Mange sprog
Grundlæggende begreber
Data typer
Operatorer

Grundlæggende programmering

Lektion 2

Mange sprog

Der findes rigtig, rigtig mange programmeringssprog

De følgende er kun nogle få eksempler

Mange sprog

Bash

Er det "sprog" man bruger i konsollen på Unix (herunder også BSD som MacOS) og Linux maskiner, men i og med at man kan bede OS om at lave en række handlinger kan man kalde det programmering.

```
echo "Hello World"
```

Mange sprog

Basic

Sproget som mange begyndte med i 80'erne og 90'erne. Som navnet siger var tanken bag at det skulle være meget simpelt, men det var så på bekostning af mere avancerede muligheder og effektivitet.

```
PRINT "Hello, world!"
```

Mange sprog

C

"Moderen" i C-familien af sprog, et af de mere effektive men også komplicerede høj-niveau sprog men der findes C-compilere til alle systemer i dag så kan man C skal man "blot" sætte sig ind i de enkelte platformes unikke biblioteker.

```
#include

int main(void)

{

   puts("Hello, world!");

}
```

Mange sprog

C++

En videreudvikling af C som i dag er det dominerende af de klassiske C sprog.

```
#include
int main()
{
   std::cout << "Hello, world!
";
   return 0;
}
```

Mange sprog

C#

Det vi fokuserer på i dette kursus

```
using System;
class Program
{
   public static void Main(string[] args)
   {
      Console.WriteLine("Hello, world!");
   }
}
```

Mange sprog

HTML

HTML er det scriptsprog man opbygger hjemmesider i, så tekst man bare skriver i et HTML dokument vises som tekst på skærmen.

Hello World!

Mange sprog

Java

Java er på trods af navnet også et sprog i C-familien, da det er inspireret af C's opbygning og måder at gøre ting på.

```
import javax.swing.JFrame;  //Importing class JFrame
import javax.swing.JLabel;  //Importing class JLabel
public class HelloWorld {
   public static void main(String[] args) {
      JFrame frame = new JFrame();        //Creating frame
      frame.setTitle("Hi!");              //Setting title frame
      frame.add(new JLabel("Hello, world!"));//Adding text to frame
      frame.pack();                       //Setting size to smallest
      frame.setLocationRelativeTo(null);    //Centering frame
      frame.setVisible(true);             //Showing frame
   }
}
```

Mange sprog

JavaScript

JS er *ikke* en afart af Java men et selvstændigt script sprog til at understøtte HTML i at lave dynamiske og interaktive sider til internettet, det kan dog bruges også til at lave selvstændige apps med de rette værktøjer.

```
document.write('Hello, world!');
```

Mange sprog

jQuery

jQuery er egentlig et JavaScript bibliotek der gør en masse funktioner lettere tilgængelige ved at præ-definere dem. Man ser det dog ofte nævnt på listen over sprog som firmaer søger - men det er ikke et selvstændigt sprog, bare en udvidelse!

```
$("body").append("Hello world!");
```

Mange sprog

Matlab

MatLab er et værktøj til at lave komplicerede beregninger med der er ekstremt elsket i universitets-miljøer, og på dette niveau af kompleksitet kan det selvfølgelig også programmeres.

```
disp('Hello, world!')
```

Mange sprog

Objective-C

Apples C-sprog rettet mod at lave programmer til deres udstyr.

```
#import
int main(void)
{
   NSLog(@"Hello, world!
");
   return 0;
}
```

Mange sprog

Pascal

Endnu et sprog tilbage fra 90'erne som mange lærte, typisk som det første efter Basic

```
program HelloWorld;
begin
  WriteLn('Hello, world!');
end.
```

Mange sprog

Python

Python fokuserer på læsbarhed og på at kunne opnå det samme som C++ og Java med færre linjer.s

```
print "Hello, world!"
```

Mange sprog

Perl

Perl blev skabt under Unix for at gøre rapportering lettere. Siden 1987 har Perl undergået store forandringer og er i dag et generelt anvendeligt programmeringssprog, som kan afvikles på flere forskellige platforme, herunder bl.a. Microsoft Windows og Unix.

```
print "Hello, world!
";
```

Mange sprog

Ruby

Et dynamisk open source programmerings sprog der lægger vægt på enkelhed.

```
puts "Hello, world!"
```

Mange sprog

Swift

Apples ny sprog med fokus på at lave iOS, macOS, watchOS, og tvOS apps.

```
println("Hello, world!")
```

Mange sprog

VBScript

VBScript er et "script"-sprog, der fortolkes via Microsoft's Windows Script Host. Sprogets syntax viser dets oprindelse som en variation af Microsoft's Visual Basic programmeringssprog.

```
MsgBox "Hello, World!"
```

Mange sprog

Visual Basic .NET

Efterfølger til programmeringssproget Visual Basic. Visual Basic .NET er et fuldgyldigt sprog på .NET platformen, men er dog ikke lige så stort som C#. Visual Basic og Visual Basic .NET, ligner hinanden en smule, men der er dog en del forskelle, og sprogene kan ikke blandes.

```
Module Module1
    Sub Main()
        Console.WriteLine("Hello, world!")
    End Sub
End Module
```

Mange sprog

SQL

SQL er det sprog man kommunikerer til database-servere med, og som sådan kan man derfor ikke bede det om at skrive "Hello World" til skærmen.

Mange sprog

De sjove sprog

Programmører har humor, den er bare ofte lidt anderledes og meget mere indforstået end "den almindelige".
Således findes der også en række sprog der enten er lavet fordi opfinderen ville give andre en udfordring ellers simpelthen ville lave noget der var så underligt at det var komisk i sig selv. Som et underholdende indslag bringer jeg tre af dem – men det er ikke noget der bliver brugt seriøst!

Mange sprog
De sjove sprog

ArnoldC

Sproget består udelukkende af one-liners fra Arnold Schwarzenegger film. Sproget blev lavet af Lauri Hartikka, der udskiftede normale kommandoer med Arnold citater. False og True er f.eks. "I lied" og "No problemo".

IT'S SHOWTIME

TALK TO THE HAND "Hello World!"

YOU HAVE BEEN TERMINATED

Mange sprog
De sjove sprog

Brainfuck

Lavet i 1993 af Urban Müller som et lille sprog der skulle underholde programmører. Sproget indeholder kun otte kommandoer og en instruktions pointer. De består hver af ét tegn, så sproget er ekstremt minimalistisk.

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-
]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------
.>+.>.
```

Mange sprog
De sjove sprog

Piet

Sproget er opkaldt efter den hollandske maler Piet Mondrian, der malede billeder bestående af firkanter. Det er også hvad program-merings sproget gør, da det kompileres af en pointer der kører gennem et billede, fra et området til det næste, og laver programmet ud fra farverne.

"Hello World!" ser således sådan ud:

Grundlæggende begreber

Grundlæggende definitioner i C# og generel programmering

Grundlæggende begreber

Indhold i et C# program

Et C # program består af følgende dele:

- Namespace deklaration
- En klasse
- Klasse metoder
- Klasse attributter
- En Main metode
- Statements og Expressions
- Kommentarer

Grundlæggende begreber

Indhold i et C# program

Noter omkring C#

- C # er case sensitive.
- Alle statements og udtryk skal afsluttes med et semikolon (;).
- Udførelsen af programmet starter ved Main metoden.
- I modsætning til Java må programmets filnavn være forskelligt fra klasse navnet.

Grundlæggende begreber

Indhold i et C# program

En variation over eksemplet vi har brugt hidtil:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace HelloWorldApplication
8   {
        0 references
9       class HelloWorld
10      {
            0 references
11          static void Main(string[] args)
12          {
13              /* my first program in C# */
14              Console.WriteLine("Hello World");
15              Console.ReadKey();
16          }
17      }
18  }
```

- Start Visual Studio > New > Project > C# > Classic Desktop > Console Application
- Skriv koden ovenfor ind i den første fil der åbner (Program.cs)

Grundlæggende begreber

Indhold i et C# program

Eksemplet følger strukturen beskrevet tidligere:

- De første linjer med **using** viser at programmet inkluderer forskellige præ-definerede *namespaces.*
  - Et program benytter typisk flere using statements.
  - Hvis vi ikke havde angivet system først skulle vi senere skrive system foran, f.eks. system.console.writeline
  - Vi kunne have lavet et statisk direktiv og sagt using.system.console i toppen hvorefter vi så ikke havde behøvet kalde console foran writeline
- Derefter deklareres ***namespace***.
  - Et namespace er en samling klasser.
  - I dette tilfælde klassen *HelloWorld*
- Den næste linje er en **klasse** (*class*) deklaration, klassen HelloWorld der indeholder den data og de metode deklarationer vores program benytter.
  Metoder definerer hvordan klasserne opfører sig, men HelloWorld klassen har dog kun én metode: Main.

Grundlæggende begreber

Indhold i et C# program

Eksemplet følger strukturen beskrevet tidligere:

- Den næste linje definerer *Main* metoden, der er starten på alle C# programmer. Main metoden angiver hvad klassen gør når udført.
- Den næste linje er /*...*/ ignoreres af compileren da det er **kommentarer**.
- Main metodens opførsel angives med statementet **Console.WriteLine("Hello World");**
  - *WriteLine* er en metode I *Console* klassen defineret i *System* namespace. Dette statement sørger for at beskeden "Hello, World!" vises på skærmen.
  - Den sidste linje **Console.ReadKey();** er til når vi kører det i VS.NET. Dette får programmet til at vente på at der trykkes en tast og forhindrer dermed skærmen fra at lukke for hurtigt når vi kører programmet fra Visual Studio .NET.

Grundlæggende begreber

Indhold i et C# program

Endnu et eksempel

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace RectangleApplication
8   {
        2 references
9       class Rectangle
10      {
11          // member variables
12          double length;
13          double width;
            1 reference
14          public void Acceptdetails()
15          {
16              length = 4.5;
17              width = 3.5;
18          }
19
            1 reference
20          public double GetArea()
21          {
22              return length * width;
23          }
24
            1 reference
25          public void Display()
26          {
27              Console.WriteLine("Length: {0}", length);
28              Console.WriteLine("Width: {0}", width);
29              Console.WriteLine("Area: {0}", GetArea());
30          }
31      }
32
        0 references
33      class ExecuteRectangle
34      {
            0 references
35          static void Main(string[] args)
36          {
37              Rectangle r = new Rectangle();
38              r.Acceptdetails();
39              r.Display();
40              Console.ReadLine();
41          }
42      }
43  }
```

- Start Visual Studio > New > Project > C# > Classic Desktop > Console Application hvis I har lukket det
- Skriv koden ovenfor ind i den første fil der åbner (Program.cs) eller lav en ny klasse hvis I fortsætter et projekt

Grundlæggende begreber

Indhold i et C# program

Nøgleord i det andet eksempel:

- **double** angiver en simpel type, der gemmer en 64-bit floating-point-værdi (15-16 cifre).
- **public** når man deklarer en metode fortæller compilere at metoden skal være bredt tilgængelig for alt i programmet, ikke kun for den klasse den hører under.
- **void** bruges som output type for en metode, hvor det angiver, at metoden ikke returnerer en værdi.
- **Length: {0}** angiver at vi vil have første værdi i length, da det kunne have været et array mere flere værdier og dermed flere pladser.

C# nøgleord

Underviseren, kurset og stedet

C# nøgleord

Reserved Keywords						
abstract	as	base	bool	break	byte	case
catch	char	checked	class	const	continue	decimal
default	delegate	do	double	else	enum	event
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	in (generic modifier)	int
interface	internal	is	lock	long	namespace	new
null	object	operator	out	out (generic modifier)	override	params
private	protected	public	readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc	static	string	struct
switch	this	throw	true	try	typeof	uint
ulong	unchecked	unsafe	ushort	using	virtual	void
volatile	while					
Contextual Keywords						
add	alias	ascending	descending	dynamic	from	get
global	group	into	join	let	orderby	partial (type)
partial (method)	remove	select	set			

Data typer

Forskellige former for data opbevares forskelligt

Data typer

Variablerne i C# er inddelt i tre typer

- Værdi typer / Value types
- Reference typer / Reference types
- Markør typer / Pointer types

Data typer
Værdi typer

Value types variabler kan direkte få en værdi. De er afledt af klassen class **System.ValueType**.

Det kan være data som i**nt, char** og **float**, der gemmer tal, bogstaver og komma tal respektivt.

Når man deklarerer en **int** type allokerer systemet automatisk hukommelse til at gemme værdien

Data typer
Værdi typer

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0}$ to 28	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

Data typer
Værdi typer

For at få den eksakte størrelse på en type eller en variabel på en bestemt platform kan man benytte **sizeof** metoden.
Udtrykket expression *sizeof(type)* giver opbevarings størrelse på objektet eller typen i bytes. Husk at størrelsen på typen er fast uanset indhold.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   using System;
8   using System.Collections.Generic;
9   using System.Linq;
10  using System.Text;
11  using System.Threading.Tasks;
12
13  namespace DataTypeApplication
14  {
        0 references
15      class Program
16      {
            0 references
17          static void Main(string[] args)
18          {
19              int i = 123;
20              Console.WriteLine("The size of short is {0}.", sizeof(short));
21              Console.WriteLine("The size of int is {0}.", sizeof(int));
22              Console.WriteLine("The size of long is {0}.", sizeof(long));
23              Console.ReadLine();
24          }
25      }
26  }
```

Data typer
Reference typer

En reference type indeholder ikke faktisk data gemt i en variabel men indeholder en reference til variablerne.

De refererer altså til en plads i hukommelsen. Således kan flere variabler af reference typerne referere til sammes plads i hukommelsen. Hvis dataen på denne plads i hukommelsen ændres af en af variablerne vil de andre automatisk ændre indhold.
Et eksempel på **built-in** reference typer er: **object**, **dynamic** og **string**.

Data typer
Objekt typer

Object Type er den grundlæggende base klasse for alle datatyper i C# Object er et alias for System.Object klassen. Objekt typer kan få værdier fra andre typer, value typer, reference typer, predefined eller user-defined typer. Men før den kan tildeles disse værdier, har den dog brug for type konvertering.

Når en value type konverteres til en object type kaldes det **boxing** og den anden vej rundt, når en object type konverteres til en value type, kaldes det **unboxing**

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   using System;
8   using System.Collections.Generic;
9   using System.Linq;
10  using System.Text;
11  using System.Threading.Tasks;
12
13  namespace DataTypeApplication
14  {
        0 references
15      class Program
16      {
            0 references
17          static void Main(string[] args)
18          {
19              object obj;
20              obj = 100; // this is boxing
21              Console.WriteLine("The stored number is {0}",obj);
22              Console.ReadLine();
23          }
24      }
25  }
```

Data typer
Dynamiske typer

Du kan gemme hvilken som helst værdi i dynamic data type variabler. Type checking af disse typer variabler tager tid ved run-time.

Dynamiske typer ligner objekt typer bortset fra at type checking for object type variabler finder sted ved compilering, hvor dynamic type variabler finder sted ved run time.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   using System;
8   using System.Collections.Generic;
9   using System.Linq;
10  using System.Text;
11  using System.Threading.Tasks;
12
13  namespace DataTypeApplication
14  {
        0 references
15      class Program
16      {
            0 references
17          static void Main(string[] args)
18          {
19              dynamic d = 20;
20              Console.WriteLine("The stored number is {0}",d);
21              Console.ReadLine();
22          }
23      }
24  }
```

Data typer
Streng typer

String Type lader en angive en hvlken som helst string værdi til en variabel. String type er et alias for System.String klassen. Den er afledt af en objekt type. Værdien i en string kan angives ved at benytte string literals i to former: quoted og @quoted.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   using System;
8   using System.Collections.Generic;
9   using System.Linq;
10  using System.Text;
11  using System.Threading.Tasks;
12
13  namespace DataTypeApplication
14  {
        0 references
15      class Program
16      {
            0 references
17          static void Main(string[] args)
18          {
19              String str = "This is a\nmulti-line\nstring";
20              Console.WriteLine(str);
21              Console.ReadLine();
22          }
23      }
24  }
```

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   using System;
8   using System.Collections.Generic;
9   using System.Linq;
10  using System.Text;
11  using System.Threading.Tasks;
12
13  namespace DataTypeApplication
14  {
        0 references
15      class Program
16      {
            0 references
17          static void Main(string[] args)
18          {
19              string str = @"This is NOT a\nmulti-line\nstring";
20              Console.WriteLine(str);
21              Console.ReadLine();
22          }
23      }
24  }
```

Data typer
Streng typer

- Strenge understøtter indlejrede (embedded) udtryk når man benytter string interpolation format.
- Hvis du vil finde længden på en streng bruger du **length**, der er en read only egenskab der hverken kan sættes eller kræver nogen parametre.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication5
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              string word;
14              System.Console.Write("Enter a word: ");
15              word = System.Console.ReadLine();
16              System.Console.WriteLine(
17                  $"The word \"{word}\" is"
18                  + $" {word.Length} characters.");
19              Console.ReadLine();
20          }
21      }
22  }
```

Data typer
Markør typer

Markør/pointer type variabler gemmer hukommelses adressen for en anden type. Pointers i C# har de samme muligheder som pointers i C eller C++.

```
int* myVariable;
```

Udtrykket *myVariable henviser til int variablen der findes på adressen indeholdt i myVariable.

Pointers kan dog let lede til usikker kode, hvilket vi kommer ind på senere.

Data typer

OPGAVE

- Lav et program der indebeholder følgende
  - Erklær en double variabel kaldet "penge" og tildel den værdien 200.50
  - Erklær en string variabel og tildel den værdien: "Jeg har"
  - Erklær en ny string variabel, og tildel den værdien: " kr. i banken"
  - Udskriv følgende: "Det er ikke for at prale men…" og en ny linje med "Jeg har 1075.50 kr. i banken"
    - Du skal her bruge de 3 variabler du har lavet, og udskrive dem på samme linie
    - Husk at der er forskel på Console.Write og Console.WriteLine

Data typer

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication5
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              double penge = 1075.50;
14              string jh  = "Jeg har ";
15              string kr = "kr i banken.";
16              Console.WriteLine("Det er ikke for at prale men...");
17              Console.Write(jh);
18              Console.Write(penge);
19              Console.Write(kr);
20              Console.ReadLine();
21          }
22      }
23  }
```

Data typer
Konvertering

Vi så tidligere hvordan man kunne unboxe en objekt type så den blev en data type. Man kan generelt konvertere mellem data typer, men husk kun at gøre det når det giver mening da man let får data tab!

- Tag eksemplet **long** til **int**, der går vi fra værdier så store som 9.223.372.036.854.775.808 til et maks på 2.147.483.647.
- En konversion der resulterer i et sådant tab kræver et **explicit cast** hvor en der ikke resulterer i data-tab og dermed ingen exception er en **implicit conversion**.

Data typer
Konvertering
Eksplicit

- Explicit cast benytter **cast** operatoren, hvor man ved at angive det man vil konvertere i parentesen viser systemet at man virkelig vil det.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              double x = 1234.7;
14              int a;
15              a = (int)x;
16              Console.WriteLine(a);
17              Console.ReadLine();
18          }
19      }
20  }
```

Data typer
Konvertering
Implicit

- Ved en implicit konvertering kan compilere ikke se nogen fejl i konverteringen, så den udfører den. **int** til **long** f.eks.
- Der er dog ekstra muligheder som system.convert, der dog kunne virker mellem få typer, og ToString() der virker med alt.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              bool boolean = true;
14              string text = boolean.ToString();
15              System.Console.WriteLine(text);
16              Console.ReadLine();
17          }
18      }
19  }
```

Data typer Konvertering

OPGAVE

- Lav et program der indebeholder følgende
  - Bed brugeren om tre bogstaver og vis dem i omvendt rækkefølge.
  - Husk kommandoen Convert.ToChar og kombiner den med Console.ReadLine

Data typer Konvertering

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              char letter, letter2, letter3;
14
15              Console.Write("Enter letter: ");
16              letter = Convert.ToChar(Console.ReadLine());
17
18              Console.Write("Enter letter: ");
19              letter2 = Convert.ToChar(Console.ReadLine());
20
21              Console.Write("Enter letter: ");
22              letter3 = Convert.ToChar(Console.ReadLine());
23
24              Console.WriteLine("{0} {1} {2}", letter3, letter2, letter);
25
26              Console.ReadLine();
27          }
28      }
29  }
```

Data typer
Arrays

I C # erklærer du arrays med firkantede parenteser

- Først angiver man typen af array.
- Dette følges af åbne og lukkede firkantede parenteser.
- Så angiver man navnet på variablen.

```
string[] sprog;
int[] telefonnummer;
bool [] fredag
```

- Arrays kan være i flere dimensioner, C# understøtter single-dimensional arrays, multidimensional arrays (rectangular arrays), og array-of-arrays (jagged arrays).S

Data typer
Arrays

- Single-dimensional arrays

```
string[] name = new string[]{"Erik"};
```

- Multidimensional arrays (rectangular arrays)

```
string[,] names = new string[]{" Erik","Britta"};
```

- Array-of-arrays (jagged arrays)

```
byte[][] scores = new byte[5][];
for (int x = 0; x < scores.Length; x++)
{
        scores[x] = new byte[4];
}
```

Data typer
Arrays

- Single-dimensional arrays

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              string[] name = new string[]{"Erik"};
14
15              foreach (string item in name)
16              {
17                  Console.WriteLine(item.ToString());
18              }
19
20              Console.ReadLine();
21          }
22      }
23  }
```

Data typer
Arrays

- Multidimensional arrays (rectangular arrays)

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              string[] name = new string[]{"Erik","Britta"};
14
15              foreach (string item in name)
16              {
17                  Console.WriteLine(item.ToString());
18              }
19
20              Console.ReadLine();
21          }
22      }
23  }
```

Data typer
Arrays

- Array-of-arrays (jagged arrays)

```
9   class Program
10  {
        0 references
11      static void Main(string[] args)
12      {
13          // Declare local jagged array with 3 rows.
14          int[][] jagged = new int[3][];
15
16          // Create a new array in the jagged array, and assign it.
17          jagged[0] = new int[2];
18          jagged[0][0] = 1;
19          jagged[0][1] = 2;
20
21          // Set second row, initialized to zero.
22          jagged[1] = new int[1];
23
24          // Set third row, using array initializer.
25          jagged[2] = new int[3] { 3, 4, 5 };
26
27          // Print out all elements in the jagged array.
28          for (int i = 0; i < jagged.Length; i++)
29          {
30              int[] innerArray = jagged[i];
31              for (int a = 0; a < innerArray.Length; a++)
32              {
33                  Console.Write(innerArray[a] + " ");
34              }
35              Console.WriteLine();
36          }
37          Console.ReadLine();
38      }
39  }
```

Operatorer

Syntaks til at udføre forskellige beregninger og handlinger

Booleans og hvorfor de er essentielle og praktiske

Operatorer

Operatorer bruges til at udføre matematiske eller logiske operationer på værdier
(eller variabler) kaldet operander for at producere en ny værdi, kaldet resultatet.

```
int difference = 4 – 2;
```

Her er – operatoren og resultatet 2 gemmes i integer data typen difference.

Operatorer er generelt inddelt i tre kategorier

- unære / unary
- binære / binary
- Ternære / tertiary

svarende til antallet af operander (en, to, og tre henholdsvis).

Operatorer
Unære

Plus og minus unære operatorer (+, -)

Hvis man har brug for at ændre en værdi fra positiv til negativ (eller omvendt) kan C# gøre det med den unære operator –

Den undære operator + har sjældent en effekt og er mest med for god ordens skyld.

Operatorer Binære

Aritmetiske binære operatorer (+, -, *, /, %)

Disse bruges til at foretage beregninger med to værdier.

```
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int numerator;
14              int denominator;
15              int quotient;
16              int remainder;
17
18              System.Console.Write("Enter the numerator: ");
19              numerator = int.Parse(System.Console.ReadLine());
20
21              System.Console.Write("Enter the denominator: ");
22              denominator = int.Parse(System.Console.ReadLine());
23
24              quotient = numerator / denominator;
25              remainder = numerator % denominator;
26
27              System.Console.WriteLine(
28              $"{numerator} / {denominator} = {quotient} with remainder {remainder}");
29
30              Console.ReadLine();
31          }
32      }
33  }
```

Operatorer Binære

Aritmetiske binære operatorer (+, -, *, /, %)

I eksemplet bliver divisionen og resten operationerne gennemført før opgaverne. Den rækkefølge, som de operatorerne udføres ier bestemt af deres forrang (precedence) og associativitet (associavity).

Rangen for operatørerne hidtil anvendt er_

1. *, / og % har højeste prioritet.
2. + og - har lavere prioritet.
3. = har den laveste rang af disse seks operatører.

Man kan også benytte + til at sætte andet end tal sammen, f.eks. flere strenge som vi har gjort tidligere.

Operatorer Binære

Compound Assignment Operators (+=, -=, *=, /=, %=)

Compound assignment operators kombinerer standard binære operator beregninger med en assignment operator.

Således giver

```
int x = 123;
x = x + 2;
```

og

```
int x = 123;
x += 2;
```

det samme.

Der findes talrige andre ”compund assignment" operatører til at give lignende funktionalitet. Du kan således også bruge assignment operatoren sammen med subtraktion, multiplikation, division, og resten (remainder) operatører.

Operatorer Unære

Forøg og formindsk operatorer (++, --)

C # omfatter særlige unære operatorer til forøgelse og formindskelse. Således inkrementerer inkrementeringsoperatoren, ++, en variabel hver gang den anvendes.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13                  int Value = 12;
14
15                  Console.WriteLine("Techniques of incrementing a value");
16                  Console.Write("Value = ");
17                  Console.WriteLine(Value);
18
19                  Value = Value + 1;
20
21                  Console.Write("Value = ");
22                  Console.WriteLine(Value);
23
24              Console.ReadLine();
25          }
26      }
27  }
```

Operatorer
Unære

Forøg og formindsk operatorer (++, --)

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int Value = 12;
14
15              Console.WriteLine("Techniques of incrementing a value");
16              Console.Write("Value = ");
17              Console.WriteLine(Value);
18
19              Value++;
20
21              Console.Write("Value = ");
22              Console.WriteLine(Value);
23
24              Console.ReadLine();
25          }
26      }
27  }
```

Operatorer
Unære

Forøg og formindsk operatorer (++, --)

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int Value = 12;
14
15              Console.WriteLine("Techniques of incrementing a value");
16
17              Value++;
18              Console.Write("Value = ");
19              Console.WriteLine(Value);
20
21              Value++;
22              Console.Write("Value = ");
23              Console.WriteLine(Value);
24
25              Value++;
26              Console.Write("Value = ");
27              Console.WriteLine(Value);
28
29              Console.ReadLine();
30          }
31      }
32  }
```

Operatorer
Unære

Forøg og formindsk operatorer (++, --)

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int Value = 12;
14
15              Console.WriteLine("Techniques of incrementing a value");
16
17              Value++;
18              Console.Write("Value = ");
19              Console.WriteLine(Value);
20
21              Value--;
22              Console.Write("Value = ");
23              Console.WriteLine(Value);
24
25              Value--;
26              Console.Write("Value = ");
27              Console.WriteLine(Value);
28
29              Console.ReadLine();
30          }
31      }
32  }
```

Operatorer
Unære

Forøg og formindsk operatorer (++, --)

Det har betydning om operatoren star foran eller bagved.

Når man skriver en ++ foran en værdi bliver værdien af variablen øget før den bliver kaldt. Hvis man derimod skriver ++ efter øges værdien først efter at den er blevet kaldt.

```
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int Value = 12;
14
15              Console.WriteLine("Techniques of incrementing a value");
16
17              Console.Write("Value = ");
18              Console.WriteLine(Value);
19
20              Console.Write("Value = ");
21              Console.WriteLine(Value++);
22
23              Console.Write("Value = ");
24              Console.WriteLine(Value);
25
26              Console.ReadLine();
27          }
28      }
29  }
```

Operatorer
Flow kontrol

Flow kontrol

Selv i simple programmer kan det være nødvendigt at styre rækkefølgen af udførelse af programmet, ikke blot fra start til slut

Al C # kode I har set hidtil har haft én ting til fælles. I hvert tilfælde er programmet udført fra den ene linje til den næste fra top til bund i rækkefølge, intet mangler. Hvis alle programmer arbejdede som dette, så ville man være meget begrænset i hvad man kunne gøre.

For at opnå dette benytter man forgrening og looping.
Forgrening udfører kode betinget, afhængigt af resultatet af en evaluering, som "only execute this code if the variable myVal is less than 10."
Looping udfører gentagne gange de samme udsagn, enten et bestemt antal gange, eller indtil en test tilstand har været nået.

Operatorer
Flow kontrol

Flow kontrol

Selv i simple programmer kan det være nødvendigt at styre rækkefølgen af udførelse af programmet, ikke blot fra start til slut

Al C # kode I har set hidtil har haft én ting til fælles. I hvert tilfælde er programmet udført fra den ene linje til den næste fra top til bund i rækkefølge, intet mangler. Hvis alle programmer arbejdede som dette, så ville man være meget begrænset i hvad man kunne gøre.

For at opnå dette benytter man forgrening og looping.
Forgrening udfører kode betinget, afhængigt af resultatet af en evaluering, som "only execute this code if the variable myVal is less than 10."
Looping udfører gentagne gange de samme udsagn, enten et bestemt antal gange, eller indtil en test tilstand har været nået.

Begge disse teknikker indebærer anvendelse af boolsk logik.

Operatorer
Flow kontrol

if statements

Et if statement er et af de mest udbredte i C#. Det evaluerer et **boolsk udtryk** (**Boolean expression**), et udtryk der resulterer i enten sandt eller) der kaldes **condition**.
Hvis tilstanden er sand så udføres **consequence statement**.

Et if statement kan eventuelt have en else clause der indeholder et **alternativt statement** der udføres hvis condition er falsk.

```
if (condition)
consequence-statement
else
alternative-statement
```

Operatorer
Flow kontrol

if statements

```
11          static void Main(string[] args)
12          {
13              int input; // Declare a variable to store the input.
14
15              System.Console.Write(
16              "What is the maximum number " +
17              "of turns in tic-tac-toe?" +
18              "(Enter 0 to exit.): ");
19
20              // int.Parse() converts the ReadLine() - return to an int data type.
21              input = int.Parse(System.Console.ReadLine());
22
23              if (input <= 0) // line 16 - Input is less than or equal to 0.
24                  System.Console.WriteLine("Exiting...");
25              else
26              if (input < 9) // line 20 - Input is less than 9.
27                  System.Console.WriteLine(
28                  $"Tic-tac-toe has more than {input}" +
29                  " maximum turns.");
30              else
31              if (input > 9) // line 26 - Input is greater than 9.
32                  System.Console.WriteLine(
33                  $"Tic-tac-toe has fewer than {input}" +
34                  " maximum turns.");
35              else
36                  // Input equals 9.
37                  System.Console.WriteLine( // line 33
38                  "Correct, tic-tac-toe " +
39                  "has a maximum of 9 turns.");
40              Console.ReadLine();
41          }
42      }
43  }
```

Operatorer
Flow kontrol

Code blocks

Med tuborg-klammer {} kan man kombinere statements til et enkelt statement, et **block statement** eller **code block** der fungerer som ét statement.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              Random r = new Random();
14              int n = r.Next(-5, 5);
15
16              Console.WriteLine(n);
17
18              if (n > 0)
19              {
20                  Console.WriteLine("The n variable is positive");
21              }
22          Console.ReadKey();
23          }
24      }
25  }
```

Operatorer
Flow kontrol

Code blocks

Code blocks kaldes ofte “scopes”. Scopes bruges til at afgøre, hvad ting et navn refererer til, et declaration space bestemmer, hvornår to ting erklæret med det samme navngive går i konflikt med hinanden.

Værdier er ikke tilgængelige uden for scope.

Operatorer
Binære
Flow kontrol

Relational og equality operators

Relationelle og ligestillings operators afgør om en værdi er større end, mindre end eller lig med en anden værdi.

C # syntaks for ligestilling bruger ==, ligesom mange andre programmering sprog gør. For at afgøre om input lig 9, bruger man input == 9 operatoren for at adskille det fra tildelings operatoren, =.

Udråbstegnet betyder *ikke* i C #, så til test for ulighed du bruger ulighed operatoren ! =.

Operatorer
Binære
Flow kontrol

Relational og equality operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Relational og equality operators

Op
Bi
Flo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication6
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int a = 21;
            int b = 10;

            if (a == b)
            {
                Console.WriteLine("Line 1 - a is equal to b");
            }
            else
            {
                Console.WriteLine("Line 1 - a is not equal to b");
            }

            if (a < b)
            {
                Console.WriteLine("Line 2 - a is less than b");
            }
            else
            {
                Console.WriteLine("Line 2 - a is not less than b");
            }

            if (a > b)
            {
                Console.WriteLine("Line 3 - a is greater than b");
            }
            else
            {
                Console.WriteLine("Line 3 - a is not greater than b");
            }
            /* Lets change value of a and b */
            a = 5;
            b = 20;

            if (a <= b)
            {
                Console.WriteLine("Line 4 - a is either less than or equal to  b");
            }

            if (b >= a)
            {
                Console.WriteLine("Line 5-b is either greater than or equal to b");
            }
            System.Console.ReadLine();
        }
    }
}
```

Operatorer
Binære
Flow kontrol

Logiske boolean operatorer (|, ||, &, && og ^)

OR, AND og eklsusivt OR

Udgaverne | og & bruges sjældent.

Operatorer
Binære
Flow kontrol

OR(||)

Uanset hvilken side af tegnet der er sandt giver udtrykket et sandt svar. Ved | tjekkes om de efterfølgende er korrekte, ved || stopper tjekket efter den første korrekte.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int hourOfTheDay = 25;
14              if ((hourOfTheDay > 23) || (hourOfTheDay < 0))
15                  Console.WriteLine("The time you entered is invalid.");
16              else
17                  Console.WriteLine("It is getting a bit late, isn't it?");
18              System.Console.ReadLine();
19          }
20      }
21  }
```

Operatorer
Binære
Flow kontrol

AND(&&)

Boolean AND operatoren && er kun sand hvis *begge* sider er korrekte.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int hourOfTheDay = 10;
14              if ((7 < hourOfTheDay) && (hourOfTheDay < 24))
15                  System.Console.WriteLine("Hi-Ho, Hi-Ho, it's off to work we go.");
16              else
17                  Console.WriteLine("It is getting a bit late, isn't it?");
18              System.Console.ReadLine();
19          }
20      }
21  }
```

Operatorer
Binære
Flow kontrol

Exclusive OR (^)

Kinserhatten ^, er "exclusive OR" (XOR) operatoren. Når den bruges på to Boolean statements bliver den kun sand når kun den ene af dem er det.

Boolean XOR tjekker *altid* begge sider.

Left Operand	Right Operand	Result
True	True	False
True	False	True
False	True	True
False	False	False

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              Console.WriteLine(true ^ false);
14              Console.WriteLine(false ^ false);
15              System.Console.ReadLine();
16          }
17      }
18  }
```

Operatorer
Unære
Flow kontrol

Logical Negation Operator (!)

Logical negation operator, eller **NOT** operator, !, vender en bool værdi om.

Den er unær så den kræver kun én operant.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              bool valid = false;
14              bool result = !valid;
15              System.Console.WriteLine($"Result = { result }");
16              System.Console.ReadLine();
17          }
18      }
19  }
```

Operatorer
Ternære
Flow kontrol

Conditional Operator (?:)

I stedet for et if-else udsagn til at vælge en af to værdier kan man bruge den betingede operatør. Den betingede operatør bruger både et spørgsmål-tegn og et kolon:

condition ? consequence : alternative

Den betingede operator er en "tredelt" operatør, fordi det har tre operander: tilstand, konsekvens, og alternativ. (Eftersom det er den eneste ternære (ternary) operator i C # kaldes den ofte "den ternære operator", men det er tydeligere at henvise til den med navn end med antallet af operander den benytter).

Ligesom de logiske operatorer kører den en slags kortslutning. Hvis betingelsen evalueres til sand, den betingede operatør evaluerer den kun konsekvens. Hvis den betingede evalueres til falsk, vurderer den kun alternativ. Resultatet af operatoren er det evaluerede udtryk.

Operatorer
Ternære
Flow kontrol

Conditional Operator (?:)

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApplication6
8   {
        0 references
9       class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              int x = 10;
14              int y = 20;
15
16              int maxValue = (x > y) ? x : y;
17              int minValue = (x < y) ? x : y;
18
19              Console.WriteLine("x = {0}, y = {1}", x, y);
20              Console.WriteLine("Largest value: {0}, smallest: {1}", maxValue, minValue);
21              System.Console.ReadLine();
22          }
23      }
24  }
25
```

Operatorer
Ternære
Flow kontrol

Null-Coalescing Operator (??)

Null-coalescing operatoren bruges når hvis den ene værdi er null så en anden skal bruges.

expression1 ?? expression2

Denne operator kører også en slags kortslutning.
Hvis expression1 ikke er null er resultatet af operationen dets værdi og det andet udtryk behandles ikke. Hvis expression1 *er* null bliver expression2 værdien af operatoren.

Opsummering

Opgaver og video der opsummerer det vi har talt om og som I kan øve videre på fremadrettet

Opsummering

OPGAVE

- Begynd at tage prøverne inde på https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169 fra en ende af.
- I behøves ikke se videoerne, men brug dem endelig som reference / genopfriskning.

Kilder

Materiale benyttet i denne lektion
Noget af det er udover pensum-listen!

Kilder

Mange sprog

- https://excelwithbusiness.com/blog/post/web-design/say-hello-world-in-28-different-programming-languages
- http://www.hongkiat.com/blog/bizarre-insane-programming-languages/
- https://esolangs.org/wiki/Hello_world_program_in_esoteric_languages

Grundlæggende begreber

- https://www.tutorialspoint.com/csharp/index.htm
- https://msdn.microsoft.com/da-dk/library/678hzkk9.aspx (double)
- https://msdn.microsoft.com/en-us/library/yzh058ae.aspx (public)

Kilder

Data typer

- https://www.nemprogrammering.dk/Tutorials/c-sharp/5-strings.php
- http://practiceexercisescsharp.blogspot.dk/p/3-basic-data-types-3.html
- https://msdn.microsoft.com/en-us/library/aa288453(v=vs.71).aspx
- http://practiceexercisescsharp.blogspot.dk/p/4-arrays-structures-and-strings.html

Operatorer

- https://www.tutorialspoint.com/csharp/csharp_operators.htm
- https://msdn.microsoft.com/da-dk/library/aa664462(v=vs.71).aspx
- https://msdn.microsoft.com/en-us/library/36x43w8w.aspx

Kilder

Operatorer

- http://www.functionx.com/csharp2/conditions/Lesson01d.htm
- https://msdn.microsoft.com/en-us/library/hh147286(v=vs.88).aspx
- https://msdn.microsoft.com/en-us/library/6a71f45d(VS.71).aspx
- https://msdn.microsoft.com/da-dk/library/ty67wk28.aspx
- https://msdn.microsoft.com/en-us/library/zakwfxx4(v=vs.100).aspx
- https://msdn.microsoft.com/da-dk/library/ms173224(v=vs.100).aspx