

# 基类/派生类同名成员 与Protected关键字

郭 炜 刘家瑛



北京大学



# 基类和派生类有同名成员的情况

```
class base {  
    int j;  
public:  
    int i;  
    void func();  
};
```

```
class derived : public base{  
    public:  
        int i;  
        void access();  
        void func();  
};
```



# 基类和派生类有同名成员的情况

```
class base {  
    int j;  
public:  
    int i;  
    void func();  
};
```

```
class derived : public base{  
    public:  
        int i;  
        void access();  
        void func();  
};
```



```
void derived::access()
```

```
{
```

```
    j = 5;    //error
```

```
    i = 5;    //引用的是派生类的 i
```

```
    base::i = 5; //引用的是基类的 i
```

```
    func();   //派生类的
```

```
    base::func(); //基类的
```

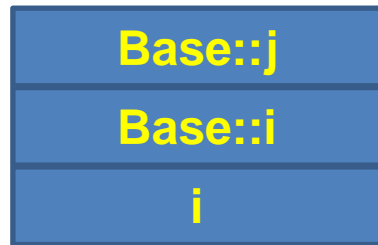
```
}
```

```
derived obj;
```

```
obj.i = 1;
```

```
obj.base::i = 1;
```

Obj占用的存储空间



Note: 一般来说，基类和派生类不定义同名成员变量



# 访问范围说明符

- 基类的private成员: 可以被下列函数访问
  - 基类的成员函数
  - 基类的友员函数
- 基类的public成员: 可以被下列函数访问
  - 基类的成员函数
  - 基类的友员函数
  - 派生类的成员函数
  - 派生类的友员函数
  - 其他的函数



# 访问范围说明符: `protected`

- 基类的`protected`成员: 可以被下列函数访问
  - 基类的成员函数
  - 基类的友员函数
  - 派生类的成员函数可以访问当前对象的基类的保护成员



# 保护成员

```
class Father {  
    private: int nPrivate;    //私有成员  
    public:  int nPublic;     //公有成员  
    protected: int nProtected; // 保护成员  
};  
class Son : public Father {  
    void AccessFather () {  
        nPublic = 1; // ok;  
        nPrivate = 1; // wrong  
        nProtected = 1; // OK, 访问从基类继承的protected成员  
        Son f;  
        f.nProtected = 1; //wrong, f不是当前对象  
    }  
};
```



```
int main(){  
    Father f;  
    Son s;  
    f.nPublic = 1; // Ok  
    s.nPublic = 1; // Ok  
    f.nProtected = 1; // error  
    f.nPrivate = 1; // error  
    s.nProtected = 1; //error  
    s.nPrivate = 1; // error  
    return 0;  
}
```