PB161: PB161 vnitro 15:00

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

Za otázku se všemi správně odpovězenými možnostmi jsou 2 body. Za část správných odpovědí je poměrný počet bodů. Za každou špatnou odpověď je -1 bod. Otázka může mít více správných odpovědí. Není povoleno používat dodatečné materiály.

1

```
class A {
protected:
  virtual void release() {}
public:
  ~A() { release(); }
};

class B : public A {
  int* m_array;
public:
 B(int length) { m_array = new int[length]; }
  void release() {
    if (m_array) delete[] m_array;
  }
  ~B() { release(); }
};

int main() {
    A* object1 = new B(10);
    B* object2 = new B(10);

    delete object1;
    delete object2;

    return 0;
}
```

Která z uvedených tvrzení jsou pravdivá?

A Uvedený kód nelze přeložit, neboť třída A nenabízí veřejný konstruktor.
B Pokud bychom deklarovali destruktor třídy A jako virtuální, tak by uvedený kód nezpůsobil žádný memory leak.
C Uvedený kód lze přeložit a nezpůsobí žádný memory leak.
D Uvedený kód způsobí memory leak dvou polí alokovaných jako new int[10].
E Uvedený kód způsobí memory leak právě jednoho pole o velikosti 10 bajtů.
F Uvedený kód způsobí memory leak jednoho pole alokovaného jako new int[10].

2

```
#include <iostream>
#include <map>
class A {
public:
    A() {  std::cout<<"A"; }
    ~A() {  std::cout<<"~A"; }
};
int main() {
  std::map<int, A*> asoc;
  for (int i = 0; i < 3; i++) asoc[i] = new A;
  asoc.clear();
  return 0;
}
```

Pro uvedený kód platí:

A dynamicky alokovaná pamět není korektně uvolněna
B všechna dynamicky alokovaná paměť je korektně uvolněna
C žádná z ostatních možností není správná
D vypíše 'AAA~A~A~A~A~A~A'
E vypíše 'AAA~A~A~A'
F vypíše 'AAA'

3

```
#include <iostream>
void print() { std::cout<< "x"; }
namespace MyNamespace {
  void print() {std::cout<< "y";}
}
namespace MyNamespace2 {
  void print() {std::cout<< "z";}
}
using namespace MyNamespace;
using namespace MyNamespace2;
int main() {
    print();
    return 0;
}
```

Pro uvedený kód platí:

A vypíše 'z'
B vypíše 'y'
C vypíše 'x'
D vypíše 'yz'
E vypíše 'xyz'
F žádná z ostatních možností není správná

4

```
#include <iostream>

class A {
public:
    _S1 void foo1() {std::cout << "1";}
    _S2 void foo2() {std::cout << "2";}
    _S3 ~A() {std::cout << "3";}
};

class B : public A {
public:
    _S4 void foo1() {std::cout << "4";}
    _S5 void foo2() {std::cout << "5";}
    _S6 ~B() {std::cout << "6";}
};

int main() {
    A* object1 = new B;
    B* object2 = new B;

    object1->foo1();
    object1->foo2();
    object2->foo1();
    object2->foo2();

    delete object1;
    delete object2;

    return 0;
}
```

Vyberte korektní možnosti hodnot specifikátorů _S u metod tak, aby došlo k vypsání textu "1245363"

A Nelze doplnit tak, aby došlo k vypsání požadovaného výstupu.
B _S1=, _S2=virtual, _S3=virtual, _S4=, _S5=, _S6=virtual
C _S1=, _S2=virtual, _S3=virtual, _S4=virtual, _S5=, _S6=
D _S1=, _S2=, _S3=, _S4=, _S5=, _S6=
E _S1=virtual, _S2=, _S3=virtual, _S4=, _S5=, _S6=virtual
F _S1=virtual, _S2=virtual, _S3=virtual, _S4=virtual, _S5=, _S6=

---

5 Která z uvedených tvrzení jsou korektní?

A STL kontejner list má menší paměťové nároky pro uložení stejného množství prvků než kontejner vector.
B STL kontejner vector se typicky používá pro datové struktury, které vyžadují rychlé vkládání nebo ubírání prvků ve středu kontejneru.
C STL kontejner map uchovává dvojici klíč a hodnota a nabízí k nim přístup prostřednictvím iterátoru.
D STL kontejner map poskytuje rychlejší přístup k danému prvku než kontejner vector, pokud známe index daného prvku.
E STL kontejner map se používá jako přímá náhrada pole prvků a nabízí i stejné operace.
F Žádná z ostatních odpovědí není správná

6

```
int main() {
    B* obj1 = new A;
    C* obj2 = new C;
    C* obj3 = new A;
    return 0;
}
```

Který z uvedených vztahů dědičnosti mezi třídami A, B, C platí v případě, že uvedený kód lze zkompilovat?

A pro uvedený kód není možné takové vztahy najít
B B je potomek A, C je potomek A, B je potomek C
C C není potomek B, A není potomek C, A není potomek B
D žádná z ostatních odpovědí není správná
E A je potomek B, A je potomek C, B není potomek C

---

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

A Pro jednu proměnnou nelze vytvořit více než dvě reference (jednu konstantní a jednu nekonstantní)
B Pokud vytvoříme proměnnou typu konstantní reference, tak není možné pomocí této reference měnit obsah odkazované proměnné
C Proměnná typu reference musí být incializována ihned při svém vzniku
D Pokud předáváme proměnnou jako parametr funkce očekávající referenci, musíme získat adresu předávané proměnné pomocí operátoru &
E Pokud je parametr funkce předávaný nekonstantní referencí ve funkci změněn, tak se změna projeví i mimo funkci

8

```
class A {
_PRAVO1_
  A(int value) : m_value(value) {}
public:
  int GetValue() const { return m_value; }
  virtual void SetValue(int value) {
    m_value = value;
  }
_PRAVO2_
  int m_value;
};

class B : _PRAVO3_ A {
_PRAVO4_
  B(int value) : A(value) {}
  int GetValue() const {
    return A::GetValue();
  }
};

int main() {
  B test(10);
  test.SetValue(11);
  int value = test.GetValue();
  return 0;
}
```

Doplňte správné hodnoty práv namísto označení _PRAVO1_, _PRAVO2_, _PRAVO3_ a _PRAVO4_ tak, aby bylo možné kód zkompilovat a zároveň dodržoval pravidla zapouzdření.

A _PRAVO1_ = protected:, _PRAVO2_ = private:, _PRAVO3_ = public, _PRAVO4_ = public:

B _PRAVO1_ = private:, _PRAVO2_ = private:, _PRAVO3_ = public, _PRAVO4_ = public:

C _PRAVO1_ = private:, _PRAVO2_ = public:, _PRAVO3_ = private, _PRAVO4_ = private:

D _PRAVO1_ = protected:, _PRAVO2_ = private:, _PRAVO3_ = private, _PRAVO4_ = public:

E _PRAVO1_ = protected:, _PRAVO2_ = public:, _PRAVO3_ = protected, _PRAVO4_ = protected:

9

```
#include <iostream>
class A {
public:
    A(int value) : m_value(value) {}
    void set(int value) { m_value = value; }
    int get() const { return m_value; }
private:
    int m_value;
};

void foo(const A v1, const A& v2, const A* v3) {
    v1.set(v2.get());
    v2.set(v3->get());
    v3->set(v1.get());
}
int main() {
    A x(1);
    A y(2);
    A* z = new A(3);

    foo(x, y, z);
    foo(x, y, z);
    std::cout<<x.get()<<" "<<y.get();
    std::cout<<" "<<z->get();
    delete z;
    return 0;
}
```

Pro uvedený kód platí:

A program nelze přeložit

B vypíše '1 2 3'

C vypíše '3 2 3'

D vypíše '3 3 3'

E vypíše '2 2 2'

F vypíše '3 2 2'

10

```
#include <iostream>
#include <list>
class A {
public: void foo() {std::cout << "A"; }
};
int main() {
  std::list<A*> sez;
  std::list<A*>::iterator i;
  sez.push_back(new A);
  sez.push_back(new A);
  sez.push_back(new A);
  for (i= sez.begin(); i!= sez.end(); i++) {
    (*i)->foo();
  }
  return 0;
}
```

Kolega vám ukazuje uvedený kód s tím, že jej nelze přeložit s chybou error: request for member 'foo' in '* iter.std::_List_iterator<_Tp>::operator-> [with _Tp = A*]()', which is of non-class type 'A*'
Jak mu odpovíte?

A Chybně vkládané prvky do kontejneru seznam. Pokud by byly prvky vkládány jako seznam.push_back(A); (jediná změna), tak by bylo možné program bez chyb zkompilovat.

B Chybně vkládané prvky do kontejneru seznam. Pokud by byly prvky vkládány jako seznam.push_back(new A()); (jediná změna), tak by bylo možné program bez chyb zkompilovat

C Využití iterátoru bez derefence.

D Uvedená chyba nemohla být překladačem vypsána.

E Chybná deklarace kontejneru seznam. Pokud by byl kontejner deklarován jako std::list<A> (jediná změna), tak by bylo možné program bez chyb zkompilovat.

F Chybně deklarovaný iterátor i. Pokud by byl iterator deklarován jako std::list<A>::iterator (jediná změna), tak by bylo možné program bez chyb zkompilovat.