

Norges teknisk–naturvitenskapelige
universitet
Institutt for datateknologi og
informatikk

TDT4102 Prosedyre- og objektorientert programmering Vår 2020

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

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 2020 benytte verktøyet VS Code som er gratis og som kan brukes under Windows, MacOS og Linux. Forelesningene vil da bedre kunne dekke det verktøyet alle studentene bruker.

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 som vil vises, det antas at de som velger å benytte andre varianter av Linux kjenner til hvordan et bibliotek lastes ned, kompileres og installeres. **Merk at de to primære platformene som anbefales og støttes i faget TDT4102 er Windows PC og Mac.** De aller aller fleste studentene i faget benytter PC 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 PC eller Mac brukere til å skifte over til GNU/Linux.

Programvare som installeres i denne øvingen er bl.a. `clang`, `make` og `FLTK`. *Denne øvingen tar utgangspunkt i Ubuntu 18.04 LTS.*

Aktuelle kapitler i boka:

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

Oppgave 0 - Installasjon av byggeverktøy og FLTK

Linjen under installerer bl.a. make og clang.

```
sudo apt install make cmake clang git libjpeg-dev libx11-dev
```

Installering av FLTK 1.4

Kildekoden til FLTK 1.4 lastes ned fra github vha kommandoen

```
git clone git@github.com:ftlk/ftlk.git
```

Mappen `ftlk` inneholder nå kildekode til FLTK, og biblioteket skal bygges og installeres. Det gjøres vha. kommandoene under.

```
cd fltk
mkdir build
cd build
cmake .. # Genererer makefiler
make # Bygger FLTK
sudo make install # Installerer FLTK
```

FLTK 1.4 er versjonen faget bruker. Vi kjenner til at FLTK 1.3 kan installeres gjennom `apt`, men vi anbefaler å bruke versjon 1.4 for å unngå uventede forskjeller. Sannsynligvis vil det gå greit å bruke FLTK 1.3, men det anbefales å installere FLTK 1.4 vha. metoden over for at øvingene skal oppføre seg som forventet. FLTK 1.3 kan installeres i Ubuntu vha.

```
sudo apt install libftlk1.3 libftlk1.3-dev
```

Installering av `graph_lib`

Boken og øvingene bygger på en headerfil og en wrapper for et grafikkbibliotek. Headerfilen `std_lib_facilities.h` vil skjule noe av kompleksiteten og gjøre programmering i C++ trygghere den første halvdelen av kurset. Wrapperen `graph_lib` gjør det mulig å bytte ut det underliggende grafikkbiblioteket (`ftlk`) uten at applikasjoner som benytter `graph_lib` må kompileres på nytt eller endres.

Last ned `TDT4102_Linux.zip` fra blackboard. Pakk ut innholdet og gå til roten av den nye stien:

```
unzip TDT4102_Linux.zip
cd TDT4102_Linux
```

Du bestemmer selv hvor du ønsker å plassere headerfiler og bibliotek. Et alternativ er å kompilere `Graph_lib` og legge det i et bibliotek, som installeres til `/usr/local`. Det følger med en Makefile som gjør dette for deg.

```
cd Graph_lib
make all
sudo make install
cd ..
```

Nå skal du ha en filstruktur der du kan se filstiene `Graph_lib`, `example` og `tdt4102_base_project`. I mappen `example` ligger et program som tester at et enkelt grafikk-prosjekt kan bygge og kjøre. For å verifisere at alt fungerer som det skal, kjør følgende:

```
cd example
make runrelease
```

Du skal nå se en rød trekant. Hvis dette fungerte uten problemer kan du rydde opp etter kompileringen ved å skrive:

```
make clean
```

Installering av VS Code

Installeringsveiledning finnes på <https://code.visualstudio.com/docs/setup/linux>. For Ubuntu går den i korte trekk ut på å laste ned og kjøre <https://go.microsoft.com/fwlink/?LinkID=760868>.

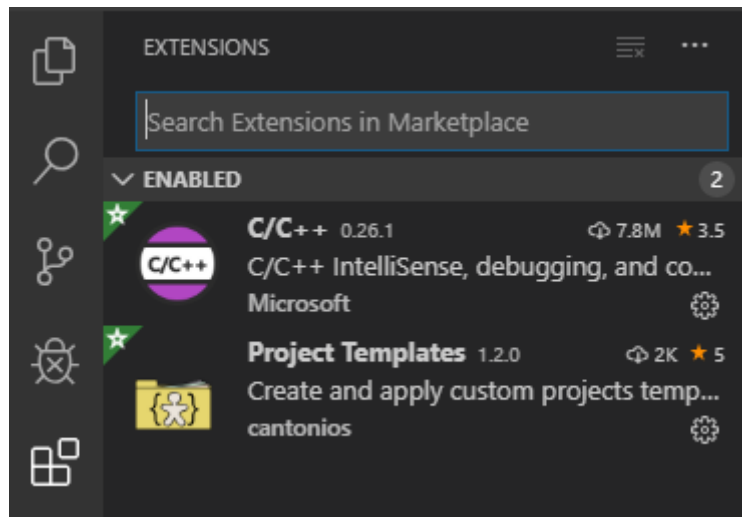
Etter at du har installert VS Code kan du gå til Oppgave 1 for å lære hvordan du kan sette opp dine egne prosjekter.

Oppgave 1 - Programmering med VS Code

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

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. Velg **File** → **Open Folder** og åpne mappen `tdt4102_base_project`, som også ligger der du pakket ut TDT4102.
3. Når du har åpnet mappen (trykket "select Folder") skal det dukke opp en popup nede til høyre med "recommended extensions". Trykk "install all" så vil VS Code installere noen utvidelser som hjelper med skriving og debugging av C++-kode. Hvis du får tilbud om "Insiders version" så velg "Don't Show Again".
4. Du kan sjekke at du har de nødvendige utvidelsene med å trykke på "extensions" ikonet i menybaren helt til venstre på skjermen. (Det nederste av fem ikoner, ser ut som fire byggeklosser.) Her bør du se "C/C++" og "Project Templates" under enabled (figur 1), eller så er de "enabled globally". Hvis de ikke er der kan du søke dem opp og installere dem selv.
5. VS Code har mange tusen forskjellige utvidelser, i dette faget trenger vi bare de to som er nevnt. Det kan fort bli forvirrende med undøvendig mange, så vi anbefaler ikke å installere flere hvis du ikke vet hva du gjør. Hvis det står flere anbefalte utvidelser enn "Project Templates" og "C/C++" er det bare å ignorere disse. Trykk på det øverste ikonet i menybaren for å komme tilbake til filutforskeren når du er ferdig.
6. Til venstre i VS Code vil du nå se en oversikt over filene i prosjektmappen din. Her skal det ligge en fil ved navn `main.cpp`, en ved navn `Makefile` og en mappe ved navn `.vscode`.
7. `Makefile` og `.vscode` inneholder innstillinger for hvordan VS Code skal compilere og kjøre koden i prosjektet ditt, og å endre dem kan gjøre at ting ikke lenger fungerer som de skal. La dem derfor være som de er! Mappene "Release" og "Debug" er der Code vil legge programmene du lager etter de er compilert, og de vil opprettes ved behov.
8. 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 så du kan redigere den, og den vil dukke opp i listen over "open editors" øverst til venstre.
9. Kompilering i VS Code gjøres ved hjelp av hurtigtasten **Ctrl+Shift+B** eller **Terminal** → **Run Build Task**. Da får du opp en meny med muligheter for å bygge i debug eller release konfigurasjon. Dette kommer vi tilbake til, men nå kan du velge "Build Release Executable".
10. Kjør programmet ditt ved å trykke **Ctrl+F5**, eller ved å gå inn i menyen **Debug** og velge **Start Without Debugging**. Det vil åpnes et terminalvindu med teksten "Hello, World!" som blir værende til du X-er det bort, eller skriver inn en bokstav og trykker enter.



Figur 1: Utvidelsene vi trenger i faget

Oppgave 1.2 - Nye prosjekter i øvingsopplegget

I løpet av øvingsopplegget kommer du til å skrive mye kode i forskjellige filer. For å slippe å kopiere alle de nødvendige filene hver gang du skal lage en ny mappe vil vi bruke en utvidelse til VS Code som heter "Project Templates" som lar oss lage prosjekter fra såkalte maler.

For å lagre en mal som kan gjenbrukes må man:

1. Åpne DT4102_base_project i VS Code
2. Trykk **Ctrl+Shift+P**, du vil da få opp et vindu hvor du kan skrive kommandoer.
3. Start å skrive "Project: Save Project as Template" til du får opp det som et alternativ.
4. Trykk enter og velg hva du vil kalle prosjekttypen din. (default-navnet TDT4102_base_project er også helt greit)

Det neste du må gjøre er å lage en mappe der du ønsker å jobbe med øvingene i løpet av semesteret (for eksempel Dokumenter/cpp/), og deretter en "Øving 0"-mappe inne i den. Når det er gjort åpner du mappen i VS Code på samme måte som vi gjorde tidligere, ved hjelp av **File** → **Open Folder** og så finne fram til Øving 0 mappen du nettopp lagde. Når du har åpnet den kan du igjen trykke **Ctrl+Shift+P** og begynne å skrive "Project: Create Project from Template". Trykk enter og velg malen du har lagd, da vil mappen Øving 0 fylles med alt du trenger for å kompilere koden, og en main.cpp-fil hvor du kan programmere.

Altså, for å lage et nytt prosjekt må man:

1. Lag en ny tom mappe
2. Åpne denne mappen i Code
3. Trykk **Ctrl+Shift+P** og skriv "Project: Create Project from Template"

4. Velg TDT4102_base_project eller det du kalte prosjekt-malen.

For å teste at alt fungerer kan du kopiere koden fra figur 2 inn i `main.cpp` (Skriv over det som ligger der fra før) og teste om du får til å kompilere og kjøre. Hvis alt har gått som det skal vil det nå åpnes et nytt vindu med en rød trekant. Dette er et eksempel på enkel grafikk, som vi kommer til å bruke en del senere i øvingsopplegget.

```
// Example program from PPP, page 415
#include "Graph.h"
#include "Simple_window.h"
int main() {
    using namespace Graph_lib;
    cout << "The New \"Hello, Graphical World!\" message\n";
    Point tl{ 100, 100 };
    Simple_window win{ tl, 600, 400, "Canvas" };

    Polygon poly;
    poly.add(Point{ 300, 200 });
    poly.add(Point{ 350, 100 });
    poly.add(Point{ 400, 200 });
    poly.set_color(Color::red);

    win.attach(poly);
    win.wait_for_button();
}
```

Figur 2: Eksempelprogram med enkel grafikk

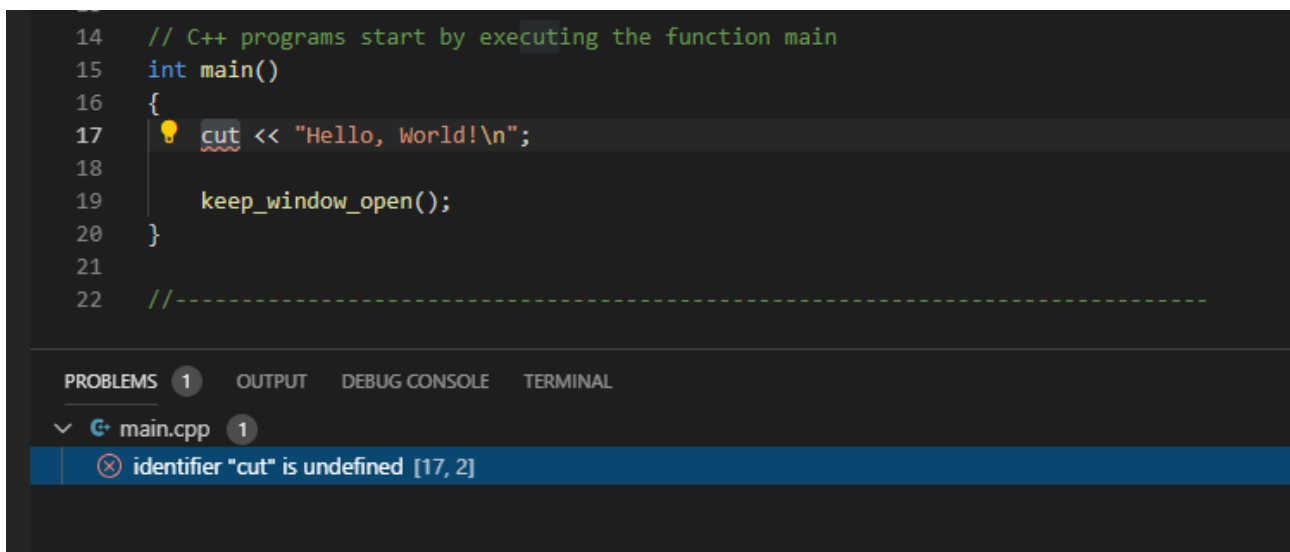
Oppgave 1.3 - Kompilering og feilmeldinger

En god del av tiden du bruker på å gjøre øvinger vil 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.

Du skal nå med vilje «ødelegge» koden din ved å endre på småting, og deretter observere hva slags feilmeldinger du får når du kompilerer. Du kan for eksempel gjøre følgende:

1. Fjern 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 krøllparantesene, skrive `cout` som `cut` osv.
4. Det skal også dukke opp feilmeldinger i Problems fanen på terminalen (se figur 3). 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.



```
14 // C++ programs start by executing the function main
15 int main()
16 {
17     cout << "Hello, World!\n";
18
19     keep_window_open();
20 }
21
22 //-----
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

main.cpp 1

✘ identifier "cout" is undefined [17, 2]

Figur 3: Problems fanen i VSCode

Oppgave 1.4 - Eksempelprogrammer fra forelesningene

Underveis i semesteret vil vi legge ut eksempelprogrammene fra forelesningene utenfor Blackboard. Du vil finne dem ved å gå til <https://github.com/LasseNatvig/cppc2020> i en webleser. Her vil alle eksempler bli lag ut i mapper med beskrivende navn. Finn fram til eksempelet du ønsker å teste, og trykk på det så får du se kildekoden. Herfra kan du velge og kopiere ut koden, for så å lime inn i et prosjekt for å kompilere og kjøre eksemplet.

GitHub er et anerkjent system for samarbeid og deling av kode. Ved å legge eksempelprogrammene her vil studentene hele tiden ha direkte adgang til siste versjon av koden, som ofte vil kunne bli justert like før, under og etter forelesningene. Måten vi foreslår for å laste ned koden er en enkel «nybegynner-metode», siden bruk av Git og GitHub *ikke* er del av pensum i dette faget.

Oppgave: Finn eksempelet som heter "graph_2/main.cpp". Opprett ett nytt prosjekt, kall det for eksempel "Forelesningseksempel" (se oppskriften i slutten av Oppgave 1.3 hvis du trenger en oppfriskning) og kopier innholdet i graph_2/main.cpp-fila inn i din nye main.cpp. Kompiler og kjør programmet. Du kan også endre navn på fila fra main.cpp til f.eks. graph_2.cpp

Oppgave 1.5 - Kompilering fra kommandolinje

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. Vi vil heller ikke benytte *Graph.h* eller *Simple_windows.h*, ettersom det er mer avansert å kompilere "for hånd". Eksempelprogrammet vi skal bruke i denne oppgaven vises i Figur 4.

```
// C++ Hello World without std_lib_facilities
#include <iostream>

int main() {
    std::cout << "Hello World" << std::endl;
    return 0;
}
```

Figur 4: Eksempelprogram uten avhengigheter på Graph.h eller Simple_window.h

For GNU/Linux kan du anse det som en alternativ måte å bygge og kjøre programmer som kan bruke Graph_lib fra læreboken og FLTK.

Du kan gjenbruke byggeoppsettet beskrevet i Oppgave 0. Kopier example-mappen som utgangspunkt for andre øvinger. Øving 3 kan f.eks. opprettes som følger:

```
cp -r example Oving3
```

Deretter kan du bygge og kjøre programmer på samme måte som beskrevet i Oppgave 0.