PB161: Vnitro Test 31.10. 15:00

Jméno a příjmení – pište do okénka	*UČO*	*Číslo zadání*
		1

Za zcela správně zodpovězenou otázku jsou 2 body. Za každou špatnou odpověď je -1 bod. Více odpovědí může být správných.

1 Která z uvedených tvrzení jsou pro jazyk C++ pravdivá?

A Korektní stav vstupního proudu lze zjistit pomocí členské metody good()

B Pokud chci vytvářet vlastní vstupní proud, dědím typicky z třídy ostream

C STL proudy tvoří objektově orientovanou hierarchii navrženou tak, aby bylo možné zaměnit jeden typ proudu za jiný (např. fstream za stringstream)

D Operátory pro výstup pro naši třídu typicky přetěžujeme s proudem typu istream.

E Pro načítání dat ze standardního vstupu se používá proud cin, pro výpis na standardní výstup se používá proud cerr

---

2

```
#include <iostream>
using std::cout;

class Test1 {
public:
    void print() { intern(); }
private:
    void intern() { cout << "T1"; }
};
class Test2 : public Test1 {
public:
    void print() { intern(); }
private:
    void intern() { cout << "T2"; }
};

void print(Test1* obj) {obj->print();}
void print2(Test2* obj) {obj->print();}

int main() {
    Test1 obj1;
    Test2 obj2;
    print(&obj1);print(&obj2);
    print2(&obj2);
    return 0;
}
Pro uvedený program platí:
```

A program nelze přeložit

B vypíše 'T1T1T2'

C vypíše 'T1T1T1'

D žádná z ostatních možností není správná

E vypíše 'T1T2T2'

---

3 Která z uvedených tvrzení jsou pro jazyk C++ pravdivá?

A Pokud vytvoříme proměnnou typu konstantní reference, tak není možné pomocí této reference měnit obsah odkazované proměnné

B Pokud je parametr funkce předávaný nekonstantní referencí ve funkci změněn, tak se změna projeví i mimo funkci

C Proměnná typu reference musí být incializována ihned při svém vzniku

D Pro jednu proměnnou nelze vytvořit více než dvě reference (jednu konstantní a jednu nekonstantní)

E Pokud předáváme proměnnou jako parametr funkce očekávající referenci, musíme získat adresu předávané proměnné pomocí operátoru &

4

```
#include <iostream>
using std::cout;

void foo(float a) {
    cout << a << " ";
}
void foo(int a, float b) {
    cout << a << b << " ";
}
void foo(int a, float b = 0.1) {
    cout << a << b << " ";
}

int main() {
    foo(1.1);
    foo(1);
    foo(1, 1.1);
}
Pro uvedený program platí:
```

A program nelze přeložit

B vypíše '1.1 1 10.1'

C vypíše '10.1 10.1 11.1'

D vypíše '1.1 10.1 11.1'

E žádná z ostatních možností není správná

5

```
struct CTest {
_PRAVO1_
  CTest(int value) : m_value(value) {}
_PRAVO2_
  int GetValue() const { return m_value; }
  void SetValue(int value) {
    m_value = value;
  }
_PRAVO3_
  int m_value;
};

int main() {
  struct CTest test(10);
  test.SetValue(11);
  return 0;
}
```

Doplňte správné hodnoty práv namísto označení _PRAVO1_, _PRAVO2_ a _PRAVO3_ tak, aby bylo možné kód zkompilovat a zároveň dodržoval pravidla zapouzdření.

A právo _PRAVO1_ nezadáno, právo _PRAVO2_ nezadáno, _PRAVO3_ = private:
B _PRAVO1_ = private:, _PRAVO2_ = public:, _PRAVO3_ = private:
C právo _PRAVO1_ nezadáno, právo _PRAVO2_ nezadáno, _PRAVO3_ = public:
D _PRAVO1_ = private:, _PRAVO2_ = private:, _PRAVO3_ = private:
E _PRAVO1_ = public:, _PRAVO2_ = public:, _PRAVO3_ = private:

6

```
#include <iostream>
using std::cout;

class Test1 {
public:
    virtual void print() { intern(); }
private:
    void intern() { cout << "T1"; }
};
class Test2 : public Test1 {
public:
    virtual void print() { intern(); }
private:
    void intern() { cout << "T2"; }
};

void print(Test1* obj) {obj->print();}
void print2(Test2* obj) {obj->print();}

int main() {
    Test1 obj1;
    Test2 obj2;
    print(&obj1);print(&obj2);
    print2(&obj2);
    return 0;
}
```

(Obtížnější otázka) Pro uvedený program platí:

A program nelze přeložit
B vypíše 'T1T1T1'
C vypíše 'T1T1T2'
D žádná z ostatních možností není správná
E vypíše 'T1T2T2'

7

```
#include <iostream>
using std::cout;

class A {
public:
    virtual void print() = 0;
};
class B : public A {
public:
    void print() { cout << "B"; }
};

void print(A* obj) {obj->print();}
void print2(B* obj) {obj->print();}

int main() {
    A obj1;
    B obj2;
    print(&obj1);print(&obj2);
    print2(&obj2);
    return 0;
}
```

Pro uvedený program platí:

A vypíše 'B'
B vypíše 'BBB'
C vypíše 'BB'
D žádná z ostatních možností není správná
E program nelze přeložit

8

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

Pro uvedený kód platí:

A vypíše '777'
B žádná z ostatních možností není správná
C vypíše '335'
D nelze přeložit
E vypíše '355'
F nelze určit, závisí na předešlém obsahu paměti

9

```
#include <iostream>
void add(int& v1, int v2, int* v3) {
    v2 += v1;
    v1 = v2;
    *v3 = v1;
}
int main() {
    int x = 1;
    int y = 2;
    int z = 3;

    add(x, y, &z);
    std::cout<<x<<" "<<y<<" "<<z;
    return 0;
}
```

Pro uvedený kód platí:

A vypíše '3 2 2'
B žádná z ostatních možností není správná
C vypíše '3 2 3'
D vypíše '1 2 3'
E vypíše '3 3 3'

10

```
#include <iostream>
using std::cout;
class X {
  public:
    X() { cout << "X"; }
    ~X() { cout << "~X"; }
};
class Y : public X {
  public:
    Y() { cout << "Y"; }
    Y(const Y& copy) { cout << "cY"; }
    ~Y() { cout << "~Y"; }
};
int main() {
    Y   obj1;
    Y   obj2;
    return 0;
}
```

A vypíše 'XYXY~Y~X~Y~X'
B vypíše 'YcYYcY~Y~Y'
C vypíše 'XYXY~X~Y~X~Y'
D žádná z ostatních možností není správná
E vypíše 'YY~Y~Y'
F vypíše 'YXYX~Y~X~Y~X'