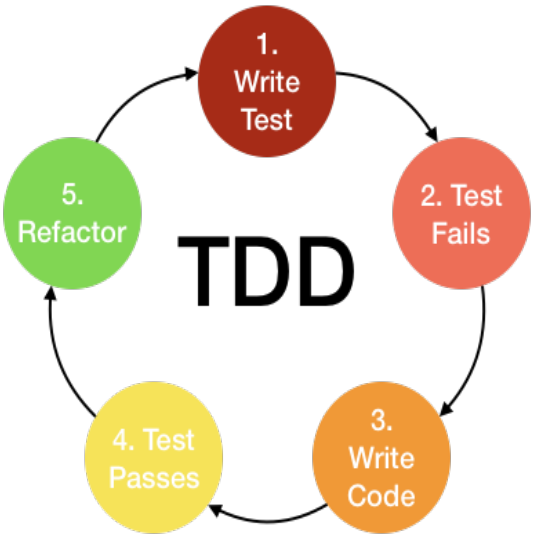
Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2019-08-29

TDD och Catch - En liten guide

Detta är en kort och övergripande guide om testdriven utveckling (Test-driven development, TDD) och om Catch, ett testramverk för C++.

Vad är TDD?

Testdriven utveckling är en metod för att se till att koden uppfyller de krav vi har på den. Tillvägagångssättet är att först skriva tester, se testen misslyckas (eftersom koden för dem inte existerar), skriva kod, se testen lyckas och sedan utföra förbättringar till koden. Denna process repeteras kontinuerligt.



Styrkan är att tester skapas för all kod som skrivs (uppnår en hög testtäckning), bra kvalité på programkoden och det skapas en dokumentation.

Vad är Catch?

Catch är ett testramverk för C++. Det innebär att vi får en rad verktyg för att skriva ett strukturerat testprogram, som kan indikera ifall ett test går igenom eller inte går igenom.
Catch är ett stort ramverk och innehåller många olika verktyg för att skapa testprogram. I denna guide kommer vi att lyfta några få av dem som krävs för att fullt fungerande testprogram. Om du vill fördjupa dig kan vi rekommendera dokumentationen för Catch.

Hur skriver vi ett test?

test_program.cc:

```
//Detta säger åt Catch att skapa en main() - gör bara detta i en .cc-fil.
#define CATCH_CONFIG_MAIN


//Här inkluderar vi alla de verktyg vi använder i Catch.
//Detta biblioteket är stort och tar tid att kompilera.
#include "catch.hpp"

//Här kan vi också inkludera .h-filerna för de funktioner,
//klasser, m.m., som vi vill testa.

//Vanligtvis brukar vi inte definiera de program som vi vill testa
//i samma fil som testprogrammet. Detta är bara för att ge ett
//fungerande exempel av en funktion som vi kan testa.
int sum(int a, int b)
{
    return a + b;
}

//Ett TEST_CASE kan innehålla flera tester som hänger ihop,
//exempelvis flera tester för en funktion.
TEST_CASE( "Sum of ints") {

   //Här kan vi deklarera variabler som vi vill använda i tester.
   int x{3};

   //Följande är olika test.

   //Blir villkoret i CHECK(villkor) sant kommer testet indikeras som godkänt.
   //Oavsett utgång fortsätter testprogrammet.
   CHECK(sum(2, x) == 5);

   //Blir villkoret i CHECK_FALSE(villkor) falskt kommer testet indikeras som
   //godkänt. Oavsett utgång fortsätter testprogrammet.
   CHECK_FALSE(sum(2, x) == 6);

   //Blir villkoret i REQUIRE(villkor) sant kommer testet indikeras som godkänt.
   //Om testet blir ej godkänt avbryts testprogrammet.
   //Tänk på att två villkor inte alltid önskvärt
   REQUIRE((sum(2, x) == 5 && x == 3));

   //Blir villkoret i REQUIRE_FALSE(villkor) falskt kommer testet indikeras som
   //godkänt. Om testet blir ej godkänt avbryts testprogrammet.
   REQUIRE_FALSE((sum(2, x) == 5 || x == 3));
}
```

Testprogrammet kompileras precis som när vi kompilerar andra huvudprogram (de som innehåller int main()) och det skapas en körbar fil. När filen körs kommer utskriften ange vilka test som gått igenom och vilka som inte gått igenom. När det inte går igenom så står det vilket/vilka som inte gått igenom och på vilken rad du hittar dem.

Det bör också lyftas att det finns verktyg för att testa om undantag har kastats. Detta kommer bli relevant i den sista laborationen (Listan).

Snabbare kompilering med catch

Det tar lång tid att kompilera testprogrammet, då Catch.hpp (som är en enorm fil) behöver inkluderas och kompileras varje gång. Detta kan vi undvika genom att göra följande:

1. Skapa en fil test_main.cc (Kommer färdig med givna filer till laboration Klockslag)
2. I den ligger följande kod:

```
#define CATCH_CONFIG_RUNNER
#include "catch.hpp"
int main(int argc, char* argv[])
{
  return Catch::Session{}.run(argc, argv);
}
```

3. Kompilera test_main.cc till en förkompilerad programmodul (.o-fil) med kompilatorflaggan -c:
   `g++ -std=c++17 -c test\_main.cc`
   Detta gör att vi inte behöver kompilera catch-biblioteket igen, och det är okej eftersom vi inte ändrar något i det givna biblioteket.
4. Ta bort raden #define CATCH_CONFIG_MAIN från test_program.cc.
5. Länka in .o-filen vid kompilering av de andra filerna:
   `g++ -std=c++17 test_main.o test_program.cc <eventuellt andra .cc-filen>`