TDIU01 (725G67) - Programmering i C++, grundkurs

Program, datatyper och IO

Eric Elfving
Institutionen för datavetenskap

7 oktober 2015

LINKÖPINGS UNIVERSITET

Översikt

- Struktur på ett C++-program
- Köra ett program
- Variabler
- Datatyper
- IO - Inläsning och utskrift

Ett första C++-program


```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    cout << "Ett litet C++-program" << endl;
7    return 0;
8  }
```

Ett första C++-program

hello.cc

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

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

```
zaza4 <203> g++ hello.cc
zaza4 <204> ./a.out
Ett litet C++-program
zaza4 <205>
```

Variabler

- Har **alltid** tre egenskaper
  - Namn
  - Datatyp
  - Värde
- Måste deklareras innan de används
- Kan ses som en ¨låda¨ i datorns internminne

```
int val;
int y {4};
```

val: ?

int

Datatyper

- Bestämmer vad vi kan göra med variabeln och vad (vilka värden) som kan lagras i den
- Vi använder ofta följande datatyper :

int	Lagrar ett heltal
double	Lagrar flyttal, en representation av de reella talen
char	Lagrar ett tecken
bool	Sanningsvärden, `true` eller `false`
string	Lagrar text, en samling av tecken (`char`)

- För djupare beskrivning se extramaterial på föreläsningssidorna på kurshemsidan!

Ett kort C++-program


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

// Ett kort program för att visa C++

int main()
{
  int x,y{5};
  const int z {6};
  y = 4;
  x = x*y; // En beräkning
  return 0;
}

/* Kommentarer kan
   vara flera rader! */
```

Inläsning och utskrift - IO

Formaterad inmatning

- Våra program blir normalt sätt bättre om vi låter användaren mata in data.
- Användaren kan oftast inte vår kod utantill, var därför duktig och ange ledtexter så att användaren vet vad den ska göra.
- Nedan presenteras formaterad inläsning. Inledande vita tecken (tabbar, blanksteg och nya rader) ignoreras varefter det inmatade tolkas beroende på variabeln där datat ska lagras.

```
int x;
cout << "Mata in ett heltal: ";
cin >> x;
```

Inläsning är buffrad!

Inläsning och utskrift - IO

Formaterad utskrift

- Skaparna av språket väljer hur data formateras vid utskrift. Här kommer några saker man kan använda sig av för att ändra utseendet.

`'\n'`	Gå till nästa rad i utskriften
`'\r'`	Börja om på samma rad i utskriften
`flush`	Skriv ut allt i utmatningsbufferten
`endl`	`flush` följt av `\n`

Inläsning och utskrift - IO

Formaterad utskrift

- Instruktioner från `<iomanip>`

`fixed`	Skriv ut flyttal i fixt format med 6 decimaler
`setprecision(N)`	Använd `N` decimaler vid utskrift
`scientific`	Visa flyttal på formatet 1.234e+02
`setw(N)`	Använd minst `N` tecken för nästa utskrift
`setfill(C)`	Använd tecknet `C` som utfyllnadstecken
`left`	Vänsterjustera utskriften
`right`	Högerjustera utskriften

Inläsning och utskrift - IO

Formaterad utskrift

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  int i {10};
  double d {3.141592};
  cout << setw(6) << setfill('.') << i << endl;
  cout << setw(6) << fixed << setprecision(2) << d << endl;
  cout << setw(10) << left << "hej" << endl;
  cout << setfill(' ');
  cout << setw(10) << right << "hej" << endl;
}
```

```
....10
..3.14
hej.......
       hej
```

Inläsning och utskrift - IO

Oformaterad inläsning

- Formaterad inläsning kastar bort inledande vita tecken och tolkar de tecken användaren matade in.
- Detta är dock inte alltid det man vill, då passar oformaterad inläsning bra.
- Antag att vi har dessa deklarationer:

```
string str;
char c;
```

`cin.get(c);`	Läs nästa tecken
`getline(cin, str);`	Läs nästa rad
`getline(cin, str, ':');`	Läs en rad eller tills nästa kolon
`cin.ignore(28, ':');`	Ignorera alla tecken tills teckne eller max 28 tecken.
`cin.peek();`	Tittar på nästa tecken.

www.liu.se

LINKÖPINGS UNIVERSITET