HÁZI FELADAT

Programozás alapjai 2.

Végleges dokumentáció

Barczikay Dániel

2024. május 12.

Tartalom

1. Kő-papír-olló játék 2
2. Program működése, specifikációja 2
3. Tervezés 3
   3.1. Fontosabb algoritmusok 3
      3.1.1. Játékos osztály 3
      3.1.2. Játékosok osztály 4
      3.1.3. Játék osztály 4
      3.1.4. Játéknapló osztály 4
   3.2. Osztálydiagramm 4
4. Megvalósítás 6
5. Tesztelés 8

1. Kő-papír-olló játék

Tervezzen objektummodellt a kő-papír-olló játék modellezéséhez! Célunk, hogy a különböző stratégiával játszó játékosokat összesorsolva megállapítsuk a legjobb stratégiát, ha van ilyen. A modellben legyenek "Játékos" objektumok, melyek egy "Napló" objektum felügyeletével játszanak. Ez utóbbi gyűjti a statisztikát.

Demonstrálja a működést külön modulként fordított tesztprogrammal! A játék állását nem kell grafikusan megjeleníteni, elegendő csak karakteresen, a legegyszerűbb formában! A megoldáshoz ne használjon STL tárolót!

2. Program működése, specifikációja

A megvalósítandó program ezen játékot szimulálja, de természetesen bármennyi kört lehet játszani és bármennyi játékos beszállhat a játékba, illetve játékosnak a számítógépet is lehet választani. A gépi játékosok között több avatar játékos figura is elérhető lesz, akik különböző stratégiával játszanak.

Egy játékban a két versenyzőt és humán játékos esetén egy-egy tétet lehet megadni. A tétet karakterek rövidítik K, mint kő, P, mint papír, O, mint olló. Emberi játékos esetén ezen adatok bevitele a képernyőn keresztül (karakteres módban) történik. Amennyiben gépi játékost választunk, akkor az aktuális tétet az avatar fogja megtenni.

A gépi avatarok előre programozott stratégia mentén játszanak. A program felépítése az OO szemlélettel támogatni fogja új típusú avatarok bővítésének lehetőségét.

Az előre megvalósított avatarok stratégiája:

- véletlen választó: A gépi játékos mindig véletlenszerűen választ tétet, semmi korábbi eredményt nem vesz figyelembe.
- előző kör ellen fogadó: A gépi avatar figyelembe veszi az előző kör eredményét. Feltételezi, hogy az ellenfele, a győztes tétjét fogja ismételni, ezért az ellen fogad
- előző kör ismétlő: A gépi avatar figyelembe veszi az előző kör eredményét. Meg akarja ismételni a győztes tétet
- statisztikai győztes ellen fogadó: A gépi avatar figyelembe veszi az előző összes játék eredményét. A statisztikailag legtöbbször nyert tét ellen fogad.
- statisztikai győztest ismétlő: A gépi avatar figyelembe veszi az előző összes játék eredményét. A statisztikailag legtöbbször nyert tétre fogad.

A program lehetőséget biztosít a játékállások fájlba történő kiírására és visszaolvasására. Ezáltal a statisztikai számítást figyelembe vevő avatarokat folyamatosan tanítani lehet.

A bevitel után a program kiértékeli az adott játék eredményét és karakteres képernyőn keresztül értesít a győztes nevéről.

A program a humán játékosok neveit egyedi azonosítónak fogja tekinteni, azaz ugyanazon névvel játszó játékosok eredményeit az adott játékos statisztikájába fogja számolni. Amennyiben az adott játékos neve még nem szerepel a listában, akkor új játékosként fogja eltárolni. Minden játékos (humán és gépi) kezdeti statisztikája 0 darab játék.

A játék során lekérhetjük az eddigi játszmák statisztikáját a játéknaplóból. Ez a statisztika megmutatja, hogy az egyes játékosok hány kört játszottak, ebből mennyi volt számára nyert, vesztett vagy döntetlen kör.

A felület menüpontjai lehetőséget biztosítanak:

- egy kör lejátszására. Azaz gépi avatar kiválasztására, vagy humán játékos esetén a név megadására. Humán játékos esetén a tét bevitelére.
- humán és gépi játékosok statisztikájának listázása
- kilépés a programból

3. Tervezés

A program szöveges üzemmódra készül, egyszerű szöveg kiíratást és adatbevitelt fog használni.Az input mezők a legalapvetőbb beviteli ellenőrzést fogják tartalmazni.

3.1. Fontosabb algoritmusok

3.1.1. Játékos osztály

Tartalmazza a játékosokra jellemző attribútumokat, mint név, nyert játékok száma, vesztett játékok száma, döntetlen játékok száma

Leglényegesebb eleme a Tipp függvény, mely minden származtatott osztály esetén más algoritmust tartalmaz.

A Humán játékos esetén a stratégiát nem tartalmaz az osztály, itt az input mezőben bekért tippet fogja megjátszani a gép.

A többi avatar, gépi játékos esetén a meghatározott algoritmusok szerint fogja a rendszer számolni a következő tippet.

Célszerű a játék során a játékosok eredményét számon tartani, az adott játékosnak hány nyertes, vesztes és döntetlen játéka volt. Így ezen statisztikák könnyen kigyűjthetőek lesznek az értékelés során.

Minden játékmenet végén az érintett két játékos statisztikai adatai frissülnek.

A játékosok a könnyebb kezelés érdekében láncolt listába vannak szervezve.

3.1.2. Játékosok osztály

A játékosok láncolt listájának kezelését szolgálja, tartalmazza az első mutatót, valamint függvényt a játékosok kiiratására.

3.1.3. Játék osztály

A játék osztály egy adott játékmenetet foglal magába. Tartalmazza a két érintett játékost, a tippeket, valamint a kiértékelő függvényt, ami visszaadja a győztes játékost.

A Kő-Papír-Olló játék szabálya viszonylag egyszerű, egy összetett IF szerkezetben ki lehet értékelni a megadott két tipp közül, hogy melyik a nyertes. Azonos tipp esetén a játék döntetlen.

A játékok a könnyebb kezelhetőség érdekében láncolt listába vannak fűzve.

3.1.4. Játéknapló osztály

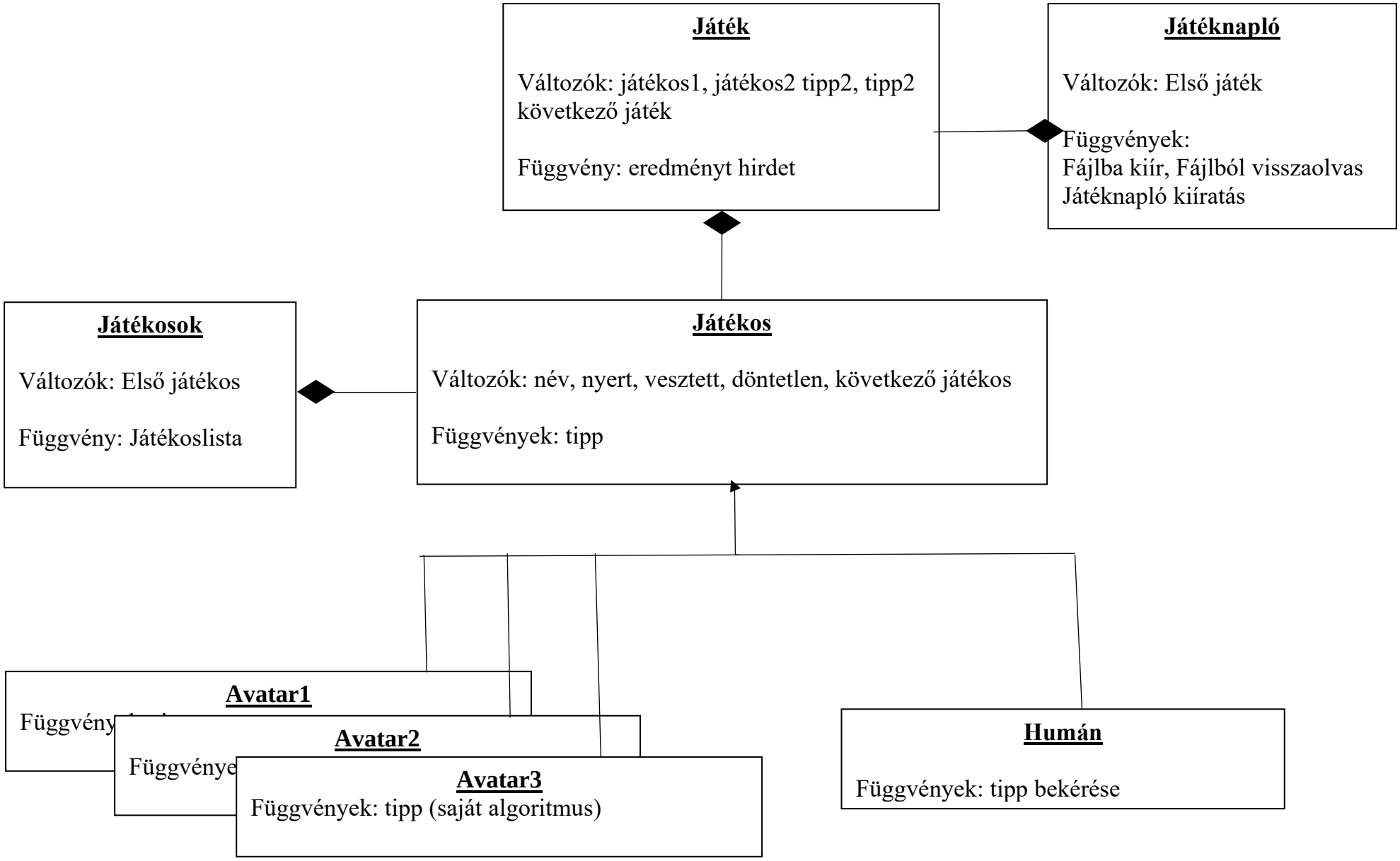
A játéknapló tartalmazza az első játékra mutató pointert.

Hogy a különböző stratégiákhoz megfelelő számú játékeset rendelkezésre álljon, a játékok eredményeit fájlba mentjük. A mentés automatikusan megtörténik a játék végén, valamint a program elején beolvasásra kerül fájlból az elérhető adatok köre.
Az adatok mentése az egyes játék körök adatainak tárolását foglalja magába, a játékosok adatait, valamint az adott tippeket.

A statisztikák a beolvasás során újra előállíthatóak.

A játéknapló függvénye tartalmazza az eddigi játéknaplók listázását is.

3.2. Osztálydiagramm


Játék
Változók: játékos1, játékos2 tipp2, tipp2
következő játék
Függvény: eredményt hirdet
Játéknapló
Változók: Első játék
Függvények:
Fájlba kiír, Fájlból visszaolvas
Játéknapló kiíratás
Játékosok
Változók: Első játékos
Függvény: Játékoslista
Játékos
Változók: név, nyert, vesztett, döntetlen, következő játékos
Függvények: tipp
Avatar1
Függvény
Avatar2
Függvénye
Avatar3
Függvények: tipp (saját algoritmus)
Humán
Függvények: tipp bekérése

4. Megvalósítás
4.1. Főbb osztályok
4.1.1. „jatek”

A „jatek” osztály foglalja magába egy darab Kő-Papír-Olló játék elemeit.
Minden játékhoz tartozik két játékos és az általuk megtett tippek. A tippelés után a kiertekel függvény kiértékeli az eredményt. Későbbi felhasználás céljából külön eltárolja a nyertes játékost és az általa tett tippet. A lejátszott játékokat láncolt listában fűzzük fel, ezért minden osztály tartalmaz egy mutatót, a következő játékra.

Metódusai között szerep az említett kiértékelés, valamint egy „kiirat” függvény, ami a standard outputra add vissza információt, valamint egy „serialize” függvény, ami a fájlba mentést támogatja.

4.1.2. „naplo”
Az objektum felel azért, hogy lejátszott játékokat egy listába, naplóba szervezze. A napló tárolni fogja a napló első játék elemét.

Metódusai között szerepel az új játék felvitele, a játéklista kiíratás szöveges formába. Ez az objektum indítja az adatok fájlba történő írását, valamint a kiírt bináris fájlok visszaolvasását.

Destruktor függvénye gondoskodik arról, hogy a játékoknak foglalt dinamikus memória a végén felszabadításra kerüljön.

4.1.3. „jatekos”
A játék résztvevőinek adatát a „jatekos” osztály tartalmazza.
A játékos természetes tulajdonsága, a neve. Ehhez tartozik egy egy karakteres egyedi azonosító, a monogramm. Ez a játék szöveges felületének kezelhetőségét támogatja, ezzel tudjuk a játékosokat kiválasztani.
A játékosoknak két csoportját különböztetjük meg az emberi és a gépi játékosok. Gépi játékos esetén a gép maga ad tippet, vagy véletlenszám vagy az előző játékok eredményeinek alapján. Emberi játékos esetén standard inputról kérjük be a saját tippjét.
Az osztály tárolni fog még önmagáról statisztikai jellegű adatokat, nevezetesen, hogy az adott játékosnak hány nyert, vesztett és döntetlen játéka volt és gondoskodik ezek növeléséről a játék során.
A játékosok láncolt listában vannak felfűzve.
Ennek az osztálynak létezik egy tisztán virtuális tagfüggvénye, a „tippel”, ami az egyes játékstratégiákat fogja megvalósítani. A gépi vagy humán osztályok ebből fognak származni és a valódi játékban már ők fognak részt venni.

4.1.4. „jatekosok”

A játékosokat láncolt listába szervezzük. Az osztály függvényei gondoskodnak, hogy a játékosok kereshetőek legyenek, vagy új emberi játékossal lehessen bővíteni a listát. A destruktor függvénye gondoskodik, hogy a játék végén az egyedek által foglalt memória felszabadításra kerüljön.

4.1.5. „statisztika”

A játék menetéről folyamatosan frissülő statisztikákat tárolunk. Ezek az a gépi játékosok tippelő algoritmusához szolgáltatnak adatok. Ilyen statisztikai adat, hogy hányszor nyert a játékok során a kő, a papír vagy az olló tipp, vagy mi volt az utolsó nyertes tipp.

4.2. Főbb algoritmusok

4.2.1. Öröklés

A játék egyik fő OO jellegét adja a játékosok kezelése. Létezik a „jatekos” ősosztály, s ebből fognak leszármazni a különböző stratégiát megvalósító tényleges játékosok.

A programban négy gépi játékos került megvalósításra, különböző algoritmusra specializálva, ezeket az „avatar” leszármazott osztályok valósítják meg. Minden osztályban az említett „tippel” függvény kerül felüldefiniálásra.
Az emberi játékosok esetén a tipp függvény a billentyűzetről kéri be a tippünket.

4.2.2. Adatok mentése fájba

Az OO szemléletet jól mutatja az információk fájlba írása is.
A játék felépítése alapján a kiíratásokat a „naplo” osztály fogja tartalmazni, hisz ő fogja össze a legfelső szinten a játék struktúráját.
Az érdemi adatokat viszont a „jatek” illetve a „jatekos” osztályok tartalmazzák, így az adatok mentése ezek azonos nevű függvényeken keresztül kerülnek láncoltan meghívásra. Minden objektum a saját adatának kiíratásáról gondoskodik.

4.2.3. Operátor felüldefiniálása

A naplo és a statisztika osztályok a képernyőre történő kiírást a „<<” operátor felüldefiniálásával valósítják meg.

5. Tesztelés

5.1. Memória szivárgás tesztelése

A memória szivárgás ellenőrzését a laborgyakorlatokon használt MEMTRACE modullal végeztem. Az osztályok include definíciójában szerepel a ”memtrace.h” állományt a standard fejlécállományok után.
A sikerességet JPorta folyamat is ellenőrizte, problémát nem detektált.

5.2. Funkcionális tesztek

A főprogram CPORTA definíciós ága 4 játékmenetet valósít meg. Ennek egyik ága az emberi játékot is szimulálja, azaz beolvasást tartalmaz a standard inputról.

5.3. Interfész tesztek

A főprogram CPORTA definíciós ága interfésztesztеket is megvalósít. Ezekkel tudom vizsgálni, hogy pl a játékosok felvitele, keresése, vagy a felüldefiniált tippelés függvény megfelelően működik-e a háttérben.
Az interfész teszteket a „gtest_lite.h” támogatja a beépített EXPECT esetek segítségével.

5.4. Lefedettségi tesztek

A JPorta felületen keresztül megtörtént a kódlefedettségi teszt is a beépített funkcionális tesztek segítségével.
A teljes lefedettséget nehezíti, hogy egy olyan játékról van szó, amiben véletlenszám generálás is van, valamint játék statisztika alapján is történik döntés. Így a kiértékelés viszonylag nagy számú lehetőséget generál.
A lefedettségi teszteket két ágon lehet növelni.

- Programozott tesztesetek
  A főprogram CPORTA definíciós ága 4 játékmenetet valósít meg. Ennek egyik ága az emberi játékot is szimulálja, azaz beolvasást tartalmaz a standard inputról. A programozott tesztesetek továbbá minden gépi játékost is megmozgatják, így törekszünk a játékban előforduló lehetőségek minél bővebb lefedésére.
- A program beépítetten tartalmazza a korábbi játéknaplók mentését és visszaolvasását. Mivel a visszaolvasás során is a rendszer alapvetően leszimulálja a játékot, a játék előrehaladtával a lefedettség fokozatosan nő.