



**Campus Nyköping**

# PROGRAMMERING I .NET C# 1 - 04

C#: Switchar och metoder

# Förra gången

- C#
  - Datatyper: int string, bool, double...
  - Operatorer: +, -, \*, /, %
  - Loopar: for, while
- Övningar
  - Labyrinten/Rutor

# Idag

- Lösningsförslag Labyrinten
  - Eventuella andra som vill visa sina lösningar?
- Switchar
- Metoder

# Övningsuppgiften från förra gången

- Labyrint 1
  - Uppgiften går ut på att rita upp ett rutnät i en konsol, med hjälp av loopar.
  - Rutnätet består av ett antal rutor med tre teckens bredd och höjd.
  - Det ni ska göra är att, m h a bokstäver, rita ut alla kanter.
  - Börja med ett rutnät av 3 x 3 rutor.
  - Därefter ska ni, med tangenttryckningar, kunna ändra storlek på rutnätet.
  - Tangent x gör rutnätet större, och tangent z gör rutnätet mindre.
- Extrauppgiften
  - I rutnätet ska du placera en ytterligare bokstav, i mitten av en ruta. Den ska symbolisera en gubbe, som ska kunna förflytta sig mellan rutorna.
  - Använd tangenterna A S D W för att flytta gubben.

# Lösningen – visa runt i koden

- Kod: Rutor
- Ett antal nästlade if-satser.
- Problem
  - Ett antal If-satser kan bli rörigt och upprepande
    - Bättre att använda switch
  - Upprepad kod
    - Bättre att skapa en metod

# switch – istället för if

- Ett enkelt och översiktligt sätt att skapa ett antal villkor av liknande sort är att använda en switch

# Switch - Strukturen

```
switch(uttryck) // Tex en variabel, som räknas ut I början, en gång
{
    case x:
        // kodblock som körs om uttrycket = x
        break;
    case y:
        // kodblock som körs om uttrycket = y
        break;
    default:
        // kodblock som körs om uttrycket varken är lika med x eller y
        break;
}
```

# Switch - exempel

- Ett enkelt och översiktligt sätt att skapa ett antal villkor av liknande sort är att använda en switch:

```
int caseSwitch = 1;

switch (caseSwitch)
{
    case 1:
        Console.WriteLine("Case 1");
        break; // Avbryter switch-satsen

    case 2:
        Console.WriteLine("Case 2");
        break; // Avbryter switch-satsen

    default:
        Console.WriteLine("Default case");
        break; // Avbryter switch-satsen
}
```



# Switch - break

- Endast ett switch i en switch-sats verkställs.
- C# tillåter inte att körningen fortsätter från en switch till nästa. På grund av detta genereras följande kod ett kompilatorfel, CS0163: "Control cannot fall through from one case label (<case label>) to another." om man inte har med ett break i varje switch.

# Switch

- Det går också att ha flera alternativ på värden som ger samma resultat:

```
switch (caseSwitch)
{
    case 1:
        Console.WriteLine("Case 1");
        break;

    case 2:
    case 3:
        Console.WriteLine("Case 2 or 3");
        break;

    case 4:
        Console.WriteLine("Case 4");
        break;

    default:
        Console.WriteLine("Default value...");
        break;
}
```

# Switch - eksempel

- `int day = 6;`
- `switch (day)`
- `{`
- `case 1:`
- `Console.WriteLine("Måndag");`
- `break;`
- `case 2:`
- `Console.WriteLine("Tisdag");`
- `break;`
- `case 3:`
- `Console.WriteLine("Onsdag");`
- `break;`
- `case 4:`
- `Console.WriteLine("Torsdag");`
- `break;`
- `case 5:`
- `Console.WriteLine("Fredag");`
- `break;`
- `case 6:`
- `case 7:`
- `Console.WriteLine("Helg");`
- `break;`
- `}`
- `// Kodens skriver ut "Helg"`

# Lärare – Chef – Elev -exemplet

```
Console.Write("Ange namn: ");
string name = Console.ReadLine();
if (name == "micke" || name == "håkan")
{
    Console.Write("Lärare");
}
else if (name == "christer")
{
    Console.Write("Chefen");
}
else
{
    Console.Write("Elev");
}
```

```
Console.Write("Ange namn: ");
string name = Console.ReadLine();
switch(name)
{
    case "micke":
    case "håkan":
        Console.WriteLine("Lärare");
        break;
    case "christer":
        Console.WriteLine("Chefen");
        break;
    default:
        Console.WriteLine("Elev");
        break;
}
```

# Switch – Några råd

- Det måste finnas ett separat case för varje normal situation
- Lägg det normala fallet först
  - Lägg de mest använda fallen först, de sällan använda sist
- Sortera fallen alfabetiskt, numeriskt eller vad som är lämpligt
  - Tänk på att koden skall vara lätt att läsa för människor!
- Använd fallet default för att hantera sådana fall som inte skall kunna inträffa under normala omständigheter

# Code- Along

- Kod: Switch
- Extra: Labyrint – ändra till switch för knapparna

# Metoder

- En metod är som ett byggblock som löser ett litet problem
- Ett stycke kod som har ett namn och kan anropas från en annan plats i programmet
- Kan ta emot parametrar och returnera ett värde
- Metoder låter oss bygga stora program av små enkla bitar
- Metoder kallas också funktioner, procedurer och subrutiner i andra språk

# Metoder – Varför?

- Mer hanterlig programmering
  - Dela upp ett stort problem i mindre bitar
  - Bättre organisation av programmet
  - Förbättra läsbarhet
  - Underlätta förståelse
- Undvika upprepningar
- Återanvändbarhet
  - Använd en metod flera gånger
- "Gömma undan" kod som man inte behöver känna till - Abstraktion

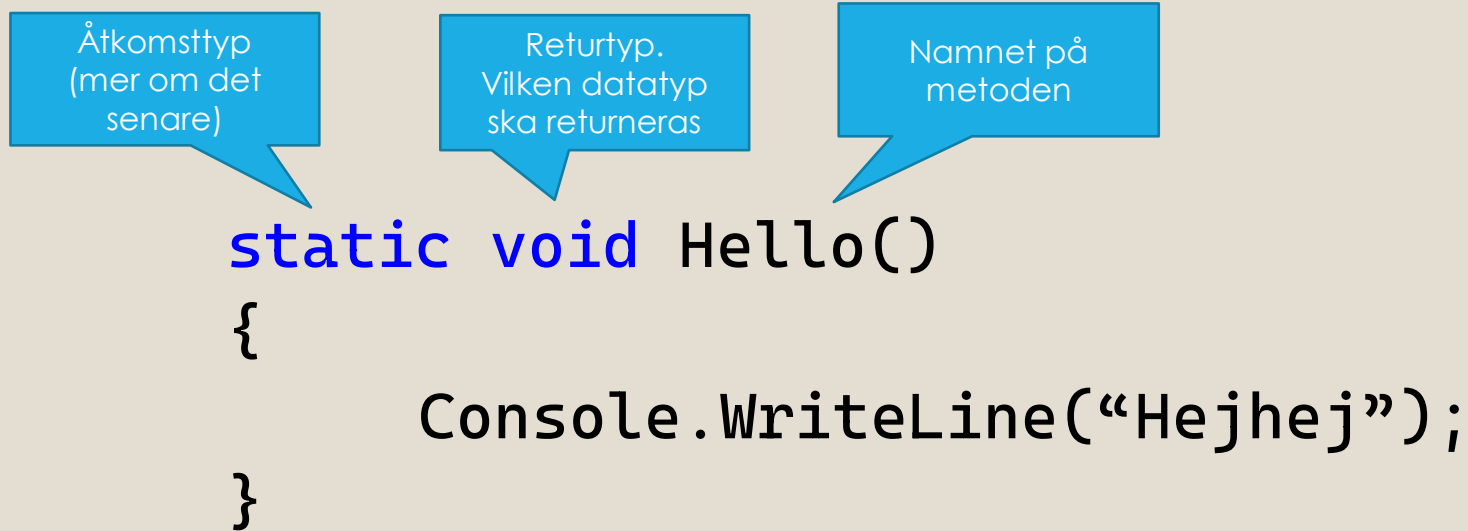


# Enklaste metoden

```
static void Hello()  
{  
    Console.WriteLine("Hejhej");  
}
```

# Deklarera metoder

- En metod är ett stycke kod med ett namn
- Varje metod har:



Returtypen void betyder att inget ska returneras

# Anropa metoden

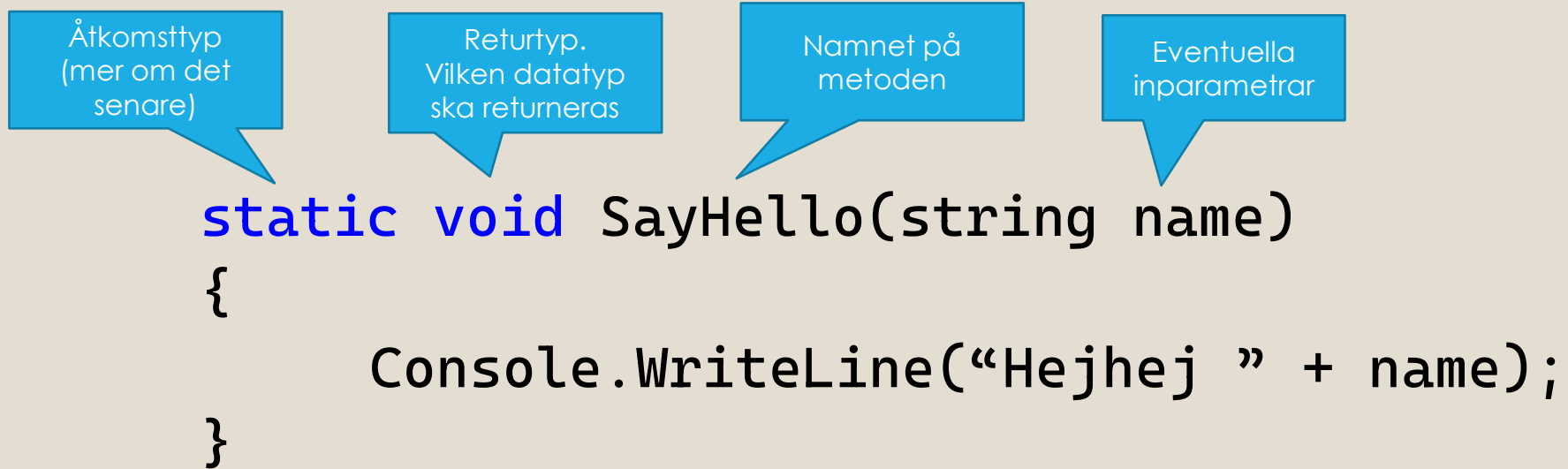
```
static void Main(string[] args)
{
    Hello();
}
```

# Metoder med inparametrar

- För att skicka information till en metod används parametrar (kallas även argument)
  - Skicka noll eller flera värden
  - Skicka värden av olika typ
  - Varje parameter har namn och typ
  - Parametrar får värden när metoden anropas
  - Parametrar kan ändra hur metoden fungerar beroende på deras värden

# Deklarera metoder med parametrar

- En metod är ett stycke kod med ett namn
- Varje metod har:



Returtypen void betyder att inget ska returneras

# Anropa metoden

```
static void Main(string[] args)
{
    SayHello("Micke");
}
```

# Exempel

```
static void Main(string[] args)
{
    SayHello("Micke");
}
```

```
static void SayHello(string name)
{
    Console.WriteLine("Hejhej " + name);
}
```

# Definiera och använd metodparametrar

- Metodens beteende beror på parametrarna
- Parametrar kan vara av godtycklig typ
  - int, double, string etc.
  - Arrayer (int[], double[], etc.)

```
static void PrintSign(int number)
{
    if (number > 0)
        Console.WriteLine("Positive");
    else if (number < 0)
        Console.WriteLine("Negative");
    else
        Console.WriteLine("Zero");
}
```



# Flera inparametrar

```
static void PrintPerson(int weight, string name)
{
    Console.WriteLine($"{name} väger {weight} kg");
}
```

# Anropa metoder med parametrar

- För att anropa en metod och skicka värden till dess parametrar:
  - Använd metodens namn, följt av en lista av uttryck för parametrarna
- Uttryck måste vara av samma datatyp som metodens parametrar (eller kompatibla)
- Om metoden vill ha en float, kan du skicka en int istället
  - "Kan" betyder inte att det är bra idé!
  - Använd samma ordningsföljd som i metodens deklaration
  - För metoder utan parametrar: glöm inte parenteserna!
    - `SayHello()`
- `PrintPerson(82, "Micke");`
- `PrintPerson (23 + 45, name);`
- `PrintPerson(GetWeight(), GetName());`

# Skicka tillbaka värden från metoder

- Istället för void anger vi vilken typ av data som skall returneras

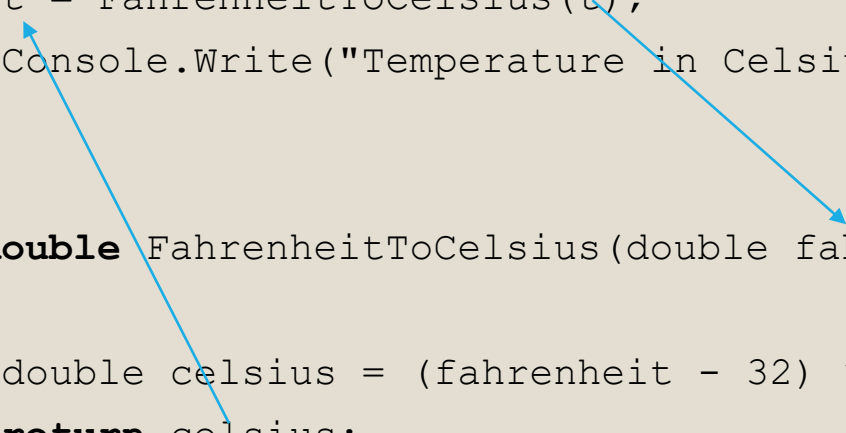
```
static int Multiply(int firstNum, int secondNum)
{
    return firstNum * secondNum;
}
```

- Metoder kan returnera alla datatyper (int, string, array etc.)
- void-metoder returnerar ingenting
- Kombinationen av metodens namn och dess parametrar kallas signatur
- Använd nyckelordet **return** för att skicka tillbaka resultatet

# Exempel

```
static void Main()
{
    Console.WriteLine("Temperature in Fahrenheit: ");
    double t = Double.Parse(Console.ReadLine());
    t = FahrenheitToCelsius(t);
    Console.WriteLine("Temperature in Celsius: {0}", t);
}

static double FahrenheitToCelsius(double fahrenheit)
{
    double celsius = (fahrenheit - 32) * 5 / 9;
    return celsius;
}
```



The diagram consists of two blue arrows. The first arrow originates from the variable `t` in the expression `t = FahrenheitToCelsius(t);` within the `Main` method and points to the parameter `fahrenheit` in the signature of the `FahrenheitToCelsius` method. The second arrow originates from the `return` statement `return celsius;` within the `FahrenheitToCelsius` method and points back to the variable `t` in the expression `t = FahrenheitToCelsius(t);` within the `Main` method, illustrating the flow of data from the function back to the caller.

# Extragrej - tryParse

```
bool success = Int32.TryParse(value, out number);  
if (success)  
{  
    Console.WriteLine($"Converted '{value}' to {number}.");  
}  
else  
{  
    Console.WriteLine($"Attempted conversion of '{value}'  
failed.");  
}
```

# Kolla om rätt siffervärden skrivs in

- Bara siffror mellan 0-5 får skrivas in:

// Om siffran **inte** går att parsea till en int eller siffran är mindre än 0 eller större än 5, så avbryt programkörningen.

```
int digit = 0;
if (!int.TryParse(Console.ReadLine(), out digit) || digit < 0 || digit > 5)
{
    Console.WriteLine("Du har matat in fel värden");
    return; // Avbryter programmet
}
```

# Code Along - Metoder

- Kod: Metoder
- Extra: Menu

# Övning 1 - Switch

- Skriv ett program som lägger till bonus till givna poäng.
  - Poängen är mellan 1 och 9.
  - Programmet läser en siffra som indata.
- Om siffran är
  - mellan 1 och 3: multiplicera med 10
  - mellan 4 och 6: multiplicera med 100
  - mellan 7 och 9: multiplicera med 1000
  - noll eller inte en siffra: skriv ut felmeddelande
- Använd en switch och avsluta med att skriva ut det nya värdet på konsolen
- Lägg in tryParse så det bara går att ange siffrorna 1-9



# Övning 2 - Switch

- Skapa en applikation som “simulerar” en hiss
  - En loop räknar upp våningarna från 0 till 4
  - Beroende på våning så ska det skrivas ut följande:
    - Våning 0: Entrèplan
    - Våning 1: Säljavdelningen
    - Våning 2: IT-avdelningen
    - Våning 3: Projekt-ledningen
    - Våning 4: Chefen
- Beroende på vilken våning så ska en switch välja vilken text som skrivs ut.
- (Använd gärna `Thread.Sleep(1000)` för att simulera att hissen tar tid)

# Övning 3 - Metoder

- Skriv ett program med en Main enligt:

```
public static void Main()  
{  
    SayHello();  
    SayGoodbye();  
}
```

- Skriv metoderna SayHello() och SayGoodbye()
- Lägg till att du kan skriva in ditt namn, och få en personlig hälsning.

# Övning 4 - Metoder

- Skapa en Console Application med tre metoder:
  - Addera() som tar två heltal och returnerar summan.
  - Momsen() som tar ett tal som inparameter och räknar ut momsen.
    - Anta att moms är 25%.
  - MomsOchBelopp() som tar ett tal och momsprocent och returnerar resultatet.
    - Skapa ett gränssnitt som använder de här metoderna. Använd tryParse.



# Övning 5 – Switch och metoder

- Bankomaten
  - Skapa en Console Application som:
    - med hjälp av en switch-sats skapar en inmatningsmeny för en bankomat. Alternativen ska vara:
      - [I]nsättning
      - [U]ttag
      - [S]aldo
      - [A]vsluta
  - Skapa fyra metoder som hanterar dessa funktioner
  - Det ska inte gå att ta ut mer pengar än vad det finns på kontot
  - Det ska inte gå att sätta in mer än 10.000 kr per gång (lagen om penningtvätt)
  - Saldot ska hela tiden vara uppdaterat.

# Länkar

- [https://www.w3schools.com/cs/cs\\_switch.asp](https://www.w3schools.com/cs/cs_switch.asp)
- [https://www.w3schools.com/cs/cs\\_methods.asp](https://www.w3schools.com/cs/cs_methods.asp)
- [C# Method Parameters \(w3schools.com\)](#)