Chris Culling

Lab report for assignment #1

4 September 2024

Table of Contents

Command Line Compilation

Exercise 1

After creating a new terminal profile in VS Code which runs Developer Command Prompt—which was a bit tricky and took some time—and navigating to the working directory, compiling the `hello.cpp` file with the `cl /EHsc hello.cpp` command worked without issues.

The files `hello.obj` and `hello.exe` were created.

Entering `hello` as a lone argument in the terminal ran the executable, and "Hello Wolrd!" was printed to the same terminal. (There is a spelling error in the print! Let's change that and compile again.)

Using `cl /EHsc hello.cpp` again modified the `hello.obj` and `hello.exe` files. The executable now prints "Hello World!". The spelling correction came through!

<Description of the problem (briefly) and how you solved it. You are encouraged to reflect on your thought process, decision-making, possible failings etc.>

The most challenging and time-consuming part of the task was the set-up and refamiliarising with the terminal. I thought it was strange that other terminals (powershell, command prompt, etc.) were unable to recognise and run the installed C++ compiler.

Command Prompt Record

```
**********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.7.5
** Copyright (c) 2022 Microsoft Corporation
**********************************************************************

C:\Program Files\Microsoft Visual Studio\2022\Community>cd
C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello

C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>cl /EHsc hello.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.37.32825 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.
```

```
hello.cpp
Microsoft (R) Incremental Linker Version 14.37.32825.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:hello.exe
hello.obj

C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>hello
Hello Wolrd!

C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>cl /EHsc hello.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.37.32825 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

hello.cpp
Microsoft (R) Incremental Linker Version 14.37.32825.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:hello.exe
hello.obj

C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>hello
Hello World!
```

Figure 1

Exercise 2

When compiling with separate linking, the `hello.obj` file is compiled but no `.exe` file is generated. Only when we use the link command (`link /out:hello.exe hello.obj`) does the `.exe` file generate.

Command Prompt Record

```
C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>cl /EHsc /c hello.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.37.32825 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

hello.cpp
```

```
C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>link /out:hello.exe
hello.obj
Microsoft (R) Incremental Linker Version 14.37.32825.0
Copyright (C) Microsoft Corporation.  All rights reserved.
```

Figure 2

Exercise 3

I modified the code and compiled just as I had done before. The program now prints “Hello World! “ followed by all input words, including the `hello` argument used to launch the program in the first place! You can easily skip this argument by starting the for loop at `int i = 1` instead of `int i = 0`.

In my case, running the program with the arguments `hello This is a TEST` printed `Hello World! hello This is a TEST` to the console.

Code

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

{

    std::cout << "Hello World! ";



    for (int i = 0; i < argc; i++)

    {

        std::cout << argv[i];

        std::cout << " ";

    }



    return 0;

}
```

Figure 3

Command Prompt Record

```
C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>cl /EHsc hello.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.37.32825 for x86
Copyright (C) Microsoft Corporation.  All rights reserved.

hello.cpp
Microsoft (R) Incremental Linker Version 14.37.32825.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:hello.exe
hello.obj

C:\Users\chris\source\repos\lab1_handout\lab1_handout\hello>hello This is a TEST
Hello World! hello This is a TEST
```

Figure 4

Sum Calculator

Exercise 4

I began by testing the `cout` and `cin` commands to make sure they work as I expect.

Then I tested `while(std::cin >> value)` to grasp how precisely it operates. I was surprised to learn that you can enter any number of arguments any number of times, as demonstrated in the Command Prompt Record below (Figure 5). I expected to be able to either enter an arbitrary amount of arguments in a single command line, or to enter any amount of arguments one command line at a time, but not both combined.) I had to adjust my code due to this unexpected behaviour in order to make the console outputs more readable.

The logic and programming portions were otherwise trivial.

Command Prompt Record

Inputs are highlighted

```
C:\Users\chris\source\repos\lab1_handout\lab1_handout\sum>sum
8 3 7
+8
+3
+7
1 2
+1
+2
5
+5
^X
= 26
```

Figure 5

Exercise 5

At first I thought the assignment was asking for a program that dumped the output to a file by default. Once I realised to just add arguments `sum > output.txt`, the task completed successfully and as expected.

Command Prompt Record

```
C:\Users\chris\source\repos\lab1_handout\lab1_handout\sum>sum >> output.txt
9 1
100
5
^X
```

Figure 6

`output.txt` content

```
+9
+1
+100
+5
= 115
```

Figure 7

Exercise 6

The outputs were shown, as usual, in the console. After exercise 5, this exercise was straight forward with no funny business.

Command Prompt Record

```
C:\Users\chris\source\repos\lab1_handout\lab1_handout\sum>sum < terms.txt
+3
+6
+9
+4
+20
= 42
```

Figure 8

`terms.txt` content

```
3 6 9 4 20
```

Figure 9

Exercise 7

Console outputted nothing. The text files appear correct. Command: `(sum < terms.txt) > output.txt`

`output.txt` content

```
+3
+6
+9
+4
+20
= 42
```

Figure 10

`terms.txt` content

```
3 6 9 4 20
```

Figure 11

Root Finder

Exercise 8

Programming the calculations and the implementations of the methods were trivial and very similar to other programming languages.

The bigger challenge was re-learning how to instruct the program to get values from a file and how to input different values on the fly (in the same runtime) so I didn't have to recompile every test. Once this obstacle was overcome, however, things got easier.

Exercise 9

Firstly, I separated the logic in `main` into designated methods (`printRoots` and `evaluate`) to clean it all up and make it more flexible to work with.

I decided to make a `getRoots` method in the `Poly2` class that returns a tuple containing the roots. ~~I hate this decision. I hate it. I cannot get it to work.~~ I did not quite understand what or how to implement *by-reference* arguments. ~~I hope that I did.~~

After thinking for hours that the compilation failures were due to the implementation of the tuples, it dawned on me that the new methods in `poly2.cpp` were not defined in the header file. After an additional journey through hell, I got the impression that methods defined in the header need to belong to a class—and I'm not sure if `main()` belongs to a default class or how that works—so I made a new class called ~~`Printer`~~ `Program` to stuff my new methods in and made an object of the class in the beginning of my program. This worked!

There was also the issue of identifying how many real roots any given polynomial had. I made a cheeky solution by adding a constant `rootless = 3.1451`—almost equal to pi, but not quite!—and had my `getRoots` method return that in the place of `root1` and/or `root2` whenever appropriate. This obviously fails if the roots actually are `3.1451`. I did this because I was not sure if or how to best check if the roots were `null` or `-ndef` or whatever other funky value C++ decides it should have. This will need exploring in the future.

Editor's note: The tuple-solution eventually worked, but did not fulfil the by-reference requirement. After grasping by-reference and turning `getRoots` into a `void` method, I realised that this solution is much simpler, and this technique will be incredibly useful to bring on in my future C++ endeavours. Instead of returning `3.1451` for non-real roots, they are simply set to `3.1451` by default before running `getRoots` and remain unchanged if no real roots were found.

Exercise 10

Wrapping the program in a `while(cin << a << b << c)` did the trick. Nothing strange here.

Exercise 11

The program obeyed the task of reading and writing to files with no changes at this point. I made sure to add an `if(a == 0) return (-c)/b;` clause to take care of the dividing by zero issue as motivated by Figure 12.

$$0 = 0 \cdot x^2 + b \cdot x + c \implies x = \frac{-c}{b}$$

Figure 12

`coeffs.txt`

```
1 2 1
2 -1 -1
1 1 1
1 2 -2
```

Figure 13

roots.txt

```
--------------------------------------------------
Enter three coefficients for a polynomial, one coefficient at a time.
Alternatively, enter '0' for all coefficients to toggle evaluation for y(x) (given an inputted x) after routine.
--------------------------------------------------
--------------------------------------------------
Coefficients selected:
a = 1, b = 2, c = 1
Root-finding started...
Found one real root: -1
--------------------------------------------------
Enter three coefficients for a polynomial, one coefficient at a time.
Alternatively, enter '0' for all coefficients to enable evaluation for y(x) (given an inputted x) after routine.
--------------------------------------------------
--------------------------------------------------
Coefficients selected:
a = 2, b = -1, c = -1
Root-finding started...
Found two real roots: 1, -0.5
--------------------------------------------------
Enter three coefficients for a polynomial, one coefficient at a time.
Alternatively, enter '0' for all coefficients to enable evaluation for y(x) (given an inputted x) after routine.
--------------------------------------------------
--------------------------------------------------
Coefficients selected:
a = 1, b = 1, c = 1
Root-finding started...
Found no real roots.
--------------------------------------------------
Enter three coefficients for a polynomial, one coefficient at a time.
Alternatively, enter '0' for all coefficients to enable evaluation for y(x) (given an inputted x) after routine.
--------------------------------------------------
```

```
--------------------------------------------------
Coefficients selected:
a = 1, b = 2, c = -2
Root-finding started...
Found two real roots: 0.732051, -2.73205
--------------------------------------------------
Enter three coefficients for a polynomial, one coefficient at a time.
Alternatively, enter '0' for all coefficients to enable evaluation for y(x) (given an
inputted x) after routine.
--------------------------------------------------
```

Figure 14