

# Imperativ Programmering og Datastrukturer

## kursusintroduktion

René Rydhof Hansen

7. september 2009

- At kunne forklare hvorfor programmering, og især imperativ programmering, er interessant.
- At kunne forklare, i oversigtsform, forskellene på et højniveausprog og et lavniveausprog samt fordele og ulemper ved samme.
- At kunne indplacere C# som et højniveausprog og argumentere for det.
- At kunne skrive (og få udført) "Hello World!" .
- At kunne skrive (og få udført) simple programmer med simpel input/ouput
- At kunne lave myTunes v1.0

# Undervisningsform

- Praktisk orienteret
- Typisk undervisningsgang:
  - **Studererledet** repetition/opgavegennemgang
  - Forelæsning
  - Øvelser (i forelæsningslokalet)
- Afleveringsopgaver
  - Antal: 3–6
  - Programmering
  - Basis for eksamen
- Eksamen
  - Mundtlig (med udgangspunkt i afleverede opgaver)

# Hvorfor skal jeg lære at programmere?

- For at kunne bruge computere til *generel* problemløsning.
- Ultimativ kontrol over computeren.
- Almen (nat-tek) dannelse: væsentlig(e) generel(le) indsigt(er).
- Programmerings-erfaring og -viden kan overføres til andre områder.
- Styrker logisk/analytisk tænkning.
- Godt redskab i mange sammenhænge, fx let at lave småprogrammer til at undersøge/udforske del-problemer
- ... udfordrende og sjovt?!?!

# Hvad er et program?

- Sekvens af *instruktioner* der kan *udføres* af computeren
  - Instruktion: udskriv (print), addér, flyt
  - Udførelse?
- Opskrift/algoritme: brug ind-data til at producere ud-data
  - Udskrivning/indlæsning (input/output)
- Kildekode (source code)
  - Programmeringssprog (C#)
- Objektkode (object code)
  - Maskinnær kode

## Et konkret program

```
Console.WriteLine(" Hello _World! " );
```

# Hvad er et program?

- Sekvens af *instruktioner* der kan *udføres* af computeren
  - Instruktion: udskriv (print), addér, flyt
  - Udførelse?
- Opskrift/algoritme: brug ind-data til at producere ud-data
  - Udskrivning/indlæsning (input/output)
- Kildekode (source code)
  - Programmeringssprog (C#)
- Objektkode (object code)
  - Maskinnær kode

## Et konkret program

```
Console.WriteLine(" Hello _World!" );
```

- Udviklet af Microsoft under ledelse af bl.a. Anders Hejlsberg i 2001.
- Moderne sprog, inspireret af bl.a. Java, C og C++
- Forsøg på at simplificere og rette fejl i andre sprog
- Vigtig brik i MS's strategi og teknologi
- Meget udbredt sprog
- Primært til Windows, men findes også til Linux
- Objekt-orienteret (**problem**)

# Abstraktionsniveauer

- Maskinkode
- Assembler
- C
- ...
- FORTRAN, COBOL
- BASIC, Pascal
- ...
- C#, Java
- ...
- SML, Haskell, Prolog

Hvordan kommer jeg fra højt niveau til lavt niveau?

- Oversætter (compiler)
- Fortolker (interpreter)

# Det første program... igen

Hello (again) World!

```
class Program
{
    public static void Main()
    {
        Console.WriteLine(" Hello , _world!" );
    }
}
```

- Input?
- Output?
- Udførelse?
  - Oversætter (compiler)
  - Fortolker (interpreter)
- Hvordan får vi programmet ind i computeren?

## Vejledning for “Hello World!”

- Start VS
- Vælg “New project”
- Vælg “Console application”
- Skriv programteksten ind
- Klik på “Start without debug” i “Debug” menuen
- Teksten “Hello World!” skulle nu komme til syne i “Output Window”.

Hello (again)<sup>2</sup> World!

```
class Program
{
    static void Main()
    {
        Console.WriteLine(" Hello , _world!" );
    }
}
```

- class? static? void? Main()?
- Console?

## Det andet program

- Vælg “Windows Forms” istedet for “console application” under “new project”
- Dobbelt-klik på den tomme form

```
class HelloWorld
{
    static void Main()
    {
        MessageBox.Show(" Hello , _World!" );
    }
}
```

# Variable og Tildeling (assignment)

- Variable er navngivne pladser i hukommelsen
- Variable indeholder *værdier*
- En tildeling (assignment) er en ordre som tildeler en værdi til en variabel (ændrer værdien af en variabel): 'x = 42'
- Re. variable i matematik

```
public static void Main()  
{  
    int x;  
    string s;  
  
    s = "117";  
    x = int.Parse(s);  
    s = "x = " + x.ToString();  
    Console.WriteLine(s);  
}
```

# Variable og Tildeling (assignment)

- Variable er navngivne pladser i hukommelsen
- Variable indeholder *værdier*
- En tildeling (assignment) er en ordre som tildeler en værdi til en variabel (ændrer værdien af en variabel): 'x = 42'
- Re. variable i matematik

```
public static void Main()  
{  
    int x;  
    string s;  
  
    s = "117";  
    x = int.Parse(s);  
    s = "x = " + x.ToString();  
    Console.WriteLine(s);  
}
```

# Erklæring og Initialisering

- Variable skal *erklæres*:

```
int x;
```

- Variable *skal initialiseres* inden de bruges. I stedet for:

```
Console.WriteLine("Foo" + s)
```

Bruges:

```
s = "BAR"
```

```
Console.WriteLine("Foo" + s)
```

# Erklæring og Initialisering

- Variable skal *erklæres*:

```
int x;
```

- Variable *skal initialiseres* inden de bruges. Istedet for:

```
Console.WriteLine("Foo" + s)
```

Bruges:

```
s = "BAR"
```

```
Console.WriteLine("Foo" + s)
```

- Hvilken slags værdier kan en variabel indeholde
  - Heltal (int): 42, 17, 117, 87
  - Flydende-komma/decimal tal (double): 3.1415926535
  - Streng: "foo bar!", "foo" + "bar"
- Konvertering mellem typer?

```
int i;
```

```
double x;
```

```
string s;
```

```
s = "FOO" ;
```

```
i = 42;
```

```
x = 4.2;
```

# Udtryk (expressions)

- Et udtryk er den kode der står til højre for lighedstegnet i en tildeling:

## Example (Udtryk)

```
r = 87.3;  
pi = 3.14;  
A = pi * r * r;  
  
s = "foo" + "bar" ;  
s = s + "baz" ;
```

# Sammenfatning og Næste gang

- Udtryk
- Simple programmer

Næste gang

- Basale OO begreber (ved Kurt Nørremark)