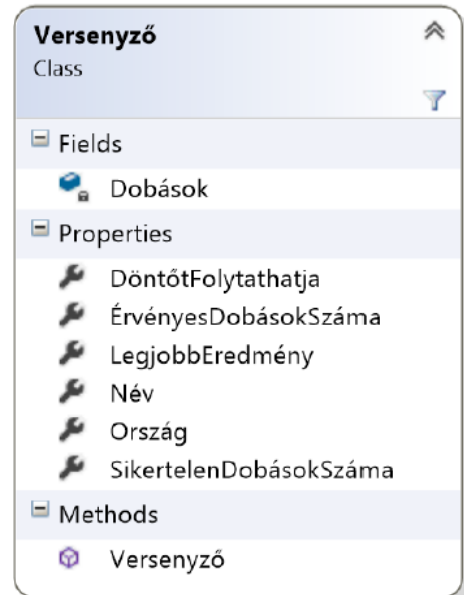


## Kalapácsvetés 2016

A 54\_481\_06\_2\_3T vizsgafeladat megoldása – a feladat szövegéből csak annyit írok le, amennyi feltétlenül kell az érthető megfogalmazáshoz. Először konzol applikációt készítettünk. Indítjuk a C# alkalmazást és **kalapacsvetes2016** néven létrehozunk egy konzol applikációt. A program írása közben több problémám is akadt, de azokat majd menetközben.

1. Osztály létrehozása. Ehhez – szokás szerint – megadnak egy képet, amely mutatja, hogy milyen adattagokat, metódusokat, illetve jellemzőket hozunk létre. Maga az osztály létrehozása a szokásos. A **Solution Explorer** ablakban helyi menüt kérünk a program néven (**kalapacsvetes2016**). A helyi menüből választjuk az **Add** lehetőséget, majd az újabb menüből az a menüsor alján megjelenő **Class...** opciót. A megjelenő párbeszédablakban beírjuk az osztály nevét, most **Versenyzo** és deklarálhatjuk az osztály elemeit.
2. Amit deklarálnunk kell: a **Dobasok** **string** tömböt. Ebben tároljuk majd a dobások értékeit. Azért kell, hogy **string** legyen, mert az x (érvénytelen dobás) is dobásnak számít. Itt nem adjuk meg az elemszámot. Azt majd a konstruktorra bízunk.



A feladat jellemzőket ír elő, de azokkal szorozott a program, így áttértem adattagokra (egyébként az adattagok könnyebben kezelhetők – én jellemzőket csak akkor használok, ha privát adattag értékét kell megjeleníteni). Természetesen nem használtam most sem ékezetes azonosítókat. Talán egyetlen apróság, hogy a konvenció szerint az adattagok neve kisbetűvel kezdődik. Azért maradtak nagybetűsek, mert a jellemzők nevei viszont nagybetűsek. Amikor adattagok lettek a jellemzőkből, már nem akartam túl sok mindent javítani.

Tehát a **Versenyzo** osztály eddigi elemei:

```
const int d = 6;
private string[] Dobasok;
public bool DontotFolytathat;
public int ErvenyesDobasokSzama;
public double LegjobbEredmeny;
public string Nev;
public string Orszag;
public int Sikertelen;
```

Az **int** típusú konstansra a tömb inicializálására van szükség (enélkül is menne az inicializálás, de olyan jól mutat a konstans)..

3. A konstruktor: Az inicializálás annyiban áll, hogy megadjuk a **Dobasok** elemszámát (azaz 6-ot). A kiírás szerint a konstruktornak egy **string** típusú paraméterrel kellene rendelkezni. Az **int** típusú paramétert azért vettem be, mert így egyszerűbben tudtam kü-

lönválasztani a döntőt folytató 8 versenyzőt. A beolvasott sorok eleje azonos az első 8 és az utána következő 4 versenyzőnél. Név, ország. Utána viszont az utolsó négy versenyző csak 3-szor kísérletezhetett és ez tömb túlsordulást eredményezett.

A konstruktor eleje:

```
public Versenyzo(string sor, int sz)
{
    Dobasok = new string[d];
    Sikertelen = 0;
    ErvenyesDobasokSzama = 0;
    string[] db = sor.Split(';');
    Nev = db[0];
    Orszag = db[1];
}
```

A paraméterként kapott sor argumentumot a **Split** metódussal daraboljuk fel, az elválasztóként megadott pontosvesszővel. A darabolás eredményét a **db string** tömbbe rakom. Ezután innen pakolom át az egyes elemeket a megfelelő változóba. A **Név** és **Orszag** már meg is kapja a megfelelő elemet.

A konstruktor következő részében a **Dobasok** tömböt töltöm fel a **db** további elemeivel. A **db** indexénél **2**-vel nagyobb értéket kell használnom, mert a **Név** és **Orszag** már elvitte az első két elemet. Itt van szükségem az **int** típusú paraméterre. Az első 8 versenyző **6** kísérletet tehetett, a többiek csak **3**-at. Így aztán ha **6** értéket akartam nekik is adni tömb túlsordulás lett. Ettől kiakadt a program is, én is (de a programírás erről szól – ezért kellene átnéznem azokat a programokat, amiket átküldök). A **DontotFolytathat** adattag itt kap értéket.

```
if (sz < 8)
{
    for (int i = 0; i < 6; i++) Dobasok[i] = db[i + 2];
    DontotFolytathat = true;
}
else
{
    for (int i = 0; i < 3; i++) Dobasok[i] = db[i + 2];
    DontotFolytathat = false;
}
```

Végül következik egy foreach ciklus, melyben a Dobasok tömb elemeit számokká konvertálok. Természetesen a az x ebből kimarad. Itt kapnak értéket a további adattagok. Az ErvenyesDobasokSzama és a Sikertelen értékét csak növelni kell a megfelelő opciónál. A LegjobbDobas 0-ról indul. Ha az aktuális dobás értéke nagyobb nála, akkor kapja annak értékét. Tehát a záróciklus:

```
LegjobbEredmeny = 0;
foreach(string i in Dobasok)
{
    if (i != "x")
```

```

    {
        ErvenyesDobasokSzama++;
        if (LegjobbEredmeny < Convert.ToDouble(i))
            LegjobbEredmeny =
                Convert.ToDouble(i);
    }
    else Sikertelen++;
}

```

Elkészült az osztálydeklaráció. Jöhet a főprogram. Lehet, számotokra egyszerűbbnek tűnne, mindent ömlesztve beírni a főprogramba. Én mégis inkább a függvények deklarálása és hívása mellett maradok. A főprogram (Main) egyszerűbb, áttekinthetőbb, ráadásul egy-egy függvény egy-egy részfeladatot old meg.

4. Adatbeolvasás. Ehhez szükségünk van a **System.IO** névtérre. Ne felejtsetek el felvenni, különben a kulcsszavak hibát jeleznek. Nálam a forrásállományok az **E:\forras** mappában vannak. Innen töltöm be őket. Így: **"E:\\forras\\olimpia2016.txt"** a fájlnev az elérési úttal. Ez az egyik megadási mód, amikor kettőzöm a \ karaktert. A **@"E:\forras\olimpia2016.txt"** lenne a másik lehetőség. Természetesen ti a saját elérési utakat írjátok az enyém helyett. Van még egy harmadik lehetőség, ehhez a forrásállományt be kell másolni az aktuális program **\bin\debug** alkönyvtárába. Ilyenkor elég csak a fájlnevet megadni.

Mielőtt bármit tennénk, deklarálunk egy **Versenyzo** típusú statikus tömböt – mondjuk 20 elemmel:

```
static Versenyzo[] ver = new Versenyzo[20];
```

Ezután deklaráljuk a **Beolvas** függvényt egész típusú referenciaváltozó paraméterrel. Ez a paraméter adja majd vissza az adatállományban lévő sorok számát (erre van szükségünk a következő feladat megoldásához). A függvénynek nincs visszatérési értéke, hisz az állományból beolvasott adatokat a **Versenyzo** típusú **ver[]** tömb tárolja. Megnyitjuk a fájlt olvasásra. Beolvasunk egy sort, majd létrehozunk egy példányt a **Versenyzo** osztályból, amelynek paraméterül átadjuk a beolvasott sort, illetve a sor számát. Ezután az **uj** változót értékül adjuk a **ver[]** tömb következő elemének. Ha elértük a fájl végét, lezárjuk az állományt:

```

static void Beolvas(ref int db)
{
    string fnev = @"E:\forras\olimpia2016.txt";
    StreamReader f = File.OpenText(fnev);
    while (!f.EndOfStream)
    {
        string sor = f.ReadLine();
        Versenyzo uj = new Versenyzo(sor, db);
        ver[db] = uj;
        db++;
    }
    f.Close();
}

```

```
}
```

5. A döntőbe jutott versenyzők számának kiírása. Ez már a Main részeként kerül megoldásra. A főprogram elején deklarálunk egy egész típusú változót. Ez lesz a referenciaparaméter a Beolvas() függvénynek. Az ebben visszakapott értéket kell kiíratnunk a képernyőre.
6. Hányan maradtak a döntőben. Ehhez a ver[] tömb elemeit végigolvassuk és ha az aktuális DontotFolytathat igaz értékű, akkor a számláló értékét növeljük. Ezt a feladatot az int visszatérési értékű Donto() függvény oldja meg. Mindössze annyit kell tennünk a főprogramban, hogy kiíratjuk a függvény által visszaadott értéket.

```
static int Donto()
{
    int darab = 0;
    for(int i = 0; i < 12; i++)
    {
        if (ver[i].DontotFolytathat) darab++;
    }
    return darab;
}
```

7. A statisztika elkészítését a **Statisztika** függvény végzi. A visszatérési érték nélküli **Statisztika** függvény elején kiíratjuk a fejléctet. Eztán **for** ciklussal végigmegyünk a **ver[]** tömb elemein és kiíratjuk a szükséges adatokat. A sorok kiírásánál a **\t** a tabulátor lenyomását helyettesíti.

```
static void Statisztika()
{
    Console.WriteLine("7. feladat: Statisztika: (név; ér-  
vényes_dobás; sikertelen_dobás; leg-  
jobb_dobás");
    for (int i = 0; i < 8; i++)
    {
        Console.WriteLine("\t {0}; {1}; {2}; {3}",  
ver[i].Nev, ver[i].ErvenyesDobasokSzama,  
ver[i].Sikertelen, ver[i].LegjobbEredmeny);
    }
}
```

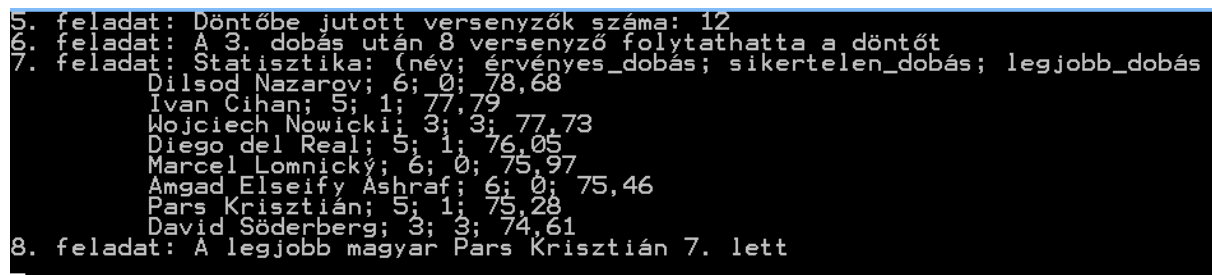
8. A magyar versenyző nevének és eredményének lekérdezését a **Magyar** függvény végzi. A függvény egész típusú értéket ad vissza. Addig lépegetek a **ver[]** tömb elemein, amíg az **Ország** nem **Magyarország** nevét tartalmazza. Ha megtaláltam, mindössze az index értékét kell megkapnom, az alapján összeállíthatom a megjelenítendő üzenetet.

```
static int Magyar()
{
    int i = 0;
    while (ver[i].Ország != "Magyarország") i++;
    return i;
}
```

Már csak a főprogram van hátra. Tulajdonképpen a megfelelő függvényhívásokat tartalmazza a feladat megoldásának sorrendjében, illetve az előírt üzeneteket kell megjeleníteni:

```
static void Main(string[] args)
{
    int db = 0;
    Beolvas(ref db);
    Console.WriteLine("5. feladat: Döntőbe jutott versenyzők  
száma: {0}", db);
    Console.WriteLine("6. feladat: A 3. dobás után {0} ver-  
senyző folytathatta a döntőt", Donto());
    Statisztika();
    Console.WriteLine("8. feladat: A legjobb magyar {0} {1}.  
lett", ver[Magyar()].Nev, Magyar()+1);
    Console.ReadKey();
}
```

A futtatás után a képernyőkép:

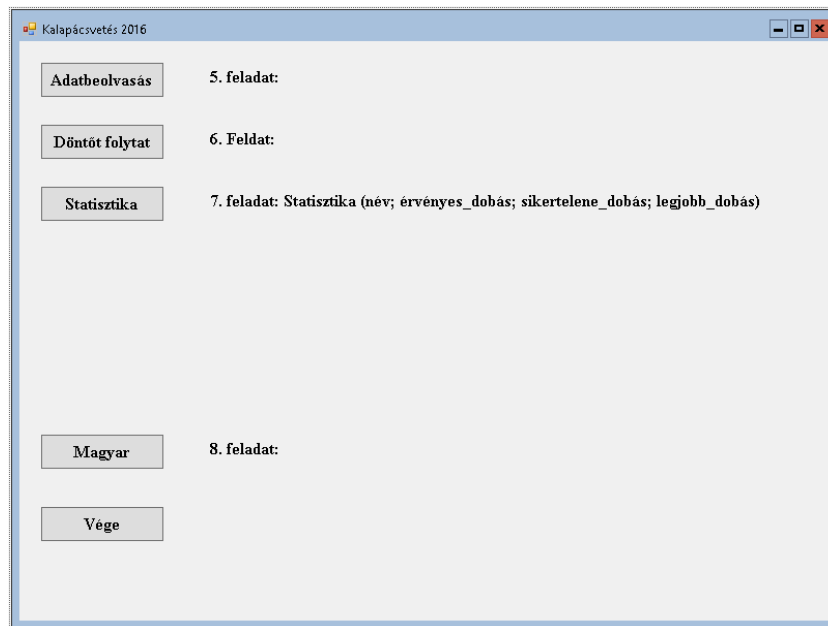


```
5. feladat: Döntőbe jutott versenyzők száma: 12
6. feladat: A 3. dobás után 8 versenyző folytathatta a döntőt
7. feladat: Statisztika: (név; érvényes_dobás; sikertelen_dobás; legjobb_dobás
Dilsod Nazarov; 6; 0; 78,68
Ivan Cihan; 5; 1; 77,79
Wojciech Nowicki; 3; 3; 77,73
Diego del Real; 5; 1; 76,05
Marcel Lomnický; 6; 0; 75,97
Amgad Elseify Ashraf; 6; 0; 75,46
Pars Krisztián; 5; 1; 75,28
David Söderberg; 3; 3; 74,61
8. feladat: A legjobb magyar Pars Krisztián 7. lett
```

## A feladat grafikus megoldása

Az osztálydeklarációval nem foglalkozom, mert ugyanazt az osztály használtam, amit a feladat konzol változatánál.

A grafikus alkalmazás elkészítéséhez **Windows Forms** applikációt indítunk. Ugyanúgy lehet a program neve **kalapacsvetes2016**, ha más mappába készítjük. Nálam most csak **kalapacs** lett a program neve. Elkészítjük az alábbi formot:



A gombokat most nem neveztem el. A címkék közül is csak a **fel5**, **fel6** és **fel8** címkék kapták az előbbi neveket. A címkék mindegyikénél **false**-re állítottam az **AutoSize** tulajdonságot és minden címke **600;20** méretű lett. Ez azért szükséges, mert így a teljes szöveget megjeleníti. Ezekre hivatkoznom kell, emiatt neveztem át. A gombokra is kell hivatkoznom, de a tervezéskor még nem sejtettem.

Vegyük észre a szembetűnő különbséget a grafikus és a konzol megoldás között. Itt az üzeneteket címkékben jelenítem meg (nem **Console.WriteLine** metódussal). Nem függvényeket deklarállok, hanem a részfeladatokhoz nyomógombot rakok fel a formra, majd az egyes nyomógombokhoz írom meg a részfeladatokat.

Mindent egybevetve kész a váz. Hasonlóan a konzolalkalmazáshoz itt is deklarálnom kell a **Versenyző** típusú tömböt. Ez viszont grafikus felületen másképp történik. Kattintsunk kettő a formon, felül megjelenik a **Form1.cs** fül. A deklarációt itt követem el. Az inicializálást viszont a **Form1** konstruktorban:

```
Versenyző[] ver;
public Form1()
{
    InitializeComponent();
    ver = new Versenyző[20];
}
```

Kész az osztály, így nekieshetünk a részfeladatok megoldásának. Első lépés az adatbeolvasás. Ehhez az **adatbeolvasás** gombon kattintunk kettőt és a **Beolvas()** függvényhez hasonlóan járunk el:

```
string fnev = "E:\\forras\\olimpia2016.txt";
StreamReader f = File.OpenText(fnev);
int db = 0;
while (!f.EndOfStream)
{
    string sor = f.ReadLine();
```

```

        Versenyzo uj = new Versenyzo(sor, db);
        ver[db] = uj;
        db++;
    }
    fel5.Text = "5. feladat: Döntőbe jutott versenyzők szá-
                ma: " + db.ToString();

```

A lényeges különbség, hogy ide kerül az 5. feladat megoldása.

A 6. feladat megoldása a **Döntőt folytat** nyomógombhoz kerül. Ez teljesen hasonló a **Döntő** függvényénél láthatóhoz. Lényeges különbség, hogy ide kerül a megjelenítendő üzenet.

```

private void button2_Click(object sender, EventArgs e)
{
    int darab = 0;
    for (int i = 0; i < 12; i++)
        if (ver[i].DontotFolytathat) darab++;
    fel6.Text = "A 3. dobás után " + darab.ToString() + "
                versenyző folytathatta a döntőt";
}

```

A **Statisztika** nyomógomb programja kissé kacifántos, bár – ha tudod, hogy mit ért teszel – akkor ez sem bonyolult. Az adatmegjelenítés miatt vannak plusz dolgok. Valahogy meg kell jeleníteni az egyes sorokat. Erre az egyik legalkalmasabb megoldás a dinamikus címkék alkalmazása. Ezek a címkék a Form1 felületre kerülnek, így a szülő objektum a **Form1.ActiveForm**. Be kell állítani a címkék helyzetét. A bal oldaltól való távolság 300 képpont. A form tetejétől való távolság változik. Ezt a le egész típusú változóval oldottam meg. Ahányszor kiírnunk egy új sort, mindannyiszor nő az értéke 20 képponttal. A címkék szélessége és magassága ugyanakkora. Az **AutoSize** itt is **false**. Ha az aktuális **DontotFolytathat** igaz, akkor előállítom a megjelenítendő sort, majd létrehozom az új címkét a megjelenítéshez.

```

private void button3_Click(object sender, EventArgs e)
{
    string sor;
    int le = 165;
    for(int i = 0; i < 12; i++)
    {
        if (ver[i].DontotFolytathat)
        {
            sor = ver[i].Nev + "; " +
                ver[i].ErvenyesDobasokSzama.ToString() + "; " +
                ver[i].Sikertelen.ToString() + "; " +
                ver[i].LegjobbEredmeny.ToString();
            Label l = new Label();
            l.Parent = Form1.ActiveForm;
            l.Left = 300; l.Top = le;
            l.Width = 500; l.Height = 20;
            l.AutoSize = false;
            l.Text = sor;

```

```

        le += 20;
    }
}

```

A 8. feladatot a **Magyar** nyomógomb oldja meg. A megoldás ugyanaz, mint a konzolalkalmazás **Magyar** függvényénél. Egyetlen eltérés a megjelenítés.

```

private void button4_Click(object sender, EventArgs e)
{
    string sor;
    int i = 0;
    while (ver[i].Orszag != "Magyarország") i++;
    sor = "8. feladat: A magyar versenyző " + ver[i].Nev + "
        " + (++i).ToString() + ". lett!";
    fel8.Text = sor;
}

```

Már csak a Vége gomb utasítása van hátra. Ez egy kicsit más, mint amit eddig alkalmaztunk (a régi megoldás is jó – de ezt is kell ismernetek).

```

private void vege_Click(object sender, EventArgs e)
{
    Application.Exit();
}

```

Most már majdnem mindent megoldottunk. Van egy bökkenő. Konzolalkalmazásnál én (a programozó) mondom meg, hogy milyen sorrendben haladjunk. Megadom a függvények hívásának sorrendjét. Na, ez grafikus felületen nem ilyen egyszerű. A képernyőn megjelenik a sok gomb. Akkor nyomkodhatom tetszésem szerint? Ez nem biztos, hogy célravezető. Szerencsére van lehetőségünk gombokat letiltani. Jelen feladatnál induláskor két gombra kattint-hatok büntetlenül. Az **Adatbeolvasás** és a **Vége** gombra. A többiek zavart okoznának a program működésében, a hibajelzéssel leállni nem szerencsés dolog.

A program indulásakor nem lehet aktív a Döntőt folytat, a Statisztika és a Magyar gomb. Ezért ezeket letiltjuk a Form1 betöltésekor:

```

private void Form1_Load(object sender, EventArgs e)
{
    button2.Enabled = false;
    button3.Enabled = false;
    button4.Enabled = false;
}

```

Csak akkor engedélyezzük, ha már az adatbeolvasás megtörtént, ekkor viszont letiltjuk az **Adatbeolvasás** gombot (ezért kellett volna nevet adni a gomboknak – akkor tudnánk melyik gomb kicsoda). Az **Adatbeolvasás** gomb teljes utasítássora:

```

private void button1_Click(object sender, EventArgs e)
{
    string fnev = "E:\\forras\\olimpia2016.txt";
    StreamReader f = File.OpenText(fnev);
}

```

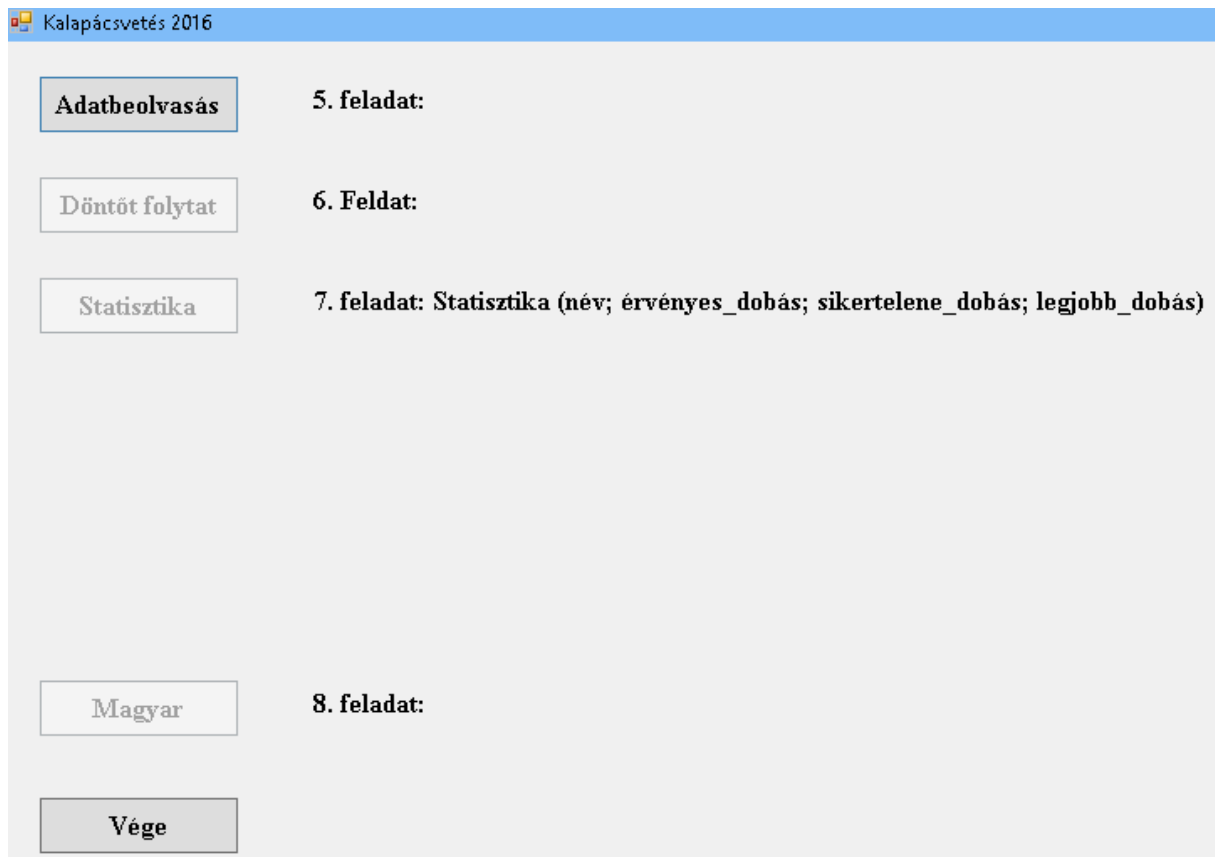


```

int db = 0;
while (!f.EndOfStream)
{
    string sor = f.ReadLine();
    Versenyzo uj = new Versenyzo(sor, db);
    ver[db] = uj;
    db++;
}
f.Close();
fel5.Text = "5. feladat: Döntőbe jutott versenyzők szá-  
ma: " + db.ToString();
button2.Enabled = true;
button3.Enabled = true;
button4.Enabled = true;
button1.Enabled = false;
}

```

Nézzük meg mit követtünk el. Futtassuk a programot:



A végeredmény:

Adatbeolvasás

5. feladat: Döntőbe jutott versenyzők száma: 12

Döntőt folytat

A 3. dobás után 8 versenyző folytathatta a döntőt

Statisztika

7. feladat: Statisztika (név; érvényes\_dobás; sikertelene\_dobás; legjobb\_dobás)

Dilsod Nazarov; 6; 0; 78,68

Ivan Cihan; 5; 1; 77,79

Wojciech Nowicki; 3; 3; 77,73

Diego del Real; 5; 1; 76,05

Marcel Lomnický; 6; 0; 75,97

Amgad Elseify Ashraf; 6; 0; 75,46

Pars Krisztián; 5; 1; 75,28

David Söderberg; 3; 3; 74,61

Magyar

8. feladat: A magyar versenyző Pars Krisztián 7. lett!

Vége