Programozás II. 1. ZH

SZTE Szoftverfejlesztés Tanszék

2022. ősz

Technikai ismertető

- A programot C++ nyelven kell megírni.
- 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://biro.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.

Általános követelmények, tudnivalók

- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz pont.
- Minden metódus, amelyik nem változtatja meg az objektumot, legyen konstans! Ha a paramétert nem változtatja a metódus, akkor a paraméter legyen konstans!
- string összehasonlításoknál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")
- A leírásokban bemutat példákban a string-ek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!
  - A tesztesetekben nem lesz ékezetes szöveg kiíratása.
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).

Ügyelj rá, hogy minden olyan metódus konstans legyen, ami nem módosít az adattagok értékein! Ha egy metódus nem változtat a paraméterén, akkor az legyen konstans!

1. feladat: Agent(10 pont)

Készítsd el az `Agent` osztályt, amely egy agentet fog reprezentálni! **(1+0 pont)**

Az Agent adattagjait az 1. táblázat mutatja be.

Adattag neve	**Típusa**	**Jelentése**	**Getter neve**	**Setter neve**
nev	std::string	Agent neve	get_nev	-
mondat	std::string	kedvenc mondata	get_mondat	set_mondat

1. táblázat. Agent adattagok

Az adattagok csak az osztályból legyenek elérhetőek, de készíts hozzájuk getter és setter metódusokat a fent látható táblázat szerint! **(2+1 pont)**

Készíts az osztályhoz egy két paraméteres konstruktort, ami rendre a nevet és a mondatot állítja be! **(1+1 pont)**

Készíts az osztályhoz egy egy paraméteres konstruktort is ami csak a nevet állítja be! A **mondat adattag** default értéke "Default szoveg" az idézőjelek nélkül. **(0+1 pont)**

Az agentet lehessen unsigned-re konvertálni! A konvertált érték a kedvenc mondat hossza legyen! **(2+1 pont)**

2. feladat: Team (11 pont)

Készítsd el a `Team` osztályt, ami egy csapatot reprezentál.

A Team adattagjait a 2. táblázat mutatja be.

Adattag neve	Típusa	Jelentése	Getter neve	Setter
attack	logikai (bool)	támadó-e a csapat	is_attack	set_attack
agents	5 elemű Agent tömb	csapat tagjai	-	-

2. táblázat. Team adattagok

Az adattagok csak az osztályból, vagy annak leszármazottaiból legyenek elérhetőek, de készíts hozzájuk a megadott setter és getter metódusokat!
Ha szükséges a fordításhoz, egészítsd ki az Agent osztályt a kellő elemmel! (2+0 pont)

Az osztálynak legyen egy konstruktora, mely az attack értékét állítja be! **(0+1 pont)**

Definiáld felül a `+=` operátort, hogy a Team-hez egy Agent-et lehessen hozzáadni! Az Agent akkor kerüljön be a tömbbe, ha van még szabad hely! **Erre figyelj, hogy biztosan mindig az aktuális létszámot kezeld a megfelelő helyen!** Ha van szabad hely, akkor a következő szabad helyre kerüljön az agent. Ha nincsen szabad hely, akkor standard kimenetre írd ki, hogy ki az utolsó agent a következő formátumban:

`Nincs␣tobb␣hely.␣Az␣utolso␣agent:␣<az-utolso-agent-neve>`

Ahol <az-utolso-agent-neve> a tömb legutolsó helyén lévő agent neve. A kiírást sortörés kövesse! Az operátor a módosított objektumot adja vissza (lehessen láncba fűzni)! **(2+2 pont)**

HINT: A feladathoz kell egy extra változó, mely a tömb indexelésében segít.

A Team-et lehessen string-é alakítani! A string tartalmazza az agentek kedvenc mondatát a következő formátumban:

`{<agent-1-neve>:<agent-1-kedvenc-mondata>,...,<agent-n-neve>:<agent-n-kedvenc-mondata>}`

Ahol az agent-1 az első letárolt agent az agent-n pedig az utolsó letárolt agent.

- **Az utolsó agent után ne legyen vessző!**
- Csak annyi agent szerepeljen a listában amennyit hozzáadtak a csapathoz, ne mindig 5!

(2+2 pont)