面向对象方法与C++程序设计

第4章　　继承

大连理工大学
主讲人-赵小薇

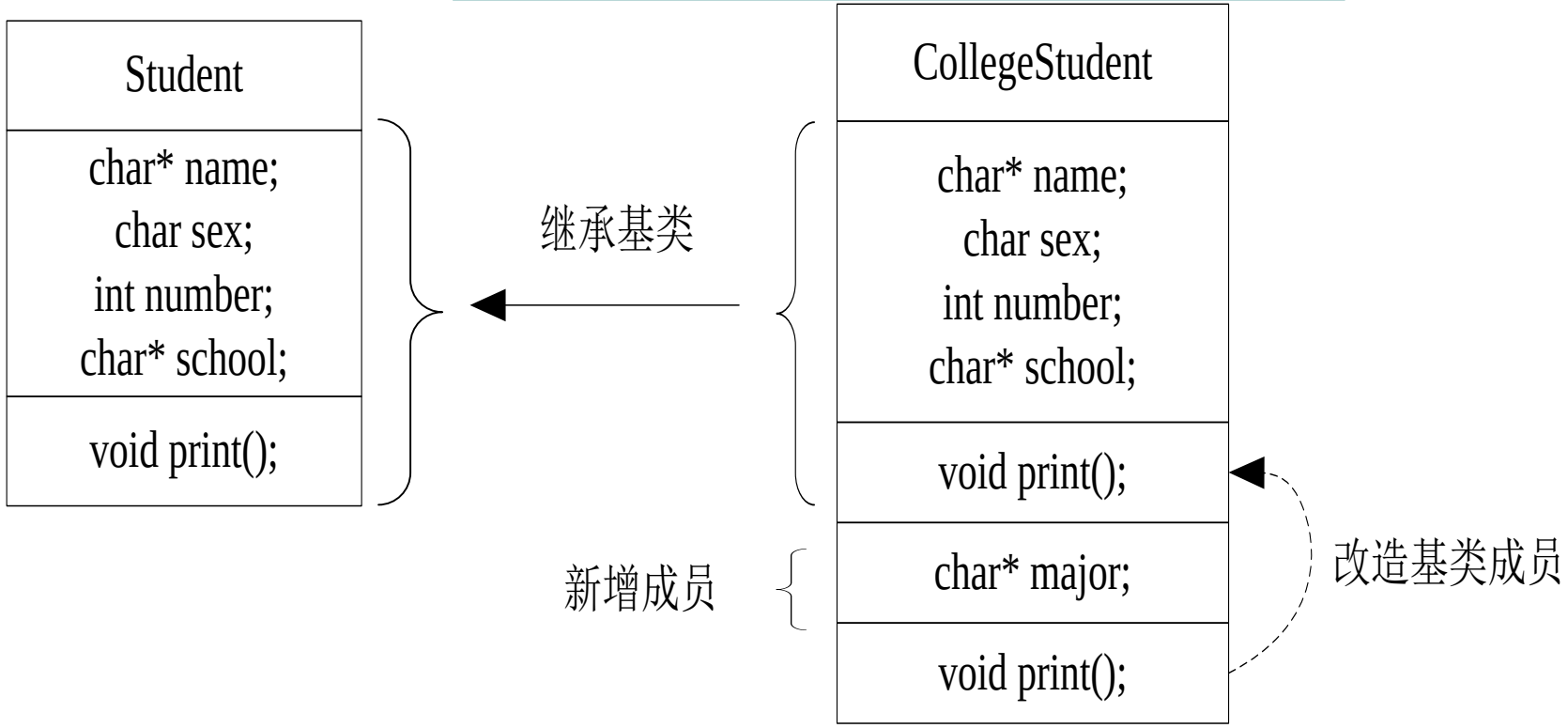
派生类的构成

派生类与基类的成员表现出相同和不同，派生类构成有三个步骤

接收基类成员

改造基类成员

添加新的成员



继承中的访问控制



- C++的继承方式有：公有继承、私有继承、保护继承

- 继承方式决定了基类成员在派生类中的访问权限，这种访问来自两个方面：
  - 派生类中的新增函数成员访问从基类继承来的成员
  - 在派生类外部（非类族内的成员），通过派生类的对象访问从基类继承的成员

公有继承



公用基类成员	在派生类中的访问属性
公有成员public	公有public
保护成员protected	保护protected
私有成员private	不可访问

访问公有继承的成员举例



```
class Student {                          // 学生类(基类)
protected:
    char name[10];
     char sex;
     int number;
     char school[10];                    // 学校
public:
     void input_data( );                 // 输入学生信息函数
     void print( );                      // 输出学生信息函数
};
```

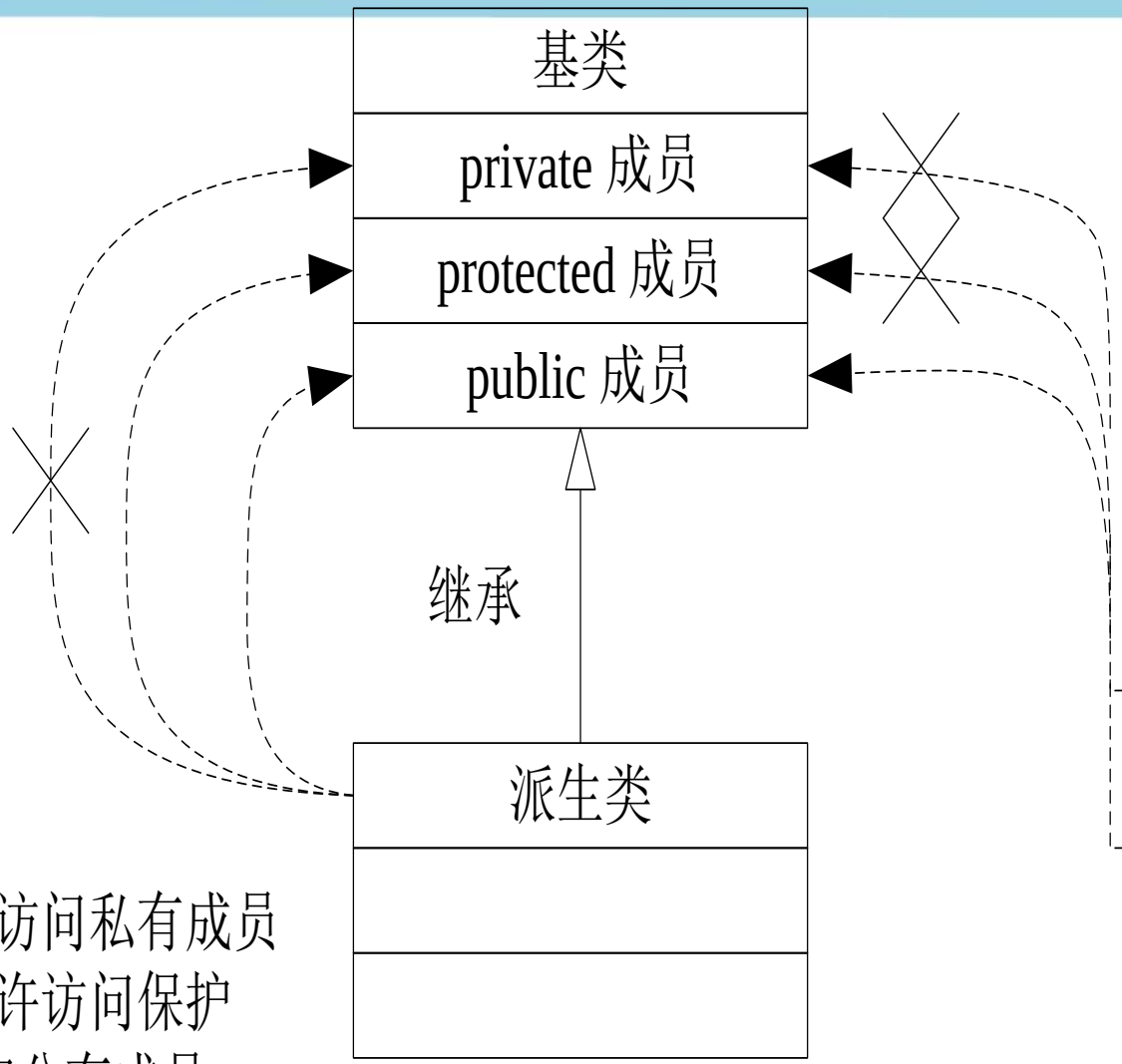
protected
访问权限？



```
class CollegeStudent : public Student{   //公有继承
private:      char major[10];                          //专业
public:
   void input_major( );                                //
   void print( ){                                      // 输出信息
     cout<< "name:" <<name << " sex:" <<sex <<endl;
    //访问基类
      cout<<"major:"<<major<<endl;      }
};
```

派生类中可直接访问基类保护成员

```
void main(){                                    //主函数
CollegeStudent cs;    cs.input_data();
cs.input_major();      cs.sex ='F';   //error!
cs.print();  }
```


基类
private 成员
protected 成员
public 成员
继承
派生类
派生类对象
继承树以外使用者
不能访问私有成员和保护成员
允许访问公有成员
不能访问私有成员
允许访问保护和公有成员

保护成员



- 访问属性为protected，表示这些成员可以被其基类和派生类访问，但是在继承树以外的类成员无法访问。

私有继承



私有基类成员	在派生类中的访问属性
公有成员public	私有private
保护成员protected	私有private
私有成员private	不可访问

私有继承举例



对于CollegeStudent来说，它访问基类Student的能力没有变化，但是所有它所继承的成员其属性全部变为私有。
事实上，在私有继承的情况下，通过派生类对象无法访问基类的任何成员。

```
class Student {
 public: void input_data( );
       ......
};
class CollegeStudent : private Student{          // 私有继承
    ......
};
int main(){
     CollegeStudent cs;
     cs.input_data();                            // 错误
      cs.input_major();
     cs.print();
      return 0;   }
```

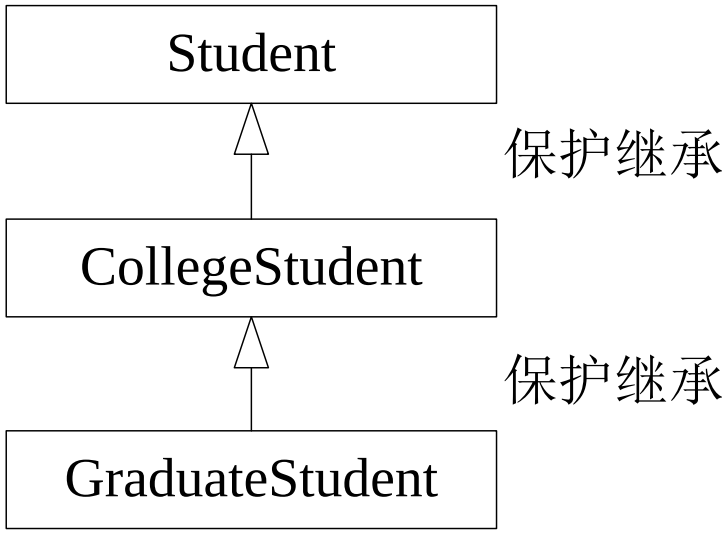
保护继承



保护基类成员	在派生类中的访问属性
公有成员public	保护protected
保护成员protected	保护protected
私有成员private	不可访问

保护继承举例







```
class Student {
    protected:
            char name[10];                  // 姓名
            char sex;                       // 性别
            int number;                     // 学号
            char school[10];                // 学校
    public:
            void input_data( ){...} // 输入学生信息函数
            void print(){...} // 输出学生信息函数
};
```



```
class CollegeStudent : protected Student{// 保护继承
protected:
    char major[10];                          // 专业
public:
    void input_major( ){cin>>major;} // 输入专业信息函数
    void print( ){                           // 输出大学生信息函数
     cout<<"name:"<<name<<endl;        // 允许访问基类保护成员
     cout<<"sex:"<<sex<<endl;          // 允许访问基类保护成员
     cout<<"number:"<<number<<endl;    // 允许访问基类保护成员
     cout<<"school:"<<school<<endl;    // 允许访问基类保护成员
     cout<<"major:"<<major<<endl;
   }
};
```



```
class GraduateStudent : protected CollegeStudent{
private:          char tutor[10];                    // 导师信息
public:
    // 输出研究生信息函数
   void print( ){
     cout<<"name:"<<name<<endl;       // 允许访问间接基类保护成员
     cout<<"sex:"<<sex<<endl;         // 允许访问间接基类保护成员
     cout<<"number:"<<number<<endl;   // 允许访问间接基类保护成员
     cout<<"school:"<<school<<endl;     // 允许访问间接基类保护成员
     cout<<"major:"<<major<<endl;      // 允许访问直接基类保护成员
     cout<<"tutor:"<<tutor<<endl;
     }
};
```



```
int main(){
    GraduateStudent gs;
    gs.input_data();        // 错误
    gs.input_major();       // 错误
    return 0;
}
```

三种继承方式的对比



- 私有继承：基类的可被继承的成员都成了其直接派生类的私有成员，无法再进一步派生，实际上私有继承相当于终止了基类成员的继续派生。
- 保护继承：基类的可被继承的成员都成了其直接派生类的保护成员，依然能被继承树中的次级子类所继承。
  - 一般情况下私有继承和保护继承的设计比较少见。一般采用不会改变基类成员访问权限的公有继承。
- 私有继承和保护继承：改变了基类成员继承后的访问属性，但对于其直接子类来说，这两种方式实际上是相同的。

最常用的继承方式是什么方式?