HÁZI FELADAT

Programozás alapjai II.

Sportegyesület - Végleges

Keresztes Csenge
B8XWST

2020. május 15.

TARTALOM

1. Feladat

Sportegyesület

A Fitt Sportegyesület nyilvántartást szeretne vezetni a csapatairól. Minden csapat rendelkezik egy névvel és egy alaplétszámmal. A sportegyesület háromféle sportággal foglalkozik:

1. labdarúgás
2. kosárlabda
3. kézilabda

A labdarúgó csapatnak két edzője van; a kosárlabda csapatnak elengedhetetlen kellékei a pom-pom lányok aminek létszámát is nyilvántartják; a kézilabda csapatok pedig évente kapnak valamekkora összegű támogatást. A nyilvántartás rendelkezzen minimum az alábbi funkciókkal:

- új csapat felvétele
- csapat törlése
- listázás.

Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz ne használjon STL tárolót!

2. Pontosított feladatspecifikáció

A feladatot objektum orientált módon kell megvalósítani. Egy olyan tárolót kell készíteni, amelyben különböző típusú, különböző adatokkal rendelkező objektumokat tárolunk, majd ebben végzünk műveleteket. Ezt heterogén kollekcióval valósítottam meg.

A feladatkiírás három fő funkciót kér: új csapat felvételét, csapat törlését illetve a csapatok kilástázását. Ezek mellett megvalósítottam a létrehozott adatbázis fájlba mentését, illetve már meglévő fájlból való beolvasást és a listába fűzést.

A teszteléshez egy olyan programot hoztam létre, amely a fent említett 5 funkciót teszteli, illetve magát a programot. A 6. teszteset az, amiben a felhasználó kipróbálhatja a menüvezérelt program működését. A tesztesetek standard inputról beolvasott adatok alapján működnek.

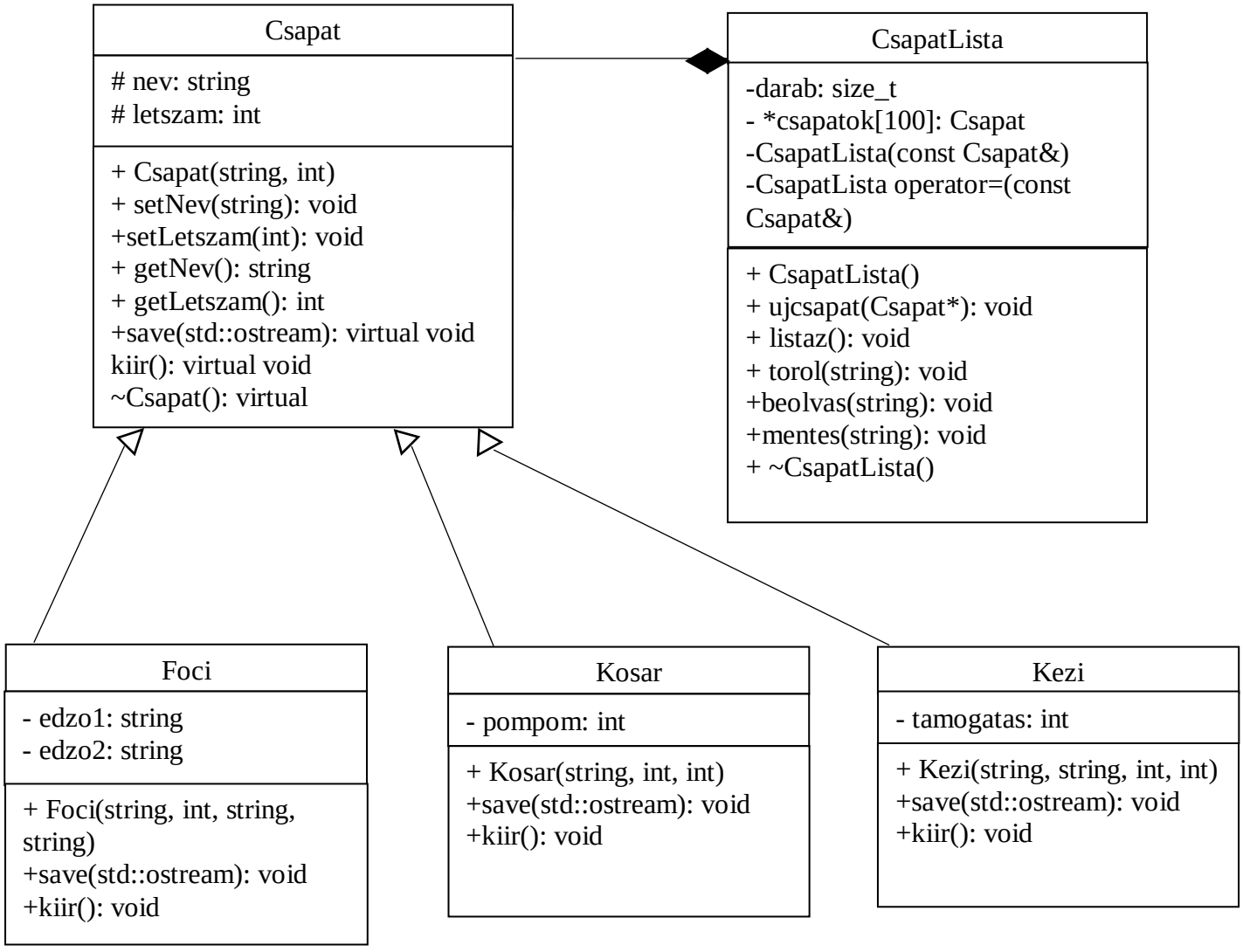
3. Terv

A feladat elkészítéséhez az osztályokat, azok kapcsolatait, az algoritmusokat, illetve a tesztprogramot kell megtervezni.

3.1. Osztályok

A feladat megvalósításához 5 osztályt fogok létrehozni:

- csapat : ez lesz az ősosztály, ami a 2 alapadatot tartalmazza: a csapat nevét illetve a létszámát
- foci: ez lesz az ősosztály egyik leszármazottja, ami a focicsapat két edzőjének nevét tartalmazza pluszba
- kezi: ez lesz a másik leszármazott, ami a kézilabdacsapatnak adott támogatás összegét tartalmazza pluszba
- kosar: ez is leszármazott, a pompom csapat létszámát tartalmazza még
- CsapatLista: a lista, amiben tároljuk a csapatokat és az adataikat, maga a tároló, a csapat osztállyal szülő-gyerek viszonyban áll. A csapatokat egy max. 100 elemet tároló dinamikus tömbben fogom tárolni



3.2. Algoritmusok

A program 5 fő összetett algoritmust fog tartalmazni:

- új csapat létrehozására képes függvényt
- csapat törlésére képes függvényt
- csapatok kilistázására képes függvényt
- txt fájlból beolvasó, majd azt a listába fűző függvényt
- a már elkészített adatbázist txt fájlba mentő függvényt

3.3. Tesztprogram

A tesztprogram 6 tesztesetet fog tartalmazni. A program indításakor a felhasználó a standard inputon megadhatja a futtatni kívánt teszteset számát, majd a konzolon megjelenő instrukcióknak megfelelően a tesztelés végigviteléhez szükséges további adatokat.

4. Megvalósítás

A feladatot a terv szerint valósítottam meg. Mindenhol használtam az *std::string* osztályt. A Csapat osztályt a class.h header fájlban valósítottam meg.

class Csapat{}

- ősosztály

Privát adattagok:

- string nev
- int letszam

Publikus tagfüggvényei:

- *Csapat(string nev, int letszam):nev(nev), letszam(letszam) {}*
  ez a konstruktor, inicializáló listát használok
- *void setNev(string nv)*
  setter, ami beállítja a csapatnevet
- *void setLetszam(int letsz)*
  setter, ami beállítja a létszámot
- *string getNev() const*
  getter, visszaadja az osztály privát adattagját, a nevet
- *int getLetszam() const*
  getter, visszaadja az osztály privát adattagját, a létszámot
- *virtual void save(std::ostream& os) const = 0*
  tisztán virtuális mentés függvény
- *virtual void kiir()=0*
  tisztán virtuális kiíró függvény
- *virtual ~Csapat() {}*
  virtuális destrukror

A Foci osztályt a foci.h fejlécfájlban valósítottam meg.

class Foci: public Csapat{}

- Csapat ősosztályból leszármazott alosztály

Privát adattagok:

- *string edzo1*
- *string edzo2*

Publikus tagfüggvényei:

- *Foci(string n, int l, string edzo1, string edzo2):Csapat(n, l), edzo1(edzo1), edzo2(edzo2) {}*
  konstruktor, inicializáló lista
- *void save(std::ostream& os) const*
  txt fájlba írja a focicsapatok adatait tabulátorral elválasztva
- *void kiir()*
  szabványos kimenetre írja egy focicsapat adatait szóközökkel elválasztva

A Kezi osztályt a kezi.h fejlécfájlban valósítottam meg.

class Kezi: public Csapat{}

- Csapat osztályból leszármazott alosztály

Privát adattagok:

- *int tamogatas*

Publikus tagfüggvényei:

- *Kezi(string n, int l, int tamogatas):Csapat(n, l), tamogatas(tamogatas){}*
  konstruktor, inicializáló lista
- *void save(std::ostream& os) const*
  txt fájlba írja a focicsapatok adatait tabulátorokkal elválasztva
- *void kiir()*
  szabványos kimenetre(konzolra) kiírja egy focicsapat adatait szóközökkel elválasztva

A Kosar osztályt a kosar.h fejlécfájlban valósítottam meg.

class Kosar: public Csapat{}

- Csapat ősosztályból leszármazott alosztály

Privát adattagok:

- *int pompom*

Publikus tagfüggvényei:

- *Kezi(string n, int l, int pompom):Csapat(n, l), pompom(pompom){}*
  konstruktor, inicializáló lista
- *void save(std::ostream& os) const*
  txt fájlba írja a focicsapatok adatait tabulátorokkal elválasztva
- *void kiir()*
  szabványos kimenetre(konzolra) kiírja egy focicsapat adatait szóközökkel elválasztva

A CsapatLista osztályt a lista.h fejlécfájlban valósítottam meg. Ez az osztály a Csapat osztállyal szülő-gyerek kapcsolatban áll.

class CsapatLista{}

- maga a tároló, a lista, a heterogén kollekciót „kezelő” osztály

Privát adattagok:

- size_t darab
  a listában szereplő csapatok száma, ez változik, de alapértelmezetten 0
- Csapat *csapatok[100]
  maximum 100 tagú dinamikus tömb, pointereket használunk heterogén kollekcióba
- CsapatLista(const Csapat&)
  másoló konstruktor, privát, hogy ne lehessen hozzáférni
- CsapatLista operator=(const Csapat&)
  értékadó operátor, privát, hogy ne lehessen hozzáférni

Publikus tagfüggvényei:

- *CsapatLista():darab(0) {}*
  konstruktor
- *void ujcsapat(Csapat *uj)*
  pointert kap paraméternek, ha a darab még nem érte el a maximálisan megengedettet(100-at), akkor hozzáfűzi a lista végére a csapatot az adataival együtt
- *void listaz()*
  szabványos kimenetre(konzolra) kiírja a virtuális kiir() függvény segítségével az adatbázisban szereplő összes csapatot az adataikkal együtt
- *void torol(string mit)*
  paraméterként a kitörlendő csapat nevét kapja meg, végigmegy a listán, ahol a név egyezik, azt kifűzi a listából, a darabszámot pedig csökkenti 1-gyel
- *void beolvas(string fajlnev)*
  megnyitja a paraméterként megadott nevű fájlt, majd beolvassa, illetve a listába fűzi a csapatokat és a hozzájuk tartozó adatokat. Mivel minden próbafájl első adata „foci” vagy „kosar” vagy „kezi” (tehát a csapat típusa), így azt olvassa be először, majd az alapján eldönti, hogy milyen típusú osztályként tárolja el az adott csapatot
- *void mentes(string fajlnev)*
  a virtuális save() függvény segítségével a paraméterként megadott nevű fájlként menti el az adatbázist
- ~CsapatLista()
  destruktor, felszabadítja a lista minden tagját

5. Tesztprogram bemutatása

A tesztprogramot a main.cpp-ben valósítottam meg. Ahogy elindul a program, a szabványos kimeneten megjelenik a 6 választási lehetőség, ezek közül lehet választani. A program a standard inputról beolvasott egész szám alapján hívja meg az egyes teszteseteket. A tesztprogram a követelményeknek megfelelően nem csak a szabványos inputról olvas be, hanem előre elkészített txt fájlból is.

Tesztesetek:

- *void test1()*

  1.tesztesetet megvalósító függvény: az új csapat létrehozása funkciót teszteli. Megjelenik a 3 választási lehetőség, hogy milyen fajta csapatot szeretne a felhasználó hozzáadni a listához. A megfelelő szám megadásával választhatja ezt ki, majd pedig adhatja meg az egyes adatokat. Azt, hogy az adott csapat valóban létrejött, úgy szemlélteti, hogy kilistázza a csapatokat (tehát azt az egyet amit megadott a felhasználó kiírja a szabványos kimenetre).

- *void test2()*

  2.tesztesetet megvalósító függvény: a listában szereplő csapatok kilistázó funkcióját mutatja be. Létre van hozva eleve 3 csapat, azokat fogja az adataikkal együtt kiírni a szabványos kimenetre.

- *void test3()*

  3.tesztesetet megvalósító függvény: a meglévő adatbázisból való csapattörlés funkciót teszteli le. Kiírja először a már meglévő csapatokat (kilistázza), majd ezek közül kell választania a felhasználónak egyet (megint a megfelelő számot kell megadnia), amelyiket törölni szeretné. Ezután újra kilistázódik a már módosult adatbázis, ezzel is szemléltetve, hogy a kiválasztott törlendő tag már nincs ott, törölve lett.

- *void test4()*

  4.tesztesetet megvalósító függvény: a fájlba mentés funkciót teszteli. Létrehozza az adatbázist egy csapattal, majd a felhasználótól kér egy nevet (fajl.txt formában), ahogyan azt el akarja menteni. A megfelelő mappában ellenőrizhető a létrejött (vagy esetleg módosult) fájl.

- *void test5()*

  5.tesztesetet megvalósító függvény: fájlból beolvasás funkciót tesztelő függvény. Kéri a felhasználótól a beolvasandó fájl nevét, beolvassa annak tartalmát, majd pedig kilistázza azt, ezzel mutatva, hogy valóban befűzte az adatokat a listába.

- *void test6()*

  6.tesztesetet megvalósító függvény: a programot kipróbáló teszteset. Menüvezérelt program, a felhasználó egész számokkal választhat a megjelent menüpontok között, az alapvető funkciókat próbálhatja ki itt is.

5.1. Memóriakezelés tesztje

A memóriakezelés tesztelését a memtrace.h segítségével végeztem, ezt minden .cpp fájlba include-oltam. Nem jelzett semmilyen memóriaszivárgási problémát a futtatások során, ahogy a JPorta sem.