Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre- og objektorientert programmering Vår 2017

Øving 0 for Mac

Frist: Ingen (frivillig øving)

Mål for denne øvingen:

- Bli kjent med programmeringsverktøy
- Lage et første program kun med teksteditor og kompilator
- Lage et første program med Xcode

Denne øvingen er mer en veiledning i hvordan å lage et program. Målet er å veilede deg gjennom prosessen å lage et første program. Hvis du gjennomfører denne øvingen og blir fortrolig med skriving, kompilering og kjøring av egne program vil du komme fortere i gang med de obligatoriske øvingene. I denne øvingen går første del ut på å kode ved hjelp av en vanlig teksteditor og kompilere fra kommandolinjen.

NB: Denne øvingen er utformet på basis av Xcode 6, 7 og 8. Dersom du bruker Xcode 4 eller 5 skal det meste være likt. Dersom du bruker Xcode 3 eller tidligere anbefales det sterkt at du oppgraderer til en nyere versjon, eller benytter Windows-maskinene på datasal for å løse oppgavene. Du kan sjekke hvilken versjon av Xcode du kjører ved å velge **Xcode** → **About Xcode** i menylinjen.

Aktuelle kapitler i boka:

- Kap. 1 (1.1 og deler av kap 1.3) i Absolute C++ (Walter Savitch)

Oppgave 1 – Bli kjent med kompilatoren

Skriving av kildekode og kompilering kan i prinsippet gjøres med enkle verktøy. En helt vanlig teksteditor[1] er alt du trenger for å skrive kode, og kompilering kan gjøres ved å kjøre kompilatoren fra terminalen.

Før du gjør denne oppgaven, skal du installere Xcode. Dette kan gjøres fra Mac App Store. Dersom du installerte Xcode for en stund siden, anbefales det sterkt at du oppgraderer til siste versjon av programvaren. Åpne XCode minst en gang før du fortsetter i tilfelle du må godta en lisensavtale (dette kan spare deg bry senere i øvingen).

I denne oppgaven kan du f.eks. bruke kildekoden som er vist nedenfor dette avsnittet eller du kan skrive av andre eksempler i boken. Kopier og lim det som er mellom strekene.

```
// Dette er et helt enkelt program som du kan kopiere og bruke i
// denne oevingen.
#include <iostream>

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

Oppgave 1.1 - Lagring av egen kildekode

1. Start en vanlig teksteditor. Kopier og lim inn eksemplet over. Opprett en ny katalog på hjemmeområdet ditt 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. Det er vanlig konvensjon at kildekodefiler for C++ har filtypenavnet (delen av filnavnet etter punktum) `cpp`. Ved å bruke riktig filtypenavn oppnår du at mange verktøy automatisk skjønner at filen inneholder C++-kode. Hvis du bruker et annet filtypenavn kan det hende at du ikke får kompilert koden.

Oppgave 1.2 - Kompilering fra kommandolinjen på Mac

1. Start opp et terminalvindu. Terminalen finner du i **Finder** → **Programmer** (i venstre sidemeny) → **Verktøy** → **Terminal**. (Du kan også søke etter **Terminal** i Launchpad eller Spotlight.)

[1] Her kan du blant annet bruke TextWrangler (http://www.barebones.com/products/textwrangler/) eller TextEdit, som følger med OS X. Dersom du skal bruke TextEdit må du velge `Format` → `Konverter til ren tekst` fra menylinjen før du begynner å skrive.

2. 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 OS X vil denne, og tilsvarende mapper som «Pictures», alltid ha engelsk navn, uavhengig av hva den heter når du ser den i Finder.)

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

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

5. 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.

6. 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
   ```

7. 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`.

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

9. Kompiler og kjør på nytt.

Oppgave 1.3 - Hva skjer hvis koden min er feil?

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 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 vanskeligere 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

   ```
   std::cout << "Hello World!" << std::endl;
   ```

2. Lagre filen, kompiler og sjekk feilmeldingen du nå får. Forstår du feilmeldingen?
3. Husk å rette opp filen igjen før du går videre.
4. Introduser andre feil og les feilemeldingene som kompilatoren gir. Du kan f.eks. prøve å slette en av krøllparantesene, skrive `cout` som `cut` osv.
5. Les feilmeldingene du får og rett opp slik at filen din blir riktig igjen etterpå.

Noen tips til å forstå feilmeldinger fra kompilatoren:

- En feilmelding fra kompilatorene `g++` og `clang++` er vanligvis på formen

  ```
  kildekode.cpp:7:5: error: 'st' has not been declared st::cout « "Hello World!"
  « std::endl;
  ```

  Første del, `kildekode.cpp:8:5`, inneholder filnavnet til filen der feilen befinner seg, samt to tall. Det første tallet er linjenummeret der kompilatoren tror feilen befinner seg, mens det andre tallet er hvor mange tegn inn på linjen den tror feilen ligger.

- Kompilatoren har ikke alltid rett i på hvilken linje feilen ligger. Ofte er det snakk om «følgefeil» fra en tidligere linje. Når du får opp mange feilmeldinger på én gang, lønner det seg ofte å se på de første feilmeldingene.
- Etter linje- og tegnnummeret står det hvilken *type* feilmelding det er snakk om. Dette kan enten være «error», «warning» eller «note». En «error» vil avbryte kompileringen uten at kompilatoren produserer en programfil. Dersom du får en «warning» eller en «note» vil kompileringen fortsette, men disse indikerer at koden din gjør noe du antakelig ikke vil at den skal gjøre. Husk at selv om koden din kompilerer, er det ikke sikkert at det resulterende programmet *virker* slik du vil! Får du en «warning» eller «note» når du kompilerer, bør du ta en ekstra kikk på koden din før du programmerer videre.
- Deretter følger beskrivelsen av feilen, ideelt uttrykt på en forståelig og ikke altfor generell måte. For mindre feilmeldinger som er grunnet i syntaksfeil og andre småfeil vil beskrivelsen vanligvis være konsis og forståelig, men når man begynner å bruke mer avanserte deler av C++ kan den være vag eller til og med direkte misledende. Da kan det være lurere å gå grundig gjennom koden i nærheten av der feilen oppstod og se om man ser en feil.
- Etter dette kommer et utdrag av koden der kompilatoren tror feilen ligger.

Feilmeldinger fra C++-kompilatoren kan være vanskelig å forstå, særlig når de er misvisende. Ikke sitt og dra deg i håret av den grunn! Både studassen din, emnets diskusjonsforum på It's learning og nettressurser som stackoverflow.com er uvurderlige hjelpemidler når det ser aller mest håpløst ut.

Oppgave 2 - Programmering med Xcode

I denne oppgaven skal vi lære å skrive og kjøre enkle programmer i Xcode. Dersom du ikke enda har installert Xcode, kan du gjøre dette i Mac App Store.

Oppgave 2.1 - Start opp Xcode

1. Start Xcode, som du finner i «Programmer»-mappen din.
2. Velg «Create a new XCode project».
3. Under «macOS» (evt. «OS X») og «Application» velg «Command Line Tool».
4. Veiviseren ber deg nå fylle ut en rekke felter. Her skal du skrive inn et produktnavn, f.eks. «Oving0» eller «HelloWorld». Organisasjonsnavn og organisasjonsidentifikator skal også fylles ut – her betyr det ikke noe hva du skriver. Pass på å velge **C++** fra språkmenyen. Lagre prosjektet i en egnet mappe. Du kan fjerne avkrysningen for «Create Git Repository» dersom du ikke vet hva dette er, eller ikke ønsker å bruke verktøyet.
5. Xcode har nå opprettet et prosjekt og en standard kildekodefil kalt `main.cpp`. Disse skal vi ikke benytte oss av. Høyreklikk på `main.cpp` i sidelinjen, og velg **Delete**, og deretter **Move to Trash**.
6. I sidelinjen har du nå et prosjekt med navnet du valgte i veiviseren. Dette prosjektet har to mapper under seg: en med navnet «Products», og en med samme navn som prosjektet. Høyreklikk på sistnevnte mappe, og velg **New File...**.
7. Fra menyen du nå får opp skal du velge **macOS** (evt. **OS X**) → **Source** → **C++ File**. Trykk **Next**, og gi filen navnet `HelloWorld.cpp`. Trykk **Create**.
8. Du skal nå ha denne filen åpen i Xcode. For ordens skyld: det ble laget *to* filer nå, men ignorer .h- eller .hpp-filen enn så lenge.
9. Kopier inn kildekoden fra oppgave 1 – erstatt alt som ligger i `HelloWorld.cpp` fra før.
10. Sjekk i Finder hvor filen `HelloWorld.cpp` ble lagret. De andre filene som Xcode oppretter trenger du ikke bry deg om.
11. Kompiler programmet ditt uten å kjøre det ved å trykke **Product** → **Build** i menylinjen (hurtigtast: `CMD+B`).
12. Kompilering etterfulgt av kjøring av programmet i Xcode gjøres ved å trykke på «play»-knappen) øverst til venstre i Xcode (hurtigtast: `CMD+R`). Gjør dette nå.
13. Du ser at programmet kjører og skriver ut teksten «Hello World!» i bunnen av program-vinduet. Gratulerer!

NB: Når du kompilerer programmer i Xcode vil det lages en programfil, som havner på et litt annet sted enn prosjektet ditt. For å finne denne, åpner du mappen «Products» i sidelinjen. Inni denne mappen ligger det et objekt med samme navn som prosjektet ditt og med et svart «terminalikon». Høyreklikk på dette objektet og velg **Show in Finder** for å finne ut hvor det ligger.

Oppgave 2.2 - Kompileringsfeil i Xcode

- Prøv å introdusere de samme feilene som du testet i Oppgave 1.3, og se hvor feilmeldingene blir skrevet ut. (Hint: Klikk på varseltrekanten i sidelinjen på venstre side.)
- Prøv å klikke på linjer i feilmeldingen og sjekk om Xcode merker linjene i koden din der den tror feilen ligger.

Oppgave 3 - Kompilering på UNIX (for interesserte)

Siden OS X er en UNIX-variant, er dette egentlig allerede dekket av det vi gjorde i Oppgave 1, men noen småting er verdt å nevne:

- I Oppgave 1 endte vi opp med en `a.out`-fil. Navnet på denne filen kan velges ved å legge til `-o filnavn` på slutten av kommandoen. Vi kan for eksempel skrive følgende:

  ```
  clang++ HelloWorld.cpp -o HelloWorld
  ```

  Denne kommandoen vil produsere en fil med navn `HelloWorld`, som da kan kjøres med kommandoen `./HelloWorld`.
- I Oppgave 1 brukte vi clang++, siden denne gir hakket mer leselige feilmeldinger, men denne vil ikke alltid være tilgjengelig. Prøv i såfall `g++` eller `c++` som alternativer.

Hvis du vil prøve å kompilere kode på en av NTNU sine servere kan du gjøre følgende:

1. Logg deg på NTNUs Linux-server for studenter ved hjelp av SSH. På Mac kan du gjøre dette ved å åpne terminalen (**Applications** → **Utilities** → **Terminal**) og skrive `ssh login.stud.ntnu.no` for å koble til serveren. Skriv deretter inn ditt vanlige NTNU-brukernavn og -passord, begge etterfulgt av `Enter`. Merk at det av sikkerhetshensyn ikke vil vises tekst når du skriver inn passordet ditt.
2. Lag deg en ny fil ved navn `HelloWorld.cpp` på et egnet sted, for eksempel med teksteditoren `nano`. For å lage en ny fil ved hjelp av `nano` (eller redigere en eksisterende) skriver du

   ```
   nano filnavn
   ```

   Når du er kommet inn i `nano` bruker du hurtigtastene som vises på bunnen av skjermen. Tegnet `^` betyr «ctrl», så `Ctrl+O` (`^O`) utfører kommandoen «WriteOut», som er et annet ord for å lagre.
3. Alternativt: Hvis du i Oppgave 1 lagret `HelloWorld.cpp` på NTNU-hjemmeområdet ditt, kan du i stedet navigere deg fram til den ved hjelp av `cd` og gjenbruke den.
4. NTNU-serveren har to C++-kompilatorer installert: `clang++` og `g++`. Hvilken av disse du bruker er vilkårlig, men `clang++` er kjent for å gi mer forståelige feilmeldinger. Kompiler `HelloWorld.cpp` ved å skrive

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

eller

```
g++ HelloWorld.cpp
```

5. Sjekk hva programfilen som ble laget heter og start denne med kommandoen `./filnavn` (merk punktumet først i kommandoen).

6. Prøv også her å introdusere feil i koden og se på feilmeldingene du får når du kompilerer filen.