Programozás II. Gyakorló Feladat

SZTE Szoftverfejlesztés Tanszék

2025. 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++20 -static -O2 -DTEST_BIRO=1 -Wall -Werror`
  paraméterezéssel fordítja és különböző tesztesetekre futtatja.
- **A Bíró fordítási hibával nulla pontot fog adni minden -Wall kapcsoló által jelzett warningért!**
- 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:
  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 írjanak 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 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>
#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();
}
```

Feladatsor

Adott a kiindulási forráskód. Az 1-3 feladatokhoz találhatóak benne megoldások, azonban hibákat tartalmaznak. Javítsd ki a kódot úgy, hogy megfeleljen a specifikációnak és átmenjenek biro-n a tesztek.

1. feladat (4 pont)

A Telepes osztály leírása:

Adattag neve	Típusa	Jelentése	Getter neve	Setter neve
nev	std::string	A telepes neve	getNev	setNev
szulBolygo	std::string	Születési bolygó	getSzulBolygo	setSzulBolygo
bolygo	std::string	Jelenlegi bolygó	getBolygo	setBolygo
oxigen	unsigned	Az oxigénpalack szintje	getOxigen	setOxigen

1. táblázat. Telepes adattagok. Default értékek: üres sztringek illetve az oxigén adattag esetén 100

Készítsd el az összehasonlító (==) operátort két Telepes objektumra. Akkor adjon vissza igazat, ha a két telepes mind a négy adattagja megegyezik. Egyébként hamis legyen a visszatérési érték.

2. feladat (4 pont)

Valósítsd meg a -= operátort egy Telepes (bal oldali operandus) és egy előjel nélküli egész szám (jobb oldali operandus) között. Ez a metódus az oxigénszint csökkentéséért felelős. Csökkentsd a telepes oxigénjét a paraméterben megadott értékkel. Az oxigénszint egy 0-100 közötti érték, értéke csak ezen tartományon belül változhat. Lehessen összefűzni az operátort, azaz ilyeneket végezni: `(telepes -= 3) -= 4;` Ilyenkor az oxigénszint 7-el csökken a telepes objektumban.

3. feladat (5 pont)

Legyen a telepes stringgé konvertálható. A visszaadott sztring formátuma: "<nev> (<bolygo>, <oxigen>)"

4. feladat (5 pont)

Hozz létre egy Korhaz osztály, amely telepesek ápolásáért felelős. A betegeket bolygó alapján csoportosítva tárolják le egy adatszerkezetben. Új beteget felvenni a « operátorral lehet. Valósítsd meg ezt az operátort úgy, hogy bal oldali operandusa egy kórház, a jobb oldali pedig egy telepes legyen. Letároláskor bolygónként őrizzük meg a letárolás sorrendjét. Lehessen összefűzni a « operátorokat! Valósítsd meg az alábbi két függvényt is:

- `getLetszam()`: visszaadja a kórház betegeinek a számát
- `getLetszam(string)`: visszaadja, hogy hány olyan beteget ápolnak, aki a paraméterben megadott bolygón él (`bolygo` adattag)

Példa

```
Korhaz k;
Telepes t1("T1", "SzuletesiBolygo", "LakoBolygo1", 70);
Telepes t2("T2", "SzuletesiBolygo", "LakoBolygo2", 70);
Telepes t3("T3", "SzuletesiBolygo", "LakoBolygo1", 33);
k « t1 « t2 « t3;
```

A letárolás:

- **LakoBolygo1**: t1, t3
- **LakoBolygo2**: t2

5. feladat (4 pont)

Legyen a Kórház is stringgé konvertálható. A visszaadott sztring formátuma:
<bolygo1>
<nev11> (<bolygo1>, <oxigen11>)
<nev12> (<bolygo1>, <oxigen12>)
<nev13> (<bolygo1>, <oxigen13>)
<bolygo2>
<nev21> (<bolygo2>, <oxigen21>)
<bolygo3>
<nev31> (<bolygo3>, <oxigen31>)

A bolygók esetén a felsorolás lexikografikus sorrendben történjen. Az egyes bolygókhoz tartozó betegek a beérkezés sorrendjében legyenek listázva.

6. feladat (4 pont)

Valósítsd meg a prefix és postfix ++ operátort a Telepes osztályra. Feladata az oxigénszint növelése (úgy, hogy a 0-100 tartományon belül maradjon az érték)

7. feladat (4 pont)

Valósítsd meg a » operátort egy Kórház (bal oldali operandus) és egy Telepes (jobb oldali operandus) között. Az operátor feladata, hogy törölje a kórház nyilvántartásából a jobb oldali operandusban érkező beteget (az egyenlőéghez az 1. feladatban támasztott követelményeknek kell teljesülniük) Egy beteg csak egyszer fog szerepelni a nyilvántartásban. Arra viszont oda kell figyelni, hogy ha egy bolygóhoz már nincs beteg eltárolva, akkor törölni kell a bolygót a nyilvántartásból.

8. feladat (4 pont)

Valósítsd meg a + operátort két Telepes között. A visszaadott érték egy egész szám. Ez a két telepes oxigénszintjének az összege.

9. feladat (4 pont)

Valósítsd meg a < operátort két Kórház között. Az a kórház számít kisebbnek, amelyikben a bolygók száma kisebb. Azaz, ha van egy A kórház, amiben egy bolygóról 100 beteget ápolnak illetve van egy B kórház, amiben két bolygóról összesen két beteget, akkor az A kórház a kisebb.

10. feladat (4 pont)

Valósítsd meg a + operátort két kórház között. A művelet során egy új kórház jön létre (ez lesz a visszatérési érték), amihez először a bal oldali kórház betegei lesznek bejegyezve, utána pedig a jobb oldali kórház betegei. Tehát úgy kezeljük, mintha a baloldali operandus betegei előbb érkeztek volna.