PB161: PB161_testvnitro_17

Jméno a příjmení – pište do okénka	*UČO*	*Číslo zadání*
		17

Za otázku se všemi správně odpovězenými možnostmi jsou 2 resp. 3 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

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

Pro uvedený kód platí:

A žádná z ostatních možností není správná
B vypíše 'z'
C vypíše 'xz'
D vypíše 'xy'
E nelze přeložit

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

A Metoda deklarovaná s klíčovým slovem const může vždy měnit obsah vnitřních atributů třídy
B Metoda deklarovaná s klíčovým slovem const nemůže být volána pozdní vazbou
C Metoda deklarovaná s klíčovým slovem const musí mít všechny parametry předávané konstantní referencí
D Pokus o změnu parametru předávaného konstantní referencí upozorní překladač už v době překladu
E Proměnná typu konstantní reference může být měněna pouze v metodě deklarované s klíčovým slovem const

3

```
#include <iostream>
#include <vector>
class A {
public:
    A() {  std::cout<<"1"; }
    ~A() {  std::cout<<"2"; }
};
int main() {
  std::vector<A*> vect;
  for (int i = 0; i < 3; i++) {
    vect.push_back(new A);
  }
  for (int i = 0; i < 3; i++) {
    delete vect[i];
  }
  vect.clear();
  return 0;
}
```

Pro uvedený kód platí:

A vypíše '111222'
B vypíše '111'
C všechna dynamicky alokovaná paměť je korektně uvolněna
D vypíše '222111'
E dynamicky alokovaná pamět není korektně uvolněna
F žádná z ostatních možností není správná

4

```
int main() {
    int value1 = 0;
    int& value2 = value1;
    int& value3 = value2;
    value1 += 10;
    value2 += 20;
    value3 = value2;
    value3 += 30;
    return 0;
}
```

Hodnota bude hodnota proměnných value1,value2 a value3 před příkazem return?

A value1=30, value2=30, value3=30
B value1=60, value2=60, value3=60
C value1=10, value2 a value3 nelze bez spuštění určit (value2 = adresa proměnné value1 zvětšená o 20 resp. 30 pro value3)
D program nelze přeložit
E value1=10, value2=30, value3=60
F value1=10, value2=20, value3=30

5

```
#include <iostream>
using namespace std;
class A {
public:
    virtual void print() const = 0;
    void print2() const {cout << "7|";}
    ~A() {cout << "4|";}
};
class B : public A {
public:
    virtual void print() const {
        cout << "1|";
    }
    ~B() {cout << "5|";}
};
class C : public B {
public:
    virtual void print() const {
        cout << "2|";
    }
    virtual void print2() const {
        cout << "3|";
    }
    ~C() {cout << "6|";}
};

int main() {
    A* var1 = new B;
    A* var2 = new C;
    var1->print();
    var2->print();
    var2->print2();
    return 0;
}
```

Program po spuštění vypíše:

A 1|2|3|4|4|
B program nelze přeložit
C žádná z ostatních odpovědí není správná
D 1|2|7|5|4|6|5|4|
E 1|2|3|5|4|6|5|4|
F 1|2|7|4|4|

6

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

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

7

```
#include <iostream>
using namespace std;
class A {
public:
    virtual void foo() const = 0;
    virtual void foo2() const = 0;
    virtual ~A() {}
};
class B : public A {
public:
    virtual void foo() const {}
};
class C : public B {
public:
    virtual void foo2() const {}
};
class D : public C {
public:
};
int main() {
    A* var1 = new D;
    delete var1;
    return 0;
}
```

Předpokládejte že výše uvedený program vypíše při překladu:
..\main.cpp: In function 'int main()':
..\main.cpp:20: error: cannot allocate an object of abstract type 'D'
..\main.cpp:16: note: because the following virtual functions are pure within 'D':
..\main.cpp:5: note: virtual void A::foo() const
..\main.cpp:6: note: virtual void A::foo2() const
Která z uvedených tvrzení jsou správná?

A Přidání modifikátoru const u metody foo ve třídě B a foo2 ve třídě C umožní kompilaci bez chyb
B Přetypování vytvářené instance typu D (proměnná var1) na typ A pomocí static_cast umožní kompilaci bez chyb
C Odstranění modifikátoru const u metody foo ve třídě A a přidání modifikátoru const u metody foo2 ve třídě B umožní kompilaci bez chyb
D K uvedené chybě při překladu nemohlo dojít, program se zkompiluje bez chyb
E Přidání chybějící dealokace dynamicky alokovaného objektu D umožní kompilaci bez chyb
F Odstranění modifikátoru const u metody foo a foo2 ve třídě A umožní kompilaci bez chyb

8

```
#include <iostream>
using namespace std;

class A {
public:
    virtual void foo1() = 0;
    void foo2() {cout << "A2";}
    void foo3() {cout << "A3";}
};

class B : public A {
public:
    void foo1() {cout << "B1";}
    void foo2() {cout << "B2";}
    void foo3() {cout << "B3";}
};

int main () {
    A* var = new B;

    var->foo1();
    var->foo2();
    var->foo3();

    return 0;
}
```

Uvedený kód vypíše na standardní výstup řetězec:

A B1B2B3
B B1A2B3
C B1A2A3
D B1B2A3
E uvedený kód nelze přeložit
F žádná z ostatních možností není správná

Tato strana je prázdná.