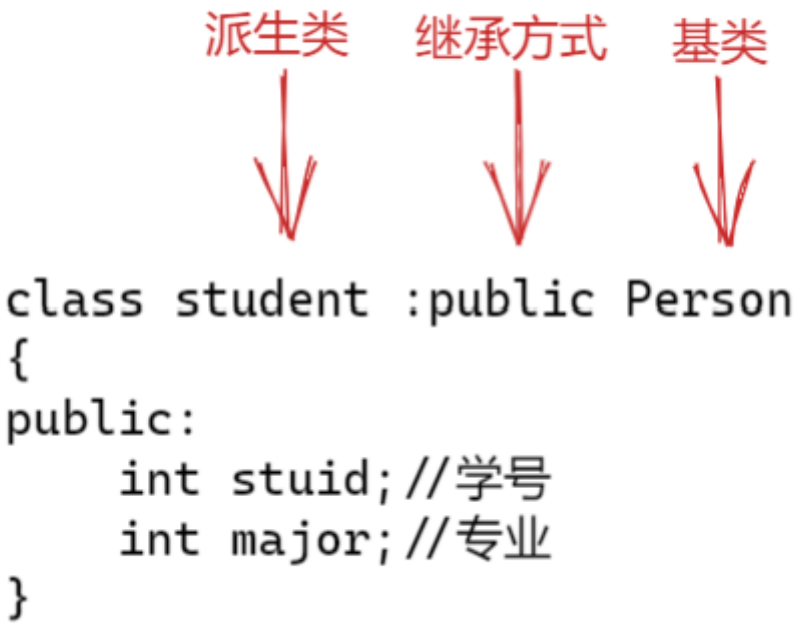
继承

继承(inheritance)机制是面向对象程序设计代码**复用**的手段，它允许程序员在**保持原有类特性**的基础上进行扩展，增加功能，这样产生新的类，我们成为**派生类**。

继承定义

定义格式



继承方式和访问限定符

1. 继承方式
    - public继承、protected继承、private继承
2. 访问限定符
    - public访问、protected访问、private访问

继承基类成员访问的变化

类成员/继承方式	public继承	protected继承	private继承
基类的public成员	派生类的public成员	派生类的protected成员	派生类的private成员
基类的protected成员	派生类的protected成员	派生类的protected成员	派生类的private成员
基类的private成员	在派生类中不可见	在派生类中不可见	在派生类中不可见

- 基类private成员在派生类中无论什么方式继承都是不可见的，这里的不可见指的是**基类的私有成员还是被继承到了派生类中，只是语法上限制派生类对象不管在类里面还是类外都不能访问它**
- 如果基类成员不想在类外直接被访问，但需要在派生类中能访问，就定义为protected(protected访问限定符是因为继承才出现的)
- 使用关键字class时默认继承方式是private，使用关键字struct时默认继承方式是public
- 总结：
  - **基类的私有成员无论什么继承方式在派生类中都是不可见的；**
  - **基类的其他成员在派生类中的访问方式=min(成员在基类中的访问限定，继承方式)，public>protected>private**

基类和派生类赋值兼容规则

1. 派生对象可以赋值给基类对象/指针/引用
2. 基类对象不能赋值给派生类对象
3. 基类的指针可以通过强制类型转换赋值给派生类的指针

```
#include<iostream>
#include<string>
using namespace std;
class Person
{
protected:
    string _name;//姓名
    int _age;
};
class Student :public Person
{
    string _id;//学号
};

int main()
{
    Person p;
    Student s;
    //派生类可以赋值给基类对象/引用/指针
    p = s;
    Person& p1 = s;
    Person *p2 = &s;

    //基类不能赋值给派生类对象
    //s = p;

    //基类的指针可以通过强制类型转换赋给派生类的指针
    Student* s1 = (Student*)p2;

    return 0;
}
```

继承中的作用域

1. 在继承体系中基类和派生类都有独立的作用域
2. 子类和父类中有同名成员，子类成员将屏蔽父类对同名成员的直接访问，这种情况叫隐藏，也叫重定义。(在子类成员中，可以使用 基类::基类成员 来访问)

派生类的默认成员函数

1. 构造函数
    - 派生类的构造函数必须调用基类的构造函数初始化基类的那一部分成员。
    - 如果基类没有默认的构造函数，则必须在派生类构造函数的初始化列表显示调用。

```
Student(const string& name,int stuid)
    :Person(name)
    ,_stuid(stuid)
{

}
```

2. 析构函数
    - 派生类的析构函数会在被调用完成后**自动调用**基类的析构函数清理基类成员，这样才能保证派生类对象先清理、派生类对象成员再清理基类成员的顺序。

```
~Student()
{
    //派生类的析构函数会在被调用完成后自动调用基类的析构函数
}
```

3. 拷贝构造函数
    - 派生类的拷贝构造函数必须要调用基类的拷贝构造函数完成基类的拷贝构造。

```
Student(const Student& s)
    :Person(s)
    ,_stuid(s.stuid)
{

}
```

4. 赋值重载函数
    - 派生类的operator=必须要调用基类的operator=完成基类的赋值。

```
Student& operator=(consy Student& s)
{
    if(this!=&s)
    {
        Person::operator=(s);
        _stuid=s._stuid;
    }
}
```

5. 派生类对象初始化先调用基类构造再调用派生类构造。
6. 派生类对象析构先调用派生类析构再调用基类析构。

继承与友元和静态成员

1. 友元关系不能被继承，也就是说基类友元不能访问派生类的私有和保护成员

2. 基类定义了static静态成员，则整个继承体系里面只有一个这样的成员。无论派生出多少个子类，都只有一个static成员实例

菱形继承

1. 单继承——一个子类只有一个直接父类
2. 多继承——一个子类有两个或以上直接父类
3. 菱形继承的产生了数据冗余和二义性，为了解决引入了虚继承(virtual关键字)
4. 继承和组合的区别——都完成类层次的复用

```
class A
{
}
class B: public A
{
}
```

A和B的关系就是继承，继承是一种白箱复用，在继承中，基类的内部细节对子类可见，继承一定程度上破坏了基类的封装，基类的改变对派生类有很大的影响，派生类和基类间依赖关系很强，耦合度很高。

```
class C
{
}
class D
{
    C c;
}
```

组合是一种黑箱复用，对象的内部细节是不可见的，优先使用组合会保持每个类的封装性。