

Norges teknisk–naturvitenskapelige
universitet
Institutt for datateknologi og
informatikk

TDT4102 Prosedyre- og objektorientert programmering Vår 2024

Øving 0 for GNU/Linux

Frist: Som for Øving 1

Mål for denne øvingen:

- Bli kjent med programmeringsverktøy
- Lage et første program med Visual Studio Code (VS Code)
- Kunne laste ned og kjøre eksempelprogram fra forelesningene med VS Code
- Lage et første program kun med teksteditor og kompilator
- Bli kjent med infobanken

Denne øvingen er en veiledning i å installere en programmeringsomgivelse slik at du kan skrive, redigere, compilere, debugge og kjøre et C++ program. **Det er nødvendig å gjennomføre og mestre det meste av det som gjennomgås i denne øvingen for å kunne utføre de obligatoriske øvingene.**

Vi vil våren 2024 benytte verktøyet VS Code som er gratis og kan brukes under Windows, MacOS og GNU/Linux.

Vær obs på at installering av alle programmene kan ta opptil flere timer, avhengig av hvor rask internettilkobling og PC du har.

Hvis du har tatt emnet tidligere og allerede har VS Code med utvidelsen TDT4102 Tools installert, kjør `TDT4102: Install required tools` og gå videre til oppgave 1.

Hvis du kjører GNU/Linux eller en annen form for Unix-liknende operativsystem (macOS har egen øving 0), er denne øvingen en generell veiledning til hvordan fagets øvinger *kan* bygges og kjøres. For å gjøre det enkelt er det i denne øvingen også kun installering i Ubuntu/Mint/Pop!OS som vil vises, det antas at de som velger å benytte andre varianter av Linux kjenner til hvordan et bibliotek lastes ned, kompiles og installeres. **Merk at de to primære platformene som anbefales og støttes i faget TDT4102 er Windows og Mac.** De aller aller fleste studentene i faget benytter Windows eller Mac, og *fagstaben har ikke kapasitet til å gi veiledning i bruk av Linux*. Hvert år har vi et mindre antall studenter i faget som er godt kjent med Linux, og dette notatet er ekstra hjelp til disse, selv om disse studentene pleier å klare seg selv. Notatet er *ikke* en oppfordring til Windows- eller macbrukere om å bytte til GNU/Linux.

Programvare som installeres i denne øvingen er bl.a. `clang` og `meson`. *Denne øvingen tar utgangspunkt i Ubuntu 22.04 LTS.*

Aktuelle kapitler i boka:

- Kapittel 0, 1 og 2 i Programming – Principles and Practice Using C++ (Second Edition)

0 Oppgave 0 - Installasjon

Hurtigoppsett

Dersom du er godt rutinert på installasjoner kan du følge den påfølgende forkortede installasjonsguiden. Vi vil likevel råde de aller fleste til å følge den detaljerte installasjonsguiden nedenfor.

1. Last ned VS Code for din distribusjon; <https://code.visualstudio.com/docs/setup/linux>
2. Gå til extension-menyen i VS Code (fire bokser i venstremenyen), og installer utvidelsen «TDT4102 Tools»
3. Trykk **Ctrl+Shift+P** og velg kommandoen **TDT4102: Install required tools**.

NB! Kun Ubuntu/Mint/Pop!OS er støttet av installasjonsverktøyet. Instruksjoner for andre distribusjoner finnes på nettsiden:

<https://tdt4102.pages.stud.idi.ntnu.no/documentation/installing/linux/#manuell-installasjon>

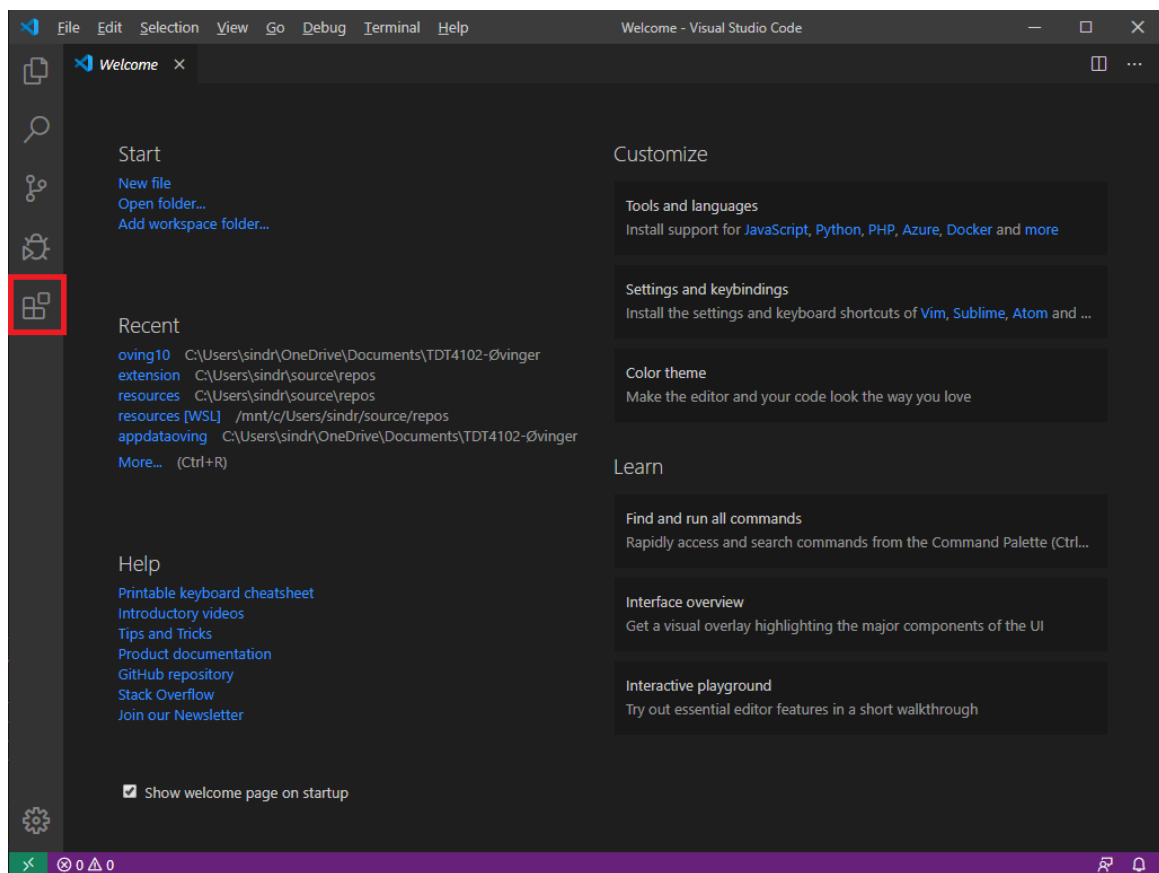
4. Godkjenn eventuelle pop-ups om tilganger og installer eventuelle programmer eller tjenester som utvidelsen ber om
5. Lukk VS Code og gå videre til oppgave 1 i denne øvingen

Detaljert installasjonsguide

Les gjennom hele installasjonsinstruksen før du begynner.

Installasjonen krever at du installerer VS Code for å skrive og redigere kode.

VS Code er en teksteditor laget for å skrive og redigere kode i forskjellige programmeringsspråk. Programmet kan lastes ned og installeres fra <https://code.visualstudio.com/docs/setup/linux>. Husk å følg instruksjonene for din distribusjon.



Figur 1: VS Code Extension-meny i rødt.

0.1 Installere TDT4102 Tools

VS Code vil la oss skrive og redigere kode, men vi trenger også en kompilator og bibliotek som følger med boken for å bygge og kjøre øvingene. For å forenkle denne prosessen har vi laget en utvidelse (extension) til VS Code. Denne utvidelsen vil laste ned og starte installasjonen av de resterende verktøyene vi trenger, samt gjøre det lett å lage nye prosjekter. **Ikke lukk datamaskinen underveis i installasjonsprosessen, dette kan ta alt fra 10 minutter til et par timer.** *Det kan hende du må skrive inn passordet ditt flere ganger for at installasjonen skal kunne fullføre.*

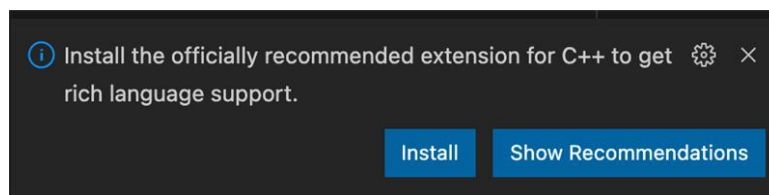
Merk at stegene i denne prosessen kan variere hvis du allerede har installert VS Code, Xcode eller andre verktøy tidligere.

1. Åpne VS Code og velg Extensions fra menyen til venstre, se **Figur 1**. (Ctrl+Shift+X)
2. Søk etter TDT4102 Tools, og trykk på Install.
3. Start installasjon av nødvendige programmer ved å trykke Ctrl+Shift+P og velg kommando TDT4102: Install required tools. Nederst til høyre i VS Code-vinduet vil det dukke opp en noen meldinger som forteller om installeringen er underveis, er fullført eller har feilet.

NB! Hvis du bruker en annen distribusjon enn Ubuntu/Mint/Pop!OS så finnes instruksjoner for installasjon av nødvendige programmer på nettsiden:

<https://tdt4102.pages.stud.idi.ntnu.no/documentation/installing/linux/#manuell-installasjon>

4. I de fleste tilfeller får man en eller flere meldinger fra VS Code om at enkelte programmer eller tjenester mangler. Dersom dette skjer trykker du **Install** og deretter **Allow** eller **Install**, hvis du får spørsmål om å godkjenne tilgang.



Figur 2: Manglende programmer under installasjon av extension i VS Code. Trykk **Install** og hvis du får spørsmål om å godkjenne tilgang, trykk **Allow**.

5. Du vil bli bedt om å restarte VSCode når installasjonen er ferdig. Gjør dette.

TDT4102-utvidelsen vil sjekke at du har de nyeste filene og malene, samt at alt du trenger er installert hver gang du åpner VS Code. Dersom det for eksempel skulle komme endringer i malene vil disse bli lastet ned automatisk, og du vil få en melding nederst i høyre hjørne av VS Code om at malene er oppdatert. Likevel kan det av og til være lurt å manuelt sjekke at alt er oppdatert ved å trykke Ctrl+Shift+P og skrive TDT4102: Force refresh of course content og TDT4102: Perform a health check of the setup.

Merk: Insider-version

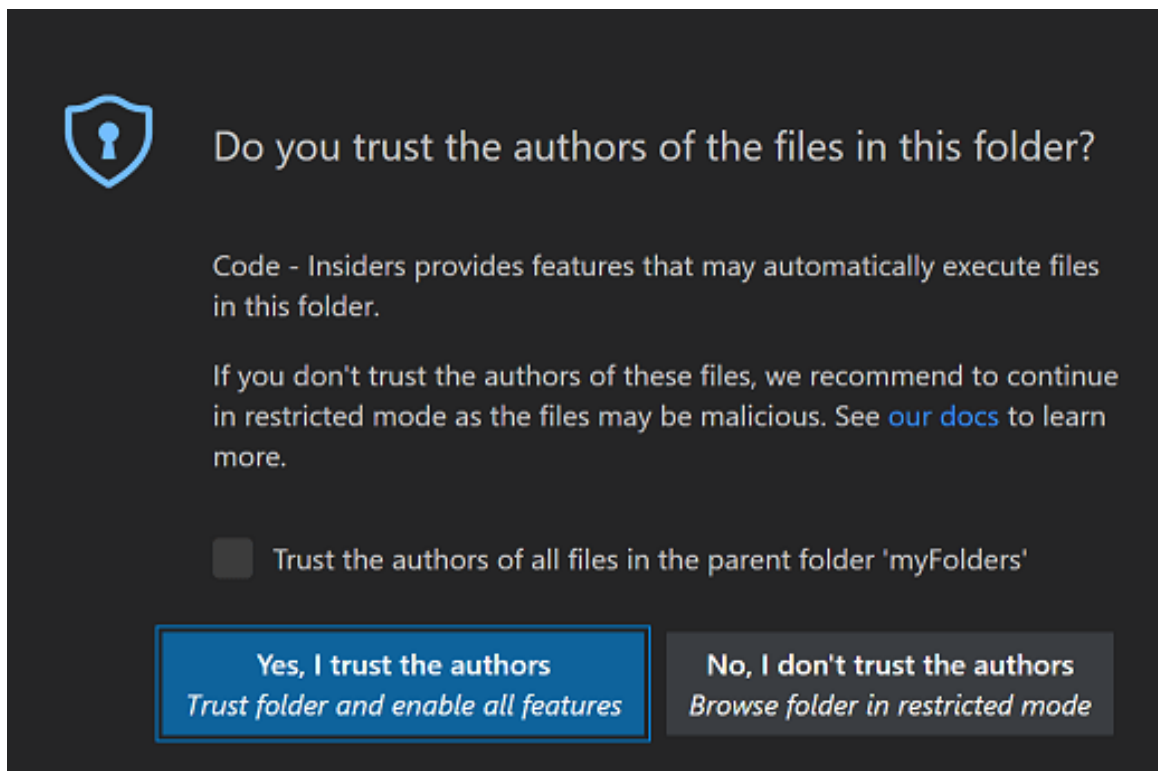
VS Code kommer til å spørre om du vil bli med i insider-programmet. Takk **nei** til det.

1 Oppgave 1 - Programmering med VS Code

I denne oppgaven antas det at du allerede har installert VS Code, som beskrevet i Oppgave 0.

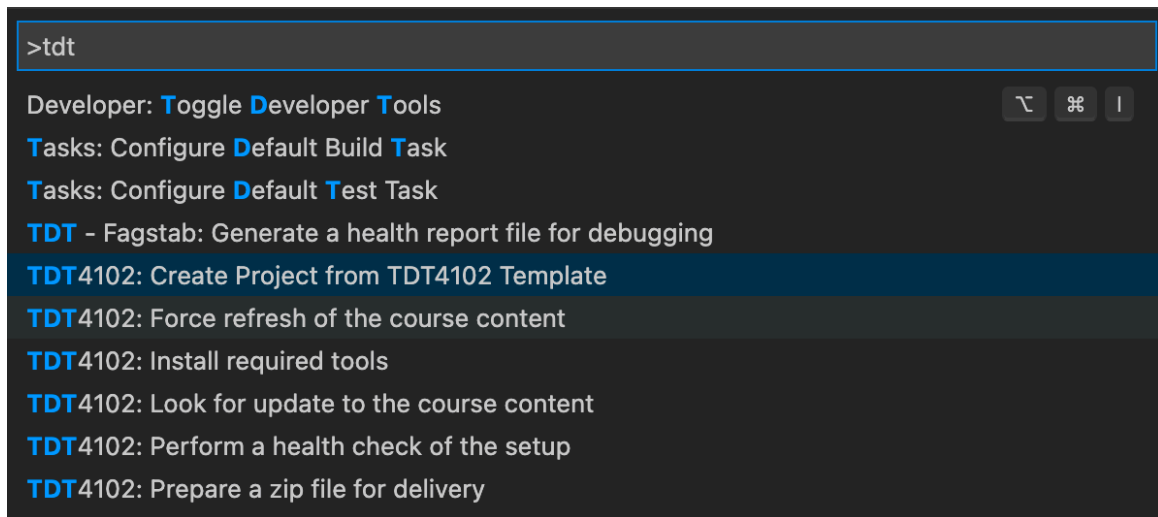
1.1 Oppgave 1.1 - Mitt første prosjekt i VS Code

1. Åpne VS Code som du normalt åpner programmer. I Ubuntu vil en standard-installasjon la deg trykke Windows-/Super-knappen og skrive `code`, da vil du få mulighet til å starte Visual Studio Code.
2. Lag en ny tom mappe et valgfritt sted. Gå inn i Visual Studio Code og trykk på **File** øverst til venstre, deretter trykk **Open folder**. Velg mappen du laget.
 - Hvis du blir spurt om du stoler på mappen **3**, trykk på **Yes, I trust the authors**. Dette er en funksjon Visual Studio Code har laget slik at man kan lese kode andre har skrevet uten å risikere å kjøre den med uhell. I dette faget ønsker vi selvfølgelig å kjøre koden vi selv skriver, og vi må derfor markere at vi stoler på filene. Dette er også nødvendig for at TDT4102-utvidelsen vi bruker i faget skal fungere, fordi denne utvidelsen hjelper oss å kjøre programmer. For mer info klikk [her](#).



Figur 3: Meny hvor du kan gi tillit til arbeidsmappen.

3. Trykk **Ctrl+Shift+P** for å få opp *kommandopaletten*. Begynn å skrive **TDT4102: Create Project from TDT4102 Template** og trykk *Enter* når du har markert valget. Velg deretter **Blank project**. Se [Figur 4](#) for hvordan kommandopaletten ser ut etter å ha skrevet tdt.



Figur 4: Kommandopaletten (Command Palette) etter å ha skrevet `tdt` første gang.

4. Til venstre i VS Code vil du nå se en oversikt over filene i prosjektmappen din. Her skal det ligge fem elementer: filene `main.cpp` og `meson.build`, og mappene `.vscode`, `builddir` og `subprojects`. `Subprojects`-mappen inneholder en del kode som vi er avhengige av for å tegne grafikk senere i kurset. `Builddir` er mappen der den ferdige maskinkoden blir lagret når du kompilerer programmet ditt.
 - Når utvidelsen gjenkjenner et korrekt oppsatt prosjekt vil det stå **TDT4102 Project** ✓ nederst i venstre hjørne av VS Code. For å sjekke at du har åpnet VS Code riktig og at alt er som det skal kan du alltid se etter denne teksten og haken når du har åpnet et prosjekt som er korrekt oppsatt for øvingsopplegget. Det vil for eksempel ikke være synlig i en tom mappe.
 - Filen `meson.build` og mappen `.vscode` inneholder innstillinger som bestemmer hvordan VS Code skal kompilere og kjøre koden i prosjektet ditt. Vi anbefaler at du ikke endrer disse filene med det første. Vi skal lære mer om noen av disse senere i faget. Hvis du mot formodning skulle endre noen av disse filene kan du nullstille konfigurasjonen ved å skrive **Ctrl+Shift+P**, skrive **TDT4102: Create Project from TDT4102 Template** og velg **Configuration only** og deretter trykke **overwrite** på alt. Ønsker du i stedet å begynne på nytt, bruk **Blank Project** og overskriv alt.
5. Når du har åpnet et eksempelprosjekt på riktig måte vil utvidelsen automatisk kjøre en setup av prosjektet. Dette vil du kunne se output fra i terminalvinduet ditt. Under kjøring vil Meson klage over at den ikke finner noen pakker, dette kan ignoreres. Om alt kjører som det skal vil det se ut som **Figure 5**.
 - Det kan hende at setup ikke fungerer helt riktig. Da kan du gjøre det manuelt. Dette gjør du ved å åpne et terminalvindu . Skriv deretter inn **meson setup builddir**.
6. Det viktigste for oss er `main.cpp`, i denne ligger kildekoden til et enkelt program som skriver "Hello, World!" til skjermen. Hvis du trykker på `main.cpp` vil filen åpnes som en fane i VS Code så du kan se og redigere den.
7. Kjør programmet ditt ved å trykke **Ctrl+F5** (det kan være du må trykke **fn**-tasten for å få tilgang til F-tastene, det blir i så fall **fn+Ctrl+F5**), eller ved å gå inn i menyen **Run and**

```
testproject 0.1

Subprojects
  animationwindow : YES
  sdl2_image_windows : YES
  sdl2_windows : YES

User defined options
  buildtype : debug
  sdl2:run_test : false
  sdl2:test : false
  sdl2:use_audio_alsa: disabled
  sdl2:use_file : disabled
  sdl2:use_haptic : disabled
  sdl2:use_joystick : disabled
  sdl2:use_locale : disabled
  sdl2:use_power : disabled
  sdl2:use_sensor : disabled

Found ninja-1.10.2 at "C:\Program Files\Meson\ninja.EXE"
```

Figur 5: Om meson setup kjører som det skal vil du få en oppsummerende output som kan se f.eks. slik ut. Outputen er ikke nødvendigvis helt lik hver gang.

Debug og velge **Build and Run Debug**. I terminalvinduet på bunnen (den kan flyttes) av **VS Code** vil teksten "Hello, World!" være skrevet. Hvis du ikke kan se terminalen trykk på **Terminal** i menyen øverst, trykk så **New Terminal** og kjør programmet igjen.

- Vi anbefaler sterkt å bruke auto save i dette faget. For å skru det av og på gå til menyen øverst og trykk på **File** så på **Auto save**. Hvis Auto Save er av vil TDT4102 Tools minne deg om å skru det på.

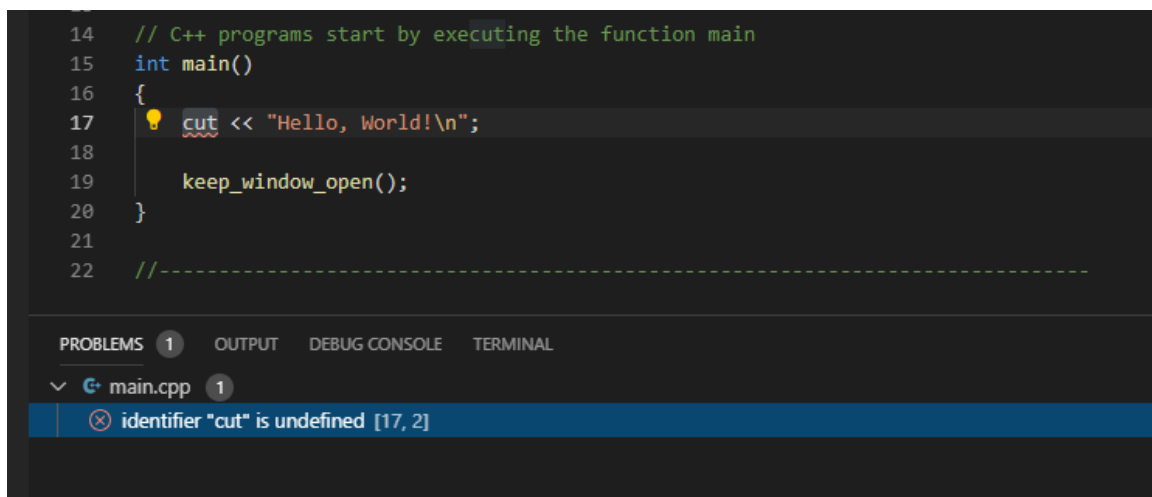
NB: Hvis du dukker opp i problemer, kan dette være på grunn av bruk av skylagringstjenester (iCloud, OneDrive, Dropbox, etc.) Sørg for at skylagringstjenestene ikke virker på arbeidsfilene.

1.2 Oppgave 1.2 - Kompilering og feilmeldinger

Når du kjører programmer som i siste avsnitt av forrige seksjon vil du ofte få feil. Derfor vil en god del av tiden du bruker på å gjøre øvinger bestå i å finne ut hva som er feil i koden din. En type feil er syntaktiske feil (enkle skrivefeil) som resulterer i kode som ikke vil kompilere. Hvis du prøver å kompilere kode som ikke er riktig skrevet vil kompilatoren gi deg feilmelding(er) som inneholder informasjon om hva som kan være galt. Noen ganger er dette forståelig informasjon, andre ganger kan det være vanskelig å skjønne feilmeldingene.

Vi antar nå at koden kjører som den skal. Du skal videre i denne oppgaven «ødelegge» koden din med vilje ved å endre på småting, og deretter observere hva slags feilmeldinger du får når du kompilerer. Du kan for eksempel gjøre følgende:

1. Slett semikolonet på slutten av linjen som skriver ut teksten «Hello World!» og kompiler på nytt.
2. Det vil nå dukke opp en feilmelding i terminalvinduet nederst, forstår du feilmeldingen?
3. Introduser andre feil og les feilmeldingene som kompilatoren gir. Du kan f.eks. prøve å slette en av krøllparentesene, skrive `cout` som `cut` osv.
4. Det skal også dukke opp feilmeldinger i Problems-fanen nederst i VS Code, se [Figur 6](#). Her kan du klikke på linjer i feilmeldingen for å se hvordan VS Code markerer linjene i koden din der den tror feilen ligger.
5. Husk å rette opp feilene i filen igjen før du går videre.



Figur 6: Problems-fanen i VSCode

NB: Det er ikke alltid Problems-fanen er pålitelig

Problems-fanen er fin for å fremheve syntaksfeil i koden din, når den fungerer. Det finnes dessverre tilfeller der fanen enten ikke fanger opp alle feil, eller at den til og med uthever ikke-feil i koden din.

Fasiten får du alltid ved kompilering av koden din, som skjer når du *bygger* prosjektet med **Ctrl+Shift+B**, eller ved kjøring av programmet. Alle feil du eventuelt har i koden din vises da som feilmeldinger i *terminalen* - denne er alltid pålitelig. Ta derfor Problems-fanen alltid med en klype salt.

1.3 Oppgave 1.3 - Eksempelprogrammer fra forelesningene

Ved å åpne kommandopaletten (`Ctrl+Shift+P`) og skrive `TDT4102: Create Project from TDT4102 Template` vil du få opp valg om å opprette et nytt prosjekt. Til nå har vi brukt malen `Blank project` og `Configuration only`. Det er også fire valg som heter `Examples`, `Exercises`, `Inspira` og `Lectures`. Ved å velge `Lectures` vil det dukke opp en liste med eksempelprogrammene fra forelesningene. Disse kan åpnes på samme måte som andre maler. Eksempelene inneholder ikke prosjektinnstillinger som `.vscode`-mappen eller `meson.build` filen. Disse prosjektinnstillingene finnes i malen `Configuration only`. Denne bruker vi for å gjøre eksempelprogrammene komplette slik at de kan kjøres fra VS Code.

For å åpne og kjøre et eksempel vil vi:

1. Lage en ny mappe og åpne den i VS code som beskrevet i oppgave 1.1.2
2. Opprette et prosjekt med malen `Configuration only` (`.vscode`, `subprojects`, `builddir` og `meson.build`)
 - `Ctrl+Shift+P` og velg `TDT4102: Create Project from TDT4102 Template`
 - Velg `Configuration Only`
3. Åpne en av malene i `Eksempler` mappen.
 - `Ctrl+Shift+P` og velg `TDT4102: Create Project from TDT4102 Template`
 - Velg `Examples`
 - Skriv inn `hello_graphics` i søkefeltet og velg eksempelet
4. Kjør eksempelprogrammet ved å trykke `Ctrl+F5` (det kan være du må trykke `fn`-tasten for å få tilgang til `F`-tastene, det blir i så fall `fn+Ctrl+F5`), eller ved å gå inn i menyen **Run and Debug** og velge **Build and Run Debug**.

Hvis alt har gått som det skal vil det nå åpnes et nytt vindu med en blå sirkel. Dette er et eksempel på enkel grafikk, som vi kommer til å bruke en del senere i øvingsopplegget.

1.4 Oppgave 1.4 - Kompilere egne filer

Nå som du har lært deg hvordan hente og kjøre eksempelprogrammer skal vi se på hvordan bygge programmer med flere filer. Det første du gjør er å hente eksempelprogrammet for Øving 0 ved følge liknende steg som i [Avsnitt 1.3](#). Altså hent ut templatene som ligger under "Exercises/O00".

Dersom du har funnet riktig oppsett skal du nå ha tre filer: `main.cpp`, `other.h` og `other.cpp`. Dere skal lære mer om `h`-filer senere. Det viktige for denne oppgaven er at vi må forteller `meson` at filen `other.cpp` eksisterer, slik at den også blir bygget. Det gjør du ved å gå inn i `meson.build` filen. Der det står `src = ['main.cpp']` må du nå legge til `other.cpp` i listen slik at det nå står `src = ['main.cpp', 'other.cpp']`.

Om du har gjort dette riktig vil du få til å bygge og kjøre programmet ditt med `f5`. Du vil måtte legge til alle nye filer du lager på denne måten gjennom faget.

1.5 Oppgave 1.5 - Lage .zip for innlevering

Utvidelsen TDT4102 Tools har en funksjon for å pakke filene i øvingen til en .zip-fil som kan leveres på BlackBoard. Det er obligatorisk å bruke denne funksjonen, slik at vi vet vi får innleveringer på riktig format. Trykk på **Ctrl+Shift+P** og skriv **TDT4102: Prepare a zip file for delivery**. Trykk enter. En .zip-fil med navn **handin.zip** vil dukke opp i marginen til venstre og ligge i mappen med øvingen. Du må selv laste opp filen på BlackBoard for å få godkjent øvingen.

NB! Du skal ikke levere noe for denne øvingen, men test ut at det fungerer å lage en .zip-fil.

1.6 Oppgave 1.6 - Infobanken

En av funksjonene til TDT4102 Tools gir det tilgang til fagets infobank. Her vil du kunne få tilgang på nyttige ressurser som artikler, videoer og øvinger. Formålet er at du enkelt skal kunne finne det du trenger inne i VS Code.

For å åpne infobanken trykk på **Ctrl+Shift+P** og skriv **TDT4102: Open infobank**. En liste over ulike ressurser vil dukke opp hvor du kan søke eller bla/scrolle til den ressursen du ønsker å bruke. Du kan søke både etter tittel eller nøkkelord.

Prøv å åpne følgende ressurser:

1. **Piazza** - fagets forum
2. **Feilmeldinger** - en artikkel som hjelper med debugging
3. **Hello graphics** - her finner du lenke til en video som forklarer hvordan man lager grafikk i c++
4. **Øving 0** - Du kan til og med finne denne øvingen i infobanken! Øvingene i faget vil bli lagt ut i infobanken i løpet av året

1.7 Oppgave 1.7 - Kompilering fra kommandolinje (frivillig oppgave)

Denne oppgaven er ikke nødvendig for å følge øvingsopplegget videre, men gir bedre innsikt i hvordan kompilering og kjøring av C++ kode fungerer, uten at Visual Studio Code gjør det for deg.

Skriving av kildekode og kompilering kan i prinsippet gjøres med enkle verktøy. En helt vanlig teksteditor er alt du trenger for å skrive kode, og kompilering kan gjøres ved å kjøre kompilatoren fra kommandolinja. Eksempelprogrammet vi skal bruke i denne oppgaven vises i **Figur 7**.

```
#include <iostream>
int main() {
    std::cout << "Hello World" << std::endl;
}
```

Figur 7: Eksempelprogram uten avhengigheter på AnimationWindow.h

1. Start en vanlig teksteditor. Nano er en veldig enkel teksteditor som følger med de fleste Linux-distribusjoner og kan brukes her. Skriv inn eksemplet fra Figur 7. Opprett en ny mappe og lagre filen din der, for eksempel med navnet `HelloWorld.cpp`. Sjekk mappen hvor filen din er lagret, og forsikre deg om at den er der og har riktig navn.
2. Start opp et terminalvindu. Terminalvinduet starter i hjemmemappen din. Dersom du ikke lagret `HelloWorld.cpp` direkte i hjemmemappen din, må du flytte deg til riktig mappe. Dette gjør du med kommandoen `cd` (change directory). Hvis du for eksempel la filen i «Dokumenter»-mappen din, vil du måtte skrive:

```
cd Documents
```

Du befinner deg nå i «Documents»-mappen din.

3. Skulle filen befinne seg i en undermappe, gjentar du `cd`-kommandoen, denne gangen med navnet til undermappen, for å gå videre dit. Ønsker du å sjekke hvilken mappe du befinner deg i, kan du gjøre dette med kommandoen `pwd`.
Dersom du skriver kommandoen `cd` uten noe etterpå (det vil si uten noen *argumenter*) vil du bli returnert til hjemmemappen din. Ønsker du å gå til mappen *over* den du befinner deg i, kan du skrive `cd ..` (to punktum). «Over» refererer her til over i *mappehierarkiet*. Befinner du deg i mappen `~/Documents/tdt4102`, vil mappen over være `~/Documents`.
4. Skriv kommandoen `ls` (list) for å se hvilke filer (eller mapper) som ligger i mappen du befinner deg i.
5. Har du funnet fram til mappen der `HelloWorld.cpp` ligger, er du klar til å kompilere programmet. Kompilatoren vi bruker i TDT4102 heter `clang++`, og du bruker denne til å kompilere programmet ditt ved å skrive:

```
clang++ HelloWorld.cpp
```

6. Sjekk nå innholdet av mappen med `ls`, og se hvilke filer som ble produsert da du kompilerte. `a.out`-filen er programmet ditt. Kjør programmet du har laget ved å skrive `./a.out`. Hvis alt har gått bra skal utskriften fra programmet ditt vises i terminalen.
7. Rediger teksten i `HelloWorld.cpp` slik at programmet skriver ut noe annet (husk å ha med hermetegnene rundt teksten).
8. Kompiler og kjør på nytt.

Her kompilerte vi et veldig enkelt program som besto av bare en fil, og helt uten grafikk. Man kan i prinsippet gjøre det samme for større programmer, men det krever at man spesifiserer hvordan filene skal lenkes sammen, hvor grafikkbiblioteket ligger og noen andre detaljer som er tungvint å gjøre for hånd. Meson som vi bruker i dette faget gjør alt dette automatisk slik at vi ikke trenger å bekymre oss for det.