Обектно ориентирано програмиране

НАСЛЕДЯВАНЕ. ПРОИЗВОДНИ КЛАСОВЕ

Преобразуване на типовете

Ако основният клас, който се наследява от производния клас е с атрибут **public**, възможно е взаимно заменяне на обекти от двата класа. Заменянето може да се извършва при инициализиране, при присвояване и при предаване на параметри на функции. Могат да се заменят обекти, псевдоними на обекти, указатели към обекти и указатели към методи. Замяната в посока “производен с основен” се счита за безопасна, докато замяната в обратната посока “основен с производен” може да предизвика проблеми.

Процесът на замяна е свързан с преобразувания, които за различните случаи са явни или неявни.

Преобразуване на типовете

...

За да покажем тези преобразувания ще използваме класовете base и der, дефинирани по следния начин:

```
class base
{
public:
   base(int x = 0) { b = x; }
   int get_b() const { return b;}
   void f() { cout << "b: " << b << endl; }
private:
   int b;
};
```

Преобразуване на типовете

...

```
class der : public base
{
public:
   der(int x = 0) : base(x) { d = 5; }
   int get_d() const { return d; }
   void f_der() {
      cout << "class der: d: " << d
      << " b: " << get_b() << endl;
   }
private:
   int d;
};
```

Преобразуване на типовете ...

Преобразуване в посока “производен с основен”

Обект, псевдоним на обект или указател към обект на производен клас се преобразуват съответно в обект, псевдоним на обект или указател към обект на основен клас чрез ***неявни стандартни преобразувания***. На практика тези преобразувания се свеждат до използване само на наследените компоненти на класа. Последното се основава на факта, че производният клас наследява всички свойства на базовия клас и може да бъде използван вместо него.

Преобразуване на типовете

...

Пример: В резултат от изпълнението на фрагмента:

```
der d; d.f_der();
base x = d; x.f();
der &d1 = d; d1.f_der();
base &y = d1; y.f();
der *d2 = &d; (*d2).f_der();
base *z = d2; (*z).f();
```

се получава

```
class der: d: 5 b: 0
b: 0
class der: d: 5 b: 0
b: 0
class der: d: 5 b: 0
b: 0
```

Преобразуване на типовете ...

Преобразуване в посока “основен производен”

Тъй като основният клас не съдържа собствените компоненти на производния клас, това преобразуване се осъществява само чрез явно указване.

Най-често се срещат следните случаи:

а) Присвояване и инициализиране на обект от производен клас с обект на основен клас

Нека x е обект на класа base, а y е обект на производния му клас der. Искаме на y да присвоим x. Стандартно се реализира чрез явно преобразуване на x в обект на клас der, т.е.

```
base x;

der y = (der)x;
```

Операцията е опасна, тъй като собствените компоненти на обекта y ще останат неинициализирани и *опитът за промяната им може да доведе до сериозни последици*. Затова някои реализации на езика, в това число и Visual C++, **не реализират това преобразуване**.

Преобразуване на типовете ...

Подобна е ситуацията при използване на указатели към обекти:

```
base *pb = new base;
der* pd = (der*)pb;
```

Извършва се явно преобразуване на pb в указател към обект на клас der. Указателят pd към обект на der не сочи към *истински обект* от клас der. Областта в паметта, свързана с указателя pd, няма собствени компоненти на класа der. Опитът за използването им **може** да предизвика сериозни проблеми, тъй като ще се използва памет, която е определена за други цели. Някои реализации на езика не реализират това преобразуване. Visual C++ 6.0 го извършва. От примера по-долу се вижда, че се извежда случайна стойност, но опитът за промяна на тази памет, **може** да е с непредвидими за програмата последици. Дефиниция на указателя pd към der може да се използва за извикване на собствена член-функция на der.

Преобразуване на типовете

...

Пример: Изпълнението на фрагмента:

```
base *pb = new base;
(*pb).f();      // или pb->f();
der *pd = (der*)pb;
(*pd).f_der();// или pd->f_der();
cout << pd->get_b() << endl;
```

води до следния резултат:

```
b: 0
class der: d: -33686019 b: 0
0
```

Преобразуване на типовете ...

б) Достъп до собствени членове на производния клас чрез обект на основния клас

Такъв достъп директно не е възможен. Непряк достъп е възможен и се осъществява чрез указатели и преобразувания.

Пример:

```
der y;
der *pd = &y;  // инициализация на указателя pd
    // към обект на der
base *pb = pd; // неявно преобразуване
```

Указателите pb и pd сочат към обекта y на класа der, но са различни. Компилаторът проверява допустимостта на използването им в зависимост **за кой клас се отнася**.

Преобразуване на типовете

...

Обръщението

```
pd->f_der();
```

е допустимо и активира, чрез указателя pd към обекта y, метода f_der(), но обръщението

```
pb->f_der();
```

е недопустимо, тъй като pb е указател към base, който няма f_der() за свой метод. За да стане възможно последното трябва да се използва явно преобразуване

```
((der *) pb)->f_der();
```

Ще напомним, че казаното се отнася само за производни класове с атрибут за област public на основния клас. За производни класове с атрибути за област private и protected не са дефинирани подобни преобразувания. Това е естествено, тъй като в тези случаи производният клас не притежава всички свойства на базовия и не може да го замести.

C++ явно преобразуване

- dynamic_cast,
- reinterpret_cast,
- static_cast
- const_cast.

C++ dynamic_cast

dynamic_cast може да се използва само с указатели и псевдоними. Изпълнява се по време на изпълнението на програмата

```
class CBase { };
class CDerived : public CBase { };
CBase b;
CBase* pb;
CDerived d;
CDerived* pd;
pb = dynamic_cast<CBase *>(&d);
// ok: derive d to base
pd = dynamic_cast<CDerived*>(&b);
// allowed but it returns NULL
```

C++ static_cast

```
class CBase { };
 class CDerived : public CBase { };
 CBase * a = new CBase;
 CDerived * b = static_cast<CDerived*>(a); //OK
```

Шаблони на класове и наследяване

Използването на шаблони на класове и наследяването може да се разгледа в следните случаи:

шаблон на клас – наследник на обикновен клас

В този случай, ако base е обикновен клас, т.е.

```
class base
{
        <тяло>
};
```

декларацията на производния на base шаблон на клас der ще има вида:

```
template <class T>
class der:<атрибут_за_област> base
{
        <тяло>
};
```

Шаблони на класове и наследяване …

обикновен клас – наследник на шаблон на клас

Ако tempbase е шаблон на основен клас

```
template <class T>
class tempbase
{<тяло>
};
```

декларацията на обикновен производен клас на шаблона tempbase трябва да задава стойност на параметъра за тип на tempbase, например int в случая и има вида

```
class der:<атрибут_за_област> tempbase<int>
{<тяло>
};
```

Шаблони на класове и наследяване …

шаблон на клас – наследник на шаблон на клас

Нека е деклариран шаблон на клас tempbase с параметър T:

```
template <class T>
class tempbase
{<тяло>
};
```

Декларацията на производен шаблон на клас на шаблона tempbase може да стане по два начина. При първия, производният шаблон на клас запазва същия параметър за тип като базовия, т.е.

```
template <class T>
class tempder1: атрибут_за_област tempbase<T>
{<тяло>
};
```

Шаблони на класове и наследяване ...

В този случай на всеки възможен базов клас съответства точно един производен. При втория начин, производният шаблон на клас въвежда и други параметри за тип:

```
template <class T, class U, ...>
class tempder2 : атрибут_за_област tempbase<T>
{<тяло>
};
```

При тази декларация на всеки възможен базов клас съответства фамилия от производни класове.