HÁZI FELADAT

Programozás alapjai 2.

Dokumentáció

Jakosa Emma Kloé
O5EPGN

Tartalom

1. Feladatválasztás

A tárgyhonlapon megtalálható feladatötletek listájából választottam a Sportegyesület elnevezésű programot nagy házi feladatomként. Egy nyilvántartást fogok készíteni az egyesület 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.

2. Feladatspecifikáció

Menü:

A programom menüvezérelt lesz. Konzolablakban nyílik meg a menü, amely az alábbi menüpontokat tartalmazza:

(1) Labdarúgás
(2) Kosárlabda
(3) Kézilabda

Az egyes menüpontok kiválasztása a hozzájuk tartozó szám lenyomásával lehetséges. Miután a felhasználó belépett egy menüpontba a 0 billentyűzet lenyomásával térhet vissza a menübe.

Ha a felhasználó kiválasztja az adott sportágat, akkor azokon belül az alábbiakra van lehetősége:

(0) Vissza a menübe
(1) Csapatok listázása
(2) Új csapat
(3) Csapat törlése

A 3 funkció részletezése:

Csapatok listázása: Az alap adatokon túl (csapatnév, létszám), mindig az adott sportágnak megfelelő plusz információk is listázásra kerülnek. Az egy csapathoz tartozó adatok egy sorban, külön oszlopban lesznek. Az első csapatot tartalmazó sortól kezdődően számozva lesznek a sorok.

Új csapat: Új csapat felvétele esetén a sportágnak megfelelő adatok bekérésére kerül sor.

Csapat törlése: A sportágnak megfelelő csapatok listázása a „Csapatok listázása” funkcióhoz hasonlóan először sorszámozva listázza a csapatneveket, majd az adott csapathoz tartozó sorszám lenyomásával van lehetőség a kiválasztott csapat törlésére.

Hibakezelés:

Az egyes adatok bekérése kérdések és utasítások segítségével valósul meg, melyek precízen megfogalmazva tisztázzák az adott esetben elvárt, megfelelő formátumot.

Ettől függetlenül a program képes lesz pár hiba kezelésére:

- Amikor a program számot vár, de karaktereket kap, akkor egy megjelenő üzenet utasítja a felhasználót, hogy számot adjon meg. (csapat létszám, pom-pom lányok száma, támogatás összege, törölni kívánt csapat sorszáma)
- Amikor a program pozitív szám helyett negatívat kap, akkor a megjelenő hibaüzenet arra utasítja a felhasználót, hogy pozitív számot adjon meg.
- Amikor a felhasználó túl nagy számot ad meg (a törölni kívánt csapat sorszámának beírásakor), akkor a megjelenő üzenet utasítja, hogy egy adott intervallumon belüli számot adjon meg.
- Amikor a bemeneten a program karaktereket vár, de a felhasználó számokat ad meg, akkor karakterekből álló szöveg megadására utasítja az üzenet a felhasználót. (csapatnév megadása)

Fájlkezelés:

A csapatokat és azoknak adatait txt fájlban fogom tárolni. Ez a fájl minden új csapat felvételekor, illetve törlésekor frissül, valamint a program indulásakor innen olvassa be a már felvett csapatok adatait.

3. Pontosított feladatspecifikáció

A program képes txt-be menteni, illetve abból beolvasni. A csapatokra mutató pointereket egy dinamikus tömbben tárolom. A programot öt osztállyal valósítom meg. Ezek közül a Sportegyesulet osztálynak úgymond tároló szerepe van, ugyanis itt tárolom dinamikus tömbben (heterogén kollekcióval) a csapatokra mutató pointereket. A Csapat egy absztrakt osztály, mely rendelkezik 3 leszármazottal (Labdarugas, Kosarlabda, Kezilabda). A csapatoknak tárolva van a neve és a létszáma, illetve a sportága (ezt én az egyszerűség kedvéért egy int értékkel azonosítottam), továbbá minden sportág rendelkezik egy plusz sajátos információval, tehát a 3 leszármazott osztálynak van egy-egy sajátos attribútuma.

Az osztályok megvalósítják az alábbi műveleteket (, melyekért az osztályok tagfüggvényei a felelősek):

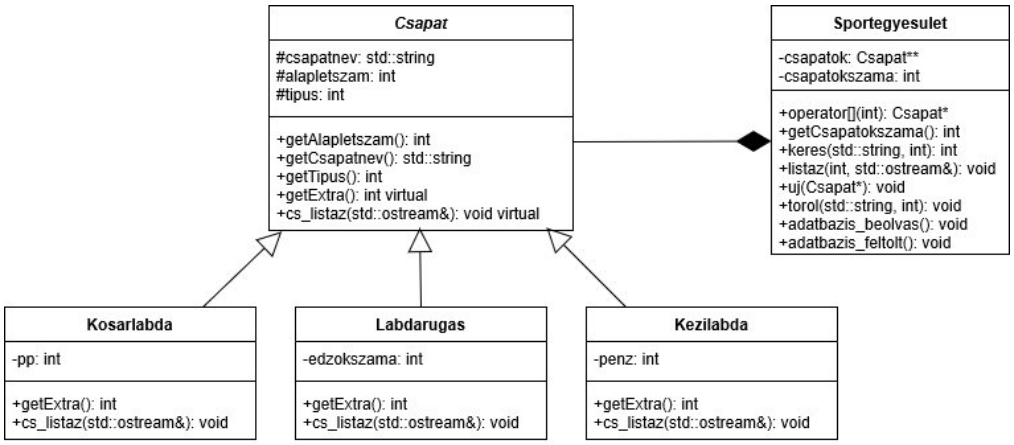
- létrehozás,
- megszüntetés,
- értékadás,
- listázás,
- elemek számának lekérdezése.

A 3 funkció (listázás, törlés, új hozzáadása) megvalósításáért felelős függvényeket a Sportegyesulet osztályban tárolom. Ezek szükség esetén virtuális függvények is lehetnek (például a csapatok és adataik listázásakor), hogy lehetőség legyen a leszármazottakban a felüldefiniálásra.

- listázó függvény (listázza a választott sportág csapatait és azoknak adatait)
- új hozzáadó függvény (hozzáfűzi az addig meglévő csapatokhoz az újat – láncolt lista)
- törlő függvény (törli a listából a törölni kívánt csapatot)

(Továbbá a program függvényei még az osztályokban lévő konstruktor, másoló konstruktor, illetve destruktor is.)

4. Algoritmusok, UML



Sportegyesulet:

Ebben tárolom a csapatokra mutató pointereket egy dinamikus tömbben, valamint segédváltozóként a csapatok számát is tárolom, hogy könnyen kezelhető legyen a tömb. A tömb könnyű kezelhetősége és a túlindexelés megelőzése érdekében felüldefiniálom a [] operátort. Ez az osztály foglalkozik a legfőbb funkciókkal, így itt találhatóak az új csapat hozzáadásáért, csapat törléséért és a listázásért felelős függvények.

Új csapat hozzáadása:

```
void Sportegyesulet::uj(Csapat* ujcsapat){
   Csapat** uj = new Csapat*[csapatokszama + 1];
   for(int i = 0; i < csapatokszama; i++){
      uj[i] = csapatok[i];
   }
   uj[csapatokszama] = ujcsapat;
   delete[] csapatok;
   csapatokszama+=1;
   csapatok = uj;
}
```

Paraméterként egy csapatra mutató pointert kap. Létrehozok egy, a már meglévőnél eggyel több elemű tömböt és ebbe átmásolom az eredeti csapatok tömb elemeit. Majd a legújabb, paraméterként kapott új csapatot is bemásolom. Felszabadítom az eredeti pointerekből álló tömbre mutató pointert és egyenlővé teszem az új tömbbel. A csapatok számát pedig növelem eggyel.

Csapat törlése:

```
void Sportegyesulet::torol(std::string nev, int type){
   if(csapatokszama == 0) throw("Egy csapat sincs a nyilvantartasban!");
   int idx;
   idx = keres(nev,type);
   if(idx < 0 || idx > csapatokszama) throw std::out_of_range("Nincs ilyen indexu csapat!");
   Csapat** uj = new Csapat*[csapatokszama - 1];
   for(int i = 0; i < idx; i++){
      uj[i] = csapatok[i];
   }
   for(int i = idx; i < csapatokszama-1; i++){
      uj[i] = csapatok[i+1];
   }
```

```
    delete csapatok[idx];
    delete[] csapatok;
    csapatokszama-=1;
    csapatok = uj;
}
```

Csapat törlése a csapatnév megadásával lehetséges, ezért ehhez a függvényhez létrehoztam egy segédfüggvényt (keres(std::string,int): int), amely a megadott név és a csapat típusa alapján megkeresi, hogy hányadik indexű helyen található a csapat, ha pedig nem találató ilyen nevű csapat, akkor hibaüzenetet dob.

A törölben először ellenőrzöm, hogy nem üres-e a tömböm, majd meghívom a keres-t. Ellenőrzöm, hogy biztos ne legyen túlindexelés, aztán létrehozok egy új, az eredetihez hasonló, csak eggyel kevesebb elemű tömböt. Majd megkezdődik a másolás az első for ciklussal, ami egészen addig megy amíg el nem érünk a törölni kívánt elemig. Ekkor kezdődik a második for ciklus, ahol az eredeti tömbben eggyel előbbre lépünk, így kimarad a törlendő elem a másolásból. Ekkor fel kell szabadítani a törölt csapatunkra mutató pointert, majd az eredeti tömbre mutató pointert, hogy aztán egyenlővé tehessük az új pointerekből álló tömbre mutató pointerrel.

Listáz:

```
void Sportegyesulet::listaz(int type, std::ostream& os){
    for(int i=0; i < csapatokszama; i++) {
        if(type==csapatok[i]->getTipus()){
            csapatok[i]->cs_listaz(os);
        }
    }
}
```

A csapatok kiírásához segítségül szolgál egy, a Csapat absztrakt osztályban lévő virtuális cs_listáz függvény. Erre a segédfüggvényre azért van szükség, mert a csapatokat sportágak szerint lehet listázni, márpedig a különböző sportágak különböző extra tulajdonsággal rendelkeznek. Így adott helyzetben az adott sportágnak megfelelő cs_listáz függvényét hívja meg a listáz függvény, hogy kiírhassa a csapatokat.

Ezeken kívül a fájlkezelést is ez az osztály végzi:

Fájlba írás:

```
void Sportegyesulet::adatbazis_beolvas(){
    std::ifstream adatbazis("adatbazis.txt");
    std::string nev;
    int tipus, letszam, extra;

    if(adatbazis.is_open()){
        while(!adatbazis.eof()){
            adatbazis >> tipus >> nev >> letszam >> extra; /* Beolvassa az egész sort és a tipustól
függöen létrehozza a csapatot */
            switch(tipus){
            case(1):
                uj(new Kosarlabda(nev,letszam,tipus,extra));
                break;
            case(2):
                uj(new Labdarugas(nev,letszam,tipus,extra));
                break;
            case(3):
                uj(new Kezilabda(nev,letszam,tipus,extra));
                break;
            default:
                return;
            }
```

```
        }
        adatbazis.close();
    }
    else std::cout << "Nem sikerult megnyitni a fajlt!";
}
```

A függvényben létrehoztam segédváltozokat, melyekbe a megnyitott fájlból soronként beolvashatom az adatokat és létrehoztam a segítségükkel a megfelelő csapatokat. Sportágtól (típustól) függ, hogy melyik leszármazott konstruktorát kell meghívni, erre használom a switch-case-t.

Fájlba írás:

```
void Sportegyesulet::adatbazis_feltolt(){ /* A txt fájlt újratölti */
    std::ofstream adatbazis;
    adatbazis.open("adatbazis.txt", std::ofstream::out | std::ofstream::trunc);
    if(adatbazis.is_open()){
        for(int i=0; i < csapatokszama ; i++){
            adatbazis << csapatok[i]->getTipus() << " " << csapatok[i]->getCsapatnev() << " " <<
csapatok[i]->getAlapletszam() << " " << csapatok[i]->getExtra() << "\n";
        }
        adatbazis.close();
    }
    else std::cout << "Nem sikerult megnyitni fajlt!";
}
```

Azt a megoldást választottam, hogy a fájlt mindig teljesen újratöltöm, így ehhez először ürítem, majd az összes csapatot beleírom, ezt egy for ciklussal valósítottam meg.

Csapat:

A csapat osztály tartalmazza a csapatok azon adatait, melyekkel sportágtól függetlenül minden csapat rendelkezik, ezeken túl minden leszármazottnak van egy további sajátos attribútuma. Mivel ez az osztály egy absztrakt osztály, így a 3 getter függvényén kívül virtuális függvényeket tartalmaz, melyeket aztán a leszármazottak felüldefiniálnak. A 3 getter függvénye a Csapat osztály által tárolt protected adattagok elérésére szolgálnak. A 4. get függvény egy virtuális függvény, ami a leszármazottak sajátos attribútumainak elérésében segédkezik. Ezen kívül itt a már fentebb említett cs_listaz függvény, amely segít a Sportegyesulet listaz függvényének. A Kosarlabda osztály által felüldefiniált cs_listaz függvény:

```
void Kosarlabda::cs_listaz(std::ostream& os){
    os << csapatnev << ", " << alapletszam << " fo, " << pp << " pompom lany" << std::endl;
}
```

5. Tesztelés

5.1 Memóriakezelés tesztje

A memóriakezelés ellenőrzését a laborgyakorlatokon is használt MEMTRACE modullal végeztem. Ehhez minden önálló fordítási egységbe include-oltam a „memtrace.h” állományt a standard fejlécállományok után. Memóriakezelési hibát nem tapasztaltam.

5.2 Lefedettségi teszt

A feladatkiírás kérte a külön modulként forduló tesztprogramot. Mivel én konzolos menüvel rendelkező programot készítettem, ezért két main-em van. Definiáltam egy TESZTEK makrót, melynek ha 1 az értéke, akkor tesztel, ha 0, akkor pedig a menü main-je él. A teszteket gtest_lite segítségével végeztem, melyek a program minden ágát lefedték. (JPORTA értékelés: 94, 36%)