

# 虚析构函数

郭 炜 刘家瑛






# 虚析构函数

## 问题:

```
class CSon{
    public: ~CSon() {  };
};
class CGrandson : CSon{
    public: ~CGrandson() {  };
}
int main(){
    CSon *p = new CGrandson;
    delete p;
    return 0;
}
```





# 虚析构函数

- 通过 基类的指针 删除 派生类对象 时

→ 只调用基类的析构函数

**Vs.**

- 删除一个 派生类的对象 时

→ 先调用 派生类的析构函数

→ 再调用 基类的析构函数



# 虚析构造函数

- 解决办法:
- 把基类的析构造函数声明为 **virtual**
  - 派生类的析构造函数 **virtual** 可以不进行声明
  - 通过 **基类的指针** 删除 派生类对象 时
    - 首先调用 派生类的析构造函数
    - 然后调用 基类的析构造函数
- 类如果定义了虚函数, 则最好将析构造函数也定义成 **虚函数**

**Note:** 不允许以虚函数作为构造函数



```
class son{
    public:
        ~son() { cout<<"bye from son"<<endl; };
};
class grandson : public son{
    public:
        ~grandson(){ cout<<"bye from grandson"<<endl; };
};
int main(){
    son *pson;
    pson=new grandson;
    delete pson;
    return 0;
}
```

输出结果: bye from son

没有执行grandson::~~grandson()!!!



```
class son{
    public:
        virtual ~son() { cout<<"bye from son"<<endl; };
};
class grandson : public son{
    public:
        ~grandson(){ cout<<"bye from grandson"<<endl; };
};
int main() {
    son *pson;
    pson= new grandson;
    delete pson;
    return 0;
}
```

输出结果: bye from grandson  
bye from son

执行grandson::~~grandson(),  
引起执行son::~~son()!!!