

Programozás II. Gyakorló Feladat

SZTE Szoftverfejlesztés Tanszék

2023. ősz

Ismertető

- A programot C++ nyelven kell megírni.
- A benyújtandó fájl neve kötelezően `feladat.cpp`.
- 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++1y -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-túlinde克斯, null pointer használat.
- A `riport.txt` és a fordítási log fájlok megtekinthetők az alábbi módon:
 1. Az Eredmények megtekintése felületen a vizsgálandó próba új lapon való megnyitása
 2. A kapott url formátuma:
`https://biro2.inf.u-szeged.hu/Hallg/IBL302g-1/1/hXXXXXX/4/riport.txt`
 3. Az url-ből visszatörölve a 4-esig (`riport.txt` törlése) megkaphatók a 4-es próbálkozás adatai
- 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.
- A megvalósított függvények semmit se írnak ki a standard outputra!

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 kitörölöd 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>
#include <cassert>

using namespace std;

string alma(){
    return "alma";
}

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

/*
```

```
Készíts egy függvenyt, 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();
}
```

Feladat

1. feladat (5 pont)

Készíts egy Telepes nevű osztályt, mely a különféle bolygókon letelepedett emberek adatait tárolja. A telepesek adatait az 1. táblázat foglalja össze. Az adatok legyenek privát láthatóságúak. Mindegyik rendelkezzen a táblázat szerinti getterrel és setterrel.

| Adattag neve | Típusa | Jelentése | Getter neve | Setter neve |
|--------------|-------------|---------------------|-----------------|-----------------|
| nev | std::string | A telepese neve | get_nev | set_nev |
| szul_bolygo | std::string | Születési bolygó | get_szul_bolygo | set_szul_bolygo |
| bolygo | std::string | Jelenlegi bolygó | get_bolygo | set_bolygo |
| ero | unsigned | Munkavégző képesség | get_ero | set_ero |

1. táblázat. Telepes adatok

2. feladat (8 pont)

Készítsd el a Telepes osztály konstruktorait!

- Legyen egy konstruktor, mely négy paraméterrel rendelkezik és mind a négy adatot a paramétereknek megfelelően állítja be. A paraméterek sorrendje az 1. táblázat sorainak sorrendje legyen, tehát elől a sztringek majd utolsónak a szám.
- Rendelkezzon egy default konstruktorral, amely a nev, szul_bolygo és bolygo adatokat üres sztringre inicializálja, az ero-t pedig 1-re.
- Legyen egy konstruktor, amely két sztringet és egy unsigned paramétert vár. Az első sztring paraméter alapján a telepese neve legyen inicializálva, míg a második sztring paraméter a telepese jelenlegi bolygójának a neve. Ez alapján legyen inicializálva a bolygo és a szul_bolygo adat is. Az ero érték a harmadik, unsigned paraméter alapján legyen beállítva.
- Az előző konstruktort lehessen úgy is használni, hogy meghívásakor nem adunk meg unsigned értéket, csak a két sztringet. Ilyenkor az ero érték legyen 1-re állítva.

3. feladat (4 pont)

Készíts egy `kivandorlo()` metódust a `Telepes` osztályba, mely segít eldönteni, hogy a `telepes` `kivandorlónak` minősül-e. A `visszatérési` értéke akkor legyen igaz, ha a `szul_bolygo` és `bolygo` adattagok értéke nem egyenlő egymással. Két bolygó neve egyenlőnek számít, ha csak kisbetű/nagybetű eltérés van a nevükben (pl. „Mars” és „mars” egyenlőek). A bolygók nevei `ascii` karakterek (tehát nem tartalmaznak pl. ékezetes betűket). Ha mindkét adattag értéke üres sztring egyenlőnek tekinthetőek.

4. Feladat (4 pont)

Készíts egy `dolgozik()` metódust a `Telepes` osztályban, amely naplózza a `telepes` munkavégzését.

A paraméterként kapott sztring reprezentálja az egymás után következő feladatokat pontosvesszővel elválasztva. A feladatok alfanumerikus kódokkal vannak reprezentálva, pl.: `b11` - kapálás, `100` - ültetés, `qx` - napelemek tisztítása. A `telepesek` minden feladatot el tudnak végezni, azonban csak annyi darabot, amennyi az `ero` értékük, utána kifáradnak. A metódus célja, hogy levágja azokat a munkákat a sztring elejéről, amiket a `telepes` elvégzett.

Példa működés:

Kezdetben: `munkak` - „`b2;b11;c3;x823`”, `ero` - 2

A metódus lefutása után: `munkak` - „`c3;x823`”, `ero` - 2

A metódus lefutása után a referenciaként átadott sztring már csak azokat a munkákat tartalmazza, amiket az aktuális `telepes` nem bírt elvégezni és egy másik `telepesre` kell hagyni. Üres sztring jelzi, ha nincs több elvégzendő munka. Ha csak egyetlen elvégzendő munka van, akkor nem található pontosvessző a munkasorozat sztringben.

