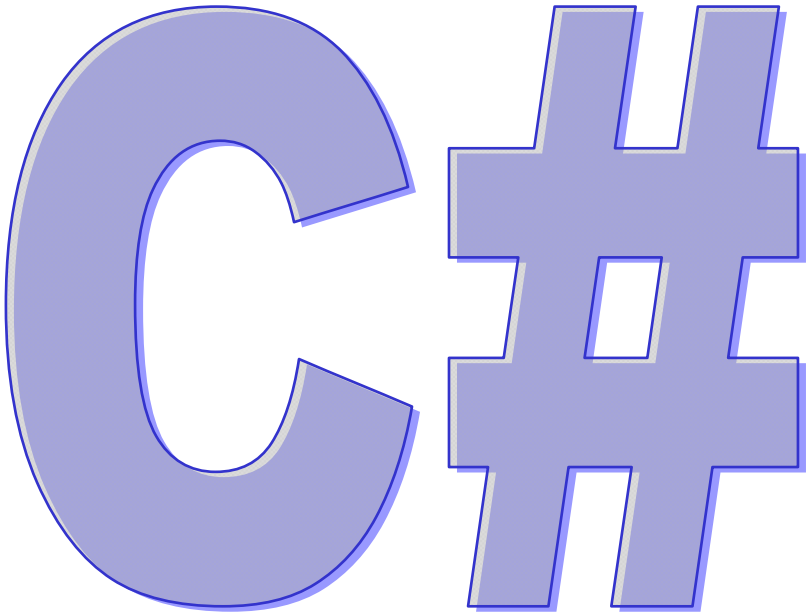
Introduktion til C# programmering.



Målbeskrivelse

Du skal efter endt undervisning have kendskab til programmet Visual C# Express. Du vil lære at bruge programmet, til at lave simple consol programmer og blive gjort bekendt med objektorienteret programmering.

I undervisningen vil følgende områder blive gennemgået:

Datatyper: Variabler og Konstanter

Metode kald

Løkkestrukturer: For, do-while og while

Betingelser: If, if-else og switch case

Grafisk brugerflade (consol skærm)

Målpinde og Delmålpinde

Eleven vil kunne fremstille små simple programmer under anvendelse af en given programmeringssoftware på baggrund af viden om programmeringssprog og elementær programopbygning og -udvikling.

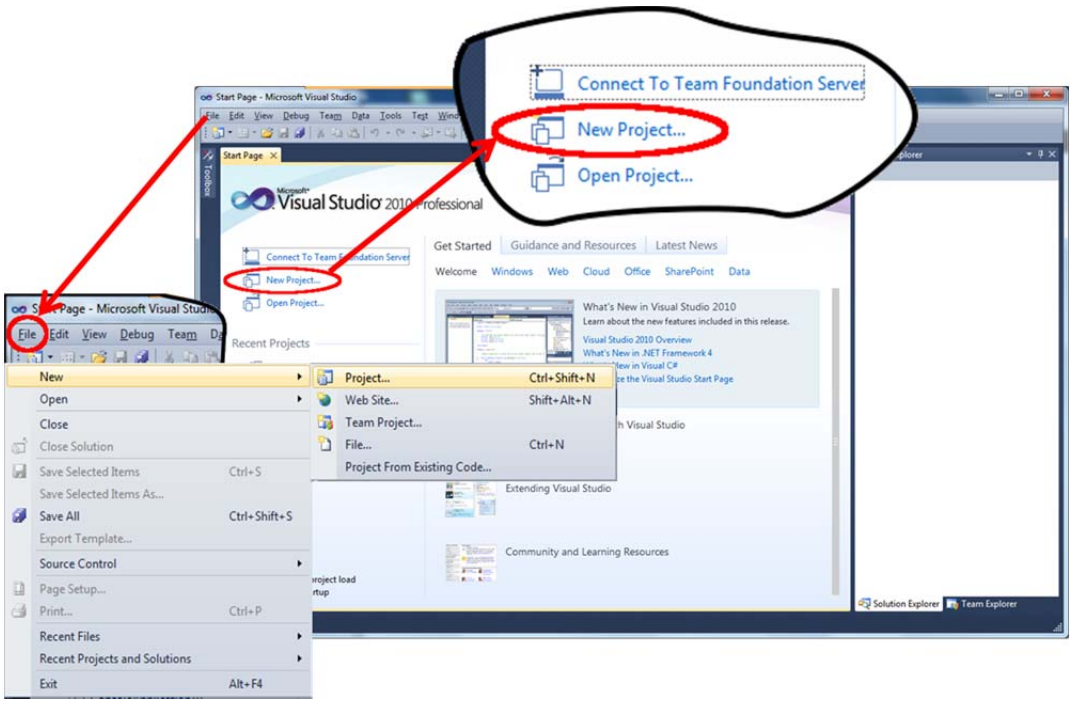
.

Udviklings software

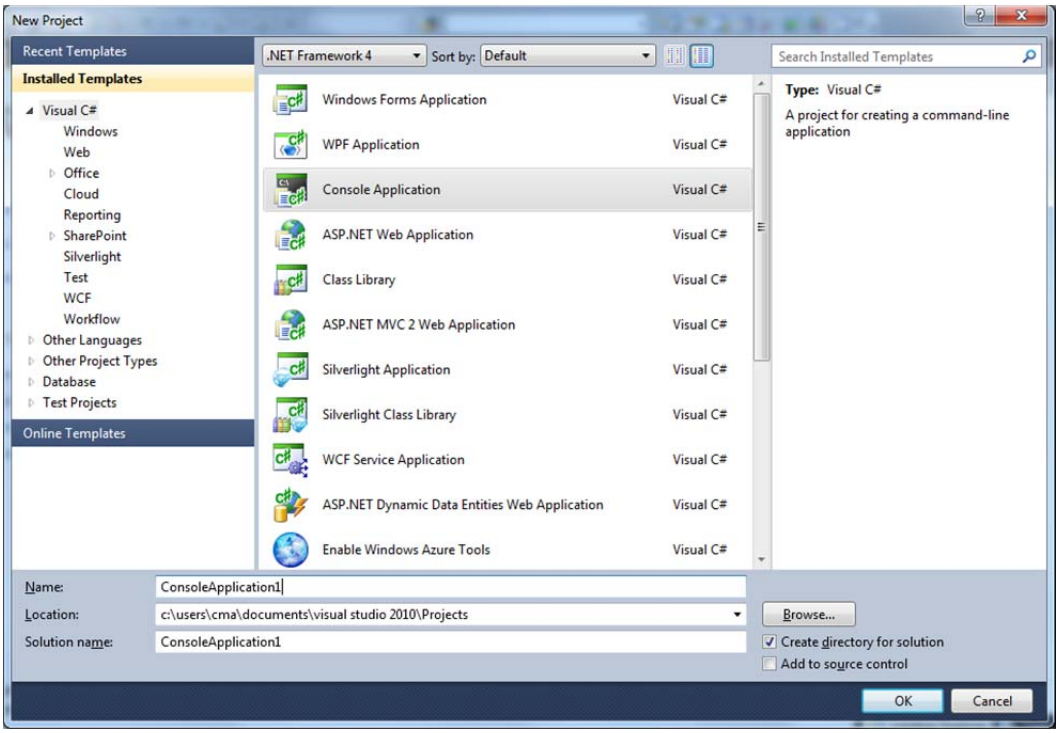
Til at udvikle programmer i C# skal der bruges et programmeringsværktøj.

Programmet hedder Microsoft Visual Studio Express og kan hentes gratis hos Microsoft.

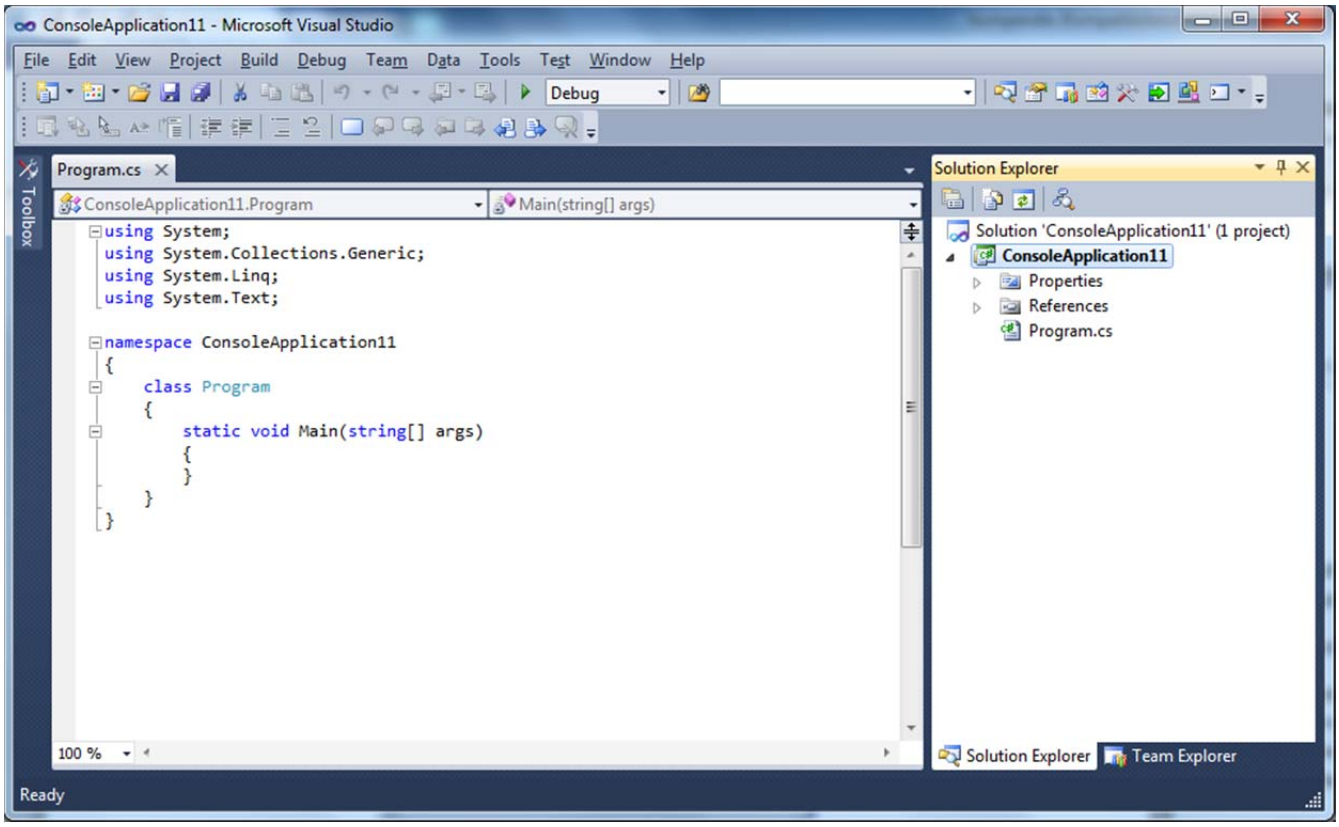
Mit første program.



Når man skal i gang med et nyt program vælges Filer/New Projekt.



Vælg **Console Application** og skriv det navn du ønsker programmet skal have i feltet **Name**.



Introduktion til C#

C# (udtales C sharp) bruger objektorienteret programmering (OOP). OOP giver mulighed for modulopdeling, som giver overblik og struktur.

C# består i sig selv af sproget (som kaldes nøgleord) og en længere række metoder som man kan kalde.

En metode kan betragtes som et underprogram som udfører en givet handling. Dette kunne f.eks. være at skrive en tekst på skærmen eller læse nogle karakterer fra tastaturet. Ideen med at lave et program er ganske nærliggende, da procedurer på en eller anden måde skal gentages, så man ikke behøver at lave alt selv flere gange. Senere i din C# karriere vil du sandsynligvis lave dine egne metoder.

Metoder der er nær beslægtet, er en slags kærnefamilie i samme klasse.

Klasser som minder om hinanden er i samme namespaces, det skal her tilføjes at namespaces kan ligge i flere niveauer.

For at kunne anvende en metode skal man angive hvilken namespaces den ligger i, samt i hvilken klasse.

Metodekald output

Vi vil her gennemgå de mest elementære metoder og beskrive hvordan de kaldes.

Console.WriteLine

Metoden ”*Console.WriteLine”* er en kommando der bruges til at udskriv til skærmen

Den nemmeste måde man kan bruge denne metode, er at skrive det der skal skrives på skærmen mellem 2 gåseøjne. ”Dette skrives på skærmen”.

Eksempel 1:

```
Console.WriteLine(”Hej, jeg hedder Ib”);

Console.ReadKey();
```

Dette eksempel vil udskrive følgende:

```
Hej, jeg hedder Ib
```

Eksempel 2:

```
Console.WriteLine(”45”);

Console.ReadKey();
```

Dette eksempel vil udskrive følgende:

```
45
```

Eksempel 3:

```
Console.WriteLine(”Ib er 45 år );

Console.ReadKey();
```

Dette eksempel vil udskrive følgende :

```
Ib er 45 år
```

Vi kommer tilbage til WriteLine senere.

Farver

Når vi snakker farver er der to typer, forgrundsfarven som er farven på bogstavet og baggrundsfarven.

For at skifte farve skal man give en variabel en værdi, som vi senere vil blive bekendt med.

Eksempel 1:

```
Console.ForegroundColor = ConsoleColor.Red;
```

Her fortæller man at forgrundsfarven skal være rød.

Eksempel 2:

```
Console.ForegroundColor = ConsoleColor.Blue;
```

Her fortæller vi så den skal være blå;

Ligeledes kan vi sætte baggrundsfarmen.

```
Console.BackgroundColor = ConsoleColor.Yellow;
```

Nu bliver baggrundsfarven gul.

Rense skærmen.

Når man starter et program er det smart at rense skærmen, så der ikke står noget gammelt på skærmen.

Til dette formål bruges metoden Clear.

Eksempel 1:

```
Console.Clear();
```

Når man kalder denne metode sletter den alt der står på skærmen, og placerer cursoren oppe i venstre hjørne af skærmen, men initialiserer stadig hele skærmen med sidst valgte baggrundsfarve.

Placering af cursoren

Man kan placere cursoren efter ejet valg et sted på skærmen inden man skriver på skærmen, så det kommer til at se pænt ud.

Til dette formål bruges metoden ”*SetCursorPosition”*.

Eksempel 1:

```
Console.SetCursorPosition(20,6);
```

Metoden skal have 2 værdier med i kaldet. Den første er x værdien og den anden er y værdien.

X værdien er den vanrette linje, som starter i venstre side og y værdien er den lodrette og starter for oven og går nedad. I dette eksempel vil den placere sig 20 karaktere inde og 6 linjer nede.

Output

Her kommen et eksempel på brug af gennemgået output metoder.

```
Console.ForegroundColor = ConsoleColor.DarkRed;
Console.BackgroundColor = ConsoleColor.Yellow;
Console.Clear();
Console.SetCursorPosition(20,6);
Console.WriteLine("Dette er vores andet program");
Console. ReadKey();
```

Her vil forgrundsfarven blive mørkerød og baggrundsfarven blive gul, derefter vil den rense skærmen, placere cursoren og til sids skrive: ***Dette er vores andet program på skærmen***.

Variabler

En variabel bruges til at gemme informationer mens programmet afvikles. Når programmet skal bruge informationen som er gemt i variablen hentes denne frem fra hukommelsen. For at programmet skal kunne bruge en variabel skal den først erklæres.

Der findes groft sagt 3 typer variabler.

Heltals variabler er variabler som indeholder hele tal, så som: 4, -23 eller 1234.
Eksempelvis en int32 (int), som er en 32 bits størrelse.

Decimaltal er variabler som indeholder kommatal, så som: 23,34, -1,12 eller 234,345.
Eksempelvis er en double: 64 bits størrelse, med 15 ciffers nøjagtighed.

Tekst variabler hedder strenge. En string kan betragtes som en række celler, hvor hver celle indeholder en karakter (bogstav, tal eller tegn) eller som eksempel sagt på en anden måde vil bynavnet *"Ballerup"* fylde 8 celler *plus en* til at angive at strengen er slut. Men det vender vi tilbage til senere.

De 3 variabler.

Type	Navn	Formål	Størelse
Int32	int	Heltal	(-2.147.483.648 til 2.147.483.647)
Double	double	kommatal	(-5,0 X 10^{-324} til 1,7 x 10^{308})
String	string	Tekst	(0 til 2.000.000.000 tegn)

Erklæring af variabel.

Når man skal anvende en variabel skal man først *erklære den*. I denne erklæring gives besked om hvor meget plads der skal reserveres i computeres hukommelse.

Dette gøres ved først at skrive hvilke type variabel man vil anvende, give variablen et navn *efter eget valg* og til sidst kan man endda også give variablen en værdi.

Som type kan man både anvende typenavnet og C# navnet. Det er altså lige meget om man skriver int eller Int32.

Eksempel 1:

```
int Varnavn1;
```

Eksempel 2:

```
int Varnavn2 = 34;
```

Eksempel 3:

```
Int32 Varnavn3;
```

Eksempel 4:

```
Int32 Varnavn4 = 56;
```

Eksempel 5:

```
double Varnavn5;
```

Eksempel 6:

```
Double Varnavn6 = 34,56;
```

Eksempel 7:

```
string Navn1;
```

Eksempel 8:

```
String Navn2 = "TEC";
```

Console.WriteLine 2

Hvis vi lige vender tilbage til metoden *"Writeline"* som kan udføre en række opgaver. Vi har tidligere brugt metoden til at udskrive tekst direkte på skærmen ved at skrive teksten i gåseøjne. Metoden kan også udskrive variabler, samt formatere disse udskrifter.

Man kan ved at skrive *variabel navnet* inde i parentesen udskrive værdien af variablen automatisk, men dette kræver selvfølgelig at variablen har en værdi, ellers kommer der en fejlmelding.

Eksempel 1:

```
int Varnavn1 = 45;

Console.WriteLine(Varnavn1);
```

Eksempel 2:

```
string tekst = "Ballerup";

Console.WriteLine(tekst);
```

Man kan også blande de 2 måder at bruge WriteLine på.

Man kan skrive en tekst og indsætte variabler i teksten.

Eksempel 3:

```
string tekst = "Ballerup";
Int32 Postnummer = 2750;

Console.WriteLine("Jeg går i skole i {0} som har postnummer{1}"  ,tekst,Postnummer);
```

Der hvor variablen skal stå sætter man en { *"tuborgparantes start"* og en } *"tuborgparates slut",* med et nummer inden i, startende med 0.

Når man skriver 0, får man den første variabel der står efter kommaet og når man skriver 1 får man den næste. I dette tilfælde kommer den udskrevne tekst til at stå ved 0'et og Postnummer ved 1 tallet.

Eksempel 3 vil udskrive: ***Jeg går i skole i Ballerup som har postnummer 2750***

Formatering af udskrifter med WriteLine

Hvis man ikke skriver nogle formaterings tegn, vælger c# selv hvordan det skal formateres. I nogle tilfælde er det ikke smart og derfor kan man lave sine egne formatering tegn.

Eksempler på formateringskoder.

Tegn	Beskrivelse	Eksempel	Output
N2	Udskriver med 2 decimaler	("{0:N2}", 123,4)	123,40
C	Udskriver den lokale valuta	("{0:C}", 356)	kr. 356,00
,3	Udskriver 22 i et felt højrestillet	3("{0, 3}", 22)	22
X	udskriver i Hexadecimalt	("{0:X2}", 15)	0F
%	Udskriver tallet med procent tegn	("{0:0%}", 2567)	2567%

For at anvende disse formaterings tegn, skal man mellem de 2 tuborgparanteser skrive et colon efter tallet og derefter formateringstegnet.
En undtagelse i denne sammenhæng, hvis man ønsker at højrestille teksten, bruges blot et komma.

Eksempel 1:

```
Double kommatal = 123.12345;
Int32 heltal = 2750;

Console.WriteLine("Kommatal med 2 decimaler {0:n2} og heltal i HEX,
{1:X}",kommatal,heltal);
```

Dette vil udskrive ***Kommatal med 2 decimaler 123,12 og heltal i HEX, ABE***

Læg mærke til at når vi videregiver værdien til variablen bruger vi punktum som komma, men i udskriften kommer der et rigtigt komma.

Dette skyldes at programmerings sproget er engelsk, som konsekvent bruger komma og punktum omvendt, men Windows er en dansk udgave.

Input.

Et program er ikke meget værd hvis man ikke kan komme med input fra tastaturet.

Når skal have input fra brugeren sker det ved at man taster *"noget ind"* på tastaturet som bliver gemt i en variabel. Variablens type afhænger af hvad brugeren skal skrive, og i de fleste tilfælde er det en streng vi skal bruge. Denne tekst kan så laves om til et tal. Strenge bruges mest til at udskrive til skærmen og kan ikke bruges direkte til beregninger.

ReadLine

Til input vil vi i første omgang bruge metoden *"ReadLine".*
Når man kalder metoden ReadLine vil den returnerer en værdi/tekst når brugeren trykker på enter. Returværdien for ReadLine er en string, derfor er vi nød til at erklære en variabel af typen string, før vi bruger ReadLine.

Da Readline returnerer en værdi, skal variabel navnet stå før metode kaldet.

Eksempel 1:

```
String Indput;

Console.Write("Hvad er det fedeste fag her på skolen ");
Indput = Console.ReadLine();
Console.WriteLine("Nå! Du kan best lide {0}",Indput);
Console.ReadKey();
```

Forklaring til koden.

Først skriver den en besked på skærmen, som brueren skal svare på.
Når brugeren skriver på tastaturet og trykker på enter vil svaret blive gemt i variablen *"Indput".* WriteLine vil da udskrive ***Nå! Du kan best lide Programmering*** ,som man forhåbentligt svarede i programmet til spørgsmålet.

Eksempel 2:

```
String Mad;

Console.Write("Hvad er din livret?
Mad = Console.ReadLine();
Console.WriteLine("Nå {0} er din livret", Mad);
Console.ReadKey();
```

Forklaring til koden.

I dette eksempel skal du lægge mærke til at variablen Mad udskrives inde i teksten.

Indlæsning af tal

Da det er smart at indlæse variabler som tekst og vi har brug for tal, må vi lave tekst variabler om til talvariabler. Dette kan gøres på 2 måder og det er op til dig hvilken du bruger.

Convert og Parse

Convert

For at bruge convert skal klassen convert kaldes med en af klassens metoder.

Først skriver man retur variablen, som er talvariablen. Derefter skriver vi navnet på klassen(convert) efterfulgt at et punktum, samt metoden(ToInt32). Inde i parenteserne skriver man tekstvariablens navn.

Eksempel 1:

```
String TekstTal;
Int32 Tal;

Console.Write("Skriv et tal: ");
TekstTal = Console.ReadLine();
Tal = Convert.ToInt32(TekstTal);
Console.WriteLine("{0} er dit tal.", Tal);
Console.ReadKey();
```

I dette eksempel vil det der står i TekstTal, blive kopiret over i Tal variablen som en tal variabel.

Parse

Parse er en tilføjelse til variablen, *et såkaldt "overload"*, som betyder vi laver en handling med variablen, der kopier indholdet at en tekst værdi over til en talvariabel. Overload principper er dog for viderekomne, så dette er ikke noget vi behøver at koncentrere os om, andet end at forstå at begrebet eksisterer.

Eksempel 2:

```
String TekstTal;
Int32 Tal;

Console.Write("Skriv et tal: ");
```

```
TekstTal = Console.ReadLine();
Tal = int.Parse(TekstTal);
Console.WriteLine("{0} er dit tal.", Tal);
Console.ReadKey();
```

Fidusen med at omdanne tekst til tal, er at vi så først kan bruge variablen til at regne videre med. Dette kunne være at lægge tal sammen, gange tal med hinanden eller ethvert andet problem man kunne ønske sig at løse.

Eller sagt på en anden måde:

En gammeldags skrivemaskine kan godt skrive tal, men den kan ikke regne.

Det kan en lommeregner, så det handler i bund og grund om at flytte sig væk fra skrivemaskinen i retning mod lommeregneren.

Eksempel 3:

```
String Temp;
Double Is,Penge,Tilbage;

Console.Write("Hvor mange penge har du: ");
Temp = Console.ReadLine();
Penge = Double.Parse(Temp);
Console.Write("Hvor meget koster den is du vil købe: ");
Temp = Console.ReadLine();
Is = Convert.ToDouble(Temp);
Tilbage = Penge - Is;
Console.WriteLine("Du har {0} kr. og har købdt en is til {1} kr. , så nu har du
      {2} kr. tilbage",Penge,Is,Tilbage);
Console.ReadKey();
```

I dette eksempel indtastede du hvor mange penge du har, som først kommer ind som tekst og bliver til tal med parse, som bliver konverteret til variabeltypen *"Double"*. Da vi har sikret os at variabelværdien *"Penge",* kan vi bruge tekstvariablen Temp igen. Denne gang bruger vi "*convert"* til at lave *Temp* om til en talvariabel.
Tilbage = Penge – Is er regnestykket, hvor vi så igen beregner hvor mange penge vi har tilbage.
"Console.WriteLine" udskriver alle tre talvariabler i rækkefølge 0'et, 1'tallet og 2'tallet, som bestemmer hvilken rækkefølge de kommer i.

IF – else

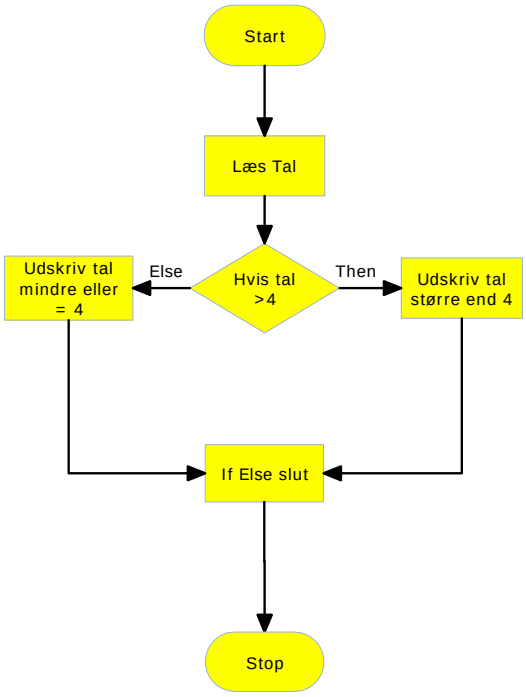
Struktur

```
if(betingelse)
{
  Sætninger
}
else
{
  Sætninger
}
```

Forklaring til ovenstående Struktur:

Hvis betingelsen i if-else sætningen er opfyldt, udføres den første sætningsblok. Ellers udføres den anden sætningsblok. Hvis der kun er én sætning i sætningsblokken kan man undlade { og }.

Flowchart



Eksempel på IF-else

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

namespace ifløkke
{
   class Program
   {
      static void Main()
      {
         //Initialisering af variable
         string vægt;
         float vægt1;

         //Indlæser vægten på brevet
         Console.Write("Indtast brevets vægt i gram : ");

         //Konverterer til kommatal
         vægt = Console.ReadLine();
         vægt1 = float.Parse(vægt);

         //Undersøger vægten på brevet, og skriver prisen

         if (vægt1 <= 0)
            Console.WriteLine("Prøv igen");
         else if(vægt1 <= 20.00)
         Console.WriteLine("Portoen er 5,50 Kr.");

         else if (vægt1 <= 100.00)
         Console.WriteLine("Portoen er 11,50 Kr.");

         else if (vægt1 <= 250.00)
         Console.WriteLine("Portoen er 23,40 Kr.");

         //Hvis vægten er over 250 gram udskrives næste linie
         else
         Console.Write("Dette er ikke et brev men en pakke");

         //Vent på tastetryk
         Console.ReadLine();
      }
   }
}
```

Forløkker

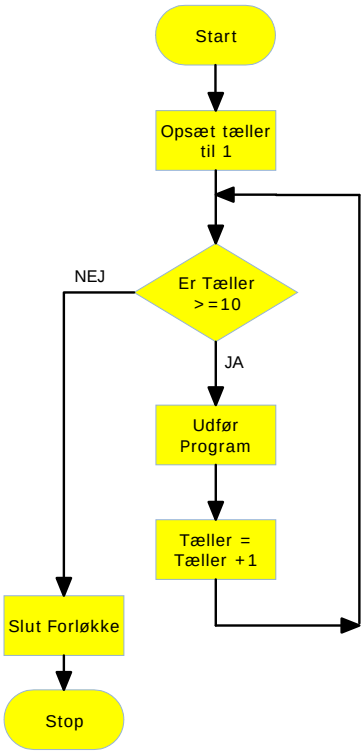
Struktur

```
for(startsætning; betingelse; slutsætning)
{
  Sætninger
}
```

Forklaring til ovenstående Struktur:

Først udføres startsætning. Så undersøges, om betingelsen har en værdi forskellig fra 0. Hvis dette er tilfældet udføres sætningsblokken, ellers springer kontrollen ud af forløkken. Efter at sætningsblokken eventuelt er udført, udføres slutsætningen. Herefter springer kontrollen til betingelsen, og det hele starter forfra.

Flowchart



Eksempel på program med forløkker

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

namespace forløkke
{
   class Program
   {
      static void Main()
      {
         for (int i = 1; i < 21; i++)

         Console.WriteLine("{0,2}",i);
         Console.ReadKey();

      }
   }
}
```

While

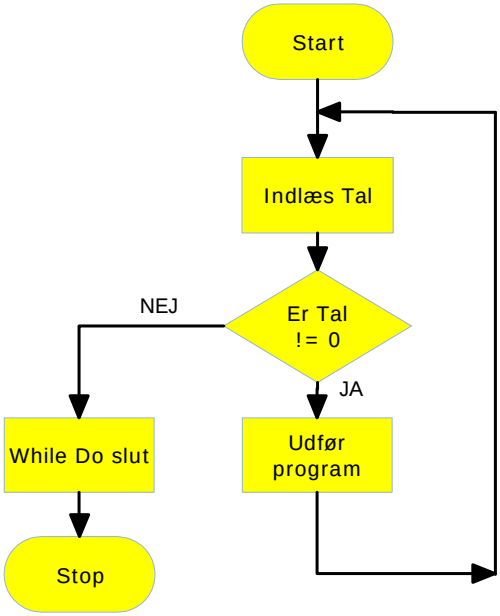
Struktur

```
while(betingelse)
{
   Sætninger
}
```

Forklaring til ovenstående Struktur:

Sætningerne i sætningsblokken bliver gentaget så længe betingelsen antager en værdi forskellig fra 0, dvs. så længe betingelsen er sand. Hvis der kun er én sætning i sætningsblokken kan man undlade { og }. Man anvender ofte While løkken til løkker, der skal gennemløbes et ukendt antal gange.

Flowchart



Eksempel på program med while

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

/* Programmet beder om at man indtaster minimum 5 tegn til sit password
 * er der ikke mindst 5 tegn bliver programmet ved med at spørge. Dette
 * foregåe i While løkken. */

namespace whiledo
{
   class Program
   {
      static void Main(string[] args)
      {
         string password;

         Console.Write("Indtast et password som er mindst 5 cifre: ");
         password = Console.ReadLine();

         while (password.Length < 5)
         {
            Console.WriteLine("Password skal være minimum 5 tegn");
            Console.WriteLine("Prøv igen. Indtast password");
            password = Console.ReadLine();
         }
         Console.Write("Mange TAK");
         Console.ReadKey();
      }
   }
}
```

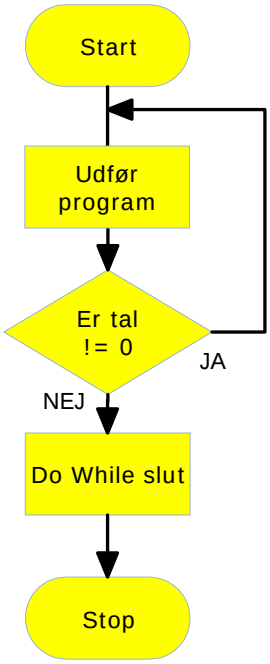
Do While

Struktur

```
do
{
  Sætninger
}
while(betingelse);
```

Forklaring til ovenstående Struktur:

En Do-While sætning gentages, indtil betingelsen får værdien 0 (falsk). Forskellen mellem en While-Do sætning og en Do-While sætning er at sætningsblokken til en Do-While sætning altid udføres mindst én gang. . Hvis der kun er én sætning i sætningsblokken kan man undlade { og }.Flowchart



Eksempel på program med Do While

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

namespace dowhile
   /* Programmet slår med en terning indtil at terningen viser 6.
    * når der kommer en sekser udskrives hvad terningen har vist under
    * forsøgene. Læg mærke til Objektet Random */
{
   class Program
   {
      static void Main()
      {
         Random RandomObj = new Random();
         int øjne;
         do
         {
            øjne = RandomObj.Next(1,7);
            Console.WriteLine("Terning viste: {0}",øjne);
         }
         while (øjne<6);
         Console.ReadKey();
      }
   }
}
```

Switch Case

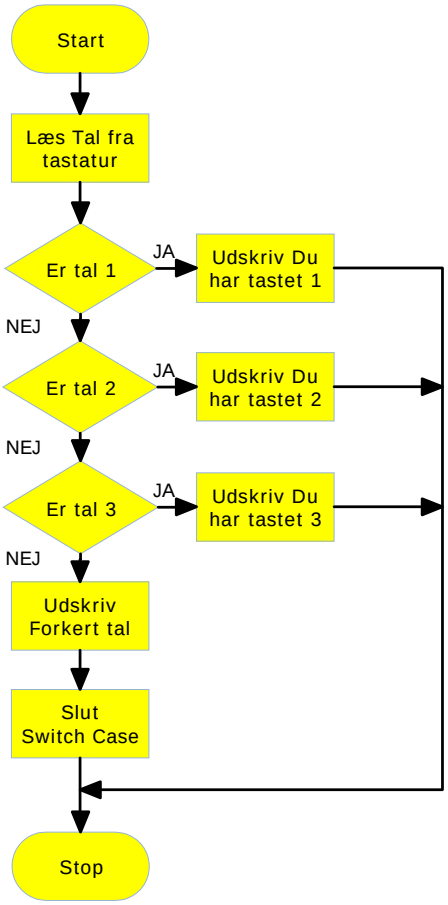
Struktur

```
switch(udtryk)
{
 case værdi_1:
   Sætninger_1
   Break;
  case værdi_2:
   Sætninger_2
   Break;
  case værdi_3:
   Sætninger_3
   Break;
O.S.V.
default:
  Sætninger_default
   Break;
}
```

Forklaring til ovenstående Struktur:

Hvis udtrykket har værdien værdi_1, bliver sætningsgruppen sætninger_1 udført, hvorefter at den øvrige del af Switch sætningen springes over pga. Break kommandoen. Det samme gælder hvis udtryk har værdien værdi_2 eller værdi_3, men så er det deres sætningsgrupper som bliver udført. Default til slut i Switch sætningen, vil blive udført, såfremt at udtryk ikke indeholder nogle af Case værdierne.

Flowchart


Start
Læs Tal fra tastatur
Er tal 1
JA
Udskriv Du har tastet 1
NEJ
Er tal 2
JA
Udskriv Du har tastet 2
NEJ
Er tal 3
JA
Udskriv Du har tastet 3
NEJ
Udskriv Forkert tal
Slut Switch Case
Stop

Eksempel på program med Switch Case

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

namespace switchcase
   /*Programmet beder om at man taster: 1, 2 eller 3. Derefter udskrives
    * hvad man har tastet. Taster man forkert forlades SwitchCase ved at
    * afvikle det som er skrevet under Default */
{
   class Program
   {
      static void Main()
      {
         int tast;
         Console.WriteLine("Tryk på 1 2 eller 3 på tastaturet: ");
         tast = Console.Read();
         switch(tast)
         {
            case '1':
             Console.WriteLine("  Du har trykket på 1 tast return");
             break;
            case '2':
             Console.WriteLine("  Du har trykket på 2 tast return");
             break;
            case '3':
            Console.WriteLine("   Du har trykket på 3 tast return");
            break;
            default:
            Console.WriteLine("   Du har ikke tastet 1,2,3 tast return");
            break;
         }
         Console.ReadKey();

      }
   }
}
```

Samlet oversigt over For, Do og While

- For løkker anvendes typisk til løkker, der skal gennemløbes et kendt antal gange.
- While-Do og Do-While løkker anvendes til løkker, der skal gennemløbes et ukendt antal gange.
- En Do-While løkke gennemløbes altid første gang, og derefter kun hvis dens betingelse er sand.
- En While-Do løkke gennemløbes kun, hvis dens betingelse er sand.
- En betingelse er sandt, hvis dens værdi er forskellig fra 0. Betingelsen er falsk, hvis dens værdi er 0.

Samlet oversigt over If-Else og Switch

- Man kan afgøre hvilke forskellige del af et program, der skal udføres med givne værdier af variablerne med sætninger med If, med If-Else og med Switch.
- Med If og med If-Else sætninger afgør de opstillede betingelser, hvilken gren af programmet kontrollen skal udføre.
- Med Switch sætninger afgør værdien af et udtryk, hvilken gren af programmet kontrollen skal vælge.

Strenge

Strenge er som før nævnt i C# en samling af karakterer. Du kan herunder kunne se nogle eksempler på, hvordan man behandler disse strenge.
Hvis man f.eks. har en streng og man ønsker at tilskrive strenge nogle værdier, kan man gøre som vist herunder:

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

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         string navn;
         string fornavn;
         string efternavn;
         fornavn = "Henrik";
         efternavn = " Andersen";
         navn = fornavn + efternavn;
         Console.WriteLine("navn: {0}", navn);
         Console.Read();

      }
   }
}
```

Ovennævnte program vil skrive Henrik Andersen i consolen når det afvikles.
En anden måde hvor man kan bruge en streng er at tildele strengen værdier efterhånden og så udskrive strengens indhold til sidst er vist herunder:

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

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         string samlettekst;
         samlettekst = ("Dette er et eksempel på ");
               samlettekst += ("hvordan at, man kan samle flere variabler i samme);
         samlettekst += (" tekst. \nMan skal bruge syntaksen +=, som så");
         samlettekst += (" tilskriver strengen værdierne efterhånden");
         Console.WriteLine("{0}", samlettekst);
         Console.Read();
```

```
        }
    }
}
```

Bilag

Ascii tabel

Når man skal bruge specielle tegn som ikke er på tastaturet kan gøre følgende:

Tryk på ALT samtidig med at du indtaster den decimale (DEC) kode på det numeriske tastatur.

På et almindeligt tastatur er det tallene til højre man skal bruge og på en bærbar er det talende i midten man skal bruge.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	