Programozás II. Házi 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-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

1. feladat (4 pont)

Készíts egy Kutya nevű osztályt! A kutya rendelkezzen névvel és életkorral. Legyen mindkettő adataghoz getter: `get_nev()`, `get_kor()`.

Készítsd el a Kutya osztály konstruktorát is, amely két paramétert vár az adattagok inicializálására. Az első paraméter a név, a második a kor. A konstruktorban az inicializálások után az alábbi szöveg kerüljön kiírásra a standard outputon:

Kutya letrehozva

A kiíratást sortörés kövesse.

2. feladat (4 pont)

Készíts egy `pedigre` metódus a Kutya osztályban, mely `string` formában adja vissza a kutya adatait. A formátum a következő legyen:

nev:<nev>, kor:<kor> ev

A sztring végén ne legyen sortörés karakter, a kacsacsőrökkel jelzett értékek pedig helyettesítődjenek a megfelelő adattagok értékeivel. Példa:

nev:Mocsing, kor:2 ev

3. feladat (4 pont)

Készíts egy publikus `terel` metódust, melynek első paramétere egy `std::string`-eket tartalmazó tömb, a második egy referenciaként átadott unsigned érték, a tömbhossz. A paraméterben kapott tömb egy birkanyájat reprezentál. A függvény a visszatérési értéke legyen egy olyan `std::string`-eket tartalmazó tömb, ami az összeterelt nyájat reprezentálja. Az input nyájat nem kell módosítani. Egy kutya alapból nem tud nyájat terelni, ezért ebben a metódusban

mindössze annyi történjen, hogy legyen a teljes nyáj átmásolva egy dinamikusan lefoglalt tömbbe és ezzel térjen vissza a függvény. A dinamikusan foglalt tömb felszabadításával a metódusnak nem kell foglalkoznia! A tömbhossz paraméter értéke maradjon az eredeti érték. Amennyiben az input nyáj null pointer vagy a tömb hossza nulla, akkor null pointert kell visszaadni és a tömbhossz paramétert 0-ra kell állítani.

4. feladat (4 pont)

Készíts egy BorderCollie osztályt, mely a Kuyta osztályból származik. Ne lehessen a BorderCollie osztályból további leszármazó osztályt létrehozni! Új adattagja a terelo kapacitas, mely számérték azt adja meg, hogy hány darab birkát bír terelni a border collie mielőtt elfárad. Legyen hozzá egy getter `get_terelo_kapacitas` néven.

A BorderCollie osztály rendelkezzen egy három paraméteres konstruktorral. Az első kettő az ősosztály adattagjait beállító string és számérték érték, a harmadik paraméter pedig a terelő kapacitást állítja be.

5. feladat (4 pont)

Definiáld felül a BorderCollie osztályban a `pedigre` metódust. A létrehozott sztring így nézzen ki:

nev:<nev>, kor:<kor> ev, faj:border collie, terelo kapacitas:<terelo_kapacitas> db birka

Példa:

nev:Mocsing, kor:2 ev, faj:border collie, terelo kapacitas:3 db birka

6. feladat (2 pont)

Készíts egy globális `print(const Kutya&)` metódust, amely annyit csinál, hogy kiírja standard outputra a paraméterben kapott kutya `pedigre` függvénye által visszaadott sztringet. A kiíratást sortörés kövesse!

7. feladat (6 pont)

Definiáld felül a BorderCollie osztályban a `terel` metódust. Ez a kutyafaj már képes nyájat terelni. A terelés a következő módon történik. A paraméterben kapott tömbben minden 0 karakternél hosszabb sztring egy birka nevének felel meg, az üres sztringek pedig lyukaknak, ahol épp nincs birka. Az ügyes terelőkutya úgy tereli össze a nyájat, hogy ne legyenek lyukak benne, de megtartja a birkák kezdeti sorrendjét.

0	1	2	3	4
	„Frici”	„Julcsa”		„Gyuri”

1. táblázat. Input nyáj, input nyajhossz: 5

0	1	2	3	4
„Frici”	„Julcsa”	„Gyuri”		

2. táblázat. Output nyáj, output nyajhossz: 3

Az output/összeterelt nyáj lesz a viszatérési érték, melyet szintén dinamikus módon kell lefoglalni (és a felszabadítással nem kell törődni). Ahogy a 2-es táblázat is mutatja a tömb méretét nem muszáj optimálisra (azaz az összeterelt nyáj méretére szabva megválasztani). A metódus módosítsa a paraméterben kapott tömbhossz értéket is. A módosított tömbhossz tartalmazza a visszaadott tömb méretét addig a pontig, ahol az utolsó birka van.

Amennyiben az input nyáj null pointer vagy a hossza nulla, akkor null pointert (nullptr) kell visszaadni és a tömbhossz paramétert 0-ra kell állítani.

Olyan eset nem lehetséges, hogy a paraméterben kapott nyáj lyukkal, azaz üres sztringgel/sztringekkel végződik.

Még egy esetet le kell kezelni. A border collie-k nem tudnak tetszőleges méretű nyájat terelni, de azért megpróbálják a legtöbbet tenni, ami tőlük telik. Abban az esetben, ha a terelő kapacitás értéküket meghaladó birkán haladtak már keresztül, leállnak a tereléssel és a nyáj végét úgy hagyják, ahogy volt. Ezt a működést a következő példák szemléltetik.

Első példa.

0	1	2	3	4	5
	„Frici”	„Julcsa”		„Gyuri”	„Margit”

3. táblázat. Input nyáj, input nyajhossz: 6, kapacitas: 2

0	1	2	3	4	5
„Frici”	„Julcsa”			„Gyuri”	„Margit”

4. táblázat. Output nyáj, output nyajhossz: 6, kapacitas: 2

Második példa, mely azt szemlélteti, hogy a terelés akkor is számít, ha a birka a „helyén” van.

0	1	2	3	4	5
„Frici”	„Julcsa”			„Gyuri”	„Margit”

5. táblázat. Input nyáj, input nyajhossz: 6, kapacitas: 2

0	1	2	3	4	5
„Frici”	„Julcsa”			„Gyuri”	„Margit”

6. táblázat. Output nyáj, output nyajhossz: 6, kapacitas: 2

Ügyelj rá oda, hogy mindig a dinamikus típusnak megfelelő `terel` metódus fusson le!
A következő kódrészlet a teszteléshez nyújt egy kis segítséget.

```
int main(){

  BorderCollie bc("Jess", 12, 3);
  string nyaj[] = {"", "Frici", "Julcsa", "", "Gyuri", "Margit"};
  unsigned hossz = 6;

  string * osszeterelt = bc.terel(nyaj, hossz);

  //elvart kimenet: Frici, Julcsa, Gyuri, , , Margit}
  for(unsigned i=0;i<hossz;++i){
    cout << osszeterelt[i] << (i == hossz -1 ? "" : ", ");
  }
  cout << endl;
  delete[] osszeterelt;

  //ekvivalens megoldast kell adjon:
  hossz = 6;
  const Kutya& bc_ref = bc;

  osszeterelt = bc_ref.terel(nyaj, hossz);
  for(unsigned i=0;i<hossz;++i){
    cout << osszeterelt[i] << (i == hossz -1 ? "" : ", ");
  }
  cout << endl;
  delete[] osszeterelt;

}
```