Informační systém Masarykovy univerzity

Zodpovězení odpovědníku (student)

odpovědník Vnitrosemestrální test 17.00

Celkem 8 otázek, za každou otázku je maximálně 2,5 bodu. Otázka může mít více správných odpovědí (má vždy minimálně jednu). Za každou správnou odpověď je přibližně poměrná část bodového hodnocení otázky, za každou špatnou odpověď je -1 bod. Není povoleno používat dodatečné materiály.

• Skrýt parametry odpovědníku.

Základní informace

- počet otázek: 8
- čas: bez omezení
- výsledky naleznete v poznámkovém bloku
  - pouze jeden nejnovější výsledek

Implicitní bodování

+1	0	−1
✓ správně	nezodpovězeno	✗ špatně

Prohlídka odpovědí

- přístupná po zodpovězení

Průchody

- nelze odpovídat (skenovací písemky)

Další nastavení

Zeleně jsou vyznačeny správné odpovědi.

1. Která z následujících tvrzení jsou pro jazyk C++ pravdivá?
   - ☐ Při předávání argumentu konstantní referencí může a nemusí být vytvořena jeho kopie, záleží na překladači.
   - ☐ Zvětšení vektoru `v` typu `vector<int>` na 20 prvků provedeme pomocí `v = vector<int>(20)`.
   - ☐ Přetěžovat (*overload*) lze pouze operátory; přetěžovat funkce není možné.
   - ☑ ✔ *Přetěžovat (*overload*) můžeme i přiřazovací operátor.
   - ☐ Metoda `end()` na libovolném prázdném kontejneru vrací iterátor `nullptr`.
   - ☑ ✔ *Mezi rozdíly mezi ukazateli a referencemi patří skutečnost, že reference se nedají přesměrovat, tj. odkazují se pořád na stejnou paměť.

body = 2.5 = 2.5

2.
```
#include<iostream>
int main() {
  int a;
  int& b = a;
  int* c = &a;
  *c = 1;
  int d = b;
  c = &d;
  b = 0;
  *c += 6;
  std::cout << a << b << *c << d;
}
```

Pro výše uvedený kód platí:

- ☑ ✔ *Vypíše „0077“.

- ☐ Vypíše „0070“.
- ☐ Nezkompiluje se, protože podle nejnovějšího standardu je v přiřazení `b = 0` typová chyba, mělo to správně být `b = nullptr`.
- ☐ Vypíše „0707“.
- ☐ Nelze určit, co vypíše, protože dochází k použití neinicializované paměti.
- ☐ Vypíše „7777“.

body = 2.5 = 2.5

3.
```
#include <iostream>
#include <vector>
using namespace std;
int main() {
  vector<int> v(2, 3);
  vector<int> w{ 2, 3 };
  for (int x : v) { cout << x; }
  for (int x : w) { cout << x; }
}
```

Výše uvedený kód vypíše:

- ☐ 3232
- ☐ 32222
- ☐ 3333
- ☐ 2323
- ☐ 22232
- ☑ ✔ *3323

body = 2.5 = 2.5

4. K druhému prvku ntice (*tuple*) `t` přistupujeme pomocí:

- ☐ `t.get<1>()`
- ☐ Pro čtení použijeme `t.get<1>(t)`, pro zápis `t.set<1>(t, hodnota)`.
- ☐ `t.get(1)`
- ☐ Pro čtení použijeme `t.get(1)`, pro zápis `t.set(1, hodnota)`.
- ☑ ✔ *`std::get<1>(t)`
- ☐ Pro čtení použijeme `std::get<1>(t)`, pro zápis `std::set<1>(t, hodnota)`.

body = 2.5 = 2.5

5.
```
#include <iostream>
using namespace std;
class X {
public:
  X() { cout << "X"; }
};
class Y {
public:
  Y() { cout << "Y"; }
};
class Z {
  X x;
  Y y;
public:
   Z() { cout << "Z"; }
};
int main() {
  Z z;
}
```

Pro výstup výše uvedeného kódu platí:

- ☐ Výstupem bude „ZYX“.
- ☐ Výstupem bude „YXZ“.
- ☐ Výstupem bude „ZXY“.

- [x] *Výstupem bude „XYZ“.
- [ ] Výstupem bude „Z“, atributy `x` a `y` budou neinicializované, a proto se na nich konstruktor nezavolá.
- [ ] Výstup není možno určit, standard C++ nespecifikuje pořadí volání konstruktorů atributů.

body = 2.5 = 2.5

6.
```
#include <iostream>
using namespace std;
class A {
public:
  A() { cout << 'A'; }
  A(const A&) { cout << 'c'; }
  A& operator=(const A&) {
    cout << '='; return *this;
  }
  ~A() { cout << '~'; }
};
int main() {
  A a;
  A b = a;
  A c{};
  c = a;
  cout << '\n';
}
```

Pro výstup výše uvedeného kódu platí:

- [ ] Na prvním řádku výstupu bude „`A=A=`“.
- [ ] Na prvním řádku výstupu bude „`AA=A=`“.
- [ ] *Na druhém řádku výstupu bude „`~~~`“.
- [x] Kód se vůbec nespustí, protože obsahuje syntaktickou chybu na třetím řádku uvnitř funkce `main`.
- [ ] Na prvním řádku výstupu bude „`AAc=A=`“.
- [ ] Žádná z ostatních nabízených odpovědí není správná.

body = -1 = -1

7.
```
#include <fstream>
#include <string>
int main() {
  using namespace std;
  ofstream output("output.txt");
  if (!output) { return 2; }
  ifstream input("input.txt");
  if (!input) { return 1; }
  string line;
  getline(input, line);
  output << line << '\n';
  input.close();
}
```

Předpokládejme, že přístupová práva aktuálního adresáře a všech souborů v něm obsažených dovolují čtení i zápis. Která z následujících tvrzení o výše uvedeném kódu platí?

- [ ] Nelze říct, jaký bude obsah souboru `output.txt` po skončení kódu, protože jsme zapomněli proud `output` uzavřít pomocí `output.close()`.
- [ ] V případě, že soubor `output.txt` neexistuje, vrátí hodnotu 2.
- [ ] Kód se nezkompiluje, protože na řádku `getline(input, line)` je chyba. Správně se metoda `getline` volá jako `input.getline(line)`.
- [ ] V případě, že soubor `input.txt` neexistuje, soubor `output` zůstane nedotčen (tj. bude obsahovat to, co obsahoval před spuštěním kódu).
- [x] *V případě, že soubor `output.txt` existuje a soubor `input.txt` neexistuje, vrátí hodnotu 1.
- [x] *V případě, že soubor `input.txt` neexistuje, bude soubor `output.txt` po spuštění kódu prázdný.

body = 2.5 = 2.5

8.
```
#include <iostream>
#include <vector>
```

```
void f(std::vector<int>& v) {
  v[0] = 1;
}
int main() {
  std::vector<int> x = { 3, 4, 5 };
  std::vector<int>& y = x;
  x[1] = 2;
  f(y);
  for (auto a : y) { ++a; }
  for (auto a : y) { std::cout << a; }
}
```

Výše uvedený kód vypíše:

- [ ] 236
- [x] ✗ 145
- [ ] 456
- [ ] 325
- [ ] *125
- [ ] 345

body = -1 = -1