Házi feladat dokumentáció

Programozás alapjai 2.

Szabó Bence – LV08C9

Választott feladat

A nagyházi feladatötletek listájáról a sportegyesületet választottam.

Feladat leírása

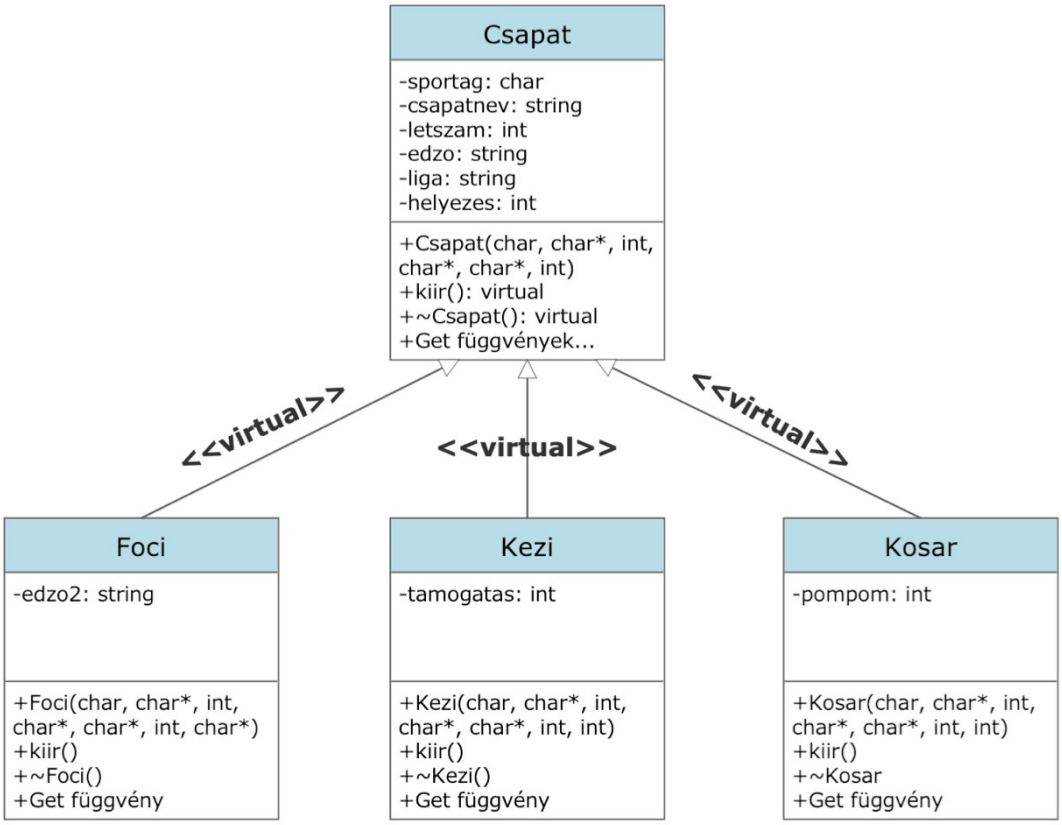
A Fitt Sportegyesület nyilvántartást szeretne vezetni a csapatairól. Minden csapat rendelkezik egy névvel és egy alaplétszámmal. A sportegyesület háromféle sportággal foglalkozik: labdarúgás, kosárlabda és kézilabda. A labdarúgó csapatnak két edzője van; a kosárlabda csapatnak elengedhetetlen kellékei 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!

Terv

A projektben 4 osztály mindenképpen szerepelni fog. Kell egy magának a csapatnak, amelyben szinte minden információ

megtalálható, illetve kell még három külön a sportágaknak. Az UML diagram a következő:



A csapat osztály az alábbi attribútumokból épül fel:

- sportag: Megadja a csapat sportágát. (F = Foci, H = Kézilabda, B = Kosárlabda. A rövidítéseknél az angol szó kezdőbtje szerepel.)
- csapatnev: Megadja a csapat nevét.
- letszam: A csapat játékosainak létszáma.
- edzo: Minden csapatnak van egy edzője, annak a nevét adja meg.
- liga: Melyik ligában szerepel a csapat. • helyezes: Megadja, hogy a csapat a ligában hanyadik helyen szerepel.

A Foci osztály az alábbi attribútumokból épül fel:

- edzo2: Mivel a focicsapatnak 2 edzője van, ezért a mások edző neve itt található.

A Kezi osztály az alábbi attribútumokból épül fel:

- tamogatas: Megadja, hogy a csapat mekkora összegű támogatást kapott.

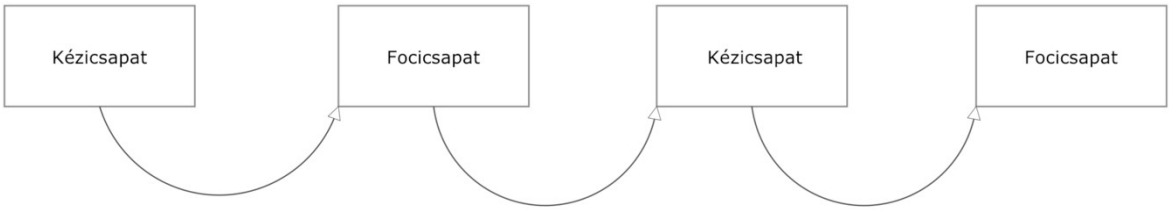
A Kosar osztály az alábbi attribútumokból épül fel:

- pompom: Megadja a pom-pom csapat létszámát.

Az UML diagramon látható, hogy kellenek get függvények is. A diagram olvashatósága végett azokat nem raktam bele. A get függvények az alábbiak lehetnek:

A get függvények azért kellenek, ha osztályon kívül le akarom kérdezni például a csapat nevét. Mivel az osztály adattagjai privátak, ezért ezek csak a tagfüggvényeknek érhetők el.

A program az adatokat láncolt listában fogja eltárolni. Az egyesületek egymás után fognak szerepelni, nem tesz megkülönböztetést a program a foci, vagy a kézi csapat között.



A megkülönböztetés a programozó számára annyiban rejlik, hogy az fő osztályban található egy olyan adattag, hogy sportág. Ezzel a különböző rangsorok készítésekor ki tudja szűrni, hogy például melyek a kézi csapatok.

A program indításakor a program automatikusan beolvassa a fájlban lévő adatokat. Egy csapat adata egy sort reprezentál a fájlban, és annak minden adata meg van adva. A fájlban az adatok pontosvesszővel (;) vannak elválasztva.

Specifikáció

A program a sportegyesület csapatainak adatait dolgozza fel. Ezen adatokkal a program különböző funkciókat fog ellátni. Például listákat fog tudni készíteni, amely alapján könnyebben meg lehet találni a keresett információt.

A program működésének elengedhetetlen feltétele a már meglévő .txt fájlok. A fájlban lévő adatok program elindításakor ezek automatikusan eltárolódnak. A felhasználónak nem lesz lehetősége a program indítása után létrehozni egy teljesen üres nyilvántartást. Lehetőség van „végtelen” csapatmennyiséget tárolni, ugyanis a program dinamikusan fog működni, azaz lehetőség van akármennyi csapatot hozzáadni, nem fog ilyen szempontból ránk szólni a program.

A felhasználói felület menüpontokból fog állni, amelyben a felhasználó gombkombinációk segítségével tud navigálni. Egy példa:

Üdvözlöm az XY Sportegyesület nyilvántartásában!
Az alábbi opciók közül tud választani:

(1) – Csapatok szerkesztése
(2) – Csapatok listázása
...
(x) – Kilépés a programból

Ha a felhasználó a csapatokat szerkeszteni szeretné akkor az egyes (1) gombot kell megnyomnia.
A főmenün kívül lesznek almenük is, ahol a részletesebb leírást kap a felhasználó, hogy pontosan mit lehet végrehajtani. Egy példa a csapatok szerkesztése almenüre:

A csapatok szerkesztése menübe lépett.
Az alábbiak közül tud választani:

(1) – Csapat felvétele
(2) – Csapat törlése
(3) – Csapat szerkesztése

Amennyiben a felhasználó a csapatokat szeretné szerkeszteni, vagy új csapatot szeretne létrehozni, akkor a csapat minden adatát meg kell adnia. Amennyiben valamilyen információ hiányzik a csapatról, azt egyszeren egy ’-’ jellel kell jelölni. Például:

Új csapat létrehozása.

Csapat név: XY
Létszám: - //Ismeretlen adat
Liga: NBI. Helyezés: 1
...

A felhasználó a programból csak a főmenün keresztül tud majd kilépni. Ilyenkor választani tud, hogy elmenti a program futása során keletkezett új adatokat, vagy sem.

Felhasználói Dokumentáció

A program futtatása után a felhasználó a menüben találja magát és innen tud navigálni különböző pontokba.

Az első dolog, amit a felhasználónak meg kell tennie, az a fájl beolvasása. Ha úgy akarja használni a programot, hogy nem olvasott be fájlt, akkor nem fogja engedni más menüpontba.

A fájl beolvasásánál a felhasználónak meg kell adnia egy fájl nevét .txt-vel a végén. Ilyen például az „adatok.txt”.

Amint beolvasta az adatokat visszakerül a főmenübe, és innen már az eddig nem elérhető menüpontokba is tud lépni.

A csapatok szerkesztése menüpontban a felhasználónak három opciója is van. Csapat hozzáadása, csapat törlése, és csapat szerkesztése.

Ha a csapat hozzáadása menüpontba lép, akkor a felvinni kívánt csapat minden adatát meg kell adni, de ezt a program úgy is ki fogja írni. A sportágak közül három opcióból választhat. ’F’ – foci, ’H’ – kézi, illetve ’B’ – kosár. A három sportágnak ez az azonosítója, viszont ha más sportágat adna meg a felhasználó, akkor ezt jelezni fogja a program. A többi adatnál pedig kedv szerint viheti fel, hogy például mi a csapat neve, vagy a létszáma.

A csapat törlése menüpontnál a felhasználó beírja a csapat nevét, és az törlésre kerül a nyilvántartásból. Ha olyan nevet ad meg, ami nincsen benne a nyilvántartásban, akkor a program ezt tudatni fogja.

A csapat szerkesztésénél pedig szintén megadja, hogy melyik csapatot szeretné szerkeszteni. Ilyenkor a program kiír minden adatot, ami annak a csapatnak van. A felhasználónak pedig egyszeren ki kell választania melyik adatot szeretné szerkeszteni. Például ha kiválasztja az egyes opciót, akkor a sportág, míg a kettes opciónál a csapatnév kerül szerkesztésre.

A hozzáadás, a szerkesztés és a törlés elvégzése után visszakerül a felhasználó a szerkesztés almenübe, és innen tud visszamenni a főmenübe.

A főmenüből még egy opció nyílik az adatok listázására. Ilyenkor a felhasználó kiválaszthatja, hogy mit szeretne kilistázni. Például ki tud íratni minden adatot, de azt is lehet, hogy csak a focicsapatokat írja ki.

Ha a felhasználó ki akar lépni a programból, akkor azt a főmenün keresztül teheti meg. Ilyenkor beüti a kilépésre szolgáló menüpontot. A program elmenti a szerkesztett nyilvántartást, amelyhez a felhasználónak meg kell adnia (úgy mint a beolvasásnál) egy fájlnevet, amelybe elmenti majd az adatokat. Ha a fájl nem létezik akkor az létrehozásra kerül, ha pedig már van ilyen, akkor az abban lévő adatokat felülírja.

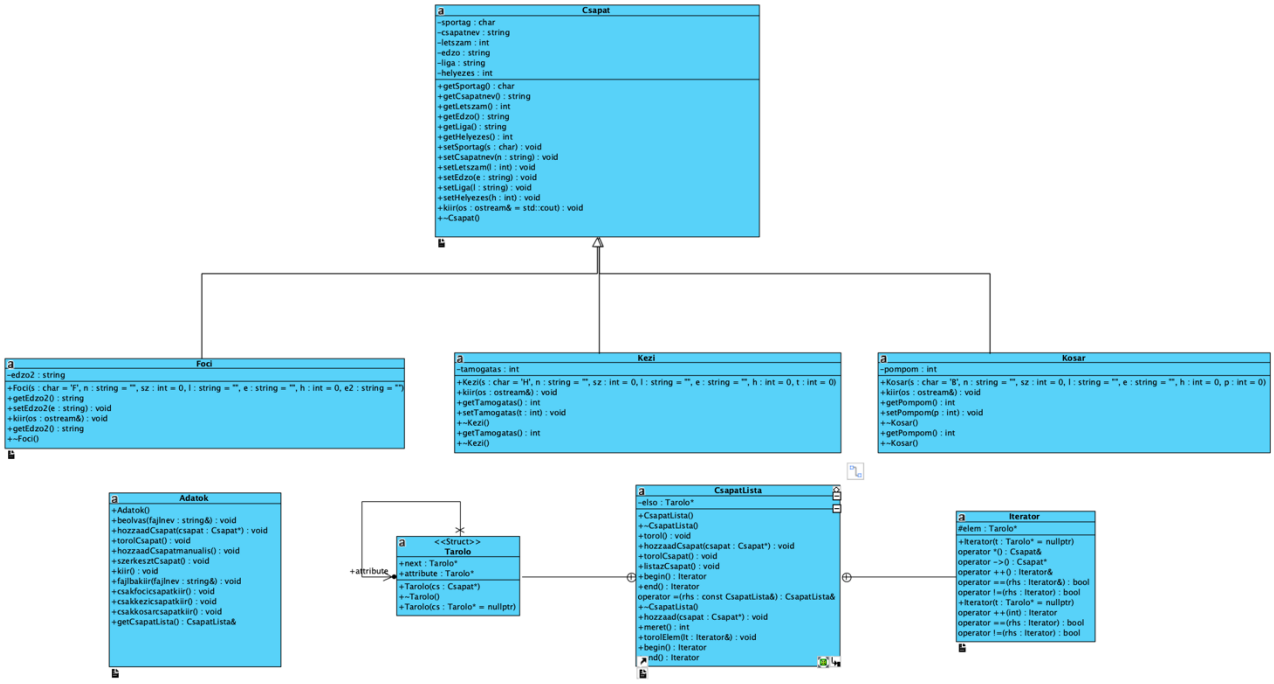
Programozói dokumentáció

A program MacOs Ventura rendszeren készült Clion fejlesztőkörnyezetben

Az egész projekt 8+1 (test.cpp) .cpp fájlt tartalmaz, 6+1 .h fájlt és 1 .hpp fájlt tartalmaz.

A projekt tervezéséhez elengedhetetlen egy igényesen felépített UML diagram. Ezzel a tervezési gondolatok és folyamatok tisztán átláthatók.

UML diagram:



A digramon tisztán látható, hogy melyik osztályok hogyan kapcsolódnak egymáshoz.

Az osztályok, illetve struktúrák:

- Csapat: ebben van a csapat alap adatai.
- Foci: plusz adattaggal bővül, amely az edzo2
- Kosar: plusz adattaggal bővült, amely a pompom
- Kezi: plusz adattaggal bővült, amely a tamogatas
- Csapatlista: „Csapat” objektumot tárol és egy láncolt listát reprezentál
- Iterator: lehetővé teszi az iterációt, a lista elemeinek bejárását a láncolt listában
- Tarolo: ez egy struktúra, amely a láncolt lista egy elemét reprezentálja.
- Adatok: Adatkezelő osztály

Fontosabb függvények elemzése:

Az alap osztályokban vannak setter, illetve getter függvények, amelyekkel könnyedén be tudjuk állítani, illetve lekérdezni az egyes adattagokat.
A kiir függvények egyszerűen kiírják a standard outputra a csapatok adatait. Minden osztálynál ez más, mert például a focicsapat más adattagokkal rendelkezhet mint a kézicsapat.

CsapatLista függvényei:

CsapatLista(): elso(nullptr) {}; :Konstuktor, amely egy üres láncolt listát inicializál.

~CsapatLista() {torol();} :Destruktor, amely törli a lista összes elemét a torol() függvény segítségével

void torol(); :Törli a láncolt lista összes elemét, és a dinamikusan foglalt területeket felszabadítja.

void hozzaad(Csapat* csapat); :Hozzáad egy Csapat objektumot a láncolt listához. A függvénynek más függvényekben is fontos szerepe van, mert az újonnan létrehozott elemet ezzel lehet hozzáadni. Például fájlból beolvasásnál.

int meret() const; :Visszaadja a láncolt lista hosszát.

void eltavolit(Iterator& It); :Törli a láncolt lista egy elemét. A függvények más függvényhívásnál fontos szerepe van, például amikor el akarunk távolítani egy csapatot a nyilvántartásból.

Iterator osztály függvényei:

Iterator(Tarolo* t = nullptr): elem(t) {}; :Konsturktor, amely beállítja az elem adattagot a megadott értékre.

Csapat& operator*() {return *elem->adat;} :Dereferáló operátor, amely visszaadja az aktuális elemhez tartzozó „Csapat” objektumot.

Csapat* operator->() {return elem->adat;}: Nyíl operátor, amely a memóriacím hozzáférési operátora. Visszaadja az aktuális elemre mutató pointer-t.

Iterator& operator++(); :Előre léptető operátora amely az aktuális elemet továbblépteti a láncolt listában, és visszatér az iterátorra mutató referenciával.

Iterator operator++(int); :Utólag léptető operátora. Először létrehoz egy másolatot az aktuális iterátorról, majd az aktuális elemet továbblépteti a láncolt listában. Végül visszatér a másolattal.

bool operator==(Iterator rhs) {return elem == rhs.elem;} : Egyenlőségvizsgáló operátora. Összehasonlítja két iterátort az aktuális elemük alapján, és visszatér az eredménnyel

bool operator!=(Iterator rhs) const{return elem != rhs.elem;}: Egyenlőtlenségvizsgáló operátora. Összehasonlítja két iterátort az aktuális elemük alapján, és visszatér az eredménnyel

Tarolo struktúra:

Tarolo(Tarolo *cs = nullptr): next(nullptr){} : Konstruktor

Adatok osztály függvényei:

void beolvas(const std::string& fajlnev); :A beolvas függvény a .txt fájl beolvasására szolgál. Megnyitja a felhasználó által megadott fájlt, és megnyitja. Ha nem tudta megnyitni, akkor kivételt dob. Ha megnyitotta, akkor az adatokat soronként olvassa be. A sort feldarabolja az elválasztókarakter mentén, és a megfelelő adattagba helyezi. Beolvassa először az adott sportágat, és az extra adattagot egy **if** elágazással a megfelelő helyre rakja. Konstruktor segítségével létrehozza azt az objektumot, majd hozzaadCsapat függvénnyel azt hozzáadja a láncolt listához.

void Adatok::hozzaadCsapat(Csapat *csapat): Hozzáadja a csapatot a láncolt listához.

void Adatok::torolCsapat(): Egy csapat törlésére szolgál a láncolt listából. Először bekéri a felhasználótól a csapat nevét, viszont ha nincsen ilyen csapat a listában, akkor kivételt dob. Ha azonban van ilyen csapat, akkor az iterátor segítségével megkeresi azt a láncolt listában, majd az eltavolit() segitsegevel törli azt a listából. Fontos, ilyenkor nem az egész lista törlődik, mint a torol() függvénynél, hanem csak az az egy elem.

void Adatok::kiir(): Kiírja a standard kimenetre a csapatok adatait. Iterátor segítségével végigmegy a láncolt listán, és meghívja az osztályok kiíró függvényeit.

void Adatok::hozzaadCsapatmanualis(): Egy csapat hozzáadására szolgál. A függvénynek a szerkesztés menüpont csapat hozzáadása pontnál van fontos szerepe. Bekéri a felhasználótól a csapat minden adatát. Először a sportágat kéri be. Ezzel eldönti, hogy a felhasználó milyen sportágú csapatot akar hozzáadni, és ehhez fogja alakítani az extra adat beolvasását.

void Adatok::szerkesztCsapat(): A csapatok szerkesztésére szolgál. Bekéri a felhasználótól a szerkesztendő csapat nevét. Ha nincs ilyen csapat, akkor kivételt dob. Ha van ilyen csapat, akkor kiírja annak minden adatát sor számozva. Ilyenkor a felhazsnáló választ, hogy mit szeretne szerkeszteni. Ha rossz számot ad be, akkor szintén kivételt dob. Amint megadta, hogy

mit szerkeszt, a program kiírja a régi adatot, és bekéri a felhasználótól az újat. Ezt a set függvény segítségével a frissíti.

void Adatok::fajlbakiir(const std::string &fajlnev): A program futása végén a program elmenti az elvégzett módosításokat. A függvény a felhasználótól bekéri a fájl nevét, amibe menteni akarja az adatokat. Ha nem tudta megnyitni a fájlt, akkor kivételt dob. Ha nem létezik ilyen fájl, akkor létrehoz egy ilyet, ha pedig már van ilyen akkor felülírja a benne lévő adatokat. A program az adatokat ugyanolyan formába írja ki a fájlba, ahogy megkapta, így azt ugyanúgy be lehet olvasni, és végezni rajta módosításokat.

void Adatok::csakfocicsapatkiir(): Csak a foci csapatokat írja ki. Ezt a sportág segítségével szortírozza, és ez alapján kerül az adat kiírásra.

void Adatok::csakkezicsapatkiir(): Csak a kézi csapatokat írja ki. Ezt a sportág segítségével szortírozza, és ez alapján kerül az adat kiírásra.

void Adatok::csakkosarcsapatkiir(): Csak a kosár csapatokat írja ki. Ezt a sportág segítségével szortírozza, és ez alapján kerül az adat kiírásra.

További függvények találhatók a menu.cpp fájlban amelyeknek leginkább a kód olvashatósága a szerepe. Itt találhatók továbbá az almenük is, amely a főmenühöz hasonlóan switch-case szerkezettel lett megoldva. A program futása során keletkező kivételek a menüben vannak kezelve.