Imperativ Programmering og Datastrukturer

Datatyper og kontrolstrukturer

René Rydhof Hansen

8. oktober 2008

Husk!

Næste forelæsning 22. oktober 2008.

Mål

- At kunne forklare hvad en datatype er, hvad den kan bruges til og hvorfor datatyper er vigtige
- At kunne give en oversigt over de mest basale datatyper i C#
- At kunne forklare forskellen på de basale datatyper
- At kunne anvende datatyper korrekt i simple programmer
- At kunne forklare hvad en "form" og en "kontrol" er

Typer

- En *type* er en mængde af værdier
- Hvorfor anvende typer?
  - Typer forbedrer programmets læsbarhed og forståelighed
  - Typer gør det muligt for computeren at generere bedre og mere effektiv kode
  - Typer gør det muligt at opdage fejl under oversættelsen
- Hvordan bruges typer?
  - I variabel-erklæringer: (`int i;`)

Simple Datatyper i C#

Type	Eksempel	Range
`bool`	`true`	`true`, `false`
`char`	`'A'`, `\u0041`	`\u0000`...`\uFFFF`
`sbyte`	`42`, `-117`	-128...127
`byte`	`42`, `87`	0...255
`short`	`-30042`, `8700`	-32768...32767
`ushort`	`32042`, `8700`	0...65535
`int`	`-1000000000`	-2147483648...2147483647
`uint`	`4000000000`	0...4294967295
`long`	`-1000000000000L`	-9223372036854775808...9223372036854775807
`ulong`	`4000000000UL`	0...18446744073709551615
`float`	`-0.999F`, `4E9F`	$\pm 10^{-44}$...$\pm 10^{38}$, 7 betydende cifre
`double`	`-0.999`, `4E9`	$\pm 10^{-323}$...$\pm 10^{308}$, 15–16 betydende cifre
`String`	`"foo"`	

Typekonvertering

- Konvertering mellem "tal typer":
  - `(int)1.17 = 1`
  - `(float)42 = 42.0`
- Konvertering fra streng til tal:
  - `int.Parse("42") = 42`
  - `double.Parse("8.7") = 8.7`
- Konvertering fra tal til streng:
  - `42.ToString() = "42"`
  - `(11.7).ToString() = "11.7"`
  - Implicit konvertering: `4 + "2" = "42"`

Scope (virkefelt)

- Det "område" i programmet hvor en given variabel "gælder"

Example

```
int x = 41;

Main()
{
  int x = 86;

  x = x + 1;
  Console.Writeline(x.ToString());
}
```

Operationer på strenge

- Sammenligning: `"foo" == "bar"`
- Konkatenering: `"foo" + "bar" = "foobar"`
- Enkelt-tegn: `"baz"[1] = 'a'`
- Del-streng: `"foobar".SubString(1,2) = "oo"`
- Fjern: `"foobar".Remove(1,2) = "fbar"`
- Længde: `"foobar".Length = 6`
- Formatering: `String.Format("{0}--{1}",17,42) = "17--42"`

Operationer på tal

- Pre-inkrementering/-dekrementering: `++x`, `--x`
- Post-inkrementering/-dekrementering: `x++`, `x--`
- Addition: `111 + 6 = 117`
- Subtraktion: `89 - 2 = 87`
- Multiplikation: `2 * 21 = 42`
- Division (`int`): `10/3 = 3`
- Division (`double`): `10/3 = 3.333333`
- Modulus: `10 % 3 = 1`

Arrays

- Indekserede variable
- Matematik: x_i for $i = 1..100$
- Erklæring af array

```
String[] titel = new String[10];
```

- Brug af array

```
titel[0] = "???"   // tilladt?
titel[1] = "foo";
titel[2] = "bar";

titel[10] = "baz"
titel[11] = "!!!"   // IKKE tilladt
```

- Alternativt:

```
String[] titel =
    { "alfa", "beta", ..., "omega" };
```

Arrays og `for`-løkker

Example

```
String[] titel = new String[10];

for(int i = 0; i < 10; i++)
{
  Console.Write("Indtast␣tal␣nr.␣" + i + ":␣");
  titel[i] = Console.Readline();
}
```

Arrays i flere dimensioner

Example

```
String[,] matrix = new String[5,10];

for(int i = 0; i < 5; i++)
{
  for(int j = 0; j < 10; j++)
    matrix[i,j] = i + "," + j;
}
```

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7	3,8	3,9
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9

Arrays i flere dimensioner

Example

```
String[,] matrix = new String[5,10];

for(int i = 0; i < 5; i++)
{
  for(int j = 0; j < 10; j++)
    matrix[i,j] = i + "," + j;
}
```

`matrix[3,5] = "XX"`

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8	2,9
3,0	3,1	3,2	3,3	3,4	XX	3,6	3,7	3,8	3,9
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8	4,9