Debreceni Egyetem

Informatikai kar

Alkalmazásfejlesztés Delphiben

Témavezető:
Dr. Bölcskei András
Egyetemi docens

Készítette:
Gerják Balázs
programozó matematikus

Debrecen, 2007

1. Bevezetés

A szakdolgozatom címe jól mutatja, hogy a Delphi programozási nyelven való alkalmazásfejlesztéssel foglalkozok. Az alkalmazás – vagy application – az Oxford Dictionary of Computing (For Learners of English) szerint:

„a computer program or set of programs designed for a particular type of real world job”

„Egy computer program, vagy computer programok egy gyűjteménye, amelyeket meghatározott valós világbeli munkára terveztek”

Ennek megfelelően terveztem és programoztam le én is a megfelelő computerprogramot, amely egy valós világbeli feladatot hivatott ellátni. Egy könyvtárprogramot írtam.

A fejlesztés – vagy development – pedig az Oxford Dictionary of Computing szerint:

„the process of designing or improving something”

„Valami tervezésének, vagy javításának a folyamata.”

Ilyen értelemben a Delphiben való alkalmazásfejlesztés annak a folyamata, ahogyan ezt a könyvtárprogramot megterveztem és javításokat eszközölve rajta végül elkészítettem.

A programot Delphiben készítettem el. A Delphi egy Pascal alapú, objektumorientált, eseményvezérelt fejlesztői környezet.

Programozásnak nevezzük azt a tevékenységet, amikor a számítógépet egy tevékenységsorozat végrehajtására utasítjuk.

A Delphi programozási nyelv objektumorientált, ami azt jelenti, hogy adott, attribútumokkal és módszerekkel rendelkező osztályokból példányosított objektumuk felhasználásával programozunk, elősegítve a kód újrahasznosítást.

Minden objektumnak van *címe*, az a memóriaterület, ahol az adatstruktúra elemei elhelyezkednek. Az adott címen elhelyezkedő értékegyüttest az objektum *állapotának* hívjuk.

A példányosítás folyamán az objektum *kezdőállapotba* kerül.

Minden objektum példányosító osztálya meghatározza azt az *interfészt*, amely definiálja, hogy más objektumok számára az ő példányainak mely attribútumai és módszerspecifikációi látszanak.

Az objektumok üzenetek formájában kommunikálnak egymással és egy üzenet hatására egy objektum lehet hogy megváltoztatja az állapotát.

Az objektumnak van *öntudata*, minden objektum csak önmagával azonos és az összes többi objektumtól különbözik.

Az objektum rendelkezik egyedi *objektumazonosítóval* (Object Identifier – OID), amelyet valamilyen nyelvi mechanizmus valósít meg.

Az OO nyelvek az osztályok között egy aszimmetrikus kapcsolatot értelmeznek, melynek neve *öröklődés*.

Az öröklődés lényege, hogy az alosztály átveszi (örökli) szülőosztályának minden attribútumát és módszerét, és ezeket azonnal fel is tudja használni. Ezen túlmenően új attribútumokat és módszereket definiálhat, az átvett eszközöket átnevezheti, az átvett neveket újradeklarálhatja, megváltoztathatja a láthatósági viszonyokat, a módszereket újraimplementálhatja.

Az osztályok eszközeinek láthatóságát szabályozza a *bezárás*. Az OO nyelvekben általában a következő bezárási szintek léteznek. *Publikus* szint esetén az eszközt látja az összes kliens osztály. *Védett* szintnél az eszközökhöz csak a leszármazott osztályok férhetnek hozzá. A *privát* szintű eszközök viszont csak az adott osztályban használhatók.

Az OO nyelvek általában megengedik a módszernevek túlterhelését. Ez annyit jelent, hogy egy osztályon belül azonos nevű és természetesen eltérő implementációjú módszereket tudunk létrehozni.

Az OO nyelvekben az objektumok memóriában kezelt konstrukciók. Egy objektum mindig *tranziens*, tehát nem éli túl az őt létrehozó programot. I/O segítségével

természetesen bármely objektum állományba menthető és azután bármikor létrehozható egy *másik* objektum, amelynek állapota ugyanaz lesz. A nem nyelvi OO rendszerek (pl. adatbázis-kezelők) ismerik a *perzisztens* objektum fogalmát. Ekkor az objektum túléli az őt létrehozó alkalmazást, bármikor újra betölthető a memóriába, és *ugyanaz* az objektum marad.

A Delphi másrészt eseményvezérelt programnyelv. Az eseményvezéreltség lényege, hogy olyan függvényhívásokat alkalmazunk, amelyek blokkolják a program futását mindaddig, amíg egy olyan esemény be nem következik, amire a programnak reagálnia kell.
A programnyelvek kétfajta eseményt különítenek el, a felhasználó által kiváltott és a rendszer, illetve a programozó által létrehozott eseményeket.

A felhasználói események biztosítják a program és a felhasználója közötti kommunikációt. A felhasználó tevékenysége eseményeket generál, (pl. billentyűlenyomás, vagy egérgomb megnyomása) amelyek bekövetkeztekor a program megfelelő utasításokat hajt végre.

A rendszeresemények pedig a program belső állapotáról hordoznak információkat, ilyeneket hozhat létre a programozó is hasonló céllal.

Az eseményvezérelt programnyelvekben mindig van egy eseménykezelő (Event Handler), ami meghatározza azt, hogy az adott esemény bekövetkeztekor melyik kódrészletnek kell végrehajtásra kerülnie és vannak eseményfigyelők, amelyek adott esemény bekövetkeztéről értesítik az eseménykezelőt. Az események általában várakozási sorba kerülnek, feldolgozásuk FIFO (First In First Out – ez azt jelenti, hogy az időben hamarabb bekövetkezett esemény kerül előbb feldolgozásra) elven működik.

Ennek megfelelően a program megírása közben objektumokat használtam és a program eseményekre reagál.

2. Objektumorientáltság és eseményvezérlés a Delphiben

A Delphi nem tisztán objektumorientált programozási nyelv, ami azt jelenti, hogy az objektumorientált programozási-eszközrendszer mellett találhatók eljárásorientált eszközök is. A Delphi megtartotta elődjének, a Pascalnak az eljárásorientált fogalomrendszerbeli változó- és rekordfogalmát.

A Pascal nyelvben a változó egy olyan programozási eszköz, amelynek van címe, van neve, amellyel a címterületet hivatkozhatjuk, van értéke, amely a címterületen helyezkedik el, illetve lehetnek attribútumai, amelyeket a hivatkozási nyelv specifikál.

A rekord egy tárban létező adataszerkezet. Általában kötött számú és kötött sorrendű mezőből áll, de a Delphi dinamikus rekordszerkezetet biztosít, ahol az egyes mezők más-más, tetszőleges típusú értékeket tartalmazhatnak. A rekordban minden mezőt megnevezünk és rá közvetlenül a mezőnevével tudunk hivatkozni.

Könyvtárprogramom megírása közben a Delphiben mint objektumorientált programozási nyelvben használtam a formokat, illetve a Delphi komponenseket, mint objektumokat.

3. A Könyvtárprogram működése

3.1 A Könyvtárprogram működése felhasználói szempontból

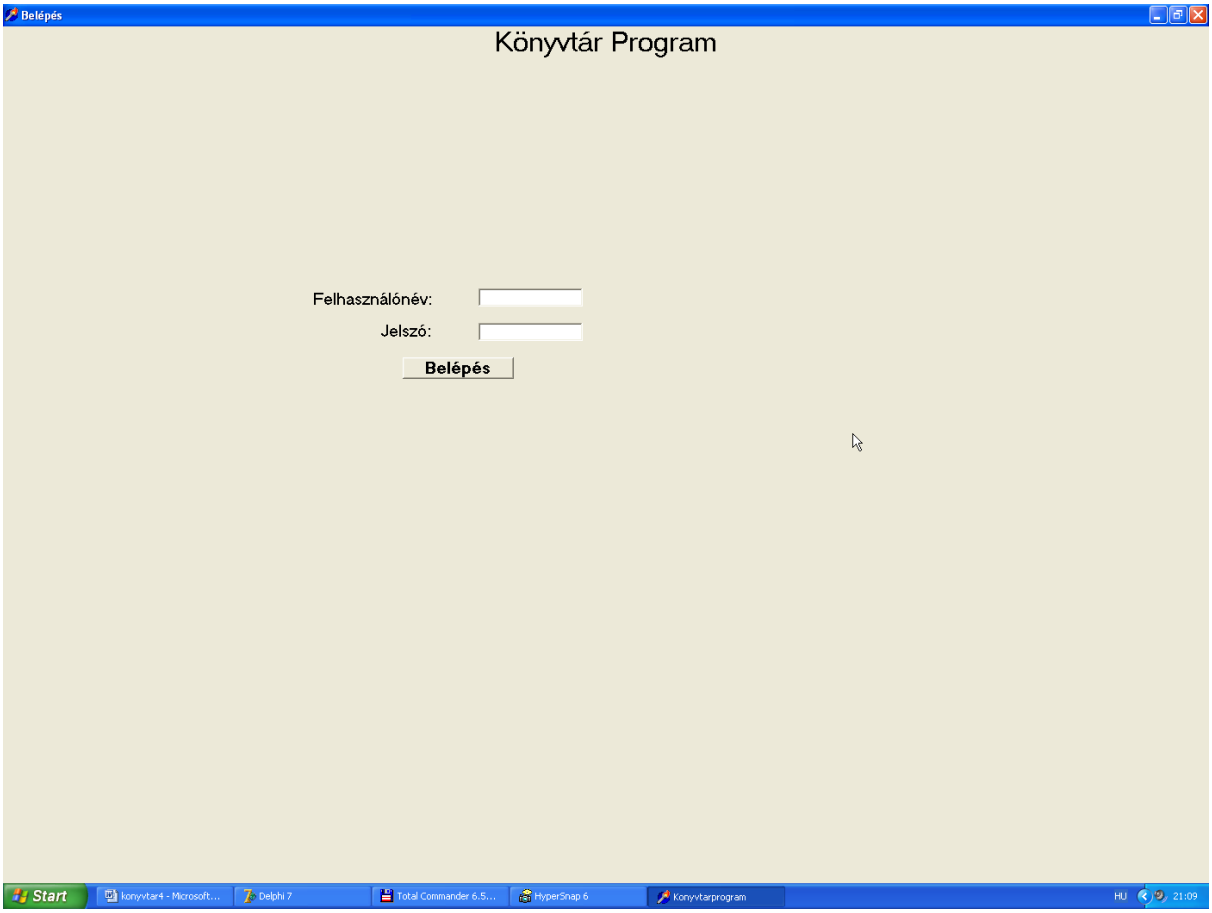
A következőkben a felhasználó szempontjából fogom a szakdolgozatom témájául szolgáló könyvtárprogram működését bemutatni. A felhasználó, amikor

először találkozik egy programmal, azokat a kérdéseket teszi fel, hogy hogyan működik, hogyan tudja majd kezelni. Programomban ezért a felhasználóbarát kezelőfelületnek kiemelt szerepet szántam. Ugyanakkor a számítástechnikában napjainkban az is fontos szemponttál vált, hogy a program „bolondbiztos” legyen, azaz a felhasználó ne tudja azt elrontani. Mint tudjuk, a „bolondbiztos” - ságot nem lehet száz százalékig biztosítani, azonban a program tervezési szakaszában és írása közben arra kell törekedni, hogy a programban ez a cél a lehető legteljesebb módon teljesüljön. Ezt a szempontot is figyelembe vettem a program megtervezése során.

Minden egyes program esetében vannak azonkívül célkitűzéseink, hogy miért fejlesztjük az adott programot, mire akarjuk használni az adott programunkat.
Egy könyvtárprogram esetében a fő cél az, hogy könyveket tudjunk nyilvántartani, kiadni kölcsönzésre, illetve visszavenni. Figyelnünk kell a könyvek kikölcsönzési idejének lejártát, hogy mennyi idő telt el azóta, illetve hogy mekkora összeget kell fizetni a kölcsönzési idő lejárta után. Új könyveket kell tudnunk felvenni a könyvtár nyilvántartásába, illetve ha elvesztek, akkor könyveket kell tudnunk törölni onnan. Ehhez hozzájárul még az is, hogy felhasználókat is számon kell tudnunk tartani, hiszen a felhasználó az, aki kiveszi, illetve visszahozza a könyveket, őmiatta, az ő számára van a könyvtárak szolgáltatása. A felhasználókat is fel kell tudnunk venni a nyilvántartásba, illetve le kell tudnunk venni onnan. A felhasználókat el kell tudnunk különíteni egymástól az adataik különbözősége alapján, illetve biztosítanunk kell számukra, hogy tudják használni a könyvtár szolgáltatásait. Természetesen nem lenne jó, hogyha minden felhasználó teljes jogokat kapna, hiszen az visszaélésekre adna alkalmat. A jogi tényezőket is figyelembe véve tehát két részre oszthatjuk a felhasználók csoportját: a könyvtárasokéra, akik sokkal több jogot élveznek a rendszeren belül mint a többiek, ők azok akik a könyvtárprogram adatbázisához hozzá tudnak férni és a mezei felhasználókéra, – akiket a továbbiakban csak felhasználóknak nevezek – és akiknek a jogaik a könyvek között való keresésre, könyv igényléseik leadására, illetve visszavonására, illetve a jelszavuk megváltoztatására terjednek ki.

A programomban ezeket a funkciókat a következőképpen oldottam meg:

A Bejelentkezés ablak



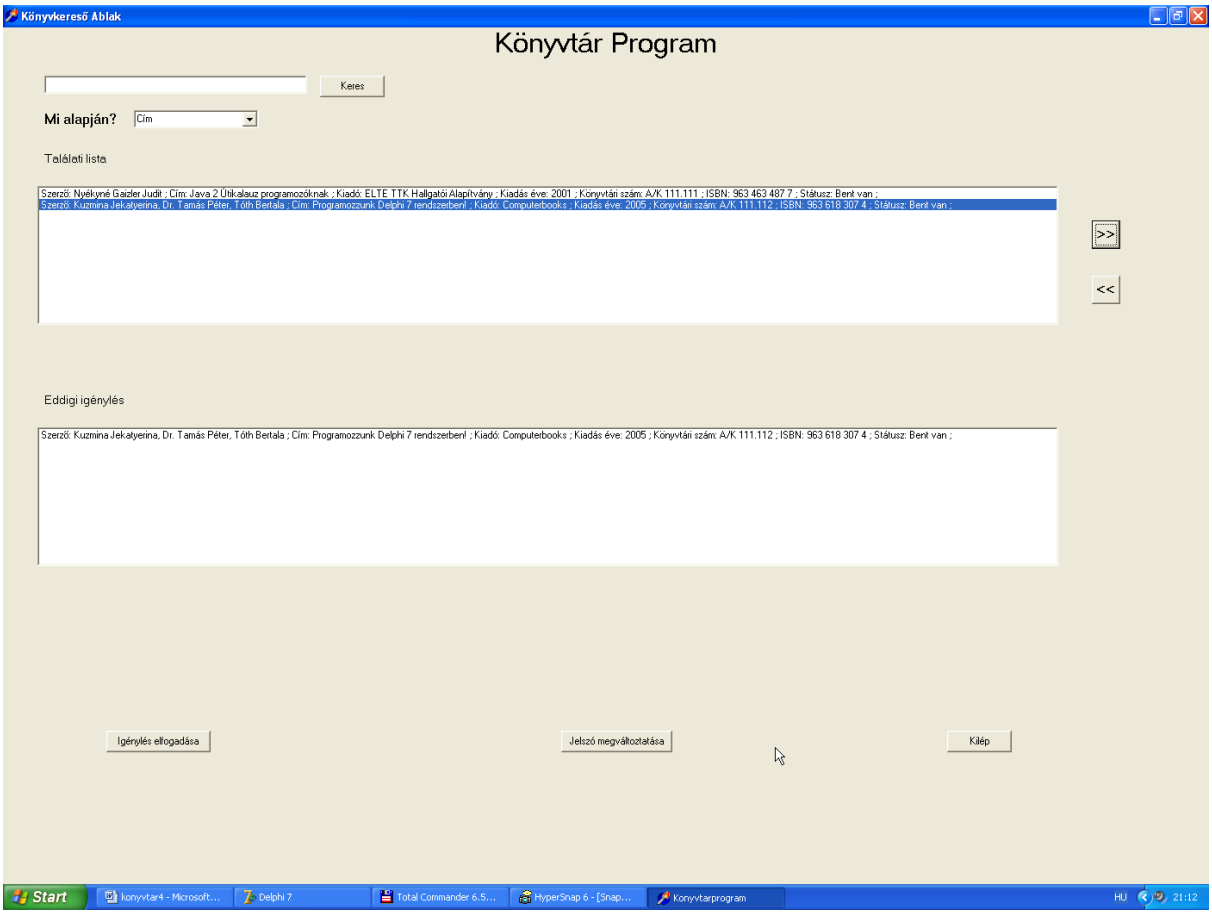
1. ábra

Amint a mellékelt ábra mutatja, a bejelentkezési képernyő egy felhasználónév és egy jelszó mezőből áll, valamint egy „Bejelentkezés” gombból. A program felhasználói felületének egyszerűsítését és a felhasználó minél kisebb mértékű összezavarását szem előtt tartva úgy döntöttem, hogy a bejelentkező képernyőn keresztül lehet majd elérni a programból a felhasználók és a könyvtárosok számára fenntartott szolgáltatásokat is, – így nem lesz számukra semmilyen külön bejelentkezési képernyő – és a program aszerint fog különbséget tenni, hogy felhasználó, vagy könyvtáros az illető, aki bejelentkezik, hogy megnézi, hogy a bejelentkezési név felhasználóhoz, avagy könyvtároshoz tartozik-e és eszerint biztosítja a számára a jogokat. Ebből következően két ember ugyanazzal a bejelentkezési névvel nem létezhet az adatbázisban. Elméletileg lehetséges, hiszen a felhasználókhoz és a könyvtárasokhoz egy String típusú Azonosító mező is bekerül az adatbázisba, azonban hogy a program kezelése minél egyszerűbb lehessen, így a különböző felhasználóneves megoldás mellett döntöttem.
A program, ahogyan az elvárható természetesen csak csillagokat ír ki a képernyőre, miközben a felhasználó vagy a könyvtáros beírja a jelszavát, hogy nehogy másvalaki is láthassa azt. A csillagos megoldást a továbbiakban is mindenhol alkalmaztam, ahogy a felhasználónak vagy a könyvtárosnak jelszót kellene bevinnie, így erre a megoldásra a szakdolgozatom további részeiben nem fogok külön hivatkozni.

A továbbiakban először sorra veszem, hogy a könyvtárprogramban mihez fér hozzá, illetve mit tud elérni egy mezei felhasználó.

A Könyvkereső ablak

Amikor egy felhasználó belép a rendszerbe, a program a könyvkereső ablakhoz irányítja át. (Lásd, kép lent.)



2. ábra

A könyvkereső ablak egy olyan funkciókat biztosító ablak, amelyeket hagyományosan bármely könyvár programnak biztosítania kell a felhasználó felé. Van egy szövegbeviteli mező, ahol a felhasználó megadhatja azt a mintát, amelyet aztán különböző szempontok szerint, – így szerző neve, könyv címe, tárgyszó, illetve kiadó – a program ráilleszt a könyvekre vonatkozó adatbázisában tárolt

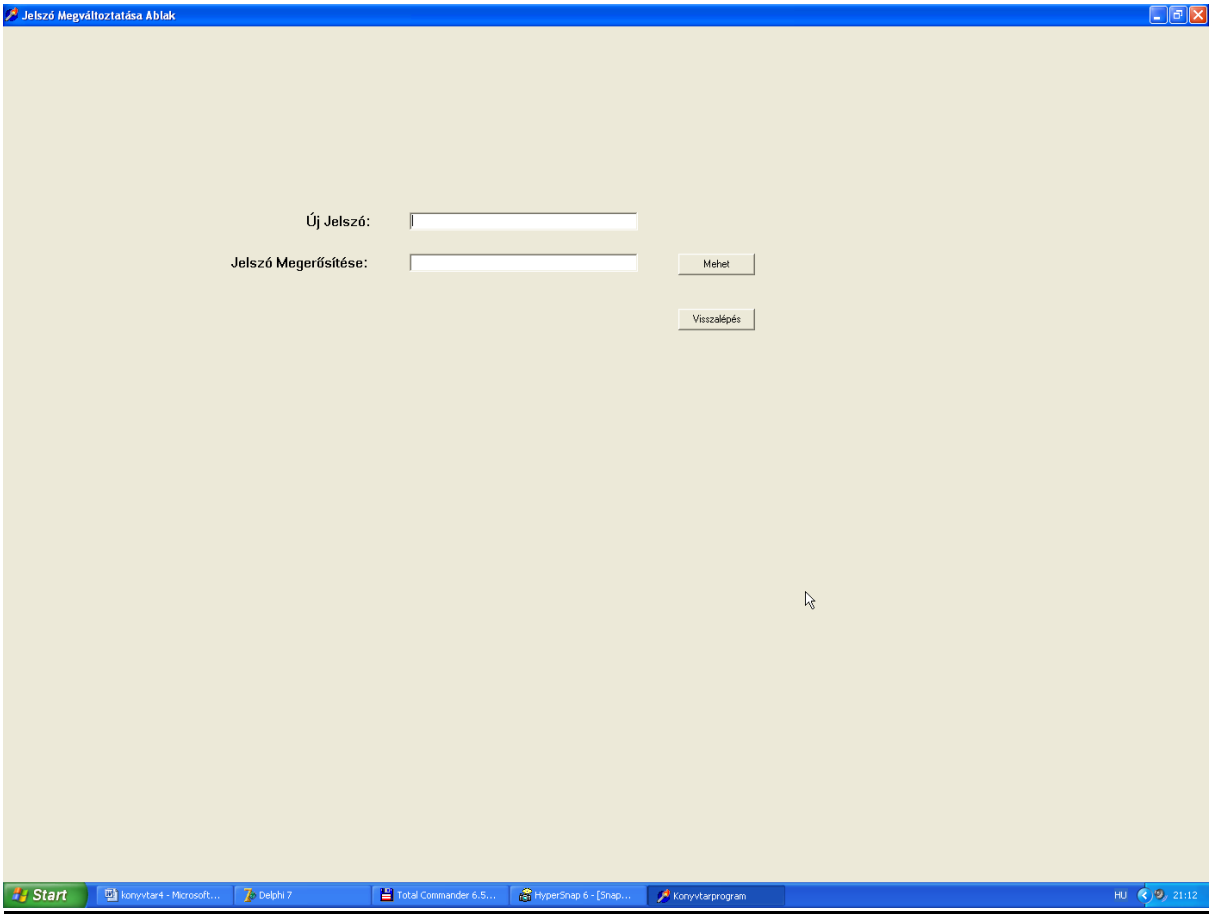
rekordoknak a megfelelő mezőire. A mintát a program a rekord megfelelő mezőjében a mező elejétől a végig kutatja és bármilyen előfordulásban megtalálja. Így nem számít a kis- és nagybetű különbség, sem az, hogy a minta a szó elején, végén, vagy a szó közepén található.

A program a találati listában írja ki a megtalálás sorrendjében azokat az adatbázis rekordokat, amelyekre illeszkedett a minta. A találati listáról azután az egér segítségével kijelölheti azokat a könyveket, amelyeket szeretne igényelni és ezeket az igényléseket a „>>” gomb megnyomásával átküldheti az igénylési listára. A program lehetőséget biztosít a felhasználónak arra is, hogy a nem kellőképpen átgondolt igényléseket az igénylési listáról később még visszavonhassa. A felhasználó az igénylési listán ugyanúgy kijelölhet sorokat és a „<<” gomb megnyomásával törölheti azokat az igénylési listáról. Az „igénylés elfogadása” gomb megnyomásával az igénylés megerősítésre kerül és a lista átküldésre kerül a könyvtárosok által elérhető szolgáltatások egyikének. (Erről később a könyvtárosok által elérhető szolgáltatásokkal foglalkozó résznél beszélek.)

Mint látható, a programban minden elektronikusan elvégezhető, nincs szükség igénylési listák kézzel való kitöltésére, néhány kattintással elintézhető a módosítás (az igénylés).

A jelszó megváltoztatása gombra kattinva a felhasználó egy újabb ablakon megváltoztathatja a jelszavát, a kilépés gomb segítségével pedig visszakerülhet a bejelentkezési képernyőre.

A jelszó megváltoztatása ablak



3. ábra

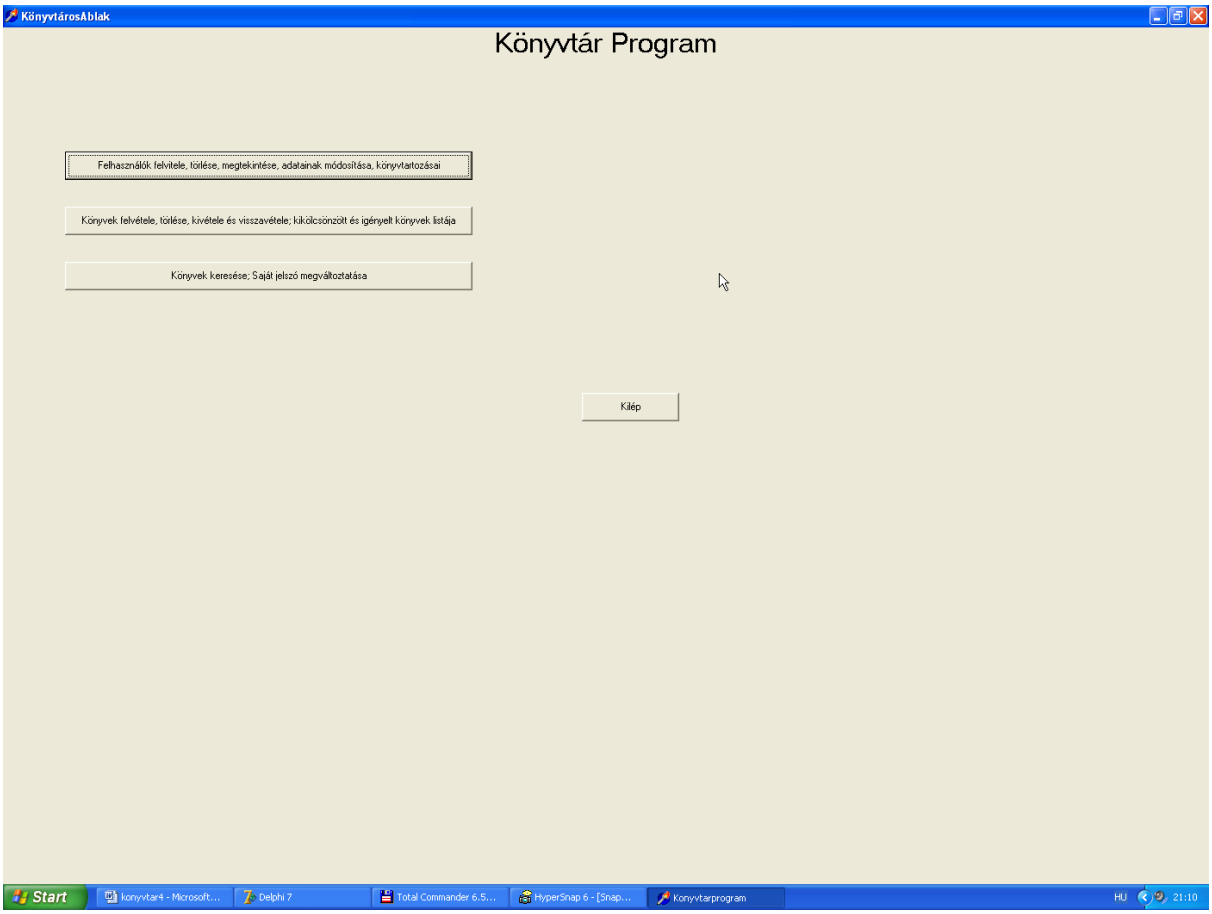
A felhasználók néha biztonsági okokból szeretnék megváltoztatni a jelszavukat. Erre lehetőségük van, hiszen ebben az ablakban az új jelszó kétszeri begépelésével és a

„Mehet” gombra kattintással ezt könnyedén megtehetik. Biztonsági szempontokat is figyelembe véve a felhasználónak természetesen kétszer kell megadnia a jelszavát, kiküszöbölendő azt az esetet, hogy egyszer megadja tévesen, megváltoztatja azt, majd csak a könyvtáros segítségével tud újra belépni a rendszerbe, mert a rendszer a hibás jelszóbevitel miatt nem engedi azt.

3.1.1 A könyvtáros által elérhető szolgáltatások

A könyvtáros ablak

A könyvtárosnak természetesen egyéb feladatokat is el kell látnia, illetve ő az, aki az adatokat módosítja a rendszerben, így neki több funkciót kell elérnie a bejelentkezés után. A könyvtáros ezért egy külön ablakra kerül a bejelentkezés után.



4. ábra

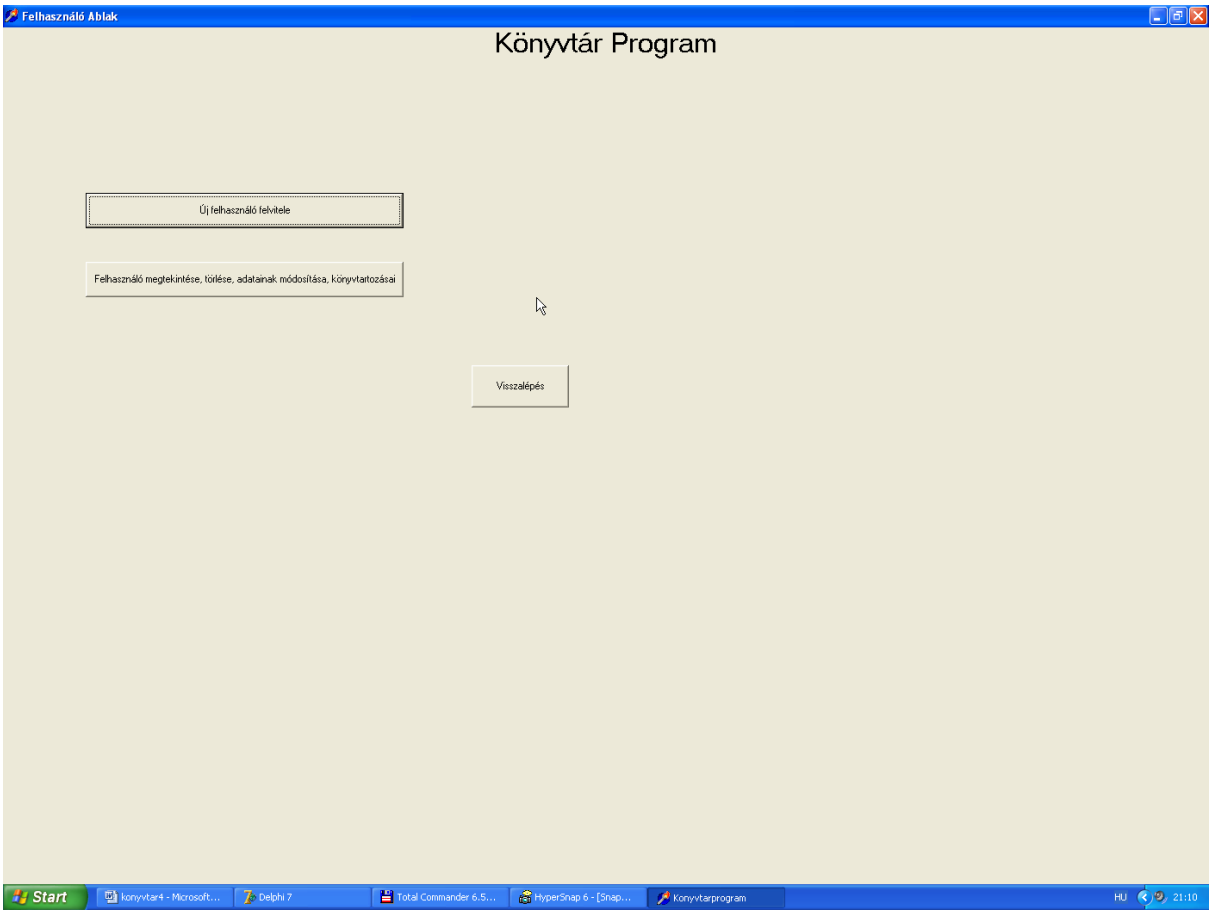
Ezen az ablakon az egyszerűbb áttekinthetőség kedvéért csoportosítva találhatóak meg azok a funkciók, amelyeket a könyvtáros a munkája során el akarhat érni.
Külön csoportot képeznek a könyvekkel végezhető műveletek, külön csoportot a felhasználókkal végezhető műveletek és végül elérheti a könyvtáros még azokat a szolgáltatásokat, amelyeket a felhasználók elérhetnek, hiszen a könyvtáros egyben maga is felhasználó és elképzelhető, hogy ő maga is szeretne könyveket igényelni,

hogy kivegye őket. Így a „Könyvek keresése; Saját jelszó megváltoztatása” ablak segítségével a korábban már ismertetett műveleteket végezheti el.
(A könyvtáros erről az ablakról léphet ki továbbá a bejelentkezési képernyőre, illetve a könyvkereső ablakról magától értetődően nem a bejelentkezési képernyőre fog visszalépni, hanem erre az ablakra.)
Hogy a felhasználó minél könnyebben kezelhesse a rendszert, fontos volt továbbá az a szempont, hogy jobban átláthassa azt. Így a különböző szolgáltatás csoportoknál külön kiírja a gombokra a program, hogy milyen alszolgáltatásokat érhetünk el a főbb szolgáltatástípusokra rákattintva.

A különböző szolgáltatástípusokat sorba véve:

3.1.2 A felhasználókhoz kötődő funkciók

Az első felhasználókkal foglalkozó ablak

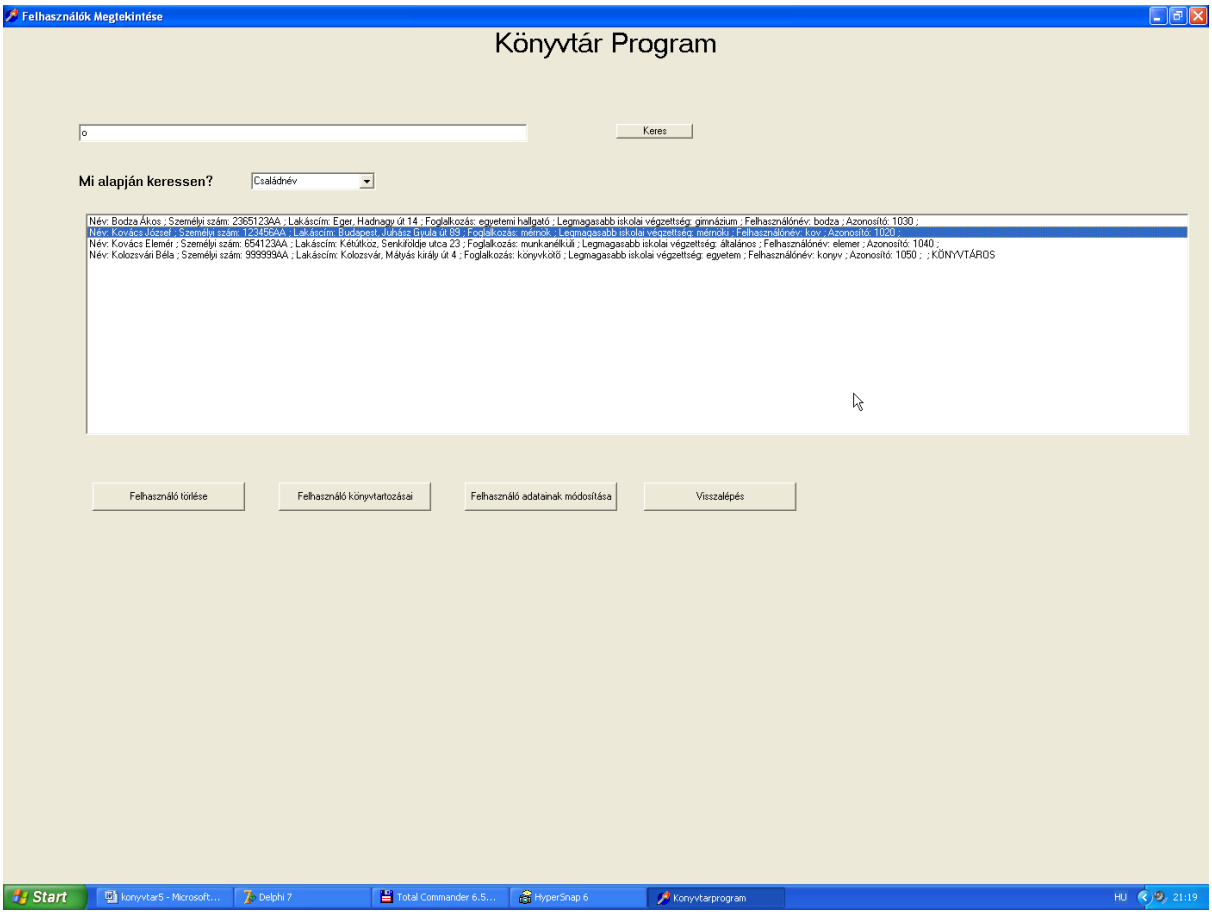


5. ábra

Az első felhasználókkal foglalkozó ablakon egyrészt az új felhasználók felvitelének ablakát lehetséges elérni, illetve egy külön csoportba szedve a felhasználókkal végezhető egyéb műveletekre lehetséges továbbmenni – így felhasználók megtekintése, törlése, adatainak módosításai, illetve könyvtartozásainak megtekintése –. A csoportosítást az igazolja, hogy amikor új felhasználót akarunk felvenni az adatbázisba, akkor a könyvtárosnak és a programnak semmilyen

előzetes információt nem kell tudnia a felhasználóról, hiszen a rendszerbeli adatai még akkor kerülnek kialakításra. Abban az esetben azonban, hogyha egy felhasználóról le akarjuk kérni az adatait, módosítani akarjuk, vagy törölni akarjuk a felhasználót, esetleg egy adott felhasználóhoz tartozó könyvtartozásokat akarjuk megnézni akkor szükség van a felhasználó azonosítására is. Ezért ez a gomb, amely ezzel foglalkozik gyűjtőjévé válik ezeknek a műveleteknek és egy külön ablakra irányítja át a kattintásnál a felhasználót, ahol legelőször is egy keresést végezhet a könyvtár felhasználói adatbázisában.

A felhasználók megtekintése ablak

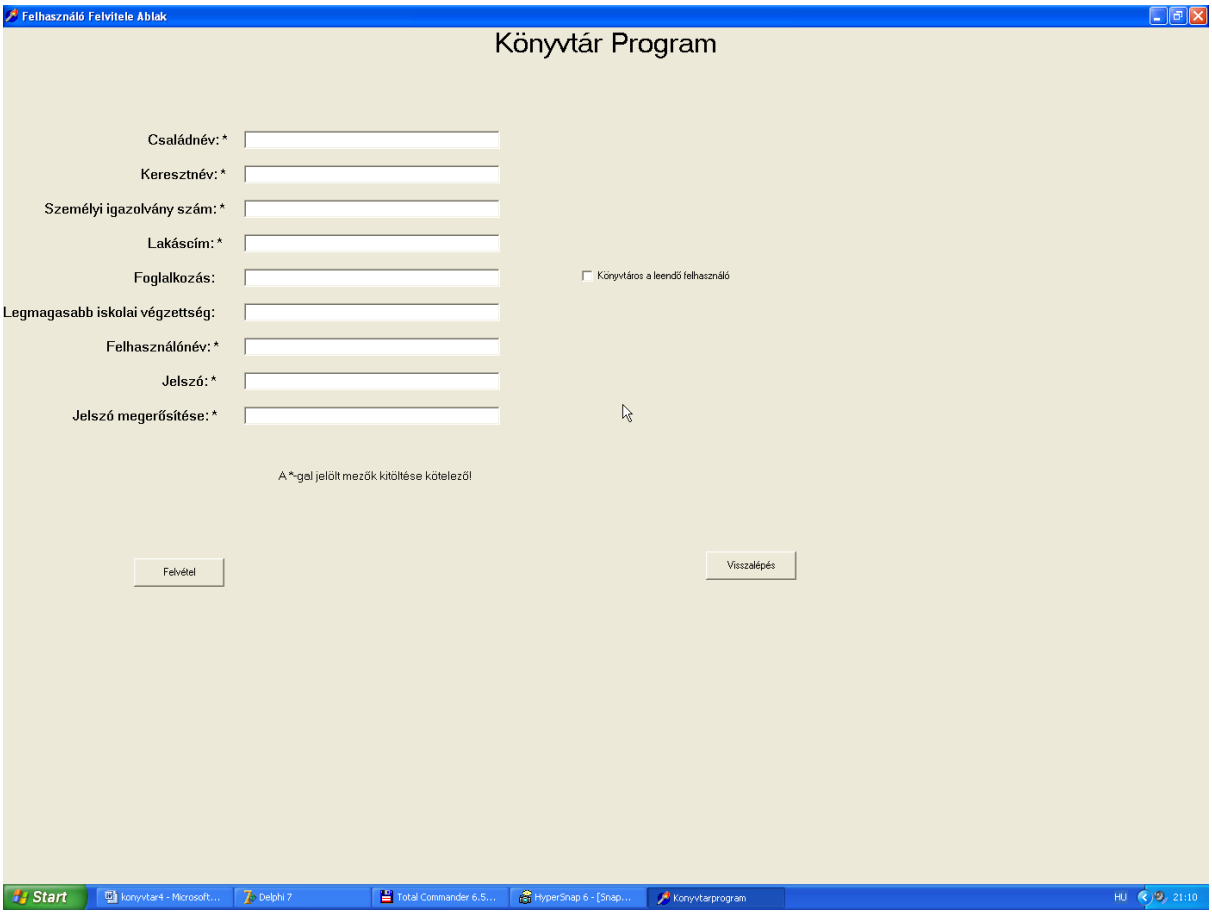


6. ábra

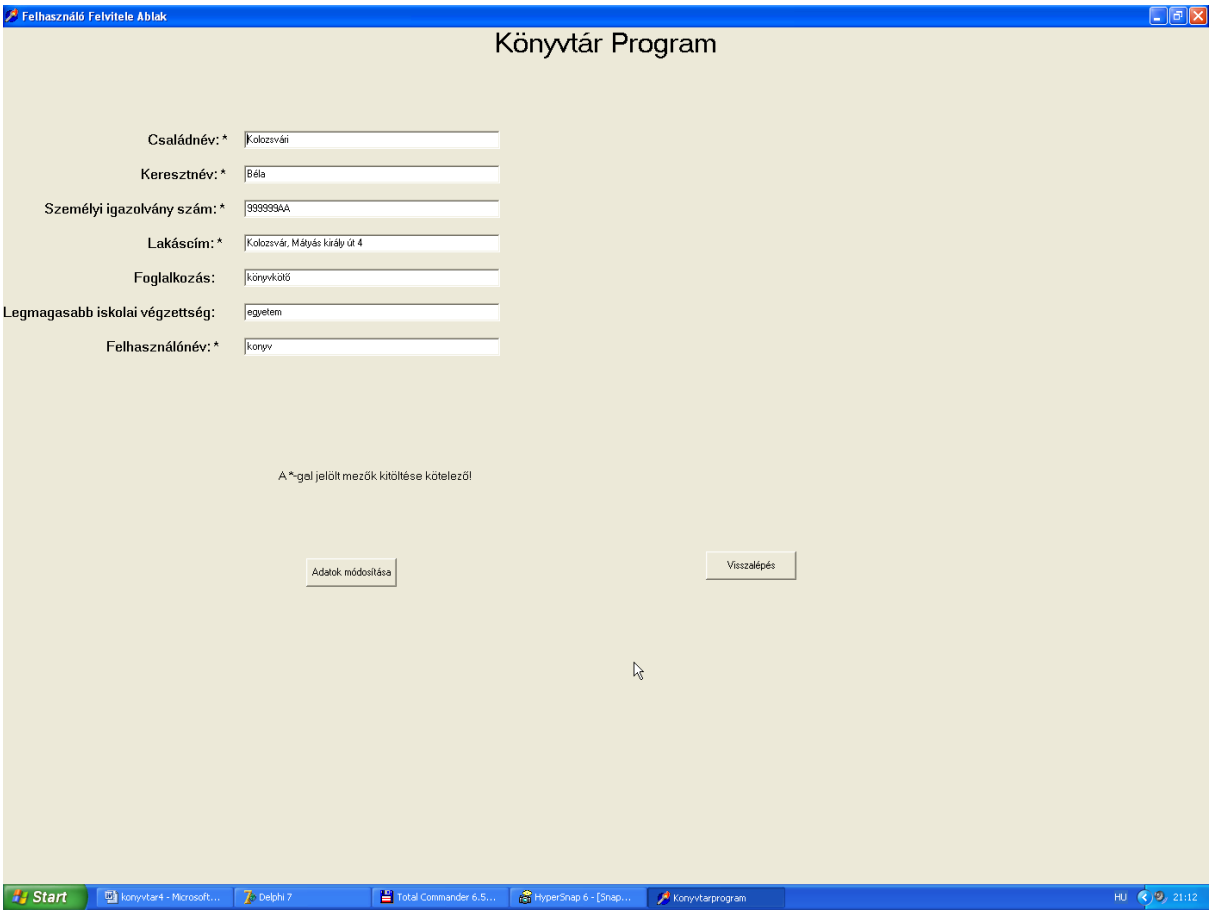
Mint korábban említettem, ebben az ablakban a felhasználói adatbázisban kereshetjük meg azokat a felhasználókat, akikről információkat szeretnénk szerezni. A keresést végezhetjük családnév, keresztnév, személyi igazolvány szám, illetve felhasználónév alapján. Arra is lehetőségünk van, hogy az összes felhasználót egyszerre megtekintsük, ebben az esetben a „Mi alapján keressen?” szöveg után következő legördülő listáról a „Mind” kiválasztásával és a „Keres” gombra való

rákattintással az összes felhasználót meg lehet tekinteni. A mintaillesztés ugyanaz a mintaillesztés, amiről a könyvek keresése ablaknál már beszéltem, így az ott elmondottak ide is érvényesek.
A keresés eredménye a középső ListBox-ban jelenik meg. Ezután az egér segítségével ki lehet jelölni egy felhasználót, majd törölni lehet az adatbázisból, módosítani lehet az adatait, vagy meg lehet nézni a könyvtartozásait.

Felhasználók felvitele, illetve adatainak módosítása


Felhasználó Felvitele Ablak
Könyvtár Program
Családnév: *
Keresztnév: *
Személyi igazolvány szám: *
Lakáscím: *
Foglalkozás:
Legmagasabb iskolai végzettség:
Felhasználónév: *
Jelszó: *
Jelszó megerősítése: *
Könyvtáros a leendő felhasználó
A *-gal jelölt mezők kitöltése kötelező!
Felvétel
Visszalépés
Start
konyvtar4 - Microsoft...
Delphi 7
Total Commander 6.5...
HyperSnap 6 - [Snap...
Konyvtarprogram
HU
21:10

7. ábra



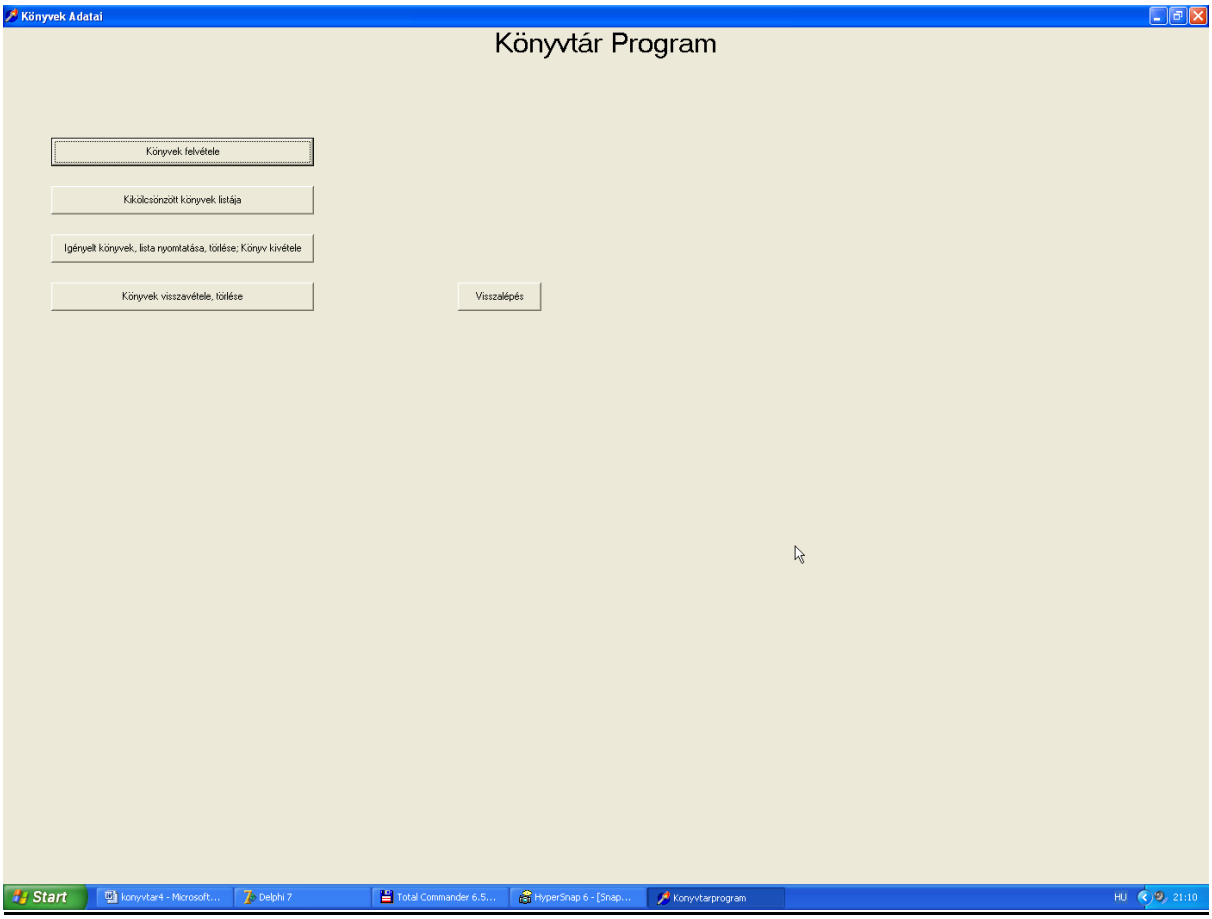
8. ábra

A felhasználók felvitele és a felhasználók adatainak módosítása ablak – a fő funkció kivételével – kisebb különbségektől eltekintve ugyanazokkal a funkciókkal rendelkezik. A felhasználók adatainak megváltoztatása ablakban a felhasználóhoz tartozó adatok – kivéve a jelszót – azonnal megjelennek a megfelelő mezőkben, így a módosítani nem kívánt adatokat nem kell újra begépelni, hanem egyszerűen az eredeti állapotában otthagyhatjuk. A program külön szól, hogyha a könyvtáros elfelejtett egy kötelezően kitöltendő adatot megadni, illetve hogyha már van olyan

felhasználó nevű felhasználó az adatbázisban, amire az eredeti felhasználónevet meg akarta változtatni.
Lehetőség van továbbá arra is, hogy egyetlen gombnyomással eldöntsük, hogy könyvtáros, vagy felhasználó legyen-e az illető, ilyen módon a program egyesíti a két felhasználótípus felhasználói felületen keresztül történő kezelését és egyszerűsíti a felhasználó számára a program átláthatóságát.
A felhasználó törlését is ezen az oldalon lehet – megerősítést követően – megtenni.

3.1.3 A könyveken elvégezhető műveletek

A könyvek adatai ablak

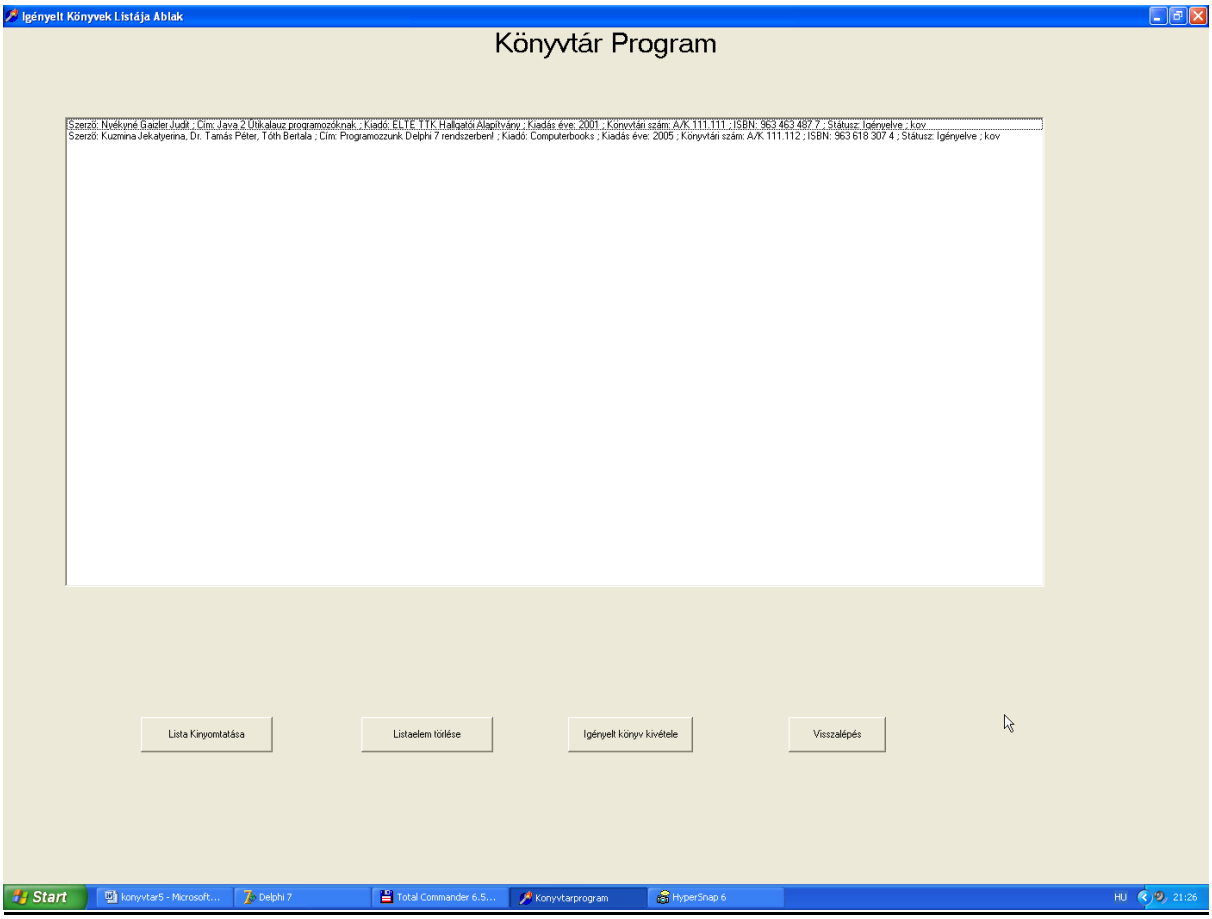


9. ábra

A könyvtáros által elvégezhető másik tevékenységcsoport a könyvekhez kötődő tevékenységek csoportja. A könyvtáros számára megjelenő első ablakban a középső gombra kattintva egy új ablak jelenik meg. Itt láthatjuk a nyomógombok szerint csoportosítva a könyveken elvégezhető műveleteket. Az első és a második műveletcsoport csak egy-egy elemből áll. A többi esetben tevékenységcsoportokról beszélhetünk, a felhasználók ablakának ismertetésekor már említett okokból

kifolyólag. Egy új könyv felviteléhez nincs szükség az adatbázisban való keresésre, csakúgy ahogy a kikölcsönzött könyvek listájának megtekintéséhez sem. Két külön csoportba sorolhatjuk azonban azokat a tevékenységeket, amelyek a könyvek kivételéhez kötődnek és másikba azokat, amelyek a könyvek visszavételéhez és törléséhez. Mind a két csoport a beépített keresőrendszer szerint különül el egymástól. Az első esetben az igényelt könyvek listájának az alapján végzünk műveleteket, a másik esetben pedig az adatbázisunkban lévő könyvek listáját vesszük alapul a módosításhoz. A könyvtáros a számára a megfelelő listához gyűjtöttük össze a megfelelő műveleteket.

Az igényelt könyvek listája ablak



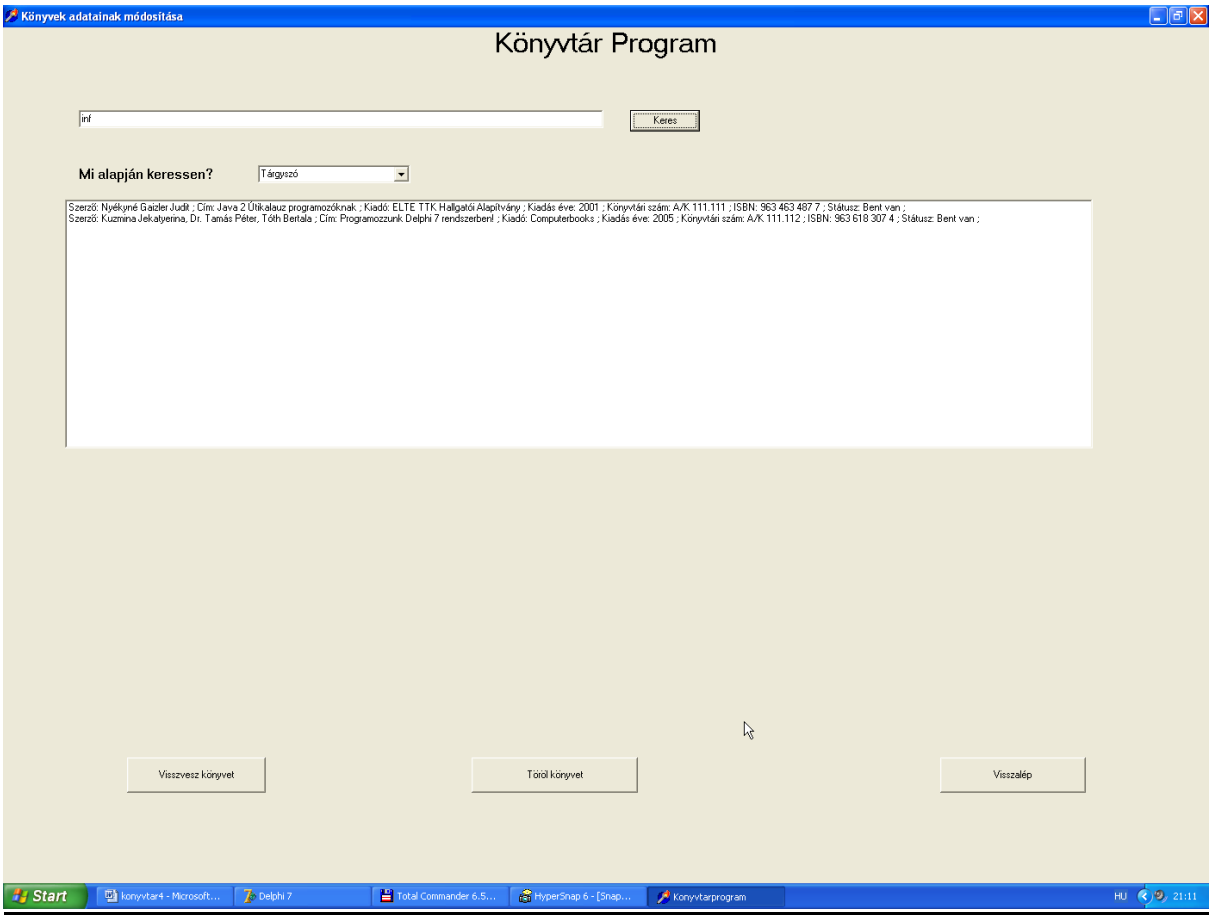
10. ábra

Az igényelt könyvek listája ablakban könyvtárosbarát módon azonnal megjelennek azok a könyvek, amelyeket az egyes felhasználók igényeltek. A lista egyes elemeire kattintva kiválaszthatjuk őket a ListBoxban és ezután a lent található gombok segítségével műveleteket végezhetünk rajtuk.

A könyvtáros munkáját megkönnyítendő a lista egészét ki is tudjuk nyomtatni, így azt elviheti magával, hogy a segítségével könnyebben ki tudja keresni a raktárból a könyveket.

A kijelölt tételt a könyvtáros ugyanakkor törölheti is a listáról, vagy pedig a kivételt elektronikusan is megerősítheti.

A könyvek adatainak módosítása ablak



11. ábra

Ebben az ablakban a könyvtáros egy keresőrendszer segítségével rákereshet a könyvekre szerző, cím, tárgyszó és kiadó, továbbá a kikölcsönző felhasználói neve alapján. A mintaillesztés az előzőekben elmondottak szerint történik. A kikölcsönző felhasználói neve alapján való keresés azért kellemes, mivel amikor a felhasználó személyesen visszahozza a könyveket, akkor a név alapján azonnal látható, hogy mely könyvek voltak a nevén és így vissza lehet venni őket.
Ebben az ablakban található továbbá a könyvek törlésére szolgáló gomb is, mivel a könyvek néha el is veszhetnek.

3.2 A Könyvtárprogram működése programozás-technikai szempontból

3.2.1 A program által tárolásra használt állományoknak a szerkezete és funkciója

Amikor a felhasználó begépeli a bejelentkezési ablakban a jelszavát, akkor a program két állományban nézi meg, hogy a begépelt felhasználónév és a jelszó egyezik-e a két állományban tárolt felhasználónév és jelszó párok valamelyikével.
Ha egyezik, akkor a rendszer belépteti a felhasználót, vagy könyvtárost.
A két állomány neve felh.dat és konyvtaros.dat.
Az állományok tartalmát a felhasználó felviteli ablak segítségével lehetséges felvinni. Attól függően, hogy a felhasználó felviteli ablakban a könyvtáros-e a felhasználó rubrika be van-e jelölve, a program az adott felhasználónak az adatait a konyvtaros.dat-ba, menti, mint könyvtárosét, vagy a felh.dat-ban, mint hagyományos felhasználóét.
Az állományokban a felhasználók adatait rekordok segítségével tároltam le.
A felhasználók rekordjainak a szerkezete:

type

```
 TFelhAdat = record
   CsaladNev: String[50];
   KeresztNev: String[30];
   SzemSzam: String[20];
   LakasCim: String[100];
   Foglalkozas: String[60];
   Vegzettseg: String[60];
   FelhNev: String[50];
   Jelszo: String[50];
   Azonosito: String[50];
 end;
```

A rekord egyes mezőinek String típusnak választása segíti a program egyszerűsítését, illetve a felesleges konverziók elkerülését is szolgálja.
A felhasználókat egymástól a felhasználónév, vagy az azonosító alapján tudjuk megkülönböztetni. Mindkettő egyedi. A program természetesen a megfelelő műveleteknél, így az új felhasználó felvitelénél, illetve a felhasználó adatainak módosításakor figyel arra, hogy ugyanazok a felhasználói nevek és azonosítók – a felh.dat-ot és a konyvtaros.dat-ot összevetve is – ne fordulhassanak elő több felhasználónál.

Amikor a felhasználó megadta a felhasználónevét és jelszavát, a program megnézi, hogy melyik állományban találta meg az egyezést és ennek megfelelően lépteti be a felhasználót, mint könyvtárost, vagy mint hagyományos felhasználót.

Ezen kívül a két állományon kívül használok még egy állományt, konyv.dat néven, amely állomány a könyvek adatait tartalmazza. Ez az állomány is rekordokban tárolja az egyes könyvek adatait. Az állomány szerkezete:

```
type
```

```
TKonyvAdat = record
  Szerzo: String[50];
  Cim: String[50];
  Kiado: String[50];
  Kiadas_eve: String[5];
  Konyvtari_szam: String[14];
  ISBN: String[16];
  Statusz: String[20];
  felhNev: String[50];
  targyszo: String[50];
  kivetel: TDateTime;
end;
```

A TKonyvAdat rekordban, csakúgy mint TFelhAdat rekordban a felesleges konverziók elkerülése érdekében String-ben történik, kivéve a kivetel mezőt, mivel ott a konverzió elkerülése érdekében ésszerűbbnek tűnt TDateTime órát és percet is kezelni képes dátumos rekordformátumot használni.

A TKonyvAdat rekordban letárolunk mindent, ami fontos lehet egy könyvtári könyv esetében. Az adatok alapján később keresést is végezhetünk a könyvtárprogram beépített keresőmotorjának segítségével.
A könyvek azonosítására a könyvtári azonosítót és az ISBN számot használjuk, mivel a könyvtári azonosító nem lehet ugyanaz két könyvnél ugyanabban a könyvtárban, illetve mivel az ISBN szám minden könyv esetében egy egyedi, 10 számjegyből álló azonosító.

3.2.2 A jelszó begépelés közbeni elrejtésének eljárása

A program a megfelelő formokon található jelszó begépelésére fenntartott szövegmezők esetében minden jelszómezőhöz fenntart egy sztringet az eredeti

jelszó letárolására, az eredeti szövegmezőben található szöveget pedig elrejti a felhasználó elől. A jelszómezőbe kerülő minden egyes karakter begépelésénél a program először kimásolja az új karaktert a jelszót ténylegesen tároló sztringbe és a szövegmezőbe annyi csillagot helyez el, amennyi karakter található a jelszót tartalmazó sztringben. A szövegmezőből való törlés esetében a program természetesen figyel arra, hogy a sztringből is töröljön.

3.2.3 A különböző ablakok létrehozása és törlése

A program bejelentkezéskor a bejelentkezés formot kezdi el futtatni, majd a bejelentkezés után újabb és újabb formokat lehet elérni. Az egyes formok különböző funkciókat látnak el, azaz a különböző felhasználói funkciók különböző formokon kaptak helyet. Amikor a felhasználó egy új funkciót akar elérni, a program a régi funkciót tartalmazó ablakot láthatatlanná teszi, az újat pedig a Create metódus segítségével elkészíti és a képernyő teljes szélességében kinagyítja. A visszalépést megkönnyítendő a program minden formhoz tartalmaz egy SzuloLeiro és egy SajatLeiro mutatót, amelynek segítségével el lehet érni azt a formot, amelyik az aktuális formot kreálta és egy aktuális formon lefuttatott Close metódus után újra láthatóvá lehet tenni.

3.2.4 A könyvek keresőjének működése

Ahogyan korábban írtam, a szövegbeviteli mezőben a felhasználó megadhatja azt a mintát, amelyet aztán különböző szempontok szerint, – így szerző neve, könyv címe, tárgyszó, illetve kiadó – a program ráilleszt a könyvekre vonatkozó adatbázisában tárolt rekordoknak a megfelelő mezőire. A mintát a program a rekord megfelelő mezőjében a mező elejétől a végig kutatja és bármilyen előfordulásban megtalálja.

Így nem számít a kis- és nagybetű különbség, sem az, hogy a minta a szó elején, végén, vagy a szó közepén található.
A program a találati listában írja ki a megtalálás sorrendjében azokat az adatbázis rekordokat, amelyekre illeszkedett a minta. A kiírásnál a rekordban található Stringeket, így a könyv szerzője, a könyv címe és a többit egyetlen egy sztringgé fűzi össze és ennek a sztringnek a végére, - ha a könyvet kivették, - TDateTime típusról String típusra való konverzió után kiírja a kivétel dátumát, illetve ha a kölcsönzési határidő – ami az eredeti beállítás alapján két hét – lejárt, akkor kiírja azt is, hogy mekkora a könyvön lévő tartozás összege. A tartozás összege az alapbeállítás szerint naponként 25 forinttal növekszik.

Amikor a felhasználó a könyvek keresőjének ablakában a találati lista és az igénylési lista között szeretné mozgatni a könyveket, akkor a rendszer csak a ListBoxokban található egyes elemeket másolja, illetve törli. Amikor a felhasználó ráklikkel az „Igénylés elfogadása” gombra, akkor – csakúgy mint a program többi részében is – nem a teljes lista mentődik, pusztán az egyes könyvekhez tartozó rekordbejegyzések módosulnak az állományokban. Más formokon pedig a módosított rekordbejegyzések alapján állítja elő a program a megfelelő listákat.
Így például az igényelt könyvek listája ablakban azok a könyvek kerülnek kiírásra, amelyek Statusz mezőjébe egy „Igényelve” sztring került bejegyzésre.
Természetesen a konyv.dat állományban található rekordok megfelelő mezőinek módosítása azt igényli, hogy a ListBoxban található, abból sztringként kinyert elemek újra rekordformátumban legyenek értelmezhetőek, azaz a művelet a sztring több sztringre és más adatszerkezetre széttagolást, azaz további konverziót igényel.
Így a programhoz írtam egy megfelelő, TKonyvAdat rekordot széttagoló és a találati listában és az igényelt könyvek listájában található ListBoxokból származó sztringeket újra TKonyvAdat rekordba átalakító programrészletet is.

A program más részében, például amikor az adott felhasználó által kivett könyveket akarjuk megtekinteni, akkor is a program a konyv.dat-ban található könyvek rekordjait, illetve a felh.dat-ban vagy a konyvtaros.dat-ban található felhasználók rekordjait egyezteti annak megfelelően, hogy az adott felhasználónévhez a

konyv.dat-ban a könyvek rekordjanak felhNev mezője megegyezik-e az adott felhasználónévvel.

Összegzés

A Delphiben fejlesztett könyvtárprogramom működőképes, felhasználókat és könyveket egyaránt képes tárolni az adatbázisában és képes az adatokat módosítani. A felhasználók eltérő jogosultságainak és hozzáférési szintjeinek problémáját is hatékonyan képes kezelni, a könyvtárosok a rendszeren belül adminisztrátori jogokkal bírnak.
Némi módosítás után a könyvtárprogram könyvtári felhasználásra is alkalmassá tehető lenne. Illetve programom fejlesztése közben azt tartottam szem előtt, hogy a lehető leginkább felhasználóbarát legyen. Ennek megfelelően az alkalmazás a belépéskor automatikusan ellenőrzi, hogy aki bejelentkezik az felhasználó, vagy könyvtáros-e és az ennek megfelelő funkciókhoz fér hozzá. A könyvkereső metódus a megfelelő fejezetben említett okoknak megfelelően felhasználóbarát. A program arra is figyelmezteti a könyvtárost, hogy a megfelelő adatok megadásánál mely mezőket hagyta kitöltetlenül, illetve arra is figyelmezteti, hogyha olyan műveletet akar végrehajtani, ami nem megengedett.
A program fejlesztése közben céljaim között szerepelt hogy egy működőképes, felhasználóbarát alkalmazást fejlesszek, amely a gyakorlatban is használható.
Ezeknek a kritériumoknak eleget tesz és megfelelő vizuális továbbfejlesztés és némi funkcióbeli bővítés után a program a gyakorlati használatban is megállná a helyét.

Irodalomjegyzék:

1. *Oxford Dictionary of Computing (For Learners of English)* ed: Sandra Pyne and Allene Tuck. Oxford University Press, 1996

2. http://www.iit.bme.hu/education/szglab5/fbeadas/tanacsok.html

3. Juhász István: *Programozás 2.* MobiDIÁK könyvtár. sorozatszerk: Fazekas István. http://mobidiak.inf.unideb.hu, 2004

4. Kuzmina Jekatyerina, Dr. Tamás Péter, Tóth Bertalan: *Programozzunk Delphi 7 rendszerben!* ComputerBooks. Budapest, 2005

5. Csordás Annamária és Mohai Gábor: *Adatszerkezetek és algoritmusok, Jegyzet dr. Juhász István előadása alapján*. http://nyf.beckground.hu/incoming/klte-jegyzetek/Jegyzetek_Konyvek/Adatszerkezetek%20es%20algoritmusok/Inf2_f.pdf

6. http://www.delphibasics.co.uk/

7. Benkő Tiborné, Benkő László, Tóth Bertalan, Varga Balázs: Programozzunk Turbo Pascal nyelven. ComputerBooks. Budapest, 1995