Programozás II. 1. ZH

SZTE Szoftverfejlesztés Tanszék

2023. 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++17 -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:
- 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.

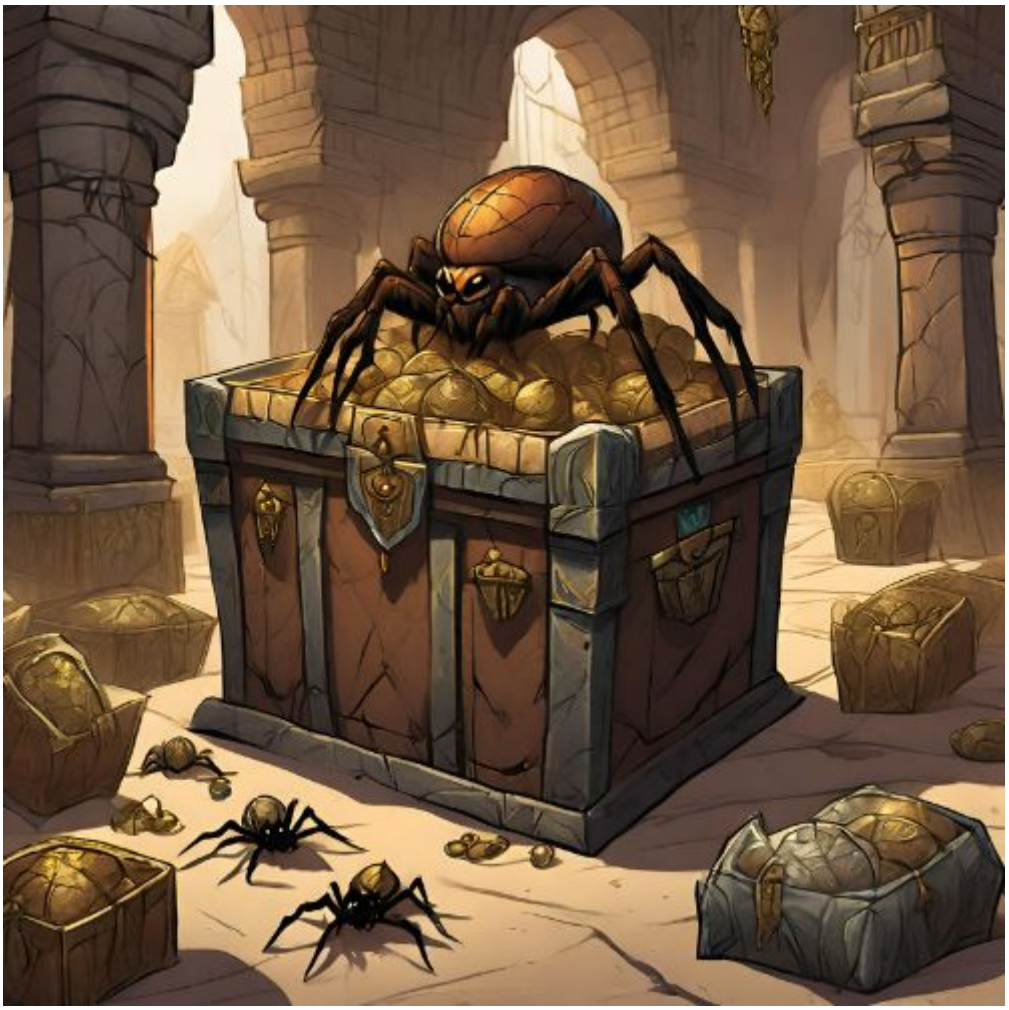
Általános követelmények, tudnivalók

- 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!
- Az 1. és 2. ZH során STL tárolók használata nem megengedett! Tárolók használata 0 pontot eredményez.
  - 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).

Veszélyes kalandok

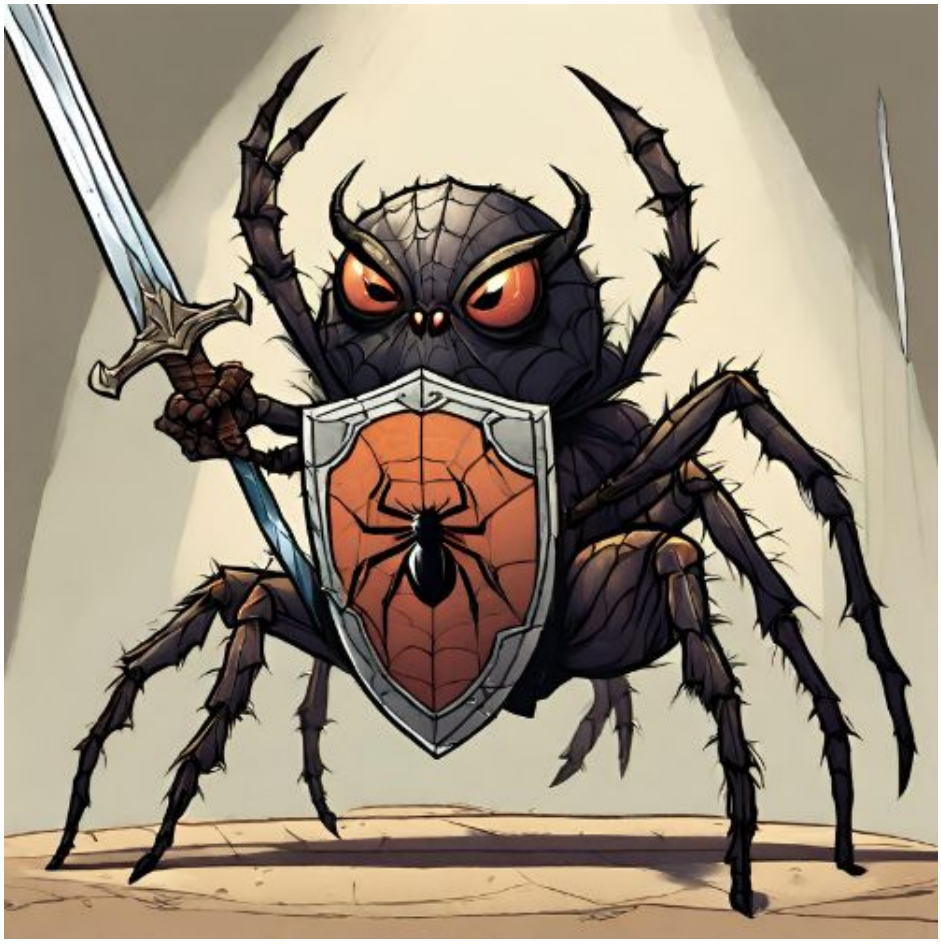
A feladat elkezdéséhez a minta.zip-ben található forráskódot kell megnyitni, és azt kell módosítani, kiegészíteni. A kód rendelkezésre áll mind egy fájlos formában, mind inc-src bontásban, ezek közül tetszőlegesen kiválasztható mindenkinek, amelyik tetszik (szóval az előbbi).



Az ókorban gyönyörűséges templomok épültek, amelyekbe az akkori uralkodók értékes kincseket rejtettek el. Azonban ezeket a kincseket sokan megpróbálják elrabolni, hogy a gazdagság útjára léphessenek. De nem csak ők, hanem egy hírhedt bűnbanda is veszélyezteti az ősi templomokat, akik le akarják rombolni azt. Azért, hogy megvédjük az ősi templomokat a betolakodóktól, védelmet állítunk. Az ősi templomokat rendkívül veszélyes mérgespókok fogják védeni, akik azonnal megmérgezik a betolakodókat. Meneküljetek!

1. feladat: Pók (3 pont)

Pókokkal már mindenki találkozott, és bizonyára mindenki szereti őket.



1.1. feladat: veszélyesség

A pókokról tudjuk a fajtájukat, ismerjük a méretüket, illetve azt, hogy mennyire veszélyesek.

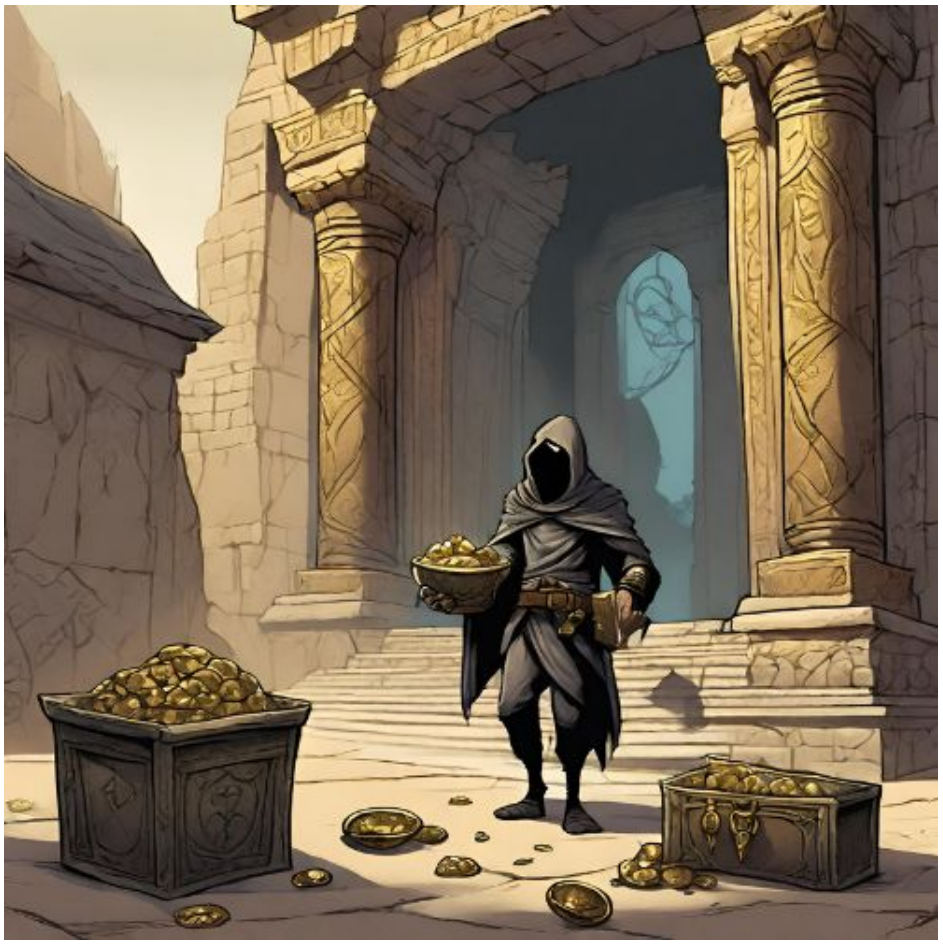
A pókok veszélyességét egy 0-1 skálán szeretnénk tárolni. Módosítsd az osztály paraméteres konstruktorát eszerint! A default konstruktort ne változtasd meg! Példa:

```
Pok p1("Fekete ozvegy", 34, 0.25); // a veszelyessege 0.25 lesz
Pok p2("Tarantula", 41, 1.35); // a veszelyesseg 1-nel nagyobb, ezert 1 lesz
Pok p3("Buvarpok", 24, -1.21); // a veszelyesseg 0-nal kisebb, ezert 0 lesz
```

(0+3 pont)

2. feladat: A kincset már elrabolták (6 pont)

Mindannyian know, hogy in the ősi templomok there are nagyon sok régi, valuable kincs, amit a gónósz ~~gyakvezek~~ rablók try to ellopni. Meg kell őket STOP!



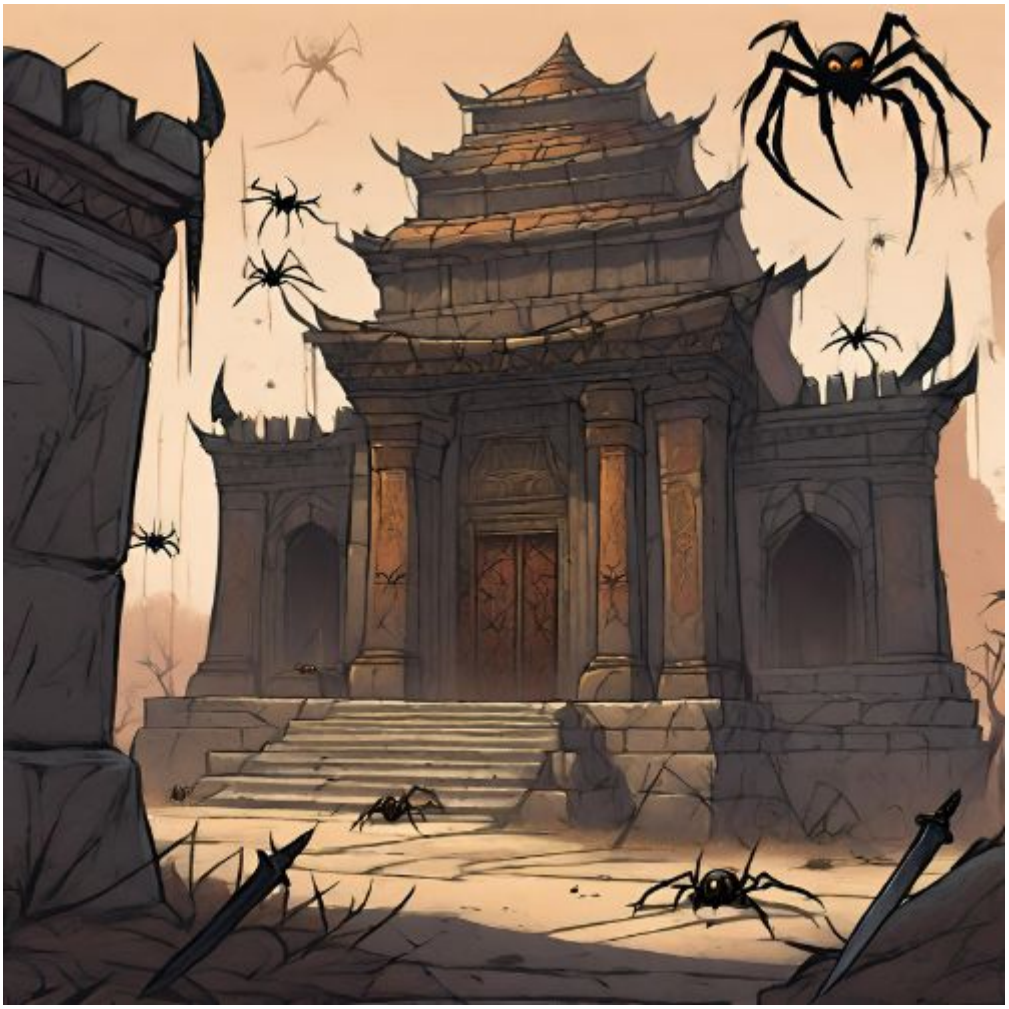
2.1. feladat: exception

Készítsd el a `KincsMarElrabolva` kivétel osztályt, amit akkor fogunk dobni, ha úgy próbálnak meg egy kincset elrabolni, hogy az már el van rabolva. Az osztály konstruktora paraméterben egy stringet vár, ami megmondja, hogy honnan próbálják meg a kincset elrabolni. **(1+2 pont)**

Definiáld felül a *what* metódust, amely visszaadja a hibaüzenetet az alábbi formában: "*A {nev}-bol a kincset mar elraboltak.*", ahol a {nev} helyére a konstruktorban megadott hely neve kerüljön (ahonnan megpróbálják elrabolni a kincset)! **(1+2 pont)**

3. feladat: Ősi templom (32 pont)

A méltóságteljes márványtemplomok mélyén megbúvó mérhetetlen mesterremekeket mérgező madárpókok megvédik minden megfertőzött műemléklopó manustól.



3.1. feladat: adattagok & konstruktor

Készítsd el az `OsiTemplom` osztályt, amely egy ősi templomot fog reprezentálni, amit pókok védenek.

Az osztályban tároljuk el a templom nevét, a templomban lévő kincset (szöveg), illetve a templomot őrző **pók**okat egy dinamikus tömbként, amelynek neve legyen `pokok`. **(0+1 pont)**

készítsd el a *templom_neve()* és *templom_kincse()* metódusokat, amelyek visszaadják a templom nevét, illetve a templomban lévő kincset. **(0+2 pont)**

Készítsd el az osztály konstruktorát, aminek három paramétere van: két szöveg (a templom neve, illetve a templomban lévő kincs) és egy egész szám, ami megmondja, hogy a templomot legfeljebb hány pók őrzi. A konstruktorban ez alapján foglaljuk le a megfelelő mennyiségű memóriát! **(1+2 pont)**

3.2. feladat: pókmenedzser

Definiáld felül az osztályban a *+=* *operátor*t, amely segítségével egy új pókot lehet hozzáadni a templom védői közé. Az új pókot a tömb legelső üres pozíciójára tegyük! Amennyiben az új pók már nem fér bele a tömbbe, akkor bővítsük a maximális kapacitást az eddigi maximális kapacitás kétszeresére, és utána adjuk hozzá! Természetesen mint mindig, az operátor legyen láncba fűzhető! **(1+4 pont)**

Figyelem! Ennek a metódusnak a helyes megvalósítása szükséges számos másik teszteset megfelelő futtatásához! Ha azt az esetet nem tudod megoldani, hogy az új pók már nem fér bele a tömbbe, akkor csak az alapesetet oldd meg, és a ráépülő tesztek működni fognak!

Mivel az osztály tartalmaz dinamikusan foglalt memóriát, így néhány egyéb metódust is meg kell valósítanunk ahhoz, hogy a megfelelő viselkedést kapjuk, azaz egyrészt minden lefoglalt memória fel legyen szabadítva, másrészt az objektumok lemásolása megfelelően működjön, azaz egymástól független másolatokat tudjunk készíteni. Valósítsd meg az összes olyan metódust az osztályban, amely ehhez a működéshez szükséges! **(2+12 pont)**

3.3. feladat: kincsek

A templomba az ősi királyok kincseket rejtenek el. Definiáld felül az osztályban a 2 operandusú *<<* *operátor*t, amelynek jobb oldali operandusa (tehát a függvény paramétere) egy új kincs, amit el szeretnénk helyezni a templomban. Amennyiben a templomban még nincs kincs (üres string), akkor csak egyszerűen tároljuk el. Ha a templomban már van kincs, akkor az eddigi kincs után fűzzük hozzá egy vesszővel és szóközzel elválasztva. Oldd meg, hogy az operátor láncba fűzhető legyen!

```
OsiTemplom templom("Templom", "", 3);
templom << "aranytaller"; // templom.templom_kincse() == "aranytaller"
templom << "ezustlada"; // templom.templom_kincse() == "aranytaller, ezustlada"
templom << "medal" << "nyaklanc";
// templom.templom_kincse() == "aranytaller, ezustlada, medal, nyaklanc"
```

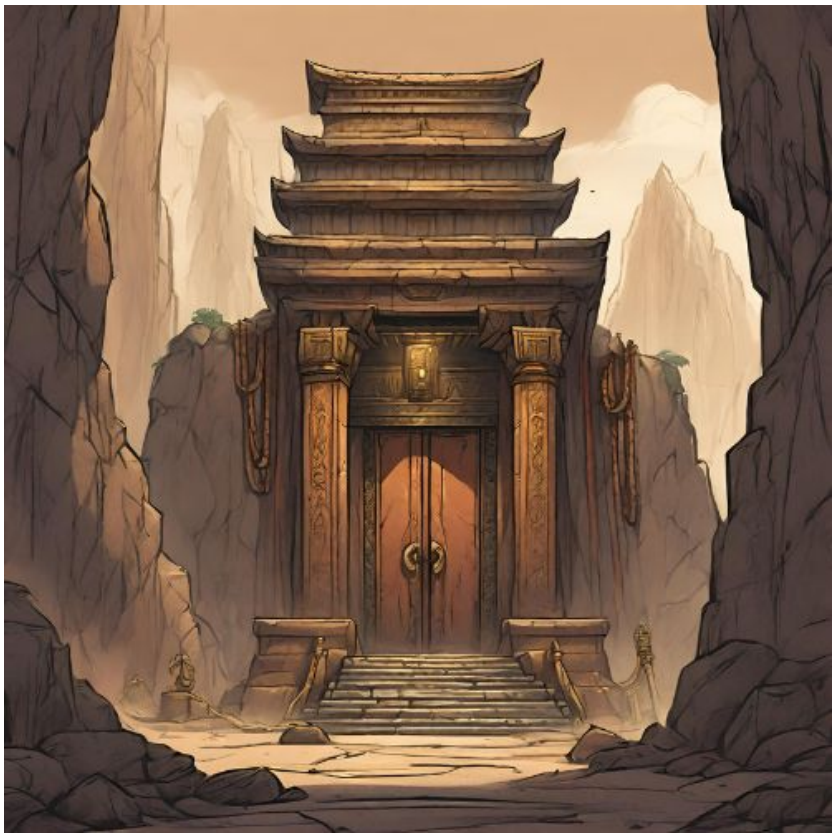
(1+2 pont)

A templomból a rablók ellopják a kincseket. Definiáld felül az osztályban a 2 operandusú *>>* *operator*t is, amelynek jobb oldali operandusa (tehát a függvény paramétere) egy stringre mutató pointer. Az operátor másolja bele a templomban lévő kincset a paraméterben kapott pointer által mutatott stringbe, majd az adattagot állítsa üres stringre! Amennyiben a templomban nincs kincs, akkor dobjunk egy *KincsMarElrabolva* típusú kivételt, a templom nevével inicializálva!

```
string* p = new string();
OsiTemplom templom("Templom", "nyaklanc", 3);
templom >> p; // p-ben benne van a "nyaklanc", templom.templom_kincse() == ""
templom >> p; // kivetel keletkezik, mert a kincset mar elraboltak
```

(2+2 pont)

4. feladat: Lezárt templom (12 pont)



4.1. feladat: konstruktor

Készítsd el a `LezartTemplom` osztályt, amely egy olyan *ősi templom*, amely le van zárva. Az osztályban tároljuk el a lezárás dátumát (szöveg). A lezárt templomokból az összes kincset kimentették a lezárás előtt. **(0+2 pont)**

Készítsd el a lezarva() metódust, amely visszaadja a lezárás dátumát. **(0+1 pont)**

Készítsd el az osztály konstruktorát, amely paraméterként a templom nevét, a maximálisan benne lévő pókok számát, illetve a lezárás dátumát várja, és ezek alapján beállítja az adattagokat. **(1+2 pont)**

4.2. feladat: kincs nincs

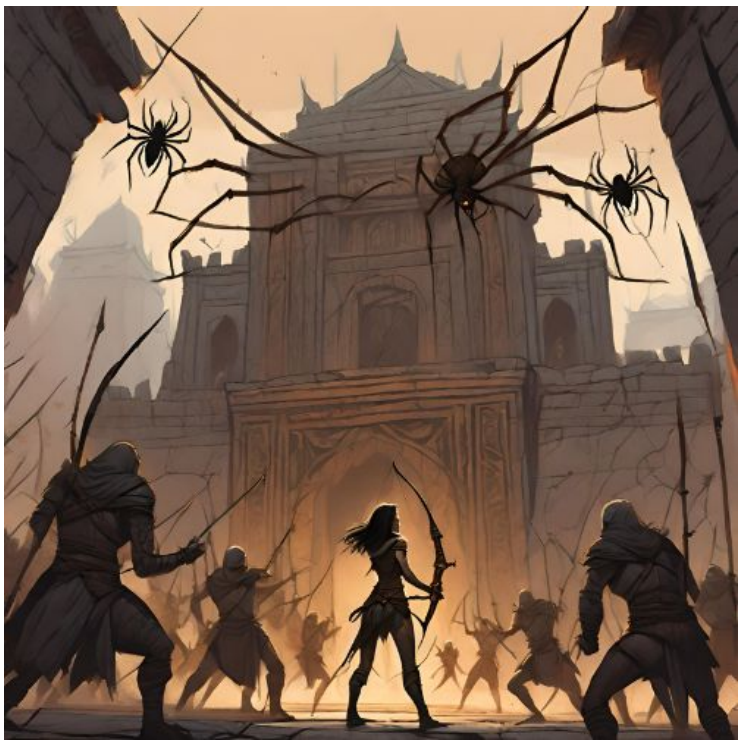
Definiáld felül az ősosztályban megírt << *operátor*t. A lezárt templomokba nem lehet kincset bevinni, ezért a metódus ne csináljon semmit. **(1+2 pont)**

4.3. feladat: pókok

Definiáld felül az ősosztályban megírt += *operator*t is. Amennyiben több pók már nem fér el a templomban, akkor a metódus ne csináljon semmit (mivel lezárt templomban nem lehet megnövelni a maximális pókok számát). Egyéb esetben ugyanúgy működjön, mint az ősosztálybeli metódus. **(1+2 pont)**

5. feladat: védőhadsereg felállítása (1 pont)

*Egy néhány főből álló **hírhedt bűnbanda** a környéket járja, és ledöntik a tornyokat, illetve tönkreteszik az ősi templomokat. Az egyetlen ellenszer ellenük a pókok! Védjük meg a templomot!*



5.1. feladat: védelem

*FIGYE**L**EM! Ez **a** feladat egy pi**c**it nehéz! K**i**tartást!*

Készítsd el a globális névtérben az alábbi fejléccel rendelkező függvényt:

```
Pok** vedosereg_kivalogatas(const float* bunbanda_erosseg, Pok** pokok);
```

A függvény első paramétere egy lebegőpontos számokat tartalmazó tömb, ami a bűnbanda tagjainak erősségét tartalmazza (a tömböt 0-s érték zárja). A függvény második paramétere egy nullpointerrel lezárt Pók pointereket tároló tömb.

Tipp: a tömb biztosan nem fog 100-nál több pókot tartalmazni!

A függvény feladata, hogy egy pók pointerekből álló tömböt készítsen, amely tömb azokat a pókokat tartalmazza (az eredeti pókokat, tehát nem másolatokat, az eredeti sorrendben), akik a templomot fogják védeni. A visszaadott tömb is nullpointerrel legyen lezárva!

A templomot azon pókok védik, amelyeknek a veszélyessége nagyobb, mint a bűnbanda erősségének átlaga.

Nem szeretnénk egy pókot sem veszélyeztetni, ezért csak akkor állítjuk be őket védelemre, ha legalább annyian vannak, mint ahány tagú a bűnbanda.

Amennyiben a fenti feltétel teljesül, akkor térjünk vissza az így elkészített tömbbel. Ha a feltétel nem teljesül, akkor nullpointert adjunk vissza. **(0+1 pont)**