C# Grunder

V34

Vad är Programmering?

- Ett sätt att skriva instruktioner till en dator
- Olika språk är bra på olika saker
- Datorer är inte smarta
- Kompilerade språk och Översatta språk
  - Exempel på kompilerade språk: C#, Java, C, C++, Kotlin
  - Exempel på översatta språk: Python, JavaScript, Ruby

Data

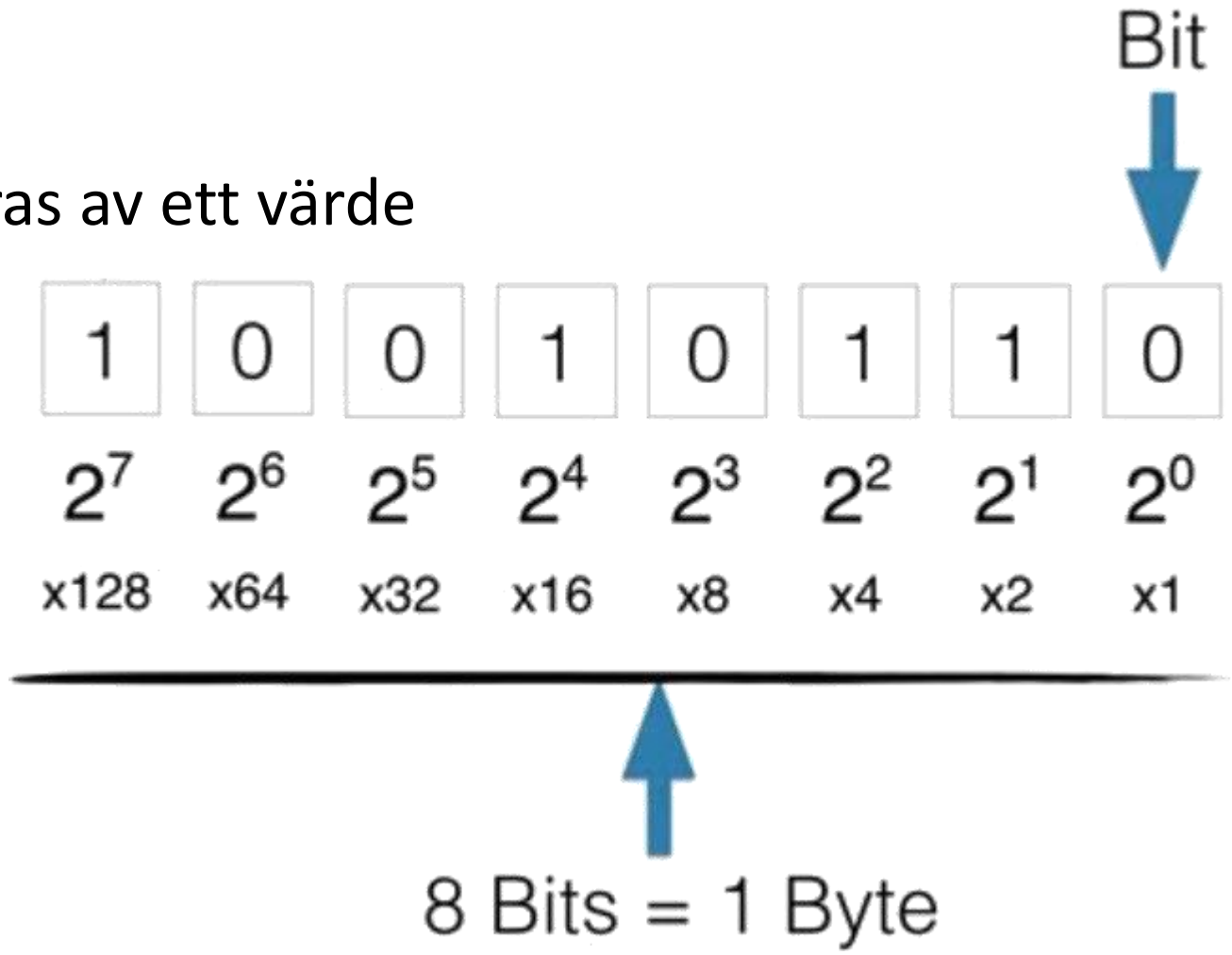
V34

Binära tal

- Vad är en bit?
  - En bit är ett tillstånd. Av eller på
  - Representeras av ett binärt tal
    - Alltså ett tal med basen 2 (0 eller 1)
- Vad är en Byte?
  - En byte är en samling av åtta bitar (01010101)

Omvandla Binära tal

- Varje bit i en byte representeras av ett värde
- 00000001 = 1
- 00000010 = 2
- 10000000 = 128
- 11111111= 255



Datamängder

- Byte (B) – 8 bitar
- Kilobyte (kB) –> 1000 bytes –> 8000 bitar
- Megabyte (MB) –> 1 000 000 bytes –> 8 000 000 bitar
- Gigabyte (GB) –> 1 000 000 000 bytes –> 8 000 000 000 bitar
- Terabyte (TB) –> 1 000 000 000 000 bytes –> 8 000 000 000 000 bitar

- Mb/s är alltså INTE det samma som MB/s.
- 1 Mb/s är alltså 125 kB/s

Övning binär data

- 01001000 01100101 01101010
- 01001110 01101001 01101011 01101100 01100001 01110011
- 01001001 01010100 00101101 01001000 11000011 10110110 01100111 01110011 01101011 01101111 01101100 01100001 01101110
- 01000111 11000011 10110110 01110100 01100101 01100010 01101111 01110010 01100111
- <Ditt namn>
- Wake up Neo
- Hur många bytes krävs för att representera din adress?
- 01000010 01110010 01110101 01101110 01110100 01100101 01100010 01100001 01100011 01101011 01100101 01101110 00100000 00110010 00110100 00001010 00110100 00110010 00110101 00110100 00110010 00100000 01001000 01101001 01110011 01101001 01101110 01100111 01110011 00100000 01001011 11000011 10100100 01110010 01110010 01100001

Filtyper i C#

Filtyper och filändelser

- **.cs** – C#-filer, alltså de filer vi fyller med kod
- **.csproj** – Projektbeskrivning, här beskrivs vilka paket och filer som ingår i ett projekt.
- **.sln** – Solutionbeskrivning, här beskrivs vilka projekt, paket, relationer som finns mellan dessa
- Obligatorska för att bygga ett projekt är minst en **.cs**-fil och en **.csproj**-fil. Solutionfilen krävs främst då man ska koppla flera saker utifrån in till ett projekt, tex flera projekt

Syntax

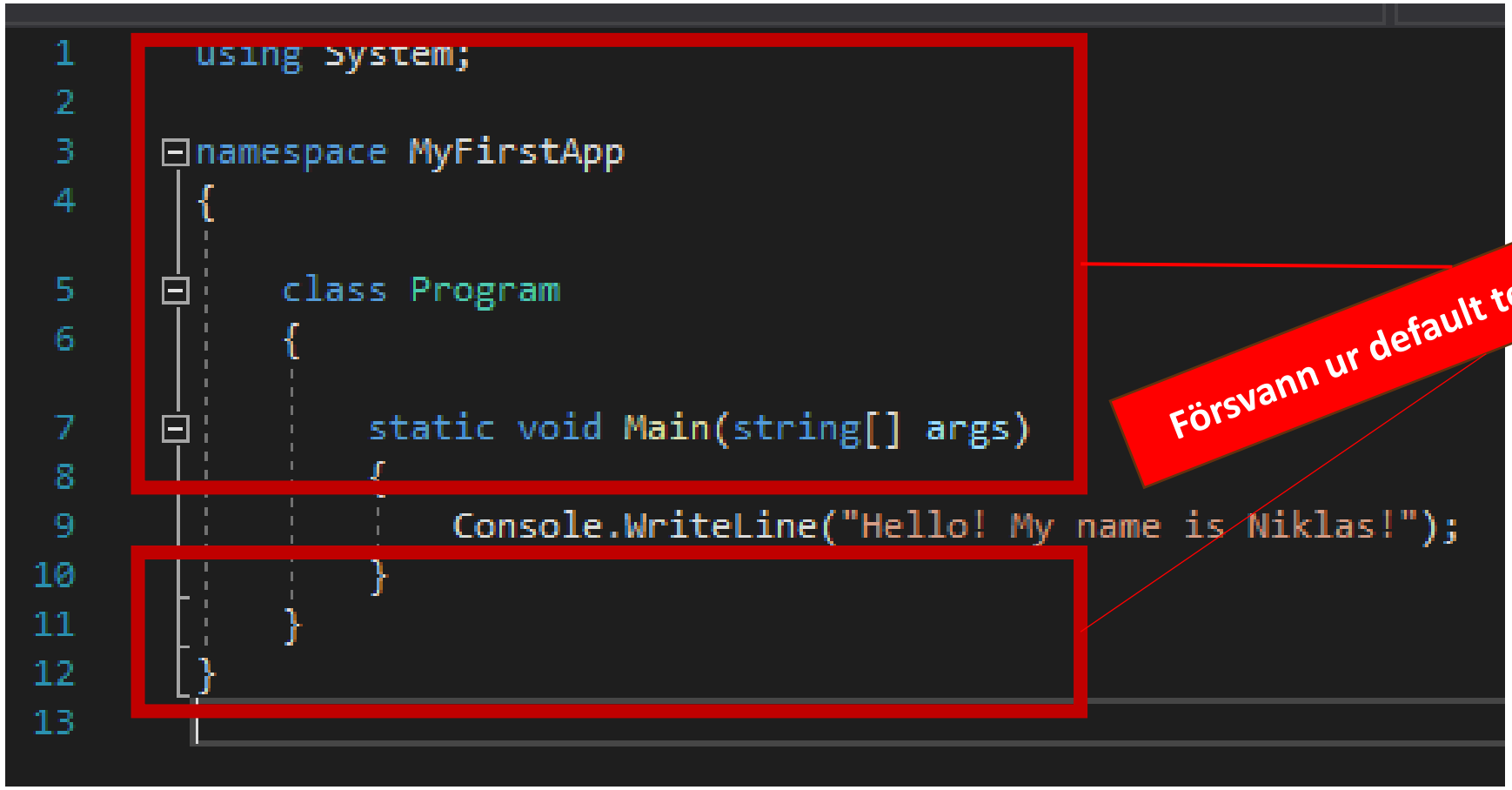
Syntax

- Definitionen av ordet Syntax
  - Läran om hur ord förbinds med varandra till högre enheter dvs. ordgrupper, satser och meningar
- För oss
  - Hur man skriver kod i ett specifikt språk

Syntax i C#

- C# är ett Objektorienterat kompilerat språk
- Kod inom ett kodblock körs i sekvens. Alltså från topp till botten
- Ett kodblock startar med **{** och avslutas med **}**
  - Måsvinge/Krullparentes/Curly braces/Curly brackets
- En rad avslutas alltid med **;**

Syntax i C#


using System;
namespace MyFirstApp
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Hello! My name is Niklas!");
}
}
}
Försvann ur default template i .NET 6

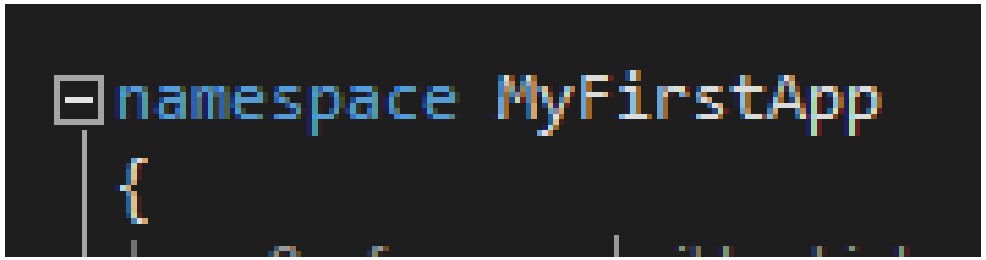
Using

- Using indikerar att man vill hämta och ha tillgång till ett bibliotek.
- Man kan ha manga “using” i samma fil.

```
using System;
```

Namespace

- Namespace är en identifierare för att kunna samla många klasser i ett paket.



Main

- Main är en metod.
  - En metod är en samling kodrader.
  - Main är den metod som körs först av allt i ett C#-program.

```
static void Main(string[] args)
{
    Console.WriteLine("Hello! My name is Niklas!");
}
```

Class

- Kortfattat är en klass en ritning för ett object.
  - Klasser används för att dela upp kod efter syfte.
  - Klasser kan vara enbart databehållare
  - Klasser kan också vara en samling körbar kod.

Kommentarer

- Kommentarer är rader som inte körs
- Kommentarer bör användas för att dokummentera sin kod
- Tips! Skriv kommentar först, sedan koden som utför det man skrivit.
- Kommentarer på en rad startar med **//**
- Kommentarer på flera rader kan göras med **/*** i början och ***/** i slutet

Variabler

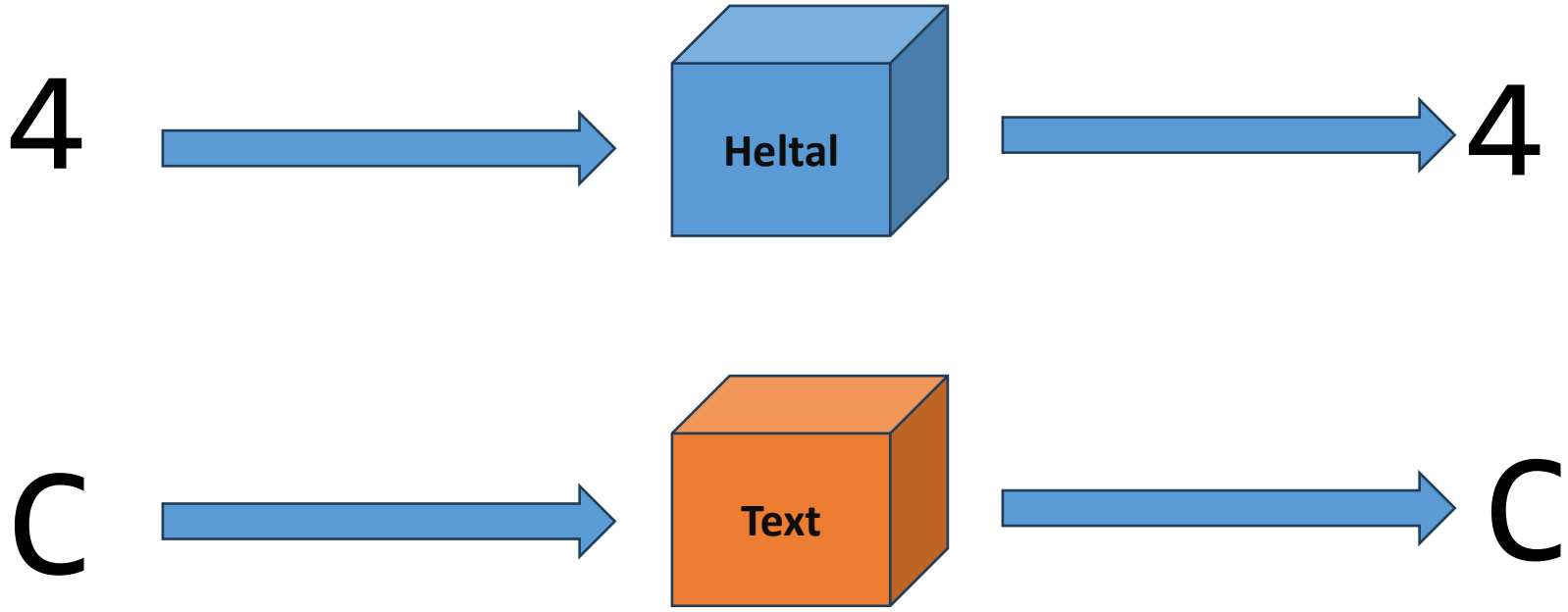
V34

Vad är en variabel?

- En variabel är en behållare för data
- En variabel måste först deklareras innan den används
- När en variabel deklareras anger man en datatype och ett namn
- Datatypen beskriver vilken sorts data som kommer att lagras och hur stort utrymme den kommer att ta upp
- Namnet är något vi väljer själva ochbör beskriva vad variabeln kommer att innehålla eller användas till

Data

Variabler



Numeriska Datatyper

V34

Integer - `int`

- Storlek: 4 Bytes
- Positiva och negativa heltal
- Minsta talet är -2 147 483 648
- Största talet är 2 147 483 648

```
int integer = 42;
```

Long - `long`

- Storlek: 8 Bytes
- Positiva och negativa heltal
- Minsta talet är -9 223 372 036 854 775 808
- Största talet är 9 223 372 036 854 775 808

```
long longInt = 42L;
```

Float - `float`

- Storlek: 4 Bytes
- Positiva och negativa decimaltal
- Minsta talet är 1.5×10^{-45}
- Största talet är 3.4×10^{38}
- Begränsning på 7 siffror

```
float decimaltal = 3.14F;
```

Double - **double**

- Storlek: 8 Bytes
- Positiva och negativa decimaltal
- Minsta talet är 5.0×10^{-345}
- Största talet är 1.7×10^{308}
- Begränsning på 15 siffror

```
double långtDecimaltal = 3.14;
```

Icke-numeriska Datatyper

V34

Character - **char**

- Storlek: 2 Bytes
- Enskilda tecken

```
char character = 'c';
```

String - `string`

- Storlek: 2 Bytes/tecken
- Text bestående av flera tecken

```
string text = "Massa text";
```

Boolean - `bool`

- Storlek: 1 bit
- Värde `true` eller `false`
- Används för att spara resultatet av en jämförelse

```
bool boolean = true;
```

Operatorer

V34

Aritmetriska operatorer

- Addition **+**
- Subtraktion **-**
- Multiplikation *****
- Division **/**
- Modulus (rest) **%**

Jämförelseoperatorer

- Större än `>`
- Mindre än `<`
- Större- eller lika-med `>=`
- Mindre- eller lika-med `<=`
- Lika med `==`
- Inte lika med `!=`

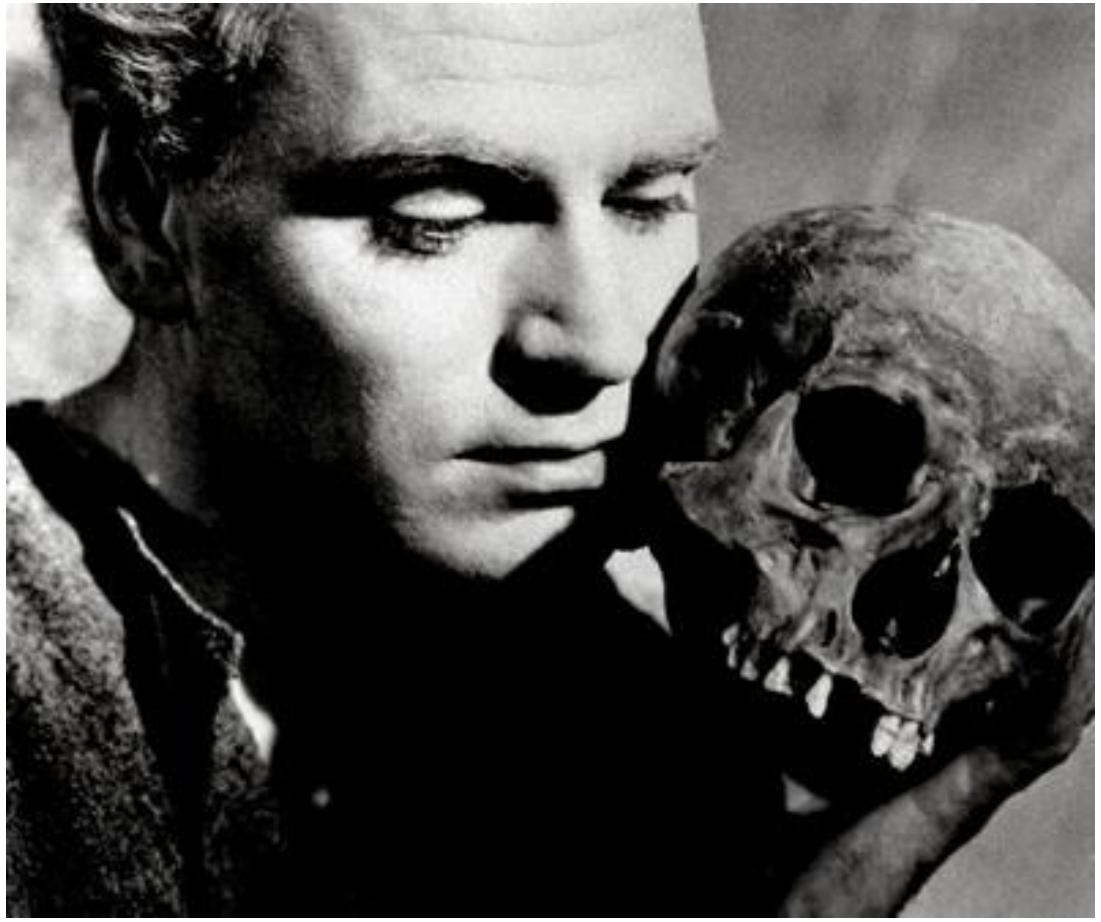
Logiska operatorer

- AND-operatorn **&&**
- OR-operator **||**

IF

V34

Att vara eller icke vara?



- Ett villkor i programmering är ett påstående som antingen är sant eller falskt.
- Ett eller flera villkor kan grupperas.
- Villkor benämns på engelska som condition.

`If` statement – ”Om” sats

- If-satsen används för att kapsla in kod som enbart ska köras om ett eller flera villkor är uppfyllda
- För att jämföra värden används lika-med komparatorn **==**
- För att kolla om en variabel inte är ett specifikt värde används **NOT** operatorn **!**

```
int a = 1;

if (a != 2)
{
    //Code to be executed
}
```

```
int a = 1;

if (a == 2)
{
    //Code to be executed
}
```

Och/Eller

- Och betecknas med **&&**
- Eller betecknas med ||
- För att gruppera villkor används **och** eller **eller** operatorer.

```
int a = 1;
bool b = true;

if (a == 1 && b)
{
    //Code to be executed
}
```

```
int a = 1;
int b = 2;

if (a == 2 || b == 2)
{
    //Code to be executed
}
```

```
int a = 1;
bool b = true;
char c = 'c';

if (a == 1 && b || c != 'c')
{
    //Code to be executed
}
```

`else`/`else if` – Annars/Annars om

- En if-sats kan byggas ut med else/else if
- Villkoren testas i sekvens
- Else kan användas för att exekvera kod om inget av de tidigare villkoren är sanna.

```
int a = 1;
bool b = true;
char c = 'c';

if (a == 2)
{
    Console.WriteLine("a är 2");
}
else if (b)
{
    Console.WriteLine("B är true");
}
else if (c != 'c')
{
    Console.WriteLine("c är inte 'c'");
}
else
{
    Console.WriteLine("inget av ovanstående är sant");
}
```

Switch

V34

`switch` – Flera möjligheter

- **`switch`**-satsen består av flera delar
- Första delen beskriver vad som ska evalueras
- En **`case`** är ett värde som skall kontrolleras
- **`break`** indikerar att **`switch`**-satsen skall avslutas
- **`default`** kan användas för att köra kod om inga utav **`case`**n är sanna.

```
char c = 'c';

switch (c)
{
    case 'a':
        Console.WriteLine("c är 'a'");
        break;
    case 'b':
        Console.WriteLine("c är 'b'");
        break;
    case 'c':
        Console.WriteLine("c är 'c'");
        break;
    default:
        Console.WriteLine("c är varken 'a', 'b' eller 'c'");
        break;
}
```

Loopar

V34

Loopar

- En loop kan användas för att repetera ett kodblock så länge ett angivet villkor uppfylls.
- Loopar finns i flera varianter – `for`/`while`/(`foreach`)
- Loopar…
  - … sparar tid
  - … ökar läsbarhet
  - … förenklar felsökning

While loop - `while`

- En while-loop repeterar kod så länge ett villkor uppfylls
- Villkoret skrivs på samma sätt som i en `if`

```
int a = 1;
while (a == 1)
{
    //Code to be executed until the condition is no longer true.
}
```

Do while loop – `do while`

- En do/while-loop kör kodblocket före villkoret kontrolleras
- Villkoret skrivs på samma sätt som i en `if`

```
int a = 1;
do {
    //Code to be executed until the condition is no longer true.
} while (a == 1);
```

For loop – **for**

- En for-loop kan användas för att upprepa kod ett bestämt antal gånger
- Första delen i en for-loop definierar en lokal variabel som kan användas inom loopens kodblock
- Den andra delen är villkoret som måste vara uppfyllt för att kodblocket skall köras
- Den tredje delen låter oss utföra en operation efter varje iteration

```
for (int i = 0; i < 10; i++)
{
    //Code to be executed until the condition is no longer true.
}
```

Break - **break**

- Break kan användas för att avbryta ett kodblock
- Break används oftast i kombination med en if-sats.
- Break avbryter kodblocket och programmet fortsätter nedanför blocket det brutit.

```
int a = 5;
for (int i = 0; i < 10; i++)
{
    //Code to be executed until the condition is no longer true.
    if (i == a)
    {
        break;
    }
}
```

Continue - `continue`

- Continue kan användas när man vill avbryta den aktuella iterationen och gå till nästa
- Continue används oftast i kombination med en if-sats.
- Continue avbryter kodblocket och programmet fortsätter med nästa iteration.

```
int a = 5;
for (int i = 0; i < 10; i++)
{
    //Code to be executed until the condition is no longer true.
    if (i == a)
    {
        continue;
    }
    //More code
}
```

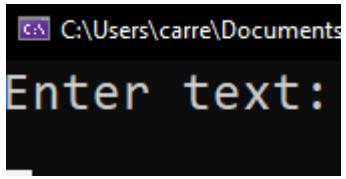
Console

V34

Console – En klass

- Console finns i biblioteket System
- Console är en klass som innehåller funktioner som låter oss styra konsollen
- Console.WriteLine() – later oss skriva ut rader med text

```
Console.WriteLine("Enter text:");
```



Write

- Console.Write() – skriver ut text utan att byta rad

```
Console.Write("apa");
Console.Write("apa");
```

```
apaapa
```

ReadKey

- Console.ReadKey() – inväntar att användaren trycker på en tangent
- Den reagerar dock ej på modifier keys.(shift,alt,ctrl)

```
Console.ReadKey();
```

ReadLine

- Console.ReadLine() – Läser och returnerar text som avslutas med ny rad (enter).

```
Console.WriteLine("What is your name?");
string name = Console.ReadLine();
Console.WriteLine("Hello " + name + "!");
```

```
What is your name?
Niklas
Hello Niklas!
```

Array

V34

Array – [ ]

- En Array är en samling värden av samma datatyp.
- En Array har ett bestämt antal element.
- Värdena inom Arrayen kan förändras men inte antalet element.

Array – [ ]

- När en Array deklareras bestäms vilken datatyp den kommer att.
- När en Array initieras får den en längd.
- Efter det kan varje position i Arrayen tilldelas ett värde.
- Alternativt kan man tilldela värden till hela Arrayen samtidigt som man definierar den.

```
string[] choices;
```

```
choices = new string[3];
```

```
string[] choices = new string[3];

choices[0] = "rock";
choices[1] = "paper";
choices[2] = "scissors";
```

```
string[] choices = new string[3] {"rock","paper","scissors"};
```

Arrayer och Loopar

- Ett vanligt sätt att använda loopar är för att leta i eller förändra Arrayer.
- For-loopen är speciellt bra för detta.

```
string[] choices = new string[3];

choices[0] = "rock";
choices[1] = "paper";
choices[2] = "scissors";

for (int i = 0; i < choices.Length; i++)
{
    Console.WriteLine(choices[i]);
}
```

Arrayer och Loopar

- En annan loop som med fördel kan användas för att leta i en array är **foreach**-loopen.
- I en foreach-loop körs ett kodblock för varje element i en Array.

```
string[] choices = new string[3];

choices[0] = "rock";
choices[1] = "paper";
choices[2] = "scissors";

foreach(string s in choices)
{
    Console.WriteLine(s);
}
```

Arrayfunktioner

- Array.Length – Är ett attribut som returnerar antalet element i en array

```
int numberOfLetters = letters.Length;
```

Arrayfunktioner

- Array.Sort() – Ger oss möjlighet att sortera en Array.
- Array.Sort() använder **Insertion sort** för arrayer mindre än 17 element och **Quicksort** för större.

```
string[] letters = new string[3];

letters[0] = "C";
letters[1] = "B";
letters[2] = "A";

Array.Sort(letters);

foreach(string l in letters)
{
    Console.WriteLine(l);
}
```

```
A
B
C
```

Arrayfunktioner

- Array.IndexOf() – En funktion för att leta efter ett specifikt element i en array och returnera positionen för det första elementet med det önskade värdet.
- Array.LastIndexOf() – Samma som IndexOf men returnerar positionen för det sista elementet med det önskade värdet.

```
int indexOfA = Array.IndexOf(letters, "A");
Console.WriteLine(indexOfA);
```

Enum

V34

Enum - **enum**

- En enum kan ses som en lista med tal som representeras av ord.

```
enum Mood
{
    HAPPY,
    NEUTRAL,
    SAD
}
```

```
Mood mood = Mood.HAPPY;

if(mood is Mood.HAPPY)
{
    Console.WriteLine($"I'm so {mood}");
}
```