Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Bevezetés a programozásba II

Feladatgyűjtemény

Összeállította:
Giachetta Roberto
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

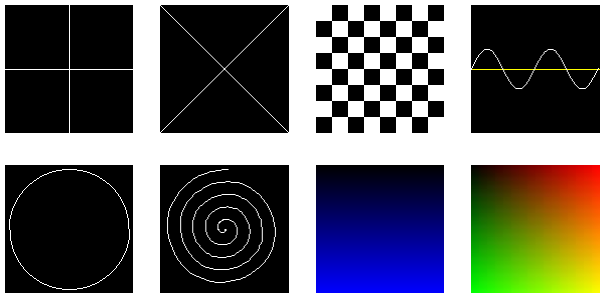
Gelencsér András
andras.gelencser@itk.ppke.hu

Frissítve:
2014. március 14.

I. Strukturált programozás

1. A grafikus könyvtár használata

1.1. Rajzoljuk ki az alábbi alakzatokat a grafikus képernyőre.



1.2. Bal egérkattintásra változtassuk meg a képernyő hátterének színét egy új, véletlenszerű színre.

1.3. Készítsünk egy olyan programot, ami elhelyez egy színes pöttyöt az egérpozícióba minden balegér gomb lenyomáskor.

1.4. Készítsünk egy törtvonal rajzoló programot, ahol a vonal törés pontjait az egérrel adhatjuk meg. A vonal kezdő pontja az első elhelyezett törés pont legyen.

1.5. Készítsünk egy programot, melyben az egér segítségével folytonos görbe vonalat tudunk rajzolni.

1.6. Készítsük el az előző program tovább fejlesztett verzióját, ahol a görbe nem egyszínű, hanem szín átmenetes.

1.7. (*) A vonal színe a szivárvány színeinek megfelelően változzon, miközben rajzoljuk ki.

1.8. Készítsünk egy olyan programot, ahol az egér gombjai segítségével különböző színű spray effektust valósítunk meg a képernyőn.

1.9. Írjuk ki a képernyő bal felső sarkába a program indítása óta eltelt másodperceket.

1.10. Készítsünk egy olyan programot, ami dobozokat helyez el a képernyőn egérklikkelésre. Az új dobozok színe 5 másodpercenként változzon meg.

1.11. Írjuk ki a grafikus képernyőn az aktuálisan lenyomott billentyűt az egérpozícióba.

1.12. Készítsünk programot, amely a képernyő bal felső sarkától az egérmutatóig húz egy piros vonalat, amely végig követi az egér mozgását.

1.13. Rajzoljunk ki egy körpályán mozgó dobozt a képernyőre.

1.14. Rajzoljuk egy 100×100-as négyzetet a képernyő bal felső sarkába, majd egérmozgás hatására mindig mozgassuk az egér felé (de csak egyesével). ESC billentyű hatására a program lépjen ki.

1.15. Fejlesszük tovább az előző programot úgy, hogy minden egérkattintásra egy újabb négyzet jelenik meg (véletlenszerű pozícióban), és minden, a képernyőn megjelenő négyzet kövesse az egeret.

1.16. Készítsünk egy olyan programot, ahol egér kattintással elhelyezhetünk egy négyzetet a képernyőn, ami egyenletes sebességgel jobbra halad, és ha eléri a képernyő szélét ott megáll.

1.17. (*)Fejlesszük tovább az előző programot úgy, hogy a dobozok ne csúszhassanak egymásra, hanem akadjanak fent egymáson.

1.18. (*)Adjunk hozzá olyan funkciót is, hogy már meglévő dobozokra klikkelve azok eltűnnek és a többi doboz, ha lehetséges, elfoglalja a felszabadult helyet.

1.19. Készítsünk programot, amely hóesést szimulál. A hópihék hulljanak a képernyő tetejéről az aljáig (majd ismét a tetejéről, tehát ciklikusan), szóköz lenyomására pedig kerüljön be újabb hópihe a programba.

1.20. Fejlesszük tovább az előző programot úgy, hogy a hópihe esés közben himbálózó vízszintes mozgást is végezzenek (némileg véletlenszerűen, tehát az egyes hópihék mozgása különböző legyen).

1.21. (*) Fejlesszük tovább az előző programot úgy, hogy szélfúvást is szimuláljuk. A szél mindig az egér irányába fúj, és erőssége függ az egérpozíció távolságától a képernyő közepéhez képest.

1.22. (*) Adjunk térhatást a hóviharhoz úgy, hogy nem csak fehér, hanem sötétebb árnyalatú (távolabb lévő) hópihéket is kirajzolunk. A közelebbi hópihék gyorsabban essenek és nagyobb hatást is fejtsen ki rájuk a szél.

1.23. (*) Fejlesszük tovább az előző programot úgy, hogy a hópihék nem egy-egy pontból állnak, hanem három vonalból álló hókristályok, amelyek középen keresztezik egymást (mint egy hatszög átlói), és (véletlenszerű) forgó mozgást végeznek esés közben.

1.24. Készítsünk író gép programot. A betűk, számok, írásjelek jelenjenek meg szöveg szerűen a kijelzőn. Figyeljünk a sortörésekre! BACKSPACE segítségével lehessen kitörölni a bevitt karaktereket (egyszerre egyet). Figyeljünk arra, hogy ha egy másik sorba kell vissza ugrani, valamint, ha már nincs több karakter, akkor ne tudjunk törölni!

1.25. Rajzoljunk ki egy dobozt, mely követi az egér kurzort. BAL, JOBB kurzor billentyűvel tudjuk szabályozni a méretét, FEL, LE kurzor billentyűvel pedig a sebességét.

Figyeljünk arra, hogy ne tűnjön el a doboz, ha elérte a kurzort vagy, ha a kurzor elhagyta a képernyőt!

1.26. Rajzoljunk ki 100 db színes dobozt, ahol minden egyes doboz egy másikat követ láncban.

1.27. Rajzoljunk ki 100 db színes muslicát (2x2es dobozok) a képernyőre, melyek egyhelyben imbolygó mozgást végeznek. SPACE billentyűvel 50 újabb muslicát lehet hozzá adni a rajhoz. BACKSPACE billentyűvel pedig törölni lehet 50-et, de figyeljünk arra, hogy ne lehessen 50db muslicánál kevesebb!

1.28. Fejlesszük a muslicás programot úgy, hogy egérklikkeléskor 100 pixel sugarú körben a muslicák a kurzorhoz repülnek.

1.29. (*) Készítsünk egy végtelenített hiperűr ugrásszerű animációt, ahol a csillagok a képernyő közepéből véletlenszerűen a képernyő széle felé tartanak egyenes irányban. A közeledő csillagok legyenek egyre fényesebbek és nagyobbak.

2. Mátrixok a grafikus felületen

2.1. Rajzoljunk véletlenszerűen kék pöttyöket a képernyőre, majd billentyű lenyomására invertálódjon a képernyőt, azaz a kék pöttyök feketék, a fekete pöttyök kékek legyenek

2.2. Osszuk fel a 400 × 400-as képernyőt 50 × 50-es mezőkre, amelyek kezdetben zöldek. Zöld mezőre kattintva az fehérre vált, fehérre kattintva pirosra, pirosra kattintva pedig zöldre.

2.3. Egészítsük ki az előző programot úgy, hogy ne csak a kattintott négyzet, hanem annak 8 szomszédja is színt váltson a szabályoknak megfelelően (ha létezik).

2.4. Egészítsük ki az előző programot úgy, hogy a felhasználó billentyűzettel megváltoztathassa a tábla méretét úgy, hogy amennyiben számbillentyűt üt le, a tábla átméreteződjön vízszintesen és függőlegesen is a megadott számúra. Az újonnan bekerült sorok és oszlopok legyenek zöldek, de a régiek ne változzanak meg.

2.5. Készítsünk programot, amely egy színes négyzetrácsot fest a képernyőre, amelynek vízszintes és függőleges méretét változtatni tudjuk úgy, hogy a megmaradt mezők megtartják színüket. Kezdetben legyen fekete a képernyő, és ha valamilyen számot leütünk a billentyűzeten, akkor akkora mátrixban rajzolja ki a színeket.

2.6. Készítsünk alkalmazást, amely beolvas egy képfájlt (.kep). A fájl nevét kérjük be konzol ablakon keresztül a felhasználótól, majd jelenítsük meg azt egy ugyanakkora méretű ablakban.

2.7. Készítsünk olyan alkalmazást, amely egy kis képet megjelenít az egér pozíciójában.

2.8. Készítsünk egy olyan alkalmazást, amely a háttérképet csak az egér egy adott környezetében teszi láthatóvá, máshol csak a fekete képernyőt látjuk.

2.9. Készítsünk egy képfeldolgozó alkalmazást, ami a betöltött képet képes megjeleníteni az eredeti színein, az invertált színein, fekete-fehérben és szürkeskálásan is. A különböző nézetek között gombnyomással lehessen navigálni.

2.10. Készítsünk egy olyan képernyőt, amin fehér zaj van („hangyafoci”).

2.11. Készítsünk egy ablaktörlő alkalmazást, ahol először csak egy „koszos felületet” látunk, de az egér balgombját lenyomva tartva le tudjuk törölni a képernyőt és megjelenik egy szép háttérkép.

2.12. Bővítsük az előző alkalmazást úgy, hogy lehessen a képet nagytani is az egérgörgő segítségével. Minden felfelé görgetés megduplázza a pixelek méretét, a bal felső sarokból indulva, minden lefelé görgetés felezi.

2.13. (*) Bővítsük az előző alkalmazást úgy, hogy a képet ne a bal felső sarokból, hanem az egérpozícióból indulva nagyítsuk.

2.14. (*) Készítsünk képnézegető alkalmazást. A program jelenítse meg a könyvtárban lévő képeket egymás után 10 másodperces késletetéssel úgy, hogy a képek között áttűnési effektet használ. Az effekt legyen a kép fokozatos lecserélése balról jobbra, majd a következő lépétetésnél jobbról balra haladva, és az időtartama legyen 2 másodperc. Tegyük fel, hogy a könyvtárban a képek 1-től számozva vannak (1.kep, 2.kep, ...), de nem tudjuk, összesen hány kép van. Az utolsó kép után ismét az első képet hozza be a program. Lehessen szóközzel azonnal léptetni a képet (így ne várja meg a 10 másodpercet, de az effekt maradjon meg).

3. Adattípusok megvalósítása

3.1. Készítsük el a téglalap típusát, amelynek megadhatjuk a méretét és a színét. Legyen lehetőség a téglalapot kirajzolni a képernyő egy adott pontjára, átszínezni, illetve átméretezni. A műveleteknél ügyeljük rá, hogy csak megfelelő értékeket fogadjon el (pl. negatív méretet ne). Készítsünk programot, amellyel a képernyőre történő bal kattintásra az egérpozícióba kirajzolunk egy téglalapot, jobb kattintásra pedig megváltoztatjuk a színét.

3.2. Készítsük el a kör típusát, amelynek megadhatjuk a sugarát és a színét. Legyen lehetőség a kört kirajzolni a képernyő egy adott pontjára, átszínezni, illetve átméretezni. A műveleteknél ügyeljük rá, hogy csak megfelelő értékeket

fogadjon el (pl. negatív sugarat ne). Készítsünk programot, amellyel a képernyőre történő bal kattintásra az egérpozícióba kirajzolunk egy kört, jobb kattintásra pedig megváltoztatjuk a színét.

3.3. (*) Készítsünk alkalmazást, amellyel tetszőleges sok téglalapot, illetve kört tudunk rajzolni a képernyőre (bal kattintásra rajzolunk az egérpozícióra, jobbal megváltoztatjuk az alakzat színét). Ehhez egészítsük ki a kör és téglalap típusokat olyan művelettel, amely megállapítja, hogy az adott pozíció rajta helyezkedik-e el.

3.4. (*) Fejlesszük tovább az előző alkalmazást. Legyen lehetőség az alakzatok áthelyezésére (bal egérgombot lenyomva tartva), valamint méretezésére (egérgörgő segítségével).

3.5. Készítsük el a kép típusát, amelynek megadjuk a .kep fájl elérési útvonalát, és ez alapján betölti, valamint eltárolja a képet, továbbá lehetőséget ad a képernyő egy adott pozíciójában történő kirajzolásra.

3.6. Készítsünk alkalmazást, amely betölt egy háttérképet, valamint egy előtér képet, amely egy figura látható. A figura kép környéke fekete színű, ehelyett a háttér jelenjen meg. A figurát lehessen balra, illetve jobbra mozgatni a képernyőn.

3.7. (*) Fejlesszük tovább az előző alkalmazást. A figura nem csak egy, hanem egymást követő 6 kép összessége, amellyel a figura mozgását tudjuk animációszerűen megjeleníteni. A mozgatás történjen úgy, hogy minden lenyomásra a figura a 6 animációs lépést véghezvigye.

3.8. Készítsünk egy visszaszámláló óra típust. Az órának megadhatjuk a hátralévő egységet (időt), illetve kirajzolhatjuk a képernyő egy adott pontjára, valamint egy metódussal csökkenthetjük eggyel az értékét, amely ha nullára csökken, a „vége” felirat jelenik meg. Készítsünk alkalmazást, amely megjeleníti az időzítőt, és minden egérmozgásra csökkenti annak értékét.

4. Algoritmusok és adatszerkezetek alkalmazása

4.1. Készítsünk egy képmegjelenítő alkalmazást, amely lehetőséget ad a kép világosítására, illetve sötétítésére. A program egérgörgő hatására növelje, illetve csökkentse az értékeket, szóköz hatására állítsa vissza az eredeti állapotot.

4.2. Készítsünk egy képmegjelenítő alkalmazást, amely a következő statisztikákat készíti el egy szürkeskálázott képről: intenzitásértékek átlaga, szórása, legvilágosabb képpont, legsötétebb képpont, fehér képpontok száma, fekete képpontok száma. A statisztikákat a program írja ki a kép alá.

4.3. Készítsünk egy képmegjelenítő alkalmazást, amely megkeresi a legvilágosabb, illetve legsötétebb képpontokat a képernyőn, és előbbieket sárgán villogva, utóbbit pirosan villogva jeleníti meg.

4.4. Készítsünk alkalmazást, amely egy konzol felületen megadott, kerek zárójeleket tartalmazó kifejezést visszaír a grafikus képernyőre úgy, hogy az egyazon zárójelbe tartozó részkifejezéseket ugyanolyan színnel rajzolja ki. A program külön jelezze, ha a megadott kifejezés hibásan zárójelezett.

4.5. Bővítsük az előző alkalmazást úgy, hogy szögletes és kapcsos zárójelpárokat is felismerjen.

4.6. (*) Szimuláljuk a memóriánk működését a grafikus felület segítségével. Egy fájlból beolvashatunk soronként utasításokat, amelyek a következők lehetnek: blokk nyitás (**{**), blokk bezárás (**}**), illetve egész szám létrehozása: (**int *<változónév>***). A program dolgozza fel a fájlt soronként, és jelenítse meg, ahogy a memóriában lévő veremben egymás után létrejönnek a változók, majd törlődnek a blokk végén.

4.7. Készítsünk egy egyszerű korházas játékprogramot. A kórházban betegek érkeznek, akiket orvosok tudnak ellátni, az orvosokat pedig a kórház veheti fel, illetve küldheti el. Az ellátás valamennyi időt vesz igénybe. Az orvosok pénzbe kerülnek, de a kórház kap állami támogatást, illetve minden kezelt beteg után plusz támogatást. A betegek száma az idő múlásával egyre növekszik, így a játék célja, hogy a korház minél később menjen tönkre úgy, hogy az orvosra várakozó betegek száma nem haladhat meg egy adott korlátot, mert akkor fellázadnak a betegek.

4.8. Bővítsük az előző programot úgy, hogy kétféle beteg érkezhet a kórházba annak függvényében, milyen súlyos az állapota. A súlyosabb betegek mindig elsőbbséget élveznek a kevésbé súlyosakkal szemben, tehát utóbbiakat csak akkor lehet kezelni, ha nincs várakozó súlyos beteg. A megvalósításhoz használjunk két sort.

4.9. Bővítsük az előző programot úgy, hogy a betegeknél sürgősség szerint 10 kategóriát különböztetünk meg, és előbb mindig a magasabb kategóriájú betegeket kell ellátni. Ehhez ne 10 sort vegyünk fel, hanem csupán egy adatszerkezetet, ahonnan nem az első várakozót, hanem mindig a legmagasabb prioritásút vesszük elő.

4.10. (*) Készítsünk játékot, amellyel egy informatikus csoportot játszhatunk egy nagyvállaltnál. Az IT csoportba felvehetünk, illetve elküldhetünk informatikusokat. A csoporthoz a vállalat részlegvezetői fordulnak segítségért (idővel egyre több), akik türelme eltérő, és a hívástól kezdve csak addig hajlandóak várni, amíg el nem fogy a türelmük, ha elfogy, akkor a csoportvezetőt (a játékost) kirúgják. A segítségnyújtás egy informatikust

igényel, és ideje problémánként különböző lehet. A csoport kap valamennyi pénzt a vállalattól, illetve minden segítségnyújtás után úgyszintén, cserébe az informatikusokat fizetni kell. A csoport költségvetése sohasem mehet 0 alá (akkor maximum kirúgnak megfelelő számú informatikust). A játék célja, hogy minél később rúgják ki a játékost.

5. Objektumorientált alkalmazások megvalósítása

5.1. Készítsünk egy alkalmazást, amelyben négyzeteket tudunk elhelyezni. A négyzeteket szóköz hatására helyezzük el, abba a pozíció, ahol az egér is található. A négyzeteket lehessen nagyítani bal kattintásra, illetve zsugorítani jobb kattintásra. Természetesen csak az a négyzet változzon, amely felett az egér elhelyezkedik.

5.2. Módosítsuk az előző alkalmazást úgy, hogy a négyzeteket egymásra ne engedje helyezni a program, nagyításnál pedig ne lehessen tovább nagyítani, ha azzal egy másik négyzetnek ütközne az elem.

5.3. Módosítsuk az előző alkalmazást úgy, hogy a vezérlést nem a főprogram, hanem egy alkalmazás osztály végzi, amely lehetőséget ad a kilépésre (ESC hatására), illetve megadható neki a kezdeti képernyőméret. A főprogram feladata csak annyi legyen, hogy példányosítja, majd elindítja az alkalmazást.

5.4. Rajzoljunk ki különböző, de állandó sebességgel guruló golyókat a képernyőre, melyek visszapattannak a képernyő széléről. Legyenek köztük, olyan golyók is, melyek megváltoztatják a színűket, a képernyő széléről történő visszapattanáskor. Használjunk öröklődést!

5.5. Egészítsük ki a programot úgy, hogy a mozgó golyókat megfoghassuk az egér segítségével. Ilyenkor a golyó álljon meg és legyen lehetőségünk arrébb helyezni. Ha elengedtük folytatja az eredeti sebességével a mozgását az új pozícióból.

5.6. (*) Fejlesszük tovább az előző programot úgy, hogy a golyók egymásról is visszapattanjanak.

5.7. (*) Rajzoljunk ki pattogó labdákat a képernyőre. Pattanjanak vissza a földről és az oldalakról, de hasson rájuk a gravitáció! Idővel egyre kisebbeket pattogjanak és teljesen álljanak meg.

5.8. Készítsünk képszerkesztő alkalmazást, amely lehetőséget biztosít szürkeskálázásra, homályosításra, fényerősség és kontrasztváltoztatásra, illetve Emboss szűrő alkalmazására. A programban lehessen a keletkezett képet elmenteni, illetve a műveleteket visszavonni. Az Emboss szűrő egy olyan élkiemelő leképezés, amely a képponttal átlósan szomszédos értékeket kivonja az képpont értékének arányos szorzatából, majd világosítja azt szürkére.

5.9. Egészítsük ki a képszerkesztő programunkat alkalmazás osztállyal, amely megkapja a megnyitandó fájl nevét, valamint a képernyő méretét. A főprogram feladata csupán annyi legyen, hogy elindítja az alkalmazás egy példányát.

5.10. (*) Egészítsük ki az előző alkalmazást az invertálás (minden szín az ellentettjére vált), valamint a küszöbölés (egy adott fényerősség felett az értékek fehérek, alatta feketék lesznek) funkcióival. Továbbá legyen lehetőség a visszavont műveleteket újra alkalmazni.

5.11. Jelenítsünk meg a képernyőn két 100*100-as piros négyzetet. Egyik kattintás hatására váltson valamilyen más színre, a másik kerüljön más helyre a képernyőn. A megvalósításhoz használjunk öröklődést.

5.12. Egészítsük ki az előző programot úgy, hogy tetszőlegesen sok négyzetet lehessen létrehozni, és mindig véletlenszerűen döntse el, hogy melyik fajtát. A program vezérlését végezze egy alkalmazás osztály.

5.13. (*) Készítsünk rajzolóprogramot, amellyel különböző alakzatokat tudunk rajzolni (négyzet, kör, egyenlő szárú háromszög). Az alakzatokat lehessen utólag méretezni, törölni, illetve áthelyezni a képernyőn az egér segítségével.

6. Grafikus felületű alkalmazások megvalósítása

6.1. Készítsünk egy kijelölő négyzet típust (checkbox) az ős vezérlőből származtatva. Bal egér gombbal lehessen változtatni a doboz bejelöltségét. Készítsünk több fajta színű, méretű illetve jelű kijelölő négyzet példányt.

6.2. Hozzunk létre egy olyan címkét, amely adott színű háttér előtt, adott színű szöveget jelenít meg és, ha fókuszba kerül, akkor a háttér kivilágosodik (highlight effect).

6.3. Készítsünk egy nyomógomb és egy címke típust egy ős vezérlőből származtatva. A címke feladata egy szöveg kiírása, míg a gombra kattintani lehet, amelynek hatására elvégez valamilyen tevékenységet. A gombból létrehozhatunk egy még speciálisabb gombot, amelynek tevékenysége egy (paraméterben megadott) címke szövegének átírása. Egy programban hozzunk létre egy címkét és egy nyomógombot, majd kattintásra a nyomógomb változtassa meg a címke feliratát.

6.4. Hozzunk létre az alap nyomógombból tovább fejlesztett gomb típusokat. Legyen kétállású gomb, amit, ha megnyomunk „beragad” és újbóli megnyomás esetén kerül újra alap állapotba. Készítsünk feliratozható gombot is, mely a példányosításkor kapott szöveget megjeleníti magán.

6.5. Készítsünk egy egysoros szövegszerkesztő vezérlőt, amelynek tartalmát futás közben tudjuk karakterenként beírni, és azt természetesen lekérdezni. Az

alfanumerikus karaktereket (40 és 177 közötti ASCII kód) dolgozzuk fel, illetve BACKSPACE billentyű hatására töröljük az utolsó karaktert. Természetesen csak akkor szerkeszthető a szöveg, ha rajta van a fókusz, ezt úgy jelöljük, hogy kirajzolunk még egy aláhúzás jelet a szöveg után. Ügyeljünk továbbá arra, hogy a szöveg túllóghat a vezérlőn, és ilyenkor csak a végét jelenítsük meg, ami még belefér a szövegdobozba. Módosítsuk a címke átíró gombunkat úgy, hogy ne előre beírt szöveget ír a címkére, hanem egy szövegdobozból olvassa ki a tartalmat, és azt írja a címkére.

6.6. Valósítsuk meg három gomb és egy egysoros szövegbeviteli mező segítségével a következőket: A mezőbe írt szöveg karaktereit csupa nagybetűre illetve csupa kisbetűre tudjuk állítani a megfelelő gomb megnyomásával, valamint felcserélhetjük a karakterek sorrendjét a harmadik gomb segítségével.

6.7. Készítsünk fájlbeolvasó vezérlőt, amellyel egy szöveges fájl tartalmát tudjuk a képernyőre írni, és nyissunk meg fájlokat vele a programban. Ügyeljünk arra, hogy az ablakba csak annyi sor kerüljön be, amennyi befér az ablakba, mivel a fájlt teljes egészében nem tudjuk megjeleníteni, biztosítanunk kell görgetést is a felhasználó számára, történjen ez a fel-le billentyűk hatására. Készítsünk egy új gombot, amely egy szövegmezőbe beírt fájlnév alapján tölti be a fájlt a vezérlőbe

6.8. Készítsünk el egy egyszerű számológép alkalmazást, mely képes az alapvető műveletek (+, -, *, /) elvégzésére és a bevitt számokat valamint az eredményt egy kijelzőn keresztül mutatja.

6.9. (*) Készítsünk egy előugró üzenet ablakot, amely megjelenít egy előugró ablakot a képernyőn, rajta egy címmel, felirattal, illetve két gombbal (OK, Mégse). A két gomb másféle tevékenységet végezhet annak függvényében, milyen konkrét gombot rendelünk hozzá. Készítsünk programot, amely betölt egy fájl tartamát a fájlmegjelenítővel, de a betöltés előtt még ellenőrzi, hogy jó-e a fájlnév. Ha nem az, akkor egy előugró üzenetben figyelmeztessen erre.

6.10. (*) Készítsünk képmegjelenítő vezérlőt, amely megkap egy fájlnevet, és megjeleníti a képi tartalmat. A megjelenítő mérete legyen rögzített, és ha a betöltött kép nagyobb, akkor a nyíl billentyűk segítségével lehessen mozgatni a megjelenített képrészletet.