Introduksjon til CMake for C++ prosjekter

CMake er et mye brukt verktøy for å handtere platformuavhengig oppsett av C++ prosjekter. Dette verktøyet kan blant annet hjelpe oss å finne programvarebiblioteker på ulike plattformer, og hjelpe oss med å kompilere C++ filer til kjørbare filer.

Et enkelt C++ program i filen `main.cpp`:

```
#include <iostream>

int main() {
  std::cout << "hello" << std::endl;
}
```

kan kompileres i en terminal på Unix-lignende systemer som MacOS og Linux:

```
g++ main.cpp
```

Dette resulterer i en kjørbar fil som heter `o.out` i samme katalog.

Når vi bruker CMake må vi først ha en `CMakeLists.txt` fil, for eksempel opprettet av juCi++:

```
cmake_minimum_required(VERSION 3.1)

project(hello)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++1y -Wall -Wextra")

add_executable(hello main.cpp)
```

Her settes i tillegg flaggene `-std=c++1y -Wall -Wextra` som aktiverer c++14 standarden og ekstra advarsler (`-Wall` og `-Wextra`). Med `add_executable()` sier vi at `main.cpp` skal kompileres til den kjørbare filen `hello`.

En kan kjøre cmake i terminalen mot katalogen der `CMakeLists.txt` filen ligger, og den kjørbare filen `hello` blir opprettet:

```
mkdir build
cd build
cmake .. # Prepare project for compilcation
make     # Compile project and make executable
./hello  # Run hello executable
```

Det er vanlig å gjøre dette i en egen byggekatalog `build` siden byggeprosessen kan opprette flere filer du ikke ønsker sammen med kildefilene.

Merk at IDE'er som juCi++ gjør dette for deg når du velger å kjøre et prosjekt.

Når du kjører `make` blir en kommando som tilsvarer følgende kjørt:

```
# in build folder:
g++ -std=c++1y -Wall -Wextra ../main.cpp -o hello
```

Flere kjørbare filer

Om du ønsker to eller flere kjørbare filer i samme prosjekt kan du legge til flere `add_executable()`-kall i CMakeLists.txt filen. For eksempel med disse kildefilene:

```
// hello1.cpp
#include <iostream>

int main() {
  std::cout << "1" << std::endl;
}
```

```
// hello2.cpp
#include <iostream>

int main() {
  std::cout << "2" << std::endl;
}
```

kan du lage to kjørbare filer `hello1` og `hello2` med følgende CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.1)

project(hello)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++1y -Wall -Wextra")

add_executable(hello1 hello1.cpp)
add_executable(hello2 hello2.cpp)
```

og kjøre `make` igjen i `build`-katalogen.

Når du velger *Compile and Run* i *Project* menyen i juCi++ kjøres den kjørbare filen som er knyttet opp mot kildefilen du jobber med i IDE'en. Du kan også velge *Start/Continue* i *Debug* menyen om du ønsker å debugge programmet ved å sette *breakpoints* og se på verdier i variabler når debuggeren stopper ved et *breakpoint*.

Kompilering av flere kildefiler til en kjørbar fil

I C++ er det vanlig å dele opp kildekoden i flere filer. Du trenger ikke gjøre det i dette kurset med mindre øvingsteksten ber om det, men det kan likevel være nyttig å vite hvordan dette kan gjøres med CMake.

I eksempelet under er kildekoden delt opp i tre filer `main.cpp`, `answer.hpp` og `answer.cpp`:

```
// main.cpp
#include "answer.hpp"
#include <iostream>

int main() {
  std::cout << "2" << std::endl;
}
```

I `main.cpp` ligger `main()`-funksjonen som kjøres først i den kjørbare filen som til slutt skal lages.

```
// answer.hpp
#pragma once

int answer();
```

I `answer.hpp` ligger signaturen til en funksjon `answer()` slik at kompilatoren kan sjekke at `answer()` blir brukt riktig i for eksempel `main.cpp`-filen. Slike signaturer legges i *header* filer som for eksempel slutter med `.hpp`.

Selve implementasjonen av `answer()`-funksjonen kan legges i en egen `cpp`-fil som her heter `answer.cpp`:

```
// answer.cpp
#include "answer.hpp"

int answer() {
  return 42;
}
```

For å kompilere de to kildefilene `main.cpp` og `answer.cpp` til en kjørbar fil kan vi bruke følgende CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.1)

project(hello)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++1y -Wall -Wextra")

add_executable(hello main.cpp answer.cpp)
```

Begge kildefilene `main.cpp` og `answer.cpp` er her lagt til i `add_executable()`-kallet.