SZAKDOLGOZAT

Kiss Albert

Debrecen

2009

Debreceni Egyetem

Informatikai Kar

A VIZUÁLIS PROGRAMOZÁS TANÍTÁSA A DEBRECENI MECHWART ANDRÁS GÉPIPARI ÉS INFORMATIKAI SZAKKÖZÉPISKOLÁBAN

Témavezető:

Nyakóné dr. Juhász Katalin Mária
tudományos főmunkatárs

Készítette:

Kiss Albert
szakvizsgázó
informatikatanár

Debrecen

2009

Tartalomjegyzék

Bevezetés **2**

1. A szakmacsoportos alapozó oktatás helye az iskola képzési profiljában **3**

2. A programozás az érettségi vizsgán **6**

3. A programozási nyelv választása az iskolában **16**

4. A vizuális programozás tanítása **20**

4.1. Az elméleti órák fontosabb elméleti tananyagrészeinek vázlata: 22

4.1.1. Az objektumorientált programozás alapfogalmai: 22

4.1.2. A 4 GL alapfogalmai, tulajdonságai: 25

4.1.3. A Turbo Delphi alapfogalmai 27

5. Ismeretek felhasználása a gyakorlatban **30**

1. feladat: Ablak átméretezése 31

2. feladat: Név bevitele 33

3. feladat: Listák 35

4. feladat: Síkidomok 37

5. feladat: Kérdés 39

6. feladat: Editor 40

7. feladat: Színkeverés 43

8. feladat: Méretezés 45

9. feladat: Kirakó 46

10. feladat: Képnézegető 47

11. feladat: Csigafutam 48

12. feladat: Grafikus szerkesztő 49

13. feladat: Külső programok indítása 51

14. feladat: Puma 52

15. feladat: Audiólejátszó 54

16. feladat: Drag&Drop 56

17. feladat: Létrehozás futás közben 58

18. feladat: MDI alkalmazás 60

19. feladat: Konzol-alkalmazás 1. 63

20. feladat: Konzol-alkalmazás 2. 64

6. Összefoglalás **65**

7. Irodalomjegyzék **66**

Bevezetés

1993 óta dolgozok a Mechwart András Gépipari és Informatikai Szakközép-iskolában. Az első két évben gépészmérnökként kizárólag gépészeti jellegű tantárgyakat tanítottam. Az iskola profilja és az informatika fejlődése miatt egyértelmű volt, hogy szükséges az iskolában oktatott másik szakirányból is megszereznem a megfelelő képesítést. Ezért 1996-ban beiratkoztam az akkor még Kossuth Lajos nevét viselő egyetem informatikatanár szakára, levelező képzésre. Az itteni tanulmányaimmal párhuzamosan az iskolában egyre több informatika és számítástechnika órát kaptam. Ekkor már az óráim egy részén algoritmizálást és programozást is kellett tanítanom. Az iskolában a programozást Pascal nyelven oktatták a kollégák, így én is azt kezdem el tanítani. Néhány évvel ezelőtt követelményként megjelent az objektumorientált programozás ismeretének igénye, és ehhez az informatikai szakmacsoportos képzésben bevezettük a vizuális programozás oktatását is. A tanulók Pascal ismeretei alapján kézenfekvő volt, hogy a Pascal nyelvre épülő Object Pascal nyelv és a hozzá kötődő Delphi fejlesztői környezetet válasszuk ehhez. A kezdeti liszensz gondokat három évvel ezelőtt megoldotta az oktatási célra ingyenesen használható Turbo Delphi Explorer 2006 megjelenése.

A szakdolgozatomban szeretném bemutatni, hogy milyen érettségi vizsgakövetelmények határozzák meg az iskolánkban folyó programozástanítást, és milyen dilemmákkal szembesültünk az utóbbi években. A dolgozatban röviden ismertetem a vizuális programozás keretében oktatásra kerülő főbb témaköröket, ezek elméleti alapjait és a gyakorlati oktatás során történő felhasználásukat az órán készítendő gyakorlati feladatok segítségével.

1. A szakmacsoportos alapozó oktatás helye az iskola képzési profiljában

A Mechwart András Gépipari és Informatikai Szakközépiskolában évfolyamonként hagyományosan két gépészeti szakmacsoportos alapozó képzésű és két informatika szakmacsoportos alapozó oktatásban részesülő osztály tanul. Az iskola a nyolcadik évfolyamosok számára kiadott felvételi tájékoztatóban nagy hangsúlyt fektet arra, hogy az „igazi" szakközépiskolai képzés nem ér véget az érettségivel, mert az addig itt eltöltött négy, illetve a két tannyelvű képzésben résztvevők számára öt év nem csak az érettségire készít fel. Ezeken az évfolyamokon jelentős óraszámban tanulnak a diákok szakmai képesítés megszerzését megalapozó tantárgyakat.

A szakképzéshez szükséges előtanulmányok hatékonyabb végzése érdekében a kerettanterv által biztosított szabad órakeretet az informatika és az idegen nyelv oktatására fordítja az intézményem.

A megnövelt informatika és informatika szakmacsoportos oktatás óraszáma szükséges ahhoz, hogy a két év alatt megszerezhető szakképesítés oktatási ideje az érettségit követően két év helyett egy évre csökkenjen, mert az érettségiig terjedő időszakban tanult szakmai tárgyak óraszáma beszámítható a szakképzés óraszámába.

A Mechwart András Gépipari és Informatikai Szakközépiskola nem vállalhatja, hogy az érettségit követő szakképzés idejét két évre tervezi. A jelentős mennyiségű szakmai alapozó ismeret beszámítása révén ezt az időt egy évre tudja az iskola csökkenteni. Ez több ok miatt is szükséges:

1. Az iskolába felvett tanulók tanulmányi eredménye jóval 4 egész felett van. A jó beiskolázási eredmény révén általánosságban megállapítható, hogy az itt tanuló diákok jó képességűek. Ezt igazolja, hogy az iskola az országos tanulmányi versenyek mindenkori utolsó öt éves eredményei alapján évek óta a szakközépiskolák között az országos rangsorban az elsők között található. Az utóbbi években pedig folyamatosan a képzeletbeli dobogón helyezkedik el. A másik fontos eredményességi mutató a továbbtanulási arány. Eszerint a tanulók több mint 70%-a már az érettségi évében felvételt nyer valamely felsőoktatási intézménybe. A felvett tanulók passzív féléveket kérnek az adott egyetemtől, illetve főiskolától, hogy még egy éve alatt megszerezhessék a megfelelő szakmai

végzettséget is. A kérvényezhető passzív félévek száma pedig nem teszi lehetővé a két éves halasztást.

2. A gépészet szakmacsoportos képzésünk egyik osztálya német két tanítási nyelvű osztály. Ebbe az osztályba járó tanulók a nyelvi előkészítő évet követően kezdik csak el a többiek számára előírt négy tanítási évet, így ők a beiratkozásuk után öt év alatt jutnak el érettségiig.

A leírtakból következik:

- A német két tanítási nyelvű osztályok esetén, ha a szakma megszerzése az érettségi után további két évet igényelne, nem sokan vállalnák, hogy hét évet töltsenek el a szakközépiskolánkban a szakma megszerzéséig, hanem érettségi után elkezdenék egyetemi és főiskolai tanulmányaikat.
- Az informatikus osztályban tanulók nagy része nem vállalná, hogy további két évet töltsön az iskolában egy adott szakma megszerzéséért, még akkor sem, ha ezek a szakmák jelenleg keresett képzettséget, megfelelő tudást és lehetőséget biztosítanak számukra.

A fentiek figyelembe vételével a szakmai alapozó oktatás tantárgyainak beszámíthatósága miatt az iskola tantestülete az iskola pedagógiai programjának elfogadásakor az informatika szakmacsoport alapozó oktatásában a 9-12. évfolyamon az alábbi tantárgyakat a táblázatban látható óraszámokkal határozta meg [1]:

Évfolyam:	9.	10.	11.	12.
Informatika szakmacsoportos alapozó ismeretek (elmélet)	***3***	***1***	***3***	***2***
A műszaki pályák világa	1			
Anyag- és eszközismeret	1			
Műszaki ábrázolás és képfeldolgozás alapjai	1			
Számítógép-programozás I.		1		
Számítógép-programozás II.			3	
Számítógép-programozás III.				1
Adatbázis-kezelés				1
Informatika szakmacsoportos alapozó gyakorlatok	***3***	***3***	***6***	***7***
Kép és hang feldolgozás, multimédia *		2		
Számítástechnikai gyakorlatok I.	3			
*Számítógép-programozási gyakorlatok I .**		1		
*Számítógép-programozási gyakorlatok II. **			5	
*Hardverismeret és gyakorlat I. **			1	
*Számítógép-programozási gyakorlatok III. **				5
*Hardverismeret és gyakorlat II. **				2
Szakmacsoportos alapozó oktatás összesen	**6**	**4**	**9**	**9**
Emelt szintű érettségire felkészítő tantárgy választható (2×2 óra)*			4	4

Jelmagyarázat az óratervhez: * Csoportbontásban kell tanítani
** Hármas - létszámhoz igazodó - csoportbontásban kell tanítani

Az óraterv alapján kiszámolható, hogy a szakmacsoportos alapozó oktatás keretében a tanulók összesen 991 órán tanulnak informatika jellegű tantárgyakat. Ebből 323 óra elmélet és 668 óra gyakorlat. Ehhez adható még a négy éven keresztül heti egy-egy órában tanult informatika és az utolsó két évfolyamon az emelt szintű érettségire felkészítő tantárgyként választható informatika fakultáció, melyet az informatikus osztályokba járó tanulók nagy része választ, így ők heti két órában további informatika oktatásban is részesülnek.

A táblázat alapján jól látható, hogy a szakmacsoportos alapozó oktatás óraszámának több mint fele a Számítógép-programozás tantárgy óraszáma. Ez a négy év alatt összesen 562 óra, melyből 180 elméleti és 382 gyakorlati óra. Ehhez adható még az informatika fakultáció óraszámának egy része a tizenegyedik és a tizenkettedik évfolyamon, ha emeltszintű informatika érettségire készülnek a tanulók.

2. A programozás az érettségi vizsgán

Az informatika jellegű tantárgyak magas óraszáma és az informatika fakultáció választása ellenére is kevés informatikus tanuló választja az emelt szintű érettségit, mert jelen helyzetben az egyetemek és főiskolák informatikus szakjaira viszonylag alacsony pontszámmal be lehet kerülni, így a tanulóknak nincs szüksége az emelt szintű érettségiért járó plusz pontokra, illetve a plusz pontokat inkább közép szintű nyelvvizsga alapján szerzik meg.

Az informatikai képzésben résztvevő tanulók az érettségi vizsga szabadon választható tantárgyaként szinte kizárólag az informatikát választják az informatikai alapismeretek helyett. Történik ez annak ellenére, hogy a szakmacsoportos alapozó oktatás keretében az elsajátított ismeretanyag alapján az utóbbi tantárgyat is választhatnák. Joggal vetődhet fel a kérdés, hogy miért van ez így? Ennek okait az alábbiakban látom:

1) A diákok a „közismereti” informatikát könnyebbnek ítélik. A tanulók jelentős részének két témakör okoz igazán nehézséget. Ezek a hardver-ismeretek és a programozás. Ezekkel a témakörökkel nem találkoznak az informatika érettségi vizsgán, ha középszintű vizsgát kívánnak tenni. Ha az emeltszintű informatika érettségi vizsgát választja a tanuló, akkor is csak a programozás szerepel az írásbeli vizsgafeladatokban. A szóbeli követelményében megtalálható ugyan egy kevés hardverismeret is, de ez egyrészt elenyésző a többi kedveltebb témakörhöz képest, valamint az érettségi vizsga eredményét az írásbeli részen mutatott teljesítmény nagyon meghatározza. Az írásbeli vizsgarészen 120, a szóbeli vizsgarészen pedig mindössze 30 pont szerezhető. Az informatikai alapismeretek tantárgyi érettségi vizsga ellenben mindkét, a diákok által nehéznek minősített témakör jelentős szerepet kap.
2) A szakirányú egyetemi és főiskolai továbbtanulásnak általában nem feltétele a szakmai alapozó tantárgyból eredményes érettségi vizsga.
3) Az előző években a szakmai érettségi tantárgyak versenyén az országos döntőben résztvevő tanulók a versenyen elért eredményük alapján emelt vagy középszintű jeles, 100%-os érettségi minősítést kaphattak a szakmacsoportos alapozó tantárgyból. Ebben a tanévben a versenyszabályzat alapján már ez a

lehetőség sem adott, így nem is jelentkeznek informatikai alapismeretek érettségire a tanulók.

4) Az előző három ok csak úgy teljes, ha megjegyezzük azt is, hogy a diákok jellemzően a kisebb ellenállás irányát, a kisebb nehézséggel járó lehetőségeket részesítik előnyben.

A leírtak alapján úgy érezhető, hogy a Mechwart András Gépipari és Informatikai Szakközépiskola informatikai szakmacsoportba járó tanulók nem szeretik a programozást. Ez csak részben igaz. A grafikus, gyakran látványos alkalmazói felülethez szokott emberek nem kedvelik a régebbi hatást keltő karakteres felhasználói felületeket, és ez igaz akkor is, ha a programfejlesztői környezet rendelkezik a karakteres felületre jellemző tulajdonságokkal. A szakközépiskolai programozás oktatásának jelentős része ilyen fejlesztői környezetben történik. Elég, ha csak az olyan legelterjedtebb programozási nyelvekre gondolunk, mint a Turbo Pascal, a C, vagy akár a C++, bár ezek ingyenes fejlesztői környezetben is megjelentek már a windowsra szabott felülettel is. Mindezek mellé sorolhatjuk azt is, hogy a programozás tanulásának kezdete sok, a későbbiekben elengedhetetlen elméleti tudás megszerzését is igényli, és valljuk be, hogy a szakközépiskolás tanulók nem az elméleti tananyagokat kedvelik a legjobban.

Ezen, a diákok többsége számára barátságtalan helyzeten javíthatnak a grafikus felülettel rendelkező programfejlesztő környezetek és a 4GL rendszerek. Azonban ezek alkalmazása esetén sem feledhetjük el, hogy a megfelelő elméleti tudás ebben az esetben sem nélkülözhető, de legalább a fejlesztő eszközök és a grafikus alkalmazás készítése barátságosabbá teheti és teszi is a programozás folyamatát.

A kerettanterv és a Nemzeti Alaptanterv előírásai alapján a programozás az informatika oktatásában részvevők számára elengedhetetlen, csakúgy, mint az algoritmikus gondolkodás fejlesztése, a problémamegoldó képesség fejlesztése, bár az érettségivel záruló képzés esetén csak az emeltszintű vizsgán kap nagy jelentőséget.

A szakközépiskolák 9-12. évfolyama számára meghatározott fejlesztési feladatok az informatika tantárgy az algoritmizálás, adatmodellezés, problémamegoldás területén a Magyar Közlöny 2008/20/II. számában [3] közöltek szerint a következők:

TÉMAKÖRÖK	TARTALMAK	BELÉPŐ TEVÉKENYSÉGEK
Algoritmizálás, adatmodellezés	Algoritmusok a gyakorlatban. Algoritmusok leírása általános eszközökkel. Programozási nyelvek és a programok. Egy fejlesztő rendszer használata. Elemi adattípusok. Konstansok és változók. Adatok bevitele, tárolása és megjelenítése, közlése. Képletek és függvények használata. Feltételes elágazások. Ciklusok típusai. Összetett adattípusok, tömbök. Típusalgoritmusok. A programkészítés lépései.	Hétköznapi tevékenységsorok, algoritmusok leírása. Adott feladat megoldásához algoritmus tervezése. A szükséges adatok és eredmények megtervezése. Elemi és összetett adatok, képletek és függvények használata. Elágazások és ciklusok alkalmazása. Fejlesztő rendszer használata. Típusalgoritmusok alkalmazása életszerű, gyakorlati feladatok megoldása során. (Összegzés, keresés, megszámlálás, rendezés.) Például, oldjunk meg a szakmai irányultságnak megfelelő feladatokat, modellezzünk egyszerű rendszereket. A kész program tesztelése és alkalmazása.
Folyamatok és rendszerek modellezése	Problémák megoldása számítógépes modellekkel.	Egyszerű modellek megismerése vagy fejlesztése, „kísérletezés” a modellekkel. Az eredmények megfogalmazása.
Problémamegoldás	Informatikai eszközök és módszerek kiválasztása és komplex alkalmazása. Tantárgyi és iskolai problémák, feladatok megoldása. Számítógéppel irányított vagy intelligens rendszerek.	Problémamegoldó tevékenység tervezése. Problémák megoldása egyénileg vagy csoportban. Kooperatív munkák, projektek. Intelligens eszközök, robotok és/vagy virtuális eszközök szabályozása, tanítása, irányítása.

A részletes érettségivizsga-követelmény a tartalomorientált kompetenciák között így szól az algoritmizálásról, modellezésről és a programozás eszközeiről:

Algoritmizálás, adatmodellezés

- A tanuló legyen képes egy programozási feladatot szabatosan megfogalmazni;
- tudjon pontos feladatmeghatározás után adatmodellt felállítani;
- tudjon használni legalább két algoritmust leíró eszközt;
- tudjon a megoldandó feladathoz algoritmust készíteni;
- legyen képes algoritmusok számítógépes megvalósítására, az elkészült algoritmus helyességének ellenőrzésére!

A programozás eszközei

- A tanuló legyen képes egy programozási feladatot adott programozási nyelven megoldani;
- legyen képes használni egy programozási nyelv fejlesztői környezetét;
- legyen képes tesztelni programját, hibát keresni, majd javítani benne!

Mindezek mellett az informatika érettségi vizsgán csak emelt szinten jelenik meg az algoritmizálás és a programozás:

Az informatika érettségi vizsga részletes érettségivizsga-követelményei [3]

Algoritmizálás; adatmodellezés, programozási ismeretek (csak emelt szinten)

TÉMÁK	VIZSGASZINTEK	
	Középszint	Emelt szint
10.1. Elemi és összetett adatok, állományszervezés, relációs adatstruktúrák 10.1.1. Egész és valós számok, logikai értékek, rekord, halmaz 10.1.3. Állományok karakterek 10.1.2. Szöveg, sorozat, tömb,		Ismerje az adattípusok osztályozásának lehetséges fajtáit. Tudjon különbséget tenni egyszerű és összetett típusok között. Tudja a felsorolt összetett típusokat definiálni. Ismerje az egyes típusokhoz tartozó műveleteket. (Numerikus, logikai, karakter-, ill. szövegműveletek; továbbá tömbből elem kiválasztása indexével, rekordból mező kiválasztása nevével, halmazműveletek; szekvenciális állományokra alkalmazható műveletek)
10.2. Elemi algoritmusok típusfeladatokra 10.2.1. Összegzés, eldöntés, kiválasztás, keresés, megszámlálás, maximum-kiválasztás, kiválogatás, elemi rendezések		Ismerje a strukturált programozás alapelveit, a lehetséges programszerkezeteket. Tudja a szükséges változókat kiválasztani, és programbeli használatukat szabatosan megfogalmazni. Tudja pontosan leírni az egyes típusfeladatok kiinduló állapotát (azaz felsorolni az értékkel rendelkező változókat és tulajdonságukat) és a várt eredményt (azaz mely változóba, milyen feltételek mellett, milyen értékeket kell visszaadnia a programnak). Tudja leírni a megfelelő algoritmusokat valamely algoritmus-leíró nyelven.
10.3. Rekurzió 10.3.1. Rekurzió a feladatok és az algoritmusok világában		Ismerje a rekurzió fogalmát. Néhány egyszerű rekurziós feladaton tudjon bemutatni a rekurzív algoritmusokat.
10.4. A programkészítés mint termék-előállítási folyamat 10.4.1. A programkészítés lépései: feladatmeghatározás, tervezés, kódolás, tesztelés, hibakeresés, hatékonyság- és minőségvizsgálat, dokumentálás		Világosan lássa a tervezés és a kódolás közötti különbséget. Tisztában legyen a tesztelés szerepével és alapelveivel. Tudjon adott feladathoz olyan tesztadatokat meghatározni, amelyek a hibás működés kiszűrésére alkalmasak.

10.5. Számítógép a matematikában, a természet- és társadalomtudományi tantárgyakban 10.5.1. Matematikai feladatok, egyszerű természettudományos szimulációs problémák, a középiskolai tantárgyakkal kapcsolatos egyszerű feladatok megoldása		Tudjon programot készíteni a felsorolt tantárgyak köréből megfogalmazott probléma megoldására, ha a megoldó módszerről részletes leírást kap.

A programozás eszközei (csak emelt szinten)

TÉMÁK	VIZSGASZINTEK	
	Középszint	**Emelt szint**
11.1. Algoritmusleíró eszközök 11.1.1. Feladatmegoldás egy algoritmus-leíró eszköz segítségével 11.1.2. Az algoritmus-leíróeszközök fajtái		Ismerje a struktogramot vagy a folyamatábrát, és a mondatszerű algoritmus-leíró eszközt. Tudjon az egyikkel programot tervezni.
11.2. Programozási nyelv 11.2.1. Egy programozási nyelv részbeni (specialitások nélküli) ismerete		Ismerjen egy programozási nyelven: típusdefiníciót, változódeklarációt, input és output utasításokat, alapvető programszerkezeteket (azaz szekvenciát, elágazást, ciklust), eljárásokat, állományból adatbeviteli és -kiviteli műveleteket.
11.3. Programfejlesztői környezet 11.3.1. Kódolási, szerkesztési eszközök valamilyen programnyelvi fejlesztői környezetben 11.3.2. Programkipróbálási eszközök valamilyen programnyelvi fejlesztői környezetben		Tudjon egy közepes nehézségű, de összetett feladatot strukturáltan megoldani az ismert programnyelven. Tudjon e felhasználóval kulturáltan kommunikáló adatbevitelt és adatkivitelt írni. Legyen képes a program különböző kimeneteinek tesztelésére alkalmas mintaadatokat adni. Tudjon nyomkövetéssel programot tesztelni.

Az informatikai alapismeretek érettségi vizsga a közismereti informatika vizsgával szemben mind közép mind emelt szinten elvár megfelelő szintű algoritmizálással és programozással kapcsolatos ismereteket. Az érettségi tantárgy részletes követelményei az algoritmizálásról, modellezésről és a programozás eszközeiről alábbiak ismeretét kérik:

Az informatikai alapismeretek érettségi vizsga részletes követelményei [3]

A) KOMPETENCIÁK

Programozási alapismeretek

TÉMÁK	VIZSGASZINTEK	
	Középszint	**Emelt szint**
7.1. Programozási alapismeretek	- Ismerje a programozási lépéseket. - Ismerjen egy programfejlesztő környezetet. - Ismerje a programozási környezet alapvető beállításait. - Értse egy egyszerű feladat algoritmusának az elkészítését és leírását valamilyen algoritmus-leíró eszköz használatával. - Alkalmazza az algoritmus szerinti program készítését. *(A további követelmények erre a programra vonatkoznak!)* - Értse egy félkész program folytatását, illetve javítását. - Értse a hibakeresést és a hibajavítást. - Alkalmazza az eredménynek képernyőn, vagy nyomtatón történő megjelenítését.	
7.2. Algoritmusok	- Értse az algoritmus-leíró eszközök használatát. - Alkalmazza az algoritmus-leíró eszközöket egyszerű feladatok elkészítéséhez és leírásához (mondatszerű leírás, struktogram). - Ismerje a rekurzió fogalmát.	
7.3. A program-készítés környezete	- Ismerje a használt fejlesztői környezetet és követelményeit.	
7.4. Programnyelv ismerete	*Értékadás, változók, konstansok* - Alkalmazza az elemi adattípusokat (egész és valós számok, logikai, karakteres, és mutató típusok). - Értse az összetett adatszerkezetek közül a tömböt. - Ismerje a szövegek (string, karakterlánc) kezelését. - Ismerje a verem, a sor és a lista struktúrákat. - Alkalmazza a megfelelő adattípust. *Vezérlési szerkezetek (Szekvenciák, elágazások, ciklusok):* - Alkalmazzon szekvenciákat. - Alkalmazzon elágazásokat. - Alkalmazzon különböző ciklusokat. (Tesztelés helye, feltételes elágazás tervezése.)	- Értse az összetett adatszerkezeteket. (Halmaz, tömb, rekord, fájl.) - Értse a rekurzív algoritmusokat.

	Függvények, eljárások - Alkalmazza a programkönyvtár fontosabb eljárásait. (Alkalmazási feltételek, paraméter átadás.) - Értse a függvények hívását. *Adatállományok:* - Ismerje a típusos állományokat (létrehozás, írás, olvasás, módosítás). - Alkalmazzon szöveges állományokat. (Írás, olvasás, hozzáfűzés szabványos eszközökkel.	- Alkalmazzon saját függvényeket és eljárásokat.
7.5. Programozási tételek	- Ismerje a programozási tételek tartalmát, alkalmazási körét. - Értse egyszerű feladatok programjainak az elkészítésénél az összegzés, eldöntés, szekvenciális keresések, szélsőérték kiválasztás, megszámlálás, belső rendezések, egyesítés, szétválogatás, kiválogatás, összefuttatás tételeinek alkalmazását. - Tudja alkalmazni a programozási alaptételeket feladatok megoldásához.	
7.6. Hibakeresés, tesztelés, hatékonyság	- Ismerje a töréspontok elhelyezésének és a lépésenkénti végrehajtásnak a módszereit.	
7.7. Az objektumorientált programozás alapjai	- Ismerje az objektum-létrehozást valamilyen vizuális nyelven. - Ismerjen egy vizuális fejlesztő eszközt egy elemi program létrehozására.	- Tudjon form- és kontrolltulajdonságokat meghatározni. - Legyen képes formot létrehozni és azon eszközöket elhelyezni. - Értse az eseményfigyelést és -kezelést. - Ismerje az objektumokhoz rendelt metóduskészletből való kiválasztást. - Ismerje a különböző metódusok és események hatását.

B) TÉMAKÖRÖK

Programozási alapismeretek

TÉMÁK	**VIZSGASZINTEK**	
	Középszint	**Emelt szint**
7.1. A feladat-megoldás lépései, módszerei	- Ismerje a hagyományos programozás lépéseit. - (Feladat meghatározása; Specifikációk, változók és típusok, a kiinduló és a végső állapot rögzítése; algoritmus készítése és rögzítése valamilyen módszerrel; kódolás; tesztelés,	

	hibakeresés javítás; hatékonyság vizsgálat; dokumentálás.) - Ismerje a fordítás és szerkesztés folyamatát. - Ismerje a strukturált programozás alapelveit.	
7.2. Algoritmusok	- Ismerjen az algoritmus-leíró eszközöket (mondatszerű leírás, folyamatábra, struktogram). - Értse programok tervezéséhez az algoritmus-leíró eszközök használatát. - Ismerje a rekurzió fogalmát.	- Alkalmazza az algoritmus-leíró eszközöket programok tervezéséhez. - Alkalmazza szövegesen megfogalmazott feladat megoldásához valamelyik algoritmus-leíró eszközt. - Ismerje a programelemzést.
7.3. A programkészítés környezete	- Ismerje a programkönyvtárak szerepét.	- Értse a program használójával kulturáltan kommunikáló adatbevitelt és adatkivitelt.
7.4. Programnyelv ismerete	- Ismerje a programkód és programnyelv fogalmát. - Ismerje a kódolás és a forrásprogram fogalmát. - Ismerje a programnyelvek generációit, típusait. - Alkalmazza az alapvető programszerkezeteket a tanult programozási nyelven (szekvencia, elágazás, ciklus). - Ismerje az utasítás, az adatok, a függvények, az eljárások fogalmát, célját a tanult programnyelvi környezetben. - Alkalmazza az elemi adattípusokat (egész számok típusa, valós számok, logikai adattípus, karaktertípus, mutató típus, felsorolás típus). - Értse a változók élettartamának és érvényességi körének fogalmát. - Ismerje a fontosabb alapfüggvényeket, a paraméter-átadás alapjait és a hívási szabályokat. - Értse az összetett adatszerkezetek használatát (szöveg és tömb). - Ismerje a verem, a sor és a lista struktúrákat. - Ismerje az egyes típusokhoz tartozó műveleteket. (Numerikus, logikai, karakter- és szövegműveletek, tömbből elem kiválasztása az indexével.) - Ismerje a strukturált feladatmegoldást.	- Értse a rekord, sor és lista adatszerkezeteket. - Értse az állományok definiálását. - Ismerje az egyes típusokhoz tartozó műveleteket (rekord elemének kiválasztása a nevével, halmazműveletek, szekvenciális és direkt állományokra alkalmazható műveletek). - Értse a külső programmodulok beépítését. - Ismerje az adattípusok osztályozását.

7.5. Programozási tételek	- Értse a legfontosabb programozási tételeket: (összegzés, eldöntés, - szekvenciális keresések, kiválasztás, szélsőérték kiválasztás, megszámlálás, metszetképzés, belső rendezések, egyesítés, szétválogatás, kiválogatás, összefuttatás). - Ismerje valamely algoritmus-leíró nyelven a kiválasztott algoritmus leírását. - Értse a programok előfeltételének és utófeltételének fogalmát. - Értse a szükséges változók kiválasztását és a programban történő szabatosan megfogalmazását.	- Alkalmazza a strukturált feladatmegoldást a tanult programnyelven. - Ismerjen rekurziót bemutató algoritmust.
7.6. Hibakeresés, tesztelés, hatékonyság	- Ismerje a tesztelés szerepét és alapelveit. - Értse a hibás működés kiszűrésére alkalmas mintaadatokat. - Értse a nyomkövetéssel történő programtesztelést. - Értse a tervezés és a kódolás közötti különbséget.	- Alkalmazza a hibás működés kiszűrésére alkalmas mintaadatokat. - Ismerje a hatékonyságvizsgálatok szempontjait.
7.7. Az objektumorientált programozás alapjai	- Ismerje a modern vizuális programozás alapjait, a vizuális objektumok fogalmát.	- Ismerje az objektumorientált programozás elvét, az objektum fogalmát, szerkezetét. - Ismerje az adatmező, a metódus és az osztály fogalmát és jelentőségét. - Értse az objektumokhoz kapcsolódó metódusokat. - Ismerje a függvényt, és az eljárást. - Ismerje az objektumok általános felépítését, kialakításuk elvét, tartalmát, az öröklődés fogalmát. - Ismerje az objektumok azonosítását, a jellemző közös tulajdonságokat kiemelését, kapcsolatok kialakítását. - Ismerje a modern vizuális programozás alapjait, a vizuális komponensek fogalmát, főbb típusait. - Ismerje az eseményvezérlést.

A részletes követelmények alapján valóban felismerhető, hogy a magas szakmacsoportos alapozó oktatás óraszáma mellett, ha a tanulók középszintű informatika érettségit választanak, a programozás elkerülhető. Ha figyelembe vesszük, hogy a szakközépiskolai képzés akkor teljes, ha az érettségi után további egy évre itt

maradnak a tanulók valamilyen szakképesítés megszerzésére, akkor a programozásra fordított magas óraszám nem felesleges, mert a szakképző évfolyamon egyes szakmák megszerzéséhez programozási ismeretek is szükségesek. A közelmúltban egészen az előző tanévig az iskolánkban megszerezhető volt a Multimédia-fejlesztő szakképesítés, melyben a vizuális programozás meghatározó szerepet töltött be. A jelenleg átalakulóban lévő szakképzési struktúra miatt ez a szakképesítés már nem szerezhető meg az iskolánkban, de helyette az informatikai alkalmazásfejlesztő, szoftverfejlesztő szakképesítés igen, melyben szintén fontos szerepet kap a programozás.

Az algoritmizálás, programozás megfelelő szintű ismerete azonban nem csak a szakképző évfolyamon fontos, hanem az érettségi után, vagy a szakma megszerzését követő informatika irányú továbbtanulás során is.

3. A programozási nyelv választása az iskolában

Az intézményemben a számítógép-programozás tantárgyak óraszámának kis része a tízedik évfolyamon, és egy nagyobb része a tizenkettedik évfolyamon nem a programozás oktatására kerül felhasználásra. Az iskola a minőségirányítási programjában meghirdette az „ECDL bizonyítványt minden diáknak” programot. Ennek teljesítése érdekében az informatikus osztályok a szükséges hét modulból öt vagy hat modult teljesítenek a tízedik évfolyam végéig. Ehhez az informatika órák mellett a szakmacsoportos alapozó oktatás óraszámának egy része is felhasználásra kerül. A tizenkettedik évfolyamon a szakmacsoportos alapozó oktatás számítógép-programozás III nevű tantárgyának keretében a programozás mellett a hálózati ismeretek, a weblapkészítés és az érettségire történő felkészítés is szerepet kap.

A fentiek figyelembevételével a programozással kapcsolatos ismeretek oktatása és ennek tényleges óraszáma az alábbiak szerint alakul a szakmacsoportos alapozó oktatás keretében. A táblázatban csak a programozásra és algoritmizálásra fordítandó éves óraszámok szerepelnek, egyéb témakörökkel, mint például szövegszerkesztés, táblázatkezelés, prezentáció, stb. most nem foglalkozok.

Évfolyam		**Témakörök, óraszámok**	
9.	*Elmélet*	-	-
	Gyakorlat	-	-
10.	*Elmélet*	Algoritmusok	16
	Gyakorlat	Algoritmusok, Pascal	16
11.	*Elmélet*	Algoritmusok Pascal Vizuális programozás	10 64 37
	Gyakorlat	Pascal Vizuális programozás	111 74
12.	*Elmélet*	Pascal Vizuális programozás	10 10
	Gyakorlat	Pascal Vizuális programozás	24 16

A táblázatból látható, hogy a tizenegyedik és a tizenkettedik évfolyamon párhuzamosan történik egy harmadik és egy negyedik generációs fejlesztői környezetben történő programozás oktatása. Ez úgy valósul meg, hogy a heti 3 elméleti

órából 2 óra az algoritmusok és a Pascal nyelv oktatásának óraszáma, 1 pedig a vizuális programozás óraszáma. Ez az arány a heti 5 gyakorlati óra esetében 3 óra illetve 2 óra.

A felsőbb évfolyamon történő hatékony párhuzamos oktatás feltétele, hogy a tizedik évfolyamon olyan alapismereteket szerezzen a tanuló az algoritmizálás és az adott programozási nyelv használatából, hogy ez alap lehessen a következő évfolyam ilyen irányú tanulásához. A tizenegyedik évfolyamon a vizuális programozás gyakorlati oktatása az egyszerű, előző évben megismert algoritmusokra és szabályokra épül, miközben az elméleti órán a 4GL fejlesztő környezetek alapfogalmait, az eseményvezérlés és a vizuális tervezés elvét valamint az objektumorientáltság fogalomrendszerét ismerik meg a tanulók. Ezen bevezető időszak során ezzel párhuzamosan a vizuális programozáshoz szükséges alap programozási nyelvet még jobban megismerik a tanulók. Ekkortól kezdve már mélyebben is taníthatók a vizuális fejlesztés elemei, komponensei, ezek tulajdonságai, metódusai és eseményei. Egyre inkább felhasználóbarát, látványos programok készíthetők. Ekkor érkezhetünk el programozás tanítása során ahhoz a ponthoz, amelyet már az előzőekben említettem. Azok a tanulók, akik érdeklődése a programozás iránt csökkent a harmadik generációs fejlesztői környezet és nyelv használata során, mostantól ismét sokkal jobban motiválhatóvá válnak.

A hatékony vizuális programozás oktatásának iskolámban feltétele, hogy az adott vizuális fejlesztő környezet által használt alap programozási nyelvet, illetve annak alapjait a tanulók már megfelelő mértékben ismerjék. Ez a feltétel általában teljesül. Az alap programozási nyelvhez alkalmazott környezet kifejezetten alkalmas a nyelv szintaktikai elemeinek megtanulására, a vezérlési szerkezeteinek megismerésére. Az alapvető probléma, hogy ezt a környezetet használva a programozást a diákok előbb-utóbb összekapcsolják a sok-sok gépeléssel, és egy idő után már nem a probléma megoldása jelent számukra kihívást, hanem a nagy mennyiségű gépelés válik unalmassá. A tanítási órákon ilyen keretek között csak viszonylag kisebb feladatok oldhatók meg a folyamatos tanári értékelés igénye mellett. Általában egy összetettebb probléma megoldására nem elegendő egy-egy alkalom, ezzel a munka folyamatossága megszakad, főként ha azt is figyelembe vesszük, hogy mindig van egy-két hiányzó tanuló, aki emiatt nem tud megfelelő ütemben haladni az adott összetett probléma megoldásával. Ezek miatt ritkán fordul elő egy-egy nagyobb munka elkészítése.

A vizuális programozás alkalmazása sok kódadminisztrációtól megkíméli a program készítőjét, és ezt a tanulók nagy örömmel tapasztalják. Könnyebb a program látványképének megtervezése, elkészítése, és könnyen látványossá tehető a program. A diákok számára ezek fontos dolgok, de azzal nekik is mindenképpen tisztán kell látniuk, hogy ha nem ismerik az alapnyelvet megfelelő mértékben, akkor még ebben a sok támogatást nyújtó fejlesztői környezetben sem tudnak eredményesen programozni. A Kecskeméti Főiskola Gépipari és Automatizálási Műszaki Főiskolai Karán 2007-ben végzett felmérés szerint a vizuális fejlesztést hallgató diákok lényegesen többet tudnak a programok szerkezetéről, működéséről. Köszönhető ez a tervezés megjelenésének a tananyagban és annak, hogy a fejlesztő eszközök automatikusan modulokra tördelik a programokat [4].

A tapasztalataik azt mutatják, hogy a programozás manapság valóban nem gépelést jelent, hanem komoly elméleti alapokra helyezett szerkezet tervezést és kódgenerálást. A hagyományos fejlesztéssel a szintaktikát meg lehet tanítani, de csak olyan kisméretű feladatokat lehet megoldani, amin keresztül az elmélet sem mélyül el.

Sokak szerint a vizuális szoftverfejlesztéssel nem lehet programozás oktatást kezdeni, mert a fejlesztő eszköz elsajátítására megy el a drága és kevés idő, és közben nem tanulnak meg programozni. Ennek ellensúlyozására történik iskolámban a programozás párhuzamos tanítása a két fejlesztési környezettel.

Felmerül a kérdés, hogy melyik programozási nyelven történjen az algoritmizálás és a programozás tanítása. Ez egy olyan kérdés, melyre valószínűleg nagyon sok féle választ el tudnánk fogadni. A következőkben egy középiskolai tanár szemszögéből nézve próbálok válaszolni erre a kérdésre. A középiskolás korosztály számára véleményem szerint a legelterjedtebb alapnyelvek közül kell választani. Ezek a Basic, a Pascal és a C.

A Basic nyelv mai jelentőségét a Microsoft szoftverei miatt tartom fontosnak, de csak a Visual Basichez szükséges ismeretek miatt.

Az alapok megértéséhez mindenképpen a Pascal-t tartom jónak, mert:

- a programozási alapelemek (adattípusok, vezérlési struktúrák) rendelkezésre állnak;
- erősen tipizált, ezért módszeres gondolkodásra szoktat;
- emberbarát nyelv;

- az erősen kötött szintakszisa miatt a fordító majdnem mindig a tényleges helyen a tényleges hibát jelzi.

A Pascal egy nagyon jó oktató nyelv azok számára, akik igazából most találkoznak a programozással nagyobb óraszámban, és csak most kezdik a programozással az ismerkedést, vagy most kezdik elmélyíteni az általános iskolában megszerzett ismereteket.
Csak ezután támogatnám a C-re történő áttérést, mert:

- bár a szintaxis másabb, de az eddig megismert dolgok jelentős része átírható ide is;
- a hatékony programozáshoz elengedhetetlen dinamikus memóriakezeléssel kapcsolatos része (mutatók, foglalás, felszabadítás) következetesebb, sokrétűbben alkalmazható, mint a Pascalban;
- a többmodulos fejlesztési része szintén jobban kiaknázható, nem olyan aszimmetrikus, mint Pacalban a program és a unit viszonya;
- nagyon sok programozási nyelv használ a C-hez hasonló szintaktikát és programszerkezeteket;
- közelebb áll a rendszerprogramozás megoldásaihoz.

Az erősen célorientált nyelveket legkorábban a C megtanulása után említeném, mert addigra szerezhető meg elég tudás az adatszervezéshez és az algoritmusban gondolkozáshoz.

Az iskolámban egy programozási nyelv magtanulására van lehetőség, és erre épül majd a vizuális programozás is. A részletesebben említett fenti két nyelven kívül nagyon sok programozási nyelv létezik még, de a két leginkább preferált Pascal és a C alapú nyelvek között kell tehát döntenünk.

Ezen előnyök mellett mindenképpen meg kell említeni, hogy egy programozó a C nyelvhez hasonló felépítésű nyelvet jobban ki tud aknázni, ellenben a típusok és mutatók itt jellemző használata miatt a középiskolás tanulók többségének sokkal nagyobb nehézséget jelent a programozás, mint Pascalban, és a középiskolában az alapképzés során nem képzett programfejlesztőket és programozókat kívánunk nevelni, hanem olyan tanulókat, akik képesek lehetnek egyetemen megtanulni az ilyen jellegű programozást a hozzá szükséges nyelvekkel együtt. Ugyanakkor gondolni kell azokra a tanulókra is, akiket ezen a téren kevesebb érzékkel áldott meg a sors. A programozás

iránt fogékonyabb tanulók számára is kisebb-nagyobb problémát okozó nyelv elsajátítása esetükben gyakran csak a sikertelenséget jelentené, ezzel pedig egy igen komoly előítélet alakulhat ki bennük a programozás iránt. Ebben a tanévben egy véleményem szerint nem teljesen szerencsés döntés eredményeként ezt sajnos tapasztalhatjuk az iskolámban. Ennek az iskolai döntésnek a hátterében az Oktatási Hivatal azon döntése volt, hogy 2010-től az érettségin használható szoftverek listájából törlik többek között a Turbo Pascal-t, a FreePascal-t és a Delphi 6 Pascal illetve Object Pascal alapú fejlesztői környezeteket is, és így Pascal alapú környezetként csak a Turbo Delphi lesz választható [5]. Ez a programozás tanítását, amely eddig iskolámban teljesen Pascal alapú volt, egy munkaközösségi döntés miatt jelentősen megváltoztatta. Történt ez annak ellenére, hogy a Turbo Delphi környezetben már évek óta történik oktatás, és az érettségi feladatai a benne használható konzol-alkalmazás segítségével egy-egy kisebb változtatás megismerése mellett tökéletesen megoldhatóak lennének. A munkaközösség úgy döntött, hogy a most tizenegyedikes tanulók a programozást C nyelven tanulják.

Jelenleg ezekben az osztályokban a programozás tantárgyi átlaga jelentősen elmarad az előző évekétől, jelezve, hogy aki fogékony a programozás iránt, annak nem okoz nagy nehézséget, aki nem fogékony, annak gyakorlatilag mindegy, a középszint viszont inkább lefelé tolódott el, kevésbé képes elsajátítani a szükséges ismereteket.

Az Oktatási Hivatal döntéséről véleményem az, hogy ez egy mesterséges irányítása a programozási nyelvek tanításának. Főleg nem tudok egyetérteni vele, ha megfigyeljük, hogy amíg az érettségizők több mint fele Pascalt használt. Ilyen arány mellett nem tartom szerencsésnek kirántani a talajt a tanárok, és azon diákok alól, akiket ebbe az irányba indítottak el, még akkor sem, ha a Turbo Delphi konzolalkalmazásának lehetőségét meghagyták.

A tanév során az Oktatási Hivatal módosította szándékát, és a FreePascalt visszahelyezte a választható szoftverek listájára.

4. A vizuális programozás tanítása

Az előző fejezetekben látható volt, hogy a vizuális programozás tanításának meghatározó része tizenegyedik évfolyamon történik. Ezen az évfolyamon a tervezett vizuális programozás témakör 37 elméleti órája és 74 gyakorlati órája az alábbi tananyagrészeket tartalmazza:

Hét	Elméleti óra (1 óra/hét)	Gyakorlati óra (2 óra/hét)
1	Az objektumorientált programozás alapfogalmai	Balesetvédelem, laborrend
2		Felépítés, kezelés alap
3	A 4GL alapfogalmai, jellemzői	Formok tulajdonságai
4	Project-ek felépítése, részei	Formok tulajdonságai
5	Komponensek ismertetése, alkalmazási lehetőségei, Tform, TButton	Formok eseményei
6	TLabel, TFont, TEdit, Tpanel, TXPManifest	Események létrehozása (Formon)
7	Stringlisták (TListBox, TComboBox, TMemo)	Projectek mentése
8	TCheckBox, TRadioButton, TRadioGroup	Az elméleti órákon megismert komponensek alkalmazásával programok készítése
9	Párbeszédablakok(TOpenDialog, TSaveDialog)	
10	Párbeszédablakok (TFontDialog, TColorDialog)	
11	TbitBtn, TGroupBox, TBevel	
12	Üzenetablakok	
13	Konverziók	
14	Inputablak, Inputfigyelés	
15	Formok közötti kommunikáció, modális formok	
16	Ablakok bezárása	
17	TToolBar, TSpeedButton	
18	TactionList, TClipBoard	
19	TMainMenu, TPopupMenu	
20	TShape, TscrollBar, TTrackBar	
21	TStringGrid	
22	Timage, TImageList	
23	TOpenPictureDialog, TSavePictureDialog, TOpenTextFileDialog, TSaveTextFileDialog	
24	TFileListBox, TDireectoryListBox TDriveComboBox, TFilterComboBox	
25	ColorListBox, TTimer, TUpDown	
26	TPrintDialog, TPrinterSetupDialog	
27	TCanvas	
28	Az Object Pascal egyéb lehetőségei (Sender, is, as, with … do)	
29	Kivételkezelés	
30	TMediaPlayer	
31	Drag and Drop	
32	TRitchEdit	
33		
34	Dinamikus létrehozás és kezelés	
35	MDI alkalmazások	
36	Konzol alkalmazás	
37		

Ebben a részben rövid vázlat segítségével szeretném bemutatni, hogy mely alapfogalmakkal kell megismertetni a tanulót az objektumorientált programozási szemlélet megértéséhez és a 4GL környezetben történő programkészítéshez.

4.1. Az elméleti órák fontosabb elméleti tananyagrészeinek vázlata:

4.1.1. Az objektumorientált programozás alapfogalmai:

a) Az objektumorientált programozás történeti áttekintése
   - Az objektumorientált programozás egy elfogadott programozási filozófia.
   - A nyelvek két fejlődési trendet követnek:
     - Eljárásorientált nyelvek
       - A programozás alapvetően eljárásokra épül, az objektumok csak egy új eszközként, lehetőségként jelennek meg.
       - Például: C++, Turbo Pascal, Object Pascal.
     - Tisztán objektumorientált nyelvek (Smalltalk, Eiffel)

b) Az objektumorientált programozás alapfogalmai
   - Az objektumorientált programozás egy programozási paradigma
     - Paradigma: azon módszerek, elméletek, szabványok sokasága, melyek együtt az ismeretek ábrázolásának egy lehetőségét adják.
     - Az objektumorientált programozás a természetes gondolkodást közelítő programozási mód.
     - Egy objektumorientált program strukturáltabb, modulárisabb, mint egy hagyományos program, ezért könnyebb bővíteni, karbantartani.
   - Objektumorientált program: együttműködő objektumok (objects) összessége.
   - Objektumok:
     - Olyan program-összetevők, amelyek egységbe foglalják az állapotukat leíró belső adatszerkezeteket és a rajtuk értelmezhető műveleteket.
     - A programot objektumokból építjük fel. Az objektumoknak saját viselkedésük, működésük és lehetőleg rejtett belső állapotuk van.
     - Egy objektumra a környezetében lévő más objektumok hatnak, ennek hatására állapotuk megváltozhat.
     - állapotukat a tulajdonságaik (properties) határozzák meg.
       - Hivatkozás: `objektum.tulajdonság`
     - Viselkedésük, működésük leírása az objektumokban szereplő metódusok szolgálnak (általában eljárásokkal és függvényekkel implementáljuk).
       - Hivatkozás: `objektum.metódus`

- Minden objektum valamilyen osztályba (`class`) tartozik.

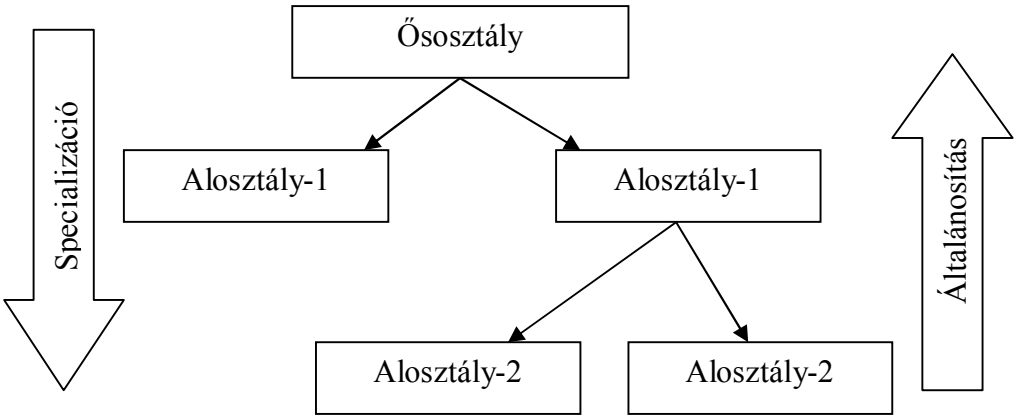
- Egységbe zárás
  - Az objektumokat, az adatok és a hozzá tartotó metódusok (eljárások, függvények alkotják).
  - ezek egy egységet alkotnak és a külvilág számára láthatatlanok.
  - Attribútumok, tulajdonságok: adatelemek
  - Metódusok: az objektum adatain elvégezhető műveletek.
  - Amikor kívülről hatás éri az objektumot akkor az adatok megváltoznak és a metódusok ezekkel az adatokkal végzik el a feladatot.
- Objektumosztály
  - Az azonos attribútumokat és metódusokat tartalmazó objektumok együttese.
  - Először mindig az osztályokat hozzuk létre és utána az objektumokat, mert ilyenkor minden egyes példány tartalmazza az osztály összes attribútumát és metódusát: ez az alapja az öröklődésnek.
  - Az osztályok definiálják az egyes objektumok állapotát leíró adatszerkezeteket és a rajtuk végezhető műveleteket (metódusokat).
  - Az egyes osztályokat az öröklődés hierarchiába rendezi.

```
Type   TForm1 = class(TForm)
       ...
       ...
var  Form1: TForm1;
```

- Láthatóság
  - A programozó megadhatja, hogy az általa definiált egyes osztályok illetve azok változói, módszerei milyen körben használhatók.
    - `public` (nyilvános) : Mindenhonnan látható.
    - `private` (magán) : Csak saját objektumon belül látszik.
    - `protected` (védett) : csak az osztályban és annak valamelyik leszármazottjában láthatók.
- Öröklődés
  - Az az eljárás, amely segítségével egy osztály felhasználhatja a hierarchiában felette álló osztályokban definiált adatszerkezeteket és módszereket.
  - Így a közös elemeket elég egyszer definiálni (a hierarchia megfelelő szintjén).

- Az alosztályokban ezeket a metódusokat módosítani lehet.
- Az alosztályok lehetnek ősei további osztályoknak, így azok metódusai tovább módosulhatnak.

➢ Öröklődési szerkezet



- Tekintsünk meg egy öröklődési láncot a TBitBtn osztály példáján keresztül:

Hierarchy

TObject
|
TPersistent
|
TComponent
|
TControl
|
TWinControl
|
TButtonControl
|
TButton

In TBitBtn
- Glyph
- Kind
- Layout
- Margin
- NumGlyphs
- Spacing
- Style

Derived from TButton
- Cancel
- Default
- ModalResult

Derived from TWinControl
- Brush
- ClientOrigin
- ClientRect
- ControlCount
- Controls
- DockClientCount
- DockClients
- DoubleBuffered
- Handle
- HelpContext
- ParentWindow
- Showing
- TabOrder
- TabStop
- VisibleDockClientCount

Derived from TControl
- Action
- Align
- Anchors
- BiDiMode
- BoundsRect
- Caption
- ClientHeight
- ClientWidth
- Constraints
- ControlState
- ControlStyle
- Cursor
- DockOrientation
- DragCursor
- DragKind
- DragMode
- Enabled
- Floating
- FloatingDockSiteClass
- Font
- Height
- Hint
- HostDockSite
- Left
- LRDockWidth
- Name
- Parent
- ParentBiDiMode
- ParentFont
- ParentShowHint
- PopupMenu
- ShowHint
- TBDockHeight
- Top
- UnDockHeight
- UnDockWidth
- Visible
- Width
- WindowProc

Derived from TComponent
- ComObject
- ComponentCount
- ComponentIndex
- Components
- ComponentState
- ComponentStyle
- DesignInfo
- Owner
- Tag
- VCLComObject

- Mi öröklődik?
  - Az új osztály örökli a szülő összes egyedváltozóját (ezek még akkor is meg vannak, ha egy `private` deklaráció miatt nem láthatók).

- Ha a leszármazott definiál egy a szülő által már használt változót, ilyenkor ez a változó eltakarja (shadowing) a szülő változóját. A takarás meg is akadályozható.
- A szülő módszerei is öröklődnek, akár nyilvános, akár védett módszerek.

➢ Osztályok leszármaztatása
  – Egy új osztályt már valamelyik meglévő alapján definiálhatunk.
  – A leszármazott osztály (Gyermek) örökli a szülője tulajdonságait, vagyis belső állapotát leíró adatszerkezetét és a viselkedést megvalósító módszereit. Ez az osztály törzsében tovább bővíthető, módosítható.
  – Minden osztálynak van szülője, a nyelv ugyanis definiál egy beépített Object nevű osztályt a hierarchia csúcsán.

➢ Többrétegűség
  – Az örökölt metódusokat az alosztályokban megváltoztathatjuk, a metódus neve azonban ugyanaz marad.
  – Ha nem akarjuk megváltoztatni a metódus működését, akkor a deklarációban nem kell felsorolni.
  – Ebben az esetben a metódus ugyanúgy működik, mint az ősosztályban.
  – Ha egy alosztálynál hivatkozunk egy metódusra, akkor:
    - Szerepel a deklarációban és az fog végrehajtódni (módosult működés)
    - Nem szerepel, örökölt metódus visszafelé megkeresi melyik ősben történt a deklaráció és végrehajtódik.

4.1.2. A 4 GL alapfogalmai, tulajdonságai:

a) A 4GL egy (vagy több) magasszintű nyelvre épülő komplex, objektumorientált programfejlesztői környezet.
  ➢ eseményvezérelt;
  ➢ a kezelői felület létrehozására speciális szerkesztőt, ún. látványtervezőt (vizuális tervező felületet) alkalmaz;
  ➢ minden 4GL eszköz objektumorientált;
  ➢ tervezési időben ami csak lehet, legyen látható;
  ➢ megkülönböztethető a tervezési idő és a futási idő;
  ➢ a megjelenítő objektumokat látványtervezőkkel helyezzük el fogadásukra alkalmas felületekre (Form-okra);

- grafikus környezetben a formok (lapok) általában egy-egy operációs rendszer szintű ablakban jelennek meg.
- A megjelenítő objektumok fajtái:
  - Passzív objektumok
    - tartalmuk, megjelenésük, és általában állapotuk is állandó;
    - kezelői eseményekre nem reagálnak, és adatokat nem jelenítenek meg;
    - az egyes lapok jobb áttekinthetőségét szolgálják, vagy emelik az alkalmazás „fényét”;
    - például: geometriai alakzatok (egyenes, kör, ellipszis, négyszög, sokszög, szabadkézi vonal), egyes feliratok, statikus ábrák (cégjelzés, háttérkép).
  - Aktív objektumok
    - az alkalmazás információfeldolgozási folyamatának szerves részét képezik;
    - tartalmuk időben változó;
    - ha egy aktív objektum e mellett kezelői beavatkozásokra is reagál, kezelőszervnek (control) nevezik;
    - szabványos aktív kezelőszervek például: nyomógombok (button), jelölőnégyzetek (checkbox), választógombok (radio button), szövegmezõk (edit), listák (listbox, combobox, memo), görgetõsávok (scrollbar), stb.

b) Beviteli eszközök kezelése
- Inputfókusz:
  - meghatározza, hogy (a parancsbillentyűket kivéve) a klaviatúra billentyűinek leütéséből származó események a kezelői felület melyik objektumára vonatkozzanak.
    - navigálási sorrend (set order)
    - megálljon-e az inputfókusz (tab stop)
- Vezérlés egérhasználattal
  - Kattintással kapcsolatos események (onClick, onDblClick, onMouseDown, onMouseUp, stb.)
  - Egér mozgatásának eseményei (onMouseEnter, onMouseLeave, stb.)

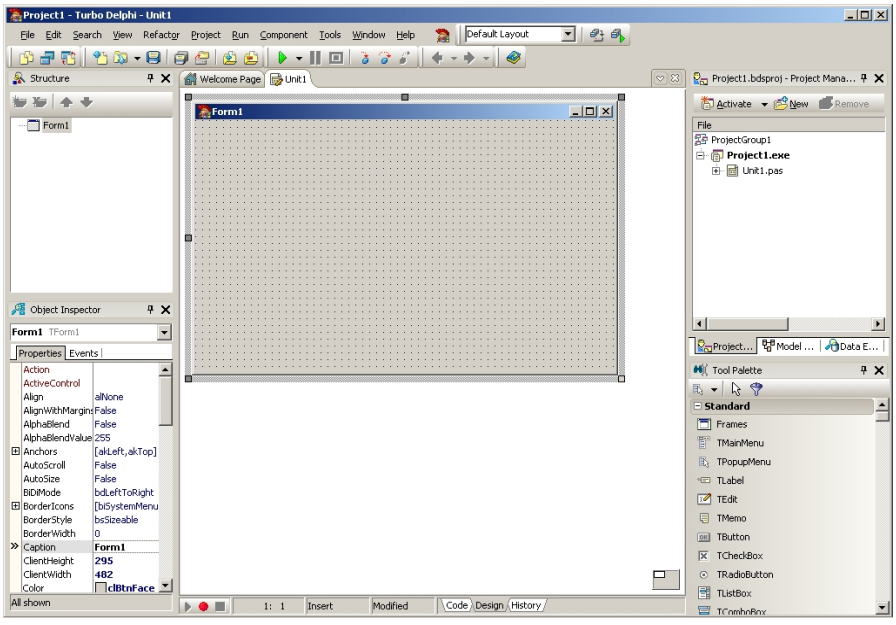
4.1.3. A Turbo Delphi alapfogalmai

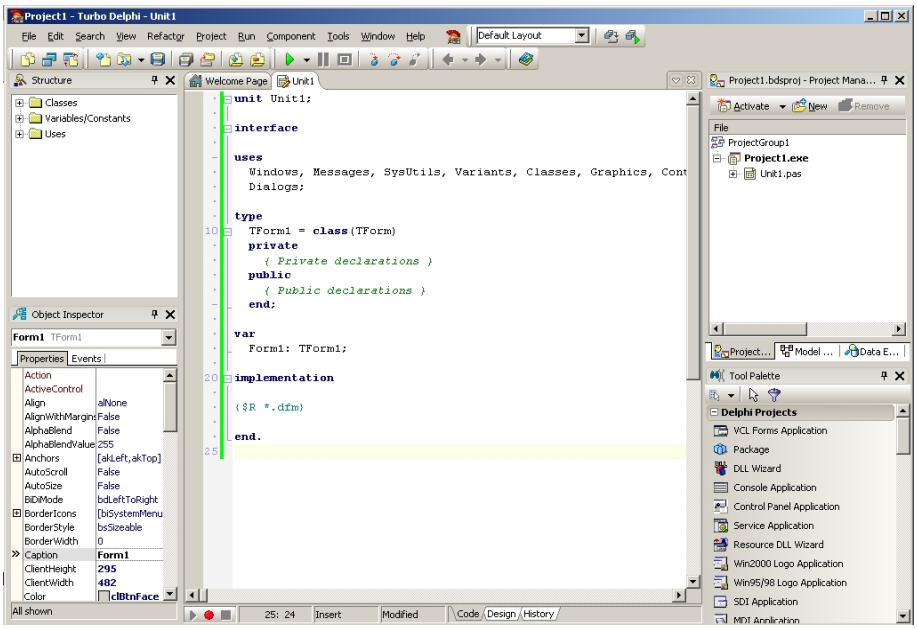
Egyszerű és könnyen kezelhető fejlesztői rendszer, mellyel gyorsan és hatékonyan lehet Windows alkalmazásokat fejleszteni. Lehetőséget biztosít a karakteres futtatási környezet használatára is, melyhez úgynevezett konzol alkalmazás is készíthető.

a) A **RAD** (Rapid Application Development) fejlesztői környezetbe integrált szolgáltatások együttese, melyek segítik a fejlesztő munkát. Fő jellemzői:
   - teljes integritás;
   - automatikus kódgenerálás és kódszinkronizáció;
   - integrált debugger;
   - gyors és intelligens fordító;
   - újrahasznosíthatóság;
   - intelligens help-rendszer.

b) A Delphi alkalmazások felépítése
   - project állomány (.dpr)
   - Form állományai (.dfm)
   - Unitok állományai (.pas)
   - Lefordított unitok és formok (.dcu)
   - Lefordított alkalmazás (.exe)
   - Egyéb állományok (.res , .cfg)

c) A fejlesztői környezet felépítése:





d) Állományok felépítése és tartalma

- Egy form állomány felépítése:

```
object Form1: TForm1
  Left = 200
  Top = 121
  Width = 488
  Height = 395
  Caption = 'Form1'
  Font.Charset=DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  PixelsPerInch = 120
  TextHeight = 16
end
```

- Egy unit állomány felépítése:

```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Classes,
     Graphics, Controls, Forms, Dialogs;
type
     TForm1 = class(TForm)
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var Form1: TForm1;
implementation
  {$R *.DFM}
end.
```

- Egy project állomány felépítése:

```
program Project1;
uses Forms,
  Unit1 in 'Unit1.pas' {Form1};
  {$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

e) Egy eseményvezérelt program futásának lépései:

a) A Delphi megvizsgálja az összes ablakot és a rajtuk lévő összes komponenst hogy történt-e valamilyen esemény, melyet az adott vezérlőelem felismer.

b) Ha észlel eseményt, melyhez nincs beépített válasz, akkor vizsgálja, hogy van-e az eseményhez rendelt eljárás.

c) Ha van, végrehajtja azt és az *a)* pont lép életbe.

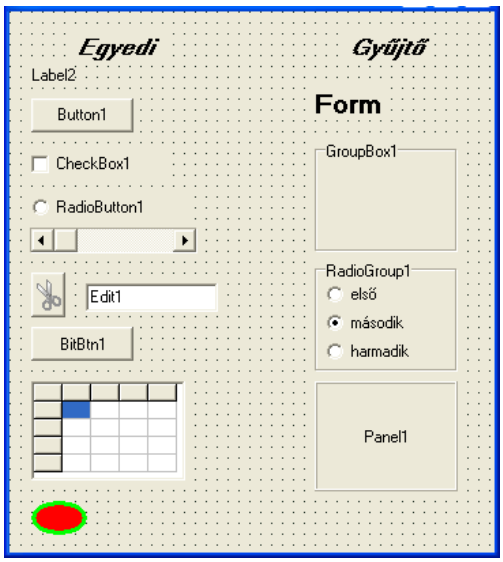
d) Ha nincs, akkor vár a következő eseményre az *a)* pont szerint.

Ezek a lépések ciklikusan követik egymást, amíg az alkalmazás fut.

Beépített választ generálnak például a címsor nyomógombjai: 

f) Komponensek

- a TComponent osztály leszármazottai
- Láthatóság szerint:
  - vizuális komponensek (VCL)
  - nem vizuális komponensek
- Jellegük szerint:
  - egyedi
  - gyűjtő
    - a gyűjtő komponens mozgatásával a ráhelyezett komponens is mozog.
    - a ráhelyezett komponens pozíciója a gyűjtő komponens nullpontjához viszonyított.



A fenti elméleti ismeretek után az egyes gyakran használt komponensek tulajdonságainak, fontosabb metódusainak és eseményeinek bemutatása következik. Ezeket az elméleti órákon megszerzett ismereteket a vele párhuzamosan haladó számítógép-programozás gyakorlat órákon használjuk fel, és itt készítünk olyan alkalmazásokat, melyekben az adott komponensek sajátosságait is alkalmazzuk.

5. Ismeretek felhasználása a gyakorlatban

A következő rész a gyakorlati órákon készülő alkalmazások közül néhánynak a rövid leírását és azokat az elkészítésükhöz szükséges új ismereteket tartalmazza, melyek az elméleti órákon hangoznak el. Az egyes feladatokhoz a könnyebb érthetőség kedvéért ablakképek is tartoznak. Minden feladatban szerepel a működés leírása, majd az, hogy az eddigi ismeretek mellett milyen új ismeretek szükségesek az adott feladat elkészítéséhez. Ezeket az ismeretek elméleti órán szerzik meg a tanulók, a gyakorlatban egy rövid ismétlés után már csak ennek alkalmazása a cél.

Az egyes alkalmazások létrehozáshoz szükség lehet még néhány gyakorlatias ismeretre, melyek lehet, hogy nem hangoznak el elméleti órákon. Ezeket minden feladathoz az egyéb ismeretek között részletezem.

A feladatok ismertetése úgy készült, hogy egy részük ez alapján önálló tanulói munka formájában is elkészíthető, de vannak olyan feladatok is, melyek elkészítéséhez több-kevesebb tanári irányítás, útmutatás szükséges.

A számítógép-programozás gyakorlatok II. tantárgy keretében ezeket az alkalmazásokat kisebb-nagyobb változtatásokkal, vagy esetleg pontosan ugyanígy elkészíttetem, illetve elkészítjük a tanulókkal. A feladatok sorrendje természetesen egyfajta nehézségi sorrendet is tükröz.

Az utolsó két feladatban a vizuális fejlesztői környezetbe integrált lehetőséget használom ki a konzol-alkalmazások készítéséhez. Erre a Turbo Delphi Explorer-be épített lehetőségre fontos figyelmet fordítani, mert az emelt szintű informatika érettségi vizsgán Pascal alapú programkészítést a vizuális fejlesztésen kívül csak ezzel, vagy a FreePascal segítségével végezhet a vizsgázó tanuló.

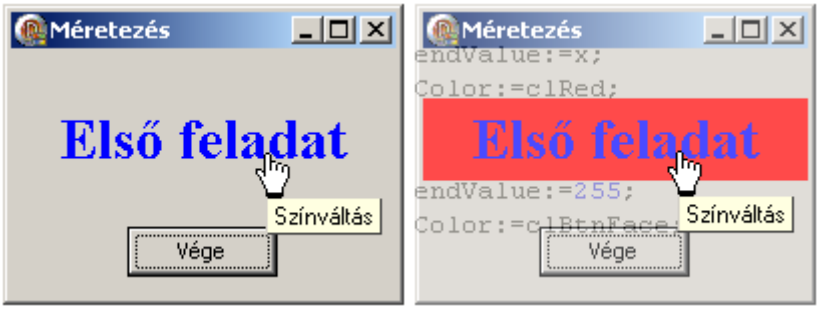
1. feladat: Ablak átméretezése

Készítsünk egy alkalmazást, mely egy ablak átméretezését csak megadott határok között engedi, a rajta lévő elemek helyzete pedig minden esetben az előírtak szerint követi az ablak méretváltozását.

Az alkalmazás ablakképe futás közben:

Működés:

- A „Méretezés” feliratú ablak szélessége 200 és 800 pixel között, magassága pedig 150 és 600 pixel között változhat. Az alkalmazás indulásakor az „Első feladat” szöveg is és a „Vége” feliratú nyomógomb is vízszintesen középen legyen, és átméretezés után is mindig vízszintesen középre kerüljenek. A felirat függőleges pozíciója ne változzon, és a gomb távolsága az ablak aljától mindig ugyanannyi legyen.
- Az elhelyezett feliraton kattintva az ablak legyen áttetsző és a felirat háttérszíne változzon pirosra, majd egy újabb kattintás eredményeként az eredeti állapot álljon vissza.
- A felirat fölött a kurzor alakja változzon meg, és jelenjen meg egy „Színváltás” tartalmú címke.



Az ablak felépítése:

Szükséges ismeretek:

TForm	Tulajdonságai:	`Name, Caption, Constraints, WindowState, Width, Height, ClientWidth, ClientHeight, AlphaBlend, AlphaBlendValue, Position`
	Eseményei:	`onCreate, onResize`
	Metódusai:	`Close`
TLabel	Tulajdonságai:	`Name, Caption, Left, Top, AutoSize, Width, Height, Alignment, Layout, Color, Transparent, Cursor, Hint, ShowHint`
	Eseményei:	`onClick`
TButton	Tulajdonságai:	`Name, Caption, Anchors`
	Eseményei:	`onClick`
TFont	Tulajdonságai:	`Color, Name, Size, (Style)`

Egyéb ismeretek:

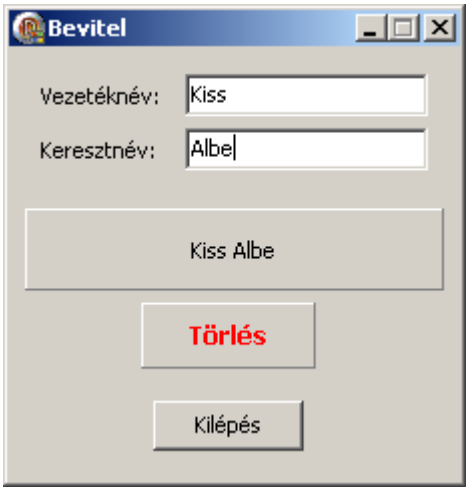
- `TFont`
- Nyomógomb (`Button1`) igazítása vízszintesen középre:

```
Button1.Left:=(Form1.ClientWidth-Button1.Width) div 2;
```

2. feladat: Név bevitele

Az alkalmazásban egy nevet tudunk megadni, külön a vezetéknevet és külön a keresztnevet. A névbevitel állapota folyamatosan követhető. A név részei együtt törölhetők.

Az alkalmazás ablakképe futás közben:



Működés:

- A „Bevitel” feliratú ablakon két darab egysoros beviteli mező egyikébe a vezetéknevet, a másikba a keresztnevet kell beírni. A beírással szinkronban egy panelen lévő felirat kövesse a gépelés változásait úgy, hogy mindig a gépelés alatt álló név aktuális állapota legyen látható. Mind a vezetéknév, mind a keresztnév legfeljebb 15 karakter hosszú lehet.
- A név törlése a „Törlés” feliratú panelre kattintva lehetséges. Ekkor a panel egy benyomódáshoz hasonló hatást mutasson.

- Az ablak ne legyen átméretezhető, és a maximalizáló gomb se legyen használható.
- Az alkalmazás indulásakor a szövegmezők üresek legyenek.

Szükséges új ismeretek:

TForm	Tulajdonságai:	`BorderStyle, BorderIcons`
TEdit	Tulajdonságai:	`CharCase, MaxLength, Text,(PasswordChar, ReadOnly)`
	Eseményei:	`onChange`
	Metódusai:	`SetFocus, Clear`
TPanel	Tulajdonságai:	`Caption, BevelInner, BevelOutet, BevelWidth, Alignment, VerticalAlignment`
	Eseményei:	`onMouseDown, onMouseUp, onMouseLeave`

Egyéb ismeretek:

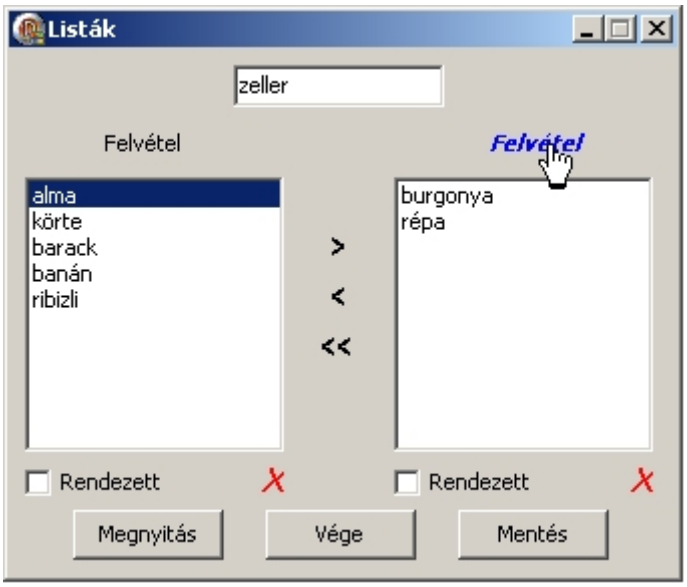
- `Application.Terminate;`
- Egy tevékenységsorozat több eseményhez rendelése a kód többszörözése nélkül: (az egyik beviteli mező `onChange` eseményét kell a másik `onChange` eseményére is beállítani).
- Ha a létrehozott alkalmazásunkat szeretnénk hasonló kinézetűre alakítani, mint a többi windows xp-n futó alkalmazás, akkor ezt alvégezhetjük egy `XPManifest` komponenssel. Ennek hatása nem minden windows stílusnál látható, de ha igen, akkor a változás egyes komponensek esetében nagyon szembetűnő.

- Ha ezt a hatást meg szeretnénk szüntetni, akkor az XPManifest komponens törlése mellett a unit `uses` sorából is törölni kell az `XPMan` unitra történő hivatkozást is.

3. feladat: Listák

A program futása során egy rövid begépelt szöveget felvehetjük két lista egyikébe, ha még nem szerepel benne az adott szöveg. A listák között elemeket mozgathatunk az egyik irányba csak egyesével a másik irányba akár csoportosan is. A listák tartalma sorba rendezhető, valamint az egyik listát feltölthetjük fájlból is, a másik tartalmát pedig fájlba is menthetjük.

Az alkalmazás ablakképe futás közben:



Működés:

- Az ablak nem átméretezhető, legfeljebb csak tálcára ikonizálható. A program futása természetesen a „Vége” gombra kattintva ér véget.
- Az egysoros beviteli mező tartalma a két „Felvétel felirat valamelyikére kattintva vehető fel az adott listába, ha még nem szerepel benne. A „Felvétel” szöveg stílusa és színe is megváltozik amikor az egér föléje ér, majd az egér lemozgatásakor a szöveg eredeti tulajdonságait kapja vissza. A „Felvétel” szövegek csak akkor aktivizálhatók, ha a beviteli mező nem üres.
- A bal oldali listában egyszerre csak egy elem, a jobb oldaliban pedig több elem is kijelölhető. A listák közötti három iránymutató segítségével az elemek áthelyezhetők az alábbi módokon:

  > : elem áthelyezése jobbra

  < : elem(ek) áthelyezése balra

  << : az összes elem áthelyezése balra

- Az adott iránymutató fölé mozgatva az egeret, az iránymutató színe változzon pirosra, és jelenjen meg az előbb megadott szöveg egy súgócímkében. Az egér lemozgatásakor az iránymutató eredeti állapota álljon vissza.
- A listák alatti jelölőnégyzetek segítségével ábécé sorrendbe rendezhetjük az adott lista elemeit.
- A listákból a kijelölt elem vagy elemek a  jelre kattintva törölhetők.
- A bal oldali listába egy párbeszédablakban kiválasztott állományból tölthetjük be az elemeket, míg a jobb oldali lista tartalmát egy mentéskor szokásos módon megadott állományba menthetjük.
- Mentéskor a szövegfájl (*.txt) és a listafájl (*.lst) fájltípusok közül választhatunk. Az alapértelmezett kiterjesztés a listafájl kiterjesztése legyen.
- Megnyitáskor e két típuson kívül legyen választható a támogatott típusok lehetőség is, amely a két típus bármelyikét jelenti.
- A program indulásakor a beviteli mező és a két lista is üres, és a leírtak szerint a „Felvétel” feliratok is inaktívak.

Szükséges új ismeretek:

TLabel	Tulajdonságai:	`Enabled, (Visible)`
	Eseményei:	`onMouseEnter, onMouseLeave`
TListBox	Tulajdonságai:	`MultiSelect, ExtendedSelect, Items, ItemIndex, Sorted`
TStrings	Tulajdonságai:	`Count`
	Metódusai:	`Add, Delete, IndexOf, Selected, Clear, SaveToFile, LoadFromFile`
TCheckBox	Tulajdonságai:	`Alignment, Checked, (AllowGrayed), State, (Wordwrap)`
TFont	Tulajdonságai:	`Style`
TOpenDialog, TSaveDialog	Tulajdonságai:	`DefaultExt, FileName, Filter, FilterIndex, Title, (Options)`
	Metódusai:	`Execute`

Egyéb ismeret:

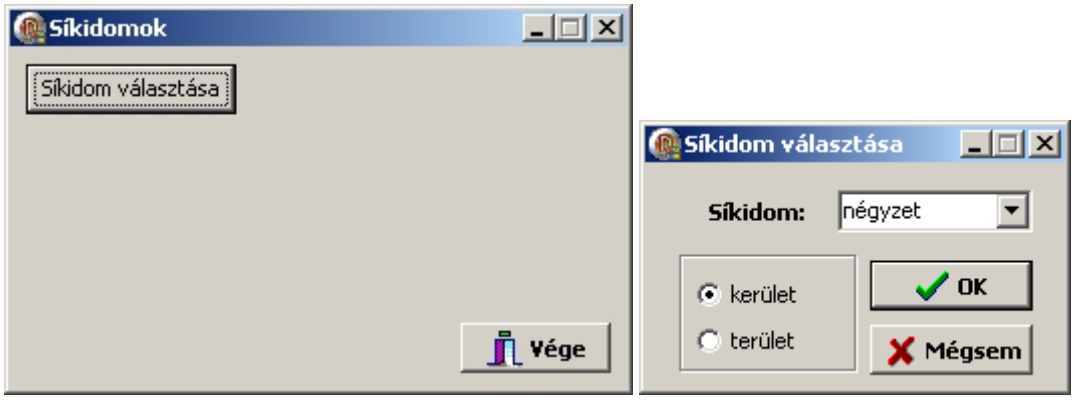
- Egy többszörös kijelölést engedélyező lista elemeinek törlésekor célszerű a törlést végző ciklust a lista végétől az eleje felé futtatni, mert törléskor változik az elemszám.

4. feladat: Síkidomok

A két ablakos alkalmazásban a második ablakon választhatunk egy síkidomfajtát, melynek az első ablakon általunk megadott megfelelő adatiból kiszámolja és egy üzenetablakban megjeleníti az adott síkidom kerületét vagy területét, attól függően, hogy melyiket szeretnénk meghatározni.

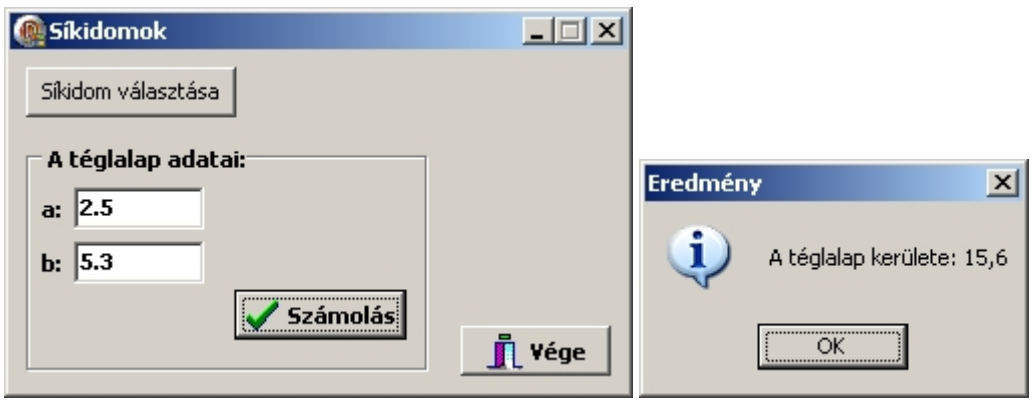
Működés:

- Az első, „Síkidomok” feliratú ablakon a „Síkidom választása” nyomógombra kattintva megjelenik a második, „Síkidom választása” feliratú ablak, melyen egy előre kitöltött legördülő listából választhatjuk ki a síkidom fajtáját. A lista alapértelmezése a lista első eleme. A lista futás közben nem szerkeszthető.

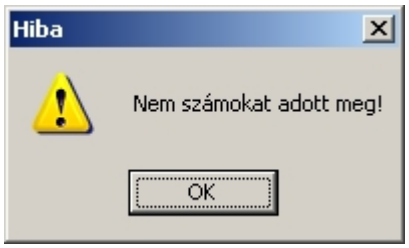


- Az „OK” vagy a „Mégsem” feliratú grafikus gombra kattintás után a második ablak bezáródik és az „OK” választása esetén az első ablakon a megfelelő feliratú csoportosított részen megfelelő számú beviteli mező jelenik meg. A mező(k) száma és az előtte lévő feliratok tartalma a kiválasztott síkidomtól függ:

síkidom	négyzet	téglalap	háromszög	kör
szükséges adat(ok):	oldal(ak) hossza			sugár
jelölés:	a	a, b	a, b, c	r



- A beviteli mezők kitöltését követően a „Számolás” gombra kattintva a program kiszámolja a második ablakon a felirat nélküli rádiógomb-csoportban jelölt síkidomjellemzőt. Alapértelmezés a kerület választása. A kiszámolt értéket egy üzenetablakban jeleníti meg a program.



- Ha a beviteli mezők üresek, vagy hibás adatot adtunk meg (például szöveget vagy nem megfelelő decimális elválasztójelet), akkor a program ezt egy üzenetablakban közölje.
- Figyeljünk arra is, hogy a jellemző méretek csak pozitív számok lehetnek, és háromszög nem készíthető bármilyen számhármasból. Ha futás közben ezek a hibák fordulnak elő, jelezze azt a felhasználó felé.

Szükséges új ismeretek:

Típuskonverziók: `Val, FormatFloat, (IntToStr)`

Új ablakok megnyitása, modális ablakok: `Show, ShowModal`

Üzenetablakok: `Application.MessageBox`
`(ShowMessage, MessageDlg)`

TGroupBox	Tulajdonságai: `Caption, Visible`
TBitBtn	Tulajdonságai: `Kind, Glyph, Layout, ModalResult, Cancel`
TRadioGroup	Tulajdonságai: `Items, Columns`
TRadioButton	Tulajdonságai: `Checked`
TComboBox	Tulajdonságai: `Items, ItemIndex, Text, DropDownCount, Style`
	Eseményei: `onKeyPress`

Egyéb ismeretek:

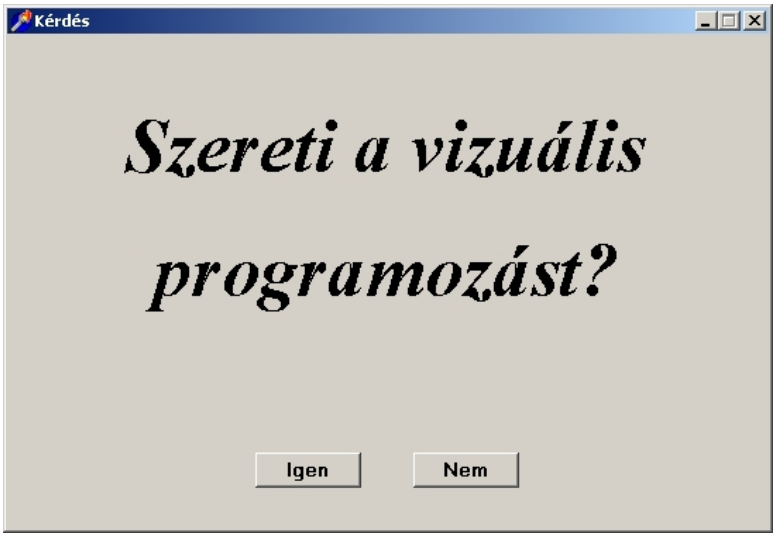
- Háromszög területének számítása Héron képletével:

$$T = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}, \text{ ahol } s = \frac{a+b+c}{2}$$

5. feladat: Kérdés

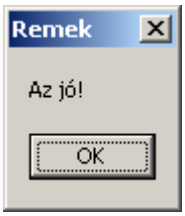
A programban feltett kérdésre az ablakon lévő megfelelő gomb kiválasztásával lehet válaszolni.

Az alkalmazás ablakképe futás közben:



Működés:

- Az ablak nem átméretezhető. A „Nem” gombra nem lehet rákattintani, mert az egérrel történő elérés hatására véletlenszerűen más helyre ugrik. Az új pozíció csak olyan lehet, hogy a gomb teljes mérete az ablakon maradjon.
- Az „Igen” gombra kattintva egy üzenetablak jelenjen meg.



- A program bezárására csak a rendszermenü és a címsor bezáró gombja szolgál.
- Meg kell oldani, hogy a „Nem” gomb más módon se legyen kiválasztható.

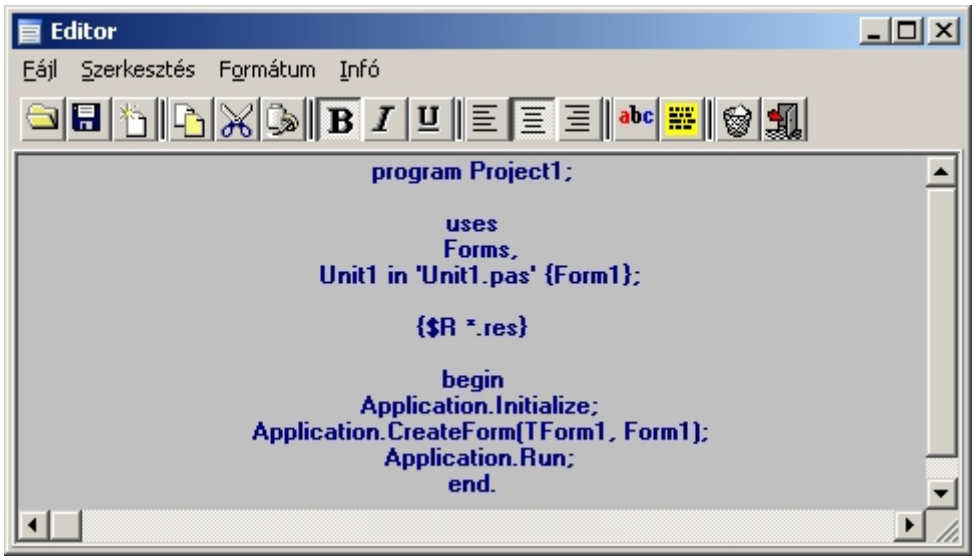
Szükséges új ismeretek:

TButton	Tulajdonságai: `TabStop, TabOrder`

6. feladat: Editor

A program egy egyszerű szövegfájlok kezelésére alkalmas editor főmenüvel, eszköztárral és helyi menüvel ellátva. Az editorban a formázási beállítások a teljes szövegre egységesen érvényesek.

Az alkalmazás ablakképe futás közben:



Működés:

- A program menüszerkezete:

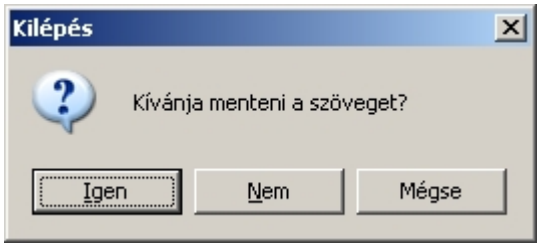
Fájl	Szerkesztés	Formátum
Új Ctrl+N Megnyitás … Ctrl+O Mentés … Ctrl+S Kilépés	Másolás Kivágás Beillesztés Kijelölt rész törlése	Karakter … Igazítás Stílus Háttér

Igazítás	Stílus	helyi menü
balra középre jobbra	félkövér dőlt aláhúzott	Másolás Kivágás Beillesztés Kijelölt rész törlése

- A program indulásakor alapértelmezésként a balra igazított normál stílus kerül beállításra.
- A megnyitáskor választható állománytípusok a Szövegfájl (*.txt), a Unit állomány (*.pas), a Project állomány (*.dpr), a Form állomány (*.dfm), valamint jelenjen meg az összes támogatott típus választási lehetősége. Mentéskor

értelemszerűen az utolsó kivételével tartalmazza a menthető fájltípusokat. Az alapértelmezett fájlnév a „Szöveg1”, az alapértelmezett kiterjesztés pedig „txt” legyen.

- A „Karakter …” menüparancs választásakor a Windows környezetben megszokott karakterbeállító párbeszédablakban állíthatjuk be a teljes szövegre alkalmazandó beállításokat.
- Az „Igazítás” almenüjének parancsai rádiógombszerűen, kölcsönös kizárással választhatók, és az itt elvégzett beállításnak az eszköztár igazítást megvalósító gombjainak benyomott állapotában is meg kell jelenni és viszont. Hasonló kapcsolatnak kell lenni a „Stílus” almenüpontjai és az írásstílust jelző és vezérlő gombok között azzal az eltéréssel, hogy itt természetesen nincs kölcsönös kizárás. Az írásstílus jelzésének és vezérlésének a gombokon, a menüben és a karakterbeállítás párbeszédablakában kell azonosnak lenni.
- A „Háttér” parancs választásakor egy színbeállító párbeszédablakban választhatjuk ki a szöveget tartalmazó terület színét.
- A menük parancsaihoz és az eszköztár gombjaihoz azonos bitképeket használjunk. A menüpontokhoz rendeljünk kiemelt karaktereket, és néhányhoz gyorsgomb funkciót is.
- Az eszköztáron elhelyezett parancsgombok mindegyikéhez rendeljünk súgócímkét, mely az adott gomb feladatát jeleníti meg.
- A program kilépéskor ajánlja fel a szöveg mentésének lehetőségét, vagy a kilépés megszakítását.



- Az „Infó” menüpontot választva egy, a fejlesztő környezetben megtalálható AboutBox típusú ablakot nyisson meg modálisan. Ezen az ablakon található elemeket módosíthatjuk.

Szükséges új ismeretek:

TForm	Tulajdonságai:	`KeyPreview, Menu`
	Eseményei:	`onCloseQuery`
TMemo	Tulajdonságai:	`Align, Lines, ScrollBars, WantTabs, WantReturns, PopupMenu`
	Metódusai:	`Clear, CopyToClipboard, CutToClipboard, PasteFromClipboard, ClearSelection`
TToolBar	Tulajdonságai:	`ButtonHeight, ButtonWidth`
TSpeedButton	Tulajdonságai:	`AllowAllUp, GroupIndex, Down, Flat, Glyph, Layout, Margins, NumGlyphs, Spacing, Transparent`
TMainMenu	Tulajdonságai:	`AutoHotKeys`
TPopupMenu	Tulajdonságai:	`Alignment, AutoPopup`
	Eseményei:	`onPopup`
TMenuItems	Tulajdonságai:	`AutoCheck, Bitmap, Checked, GroupIndex, RadioItem, ShortCut`
TFontDialog	Tulajdonságai:	`Font, Options`
	Metódusai:	`Execute`
TColorDialog	Tulajdonságai:	`Color, Options`
	Metódusai:	`Execute`
TClipBoard	Tulajdonságai:	`AsText`
	Metódusai:	`(Assign, Clear, Free, AssignTo)`
TActionList	-	
TAction	Metódusai:	`Execute`

Egyéb ismeretek:

- A beépített elrendezések a `File/New/Other…` menühívás sorozattal a `New Items` ablakban érhetők el.

7. feladat: Színkeverés

A program három gördítősáv segítségével az RGB rendszer színkomponenseinek intenzitása alapján keveri ki a színt és annak komplementer színét.

Az alkalmazás ablakképe futás közben:

Működés:

- A nem átméretezhető ablakon lévő gördítősávok segítségével állíthatjuk be a kevert szín piros, zöld és kék összetevőjének intenzitását. Az intenzitás 0-tól 255-ig terjedő érték lehet. A gördítősávok előtt egy-egy grafikus elemmel jelöljük, hogy melyik szín intenzitását állíthatjuk a segítségével.
- A kevert szín határozza meg a kijelző panel színét, a panel felirata pedig ennek kiegészítő színe lesz. A kiegészítő szín színkomponensenkénti 255-re történő kiegészítéssel állítható elő.
- Bármely gördítősáv csúszkáját is változtatjuk, az aktuális kevert színnek és kiegészítő színének követnie kell a változást. Az egyes komponensek intenzitását az adott gördítősáv mögötti szám is jelzi.
- A program indulásakor a kijelző panel színe fekete, a szövegének színe ennek kiegészítője, a fehér legyen.

Szükséges új ismeretek:

Konverzió : `IntToStr` függvény

TShape	Tulajdonságai: `Shape, Brush, Pen`
TPen	Tulajdonságai: `Color, Style, Width`
TBrush	Tulajdonságai: `Color, Style`
TScrollBar	Tulajdonságai: `Min, Max, Position, Kind, Large Change, Small Change`
	Eseményei: `onChange, onScroll`

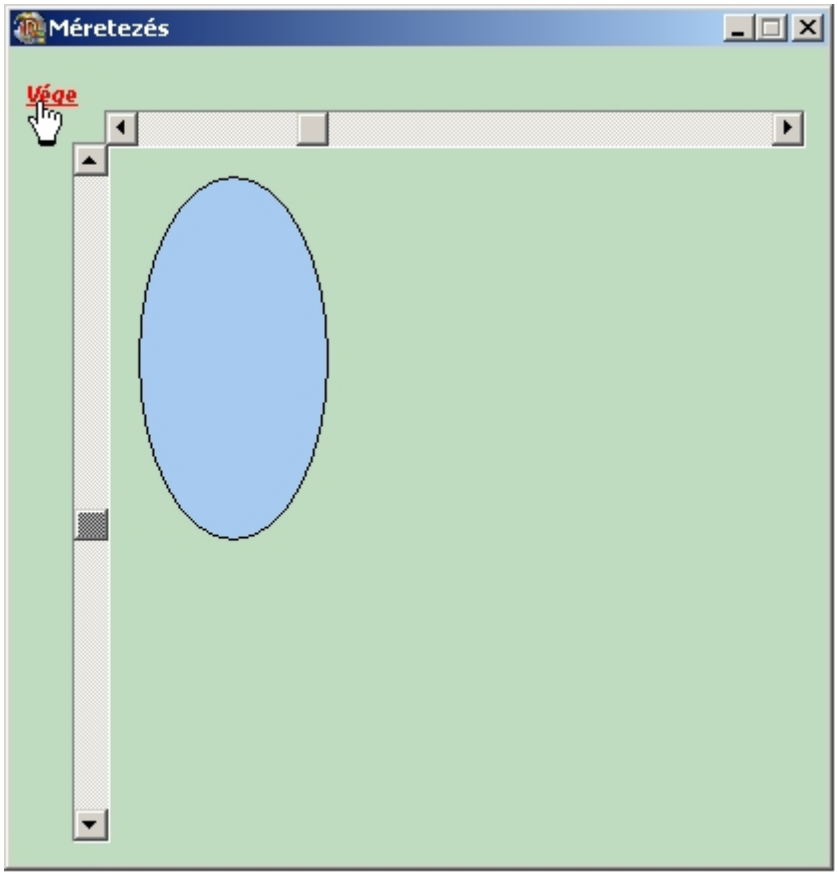
Egyéb ismeretek:

- `Color:=RGB(red, green, blue:byte);`
- Fekete szín esetén mindhárom színkomponens intenzitása 0

8. feladat: Méretezés

Az alkalmazás ablakában egy grafikus elemet méretezünk át gördítősávok segítségével.

Az alkalmazás ablakképe futás közben:



Működés:

- A gördítősávok segítségével átméretezhetjük az eredetileg kör alakú grafikus alakzatot.
- Figyeljünk rá, hogy az alakzat méretét a csúszka egyik széle határozza meg.
- A programból történő kilépés a „Vége” feliratra kattintva történhet. A felirat írásstílusa az egér hatására megváltozik. A példában a piros félkövér karakterek aláhúzottá is válnak.

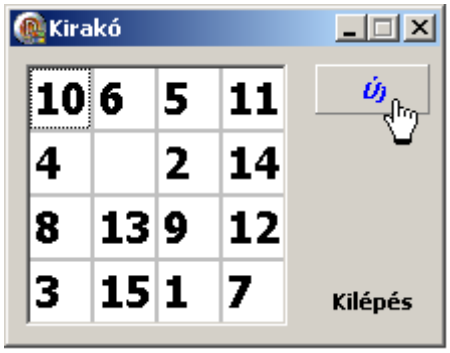
Szükséges új ismeretek:

A feladat megoldásához ugyanazok az ismeretek szükségesek, mint az előző, Színkeverés nevű feladathoz.

9. feladat: Kirakó

A program a jól ismert kirakós játék számítógépes változata.

Az alkalmazás ablakképe futás közben:



Működés:

- Az alkalmazás indításakor, majd később az „Új” gombra kattintva is a rácsot véletlenszerűen tölti fel a program számokkal 1-től 15-ig. és egy üres hellyel úgy, hogy minden érték csak egyszer fordulhat elő.
- Feltöltés után valamelyik, üres mező melletti elemre kattintva a kiválasztott elem kerül az üres helyére, és a helye lesz üres. Figyeljünk, hogy csak vízszintes vagy függőleges mozgatás végezhető, átlós irányú nem.
- Az „Új” és a „Kilépés” feliratú gombok látványterve és vizuális hatásai azonosak.

Szükséges új ismeretek:

TSringGrid	Tulajdonságai:	`ColCount, RowCount, DefaultColWidth, DefaultRowHeight, FixedRows, FixedCols, FixedColor, GridLineWidth, VisibleColCount, VisibleRowCount, Options`
	Eseményei:	`onSelectCell`

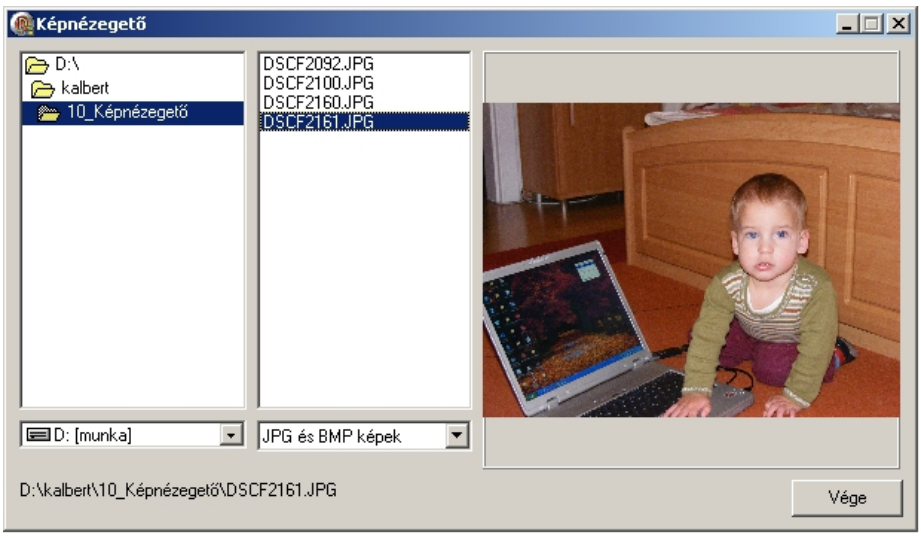
Egyéb ismeretek:

- Egy cella tartalmának elérése: `StringGrid1.Cells[i,j]`
- A már elhelyezett számok vizsgálatához célszerű halmazba gyűjteni a már kisorsolt értékeket.
- A csak vízszintes vagy csak függőleges szomszédság megállapítására az üres és a kiválasztott cellák indexe csak eggyel térhet el és csak az egyik, ezért célszerű lehet az `xor` logikai művelet alkalmazása.

10. feladat: Képnézegető

Az alkalmazás egy fájlböngésző részben kiválasztott képet jeleníti meg az ablak egy kijelölt részén torzulásmentesen középen.

Az alkalmazás ablakképe futás közben:



Működés:

- A fájlok listájára beállított szűrési lehetőségek csak bmp és jpg kiterjesztésű állományok megjelenítését tegyék lehetővé külön-külön is és együtt is.
- Ha a kép kisebb, mint a rendelkezésre álló terület, akkor átméretezés nélkül, egyébkent pedig arányosan kicsinyítve mindig a képterület közepére igazítottan jelenjen meg az eredeti oldalarányok megtartásával.
- A képek a fájlnév kiválasztásakor automatikusan töltődjön be az adott helyre.
- Ha olyan meghajtót választunk, amelyben nincs elérhető adathordozó, akkor a program figyelmeztessen erre.
- A kiválasztott kép teljes elérési útvonala jelenjen meg az ablak alján.

Szükséges új ismeretek:

TDriveComboBox	Tulajdonságai:`Drive, TextCase, DirList`
TDirectoryListBox	Tulajdonságai:`Drive, FileList`
TFileListBox	Tulajdonságai:`Directory, Mask`
TFilterComboBox	Tulajdonságai:`FileList`
TImage	Tulajdonságai:`Autosize, Stretch, Center, Picture`
TPicture	Metódusai: `LoadFromFile, (SaveToFile)`

11. feladat: Csigafutam

A program csigák versenyét szimulálja, feltéve, hogy azok egymással párhuzamosa, egyenesen haladnak.

Az alkalmazás ablakképe futás közben:

Működés:

- A program indulásakor a csigák a sávjuk legelején helyezkednek el. Az „Új futam” feliratra kattintva is ezt a helyzetet kell előidézni az adott futam megszakítása után. A futam a „Start” feliratra kattintva indul és addig tart, amíg az első csiga eléri a célvonalat, vagy amíg meg nem szakítjuk a futamot az „Új futam” feliratra kattintva.
- A csigák mozgatása egy időzítő és egy véletlen szám generálásával történik úgy, hogy adott időtartam elteltével a véletlenszerűen kiválasztott csiga halad előre valamekkora beállított értékkel.
- A példaprogramban a pálya hasznos hossza 420 pixel, a beállított léptetés értéke 5 pixel. A léptetési időköz beállítására a „Tempó:” szöveg mögötti mezőben végezhető el a léptető gombok segítségével 1-től 10-ig megadott egész értékkel úgy, hogy a nagyobb érték nagyobb sebességet, azaz gyakoribb léptetést jelentsen.

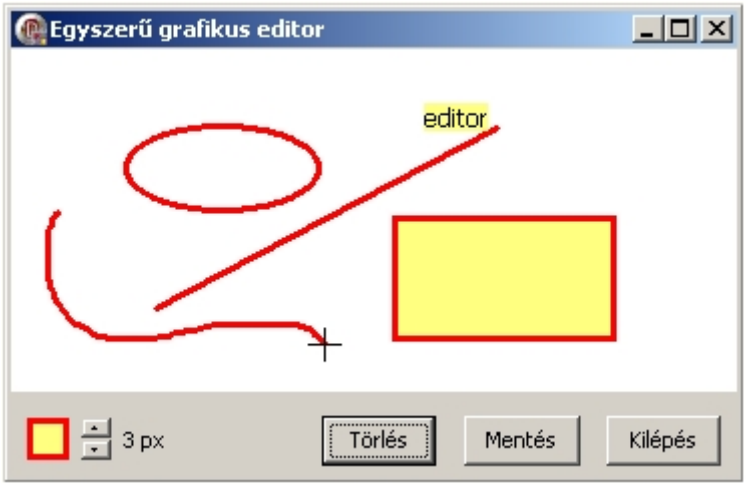
Szükséges új ismeretek:

TTimer	Tulajdonságai: Enabled, Interval
	Eseményei: onTimer
TUpDown	Tulajdonságai: Associate, Increment, Orientation, Position,

12. feladat: Grafikus szerkesztő

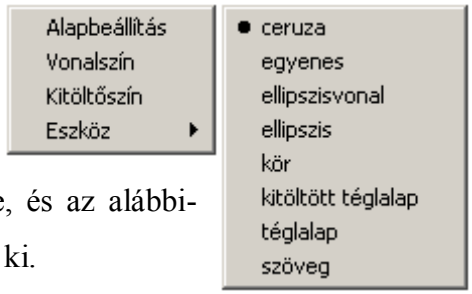
Egyszerű alakzatok rajzolására alkalmas grafikus alkalmazás.

Az alkalmazás ablakképe futás közben:



Működés:

- Az átméretezhető ablak nagyobb részét egy fehér színű rajzvászon borítja, az alsó rész bal oldalán pedig tájékoztató információk láthatók az aktuális beállításokról. Itt láthatjuk a beállított vonalszínt, ami a keret színe, a kitöltőszínt és a vonal vastagságát. A vonal vastagsága 1 pixeltől 6 pixelig léptetéssel állítható. A vonalvastagság állításával a minta szegélyvastagsága ne változzon meg.
- Az ablak átméretezésekor az alsó szürke sáv magassági mérete nem változik, és a bal oldali információjelzők pozíciója a bal alsó sarokhoz viszonyítva, a gombok pozíciója pedig a jobb alsó sarokhoz képest ne változzon meg.
- A „Törlés” gombra kattintva a teljes vászon tartalma törlődik.
- A mentés során csak bitkép (bmp) formátumban menthetünk egy szokásos mentés párbeszédablak segítségével.
- A rajzolás egyéb beállításait a vászonhoz kötött helyi menüben végezhetjük el.



- Az „Alapbeállítás” fehér kitöltőszínt, egy pixel vastag fekete vonaszínt állít be, és az alábbiakban részletezett ceruza eszközt választja ki.
- Az „Eszköz” menüpont almenüjében választhatjuk ki a rajzolni kívánt alakzatot. Az almenü rádiógombszerűen működik, és az aktuális rajzelemet mindig jelöljük.
- A program indulásakor az alapbeállítás érvényes a ceruza eszközzel együtt.

Szükséges új ismeretek:

TForm	Tulajdonságai: `Tag, DoubleBuffered`
TSavePictureDialog	Tulajdonságai: `FileName`
	Metódusai: `Execute`
TImage	Tulajdonságai: `Align`
	Eseményei: `onMouseDown, onMouseUp, onMouseMove`
TCanvas	Metódusai: `MoveTo, LineTo, Ellipse, Rectangle, TextOut, FillRect, (Arc, Pie, FloodFill)`
TBitmap	Tulajdonságai: `Width, Height`
TPicture	Metódusai: `SaveToFile`

Egyéb ismeretek:

- A vászon tartalmának törlése legegyszerűbben egy teljes vászon méretű kitöltött téglalap rajzolásával oldható meg, ha a kitöltőszín fehér:

```
Image1.Canvas.Brush.Color := clWhite;
Image1.Canvas.FillRect(Rect(0,0,Image1.Width,Image1.Height));
```

- A `FillRect` metódus paramétere egy `TRect` típusú is lehet. Ekkor:

```
Teglalap:Trect;
...
With Teglalap do
  Begin
    Left:=0; Top:=0;
    Right:= Image1.Width;
    Bottom:= Image1.Height;
  End;
Image1.Canvas.FillRect(Teglalap);
```

13. feladat: Külső programok indítása

Az alkalmazás megmutatja hogyan lehet elindítani egy másik programot az alkalmazásunkból.

Az alkalmazás ablakképe futás közben:

Működés:

- A listából kiválasztott alkalmazás a nevének sorosa kattintva indítható.
- Az egymás alatt lévő feliratok mindegyikének stílusa megváltozik, ha az egérrel fölé állunk, majd ez egeret elmozgatva róla a stílus ismét az eredeti lesz. Ezt a megfelelő kód többszörözése nélkül kell megoldani.
- A „Kép?” feliratú jelölőnégyzet kiválasztásával a listában éppen jelölt program ikonját is megjeleníthetjük. Ehhez az ikonokat külön fájlokba el kell készíteni.

Szükséges új ismeretek:

TImageList	Tulajdonságai: `AllocBy, BlendColor, Height, Width`
	Metódusai: `GetBitmap, GetImages`

Egyéb ismeretek:

- A `sender` paraméter, az `is` és az `as` operátorok megismerése után a feliratok mindegyikére alkalmazható azonos kód készítéséhez szükséges szerkezet készíthető

```
With sender as TLabel do
Begin
...
End;
```

- A program indításához a `Winexec` eljárás hívása szükséges, melyben ismerni kell az indítandó program pontos elérési útvonalát. Ennek általánosításához az Interneten találhatók programkódok, melyek segítségével meghatározhatjuk a fontosabb könyvtárak elérési útvonalát *(például: Windows, Program Files, System)*. Az így megtalált útvonal adatának konverziójához szükséges lehet a `PChar` függvény használata is.

14. feladat: Puma

Az alkalmazás bemutatja, hogy készíthetünk a megszokottól eltérő alakú ablakkal rendelkező programot is.

Az alkalmazás ablakképe futás közben:

Működés:

- Az ablak a képen kívül csak egy csúszkát és egy kilépés feliratot tartalmaz
- A csúszka segítségével az ablak átlátszóságát állíthatjuk
- Az ablak nem rendelkezik címsorral, ezért a mozgatását az ablak belsejénél fogva kell megoldani.

Szükséges új ismeretek:

TForm	Tulajdonságai: `TransparentColor, TransparentColorValue`
TTrackBar	Tulajdonságai: `Frequency, LineSize, Max, Min, Position, Orientation, TickMarks`
	Eseményei: `onChange`

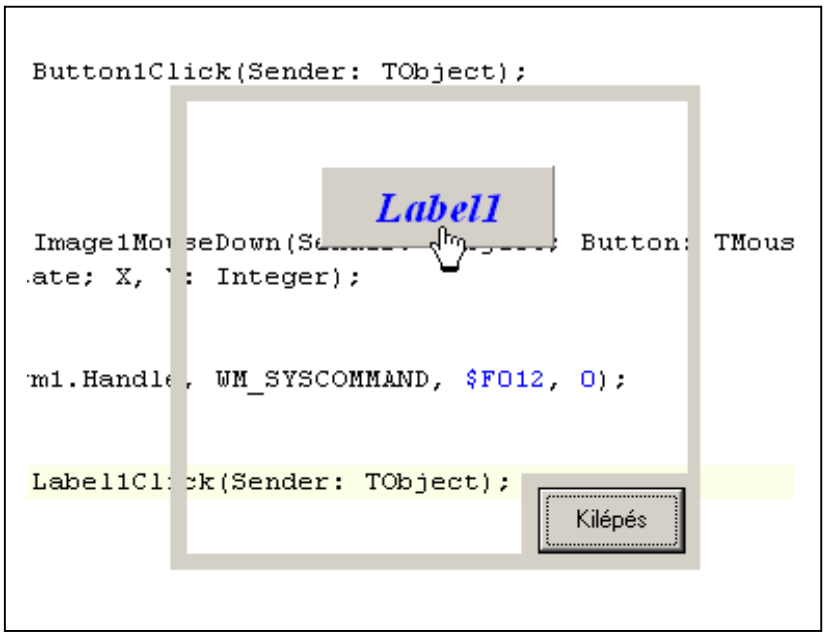
Egyéb ismeretek:

- Az átlátszóság (`AlphaBlendValue`) értékét nem célszerű túl kis értékre állítani, mert akkor a „Kilépés” feliratot sem tudjuk elérni a kilépéshez.

- Az ablak kialakításához az ablak teljes kliensterületét kitöltő kép szükséges, melyen a megjeleníteni kívánt részen kívül minden egyéb területet egyszínűre kell színezni, majd ezt a színt kell beállítani az ablak átlátszó színének.
- Az ablakot a kép segítségével tudjuk mozgatni. Az ehhez szükséges kódrészlet a kép `onMouseDown` eseményében:

```
ReleaseCapture;
SendMessage(Form1.Handle, WM_SYSCOMMAND, $F012, 0);
```

- Készíthetünk képek nélkül is tetszőleges alakú ablakokat, ehhez egyszerűen csak a rá helyezett, színezhető komponensek színét kell helyesen megválasztani. Például:

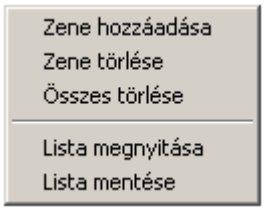


15. feladat: Audiólejátszó

Egy testreszabott alakú audiólejátszó program, melyben a lejátszás vezérlése a kép megfelelő részeivel lehetséges.

Az alkalmazás ablakképe futás közben:

Működés:



- A zeneszámok listájának kezelésére a listához rendelt helyi menü alkalmazható.
- A zene hozzáadása egy, a windows-ban megszokott megnyitás párbeszédablak segítségével történik. A programmal `mp3`, `mid`, `wav` kiterjesztésű állományokat lehessen megnyitni.
- A „Zene törlése” a kijelölt zeneszámok törlését végezze el a listáról, míg az „Összes törlése” paranccsal a lista minden bejegyzése törlődjön.
- A lejátszási lista legyen elmenthető `lst` kiterjesztéssel, és természetesen az ilyen kiterjesztéssel rendelkező listákat tudja megnyitni is a program. Figyeljünk, hogy a fájlok csak akkor lesznek lejátszhatók, ha a teljes elérési útjuk ismert, és azt mentjük el, majd töltjük be.
- A lejátszás vezérlését úgy oldjuk meg, hogy a kép részét képező megfelelő feliratok változzanak meg az egér rámozgatásának hatására, majd lemozgáskor álljon vissza az azt megelőző állapot. Például:

  PREV PREV , NEXT NEXT , STOP STOP , PLAY PLAY .

- Hasonló módon használjuk a kikapcsolás gombot ( ) is a kilépéshez.
- A kiválasztott zeneszám legyen azonnal elindítható úgy is, ha a kijelzőn duplán kattintunk a hozzá tartozó bejegyzésen.

- Ha szükséges használjuk a kivételkezelés lehetőségeit is.
- A listában a körbejárás legyen megoldott, azaz az utolsó zeneszám után a következő zene ez első, illetve az első előtti az utolsó.

Szükséges új ismeretek:

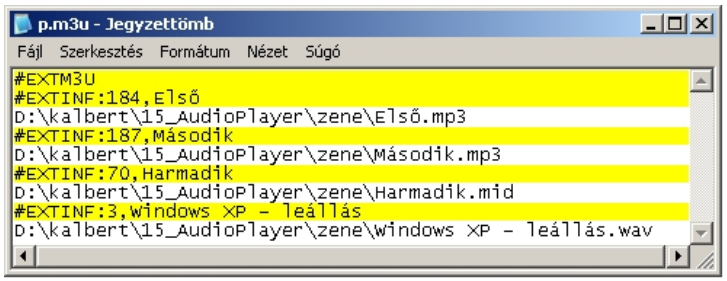
TMediaPlayer	Tulajdonságai:	`AutoEnable, AutoOpen, DeviceType, FileName, ColoredButtons, EnabledButtons, VisibleButtons`
	Metódusai:	`Stop, Open, Play, (Close, Pause, Back, Next)`

Egyéb ismeretek:

- A zeneszámok hozzáadása során figyeljünk arra, hogy a listában csak a fájl neve látható, viszont azt kiválasztva a fájl lejátszásának elindításához szükséges a teljes elérési útvonala, ezért azt is el kell tárolni.
- Hibakezelés: Feltételekkel nehezen kezelhető hibalehetőség, hogy bár a médialejátszó komponenst még nem nyitottuk meg, de mégis a stop gombra kattinthatunk. Ez futási hibát eredményez. Ennek egy lehetséges kezelését mutatja be az alábbi kivételkezelő kódrészlet, melyben a hiba bekövetkezésekor semmit sem hajtatunk végre a programmal:

```
try
  MediaPlayer1.Stop;
except on EMCIDeviceError do ;
end;
```

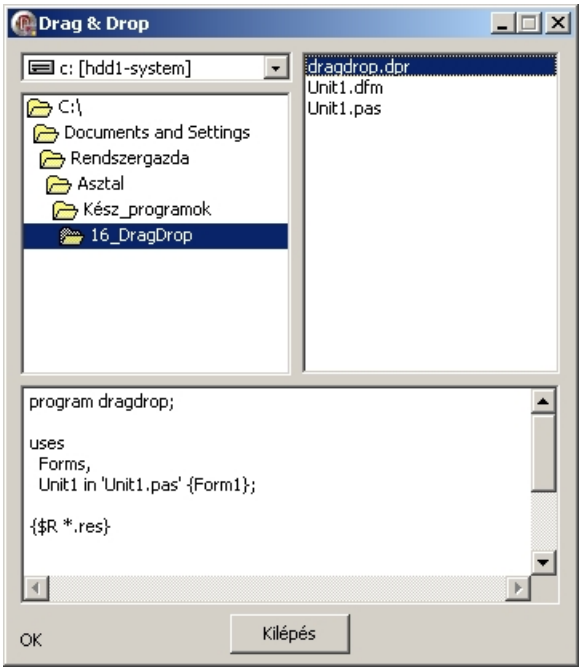
- A listák kezelésekor megengedhetjük az elterjedt `m3u` kiterjesztéssel történő mentést is, de ekkor célszerű megengedni az `m3u` fájlok megnyitását is a programban. Egy másik programmal (például a WinAmp-pal) létrehozott m3u kiterjesztésű fájlok az állományok elérési útvonalán kívül egyéb információkat is tartalmaznak az adott állományról. Ezért a listába történő betöltés során a felesleges sorokat el kell távolítani. A képen egy m3u fájl tartalmát látjuk, melyben az eltávolítandó sorokat jelöltem.



16. feladat: Drag&Drop

A program szöveges állományok tartalmának megjelenítését teszi lehetővé úgy, hogy a megjelenítendő állományt a megjelenítést végző ablakrészbe vonszoljuk.

Az alkalmazás ablakképe futás közben:



Működés:

- A fájllista ablakrészében megjelenő típusok a `txt`, a `dpr`, a `dfm` és a `pas` kiterjesztésű állományok lehetnek. A kiválasztott állomány tartalma az ablak alsó, megjelenítő ablakrészébe vonszolva jelenik meg. Az ablak bármely más részébe vonszolva az állományt, a tartalom természetesen nem jelenik meg.
- A vonszolás kezdetét, annak befejezésekor a sikerességét vagy sikertelenségét az ablak bal alsó sarkában lévő szöveg jelezze. A program indulásakor nem látszik szöveg, egyébként pedig rendre például a „Start”, az „OK” vagy a „--” szöveg jelenhet meg.

Szükséges új ismeretek:

TFileListBox	Tulajdonságai:	`DragCursor, DragMode, Mask`
	Eseményei:	`onStartDrag, onEndDrag`
TMemo	Tulajdonságai:	`DragCursor, DragKind, DragMode`
	Eseményei:	`onDragDrop, onDragOver`
TMouse	Tulajdonságai:	`DragImmediat, DragTreshold, IsDragging`

Egyéb ismeretek:

- Az `onStartDrag` esemény automatikusan bekövetkezik a bal egérgomb megnyomásakor. Azt, hogy ténylegesen hány pixel kurzorelmozdulás után kezdődjön a vonszolás a `DragTreshold` tulajdonsággal adhatjuk meg.
- Azt, hogy a szövegmegjelenítő `Memo1` nevű komponens képes-e fogadni a fölé vonszolt elemet, az `onDragOver` eseményében dönthetjük el. Ha a fájllistáról húzott elem fogadását engedélyezni szeretnénk, akkor ezt az `Accept` paraméter alábbi módon történő igazra állításával tehetjük meg:

```
Accept := Source is TFileListBox;
```

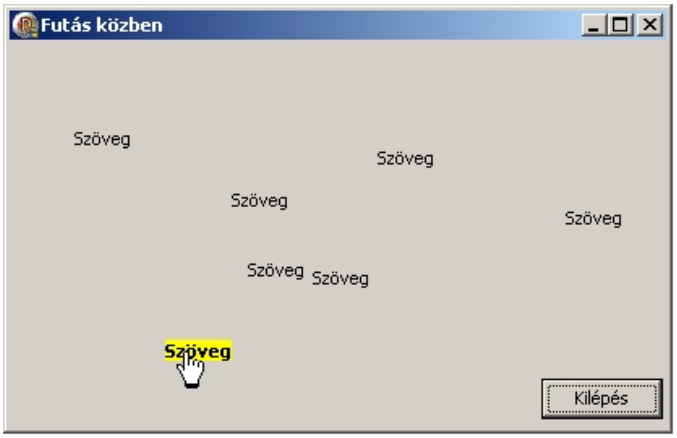
- A `Memo1` komponens `onDragDrop` eseményében adhatjuk meg, hogy a vonszolás végeztével, mi történjen a vonszolt elemmel:

```
if Source is TFileListBox then ...
```

17. feladat: Létrehozás futás közben

Ez az egyszerű alkalmazás azt mutatja be, hogyan kell eseményekkel vezérelhető ablakelemeket létrehozni a program futása közben.

Az alkalmazás ablakképe futás közben:



Működés:

- A program indításakor egy átméretezhető ablak nyílik meg, melyen csak a jobb alsó sarkához kötött nyomógomb látható.
- Az egérrel az ablakon kattintva a kattintás helyén egy „Szöveg” feliratú címkét hozunk létre. Valamely létrehozott címke fölé állva a háttérszíne és a szöveg stílusa változzon meg. A kurzorral elhagyva a címkét, annak megjelenése az eredeti tulajdonságoknak megfelelő legyen.
- A címkén kattintva véglegesen tüntessük el azt az ablakról.

Szükséges új ismeretek:

TLabel	Tulajdonságai: `Owner, Parent`
	Metódusai: `Create, Free, Destroy`

Egyéb ismeretek:

- A kattintás helyének meghatározásához az `onClick` esemény helyett az `onMouseUp` eseményt használjuk, melyben a létrehozott címkének a tulajdonságait és eseményeit is beállítjuk:

```
with TLabel.Create(Self) do begin
  Parent:=Form1;
  Left:= X+1;
  Top:= Y;
  Caption:='Szöveg';
```

```
    Color:=clYellow;
    Transparent:=True;
    Cursor:=crHandPoint;
    onMouseEnter:=EgerRa;
    onMouseLeave:=EgerLe;
    onClick:=Kattintas;
  end;
```

- A létrehozott címkéhez definiált három esemény eljárását deklarálnunk kel és a tartalmát is el kell készítenünk. A deklarálást a `TForm1` típuson belül végezzük el.

```
procedure EgerRa(Sender: TObject);
procedure EgerLe(Sender: TObject);
procedure Kattintas(Sender: TObject);
```

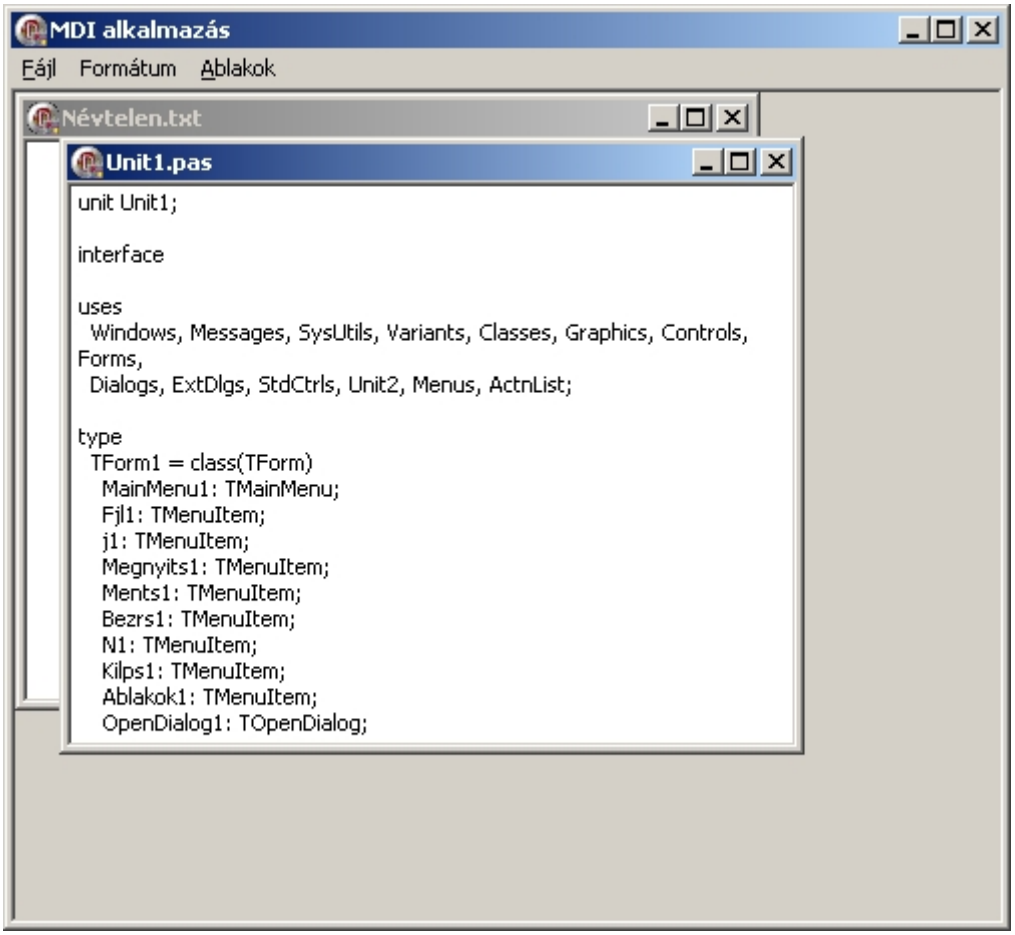
- Az eljárások formális paraméterlistájának olyannak kell lenni, hogy az `onMouseEnter`, az `onMouseLeave` illetve az `onClick` események paramétereinek megfeleljenek.
- Ezután az egyes események tartalma elkészíthető:

```
procedure TForm1.EgerRa(Sender: TObject);
begin
  with Sender as TLabel do begin
    Font.Style:=[fsBold];
    Transparent:=False;
  end;
end;

procedure TForm1.EgerLe(Sender: TObject);
begin
  with Sender as TLabel do begin
    Font.Style:=[];
    Transparent:=True;
  end;
end;

procedure TForm1.Kattintas(Sender: TObject);
begin
  (Sender as TLabel).Free;
end;
```

18. feladat: MDI alkalmazás

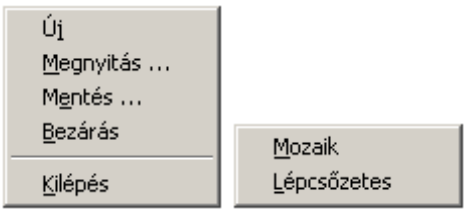
Az alkalmazás egy olyan egyszerű editor, melyben egyszerre több szöveges állomány is megnyitható és kezelhető.

Az alkalmazás ablakképe futás közben:

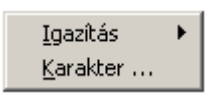


Működés:

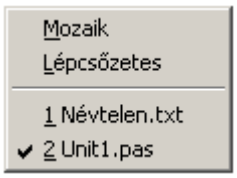
- Az alkalmazás indulásakor csak az üres keretprogram legyen látható a saját főmenüjével, melynek két eleme a `Fájl` és az `Ablakok` menü.



- A Formátum menü a megnyitott állományok menüje, mely a megnyitásuk után jelenik meg a keretprogram menüszerkezetében. Ebben a szokásos igazítások egy almenüből rádiógomb módjára egymást kizáróan választhatók. A karakterbeálltást szokásos karakterbeállító párbeszédablak segítségével végezzük el.



- A megnyitás során kiválasztott állomány neve a megnyíló ablak címsorában jelenjen meg. Ha új üres ablakot nyitunk meg, akkor annak címsorába a „Névtelen.txt” állománynév kerüljön.
- A megnyitott ablakok listája az Ablakok menüben legyen látható.



- A Mentés az aktív ablak tartalmának mentését biztosítja egy mentés párbeszédablak segítségével, a Bezárás parancs pedig az aktív ablakot zárja be. Figyeljünk, mert ha nincs aktív ablak, akkor se a mentés se a bezárás nem értelmezhető.
- Mind megnyitáskor, mind mentéskor a txt, dpr, pas, dfm állományok közül választhatunk.

Szükséges új ismeretek:

TForm	Tulajdonságai: FormStyle, WindowMenu
	Metódusai: Tile, Cascade
TMenuItem	Tulajdonságai: GroupIndex

Egyéb ismeretek:

- A gyermekablakként használt Form2 ablakot a program automatikusan létrehozza. Ezt le kell tiltanunk a fejlesztői környezet Project/Options menüpontján belül a Forms kategóriában.
- A gyermekablak megnyitásakor létre kell hozni azt, és be kell tölteni a kiválasztott állományt, vagy új ablak esetén üresen kell hagyni. Mindkét esetben el kell készíteni az ablak feliratát is:

```
if OpenDialog1.Execute then
  with TForm2.Create(Self) do begin
    Caption := ExtractFileName(OpenDialog1.FileName);
    Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
  end;
```

- vagy üres ablak esetén:

```
with TForm2.Create(Self) do Caption:='Névtelen.txt';
```

- A gyerekablak megnyitása után, ha maximalizáljuk azt, később nem tudjuk kisebb méretűre állítani. Ehhez a Form2-re egy MainMenu komponenst kell elhelyezni,

melynek hatására a gyerekablak címsorában az előző méretre állító gomb is megjelenik.

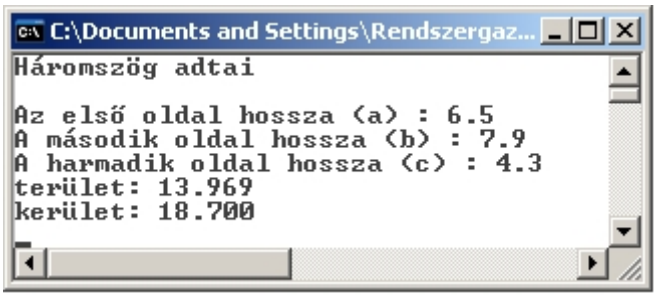
- Ebbe a `MainMenu` komponensbe hozzuk létre az ablakhoz kötött Formátum menüt is, mely megfelelő `GroupIndex` érték esetén az ablak megnyitásakor hozzáadódik a keretalkalmazás menüjéhez.
- A főablak és a gyerekablak menüi együtt határozzák meg a teljes menüszerkezetet. A főmenük és az alájuk rendel parancsok sorrendjét is a hozzájuk rendelt `GroupIndex` határozza meg. A feladatban például a `Fájl` és az `Ablakok` menü között jelenítjük meg a gyerekablakhoz kötött `Formátum` menüt. Ehhez a `Fájl` menühöz kell a legkisebb, az `Ablakok` menühöz a legnagyobb `GroupIndex` értéket rendelni, és a kettő közötti értékkel kell rendelkeznie a `Formátum` menünek. Ha két `GroupIndex` érték megegyezik, akkor a főablak adott menüjét a gyerekablak azonos `GroupIndex`-ű menüje felülírja.
- A megnyitott gyerekablakok számát az `MDIChildCount` tulajdonsággal tudhatjuk meg.
- Az `ActiveMDIChild` az aktív gyerekablakot azonosítja, mely a `Close` metódusával be is zárható. Ha nincs aktív gyerekablak, akkor a `nil` értéket adja eredményül.
- A fő ablak bezárásakor meghívja az összes gyerekablak `onCloseQuery` és `onClose` eseményét, és ha minden gyerekablak bezáródott, akkor a fő ablak is bezáródik. A gyerekablakok által foglalt erőforrások felszabadítását a gyerekablakokat reprezentáló `Form2 onClose` eseményében végezhetjük el az `Action` paraméter megfelelő beállításával:

```
Action := caFree;
```

19. feladat: Konzol-alkalmazás 1.

Az alkalmazás egy hagyományos felületen valósítja meg egy háromszög területének és kerületének meghatározásához szükséges adatok bekérését, majd kiírja az eredményeket.

Az alkalmazás ablakképe futás közben:



Működés:

- A háromszög adatait a három oldalának hosszából határozzuk meg. Az oldalhossz értékének csak pozitív számot fogadjon el a program. Helytelen adat megadásakor jelezze ezt a felhasználónak, és mindaddig kérje újra az adatot, amíg megfelelő értéket nem ad meg.
- Ha mindhárom érték megfelelő, ellenőrizni kell, hogy a megadott oldalhosszakból szerkeszthető-e háromszög. Ha igen, kiszámolja és megadott pontossággal kiírja az eredményeket a képernyőre a program, ellenkező esetben jelzi a felhasználónak, hogy az adatok alapján nem készíthető háromszög. Mindkét esetben biztosítsuk, hogy az ablak ne záródjon be automatikusan a kiírás után, hanem legyen lehetőségünk elolvasni a kiírásokat.

Egyéb ismeretek:

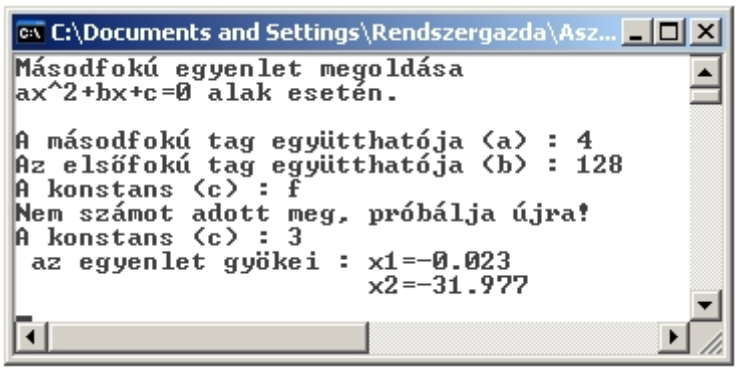
- A Pascalban megismert alapvető, `crt` unitban szereplő képernyőkezelő eljárások és egyes billentyűzet-kezeléssel kapcsolatos függvények nem alkalmazhatók. Ilyenek például: `clrscr`, `gotoxy`, `readkey`, `keypressed`. Ezek helyettesítése összetettebb eljáráskódok, függvénykódok írását igényli, melyekkel most nem foglalkozunk.
- A szöveg kiíratásakor a magyar ékezetes karakterek helyett egyéb karakterek jelenhetnek meg. Ennek elkerülésére alkalmazhatjuk a problémás karakterek ASCII kódja alapján történő megjelenítését. Például:

  `write('második');` helyett `write('m',#160,'sodik);`

20. feladat: Konzol-alkalmazás 2.

Az előző feladathoz hasonlóan ez az alkalmazás is hagyományos, karakteres ablakban fut. A program egy $ax^2+bx+c=0$ alakú másodfokú egyenlet `a`, `b` és `c` együtthatók értékeit kéri be a felhasználótól, és ha létezik, eredményként megadja az egyenlet valós gyökét vagy gyökeit.

Az alkalmazás ablakképe futás közben:



Működés:

- Az együtthatók értékének csak számokat fogadhat el a program, ellenkező esetben jelezze a hibás adat megadását a felhasználónak, és kérje újra az aktuális együttható értékét mindaddig, amíg számot nem ad meg.
- A program legyen képes felismerni, ha az adatok alapján az egyenlet elsőfokú, és így csak egy gyöke van, valamint kezelje azt is, ha az egyenlet másodfokú ugyan, de egy valós gyökkel, vagy akár egyetlen valós gyökkel sem rendelkezik.
- Készítsük fel a programot arra az esetre is, ha az `a` és `b` együttható értéke is 0.

6. Összefoglalás

A szakdolgozatomban röviden ismertettem az érettségi vizsgakövetelményeiből származó dilemmákat a programozás oktatása körül. A leírtakat még jobban erősíti az, hogy az Oktatási Hivatal tájékoztatása szerint a természettudományos tárgyak meghatározóbbá tétele érdekében várhatóan bevezetik az ötödik, még szabadon választható érettségi tantárgyak körének szűkítését, miszerint ez csak valamilyen természettudományos tantárgy lehet. Ebbe a körbe nem tartozik sem az informatika, sem az informatikai alapismeretek.

A bemutatás során megpróbáltam érzékeltetni, hogy mire lehet elegendő a magas óraszámú informatikai alapozó oktatás tantárgyon belül a vizuális programozásra fordított idő. Ezen óraszám keretében a tanulókkal megkedveltethető a programozás is, főleg azzal az eszközkészlettel, amit egy 4GL vizuális fejlesztőkörnyezet a tanuló kezébe ad.

A szakdolgozatban inkább a vizuális programozás gyakorlati oldalát emeltem ki, mert itt érezhetők igazán a lehetőségek még akkor is, ha nem szóltam az adatbáziskezelés, a hálózati kommunikáció, az ActiveX és a többszálú programok készítésének lehetőségeiről, csak úgy mint ahogy a bonyolultabb ablakfelépítésekről sem.

A tanulók nagy előszeretettel alkalmaznának kisebb-nagyobb trükköket is a programjaikban. Azoknak a tanulóknak, akiket megfertőzött a vizuális programozásban rejlő látvány és lehetőség, ajánlani szoktam a www.swissdelphicenter.ch weblapot, ahol több ezer praktikus és akár vicces, tipp (tálca eltüntetése, asztalra rajzolás, stb.) is megtalálható. Ezek felhasználásával jelentősen bővül a tanulókkal történő differenciált foglalkozás lehetősége is.

7. Irodalomjegyzék

[1] A Mechwart András Gépipari és Informatikai Szakközépiskola helyi tanterve

[2] 100/1997. (IV.13.) Kormányrendelet az érettségi vizsga vizsgaszabályzatának kiadásáról

[3] 2. számú melléklet a 100/1997. (IV.13.) Kormányrendelethez

[4] Vizuális módszerek oktatásának hatása a hallgatók programozási hibáira – (tanulmány) - Kecskeméti Főiskola, GAMF Kar, Kalmár Sándor Intézet - 2007

[5] Reményi Zoltán – Informatika érettségi tapasztalatai és változásai (előadásvázlat) – 2008.08.21-22.