Laboration 1

Uppgift 1

Filerna ”hello.exe” och ”hello.obj” skapades.

Vid exekvering skrivs ”Hello Wolrd!” ut i konsollen där filen exekverades. (Felstavat)

Ändrade felet med kommandot ”notepad hello.cpp”, redigerade felet och exekverade kommandot ”cl /EHsc hello.cpp”. Då skrevs ”Hello World!” ut.

Uppgift 2

Kommandot ”cl /EHsc /c hello.cpp” skapar filen ”hello.obj”

Kommandot ” cl /EHsc /Fe:hello.exe hello.cpp” skapar filen ”hello.exe”

Uppgift 3

Kod:

```
std::cout << "Hello World! Nice to see you, ";
for (int i = 1; i < argc; i++)
{
    std::cout << argv[i];

    if (i == argc - 1) //Place space or exclamation mark if end of scentence.
    {
        std::cout << "!";
    }
    else
    {
        std::cout << " ";
    }
}
return 0;
```

Körexempel:

```
C:\dev\git\DA378A\lab1\hello>hello Oscar Strandmark
Hello World! Nice to see you, Oscar Strandmark!
```

Uppgift 4

Kod:

```
short val;
short sum = 0;
std::cout << "Enter numbers to add." << std::endl;
while (std::cin >> val)
{
    sum = sum + val;
}
std::cout << sum << std::endl;
```

Körexempel:

```
C:\dev\git\DA378A\lab1\sum>sum
Enter numbers to add.
1
2
3
4
^Z
10
```

Uppgift 5

Input:

```
C:\dev\git\DA378A\lab1\sum>sum > sum.txt
1
2
3
4
^Z
```

Sum.txt:

```
Enter numbers to add.
10
```

Allting som skrivs i output-strömmen hamnar i textfilen istället.

Uppgift 6

Cmd:

```
C:\dev\git\DA378A\lab1\sum>sum < terms.txt
Enter numbers to add.
15
```

Terms.txt:

```
1 2 3 4 5
```

Outputströmmen hamnar i kommandotolken. Den innehåller allting som skrivs av utströmmen.

Uppgift 7

CMD:

```
C:\dev\git\DA378A\lab1\sum>(sum < terms.txt) > sum.txt
```

Sum.txt efter exekvering:

```
Enter numbers to add.
15
```

Uppgift 8

```
Poly2 polyA = Poly2(1, 2, 1);
Poly2 polyB = Poly2(2, -1, -1);
Poly2 polyC = Poly2(1, 1, 1);

std::cout << polyA.eval(-1) << std::endl;
std::cout << polyB.eval(4) << std::endl;
std::cout << polyC.eval(4) << std::endl;

polyA.findRoots();
polyB.findRoots();
polyC.findRoots();
```

```
C:\dev\git\DA378A\lab1\poly>poly2
0
27
21
One real root:
-1

Two real roots:
1
-0.5

2 imaginary roots

C:\dev\git\DA378A\lab1\poly>
```

Använde exempel-polynomerna som fanns i uppgiften.

Uppgift 9

Totala exekveringsresultat kan ses till höger i bilden, vänstra sidan visar hur print-satserna ser ut. Det till höger repeterades för varje polynom.

```
std::cout << "Roots found: " << rootCount << std::endl;
if (rootCount > 0)
{
    if (rootCount == 1) {
        std::cout << rootA << std::endl;
        std::cout << polyC.eval(rootA) << std::endl;
    }
    if (rootCount == 2) {
        std::cout << rootA << std::endl;
        std::cout << rootB << std::endl;
        std::cout << polyC.eval(rootA) << std::endl;
        std::cout << polyC.eval(rootB) << std::endl;
    }
}
```

```
C:\dev\git\DA378A\lab1\poly>poly2
0
27
21
Roots found: 1
-1
0
Roots found: 2
1
-0.5
0
0
Roots found: 0

C:\dev\git\DA378A\lab1\poly>
```

Om man matar in en rot i eval() bör 0 returneras om roten är korrekt.

Exempel:

$$Polynomet\ y(x) = x^2 + 2x + 1\ har\ en\ rot.\ Denna\ roten\ \text{är}\ -1.\ eval(x) = y(x)$$

$$-1^2 - 2 + 1 = 0$$

Uppgift 10

Kände att ett y/n input var mer intressant sätt men kan fixa om det inte tillåts.

```
float coef1;
float coef2;
float coef3;

while (1)
{
    std::cout << "Mata in 3 tal." << std::endl;
    //Read coeffs
    std::cin >> coef1;
    std::cin >> coef2;
    std::cin >> coef3;

    //Calc roots
    Poly2 polynom = Poly2(coef1, coef2, coef3);
    int rootCount;
    float rootA;
    float rootB;
    polynom.findRoots(rootCount, rootA, rootB);
    std::cout << "Roots found: " << rootCount << std::endl;
    if (rootCount > 0)
    {
        if (rootCount == 1) {
            std::cout << rootA << std::endl;
            std::cout << std::endl;
        }
        if (rootCount == 2) {
            std::cout << rootA << std::endl;
            std::cout << rootB << std::endl;
            std::cout << std::endl;
        }
    }

    //Again?
    char c;
    std::cout << "Vill du forts. skapa polynom? y/n" << std::endl;
    std::cin >> c;
    if (c != 'y') {
        break;
    }
}
```

```
Developer Command Prompt for VS 2019

C:\dev\git\DA378A\lab1\poly>poly2
Mata in 3 tal.
1
2
1
Roots found: 1
-1

Vill du forts. skapa polynom? y/n
y
Mata in 3 tal.
2
-1
-1
Roots found: 2
1
-0.5

Vill du forts. skapa polynom? y/n
y
Mata in 3 tal.
1
1
1
Roots found: 0
Vill du forts. skapa polynom? y/n
n

C:\dev\git\DA378A\lab1\poly>
```

Uppgift 11

Detta är innehållet av roots.txt efter exekvering →

Man hade kunnat hantera singulariteten genom använda en formel för att hitta var den linjära funktionen korsar x-axeln istället när $a = 0$.

$$y = mx + b$$

$$y - b = mx$$

$$\frac{y - b}{m} = x$$

Den formeln borde fungera för att hitta var $y(x) = 0$

Om a = 0 är polynomet av första graden och är en linjär funktion. (mx+b)

roots.txt* poly2.cpp

```
Root-finding started...
Found polynomial coeffs:
1
2
1

Roots found:
-1
Eval:
0
Found polynomial coeffs:
2
-1
-1

Roots found:
1
-0.5
Eval:
0
0
Found polynomial coeffs:
1
1
1

No roots found.
Found polynomial coeffs:
1
2
-2

Roots found:
0.732051
-2.73205
Eval:
-1.19209e-07
4.76837e-07
```

Uppgift 13 (Inte den valfria, så egentligen 14)

Eftersom uppgiften endast säger man ska försöka förstå dem skriver jag bara mina anteckningar om dem här.

- Till för att organisera kompilering av kod.
- Definierar hur en process för att kompilera kod till en viss grupp källkodsfiler.
    - Definierar både hur den ska kompileras och hur den ska tas bort via clean.
- Innehåller typ konstanta variabler som kan peka på namn till filer/mappar

Uppgift 14 (Sista uppgiften, egentligen 15)

Antar att när man skriver ”nmake all” följer den anvisningarna efter ”all:” i makefilen. Man ser ut att kunna skapa macros/funktionsliknande grejer som kan innehålla flera kommandon i sig