类的继承：继承类可以访问基类的public成员和protected成员，但不可以访问基类的private成员。

例：程序class\_test13

advisor.h

#ifndef ADVISOR\_H

#define ADVISOR\_H

class Advisor

{

public:

Advisor(int number);

private:

int number\_of\_meeting\_;

};

#endif // ADVISOR\_H

advisor.cpp

#include "advisor.h"

#include <iostream>

Advisor::Advisor(int number)

{

number\_of\_meeting\_ = number;

std::cout << "Advisor()" << std::endl;

}

student.h

#ifndef STUDENT

#define STUDENT

class Student

{

public:

Student(char\* name\_ptr);

void AddCourse(int hours, float grade);

int GetHours();

float GetAverage();

void Display();

protected:

char name\_[40];

int semester\_hours\_;

float average\_;

};

#endif // STUDENT

student.cpp

#include <iostream>

#include <string>

#include "student.h"

using namespace std;

Student::Student(char\* pName)

{

cout << "Student()" << endl;

strncpy(name\_, pName, sizeof(name\_));

average\_ = semester\_hours\_ = 0;

}

void Student::AddCourse(int hours, float grade)

{

average\_ = (semester\_hours\_ \* average\_ + grade);//总分

semester\_hours\_ += hours;//总修学时

average\_ /= semester\_hours\_;

}

int Student::GetHours()

{

return semester\_hours\_;

}

float Student::GetAverage()

{

return average\_;

}

void Student::Display()

{

cout << "name= \"" <<name\_ << "\"" //注意这里输出的写法，若写成cout<<"name= \"<<name<<"\"，则会报错，若写成cout<<"name= "<<name，则输出的字符串没有引号

<< ", hours=" << semester\_hours\_

<< ", average=" << average\_ <<endl;

}

graduatestudent.h

#ifndef GRADUATESTUDENT

#define GRADUATESTUDENT

#include "advisor.h"

#include "student.h"

class GraduateStudent : public Student // 类的继承

{

public:

GraduateStudent();

public:

int GetQualifier();

protected:

Advisor advisor\_; //一个类以另外一个类对象作数据成员，叫类的组合

int qualifier\_grade\_;

};

#endif // GRADUATESTUDENT

graduatestudent.cpp

#include "graduatestudent.h"

#include <iostream>

// 必须在构造函数的初始化列表中对advisor\_对象初始化，因为Advisor类没有默认构造函数

// 必须调用父类Student的构造函数，因为Student没有默认的构造函数

GraduateStudent::GraduateStudent() : advisor\_(5), Student("no name")

{

std::cout << "GraduateStudent()" << std::endl;

}

int GraduateStudent::GetQualifier()

{

semester\_hours\_ = 0; // 可以直接访问父类的protected成员

return qualifier\_grade\_;

}

int main(int argc, char\* argv[])

{

GraduateStudent gs;

gs.AddCourse(3, 4.5);

gs.Display();

return 0;

}

输出为：

Student()

Advisor()

GraduateStudent()

name= "no name", hours=3, average=1.5

可以看出：构造派生类GraduateStudent的对象时，先调用父类Student的构造函数，然后调用成员变量Advisor的构造函数，最后调用GraduateStudent的构造函数。



GraduateStudent对象的内存布局

可以看出GraduateStudent类对象gs中包含有Student对象空间，用this指针访问Student成员与访问自己增加的成员没有任何区别。