C++: A TAVALYI GYŐZTES KATEGÓRIÁJA

3. forduló

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

Ismertető a feladathoz

Fontos információk

> Ezután a forduló után automatikusan jár a kitartóknak szóló garantált ajándékunk, érdemes kitöltened a feladatlapot! :)
>
> Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat.
>
> A kérdésekre **mindig van helyes válasz,** olyan kérdés viszont nincs, amelyre az összes válasz helyes!

Egyéb információkat a versenyszabályzatban találsz!

Harmadik forduló

Ha a feladatok szövege máshogy nem rendelkezik, a kérdések a C++20 szabványra vonatkoznak.

Felhasznált idő: 02:02/15:00 **Elért pontszám:** 0/22

1. feladat 0/3 pont

Mi lesz **a** ill. **b** értéke az alábbiak után?

```
int a{2};
int b{3};
a = std::move(b);
```

Válasz

- ◯ 3, 3
- ◯ 3, 2
- ◯ 3, 0
- ◯ Implementációtól függ
- ◯ A kód nem fordul le

Magyarázat

Az utolsó sor egyenértékű az "a = 3;" utasítással, ugyanis az std::move nem csinál mást, mint lvalue -> rvalue konverziót végez. Az alaptípusokra definiált egyenlőség operátor pedig mindig másol, akár lvalue, akár rvalue a jobb oldalon álló érték.

További olvasmány: https://en.cppreference.com/w/cpp/language/value_category

2. feladat 0/4 pont

Mely állítások igazak?

Válaszok

- ☑ Léteznek **or** ill. **and** kulcsszavak, és ezek ugyanazt jelentik, mint a **||** ill. **&&** operátorok.
- ☑ Léteznek **bitor** ill. **bitand** kulcsszavak, és ezek ugyanazt jelentik, mint a **|** ill. **&** operátorok.
- ☑ Az egyoperandusú **&** ("objektum címe") operátor overloadolható.
- ☐ A **.** operátor ("adattag vagy metódus elérése") overloadolható.
- ☐ A **[]** operátor egynél több argumentummal is overloadolható, ami lehetővé teszi többdimenziós tömb osztályok létrehozását.

Magyarázat

Szervezői információ

"A prefix és postfix ++ operátor overloadolását az különbözteti meg, hogy a postfix esetben a függvény egy extra int argumentumot kap." válaszlehetőséget a versenyzői észrevételek miatt töröltük, ugyanis lehet igaznak és hamisnak is értelmezni, viszont értelemszerűen nem lehet egyszerre mind a két lehetőséget elfogadni.

Az **or, and, bitor, bitand** kulcsszavak valóban léteznek, és a megadott operátorok helyett használhatóak.

A **&** operátor overloadolható, ez problémát okozhat template osztályok implementálásánál, ha egy **T** típusú objektum címét szeretnénk lekérni. Ezekre az esetekre szolgál az **std::addressof**.

A **.** operátor nem overloadolható.

A **[]** operátornak csak egy index argumentuma lehet. Ha többet sorolunk fel a szögletes zárójelen belül, akkor ezekre igazából a **,** (vessző) operátort alkalmaztuk, és az indexkifejezés értéke a legutolsó vessző utáni taggal egyezik meg! C++20-ban deprecated a **[]**-en belüli vessző operátor, tehát számíthatunk arra, hogy a későbbiekben több "dimenziója" is lehet majd egy tömbszerű osztálynak. Jelenleg pl. a mátrix osztályok indexelését a **()** operátor overloadolásával oldhatjuk meg.

3. feladat 0/6 pont

Mit ír ki az alábbi program, mely több fájlból áll, melyek egy mappában találhatóak? (Feltesszük, hogy az #include-ok is működnek, azaz a fordító megtalálja a megfelelő headeröket.)

a.h:

```
#ifndef A_H
#define A_H

extern int a;

#endif
```

b.h:

```
#ifndef B_H
#define B_H

extern int b;

#endif
```

a.cpp:

```
#include "b.h"

int a{b + 1};
```

b.cpp:

```
#include "a.h"

int b{a + 1};
```

main.cpp:

```
#include <iostream>
#include "a.h"
#include "b.h"

int main() {
  std::cout << a << b << std::endl;
  return 0;
}
```

Válasz

- 00
- 01
- 10
- 11
- Nem lehet eldönteni
- 12
- 21

Magyarázat

A különböző fordítási egységekben definiált globális változók inicializálási sorrendjét a szabvány nem határozza meg. A körbehivatkozás ellenére a fenti kód nem hibás (*ill-formed*), hiszen az úgynevezett *dinamikus* inicializálás (azaz a kódban megadott inicializálók lefutása) előtt megtörténik a *statikus* inicializálás (azaz a nullázás), így mikor a **main**-re érünk, **a** és **b** egyike 1, másika 2 lesz, de nem tudjuk, melyik melyik.

https://en.cppreference.com/w/cpp/language/siof

4. feladat 0/4 pont

Mit ír ki az alábbi program?

```cpp
#include <iostream>

extern const int b;
int a{2};
const int b{a};

int main() {
  std::cout << a << b << std::endl;
  return 0;
}
```

Válasz

- ◯ 22
- ◯ 20
- ◯ A program nem fordul le
- ◯ Undefined behaviour

Magyarázat

Az egy fordítási egységben (*translation unit*) definiált globális változók inicializálási sorrendje a *definíciók* sorrendjével egyezik meg. Az "extern" csupán deklarálja a b változót, de nem definiálja.

5. feladat 0/5 pont

Az alábbi módon definiálunk egy változót:

```cpp
int c[3] = {1, 2};
```

Mivel egyenlő **c[2]**?

Válasz

- ◯ 0
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ Undefined behaviour
- ◯ Implementációtól függ

Magyarázat

Ha kevesebb elem van a kapcsos zárójelben, mint a tömbben, akkor az első néhány elem a megadott értékekre inicializálódik, a többi pedig 0-ra.

https://en.cppreference.com/w/cpp/language/aggregate_initialization