Debreceni Egyetem

Informatikai Kar

Informatikus könyvtáros szak

Könyvtári alkalmazás fejlesztés

Vezető tanár:

Dr. Boda István

Készítette:

Szabó Adrienn

Informatikus könyvtáros

Matematika tanár

Debrecen

2008.

Tartalomjegyzék

I) Előszó

III. éves hallgatóként kerültem a hajdúböszörményi Pedagógiai Főiskolai Kari Könyvtárba, mint gyakornok. Ezt követően a mai napig kisegítő könyvtárosként dolgozom. Gyakorlatvezetőm lelkiismeretes munkája révén kimerítően megismerkedtem a cédulakatalógus felépítésével és használatával, a gyarapítási, köttetési és törlési folyamatokkal. A tájékoztatásban és kölcsönzésben is alkalmam volt kipróbálni magamat.

Történt, hogy egy hallgató egy elektronikus dokumentumot szeretett volna kikölcsönözni, és nem tudtam, hogy hol találhatom meg. Hiszen katalógust eddig nem vezettek róluk. Ezt jeleztem a felettesemnek, aki szintén meglátta a probléma mértékét. Ekkor kaptam azt a feladatot, hogy készítsek az elektronikus dokumentumokról egy listát. Ezt kezdetben a Microsoft Office táblázatkezelőjében tettem meg. Minden szép és jó volt, csak sajnálattal kellett tapasztalnom, hogy az ebben való keresés nehézkes, és bonyolult.

Ugyanebben a félévben hallgattam az adatbázis-kezelés című órát, ahol az SQL-en volt a hangsúly. Ekkor jött a gondolat, hogy egy adatbázis-kezelő programmal kellene ezt a katalógust elkészíteni, hiszen ezekben a keresés nem túl nehéz, és egy kevés számítógépes ismerettel szinte azonnal elsajátítható. Az engedélyt meg is kaptam, hogy egy ilyen rendszert hozzak létre.

A táblák megszerkesztése nem is okozott gondot, hiszen az egyetemen jelest kaptam a gyakorlaton, illetve még középiskolai tanulmányaim folyamán sikeresen letettem az ECDL 7 modulos vizsgáját, melynek egyik része a Microsoft Acces volt.

A táblák, lekérdezések, űrlapok és jelentések elkészítése után viszont egy kicsit elakadtam és segítségre volt szükségem. Ekkor próbáltam mindenfele érdeklődni, hogy hogyan lehetne tovább haladni, de mindig „falba” ütköztem. Mígnem találkoztam egy főiskolai tanárral, aki erre specializálódott, és megmutatta, hogyan lehet, a műveletek mögé látni. Tanácsára kezdtem el a Visual Basic programozási nyelvet tanulni. Bár elég sok mindent megértek és le tudok írni ennek a nyelvnek a segítségével, de még nagyon az elején járok.

2007 nyarán készültem el a programmal. Mivel ezt a programot nem átemeltem, hanem saját magam alkottam meg – kis segítséggel -, arra gondoltam, hogy ezt szeretném szakdolgozati témaként beadni.

II) Tárgyalás

Mint azt már az előszóban említettem a dolgozatom témája két részből tevődik össze. Elsőként szeretném azt a programot bemutatni, amiben ez az egész katalógus készült.

1. Az Microsoft Access 2003 adatbázis kezelő program

A Microsoft Access egy Windows környezetbe illesztett relációs adatbázis-kezelő rendszer, melynek segítségével könnyedén hozhatunk létre és tarthatunk karban adatbázisokat és adatbázis-objektumokat.

Mivel ennek a szakdolgozatnak nem fő témája az adatbázis kezelő rendszer, ezért az alapvető dolgokat nem kívánom tárgyalni, mint pl.:

- A program indítása
- Új adatbázis létrehozása
- Az adatbázis ablakok
- Adatbázis mentése
- Adatbázis megnyitása
- Adatbázis bezárása és kilépés a programból

Úgy gondolom sokkal fontosabb az *objektumok* és a *műveletek* rövid ismertetése

1.1. Objektumok

Az adatbázis file (amely mdb kiterjesztésű) az alábbi objektumokat tartalmazza:

- adattáblák (tables): az adatbázis adatait tárolja. Az oszlopok nevét *attribútumok*nak nevezzük, és a továbbiakban ezt is használom.
- lekérdezések (queries): az adatbázisban tárolt adatok egyszerű kinyerésére szolgál különböző kérdési formák segítségével.
- űrlapok (forms): az adatbázis kezelő felhasználó barát felülete. Ennek segítségével egyszerűbbé válik az adatbevitel, keresés és módosítás.

- jelentések (reports): egyedi igények szerint kialakított és szervezett információkat tartalmazza.
- makrók (macros): olyan művelet vagy műveletcsoport, amely a feladatok automatizálására szolgál.
- modulok (modules): saját adatbázis-kezelő felület fejlesztésének környezete

A fenn sorolt elemeket (a makrók és modulok kivételével) a felhasználónak kell létrehozni. Ezek létrehozása 2 féleképpen történhet.

- Tervező nézetben: Itt a felhasználó saját maga hozza létre a kívánt objektumot.
- Varázsló segítségével: Ez egy felhasználó barát felület. Használata egyszerűbb, emiatt közkedveltebb. [1]

1.2. Műveletek, relációk

Az adattáblák (*a továbbiakban röviden csak táblák*) közötti kapcsolatokat az oszlopok értékeinek azonosságán keresztül valósítjuk meg. Nagyon fontos a minimális redundancia elve, ami csak a táblák összekapcsolásához szükséges redundanciát tartalmazza. A redundancia szó jelentése: „újabb információt nem adó felesleg a közleményben, amely nélkül azonban a megértés nehezebbé válna”.[2]

Szükséges még megemlíteni a reláció lényegét. A táblák attribútumait jelölje A1, A2,…, AN véges halmaz. Az A1×A2×…×AN Descartes-féle szorzathalmazt jelölje *R*. Ekkor R$\subseteq$*R* esetén R-t relációnak mondjuk és *egyed*nek a reláció, tulajdonságnak az attribútum felel meg.

A relációk elvégzéséhez viszont szükségünk van egy *Kulcs*ra. Ez az attribútumok azon legszűkebb halmaza, amely egyértelműen azonosítja a reláció egy sorát. Szükséges, hogy minden sorban különböző legyen, és mindig ki legyen töltve. Megkülönböztetünk egyszerű és összetett kulcsot aszerint, hogy egy vagy több attribútum alkotja.

A táblákon végezhető műveletek szempontjából 2 fajta műveleti csoportot különböztethetünk meg. Az egyik a relációs algebra műveletei, a másik pedig a halmazműveletek.

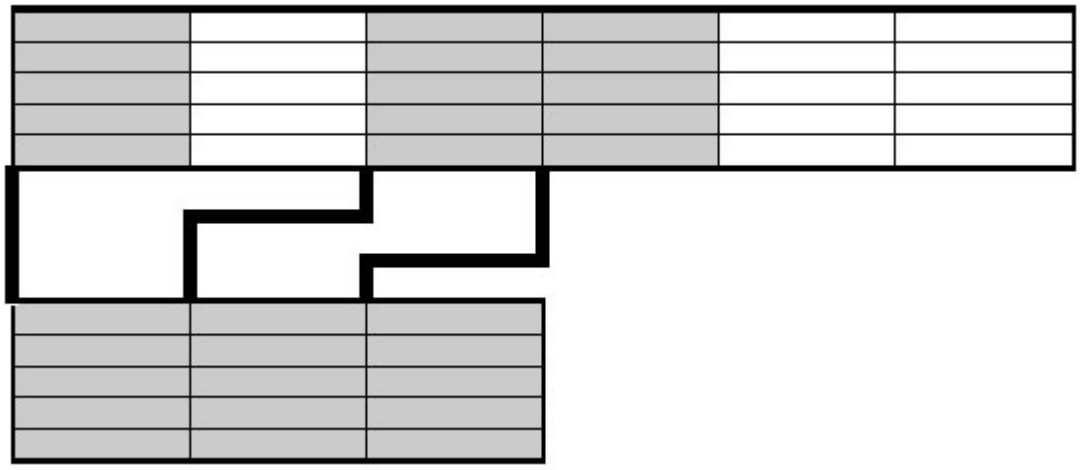
A relációs algebra műveletei:

„A relációkkal kapcsolatban a matematika egy külön ága fejlődött ki. A matematikusok műveleteket definiáltak a relációkra és a halmaz műveleteket is alkalmazták a relációkra.” [3] Röviden e műveletekről.

- Projekció
- Szelekció
- Descartes szorzat
- Természetes összekapcsolás

a. Projekció

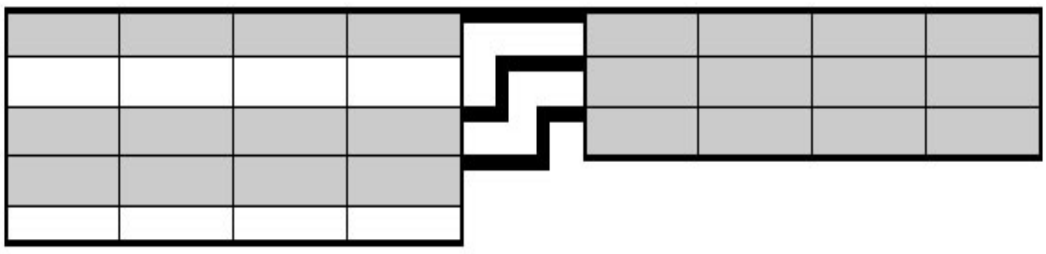
Tetszőleges reláció oszlopokra szűkítését jelenti.



1. ábra: A projekció szemléltetése

b. Szelekció

Tetszőleges reláció sorokra történő szűkítését jelenti.



2. ábra: A szelekció szemléltetése

c. Descartes szorzat

Két reláció sorainak egymás mellé állítása az összes lehetséges kombinációban.

1		
2		
3		
4		

1			
2			
3			

1			1			
2			2			
3			3			
4			1			
1			2			
2			3			
3			1			
4			2			
1			3			
2			1			
3			2			
4			3			

3. ábra: a Descartes szorzat szemléltetése

d. Természetes összekapcsolás

Két reláció természetes összekapcsolásán azt a relációt értjük, amikor a két relációnak csak azon sorait párosítjuk össze, amelyek értékei megegyeznek a két reláció összes közös attribútumaival.

A1	A2
a	b
b	c

A1	A2	A3
a	b	c
b	c	c
c	b	b

A1	A2	A3	A4
a	b	b	c
b	c	c	c

4. ábra: Természetes összekapcsolás szemléltetése

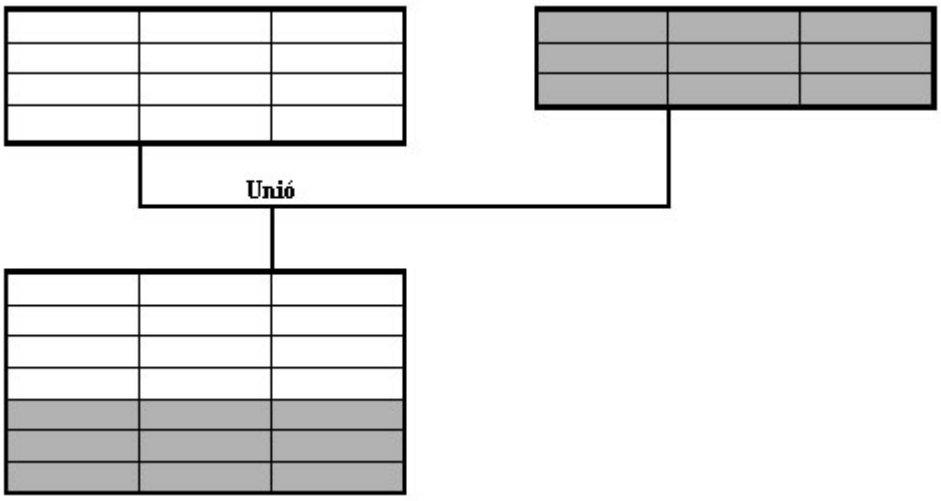
Halmaz műveletek

A halmazokkal kapcsolatos alapvető műveleteket, unió metszet, különbség, a relációkra is értelmezzük. Minden értelmezett halmazművelethez legalább két tábla szükséges, viszont a különbség esetében több nem lehet. Ezek a műveletek azonos dimenziójú relációk között hajthatók végre, ami azt jelenti, hogy a műveletbe bevont két reláció oszlopainak meg kell egyeznie az attribútumokban és a tárolt adat típusában is. Röviden ezekről, a műveletekről.

- Unió
- Különbség
- Metszet

a) Unió

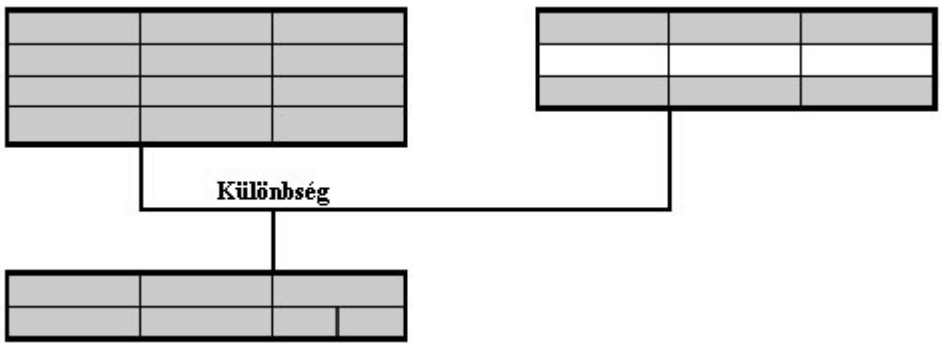
A két reláció egyesítését jelenti. Ekkor a két reláció dimenziója megegyezik az unió által kapott reláció dimenziójával.



5. ábra: Az unió szemléltetése

b) Különbség

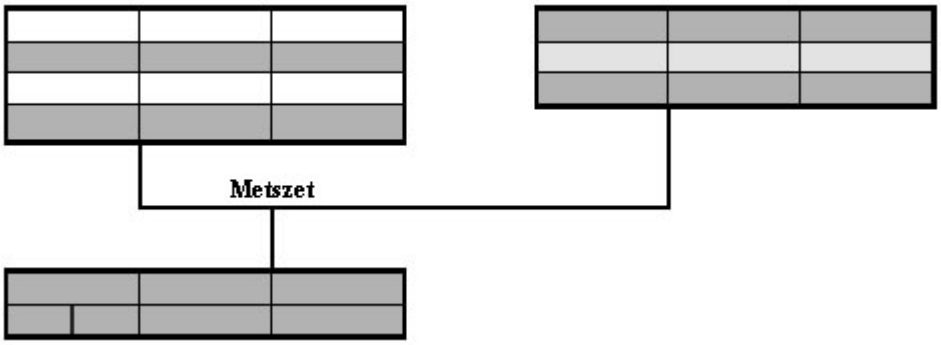
Különbség alatt azt a relációt értjük, amikor az első relációból kivesszük azokat a sorokat, amelyek a másodikban szerepelnek.



6. ábra: A különbség szemléltetése

c) Metszet

2 vagy több azonos szerkezetű reláción végezhető. Az eredmény halmaz azokat az sorokat fogja tartalmazni, amelye mindegyik relációban változatlanul szerepel. [4]



7. ábra: A metszet szemléltetése

1.3. Az adatbázis adatmodellje

A feladat során a következő egyedek és tulajdonságok adatai voltak szükségesek.

- Formátum

A hangzó dokumentumok könnyebb azonosítása érdekében a következő tábla a formátumot tartalmazza. Ez azért nagyon fontos, mert a különböző formátumú dokumentumokat különböző helyen szeretné tárolni a könyvtár:

Elnevezés	Tárolási típus	Leírás
Formátumazonosító	Szám	A formátumokat egyedi azonosító számmal láttam el. (új formátum megjelenése esetén automatikusan új számot kap)
Formátum	Szöveg	A formátum elnevezése

1. Táblázat: A Formátum tábla felépítése

- Dokumentumok

A hangzó dokumentumok tárolására egy olyan táblára van szükségem, ahol minden információt megtalálhatok, ami egy adott egyedet jellemez.

Elnevezés	Tárolási típus	Leírás
Nyilvántartási szám	Szám	Ez az elsődleges kulcs. A sorszámokat a leltárkönyvben kapott száma határozza meg.
Nyilvántartási azonosító	Szám	Ezt a nyilvántartási számból képezzük egy 'EH-' eléírásával. Erre azért van szükség, mert a könyvtár ezzel az előtaggal különbözteti meg az Egyéb Hangzó dokumentumokat.
Szerzők	Szöveg	A szerző(k) neveit tartalmazza

Cím	Szöveg	A dokumentum címe
Formátum kódja	Szám	A formátum kódja
Média sorszáma	Szám	A dokumentum formátumonkénti csoportosításkor kapott száma. Erre azért van szükség, mert ez fogja megmutatni a polcon elfoglalt helyét.
Darab	Szám	Megmutatja, hogy az adott dokumentum mennyi egységből áll.
Törölve	Szöveg	Ha ez a rész ki van töltve, akkor a polcokon való keresés felesleges. Tartalmazza a törlési jegyzék számát
Megjegyzés	Szöveg	Itt bármilyen megjegyzés tehető. Én a mellékletek esetén a fődokumentum leltári számát tüntettem fel.
Melléklet	Igen/Nem	A logikai négyzet kipipálása azt mutatja, hogy a szóban forgó dokumentum valami melléklete
Tok nélküli	Igen/Nem	Erre azért van szükség, mert sok melléklet tok

		nélkül, vagy papír tokban érkezett. Ezek pedig kezelhetetlenek. A rendszerezésükre egy CD tartó tokot állandósítottunk.

2. Táblázat: A dokumentum tábla felépítése

2. A Visual Basic programozási nyelv

2.1. Kialakulása, története

A BASIC (Beginner's All-purpose Symbolic Instruction Code) programozási nyelv félig hazai találmánynak mondható, hiszen 1964-ben Kemény János és Thomas Kurtz fejlesztette ki a Dartmouth College-ben.

A nyelv alapjául a Fortran programozási nyelvet vették. A BASIC egy interpretált nyelv, ami azt jelenti, hogy a gép egy saját maga által használt nyelvre fordítja le a programot annak futása közben.

Elterjedése érdekében kitüntetett szerepet kaptak a ’80-as évek, amikor a számítógép térhódítása kezdődött meg. Ugyanis a gépeket egy BASIC értelmezővel adták, amelyek már be voltak építve a gépekbe. A probléma csak ott kezdődött, hogy a különböző típusokon megírt programok inkompatibilisek voltak. Azaz nehézkes volt az egyszerre alkalmazás. Ez szinte mindig egy átírást igényelt.

A könnyebbséget a DOS megjelenése és elterjedése jelentette, hiszen az újabb verzióiba, már beépítésre került a BASIC Qbasic változata. A Qbasic nyelvvel az volt a probléma, hogy a futtatható fájl létrehozására nem volt képes.[5]

A Visual Basic (VB) nem csak egy programozási nyelv, hanem egy fejlesztői környezet is, melyet a Microsoft cég tökéletesített.

A VB az eredeti BASIC nyelvből alakult ki. 1991-ben látott napvilágot a 1.0-ás verzió, amely a Windows 3.0 platformra készült. Eseményvezérelt valamint korlátozott mértékben objektumorientált. Az eseményvezérelt nyelv azt jelenti, hogy a kód egyes részei események hatására lépnek működésbe. Pl.: kattintás. A nyelv nagy sikereket ért el, hiszen egy olyan kényelmes és hatékony eszközzé vált, amellyel a fejlesztők már vizuálisan tudták tervezni a Windows alkalmazásokat. De nem csak a fejlesztők, hanem a felhasználók is nagy kedvencévé vált. Sokan kezdték ezzel a nyelvvel a programozási tanulmányaikat, hiszen egyszerűen, gyorsan lehetett tanulni, és mint már azt az előbb is említettem, hatékonyan lehetett alkalmazni. Ezért a Microsoft a minél inkább a fejlesztése mellett döntött, hogy ezáltal még jobban tudja a felhasználói feladatát könnyíteni.

Viszont akármennyire is egyszerű és gyors a VB nem tudta felvenni a versenyt a C++-szal. Ennek oka, hogy a VB nyelvi eszközei korlátozottabbak:

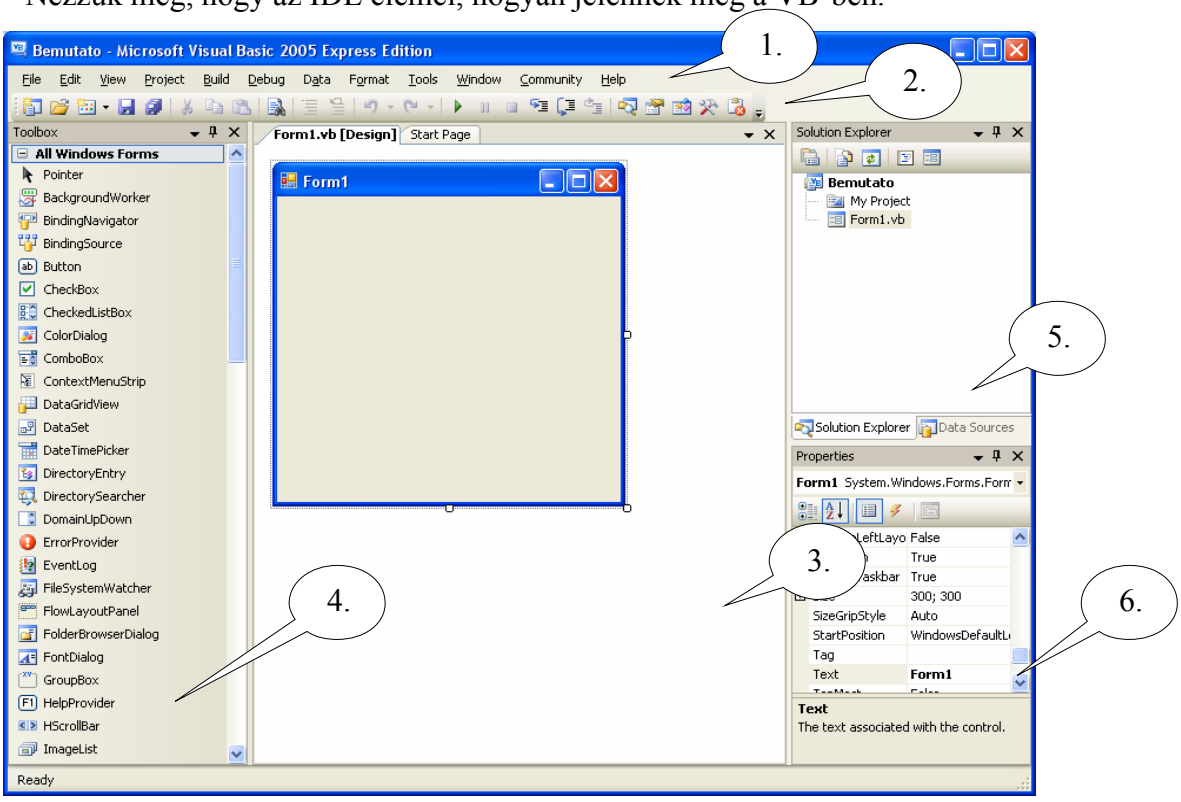
- nem támogatja a mutatókat
- nem kötelező benne a változók deklarálása
- nem kezeli szigorúan a típusokat.

Ez a gyors fejlesztés szempontjából előnyös, viszont rontja a program hatékonyságát.[6]

2.2. Felépítése

A VB vizuális tervező munkafelületét integrált fejlesztőkörnyezetnek (Integrated Devolopment Environment röviden: IDE) nevezzük. Ez azt jelenti, hogy egy közös környezetben sok funkció van egyesítve.

Nézzük meg, hogy az IDE elemei, hogyan jelennek meg a VB-ben. [7]



8. ábra: AVisual Basic grafikus felülete

1. Menüsor. Itt találhatóak a VB-ben történő munkához szükséges parancsok. Ha jól megfigyeljük, a szokásos funkciókon kívül olyan menüket is találhatunk, amelyek kifejezetten a programozás segítik elő. Ezek a *Project* és a *Format.*
2. Eszköztárak. Itt olyan gombokat találunk, amelyek a menüsorban is szerepelnek. Szerepük a gyorsabb elérés.

3. Űrlaptervező (Form Designer). Az alkalmazás kezelőfelületének megtervezésére szolgál. Fontos tudnunk, hogy minden alkalmazáshoz külön űrlaptervező ablak tartozik.
4. Eszközkészlet. (Toolbox). Abban nyújt segítséget, hogy a különböző vezérlőelemeket a lehető legegyszerűbben tudjuk elhelyezni a Form ablakon.
5. Projektböngésző (Solution Explorer). Itt találhatjuk meg azokat az űrlapokat és modulokat, melyeket a projektünk kezel. Ez azért nagyon fontos, hiszen egy projekt több fájlt is tartalmazhat. Így könnyebben átláthatjuk ezeket. Itt tudunk továbbá további projektfájlokat hozzáadni a programunkhoz, illetve ezeket átnevezni.
6. Tulajdonságok (Properties). A projekt felületek tervezése során itt tudjuk beállítani azok tulajdonságait. Pl.: méret, szín, stb.[8]

2.3. A programkód készítésének alapjai

Egy programrendszer, programozási nyelv megismerésekor a következő dolgok ismerete elengedhetetlen:

- Milyen típusú adatokat kezel a rendszer
- Milyen műveleti jeleket (operátorokat), kifejezéseket tartalmaz
- Milyen függvényekkel rendelkezik
- Hogyan kell értelmezni (esetleg írni) egy programnyelvi utasítást, eljárást.

Az alábbi táblázat az Access 2003 táblamező adattípusokat tartalmazza:

Adattípus	**Tartomány, megjegyzés**
Szöveg (Text)	Tetszőleges karaktersor, maximum 255 karakterig
Feljegyzés (Memo)	Tetszőleges karaktersorozat 1-től kb. 64000 karakterig
Szám (Number)	A szám adattípus részletezése a külön táblázatban található.
Dátum/Idő (Date /Time)	1100. január 1.-től 9999. december 31.-ig terjedő dátum és/vagy idő.
Pénznem (Currency)	Pénzügyi számítások adattípusa: -922377203685477580 8-tól

	9223372036854775807-ig.
Számláló (AutoNumber)	Egyedi sorszámozás és véletlenszerű számérték az adott mezőbe. Hosszú egész vagy Többszörözési azonosító mezőméret adható hozzá
Igen/Nem (Yes/No)	Két lehetséges érték
OLE objektum (OLE Object)	Csatolt vagy beágyazott OLE objektum (táblázat, videó, stb,)
Hiperhivatkozás (Hyperlink)	Fájl elérési út vagy internet cím
Keresés varázsló (Lookup Wizard)	Nem adattípus. Más táblából vagy listából kiválasztható adatok.

3. Táblázat: az Acces 2003 táblamezői

A szám adattípus mezőméret szerinti megkülönböztetése:

Adattípus – mezőméret	**Leírás**	**Méret**	**Tartomány**
Bájt (Byte)	Előjelnélküli szám	1 bájt	0-tól 255-ig
Egész (Integer)	Fixpontos egész szám	2 bájt	-32768-tól 32767-ig
Hosszú egész (Long Integer)	Fixpontos egész szám	4 bájt	-2147483648-tól 2147483647-ig
Egyszeres (Singel)	Egyszeres pontosságú lebegőpontos szám	4 bájt	-3,402823E38-tól -1,401298E-45-ig negatív értékekre; 1,401298E-45-től 3,402823E38-ig pozitív értékekre
Dupla (Double)	Dupla-pontosságú lebegőpontos szám	8 bájt	1,79769313486232E308-től -4,94065645841247E-324-ig negatív értékekre; 4,94065645841247E-324-tól 1,79769313486232E308-ig pozitív értékekre

Többszörözési azonosító (Replication ID)	Adatbázis többszörözés esetén használatos a kópiák szinkronizálására szolgáló egyedi azonosítók	16 bájt	

4. Táblázat: A Visual Basic számtípusai és jellemzői

Az Access 2003 adatbáziskezelő-rendszerben a VBA szűkített (a Visual Basic programozási nyelv szűkítése) programozási nyelv ismeretében írhatunk, értelmezhetünk programkódot. A programkód különböző modulokban (programokban) jelenik meg. A program utasítások sorozata. Ezeknek az utasításoknak az írásakor eleget kell tennünk az adott programozási nyelv formai és nyelvtani előírásainak.

A dolgozat terjedelmét szem előtt tartva, csak röviden összegezem azokat azt minimális tudásanyagot, amely elengedhetetlen ahhoz, hogy egy modulban megjelenő programkódot megértsünk, és ezek alapján egyszerűbb utasításokat is megfogalmazzunk.

2.3.1 Szintaktikai elemek

Az utasítások egyik leggyakrabban használt típusa az értékadó utasítás, amely tartalmaz egy "=" operátort. Az egyenlőségjel bal oldalán áll a változó, a jobb oldalán pedig az az érték, vagy kifejezés, amit ennek a változónak akarunk adni. A változó – ugyanúgy mint a Javaban - egy névvel azonosított memóriabeli tároló helyet jelent, mely olyan adatot tartalmaz, amely a program végrehajtása közben bármikor módosítható. Így a változónak van egy egyedi neve, amely egyértelműen meghatározza a hatókörén belül. A változó adattípusának megadása (deklarációja) nem kötelező.

Például az következő utasítás az *nVálasz* nevű változóba helyezi az *MsgBox ()* belső függvény által visszaadott értéket:

nVálasz = MsgBox (" Kriktikus hiba!", vbAbortRetryIgnore + vbCritical,_-

"Üzenet a felhasználónak")

A fenti példában jól megfigyelhető, hogy egy utasítást több sorban is elhelyezhetünk. A sorfolytató jel két karakter, egy szóköz és egy aláhúzás jel " _". A sorfolytató jelet követően megjegyzést nem írhatunk. A megjegyzés egyébként az utasítás végén, vagy új sorban is tehető. Ennek viszont aposztróf (') jellel kell kezdődnie.

Lehetőség van viszont arra is, hogy egy sorba több utasítást is kiadjunk. Ebben az esetben az utasításokat a kettőspont karakterrel kell elválasztani. (":")

2.3.2 Változó, tömb deklarációk

Minden változót használata előtt definiálni kell. Ekkor megadhatjuk az adattípusát is. Ezt az eljárást deklarációnak nevezzük. A Visual Basicben ez a művelet leggyakrabban a Dim utasítással történik, a következő formában:

Dim változónév [As adattípus]

Amennyiben a deklarálás nem a használat előtt történik, hanem egy eljárás belsejében, akkor azt a változót csak és kizárólag abban az eljárásban használhatjuk. Vagyis az érvényessége csak helyi.

A változó deklarálására két kulcsszó is alkalmas, amely funkciójukat tekintve mást jelentenek. A *Public* kulcsszóval deklarált változó érvényessége az alkalmazásra terjed ki. A *Static* kulcsszóval definiált változó azzal a tulajdonsággal rendelkezik, hogy az utasítás végrehajtása után az értékét megőrzi.

Ha a modul deklarációs részében (a modul eleje) definiálunk a *Dim* utasítással változót, akkor az modul szintű lesz. Ez azt jelenti, hogy érvényessége a modul összes eljárására kiterjed. A *Dim* utasítás helyett viszont használjuk még a *Private* kulcsszót is. Ennek az értelme ugyan az.

Amennyiben olyan változót használunk, amit előzőleg nem definiáltunk, akkor a Visual Basic azt egy Variant adattípusú változóként értelmezi.

De ez csak abban az esetben tehető meg, ha a modul deklarációs részében nem helyeztük el az *Option Explicit* utasítást. Ekkor ugyanis használat előtt kötelező a változók definiálása.

A tömb egyszerű hivatkozást biztosít változók (tömbelemek) sorozatára. A tömbelemet egy (vagy több) szám (index) segítségével adjuk meg. Ennek megfelelően megkülönböztetünk egy dimenziós (egy index), kétdimenziós (két index: sor, oszlop) stb. tömböket. A Visual Basic tömbelemként megengedi tömb használatát is. A tömbök (indexek) alsó és felső korláttal rendelkeznek. A tömbindexek folytonosak (mint egész szám) a két korlát között. A tömb minden eleme azonos adattípusú. Variant adattípus esetén természetesen ez jelentheti azt, hogy az egyes tömbelemek másfajta adatot tartalmazhatnak.

A tömbök deklarációja megegyezik a változókéval. Típusukat tekintve viszont beszélhetünk statikus és dinamikus tömbökről.

Fixméretű vagy statikus tömbök esetén a definícióban megadjuk az alsó és felső korlátot. A következő példában egy egydimenziós, 16 elemű (0-tól 15-ig), tömböt definiálunk, amely szöveges adattípusú:

Private aNevek (10) As String

Ebben az esetben az alsó korlát 0. A következő példában egy kétdimenziós (mátrix), 12 elemű, egyszeres pontosságú adatokat tartalmazó tömböt definiál:

Dim aMarix (3 To 5, 2 To 5) As Single

Elsőként a sorindexet adjuk meg, ami 3 és 5 között van. Ezt követően az oszlopindex megadása következik, ami ebben az esetben a 2 és 5 közé esik.

A dinamikus tömb deklarációjában csak a tömbnevet adjuk meg, majd futásidőben újradeklarálva mindig csak a szükséges helyfoglalás hajtódik végre. A következő példa egy dinamikus tömböt definiál:

Dim aTomb ()

Fontos, hogy a futásidőben a szükséges méretet be kell állítani. Ezt akkor kell megtenni, amíg nem hivatkozunk rá. Ha az intI változó pillanatnyi értéke 20, akkor az alábbi újradefiniálás után a tömb 21 elemét használhatjuk:

ReDim aTomb (intI)

Az ismételt deklaráció hatására a tömb elveszíti előző értékeit. Amennyiben azt akarjuk, hogy a tömbben eddig tárolt elemek is megmaradjanak, akkor a *ReDim* után a *Preserve* kulcsszó használatát követeljük meg.

Ha azt szeretnénk elérni, hogy a felső korlát magasabb legyen, akkor az Ubound() függvényt kell használnunk. Ez az utasítás 1-gyel növeli a felső korlát értékét, mégpedig úgy, hogy megtartja az addig definiált tömbelemeket.

ReDim Preserve aTomb (Ubound(aTomb)+1)

2.3.3 Konstans deklarációk

Azok az értékeke, amelyek függnek az adattípustól, konstansként definiáljuk. Ezek értéke a definiálást követően nem változtatható. A konstansok lehetnek beépítettek, de magunk is deklarálhatunk. Ha eldöntöttük, hogy melyiket szeretnénk használni, akkor a kifejezés értékére a konstans nevével hivatkozva tudjuk meghívni.

[Public] Private] Const konstansnév[As típus]=kifejezés

2.3.4 Vezérlési szerkezetek

A vezérlési szerkezetek törést jelentenek az utasítások egymás utáni végrehajtásában. Vezérlési szerkezetekről beszélünk elágazások és ciklusok estén. Az elágazások alkalmazásakor egy meghatározott feltétel teljesüléséhez közi az utasítás lefutását. A ciklusok pedig olyan utasításcsoportot jelent, amelyet többször meg szeretnénk ismételni. Az ismétlések számának meghatározása itt is feltételhez köthető. A feltétel mindig logikai típusú kifejezés. Amíg a feltétel Igaz (True) értéket vesz fel

lefut a program, ha viszont Hamis (False) értékkel rendelkezik, akkor már nem fog végrehajtódni az utasítás..

Feltétel nélkül ugró utasítás végrehajtására a *Go To* címke szolgál Használata során megjelölünk címkével egy programsort. A program végrehajtáskor ide érve automatikusan a megjelölt programsorhoz ugrik. Ez a címke egy tetszőleges azonosító, amelyet kettősponttal zárunk le

Elágazások alatt több féle szerkezetet is értünk. Ezek a következők:

- If…Then szerkezet

If feltétel Then utasítás	*If feltétel Then*
	utasítások
	End If

5. Táblázat: Az If…Then szerkezet szintaktikája

A feltétel teljesülése esetén a program végrehajtja a *Then* utáni utasítást, vagy a *Then* és *End If* közötti utasításokat

- If…Then…Else szerkezet

If feltétel1 Then
[utasítások-1]
[ElseIf feltétel2 Then
[utasítások-2]]
[ElseIf feltétel3 Then
[utasítások-3]]
...
[Else
[utasítások-n]]
End If

6. Táblázat: Az If…Then…Else szerkezet szintaktikája

A program a *feltétel1* vizsgálatával kezdődik. Ha ennek az értéke igaz, akkor az *utasítások-1* kerül végrehajtásra, majd az *End If*-re ugrik, amivel befejezi ennek az elágazásnak működését

Ha a *feltétel1*-re hamis értéket kapunk, akkor a program megvizsgálja a *feltétel2*-t. Ennek teljesülése esetén az *utasítások-2* hajtódik végre., majd az *End If*-re ugrik és tovább folytatja a program működését.

Ha a *feltétel2* nem teljesül, akkor a kiértékelés az előző megfontolás mintájára folytatódik. Amennyiben van *Else* ág, és addig minden feltétel kiértékelése hamis értéket adott, akkor az itt szereplő utasítások kerülnek végrehajtásra. Mivel az *Else* ág megadása nem kötelező, ezért megtörténhet, hogy az *If...Then...Else* szerkezetből semmi nem hajtódik végre.

- Select … Case szerkezet:

Select Case tesztkifejezés
[Case kifejezéslista1
[utasításblokk-1]]
[Case kifejezéslista2
[utasításblokk-2]]
...
[Case Else
[utasításblokk-n]]
End Select

7. Táblázat: A Select…Case szerkezet szintaktikája

Speciális, többágú elágazást tesz lehetővé a *Select ... Case* szerkezet. Először a *Select Case* utáni *tesztkifejezés 1* kiértékelésre. Amennyiben ez az érték a kifejezéslista1 értékkel egyezik meg, akkor az *utasításblokk-1* kerül futtatását követően, az *End Select* után folytatódik a program.

Ha a tesztkifejezés értéke nem egyezik meg a *kifejezéslista1* értékével, akkor az utasításblokk-1 kimarad és a 2. Case ágban feltüntetett utasításblokk-2 megvizsgálásával folytatódik. Ennek teljesülése estén az utasításblokk-2 kerül végrehajtásra.

Ez a meggondolás folytatódik tovább, ha a kifejezéslista2 sem mutatna igaz értéket. Ez a vizsgálat mindaddig folytatódik, míg nem talál egy igaz értéket, vagy nem marad már több elágazásunk. Amennyiben egyik ág sem hajtható végre adott feltételek mellett, a program egyszerűen kihagyja, vagy a Case Else ágat hajtja végre. Bár ennek megadása szintén nem kötelező.

Ciklusok esetében is több féle ciklusról beszélhetünk

- Do … Loop szerkezet:

Elöltesztelős:	Hátultesztelős:
Do While feltétel *utasítások* *Exit Loop* *Utasítások* *Loop*	*Do* *Utasítások* *Exit Loop* *Utasítások* *Loop While feltétel*

8. Táblázat: A Do…Loop szerkezet szintaktikája elöl- és hátultesztelős esetekben

Az elöltesztelős szerkezet esetében megeshet, hogy nem történik futtatás. Ez abban akkor fordul elő, amikor a feltétel elsőre hamis értéket ad. Ezzel szemben a hátultesztelős szerkezet egyszer mindenképp lefut, mert az utasítás végrehajtását követően fogja a feltételt megvizsgálni. Ha a feltételek igaz értékre értékelődnek ki, akkor a ciklusmag utasításai végrehajtásra kerülnek mindaddig, amíg a feltétel fennáll.

A *While* kulcsszó helyett az *Unt*il is használható.

- For … Next szerkezet:

For ciklusváltozó = kezdőérték To végérték [Step lépésköz] *utasítások* *Exit For* *utasítások* *Next [ciklusváltozó]*

9. Táblázat: A For…Next szerkezet szintaktikája

A For … Next szerkezetű ciklust akkor használjuk, ha előre tudjuk, hogy hányszor kell a ciklusmagban levő utasításokat végrehajtani. A kezdőérték, végérték, lépésköz numerikus típusú konstans vagy kifejezés. A ciklusváltozó numerikus típusú változó. Az utasításban a Step nem kötelező. Ennek hiányában a lépésköz értéke 1.A lépésköz adja meg a program számára, hogy a ciklus befejezésekor mennyivel változtassa meg a kezdőértéket. Ekkor a program megvizsgálja, hogy ez a megváltozott kezdőérték a megadott végértéktől kisebb-e. Amennyiben igen, a ciklus tovább folytatódik. Viszont, ha hamis értéket kapunk, akkor a ciklus nem hajtódik végre és átugrik rajta.

- For Each … Next szerkezet:

For Each csoportelem In csoport *utasítások* *Next csoportelem*

10. Táblázat: For Each…Next szerkezet szintaktikája

Ez egy speciális ciklus. Futtatása során az In mögött megadott objektumtömb vagy gyűjtemény elemein hajtja végre a ciklusmagban felsorolt utasításokat.

2.3.5 Eljárások

A programozási feladatot általában kisebb logikai egységekre, komponensekre osztjuk. Ezeket a komponenseket eljárásoknak nevezzük. Az eljárásokat általában az ismétlődő vagy közös feladatok megvalósítására szervezzük.

A Visual Basic az alábbi eljárástípusokat kezeli:

- A Sub típusú eljárás, amely nem ad vissza értéket.
- A Function típusú eljárás, amely visszaad egy értéket.
- A Property típusú eljárás, amely definiálhat saját objektum tulajdonságot, visszaadhat objektum tulajdonság értéket, vagy létrehozhat objektumhivatkozásokat. Ezt osztálymodulokban használjuk.

- A Sub típusú eljárás:

[Private] Public][Static] Sub eljárásnév (argumentumlista) *utasítások* *End Sub*

11. Táblázat: A Sub típusú eljárás szintaktikája

Az argumentumlistában a változók definiálása történik. Mégpedig azoké, amelyek fogadják a hívó részről átadott paramétereket. Alapértelmezés szerint az argumentumok adattípusa Variant. A teljes szintaktika egy argumentumra a következő:

[Optional] FbyVal | By Ref] [ParamArry] változónév [()] [As adattípus] [=érték]

Argumentumrész	Leírás
Optional	Nem kötelező. A kulcsszó azt jelzi, hogy egy argumentum nem kötelező. Ha az argumentum nem kap paramétert, akkor Empty az értéke. Az IsMissing () függvény segítségével vizsgálható, hogy volt paraméterátadás vagy sem. A nem kötelező argumentum számára definiálható kezdő érték is, amit akkor vesz fel, ha nem kap paramétert.
ByVal	Nem kötelező. Azt eredményezi, hogy az argumentum értékkel adódik át.
ByRef	Nem kötelező. Azt eredményezi, hogy az argumentumhivatkozással (memória címmel) adódik át. ByRef az alapértelmezett Visual Basicben.
ParamArray	Nem kötelező. Variant típusú, határozatlan számú elemet adhatunk át az argumentumlista utolsó argumentumú Optional tömbjébe. A ParamArray kulcsszó támogatja határozatlan számú paraméter átadását tömbbe. Nem használható a ByVal, ByRef, és Optional kulcsszavakkal.
változónév	Kötelező. A változó neve, ami fogadja az átadott paramétert.
adattípus	Nem kötelező. Megengedett adattípusok: Byte, Boolean, Integer, Long, Currency, Single, Double, Decimal (not currently supported), Date, String (változó hosszúságú csak), Object, Variant. Ha a paraméter nem Optional, akkor felhasználói adattípus (a Type utasítással definiált) vagy OLER automatizmus típus (Application, Sheet stb.) is megengedett.
érték	Nem kötelező. Bármely konstans vagy konstans kifejezés. Csak az Optional kulcsszóval együtt használható.

12. Táblázat: Az argumentumlista magyarázata

Kétféle eljárást különböztetünk meg. Az egyik az általános jellegű, míg a másik az eseményvezérelt eljárások. Az általános jellegű eljárásokat megfelelő szintaktikával meg kell hívni, hogy a vezérlés az eljárásra kerüljön. Az eseményvezérelt eljárások viszont valamely esemény bekövetkezésére (pl. egérkattintás) kerülnek végrehajtásra. Az alábbi táblázat a Form objektumok és egyéb vezérlés objektumok eseményvezérelt eljárásnév-adási szintaktikáját mutatja:

Vezérlés események szintaktikája	Form események szintaktikája
Private Sub vezérlésnév-eseménynév (argumentumok) utasítások *End Sub*	*Private Sub Form-eseménynév (argumentumok) utasítások* *End Sub*

13. Táblázat: Az eseményvezérelt eljárásnév-adás szintaktikája

Az általános jellegű eljárások meghívása kétféle képen valósítható meg:

Call Eljárásnév (paraméterlista)	Eljárásnév paraméterlista

- A Function típusú eljárás:

[Private [Public][Static] Function eljárásnév (argumentumlista) [As típus] *utasítások* *End Function*

14. Táblázat: A Function típusú eljárás szintaktikája

Az argumentumok azon változók definíciói, melyek fogadják a hívó részről átadott értékeket. A típus argumentum a függvényérték adattípusát jelenti, amit az *eljárásnév = kifejezés* értékadó utasítással adhatunk meg.

2.3.6 Osztály, objektum

Az osztály és objektum hasonló fogalmak, de nem teljesen azonosak. Az osztály tartalmazza azokat az információkat, hogy egy objektum milyen tulajdonságokkal rendelkezik. Tehát az osztály egyféle tervrajza egy objektumnak.

Az objektum örökli az osztály jellemzőit, viselkedését. Az objektum rendelkezik tulajdonságokkal, jellemzőkkel, attribútumokkal.

Amikor egy objektumhoz létrehozunk egy eseményvezérelt eljárást, akkor az Access 2003 hozzáad egy eseményvezérelteljárás-sablont az űrlapmodulhoz vagy jelentésmodulhoz. Ezt követően azt a programkódot kell megadnunk, amely az általunk kívánt módon válaszol az esemény bekövetkezésére.

A következő táblázat az események betűrendes listáját tartalmazza:[9]

Esemény	Eseménytulajdonság
Activate	Aktiválásra
AfterDelConfirm	Megerősítés törlés után
AfterInsert	Beszúrás után
AfterUpdate	Frissítés után
ApplyFilter	Szűrésre
BeforeDelConfirm	Megerősítés törlés előtt
BeforeInsert	Beszúrás előtt
BeforeUpdate	Frissítés előtt
Change	Módosításra
Click	Kattintásra
Close	Bezárásra
Current	Jelenlegire
DblClick	Dupla kattintásra
Deaktivate	Deaktiválásra
Delete	Törlésre
Enter	Belépésre
Error	Hibára
Exit	Kilépésre
Filter	Szűrőre
Format	Formázásra
GotFocus	Fókusz vételekor
Initialize	Nincs
IrtemAdded	Nincs
ItemRemoved	Nincs
KeyDown	Billentyű lenyomásra
KeyPress	Billentyű leütésre

KeyUp	Billentyű felengedésére
Load	Betöltésre
LostFocus	Fókusz elvesztésekor
MouseDown	Egérgomb lenyomására
MouseMove	Egérmozgásra
MouseUp	Egérgomb felengedésére
NoData	Ha nincs adat
NothlnList	Ha nincs a listában
Open	Megnyitásra
Page	Oldalra
Print	Nyomtatásra
Resize	Méretezésre
Retreat	Visszatérésre
Terminate	Nincs
Timer	Időzítésre
Unload	Kiürítésre
Updated	Frissítésre

15. Táblázat: A Visual Basic eseményeinek betűrendje és ezek jelentése

3. A program ismertetése

A program elkészítése során elsőként az Accesben hoztam létre a számomra szükséges táblákat, mezőket, lekérdezéseket. Ezek kódjait a program automatikusan megírta. Ennek megírása nem lett volna lehetetlen, de elég nagy munkát igényel. Ehhez pedig a programozói ismeretem nem elég nagy. Ezért ebben a részben csak a magam által megírt szubrutinokat ismertetem részletesen.

Leírás	Magyarázat
Private Sub AdatbevitelGomb_Click() ŰrlapNyitás ("Adatbevitel") ŰrlapZárás ("Kereső felület") End Sub	Ez a szubrutin egyfajta hidat képez az Adatbeviteli felület és a Kereső felület között. A művelet kattintásra hajtódik végre.
Private Sub Form_Load() DoCmd.Maximize MenüsorokKi Törlés Frissít End Sub	Ebben az utasítás tölti be a Kereső felületet, ahol a menüsort kikapcsolja. Erre adatvédelmi okokból volt szükség. Ezt követően egy Törlés és egy Frissít parancs hajtódik végre.
Private Sub Formátum_AfterUpdate() Frissít End Sub	Ez az utasítás akkor fut le, amikor a formátumot kiválasztja a felhasználó. A

	kiválasztást követően egy azonnali Frissítés megy végbe.
Private Sub KilépésGomb_Click() Kilépés End Sub	A kilépés gombra történő kattintáskor lépjen ki a programból

Private Sub NyomtatásiképGomb_Click()
On Error GoTo Err_hibakezelés
  Refresh
  NyomtatásiKép ("Nyomtatáshoz")
Exit_hibakezelés:
  Exit Sub
Err_hibakezelés:
  MsgBox "Hiba a [NyomtatásiképGomb_Click] rutinban van!" _
  & vbCrLf & " Hibakód = " & Err.Number _
  & vbCrLf & "Hibaleírás = " & Err.Description
  Resume Exit_hibakezelés
End Sub
	A kereső felületen való kereséskor a találatok kinyomtatására is mód van. Ezzel a gombbal az nyomtatási kép megtekintése válik elérhetővé.

Private Sub NyomtatásGomb_Click()
On Error GoTo Err_hibakezelés
  Refresh
  Nyomtatás ("Nyomtatáshoz")
Exit_hibakezelés:
  Exit Sub
Err_hibakezelés:
  MsgBox "Hiba a [NyomtatásGomb_Click] rutinban van!" _
  & vbCrLf & " Hibakód = " & Err.Number _
  & vbCrLf & "Hibaleírás = " & Err.Description
  Resume Exit_hibakezelés
End Sub
	Ha a nyomtatási képet megfelelőnek találjuk, akkor ez a parancssor fogja végrehajtani a noymtatást.

Private Sub Törlés()
  On Error GoTo Err_Lekérdez_Click
  Me.Szerző = ""
  Me.Cím = ""
  Me.Formátum = ""
  Me.Darab = ""
  Me.Ár = ""
  Me.Törölve = ""
  Me.Megjegyzés = ""
  Me.Nyilvántartási_szám = ""
  Me.Nyilvántartási_azonosító = ""
  Me.Melléklet = ""
  Me.Tok_nélküli = ""
	Ez a szubrutin egy függvény. Futtatása során a Szerző, Cím, Formátum, Darab, Ár, Törölve, Megjegyzés, Nyilvántartási szám,

Frissít Exit_Lekérdez_Click: Exit Sub Err_Lekérdez_Click: MsgBox Err.Description Resume Exit_Lekérdez_Click End Sub	Nyilvántartási azonosító, Melléklet, Tok nélküli adatmezőket üresre állítja a kereső felület keresési területén.
Private Sub TörlésGomb_Click() On Error GoTo Err_Lekérdez_Click Törlés Exit_Lekérdez_Click: Exit Sub Err_Lekérdez_Click: MsgBox Err.Description Resume Exit_Lekérdez_Click End Sub	Ez a rutin fogja a előbbiekben definiált törlés függvényt meghívni
Private Sub Űrlapfej_DblClick(Cancel As Integer) MenüsorBe End Sub	A kereső felületen a menüsor ki van kapcsolva, hogy a felhasználók ne tudjanak az adatokban módosításokat végrehajtani. E rutin segítségével dupla kattintásra a menüsor visszaugrik.

16. Táblázat: A kereső felületben megírt programrészek és magyarázata

Az adatbeviteli felület és a kereső felület közötti váltást a következő képen oldottam meg:

Leírás	Magyarázat
Private Sub KeresőGomb_Click() ŰrlapZárás ("Adatbevitel") ŰrlapNyitás ("Kereső felület") End Sub	Az adatbeviteli felületen

Private Sub KilépésGomb_Click() Kilépés End Sub	Ez az utasítás a kilépés gombra történő kattintást követően hívja meg a kilépés függvényt, amit a következő táblázatban mutatok be.
Private Sub Nyilvántartási_azonosító_GotFocus() Dim x As String x = "0000" & Me.Nyilvántartási_szám Me.Nyilvántartási_azonosító = "EH-" & Right(x, 4) End Sub	Ez a függvény a nyilvántartási azonosító generálására szolgál. A nyilvántartási azonosítót úgy kapjuk meg, hogy a nyilvántartási szám elé EH (mint Egyéb Hangzóanyagok) rövidítést teszünk. Viszont, hogy ezeket jól tudjuk kezelni a számokat előbb 4 számjegyűre egészítetem ki a programmal a szám elé írt megfelelő számú 0 segítségével
Private Sub KombináltLista23_AfterUpdate() ' A vezérlőelemmel egyező rekord keresése. Dim rs As Object	A vezérlőelemmel egyező rekord keresésére szolgál

Set rs = Me.Recordset.Clone rs.FindFirst "[Nyilvántartási szám] = " & Str(Nz(Me![KombináltLista23], 0)) If Not rs.EOF Then Me.Bookmark = rs.Bookmark End Sub	

17. Táblázat: Az Adatbeviteli felületen megírt programrészek és magyarázata

Public Sub Kilépés()
On Error GoTo Err_hibakezelés
ÜzenetSzöveg = "Valóban kiszeretnél lépni? "
Stilus = vbYesNo + vbDefaultButton2 + vbQuestion
Cím = "Kilépés a programból..."
 	
Válasz = MsgBox(ÜzenetSzöveg, Stilus, Cím)
 	
If Válasz = vbYes Then
DoCmd.Quit acQuitSaveAll
Else
DoCmd.CancelEvent
End If
 	
Exit_hibakezelés:
Exit Sub
Err_hibakezelés:
MsgBox "Hiba a [Kilépés] rutinban van!" _
& vbCrLf & " Hibakód = " & Err.Number _
& vbCrLf & "Hibaleírás = " & Err.Description
Resume Exit_hibakezelés
End Sub
	A kilépés végrehajtását szolgáló függvény

Public Sub RekordTörlése() On Error GoTo Err_hibakezelés ÜzenetSzöveg = "Valóban törölni szeretnéd az aktuális rekordot? " _	Ez a függvény a rekordtörlés menetét írja le. Hibakezelést és elágazást is

& vbCrLf & vbCrLf & " A törlés végleges, és nem vonható vissza!" _
& vbCrLf & vbCrLf & " Törlés végrehajtása [Igen] gomb." _
& vbCrLf & " Törlés megszakítása [Nem] gomb."
Stilus = vbYesNo + vbDefaultButton2 + vbQuestion
Cím = "Aktuális rekord törlése..."
Válasz = MsgBox(ÜzenetSzöveg, Stilus, Cím)
 	
If Válasz = vbYes Then
ÜzenetSzöveg = "Az aktuális rekordot végleg töröltem!"
Else
ÜzenetSzöveg = " Nem történt törlés!"
DoCmd.CancelEvent
End If
MsgBox ÜzenetSzöveg
 	
Exit_hibakezelés:
Exit Sub
Err_hibakezelés:
MsgBox "Hiba a [RekordTörlése] rutinban van!" _
& vbCrLf & " Hibakód = " & Err.Number _
& vbCrLf & "Hibaleírás = " & Err.Description
Resume Exit_hibakezelés
End Sub
	tartalmaz.

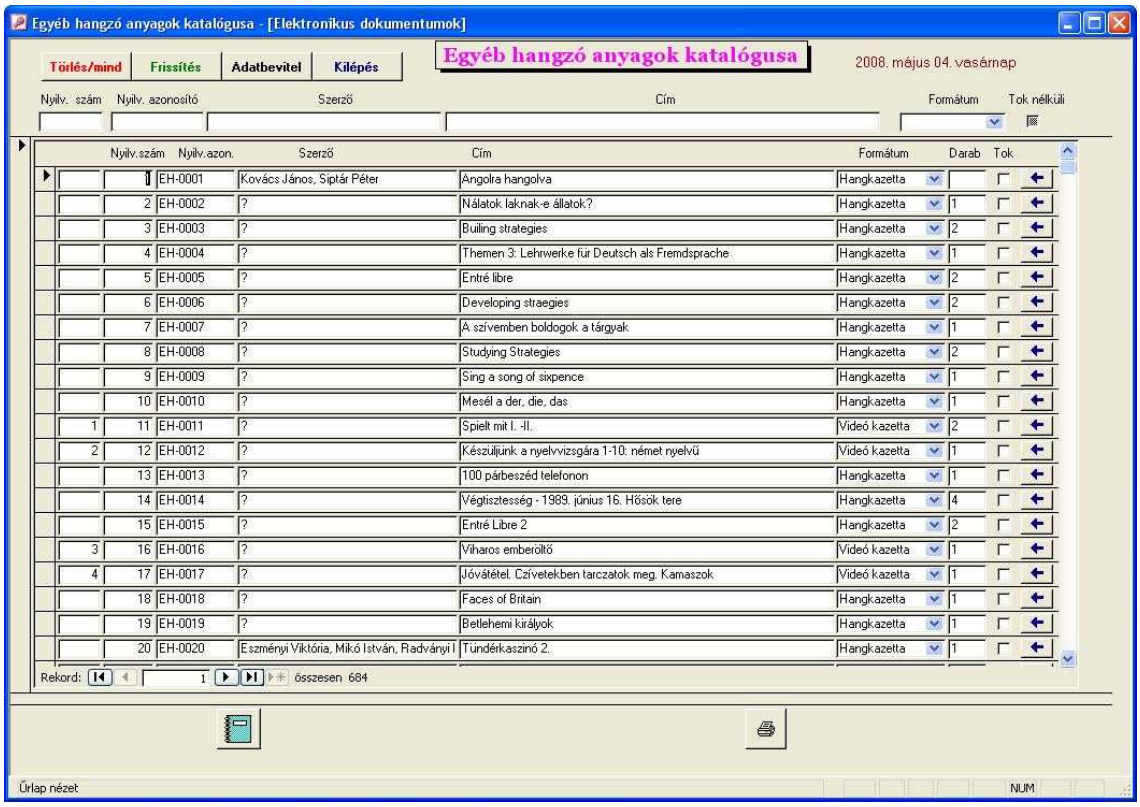
Public Sub Módosítás() On Error GoTo Err_hibakezelés ÜzenetSzöveg = " Tényleg szeretnéd a módosítást?" _ & vbCrLf & vbCrLf & " Módosítás végrehajtása [Igen] gomb." _ & vbCrLf & " Módosítás megszakítása [Nem] gomb." Stilus = vbYesNo + vbDefaultButton2 + vbQuestion Cím = "Adatmódosítás..."	Ez a függvény a módosítás menetét írja le. Hibakezelést és elágazást is tartalmaz.

Válasz = MsgBox(ÜzenetSzöveg, Stilus, Cím)
 	
If Válasz = vbYes Then
ÜzenetSzöveg = "A módosítás megtörtént!"
Else
ÜzenetSzöveg = "Nem történt módosítás!"
DoCmd.CancelEvent
End If
 	
Exit_hibakezelés:
Exit Sub
Err_hibakezelés:
MsgBox "Hiba a [Módosítás] rutinban van!" _
& vbCrLf & " Hibakód = " & Err.Number _
& vbCrLf & "Hibaleírás = " & Err.Description
Resume Exit_hibakezelés
End Sub	

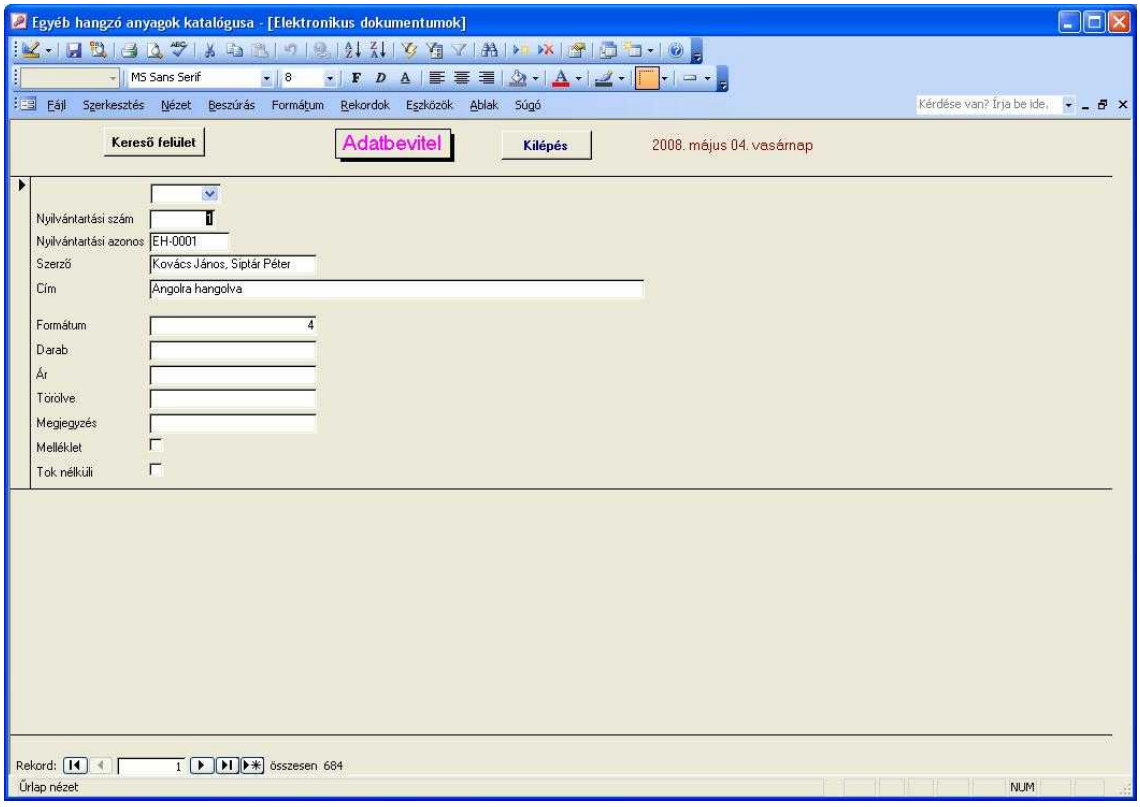
Public Sub MenüsorokBe() On Error GoTo Err_hibakezelés DoCmd.ShowToolbar "Menüsor", acToolbarYes DoCmd.ShowToolbar "Űrlap nézet", acToolbarYes DoCmd.ShowToolbar "Formázás (űrlap/jelentés)", acToolbarYes DoCmd.ShowToolbar "Adatbázis", acToolbarYes DoCmd.ShowToolbar "Űrlaptervezés", acToolbarYes Exit_hibakezelés: Exit Sub Err_hibakezelés: MsgBox "Hiba a [MenüsorokBe] rutinban van!" _ & vbCrLf & " Hibakód = " & Err.Number _ & vbCrLf & "Hibaleírás = " & Err.Description Resume Exit_hibakezelés End Sub	Ez a függvény a Menüsor bekapcsolását elősegítő függvény. Megadja, hogy mi aktivizálódjon futtatásakor.

18. Táblázat: Egyéb függvények

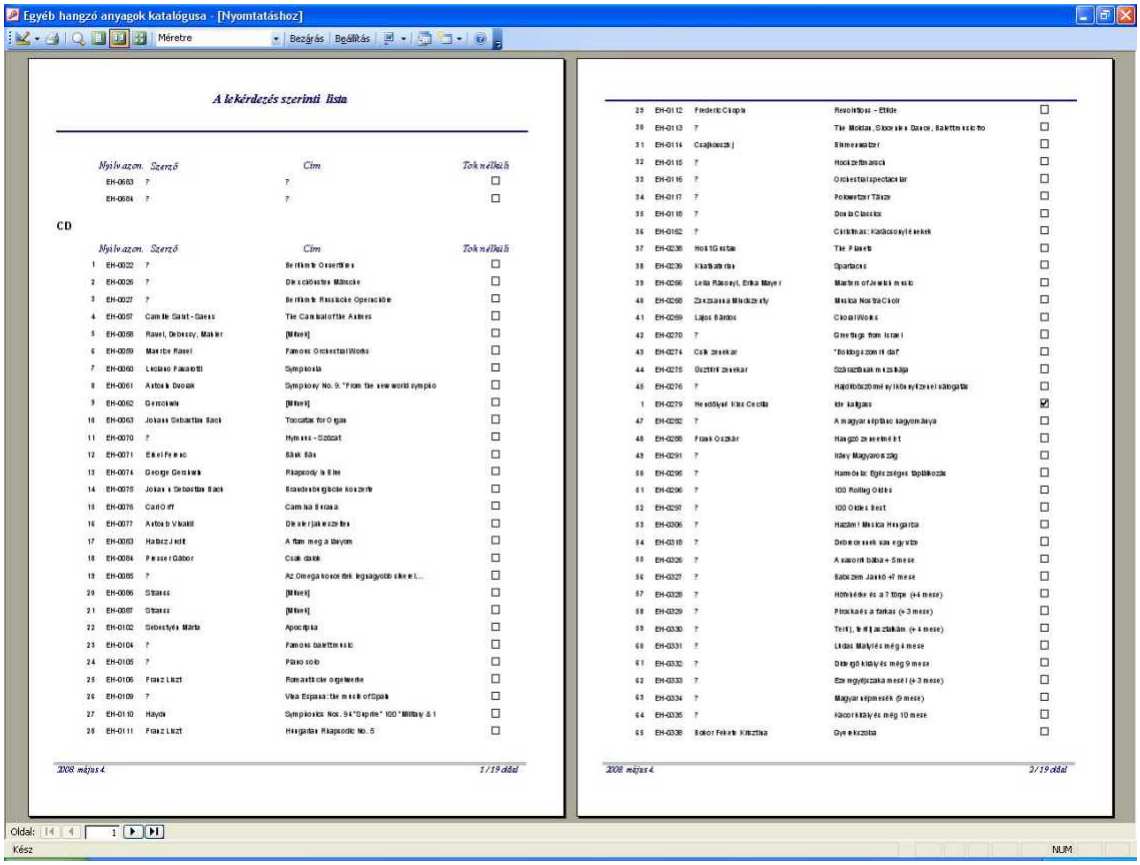
4. Képek a programból



9. ábra: A program kereső felülete



10. ábra: A program adatbeviteli felülete



11. ábra: Nyomtatási kép

12. ábra: Kilépéskor megjelenő üzenet

III) Összegzés

Jelenlegi állapotában a program alkalmas feladatának betöltésére, annak ellenére, hogy még nem tökéletes. A programot jelenleg is használják a hajdúböszörményi Pedagógiai Főiskolai Könyvtár dolgozói.

A programban vannak még kiaknázatlan területek a fejlesztést, tökéletesítést illetően. Az egyik ilyen lehetőség a táblák normalizálása. Ugyanis a dokumentumok tábla nem az ideális alakban szerepel. Ez azért van így, mert egy dokumentumnak több szerzője is van. Ennek megoldása nem lehetetlen, csak az idő hiányában még nem került megvalósításra.

Fejlesztési lehetőséget látok még abban is, hogy a program indításakor egy üdvözlő és bejelentkező képernyő jelenjen meg. Ez által könnyen lehetne mérni a program kihasználtságát. A belépő azonosító az olvasójegy száma lenne, és jelszóval pedig csak a könyvtárosok rendelkeznének. Ennek nagy előnye lenne, hogy a nem jelszóval történő belépés során az adatbeviteli gomb letiltásra is kerülhetne. Ekkor a hallgatók biztos nem tudnának az adatokhoz hozzáférni, csak olvasni.

Egy másik fejlesztési elgondolásom nagyon időigényes, ezért nem is foglalkoztam eddig vele. Az volt az elgondolásom, hogy egy pár soros ajánlást kellene minden rekordhoz csatolni. Ezáltal az olvasó könnyebben el tudja dönteni, hogy szüksége van-e a kiválasztott dokumentumra. Az ott eltöltött gyakorlati és kisegítő könyvtárosi időm alatt sokan érdeklődtek afelől, hogy mit tartalmaz az adott lemez vagy kazetta. Vagy, hogy milyen dokumentumot tudnék ajánlani, amin a tavasszal kapcsolatos mesék, dalok vannak. Ezt a sort tudnám még folytatni, hiszen a gyakorolni készülő óvónő jelöltek folyamatos programokkal látták el a gyerekeket.

Idézetek listája

1. Adatbázis-kezelés : Microsoft Access97/ BodnárIbolya, Nagy Zoltán. - Budapest : PC-START Stúdió, 1998. - 262 p. : ill. ; 24 cm

2. Idegen szavak és kifejezések szótára / főszerk. Bakos Ferenc. – Budapest: Akadémia Kiadó, 2005. – p.

3. Microsoft Acces http://209.85.129.104/search?q=cache:qSe9KdTJKFQJ:kvkmai.uw.hu/documents/tantargy/Villamosmernok_adatbaziskezeles.doc+Microsoft+access+kialakul%C3%A1sa&hl=hu&ct=clnk&cd=1&gl=hu&lr=lang_hu (2008. február 3.)

4. Siki Zoltán: Adatbáziskezelés és tervezés http://www.agt.bme.hu/szakm/adatb/db4.htm (2008. február 3.)

5. Basic http://hu.wikipedia.org/wiki/BASIC (2008. február 3.)

6. Visual Basic http://hu.wikipedia.org/wiki/Visual_Basic_programozási_nyelv (2008. február 3.)

7. Tanuljuk meg a Visual Basic 2005 használatát 24 óra alatt / James Foxall ; [ford. Batiz Judit et al.]. - Budapest : Kiskapu, cop. 2006. - XV, 511 p. : ill. ; 24 cm + CD

8. Visual Basic ismeretek / Péteri József ; [kiad. a Budapesti Gazdasági Főiskola Pénzügyi és Számviteli Főiskolai Kar]. - Budapest : Budapesti Gazdasági Főiskola, 2000. - 75 p. ; 30 cm

9. Fejezet 18 – PTE PMMK http://e-oktat.pmmf.hu/graf_rendsz_fej_19 (2008. április 28.)

Irodalom jegyzék

1. Idegen szavak és kifejezések szótára / főszerk. Bakos Ferenc. – Budapest: Akadémia Kiadó, 2005. – 723 p.; 25 cm

2. Adatbázis-kezelés Visual Basic-ben / Demeter M. Ibolya. – Budapest: Panem, 1998. - 323 p. : ill. ; 22 cm

3. Adatbázis-kezelés : Microsoft Access97/ BodnárIbolya, Nagy Zoltán. - Budapest : PC-START Stúdió, 1998. - 262 p. : ill. ; 24 cm

4. Programozzunk Visual Basic rendszerben! / Kuzmina Jekatyerina, TamásPéter, Tóth Bertalan. – Budapest : Computerbooks, 2003. - VII, 434 p. : ill. ; 24 cm + 1 diszk mell.

5. Visual Basic 6.0 : lépésről lépésre / Demeter M. Ibolya. - Budapest : Panem, [2002]. – 567 p.: ill. ; 22 cm

6. Visual Basic ismeretek / Péteri József ; [kiad. a Budapesti Gazdasági Főiskola Pénzügyi és Számviteli Főiskolai Kar]. - Budapest : Budapesti Gazdasági Főiskola, 2000. - 75 p. ; 30 cm

7. Tanuljuk meg a Visual Basic 2005 használatát 24 óra alatt / James Foxall ; [ford. Batiz Judit et al.]. - Budapest : Kiskapu, cop. 2006. - XV, 511 p. : ill. ; 24 cm + CD

8. Microsoft Visual Basic 6.0 : programozói kézikönyv / [ford. és szakmai ellenőrzés Consell Pannonia Kft.]. – Budapest : Park K., [1998]. - 968 p. : ill. ; 24 cm

9. Visual Basic.NET : fekete könyv / Steven Holzner ; [ford. Lénárt Szabolcs]. - Budapest : Perfact-Pro Kft., [2002]. - ill. ; 26 cm
   1. köt. - XXVI, 580, [1] p. + 1 CD-ROM mell.
   2. köt. - XIV, p. 584-1064. p.

Internetes források

1. Acces 2003 súgója
   http://office.microsoft.com/hu-hu/access/FX100646921038.aspx?CTT=96&Origin=CL100570041038 (2008. február 3.)

2. Siki Zoltán: Adatbáziskezelés és tervezés
   http://www.agt.bme.hu/szakm/adatb/db4.htm (2008. február 3.)

3. Microsoft Acces
   http://209.85.129.104/search?q=cache:qSe9KdTJKFQJ:kvkmai.uw.hu/documents/tantargy/Villamosmernok_adatbaziskezeles.doc+Microsoft+access+kialakul%C3%A1sa&hl=hu&ct=clnk&cd=1&gl=hu&lr=lang_hu (2008. február 3.)

4. Visual Basic 6.0 magyarul
   http://www.palkornel.hu/vb6hu/ (2008. február 3.)

5. Visual Basic
   http://hu.wikipedia.org/wiki/Visual_Basic_programozási_nyelv (2008. február 3.)

6. Visual Basic 2008 Express Edition (angol)
   http://www.microsoft.com/express/vb/Default.aspx (2008. február 3.)

7. Microsoft Termékinformációs központ: Microsoft® Visual Basic® .NET 2003 Standard Edition
   http://www.microsoft.com/products/info/product.aspx?view=34&pcid=3897bf96-7aa3-41e9-88d0-bc3d075ad436&type=ovr (2008. február 3.)

8. Basic
   http://hu.wikipedia.org/wiki/BASIC (2008. február 3.)

9. Fejezet 18 – PTE PMMK
   http://e-oktat.pmmf.hu/graf_rendsz_fej_19 (2008. április 28.)

Ábra jegyzék

Táblázat jegyzék