Programmering i C++
EDAF30
Något om C#

De sista pusselbitarna

Innehåll

- Hello World
- Lite historik
- Tillgänglighet
- Jämförelse med C++
- Ny for-variant (foreach)
- Egenskaper (properties)
- Indexerare (indexer)
- Delegater (delegates)
- Mallar (generics)

Hello World

Klassiskt testprogram

Exempel

```
using System;
class HelloWorld {
    static void Main(string []args)
    {
        if (args.Length == 0)
            Console.WriteLine("Hello World");
        else {
            string name = args[0];
            Console.WriteLine("Hello, {0}", name);
        }
        Console.ReadKey();  // Vänta på inmatning
    }
}
```

Lite historik

1996 Anders Hejlsberg –> Microsoft

2000 Standardisering av C# och CLI påbörjad

2001 Ecma-standard (v. 1.2)

2003 ISO-standard (v. 1.2)

2005 Version 2.0 (Ecma & ISO)

2007 Version 3.0

2010 Version 4.0

2012 Version 5.0

Tillgänglighet

Visual C# Referensimplementation. Endast Windows
Visual Studio Express 2013

Mono Linux, Windows, Mac OS
`http://mono-project.com/`

DotGNU Linux, Mac OS m fl. Projektet lagt på is 2012
`http://www.gnu.org/software/dotgnu/`

Jämförelse med C++

- Inga globala variabler eller funktioner
- Pekare kan bara användas i kodblock märkta med nyckelordet `unsafe`
- Säkrare (starkare) typning
- Garbage collector används
- Kompilering till CIL (Common Intermediate Language)
  Exekvering på CLR (Common Language Runtime)
  (ett slags virtuell maskin)

Ny for-variant: foreach

Exempel på ny struktur (finns liknande i Java och C++11)

```
using System;

public class ForeachTest {
    public static void Main() {
        int sum = 0;
        int[] nums = new int[10];

        for(int i = 0; i < 10; i++)
            nums[i] = i;

        foreach(int x in nums) {
            Console.WriteLine("Value is: " + x);
            sum += x;
        }
        Console.WriteLine("Summation: " + sum);
        Console.ReadKey();  // Vänta på inmatning
    }
}
```

Egenskaper (properties)

Systematisering av set och get

```
public class Customer {
    private int m_id = -1;
    public int ID {
        get { return m_id; }
        set { m_id = value;}
    }
    private string m_name = string.Empty;
    public string Name {
        get { return m_name; }
        set { m_name = value;}
    }
}
```

Egenskaper (properties)

Användning av properties

```
using System;

public class CustomerTest {
    public static void Main() {
        Customer cust = new Customer();
        cust.ID = 1;
        cust.Name = "Nils Nilsson";
        Console.WriteLine(
                        "ID: {0}, Name: {1}",
                        cust.ID, cust.Name);
        Console.ReadKey();
    }
}
```

Indexerare (indexer)

Att "vektorisera" en struktur

```
class IntIndexer
{
    private string[] myData;
    public IntIndexer(int size)
    {
        myData = new string[size];
        for (int i = 0; i < size; i++)
        {
            myData[i] = "empty";
        }
    }
    public string this[int pos]
    {
        get { return myData[pos]; }
        set { myData[pos] = value; }
    }
    // Main-funktionen
}
```

Indexerare (indexer)

Huvudprogram för indexerare

```
using System;

class IntIndexer
{
    public static void Main(string[] args)
    {
        int size = 10;
        IntIndexer ind = new IntIndexer(size);
        ind[9] = "En sträng";
        ind[3] = "En till";
        ind[5] = "Någonting";
        Console.WriteLine("\nIndexer Output\n");
        for (int i = 0; i < size; i++)
        {
            Console.WriteLine("ind[{0}]: {1}", i, ind[i]);
        }
    }
}
```

Delegater (delegates)

Motsvarar funktionspekare i C/C++

```
using System;

// Själva delegatdeklarationen
public delegate int Comparer(object obj1, object obj2);

// "Testklass"
public class Name {
    public string FirstName = null;
    public string LastName = null;
    public Name(string first, string last) {
        FirstName = first;
        LastName = last;
    }

// Forts. följer ...
```

Delegater (forts.)

```
    // En möjlig delegat
    public static int CompareFirstNames(object name1, object name2)
    {
        string n1 = ((Name)name1).FirstName;
        string n2 = ((Name)name2).FirstName;
        if (String.Compare(n1, n2) > 0)
            return 1;
        else if (String.Compare(n1, n2) < 0)
            return -1;
        else
            return 0;
    }

    public override string ToString()
    {
        return FirstName + " " + LastName;
    }
} // Slut på testklassen Name
```

Delegater (forts.)

```
// Huvudprogrammet
class SimpleDelegate {
    Name[] names = new Name[4];
    public SimpleDelegate() {
        names[0] = new Name("Nils", "Nilsson");
        names[1] = new Name("Otto", "Olsson");
        names[2] = new Name("Nils", "Jonsson");
        names[3] = new Name("Lena", "Ohlsson");
    }
    static void Main(string[] args) {
        SimpleDelegate sd = new SimpleDelegate();
        // Här skapas delegat-objektet (här en "jämförare")
        Comparer cmp = new Comparer(Name.CompareFirstNames);
        Console.WriteLine("\nBefore Sort: \n");
        sd.PrintNames();
        // Lägg märke till delegat-argumentet!
        sd.Sort(cmp);
        Console.WriteLine("\nAfter Sort: \n");
        sd.PrintNames();
    }  // Forts. följer ...
```

Delegater (forts.)

Metoden Sort som utnyttjar delegaten (en komparator)

```
public void Sort(Comparer compare) {
    object temp;
    for (int i=0; i < names.Length; i++) {
        for (int j=i; j < names.Length; j++) {
            // Använd delegaten "compare" som om det vore en metod
            if ( compare(names[i], names[j]) > 0 ) {
                // Byt plats
                temp = names[i];
                names[i] = names[j];
                names[j] = (Name)temp;
            }
        }
    }
}

// Forts. följer ...
```

Delegater (forts.)

Till sist: PrintNames (observera användningen av `foreach`!)

```
    public void PrintNames()
    {
        Console.WriteLine("Names: \n");
        foreach (Name name in names)
        {
            Console.WriteLine(name.ToString());
        }
    }

}   // SimpleDelegate
```

Mallar

Fungerar som i Java:

```
using System;

class Test<T> {
    public static void swap(ref T a, ref T b) {
        T tmp = a;
        a = b;
        b = tmp;
    }
}

class Testa {
    public static void Main() {
        string s1 = "hej", s2 = "tjo";
        Console.WriteLine("Före:  {0} - {1}", s1, s2);
        Test<string>.swap(ref s1, ref s2);
        Console.WriteLine("Efter: {0} - {1}", s1, s2);
    }
}
```