Campus Nyköping

PROGRAMMERING I .NET C#1 - 03

C#: Datatyper, villkor och loopar

Förra gången

- Ny kurs!
- Grunderna
  - Programmering
  - .NET
  - C#

Idag

- C#
  - Datatyper
  - Operatorer
  - Loopar

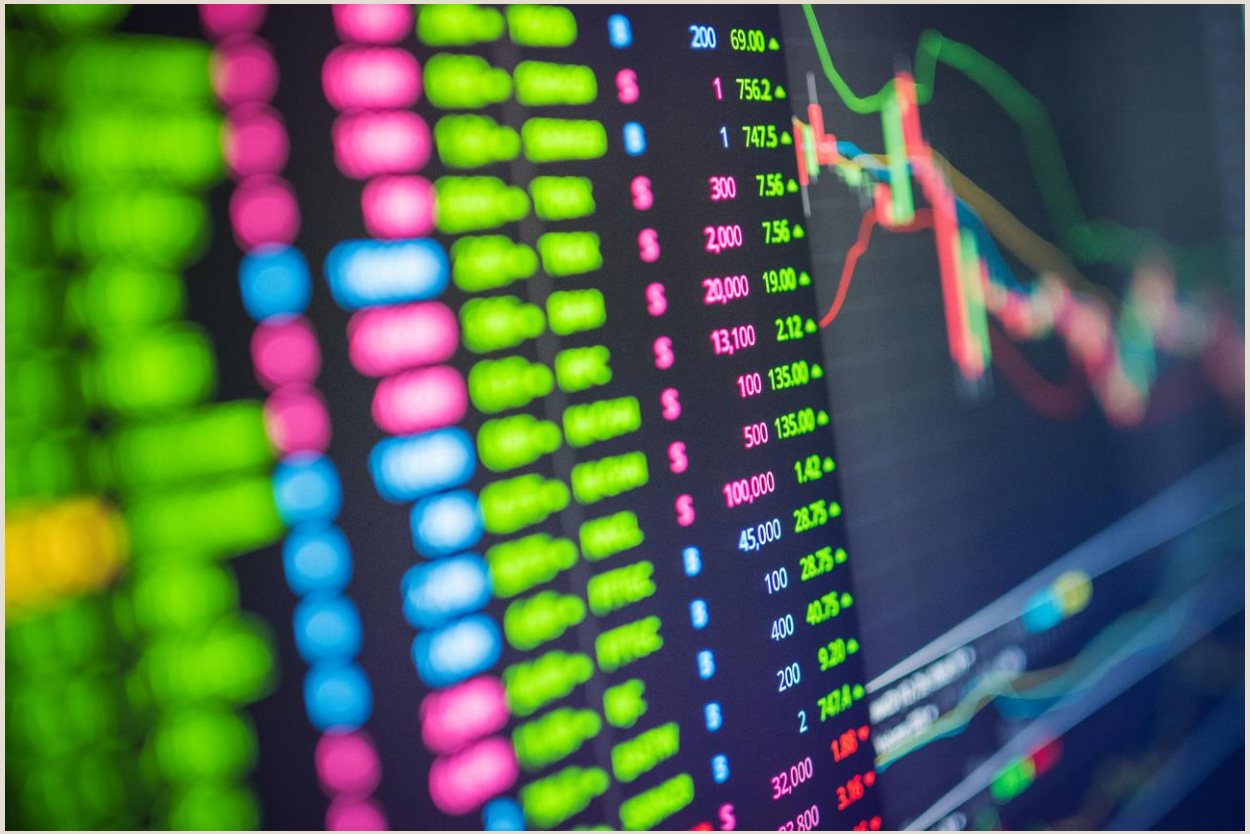
Först: Kommentarer

- En ibland viktig sak är att kommentera sin kod.
- Ha för vana att alltid kommentera din kod under kursen.
  - Flera som arbetar med samma kod
  - Äldre kod man glömt bort
  - Enklare att hitta till olika delar av koden
  - Ska göras med viss måtta
  - Men hellre att koden är självförklarande

- `// Skriver ut Hello World`
- `Console.WriteLine("Hello World!");`

- Flera rader:
  - `/* The code below will print the words Hello World`
  - `to the screen, and it is amazing */`
  - `Console.WriteLine("Hello World!");`

Mer om datatyper



Varför välja datatyp?

- Minnesåtgång
- Begriplighet
  - Man kan skriva såhär: string number = “142575”;, men det är inte lätt att förstå
- Framtida beräkningsbehov, t ex medelvärde
- Säkrare programmering. Konstiga värden kan inte ta sig in lika lätt.

Primitiva datatyper

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
bool	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter, surrounded by single quotes
string	2 bytes per character	Stores a sequence of characters, surrounded by double quotes

Datatyper

- `int myNum = 5;` `// Integer (whole number)`
- `double myDoubleNum = 5.99D;` `// Floating point number`
- `char myLetter = 'D';` `// Character`
- `bool myBool = true;` `// Boolean`
- `string myText = "Hello";` `// String`

var

- **var** låter kompilatorn själv bedöma vilken datatyp som ska användas.

- var myText = “Hejsan”
- var myNumber = 123

- Lätt att använda, men blir ibland fel…

Datatyper - string

- Datatypen string används för att lagra en sekvens av tecken (text). Strängvärden måste omges av dubbla citat
- `string greeting = "Hello World";`
- Ibland vill man ha tecknet " (citat-tecken) inne i en sträng, använd då escape-tecknet \.
- Exempel:
  - `Var text = ”Vad \”snäll\" du är idag”;`

Datatyper - int

- Datatypen **int** kan lagra heltal från -2 147 483 648 till 2 147 483 647 (Två miljarder)
- I allmänhet är datatypen **int** den datatyp som **föredras** när vi skapar variabler med ett numeriskt värde.
- `int myNum = 100000;`
- När fungerar inte int?
  - Större värden
  - Tal med decimal nogrannhet

Andra heltalstyper

- sbyte (-128 – 127): 8 bitar med tecken
- byte (0 – 255): 8 bitar utan tecken
- short (-32 768 – 32 767): 16 bitar med tecken
- ushort (0 – 65 535): 16 bitar utan tecken
- **int (-2 147 483 648 – 2 147 483 647): 32 bitar med tecken**
- uint (0 – 4 294 967 295): 32 bitar utan tecken
- long (-9 223 372 036 854 775 808 – 9 223 372 036 854 775 807): 64 bitar med tecken
- ulong (0 – 18 446 744 073 709 551 615): 64 bitar utan tecken

Datatyper - long

- Datatypen **long** kan lagra hela tal från -9 223 372 036 854 775 808 till 9 223 372 036 854 775 807. Detta används när int inte är tillräckligt stor för att lagra värdet. Observera att du *bör* avsluta värdet med ett "L”
- `long myNum = 1534578934958739L;`

Datatyper - float

- Du bör använda en flyttalstyp närhelst du behöver ett tal med en decimal, till exempel 9,99 eller 3,14515.
- Datatypen **float** kan lagra bråktal från 3,4e−038 till 3,4e+038. Observera att du bör avsluta värdet med ett "F”
- `float myNum = 5.75F;`

Datatyper - double

- Datatypen **double** kan lagra bråktal från 1,7e−308 till 1,7e+308. Observera att du kan avsluta värdet med ett "D" (även om det inte krävs)
- `double myNum = 19.99D;`

- Använda float eller double?
  - Precisionen för ett flyttalsvärde anger hur många siffror värdet kan ha efter decimalkommat. Precision på **float** är bara sex eller sju decimalsiffror, medan **double** har en precision på ca 15 siffror. Därför är det säkrare att använda double för de flesta beräkningar.

Datatyper - Decimal

- Det finns en speciell typ av flyttal som har högre precision än float eller double: decimal (±1,0 x 10-28 – ±7,9 x 1028): 128 bitar, 28 – 29 siffrors precision
- Används när man räknar med pengar eller andra tillfällen när precision är viktigt
- Inga avrundningsfel
- Nästan ingen förlust av precision
- Standardvärde för decimal är 0.0M (M är suffixet för sådana här decimaltal)

Datatyper - bool

- Datatypen **bool** deklareras med nyckelordet bool och kan bara ta värdena sant eller falskt
  - `bool isCSharpFun = true;`
  - `bool isFishTasty = false;`

Datatyper - char

- Datatypen char används för att lagra ett enskilt tecken. Tecknet måste vara omgivet av enstaka citat, som 'A' eller 'c'
- `char myGrade = 'B';`

Exempel

- byte centuries = 20; // Usually a small number
- ushort years = 2022;
- uint days = 730480;
- ulong hours = 17531520; // May be a very big number
- Console.WriteLine("{0} centuries is {1} years, or {2} days, or {3} hours.", centuries, years, days, hours);

Konvertera mellan datatyper

- Konvertera en sträng till en int.
  - int numberInt = int.Parse(number);
  - alla datatyper kan konverteras såhär: valfridatatyp.Parse(sträng)
- Förvandla en int till en float
  - float numberFloat = numberInt;
- Convert
  - Double doubleNum = Convert.ToDouble(numInt);
- Casting
  - int numInt = (int) numDouble;
- För att programatiskt ta reda på vilken datatype vi har på en variable:
  - Type t = numInt.GetType();

Variabelnamn

- ööå bör inte användas som variabelnamn om vi använder engelska namn.
- Vi kan inte ha två variabler som heter samma sak.
- Vi får också vara uppmärksamma på anda "låsta" namn som t ex int double string.
- Använder vi stora o små bokstäver i namnen så måste vi följa det exakt senare, d vs mittHELTAL och mittHeltal är inte samma variabel.
- Tydlighet är viktigt:
  - a = b * c;
    - är korrekt, men dess syfte är inte uppenbart. Jämför detta med:
  - weekly_pay = hours_worked * hourly_pay_rate;
- Det är att rekommendera att använda engelska variabelnamn.

Variabelnamn

Variable Name	**Legality**
Percent	Legal
y2x5__w7h3	Legal
yearly_cost	Legal
_2010_tax	Legal, but not advised
checking#account	Illegal; contains the illegal character #
double	Illegal; is a C keyword
9byte	Illegal; first character is a digit

Variabelnamn

- ANVÄND variabelnamn som är beskrivande.
- Anta och hålla fast vid en stil för att namnge dina variabler.
- Namnge INTE dina variabler med bara versaler i onödan.

Variabelnamn – C#

Variabelnamn I C# rekommenderas skrivas med sk lowerCamelCase

Första ordets första bokstav skrivs med liten bokstav.
Andra, och de följande ordens första bokstav skrivs med stor bokstav.

Exempel:
yearly**I**ncome, **m**y**B**irth**C**ity

Kind	Rule
Private field	_lowerCamelCase
Public field	UpperCamelCase
Protected field	UpperCamelCase
Internal field	UpperCamelCase
Property	UpperCamelCase
Method	UpperCamelCase
Class	UpperCamelCase
Interface	IUpperCamelCase
Local variable	lowerCamelCase
Parameter	lowerCamelCase

Operatorer

- Addition: Operatorn + adderar två operander. Till exempel x+y.
- Subtraktion: Operatorn - subtraherar två operander. Till exempel x-y.
- Multiplikation: Operatorn * multiplicerar två operander. Till exempel x*y.
- Division: Operatorn / delar den första operand med den andra. Till exempel x/y.
- Modulus: Operatorn % returnerar återstoden när första operand divideras med den andra. Till exempel x%y.
  - 10 % 5 = 0 eftersom 10 är jämt delbart i 5
  - 9%4 = 1 eftersom 9 inte är jämt delbart med 4, utan lämnar 1 i rest.

Code along - Kalkylatorn

- Kod: Calculator

Bool och villkor

Först lite parenteser

- { och }
  - Klammerparenteser
    - De används i diverse programspråk för att markera början och slut på kod-block.
    - Andra namn: måsvingar, krullparentes,
- [och]
  - Hakparentes
    - Används bl a för matriser och listor.
- (och)
  - Parentes eller Rundparentes
    - Anger grupper av termer eller tal

Datatypen bool

- Deklareras med nyckelordet bool
- Har två möjliga värden: true och false
- Är användbart i logiska uttryck och villkor
- Standardvärdet är false

Villkor - Jämförelseoperatorer

- De vanligaste villkoren
  - Lika med: a == b (dubbla likamed-tecken)
  - Mindre än: a < b
  - Mindre än eller lika med: a <= b
  - Större än: a > b
  - Större än eller lika med: a >= b
  - Inte lika med: a != b

  - Villkoret ska vara **sant** för att något ska hända

Jämförelseoperator

- Lika ==
- Olika (ej lika) !=
- Större än >
- Större än eller lika >=
- Mindre än <
- Mindre än eller lika <=
- Exempel:
  - `bool result = (5 <= 6);`
  - `Console.WriteLine(result); // True`

Exempel

```
int a = 1;
int b = 2;
bool greaterAB = (a > b);
Console.WriteLine(greaterAB); // False
bool equalA1 = (a == 1);
Console.WriteLine(equalA1); // True
```

Fler exempel

- age <= 100
- temp > 0
- income > 10000
- name == “micke”
- newsletter == true

Villkor

- Vi ska gå igenom tre villkorssatser, som avgör om delar koden ska köras, eller inte:
  - if
  - else
  - elseif

Villkorssats - if

```
if(villkor)
{
  Kod…
}
```

Koden mellan klammerparenteserna körs enbart om ett givet villkor är sant.

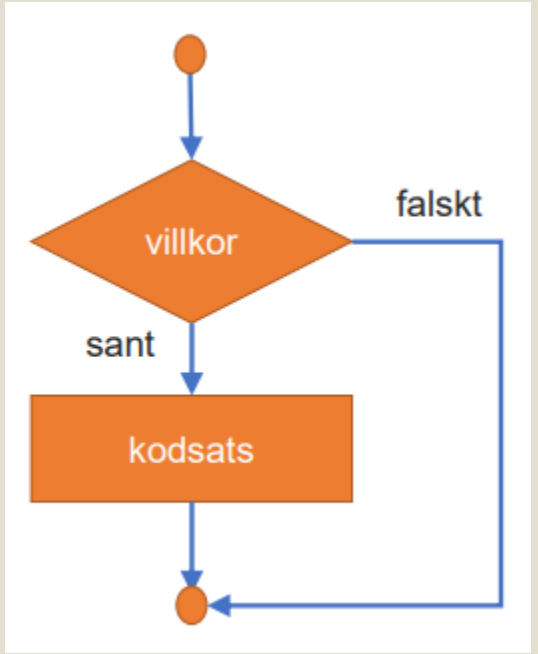
Det körs bara en gång.

Villkor

- Den viktigaste delen I de här kodexemplen är **villkoret**.
- Det gäller att skriva ett villkor som stämmer med det vi vill åstadkomma.
- Vanligaste villkoret är att en viss variabel når ett visst värde.

Villkorssats – If - Exempel

```
If(name == “micke”)
{
    Console.Write(“Välkommen hem”);
}
```



Logiska operatorer

- Logisk INTE Logical NOT !
- Logisk OCH Logical AND &&
- Logisk ELLER Logical OR ||

Flera villkor

- Med hjälp av OCH och ELLER kan man sätta upp flera villkor på samma gång.
- Exempel

```
If(name == "micke" && timme > 18)
{
        Console.Write("God kväll, herr Engström");
}
```

&& betyder OCH: Båda villkoren måste vara sanna

|| betyder ELLER: minst ett av villkoren måste vara sanna.

Exempel:

```
if(timme < 9 || timme > 18)
{
        Console.Write("Butiken är stängd");

}
```

Fler villkor och Else

```
If(name == “micke” || name = “håkan”)
{
    Console.Write(“Lärare”);
}
else
{
    Console.Write(“Elev”);
}
```

Fler villkor

```
If(name == “micke” || name = “håkan”)
{
    Console.Write(“Lärare”);
}
Else if (name == “christer”)
{
    Console.Write(“Chefen”);
}
Else
{
    Console.Write(“Elev”);
}
```

Loopar

Loopar

- Att upprepa sin kod oändligt eller ett visst antal gånger, är fundamental för programmering. Vi har flera olika sätt att upprepa koden och styra programmet, och vi ska idag gå igenom följande tre varianter av loopar:
  - while
  - do…while
  - for

Loopar - while

```
while(villkor)
{
    Kod...
}
```

Koden mellan klammerparenteserna upprepas så länge ett givet villkor är sant.
Den testar tillståndet innan du utför loopen.

Loopar – do…while

```
do
{
    Kod...
}
while(villkor)
```

Koden mellan klammerparenteserna upprepas så länge ett givet villkor är sant.
Den testar tillståndet efter du utför loopen. Det innebär att loopen alltid körs minst en gång.

Loopar - for

```
for(int x = startvärde; villkor för x; inkrementering)
{
   Kod...
}
```

Koden mellan klammerparenteserna upprepas så länge ett givet villkor är sant.

Värdet på x räknas samtidigt upp eller ner.

Loopar - while

```
while(true)
{
    Kod...
}
```

- Enklaste exemplet: En oändlig loop, eftersom villkoret alltid är sant!

Loopar - for

```
for(int x = 1; x <= 10; x++)
{
        Console.Write(x);
}
```

Loopar - while

```
int x = 0;
while(x < 100)
{
        Console.Writeline(x);
        x = x + 1;  // Viktigt!
}
```

- Andra sätt att skriva x = x + 1;
  - X++
  - x += 1

Loopar – do…while

```
int x = 0;
do
{
        Console.Writeline(x);
        x = x + 1;  // Viktigt!
}
while(x < 100)
```

Code-along

- Demo av villkor och loopar

Övningar - Blandat

- Skapa en Console Application som:

1. …frågar om AIK (eller annat valfritt lag) är Sveriges bästa lag. Om användaren svarar ja skriver programmet ut ”Helt rätt!”, annars skrivs det ut ”Väldigt fel!”.
2. …frågar efter temperaturen i Svedala och Jukkasjärvi. Programmet berättar sedan var det är kallast.
3. …lägg till Visby på förra punkten och berätta var det är varmast.

Övning - Omvandlaren

- Uppgift från förra gången, förtydligande
- Konvertera mellan olika enheter
  - Skapa en consoleapplikation som tar in ett valfritt numreriskt värde.
  - Visa en lista på ett antal vikt och längd-omvandlingar.
    - T ex om du matar in siffran 12 ska listan se ut ungefär såhär:
      - 12 meter = 13.12 yards
      - 12 kg = 26.45 pound (lb)
      - 12 grader Celsius = 53.6 Fahrenheit
      - 12 kilowatt = 16.08 hästkrafter
      - …lägg gärna på fler konverteringar
- Länkar:
  - https://www.mathsisfun.com/metric-imperial-conversion-charts.html
  - https://www.mathsisfun.com/temperature-conversion.html
  - https://www.mathsisfun.com/unit-conversion-tool.php

Övning - Sagan2

- Utgå från övningen: Sagan, och gör extrauppgiften, så du kan ange om djuret är en *han* eller *hon*.

- Uppgiften var:
  - Sagan: Skapa en applikation som bygger en saga enligt följande upplägg:
    - Det var en gång en <ålder> år gammal <djursort> som hette <namn>. En dag var <namn> ute på en promenade I skogen, och mötte en stor varg. Vargen bet av ett ben **på <honom eller henne>.** <namn> sprang snabbt hem på sina <antalben – 1>. Så var sagan slut.

    - Extrauppgift: Lägg till inmatningsfält för om djuret är hona eller hane.
      - Utgå från din befintliga kod, skriv inte om programmet från början.

Övning – Räkna till hundra

- Skapa en Console Application som med hjälp av en **while loop** skriver ut alla tal mellan 1 och 100, och markerar jämna nummer med en hakparentes omkring sig. (1 [2] 3 [4] 5 [6]...)
- Gör samma sak med en ”for-loop” istället. Markera alla ojämna nummer.

Övning multiplikationstabellen

- Gör en console-application som skriver ut multiplikationstabellen enligt nedanstående exempel, ändå till 10x10:

```
    1  2  3...
1   1  2  3
2   2  4  6
3   3  6  9
...
```

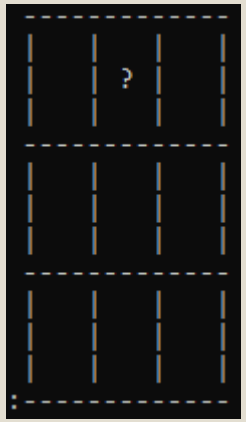
Övning – Bish-Bosh

- Skriv ett console-app som heter ‘Bish-Bosh’.
- Bish-Bosh listar alla tal mellan 1 och valfritt tal, som ska matas in av användaren.
- Det ska gå att välja två ytterligare tal, sk Bish och Bosh-tal, även de matas in av användaren.
- Sedan loopas talen från 1 till det valfria talet igenom.
- Om talet är jämnt delbart med Bish-talet visas ‘Bish’ istället för talet.
- Om talet är jämnt delbart med Bosh-talet visas ‘Bosh’ istället för talet.
- Om talet är jämt delbart med både Bish och Bosh-talet visas ’Bish-Bosh’ istället för talet.
- I exemplet är Bish-talet 4 och Bosh-talet är 6

1	2	3	Bish	5	Bosh	7	Bish	9
10	11	Bish-Bosh	13	14	15	Bish	17	Bosh
19	Bish	21	22	23	Bish-Bosh	25	26	27
Bish	29	Bosh	31	Bish	33	34	35	Bish-Bosh
37	38	39	Bish	41	Bosh	43	Bish	45
46	47	Bish-Bosh	49	50	51	Bish	53	Bosh

Svårare övning – “Labyrinten”

Se separate document: Övningsuppgift - Labyrinten



Länkar

- C# Data Types (w3schools.com)
- C# Type Casting (w3schools.com)
- C# Operators (w3schools.com)
- C# Booleans (w3schools.com)
- C# If ... Else (w3schools.com)
- C# While Loop (w3schools.com)
- C# For Loop (w3schools.com)

- C# Type Conversion (With Examples) (programiz.com)
- https://www.tutorialspoint.com/csharp/csharp_loops.htm
- https://www.tutorialspoint.com/csharp/csharp_decision_making.htm