HÁZI FELADAT

Programozás alapjai 2.

Végleges

Boros Gergő
IGMEF9
L1-R4B

2020. május 16.

TARTALOM

1. Feladat

Sportegyesület

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

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.

2. Feladatspecifikáció

A kész C++ program teljesen menüvezérelt lesz. A felhasználó a konzolablakból a főmenüben számok segítségével kiválaszthatja, hogy melyik menüpontot szeretné választani:

(1) Új csapat felvétele
(2) Csapat törlése
(3) Listázás
(4) Kilépés

Maga a program könnyen navigálható, hiszen az egyes menükből mindig vissza tudunk térni a főmenübe a 0 gomb megnyomásával. Az egyes funkciók kezelésekor, menüpontok választásánál megjelennek utasítások, kérdések és az adott helyzetben elvárt adatformátumok is, melyek segítik a felhasználót a program működtetésében.

Funkciók használata, adatok bekérése a billentyűzetről történik. Az új csapat felvétele opció választása után megadhatjuk a csapat sportágát, nevét, alaplétszámát, továbbá a speciális tulajdonságát (a labdarúgó csapat edzőit, a kosárlabda csapat pom-pom lányainak számát, a kézilabda csapat évente kapott támogatásának összegét), viszont egy sportágban nem lehetnek ugyanolyan nevű csapatok. Az új csapat adatai a felvétel után egy szöveges fájlban(.txt) tárolódnak az összes csapattal együtt. A csapat törlése lehetőség választása után szintén meg kell adni a csapat nevét, azonban szükséges még a sportág is, ugyanis lehet ugyanolyan nevű csapat két különböző sportágban. A felhasználó a listázás funkció választása után kiválaszthatja a főmenühöz hasonlóan, hogy mely sportág csapatait szeretné látni:

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

A program képes lesz a hibakezelésre is az alábbi kivételek esetében:

- Ha egy sportágban pontosan ugyanolyan nevű csapatot akar létrehozni a felhasználó.
- Ha nem megfelelő a számadatot kap a program, például alaplétszám vagy pom-pom lányok száma esetén negatív számot, akkor szintén hibaüzenet jelenik meg.
- Ha a program számot vár a billentyűzetről és a felhasználó karaktert ad meg, hibaüzenettel szembesül, ami felhívja a figyelmet az elvárt formátumra.

3. Pontosított feladatspecifikáció

A program a sportegyesület csapatait a `Sportegyesulet` osztályon belül egy tömbben tárolja, mely a csapatokra mutató pointereket (heterogén kollekció) tartalmaz. Az egyes csapatok nevét, létszámát és tulajdonságaikat egy szöveges állományban (.txt fájlban) is tároljuk, melyből akár később ki is olvashatunk, vagy hozzá is fűzhetünk. Maga a `Csapat` osztály három leszármazottal rendelkezik: `Labdarugas`, `Kezilabda`, `Kosarlabda`. Ennek a három osztálynak vannak különleges tulajdonságaik, azonban rendelkeznek közös műveletekkel, amik végrehajtására képesek:

- létrehozás,
- megszüntetés,
- értékadás,
- listázás.

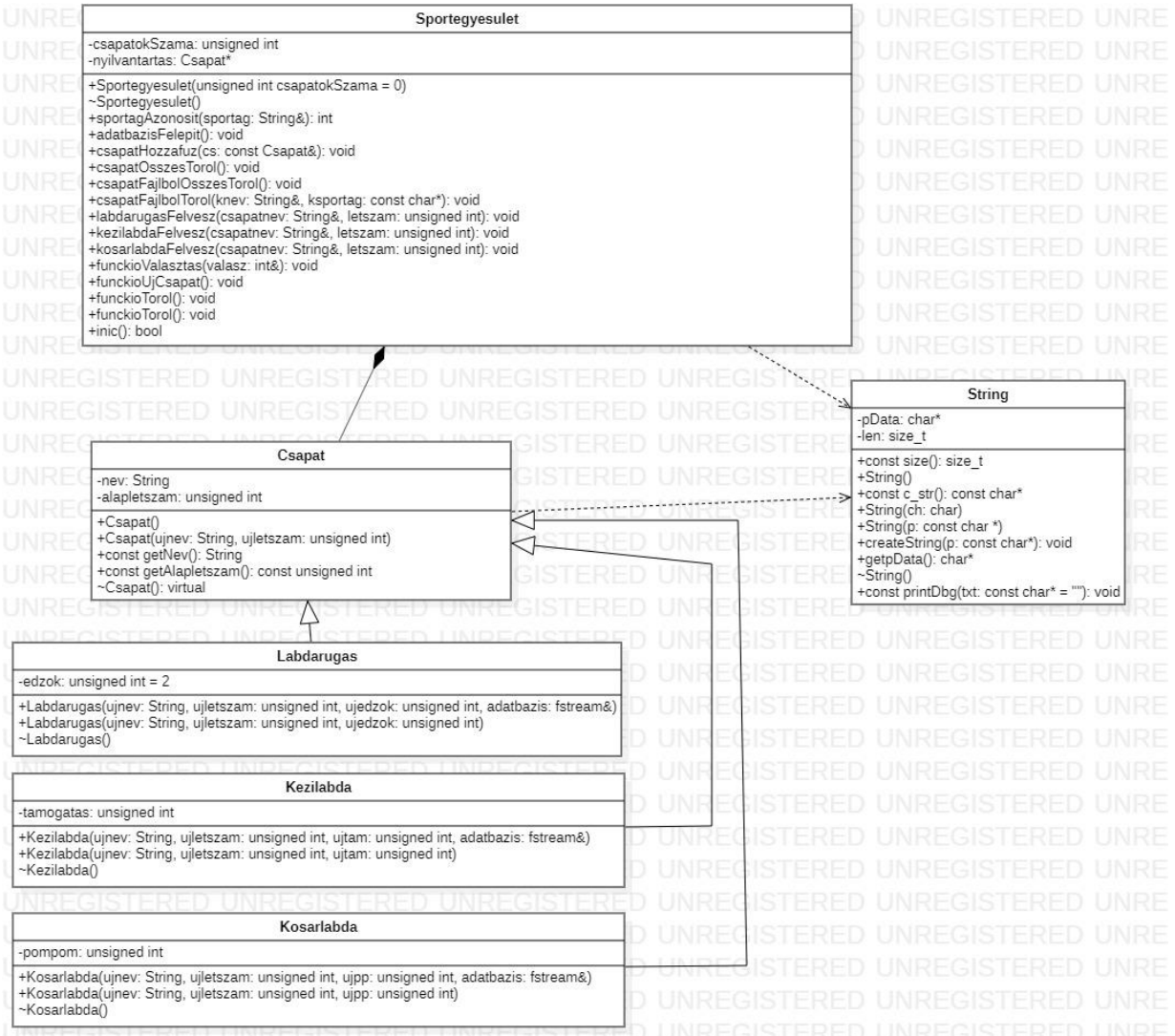
Mivel a feladat esetében STL tároló nem használható, ezért egy saját `String` osztály is létezik, amely jóval kényelmesebbé teszi a csapatok neveivel történő műveleteket.

Az egyes tesztesetekhez, olyan program is elkészül, mely meghívja a program minden függvényét legalább egyszer, illetőleg a program fontosabb ágait lefuttatja. Maga a megvalósítás teljesen moduláris, a főprogram és a tesztprogram is külön fordítható. A tesztprogram a standard inputról beolvasott adatok alapján hajtja végre a fentebb említett műveleteket. A tesztadatok között direkt hibásan megadottak is lesznek, amelyre a program hibaüzenetekkel válaszol.

4. Terv

A feladat több osztályból és egy tesztprogramból álló projekt megtervezését igényli.

4.1. Objektum terv



A `Sportegyesület` osztály minden `Csapat` osztályból származtatott objektum tárolására alkalmas. A sportágak csapatainak alaposztályából (itt `Csapat`) származtatott objektumokat (`Labdarúgás`, `Kézilabda`, `Kosárlabda`), alaposztályukra mutató pointereket (`Csapat*`) tartalmazó tömbbel tároljuk. Tehát magát a Sportegyesületet egy heterogén kollekcióval modelleztük. A `Sportegyesület` egy nagy tároló, ami az egyes csapatok közös ősére (`Csapat`) mutató pointereket tárol.

Ezt a tervet kiegészíti egy saját `String` osztály, amely megkönnyíti a csapatok neveivel történő munkát az előre megírt `String` osztálybeli tagfüggvények, műveletek segítségével.

4.2. Algoritmusok

4.2.1. A program algoritmusai

Az összes funkció elvégzéséért nyilvánvalóan a megfelelő osztályok tagfüggvényei felelnek. A menüvezérelt felületért a `Sportegyesulet` osztály a felelős. A program képes lesz az alábbi funkciókra függvények segítségével:

- listázni bizonyos csapatok nevét és tulajdonságait, legyenek ezek egy sportágban vagy nem
- létrehozni egy új csapatot, hozzáfűzve ezzel a meglévőekhez
- törölni már meglévő csapatot

Ezeken mellett természetesen megtalálható az osztályokban a konstruktor, másoló konstruktor és destruktor is.

4.2.2 A tesztprogram algoritmusai

A tesztprogram a standard bemenetről egy teszteset sorszámot kér be, majd különböző elágazásokon keresztül teszteli a program főbb ágait, melyek a különféle funkciókat vezérlik.
A tesztprogram leteszteli az új csapat létrehozását, csapatok listázását, illetve törlését is.

5. Megvalósítás

A program moduláris megvalósítása összesen 5 osztály megvalósítását igényelte. A *sportegyesulet.cpp*-ben található *Sportegyesulet* osztály tartalmaz egy *Csapat* objektumokból álló dinamikus méretű tömböt, amely segítségével tudjuk tárolni a különböző sportágak csapatait, legyen az *Labdarugas, Kezilabda,* vagy *Kosarlabda.* A *Sportegyesulet* osztály a gerince a programnak, hiszen a felhasználó által választott funkciókat ennek az osztálynak a tagfüggvényei látják el. A *csapatok.h-ban* deklarált tagfüggvények, konstruktorok, destruktorok lehetővé teszik, hogy könnyen létrehozhassunk, töröljünk akármilyen sportágú csapatokat, hozzáfűzve ezzel őket a *Sportegyesulet* nyilvántartásához. A szöveges adatok kezelésére, pedig egy saját String osztály elkészítése is létrejött, ami az operator overloading segítségével nagyban megkönnyíti a szövegekkel kapcsolatos műveletek elvégzését, hiszen STL tárolót nem lehetett használni. A továbbiakban bemutatom a legfőbb osztályokat és a tesztprogramot.

5.1 Az elkészített főbb osztályok tagfüggvényei

```
class Sportegyesulet  {
    unsigned int csapatokSzama;
    Csapat* nyilvantartas;
public:
      //A konstuktor inicializáló listával:
      Sportegyesulet(unsigned int csapatokSzama = 0) :
csapatokSzama(csapatokSzama), nyilvantartas( new Csapat[csapatokSzama]) { }

      //Destruktor:
    ~Sportegyesulet() { csapatokSzama = 0; delete [] nyilvantartas; }

      //Értékadó operátor
    Sportegyesulet& operator=(const Sportegyesulet &s);

      //Indexelő operátororok
    Csapat &operator[](unsigned int ii);
    const Csapat operator[](unsigned int ii) const;

      //Sportágat lehet vele azonosítani név alapján
    int sportagAzonosit(String &sportag);

      //Beolvassa és felépíti a Sportegyesulet nyilvantartas tömbjét az
      objektumok adatait tartalmazó txtből,
      //ezzel megőrízve a program perzisztenciáját
      //Egyik legfontosabb függvénye a programnak
    void adatbazisFelepit();

      //A nyilvantartas tömb végéhez ad egy elemet
    void csapatHozzafuz(const Csapat &cs);

      //Törli az összes csapatot a nyilvantartas tömbből
    void csapatOsszesTorol();

      //Wipeolja a csapatokat tartalmazó txt-t
    void csapatFajlbolOsszesTorol();

      //Egy adott nevű és sportágú csapatot kitöröl az összes csapatot
      tartalmazó txt-ből
    void csapatFajbolTorol(String &knev, const char* ksportag);

      //Az alábbi három függvény segítségével tudunk hozzáfűzni a
      nyilvántartás tömbhöz, ezzel egyben tartva az adatbázist
      //A méretet mindig dinamikus növeljük
      //Az új elem a tömb végére kerül
    void labdarugasFelvesz(String& csapatnev, unsigned int letszam);
    void kezilabdaFelvesz(String& csapatnev, unsigned int letszam);
    void kosarlabdaFelvesz(String& csapatnev, unsigned int letszam);
```

```
      //Leellenőrzi, hogy létezik-e már egy adott sportágban egy adott
      névvel csapat, ha nem létrehozzuk
    void funkcioEllenoriz(int& valasz);

      //Meghatározza melyik funkciót választotta a felhasználó
    void funkcioValasztas(int &valasz);

      //Új csapat létrehozásának működését kezeli
    void funkcioUjCsapat();
      //Törlés funkció kezelésére szolgáló függvény
    void funkcioTorol();
      //A listázás funkció mindenféle kezelésére való függvény
    void funkcioListazas();
      //Az elérhető funkciók kiválasztására szolgál
    bool inic();
};

class Csapat {
    String nev;
    unsigned int alapletszam;

public:
    Csapat() {}

    Csapat(String ujnev, unsigned int ujletszam)
    : nev(ujnev), alapletszam(ujletszam){ }

    const String getNev() const { return nev; }

    const unsigned int getAlapletszam() const { return alapletszam; }

    virtual ~Csapat() { }
};

class Labdarugas : public Csapat {
    unsigned int edzok = 2;
public:
    Labdarugas(String ujnev, unsigned int ujletszam, unsigned int ujedzok,
    std::fstream &adatbazis): Csapat(ujnev, ujletszam), edzok(ujedzok) {}

      //Konstruktor, mely beleírja a csapat adatait a txt-be
    Labdarugas(String ujnev, unsigned int ujletszam,unsigned int ujedzok) :
    Csapat(ujnev, ujletszam), edzok(ujedzok){
    std::ofstream adatbazis;
    adatbazis.open("adatbazis.txt", std::ios_base::app);
    adatbazis << "Labdarugas " << getNev() << " " << getAlapletszam()<<""
    << edzok << std::endl;
    adatbazis.close();
    }

    ~Labdarugas() {}
};
```

5.2 Tesztprogram bemutatása

A *tesztek.cpp* állományban található tesztprogram a standard bemenetről olvas be pozitív egész számokat, amelyek alapján különféle teszteseteket hajt végre, lefedve ezzel a program fő ágait. A főprogram egész szám beolvasása után eldobja a sor többi részét, majd meghívja a megfelelő tesztesetet.

5.2.1.A tesztprogram algoritmusai

`void tesztUjCsapat(Sportegyesulet &s)`

1. tesztesetet megvalósító függvény. Új csapat funkcióját teszteli, amely mindhárom sportág esetében ugyanaz

`void tesztSportagListaz(Sportegyesulet &s)`

2. tesztesetet megvalósító függvény. Egy adott sportág csapatainak listázását teszteli

`void tesztOsszesListaz(Sportegyesulet &s)`

3. tesztesetet megvalósító függvény. A Sportegyesület összes csapatának listázását teszteli

`void tesztAdottCsapatTorol(Sportegyesulet &s)`

4. tesztesetet megvalósító függvény. Egy bizonyos nevű csapatot töröl a nyilvántartásból.

`void tesztOsszesCsapatTorol(Sportegyesulet &s)`

5. tesztesetet megvalósító függvény. Kitörli a Sportegyesület összes csapatát a nyilvántartás dinamikus tömbből.

6. Tesztelés

A főprogram tesztelése során az osztályok funkcionális tesztjeit a következőképpen alakítottam ki:

- új csapat funkció tesztelése, különböző sportágú csapatok esetében is
- csapatok listázása funkció tesztelése, különböző sportágú csapatok esetében is
- egy adott csapat törlése név és sportág alapján
- összes csapat törlése funkció kipróbálása

6.1 A funkcionális tesztek

tesztUjCsapat

```
1       //Uj Labdarugas csapat teszteles
1
1
Ujfocicsapat
23
--------- end ----------
```

```
1       //Uj Kezilabda csapat teszteles
1
2
Ujkezicsapat
24
50000
--------- end ----------

1       //Uj kosarlabda csapat tesztelese
1
3
Ujkosarcsapat
25
10
--------- end ----------
```

tesztSportagListaz

```
1 //Labdarugas csapatok listazasa teszteles
2
1
--------- end ----------

1 //Kezilabda csapatok listazasa teszteles
2
2--------- end ----------


1 //Kosarlabda csapatok listazasa teszteles
2
3
--------- end ----------
```

tesztOsszesListaz

```
1 //Osszes csapat listazasa teszteles
3
4
--------- end ----------
```

tesztAdottCsapatTorol

```
1 //Adott csapat torlese teszteles
4
1
Ujfocicsapat
1
--------- end ----------
```

```
testOsszesCsapatTorol

1 //Osszes csapat torlese teszteles
5
3 //Kilepes
--------- end ----------
```

6.2 Memóriakezelés tesztje

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