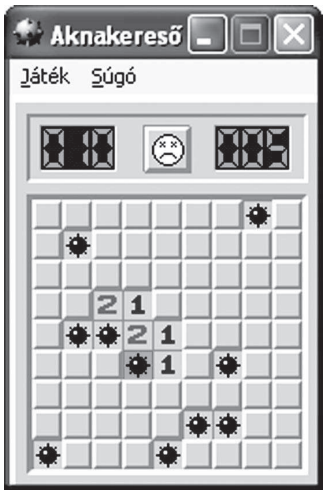
9. Játék a véletlennel

Véletlenszámok generálása

Eddigi kifejezéseink előre meghatározott mennyiségekkel (változókkal, literálokkal) végeztek műveleteket. Sok esetben azonban szükségünk van véletlenszerűen választott számokra. A véletlenszámok fontosak a titkosításban (kriptográfia), a játékprogramokban, továbbá a valóságos folyamatok számítógépes szimulációjánál, elemzésénél.



A játékprogramok gyakran alkalmaznak véletlenszámokat

A programok nem képesek valóban véletlen választásra, úgynevezett álvéletlen számsorozatokat hoznak létre. Kiindulnak egy alkalmasan választott kezdőértékből, majd a sorozat következő tagját az előző tag alapján számítják ki. A sorozattól elvárjuk, hogy ne legyen könnyen felismerhető a szabályosság és az ismétlődés az elemek között.

A sorozat kezdőértékét például a számítógép rendszerórája által mutatott időből vagy a programozó által megadott számból képezik. Ez utóbbi esetben a program többszöri futtatása során mindig ugyanazt a sorozatot kapjuk, ami főleg a program tesztelésénél fontos.

Véletlenszámokat a *Véletlenszám* objektumosztály metódusaival képezünk. A metódusok meghívásához először létre kell hoznunk a *Véletlenszám* osztályhoz tartozó objektumot.

Eddigi programjainkban már készítettünk új objektumokat. Amikor az eszközkészletből egy vezérlőelemet, például egy szövegdobozt helyeztünk az űrlapra, akkor a szövegdobozosztály egy objektumát hoztuk létre. A hozzá tartozó forráskódot a fejlesztőrendszer illesztette a programba, és általában elrejtette előlünk. A továbbiakban mi magunk deklarálunk és hozunk létre objektumokat.

Érték és hivatkozás típusú változók

Az objektumok létrehozása előtt röviden ki kell térnünk a változónevek és a nekik megfelelő memóriaterületek kapcsolatára. A numerikus változók azt a területet szimbolizálják, ahol a változó értéke található a memóriában. Fogalmazhatunk úgy is, hogy az értéket maga a változó tartalmazza. Ezzel ellentétben az objektumokhoz kapcsolódó változók csak annak a memóriaterületnek a kezdőcímét jelzik, ahol az objektumot (tulajdonságokat, metódusokat stb.) leíró kód elhelyezkedik a memóriában. A változó nem „értéket”, hanem az objektumot tároló területre mutató hivatkozást (memóriacímet) tartalmaz.

Érték típusú változó: a változó értékét tartalmazza.
Hivatkozás típusú változó: azt a címet tartalmazza, ahol a változó kódja megtalálható a memóriában.

A numerikus (továbbá a karakter- és a logikai) változók az érték típushoz tartoznak, a sztringek, objektumok hivatkozás típusúak[10]. A hivatkozás típusnak a későbbiekben látjuk majd néhány következményét.

Új objektum létrehozása

Egy objektum létrehozása két lépésből áll:

1. Deklarálunk egy új, az objektumra mutató változót. A deklarációban jelezzük az objektum típusát (osztályát):
   VÁLTOZÓ *Változónév* MINT *Osztály*
2. Az *Új* operátorral létrehozzuk az objektumot, és hozzárendeljük a változóhoz:
   Változónév = Új *Konstruktor*(*argumentumok*)

Az *Új* operátor után az objektumosztály egy speciális metódusa, az úgynevezett konstruktor áll.

Konstruktor: egy új objektum létrehozásakor végrehajtásra kerülő metódus. A konstruktor ad kezdőértéket az objektum tulajdonságainak, és hajt végre más, a létrehozással járó tevékenységet.

A konstruktor neve gyakran megegyezik az objektumosztály nevével. A konstruktornak a többi eljáráshoz hasonlóan lehetnek argumentumai. Az objektum létrehozásánál ne feledkezzünk meg az *Új* operátor alkalmazásáról, a konstruktor meghívásáról!

Véletlenszámok választása

Véletlenszámok használatához létre kell hoznunk egy új, véletlenszám-objektumot. A konstruktornak megadhatunk egy egész számot, amelyből a program a véletlenszámsorozat kezdőértékét képezi:

VÁLTOZÓ *Változónév* MINT Véletlenszám
Változónév = Új Véletlenszám(*egész szám*)

Ha a konstruktort argumentum beírása nélkül hívjuk meg, akkor a program véletlenszerűen választ kezdőértéket a sorozathoz.[11] Így az egyes futtatásoknál más-más számsorozatot kapunk:

Változónév = Új Véletlenszám()

[10] Több programozási nyelvben a sztring is érték típusú változó.
[11] Egyes programozási nyelvekben ehhez végre kell hajtani egy speciális utasítást, például: *randomize*.

A véletlenszám-objektum legfontosabb metódusai:

Következő()	nem negatív egész véletlenszámot eredményez;
Következő(max)	véletlen egész szám a [0; *max*) intervallumból;
Következő(min, max)	véletlen egész szám a [*min*, *max*) intervallumból;
KövetkezőTört()	véletlenszám a [0; 1) intervallumból.

Figyeljünk arra, hogy balról zárt, jobbról nyílt intervallumok szerepelnek az értékkészletben! A visszatérési érték lehet egyenlő az alsó határral, de kisebb, mint a felső határ.

Az egyes programozási nyelvek nem mindig rendelkeznek az objektum összes itt felsorolt metódusával.

1. gyakorlat. Írjunk programot, amely a *Következő* parancsgomb hatására kockadobást szimulál! (Kiír egy véletlenszerűen választott számot az [1; 6] intervallumból.)

2. gyakorlat. Írjunk programot, amellyel kockapókert lehet játszani. A kockapókerben 5 dobókockát kell egyszerre feldobni. Ha az összes kockával 6-ost dobtunk, akkor a program gratuláljon a felhasználónak!

Konstansok alkalmazása

A kockadobásra írt programokat könnyen átalakíthatjuk úgy, hogy tetszőleges számú oldallappal rendelkező „kockát” szimuláljanak. A metódushívásban szereplő, felső határt jelző argumentumot azonban több utasításban kell kijavítani. A forráskódban szereplő kifejezésekbe nem célszerű literálokat írni. Ugyanaz az érték több helyen előfordulhat, így nehézkessé válik az utólagos módosítás. A forráskód olvashatósága is nő, ha a konkrét értékek helyett beszédes elnevezéseket alkalmazunk. Az olvashatóság érthetővé teszi a forráskódot, megkönnyíti a hibakeresést, a hibajavítást.

A literálokat konstansokkal (állandókkal) helyettesíthetjük a kifejezésekben.

Konstans (fordítási konstans, sztatikus konstans): névvel ellátott érték. A konstansnak megfelelő értéket (számot, karaktersorozatot stb.) a fordítóprogram a forráskódban behelyettesíti a konstans azonosítójának a helyére.

A konstansok deklarációja hasonlít a változók deklarálásához:

KONSTANS *név* MINT *típus* = *érték*

A konstansok értékét literállal adjuk meg, vagy a változók kezdőértékéhez hasonlóan olyan kifejezéssel definiáljuk, melyet a fordítóprogram ki tud értékelni. Ügyeljünk arra, hogy a deklarációt kivéve konstans nem állhat egy értékadó utasítás bal oldalán!

A konstans értéke a futás során nem módosítható. A kifejezésekben szerepelhetnek konstansok, illetve egyetlen konstanst szintén kifejezésnek tekintünk.

3. gyakorlat. Módosítsuk az 1. gyakorlat programját! A felső határ értékét adjuk meg konstansként!

Konstansok helyett változókat is alkalmazhatnánk. A konstansok értékét azonban a fordítóprogram már a megfelelő módon kódolva behelyettesíti a kifejezésbe, így gyorsabbá, hatékonyabbá válik a program futása.

Az egyes programozási nyelvek beépített konstansokkal segítik a programozó munkáját. Ilyen lehet a π sok tizedesre megadott értéke, vagy az objektumok *Szöveg* tulajdonságánál felhasználható *Újsor* sztringkonstans. A beépített konstansokat nem kell deklarálni.

4. gyakorlat. Rendelkezik-e a kiválasztott programozási nyelv a π beépített konstanssal? Ha igen, jelenítsük meg az értékét!

A képek futásidejű módosítása

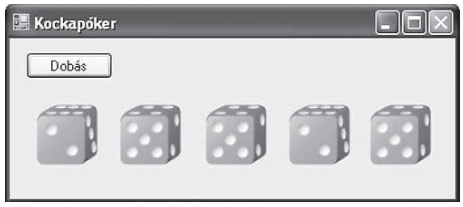
Látványossá tehetjük a kockadobást szimuláló programjainkat, ha egy képdobozobjektummal megjelenítjük a dobókocka megfelelő lapját. Ehhez az objektum *KépHelye* tulajdonságát kell módosítani. A tulajdonság értéke a képfájl elérési útja:

KépdobozObjektumnév.KépHelye = "*elérési út*"

Az elérési utat sztringkifejezésként is megadhatjuk. Használhatunk relatív vagy abszolút elérési utat. Ne feledkezzünk meg a fájl kiterjesztéséről!

```
Kocka.KépHelye = "Kocka" & Dobás & ".png"
```

5. gyakorlat. Módosítsuk az 1. és 2. gyakorlat programját úgy, hogy megjelenítse a kockákat! A képfájlokat a tankönyv forrásfájljai között találjuk.

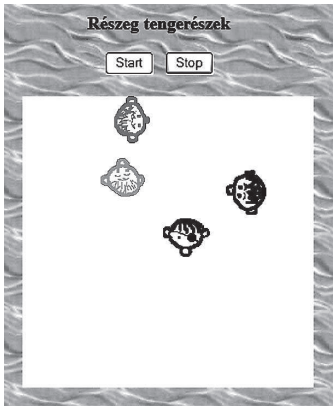


Az elérési út megadásánál ügyeljünk arra, hogy a program futása közben elérhető legyen a képfájl! A képeket célszerű a lefordított program mappájának egy almappájába helyezni, és relatív elérési utat használni.

Programozási összefoglaló: Az aktuális mappa és a felhasználó *Dokumentumok* mappájának elérési útja

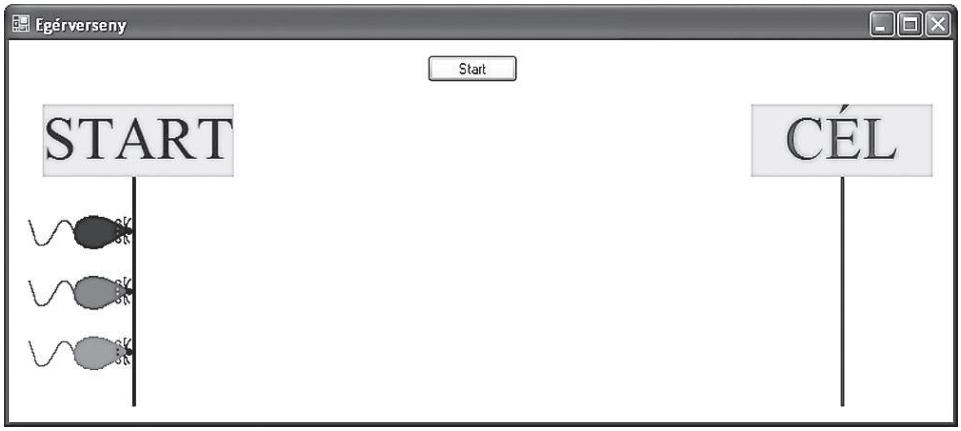
Feladatok

1. Készítsünk programot, amely a *Választ* gombra kattintáskor kiír 5 véletlenszámot az [1; 90] intervallumból! Használhatjuk-e a programot a lottósorsolás szimulálására?
2. Készítsünk programot, amely bekér egy pozitív egész számot, majd dobást szimulál egy olyan dobókockával, melynek megadott számú lapja van!
3. Írjunk programot, amely kattintásra olyan véletlenszámot választ, melynek értéke
   a) 5-tel osztható egész szám az [1; 100] intervallumból;
   b) 0,2; 0,4; 0,6 vagy 0,8;
   c) 1; 10; 100 vagy 1000;
   d) a [20; 30], [50; 60] vagy [80; 90] intervallum valamelyikébe eső egész szám;
   e) 0 vagy 1;
   f) –1 vagy +1;
4. Készítsünk programot, amely egy véletlenszerűen választott, 4 bitből álló sorozatot ír ki a képernyőre! Ha mind a 4 bit 0, akkor egy üzenetablakban jelenjen meg a „Semmi!”, ha pedig mind a 4 bit 1-es, akkor a „Szuper!” üzenet!
5. Szimuláljuk a számegyenesen egy részeg ember mozgását! Az origóból kiindulva egérkattintásra véletlenszerűen lépjen egyet balra vagy jobbra! A program írja ki minden lépés után, hogy hol helyezkedik el az emberünk! Üzenetablak jelezze, ha visszaért az origóba!
6. Módosítsuk az előző feladatot úgy, hogy az imbolygás a koordinátasíkon menjen végbe! Egy-egy lépésnél mindkét koordináta külön-külön –1-gyel, 0-val vagy +1-gyel változzon. A program írja ki a lépések után a koordinátákat, illetve az origótól mért távolságot!
7. Készítsünk programot, mellyel Taylor véletlenszerűen bolyong a képernyőn az ablak közepétől indulva! A képfájlt a forrásfájlok között találjuk. Egy címkén jelezzük folyamatosan az indulási helytől mért távolságát!
8. Készítsünk programot, mellyel Taylor és Mary véletlenszerűen bolyong a képernyőn!
9. Írjunk programot, amely négy részeg tengerészt jelenít meg egy szigeten! A szigetet egy zöld négyzet jelképezze a kék hátterű ablak közepén. A tengerészek véletlenszerűen bolyongjanak. A fürgébbek gyakran, de kicsit lépjenek, a lomhábbak ritkábban, de nagyobbat. A tengerészek ne menjenek bele a kék vízbe!



10. Készítsünk programot, amely két dobókockával dob, majd kiírja a kockák által mutatott számok összegét a képernyőre! Jelenítsük meg a dobókockákat is.
11. Írjunk programot, amely szimulálja a kockadobást! A program egy bizonyos ideig jelenítse meg egymás után egy kocka véletlenszerűen választott lapjait, majd hagyja meg az utoljára választott lapot.
12. Jelenítsünk meg a képernyőn egy kártyalapot, amely 2 másodpercenként véletlenszerűen megváltozik egy másik lapra! A kártyákat a forrásfájlok *Kártya* mappájában találjuk.

13. Írjunk játékprogramot! A játékos véletlenszerűen húz egy lapot a kártyacsomagból, majd visszateszi, és a teljes csomagból húz megint egy lapot. Értékeljük a húzott lapokat a következő pontozással!
    Mind a kétszer ugyanaz a lap: 30 pont
    Két különböző színű, de egyforma figurájú lap: 15 pont
    Két egyforma színű, de különböző figurájú lap: 5 pont
    Két különböző színű és eltérő figurájú lap: –1 pont
14. Készítsük el a következő játékprogramot! Kezdetben 100 ponttal rendelkezünk. A programnak egy szövegdobozban ajánljunk fel belőle meghatározott mennyiséget. A program ezután véletlenszerűen eldönti, hogy nyertünk vagy vesztettünk ennyi pontot. A képernyőn jelenjen meg pontjaink alakulása. Üzenetablak jelezze, ha elfogytak a pontjaink!
15. Rendezzünk egérversenyt! Jelenítsünk meg az ablakban három egeret, melyek a *Start* gombra kattintáskor elindulnak balról jobbra, véletlenszerűen változtatva a sebességüket. Az nyer, amelyik először éri el az ablak jobb szélét.



Egérverseny a képernyőn

Készítsünk programot a következő játékokhoz! A program kérje be a játékos lépését, majd lépjen véletlenszerűen, és jelenítse meg a játék állását! Üzenetablakban jelezzük a program végét és a nyertest! Melyik játéknál rendelkezünk biztosan nyerő stratégiával? Függ-e attól, hogy ki kezdi a játékot? Keressünk könyvekben vagy az interneten választ a kérdésekre!

16. Egy százszorszépnek 38 szirma van. Felváltva letépjük egy vagy két szirmát. Az nyer, akinek az utolsó szirom jut. Páratlan számú sziromból kiindulva is játsszunk!