Datum: Jméno a Příjmení:.....
Předmět: .....PJC

A

1. Jaký je rozdíl mezi *statickou vazbou* a *dynamickou vazbou*? Demonstrujte na vhodném příkladu. [2B]

   viz přednáška

2. Pokud je v inicializátoru chyba tak ji opravte. [2B]

```
struct X {
      int base, rem;
      X (int i, int j): base(i), rem(base % j) { }
};
```

   Řešení:

```
Bez chyby.
```

3. Předpokládejme, že Point je třída s veřejným kopírujícím konstruktorem. V následujícím fragmentu programu označte řádky na kterých bude volán kopírující konstruktor třídy Point. [3B]

```
Point global;

Point * foo_bar(Point arg)
{
      Point local = arg; ZDE
      Point *heap = new Point(global); ZDE
      *heap = local;
      Point pa[ 4 ] = { local, *heap }; ZDE
      return heap;
}
```

4. Do následující třídy dopište kopírující konstruktor (uvažujte deep-copy), destruktor a operátor přiřazení. [4B]

```
struct FunnyString {
      FunnyString(): pstring(new std::string) {
            *pstring+=" funny string";
      }
private:
```

```
        std::string *pstring;
};
```

Řešení:

```
FunnyString::FunnyString(const FunnyString& other) {
        pstring = new std::string(*(other.pstring));
}
FunnyString::~FunnyString() { delete pstring; }
FunnyString& FunnyString::operator=(const FunnyString& other) {
        if(this == &other) return *this;
        delete pstring;
        pstring = new std::string(*(other.pstring));
}
```

5. Co bude na výstupu po provedení následujícího programu? [4B]

```
#include <iostream>

using namespace std;

struct A {
      A(int a) { cout << "konstruktor A" << endl;}
     virtual int x() = 0;
     virtual ~A() { cout << "destruktor A" << endl;}

};

struct B : public A {
      B(int a, double b):A(a) {cout << "konstruktor B" << endl;}
     int x() {cout << "X" << endl; }
     ~B() { cout << "destruktor B" << endl; }
};

int main(void) {
      B b(10, 2.0);
      A *x = new B(1, 3.0);
      x->x();
      *x = b;
      delete x;
     return 0;
}
```

Řešení:

```
konstruktor A
konstruktor B
konstruktor A
konstruktor B
X
destruktor B
destruktor A
destruktor B
destruktor A
```

6. Navrhněte a implementujte třídu SComplex(definice operace + a * viz nápověda) tak, aby šel následující kód přeložit. [5B]:

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

int main(void) {
  SComplex c(2.3, 4.0);
  SComplex c1(1.3, 3.4);
  SComplex c3 = c * c1 + c1 * c1;
  SComplex c4 = 1; // c4 = (1,0)
  SComplex c5 = 1.0 + c4;
  cout << c3 << endl;
}
  Nápověda:     A = (a,b)
                B = (c,d)
                A + B = (a*c - a+c, b*d - b+d)
                A * B = (a*c – b*d, a*d + b*c)
```

Řešení:

```
  Návrh třídy bez implementace:

  struct SComplex {
         double a, b;
         SComplex(double _a, double _b = 0.0):a(_a),b(_b) {}
         SComplex operator+(const SComplex& other);
         SComplex operator*(const SComplex& other);

  };
  ostream& operator<<(ostream &out, const SComplex &s);
  SComplex operator+(double num, const SComplex& s);
```