

C# 6 - a Gyakorlat – Windows Form alkalmazások

Ugye emlékszik, hogy az órán meg nem oldott feladatokat otthon kell megoldani?

(A mostaniakat azonban elég csak a zh után, és mivel több hasonló is van köztük, szemezgethet belőlük.)

A 12. feladatot mindenképpen oldja meg! – de csak a ZH UTÁN

1. feladat:

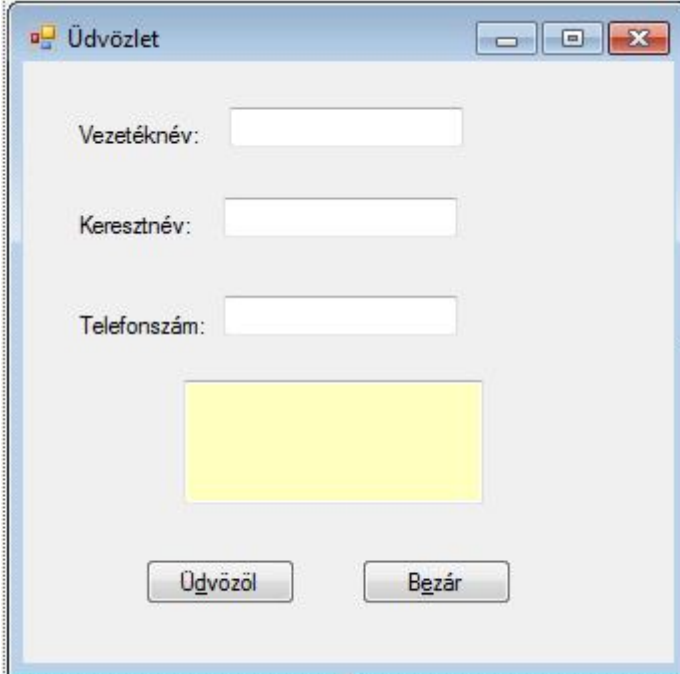
Beszélgjünk meg a *Grafikus_felulet_letrehozasa.pdf* fájl tartalmát.

NAGYON FONTOS: A leírtak alapján otthon oldja meg önállóan. Ez nagyon fontos a továbbiak megértéséhez.

Most a megértésen van a hangsúly, ezért másolható módon elér minden kód-részletet. Ezeket a kódrészleteket az *elso_grafikus_felulet* nevű mappa tartalmazza.

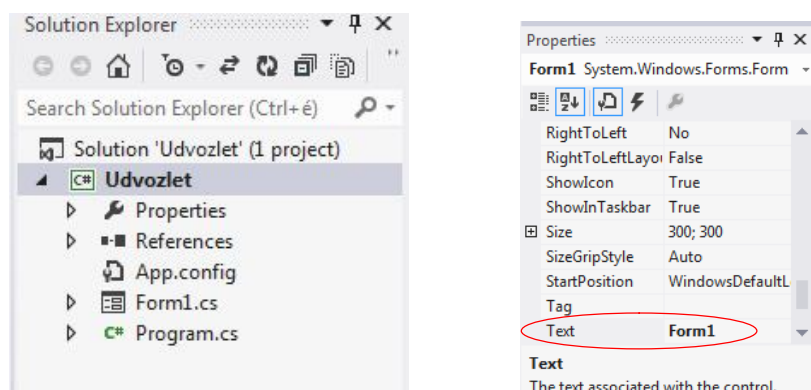
2. feladat

Készítsünk Windows Form alkalmazást, amelyben bekérjük a felhasználó vezeté- és keresztnévét, valamint a telefonszámát. Helyezzünk el egy nyomógombot az űrlapon, amelyre ha rákattint a felhasználó, akkor név szerint üdvözli őt az alkalmazás, és a név alatt a telefonszáma is megjelenik egy többsoros, csak olvasható szövegdobozban. Egy másik nyomógombra kattintva kilépünk az alkalmazásból. Az alkalmazás formaterve a következő:



Segítség a megoldáshoz:

Hozzunk létre egy új projektet, legyen a neve Udvozlet. A *New Project* ablakban most a sablonok közül a legelső, a **Windows Forms Application** sablont válasszuk ki, majd nyomjuk meg az OK gombot. Létrejön az új projekt, amelynek elemeit a Solution Explorerben láthatjuk. A **Form1.cs** fájl tartalmazza a fő űrlap általunk megírandó kódját, tulajdonságait. A Properties ablakban megváltoztathatjuk az ablak tulajdonságait, például a Text tulajdonság legyen *Üdvözet*. A Form Text tulajdonsága az ablak címének megadására szolgál.



A munkaterületen megjelenik egy üres űrlap (ablak), amelyre a képernyő bal szélén látható **ToolBox** –ból helyezhetünk el vezérlő elemeket.

Helyezzük el a következő vezérlőelemeket az űrlapon, és állítsuk be a tulajdonságaik értékét az alábbiakra:

Vezérlőelem Objektum	Tulajdonság	Tulajdonság érték
label1	Text	Vezetéknév
label2	Text	Keresztnév
label3	Text	Telefonszám
textBox1	Name	txtVeznev
textBox2	Name	txtKernev
textBox3	Name	txtTelefon
textBox4	Name	txtUdvozlet
	ReadOnly	True
	MultiLine	True
	TabStop	False
button1	Name	btnUdvozles
	Text	Ü&dvözöl
button2	Name	btnBezaras
	Text	B&ezár

A **txtUdvozlet** szövegdoboz **TabStop** beállítása azt jelenti, hogy nem lehet majd elérni a tabulátor gombbal (nem parkol rajta a kurzor). A nyomógombok text tulajdonságánál megadott & jel az adott betű előtt lehetővé teszi, az ún. „gyors” billentyűk használatát. Az ALT+az & jel mögött lévő karakter lenyomása ugyanazt eredményezi, mint a nyomógombra kattintás. Futtatáskor az Alt gomb megnyomásakor válnak láthatóvá a jelölt karakterek.

Készítsük el a két nyomógombra való kattintás eseménykezelőjét!

Egy vezérlőelem **Properties** ablakában az **Events** eszközgombon kattintva az adott vezérlő által kezelhető eseményeket nézhetjük meg. A kezelni kívánt esemény nevére duplán kattintva létrejön a kezelőmetódus váza a Form1 osztályban és a kurzor a metódus kódjához ugrik. Természetesen nekünk kell megírni a metódus törzsét.



Minden vezérlőhöz tartozik egy általánosan kezelt (default) esemény, ezt az adott vezérlőn duplán kattintva generálja a fejlesztő környezet.

Mivel nyomógomb esetén a gombnyomás (.NET-ben **Click**) esemény a leggyakoribb, ezért most ezt úgy is generálhatjuk, hogy duplán kattintunk a nyomógombon. Ezzel a lehetőséggel élve nem választhatjuk meg az eseménykezelő metódus nevét, azt a vezérlőobjektum nevéből és az esemény nevéből rakja össze a kódgenerátor. (Ha mi akarjuk megadni a nevét, akkor csak egyszerűen gépeljük be az esemény neve mellé a tulajdonságablakban.)

Írjuk meg az eseményhez tartozó kódot.

```
private void btnUdvozles_Click(object sender, EventArgs e) {
    txtUdvozlet.Text = "Üdvözöllek " + txtVezNev.Text + " " + txtKerNev.Text + "!" +
        Environment.NewLine + "Telefon: " + txtTelefon.Text;
}
```

(Új sort a "\n" hozzáadásával is generálhatunk.) Térjünk vissza tervező nézetbe, és készítsük el a második gomb Click eseménykezelőjét is.

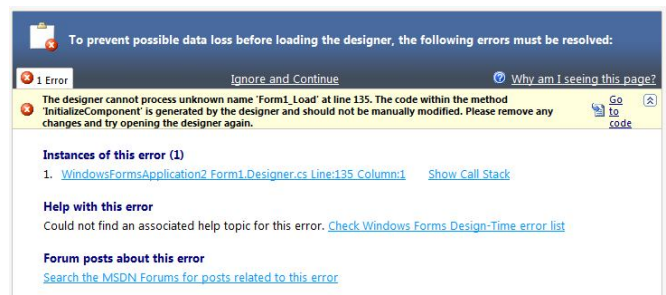
```
private void btnBezaras_Click(object sender, EventArgs e) {
    this.Close();
}
```

Futassuk a programot! Változtassuk meg az Üdvözet szövegdoz háttérszínét tetszőleges színre. (A színek, betűtípusok, stb. változtatásának módjáról a segédletben olvashat.)

FONTOS MEGJEGYZÉS: Ha utólag meggondolja magát, és ki akar **törölni** egy vezérlőhöz tartozó eseménykezelő metódust, akkor azt **a tulajdonságablakban kell** megtennie, **semmiképpen sem** a Form1 osztály kódjában, mert a háttérben futó generált kódból is ki kell törölni az eseményfigyelésre vonatkozó információt. (Properties / events – és kitörli a fölösleges eseményt.)

Ha netalántán mégis belefutott ebbe a hibába, és hamarabb törölte ki az eseménykezeléshez tartozó metódust, akkor a design felületen ilyen hibajelenség fogadja:

Ekkor már csak az a megoldás adódik, hogy nyissa meg a Form1.Designer.cs fájlt, és ott „gyalog” törölje ki a kifogásolt hivatkozást.

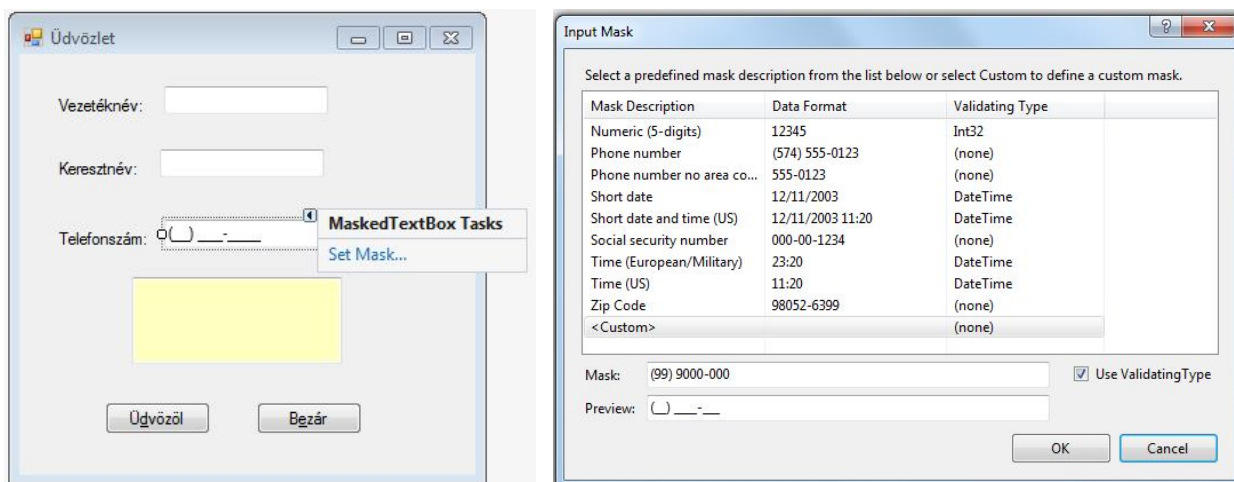


3. feladat

Módosítsuk a programot úgy, hogy csak akkor jelenjen meg az üdvözlő szöveg, ha a felhasználó minden mezőt kitölt. Figyelmeztessük egy felbukkanó üzenet (**MessageBox**) ablakban a mulasztására. Az üdvözlő kiírása után töröljük a bevitt adatokat az adatbeviteli mezőkből. A telefonszámot tartalmazó szövegdobozt cseréljük ki maskedTextBox-ra, ami lehetővé teszi, hogy előre megformázott módon adjunk meg bevitt adatokat.

Segítség a megoldáshoz:

A maszk formátumleíró nyelve a VB6-beli MaskEd vezérlőelem, illetve az Access maszknyelvéhez hasonlít. Rendelkezésre áll maszkszerkesztő (Mask Editor) is, amely hasznos maszkok listáját tartalmazza. Ez a szerkesztő azt is lehetővé teszi, hogy a fejlesztő egyedi maszkokat hozzon létre, és ki is próbálja őket. Futásidőben a MaskedTextBox vezérlőelem eseményt vált ki, ha a szövegmezőbe érvénytelen karakter kerül. A Mask Editor vagy a vezérlőelem **Mask** tulajdonságán keresztül érhető el, vagy a képen látható módon a vezérlőelem jobb felső sarkában található smartTag-ra kattintással megjelenített felbukkanó menüből. (Set Mask...)



A megadott maszkkal azt érjük el, hogy csak számokat gépelhetünk a szövegdobozba. (A 9-cel jelölt hely opcionális, a 0-val jelölt helyre számot vár.)

A hibaüzenet kiírásához a **MessageBox** osztály **Show** metódusára van szükségünk. Ennek legegyszerűbb paraméterezési módja, ha a kiírandó szöveget adjuk meg paraméterként.

Formája:

```
MessageBox.Show("szöveg", "üzenetdoboz_címe");
```

Szövegdoboz tartalmának törlése:

```
textBox1.Clear();      metódushívással vagy
textBox1.Text="";      Text tulajdonság értékének megadásával
```

4. feladat

Házi feladatként módosítsa tovább az előző feladatot.

1. Az adatmezők kiürítése után még azt is oldja meg, hogy a vezetéknév bevitelén villogjon a kurzor.

2. A felhasználó számára kicsit kényelmetlen, hogy az adatok begépelése után meg kell fognia az egeret, és rákattintani a gombra. (De az is, ha tabulátorokkal vándorol a gombra.) Ezért célszerű lenne figyelni az enter lenyomását is. Oldja meg, hogy az enter megnyomására írja ki az üdvözlő szöveget (vagy a hibajelzést, ha nincsenek kitöltve az adatok).

Megoldhatja úgy is, hogy csak a telefonszám megadása után figyel az entert, de úgy is, hogy minden egyes adat begépelése után.

Az ügyetlenebb felhasználók kedvéért a gombnyomás eseményt is hagyja meg. Arra figyeljen, hogy NE kódismétléssel oldja meg a feladatot!

3. Talán észrevette, hogy a maszkolás ellenére elfogadja, ha a számok helyett szóközt írunk, vagy egyáltalán nem írunk semmit. Így félkész telefonszámokat is elfogad. Oldja meg, hogy csak a teljes telefonszámot tekintse valós adatnak. (Vagyis: a körzetszám elmaradhat, és a telefonszám lehet hatjegyű is és hétjegyű is.)

4. Ha nagyon ügyes és türelmes, akkor oldja meg, hogy valóban csak helyes telefonszámokat fogadjon el, vagyis kötelező legyen körzetszámot írni, ha az csak egyjegyű, akkor a telefonszámnak kell hétjegyűnek lennie, ha kétjegyű, akkor a telefonszám hatjegyű. (Első eset a budapesti számok, második a vidékiek.)

Segítség a megoldáshoz:

A **Focus()** metódus ráállítja a fókuszt az adott vezérlőelemre.

Az enter megnyomásának figyelése:

KeyDown esemény

Az enter figyelése:

```
if (e.KeyCode == Keys.Enter)
```

ahol e a metódus paraméterében szereplő EventArgs típusú változó.

A teljes maszkolás figyelése: a maszkolt textbox **MaskCompleted** tulajdonsága.

5. feladat

A következő feladat a string-műveletek gyakorlására szolgál, ezért engedünk meg most egy-két helyesírási hibát.

Egy szövegdobozba kérjünk be egy nevet. Az **Üdvözlés** gomb megnyomására határozzuk meg belőle a *vezetéknévet* és a *keresztnevet* - feltesszük, hogy egy szóköz választja el őket -, majd írjuk ki egymás alá egy-egy csak olvasható szövegmezőbe a **vezetéknévet csupa nagybetűvel**, a **keresztnevet csupa kisbetűvel**, valamint alájuk a **teljes név hosszát** is!

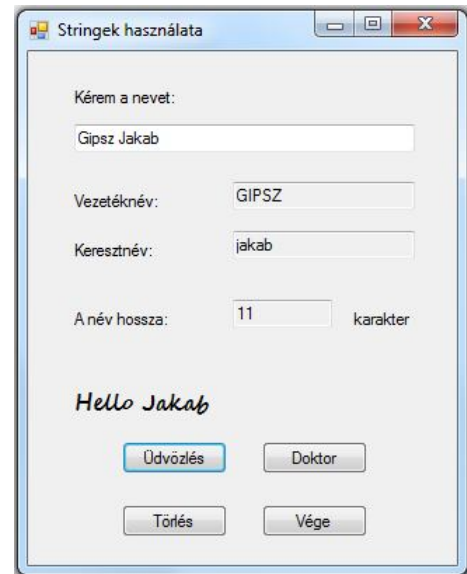
Legalulra - csak a keresztnévet felhasználva - írjunk ki egy címkébe egy **üdvözlést**! A címke (Label) betűtípusa (Font tulajdonság) legyen Segoe Script, félkövér, 12-es betűméret.

A **Doktor** gomb megnyomására a név elejére **szúrjuk be a Dr.** rövidítést!

A **Törlés** gomb megnyomására töröljünk le minden mezőt!

A **Vége** gomb az ablak bezárására szolgál.

A form fejlécébe írjuk be: *Stringek használata*



Segítség a megoldáshoz:

Egy karakter pozíciójának meghatározása a stringben: **IndexOf(karakter)** metódussal

Egy string hosszát a **Length** tulajdonsága adja meg.

Rész-string képzése: a **Substring(kezdőpozíció, hossz)** vagy a **Substring(kezdőpozíció)** metódussal – az utóbbi a kezdőpozíciótól a string végéig adja meg a rész-string-et.

Nagybetűssé alakítás **ToUpper()** metódussal

Kisbetűssé alakítás **ToLower()** metódussal

Stringbe beszúrás **Insert(kezdőpozíció, string)** metódussal

További megjegyzések:

1. Próbálja ki, hogy a címkefeliratokat nem a tulajdonságok ablakban állítja be, hanem kódból, a form **Form_Load** eseményének hatására, mégpedig `label1.Text = "Kérem a nevet:"`; stb. módon.

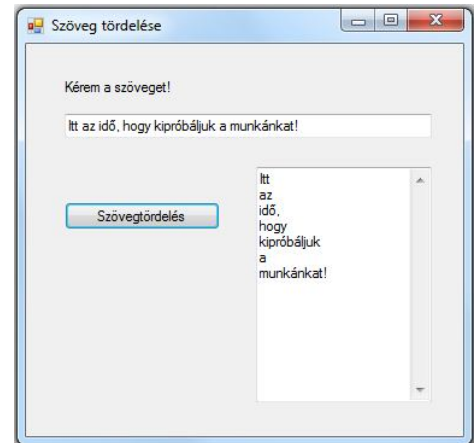
2. A feladat megoldása során használja helyesen a Dr. jelzöt, vagyis a Dr. és a név között legyen egy szóköz. Oldja meg, hogy ebben az esetben is külön tudja választani a vezetéknévet (Dr. nélkül) és a keresztnévet!

6. feladat

Készítsen programot, amelyben egy szövegdobozban megadunk egy karaktersorozatot (mondatot), és a Szövegtörölés nyomógomb megnyomására egy több soros szövegdobozban szavakra (elemekre) tördelve mutatjuk meg az eredeti szöveget. Minden szó új sorba kerüljön.

A feladat megoldásához használja a String osztály Split() metódusát. A metódus többféle módon is paraméterezhető, a legegyszerűbb, ha az elválasztó karaktert adjuk meg paraméterként:

```
private string szoveg;  
private char elvalaszto;  
private string[] tordeltSzoveg;  
tordeltSzoveg = szoveg.Split(elvalaszto);
```



További megjegyzések:

1. Oldja meg, hogy a szöveg végén nyomott enter hatására is hajtsa végre a szövegtördelést.
2. Oldja meg, hogy a tördelésre használt textBox felületen csak akkor jelenjen meg a scrollbar, ha a beírt sorok száma ezt megkívánja. (Pl. legalább 10 sort gépelünk – de ne égesse be a 10-t!)
3. Próbálja ki, hogy a tördelésre szánt textBox helyett richTextBox-ot használ. (Ekkor automatikusan csak akkor jelenik meg a scrollbar, ha szükség van rá.)

7. feladat

Számolás numerikus alpműveletekkel
Készítsük el az alábbi formot!

A form **fejlécébe** írjuk be: *Aritmetikai műveletek*

Az **Összeadás**, **Kivonás**, **Szorzás**, **Osztas**, **Maradék** gombokra kattintva a megfelelő műveletet elvégezzük az Operandus1 és Operandus2 szövegmezőkben levő adatokat felhasználva (a baloldali operandusa az Osztas és Maradék műveletnek az Operandus1 legyen!), és az eredményt kiírjuk a **csak olvasható Eredmény** szövegmezőbe.

A **Töröl** gomb az összes szövegmező tartalmát törli.

A **Vége** gomb az ablak bezárására szolgál.

- A feladatot **egész (int)** típusú adatokra készítsük el! Teszteljük úgy is az Osztas és Maradék műveleteket, hogy az Operandus2 (azaz az osztó) értéke 0 ! **Jegyezzük meg, mi történt!**
- Módosítsuk a programot úgy, hogy **valós (double)** típusú adatokkal dolgozhassunk! FIGYELEM! Valós típusú adat bevitelénél tizedesvessző van, nem tizedespont! Teszteljük úgy is az Osztas és Maradék műveleteket, hogy az Operandus2 (azaz az osztó) értéke 0 !

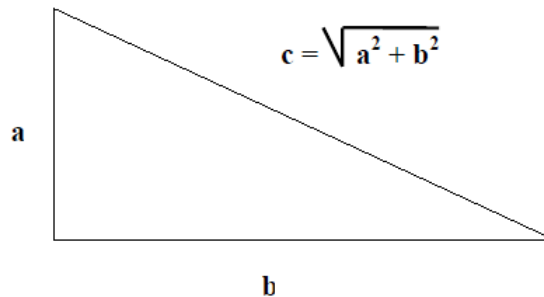
Ugyanúgy működött valós adatok esetén a program, mint egész típusú adatokra?

Megjegyzés: A ToString() metódus paraméterezésével a kiírandó double érték tizedes jegyeinek számát is megadhatjuk:

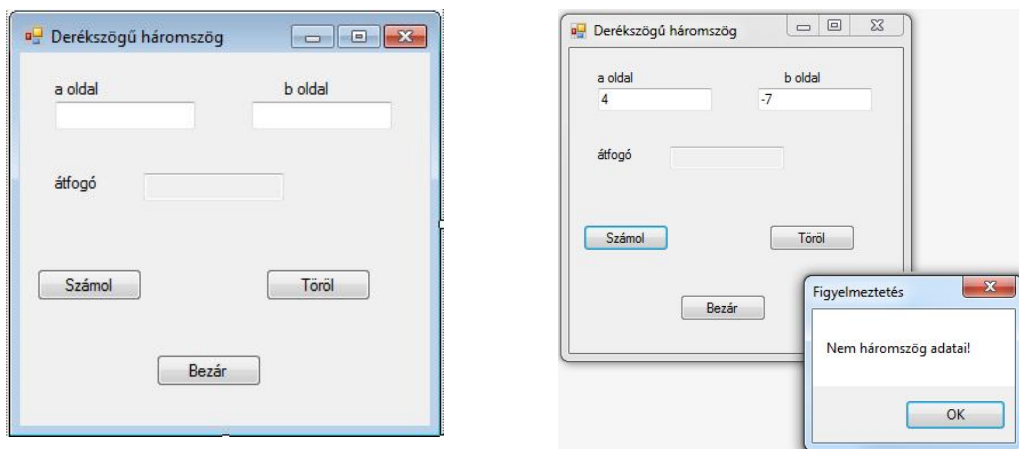
ToString("F") két tizedes
ToString("F5") 5 db tizedesjegy (általában „Fn” – n db tizedesjegy)
ToString() 15 számjegy.

8. feladat

Egy derékszögű háromszög két (valós) befogóját bekérve az átfogó kiszámítása.



Készítsük el az alábbi szerkezetű formot:



A form fejlécébe írjuk be: *Derékszögű háromszög*

Egy-egy szövegmezőbe bekérjük a két befogót.

A **Számol** gomb megnyomására helyes adatok esetén meghatározzuk az átfogót, és kiírjuk a csak olvasható harmadik szövegmezőbe.

Logikailag hibás (pl. negatív szám, vagy 0), vagy hiányzó adat esetén adjunk hibajelzést (MessageBox.Show(szöveg)).

A **Töröl** gomb az összes szövegmező tartalmát törli.

A **Bezár** gomb az ablak bezárására szolgál.

9. feladat

Két valós számot (X és Y) bekérve írjuk ki a **számtani**, **mértani** és **harmonikus** közepüket!

Számtani közép: $\frac{X + Y}{2}$

Mértani közép: $\sqrt{X \cdot Y}$

Harmonikus közép: $\frac{1}{\frac{1}{X} + \frac{1}{Y}}$

Készítsük el az alábbi szerkezetű formot:

A form fejlécébe írjuk be: Közepék

Egy-egy szövegmezőbe bekérjük a két számot.

A **Számol** gomb megnyomására helyes adatok esetén meghatározzuk a közepeket, és kiírjuk a csak olvasható szövegmezőkbe **5 tizedes** pontossággal.

Hiányzó, vagy formai hibás adat esetén adjunk hibajelzést.

Amelyik közép nem számolható, annak textbox mezőjébe írjuk be “Nem számolható”, a többi közép értékét határozzuk meg, és írjuk ki!

A **Töröl** gomb az összes szövegmező tartalmát törli.

A **Bezár** gomb az ablak bezárására szolgál.

Ajánlott tesztadatok (pl. ezekkel az adatokkal nem számolható a jelzett közép):

X	Y	nem számolható közép
0	bármely szám	harmonikus
1	-1	mértani, harmonikus
5	-4	mértani

10. feladat

Készítsen számkitaláló programot! A gép „gondol” egy számot pl. 1 és 100 között, és a felhasználónak kell tippeléssel kitalálni az értéket. Minden tipp után, a program megmondja, hogy a tipp kisebb, vagy a tipp nagyobb, mint a gondolt szám, s közben számolja a tippeket. Ha eltalálta a felhasználó a számot, akkor „Gratulálok, eltaláltad!”, üzenetet írja ki. Az **ablak címsorában** jelenjen meg a játék neve, és az is, hogy **hányadik játékot** játszunk. Az **Értékel** gomb megnyomásakor értékeli ki a gép a tippet. Az **Új játék** gomb megnyomásakor új játékot kezdünk.

Használjon véletlenszám-generátort a kitalálandó szám létrehozására.

**11. feladat**

Írjon programot a következőkre:

A form betöltésekor jelenjen meg a felső szöveg-dobozban egy aláhúzásokból álló véletlen hosszúságú szöveg.

A Tipp gomb hatására cseréljük ki a szöveg adott sorszámú elemét a megadott betűre.

A jobb olvashatóság kedvéért az eredeti szöveg minden egyes „éles” karaktere után legyen egy szóköz, amelyet nem cserélünk.

Segítség: Az új szöveg az eddigi szöveg eleje + a megadott betű (string formában) + az eddigi szöveg vége.

Továbbfejlesztések:

a/ Csak akkor engedjen cserélni, ha még aláhúzás szerepel az adott helyen, egyébként adjon hibaüzenetet.

b/ A szövegdoboz egy (egyelőre konstansként) megadott kitalálendő szó betűinek helyét jelezze, a Tipp gomb hatására csak akkor cserélődjön ki az aláhúzás a megadott betűre, ha a tippelő eltalálta a mögötte lévő betűt.

c/ Rakjon fel egy „Megfejtés” feliratú gombot is, amelyet megnyomva a szövegdobozban megláthatjuk a kitalálendő szót.

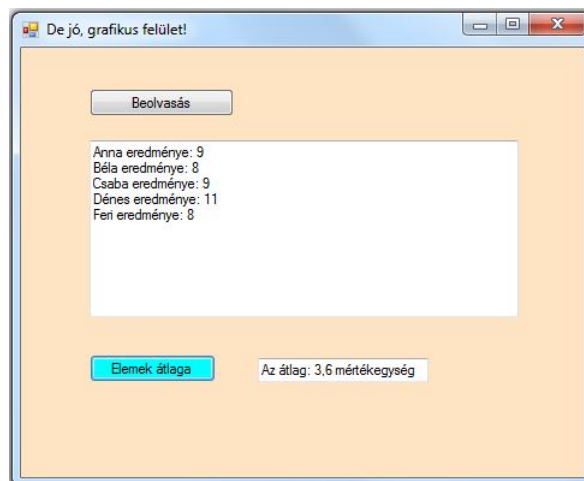
d/ Az induló szót véletlenül generálja valahány előre megadott szó közül. (A szavakat tartalmazó tömb véletlen sorszámú eleme legyen a kitalálendő szó.) A szavakat adatfájlból olvassa be.

e/ Továbbfejlesztheti egy akasztófa játékká.

12. feladat

Vegye elő egy korábban megoldott feladatát (lehet az, amelyikkel a zh-ra készült vagy akár ismét megoldhatja a zh-feladatot is).

Oldja meg, hogy a feladat eredményének néhány részlete grafikus felületen jelenjen meg. Például valami ehhez hasonló módon:



Segítség:

Hozzon létre egy új windows form alkalmazást. A projekthez adja hozzá a korábbi projekt alapsztályait (ugyanúgy, ahogy új osztályt hoz létre, csak Add existing item...)

Arra figyeljen, hogy azonos legyen a névtér.

A vezérlés megfelelő metódusait pedig másolja át az egyes eseményekhez.

Pl:

```
private void btnOlvasas_Click(object sender, EventArgs e) {  
    AdatBevitel();  
    Kiir();  
}
```

és mögé másolja az AdatBevitel(), illetve Kiir() metódust (természetesen a szükséges deklarációkkal együtt).

Az adatbevitelen semmit sem kell változtatni (persze az adatfájlt be kell másolni a megfelelő helyre), a kiíratást pedig értelemszerűen át kell alakítani, hiszen nem konzolos felületre akarunk írni.

A többsoros textbox-ot az AppendText() metódussal is fel lehet tölteni, csak persze figyeljen a soremelésre ("n").

Hasonlóan tudja megoldani a többi funkciót is.

Ha még játékos kedve is van, akkor megoldhatja azt is, hogy a gomb fölé mozgatva az egeret változzon meg a gomb színe, kilépve onnan pedig változzon vissza az eredeti színre.