

函数重定义和 派生类的继承方式

- 对于基类中的成员函数，可以在派生类中对其重新定义、实现新的功能。
例如，我们可以在Student和Person类中都定义一个DisplayInfo()函数：

```
void Person::DisplayInfo()
{
    cout<<"个人信息："<<endl
        <<"姓名："<<m_name<<endl
        <<"性别：";
    if (m_sex==true) cout<<"男"<<endl;
    else cout<<"女"<<endl;
}
```

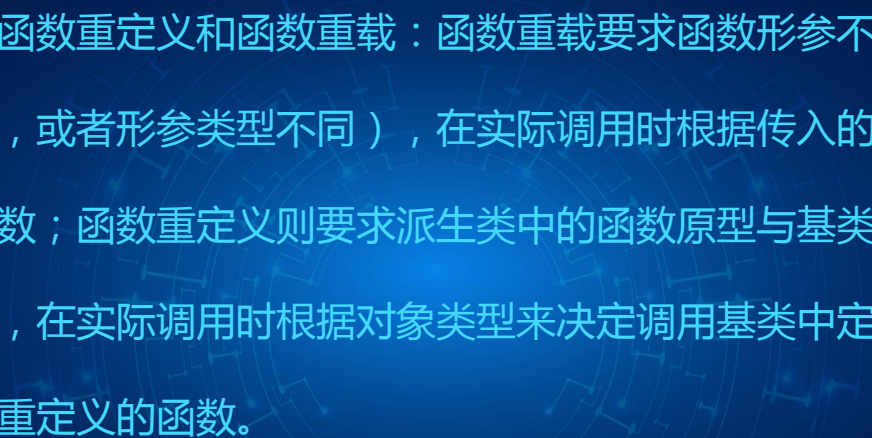
```
void Student::DisplayInfo()
{
    cout<<"学生信息："<<endl
        <<"学号："<<m_sno<<endl
        <<"姓名："<<GetName()<<endl
        <<"性别：";
    if (GetSex()==true) cout<<"男"<<endl;
    else cout<<"女"<<endl;
    cout<<"专业："<<m_major<<endl;
}
```

- 这样，Student类对从Person类继承的DisplayInfo()函数进行了重定义。当使用Person类对象调用时，就会调用Person类中定义的DisplayInfo()函数；而当使用Student类对象调用时，就会调用Student类中定义的DisplayInfo()函数。例如，在主函数中使用如下语句：

```
Person person;  
person.SetName("李四");  
person.SetSex(false);  
person.DisplayInfo();
```

可以在屏幕上输出：

```
个人信息：  
姓名：李四  
性别：女
```



提示：区分函数重定义和函数重载：函数重载要求函数形参不同（或者形参个数不同，或者形参类型不同），在实际调用时根据传入的实参来决定调用哪个函数；函数重定义则要求派生类中的函数原型与基类中的函数原型完全一样，在实际调用时根据对象类型来决定调用基类中定义的函数还是派生类中重定义的函数。

- 在定义派生类时，可以指定的继承方式包括public（公有继承）、protected（保护继承）和private（私有继承，缺省方式）3种。派生类从基类继承过来的成员的访问控制方式由两点决定：该成员在基类中的访问控制方式；定义派生类所采用的继承方式。

| 访问控制方式 继承方式 | public | private | protected |
|----------------|-----------|---------|-----------|
| public | public | 不可访问 | protected |
| private | private | 不可访问 | private |
| protected | protected | 不可访问 | protected |