const;

    Fraction Add(const Fraction& f) const;
    Fraction Subtract(const Fraction& f) const;
    Fraction Multiply(int i) const;
    Fraction Divide(int i) const;

    void Num(int num);
    void Den(int den);

    void Write(std::ostream& sout) const;
    void Read(std::istream& sin);
private:
    int m_num;
    int m_den;
};

```

```
1/2 + 3/4 = 10/8  
3/4 - 1/2 = 2/8  
3/4 * 3 = 9/4  
1 2  
Read = 1/2  
Press any key to continue . . .
```

Test data:

1 2

Sample output:

Read = 1/2

Reflection:

This is pretty simple as its fairly similar to C#. I just need to remember that I need to use things like 'Fraction::' in the .cpp

Metadata:

Fraction class

Further information:

n/a

Q3. Operators in Fraction

Question:

Implement a class member operator operator+, class member operator operator-, class member operator operator* and a auxiliary operator operator*.

Solution:

```
1/2 + 3/4 = 10/8
3/4 - 1/2 = 2/8
3/4 * 3 = 9/4
3 * 3/4 = 9/4
1 2
Read = 1/2
Press any key to continue . . .
```

```
Fraction Fraction::operator+(Fraction f) const
{
    Fraction result = this->Add(f);
    return result;
}

Fraction Fraction::operator-(Fraction f) const
{
    Fraction result = this->Subtract(f);
    return result;
}

Fraction Fraction::operator*(int i) const
{
    Fraction result = this->Multiply(i);
    return result;
}

Fraction operator* (int i, Fraction& f)
{
    Fraction result = f.Multiply(i);
    return result;
}
```

Test data:

1 2

Sample output:

Read = 1/2

Reflection:

This is pretty straight forward although in a scenario like this I'm not sure why you would even bother if the class member operators, auxiliary seems so much better.

Metadata:

Operators in fractions

Further information:

n/a

Q4. Parameters




Question:

The goal is to swap the value of a and b. First place a breakpoint on 'myswap(a,b);' then run and open the disassembly, registers and memory. On memory set it to look at the stack. Step through the program and watch how the values of a and b are pushed onto the stack. Rewrite the previous code to pass the variables a and b into the myswap function by reference, rather than by value. Repeat the debugging process from the previous section. Find the address of a in the appropriate register, and use the address to look at the memory that holds the value for the variable a.

Solution:

[illegible]

Name	Value	Type
lhs	10	int
rhs	20	int
temp	5181500	int

Name	Value	Type
 lhs	20	int
 rhs	10	int
 temp	10	int

Test data:

n/a

Sample output:

n/a

Reflection:

I understand the normal programming of this but I still don't really get assembly language. This is just instead of making a copy of the passed in variable in the function it is just using the original variable.







Metadata:

Passing by reference

Further information:

n/a

Solution:

Name	Value	Type
 _formal	1	int
 _formal	0x00877da0 {0x00877da8 "C:\\Users\\DBate\\Documents\\UniWork\\60...	char **
 result1	30	int
 result2	15	int
 value1	10	int
 value2	20	int

Test data:

n/a

Sample output:

n/a

Reflection:

The function making a copy of the value to return. I understand the logic and code of it but the assembly language has me lost.

Metadata:

Return by value

Further information:

n/a

Week 6 – Lab F

Date: 08/03/2021

Q1. Template grid

Question:

Turn the Grid class into a template class.

Solution:

```
#pragma once
#include<iostream>
#include<fstream>
template<class T>
class Grid
{
public:
    Grid(void)
    {
    }
    ~Grid(void)
    {
    }

    void LoadGrid(const char filename[]) { ... }
    void SaveGrid(const char filename[]) { ... }

    void LoadGrid(std::istream& in) { ... }
    void SaveGrid(std::ostream& out) { ... }

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

template<class T>
void operator<<(std::ostream& out, Grid<T>& grid)
{
    grid.SaveGrid(out);
}

template<class T>
void operator>>(std::istream& in, Grid<T>& grid)
{
    grid.LoadGrid(in);
}
```

Test data:

n/a

Sample output:

n/a

Reflection:

I can see how this could be useful so you wouldn't need to make different functions for all the different types. E.g. int, float, double

Metadata:

Template

Further information:

n/a

Q2. Template grid (floats)

Question:

Change the code in main() so that you can store float values.

Solution:

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

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

    system("pause");
}
```

```
m_grid 0x00b7fcf8 {0x00b7fcf8 {1.00000000, 2.00000000, 3.00000000, 4.00000000, 5.00000000, 6.00000000, 7.00000000, ...}, ...}
[0] 0x00b7fcf8 {1.00000000, 2.00000000, 3.00000000, 4.00000000, 5.00000000, 6.00000000, 7.00000000, 8.00000000, ...}
[1] 0x00b7fd1c {2.00000000, 3.00000000, 4.00000000, 5.00000000, 6.00000000, 7.00000000, 8.00000000, 9.00000000, ...}
[2] 0x00b7fd40 {3.00000000, 4.00000000, 5.00000000, 6.00000000, 7.00000000, 8.00000000, 9.00000000, 1.00000000, ...}
[3] 0x00b7fd64 {4.00000000, 5.00000000, 6.00000000, 7.00000000, 8.00000000, 9.00000000, 1.00000000, 2.00000000, ...}
[4] 0x00b7fd88 {5.00000000, 6.00000000, 7.00000000, 8.00000000, 9.00000000, 1.00000000, 2.00000000, 3.00000000, ...}
[5] 0x00b7fdac {6.00000000, 7.00000000, 8.00000000, 9.00000000, 1.00000000, 2.00000000, 3.00000000, 4.00000000, ...}
[6] 0x00b7fdd0 {7.00000000, 8.00000000, 9.00000000, 1.00000000, 2.00000000, 3.00000000, 4.00000000, 5.00000000, ...}
[7] 0x00b7fdd4 {8.00000000, 9.00000000, 1.00000000, 2.00000000, 3.00000000, 4.00000000, 5.00000000, 6.00000000, ...}
[8] 0x00b7fe18 {9.00000000, 1.00000000, 2.00000000, 3.00000000, 4.00000000, 5.00000000, 6.00000000, 7.00000000, ...}
```

```
1 2 3 4 5 6 7 8 9.1
2 3 4 5 6 7 8 9.1 1
3 4 5 6 7 8 9.1 1 2
4 5 6 7 8 9.1 1 2 3
5 6 7 8 9.1 1 2 3 4
6 7 8 9.1 1 2 3 4 5
7 8 9.1 1 2 3 4 5 6
8 9.1 1 2 3 4 5 6 7
9.1 1 2 3 4 5 6 7 8
Press any key to continue . . .
```

Test data:

Changed all the 9's in Grid1.txt to 9.1

Sample output:

```
1 2 3 4 5 6 7 8 9.1
2 3 4 5 6 7 8 9.1 1
3 4 5 6 7 8 9.1 1 2
4 5 6 7 8 9.1 1 2 3
5 6 7 8 9.1 1 2 3 4
6 7 8 9.1 1 2 3 4 5
7 8 9.1 1 2 3 4 5 6
8 9.1 1 2 3 4 5 6 7
9.1 1 2 3 4 5 6 7 8
```

Reflection:

Operator<< will convert a float such as 1.0 to an int implicitly because the .0 isn't necessary.

Metadata:

Grid<float>

Further information:

n/a

Q3. Binary search

Question:

implement two versions of a binary search, one using iteration, the other using a recursive function.

Solution:

```
int binarySearch(int size, int pCandidate, int pArray[])
{
    int low = 0;
    int high = size;

    while (low <= high)
    {
        int middle = (high + low) / 2;

        if (pCandidate == pArray[middle])
        {
            return middle;
        }
        else if (pCandidate < pArray[middle])
        {
            high = middle - 1;
        }
        else
        {
            low = middle + 1;
        }
    }
    return -1;
}
```

```
bool binarySearch(int* list, int size, int value)
{
    const int middle = (size) / 2;
    if (value == list[middle])
    {
        return true;
    }
    if (middle != 1)
    {
        if (value < list[middle])
        {
            return binarySearch(list + 1, size - middle, value);
        }
        else if (value > list[middle])
        {
            return binarySearch(list + middle + 1, size - (middle + 1), value);
        }
    }
    return false;
}
```

Test data:

11

10

69

65

Sample output:

Is in array

Isn't in array

Is in array

Isn't in array

Reflection:

I definitely prefer the iterative approach it is a lot easier to read. However, in terms of design intent the recursive way seems better as it passes the array in using a pointer.

An array as a parameter implicitly gets converted to a pointer. So, `int[]` is the same as `int*`

Metadata:

Binary Search

Further information:

n/a

Week 7 – Lab G

Date: 15/03/2021

Q1. Parasoft

Question:

Set up and test Parasoft. Then correct the problems that cause the Severity 3 errors

Solution:

10 test results

- [2] >Severity 1 - Highest
- [7] >Severity 3 - Medium
- [1] >Severity 4 - Low

C/C++test Report

User configuration: 500083 rule set
2021-03-18T12:32:05+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 [0:03:39]	0	10	1666	3	3	60	60	

All Tasks by Severityby: [Category](#) Severity

[2] **Severity 1 - Highest**

[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)

[7] **Severity 3 - Medium**

[4] Declare parameters or local variable as const whenever possible (CODSTA-CPP-53-3)

[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] **Severity 4 - Low**

[1] Avoid unused local variables (OPT-02-4)

Tasks by AuthorBack to Top

Author	Tasks		
	suppressed	Total	recommended
DBate	0	10	10

©Parasoft Corp. - C++test 2020.1.0.20200422B856 Reporting System

7 test results

- +... [1] >Severity 1 - Highest
- +... [4] >Severity 2 - High
- +... [2] >Severity 4 - Low

C/C++test[®] Report

User configuration: 500083 rule set
 2021-03-23T15:47:27+00:00

STATIC ANALYSIS

Project Name	Tasks			Files		Lines	
	suppressed	total	per 10,000 lines	checked	total	checked	total
Parasoft	0	7	1060	3	3	66	66
Total [0:00:06]	0	7	1060	3	3	66	66

All Tasks by Category
by: Category Severity

[2] **Initialization (INIT)**

[1] All member variables should be initialized in constructor (INIT-06-1)

[1] An assignment operator shall assign all data members (INIT-11-2)

[1] **Memory and Resource Management (MRM)**

[1] A copy constructor shall copy all data members and bases (MRM-41-2)

[1] **Object Oriented (OOP)**

[1] Check for assignment to self in operator= (OOP-34-4)

[3] **Optimization (OPT)**

[2] Eliminate unused parameters (OPT-03-2)

[1] Avoid unused local variables (OPT-02-4)

Tasks by Author
Back to Top

Author	Tasks		
	suppressed	Total	recommended
Daniel Bates	0	7	7

©Parasoft Corp. - C++test 2020.1.0.20200422B856 Reporting System

Line 27 in Utility.cpp had the const keyword added

In main.cpp int argc was changed to const int argc

Assignment and copy operators were added to Utility.h

Test data:

n/a

Sample output:

n/a

Reflection:

This is so useful for picking up on things that can be easy to miss. It will also provide you with a document containing all the errors which will be useful to have open next to the code while I'm fixing bugs. I'll be using this a lot

Metadata:

Parasoft

Further information:

n/a

Week 8 – Lab H

Date: 22/03/2021

Q1. PersonNode

Question:

Add member methods to PersonNode so that the name and age can be changed and returned. make the AddressBookSLL class a friend of the PersonNode class

Solution:

```
#pragma once

#include <string>
using namespace std;

class PersonNode
{
    friend AddressBookSLL;

public:
    PersonNode(void);
    PersonNode(const string& name, int age);
    ~PersonNode(void);

private:
    string m_name;
    int m_age;
    PersonNode* m_next;

    int getAge();
    void setAge(int pNewAge);
    string getName();
    void setName(string pNewName);
};

PersonNode::getAge()
{
    return m_age;
}

PersonNode::setAge(int pNewAge)
{
    m_age = pNewAge;
}

PersonNode::getName()
{
    return m_name;
}

PersonNode::setName(string pNewName)
{
    m_name = pNewName;
}
```

Test data:

n/a

Sample output:

n/a

Reflection:

Getters and setters are the same as in C#, I'm not sure if there is a better way I could do it though. Declaring a friend class was simpler than I thought it would be.

Metadata:

PersonNode

Further information:

n/a

Q2. AddressBookSLL

Question:

implement the functionality to add a new person's using the following method prototype:

```
void AddPerson(const string& name, int age);
```

Solution:

```
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* temp = m_head->m_next;
        while (temp->m_next != nullptr)
        {
            temp = temp->m_next;
        }
        temp->m_next = new PersonNode(name, age);
    }
}
```

Test data:

```
book.AddPerson("Darren", 21);
```

```
book.AddPerson("Dawn", 42);
```

```
book.AddPerson("Steven", 18);
```

```
book.AddPerson("Sue", 27);
```

Sample output:

The new person was added to the Address Book

Reflection:

-> is used to access member variables of a object pointer. Nullptr can be used to check if a pointer has been initialised to anything.

Metadata:

AddressBookSLL

Further information:

n/a

Q3. Find, Delete, Output

Question:

Add a 'FindPerson' method and a 'DeletePerson' method, as well as, the functionality to output every entry of the address book

Solution:

```
const PersonNode* AddressBookSLL::FindPerson(const string& name) const
{
    PersonNode* tempPointer = m_head;

    while (tempPointer != nullptr)
    {
        if (tempPointer->getName() == name)
        {
            return tempPointer;
        }
        else
        {
            tempPointer = tempPointer->m_next;
        }
    }

    return tempPointer;
}

bool AddressBookSLL::DeletePerson(const string& name)
{
    PersonNode* current = m_head;

    if (current != nullptr && current->getName() == name)
    {
        PersonNode* del = current;
        m_head = current->m_next;
        delete del;
        return true;
    }

    while (current != nullptr)
    {
        if (current->m_next->getName() == name)
        {
            PersonNode* del = current->m_next;
            current->m_next = del->m_next;
            delete del;
            return true;
        }
        else
        {
            current = current->m_next;
        }
    }

    return false;
}

void AddressBookSLL::OutputBook()
{
    PersonNode* current = m_head;

    while (current != nullptr)
    {
        std::cout << current->getName() << ' ' << current->getAge() << std::endl;

        current = current->m_next;
    }

    std::cout << std::endl;
}
```

Test data:

Call the methods created

Sample output:

They all work as intended

Reflection:

While loops seem to be the best for navigating a Linked List. I don't see the point in having the getters and setters in the PersonNode class if the AddressBookSLL is just going to be a friend of the class anyway

Metadata:

Navigating a Linked List

Further information:

n/a

Week 9 – Lab I

Date: 29/03/2021

In my implementation of the task, I first created two new classes that would act as the data structures for my nodes and arcs.

The node class contained seven key member variables, a string called 'mPlaceName', an int called 'mReferenceCode', a vector of type arc pointer called 'mLinks' and four doubles, 'mLatitude', 'mLongitude', 'mX' and 'mY'. These variables are all named in a way that is clear what data is stored within it. However, for 'mLinks' it should be noted that this variable will only contain arcs that go out from the node, it is a part of and does not contain the arcs coming into the node. The class also contains some methods, with the exception of one, these are all just getter methods. The exception is the 'AddLink' method this was created to avoid having to call the 'GetLinks' method and then calling 'push_back' on what was returned from 'GetLinks'. I have no real critiques for how I've done this class, except I would probably prefer to store the doubles as floats to improve memory management. However, the LLtoUTM function provided requires that the variables be doubles.

```
#pragma once

#include<string>
#include<vector>

class Arc;

class Node
{
public:
    Node(const std::string pPlaceName, const int pReferenceCode, const double pLatitude, const double pLongitude);
    ~Node();
    int GetReferenceCode();
    double GetX();
    double GetY();
    std::vector<Arc*> GetLinks();
    std::string GetName();
    void AddLink(Arc* pLink);
private:
    std::string mPlaceName;
    int mReferenceCode;
    double mLatitude, mLongitude, mX, mY;
    std::vector<Arc*> mLinks;
};
```

The arc class contained four key member variables, a string called 'mTransportMode', a double called 'mDistance' and two node pointers, 'mNodeA' and 'mNodeB'. Again these variables are named in a way that makes them clear, 'mNodeA' and 'mNodeB' could be clearer as this doesn't clearly convey that 'mNodeA' is the node that this arc starts from and that 'mNodeB' is where this arc ends. The class also contains some methods but these are all just getter methods. The main thing I would do differently is instead of having 'mTransportMode' as a string I could make it an enumerator to reduce the chance of error, I wasn't sure how I couldn't implement this successfully when I read from files so I decided to keep it simple with just a string.

```
#pragma once

#include<string>

class Node;

class Arc
{
public:
    Arc(Node* pNodeA, Node* pNodeB, const std::string pTransportMode);
    ~Arc();
    Node* GetNode(std::string pRequestesNode);
    std::string GetTransportMode();
    double GetDistance();

private:
    Node* mNodeA;
    Node* mNodeB;
    std::string mTransportMode;
    double mDistance;
};
```

With these two classes created I needed to add to the member variables of the provided navigation class. I only need to add a new vector or type node pointer called 'mPlaces' in order to store all the nodes created in the 'BuildNetwork' function.

BuildNetwork

This function reads the given Places.csv and Links.csv and creates nodes and arcs based off this data. So, this function has two core parts to it. First the creation on the nodes, this starts with reading the file until a comma is found, this indicates the end of a piece of data, this data is stored in the appropriate member variable. This is done four times for the node to get the name of the place, its reference code, its latitude and longitude, the data is collected in this order. Once this is done the node is created, then a pointer referencing this node is created and stored in the 'mPlaces' vector. This process is repeated until the function can no longer retrieve a new line from the csv file, this indicates that the file has been completely read and so arcs can now start being created.

```
bool Navigation::BuildNetwork(const std::string& fileNamePlaces, const std::string& fileNameLinks)
{
    std::fstream finPlaces(fileNamePlaces);
    std::fstream finLinks(fileNameLinks);

    if (finPlaces.fail() || finLinks.fail()) return false;

    if (finPlaces.is_open())
    {
        char line[255];

        while (finPlaces.getline(line, 255, ','))
        {
            std::string placeName;
            int referenceCode;
            float latitude, longitude;

            placeName = std::string(line);

            finPlaces.getline(line, 255, ',');
            std::istringstream inReferenceCode(line);
            inReferenceCode >> referenceCode;

            finPlaces.getline(line, 255, ',');
            std::istringstream inLatitude(line);
            inLatitude >> latitude;

            finPlaces.getline(line, 255);
            std::istringstream inLongitude(line);
            inLongitude >> longitude;

            Node* const ptr = new Node(placeName, referenceCode, latitude, longitude);
            Navigation::mPlaces.push_back(ptr);
        }
        finPlaces.close();
    }
}
```

The arc half of the function is largely the same as the node half the key difference is that each line only contains three pieces of data and the creation of the arc differs slightly. The way the creation of the arc differs from the creation of a node is that using the data read from the csv file the function needs to create a node pointer that references the node with the same reference code as the one provided by the csv data, this is done using a function I will cover later in this lab book. This is done twice to create a pointer for what will become nodeA and nodeB. Now the function will have the data it needs to create a new arc, the arc is created and then an arc pointer is made and stored in the 'mLinks' member variable of the appropriate node. Then another arc is created that has the opposite direction, so nodeA becomes nodeB and vice-versa, then another arc pointer is made and stored in the appropriate nodes 'mLinks' member variable. I don't have any critiques on this function I don't think there is much to improve on in this.

```

if (finLinks.is_open())
{
    char line[255];

    while (finLinks.getline(line, 255, ','))
    {
        int referenceCodeA, referenceCodeB;
        std::string transportMode;

        std::istringstream inReferenceCodeStart(line);
        inReferenceCodeStart >> referenceCodeA;

        finLinks.getline(line, 255, ',');
        std::istringstream inReferenceCodeDestination(line);
        inReferenceCodeDestination >> referenceCodeB;

        finLinks.getline(line, 255);
        std::istringstream inTransport(line);
        inTransport >> transportMode;

        Node* tempNode = SearchByReferenceCode(referenceCodeA);
        Arc* ptr = new Arc(tempNode, SearchByReferenceCode(referenceCodeB), transportMode);
        tempNode->AddLink(ptr);

        tempNode = SearchByReferenceCode(referenceCodeB);
        ptr = new Arc(tempNode, SearchByReferenceCode(referenceCodeA), transportMode);
        tempNode->AddLink(ptr);
    }
    finLinks.close();
}

return true;
}

```

ProcessCommand

This function is very simple it just reads the provided command and calls the correct function that the command refers to as well as storing and giving any data needed to the function. For example, the FindDist function needs two reference codes that are provided in the command. Once the function has been completed a blank line is output to '_outFile' and true is returned. The only improvement I could make with this function would be to use a switch statement instead of a series of if statements. But this would require that the command be an enumerator instead of a string.


```

bool Navigation::ProcessCommand(const std::string& commandString)
{
    std::istringstream inString(commandString);
    std::string command;

    inString >> command;

    if (command == "MaxDist")
    {
        MaxDist();

        _outFile << std::endl;

        return true;
    }
    else if (command == "MaxLink")
    {
        MaxLink();

        _outFile << std::endl;

        return true;
    }
    else if (command == "FindDist")
    {
        int referenceCodeStart, referenceCodeEnd;

        inString >> referenceCodeStart >> referenceCodeEnd;

        FindDist(referenceCodeStart, referenceCodeEnd);

        _outFile << std::endl;

        return true;
    }
    else if (command == "FindNeighbour")
    {
        int referenceCode;

        inString >> referenceCode;

        FindNeighbour(referenceCode);

        _outFile << std::endl;

        return true;
    }
    else if (command == "Check")

```

```

else if (command == "Check")
{
    int currentNode, nextNode;
    std::string transportMethod;

    inString >> transportMethod;
    inString >> currentNode;

    do
    {
        inString >> nextNode;

        if (currentNode == nextNode || !Check(transportMethod, currentNode, nextNode))
        {
            break;
        }
        else
        {
            currentNode = nextNode;
        }
    } while (!inString.fail());

    _outFile << std::endl;

    return true;
}
else if (command == "FindRoute")
{
    std::string transportMode;
    int referenceCodeStart, referenceCodeEnd;

    inString >> transportMode >> referenceCodeStart >> referenceCodeEnd;

    FindRoute(transportMode, referenceCodeStart, referenceCodeEnd);

    _outFile << std::endl;

    return true;
}
else if (command == "FindShortestRoute")
{
    std::string transportMode;
    int referenceCodeStart, referenceCodeEnd;

    inString >> transportMode >> referenceCodeStart >> referenceCodeEnd;

    FindShortestRoute(transportMode, referenceCodeStart, referenceCodeEnd);

    _outFile << std::endl;

    return true;
}

return false;
}

```

MaxDist

This function gets every combination of places and uses the 'CalculateDistance' function I created to find out the straight-line distance between the two places. If that distance is then larger than the currently stored distance then that becomes the new distance and the two nodes that make up this distance are stored. Once every combination has been checked the largest distance and the name of the places that make up this distance are output to the '_outFile'. For this function I don't think there is any way I could really improve upon it.

```
void Navigation::MaxDist()
{
    Node* a = nullptr;
    Node* b = nullptr;
    double distance = 0;

    for (int i = 0; i < mPlaces.size(); i++)
    {
        for (int j = 0; j < mPlaces.size(); j++)
        {
            if (distance < CalculateDistance(mPlaces[i]->GetX(), mPlaces[i]->GetY(), mPlaces[j]->GetX(), mPlaces[j]->GetY()))
            {
                distance = CalculateDistance(mPlaces[i]->GetX(), mPlaces[i]->GetY(), mPlaces[j]->GetX(), mPlaces[j]->GetY());
                a = mPlaces[i];
                b = mPlaces[j];
            }
        }
    }

    _outFile << a->GetName() << ", " << b->GetName() << ", " << distance << std::endl;
}
```

MaxLink

This function looks at every places' links and checks the distance covered in the link. This is simply done by using the 'GetDistance' getter that is in the arc class and compares the distance to the current 'maxLink' distance. Once all links of all places have been checked the 'maxLink' has both nodes and the distance of the link output to the '_outFile'. Again, I don't think there is anything to improve on in this function.

```
void Navigation::MaxLink()
{
    Arc* maxLink = nullptr;

    for (int i = 0; i < mPlaces.size(); i++)
    {
        for (int j = 0; j < mPlaces[i]->GetLinks().size(); j++)
        {
            if (maxLink == nullptr)
            {
                maxLink = mPlaces[i]->GetLinks()[j];
            }
            else if (mPlaces[i]->GetLinks()[j]->GetDistance() > maxLink->GetDistance())
            {
                maxLink = mPlaces[i]->GetLinks()[j];
            }
        }
    }

    _outFile << maxLink->GetNode("A")->GetReferenceCode() << ", " << maxLink->GetNode("B")->GetReferenceCode() << ", " << maxLink->GetDistance() << std::endl;
}
```

FindDist

This function uses the 'CalculateDistance' function I created to get the distance between two given nodes. The name of the nodes and the distance is then output to the '_outFile'. The only ways I can think to improve this function would either be to pass in the node pointer instead of the reference code or perhaps create a function that allows me to get certain information about a node using only a reference code to avoid having to make a temporary node pointer.

```
void Navigation::FindDist(const int& pReferenceCodeStart, const int& pReferenceCodeEnd)
{
    Node* const startNode = SearchByReferenceCode(pReferenceCodeStart);
    Node* const endNode = SearchByReferenceCode(pReferenceCodeEnd);
    _outFile << startNode->GetName() << ", " << endNode->GetName() << ", " << CalculateDistance(startNode->GetX(), startNode->GetY(), endNode->GetX(), endNode->GetY()) << std::endl;
}
```

FindNeighbour

This function gets all the nodes that have links connecting them to the given node and then outputs the node's reference code to the '_outFile'. Again, I don't think I could improve on this.

```
void Navigation::FindNeighbour(const int& pReferenceCode)
{
    std::vector<Arc*> tempLinks = SearchByReferenceCode(pReferenceCode)->GetLinks();
    for (int i = 0; i < tempLinks.size(); i++)
    {
        _outFile << tempLinks[i]->GetNode("B")->GetReferenceCode() << std::endl;
    }
}
```

Check

This function checks to see if you can go from the first given node to the second given node using the chosen transport mode. This is done by looping through all the first given node's links and checking if the current link goes to the second given node. If it does then the 'IsTraversableBy' function is used to check if you can use the link with the chosen transport mode. If you can then the function returns true and the two reference codes followed by 'PASS' will be outputted to '_outFile' and the 'Check' function will return true. If the 'IsTraversableBy' method returned false the loop would continue. If by the end of the loop no viable link is found then the two reference codes followed by 'FAIL' is outputted to the '_outFile' and the function returns false. I don't think I could improve on this. I think using the 'IsTraversableBy' function was a good idea as knowing if I can travel down a link is very useful information to be able to retrieve. The code for this function is found on the following page.

```
bool Navigation::Check(const std::string& pTransportMode, const int& pReferenceCodeA, const int& pReferenceCodeB)
{
    Node* const currentNode = SearchByReferenceCode(pReferenceCodeA);
    for (int i = 0; i < currentNode->GetLinks().size(); i++)
    {
        Arc* const currentLink = currentNode->GetLinks()[i];
        if (currentLink->GetNode("B")->GetReferenceCode() == pReferenceCodeB)
        {
            if (IsTraversableBy(pTransportMode, currentLink->GetTransportMode()))
            {
                _outFile << pReferenceCodeA << ", " << pReferenceCodeB << ",PASS" << std::endl;
                return true;
            }
        }
    }
    _outFile << pReferenceCodeA << ", " << pReferenceCodeB << ",FAIL" << std::endl;
    return false;
}
```

FindRoute

This function finds the first valid route that can be found for a chosen transport mode. This is done by storing the given start node as the current node and the given end node as the destination. The straight line distance between these two nodes is then calculated and stored. The function then loops through the current nodes links and looks for a traversable link that brings you closer to the destination. If a link that can be taken and gets you closer to the destination is found, then the node that the link takes you to becomes the stored next and the loop is broken out of. The next node is then checked to make sure it's not a null pointer, if it is then that means no link was found so 'FAIL' is output to the '_outFile'. If the next pointer does have something stored in it then the current node reference code is stored into the route vector and the current becomes the next node and next becomes a null pointer again. Then the current is checked to see if its then same as the destination, if it is then the route is complete so the final node will be added to the route vector and a loop will be carried out to output all of the reference codes in the route to '_outFile'. If this is not the case though then the process will be repeated until no viable path can be found or a complete route is found. I'm sure that this function can probably be improved upon. But, I'm not sure how I would improve it, maybe with assembly language programming. I think the speeds I'm getting with this function are good its definitely less then I was expecting.

```
void Navigation::FindRoute(const std::string& pTransportMode, const int& pReferenceCodeStart, const int& pReferenceCodeEnd)
{
    Node* current = SearchByReferenceCode(pReferenceCodeStart);
    Node* next = nullptr;
    Node* const destination = SearchByReferenceCode(pReferenceCodeEnd);
    double distanceLeft = CalculateDistance(current->GetX(), current->GetY(), destination->GetX(), destination->GetY());
    std::vector<int> route;

    do
    {
        for (int i = 0; i < current->GetLinks().size(); i++)
        {
            Arc* const currentLink = current->GetLinks()[i];
            double const distanceFromNext = CalculateDistance(currentLink->GetNode("B")->GetX(), currentLink->GetNode("B")->GetY(), destination->GetX(), destination->GetY());
            if (IsTraversableBy(pTransportMode, currentLink->GetTransportMode()) && distanceLeft > distanceFromNext)
            {
                next = currentLink->GetNode("B");
                distanceLeft = distanceFromNext;
                break;
            }
        }

        if (next == nullptr)
        {
            _outFile << "FAIL" << std::endl;
            break;
        }

        route.push_back(current->GetReferenceCode());
        current = next;
        next = nullptr;

        if (current == destination)
        {
            route.push_back(current->GetReferenceCode());
            for (int i = 0; i < route.size(); i++)
            {
                _outFile << route[i] << std::endl;
            }
        }
    } while (current != destination);
}
```

FindShortestRoute

This function works largely the same as the 'FindRoute' function. The only difference is that instead of the function breaking out of the loop as soon as a viable link is found the loop will continue. The loop will also now check to see how much distance is covered in each viable link, if the current link that is being checked covers a greater distance than the link that currently covers the greatest distance then this new link will be used instead. If not though the link that is currently found to cover the greatest distance will still be used. Again, I'm sure this can be improved upon I'm just not sure how I would do it.

```

void Navigation::FindShortestRoute(const std::string& pTransportMode, const int& pReferenceCodeStart, const int& pReferenceCodeEnd)
{
    Node* current = SearchByReferenceCode(pReferenceCodeStart);
    Node* next = nullptr;
    Node* const destination = SearchByReferenceCode(pReferenceCodeEnd);
    double distanceLeft = CalculateDistance(current->GetX(), current->GetY(), destination->GetX(), destination->GetY());
    std::vector<int> route;

    do
    {
        for (int i = 0; i < current->GetLinks().size(); i++)
        {
            Arc* const currentLink = current->GetLinks()[i];
            if (IsTraversableBy(pTransportMode, currentLink->GetTransportMode()))
            {
                const double distanceFromNext = CalculateDistance(currentLink->GetNode("B")->GetX(), currentLink->GetNode("B")->GetY(), destination->GetX(), destination->GetY());
                if (distanceLeft > distanceFromNext)
                {
                    next = currentLink->GetNode("B");
                    distanceLeft = distanceFromNext;
                }
            }
        }

        if (next == nullptr)
        {
            _outFile << "FAIL" << std::endl;
            break;
        }

        route.push_back(current->GetReferenceCode());

        current = next;
        next = nullptr;

        if (current == destination)
        {
            route.push_back(current->GetReferenceCode());

            for (int i = 0; i < route.size(); i++)
            {
                _outFile << route[i] << std::endl;
            }
        }
    } while (current != destination);
}

```

IsTraversableBy

This function checks to see if you can travel a link based on the link's transport mode and the transport mode you are travelling by. This is done using if, else if and nested if statements to check. If your chosen transport mode is by foot then true will be returned by the function as you can travel anywhere by foot. If your transport mode is bike then unless the link's transport mode is by foot then you can travel down it. If your transport mode is car then unless the link's transport mode is by foot, bike or rail then you can travel down it. If your transport mode is bus then the link's transport mode must be by bus or ship if not then you can't travel down it. Finally, if your transport mode is either ship or rail then the link's transport mode must be the same as yours to travel down it. I think this function could be greatly improved by using a switch statement but, as I mentioned early, this would require 'mTransportMode' to be an enumerator which I decided against.

```

bool Navigation::IsTraversableBy(const std::string& pTransportMethod, const std::string& pLinkTransportMode)
{
    if (pTransportMethod == "Foot")
    {
        return true;
    }
    else if (pTransportMethod == "Bike")
    {
        if (pLinkTransportMode != "Foot")
        {
            return true;
        }
    }
    else if (pTransportMethod == "Car")
    {
        if (pLinkTransportMode != "Foot" || pLinkTransportMode != "Bike" || pLinkTransportMode != "Rail")
        {
            return true;
        }
    }
    else if (pTransportMethod == "Bus")
    {
        if (pLinkTransportMode == "Bus" || pLinkTransportMode == "Ship")
        {
            return true;
        }
    }
    else if (pTransportMethod == "Ship" || pTransportMethod == "Rail")
    {
        if (pLinkTransportMode == pTransportMethod)
        {
            return true;
        }
    }
    return false;
}

```

SearchByReferenceCode

This function loops through the 'mPlaces' variable and checks if the current node's reference code matches the passed in reference code, if it does then the node pointer is returned. If a matching node is never found then a null pointer is returned. The only improvement I would make on this function is what happens if a matching reference code can't be found, I think returning a null pointer isn't the best way I could have done this.

```

Node* Navigation::SearchByReferenceCode(const int& pReferenceCode)
{
    for (int i = 0; i < mPlaces.size(); i++)
    {
        if (mPlaces[i]->GetReferenceCode() == pReferenceCode)
        {
            return mPlaces[i];
        }
    }
    return nullptr;
}

```

CalculateDistance

This function uses Pythagoras to calculate the straight line distance of two given nodes and then returns that distance. This can't really be improved as it's a set formula.

```
double Navigation::CalculateDistance(const double pNodeA_X, const double pNodeA_Y, const double pNodeB_X, const double pNodeB_Y)
{
    const double distance = sqrt((((pNodeA_X - pNodeB_X) * (pNodeA_X - pNodeB_X)) + ((pNodeA_Y - pNodeB_Y) * (pNodeA_Y - pNodeB_Y))));
    return distance;
}
```

Command	Debug – x86 (µs)	Release – x86 (µs)	Debug – x64 (µs)	Release – x64 (µs)
BuildNetwork	25585.2	3223.2	24157.8	2745.0
MaxDist	8374.0	166.8	8409.2	214.8
MaxLink	3002.2	171.4	2520.8	159.6
FindDist	74.6	34.2	70.8	31.4
FindNeighbour	85.8	43.8	98.2	43.6
Check (PASS)	122.2	37.4	121.2	61.4
Check (FAIL)	64.8	21.6	80.0	29.0
FindRoute (PASS)	129.4	40.4	133.2	29.4
FindRoute (FAIL)	78.0	24.8	69.8	23.4
FindShortestRoute (PASS)	128.4	33.2	133.4	39.2
FindShortestRoute (FAIL)	64.2	18.0	70.2	24.0

*Average taken over 5 runs of the program using the data set provided in the ACW template