

2. beadandó feladat: Windows Forms grafikus felületű alkalmazás

Közös követelmények:

- A programot kétrétegű (modell/nézet) architektúrában kell felépíteni, amelyben a megjelenítés rétege elkülönül a játéklógikától, amely tartalmazza az adatkezelést. A modell nem tartalmazhat semmilyen grafikus felületbeli osztályra történő hivatkozást, csak eseményeket küldhet a grafikus felületnek. A nézet nem tartalmazhat semmilyen játékbeli adatot.
- A megvalósításnak felhasználóbarátnak, és könnyen kezelhetőnek kell lennie. A szerkezetében törekednie kell az objektumorientált szemlélet megtartására.
- A program játékfelületét dinamikusán kell létrehozni futási időben. A megjelenítéshez lehet vezérlőket használni, vagy elemi grafikát. Egyes feladatoknál különböző méretű játéktábla létrehozását kell megvalósítani, ekkor ügyelni kell arra, hogy az ablakméret mindig alkalmazkodjon a játéktábla méretéhez.
- A programnak támogatnia kell új játék kezdését, és jeleznie kell, ha véget ért a játék (azután ne lehessen tovább játszani). Amennyiben lehet szüneteltetni a játékot, a szünet alatt ne teljen az idő, és ne lehessen semmilyen tevékenységet végezni a játékban.
- A dokumentációnak tartalmaznia kell a feladat elemzését, felhasználói eseteit (UML felhasználói esetek diagrammal), valamint a program szerkezetének leírását (UML osztálydiagrammal).

Feladatok:

1. Snake

Készítsük programot, amellyel a klasszikus kígyó játékot játszhatjuk.

A játékos egy, kezdetben a képernyő közepén induló, 5 hosszú kígyóval halad a képernyőn, minden másodpercben a legutoljára beállított irányba, ha a játékos nem változtatja meg azt egy megfelelő billentyű lenyomásával. A lehetséges haladási irányok balra, jobbra, felfelé, illetve lefelé. A játékmezőn véletlenszerű pozícióban mindig megjelenik egy tojás, amelyet a kígyóval meg kell etetni. Minden etetéssel eggyel nagyobb lesz a kígyó.

A játék célja, hogy a kígyó minél tovább elkerülje az ütközést a képernyő szélével, illetve saját magával.

Legyen lehetőség új játék kezdésére. A program folyamatosan jelezze az eltelt időt (másodperceket), és a játék végén kérje be a játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (leghosszabb kígyót elérő) játékosról, ahol megjeleníti a játékosok nevét, idejét és hosszát. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

2. Fény-motor párbaj

Készítsünk programot, amellyel a Tronból ismert fény-motor párbajt játszhatjuk felülnézetből. Két játékos játszik egymás ellen egy-egy olyan motorral, amely fénycsíkot húz maga mögött a képernyőn. A motor minden másodpercben a legutoljára beállított irányba

halad feltéve, hogy a játékos nem változtatja meg azt egy megfelelő billentyű lenyomásával. A lehetséges haladási irányok balra, jobbra, felfelé, illetve lefelé. Az a játékos veszít, aki előbb neki ütközik a másik játékos fénycsíkjának vagy a képernyő szélének.

Legyen lehetőség új játék kezdésére. A program folyamatosan jelezze az eltelt időt (másodperceket), és a játék végén kérje be a győztes játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (legtöbb idővel nyerő) játékosról, ahol megjeleníti a játékosok nevét és idejét. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

3. Aszteroidák

Készítsünk programot, amellyel az aszteroidák játékot játszhatjuk. A feladatunk az, hogy egy űrhajó segítségével átnavigáljuk egy aszteroidamezőn. Az űrhajóval a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg az aszteroidák, amelyek folyamatosan közelednek állandó sebességgel a képernyő alja felé. Az idő múlásával egyre több aszteroida jelenik meg egyszerre, így idővel elkerülhetetlenné válik az ütközés. A játék célja az, hogy az űrhajó minél tovább elkerülje az ütközést.

Legyen lehetőség új játék kezdésére. A program folyamatosan jelezze az eltelt időt (másodperceket), és a játék végén kérje be a játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (legtöbb idővel nyerő) játékosról, ahol megjeleníti a játékosok nevét és idejét. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

4. Gyorsulás

Készítsünk programot, amellyel az alábbi motoros játékot játszhatjuk. A feladatunk, hogy egy gyorsuló motorral minél tovább tudjunk haladni. A gyorsuláshoz a motor üzemanyagot fogyaszt, egyre többet. Adott egy kezdeti mennyiség, amelyet a játék során üzemanyagcellák felvételével tudunk növelni.

A motorral a képernyő alsó sorában tudunk balra, illetve jobbra navigálni. A képernyő felső sorában meghatározott időközönként véletlenszerű pozícióban jelennek meg üzemanyagcellák, amelyek folyamatosan közelednek a képernyő alja felé. Mivel a motor gyorsul, ezért a cellák egyre gyorsabban fognak közeledni, és mivel a motor oldalazó sebessége nem változik, idővel egyre nehezebb lesz felvenni őket, így egyszer biztosan kifogyunk üzemanyagból. A játék célja az, ez minél később következzen be.

Legyen lehetőség új játék kezdésére. A program folyamatosan jelezze az eltelt időt (másodperceket), és a játék végén kérje be a játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (legtöbb idővel nyerő) játékosról, ahol megjeleníti a játékosok nevét és idejét. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

5. Tankcsata

Készítsünk programot, amellyel a klasszikus tankcsata játékot játszhatjuk. A játékosok egy-egy tankot irányítanak a képernyő bal, illetve jobb szélén, és tudják a tankok csövének szögét, illetve a lövés erősségét állítani megfelelő billentyűzetkombinációval. A játékosok felváltva

lőhetnek, amikor is a löveg ferdehajtással az erősségnek és szögnek megfelelően célba ér. Az a játékos győz, aki előbb eltalálja a másik játékos tankját (vagyis elég közel lő hozzá).

A játékot nem csak sík terepen, hanem dombos, hegyes vidéken is lehet játszani. Ennek érdekében a programban lehessen véletlenszerű domborzatmagasságot generálni, vagy beolvasni azt alkalmas fájlból. Arra mindig ügyeljünk, hogy ugyan a két tank lehet különböző magasságban, de alattuk mindig sík legyen a terep.

Legyen lehetőség új játék kezdésére, valamint az ellenfelek számának megadásával, játék szüneteltetésére, mentésére és betöltésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt nem kell elmenteni).

6. Aknamező

Készítsünk programot a következő játékra. A játékban egy tengeralattjárót kell irányítanunk a képernyőn (balra, jobbra, fel, illetve le), amely felett ellenséges hajók köröznek, és folyamatosan aknákat dobnak a tengerbe. Az aknáknak három típusa van (könnyű, közepes, nehéz), amely meghatározza, hogy milyen gyorsan süllyednek a vízben (minél nehezebb, annál gyorsabban).

Az aknákat véletlenszerűen dobják a tengerbe, ám mivel a hajóskapitányok egyre türelmetlenebbek, egyre gyorsabban kerül egyre több akna a vízbe. A játékos célja az, hogy minél tovább elkerülje az aknákat. A játék addig tart, ameddig a tengeralattjárót el nem találta egy akna.

Legyen lehetőség új játék kezdésére, valamint játék szüneteltetésére. A program folyamatosan jelezze a játékidőt, illetve az elkerült akna számát, és a játék végén kérje be a játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (legtöbb idővel nyerő) játékosról, ahol megjeleníti a játékosok nevét és idejét. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

7. Malom

Készítsünk programot, amellyel a közismert malom játékot játszhatjuk. A malom játékban két játékos egy 24 mezőből álló speciális játéktáblán játszhatja, ahol a mezők három egymásba helyezett négyzetben helyezkednek (mindegyikben 8, a sarkoknál és a felezőpontoknál), melyek a felezőpontok mentén össze vannak kötve. Kezdetben a tábla üres, és felváltva helyezhetik el rajta bábuikat az üres mezőkre. Az elhelyezés után a játékosok felváltva mozgathatják bábuikat a szomszédos (összekötött) mezőkre. Amennyiben egy játékos nem tud mozgatni, akkor passzolhat a másik játékosnak. Ha valakinek sikerül 3 egymás melletti mezőt elfoglalnia (rakodás, vagy mozgatás közben), akkor leveheti az ellenfél egy általa megjelölt bábuját. Az a játékos veszít, akinek először megy 3 alá a bábuk száma a mozgatási fázis alatt.

Legyen lehetőség új játék kezdésére, szüneteltetésére, mentésére és betöltésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt nem kell elmenteni).

8. Bombázó

Készítsünk programot az alábbi bombázó játékra. A játékot egy $n \times n$ -es táblán játszunk, amelyre véletlenszerűen falakat, valamint ellenfeleket helyeztünk el. Kezdetben a játékos a bal alsó sarokban helyezkedik el, és négy irányba (balra, jobbra, fel, vagy le) mozoghat, de ha találkozik (egy pozíciót foglal el) valamely ellenféllel, akkor meghal.

Az ellenfelek folyamatosan bolyonganak a pályán úgy, hogy elindulnak valamilyen irányba, és ha falnak ütköznek, akkor elfordulnak egy véletlenszerűen kiválasztott másik irányba. A játékos feladata az, hogy végezzen az ellenfelekkel úgy, hogy bombákat helyez a pályán. A bombát az aktuális pozíciójára tudja lerakni, és azok rövid időn belül robbannak, megsemmisítve a 3 sugáron belül található ellenfeleket (falon át is), illetve magát a játékost is, ha nem menekül onnan.

Legyen lehetőség új játék kezdésére a táblaméret (5×5 , 10×10 , 20×20), valamint az ellenfelek számának megadásával, játék szüneteltetésére, mentésre és betöltésre. Ügyeljünk arra, hogy az ellenfelek, vagy a játékos ne falon kezdjenek.

9. Labirintus

Készítsünk programot, amellyel egy labirintusból való kijutást játszhatunk. A játékos a labirintus bal alsó sarkában kezd, és a feladata, hogy minél előbb eljusson a jobb felső sarokba úgy, hogy négy irányba (balra, jobbra, fel, vagy le) mozoghat, és elkerüli a labirintus sárkányát.

A labirintus felépítését fájlokból olvassuk be, amelyeket előre legyártottunk, és a program véletlenszerűen választ közülük. Minden labirintusban van több kijutási útvonal. A sárkány egy véletlenszerű kezdőpozícióból indulva folyamatosan bolyong a pályán úgy, hogy elindul valamilyen irányba, és ha falnak ütközik, akkor elfordul egy véletlenszerűen kiválasztott másik irányba. Ha a sárkány a játékosal szomszédos területre jut, akkor a játékos meghal.

Mivel azonban a labirintusban sötét van, a játékos mindig csak 3 sugarú körben látja a labirintus felépítését, távolabb nem.

Legyen lehetőség új játék kezdésére (legyen legalább 6 különböző labirintus, legalább háromféle méretben, pl. 8×8 , 16×16 , 24×24 mező), valamint a játék szüneteltetésére. Ügyeljünk arra, hogy a játékos, vagy a sárkány ne falon kezdjenek.

10. Menekülj

Készítsünk programot, amellyel az alábbi menekülő játékot játszhatjuk. A játékot egy $n \times n$ -es táblán játszunk. Kezdetben a játékos a képernyőn felül és középen, két üldöző pedig az alsó sarkokban helyezkedik el. A játékos és az üldözők vízszintes, illetve függőleges irányú mozgásra képesek. Az üldözők mindig a játékos felé haladnak úgy, hogy ha a függőleges távolság a nagyobb, akkor függőlegesen, ellenkező esetben vízszintesen mozognak. Az üldözőket a program irányítja, és meghatározott időközönként mozognak a játékos felé. A játékost a nyilakkal lehet a megfelelő irányba mozgatni. Az üldözők célja a játékos elfogása, ami bekövetkezik, ha valamelyikük hozzáér. A pályán elhelyezkedik még véletlenszerűen n akna. Ha a játékos vagy az üldözők egy aknához érnek, az akna felrobban és megsemmisíti azt, aki hozzáért. A játékos feladata megmenekülni, amihez az üldözőket az aknák segítségével meg kell semmisítenie.

Legyen lehetőség új játék kezdésére a táblaméret megadásával (12×12, 24×24, 36×36), játék szüneteltetésére, mentésére és betöltésére. Ügyeljünk arra, hogy az aknák egymáshoz ne érjenek, illetve kezdetben a játékoshoz és üldözőkhöz sem.

11. Lopakodó

Készítsünk programot, amellyel az alábbi lopakodós játékot játszhatjuk. A játékot egy $n \times n$ -es táblán játsszuk, amelyre véletlenszerűen falakat, valamint őrköt helyeztünk el. Kezdetben a játékos a bal alsó sarokban helyezkedik el, és négy irányba (balra, jobbra, fel, vagy le) mozoghat.

A játékos feladata, hogy eljusson a jobb felső sarokba úgy, hogy egyik őr sem fedezze őt fel. Az őrk folyamatosan bolyonganak a pályán úgy, hogy elindulnak valamilyen irányba, és ha falnak ütköznek, akkor elfordulnak egy véletlenszerűen kiválasztott másik irányba. Az őrk látják a körülöttük lévő szomszédos területeket, valamint az előttük lévő 3 területet (de falon nem tudnak átlátni).

Legyen lehetőség új játék kezdésére a táblaméret megadásával (5×5, 10×10, 20×20), valamint az ellenfelek számának megadásával, játék szüneteltetésére, mentésére és betöltésére. Ügyeljünk arra, hogy az őrk, vagy a játékos ne falon kezdjenek.

12. Harcos robotmalacok csatája

Készítsünk programot, amellyel a következő kétszemélyes játékot tudjuk játszani. A játék egy $n \times n$ -es táblán játszható, ahol két harcos robotmalac helyezkedik el, kezdetben a két ellentétes oldalon, a középvonaltól eggyel jobbra, és mindkettő előre néz. A malacok lézerágyúval és egy támadóököllel vannak felszerelve.

A játék körökből áll, minden körben a játékosok egy programot futtathatnak a malacokon, amely öt utasításból állhat (csak ennyi fér a malac memóriájába). A két játékos először leírja a programot (úgy, hogy azt a másik játékos ne lássa), majd egyszerre futtatják le őket, azaz a robotok szimultán teszik meg a programjuk által előírt 5 lépést.

A program az alábbi utasításokat tartalmazhatja:

- előre, hátra, balra, jobbra: egy mezőnyi lépés a megadott irányba, közben a robot iránya nem változik.
- fordulás balra, jobbra: a robot nem vált mezőt, de a megadott irányba fordul.
- tűz: támadás előre a lézerágyúval.
- ütés: támadás a támadóököllel.

Amennyiben a robot olyan mezőre akar lépni, ahol a másik robot helyezkedik, akkor nem léphet (átugorja az utasítást), amennyiben a két robot ugyanoda akar lépni, akkor egyikük se lép (mindkettő átugorja az utasítást).

A két malac a lézerrel és az ököllel támadhatja egymást. A lézer előre lő, és függetlenül a távolságtól eltalálja a másikat. Az ütés pedig valamennyi szomszédos mezőn (8) eltalálja a másikat. A csatának akkor van vége, ha egy robotot háromszor eltaláltak.

Legyen lehetőség új játék kezdésére a táblaméret megadásával (4×4, 8×8, 16×16), valamint játék betöltésére és mentésére. A program folyamatosan jelezze a játékosok sérülésszámát.

13. Gyártósor

Készítsünk programot, amellyel a következő kétszemélyes játékot tudjuk játszani. A játék egy $2n$ -es táblán (a gyártósoron) játszható, ahol mindkét játékos n mező birtokában van (a mezők fehérek és feketék). Ezek a mezők véletlenszerűen helyezkednek el a táblán. A gyártósoron termékek haladnak át, összesen $3n$ termék, amelyek 3 színben lehetnek (kék, piros, zöld, mindegyikből n darab).

A játék körökre osztott, ahol is minden körben felkerül egy új, véletlenszerű termék a gyártósor elejére. A két játékos megad egy számot 1 és 3 között (úgy, hogy azt a másik játékos ne lássa). Amennyiben a két szám különbözik, úgy a gyártósorok a termékek a kettő összegének megfelelő mezőt haladnak előre (vagy lekerülnek a gyártósorról, ha túllépik a végét). Amennyiben a két szám megegyezik, a gyártósor leáll, ekkor az összes termék az aktuális helyén lekerül a gyártósorról, és a játékosok pontot szereznek annak függvényében, milyen termékek voltak a mezőiken (a piros 3, a kék 2, a zöld 1 pontot ér).

A játék addig tart, amíg valamennyi termék le nem került, vagy át nem futott a gyártósoron. Legyen lehetőség új játék kezdésére a táblaméret megadásával (8, 12, 16), valamint játék betöltésére és mentésére. A program folyamatosan jelezze a játékosok pontszámát.

14. Kiszúrós amőba

Készítsünk programot, amellyel a közismert amőba játék alábbi változatát játszhatjuk. Adott egy $n \times n$ -es tábla, amelyen a két játékos felváltva x, illetve o jeleket helyez el. Csak olyan mezőre tehetünk jelet, amely még üres. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 5 egymással szomszédos jelet vízszintesen, függőlegesen vagy átlós irányban. A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjére kattintva helyezhessük el a megfelelő jelet. Természetesen csak a szabályos lépéseket engedje meg a program, és ismerje fel, ha a játék véget ért. Ilyenkor jelezze ki a győzelmet jelentő 5 jelet (ha több lehetőség van, akkor egy tetszőlegeset), illetve írja ki, hogy döntetlen, ha betelt a tábla.

A kiszúrás a játékban az, hogy ha egy játékos eléri a 3 egymással szomszédos jelet, akkor a program automatikusan törli egy jelét egy véletlenszerűen kiválasztott pozícióról (nem biztos, hogy a hármastól), ha 4 egymással szomszédos jelet ér el, akkor pedig kettőt.

Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), valamint játék betöltésére és mentésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentjük el és töltjük be).

15. Potyogós amőba

Készítsünk programot, amellyel a potyogós amőba játékot lehet játszani, vagyis az amőba azon változatát, ahol a jeleket felülről lefelé lehet beejteni a játékmezőre. A játékmező itt is $n \times n$ -es tábla, és ugyanúgy X, illetve O jeleket potyogtathatunk a mezőre. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 4 egymás melletti jelet (vízszintesen, vagy átlósan).

A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjére kattintva helyezhessük el a megfelelő jelet. Természetesen csak a szabályos

lépéseket engedje meg a program, mindig jelezze a következő játékos, és ismerje fel, ha a játék véget ért. Ilyenkor jelezze ki a győzelmet jelentő 4 jelet (ha több lehetőség van, akkor egy tetszőlegesen), illetve írja ki, hogy döntetlen, ha betelt a tábla (játék vége után ne lehessen tovább játszani).

Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), valamint játék betöltésére és mentésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentjük el és töltjük be).

16. Aknakereső

Készítsünk programot, amellyel az aknakereső játékot játszhatjuk. A játékosok feladata egy $n \times n$ -es játéktábla felderítése az aknák kihagyása mellett úgy, hogy egymás után felfedi a mezőket.

Az aknákat véletlenszerűen helyezzük el a játéktáblán. Amennyiben a felfedezett mező nem akna, akkor megjelenik rajta a szomszédos 8 mezőben található aknák száma. Amennyiben ez nulla, akkor nem csak a játékmező kerül felfedezésre, hanem az a teljes terület, amelyet nem nulla értékű mezők határolnak (ekkor ezek a mezők is felfedésre kerülnek). A játék addig tart, amíg a játékos aknára nem lép, vagy nem sikerül az összes nem akna mezőt felfednie.

Legyen lehetőség új játék kezdésére a táblaméret (10×10, 20×20, 30×30), valamint az aknák számának (10, 30, 50) megadásával, valamint játék szüneteltetésére. A program folyamatosan jelezze az eltelt időt (másodpercek), és a játék végén kérje be a játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (leggyorsabb) játékosról, ahol megjeleníti a játékosok nevét, idejét táblamérettel és aknaszámmal. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

17. Go

Készítsünk programot, amellyel a Go játék egyszerűsített változatát játszhatjuk. A Go játékban a játéktábla egy $n \times n$ -es pontokból álló felület, ahol a pontokra a játékosok felváltva köveket helyezhetnek (tradicionálisan fehér, illetve fekete színűt). A lerakott köveknek élete van, ami a négy szomszéd mezőből szabad mezők száma. Egy csoport olyan kövek halmaza, amelyek szomszédosan összeérnek. A játékos akkor keríti be a másik játékos egy csoportját, ha azok élete elfogy. Ekkor a kövek fogságba kerülnek, és levehetőek a tábláról (ekkor a terület újra üres lesz, és lehet oda követ helyezni).

A játék célja minél több fogoly ejtése, amennyiben ez egyenlő, a játék döntetlen. A játék tartson meghatározott körszámig.

Legyen lehetőség új játék kezdésére a táblaméret megadásával (9×9, 13×13, 19×19), játék szüneteltetésére, mentésére és betöltésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentjük el és töltjük be).

18. Négyzetek

Készítsünk programot, amellyel az alábbi négyzetek játékot játszhatjuk. A négyzetek játékban a játéktábla egy $n \times n$ -es pontokból álló felület, amelyen a játékosok két pont között vonalakat

húzhatnak (vízszintesen, vagy függőlegesen) felváltva két olyan pont között, ahol eddig még nem húztak vonalat. A játék célja, hogy a játékosok a húzogatással négyzetet tudjanak rajzolni (azaz ők húzzák be a negyedik vonalat, független attól, hogy az eddigieket melyikük húzta). Ilyen módon egyszerre két négyzet is elkészülhet. Ha egy játékos berajzolt egy négyzetet, akkor ismét ő következik. A játék addig tart, amíg lehet vonalat húzni a táblán, és az nyer, akinek több négyzete lesz a végén.

Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), játék szüneteltetésére, mentésére és betöltésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt is mentsük el és töltsük be).

19. Reversi

Készítsünk programot, amellyel az alábbi Reversi játékot játszhatjuk. A játékot két játékos játssza $n \times n$ -es négyzetrácsos táblán fekete és fehér korongokkal. Kezdekor a tábla közepén X alakban két-két korong van elhelyezve mindkét színből. A játékosok felváltva tesznek le újabb korongokat. A játék lényege, hogy a lépés befejezéseként az ellenfél ollóba fogott, azaz két oldalról (vízszintesen, függőlegesen vagy átlósan) közrezárt bábuít (egy lépésben akár több irányban is) a saját színünkre cseréljük.

Mindkét játékosnak, minden lépésben ütnie kell. Ha egy állásban nincs olyan lépés, amivel a játékos ollóba tudna fogni legalább egy ellenséges korongot, passzolnia kell és újra ellenfele lép. A játékosok célja, hogy a játék végére minél több saját színű korongjuk legyen a táblán.

A játék akkor ér véget, ha a tábla megtelik, vagy ha mindkét játékos passzol. A játék győztese az a játékos, akinek a játék végén több korongja van a táblán. A játék döntetlen, ha mindkét játékosnak ugyanannyi korongja van a játék végén.

Legyen lehetőség új játék kezdésére a táblaméret megadásával (10×10, 20×20, 30×30), játék szüneteltetésére, mentésére és betöltésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépésig tart, ezt is mentsük el és töltsük be).