ZÁRÓDOLGOZAT

Orosz Balázs

2024

Magyar Agrár- és Élettudományi Egyetem

Károly Róbert Campus

Műszaki Intézet

felsőoktatási szakképzés

Egy gépírást fejlesztő program bemutatása

Belső konzulens: Dr. Novák Tamás
egyetemi docens
Alkalmazott Informatika Tanszék

Készítette: Orosz Balázs

2024

Tartalomjegyzék

1. Bevezetés

A programozás hosszú évek óta az életem szerves része. Középiskolai tanulmányaimat informatikai szakon kezdtem el, az érettségi után, pedig elvégeztem a gazdasági informatikus képzést. Folytatni akartam tovább a tanulást, csak nem tudtam melyik egyetemen, így választottam a gyöngyösi egyetemet. Nézegettem, hogy melyik szak lenne megfelelő nekem, ezért választottam a Programtervező informatikus szakot. A szakot 2022-ben kezdtem el, mivel ez egy FOSZK képzés, ezért egy csak két éves, azaz 4 féléves a szak. Szerettem volna az akkori tudásomat tovább fejleszteni. Rengeteg mindent tanultam az egyetemi évek alatt. Jobban belemélyedtem a programozásba. Sok új dolgot tudtam meg a C# belül, nem is gondoltam volna, hogy tudnak még új felhasználási módot mutatni. Több programot is használtunk. mint például a PowerShell-t, BASH-t és persze a hálózat sem maradhat ki a Cisco-val. A munka világában elég széleskörű a választék. Én egy játék fejlesztő cégnél vagy egy ilyen játékokkal foglalkozó cégnél szívesen dolgoznék, akár fejlesztőként vagy tesztelőként. Későbbiekben szeretnék foglalkozni programozással, nem feltétlen programozó szeretnék lenni, de szeretném, ha továbbra is az életem része maradna.

Záródolgozatomban egy általam készített programot mutatok be. A program egy gépírást fejlesztő játék, amelyet C# programozási nyelvben írtam. A dolgozatom során elemzem a C# funkcióit, melyek kiemelkedő fontosságúak voltak, illetve a programozás során felmerülő problémákat. A program működésének részletezése során, a játék belső mechanizmusa mellett kitérek az adatbázis összetételére, a játéktér változására is. Végül a dolgozatomat a fejlesztési lehetőségek taglalásával zárom.

2. Programozási nyelv választása

Egy játék megalkotásához számos programozás nyelv közül lehet választani, például Python, Unix vagy a C#. Az általam preferált opció a Visual Studio által fejlesztett C# programozási nyelv. A C# 2019-es verziójában dolgoztam. Általánosan elmondható róla, hogy magasszintű programozási nyelvként használata könnyebb és gyorsabban tanulható, emiatt logikus választás egy kezdő vagy fejlődni akaró programozónak. Ezen felül a C# sokoldalúságának köszönhetően használható Windows-on vagy éppen Mac-en. Emiatt a platformfüggetlenség miatt az általam készített játék a felhasználó számára legkedvezőbb felületen használható. A következőkben kiemelném a C# néhány további fontos jellemzőét, amelyek hasznomra váltak a program készítése során.

-A C# egyszerűen és gyorsan dolgoz fel nagy mennyiségű adatot, ami egy gépíróprogram hatékonyságában elengedhetetlen, épp úgy, mint a válaszidő feldolgozásakor.

-A C# a Visual Studio részeként rengeteg lehetőséggel rendelkezik. A rendelkezésre álló, külön a C# részére optimalizált fejlesztési környezet megkönnyíti a hibakeresést, sőt további bővítmények integrálását is lehetővé teszi.

-Kezdő programozóknak könnyen tanulható nyelvek

- JavascriptPython
- JAVA
- C / C++
- PHP

Az összes nyelvet könnyen és egyszerűen el lehet sajátítani, fontos a kitartás és a szorgalom hozzá. Én a C# console változattal kezdtem egy évig tanultam, utána kezdtem el a grafikus programozást. Itt az egyetemen tanultam Javascript-et, Python-t és PHP is. Mind a három programozási nyelvből kaptam ízelítőt. Eléggé hasonlítanak egymásra mégis másképp kell őket használni.

3. Adatbázis összetétele

A programok írása során beágyazott adatbázist készítettem és használtam fel. Ezen játék esetén az adatbázis elemei konkrétan a játék elemeivé váltak. Az adatok a https://szotar.com/szokereso/hossz/4-betus és a https://szavak.info/4-betus-szavak/ weboldalról származnak. A weboldalakon talált szavaknak egyik fontos kritériuma volt, hogy a játék nyelvén, vagyis magyarul jelenjenek meg. Ezen kívül a játék beállításainak megfelelően 3, 4, 5 vagy 6 betűs szavakra volt szükségem. Az ilyen hosszúságú szavak alkalmasak arra, hogy könnyen megismerhető legyen a billentyűzetet, majd idővel elsajátíthatóvá váljon a tízujjas vakírás. Az elemek véletlenszerűen lettek kiválasztva, viszont fontos szempont volt, hogy a lehető legtöbb variációval kerüljenek ide szavak. Az adatbázisban összesen 2220db szó szerepel, amelyből 1523db 4 betűből álló szó. A játékot a betűszámok, a szavak hossza osztja nehézségi szintekre. A 4 betűből álló szavak jelenítik meg a normál, átlagos szintet, így ez válik a legkeresettebbé, és a legtöbb szót igénylő szintté.

4. A program működése

4.1 Lista:

Egy tömb kategóriájú kiterjesztett elem, aminek nincs meghatározva a nagysága, így bármilyen hosszú lehet. A típusa nincs meghatározva, így bármiként lehet deklarálni.

A következő módon lehet deklarálni, hogy string típusú legyen:

List<string> lista = new List<string>();

A sor végén szereplő dupla zárójel egy konstruktor, egy speciális metódus, amely az új létrehozott objektumot alaphelyzetbe állítja be.

Hozzáadni a listához a következő paranccsal tudunk:

lista.Add();

Létrehoztam 4 listát, „**Lista**”, „**Lista2**”, „**Lista3**” és „**Lista4**” néven. Ezekben a listákban kerültek a különböző hosszúságú szavak. A lista a hosszúságuk szerint lett elnevezve, illetve létrehozva.

1. ábra Listák Létrehozása

```
List<string> Lista = new List<string>();
List<string> Lista2 = new List<string>();
List<string> Lista3 = new List<string>();
List<string> Lista4 = new List<string>();
```

(Forrás: Saját szerkesztés)

Ezután létre hoztam a „**beolvasás1**”, „**beolvasás2**” és „**beolvasás3**” nevű változókat amit, a „**StreamReader**” segíti a hogy be lehessen olvasni a „**szavak1.txt**”, „**szavak2.txt**”, „**szavak3.txt**” illetve „**szavak4.txt**” fájlokat amikbe 3, 4, 5, és 6 betűs szavakat tárolok. Csináltam egy string típusú „**sor**” nevű változót. A „**while**” a beolvasást segíti, hogy amíg végig nem ér a txt fájlon addig olvassa be az adatokat. A „**sor**” nevű változóba beillesztettem a szavakat, utána pedig hozzáadtam a hozzá tartozó Listához. A „**while**” utána pedig lekell zárni a beolvasást a „**beolvasás.Close()**”illetve a hozzájuk tartozó metódussal.

4.2 Ciklus utasítás:

A programomban az egyik legtöbbször előforduló elem a ciklus utasítás. Ezt az elemet akkor használjuk, amikor egy számítást vagy kiválasztást meg szeretnénk ismételni. 2 fajtáját emelném ki, amik közül én csak az egyiket használtam: „for” és „while”.

2. ábra Lista beolvasása

```
System.IO.StreamReader beolvasás1 = new System.IO.StreamReader("szavak1.txt");
string sor;
while (!beolvasás1.EndOfStream)
{
    sor = beolvasás1.ReadLine();
    Lista.Add(sor);
}
beolvasás1.Close();
//
System.IO.StreamReader beolvasás2 = new System.IO.StreamReader("szavak2.txt");
string sor2;
while (!beolvasás2.EndOfStream)
{
    sor2 = beolvasás2.ReadLine();
    Lista2.Add(sor2);
}
beolvasás2.Close();
//
System.IO.StreamReader beolvasás3 = new System.IO.StreamReader("szavak3.txt");
string sor3;
while (!beolvasás3.EndOfStream)
{
    sor3 = beolvasás3.ReadLine();
    Lista3.Add(sor3);
}
beolvasás3.Close();
//
System.IO.StreamReader beolvasás4 = new System.IO.StreamReader("szavak4.txt");
string sor4;
while (!beolvasás4.EndOfStream)
{
    sor4 = beolvasás4.ReadLine();
    Lista4.Add(sor4);
}
beolvasás4.Close();
```

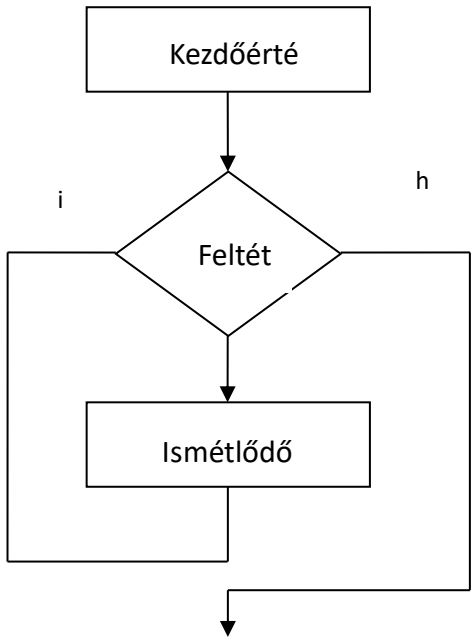
(Forrás: Saját szerkesztés)

while (!beolvasás1.EndOfStream){

sor = beolvasás1.ReadLine();

Lista.Add(sor);

A while ciklus utasítás formája: while (kifejezés) utasítás;



Ezután létrehoztam a szavak tároló Labeleket „**elsőszó**”, „**másodikszó**” és „**harmadikszó**” néven,

3. ábra Szó változók létrehozása

```
Label elsőszó = new Label();
Label másodikszó = new Label();
Label harmadikszó = new Label();
```

(Forrás: Saját szerkesztés)

Miután megalkottam a szó változókat elhelyeztem őket. A labeleknek beállítottam a magasságukat, hogy a szavak rendesen látszódjanak bennük. A „**Font**” tulajdonságai közé tartozik, hogy:

-Lehet állítani a betűnek a típusát

-Lehet állítani a betűknek a nagyságát

-Lehet állítani, hogy dőlt, félkövér vagy aláhúzott legyen

A „**Backcolor = Color.Transparent**” a kép hátterét változtatja meg, hogy a felesleges tartalmakat ne lássuk és csak a képen legyen a fókusz.

A „**this.Controls.Add**” adja hozzá a „**form**”-hoz hogy láthassuk futás idő közben.

4. ábra Szavak elhelyezkedése

```
elsőszó.Location = new Point(400, 80);
elsőszó.Font = new Font(FontFamily.GenericSansSerif, 16F, FontStyle.Italic);
elsőszó.BackColor = Color.Transparent;
elsőszó.Height = 40;
this.Controls.Add(elsőszó);
másodikszó.Location = new Point(600, 80);
másodikszó.Font = new Font(FontFamily.GenericSansSerif, 16F, FontStyle.Italic);
másodikszó.BackColor = Color.Transparent;
másodikszó.Height = 40;
this.Controls.Add(másodikszó);
harmadikszó.Location = new Point(800, 80);
harmadikszó.Font = new Font(FontFamily.GenericSansSerif, 16F, FontStyle.Italic);
harmadikszó.BackColor = Color.Transparent;
harmadikszó.Height = 40;
this.Controls.Add(harmadikszó);
kereső.Location = new Point(565, 600);
kereső.Font = alap;
this.Controls.Add(kereső);
```

(Forrás: Saját szerkesztés)

Létrehoztam 1 szövegdobozt „**kereső**” néven, ami a szavak beírásában fog segíteni.

5. ábra Kereső ablak

```
TextBox kereső = new TextBox();
```

(Forrás: Saját szerkesztés)

Csináltam egy „kereső.TextChanged” esemény, ami akkor következi be ha megváltozik a szövegdoboz avagy elkezdünk bele írni.

6. ábra Kereső változás

```
kereső.TextChanged += (sender1, ex) => this.Keresés();
```

(Forrás: Saját szerkesztés)

Amint a 4. képen is látható alul a „**kereső**” is el kell helyezni és hozzá is kell adni a „**form**”-hoz. Csináltam „**Font**” típusú változókat, „**alap**”, „**medium**”, és „**Medium**” néven. Mindegyik tartalmaz egy betű típust és egy betű nagyságot.

7. ábra Betű stílusok

```
Font medium = new Font("Arial", 18);
Font Medium1 = new Font("Arial", 30);
Font alap = new Font("Arial", 20);
```

(Forrás: Saját szerkesztés)

Miután megcsináltam a „**alap**” nevű változót, hozzá adtam a „**kereső**” nevű szövegdobozhoz.

8. ábra Kereső hozzáadása a programhoz

```
kereső.Location = new Point(565, 600);
kereső.Font = alap;
this.Controls.Add(kereső);
```

(Forrás: Saját szerkesztés)

Minden ilyen jellegű programban van „**hiba**”, illetve „pont” számláló így jártam el én is, a „**számlálónak**” elnevezett pont számolásának hoztam létre. A „**hiba**” nevű változót, pedig az esetleges mellé ütések vagy hibák számlálására hoztam létre.

9. ábra Pontok és Hibák

```
Label számláló = new Label();
Label hiba = new Label();
```

(Forrás: Saját szerkesztés)

A „**számláló**” Labelnek az értékét az „**összeg**” nevű változóban tároltam. A kiíratásához szükséges volt használni a „**ToString()**” metódust hiszen anélkül csak egy „**int**” típusú változó lenne. Az elhelyezés pedig a 85, 250 kordinátára raktam. A szélességét a szavak hosszúságára raktam, hiszen egy 6 betűs szó hosszabb, mint egy 3 vagy 4 betűs, ezért a magasságot 30, a szélességet pedig 150 állítottam be. A számlálóhoz hozzá rendeltem a „**medium, Font**” változót. A „**this.Controls.Add**” paranccsal pedig hozzá adtam a „**form**” -hoz, hogy látható lehessen.

10. ábra Pontok és Hibák helyezkedése

```
számláló.Text = "Pont: " + összeg.ToString();
számláló.Location = new Point(85, 250);
számláló.Width = 150;
számláló.Height = 30;
számláló.BackColor = Color.Transparent;
számláló.Font = medium;
this.Controls.Add(számláló);
hiba.Text = "Hiba: " + Hiba.ToString();
hiba.Location = new Point(85, 280);
hiba.Width = 150;
hiba.Height = 30;
hiba.BackColor = Color.Transparent;
hiba.Font = medium;
this.Controls.Add(hiba);
```

(Forrás: Saját szerkesztés)

Létrehoztam egy „**időzítő**” nevű labelt, aminek a hátterét egyesítettem az alap háttérrel.

11. ábra Időzítő

```
időzítő1.Location = new Point(50, 80);
időzítő1.BackColor = Color.Transparent;
```

(Forrás: Saját szerkesztés)

Készítettem egy „**Timer1_Tick**” eseményt, amivel az idő korlátot adtam meg. Az „**időzítő**” labelhez hozzá rendeltem a „**Medium**” nevű „**Font**” változót. A magasságát beállítottam 100-ra, hogy sokkal láthatóbb legyen. A „**időzítő.Visible = true**” parancs a láthatóságát engedélyezi.

A „**sec**” nevű „**int**” változó a másodperc számítására fog szolgálni. A „**min**” nevű „**int**” változó pedig a perc számítására.

12. ábra Perc és Másodperc

```
public int sec = 00;
public int min = 00;
```

(Forrás: Saját szerkesztés)

A „**sec--**” arra való, hogy lefelé számoljon. Itt fog kezdődni a vizsgálat, hogy ellenőrzi, hogy a „**min**” változó egyenlő-e nullával és hogy a „**sec**” nevű változó is egyenlő-e nullával. Ha igen akkor a „**timer**” megáll a „**timer1.Stop()**” metódussal. A „**kereső.ReadOnly = true**” pedig arra szolgál, hogy mivel már lejárt az időnk akkor már ne lehessen írni a szövegdobozba. A „**MessageBox.Show**” előhoz egy felugró ablakot, ami tartalmazni fogja, hogy mennyi pontot értünk el és hogy hány hibát ejtettünk.

A következő vizsgálatban a „**sec**” változót ismét ellenőrizzük, hogy egyenlő-e nullával és hogy a „**min**” változó nem egyenlő-e nullával. Ha a feltétel igaz akkor a perc változóból levon 1-et és a másodperc változónak az értékét 59-re átírja.

13. ábra Időzítő beállítása és kinézese

```
private void Timer1_Tick(object sender, EventArgs e)
{
    időzítő1.Font = Medium1;
    időzítő1.Height = 100;
    időzítő1.Visible = true;
    sec--;
    if (min == 0 && sec == 0)
    {
        timer1.Stop();
        kereső.ReadOnly = true; //ha lejár az idő letiltja a textboxot
        MessageBox.Show("Lejárt az idő" + "\n" + "Összesen gyüjtött pontod: " + összeg.ToString() + "\n" + "Összesen szerzett hiba: " + Hiba.ToString());
    }
    if (sec == 0 && min != 0)
    {
        min--;
        sec = 59;
    }
    if (sec > 9)
    {
        időzítő1.Text = Convert.ToString(min + ":" + sec);
    }
    else
    {
        időzítő1.Text = Convert.ToString(min + ":0" + sec);
    }
}
```

(Forrás: Saját szerkesztés)

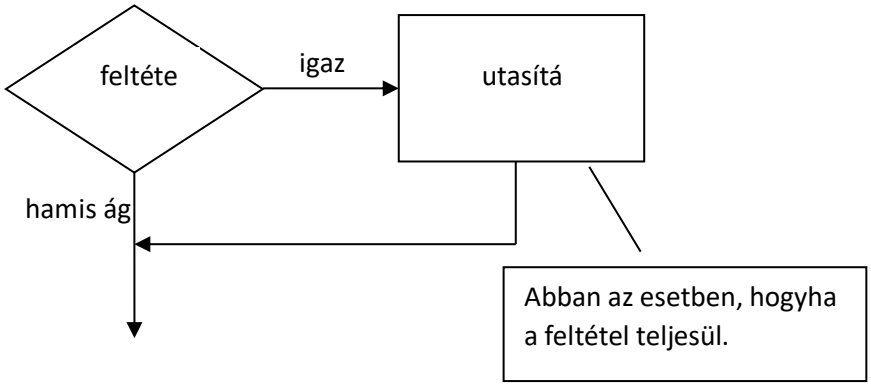
4.2.1 „Ha” függvény:

Programozás közben gyakran használunk olyan parancsot, aminek meg kell felelnie egy adott feltételnek. A programban számos alkalommal előfordul a „Ha” függvény használata. Itt történik a szónak a felismerése, eldöntése, hogy az elsőszámú szóval egyezik-e a begépelt szöveg.

```
if (kereső.Text == elsőszó.Text) {
        összeg++;
        Lista.Remove(elsőszó.Text);
        elsőszó.Text = Lista[rd];
        kereső.Text = "";
        ágyúbal();
        timer2.Start();
        boomtf1 = true;
}
```

Igaz feltétel esetén a program a megadott utasításokat teljesíti. Amennyiben a feltétel hamis értékű a programnak nem kell végrehajtania az utasításokat. A program további működése ettől függetlenül zavartalanul folytatódik.

```
else if (kereső.Text == másodikszó.Text) {
        összeg++;
        Lista.Remove(másodikszó.Text);
        másodikszó.Text = Lista[rd];
        kereső.Text = "";
        ágyúközép();
        timer2.Start();
        boomtf2 = true;
}
```

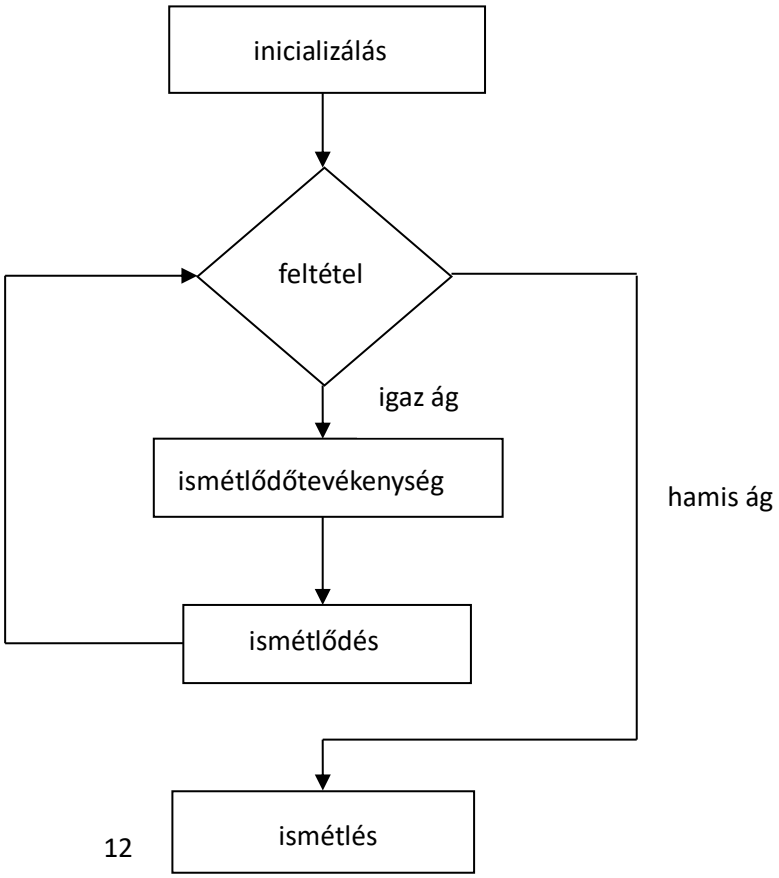


Amennyiben a zárójelben lévő kifejezés értéke igaz, az azt követő utasítás végrehajtódik. Ezután ismét kiértékelődik a kifejezés, igaz érték esetén megint végrehajtódik. Ez a ciklus addig folytatódik, amíg az érték hamis nem lesz.

A for ciklus utasítás formája:

for ciklusváltozó := kifejezés1 to kifejezés2 do utasítás ;

Először a kifejezés1 értékét veszi fel a ciklus változó, majd végrehajtja az utasítást. Ezután a ciklus változó eggyel növekszik, végrehajtja az utasítást. Ezt addig ismétli, amíg a kifejezés egyenlő nem lesz a kifejezés2 értékével. Ilyenkor utoljára végrehajtja az utasítást. Ha kifejezés2 értéke kisebb, mint kifejezés1, az utasítás nem hajtható végre. Ha kifejezés1 értéke egyenlő kifejezés2-vel, az utasítás csak egyszer hajtható végre.



Maga a program egy start gombbal kezdődik, amivel előhívjuk a játék felületet. A bal felső sarokban láthatjuk a menüsort. A Beállítások menüpontban találunk egy „**Új játék**” és „**Kilépés**” opciót. A „**Stílusok**” menüpontban található 3 különböző tervezés:

- *Régies*
- *Lány*
- *Fiú*

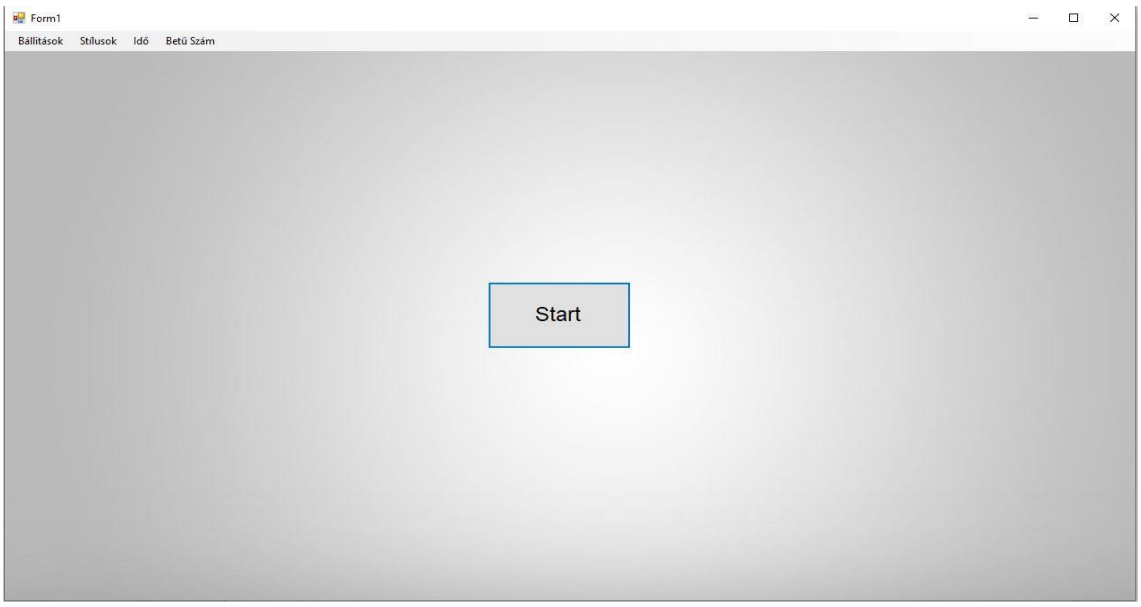
Az „**Idő**” beállításához is létrehoztam egy pontot, hiszen vannak olyan feladatok, amiben nem csak a helyes szavak száma számít, hanem hogy mennyi idő alatt gépelünk le annyi szót, ezért lehet választani:

- ✓ *1 perc*
- ✓ *3 perc*
- ✓ *5 perc*
- ✓ *10 perc*

A betűk hosszúságának is csináltam egy menüsort, amit „**Betű Szám**” -nak neveztem el. Itt lehet kiválasztani, hogy hány betűs szavakat szeretnénk játszani:

- ❖ *3 betűs*
- ❖ *4 betűs*
- ❖ *5 betűs*
- ❖ *6 betűs*

14. ábra Kezdőképernyő



(Forrás: Saját szerkesztés)

4.2.2 Gomb:

Gomb minden program párbeszédablakában megtalálható. Funkcióját tekintve leegyszerűsíti a kattintásos események használatát. Működésekor megtörténik valamilyen esemény, például egy másik ablak felugrása. Programom írása során gyakran használtam..

4.2.3 Címke:

A címke vezérlő, aminek segítségével a felhasználó statikus szöveget jeleníthet meg. Általánosságban elmondható, hogy ez a leggyakrabban használt vezérlőtípus, mivel ennek kezelése a legkönnyebb. Használata során más vezérlőket írnak le a felületen, például szövegmezők. A címke használata a programomban kulcsfontosságú, mivel ezen jelennek meg a különböző hosszúságú szavak. Az időzítő, hiba, illetve a pont is címkéken jellenek meg.

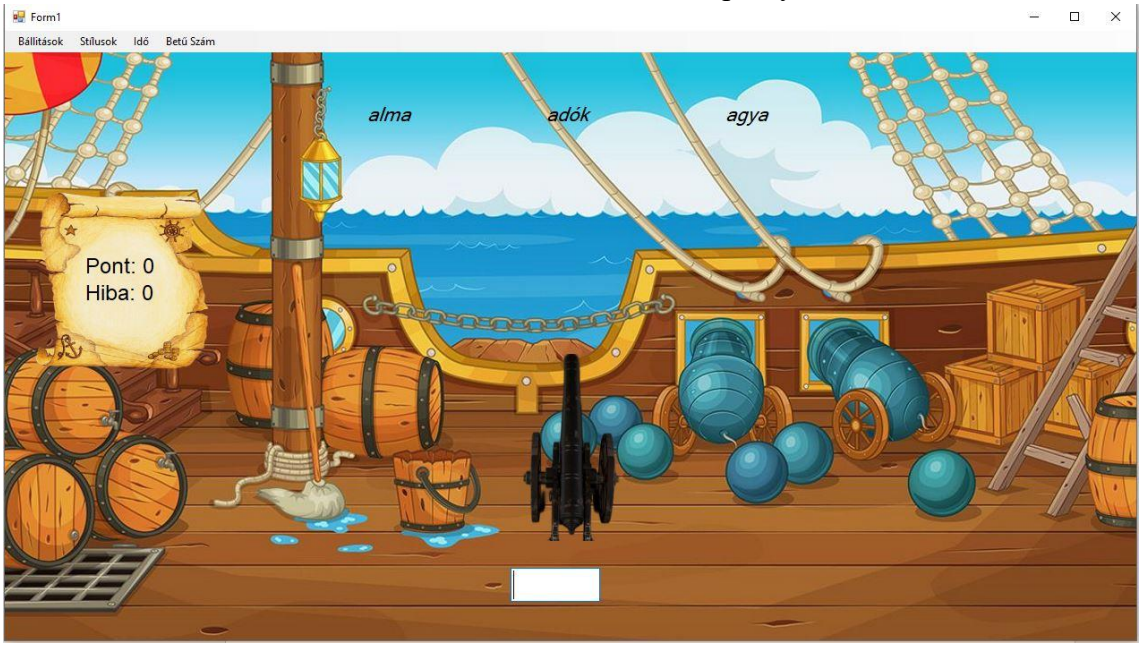
példa a címke készítésére:

```
Label időzítő=New Label;
```

4.2.4 Időzítő:

A feladata, hogy az előre meghatározott időintervallum letelte után egy adott esemény történjen. Az időtartamot, ami után az esemény történik millimásodpercben kell megadni, 1000 millimásodperc = 1 másodperc. Ahhoz, hogy működjön az „Enable” tulajdonságot kell megadnunk, anélkül nem indul el. Programomban az időzítőt a játék időtartamának megadásakor alkalmaztam. A felhasználó igénye szerint tudja beállítani a kívánt időt,

15. ábra Fiú stílus kezdőképernyő



(Forrás: Google, saját szerkesztés)

A „**15. sz.**” képen látható a „**Fiú**” stílusú játéktér. Az „**Idő**” és a „**Betű Szám**” beállítása után már kezdhetjük is a program használatát. Az idő azonnal elindul, ha elkezdünk írni a szövegdobozba. Ha a képen látható „alma” szót beírjuk akkor az ágyú balra fordul és egy lövés hatására eltűnik a szó és csak a robbanás jelenik meg egy pár pillanatra, és ugyan így jár el az ágy a többi 2 szóval. Ha mind a 3 szó elfogyott akkor újra töltődőnek a „**labellek**” és folytathatjuk tovább a játékot.

16. ábra Fiú háttér

```
//háttér
Image hatter = new Bitmap(@"boybg1.png");
this.BackgroundImage = hatter;
//
//láthatóság
ágyu.Visible = true;
//
```

(Forrás: Saját szerkesztés)

A háttér kiválasztásánál figyeltem arra, hogy a stílushoz illeszkedjen. Készítettem „**hatter**” nevű „**Image**” változót, amibe beillesztettem a „**boybg1.png**” nevű hátteret. A „**this.Background = hatter**” paranccsal hozzá rendeltem a „**form**” hátteréhez. A ágyúnak a láthatóságát igazzá tettem hogy láthassuk majd a mozgását.

Létrehoztam egy „**ágyu**” nevű „**PictureBox**”-ot amibe beolvastam a „**Image.Fromfile**” parancs segítségével a „**cannonmiddle.png**” képet. Az ágyú magasságát 212 pixel magasra és 152 pixel szélesre állítottam. Az elhelyezkedését a képernyő szélességének a fele mínusz 83 pixel adta és a magasságának a fele. A hátterét a Picturebox-nak láthatatlanná tettem hogy beleolvadjon a „**form**” hátterébe. Az ágyú képet a Pictureboxnak megfelelően középre igazítja a „**PictureBoxSizeMode.CenterImage**” parancs.

17. ábra Ágyú elhelyezkedése

```
ágyu.Image = Image.FromFile("cannonmiddle.png");
ágyu.Height = 212;
ágyu.Width = 152;
ágyu.Location = new Point(((this.Width) / 2) - 83, ((this.Height) / 2));
ágyu.BackColor = Color.Transparent;
ágyu.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(ágyu);
//
```

(Forrás: Saját szerkesztés)

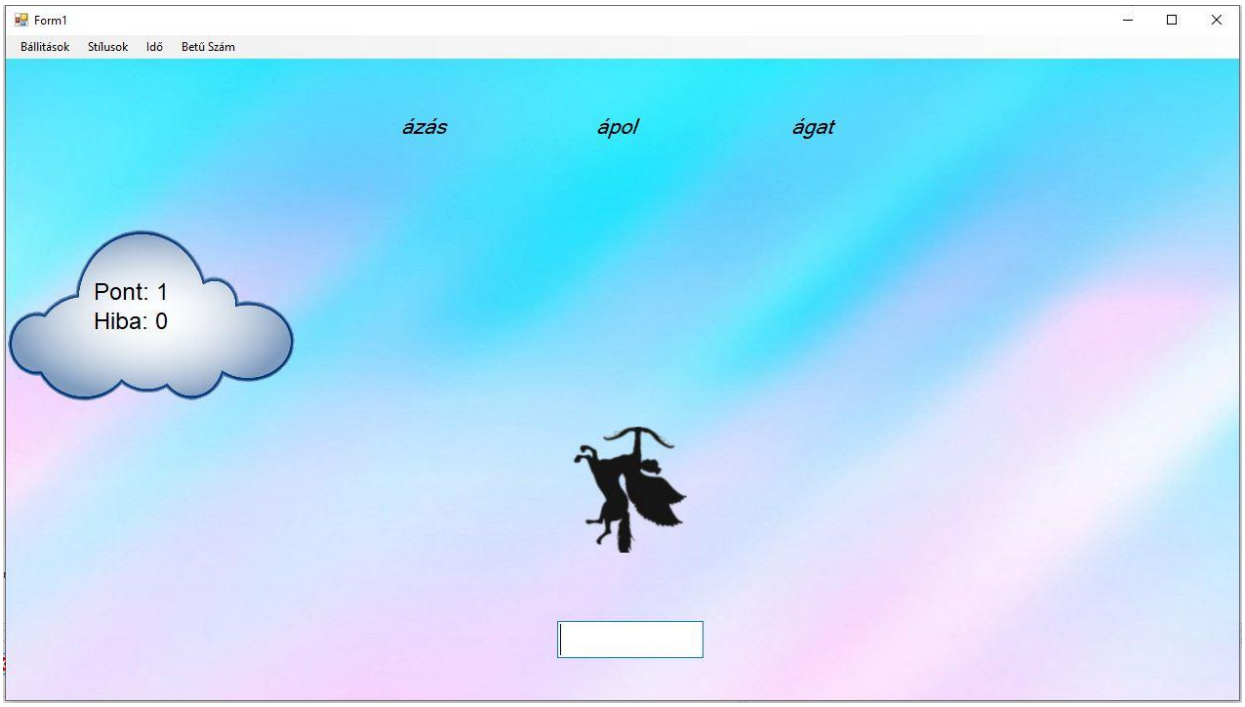
Az „**ágyu**”-nak a robbanás animációját 3 képből készítettem el, ezek egy sorozat képként funkcionálnak, egymás után következnek. A képeknek a méretét fokozatosan állítottam, hogy folyamatosan növekedjenek. Ezek a „**boom PictureBox**”-ok a 3 „**Label**”-hez tartoznak és a robbanás effektust adják hozzá.

18. ábra Robbanás Effekt

```
boom1.Height = 131;
boom1.Width = 200;
boom1.Location = new Point(310, 45);
boom1.BackColor = Color.Transparent;
boom1.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(boom1);
boom1.Visible = false;
//
boom2.Height = 131;
boom2.Width = 200;
boom2.Location = new Point(500, 45);
boom2.BackColor = Color.Transparent;
boom2.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(boom2);
boom2.Visible = false;
//
boom3.Height = 131;
boom3.Width = 200;
boom3.Location = new Point(740, 45);
boom3.BackColor = Color.Transparent;
boom3.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(boom3);
boom3.Visible = false;
```

(Forrás: Saját szerkesztés)

19. ábra Lány Stílus kezdőképernyő



(Forrás: Google, saját szerkesztés)

Csináltam egy „**Lány**” nevű menüpontot, ami színesebb élmény nyújt a felhasználók számára. Itt az ágyú helyette egy kupidó áll, aki tűzijáték formában eltünteti a szavakat. Folyamatosan változtatja az irányát a szavaknak megfelelően.

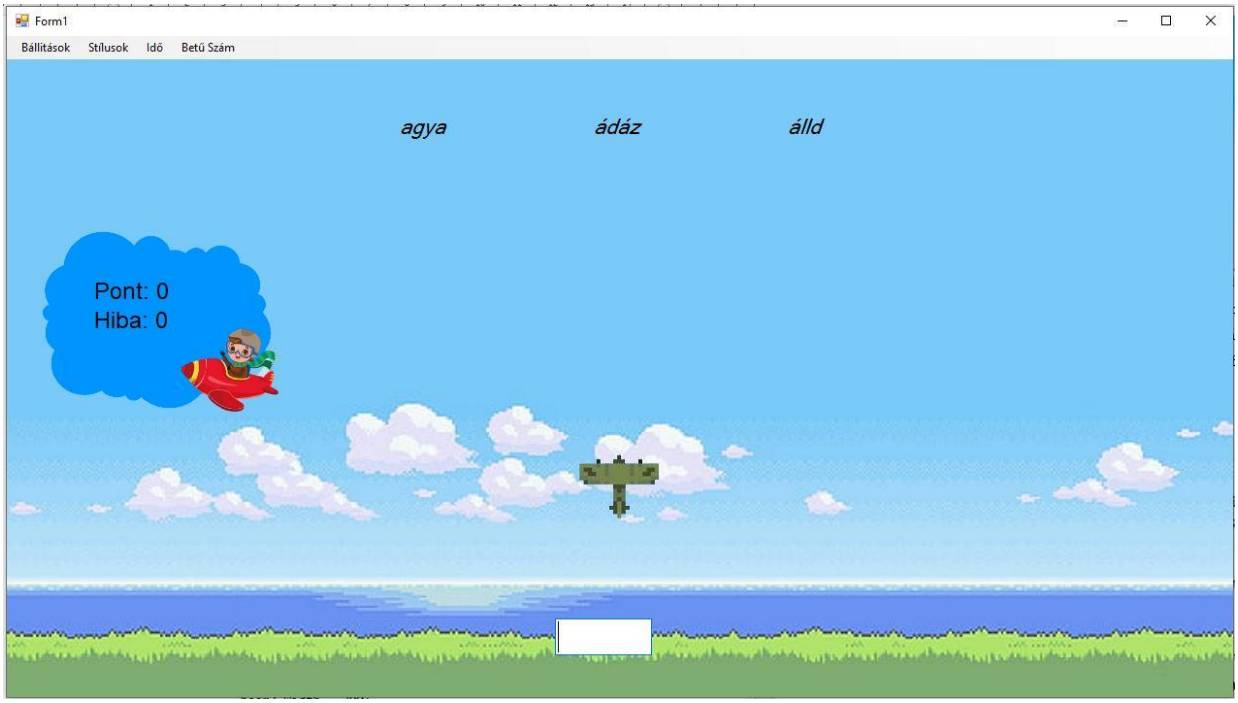
20. ábra Lány háttér

```
ágyu.Visible = true;
időzítő1.Visible = false;
timer1.Stop();
kereső.Focus();
//háttér
Image ghatter = new Bitmap(@"girlbg.jpg");
this.BackgroundImage = ghatter;
//
//képek
ágyu.Image = Image.FromFile(téma+"middle.png");
//
```

(Forrás: Saját szerkesztés)

A „**Lány**” hátteret „**girlbg**”-nek neveztem és a „**this.BackfroundImage**” paranccsal hozzá adtam a formhoz. Az „ágyu” változót folyamatosan változtatom a kép a helyes szó irányának megfelelően.

21. ábra Régies stílus kezdőképernyő



(Forrás:Google, Saját szerkesztés)

Ez a Stílus inkább az idősebb korosztály számára lehet szimpatikusabb, hiszen a régi „*old shcool*” játékosok ilyen felületen játszottak többet. A parancsokat felhasználtam a „**Régies**” Stílushoz is, a robbanás animáció itt is megjelenik, mint a „**Fiu**” Stílusnál.

22. ábra Régies háttér

```
időzítő1.Visible = false;
kereső.Focus();
timer1.Stop();
//háttér
Image hatter5 = new Bitmap(@"régibg1.jpg");
this.BackgroundImage = hatter5;
//
//képek
ágyu.Image = Image.FromFile(téma+"middle.png");
//
boom1.Height = 131;
boom1.Width = 200;
boom1.Location = new Point(310, 45);
boom1.BackColor = Color.Transparent;
boom1.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(boom1);
boom1.Visible = false;
//
boom2.Height = 131;
boom2.Width = 200;
boom2.Location = new Point(500, 45);
boom2.BackColor = Color.Transparent;
boom2.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(boom2);
boom2.Visible = false;
//
boom3.Height = 131;
boom3.Width = 200;
boom3.Location = new Point(740, 45);
boom3.BackColor = Color.Transparent;
boom3.SizeMode = PictureBoxSizeMode.CenterImage;
this.Controls.Add(boom3);
boom3.Visible = false;
```

(Forrás: Saját szerkesztés)

5. Elkészítés során felmerülő problémák

5.1 A szavak kiválasztása és feltöltése

A következő ábrán látható a kódsor, ami a programban szerepel. Ezt fogom részletezni soronként. A többi szó kereséshez is ezt a kódsort használtam csak más változó elnevezésekkel.

23. ábra Szóhossz keresés

```
if (szóhossz == 3)
{
    if (kereső.Text == elsőszó.Text)
    {
        összeg++;
        Lista.Remove(elsőszó.Text);
        elsőszó.Text = Lista[rd];
        kereső.Text = "";
        ágyúbal();
        timer2.Start();
        boomtf1 = true;

    }
    else if (kereső.Text == másodikszó.Text)
    {
        összeg++;
        Lista.Remove(másodikszó.Text);
        másodikszó.Text = Lista[rd];
        kereső.Text = "";
        ágyúközép();
        timer2.Start();
        boomtf2 = true;
    }
    else if (kereső.Text == harmadikszó.Text)
    {
        összeg++;
        Lista.Remove(harmadikszó.Text);
        harmadikszó.Text = Lista[rd];
        kereső.Text = "";
        ágyújobb();
        timer2.Start();
        boomtf3 = true;
    }
    else
    {
        if (kereső.Text.Length == szóhossz)
        {
            MessageBox.Show("eggyel sem egyezik");
            kereső.Text = "";
            Hiba++;
        }
    }
}
```

(Forrás: Saját szerkesztés)

A szavak kiválasztását és ellenőrzését a képen látható módon oldottam meg. Az elején, mikor kiválasztottuk, hogy milyen hosszúságú szavakat szeretnénk be gépelni, akkor a „szóhossz” változóba kerül bele. A kereső, amibe a szavakat írjuk, ellenőrzi, hogy amit beírunk szó egyezik-e bármelyik három szóval, amint egyezést talál a pontszámunkhoz hozzá ad egyet, az adatbázisból kitörli ideiglenesen azt a szót, így elkerülhető az, hogy megismétlődjön ugyan az a szó. Ezután az adatbázisból kiszed egy szót, amit egy véletlenszerűen generált szám fog meghatározni, ezáltal az a szó fog bekerülni a programba. A kereső ezekután törli a benne lévő adatot és üres lesz. Utána az ágyú fordul oda ahhoz a szóhoz, amit sikerült beírnunk majd elkezdődik a robbanás effektus.

Ha Véletlenül nem sikerült volna egyik szót sem eltalálni akkor egy ablak nyílik meg, ami tájékoztat, hogy nem egyezik egyik számmal sem.

5.2 A robbanás effektus elkészítése

Itt a robbanást fogom szemléltetni, hogy hogyan sikerült három kép segítségével majdhogynem egy animációt készíteni.

24. ábra Robbanás élesítés

```
private void timer2_Tick(object sender, EventArgs e)
{
    if (boomtf1==true)
    {
        if (számláló2 < 4)
        {
            boom1.Image = Image.FromFile(téma + számláló2 + ".png");
            számláló2++;
        }
        else
        {
            boom1.Image = null;
            boomtf1 = false;
            számláló2 = 1;
            boomtf4 = true;
            timer2.Stop();
        }
    }
    else if (boomtf2 == true)
    {

        if (számláló2 < 4)
        {
            boom2.Image = Image.FromFile(téma + számláló2 + ".png");
            számláló2++;
        }
        else
        {
            boom2.Image = null;
            boomtf2 = false;
            számláló2 = 1;
            boomtf5 = true;
            timer2.Stop();

        }
    }
```

(Forrás: Saját szerkesztés)

A robbanás effektus sok fejtörést igényelt, A boomft1 változó alapértelmezettben hamisra van állítva. Az előző oldalon leírtam, hogy amit elfogadja a szót elindul a robbanás effektus. Itt végzi el az ellenőrzést, hogy ténylegesen igaz-e a változó. Ha igaz akkor elindít még egy feltételt, ami a száláló2 változót ellenőrzi. Ez arra szolgál, hogy alapértelmezettként egy a változó értéke, így ennyiről kezdi az ellenőrzést. A feltétel teljesülésekor megjeleníti a képet a robbanásról. három darab képet csináltam különböző méretekben, hogy minél élethűbb legyen az effektus. a legkisebb képpel kezdi az egyes szám miatt. Ha megjelenítette a képet akkor hozzá add a változóhoz egyet és előrről kezdi az egészet. Ezt addig csinálja, amíg a számláló2 változó kisebb lesz, mint 4. Utána életbe lép az else-ág, ami eltünteti a robbanás képet, ismét hamisra állítja a változót és visszakapja az eredeti értékét a számláló2. Ennek az ágnak még van egy boomtf3 része is, mivel három darab képből készült a robbanás.

6. Továbbfejlesztési lehetőségek

Napjainkban egyre többen használják egyre gyakrabban a számítógépet a munkájuk során, ha jobban tudnák használni a 10 ujjas vakírást az meggyorsítaná a munkájukat is hiszen könnyebben meg tudnák osztani figyelmüket a gépelés és az ezzel párhuzamosan folyó tevékenységük között.

6.1 **Különböző nyelveken történő gyakorlás**: Nagyon fontos, hogy nem csak a magyar nyelvű munkában, hanem az idegen nyelven folytatott munkában is minél gyorsabban és hibátlanabbul dolgozzunk. Ezért lehet szükség angol vagy német nyelven történő gyakorlásban is. Főleg a német nyelv esetén, hiszen ott sok speciális karakter, illetve billentyű kombináció használandó például: ä, ß.

6.2 **Más textúrájú**: Ezek a textúrák főként gyereknek készült, kellemesebb közeget képes teremteni, így számokra. Továbbá lehetne sokkal letisztultabb tematikájú kinézeteket is alkotni.

6.3 **Hosszabb szavak**: Az írásuk nagyobb figyelem összpontosítást igényel. Ennek gyakorlására egészen hosszú (például: 25-30 karakteres) szavak is bekerülhetnek a gyakorló programba.

6.4 **Kerek mondatok**: A mindennapi életben nem szavakat, hanem mondatokat használunk a kommunikáció során. Vannak olyan mondatok, amelyek gyakran előfordulnak egyes, esetekben ezeknek a mondatoknak gyakorlására is összpontosíthat egy továbbfejlesztett programváltozat.

6.5 **Több soros szöveg**: A tanulás egy bizonyos pontja után szükség van folyamatos szövegek gyakorlására is, hiszen az összefüggő szövegek írása során zavart okozhat, ha ugyan az a szó többször is előfordul a mondatokban.

6.6 **Bal és jobb kezes szavak**: Az írás során nehézséget okoz, ha egymás után több karakter leütéséhez ugyan azt a kezet kell használni, ezért az írásbiztonság miatt célszerű az ilyen szavaknak a külön gyakorlása.

6.7 **Teljes képernyő:** Fontos lenne, hogy a program illeszkedjen a felhasználó képernyőjéhez. Jelenpillanatban egy előre beállított nagyságban elérhető.

7. Ábrajegyzék

NYILATKOZAT a záródolgozat nyilvános hozzáféréséről

és eredetiségéről

A hallgató neve: Orosz Balázs

A Hallgató Neptun kódja: GWV8BS

A dolgozat címe: Egy gépírást fejlesztő program bemutatása

A megjelenés éve: 2024

A konzulens intézetének neve: Műszaki Intézet

A konzulens tanszékének a neve: Alkalmazott Informatikai Tanszék

Kijelentem, hogy az általam benyújtott záródolgozat egyéni, eredeti jellegű, saját szellemi alkotásom. Azon részeket, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem, és az irodalomjegyzékben szerepeltettem.

Ha a fenti nyilatkozattal valótlant állítottam, tudomásul veszem, hogy a záróvizsga-bizottság a záróvizsgából kizár és a záróvizsgát csak új dolgozat készítése után tehetek.

A leadott dolgozat, mely PDF dokumentum, szerkesztését nem, megtekintését és nyomtatását engedélyezem.

Tudomásul veszem, hogy az általam készített dolgozatra, mint szellemi alkotás felhasználására, hasznosítására a Magyar Agrár- és Élettudományi Egyetem mindenkori szellemitulajdon-kezelési szabályzatában megfogalmazottak érvényesek.

Tudomásul veszem, hogy dolgozatom elektronikus változata feltöltésre kerül a Magyar Agrárés Élettudományi Egyetem MATER Hallgatói Dolgozatok repozitoriumába. Tudomásul veszem, hogy a megvédett és - nem titkosított dolgozat a védést követően - titkosításra engedélyezett dolgozat a benyújtásától számított 5 év eltelte után nyilvánosan elérhető és kereshető lesz az Egyetem MATER Hallgatói Dolgozatok repozitoriumában.

Kelt: 2024 év 04 hó 21 nap

Orosz Balázs

Hallgató aláírása

MATE Szervezeti és Működési Szabályzat
III. Hallgatói Követelményrendszer
III.1. Tanulmányi és Vizsgaszabályzat
6.13. sz. függeléke: A MATE egységes szakdolgozat / diplomadolgozat / záródolgozat / portfólió készítési útmutatója 4.1. sz. melléklete: Konzulensi nyilatkozat

NYILATKOZAT

____Orosz Balázs_____ (név) (hallgató Neptun azonosítója: _GWV8BS__) konzulenseként nyilatkozom arról, hogy a záródolgozatot áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A záródolgozatot a záróvizsgán történő védésre **javaslom / nem javaslom**

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem

Kelt: _____2024_____ év _____04_____ hó ____21__ nap

_____
belső konzulens