

# 派生类的构造函数

郭 炜 刘家瑛





# 衍生类的构造函数

- 衍生类对象 包含 基类对象
- 执行衍生类构造函数之前, 先执行基类的构造函数
- 衍生类交代基类初始化, 具体形式:  
构造函数名(形参表): 基类名(基类构造函数实参表)  
{  
}



# 派生类的构造函数

```
class Bug {  
    private :  
        int nLegs;    int nColor;  
public:  
    int nType;  
    Bug (int legs, int color);  
    void PrintBug () {  };  
};  
class FlyBug: public Bug { // FlyBug是Bug的派生类  
    int nWings;  
public:  
    FlyBug(int legs, int color, int wings);  
};
```



```
Bug::Bug( int legs, int color) {  
    nLegs = legs;  
    nColor = color;  
}
```

//错误的FlyBug构造函数:

```
FlyBug::FlyBug (int legs, int color, int wings) {  
    nLegs = legs;    // 不能访问  
    nColor = color;  // 不能访问  
    nType = 1;       // ok  
    nWings = wings;  
}
```

//正确的FlyBug构造函数:

```
FlyBug::FlyBug (int legs, int color, int wings):Bug(legs, color) {  
    nWings = wings;  
}
```

表达式中可以出现:  
FlyBug构造函数的参数



```
int main() {  
    FlyBug fb ( 2,3,4);  
    fb.PrintBug();  
    fb.nType = 1;  
    fb.nLegs = 2 ; // error.nLegs is private  
    return 0;  
}
```



```
FlyBug fb (2,3,4);
```

- 在创建派生类的对象时,
  - 需要调用基类的构造函数:
    - 初始化派生类对象中从基类继承的成员
  - 在执行一个派生类的构造函数之前,
    - 总是先执行基类的构造函数



## 调用基类构造函数的两种方式

- 显式方式:

派生类的构造函数中 → 基类的构造函数提供参数

```
derived::derived(arg_derived-list):base(arg_base-list)
```

- 隐式方式:

派生类的构造函数中, 省略基类构造函数时

派生类的构造函数, 自动调用基类的默认构造函数

- 派生类的析构函数被执行时, 执行完派生类的析构函数后, 自动调用基类的析构函数



```
class Base {
    public:
        int n;
        Base(int i):n(i)
        {    cout << "Base " << n << " constructed" << endl;    }
        ~Base()
        {    cout << "Base " << n << " destructed" << endl;    }
};

class Derived:public Base {
    public:
        Derived(int i):Base(i)
        {    cout << "Derived constructed" << endl;    }
        ~Derived()
        {    cout << "Derived destructed" << endl;    }
};

int main() {    Derived Obj(3); return 0; }
```





## ▀ 输出结果:

Base 3 constructed

Derived constructed

Derived destructed

Base 3 destructed

# 包含成员对象的派生类的构造函数

```
class Skill {  
    public:  
        Skill(int n) { }  
};  
class FlyBug: public Bug {  
    int nWings;  
    Skill sk1, sk2;  
    public:  
        FlyBug(int legs, int color, int wings);  
};  
FlyBug::FlyBug( int legs, int color, int wings):  
    Bug(legs, color), sk1(5), sk2(color) {  
    nWings = wings;  
}
```

表达式中可以出现:  
FlyBug构造函数的  
参数, 常量



- 创建 派生类的对象 时, 执行 派生类的构造函数 之前:
  - 调用 基类 的构造函数
    - 初始化派生类对象中从基类继承的成员
  - 调用 成员对象类 的构造函数
    - 初始化派生类对象中成员对象
- 执行完 派生类的析构函数 后:
  - 调用 成员对象类 的析构函数
  - 调用 基类 的析构函数
- 析构函数的调用顺序与构造函数的调用顺序相反