## Uppgift 1

Kompilera hello.cpp med kommandot: **cl /EHsc hello.cpp**

Vilka nya filer genererades?

* Hello.exe
* Hello.obj

Exekvera programmet med: hello  
Vad skrevs ut, och var?

* Vad: Hello Wolrd!
* Var: Kommandofönstret

Syns något fel i utskriften? Rätta i så fall felet och gör om processen

* Fel = Wolrd
* Skriv om och kompilera och skriv sedan hello för att få en korrekt utskrift i CMD.

## Uppgift 2

Radera de genererade filerna. Kompilera hello.cpp igen, nu med separat länkning.  
Undersök vilka filer som skapas efter kompilering respektive länkning

* Efter kompilering skapas **hello.obj**
* Efter länkning skapas **hello.exe**

## Uppgift 3



## Uppgift 4

#include <iostream>

float val **=** 0**;**

float sum **=** 0**;**

int main**()**

**{**

**while(**std**::**cin **>>** val**)**

**{**

sum **+=** val**;**

**}**

std**::**cout **<<** sum**;**

**return** 0**;**

**}**

## Uppgift 5

Hamnar i sum.txt och ger en 0:a eftersom det finns inga tal att addera.

## Uppgift 6

Skapa en text-fil, terms.txt, med en serie tal separerade med mellanslag. Talen  
kan vara på samma rad eller på flera rader. Dirigera input-strömmen till  
sum.exe, från konsolen till att komma fån terms.txt:  
sum < terms.txt  
Var hamnar output-strömmen? Vad innehåller den?

* Var: CMD fönstret
* Vad: Summan av talen (550) i terms.txt

## Uppgift 7

Summan av talen i numbers.txt (45) skrivs till sum.txt

## Uppgift 8

#include "poly2.h"

#include <cmath>

#include <iostream>

float x\_square**;**

float discriminant**;**

float x1**;**

float x2**;**

Poly2**::**Poly2**(**float a**,** float b**,** float c**)**

**{**

**this->**a **=** a**;**

**this->**b **=** b**;**

**this->**c **=** c**;**

**}**

float Poly2**::**eval**(**float x**)**

**{**

x\_square **=** pow**(**x**,** 2**);**

**return** **(**a**\***x\_square**)** **+** **(**b**\***x**)** **+** c**;**

**}**

void Poly2**::**findRoots**()**

**{**

discriminant **=** pow**(**b**,** 2.0**)** **-** **(**4.0 **\*** a **\*** c**);**

**if** **(**discriminant **==** 0**)**

**{**

std**::**cout **<<** "Rot x = " **<<** **(** **(-**b**)** **/** **(**2.0 **\*** a**)** **)** **<<** std**::**endl**;**

**}**

**else** **if** **(**discriminant **>** 0**)**

**{**

x1 **=** **(-**b **+** sqrt**(**discriminant**))** **/** **(**2.0 **\*** a**);**

x2 **=** **(-**b **-** sqrt**(**discriminant**))** **/** **(**2.0 **\*** a**);**

std**::**cout **<<** "Rot x1 " **<<** x1 **<<** std**::**endl**;**

std**::**cout **<<** "Rot x2 " **<<** x2 **<<** std**::**endl**;**

**}**

**else**

**{**

std**::**cout **<<** "Roots = imaginary"**;**

**}**

**}**

## Uppgift 9

### Poly2.h

void findRoots**(**float **&**x1**,** float **&**x2**,** int **&**rootNbr**);**

### poly2.cpp

#include "poly2.h"

#include <cmath>

#include <iostream>

float x\_square**;**

float discriminant**;**

Poly2**::**Poly2**(**float a**,** float b**,** float c**)**

**{**

**this->**a **=** a**;**

**this->**b **=** b**;**

**this->**c **=** c**;**

**}**

float Poly2**::**eval**(**float x**)**

**{**

x\_square **=** pow**(**x**,** 2**);**

**return** **(**a**\***x\_square**)** **+** **(**b**\***x**)** **+** c**;**

**}**

void Poly2**::**findRoots**(**float **&**x1**,** float **&**x2**,** int **&**rootNbr**)**

**{**

discriminant **=** pow**(**b**,** 2.0**)** **-** **(**4.0 **\*** a **\*** c**);**

**if(**discriminant **==** 0**)**

**{**

rootNbr **=** 1**;**

x1 **=** **(-**b**)** **/** **(**2.0 **\*** a**);**

**}**

**else** **if** **(**discriminant **>** 0**)**

**{**

rootNbr **=** 2**;**

x1 **=** **(-**b **+** sqrt**(**discriminant**))** **/** **(**2.0 **\*** a**);**

x2 **=** **(-**b **-** sqrt**(**discriminant**))** **/** **(**2.0 **\*** a**);**

**}**

**else**

**{**

rootNbr **=** 0**;**

**}**

**}**

### Polysolver.cpp

#include <iostream>

#include "poly2.h"

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

**{**

float x\_var **=** **-**0.5**;**

float value\_x1 **=** 0**;**

float value\_x2 **=** 0**;**

int rootNbr **=** 0**;**

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

//Poly2 poly1(1.0,2.0,1.0);

Poly2 poly2**(**2**,** **-**1**,** **-**1**);**

//Poly2 poly3(1.0, 1.0, 1.0);

//Poly2 poly4(1, -5, 7);

float evalValue **=** poly2**.**eval**(**x\_var**);**

std**::**cout **<<** "Eval when x = " **<<** x\_var **<<** " is " **<<** evalValue **<<** std**::**endl**;**

poly2**.**findRoots**(**value\_x1**,** value\_x2**,** rootNbr**);**

**if** **(**rootNbr **==** 1**)**

**{**

std**::**cout **<<** "x1 = " **<<** **(** value\_x1 **)** **<<** std**::**endl**;**

**}**

**else** **if** **(**rootNbr **==** 2**)**

**{**

std**::**cout **<<** "x1 = " **<<** value\_x1 **<<** std**::**endl**;**

std**::**cout **<<** "x2 = " **<<** value\_x2 **<<** std**::**endl**;**

**}**

**else**

**{**

std**::**cout **<<** "Roots = imaginary"**;**

**}**

**return** 0**;**

**}**

## Uppgift 10

include <iostream>

#include "poly2.h"

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

**{**

float value\_x1 **=** 0**;**

float value\_x2 **=** 0**;**

int rootNbr **=** 0**;**

float coeff\_1 **=** 0**;**

float coeff\_2 **=** 0**;**

float coeff\_3 **=** 0**;**

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

**while(**std**::**cin **>>** coeff\_1 **>>** coeff\_2 **>>** coeff\_3**)**

**{**

Poly2 poly2**(**coeff\_1**,** coeff\_2**,** coeff\_3**);**

poly2**.**findRoots**(**value\_x1**,** value\_x2**,** rootNbr**);**

**if** **(**rootNbr **==** 1**)**

**{**

std**::**cout **<<** "x1 = " **<<** **(** value\_x1 **)** **<<** std**::**endl**;**

**}**

**else** **if** **(**rootNbr **==** 2**)**

**{**

std**::**cout **<<** "x1 = " **<<** value\_x1 **<<** std**::**endl**;**

std**::**cout **<<** "x2 = " **<<** value\_x2 **<<** std**::**endl**;**

**}**

**else**

**{**

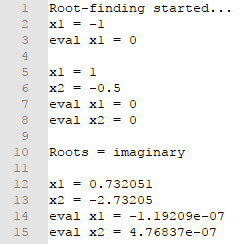
std**::**cout **<<** "Roots = imaginary" **<<** std**::**endl**;**

**}**

**return** 0**;**

**}**

## Uppgift 11

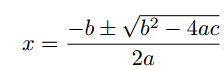


## ”Uppgift 11/12”

Ordningen på polynomet är 1 om a = 0. Det vill säga ekvationen kommer ha ett nollställe då x = .

Om b = 0 kommer det finnas två rella rötter endast om c eller a är negativ med tanke på determinanten.

Om c = 0 finns två rella rötter

Om a = b = 0 kommer det inte finnas några nollställen enligt 

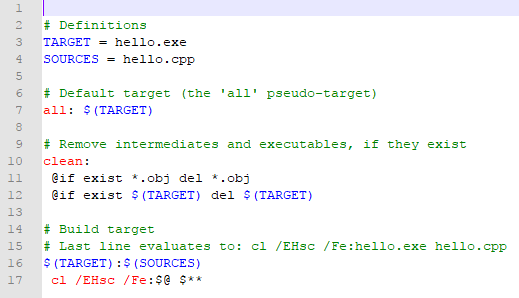
## Uppgift 13

Efter nmake all:

* Hello.exe och Hello.obj skapas

Efter nmake clean:

* Hello.exe och Hello.obj raderas



Genom att kalla på nmake all kompileras hello.cpp. Man slipper skriva nmake hello.exe. På rad 16 sker en dereference av makron för att kunna bygga cpp filen.

$(TARGET) = hello.exe (dereferencing macro)

$(SOURCES) = hello.cpp (dereferencing macro)

Clean är också en pseudotarget som gör så att man slipper skriva @if exis…..

## Uppgift 14

Mappen bin med filerna Hellopi.exe och Main.obj skapas.

Ungefär liknande process sker men man flyttar filerna till mappen bin från src. Dvs man kopierar och raderar filerna.