Műszaki Iskola

Ada

**DIPLOMAMUNKA**

A Műszaki Iskola tanműhelyének raktár nyilvántartása adatbázissal .NET

környezetben

A tanuló neve:Bakhit Patrik

Osztály:IV-3

Tantárgy:Programozás

Tanár: Jasztrebinác Edit

Szakterület:Elektrotechnika

Oktatási profil:Számítógépek elektrotechnikusa

Tanév: 2017/2018

--

# **Tartalomjegyzék**

[**Tartalomjegyzék** 3](#_Toc515574373)

[**Bevezető** 4](#_Toc515574374)

[**Kezdőlap** 5](#_Toc515574375)

[2.1.Felhasználói felület 5](#_Toc515574376)

[2.1.1.Bejelentkezés felülete 5](#_Toc515574377)

[2.2.Oledb 6](#_Toc515574378)

[2.3.Programrészek 6](#_Toc515574379)

[2.3.1.Bejelentkezés gomb működése 7](#_Toc515574380)

[2.3.2. Vissza gomb működése 9](#_Toc515574381)

[2.3.3. Beállítások gomb 9](#_Toc515574382)

[**Adatbázis kezelő** 10](#_Toc515574383)

[3.1.Felhasználói felület 10](#_Toc515574384)

[3.1.1.A felhasználói fő felület 10](#_Toc515574385)

[3.1.2 A felhasználói mellék felület 11](#_Toc515574386)

[3.2.Adatbázis táblázat 11](#_Toc515574387)

[3.3.Adat hozzáadás 14](#_Toc515574388)

[3.4.Adat módosítás 16](#_Toc515574389)

[3.5.Adat törlés 17](#_Toc515574390)

[3.6.Előzmények 18](#_Toc515574391)

[3.7.Fejésztési lehetőségek 19](#_Toc515574392)

[**4.Befejezés** 19](#_Toc515574393)

[**Ábrajegyzék** 20](#_Toc515574394)

[**Felhasznált Irodalom** 20](#_Toc515574395)

# **Bevezető**

A 2017/2018-as évi diplomamunka feladatatomnak a A Műszaki Iskola tanműhelyének raktár nyilvántartása adatbázissal .NET környezetben-t választottam. Programozást 3.éve tanulom, 2.osztályos korom óta, az első két évben a C++ programozási nyelvben programoztuk. A C++ programozási nyelvet **[Bjarne Stroustrup](https://hu.wikipedia.org/wiki/Bjarne_Stroustrup" \o "Bjarne Stroustrup)** írta meg 1983-ban, ő úgy vélte, hogy a hagyományos C nyelv elavult és ezért készítette a C++-t. Mikor 4.osztályosok lettünk elkezdtünk dolgozni a .NET C#-al. A C# objectum orientált programozási nyelv a Microsoft által jött létre a .NET keretrendszer részeként. A nyelv készítésekor a C++-t és a Java-t vették alapul. A .Net egy ingyenes platformfüggetlen szoftver, amit sok különböző tipusú alkalmazások készítésére használnak. A .NET-el lehetőségünk nyillik több nyelvet, szerkesztőt és könyvtárat használni web, mobil, asztaligép és sok sok más alkalmazás elkészítésére. Három nyelvet lehet használni a .NET környezetben C#, F# és Visual Basic. A .NET C#-nak több felhasználási lehetősége van, készíthetünk vele Windows kliens alkalmazást, XML web szolgáltatást, adatbázis alkalmazásokat, stb. Fejlett kód szerkesztővel és könnyen kezelhető felhasználói felülettel látták el, és még sok más eszközzel hogy megkönnyítsék a progamozást. A C# roppant kifejező, mégis könnyen megtanulható nyelv. Azok akik a C,C++ vagy a Java programozó nyelveken dolgoztak, azoknak sokkal gyorsabban állnak át a nyelvre. Sok olyan funkciót raktak bele mint ami a C-ben és a C++-ban nem található meg, az összeszámlálás, a jogátruházás, a lambda kifejezéseket, és közvetlen memória hozzáférés(DMA). Mint egy objektumorientált nyelv a C# támogatja a beágyazásokat, az öröklődéseket és a polimorfizmust. Egy C# programot 6 fordítóprogramban lehet megírni, a [Roslyn](https://en.wikipedia.org/wiki/.NET_Compiler_Platform), a SharpDevelop, a Portable.NET, a Mono, a Visual C# Express, és a Visual C#. Mi az iskolában SharpDevelopban programoztunk, ami egy ingyenes és nyilt forrású fordító program. Én otthon kipróbáltam a Visual C#-t amit egy Visual Studióként is ismert. Nekem megtetszettek a kisebb fejlesztések amiket a Visual Studióban meglehetett találni. Ezért döntöttem úgy, hogy ezt a diplomamunkámat Visual Studióban fogom megírni. Beszereztem a program ingyenes verzióját a Visual C# Express-t, úgy véltem, hogy azok a funkciók amiket az ingyenes verzióban megtalálhatók, elég lesz majd a programom elkészítéséhez. Eleinte kicsit szokatlan volt a felhasználói felület, de gyorsan hozzászoktam.

1. ábra-C++

2. ábra-Visual Studio

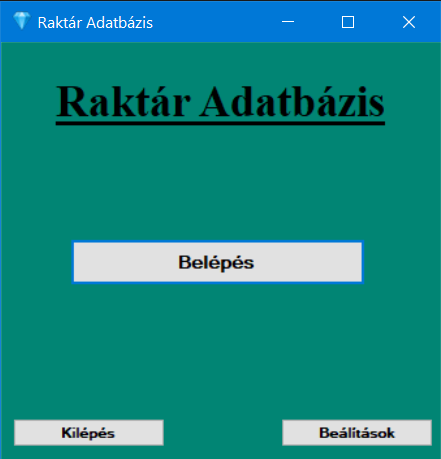
3. ábra-Bjarne Stroustrup

# **Kezdőlap**

## 2.1.Felhasználói felület

Mikor a felhasználó elindítsa a programot egy zöld háttért 468x443-as pillant meg, a zöld szín a későbbiekben változtatható lesz. Az ablakon látható a program címe nagyban egy szöveg mezőben. Látható még 3 gomb, egy a bejelentkezésnek, 1 a beállításoknak és 1 a kilépésre. Viszont ezt csak a felhasználó láthatja így, valójában vannak még elemek a háttérben elrejtve amiket majd a későbbiekben fogok részletezni.

### 2.1.1.Bejelentkezés felülete

A bejelentkezés gombra kattintva több folyamat is lejátszódik, a szöveg mezőben lévő szöveg átneveződik “Belépés”-re, ezt a .**Text** tulajdonsággal érem el, ami megváltoztatja a hozzácsatolt elem szövegét. Ezután eltűnik a belépés gomb, amit a .**Visible= false** tulajdonsággal érem el, a false ,angolban hamist jelent vagyis egyszerűen mondva, ez a tulajdonsággal azt jelenti, hogy a hozzácsatolt elem visibility-je , magyarul láthatósága, hamis értéket kap. Miután csináltunk egy kis helyet az ablakon, megjelenítsük a 4.gombot, ugyanazt a metódust használjuk mint az eltüntetésnél, csak **Visible= false** helyett **Visible= true**-t használunk, a true az angolban igazt jelent, vagyis azt mondjuk, hogy az elem láthatósága igaz. Viszont egy gombbal nem sokat érünk el ezért kellenek még a szintén elrejtett szövegdobozok és cimkék. Az különbség a szövegdoboz és a szövegmező között, hogy a szövegdobozba lehet értéket bevinni, akkor is mikor a program fut, vagyis egy beviteli adat gyüjtőként szolgál. A **.Text** tulajdonságot kihasználva megváltoztatjuk az eddigi kilépés gomb nevét „Vissza“ névre, illetve az újonnan megjelent gombunknak is nevet adunk a .**Text** segítségével, ezt „Belépés“ névre írjuk át, utalva jövőbeli funkciójára. Ezentúl még egy elem jelenik meg az ablakon, egy úgynevezett jelölőnégyzet, aminek a neve „Vendég“.

4. ábra-Kezdőlap

## 2.2.Oledb

Az Oledb kapcsolatot használjuk arra hogy a programra kapcsoljunk egy Access adatbázist. Az interneten sokan mondták, hogy jobban megéri SQL adatbázist használni Access helyett, mondván hogy biztonságosabb, de a feladathoz tökéletesen megfelel az Access adatbázis. Az Oledb-t a Microsoft fejlesztette ki, a célból, hogy megkönnyítse az adat átvitelt. Ahhoz hogy használni tudjuk, ahhoz meg kell hívnunk az **using System.Data.OleDb**-t. Csakis ezután érhetőek el a hozzá kapcsolódó parancsok és metódusok. Szükségünk van egy objektumra a **OleDbConnection**-ből amit én conn-nak nevezem el az angol connection szóbol, ami kapcsolatot jelent. Mivel hiába van egy Oledb-s objektumunk ha nincs neki megmondva hogy hova csatlakozzon, ezért a fő oldalunk konstruktorába elhelyeztük a conn.ConnectionString =“...“, amiben meg fogalmazzuk a **provider**-t, magyarul a szolgáltatót, ami megmondja a programnak, hogy milyen szolgáltatótól várjon kapcsolatot, a mi esetünkben a **Microsoft.ACE.OLEDB.12.0** használtunk. Ezután **Data Source**-t adtuk meg ami magyarul adat forrást jelent, ebből értendő, hogy ezzel megadjuk, hogy hova próbáljon meg csatlakozni.

**private OleDbConnection conn = new OleDbConnection();**

**conn.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=" + ConfigurationManager.AppSettings["udir"] + ";Persist Security Info=False;";**

## 2.3.Programrészek

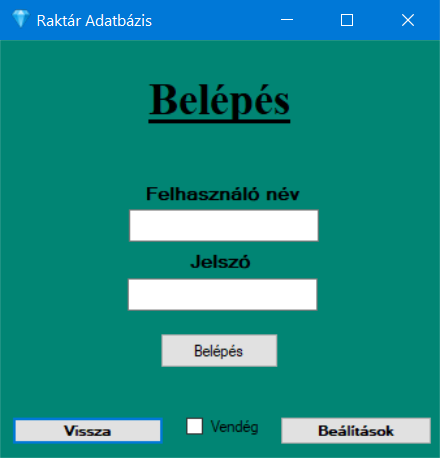
A programnak a kezdő ablakja nagyon egyszerűen működik, A bejelentkezés gombot megnyomva hozzáférhetünk a bejelentkezéshez, és kért adataikat beírva tovább haladhatunk a programban. A Kilépés gombbal visszafelé haladunk a programban és a végén ki is lép a programból. A beállítások gombbal kinyílik a beállítások ablak.

### 2.3.1.Bejelentkezés gomb működése

Miután megnyomjuk a bejelentkezés gombot és véghezmegy az összes változtatás az ablakon, a felhasználó 2 szöveg dobozt és 3 gombot lát, és persze a szöveg mezőket. A felhasználó kétféleképpen jelentkezhet be a programba. Az egyik, hogy adminként beírja a szöveg dobozokba a felhasználó nevét és a jelszavát, a másik, hogy mint vendég lépjen be. Mint vendég a felhasználó csak egy kevés részéhez férhet hozzá. Ahhoz, hogy valaki adminként regisztrálni tudjon, szüksége lesz egy már bejelentkezett adminra, de ezt majd később részletezem. Vendég valaki úgy lesz hogyha kiválasztja a belépés gomb alatt található jelölőnégyzetet, mikor ezt a felhasználó kiválasztja, ehhez egy olyan függvényt alkalmaztam ami csak akkor aktiválódik mikor megváltozik a jelölőnégyzet értéke. Először a függvényben az **if** segítségével meghatározzuk, hogy éppen kivan-e választva a jelölőnégyzet ha igen akkor a jelszó szövegdoboz és szövegmező eltűnik és a 4.gomb áthelyezkedik az ablakon belül, hogy ez megtörténjen a .Location metódussal érjük el, amit új pont objectumal meghatároztunk hogy mennyivel menjen arréb, **b4.Location = new Point(123, 160).** Ha ajelölőnégyzet nem volt kiválasztva akkor az elrejtett jelszó szövegdoboz és szövegmező megjelennek és a gomb visszaugrik eredeti helyére. Miután a felhasználó beírta a kért adatokat megnyomja a 4.gombot ami rögtön leellenőrzi, hogy vendég- vagy adminként akar a felhasználó belépni. Ha a jelölőnégyzet értéke igaz akkor megkezdi a program és elmenteni a belépési adatokat. Először kinyitja az Oledb connectiont az **.Open()** metódussal, létrehozunk az Oledb parancs objectumot, majd megírjuk hozzá a lekérdezés szöveget, a lekérdezés szöveg az SQL nyelven van írva, ami egy standard nyelv az adatbázisok adatainak a tárolására, manipulálására és megszerzésére. A programhoz egy egyszerű insert vagyis magyarul beillesztés kódot használtam.

**Insert into táblanév(oszlop1,oszlop2,...) values(érték1,érték2,...).**

Értékként megadtam a felhasználó előzőleg beírt felhasználó nevét és a belépési dátumot és a belépés idejét, ezeket az adatokat egy adatbázisban mentettem el**.** Azért, hogy ne keljen használni Dataset-et az adatbázis megváltoztatásához a **.ExecuteNonQuery** metódust használtuk, egy **g** nevű változónak 1-es értéket adtam azért, hogy a jövőben tudjuk hogy a felhasználó vendég, és legvégül lezárjuk a kapcsolatot, és megjelenítsük a felhasználó számára a következő ablakot.

****Abban az esetben ha a felhasználó nem vendég hanem admin akkor le ellenőrizük hogy a beírt felhasználó név és jelszó kombináció megtalálható-e az elmentett adminok között. Itt a **Select \* oszlop1,oszlop2 from táblanév** SQL kódot használjuk, a \* megjelőlés arra szolgál, hogy megmondjuk a programnak hogy minden oszlopot kérünk abból a táblából. Majd lefuttatom az Oledb olvasóját, **OleDbDataReader.** Ezután meg kell néznünk, hogy van-e egyezés az adatok között, ehhez létrehozunk egy **c** változót aminek 0 értéket adunk meg majd egy **while** függvény segítségével megírtam, hogy ameddig a program olvassa az adatbázist a **c**-t növelje minden egyes alkalommal mikor van egyezés. Ha a **c** értéke megegyezik 1-el akkor van egyezés. Ezután lefutatjuk még egyszer ugyanazt a programrészt amit a vendégnél használtunk azért, hogy elmentsük a belépő nevét és belépésének idejét és szintúgy mint a vendégnél tovább engedjük a második ablakra.

**5**. ábra-Bejelentkezés

**conn.Open();**

**OleDbCommand cmd = new OleDbCommand();**

**cmd.Connection = conn;**

**cmd.CommandText = "select \* from Admin where user\_name='" + nevBox.Text + "' and [password]='" + jelBox.Text + "'";**

**OleDbDataReader read = cmd.ExecuteReader();**

**int c = 0;**

**while (read.Read())**

**{**

**c++;**

**}**

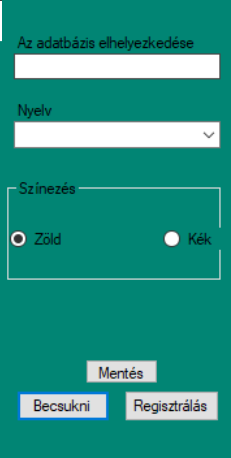
Ha a felhasználó nem adta meg jól a nevét vagy a jelszavát, akkor egy felugró ablakban ezt a szöveget mutassa meg a felhasználónak“ **Nem jó a felhasználó név vagy a jelszó, kérem próbálja újra!“ .**

### 2.3.2. Vissza gomb működése

Ennek a gombnak 2 funkciója van, hogy visszafelé haladjon a programban és hogy kilépjen a programból mikor a felasználó befejezte a használatát. Egy egyszerű módszerrel meghatározom hogy éppen ki kell lépni vagy visszafelé haladni, a Bejelentkezés felülete alcím alatt említettem, hogy a visszafelé gomb nevét megváltoztatom, a program megnézi, hogy jelenleg éppen mi a neve a gombnak, ha a neve **„Vissza“** akkor a bejelentkezés gomb által változtatott adatok vissza lesznek alakítva, a vissza gombbal együtt ami most „**Kilépés**“ néven fog szerepelni, ebben az esetben ha megnyomjuk akkor egy egyszerű paranccsal leállítom a program működését, **System.Windows.Forms.Application.Exit()** .

6. ábra-Beálítások

### 2.3.3. Beállítások gomb

A beállításokat 2 helyről lehet elérni, a bejelentkező oldalról és az Adatbázis kezelő ablakról. Ha a bejelentkező oldalon lépsz be akkor okvetlen csakis vendég jogosultsággal rendelkezel, vagyis nem érhetsz el minden lehetőséget. A beállítás gombbal lehetőségünk lesz beállítani az adatbázis helyét, az oldalak színét, és hogy milyen nyelven írja a problémákat és értesítéseket a program.

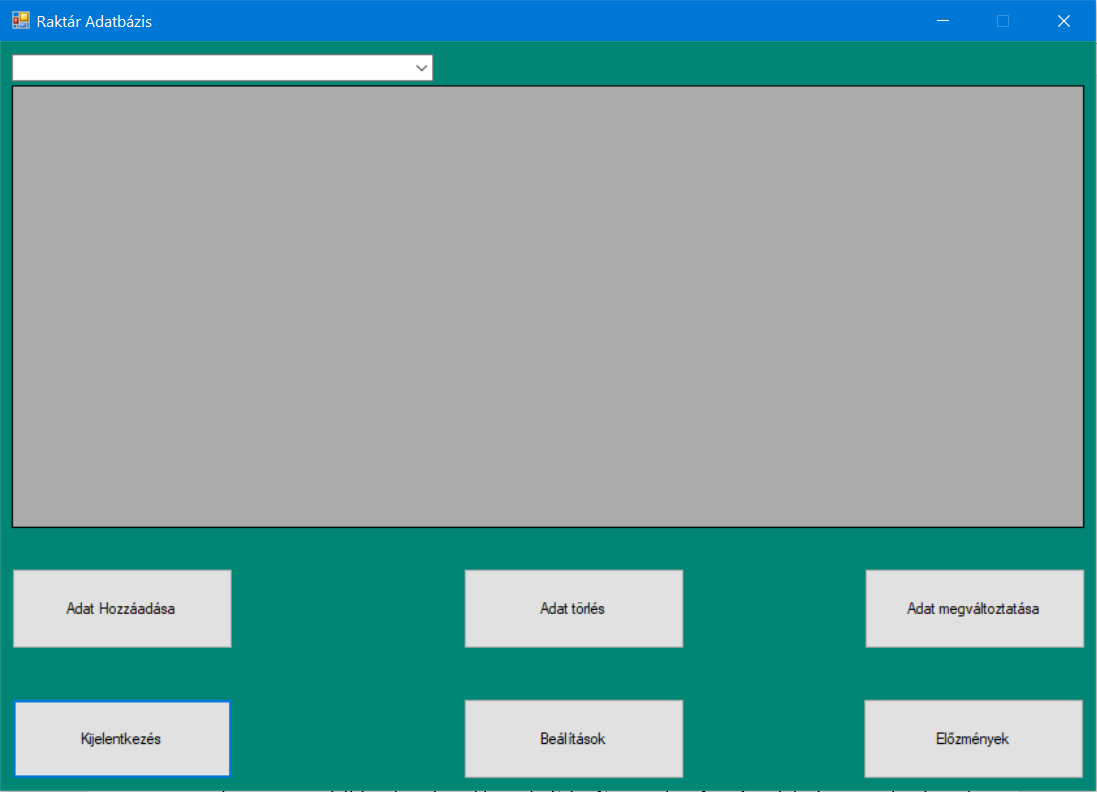
# **Adatbázis kezelő**

A programom második részében a adatbázisokat tudjuk kezelni, lehetőségünk van az adatbázishoz hozzáadni értéket, módosítani és törölni adatokat az adatbázisokból. Ezek mellett a felhasználó még megtekintheti az előzményeket illetve a beállításokon is állíthat.

## 3.1.Felhasználói felület

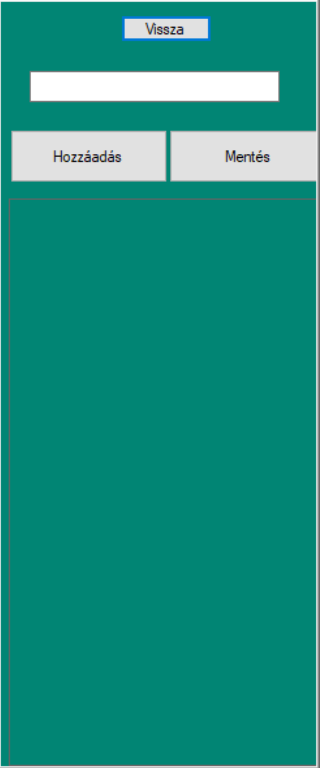
### 3.1.1.A felhasználói fő felület

Mikor a felhasználó először megpillantja a felületet akkor láthat rajta 6 gombot, 1 nagy **DataGridView**-t és egy annak a tetején egy lenyíló listát(**ComboBox**). A gombok közül kettő megnyit egy új ablakot, az egyik gomb a beállításokat nyissa meg, szintén egy új ablakba, egy másikkal megjelenítettem a **DataGridView-**n az előzményeket. A gombok hozzáférhetősége függ a felhasználó jogosultságától, ha a felhasználó adminként lép be akkor minden gombhoz hozzáférhet és minden adatot megtekinthet, viszont ha a felhasználónak vendég jogosultsága van akkor csak a kijelentkezés és a beállítások gombhoz van hozzáférése. Emelett még megtekintheti az adatbázis tábláit, de változtatást nem tehet bennük.



7. ábra-Fő oldal

### 3.1.2 A felhasználói mellék felület

Mikor megnyomjuk a hozzáadás vagy a módosítás gombot akkor megjelenik a fő oldal bal oldalán, a fő oldal magasságával megfelelő magasságú és körülbeül ¼ széles ablak. Ezen az ablakon van 4 gomb, egy vissza gomb, és 2 az adat kezeléséhez szükséges, van még egy szöveg doboz, egy nagy szöveg mező és egy kombinált szövegbeviteli doboz, és még van egy gomb a mentett adatok törlésére. A mellékoldalakat úgy készítettem el hogy automatikusan kövessék a fő oldal pozicióját, vagyis ha a fő oldalt az egérrel arrébb húzom akkor a mellék oldalak vele együtt mozogjanak, és mikor gomb kattintásra megjelennek akkor is rögtön fő oldal szélén álljanak. Az ötlet arra alapszik, ha a főoldal elhelyezkedése megváltozik akkor a mellék oldalt a fő oldalhoz visszonyítva egy megadott távolságot arrébb ugrik. Ezt a **Form-**nak a .**Location** tulajdonságávalértem el, fogtam és egy új ponttal meghatároztam, hogy mennyivel menjen arréb az oldal,

8. ábra-Mellék Oldal

**this.Location = new Point(this.Location.X + 47, this.Location.Y + 88).**

Természetesen ez nem volt elég hiszen így még ha a fő oldalt arrébb húzom a kurzor segítségével, és csak utána jelenítem meg a mellék táblát akkor az a fő oldal eredeti helyén fog megjelenni, és majd csak akkor fog a fő oldal után menni ha azt újfent elmozdítom. Ennek a megelőzése érdekében beleírtam a kódba még 4 darab elhelyezkedés változtatást. Úgy oldottam meg, hogy amikor a fő oldalon megnyomjuk az egyik gombot ami előhozná a mellék ablakot, a fő oldal rögtön utána elmozdul 1-1 egységet az **X** és **Y** koordinátákon,

**this.Location = new Point(this.Location.X + 1, this.Location.Y - 1),**

majd mivel attól még hogy szinte meg sem látszik az az 1-1 egységnyi változtatás azért nem szeretném otthagyni úgy hogy rögtön vissza is állítom,

**this.Location = new Point(this.Location.X - 1, this.Location.Y + 1)**

Mivel elmozdítottuk a fő oldalt rögtön miután a mellék oldal megjelent ezért a mellék oldal rögtön el is kezdik követni a fő oldal helyzetét.

## 3.2.Adatbázis táblázat

Az adatbázis táblázat helyezésekor igyekeztem minnél praktikusabbra és flexibilisre megcsinálni. A táblázat alapértelmezetten üres, a felhasználónak kell a kombinált szövegbeviteli doboz segítségivel kiválasztani a 3 jelenlegi adatbázisban szereplő táblázatot. A flexibilitás érdekében a következőképpen oldottam meg, hogy a program automatikusan érzékeljen minden egyes táblát az adatbázisban. Ugyanúgy mint a bejelentkézésnél az Oledb kapcsolatot hozunk létre és a konstruktorban beállítjuk az adatbázisunk elhelyezkedését. Készítettem egy új függvényt ehhez a folyamathoz, hiszen akkor még nem tudtam hogy fogom-e töbször is használni ugyan ezt a folyamatot a programom írása alatt. Szükségem volt egy változóra ezért létrehoztam az OP nevű változómat ami arrra szolgál, hogy megtudjam mondani a programnak hogy volt-e már futtatva, ha ez nincs megmondva neki akkor minden egyes kattintás a kombinált szövegbeviteli dobozra, még egyszer hozzáadná ugyanazokat az értékeket, így viszont egy sima **if** segítségével le lehet ezt ellenőrizni. Megnyitottam az adatbázishoz a kapcsolatot a kapcsolat**.Open()** metódussal, majd létrehoztam egy **Datatable**-t amiben eltároltam az adatbázis tábláit, **.GetSchema("TABLES").** Ez természetesen még nem elég, két problémával álltam szemben, az egyik hogy a Access adatbázisban vannak olyan táblák is amiket a felhasználó nem lát, és ezekre mink nem vagyunk kíváncsiak szóval meg kellett szabadulnom tőlük, ezt úgy tudtam megoldani hogy kiszűrtem a táblákat a tipusuk alapján, ez a megoldás meg rávilágított a második problémámra is ami pedig hogy vannak olyan táblák amik jó tipusúak, de szintén a felhasználó számára nem érdekesek. Megfigyelésem szerint ezek a táblák mindig „**Msys**“**-**el kezdődnek, így csak megkellett határoznom melyikben van „**Msys**“és melyikben nincsen. A programban létrehoztam egy **foreach**-et, a foreach arra szolgál hogy táblázatokban, listákban, illetve tömbökben való adatokat egyesével el tudjuk érni, **foreach (DataRow r2 in dt2.Rows)-**minden adatsor amit **r2** változóként kezelünk, megkapja az előbb elkészített **DataTable** sorainak az értékét, és végig halad mindegyiken. Létrehoztam még kettő szöveges változót, az egyiknek megadtam az r2 szöveges adattá alakított értékeit. A másik változóm megkapja az első változóm **.Substring()-**el megmetszett értékét. **str.Substring(0,4)** ez azt jelenti hogy a 0-tól a negyedik character-ig vágja le és helyezi el az **asd** megnevezésű változóba. Egy **if**-el megvizsgálom hogy melyik **asd** értéke felel meg a „**Msys**“-nek, ezzel jelezve, hogy az a tábla nem kell nekünk**,**  ahol ezt találta ahoz az értékhez tartozó táblát át írja üres értékké. Majd futtatam még egy **foreach-**et, ahol egy **if-**el kifejeztük, ha a tábla neve nem egyezik meg üres értékel akkor bele helyezi a kombinált szövegbeviteli doboz értékei közé.

**if (r["TABLE\_NAME"].ToString() != ""){**

**comboBox1.Items.Add(r["TABLE\_NAME"].ToString()); }.**

Még mindig nincsenek leszűrve a táblák, hogy csak azt mutassák ami nekem kellett, ezért még a **foreach** elejére létrehozok egy új **Datatable**-t**.**

**DataTable dt = conn.GetSchema("TABLES")**

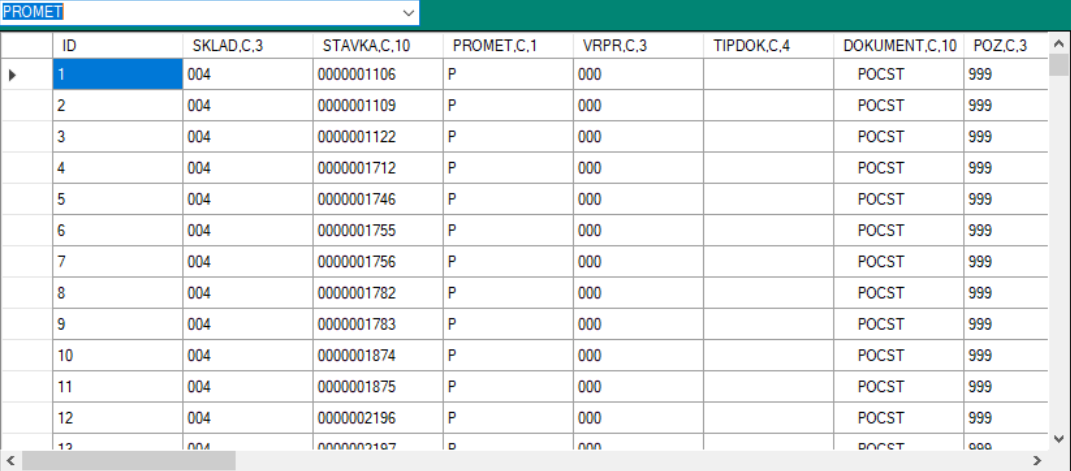
**.AsEnumerable()**

**.Where(x => x.Field<string>("TABLE\_TYPE") == "TABLE")**

**.CopyToDataTable();**

Ezzel a kóddal mondtam meg a programnak, hogy csak a megfelelő tipusú táblázatokat írja ki. A legvégén még az **OP** változó értékét átraktam 1-re, és így nem fogja többet lefuttatni ezt a programrészt és nem fogja ismételten belehelyezni a kombinált szövegbeviteli dobozba. A függvény meghívása a kombinált szövegbeviteli doboz kattintás eseményébe helyeztem el, vagyis a függvényünk akkor fog majd lefutni ha a felhasználó rákattint a kombinált szövegbeviteli dobozra. Viszont ezzel még csak belehelyeztük a kombinált szövegbeviteli dobozba a táblákat, következőre a **ComboBox**-ból kiválasztott táblát kellene megjelenítetni a **DataGridView**-ben, erre is készítettem egy új függvényt, amit **table**-nek neveztem el. Abban az esetben ha szükségem lesz rá akkor akár több helyről is megtudjam hívni. Mivel az adatbázishoz már csatlakozott, meg az adatbázis helyét már megírtam az előző függvényemnél, ezért most csak meg kell nyitnom a **.Open()** metódussal az új függvényemen belül, létrehozzuk az **OleDbCommand cmd = new OleDbCommand()** objectumot, a **cmd.Connection**-ét rákapcsoltam az adatbázis kapcsolatára, amit én általában **conn**-nak nevezek el. Szükség van a kapcsolat lekérdezés szövegére amit SQL-ben van írva. A feladatom csak annyi hogy kiírassam a **DataGridView-**be a táblázatot, szóval nem is nehéz és nem is hosszú a kód. Az SQL **Select** parancsát használom, ami után kifejezem a „\*“ segítségével hogy a tábla összes elemére szükségem van, a **from** szócskával megmondom hogy a következőre beírt érték a Tábla neve, **query = " select \* from " + comboBox1.Text.** A **combobox1.Text** az a kombinált szövegbeviteli dobozban kiválasztott érték, vagyishogy melyik tábla legyen alkalmazva. Ha ez megvan akkor létrehozom a **OleDbDataAdapter da = new OleDbDataAdapter(cmd),** és a tábázatot **DataTable dt = new DataTable();** Majd az adapter **da** segítségével átszállítom az adatokat a **dt**-be és **da.Fill(dt)** paranccsal meg beolvasom.

Legvégül csak meghatároztuk a **DataGridView-**adat állományát **dataGridView1.DataSource = dt,** Majd ezután bezárjuk a kapcsolatot. Mikor a kombinált szövegbeviteli doboz értéke megváltozik akkor az utóbbi függvény lefut és a **DataGridView**-ben megjelenik az adatbázis táblázata.



9. ábra-Tálázat

## 3.3.Adat hozzáadás

Az adat bevitelhez a felhasználónak a második lapon látható „Adat Hozzáadása“ gombbal tudja elkezdeni. Miután a felhasználó rákattintot a gombra a harmadik oldal is megjelenik. Ahhoz hogy hozzátudjunk adni adatot ki kell nyitni a fő oldalon egy táblázatot, ez azért kell mert a táblázatra kattintva tudjuk megmondani a programnak, hogy hova szeretnénk az adatot beszúrni. Mivel a program egy ablakon alkalmazza az adat bevitelt és az adat módosítást, a programnak valahogyan meg kell határozni, hogy a felhasználó éppen melyik folyamatot akarja lefuttatni, ezt a fő oldalon deklarált **i** változó segítségével oldottam meg. Amikor a felhasználó megnyomja a fő oldalon a „Adat Hozzáadása“ gombot akkor az **i** változó értéke az alapértelmezett 0-ról 1 re növekszik, ezt az értéket a deklaráláskor **public-**ra raktam így a mellékablakon is eltudtam érni a fő oldal objectumán keresztül, a mellékoldalon látható „Mentés“ gombbal hozzáférek az **i** változóhoz, **f2.i** . Majd megvizsgálom, hogy az értéke egyenlő-e 1-el, ha igen akkor a program átugrik a **add()** névre keresztelt függvényhez.

**if (f2.i == 1)**

**{ add(); }**

**else if (f2.i == 2)**

**{ mod(); }**

A függvényben először megvizsgálom, hogy a felhasználó által kiválasztott cella mező az ID oszlopában van-e, mert nem lehet a táblázat elsődleges kulcsába értéket elhelyezni, ha a felhasználó az ID mezőt választotta ki akkor egy felugró ablakban a program felszóllítsa a felhasználót, hogy válaszon ki egy másik cellát, **MessageBox.Show("Az tábla elsődleges kulcsához nem tud hozzáadni adatot, próbáld újra!").** Abban az esetben ha az adatbázis táblázatában az elsődleges kulcsot másként írják be, arra meghatároztam még három lehetséges változatát az elsődleges kulcs jelőlésének**, ID , Id, id**, vagy **iD**, neveken szintén ugyanazt a felhívást kapja a felhasználó. Ha a felhasználó jó cellát választott ki akkor a program elmenti a cella és a szövegdoboz adatait. Ehhez két változót hoztam létre , **Data** és **Field** néven, a Datába mentem el a szöveg doboz értékét egy meghatározott formában, **Data += "'" + textBox1.Text + "',".** Ezt a formátumot úgy terveztem, hogy a „Mentés“ gombra kattintva elmentse a Datában a szöveg doboz értékét, azért hogy ne írják az adatok egymást felül a **+=** -t használom, ami annyit jelent hogy nem felül írja a benne lévő értéket, hanem hozzáadja az eddigi értékhez, mivel szövegről vagy **string** értékről beszélünk ezért fogja és összekapcsolja a szöveget, erre majd a későbbiekben szükségünk lesz. Ugyanez történik meg a Field változóval csak itt a cellához tartozó oszlopokat mentsük el.

**Field += "[" + f2.data.CurrentCell.OwningColumn.Name + "],"**

A szögletes zárójelek arra kellenek, hogy ha esetleg az oszlop nevében vessző található akkor az ne zavarja össze a programot. Majd végül hogy a felhasználó lássa, hogy milyen adatot adott hozzá a szöveg mezőbe kiírja őket.

**label1.Text += "Table:" + f2.cb.Text + "\nField:" + f2.data.CurrentCell.OwningColumn.Name + "\nNew Value:" + textBox1.Text + "\n---------------------------\n"**

Az adatokat egészen addig lehet hozzáadni amíg ki nem fogyunk az oszlopokból vagy ha a felhasználó úgy itéli meg, hogy elég volt, ilyenkor a **„Hozzáadás“** gombbal lehet hozzáadni a táblázathoz az adatokat. Ugyanúgy mint az első gombnál le kell ellenőrizni, hogy a felhasználó melyik folyamatott futassa, szintén az **i** változót használjuk fel erre a célra. Ha az **i** értéke megegyezik-e 1-el , ha igen akkor az **add2()** nevű függvényhez ugrik a program. Az adat hozzáadáskor az a probléma állt fent hogy a program nem követte az adatbázis által elkezdett **ID**  felsorolást hanem sajátmagának a saját értékeivel számozta a programot, aminek következtében az adatbázisban ismétlődő értékek jelentek meg, ami pedig nem megengedhető. A probléma megoldását egy **ID** nevű változó létrehozásával kezdtem. Majd **foreach-**ben végig mentem a **DataGridView** összes során**.**

**foreach (DataGridViewRow dr in f2.data.Rows)**

Egy változóba amit **index**-nek neveztem el a következő kóddal megszereztem a sor első oszlopának az értékét, **dr.Cells[0].Value** ,majd megnéztem hogy nagyobb-e mint az én ID változóm, ha igen akkor az **ID** változóm meg kapja az értékét, **ID= Convert.ToInt32(index).**

**var index =dr.Cells[0].Value;**

**if (Convert.ToInt32(index) > ID)**

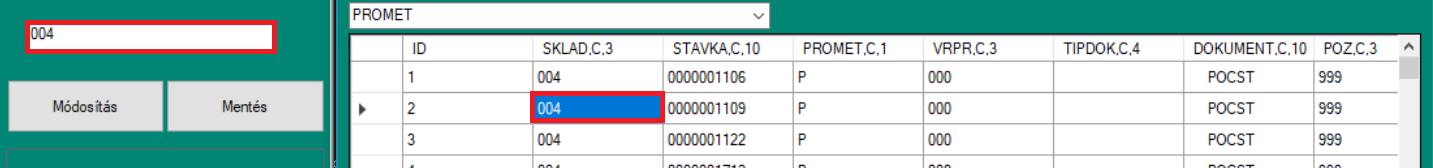
**{ID= Convert.ToInt32(index);}**

Miután végigment az összes oszlopon és meghatározta a legnagyobb értéket, az **ID++ -**t használva megnöveltem 1-el az **ID** értékét. Ezután szükségem volt az Oledb kapcsolatra, mint eddig mindegyik oldalamon **conn-**nak neveztem el, és meghatároztam az adatbázis tartózkodási helyét is, miután ezek megvoltak meg tudtam nyitni a **conn.Open()**-el a kapcsolatot. Az **OledbCommandot** is létrehoztam **cmd** néven. A lekérdezést az SQL **insert** parancsával írtam meg, amiben megadtam a jelenlegi tábla nevét illetve a megfelelő formában lévő **Field** és **Data** változókat.

**"insert into " + f2.cb.Text + "([ID]," + Field + ") values('"+ID+"'," + Data + ");"**

Itt látszik, hogy miért volt fontos a **Field** és **Data** helyes megírása. Az **ID**-t is hozzáadtam, így mindig a legnagyobb ID oszlop érték után fogja behelyezni az értékünket, elkerülve az esetleges dupla értékeket. Az SQL lekérdezést belehelyezük a **cmd.CommandText**-be és **cmd.ExecuteNonQuery()-el,** fejezem a kiíratást. Egy felugró ablakban jelzem a felhasználónknak, hogy sikeres lett a hozzáadás és lezárom a kapcsolatot, conn.Close().A **Field** és **Data** változót kiürítem, hogy a következőre új adatokat lehessen bevinni. Ezután a felhasználó ha akar akkor vihet be mégtöbb adatot vagy akár a „Vissza” gombbal lecsukja az ablakot.

## 3.4.Adat módosítás

Az adat módosítást a felhasználónak a második lapon látható „Adat módosítása“ gombbal tudja elkezdeni. Mikor ez a gomb működésbe lép a program az előbb emített **i** változó értékét 2-re állítja, ezután még meghívja a **func** függvényt ami megnyitja a 3.oldalt és az esetleges ismételt gombnyomásnál lecsukja azt, emellett még végre hajtja az oda vissza helyzet változtatást annak érdekében hogy az oldal a fő oldal mellé legyen csatolva. A felhasználónak szüksége van egy kiválaszott táblázatra, a szövegdobozban az érték autómatikusan változik arra ami a táblázatban a kijelölt mezőben van, így könnyítve meg a felhasználü munkáját az esetleges hosszabb szöveg adat módosítása esetén.

10. ábra-Egyenlő értékek

A felhasználónak ezután csak módosítani kell a szöveg dobozban lévő értéket és a Mentés gombra kattintva a program ismételten meghatározza a felhasználó alatt futtatott folyamatot, hogy az hozzáadás vagy módosítás, ha az **f2.i** megegyezik 2-vel akkor a **mod()** függvény fog lejátszódni, ha az értéke viszont 1 akkor az adathozzáadás fog lefutni. Ez alkalommal is igénybe veszük a **Field** és a **Data** változót, ahol megadjuk nekik a kiválasztott cellához tartozó oszlopot és a szöveg dobozban lévő értéket. Annyi különbséggel, hogy itt a két értéket még egybe helyezem az **LL** névre keresztelt szöveges változóba.

**Field = "[" + f2.data.CurrentCell.OwningColumn.Name + "]";**

**Data = " ' " + textBox1.Text + " ' ";**

**LL += Field + "=" + Data + ",";**

**MessageBox.Show("Addat hozzá adva");**

Ezt a folyamatot a kijelölt sorban a felhasználó addig hajthatja végre amedig van még nem változtatott oszlopa, ezutána „Módosítás” gombbal képes a módosítás folyamatát elvégezni. A program még egyszer leellenőrzi, hogy a felhasználó hozzáadást vagy módosítást végez-e ha igen , vagyis az **f2.i** értéke 2, akkor meghívja a **mod2()** függvényt. Ez alkalommal is kinyitom az Oledb kapcsolatot, elkészítem a **cmd** objectumot, annak a kapcsolat tulajdonságát a **conn-**ra állítom rá majd az SQL kódban megírom a módosítást a fő oldalon lévő lenyíló mező által kiválasztott táblát, az előzőleg elkészített **LL** változót és az éppen kiválasztott sort az index változóban tárolva.

**var index = Convert.ToString(f2.data.CurrentRow.Cells[0].Value);**

**string q = "update "+f2.cb.Text+" set "+LL+" Where ID="+index+" ;";**

**cmd.CommandText = q;**

A kapcsolatot lezárjuk miután a **cmd** parancs tulajdonsága meg kapta a **q** változó által tárolt SQL kódot. A sikeres adat bevitel utána a program egy felugró ablakban jelenti hogy a program sikeresen módosította az adatokat, emellett a program a szövegmezőbe is kiírja a változtatott adatokat, és az **LL,Data** és a **Field** mezőket kiüríti.

## 3.5.Adat törlés

A többi művelettel ellentétben ez a művelet nem a mellék oldalakon megy végbe, ez a fő oldalon játszódik le. A felhasználónak ki kell választania hogy melyik táblából akar kitörölni adatot, miután ez meg van, ki kell választani annak a sornak az elsődleges kulcsát,amit ki szeretne törölni, az elsődleges kulcs általában a táblázat első sora. Abban az esetben ha ezek megvannak akkor rá lehet kattintani az „Adattörlés” gombra, ha a feltételek nem adottak akkor a program felszólítsa a felhasználót hogy válaszon ki egy táblát. Az első dolog amit a program megtesz, hogy megkérdezze a felhasználót biztosan szeretné-e kitörölni az adatot.

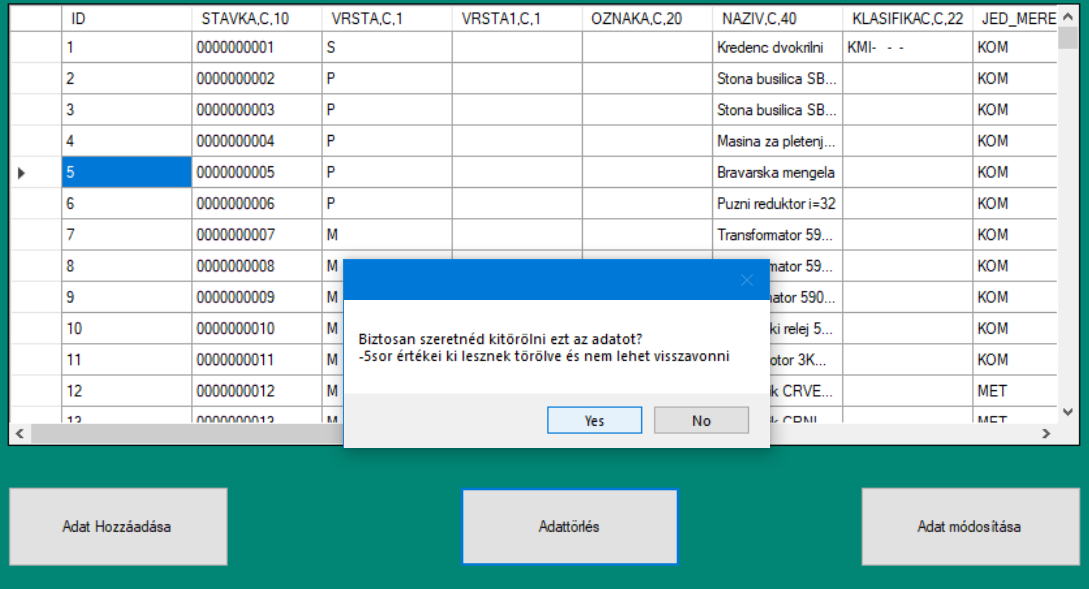
A **YesNo MessageBox** úgy működik hogy a felugró ablakon kérdezett kérdésre igennel vagy nemmel válaszol a felhasználó, ezt az igaz hamis értéket egy változóban eltárolja és egy **if-**el le lehet ellenőrizni hogy a felhasználó melyik választ adta meg.

**DialogResult dialogResult = MessageBox.Show("Biztosan szeretnéd kitörölni ezt az adatot?\n-"+ data.CurrentCell.Value +"sor értékei ki lesznek törölve és nem lehet visszavonni", "", MessageBoxButtons.YesNo);**

**if (dialogResult == DialogResult.Yes)**

**{...}**

**else if (dialogResult == DialogResult.No)**

**{}**

11. ábra-Kérdezés

Miután a program biztos benne, hogy a felhasználó szeretné kitörölni az adatot, ismételten megnyitja az Oledb kapcsolatot és a parancs objectum létrehozása, és a kapcsolatra való ráállítás után egy újabb SQL kódot használunk. Ebben a műveletben a **DELETE FROM** parancsot használom.

**conn.Open();**

**OleDbCommand cmd = new OleDbCommand();**

**cmd.Connection = conn;**

**cmd.CommandText = "DELETE FROM " + cb.Text + " where " + data.CurrentCell.OwningColumn.Name + " = " + data.CurrentCell.Value + ";";**

A **data.CurrentCell.OwningColumn.Name** megmondja a programnak, hogy melyik oszlop van kiválasztva, ennek az értéknek kell lennie az elsődleges kulcsnak. Abban az esetben ha a felhasználó nem elsődleges kulcsot választott ki akkor egy hiba üzenet jelzi neki ezt és megkéri hogy válasszon ki egy másik mezőt.

## 3.6.Előzmények

Ez a funkció arra szolgál hogy az admin felhasználók képesek legyenek követni hogy a többi felhasználó milyen adatokat változtatott meg a múltban, egy raktár adatbázisnál fontos hogy mindig tudják követni hogy ki mikor milyen adatot változtatott meg az adatbázison. Esetleges félreértések megoldására könnyű és egyszerű módszer. A feladathoz létrehoztam egy **History** néven szereplő access adatbázist aminek megadtam egy idő oszlopot, hogy tudjuk mikor lett az adatbázisban változtatva, a változás módját kifejező oszlop a **FUNCTION** névre halgató szöveges oszlop, leírja hogy milyen fajta változás történt, és a változást leíró oszlop a **Changes**. A programon belül a fő oldal és a mellékoldalba létrehoztam egy egy **conn2** Oledb kapcsolást majd egy külön függvényt mind a három folyamatunknak. Először a törlésnek hoztam létre a **delh** nevű függvényt megnyitottam a kapcsolatot a nemrég elkészített adatbázissomal, majd a **cmdh** parancs objectumot is,és rákötöttem a kapcsolatra. Az SQL kódom egy kicsivel hosszabb volt mint eddig.

**string q = "insert into History([Time],[Function],[Changes]) values('" + DateTime.Now.ToString("yyyy-MM-dd h:mm:ss") + "','Delete','DELETED ROW " + data.CurrentCell.Value + " From " + comboBox1.Text + "');";**

**cmdh.CommandText = q;**

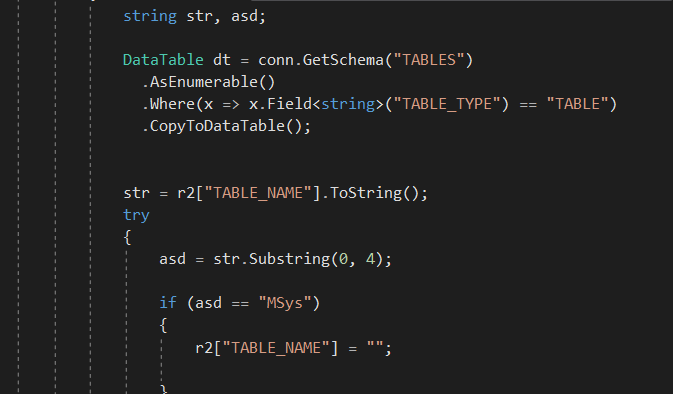
Majd előkészítettem a többi függvényt is, a **addh** és a **modh**, mindekettőn annyit változtattam hogy megfeleljen a feladatához.

## 3.7.Fejlesztési lehetőségek

A programon számos helyen lehet fejleszteni:

* A nyelv és szín beállítások kivitelezése.
* A programnak néha nehezére esik megnyitni az adatbázisokat, ezt szeretném valahogy kiküszöbölni.
* A Beállítások rendes és praktikus kivitelezése
* A hiba üzenetek tartalmazzák még az eredeti hiba üzenetet ami egy átlagos felhasználónak érthetetlen és az esetleges adattolvajoknak értékes információt tud adni.

# **4.Befejezés**

Nagyon érdekesnek és tanulságosnak találtam a programommal a munkát, rengeteg dolgot tanultam belőle nem csak a C#-ról hanem a magáról a programozásról, hogy mennyire fontos, hogy tudjam rendszerezni az függvényeimet, mert a végén nem fogok megtalálni semmit és perceket is elvesztegetek vele, hogy megtaláljak egy egyszerű kis változót. Személyesen nagyon szerettem a táblázattal dolgozni rájönni, hogy melyik tulajdonság mire való és hogyan tudom felhasználni a munkámban.

12. ábra-Kedvenc kódom

# **Ábrajegyzék**

[1. ábra-C++ 4](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572686)

[2. ábra-Visual Studio 4](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572687)

[3. ábra-Bjarne Stroustrup 4](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572688)

[4. ábra-Kezdőlap 5](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572689)

[**5**. ábra-Bejelentkezés 8](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572690)

[6. ábra-Beállítások 9](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572691)

[7. ábra-Fő oldal 10](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572692)

[8. ábra-Mellék oldal 11](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572693)

[9. ábra-Táblázat 13](#_Toc515572694)

[10. ábra-Egyenlő értékek 16](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572695)

[11. ábra-Kérdezés 18](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572696)

[12. ábra-Kedvenc kódom 20](file:///C:\Users\GPFlopi\Desktop\Diplomamunka.docx#_Toc515572697)

# **Felhasznált Irodalom**

* [**https://hu.wikipedia.org/wiki/C%2B%2B**](https://hu.wikipedia.org/wiki/C%2B%2B)
* [**https://www.microsoft.com/net/learn/what-is-dotnet**](https://www.microsoft.com/net/learn/what-is-dotnet)
* [**https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework**](https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework)
* [**https://www.w3schools.com/sql/**](https://www.w3schools.com/sql/)