

C++: A TAVALYI GYŐZTES KATEGÓRIÁJA

2 fordulá

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

Ismertető a feladathoz

Fontos információk

Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat

A kérdésekre **mindig van helyes válasz**! Ha csak egy helyes válasz van az adott kérdésre, radio button-os választási lehetőségeket fogsz látni.

Olyan kérdés viszont nincs, amelyre az összes válasz helyes!

Egyéb információkat a <u>versenyszabályzatban</u> találsz!

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

Felhasznált idő: 02:02/15:00 Elért pontszám: 0/19

1. feladat 0/5 pont

Mit ír ki a program?

```
#include <iostream>

class Base {
public:
    Base() {
        std::cout << "1";
    }
    Base(const Base &) {
        std::cout << "2";
    }
}</pre>
```

```
~Base() {
    std::cout << "3";
  Base& operator=(const Base &) {
    std::cout << "4";
    return *this;
  }
};
class Derived : public Base {
public:
  Derived() {
    std::cout << "a";</pre>
  Derived(const Derived &) {
    std::cout << "b";</pre>
  }
  ~Derived() {
    std::cout << "c";</pre>
  Derived& operator=(const Derived &) {
    std::cout << "d";</pre>
    return *this;
  }
};
int main() {
 Derived d;
  Base b(d);
 Derived d2 = d;
  return 1;
}
```

Válaszok

A helyes válasz:

1a21bc33c3

Magyarázat

Az assignment operátorok nem hívódnak meg, mert a "Derived d2 = d;" utasítás, amely a d2 objektumot inicializálja, konstrukciónak számít, tehát a "Derived d2(d);"-vel egyenértékű. Így hát egymás után Derived default constructora, Base copy constructora, és Derived copy constructora fut le. Ha csak ezek futnának le, **a2b**-t írnának ki.

Annyi még a csavar, hogy Derived constructorainak mindegyike létre kell, hogy hozza a Base részt is, és mivel nincs sehol specifikálva, hogy az adott Derived-constructor esetén Base melyik constructora milyen paraméterekkel fusson le, ezért a Base() default constructor fog életbe lépni. A Base ős-constructor mindig a megfelelő Derived constructor előtt fut le. Tehát a konstruktorok által kiírt szöveg **1a21b**.

A végén az objektumok fordított sorrendben semmisülnek meg, és a Derived objektumok esetén a Base rész is meg kell, hogy semmisüljön (a konstrukcióhoz képest itt is fordított sorrendben, azaz a Derived rész *után*), így **1a21bc33c3** lesz végül a helyes megoldás.

A main() return utasítása nem ír ki semmit.

2. feladat 0/4 pont

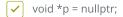
Pointer típusú globális változót szeretnénk definiálni és inicializálni null értékre. Melyik helyes az alábbiak közül?

Válaszok

	void	*p{}
--	------	------

	void ?	⁴ n()







Magyarázat

A {}-stílusú inicializáció C++11 óta működik, value initializationnek felel meg, azaz az alaptípusokat nullázza.

A ()-inicializáció valójában egy függvénydeklaráció, ld. "most vexing parse". "Anything that looks like a declaration is one." Ez a kódsor egy 0 argumentumú, void* visszatérési értékű függvényt deklarál.

Bár az int literálok nem konvertálódnak implicit módon void*-gá (ellentétben a C-vel), ez alól kivétel a 0. Jobb alternatíva C++11 óta a nullptr.

A külön inicializálás nélküli "void* p;" definíció szintén jó, mivel egy globális változóról van szó, és az azok által lefoglalt memória a szabvány szerint 0-ra inicializálódik.

3. feladat 0/3 pont

Mit ír ki az alábbi program, ha az adott implementáció a char értékek kódolására az ASCII kódtáblát használja?

```
#include <iostream>
int main() {
  const char c = 'A';
  std::cout << c << +c << std::endl;</pre>
```

/álasz			
A65			
AA			
6565			
65A			
A program nem fordul le			
Magyarázat			

4. feladat 0/3 pont

Mi az alábbi kifejezés értéke?

3U > -1

Válasz

false

true

Fordítási hiba

Magyarázat

A bal oldal típusa unsigned, a jobb oldalé int. Ezek ugyanakkora típusok, az egyik előjeles, a másik előjel nélküli. Ilyenkor az a szabály, hogy az előjeles típusú érték konvertálódik az előjel nélküli érték típusára.

-1-ből a konverzió során tehát az unsigned lehetséges legnagyobb értéke lesz, ami legalább 65535, tehát nagyobb, mint 3.

5. feladat 0/4 pont

```
#include <iostream>
#include <vector>

int main() {
   const std::vector<int> v{1, 2, 3, 4};
   (*) {
     std::cout << x;
   }
   return 0;
}</pre>
```

Válaszok

for (int x: v)

✓ for (unsigned x: v)

for (const int x: v)

for (int& x: v)

Magyarázat

Az int nyilván jó, az int& pedig nem, hiszen a vektor const.

Kevésbé magától értetődő, de az unsigned és a const int is jók. Ugyanis ez a fajta for -ciklus annak felel meg, mintha egy iterátorral végigmennénk az adatstruktúrán, és a ciklusmag legelején inicializálnánk a deklarációban szereplő x változót az iterátor aktuális tartalmára. Az iterátor const int&-re mutat, ebből pedig mind az unsigned, mind a const int inicializálható. Nem baj tehát, ha x const, hiszen nem x, hanem a (láthatatlan) iterátor a ciklusváltozó.

https://en.cppreference.com/w/cpp/language/range-for

Legfontosabb tudnivalók Kapcsolat Versenyszabályzat Adatvédelem

© 2022 Human Priority Kft.

KÉSZÍTETTE

Megjelenés

➡ Világos ❖