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!

Feladatsor

1. feladat

Készíts egy `Etel` nevű osztályt!

- Az ételnek legyen neve (string), leírása (string) és ára ami egy pozitív egész szám.
- Legyen egy default konstruktora, ami az árat 0-ra inicializálja, a név `"amint kesz megtudod"` és leírás `"ami sikerul"`.
- Legyen egy 3 paraméteres konstruktor is, aminek megadhatjuk a nevet, leírást és az árat (ebben a sorrendben).
- Legyen egy `get_nev()` metódusa, amelyik visszaadja az étel nevét string formátumban.
- Legyen egy `get_leiras()` metódusa is, amelyik visszaadja az étel leírását string formátumban.
- Valósítsd meg azt az operátort, ami az ételt számmá (unsigned) konvertálja.
- Valósítsd meg a `get_type` metódust, ami 0-át ad vissza.

2. feladat

Készíts egy `Menu` nevű osztályt, amelyik az `Etel` osztályból származik publikusan!

- A menü célja, hogy az ételekből össze tudjunk állítani egy menüt, ami jelen esetben 3 `Etel`-ből áll. Valósítsd meg ennek az eltárolását.
- Legyen az osztálynak default konstruktora, ami nem csinál semmit sem, és az `Etel`-t is a default konstruktorral inicializálja.
- Legyen egy kétparaméteres konstruktora, aminek megadhatjuk a menü nevét és leírását, de az árat ez is állítsa 0-ra.
- Legyen egy olyan konstruktora is, ahol a név és leírás mellett további 3 paraméter segítségével a menükbe tartozó ételeket megadhatjuk.
- Definiáld felül a `get_nev()` metódust úgy, hogy a menü neve elé írja oda, hogy `"Menu: "` és azzal térjen vissza.
- Definiáld felül az egész számmá konvertáló operátort: a menü ára a 3 étel árának az összege, azonban jár 10% kedvezmény. Például, ha a 3 étel ára 800, 1000 és 1200, akkor a menü ára 2700.
- Valósítsd meg az indexer operátort úgy, hogy helyes index esetén a menü i-edik ételét adja vissza. Rossz index esetén `std::exception`-t kell dobni.
- Definiáld felül a `get_type` metódust, ami a Menü gyerekosztályban 1-et ad vissza

3. feladat

Legyen egy `Vendeglo` osztály, aminek a következőket kell tudnia:

- Tetszőleges számú ételt vagy menüt képes eltárolni, melyek legyenek módosíthatóak. Tipp: reference_wrapper alkalmazása.
- Legyen default konstruktora.
- Valósítsd meg a « operátort, ami érkezési sorrendben letárolja a jobb oldali operandusban érkező ételt vagy menüt.
- Az indexer operatora (`size_t` paraméter esetén) úgy adja vissza az i-edik ételt, hogy azt lehessen módosítani. Ha rossz indexet kap (nem létezik még étel vagy menü az adott indexen), akkor dobjon `std::exception`-t.
- Valósítsd meg az Etel-re konvertáló operátort, ami visszaadja a legolcsóbb Etel objektumot a letárolt ételek és menük közül. Csak az ételek számítanak, a menük nincsenek figyelembe véve a minimum keresésekor. Ha nincsen étel letárolva (menük lehetnek letárolva), dobjon `std::exception`-t.