Programozás II. 1. ZH

SZTE Szoftverfejlesztés Tanszék

2023. ősz

Technikai ismertető

- A programot C++ nyelven kell megírni.
- A megoldást a *Bíró* fogja kiértékelni.
  - A Feladat beadása felületen a Feltöltés gomb megnyomása után ki kell várni, amíg lefut a kiértékelés. **Kiértékelés közben nem szabad az oldalt frissíteni vagy a Feltöltés gombot újból megnyomni** különben feltöltési lehetőség veszik el!
- Feltöltés után a *Bíró* a programot `g++` fordítóval és a
  `-std=c++17 -static -O2 -DTEST_BIRO=1`
  paraméterezéssel fordítja és különböző tesztesetekre futtatja.
- A program működése akkor helyes, ha a tesztesetek futása nem tart tovább 5 másodpercnél és hiba nélkül (0 hibakóddal) fejeződik be, valamint a program működése a feladatkiírásnak megfelelő.
- A *Bíró* által a `riport.txt`-ben visszaadott lehetséges hibakódok:
  - Futási hiba 6: Memória- vagy időkorlát túllépés.
  - Futási hiba 8: Lebegőpontos hiba, például nullával való osztás.
  - Futási hiba 11: Memória-hozzáférési probléma, pl. tömb-túlindexelés, null pointer használat.
- A riport.txt és a fordítási log fájlok megtekinthetőek az alábbi módon:
- A programot 20 alkalommal lehet benyújtani, a megadott határidőig.
- A programban szerepelhet `main` függvény, amely a pontszámításkor nem lesz figyelembe véve. Azonban ha fordítási hibát okozó kód van benne az egész feladatsor 0 pontos lesz.

Általános követelmények, tudnivalók

- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz pont.
- Minden metódus, amelyik nem változtatja meg az objektumot, legyen konstans! Ha a paramétert nem változtatja a metódus, akkor a paraméter legyen konstans!
- string összehasonlításoknál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")

- A leírásokban bemutat példákban a string-ek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!
  - A tesztesetekben nem lesz ékezetes szöveg kiíratása.
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

Lokális tesztelés

A minta.zip tartalmaz egy kiindulási feladat.cpp-t, amit a megoldással kiegészítve lokális tesztelésre használhattok. A fordítás az előbbiekben leírt módon történjen. A fájl felépítése a következő.

```
#include <iostream>
#include <string>
#include <cassert>

using namespace std;

//////////////////////////
//Ide dolgozz!!
//////////////////////////

//=== Teszteles bekapcsolasa kikommentezessel
//#define TEST_alma
//=== Teszteles bekapcsolas vege

#if !defined TEST_BIRO
/*
Keszits egy fuggvenyt, ami visszaadja az alma sztringet!
*/
void test_alma(){
  #ifdef TEST_alma && !defined TEST_BIRO
   string s = alma();
   assert(s == "alma");
  #endif
}
int main(){
  test_alma();
}
#endif
```

Ha megoldottad az alma feladatot, úgy tudod tesztelni, ha kitörlöd a kommentjelet a `#define TEST_alma` sor elől. Ekkor **újrafordítás** után le fog futni a `test_alma()` függvény tartalma is. Ha a visszaadott sztring nem az elvárt, az assert() függvény ezt jelezni fogja. A **define-ok módosítása nem javasolt**, fordítási hibát idézhet elő a biro-n való teszteléskor! **A tesztelőkód nem végez teljes körű tesztelést!** Saját felelősségre bővíthető. A sikeres megoldás után a feladat.cpp tartalma (mely biro-ra is feltölthető):

```
#include <iostream>
#include <string>
```

```cpp
#include <cassert>

using namespace std;

string alma(){
  return "alma";
}

//=== Teszteles bekapcsolasa kikommentezessel
#define TEST_alma
//=== Teszteles bekapcsolas vege

/*
Keszits egy fuggvenyt, ami visszaadja az alma sztringet!
*/
void test_alma(){
  #ifdef TEST_alma && !defined TEST_BIRO
  string s = alma();
  assert(s == "alma");
  #endif
}

int main(){
  test_alma();
}
```

1. Feladat: Osztály halmazba pakolása

Adott a kiindulási fájlban egy Osztaly nevű osztály, ami iskolai osztályokat reprezentál. Egy előjeltelen egész számban tárolja, hogy hanyadik évfolyamos az osztály, a kód adattag pedig azt adja meg, hogy 'a' vagy 'b' vagy 'c' stb osztályról van-e szó. Bővítsd úgy az osztályt, hogy be tudjuk pakolni egy halmazba. A rendezés az alábbi módon történjen:

- A felsorolásban előrébb vannak az alacsonyabb évfolyamok. Pl. 9a $<$ 10a, 9b $<$ 10a
- Ha két osztály évfolyama megegyezik, akkor az van előrébb a felsorolásban, amelyiknek a kódja lexikografikusan kisebb. Pl. 9b $<$ 9c, 9a $<$ 9b

Készítsd el az `osztaly_set` nevű függvényt, ami a paraméterében egy osztályokat tároló vektort kap és átmásolja a vektor elemeit egy halmazba. Ezzel a halmazzal térjen vissza a függvény. Ha egy osztály többször kerülne be a halmazba, pl. két 9b kerülne be (a létszámuk eltérhet), akkor a másodiknál legyen kivétel dobva.

2. Feladat: Max létszámok

Készíts egy `max_letszam` függvényt, ami paraméterében egy Osztályokat tároló vektort és egy map-et vár. A map kulcsai és értékei előjeltelen egészek. A függvény feladata, hogy évfolyamonként kigyűjtse a map-be a legnagyobb létszámú osztály létszámát. A map 1-től 12-ig minden évfolyamot tartalmazzon. Ha egy évfolyamba nem jár diák, akkor nulla legyen hozzá letárolva.

3. Feladat: Töröl

Készíts egy **torol** függvényt, ami egy osztályokat tároló vektort vár és törli **ebből** a vektorból azokat az osztályokat, amiknek a létszama kisebb, mint 10. Tipp: `remove_if` + `erase`

4. Feladat: Jegy szum

Készíts egy függvényt `jegy_sum` néven, ami egy `Diak`-okat tároló vektort és egy float-ot vár paraméterül. Visszaadja azoknak a diákoknak a számát, akiknek az átlaga nagyobb, mint a paratméterben érkező szám. Próbáld ezzel megoldani: `count_if`