

PB161: Vnitrosemestrální písemka vzor 12.00

Jméno a příjmení – pište do okénka	UČO	Číslo zadání
		1

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.

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

`class A {`
`public:`
 `virtual void fun1() {cout << "A1";}`
 `void fun2() {cout << "A2";}`
 `virtual void fun3() = 0;`
`};`

`class B : public A {`
`public:`
 `virtual void fun1() {cout << "B1";}`
 `void fun2() {cout << "B2";}`
 `void fun3() {cout << "B3";}`
`};`

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

 `var->fun1();`
 `var->fun2();`
 `var->fun3();`
`}`

Pro výše uvedený kód platí:

- A** vypíše 'B1A2B3'
- B** vypíše 'A1A2B3'
- C** vypíše 'B1B2B3'
- D** nezkompiluje se, protože metoda fun3() ve třídě B není virtual
- E** skončí s chybou za běhu
- F** nezkompiluje se, protože A je abstraktní třída a není možno deklarovat ukazatel na abstraktní třídu

2 `class A {`
 `int value;`
`public:`
 `A() : value(42) {}`
 `int getValue() { return value; }`
 `virtual void fun() = 0;`
`protected:`
 `void setValue(int v) { value = v; }`
`};`

`class B : public A {`
`private:`
 `const char * name;`
`public:`
 `void fun() { setValue(17); }`
`};`

`class C : public B {`
`protected:`
 `void boo() { setValue(19); }`
`};`

`int main() {`
 `A * ptr = new C;`
 `ptr->fun();`
 `return ptr->getValue();`
`}`

Pro výše uvedený kód platí:

- A** nezkompiluje se, protože třída C nemá metodu fun()
- B** nezkompiluje se, protože metoda fun() nemůže přistupovat k metodě setValue()
- C** nezkompiluje se, protože třída C není přímým potomkem třídy A
- D** zkompiluje se a po spuštění vrátí hodnotu 42
- E** zkompiluje se a po spuštění vrátí hodnotu 17
- F** dynamicky alokovaná paměť nebude korektně uvolněna (tzv. memory leak)

3

```
#include <iostream>
int main() {
    int a;
    int * b = &a;
    int & c = a;
    int d = a;
    d = 1;
    a = 2;
    b = &d;
    c = d;
    *b += 4;
    c += 7;

    std::cout << a << *b
               << c << d;
}
```

Pro výše uvedený kód platí:

- A** nezkompiluje se
- B** vypíše '8585'
- C** vypíše '5555'
- D** vypíše '8888'
- E** vypíše '2151515'
- F** nelze určit, co vypíše, protože dochází k použití ne-inicializované paměti

4

```
#include <iostream>
using std::cout;
class X {
public:
    X() { cout << "X"; }
    virtual ~X() { cout << "~X"; }
};

class Y : public X {
public:
    Y() { cout << "Y"; }
    ~Y() { cout << "~Y"; }
    Y(const Y&) { cout << "cY"; }
};

int main() {
    X * ptr = new Y;
    X obj1;
    delete ptr;
}
```

Pro výše uvedený kód platí:

- A** vypíše 'YX~X~X'
- B** vypíše 'YX~Y~X~X'
- C** vypíše 'YX~X~Y~X'
- D** dynamická paměť je správně dealokována
- E** dynamická paměť není správně dealokována, protože ptr je ukazatel typu 'X*', a pro správnou dealokaci bylo třeba napsat `delete dynamic_cast<Y*>(ptr);`
- F** vypíše 'XYcYX~Y~X~X'

5

```
#include <iostream>
#include <vector>
using std::cout;
using std::vector;
class A {
public:
    A() { cout << "1"; }
    ~A() { cout << "2"; }
};

int main() {
    vector<A*> vec;
    vec.reserve(2);
    for (int i = 0; i < 3; ++i) {
        vec.push_back(new A);
    }
    vec.clear();
}
```

Pro výše uvedený kód platí:

- A** vypíše '111'
- B** vypíše '111222'
- C** dojde k chybě za běhu, protože se pokoušíme vložit víc prvků, než kolik jsme si ve vektoru rezervovali místa
- D** vypíše '11111'
- E** dynamicky alokovaná paměť je správně uvolněna
- F** dynamicky alokovaná paměť není správně uvolněna

6

```
#include <iostream>
void print() { std::cout << "x"; }
namespace A {
    void print() { std::cout << "y"; }
    namespace B {
        void print() { ::print(); }
    }
}
namespace C {
    void fun() {
        using namespace A;
        A::print();
        B::print();
    }
}

int main() {
    using namespace C;
    print();
    fun();
}
```

Pro výše uvedený kód platí:

- A** vypíše 'yx'
- B** vypíše 'xxx'
- C** vypíše 'xyy'
- D** vypíše 'yyx'
- E** nezkompiluje se, protože není jasné, která funkce `print()` se má zavolat ve funkci `main()`
- F** žádná z ostatních možností není správná

- 7** Která z uvedených tvrzení jsou pro jazyk C++ pravdivá?
- A** Při přetěžování funkcí musí být zachován počet parametrů.
 - B** Přetěžovat lze pouze členské metody tříd, nikoli globální funkce.
 - C** Přetížené metody se mohou lišit jen ve specifikátoru 'const'.
 - D** Vícenásobná dědičnost v C++ existuje, ale dědit se smí jen od čistě abstraktních tříd.
 - E** Statické metody třídy nemají přístup k ukazateli 'this', protože se nevolají na konkrétním objektu.
 - F** Přetěžovat můžeme i destruktory třídy.

- 8**
- ```
#include <iostream>
#include <fstream>
#include <string>

int main() {
 using namespace std;
 ifstream input("input.txt");
 ofstream output("output.txt");
 if (!input) {
 return 1;
 }
 if (!output) {
 return 2;
 }
 string line;
 getline(input, line);
 output << line << "\n";
 input.close();
}
```
- Pro výše uvedený kód platí:
- A** v případě, že soubor 'input.txt' neexistuje, vrátí hodnotu 1
  - B** v případě, že soubor 'input.txt' existuje a soubor 'output.txt' neexistuje, vrátí hodnotu 2
  - C** v případě, že soubor 'input.txt' neexistuje, soubor 'output.txt' zůstane nedotčen (tj. bude obsahovat to, co obsahoval před spuštěním kódu)
  - D** pokud soubor 'input.txt' existuje a obsahuje alespoň jeden řádek, bude po spuštění kódu v souboru 'output.txt' první řádek ze souboru 'input.txt' a nic víc; veškerý původní obsah 'output.txt' bude smazán
  - E** nelze říct, jaký bude obsah souboru 'output.txt' po skončení kódu, protože jsme zapomněli proud 'output' uzavřít
  - F** v případě, že oba soubory 'input.txt' a 'output.txt' existují, kód vrátí hodnotu 0

*Tato strana je prázdná.*