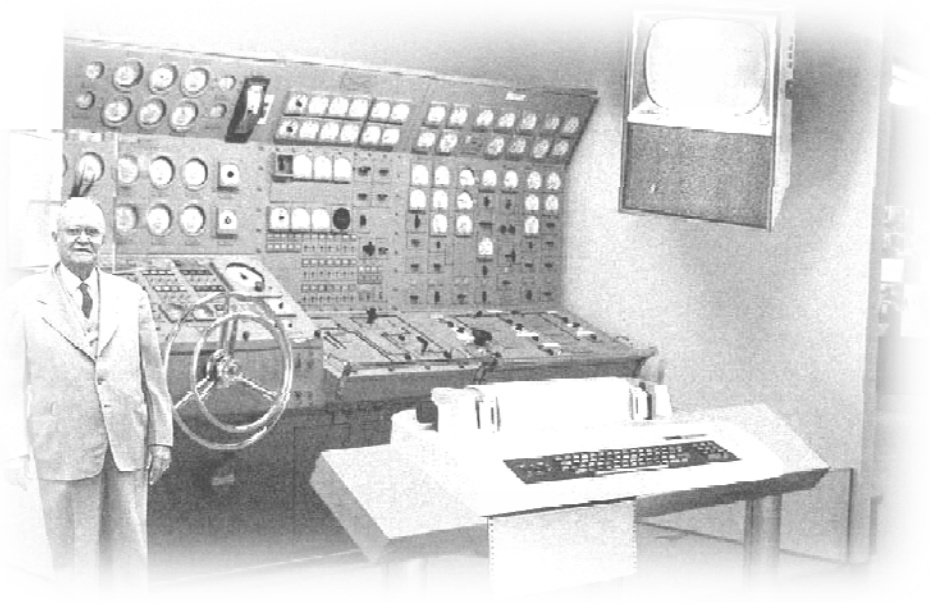
Számítógépes mérésvezérlés



Fizika Laboratórium KF
2011/2012 őszi félév

BME TTK Fizika Tanszék

Tudnivalók

- Programozás (2 alkalom)
- Mérésvezérlési feladatok (3 alkalom)
- Laboratóriumi mérések (3 alkalom)
- Segédanyagok, csoportbeosztás:
  - http://dept.phy.bme.hu/education/szilfiz_labor.html
  - login/pass: ttk/fizikus
- Visual Studio 2005
  - Beszerzés egyetemi hálózatról:
  - ftp://software.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

Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
real-time kiértékelés

Eszközök:

Mérőműszer: valamilyen fizika mennyiség mérésére;
Számítógép: adatok gyűjtése, megjelenítése, feldolgozása

Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
real-time kiértékelés

Eszközök:

<u>Mérőműszer</u>: valamilyen fizika mennyiség mérésére;
Számítógép: adatok gyűjtése, megjelenítése, feldolgozása

Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
real-time kiértékelés

Eszközök:

<u>Mérőműszer</u>: valamilyen fizika mennyiség mérésére;
Számítógép: adatok gyűjtése, megjelenítése, feldolgozása

Számítógépes mérésvezérlés

Feladatok: automatizált mérés, adatgyűjtés
real-time kiértékelés

Eszközök:

Mérőműszer: valamilyen fizika mennyiség mérésére;
Számítógép: adatok gyűjtése, megjelenítése, feldolgozása

Kommunikáció:
szabványos csatolófelületek:

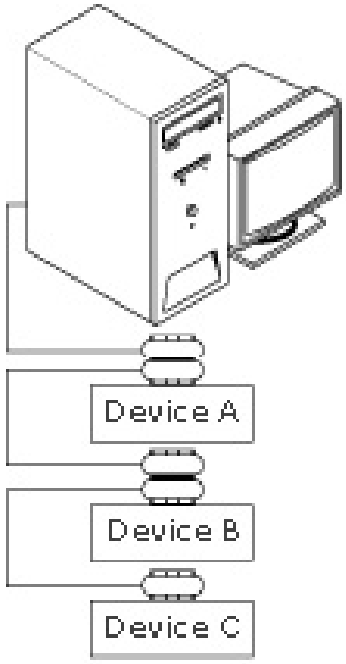
- RS-232

- USB

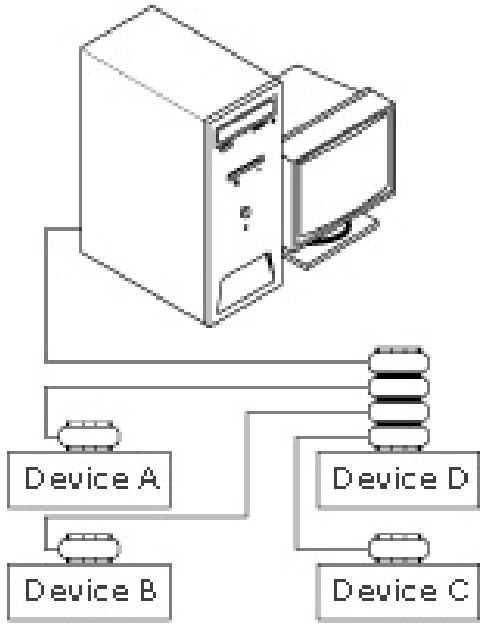
- LPT

- GPIB

...

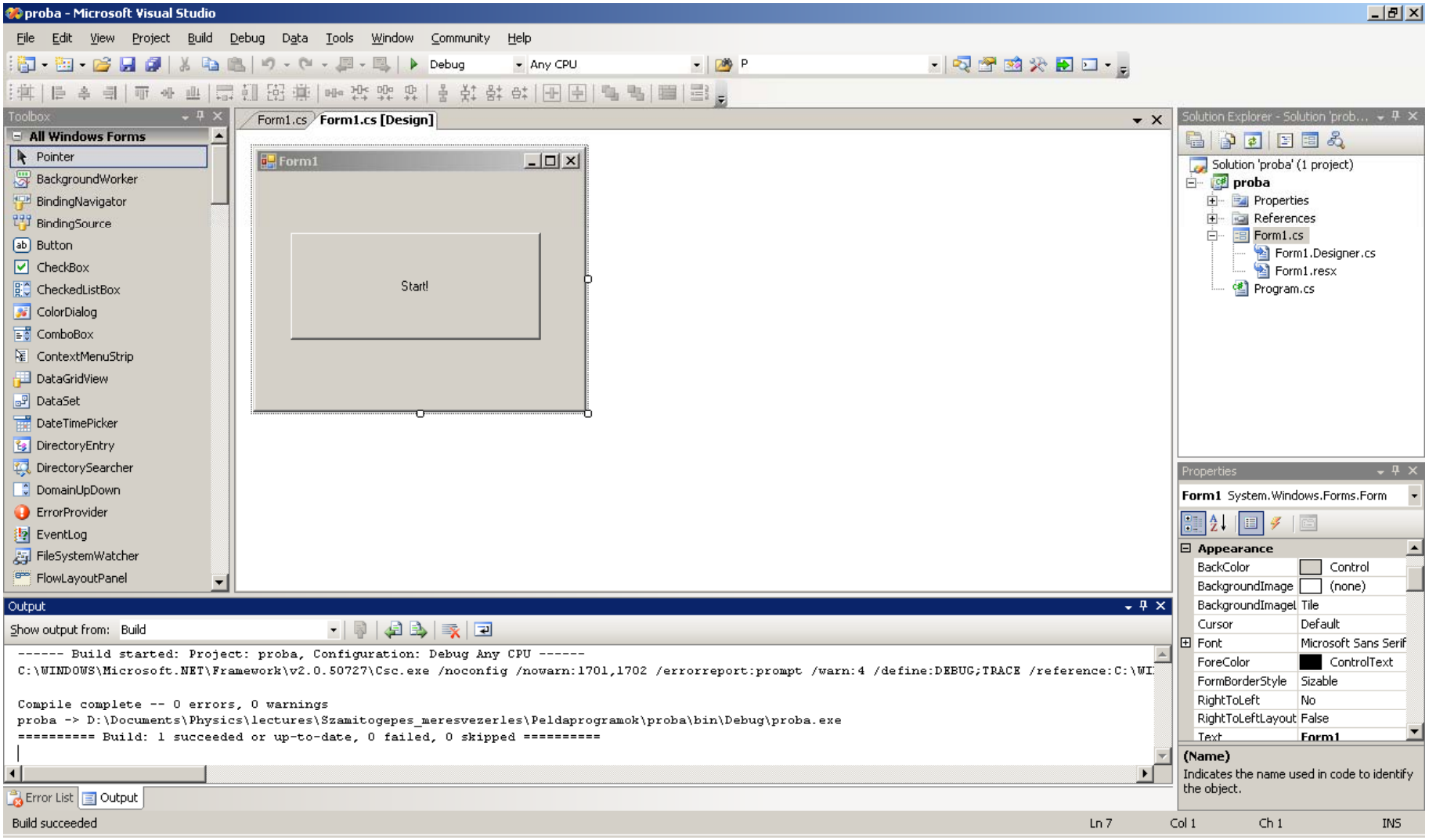


a. Linear Configuration

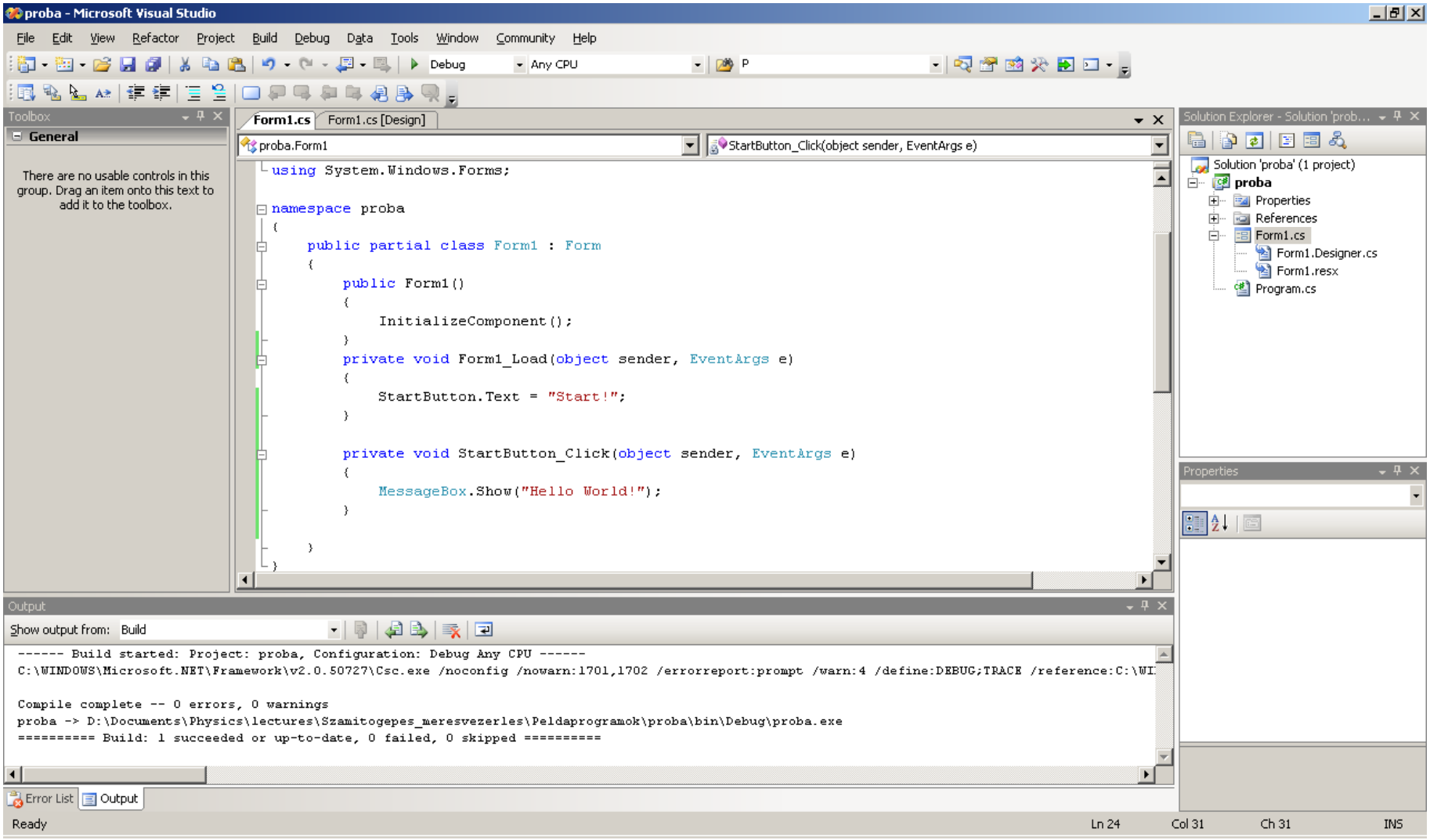


b. Star Configuration

Objektumorientált programozás – Visual Studio 2005



Objektumorientált programozás – Visual Studio 2005



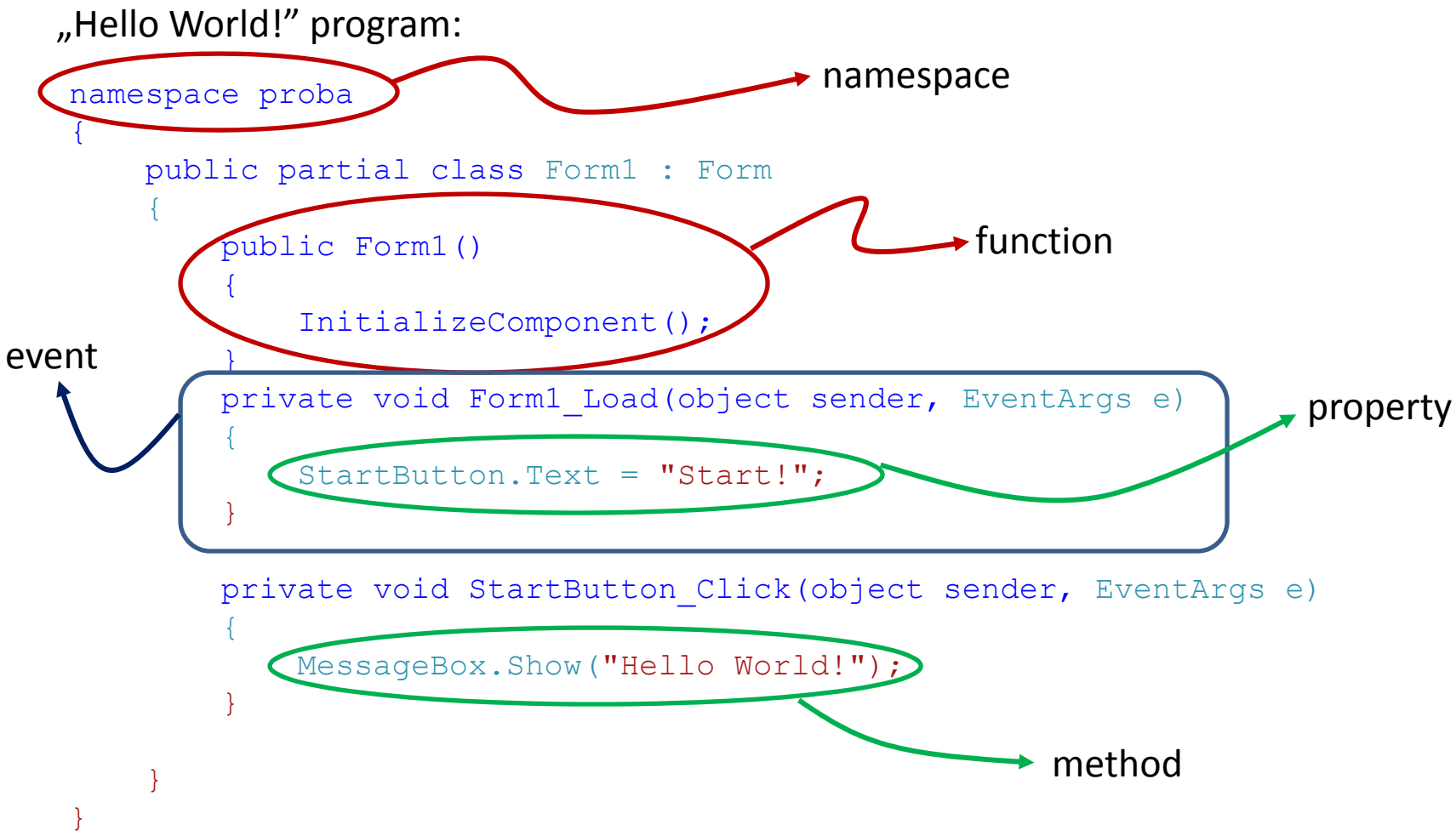
Objektumorientált programozás – Visual Studio 2005

„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!");
        }

    }
}
```

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:

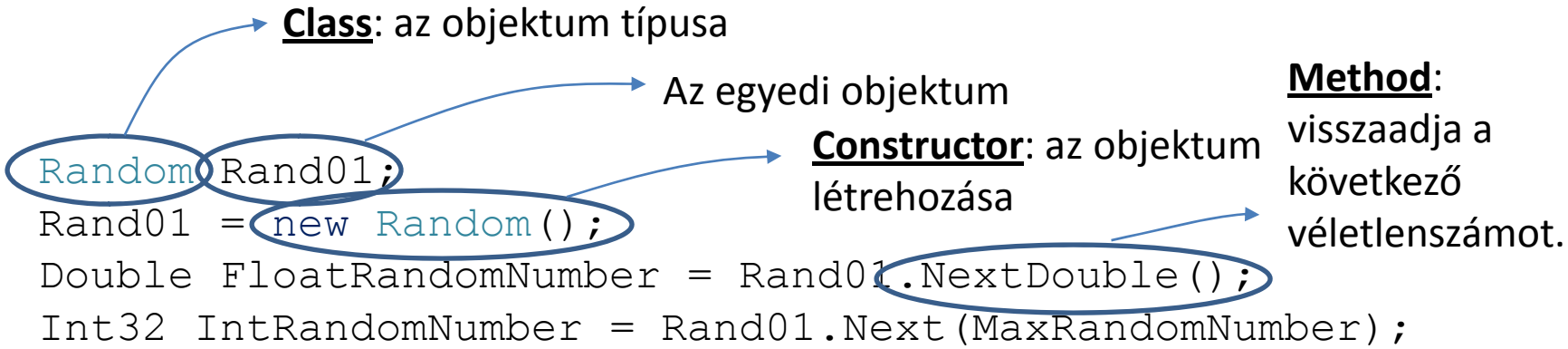
- gyors fejlesztés
- modularitás
- eseményvezérelt működés

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:



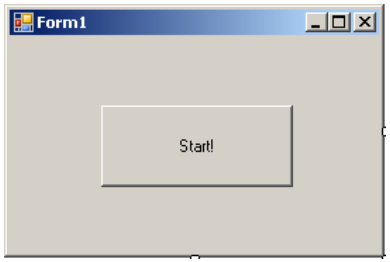
Objektumorientált programozás - alapok

Mi is az az objektum?

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

Forrás: Wikipédia

Példa: nyomógomb:



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;
```

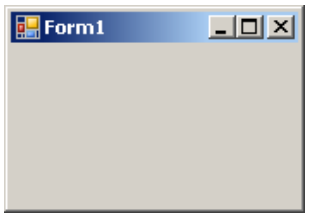
Properties	
Name	Az objektum azonosítója
Text	A gomb felirata

Methods	
Hide	Elrejti 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	Elrejti 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	Elrejti 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 :


```
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)
{
        Console.WriteLine(line);
}
reader.Close();
```

Objektumok

StreamWriter, SaveFileDialog példa

```
using System.IO;

...

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 paraméterezhető)
`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”
```

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)
{
    Console.WriteLine(i);
    i++;
}
```

`do` ciklus:

```
int i = 0;
do
{
     Console.WriteLine(i);
     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 is konvertálhatók stringgé, pl. Convert.ToString(TextBox).

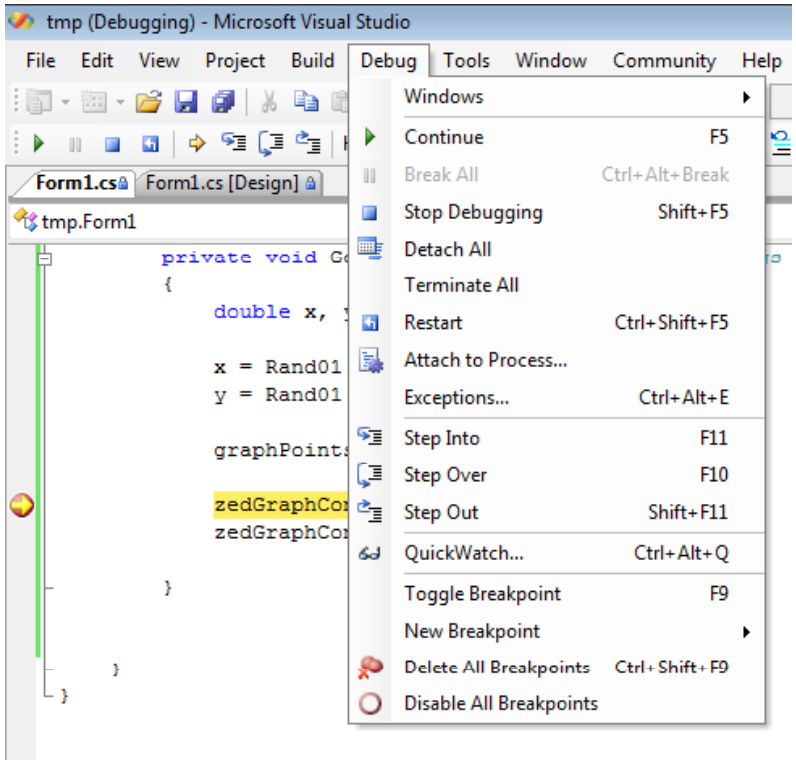
Debugging

```
private void Gomb_Click(object sender, EventArgs e)
{
    double x, y;

    x = Rand01.NextDouble();
    y = Rand01.NextDouble();

    graphPoints.Add(x, y);

    zedGraphControl1.AxisChange();
    zedGraphControl1.Refresh();

}
```

breakpoint

Debug > QuickWatch:

- kifejezések kiértékelése
- objektumok manipulálása

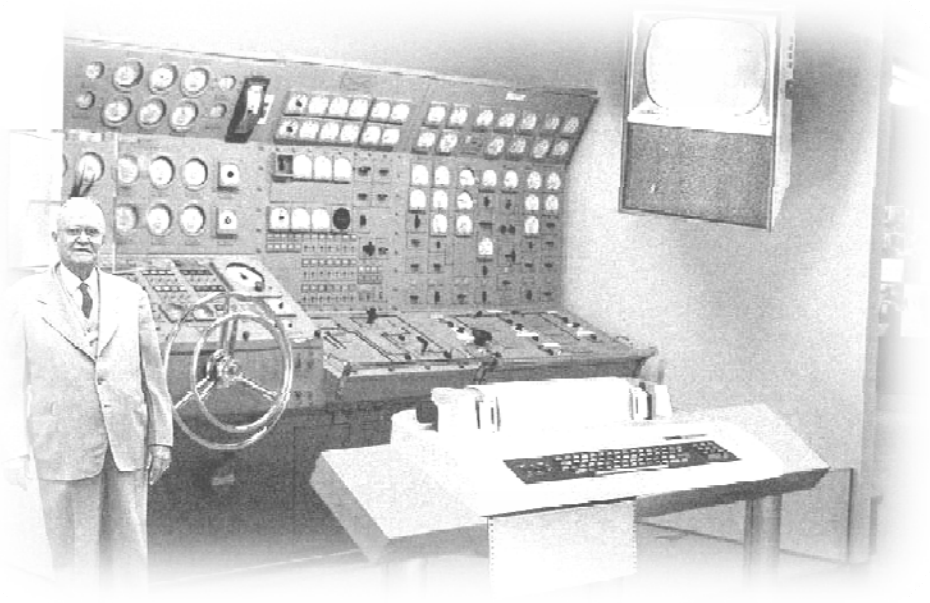


Változó nyomon követése:
debug módban jobb klikk a változóra > Add watch

Gyakorló feladatok

1. Módosítsuk a „Hello World!” programot a következőképpen: tegyünk rá szövegablakot (TextBox), a meglévő gomb lenyomására növelje eggyel a szövegablakban kiírt szám értékét (a szám 0-ról induljon).
2. Tegyünk még egy gombot (Button) a Formra, erre kattintáskor a TextBox szövegét írja ki egy MessageBox!
3. Írjuk ki a Textboxba valamely tulajdonságát a Formnak az egyik gomb lenyomásának hatására (pl.: Name, Left, Text, Font, Top ...)
4. Kérjünk be két számot plusz jellel elválasztva TextBoxba, majd írjuk ki az eredményt egy másik TextBoxba!
5. Írjuk ki az egész számokat 1-től 100-ig, vesszővel elválasztva egy file-ba felhasználva egy SaveFileDialog objektumot!
6. A 01.dat 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!

Számítógépes mérésvezérlés
2. előadás



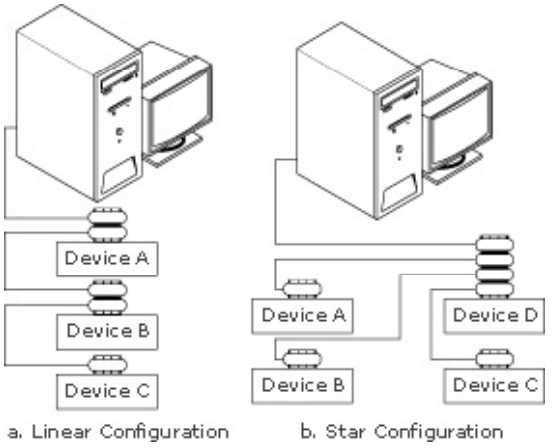
Fizika Laboratórium KF
2011/2012 őszi félév

BME TTK Fizika Tanszék

Az előző rész tartalmából…

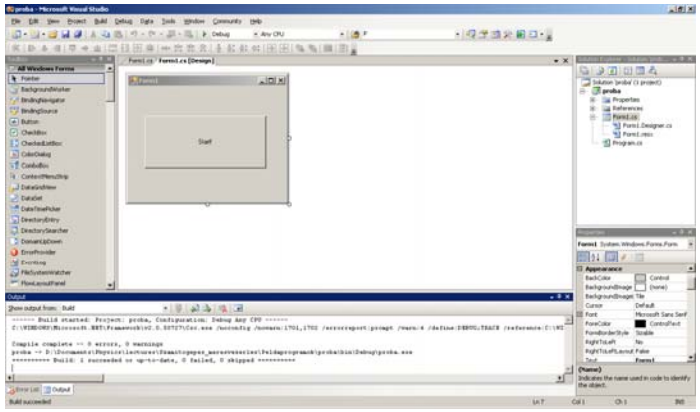
Számítógépes mérésvezérlés

Számítógép-műszer kommunikáció

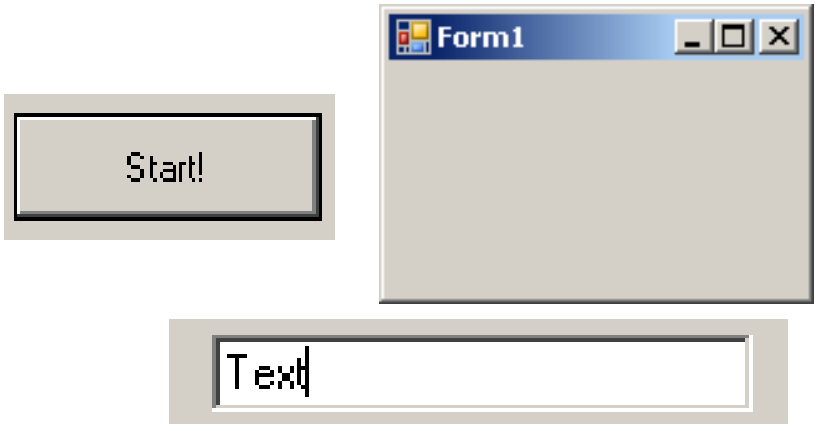


a. Linear Configuration b. Star Configuration

Visual Studio 2005



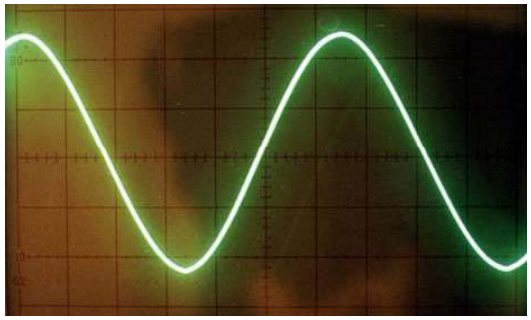
Alapvető objektumok



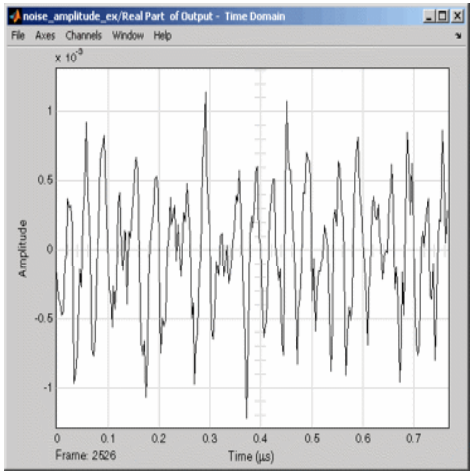
Ami még hiányzik…

A beérkezett adatok grafikus ábrázolása:

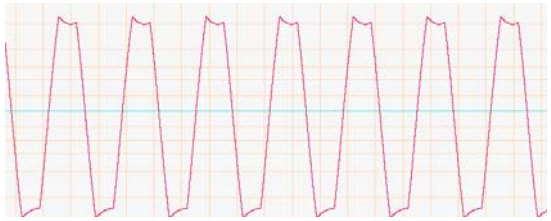
Kiértékelés „szemre”



Zajos adatsor



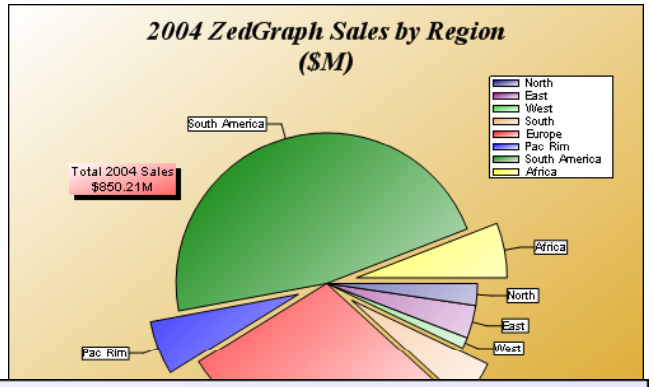
Mérőműszerek hibás működésének észlelése

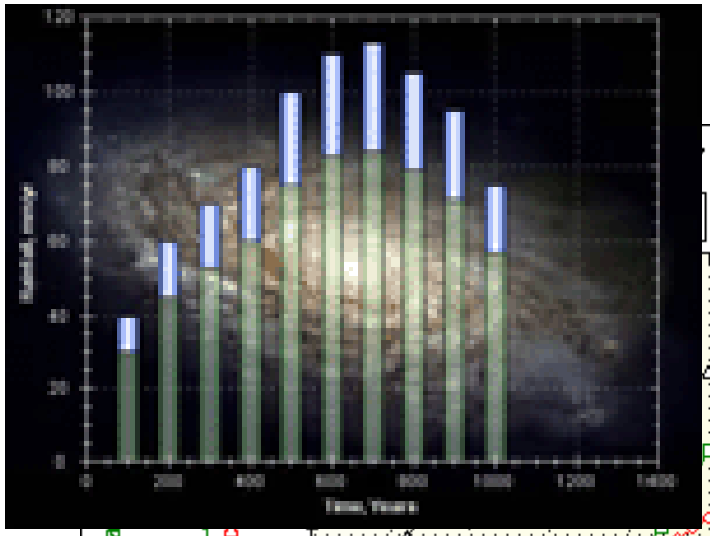


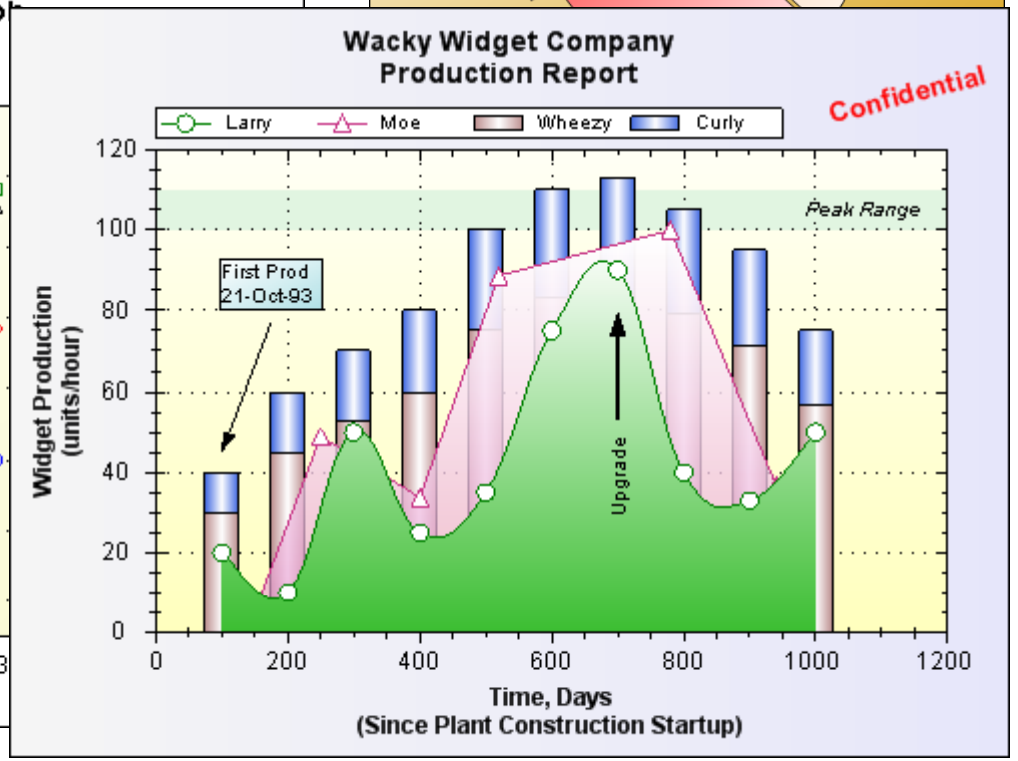
Platformok

A beérkezett adatok grafikus ábrázolása:

ZedGraph (open-source, forrás: http://zedgraph.org)





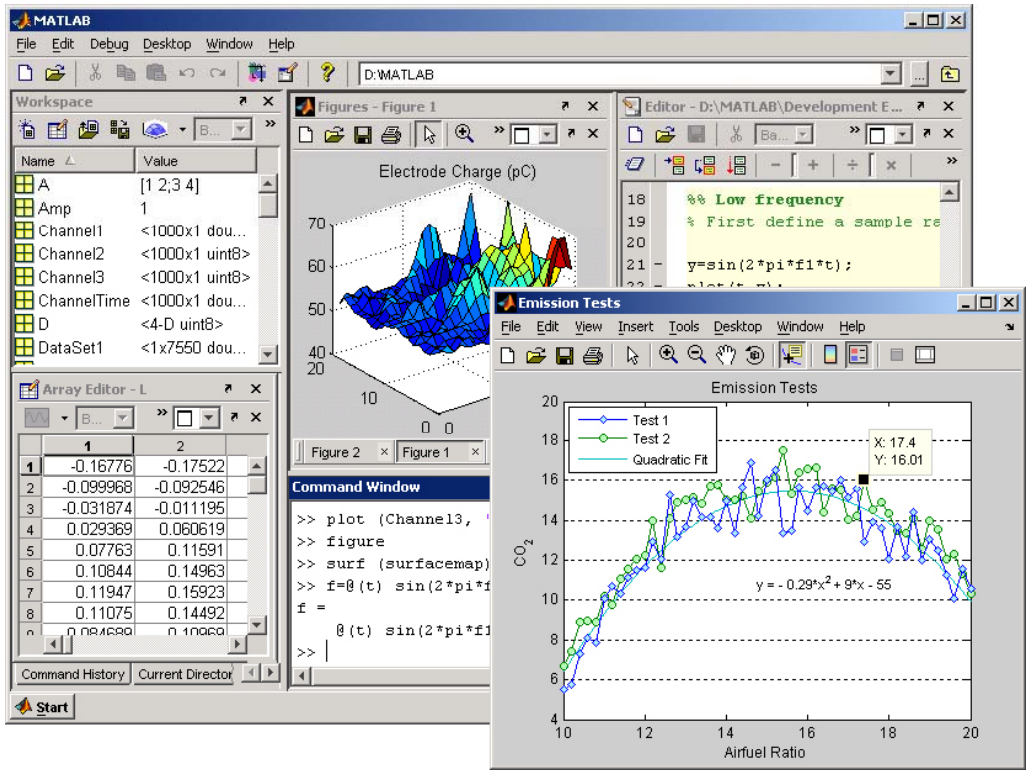


Platformok

Matlab

+ Fejlett grafikus ábrázolás
+ Matematikai támogatás
+ Multiplatform
- Műszerkommunikációhoz kiegészítő csomagok szükségesek
- Erőforrásigényes

http://www.mathworks.com

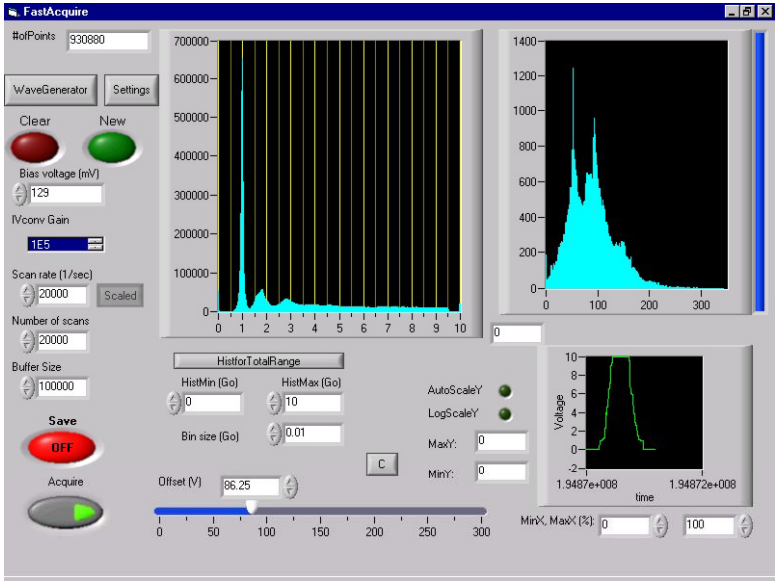


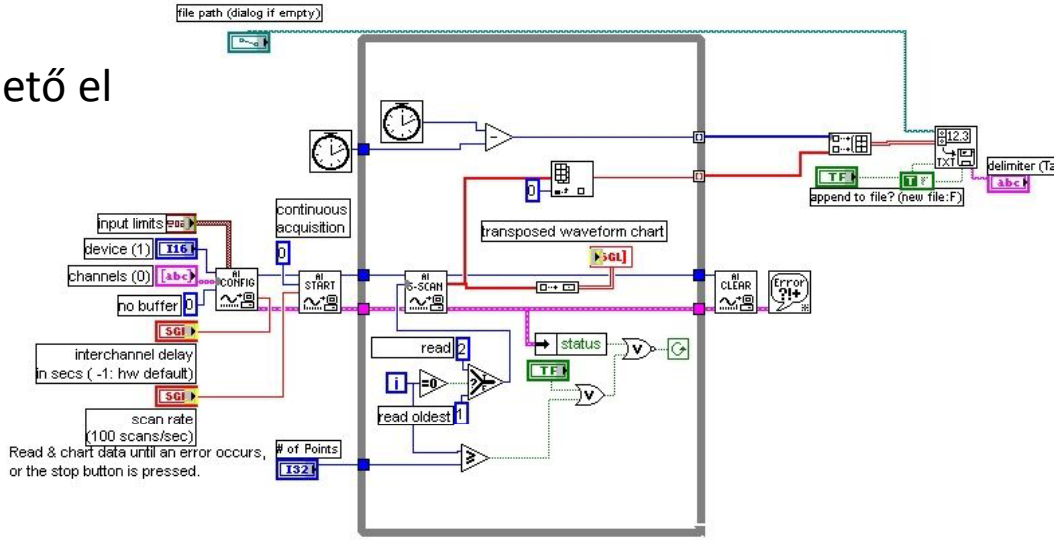
Platformok

National Instruments
Measurement Studio + Labview

http://www.ni.com

+ Felhasználóbarát felület
+ Fejlett grafikus ábrázolás
+ Közvetlen támogatás NI eszközökhöz (mérőkártyák)
- Zárt platform
- Csak MS operációs rendszerekre érhető el





Kommunikáció műszerekkel: soros port

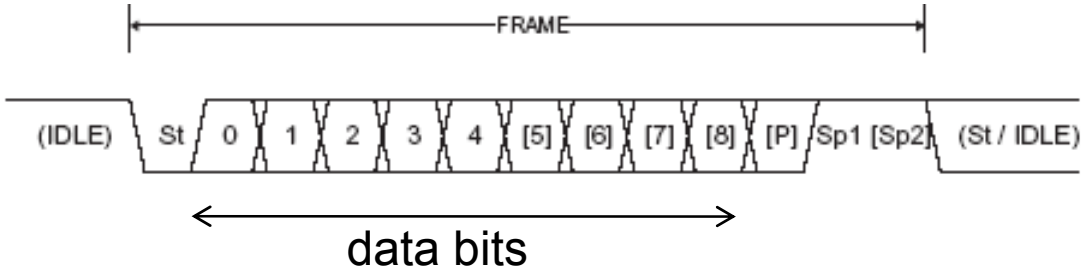
RS232: (EIA RS-232C-1969) elterjedt digitális kommunikáció számítógép és perifériák között.

Legfontosabb jellemzők:

- soros kommunikáció:

  Start bit – Adatbitek – Paritás – Stopbit

- fix, szabványos adatátviteli sebességek (pl. 9600 baud, 19200 baud)
- csillagpontos kialakítás: egy portra egy periféria csatlakozhat
- kommunikáció: 2 ér (RX, TX)
- protokoll:
  - No Flow Control
  - Hardware Flow Control
  - Software Flow Control (Xon/Xoff)



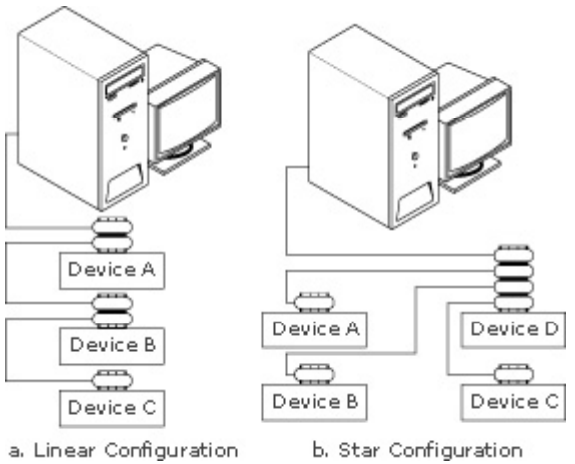
Kommunikáció műszerekkel: GPIB port

GPIB: (ANSI/IEEE Standard 488.1-1987) általánosan használt interface a számítógép és a műszerek közötti kommunikációra

Legfontosabb jellemzők:

- 8 bites párhuzamos kommunikáció
- maximális 1Mbit/sec sebességgel
- vezérlő készülék (számítógép) mellett még 14 további műszer csatlakoztatható
- készülékeket egy speciális kábellel (24 árnyékolt vezeték) kell összekötni, összekötés topológiája tetszőleges
- minden készüléknek van egy egyértelmű címe
- (PrimaryAddress), ami 0-30 közötti szám

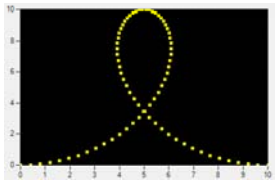
Műszerek kapcsolása:



Measurement Studio objektumok

ScatterGraph:

```
using NationalInstruments.UI;
```



Properties	
Plots[index]	A ScatterGraph objektumon belüli grafikonok gyűjteménye
Plots[0].XAxis.Range	Az x tengely skálájának szélsőértékei

Methods	
PlotXY(x_i, y_i)	Az (x_i, y_i) adatpárokat ábrázolja az adott grafikonon
PlotXYAppend(x_i, y_i)	Kiegészíti az adott grafikont az (x_i, y_i) pontpárokkal
ClearData()	Törli a grafikont

A használatához adjuk hozzá a megfelelő hivatkozást a függvénytárhoz: jobb klikk a *Solution explorer* ablakban a *References-re*, itt a .NET fülön keressük meg a *National Instruments User Interface Library*-t.

Tipp: a *Debug* menüpont alatt található *QuickWatch* eszközzel térképezzük fel a ScatterGraph objektumot!

Measurement Studio objektumok

GPIB kezelése: NI4882 Device

```
using NationalInstruments.NI4882;
```

Példaprogram:

```
NationalInstruments.NI4882.Device LockInDevice;
byte port = 8;
LockInDevice = new NationalInstruments.NI4882.Device(0, port);

LockInDevice.Write("*IDN?");
string valasz = LockInDevice.ReadString();

LockinDevice.Dispose();
```

További objektumok

Timer:

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

Properties	
Interval	Két Tick közötti időköz msec-ban
Enabled	A Timer futását engedélyezi, illetve tiltja

Events	
Tick	A beállított időköz leteltekor fut le

Random:

```
using System;
```

Constructor
Random rand01 = new Random();

Methods	
NextDouble()	A következő pszeudorandom számot adja 0 és 1 között
Next(min,max)	A megadott határok közötti egész véletlenszámot ad vissza

További objektumok

SerialPort:

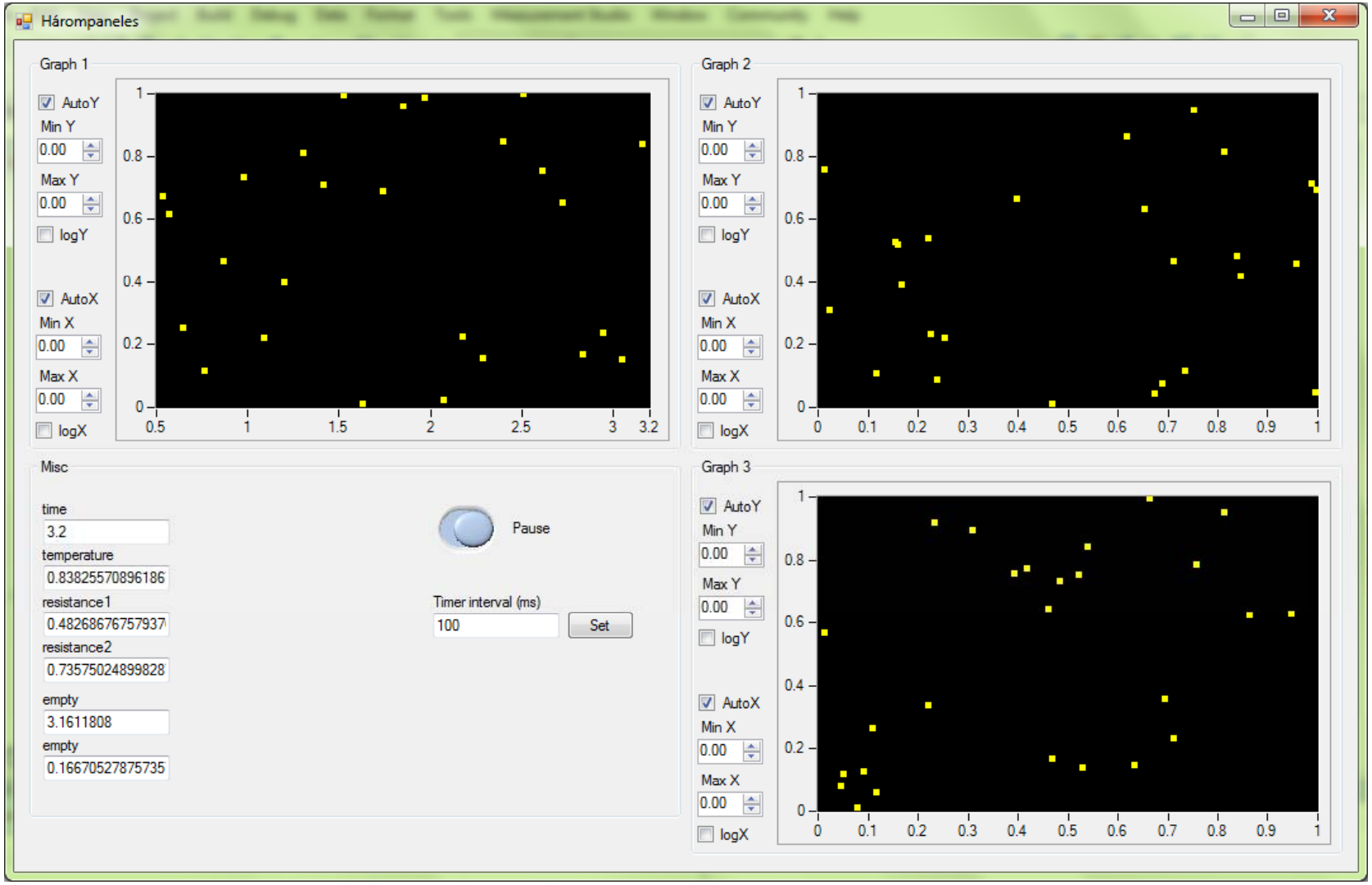
```
using System.IO.Ports;
```

Példa:

```
if (Serial1.IsOpen)
        Serial1.Close();

Serial1.PortName = ”COM1” // az eszközkezelőben látható portnév
// hasznos: string[] ports = SerialPort.GetPortNames();
Serial1.BaudRate = 38400;
… // további beállítások (DataBits, StopBits, Parity…)

Serial1.Open();


Serial1.WriteLine("*IDN?");
string valasz = Serial1.ReadLine();


Serial1.Close();
```

Példaprogram

Timer, ScatterGraph példa



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

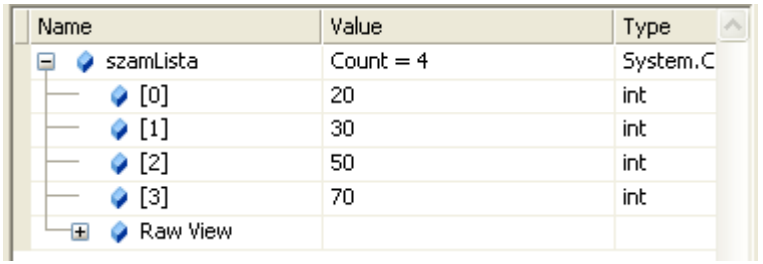
csak a button1_Click() függvényből érhető el

Listák C#-ban: <list>

Listák: a C# dinamikus tömbjei

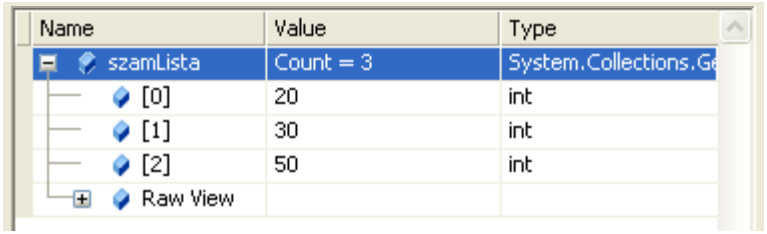
Példa: egész számokból álló lista

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




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

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



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

Gyakorló feladatok

7. Ábrázoljuk grafikonon az input.txt [X \t Y] tartalmát! Gombnyomásra alkalmazzunk egy választható ablakszélességű mozgóátlagot az adatsoron és jelenítsük meg az eredményt! Az így készült, átlagolt adatsort gombnyomásra mentsük új fájlba!
8. Írjuk át a fenti programot a következőképpen: olvassuk be a fájlt soronként, fél másodpercenként egy adatpárt, és valós időben készítsük el hozzá a mozgóátlagos adatsort, ugyanazon grafikonon jelenítsük is meg az eredetivel együtt!
9. Készítsünk Lissajous-görbe generáló programot: két Textboxban legyen beállítható az X és Y csatorna frekvenciája, és legyen változtatható a két jel fázisa!
10. Generáljunk (x,y) véletlenszám-párokat 0 és 1 között! Határozzuk meg annak a valószínűségét, hogy az így meghatározott pont az origó körüli egységsugarú negyedkörön belülre esik! Ennek érdekében egy grafikonon ábrázoljuk a negyedkört és a pontokat, egy másikon pedig a valószínűséget a generált pontok számának függvényében. A pontokat fél másodpercenként sorsoljuk, a kapott koordinátákat és valószínűségeket mentsük el egy fájlba!

Álvéletlenszámok

„Bárki, aki aritmetikai módszerekkel akar előállítani
egy véletlenszámot, a bűn állapotában leledzik.”
Neumann János

Példa álvéletlenszámok generálására:

$$X_{n+1} = (a \cdot X_n + c) \bmod m$$

m : modulus $\quad m > 0$

a : szorzófaktor $\quad 0 \le a < m$

c : inkrementum $\quad 0 \le c < m$

Pl. $m = 10,\ X_0 = a = c = 7 - \text{re}$

$7, 6, 9, 0, 7, 6, 9, 0, \ldots$

X_0 : seed

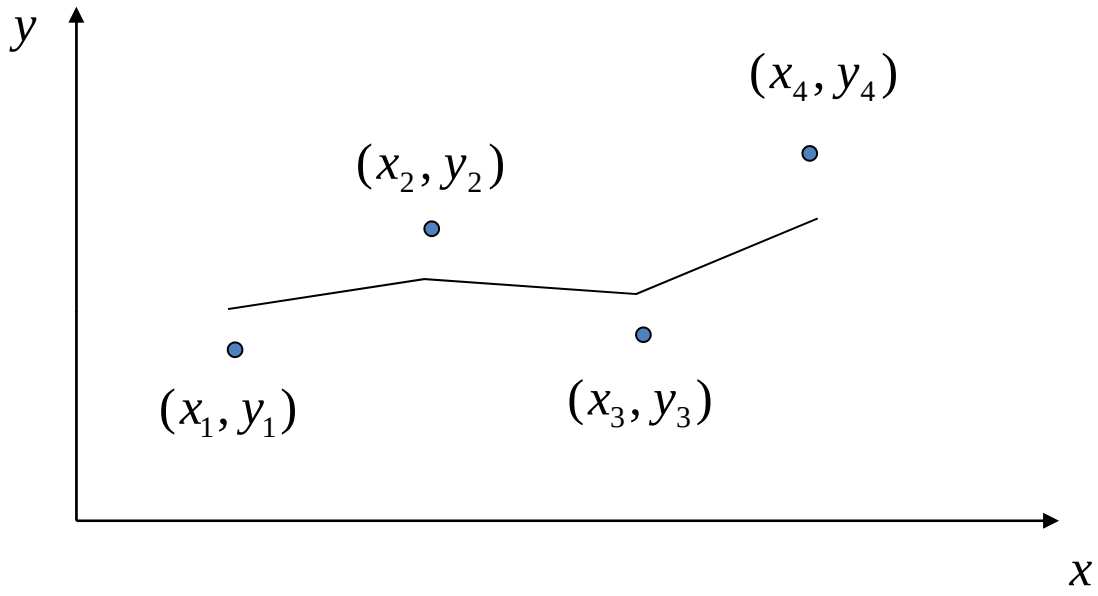
(A beépített Random osztály másképp működik.)

Emlék:

```
Random Rand01;
Rand01 = new Random();
Double FloatRandomNumber = Rand01.NextDouble();
Int32 IntRandomNumber = Rand01.Next(MaxRandomNumber);
```

forrás: Donald Knuth, The Art of Computer Programming

Mozgóátlag



Mozgóátlag 3 adatpont szélességű ablakkal:

„szimmetrikus” formula:

$$(\tilde{x}_2, \tilde{y}_2) = \left(x_2, \frac{1}{3}(y_1 + y_2 + y_3) \right)$$

„haladó” formula:

$$(\tilde{x}_2, \tilde{y}_2) = \left(x_2, \frac{1}{3}(y_2 + y_3 + y_4) \right)$$