TDP004 - Objektorienterad programmering

Grunder i C++

Rasmus Jonsson & Pontus Haglund

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

1 Kursintroduktion
2 Mål med föreläsningen
3 Grunder i C++
4 Variabler
5 Mer IO

Kursintroduktion

Personal

Se kurssidan

Kursintroduktion

Kursmål

- Redgöra ingående för begrepp, designprinciper, metoder och tekniker som används inom objektorienterad programmering.
- Använda objektorienterad programdesign för att lösa problem på ett korrekt och lämpligt sätt.
- Konstruera ett objektorienterat program som löser ett realistiskt mindre problem.

Kursintroduktion

Ändringar - Evaliuate

- Berätta om ändringar

Kursintroduktion

- Labbserie
  - Registrera er på WebReg (deadline idag)!
  - 7 laborationer (0-6)
  - Redovisning, kodinlämning och komplettering
  - Laborationerna kräver mycket tid (första mjuka deadline redan imorgon)
  - Hård deadline: sista passet
  - Ungefär ett labborationspass per dag
  - Mjuka deadlines ger bonus på tentan (5 min)

Kursintroduktion

Kursmoment, Tentamen

- Dugga: (se schema kurssidan)
- Tenta: (se schema kurssidan)

Kursintroduktion

Kursmoment, Tentamen

- Dugga: (se schema kurssidan)
  - 4 Uppgifter
  - Klarar du duggan behöver du inte skriva tentat
  - Klarar du en uppgift på duggan kan den tillgodoräknas till tentan för betyg 3
- Tenta: (se schema kurssidan)

Kursintroduktion

Kursmoment, Tentamen

- Dugga: (se schema kurssidan)
- Tenta: (se schema kurssidan)
  - 5 Uppgifter
  - Betyg ges av antalet lösta uppgifter under bestämd tid
  - Bonustid räknas enadst för högre betyg
  - Tillgodoräknande av uppgifter från duggan räknas endast till betyg 3

Kursintroduktion

Kursmoment, Poäng

- Registrera er så snart som möjligt i WebReg
- 33 poäng behövs för godkänt i kursen (65 är max)
- 44 poäng ger 5 min bonus på tentan
- 55 poäng ger 10 min bonus på tentan

Kursintroduktion

Kursmoment, Poäng

- Lektioner
  - Förberedelseuppgifter (1 poäng per löst uppgift)
  - Finns 13 uppgifter för varje lektion
  - Uppgifter redovisas på tavlan
  - Maximalt 8 poäng per lektion
  - 4 lektioner under kursens gång
- Dojo

Kursintroduktion

Kursmoment, Poäng

- Lektioner
- Dojo
  - Första timmen som lektionen, varje uppgift ger nu 2 poäng
  - Andra timmen löses ett dojoproblem
  - Poäng tilldelas gruppen av assitenten, maximalt 11 poäng per dojo
  - 3 dojos under kursens gång

Kursintroduktion

TDP005

- Högt tempo i kursen
- Går endast under November
- förväntad arbetstid är 40 timmar i veckan
- C++ kan vara svårt, ställ alltid frågor!
- Ha källkritik mot info på nätet

Kursintroduktion

TDP004 - några kommentarer

- Föreläsningsslidsen är inte heltäckande i sig(kom på föreläsningarna)
- Publicerade versioner är något annorlunda från de vi använder under föreläsningarna
- Läsinstruktioner är inte specifika för en bok, slå upp koncept i index
- Läs lite innan föreläsningarna

1 Kursintroduktion
2 Mål med föreläsningen
3 Grunder i C++
4 Variabler
5 Mer IO

Mål med föreläsningen

Efter föreläsningen skall studenten kunna:

- Skriva ett första C++ program.
- Veta vad de olika delarna av main funktionen gör.
- Veta vad include innebär.
- Kompilera ett C++ program och köra detta.
- Hantera `cout` och `cin` strömmar samt hantera några vanliga ström opperatorer.
- Definiera och använda variabler

LINKÖPING UNIVERSITY

Grunder i C++

Ett första C++-program

program.cc

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Ett litet C++-program" << endl;
    return 0;
}
```

Grunder i C++

Ett första C++-program

program.cc

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Ett litet C++-program" << endl;
    return 0;
}
```

```
$ g++ program.cc
$ ./a.out
Ett litet C++-program
```

Grunder i C++

Ett första C++-program

- `main` är start punkten för programmet
- Slutet på en rad markeras med `;`
- `cout` skriver text till konsolen
- `return 0` säger åt programmet att avsluta
- `#include <iostream>` och `using namespace std` låter oss använda `cout`

Grunder i C++

Komplilering

- En kompilator är ett speciellt program
- Den översätter *källkodsfiler* till *exikverbara filer*
- En exikverbar fil innehåller maskinkod som datorn kan köra
- Det finns många olika C++-kompilatorer men i denna kurs kommer vi att använda g++

Grunder i C++

Komplilering

- För att kunna Komplilera `program.cc` kör följande kommando i terminalen `g++ program.cc`
- Om inget skrivs ut så har kompileringen lyckats
- Detta kommer att skapa en exekverbar fil som heter `a.out`
- För att köra ditt program skriver du `./a.out` i terminalen

Grunder i C++

Kompileringsflagor

```
g++ -Wall -Wextra -Wpedantic -Werror -std=c++17 program.cc
```

Grunder i C++

Kompileringsflagor

- Flaggor kan användas för att använda eller konfigurera olika funktioner i kompilatorn
- `-Wall -Wextra -Wpedantic` kommer att lägga till mer varningar från kompilatorn vilket hjälper oss att bli bättre programmerare
- `-Werror` kommer att göra om alla varningar till fel, detta betyder att programmet inte kommer att kompileras om du har en varning.
- `-std=c++17`, `-std=c++14` eller `-std=c++11` låter oss lägga välja C++ version, standard bör vara `C++17`
- **Rekommendation:** Skapa ett alias

Grunder i C++

Skapa alias

Vi rekommenderar att ni har två separata alias, dessa kommer att finnas på tentan

```
w++17='g++ -std=c++17 -Wall -Wextra -Wpedantic'
e++17='g++ -std=c++17 -Wall -Wextra -Wpedantic -Werror'
```

C++ basics

Skapa alias

- Detta låter oss använd `w++17` och `e++17`som vår kompilator som automatiskt har alla flaggor
- **Exempel:**

  `w++17 program.cc`

  kommer att vara samma som

  `g++ -std=c++17 -Wall -Wextra -Wpedantic program.cc`

1 Kursintroduktion
2 Mål med föreläsningen
3 Grunder i C++
4 Variabler
5 Mer IO

Variabler

Grunder

```
int main()
{
  int i;
  i = 3;

  float f(1.2);
  double d = 2.3;
  char c{'d'};
}
```

Variabler

Grunder

```
int main()
{
  int i{3};
  float f{1.2}
  double d{2.3};
  char c{'d'};
}
```

Variabler

Grunder

```
int main()
{
  int i{3};
  float f{1.2}
  double d{2.3};
  char c{'d'};
}
```

```
int main()
{
  int x{3};
  cout << "x = "
       << x << endl;
}
```

Variabler

Grunder

- Variabler har olika typer: heltal, decimal tal, karaktärer etc.
- Typer definierar vilka värden som kan sparas i variablerna.
- En variable kan aldrig byta typ.
- De flesta variablerna kan skrivas till `cout`

Variabler

string

```
#include <iostream>
#include <string>

using namespace std;
int main()
{
  string str {"hello"};
  cout << str << '\n'
       << str.size() << '\n'
       << str.front() << endl;
  return 0;
}
```

Variabler

string

```
#include <iostream>
#include <string>

using namespace std;
int main()
{
  string str {"hello"};
  cout << str << '\n'
       << str.size() << '\n'
       << str.front() << endl;
  return 0;
}
```

```
$ ./a.out
hello
5
h
```

Variabler

`string`

- `string` är definierad `#include <string>`
- För att komma åt `#include <string>` använd
  `using namespace std;`
- Representerar text (en sekvens av karaktärer).
- Har en massa funktionalitet inbyggd som inte andra datatyper har.

Variabler

`const`

```
int x{5};
x = 7;

int const y{7};
y = 9; // kommer inte att kompilera

const int z{9};
```

Variabler

`const`

- Variabler kan bli markerade som konstanter med nyckelordet `const`
- Du kan placera `const` före eller efter datatypen.
- Vi rekommenderar att ni placerar den efter datatypen, varför kommer att förklaras i en senare föreläsning.

1 Kursintroduktion
2 Mål med föreläsningen
3 Grunder i C++
4 Variabler
5 Mer IO

LINKÖPING UNIVERSITY

Mer IO

Inläsning

program.cc

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
  cout << "Skriv in ett ord och ett numer: "
       << flush; // Detta för att flusha och få cursorn
                 // att stanna kvar på samma rad
  string word{};
  int number{};
  char letter{};
  cin >> word;
  cin >> number;
  cin >> letter;
  return 0;
}
```

Mer IO

Inläsning

- För att läsa in värden till variabler används `cin`.
- Det finns en buffer som läsningen kommer läsas från i första hand.
- Prompten kommer endast att dyka upp i terminalen om och endast om buffern är tom.
- `cin` kommer att läsa från denna buffern tills den är tom.
- De flesta datatyperna kan läsas från `cin`.

Mer IO

Inläsning

- Blanksteg (whitespace) är sådanna karaktärer som space, nyrad (newline) och andra karaktärer som inte har ett synligt tecken.
- När vi läser från buffern kommer `cin` att ignorera alla blanksteg fram tills den når ett icke-blankstegstecken.
- Om `cin` hittar ett blanksteg (eller en karaktär som inte passar datatypen) när den läser avslutas inläsningen.

Mer IO

getline

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string line{};
  cout << "Skriv in din rad här: ";
  getline(cin, line);
  cout << "Din rad var följande: "
       << line << endl;
}
```

Mer IO

`getline`

- Vi använder `getline` för att läsa hela rader istället för enstaka ord.
- Vi ger den cin `cin` och en sträng variabel vi vill lagra raden i.
- Den kommer att läsa tills vi når ett nyradstecken (`\n`) och sedan lagra den i variabeln.
- Sedan tar den bort nyradstecknet från buffern.

Mer IO

ignore

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string line{};
  getline(cin, line);
  cin.ignore(1000, '\n');
}
```

Mer IO

`ignore`

- `cin.ignore` kommer att ta bort saker från strömmen
- vi ger `cin.ignore` två saker; hur många karaktärer den ska ignorera och vilken karaktär *delimiter* ska vara.
- `cin.ignore` antingen den specificerade mängden tecken (exemplet ovan har 1000 tecken) eller tills den hittar *delimiter* (i exemplets fall `\n`) karaktären, den väljer den som händer först.

Mer IO

Modifierare

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
  cout << setw(10) << "Lite"
       << '|' << right
       << setw(10) << "mer"
       << endl;
  cout << setfill('-')
       << setw(21) << "C++ kod"
       << endl;
}
```

Mer IO

Modifierare

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
  cout << setw(10) << "Lite"
       << '|' << right
       << setw(10) << "mer"
       << endl;
  cout << setfill('-')
       << setw(21) << "C++ kod"
       << endl;
}
```

```
$ ./a.out
Lite       |       mer
----------C++ kod
```

Mer IO

Modifierare

- För mer avancerad formatering kan vi inkludera
  `#include <iomanip>`
- `setw(10)` kommer att garantera att nästa värde ska skrivas ut skrivs ut *med åtminstone* 10 tecken.
- Om det utskrivna värdet är mindre än 10 karaktärer så kommer resten av platserna bli utfyllda med blanksteg(space) som.

Mer IO

Modifierare

- Vi kan modifiera egenskaperna hos `setw`
- `right` placerar utfylnads blanksteg före värdet istället för efter.
- `setfill` kommer att byta ut blanksteg mot någon annan karaktär.
- Både `right` och `setfill` är så kallade *sticky*, med detta menas att deras effekt kvarstår fram tills de aktivt ändras.

Mer IO

Modifierare

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
  double pi{3.141592654};
  double zeros{0.010000};

  cout << pi << '\n'
       << setprecision(8) << pi << '\n'
       << endl;
  cout << zeros << '\n'
       << fixed << zeros
       << endl;
}
```

Mer IO

Modifierare

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
  double pi{3.141592654};
  double zeros{0.010000};

  cout << pi << '\n'
       << setprecision(8) << pi << '\n'
       << endl;
  cout << zeros << '\n'
       << fixed << zeros
       << endl;
}
```

```
$ ./a.out
3.14159
3.1415926

0.01
0.010000
```

Mer IO

Modifierare

- För att bestämma precisionen på decimal tal använder vi `setprecision`.
- `setprecision` tar in ett argument vilket bestämmer antalet decimaler som skall skrivas ut.
- `fixed` skriver ut decimalerna precis som de var instantierade.
- Det finns många fler manipulatorer för detta i `#include <iomanip>`.
- Observera att dessa är också sticky.

Mer IO

Modifierare

Det finns många fler funktioner i `#include <iomanip>`. Mer information av dessa kan ni hitta på:
https://en.cppreference.com/w/cpp/io/manip

www.liu.se

LINKÖPING UNIVERSITY