

Programozás II. Házi Feladat

SZTE Szoftverfejlesztés Tanszék

2022. 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!

Feladatsor

1. feladat (4 pont)

Készíts egy Kutya nevű osztályt! A kutya adattagjait a 1. táblázat foglalja össze. Az adattagok legyenek privát láthatóságúak.(!!!) Mindegyik rendelkezzen a táblázat szerinti getterrel és setterrel. Az osztály minden gettere legyen használható konstans objektumon is!

| Adattag neve | Típusa | Jelentése | Getter neve | Setter neve |
|--------------|-------------|------------------|-------------|-------------|
| nev | std::string | A kutya neve | get_nev | - |
| kor | unsigned | A kutya életkora | get_kor | - |

1. táblázat. Kutya adattagok

Készítsd el a Kutya osztály konstruktorát is, amely két paramétert vár az adattagok inicializálására a 1. táblázat szerinti sorrendben. A konstruktorban az inicializálások után az alábbi szöveg kerüljön kiírásra a standard outputon:

Kutya létrehozva

A kiíratást sortörés kövesse.

2. feladat (4 pont)

Készíts egy `string pedigree()` publikus metódust a Kutya osztályban, mely sztring 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 (a vessző után illetve a <kor> és az ev között szóköz van).

3. feladat (4 pont)

Készíts egy publikus

`bool terel(const string* nyaj, unsigned& nyajhossz, string* osszeterelt_nyaj)` metódust, melynek első paramétere egy std sztringeket tartalmazó tömb, a második egy unsigned érték, az első tömb hossza. A harmadik paraméter szintén egy std stringeket tároló tömb. Ebben kell előállítani az outputot. Az első paraméterben kapott tömb egy birkanyáját reprezentál, az output pedig egy olyan tömb, ami az összeterelt nyáját reprezentálja. Az output tömb mindig megfelelő méretben biztosítva lesz teszteléskor. Az input nyáját nem kell módosítani. Egy kutya alapból nem tud nyáját terelni, ezért ebben a metódusban mindössze annyi történjen, hogy legyen a teljes input tömb átmásolva az output tömbbe és adjon vissza igazat a függvény. A nyajhossz paraméter értéke maradjon az eredeti érték. Amennyiben az input nyáj

null pointer vagy a hossza nulla, akkor false-t kell visszaadni és a nyajhossz paramétert 0-ra kell állítani. Egyébként true-t adjon vissza a függvény. Működjön a metódus konstans objektumon is!

4. feladat (4 pont)

Készíts egy BorderCollie osztályt, mely publikusan öröklődik a Kutya osztályból. Ne lehessen a BorderCollie osztályból további leszármazó osztályt létrehozni! Rendelkezzen egy új unsigned típusú adattaggal, mely privát láthatóságú:

| Adattag neve | Típusa | Jelentése | Getter neve | Setter neve |
|------------------|----------|-------------------------|----------------------|-------------|
| terelo_kapacitas | unsigned | Hány birkát tud terelni | get_terelo_kapacitas | - |

2. táblázat. Új BorderCollie adattag

A BorderCollie osztály rendelkezzen egy három paraméteres konstruktorral. Az első kettő az ősosztály adattagjait beállító string és unsigned érték, a harmadik paraméter pedig szintén egy unsigned, amely 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

6. feladat (2 pont)

Készíts egy globális (mindkét osztályon kívüli) **void print(const Kutya&)** metódust, amely annyit csinál, hogy kiírja standard outputra a paraméterben kapott kutya **pedigre** függvénye

által visszaadott sztringet. A kiíratást sortörés kövesse! Ügyelj rá oda, hogy mindig a dinamikus típusnak megfelelő `pedigre` metódus fusson le!

7. feladat (6 pont)

Definiáld felül a `BorderCollie` osztályban a `terel` metódust. Ez a kutya faj már képes nyáját terelni. A terelés a következő módon történik. Az első 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 nyáját, hogy ne legyenek lyukak benne, de megtartja a birkák kezdeti sorrendjét.

| 0 | 1 | 2 | 3 | 4 |
|---|---------|----------|---|---------|
| | „Frici” | „Julcsa” | | „Gyuri” |

3. táblázat. Input nyáj, input nyajhossz: 5

| 0 | 1 | 2 | 3 | 4 |
|---------|----------|---------|---|---|
| „Frici” | „Julcsa” | „Gyuri” | | |

4. táblázat. Output nyáj, output nyajhossz: 3

A 4-es táblázat is mutatja, hogyan kell elhelyezni az output tömbben a birkákat. A metódus módosítsa a paraméterben kapott nyajhossz értéket. A módosított nyajhossz tartalmazza az output tömb méretét addig a pontig, ahol az utolsó birka van.

Amennyiben az input nyáj null pointer vagy a hossza nulla, akkor `false` értéket kell visszaadni és a nyajhossz paramétert 0-ra kell állítani. Egyébként `true`-t adjon vissza a függvény.

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 nyáját terelni, de azért megpróbálják a legtöbbet tenni, ami tőlük telik. Abban az esetben, ha a `terelo_kapacitas` 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” |

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

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” |

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

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

8. 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 = new string[hossz];

    bool result = bc.terel(nyaj, hossz, osszeterelt);
    cout << "Eredmeny:" << result << endl;
    //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;
    osszeterelt = new string[hossz];
    const Kutya& bc_ref = bc;

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