

Dědičnost tříd

Umožňuje děděním jedné nebo více tříd vytvářet nové třídy.

Děděná třída – základní třída

Dědicí třída – odvozená třída

Dědění jedné třídy

```
class A { };
```

```
class B: public A { };
```

člen třídy A	jeho viditelnost v B
<code>private</code>	---
<code>protected</code>	<code>protected</code>
<code>public</code>	<code>public</code>

```
class B: protected A { };
```

člen třídy A	jeho viditelnost v B
<code>private</code>	---
<code>protected</code>	<code>protected</code>
<code>public</code>	<code>protected</code>

```
class B: private A { };
```

```
class B: A { };
```

člen třídy A	jeho viditelnost v B
<code>private</code>	---
<code>protected</code>	<code>private</code>
<code>public</code>	<code>private</code>

Konstruktory

Pokud chceme, aby se v děděné třídě *A* použil její konstruktor, který má parametry, musí být volání tohoto konstruktoru uvedeno v konstruktoru dědicí třídy *B*.

```
class A
```

```
{  
    A(...) { }  
};
```

```
class B: public A
```

```
{  
    B(...) : A(...) { }  
};
```

```

class Osoba
{
    string jmeno;
public:
    Osoba(const char *j): jmeno(j) { }
};

class Student: public Osoba
{
    int rocnik;
public:
    Student(const char *j,int r): Osoba(j),rocnik(r) { }
};

```

```

struct A
{
    A() { cout << " A" << endl; }
    ~A() { cout << "~A" << endl; }
};

struct B: A
{
    B() { cout << " B" << endl; }
    ~B() { cout << "~B" << endl; }
};

{ B b; }

A
B
~B
~A

```

Přístup k proměnným a funkcím se stejným jménem použitím operátoru rozlišení.

```

float a;

void f()
{
    char a; a='z'; ::a=1.2;
}

class A
{
protected:
    int a;
    void f() { }
};

class B: public A
{
    char a;
    void f() { }
}

```

```

public:
    void g() { a='c'; A::a=7; ::a=2.5;
              f(); A::f(); ::f(); }
};

```

Virtuální funkce

Umožňují polymorfismus objektů – přístup k objektům různého typu přes stejné rozhraní.

Úloha:

Sestavit základní třídu s rozhraním pro plošné objekty:

- Rozhraní v základní třídě bude funkce pro výpočet obsahu plošného objektu.
- Děděním základní třídy sestavit třídy pro čtverec, obdélník a kruh.
- Vytvořit pole ukazatelů na objekty uvedených tříd.

```

class PlosnyUtvár
{ protected:
    float a;
public:
    float obsah() { return 0; }
};

class Ctverec: public PlosnyUtvár
{ public:
    Ctverec(float a): a(a) { }
    float obsah() { return a*a; }
};

class Obdelnik: public PlosnyUtvár
{ float b;
public:
    Obdelnik(float a, float b) a(a), b(b) { }
    float obsah() { return a*b; }
};

class Kruh: public PlosnyUtvár
{ public:
    Kruh(float r) { a=r; }
    float obsah() { return M_PI*a*a; }
};

PlosnyUtvár *u[]=
    { new Ctverec(3), new Obdelnik(2,3), new Kruh(2) };

for (int i=0; i<3; ++i) cout << u[i]->obsah() << endl;

```

0
0
0

Funkce *obsah* musí být virtuální – v základní třídě se u ní uvede klíčové slovo **virtual**:

```
class PlosnyUtvar
{ protected:
    float a;
    public:
        virtual float obsah() { return 0; }
};
```

9
6
12.5664

Definice funkce *obsah* ve třídě *PlosnyUtvar* je pouze pro vytvoření rozhraní, její tělo je formální a nemá žádný význam. Postačí použít čistě virtuální funkci – její zápis má za hlavičkou uvedeno **=0**; a funkce nemá žádné tělo.

```
class PlosnyUtvar
{ protected:
    float a;
    public:
        virtual float obsah() =0;
};
```

Abstraktní třída

Obsahuje aspoň jednu čistě virtuální funkci. Nelze deklarovat (vytvořit) žádný její objekt (čistě virtuální funkce nemá definici – nemá tělo). Abstraktní třídu lze použít výlučně jako základní třídu pro dědění – abstraktní základní třída.

Řešení předchozí úlohy bez použití virtuální funkce

```
class PlosnyUtvar
{ protected:
    float a;
};

enum Typ { CTVEREC, OBDELNIK, KRUH };

struct TypUkaz { Typ typ; PlosnyUtvar *u; };

TypUkaz tu[]={ { CTVEREC, new Ctverec(3) },
               { OBDELNIK, new Obdelnik(2,3) },
               { KRUH, new Kruh(2) }
};
```

```

for (int i=0;i<3;++i) { PlosnyUtvar *u=tu[i].u;
    switch (tu[i].typ)
    { case CTVEREC:  cout << ((Ctverec *)u)->obsah() << endl;
                        break;
      case OBDELNIK: cout << ((Obdelnik *)u)->obsah() << endl;
                        break;
      case KRUH:     cout << ((Kruh *)u)->obsah() << endl;
                        break;
    }
}

```

Dědění více tříd

Lze vytvořit dědicí třídu současným děděním více tříd.

```

class A1 { };

class A2 { };

class A3 { };

class B: public A1, public A2, public A3 { };

```

V konstruktoru dědicí třídy musíme uvést volání těch konstruktorů děděných tříd, které mají parametry.

```

class A1 { A1(...) { } };

class A2 { A2() { } };

class A3 { A3(...) { } };

class B: public A1, public A2, public A3
{
    B(...): A1(...), A3(...) { }
};

```

Pokud se podědí členy tříd se stejným jménem, je při jejich použití nutné operátorem rozlišení stanovit, který z členů se používá.

```

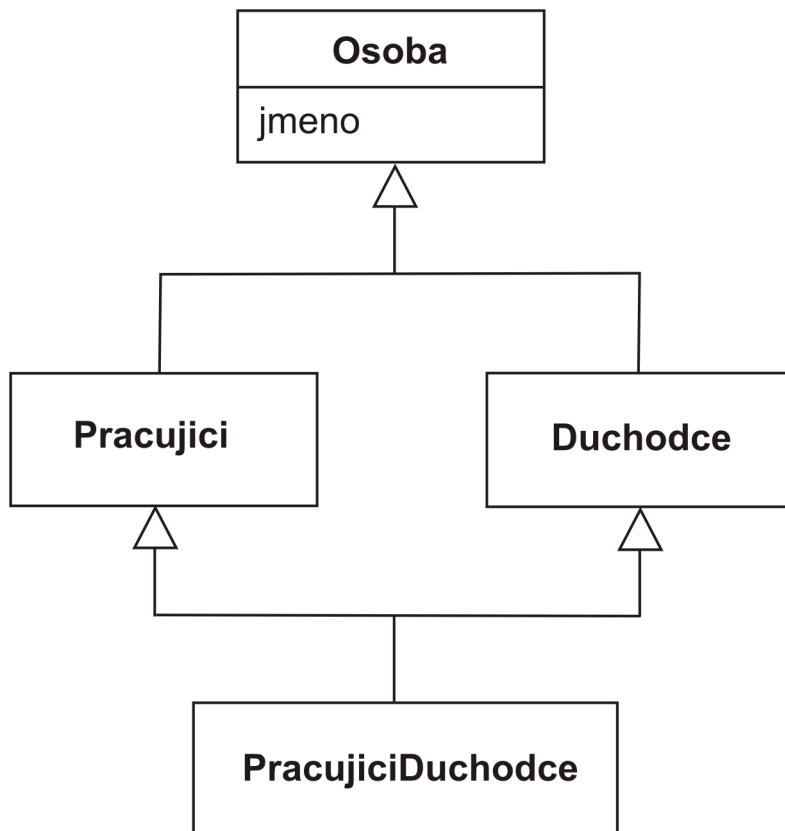
class A1
{
    protected: float a;
};

class A2
{
    protected: int a;
};

```

```
class B: public A1, public A2
{
    float f() { return A1::a * A2::a; }
};
```

Zdvojení členů vzniklé děděním dvou třídy, které samy vznikly děděním stejné třídy.



```
class Osoba
{ protected:
    string jmeno;
};

class Pracujici: public Osoba { };

class Duchodce: public Osoba { };

class PracujiciDuchodce: public Pracujici, public Duchodce
{
    // jméno zděděno 2× - Pracujici::jmeno a Duchodce::jmeno
};
```

Virtuální základní třída

Je základní třída, která je v deklaraci dědicí třídy označena klíčovým slovem **virtual**. Pokud dojde k násobnému dědění členů virtuální základní třídy, jsou její členy děděny jen jedenkrát.

```
class Osoba
{ protected:
```

```
    string jmeno;
};

class Pracujici: virtual public Osoba { };

class Duchodce: virtual public Osoba { };

class PracujiciDuchodce: public Pracujici, public Duchodce
{
    // proměnná jmeno je zděděna jen jedenkrát
};
```

Omezení dědičnosti

Je-li třída označena klíčovým slovem **final**, nelze ji dědit.

```
class A { };

class B final: public A { };

class C: public B { };           // chyba!! třídu B nelze dědit
```