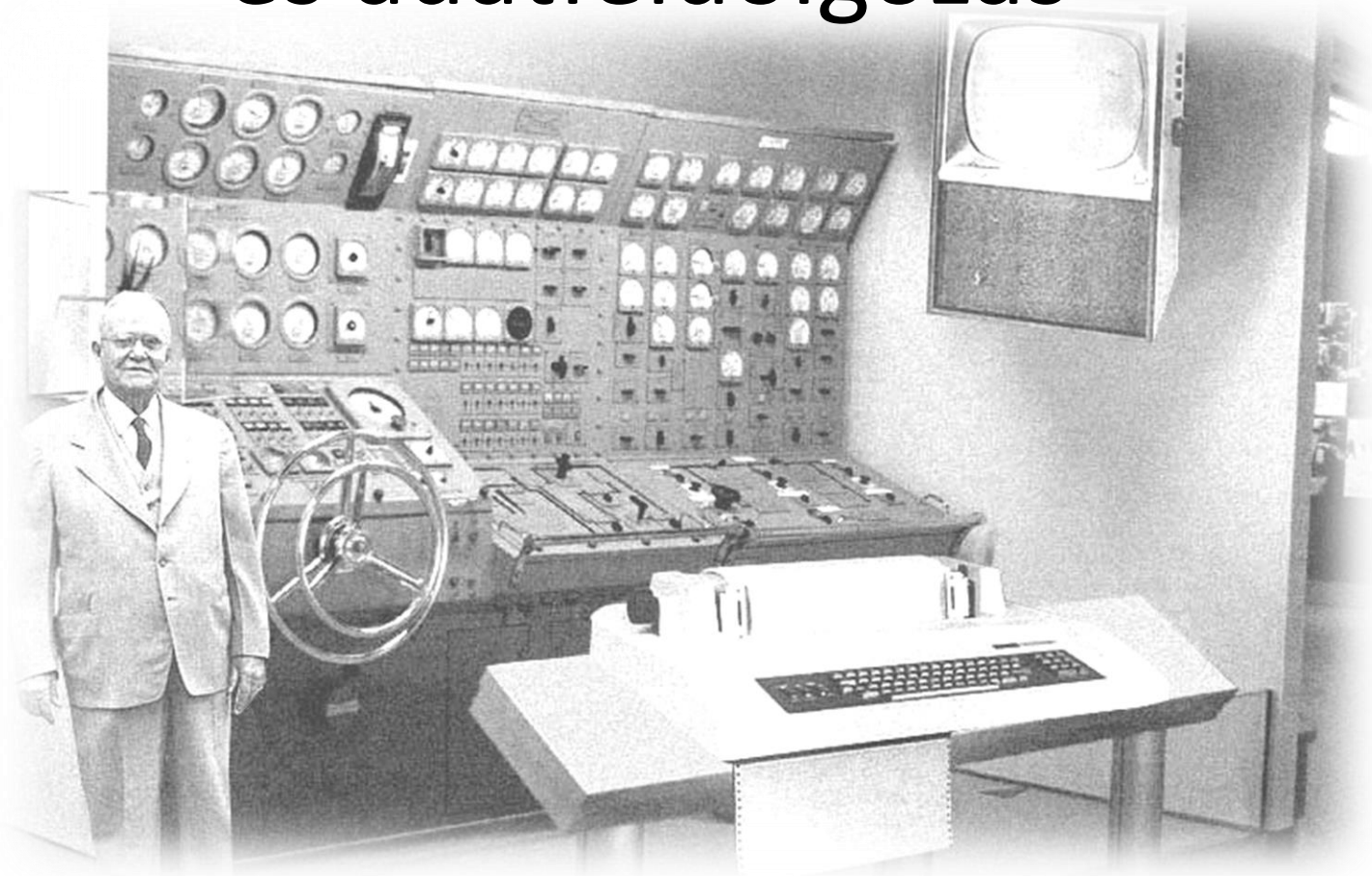


Mérési adatgyűjtés és adatfeldolgozás



BME TTK Fizika Tanszék
2013/2014 tavaszi félév

Tartalom

- Alapok
 - Objektumorientált programozás koncepciója
 - Mérésvezérlés számítógéppel
 - Adatok feldolgozása, ábrázolása
- Programozási feladatok Visual Studio 2005 környezetben
 - Grafikus felület fejlesztése
- Soros porti műszer programozása
 - Kommunikáció a műszerrel
 - Adatok grafikus megjelenítése
- Összetett mérésvezérlési és -kiértékelési feladat önálló megoldása
 - Digitális multiméter használata
 - Mérőkártya programozása USB porton

Információk

- Honlap
 - http://fizipedia.bme.hu/index.php?title=Mérési_adatgyűjtés_és_feldolgozás
- Visual Studio 2005
 - Beszerzés egyetemi hálózatról:
 - ftp://szoftver.eik.bme.hu/MicrosoftCampus/Regi_verziok/VisualStudio/VisualStudio2005/
(javasolt: *files.zip, ne *image.zip)
 - Segítség az egyetemi hálózat eléréséhez kívülről:
<http://www.hszk.bme.hu/mittegyek.html#bmevpnproxy>
 - John Sharp: Microsoft Visual C# 2008
 - Tutorial: <http://www.tutorialspoint.com/csharp/index.htm>

Tárgykövetelmények

- **Jelenléti követelmények:** A félévközi jegy megszerzésének szükséges feltétele az összes laborgyakorlat teljesítése.
- **Félévközi számonkérések:** A laboratóriumi gyakorlatok elvégzése előzetes önálló felkészülést igényel. A gyakorlatok elején a hallgatók felkészültségét rövid írásbeli számonkéréssel ellenőrizzük.
- **Osztályzat:** ZH 20 pont + jegyzőkönyv 100 pont
 - 40% alatt elégtelen (1)
 - 40% és 55% között elégséges (2)
 - 55% és 70% között közepes (3)
 - 70% és 85% között jó (4)
 - 85% felett jeles (5)
- Amennyiben egy leadott jegyzőkönyv, program vagy írásbeli dolgozat vagy ezeknek egy része bizonyíthatóan nem önálló munka eredménye, akkor azt automatikusan - a másolás mértékétől függetlenül - az adott munkára adható maximális pontszám mínusz egyszeresével értékeljük!

Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
valósidejű kiértékelés

Eszközök:

Mérőműszer: valamilyen fizika mennyiség mérésére;

Számítógép: adatok gyűjtése, megjelenítése, feldolgozása



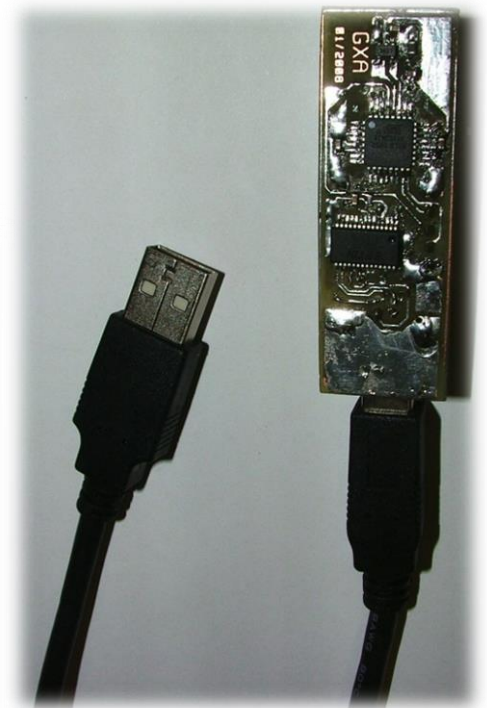
Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
valósidejű kiértékelés

Eszközök:

Mérőműszer: valamilyen fizika mennyiség mérésére;

Számítógép: adatok gyűjtése, megjelenítése, feldolgozása



Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
valósidejű kiértékelés

Eszközök:

Mérőműszer: valamilyen fizika mennyiség mérésére;

Számítógép: adatok gyűjtése, megjelenítése, feldolgozása

Kommunikáció: szabványos
csatolófelületek:

- RS-232



- USB



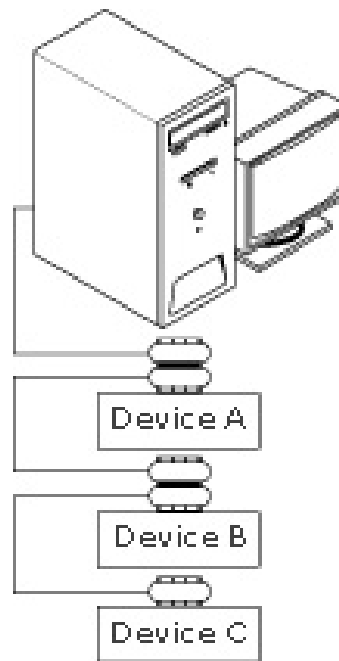
- LPT



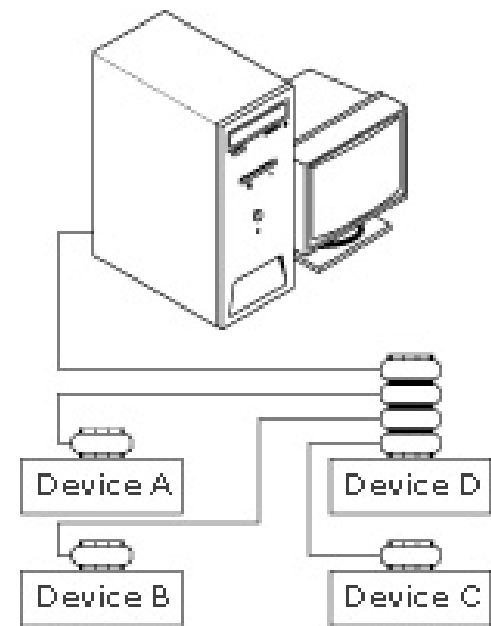
- GPIB



...



a. Linear Configuration



b. Star Configuration

Objektumorientált programozás - alapok

Mi is az az objektum?

A program olyan egysége, ami kommunikál a többi objektummal:
üzeneteket kap, adatokat dolgoz fel, üzeneteket küld.

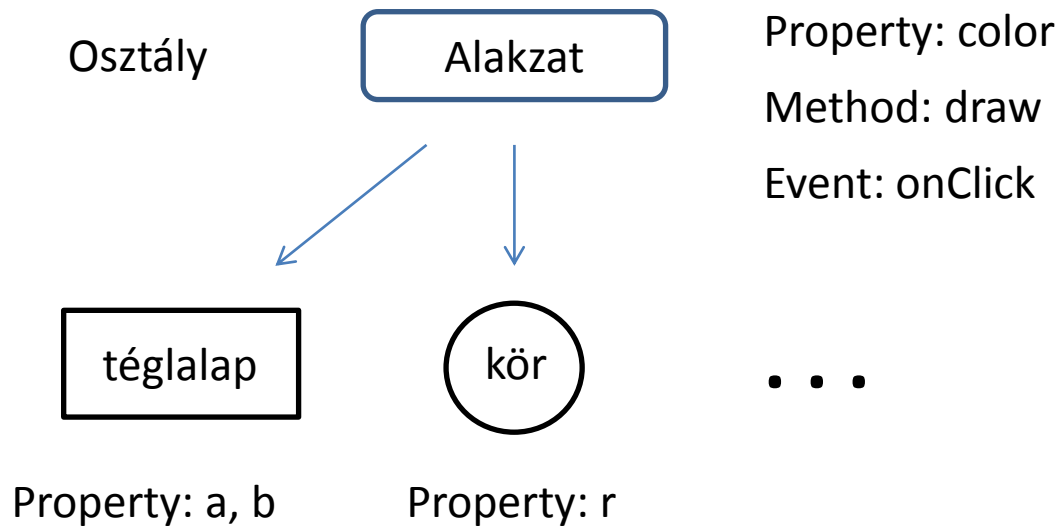
Forrás: Wikipédia

Mérésvezérlés:

- gyors fejlesztés
- modularitás
- eseményvezérelt működés

Objektumorientált programozás - alapok

Példa: rajzolóprogram



- gyors fejlesztés
- modularitás
- eseményvezérelt működés

C#

- Objektorientált, eseményvezérelt, általános, ... programnyelv

- .NET Framework



- CLR – Common Language Runtime
- Class Library

- MONO (LINUX)



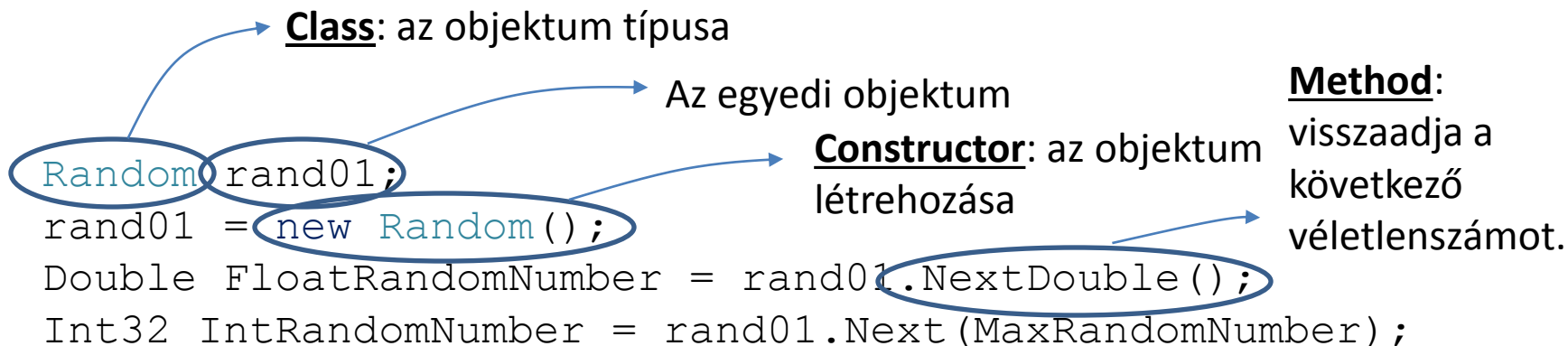
Objektumorientált programozás - alapok

Mi is az az objektum?

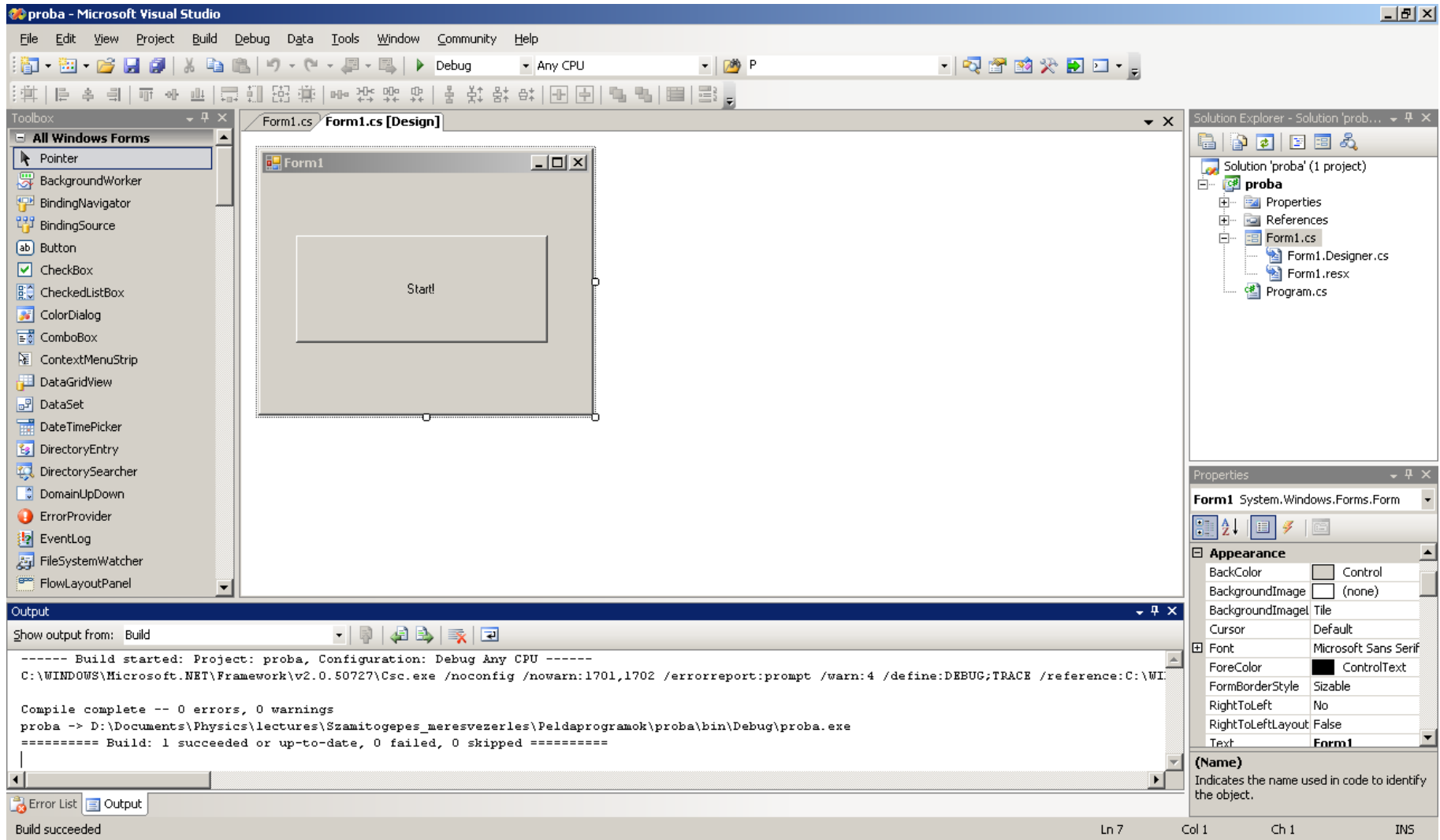
A program olyan egysége, ami kommunikál a többi objektummal:
üzeneteket kap, adatokat dolgoz fel, üzeneteket küld.

Forrás: Wikipédia

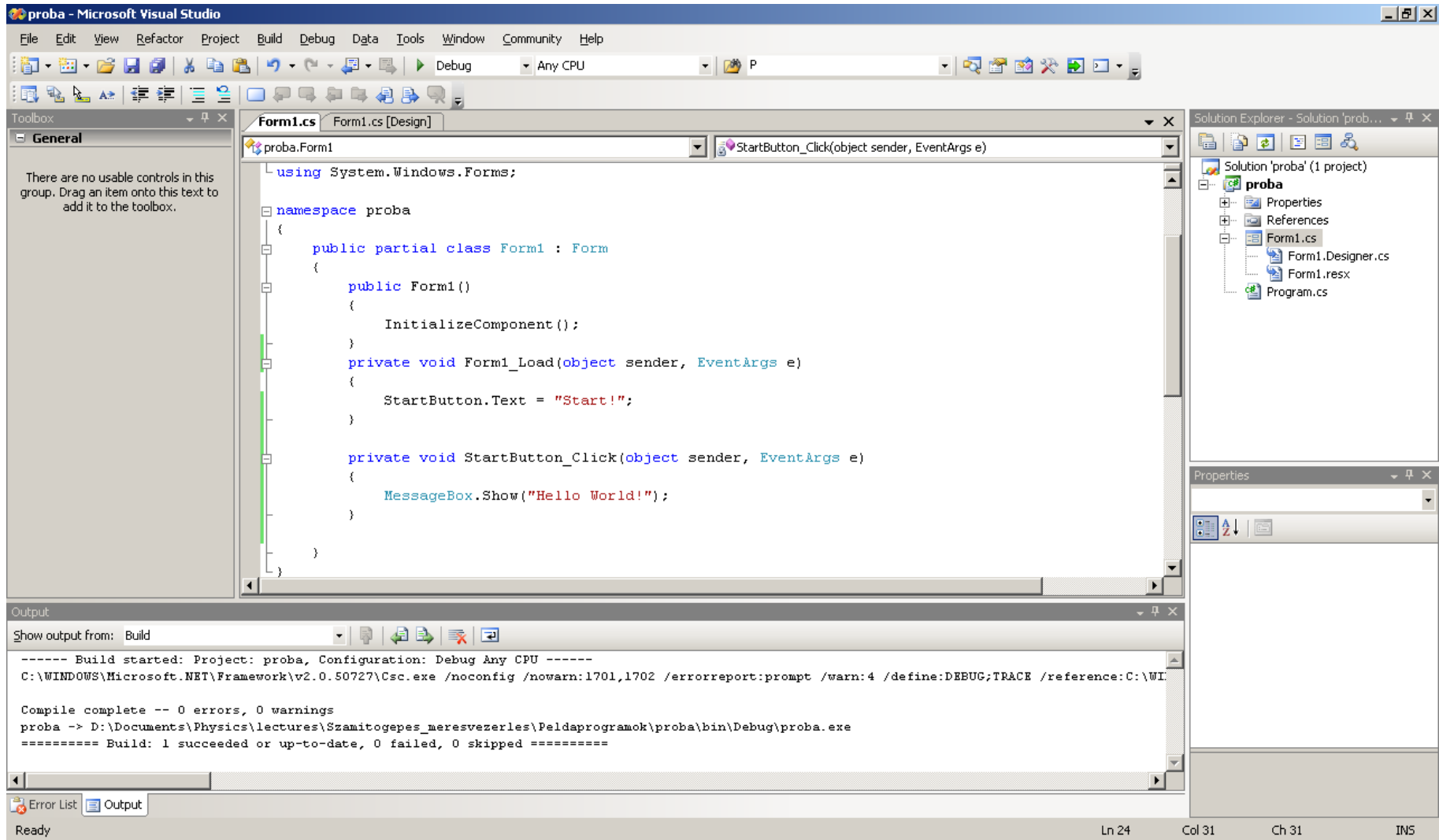
Példa: véletlenszám generálása:



Objektumorientált programozás – Visual Studio 2005



Objektumorientált programozás – Visual Studio 2005



Objektumorientált programozás – Visual Studio 2005

„Hello World!” program:

```
namespace proba
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            StartButton.Text = "Start!";
        }

        private void StartButton_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello World!");
        }
    }
}
```

Objektumorientált programozás – Visual Studio 2005

„Hello World!” program:

```
namespace proba
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            StartButton.Text = "Start!";
        }
        private void StartButton_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello World!");
        }
    }
}
```

namespace

function

event

property

method

Objektumorientált programozás - alapok

Mi is az az objektum?

A program olyan egysége, ami kommunikál a többi objektummal:
üzeneteket kap, adatokat dolgoz fel, üzeneteket küld.

Forrás: Wikipédia

Mérésvezérlés:

- gyors fejlesztés
- modularitás
- eseményvezérelt működés

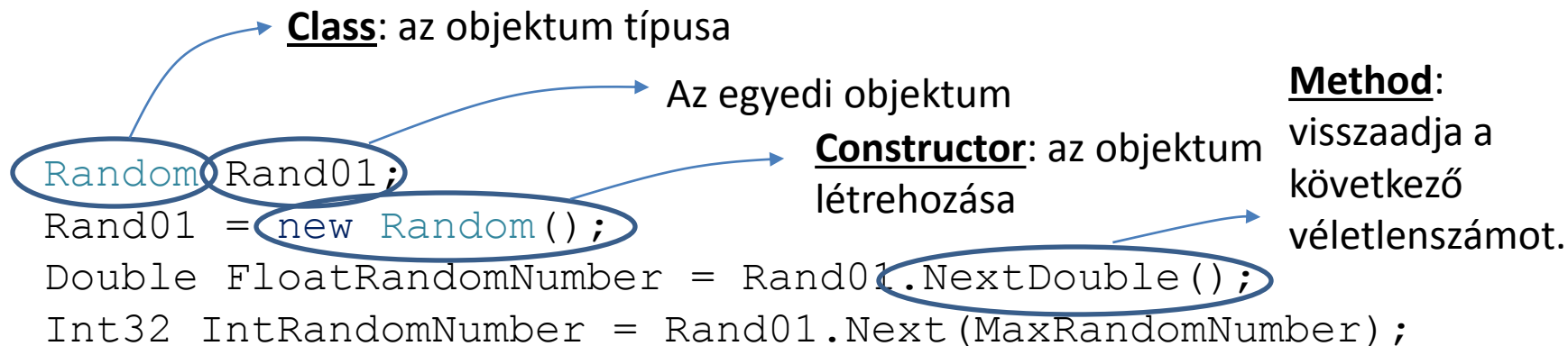
Objektumorientált programozás - alapok

Mi is az az objektum?

A program olyan egysége, ami kommunikál a többi objektummal:
üzeneteket kap, adatokat dolgoz fel, üzeneteket küld.

Forrás: Wikipédia

Példa: véletlenszám generálása:

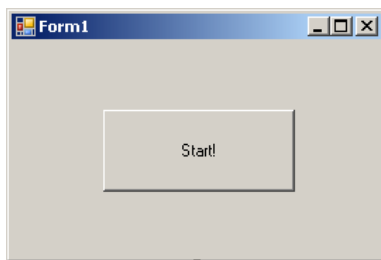


Objektumorientált programozás - alapok

Mi is az az objektum?

A program olyan egysége, ami kommunikál a többi objektummal:
üzeneteket kap, adatokat dolgoz fel, üzeneteket küld.

Példa: nyomógomb:



Forrás: Wikipédia

Property: a StartButton
objektum egyik tulajdonsága

```
StartButton.Text="Start!"  
private void StartButton_Click(object sender, EventArgs e)  
{  
  
}
```

Event: a StartButtonhoz kötődő
esemény bekövetkezésekor fut le

Objektumorientált programozás - alapok

Mi is az az objektum?

A program olyan egysége, ami kommunikál a többi objektummal: üzeneteket kap, adatokat dolgoz fel, üzeneteket küld.

Forrás: Wikipédia

Összefoglalás:

- **Class**: az objektum típusa;
- **Object**: az egyedi objektum;
- **Method**: az objektumra jellemző képesség;
- **Property**: az objektum egyik tulajdonsága;
- **Event**: az objektumhoz kötődő esemény.

Objektumok

Button:

```
using System.Windows.Forms;
```



Properties

Name	Az objektum azonosítója
Text	A gomb felirata

Methods

Hide	Elrejt a gombot
Show	Megmutatja a gombot

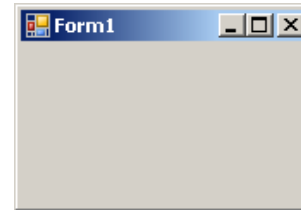
Events

Click	Kattintáshoz tartozó esemény
-------	------------------------------

Objektumok

Form:

```
using System.Windows.Forms;
```



Properties

Name	Az objektum azonosítója
Text	A Form fejléce

Methods

Show	Megnyitja a Formot
Close	Bezárja a Formot

Events

Load	A Form megjelenésekor fut le
Close	A Form bezárásakor fut le

Objektumok

TextBox:



```
using System.Windows.Forms;
```

Properties

Name	Az objektum azonosítója
Text	A szövegdoboz tartalma

Methods

Hide	Elrejt a szövegdobozt
Show	Megmutatja a szövegdobozt

Events

TextChanged	A szövegdoboz tartalmának változásakor fut le
Click	A szövegdobozra kattintáskor fut le

Objektumok

Label:



```
using System.Windows.Forms;
```

Properties

Name	Az objektum azonosítója
Text	A címke tartalma

Methods

Hide	Elrejt a címkét
Show	Megmutatja a címkét

Events

VisibleChanged	A címke elrejtésekor/megjelenésekor fut le
Click	A címkére kattintáskor fut le

Objektumok

StreamWriter, StreamReader:

```
using System.IO;
```

Constructor

```
StreamWriter FileWriter = new StreamWriter("File Neve");
```

```
StreamReader FileReader = new StreamReader("File Neve");
```

Methods

Write("Text")	Szöveget ír a megnyitott file-ba
---------------	----------------------------------

WriteLine("Text")	Szöveget ír és új sort kezd
-------------------	-----------------------------

Read()	Beolvassa a következő karaktert
--------	---------------------------------

ReadLine()	Egy egész sort olvas be
------------	-------------------------

Close()	Bezárja a file-t
---------	------------------

Properties

EndOfStream	Jelzi, ha elértük a file végét
-------------	--------------------------------

Objektumok

OpenFileDialog, SaveFileDialog :

 openFileDialog1

 saveFileDialog1

```
using System.Windows.Forms;
```

Methods

ShowDialog()

Megnyitja az ablakot

Reset()

Törli az objektum beállításait

Properties

FileName

A kiválasztott file elérési útja

Title

Az ablak fejléce

InitialDirectory

Alapértelmezett elérési út

DefaultExt

Alapértelmezett kiterjesztés

Objektumok

StreamReader példa:

```
using System.IO;
```

```
...
```

```
StreamReader reader = new StreamReader("filename.txt");  
string line;
```

```
while ((line = reader.ReadLine()) != null)  
{  
    TextBox1.AppendText(line);  
}  
reader.Close();
```

Objektumok

StreamWriter, SaveFileDialog példa

```
using System.IO;
```

```
...
```

```
// a SaveFileDialog objektum létrehozva a designer-ben, vagy:  
// SaveFileDialog sfDialog = new SaveFileDialog();
```

```
...
```

```
if(sfDialog.ShowDialog() == DialogResult.OK)  
{  
    StreamWriter writer = new StreamWriter(sfDialog.FileName);  
    writer.WriteLine("your text");  
    writer.Close();  
}
```

C# alapok

Deklaráció:

```
int i;
```

Inicializáció:

```
i = 5;
```

Egyszerre:

```
double j=1.5;
```

Int32 ↔ int, Int64 ↔ long

Tömbök:

```
double[] data = new double[16];  
data[0]=1.5;  
data[15]=2.3;
```

C# alapok

Függvények:

```
private Int32 Function(arglist)
{
    ...
}
```

private: csak az adott osztályon belülről érhető el
public: kívülről is elérhető

Függvényhívás:

```
Int32 x = Function(arglist);
```

Overloading!

Típuskonverzió:

```
x = Convert.ToDouble(Object);
string = Convert.ToString(Object);
i = Convert.ToInt(Object);
...
```

Stringek:

```
string Text = "Hello";
int length = Text.Length;
string Part = Text.Substring(start, hossz);
int index = Text.IndexOf(char);
Text = Object.ToString("Format");
```

C# alapok

string manipuláció

```
string Text = "  alma  ";
```

string hossza:

```
int length = Text.Length;
```

Trim() : eltávolítja a white space-t a string elejéről és végéről (tovább paraméterevezhető)

TrimStart(), TrimEnd() : hasonlóan, de csak a string elejéről vagy végéről

```
newText = Text.Trim(); // "alma"  
newText = Text.TrimStart(); // "alma  "  
newText = Text.TrimEnd(); // "  alma"
```

Substring() :

```
// Text.Substring(start, length);  
newText = Text.Substring(0, 4) // "  al"
```

Split() :

```
string text = "6+3";  
string[] numbers = text.Split('+');  
// numbers[0] = "6"  
// numbers[1] = "3"
```

C# alapok

string manipuláció

IndexOf() : a keresett karakter indexét adja vissza (ha nincs találat, -1-et)

```
int index = Text.IndexOf('m'); // index=4
```

ToString() :

```
// Text = Object.ToString("Format");  
double szam = 5.0133;  
Text = szam.ToString("0.00"); // fix 2 tizedesjegy, 5.01
```

C# alapok

Karakterek:

```
char c='g';  
c=(char)103;           //ASCII 'g' karakter  
string Text=c.ToString();  
char[] Text2=Text.ToCharArray(); //string->karakterlánc  
Text2[0]=c;           //karakterlánc feltöltése
```



Speciális karakterek:

```
char c;  
c='\t';               //Tabulátor  
c='\n';               //új sor  
c='\r';               //Carriage return  
c='\\';               //Backslash  
c='\'';               //Idézőjel  
c='\"';               //Dupla idézőjel
```


C# alapok

if elágazás:

```
int seconds = 0;
int minutes = 0;
...
if (seconds == 59)
{
    seconds = 0;
    minutes++;
}
else
    seconds++;
```

Másik példa:

```
if (day == 0)
    dayName = "Sunday";
else if (day == 1)
    dayName = "Monday";
...
else if (day == 6)
    dayName = "Saturday";
else
    dayName = "unknown";
```

switch elágazás :

```
switch (day)
{
    case 0 :
        dayName = "Sunday";
        break;
    case 1 :
        dayName = "Monday";
        break;
    case 2 :
        dayName = "Tuesday";
        break;
    ...
    default :
        dayName = "Unknown";
        break;
}
```

- Csak beépített adattípusokra (pl. int, string)
- A felvett értéket konstanshoz kell hasonlítani

C# alapok

while ciklus:

```
int i = 0;
while (i < 10)
{
    MessageBox.Show(i.ToString());
    i++;
}
```

do ciklus:

```
int i = 0;
do
{
    MessageBox.Show(i.ToString());
    i++;
}
while (i < 10);
```

for ciklus:

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

Példa több inicializációra:

```
for (int i = 0, j = 10; i <= j; i++, j--)
{
    ...
}
```

C# alapok

Math osztály

Matematikai függvények:

Math.Cos(rad) : koszinusz függvény

Math.Sin(rad) : szinusz függvény

Math.Min(szám1, szám2) : a kisebbik számot adja vissza

Math.Abs(szám) : abszolútérték függvény

Math.Pow(alap, kitevő) : hatványfüggvény

Math.Exp(x) : e^x

Math.Round(szám, tizedesjegyek) : kerekítés

Math.Sqrt(szám) : négyzetgyökvonás

...

Beépített állandók:

Math.PI: π

Math.E: e

C# alapok

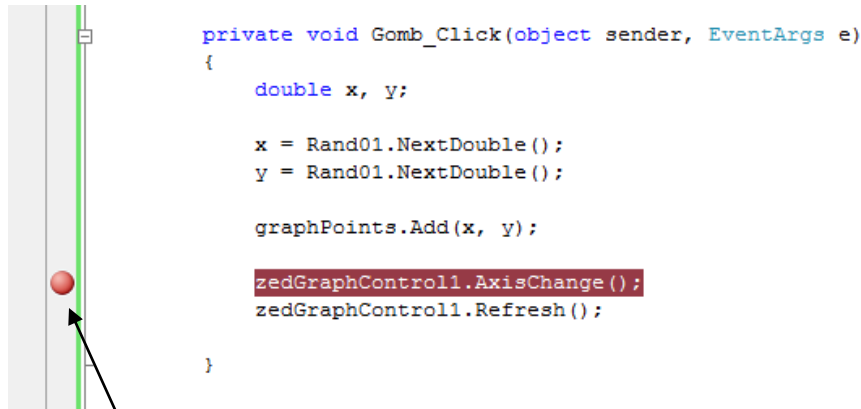
Buktatók

1. Egész számok osztása:

```
double d;  
d = 4/5; // d = 0  
d = (double) 4/5; // d = 0.8
```

2. A `Convert.ToDouble()` érzékeny a Windows területi beállításaira (tizedes elválasztójel).
3. Két darab, közvetlenül egymás után konstruált `Random` objektum ugyanazt az álvéletlen számsorozatot fogja szolgáltatni.
4. `Textbox.TextChanged()` lefut már egyetlen karakter begépelése után.
5. Összetett objektumok nem konvertálhatók, pl. `Convert.ToDouble(TextBox1)`.
Helyesen: `TextBox1.Text`.
6. A `Substring()` metódus argumentumában nem a kezdő- és végindex, hanem a kezdőindex és a hossz szerepel.

Hibakeresés (debugging)

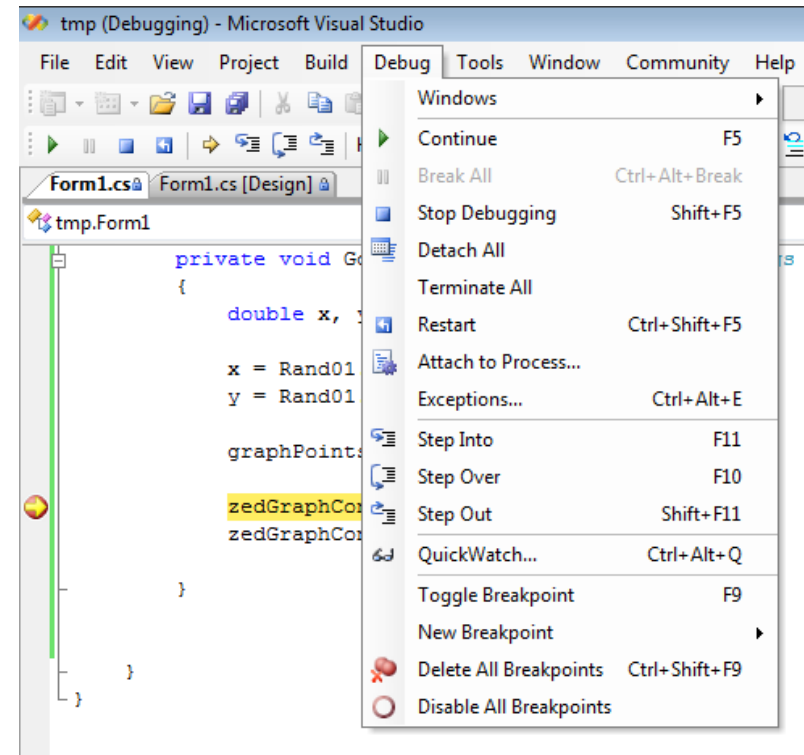


breakpoint

Változó nyomon követése:
debug módban jobb klikk a változóra > Add watch

Debug → QuickWatch:

- kifejezések kiértékelése
- objektumok manipulálása



Változók hatásköre (scope-ja)

```
namespace helloworld
{
    public partial class Form1 : Form
    {
        string globalText = "Hello World!";

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            MessageBox.Show(globalText);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string text = "Message";
            MessageBox.Show(text);
        }
    }
}
```

az osztályon belül minden függvényből elérhető

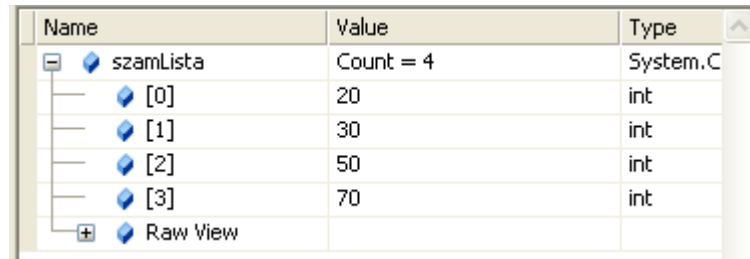
csak a button1_Click() függvényből érhető el

Listák C#-ban: <list>

Listák: a C# dinamikus tömbjei

Példa: egész számokból álló lista

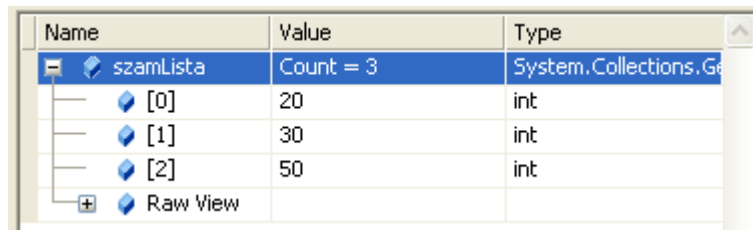
```
List<int> szamLista = new List<int>();  
szamLista.Add(20);  
szamLista.Add(30);  
szamLista.Add(50);  
szamLista.Add(70);
```



Name	Value	Type
szamLista	Count = 4	System.C
[0]	20	int
[1]	30	int
[2]	50	int
[3]	70	int
Raw View		

```
int hossz = szamLista.Count; // hossz = 4  
int elem = szamLista[2]; // elem = 50
```

```
szamLista.RemoveAt(3);
```



Name	Value	Type
szamLista	Count = 3	System.Collections.Ge
[0]	20	int
[1]	30	int
[2]	50	int
Raw View		

További metódusok: Clear(), Find(), Sort() ...

Gyakorló feladatok

1. Módosítsuk a „Hello World!” programot úgy, hogy gombnyomásra egy TextBox szövegét írja ki a MessageBox!
2. Írjuk ki valamely tulajdonságát a StartButton-nak a gomb lenyomásának hatására (pl.: Text, Font, TextAlign...)
3. Kérjünk be két számot plusz jellel elválasztva TextBoxba, majd írjuk ki az eredményt egy másik TextBoxba!
4. Írjuk ki az egész számokat 1-től 100-ig, vesszővel elválasztva egy file-ba felhasználva egy SaveFileDialog objektumot!
5. Az XY_data.txt file összetartozó [X,Y] adatokat tartalmaz. Olvassuk be a tartalmát és számoljuk ki külön-külön az X és Y adatok átlagát és szórását! Használjuk az OpenFileDialog objektumot!

A szórás számítása:
$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2 = \left(\frac{1}{N} \sum_{i=1}^N x_i^2 \right) - \mu^2$$

where

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i$$