PB161

3. cvičení

Lukáš Ručka

7. října 2011

Class vs. struct

Class vs. struct

- Implicitní práva

Reference

Reference

- Určitá podobnost s ukazatelem

Reference

- Určitá podobnost s ukazatelem
- Dovolují předat funkci data, která má funkce měnit

Reference

- Určitá podobnost s ukazatelem
- Dovolují předat funkci data, která má funkce měnit
- Úspora paměti, nižší režie

Reference

- Určitá podobnost s ukazatelem
- Dovolují předat funkci data, která má funkce měnit
- Úspora paměti, nižší režie

```
experiment a(100, 200);
experiment b(200, 300);
experiment & soucasny = a;
soucasny = b;
```

Reference

```
void proved_pokus(experiment * e);
```

vs.

```
void proved_pokus(experiment & e);
```

Reference - k zamyšlení

```
experiment e(100, 200);
proved_pokus(e);
```

Dědičnost

```
class A {
public:
    int a;
};

class B: public A {
public:
    void set_a(int value){ a = value; }
    int get_a() const { return a; }
};

...

  A alfa;
  B beta;
  A * delta = &beta;
```

Dědičnost

```
class A {
private:
    int a;
};

class B: public A {
public:
    void set_a(int value){ a = value; }
    int get_a() const { return a; }
};

...

  A alfa;
  B beta;
  A * delta = &beta;
```

Dědičnost

```
class A {
protected:
    int a;
};

class B: private A {
public:
    void set_a(int value){ a = value; }
    int get_a() const { return a; }
};

...

  A alfa;
  B beta;
  A * delta = &beta;
```

Jednoduchá třída

- Z minula

```
class jednoducha {
public:
    jednoducha(int c, int d)
        :a(c),b(d)
    { ; }

    int secti(){ return a+b; }

private:
    int a;
    int b;
};
```

Kopírovací konstruktor

Navrhněte třídu experiment, která:

- bude mít dva celočíselné atributy a ukazatel na první z těchto atributů (vše veřejné)
- bezparametrický konstruktor a konstruktor (int a, int b)

Kopírovací konstruktor

Navrhněte třídu experiment, která:

- bude mít dva celočíselné atributy a ukazatel na první z těchto atributů (vše veřejné)
- bezparametrický konstruktor a konstruktor (int a, int b)

```
class experiment {
public:
    experiment()
        :a(0),b(0),ta(&a)
    { ; }

    experiment(int c, int d)
        :a(c),b(d),ta(&a)
    { ; }

    int a;
    int b;
    int * ta;
};
```

Kopírovací konstruktor

Přidejte main, který:

- vytvoří instanci třídy experiment pomocí parametrického konstruktoru
- vytvoří druhou instanci třídy experiment identickou s první (avšak bez volání operátoru =)
- pro obě instance změní hodnotu prvního atributu
- vypíše obě hodnoty prvního atributu pomocí ukazatele každé instance

Kopírovací konstruktor

Přidejte main, který:

- vytvoří instanci třídy experiment pomocí parametrického konstruktoru
- vytvoří druhou instanci třídy experiment identickou s první (avšak bez volání operátoru =)
- pro obě instance změní hodnotu prvního atributu
- vypíše obě hodnoty prvního atributu pomocí ukazatele každé instance

```
int main(){

    experiment prvni(20, 30);
    experiment druhy(prvni);
    prvni.a = 30;
    druhy.a = 90;

    cout << *prvni.ta << " " << *druhy.ta << endl;

    return 0;
}
```

Kopírovací konstruktor

Kdo ví kde je chyba?

Kopírovací konstruktor

Kdo ví kde je chyba?

```
class experiment {
public:
    experiment()
        :a(0),b(0),ta(&a)
    { ; }

    experiment(int c, int d)
        :a(c),b(d),ta(&a)
    { ; }

    experiment(const experiment & orig)
        :a(orig.a),b(orig.b),ta(&a)
    { ; }

    int a;
    int b;
    int * ta;
};
```

Přiřazovací operátor

Napište přiřazovací operátor pro třídu experiment

Přiřazovací operátor

Napište přiřazovací operátor pro třídu experiment

```
experiment & operator=(const experiment & orig){
    a = orig.a;
    b = orig.b;

    return *this;
}
```

Přiřazovací operátor

Problém sebepřiřazení

Přiřazovací operátor

Problém sebepřiřazení

```
experiment & operator=(const experiment & orig){
    if (&orig != this){
        a = orig.a;
        b = orig.b;
    }

    return *this;
}
```