Norges teknisk–naturvitenskapelige universitet
Institutt for datateknologi og informatikk

TDT4102 Prosedyre- og objektorientert programmering Vår 2020

Øving 0 for Mac

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, kompilere, 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.

Aktuelle kapitler i boka:

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

Oppgave 0 - Installasjon

Installasjonen krever at du installerer to programmer, **Visual Studio Code** for å skrive og redigere kode, og **clang** for å kompilere å kjøre den. I tillegg trengs det noen filer for å kunne bruke kode fra pensumboka. Disse vil ligge i TDT4102mappen på blackboard.

Start med å laste ned tdt4102.zip fra blackboard og pakk den ut et sted du husker, f.eks. kan du lage en mappe for cpp under dokumenter. Merk at om du bruker Safari pakkes filen ut av seg selv. Kopier deretter katalogen som heter `tdt4102` inn i katalogen som heter `/Library` (Evt. `/Bibliotek` om du har norsk versjon). Sørg for at `tdt4102`-mappen har nøyaktig dette navnet, slik at den fullstendige stien blir `/Library/tdt4102`. En måte for å finne `/Library` er å åpne Finder, velge "Gå" oppe i menylinja, velge "Gå til mappe", og så skrive inn `/Bibliotek` (snarvei: `cmd+shift+g` og lime inn «`/Library`» - inkludert skråstrek).

Det er en vanlig feil å legge katalogen i `/Users/<brukernavn>/Library`. For at prosjektet skal fungere *må* katalogen `tdt4102` ligge som beskrevet over. Får du feil om at prosjektet ikke finner f.eks. `std_lib_facilities.h`, er feil plassering av denne mappen årsaken.

VS Code er en teksteditor laget for å skrive og redigere kode i forskjellige programmeringsspråk. Den kan lastes ned og installeres fra https://code.visualstudio.com. Last ned ved siste versjon ved å trykke den store blå "Download for Mac" knappen. Det lastes ned en .zip fil, pakk den ut med å trykke på den. Til slutt er det bare å flytte programmet fra Downloads til Applications, dette kan gjøres i finder ved å dra og slippe.

clang er en kompilator, altså programmet som oversetter koden du skriver i C++ til forståelige instruksjoner for datamaskinen. For å installere den åpner du et terminalvindu, terminalen finner du i **Finder** → **Programmer** (i venstre sidemeny) → **Verktøy** → **Terminal**. Alternativt kan du også søke etter **Terminal** i Launchpad eller Spotlight. I terminalen skriver du inn kommandoen "xcode-select --install". Det vil da dukke opp en tekstboks som spør om du ønsker å installere command line developer tools. Man må velge ja i denne, og deretter godta en lisensavtale fra Apple, så vil kompilatoren lastes ned på maskinen din.

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 med å gå inn i applications-mappen i Finder, finn Visual Studio Code, høyreklikk og velg åpne. Første gang du åpner programmet kan du få en advarsel om at Apple ikke kan verifisere utgiver, her må man trykke ”åpne likevel” så kan det åpnes på vanlig måte senere. Dersom ”åpne likevel” ikke er et alternativ må du trykke ”ok”, og deretter høyreklikke på programmet og velge ”open”. Da skal ”åpne” være et alternativ. Dette trenger du bare å gjøre en gang. Neste gang holder det å klikke på programmet.

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.

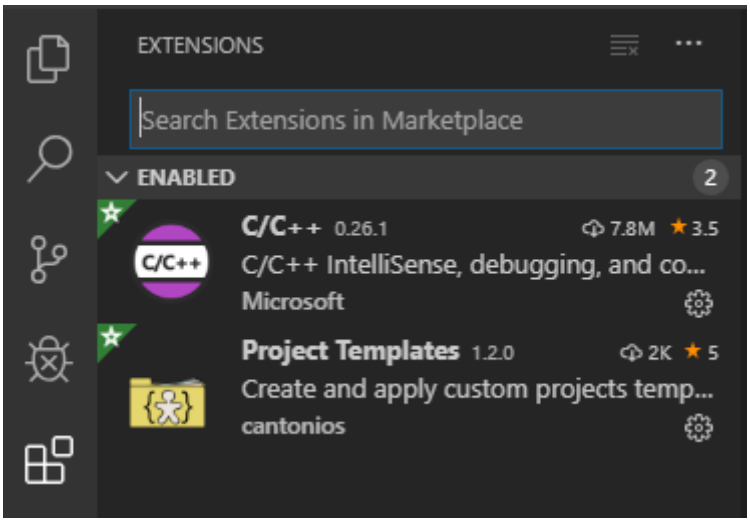
5. VS Code har mange tusen forskjellige utvidelser, i dette faget trenger vi bare de to som er nevnt. Det kan fort bli forvirrende med undøvendig mange, så vi anbefaler ikke å installere flere hvis du ikke vet hva du gjør. Hvis det står flere anbefalte utvidelser enn ”Project Templates” og ’C/C++” er 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 kompilere og kjøre koden i prosjektet ditt, og å endre dem kan gjøre at ting ikke lenger fungere 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 kompilert, 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 `Cmd+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”.



Figur 1: Utvidelsene vi trenger i faget

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.

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 `Cmd+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 `Cmd+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 `Cmd+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();
}
```

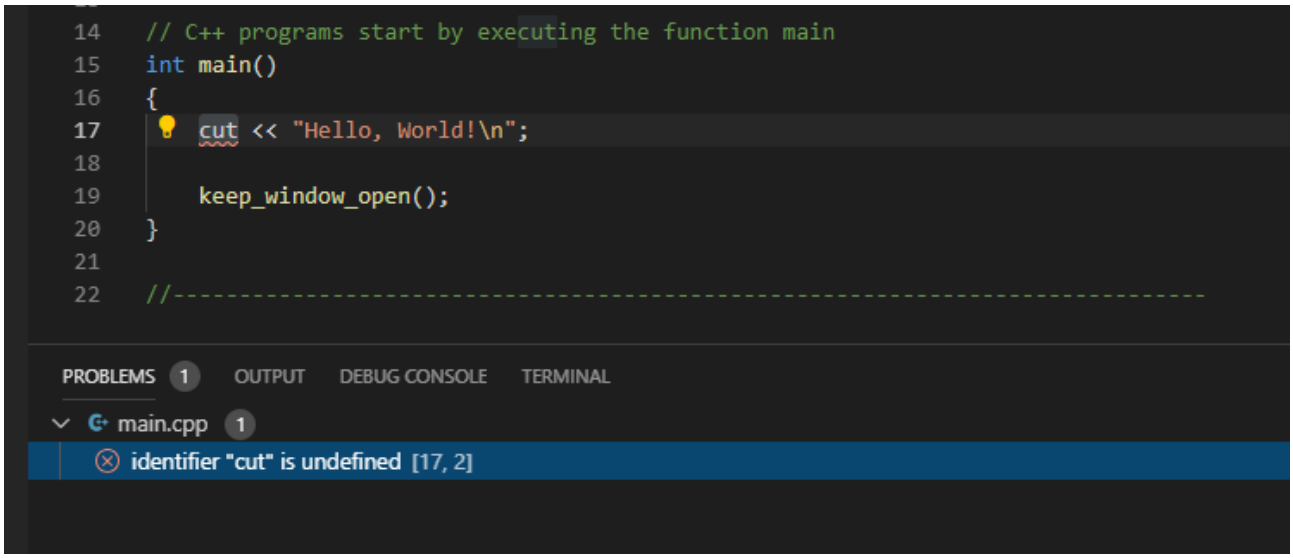
Figur 2: Eksempelprogram med enkel grafikk

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.



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.

1. Start en vanlig teksteditor. TextEdit er en veldig enkel teksteditor som følger med Mac og kan brukes her. For å skrive kode i TextEdit må man, før man begynner å skrive kode, velge **Format** → **Konverter til ren tekst**, slik at det ikke lagres ekstra informasjon om

```
// 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

stil og format i kodefilen. Kopier og lim inn eksemplet fra Figur 4. 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. Terminalen finner du i **Finder** → **Programmer** (i venstre sidemeny) → **Verktøy** → **Terminal**. Alternativt kan du også søke etter **Terminal** i Launchpad eller Spotlight.

3. 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` (**c**hange **d**irectory). Hvis du for eksempel la filen i «Dokumenter»-mappen din, vil du måtte skrive:

   ```
   cd Documents
   ```

   Du befinner deg nå i «Documents»-mappen din. (Merk at dersom du kjører norsk utgave av MacOS vil denne, og tilsvarende mapper som «Pictures», alltid ha engelsk navn, uavhengig av hva den heter når du ser den i Finder.)

4. 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 `/Users/dittbrukernavn/Documents/tdt4102`, vil mappen over være `/Users/dittbrukernavn/Documents`.

5. Skriv kommandoen `ls` (**l**i**s**t) for å se hvilke filer (eller mapper) som ligger i mappen du befinner deg i.

6. Vi skal nå sjekke at «Command Line Tools» er installert. Dette skal i utgangspunktet følge med nyere versjoner av XCode. For å sjekke at alt er installert kan du skrive `clang++` i terminalvinduet. Om «Command Line Tools» *ikke* er installert, får du en feilmelding som sier `Command not found`, samtidig som du får en forespørsel om du vil installere verktøyet. Installer verktøyet. Dersom du får (feil-)meldingen `clang: error: no input files` er alt installert.

7. Har du funnet fram til mappen der `HelloWorld.cpp` ligger, er du klar til å kompilere programmet. Kompilatoren som følger med Xcode heter `clang++`, og du bruker denne til å kompilere programmet ditt ved å skrive:

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

8. 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.

9. Rediger teksten i `HelloWorld.cpp` slik at programmet skriver ut noe annet (husk å ha med hermetegnene rundt teksten).

10. Kompiler og kjør på nytt.