Command-line compiling

Uppgift 1

- “Cl /EHsc hello.cpp” genererar två nya filer ”hello.exe” och ”hello.obj”.
- Exekverar man hello (hello.exe), så skrivs ut ”Hello Wolrd!”.
- World skrivs fel, nämligen Wolrd

Uppgift 2

Såhär gjorde jag:

1. ”cl /c hello.cpp”, där /c kompilerar utan att länka

En fil, ”hello.obj” genereras

2. ”link hello.obj”, där vi länkar en exekverbar fil till hello objektet

En fil, ”hello.exe” generas

Uppgift 3

”argc” represents the count of strings in ”argv”, which we add 1 and hope over it as it is the name of the file “hello Hadi Saghir”. However, we can just name it argc (argument count) and argv (argument vector): int main (int argc, char* argv).

Kalkylator

Uppgift 4

```
#include <iostream>

int main(){

int val, sum;

sum = 0;

while(std::cin >> val){

sum += val;

            }

std::cout << sum;
```

}

Uppgift 5

sum > sum.txt (overwrite)

sum >> sum.txt (append)

Uppgift 6

utskrift hamnar på konsolen. Det innehåller resultat, I guess.

Uppgift 7

(sum < terms.txt) > sum.txt (add terms and overwrite in sum.txt)

Uppgift 8

Filled //todo

Uppgift 9

1. Updated poly2.h
   a. Included vector
   b. Changed return type
2. Updated poly2.cpp
   a. Changed return type and edit method accordingly
3. Updated polysolver.cpp
   a. Include vector and include poly2.cpp instead of poly2.h
   b. Added method printRoots before main so it’s defined

Uppgift 10

Read arguments, make sure arguments are argc – 1 is divisable by three, and then repeat the steps of uppgift 9.

```
#include <iostream>

#include "poly2.cpp"

#include <vector>
```

```
void printRoots(const std::vector<float>& roots, Poly2 poly) {
        if (roots.size() == 0) {
        std::cout << "No real roots." << std::endl;
        } else if (roots.size() == 1) {
        std::cout << "One real double root: " << roots[0] << std::endl;
        } else {
        std::cout << "Two real roots : " << roots[0] << " and " << roots[1] << std::endl;
        }

std::cout << "Evaluate poly at x = 3: " << poly.eval(3) << std::endl;
}

int main(int argc, const char* argv[])
{
if ((argc - 1) % 3 != 0){
std::cout << "fr?" << std::endl;
return 1;
}

std::cout << "Root-finding started..." << std::endl;

for(int i = 1;  i < argc; i = i + 3){
float a = std::atoi(argv[i]);
        float b = std::atoi(argv[i + 1]);
        float c = std::atoi(argv[i + 2]);

Poly2 poly(a,b,c);
                printRoots(poly.findRoots(), poly);
}

return 0;
```

}

Uppgift 11

I want to avoid std::cin to perserve the control of parameters.

Uppgift 12

I added std::fabs() and compare it to epsilon. I encountered Epsilon in deep learning as well.

“Epsilon (ε) is a concept that appears in various fields of mathematics and computer science, including deep learning and numerical computations involving floating-point numbers. While the term "epsilon" itself might refer to a small positive constant, its use is tied to the idea of precision, accuracy, and handling numerical errors.”

Uppgift 13

Uppgift finns inte

Uppgift 14

Summary of CMake: specify compilation protocols, lists directory, lists classes and finally links where the class can be find.

CMake in MathFunctions: it will be used to link an executable with an actual file in a library (directory).

Uppgift 15

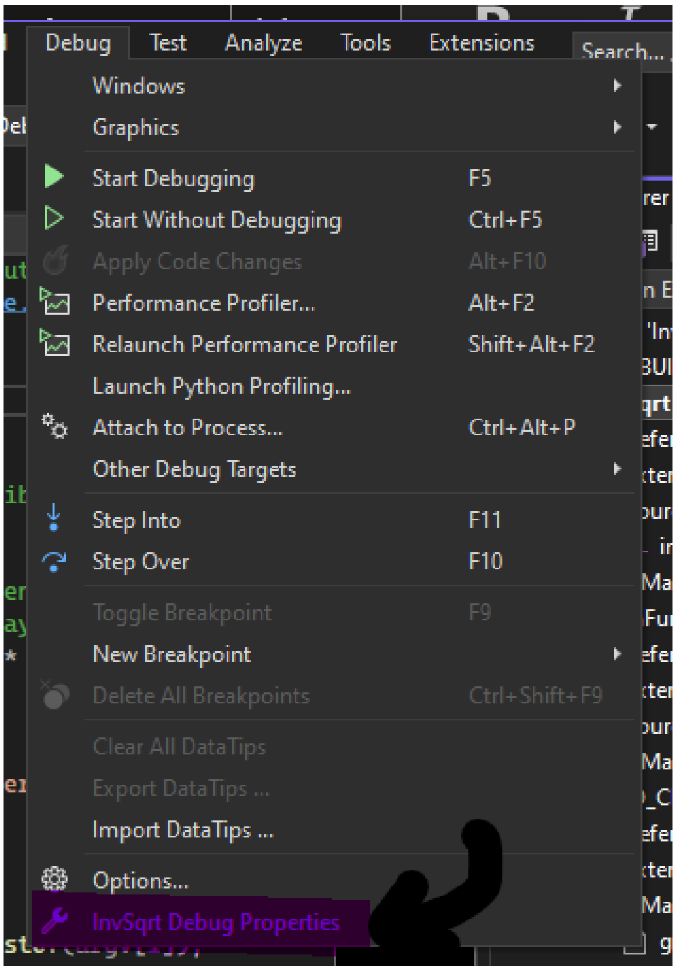
The inverse square root of 9 is 0.332953

Uppgift 15

After some search, the “correct” call for my version of vs2022 is:

```
cmake -G "Visual Studio 17 2022" -B invsqrt_build_vs invsqrt
```

Lyckats få det att funka


Debug
Test
Analyze
Tools
Extensions
Search...
Windows
Graphics
Start Debugging F5
Start Without Debugging Ctrl+F5
Apply Code Changes Alt+F10
Performance Profiler... Alt+F2
Relaunch Performance Profiler Shift+Alt+F2
Launch Python Profiling...
Attach to Process... Ctrl+Alt+P
Other Debug Targets
Step Into F11
Step Over F10
Toggle Breakpoint F9
New Breakpoint
Delete All Breakpoints Ctrl+Shift+F9
Clear All DataTips
Export DataTips ...
Import DataTips ...
Options...
InvSqrt Debug Properties