**计算机科学与工程学院实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验课程名称** | | **C++实验课** | | | **实验总成绩** |  |
| **专业** | **物联网工程** | | **班级** | **物联网2002** | **指导教师签字** |  |
| **学号** | **20203578** | | **姓名** | **陆晓晓** | **实验报告批改时间** |  |
| **实验报告分项成绩**   |  |  |  | | --- | --- | --- | | **序号** | **实验项目** | **成绩** | | **1** | **类的相关 & 访问属性成员 &构造/析构函数** |  | | **2** | **数组 & 字符串 & 字符数组** |  | | **3** | **类的继承& 基类成员的访问控制 & 虚基类** |  | | **4** | **运算符重载 & 虚函数动态多态性** |  | | **5** |  |  | | | | | | | |
| **实验课程总结**  从以下方面总结：1.实验体现知识应用和初步研究能力；2. 反映基本观察、发现问题和分析问题能力；3. 实验项目内容或者实验课程是否存在问题及下一年度改进意见）4.其他方面   1. 通过这四节c++实验课，编写了关于类、字符串、类的继承、多态、运算符重载、虚函数等方面的代码。并且在编写代码的过程中，对所提出的要求进行分析，进行了更一步的实现，很好得完成了实验要求。对c++这门课的知识点覆盖面极广，使得书本上枯燥的代码得以在上机的过程中得以实践。虽然编写代码伊始，遇到了不少困难，但是通过不断的学习，我最终都能将问题很好得解决，在这个过程中，也加深了我们对各种c++的理解。我在这之中遇到了很多的障碍，但最终通过探索，将问题得到了很好得解决。极大的锻炼研究能力。这个研究能力是全方面的：问题的分析，问题的解决，问题的总结等多个方面。 2. 发现问题的能力得到很大的提升，在老师已经提出的要求之下，进行了更深一步的拓展，使得程序功能更加完善，效果更加好。分析问题的能力也有了很大的进步。在求解一些问题的时候，并没有很好的看清要求，使得很长一段时间去做一些无用的事情，浪费了极大的时间。我也从中吸取了教训：之后再完成此类任务时，一定要好好审题，好好得分析问题之后，再进行分工以及代码的编写。 3. 实验项目内容丰富，且贴近生活，使得学生能够很好得理解并解决问题，有实践性的同时又不失挑战性。实验课程设计比较合理，总共四节实验课，频率适中。对于一些代码能力强的同学，能在课上快速完成任务；对于一些能力偏弱的同学，需要通过询问同学或者查阅书籍等方式，也可将任务完成。   总体来说，c++实验课在各个方面都能使学生得到全方面的能力提升。希望c++实验课能够一直开设下去，使更多的同学在实验过程中得到进步。 | | | | | | |

**实验一**

1. **实验目的**

1.掌握类的定义和使用2.掌握类的定义和对象的声明3.复习具有不同访问属性的成员的访问方式4.观察构造函数和析构函数的执行过程5.学习类的组合使用方法6.使用IDE的debug调试功能观察程序流程，跟踪观察类的构造函数、析构函数、成员函数的执行顺序。

**2.实验要求**

1.定义一个CPU类，包含属性 等级（rank），rank为枚举类型CPU\_Rank,定义为 enum CPU\_Rank{P1=1,P2,P3,P4,P5,P6,P7} 频率（frequency），frequency为单位是MHz的整型数 电压（voltage），voltage为浮点型的电压值 有两个公有成员函数 run()、stop() 观察构造函数和析构函数的调用顺序。

2.定义并实现一个简单的Computer类 包含数据成员(芯片、内存以及光驱等): 芯片（cpu），cpu为CPU类的一个对象； 内存（ram），ram为RAM类的一个对象； 光驱（cdrom），cdrom为CDROM类的一个对象； 有两个公有成员函数run、stop。3. （选做）设计一个用于人事管理的People(人员)类。考虑到通用性这里只抽象出所有类型人员都具有的属性： 编号（number） 性别（sex） 身份证号（id）等 出生日期（birthday），这里出生日期定义为“日期” 类的内嵌子对象。用成员函数实现对人员信息的录入和显示。 要求包括：构造函数和析构函数、拷贝构造函数、内联成员函数、组合。

**3.实验内容（各题目的题干）**

1.首先定义枚举类型CPU\_Rank，例如enum CPU\_Rank{P1=1, P2, P3, P4, P5, P6, P7}，再定义CPU类，包含等级（rank）、频率（frequency）、电压（voltage）等私有数据成员，定义成员函数run、stop，用来输出提示信息，在构造函数和析构函数中也可以输出提示信息。在主程序中定义一个CPU的对象，调用其成员函数，观察类对象的构造与析构顺序，以及成员函数的调用。程序名：lab1\_1+学号.cpp 。

2.使用debug调试功能观察程序lab1\_1+学号.cpp的运行流程，跟踪观察类的构造函数、析构函数、成员函数的执行顺序。

3.调试操作步骤如下：（以VC++ 6.0为例）1）单击组建 | 开始调试 | Step Into命令，或按下快捷键F11，系统进入单步执行状态，程序开始运行，一个DOS窗口出现，此时Visual Studio 中光标停在main()函数的入口处；2）从Debug菜单或Debug工具栏中单击Step Over，此时，光标下移，程序准备执行CPU对象的初始化；3）单击Step Into，程序准备执行CPU类的构造函数；4）连续单击Step Over，观察构造函数；5）此时程序准备执行CPU对象的run()函数，单击Step Into，程序进入run()成员函数，连续单击Step Over，直到回到main()函数；6）继续执行程序，参照上述的方法，观察程序的执行顺序，加深对类的构造函数、析构函数、成员函数的执行顺序的认识；7）再试试Debug菜单栏中别的菜单项，熟悉Debug的各种方法。

4.首先定义CPU类（可使用第一题中的CPU类）、RAM类、CDROM类。再定义Computer类；定义私有数据成员cpu、ram、cdrom，定义公有成员函数run、stop，可在其中输出提示信息。在主程序中定义一个Computer的对象，调用其成员函数，观察类对象及其成员变量的构造与析构顺序，以及成员函数的调用。程序名：lab1\_2+学号.cpp 。

5.使用debug调试功能观察lab1\_2+学号.cpp程序的运行流程，跟踪观察类的构造函数、析构函数、成员函数的执行顺序，特别注意观察成员变量的构造与析构顺序。

**4.源程序（加注释）**

**1.1**

1. #include<iostream>
2. **using** **namespace** std;
3. **enum** CPU\_Rank { P1 = 1, P2, P3, P4, P5, P6, P7 };
4. **class** CPU {
5. **public**:
6. CPU\_Rank rank;
7. **int** frequency;
8. **double** voltage;
9. CPU(CPU\_Rank a, **int** b, **double** c) {
10. **this**->rank = a;
11. **this**->frequency = b;
12. **this**->voltage = c;
13. cout << "Construct CPU" << endl;
14. }
16. ~CPU()
17. {
18. cout << "Destruct CPU" << endl;
19. }
20. **void** Run()
21. {
22. cout << "CPU run" << endl;
23. }
24. **void** Stop()
25. {
26. cout << "CPU stop" << endl;
27. }
28. };


32. **int** main()
33. {
34. CPU a(P6, 300, 2.8);
35. a.Run();
36. a.Stop();
37. **return** 0;
38. }

**1.2**

1. #include<iostream>
2. **using** **namespace** std;
4. **class** CPU {
5. **public**:
6. CPU() {
7. cout << "Construct CPU" << endl;
8. }
9. CPU(**const** CPU& p) {
10. cout << "CPU copy constructor" << endl;
11. }
12. ~CPU() {
13. cout << "Destruct CPU" << endl;
14. }
15. };
17. **class** RAM {
18. **public**:
19. RAM() {
20. cout << "Construct RAM" << endl;
21. }
22. RAM(**const** RAM& p) {
23. cout << "RAM copy constructor" << endl;
24. }
25. ~RAM() {
26. cout << "Destruct RAM" << endl;
27. }
28. };
30. **class** CDROM {
31. **public**:
32. CDROM() {
33. cout << "Construct CDROM" << endl;
34. }
35. CDROM(**const** CDROM& p) {
36. cout << "CDROM copy constructor" << endl;
37. }
38. ~CDROM() {
39. cout << "Destruct CDROM" << endl;
40. }
41. };
43. **class** Computer {
44. **public**:
45. CPU cpu;
46. RAM ram;
47. CDROM cdrom;
48. Computer(CPU a, RAM b, CDROM c):cpu(a),ram(b),cdrom(c) {
49. cout << "Construct Computer " << endl;
50. }
51. Computer(Computer& p):cpu(p.cpu),ram(p.ram),cdrom(p.cdrom) {
52. cout << "Computer copy constructor" << endl;
53. }
54. ~Computer() {
55. cout << "Destruct Computer" << endl;
56. }
57. };
59. **int** main() {
60. CPU cpu;
61. RAM ram;
62. CDROM cd;
63. Computer computer(cpu, ram, cd);
64. Computer computer\_copy(computer);
66. **return** 0;
67. }

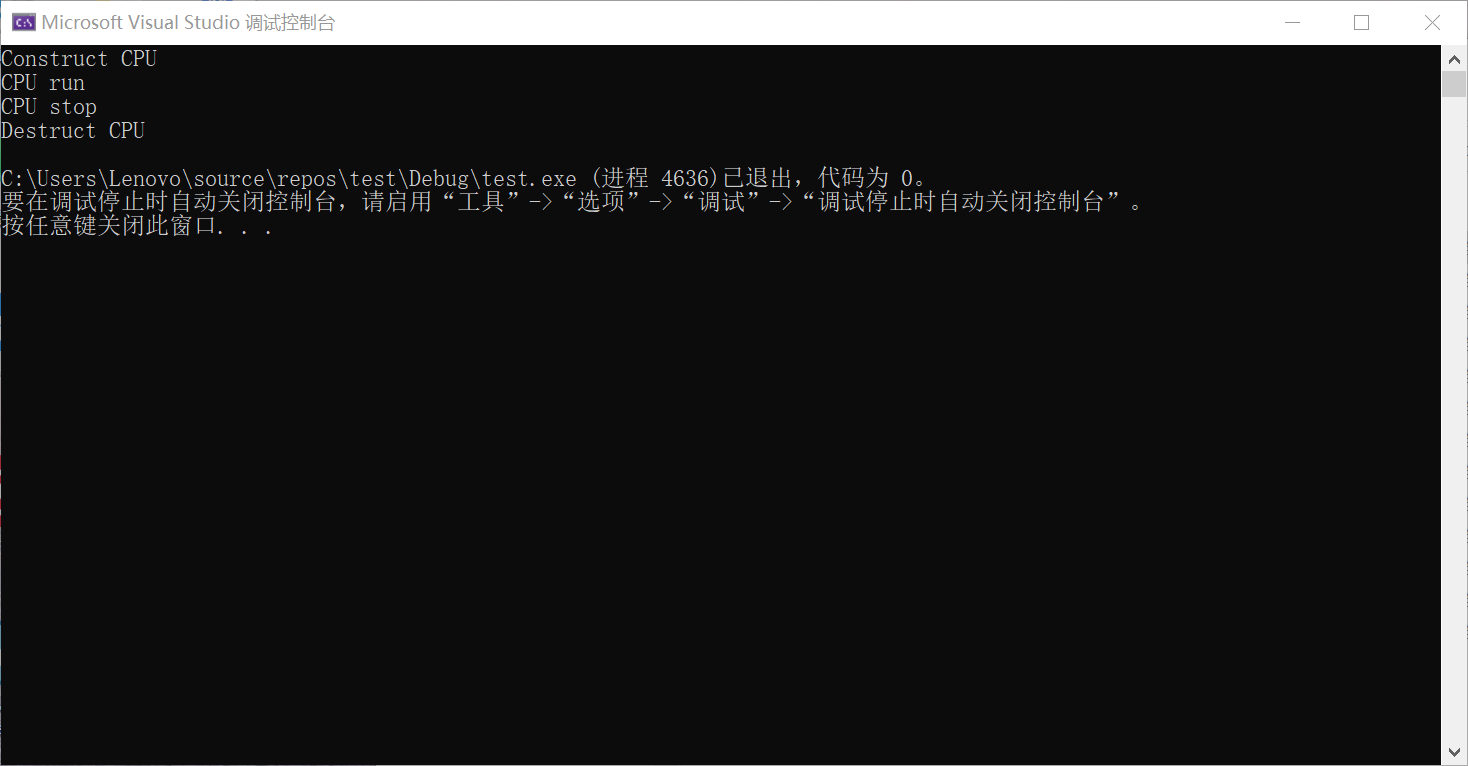
**1.3（选做）**

1. #include<iostream>
2. #include<math.h>
3. #include<string>
4. #include<iomanip>
5. **using** **namespace** std;
6. **class** people
7. {
8. **public**:
9. people()
10. {
11. cout<<"默认构造函数"<<endl;
12. }
13. people(string a,string b,string c,string d):number(a),sex(b),id(c),birthday(d)
14. {
15. cout<<"构造了此函数"<<endl;
16. }
17. people(people&a):number(a.number),sex(a.sex),id(a.id),birthday(a.birthday)
18. {
19. cout<<"拷贝此类型的对象"<<endl;
20. }
21. ~people()
22. {
23. cout<<"析构此类型对象"<<endl;
24. }
25. **void** show()
26. {
27. cout<<"编号是："<<number<<endl<<"性别是："<<sex<<endl;
28. cout<<"身份证号是："<<id<<endl<<"出生日期是："<<birthday<<endl;
29. }
30. **private**:
31. string number;
32. string sex;
33. string id;
34. string birthday;
35. };
36. **int** main ()
37. {
38. string a,b,c,d;
39. cin>>a>>b>>c>>d;
40. people people1;
41. people people2(a,b,c,d);
42. people2.show();
43. people people3(people2);
44. people3.show();
46. }

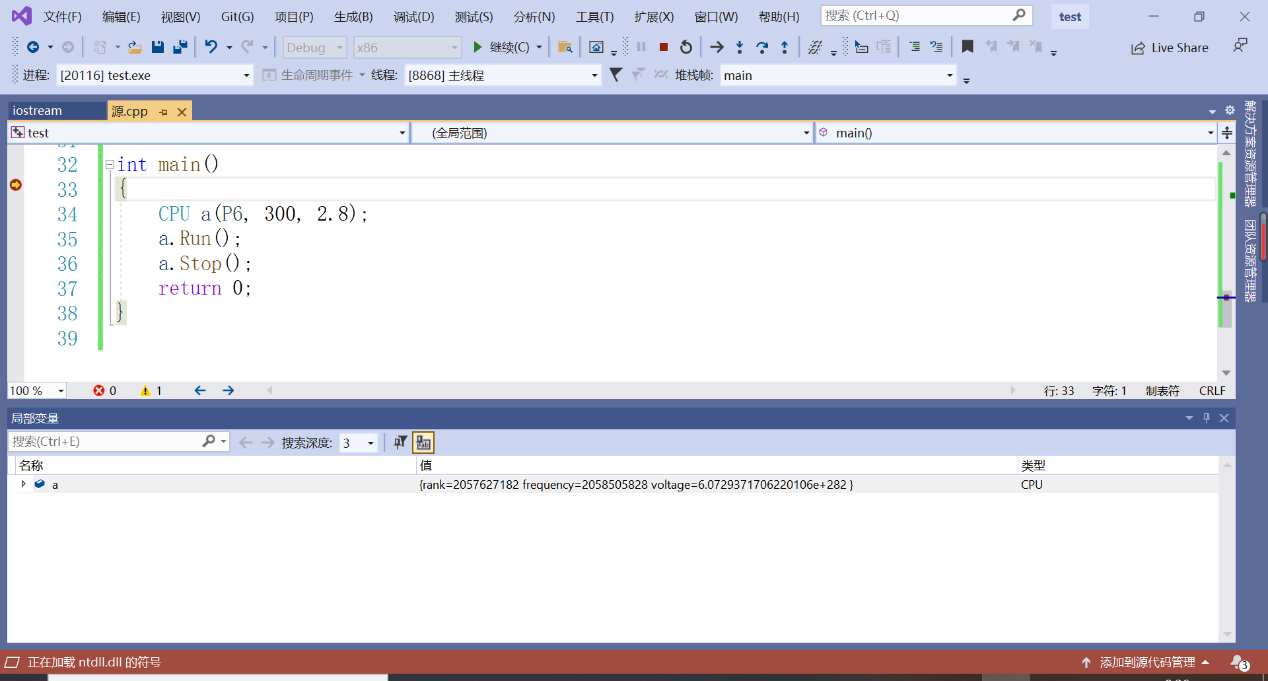
**5.运行结果（贴图）**

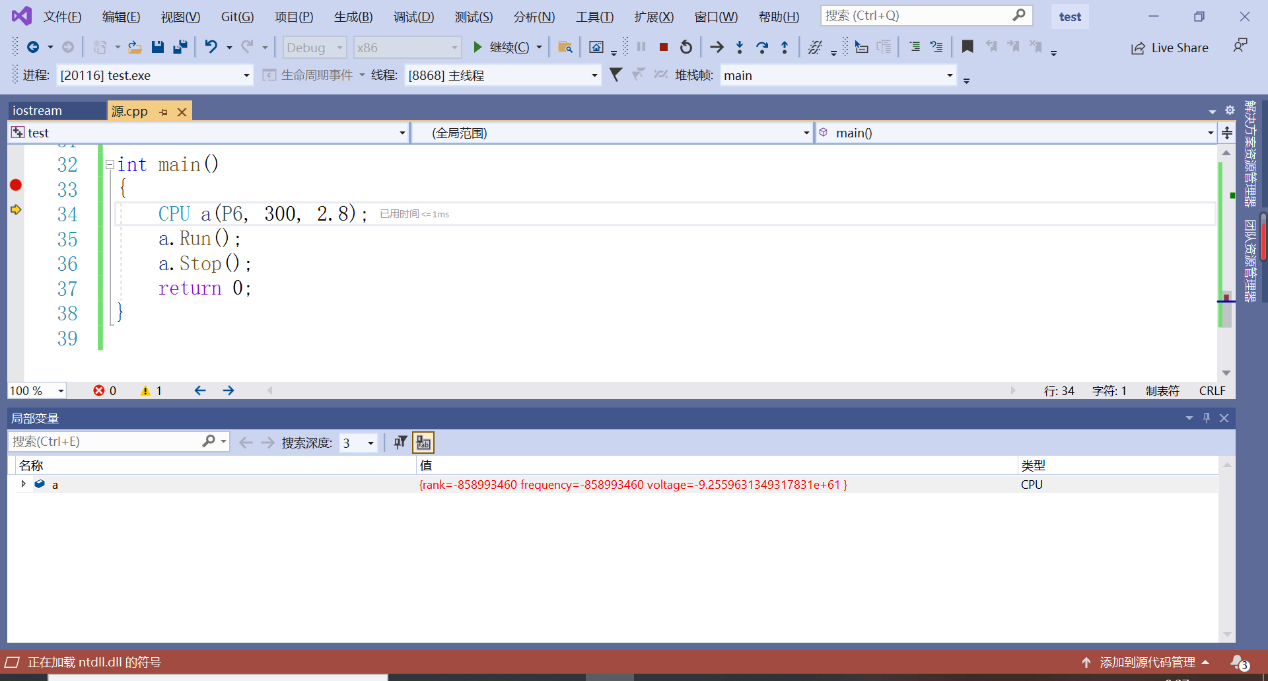
**第一次实验截屏要求：除题目运行结果截屏外，还需要有debug截屏。第一题需要有debug部分的截屏，第一题按照调试操作步骤截屏。**

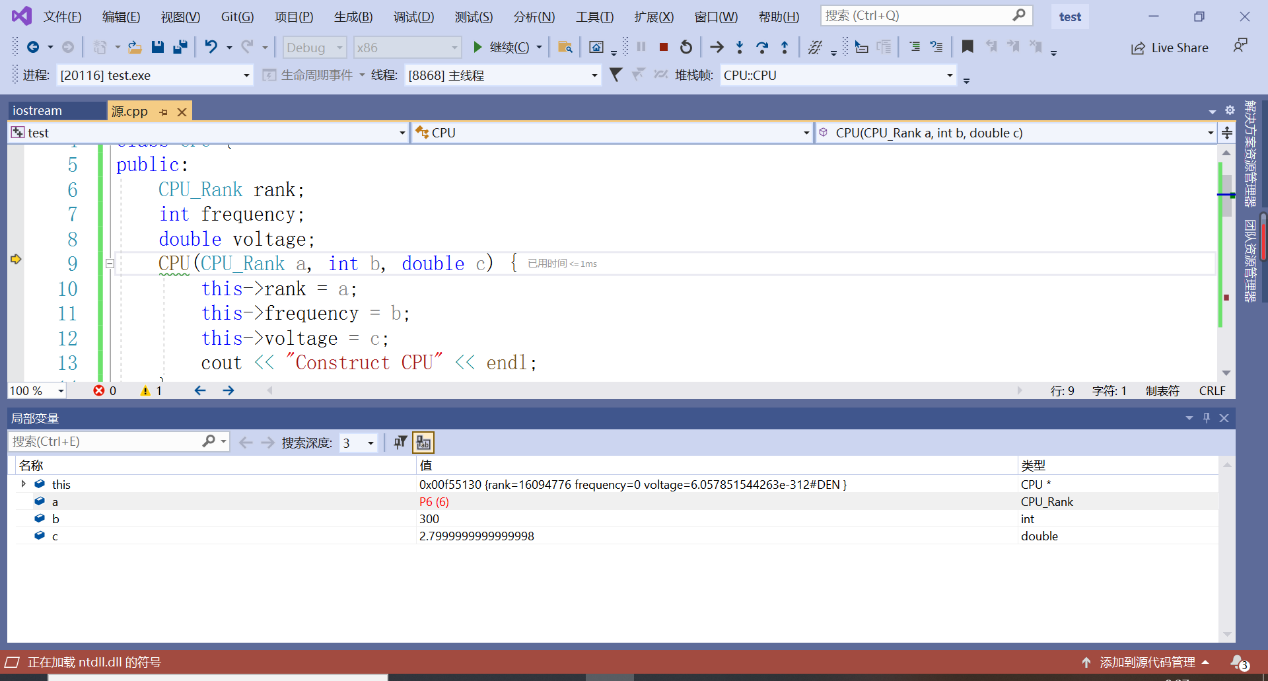
* 1. **结果**

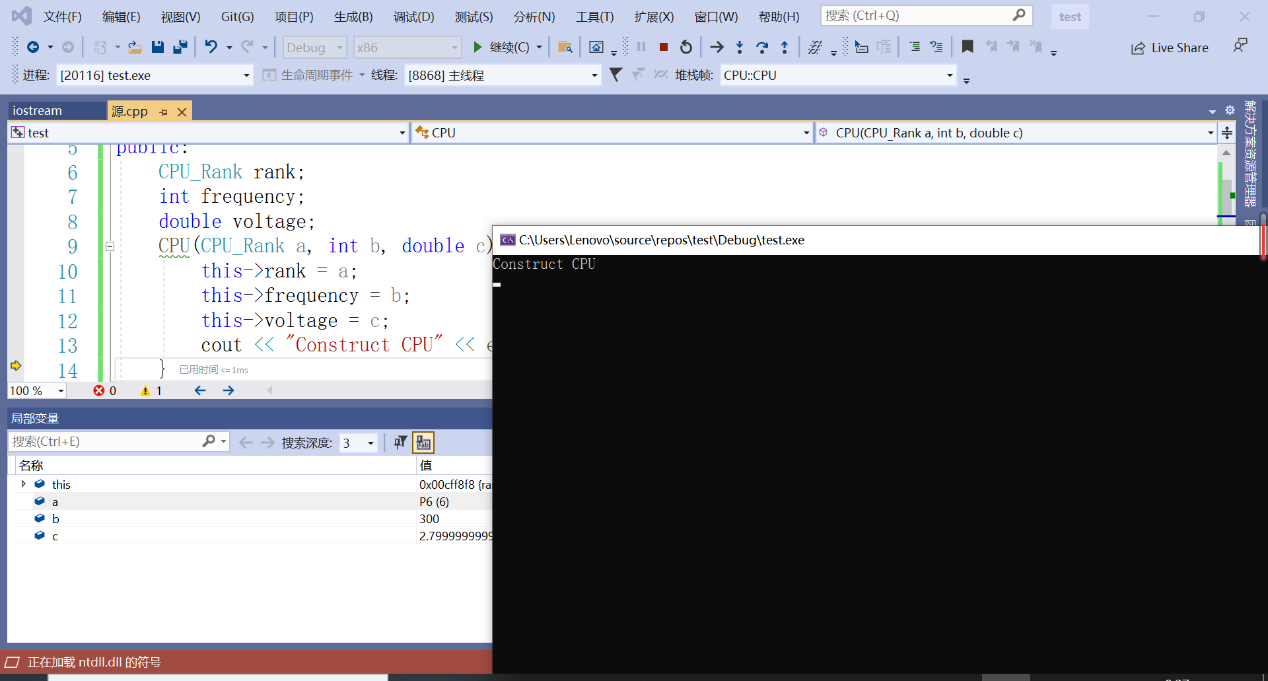


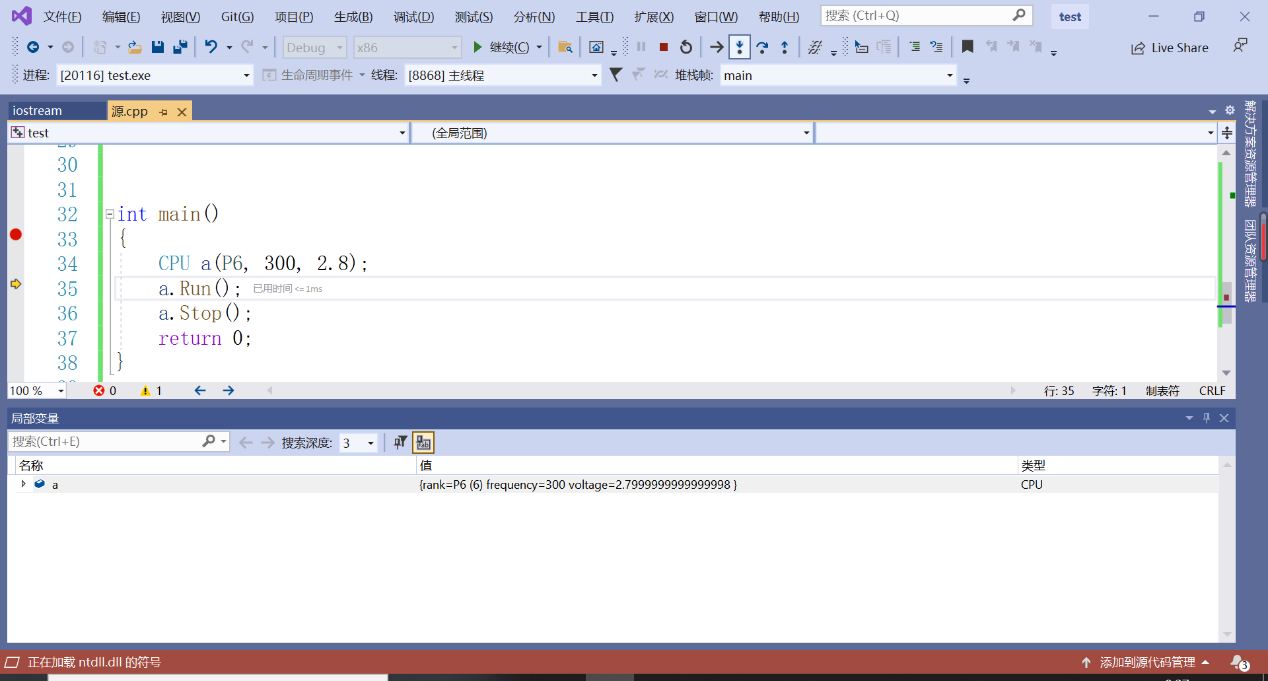
**1.1 debug**

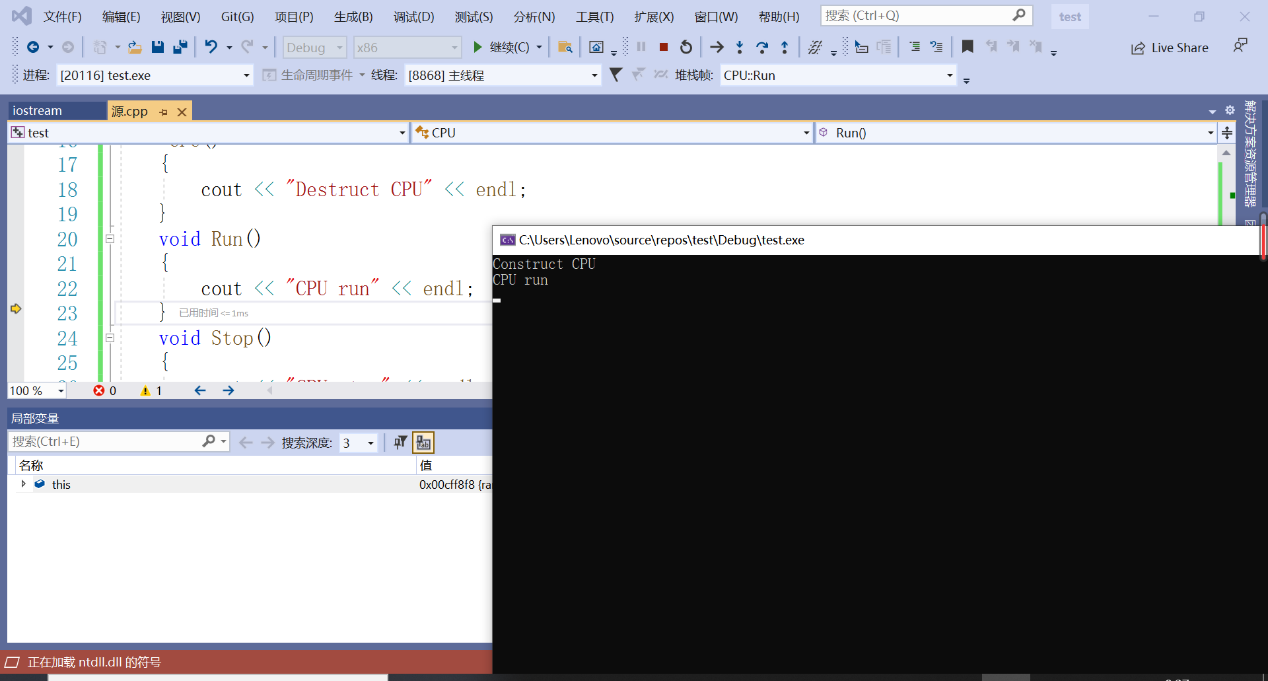


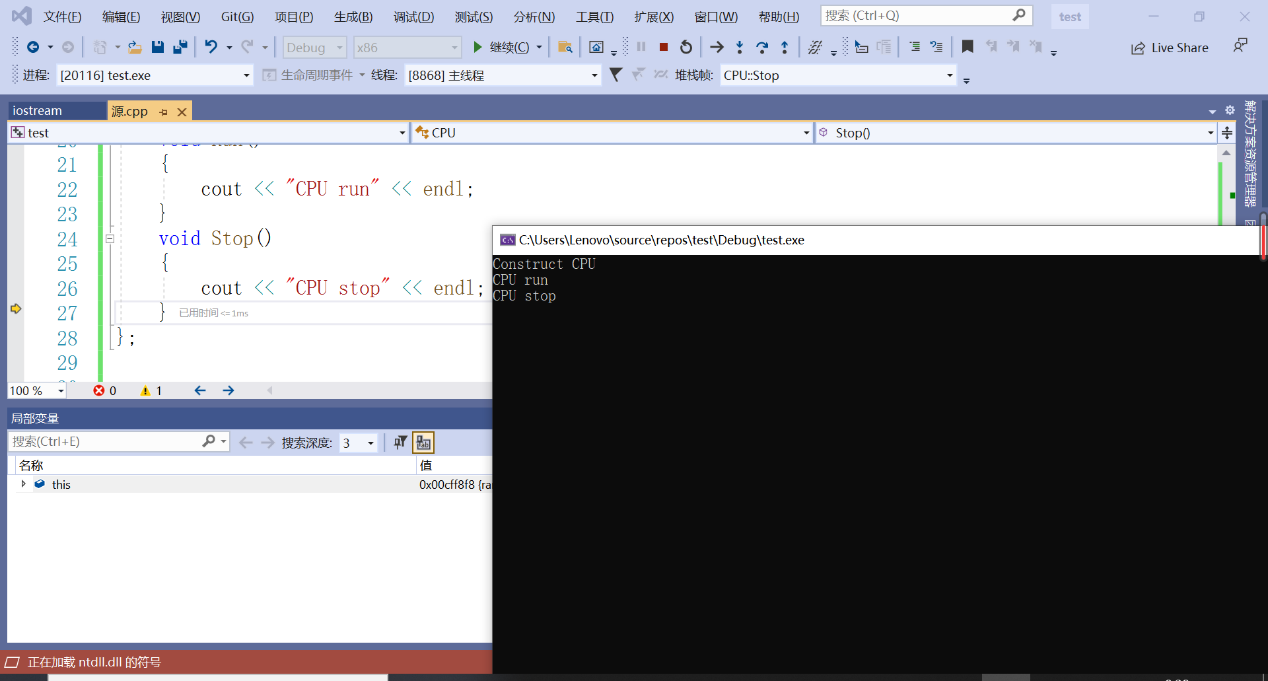
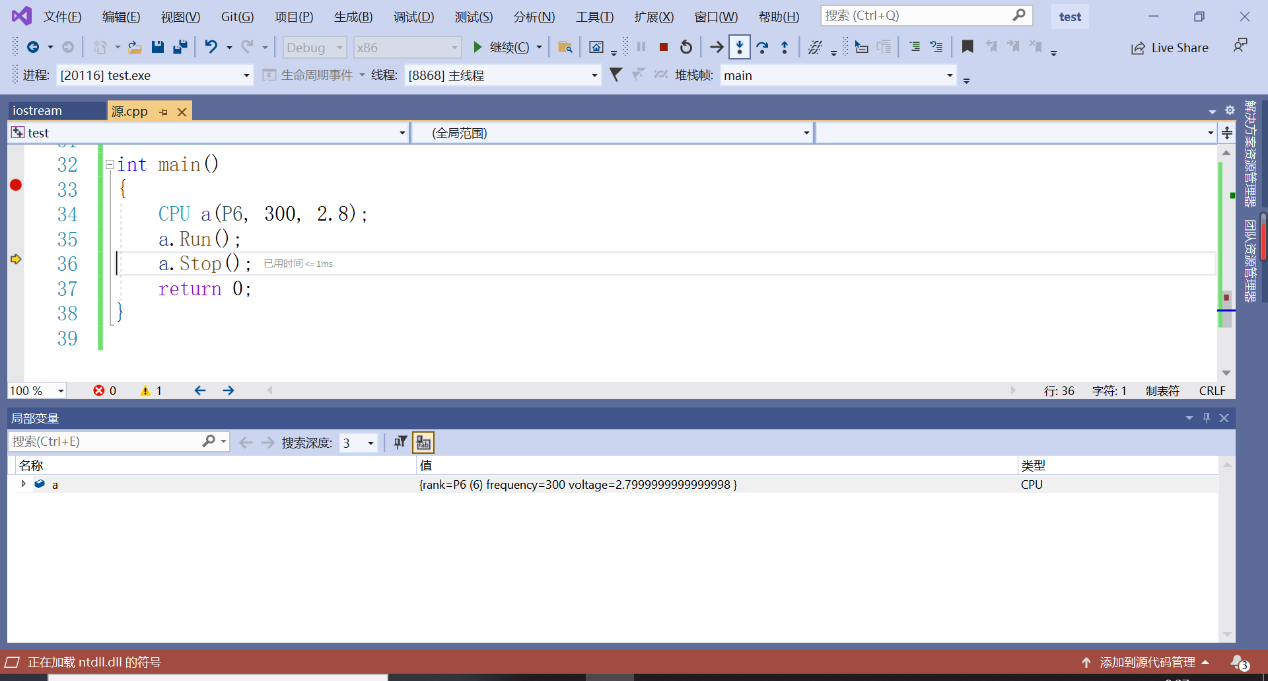


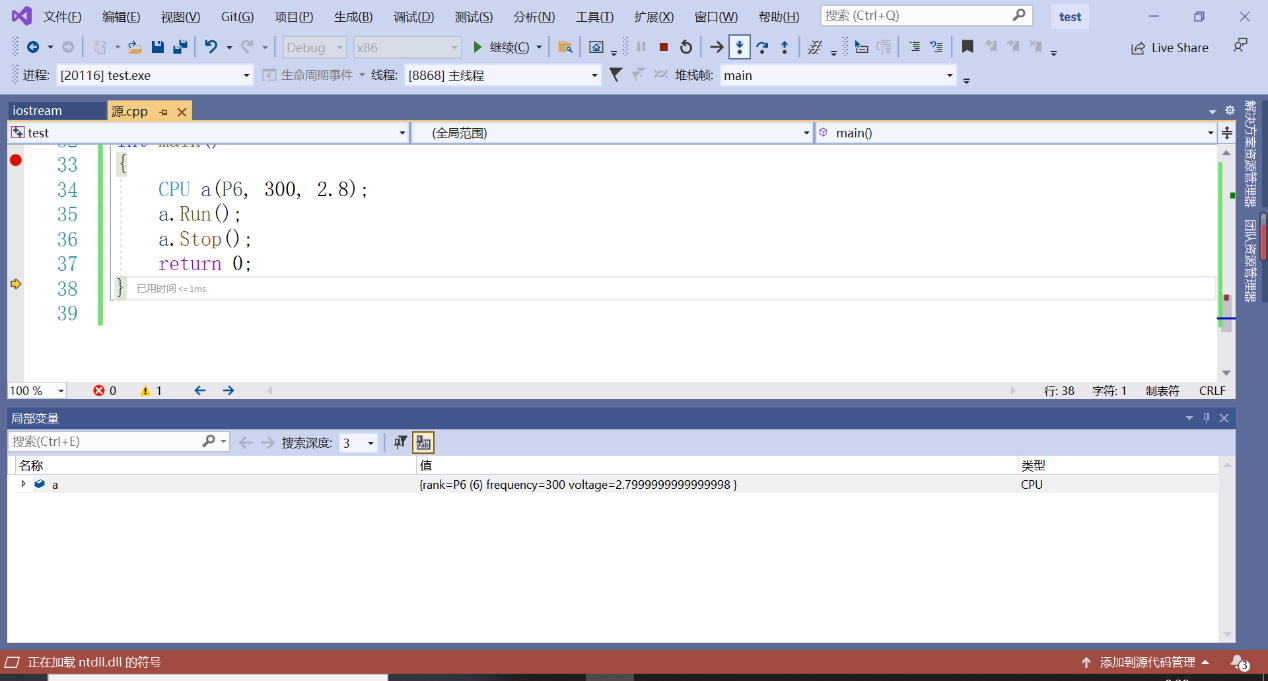
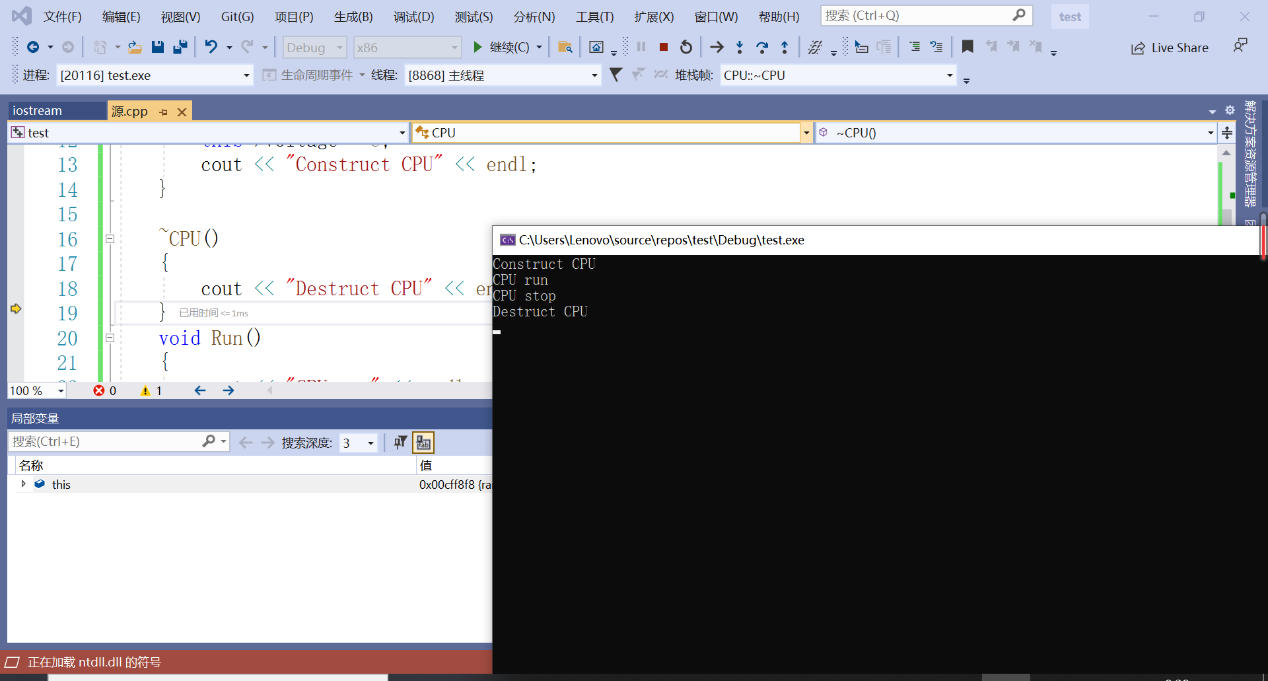
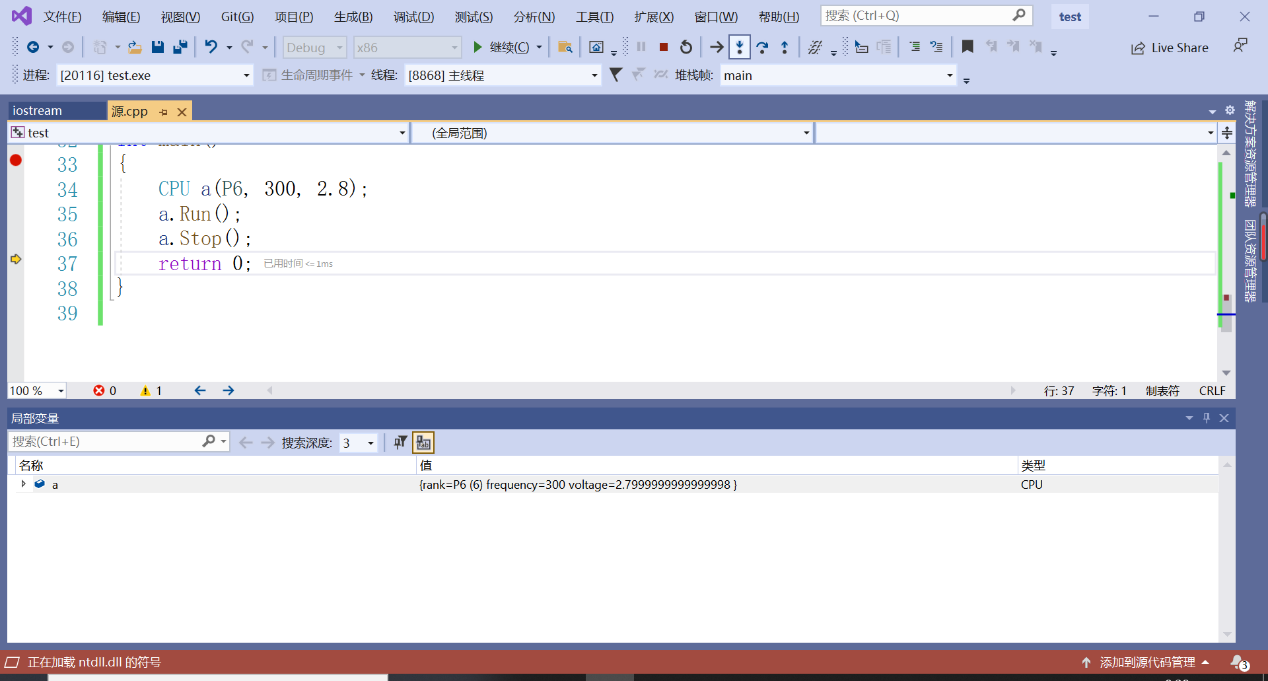








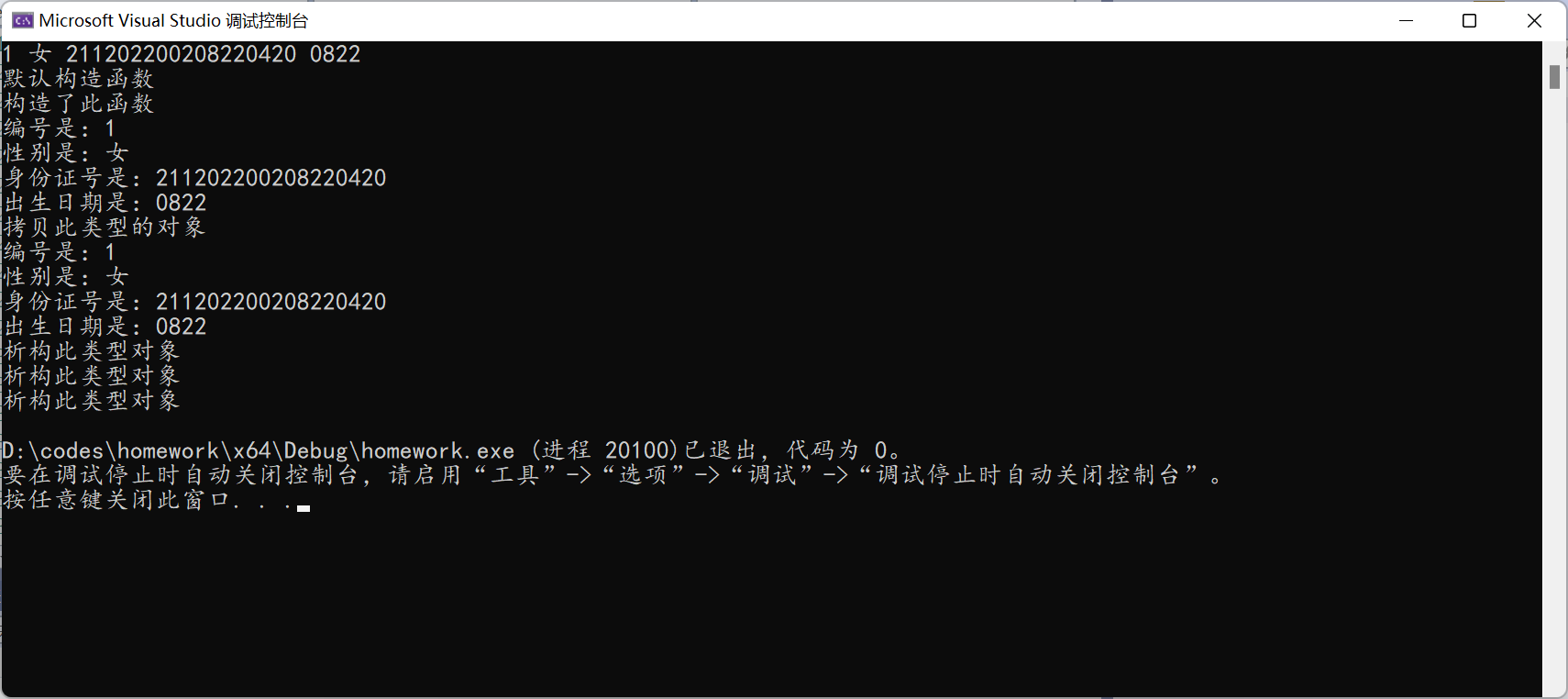




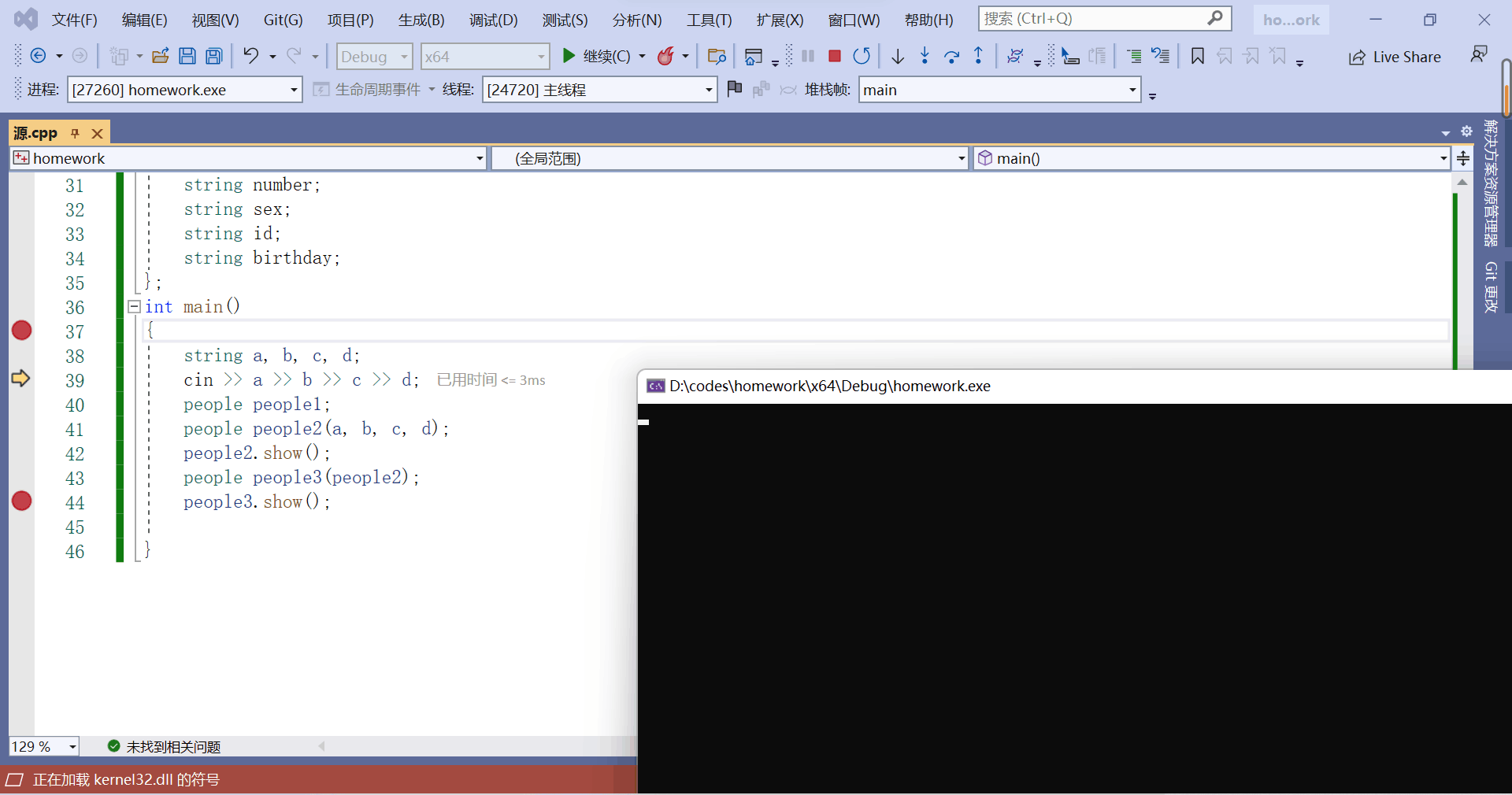
* 1. **结果**

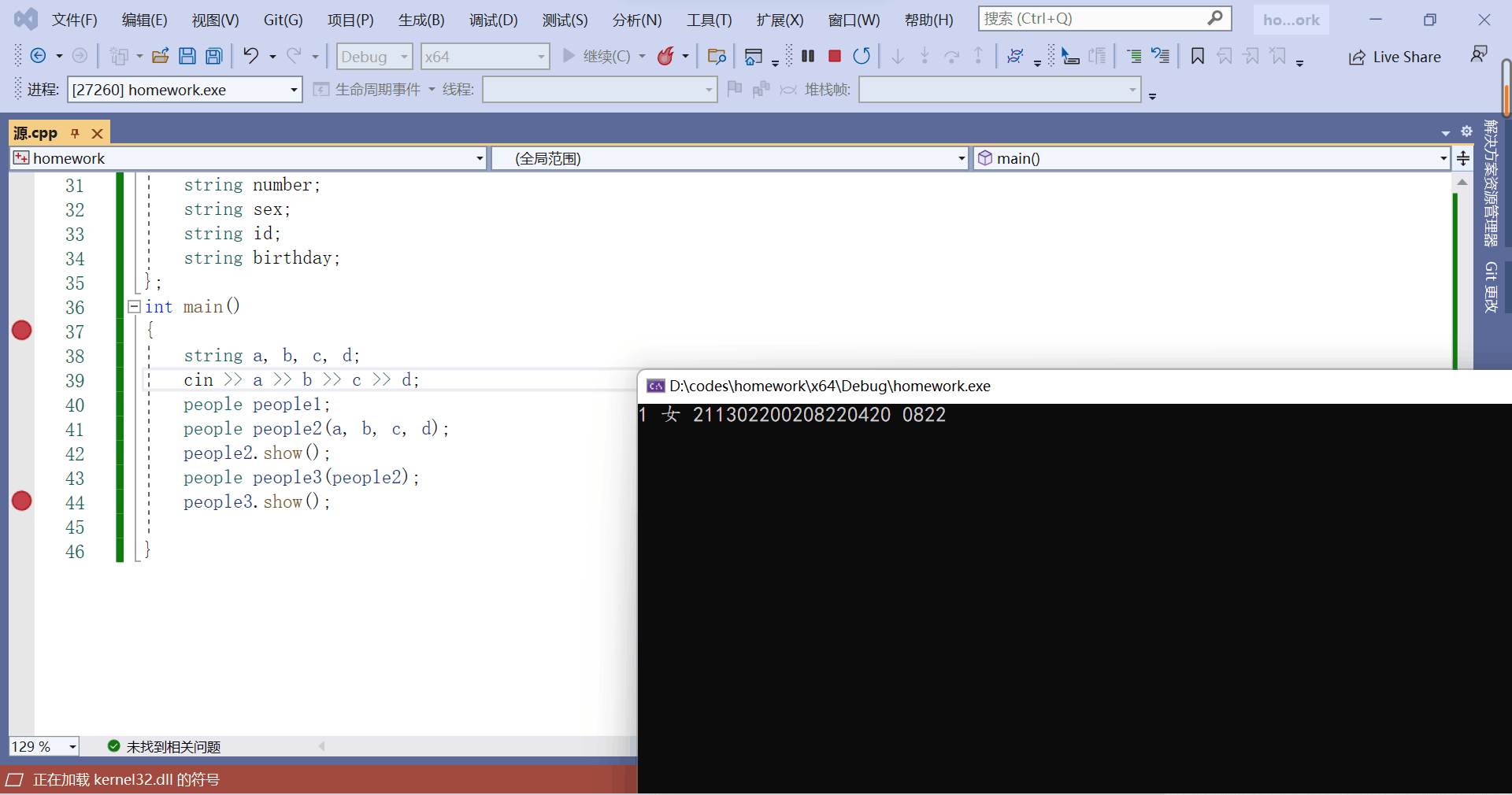


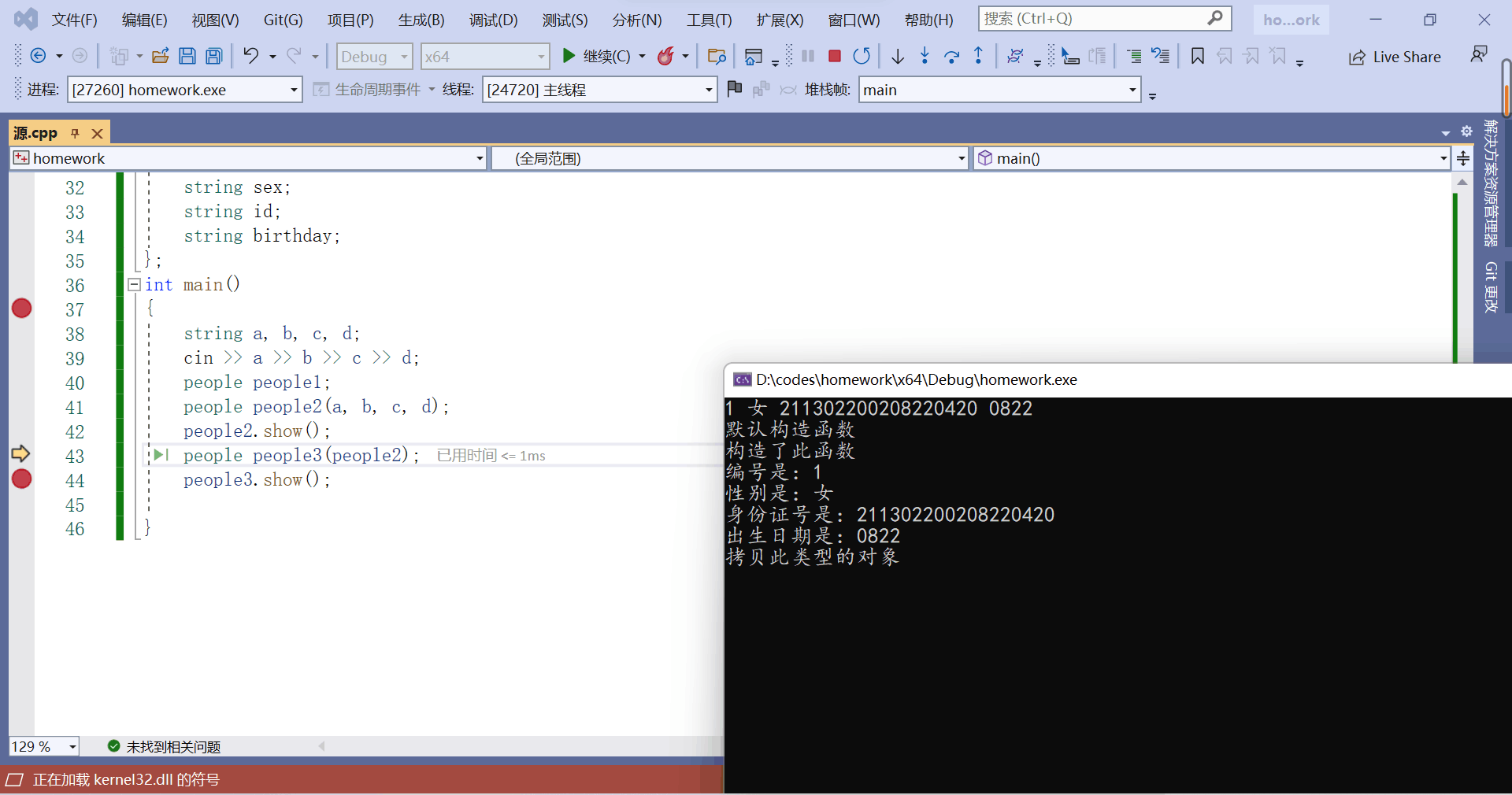
**1.3结果**

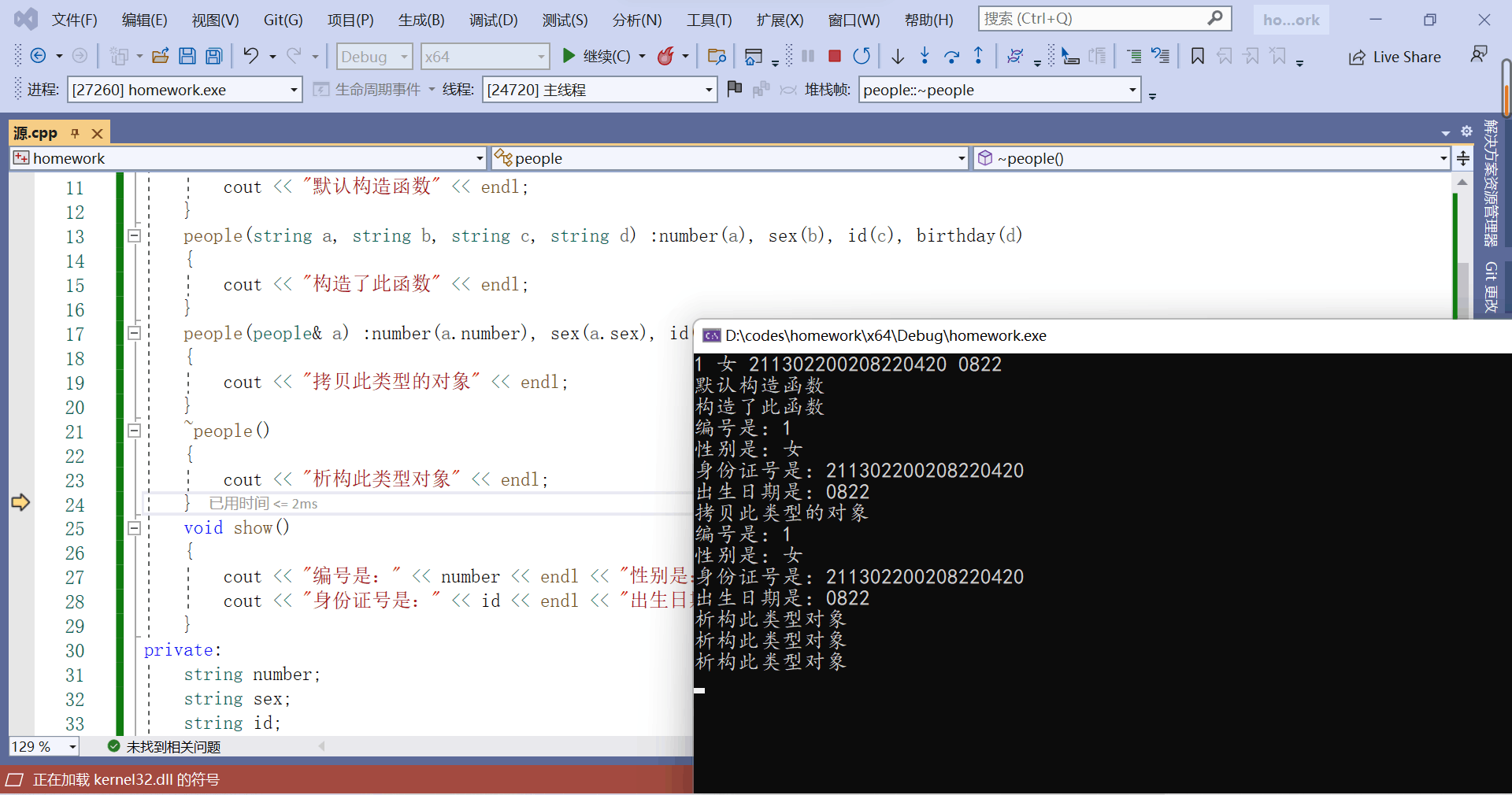


**1.3debug**









1. **心得体会：学到了什么；遇到的问题及解决方法等（可结合思考题）**

**思考题：**

1. 如何定义一个类？

Class 后面加上自己要构建类的名字

然后加上一个大括号里面用public和private来声明自己的函数类型和数据成员类型。

1. 如何定义类的构造函数和析构函数？

构造函数：类名（）{}括号里面可以有传入的参数，然后用这些参数来给自己的私有数据成员赋值。如果没有参数，就是默认构造函数，如果有私有数据成员会系统随机为成员赋值。

1. 类的各成员函数的执行顺序是怎样的？

先是构造函数，之后调用类的对象的函数，当类的对象使用结束之后，在对其调用析构函数。

1. 类组合时对象的构造顺序是怎样的？

如果对象是类的数据成员，先对这些数据成员的对象进行构造，然后再按照声明的顺序对此类的其他一些私有数据进行赋值。

**心得体会：**

1. 通过本次实验课，我掌握类的定义和使用以及类的定义和对象的声明。

2.我掌握了不同访问属性（public,private,protected）的成员的访问方式。3.通过debug，我追踪观察构造函数和析构函数的执行过程。4.我掌握了关于成员对象以及封闭类的使用方法，以及使用时需要注意的地方。

5.一开始在解第一题遇到了不少的困难，后面经过多般调试，终于了解了成员对象和类的构造函数和析构函数调用顺序，最后顺利解出此题。

**实验二**

**1.实验目的**1.学习使用数组2.学习字符串数据的组织和处理3.学习标准C++库的使用4.掌握指针的使用方法5.练习通过Debug观察指针的内容及其所指的对象的内容6.练习通过动态内存分配实现动态数组，并体会指针在其中的作用7.分别使用字符数组和标准C++库练习处理字符串的方法

1. **实验要求**

1.编写并测试3\*3矩阵转置函数，使用数组保存3\*3矩阵。2.使用动态内存分配生成动态数组来重新完成上题，使用指针实现函数的功能。3.编程实现两字符串的连接。要求使用字符数组保存字符串，不要使用系统函数。4.使用string类定义字符串对象，重新实现上一小题。5.定义一个Employee类，其中包括属性： 姓名（name）、街道地址（address）、城市（city）、邮编(postalcode)，以及display()、change\_name()等函数。其中，display()显示姓名、街道地址、城市和邮编属性，change\_name()改变对象的姓名属性。 要求：测试时，首先通过接收键盘输入录入Employee对象的各个属性，再调用display()函数显示Employee对象的各个属性的初始值，然后调用change\_name()函数更改对象的姓名，最后调用display()函数显示新的Employee对象的各个属性的值。6.使用上一小题中定义的Employee类（提示需要在上一题基础上做改动）定义对象数组emp[5]，使用循环语句把数据显示出来。 要求：测试时，首先通过接收键盘输入录入的5个对象数组emp[5]的各个属性（包括姓名、街道地址、城市、邮编），然后使用循环语句调用display()函数把5个对象数组emp[5]的各个属性显示出来。

7. （选做）修改实验一的选做实验中的people（人员）类。具有的属性如下： 姓名 char name[11] 编号 char number[7] 性别 char sex[3] 出生日期 birthday，“出生日期”定义为一个“日期（date）”类内嵌对象 身份证号 char id[16] 用成员函数实现对人员信息的录入和显示。 要求包括：构造函数和析构函数、拷贝构造函数、内联成员函数、聚集。在测试程序中定义people类的对象数组，录入数据并显示。

**3.实验内容（各题目的题干）**1.编写矩阵转置函数，输入参数为3\*3整形数组，使用循环语句实现矩阵元素的行列对调，注意在循环语句中究竟需要对哪些元素进行操作，编写main（）函数实现输入、输出。

2.改写矩阵转置函数，参数为整型指针，使用指针对数组元素进行操作，在main（）函数中使用new操作符分配内存生成动态数组。通过Debug观察指针的内容及其所指的对象中的内容。

3.编程实现两字符串的连接。定义字符数组保存字符串，在程序中提示用户输入两个字符串，实现两个字符串的连接，最后用cout语句显示输出。用cin实现输入，注意，字符串的结束标志是ASCII码0，使用循环语句进行字符串间的字符拷贝。

4.使用string类定义字符串对象，编程实现两字符串的连接。在string类中已重载了运算符“+=”实现字符串的连接，可以使用这个功能。

5.定义Employee类，Employee类具有姓名、街道地址、城市和邮编等私有数据成员，display()中使用cout显示姓名、街道地址、城市和邮编等属性，change\_name()改变类中表示姓名属性的数据成员。在主程序中定义这个类的对象并对起进行操作。

6.在上一题定义的Employee类基础上使用Employee类定义对象数组emp[5]，并在录入对象数组的信息后使用循环语句把数据显示出来。

7. （选做）修改实验一的选做实验中的people（人员）类。具有的属性如下：

姓名 char name[11]

编号 char number[7]

性别 char sex[3]

出生日期 birthday，“出生日期”定义为一个“日期（date）”类内嵌对象

身份证号 char id[16]

用成员函数实现对人员信息的录入和显示。

要求包括：构造函数和析构函数、拷贝构造函数、内联成员函数、聚集。在测试程序中定义people类的对象数组，录入数据并显示。

**4.源程序（加注释）**

**2.1**

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main()
5. {
6. **int** a[3][3];
7. **for** (**int** i = 0; i < 3;i++){
8. **for** (**int** j = 0; j < 3;j++){
9. cin >> a[i][j];
10. }
11. }
12. **for** (**int** j = 0; j < 3;j++){
13. **for** (**int** i = 0; i < 3;i++){
14. cout << a[i][j] << " ";
15. }
16. cout << endl;
17. }
19. **return** 0;
20. }

**2.2**

1. #include<iostream>
2. #include<string>
3. #include<cmath>
4. **using** **namespace** std;
5. **void** change(**int**(\*a)[3])
6. {
7. **for** (**int** i = 0; i < 3; i++)
8. {
9. **for** (**int** j = 0; j < 3; j++)
10. {
11. cout << \*(\*(a + j) + i) << " ";
12. }
13. cout << endl;
14. }
15. }
16. **int**  main()
17. {
18. **int**(\*a)[3] = **new** **int**[3][3];
19. **for** (**int** i = 0; i < 3;i++)//行
20. {
21. **for** (**int** j = 0; j < 3;j++)//列
22. {
23. cin >> \*(\*(a + i) + j);
24. }
25. }
26. change(a);
27. **delete**[]a;
28. }

**2.3**

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main()
5. {
6. **char** ans[100000];
7. **int** cnt = 0;
8. **char** ch;
9. ch = getchar();
10. **while**(ch != ' '){
11. ans[cnt++] = ch;
12. ch = getchar();
13. }
14. ch = getchar();
15. **while**(ch != '\n' && ch != EOF){
16. ans[cnt++] = ch;
17. ch = getchar();
18. }
19. **for** (**int** i = 0; i < cnt;i++){
20. cout << ans[i];
21. }
22. **return** 0;
23. }

**2.4**

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main()
5. {
6. string a, b;
7. cin >> a >> b;
8. cout << a << b;
9. **return** 0;
10. }

**2.5**

1. #include <iostream>
2. #include <string>
3. **using** **namespace** std;
4. **class** Employee
5. {
6. **public**:
7. string name;
8. string address;
9. string city;
10. string postalcode;
11. Employee(string a, string b, string c, string d){
12. **this**->name = a;
13. **this**->address = b;
14. **this**->city = c;
15. **this**->postalcode = d;
16. }
17. **void** display()
18. {
19. cout << "name:" << **this**->name << endl;
20. cout << "address:" << **this**->address << endl;
21. cout << "city:" << **this**->city << endl;
22. cout << "postalcode:" << **this**->postalcode << endl;
23. }
24. **void** change\_name(string n)
25. {
26. **this**->name = n;
27. }
28. };
30. **int** main()
31. {
32. string n,a,c,p;
33. cin>>n;
34. cin>>a;
35. cin>>c;
36. cin>>p;
37. Employee emp(n,a,c,p);
38. emp.display();
39. cin>>n;
40. emp.change\_name(n);
41. emp.display();
42. }

**2.6**

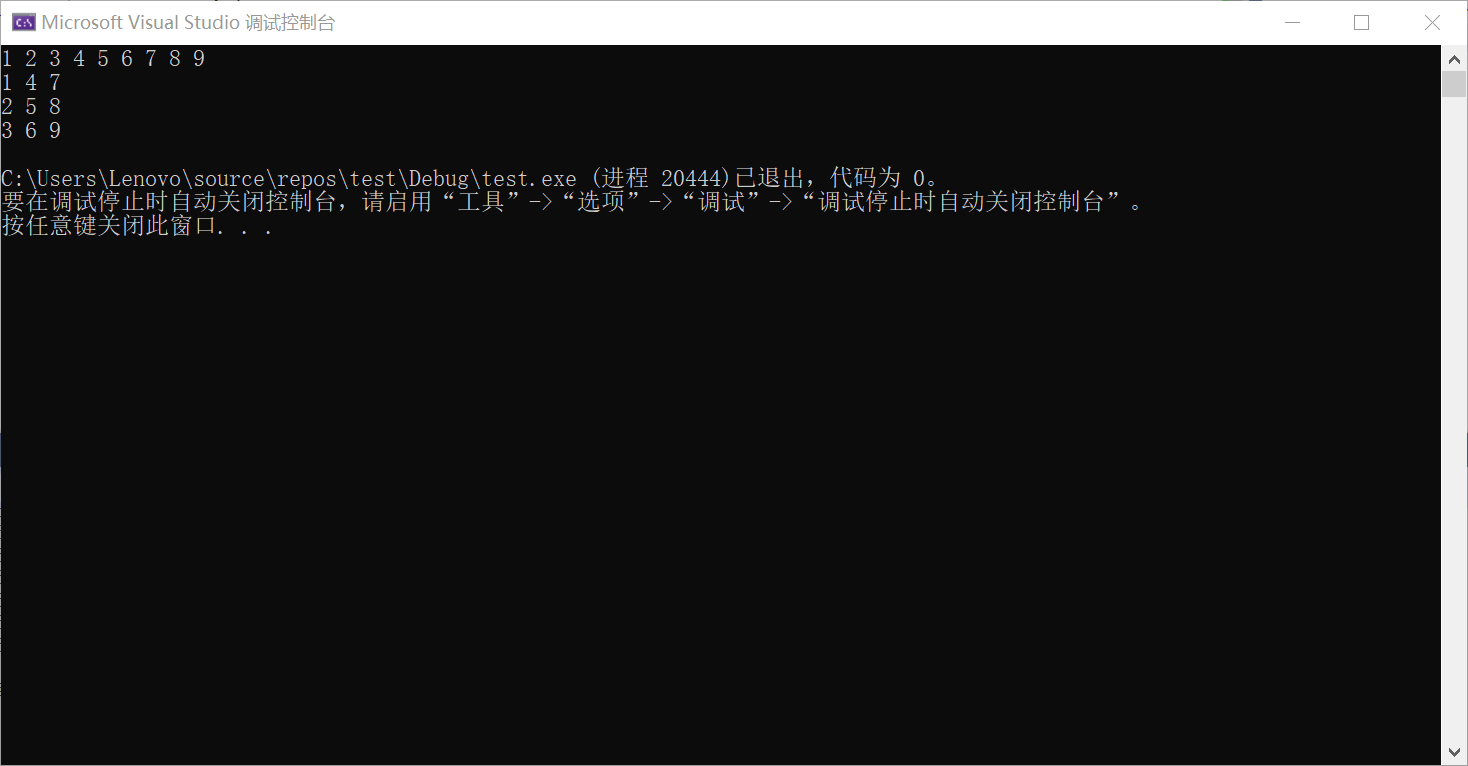
1. #include <iostream>
2. #include <string>
3. **using** **namespace** std;
4. **class** Employee
5. {
6. **public**:
7. string name;
8. string address;
9. string city;
10. string postalcode;
12. Employee(string a, string b, string c, string d){
13. **this**->name = a;
14. **this**->address = b;
15. **this**->city = c;
16. **this**->postalcode = d;
17. }
18. Employee()
19. {
21. }
22. **void** display()
23. {
24. cout << "name:" << **this**->name << endl;
25. cout << "address:" << **this**->address << endl;
26. cout << "city:" << **this**->city << endl;
27. cout << "postalcode:" << **this**->postalcode << endl;
28. }
29. **void** change\_name(string n)
30. {
31. **this**->name = n;
32. }
33. };
35. **int** main()
36. {
37. string n,a,c,p;
38. **int** i,j;
39. Employee emp[5];
40. **for**(i=0;i<5;i++)
41. {
42. cin>>n;
43. cin>>a;
44. cin>>c;
45. cin>>p;
46. emp[i]=Employee(n,a,c,p);
47. }
48. **for**(i=0;i<5;i++)
49. {
50. cout<<"Employee"<<i+1<<endl;
51. emp[i].display();
52. }
53. }

**2.7选做：**

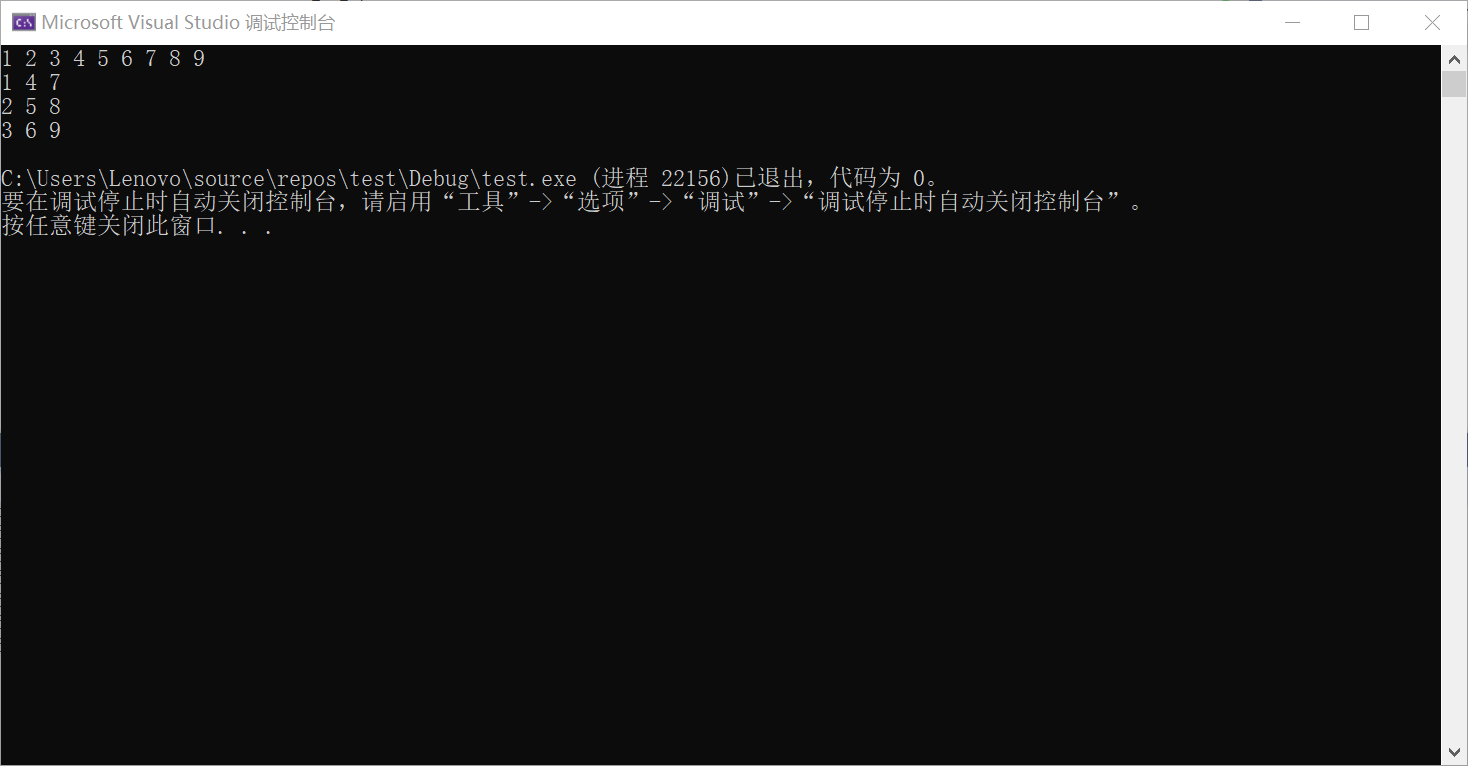
1. #include<iostream>
2. #include<math.h>
3. #include<string>
4. #include<iomanip>
5. #include<cstring>
6. **using** **namespace** std;
7. **class** people
8. {
9. **public**:
10. people()
11. {
12. cout<<"默认构造函数"<<endl;
13. }
14. people(**char** n[],**char** a[],**char** b[],string d,**char** c[])
15. {
16. strcpy(name,n);
17. strcpy(number,a);
18. strcpy(sex,b);
19. strcpy(id,c);
20. birthday=d;
21. cout<<"构造了此函数"<<endl;
22. }
23. people(people&a)
24. {
25. strcpy(name,a.name);
26. strcpy(number,a.number);
27. strcpy(sex,a.sex);
28. strcpy(id,a.id);
29. birthday=a.birthday;
30. cout<<"拷贝此类型的对象"<<endl;
31. }
32. ~people()
33. {
34. cout<<"析构此类型对象"<<endl;
35. }
36. **void** show()
37. {
38. cout<<"姓名是："<<name;
39. cout<<"编号是："<<number<<endl<<"性别是："<<sex<<endl;
40. cout<<"身份证号是："<<id<<endl<<"出生日期是："<<birthday<<endl;
41. }
42. **private**:
43. **char** name[11];
44. **char** number[7];
45. **char** sex[3];
46. string birthday;
47. **char** id[16];
48. };
49. **int** main ()
50. {
51. **char** n[11];
52. **char** a[7];
53. **char** b[3];
54. string d;
55. **char** c[16];
56. cin>>n>>a>>b>>c>>d;
57. people people1;
58. people people2(n,a,b,d,c);
59. people2.show();
60. people people3(people2);
61. people3.show();
63. }

**5.运行结果（贴图） 第二次实验截屏要求：除题目运行结果截屏外，还需要有debug截屏。第二题需要有debug部分的截屏，第二题按照调试操作步骤截屏。**

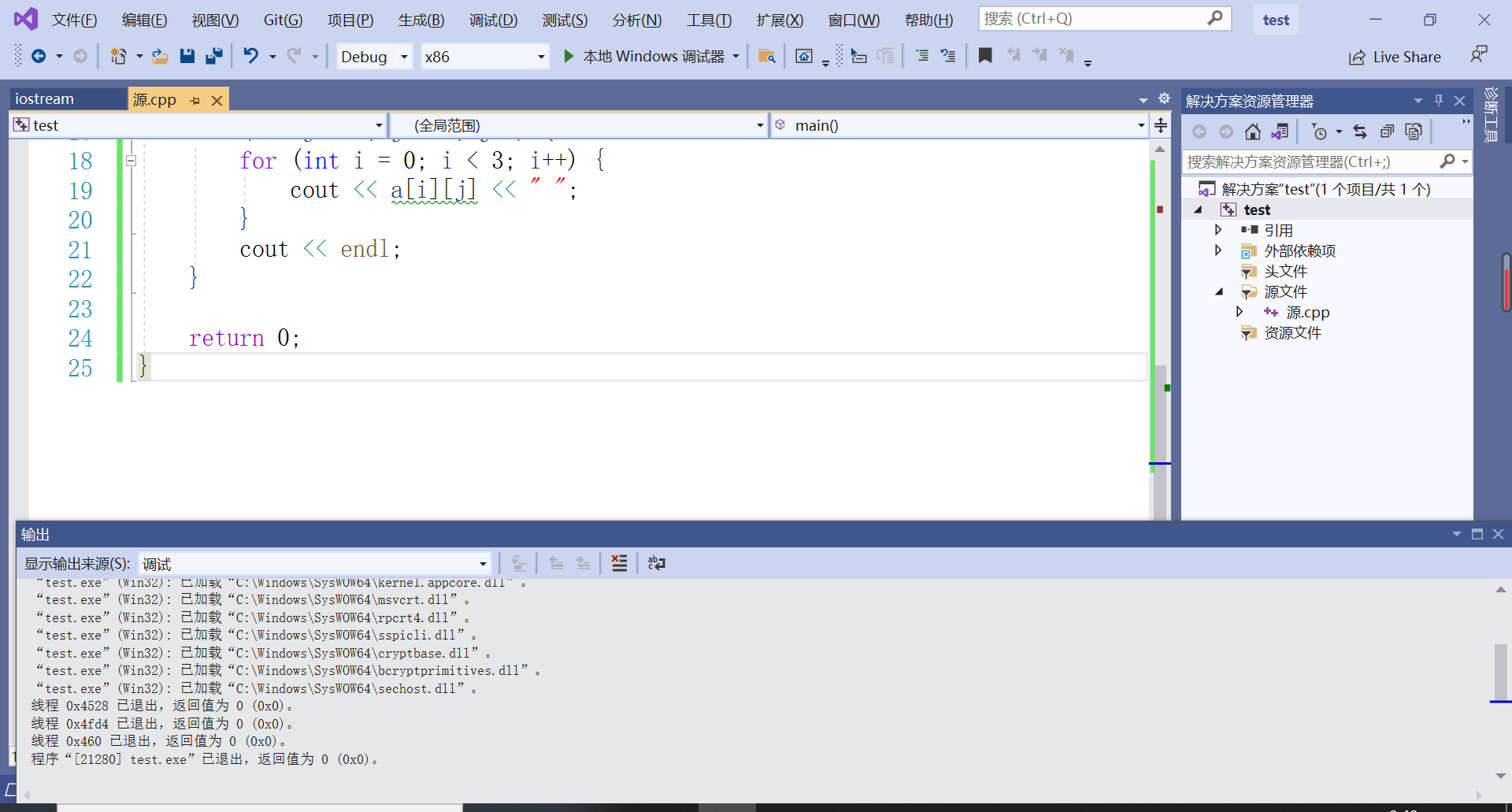
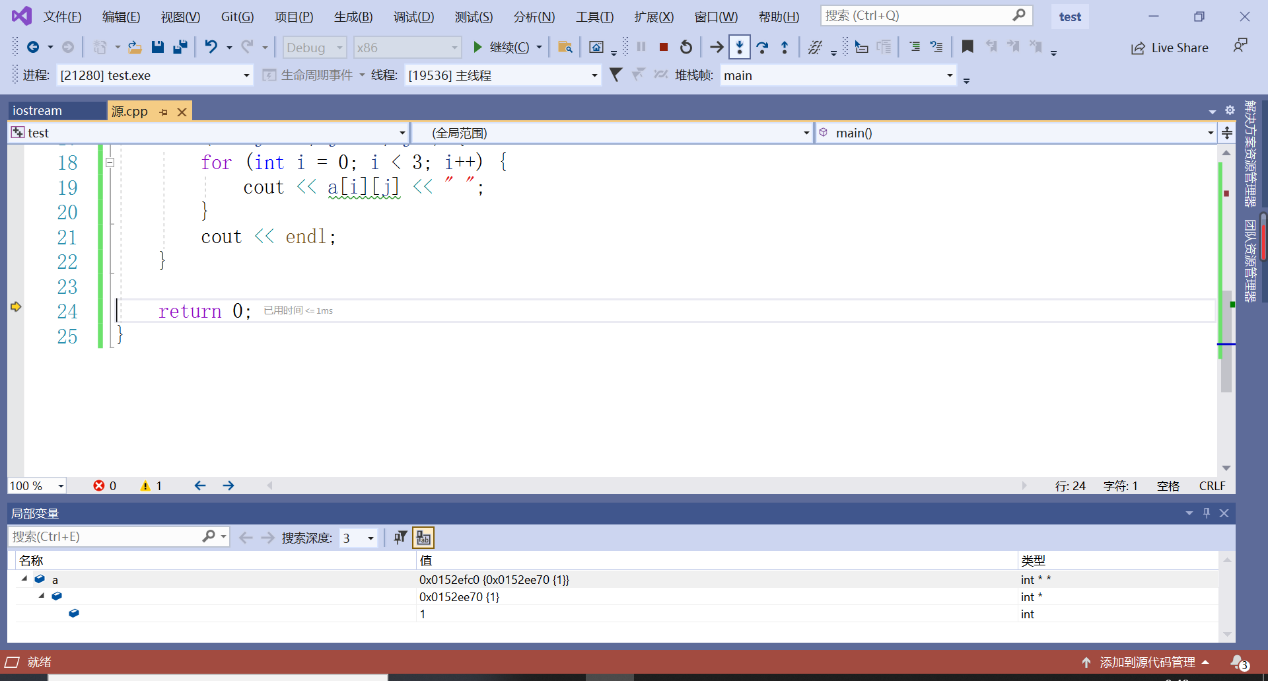
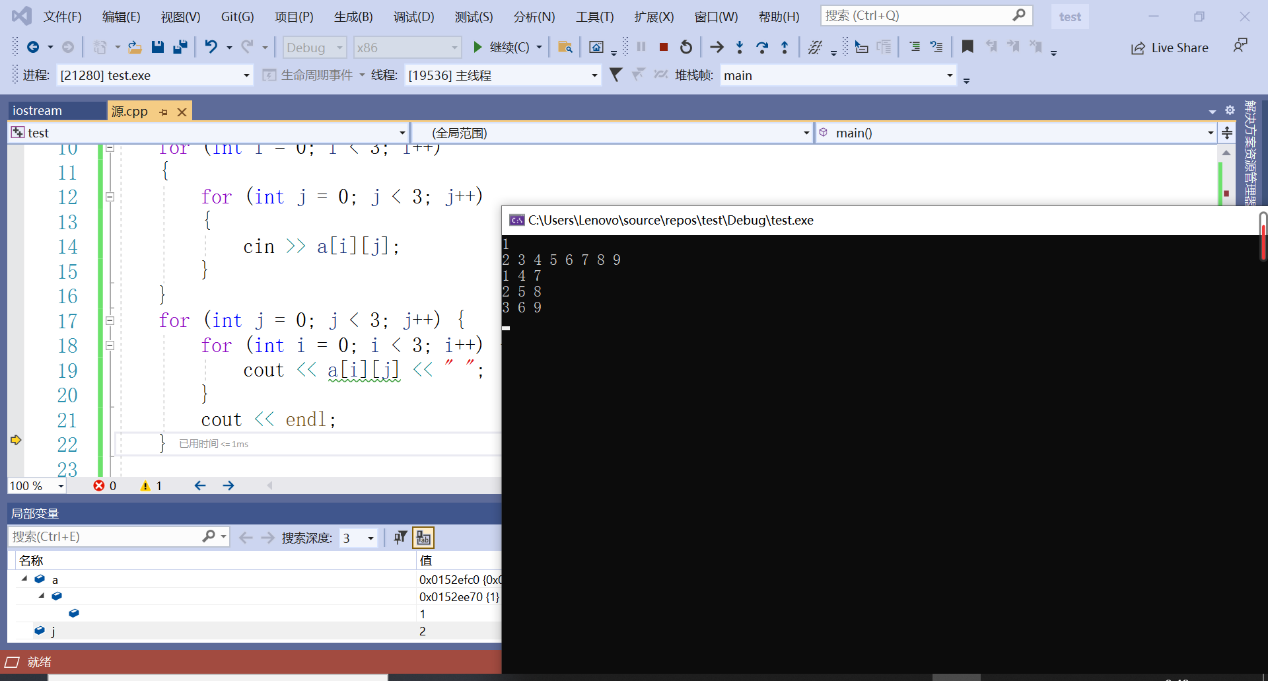
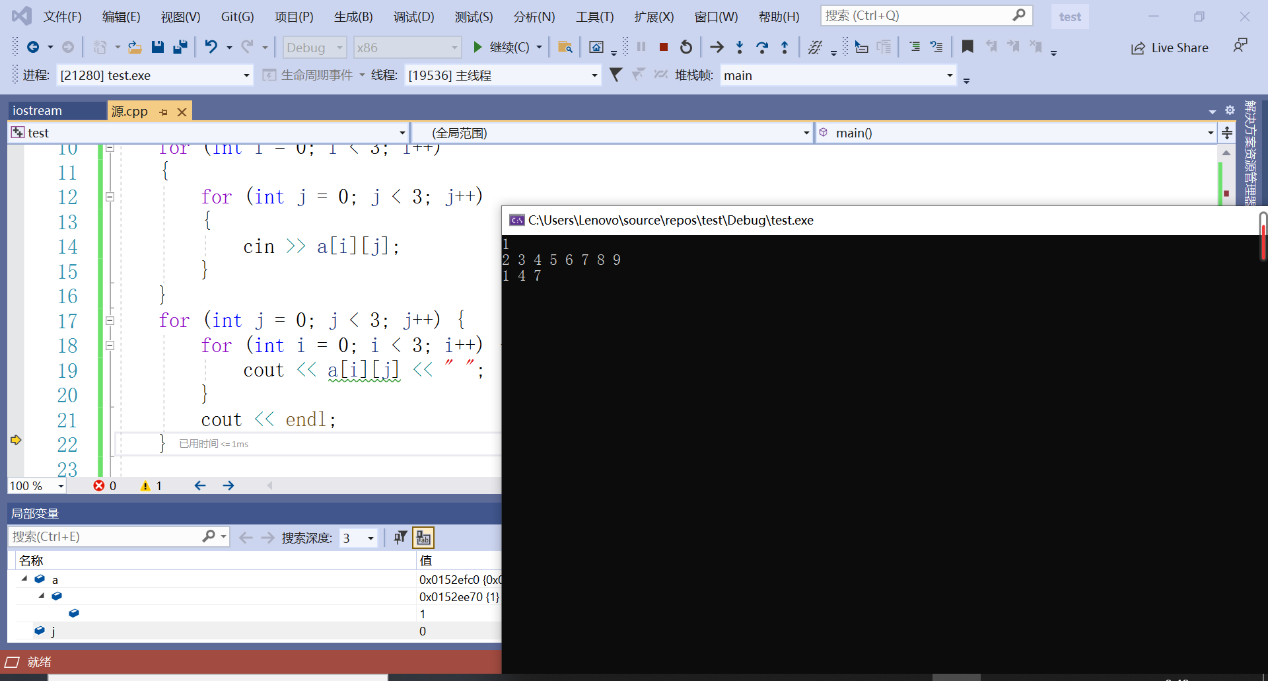
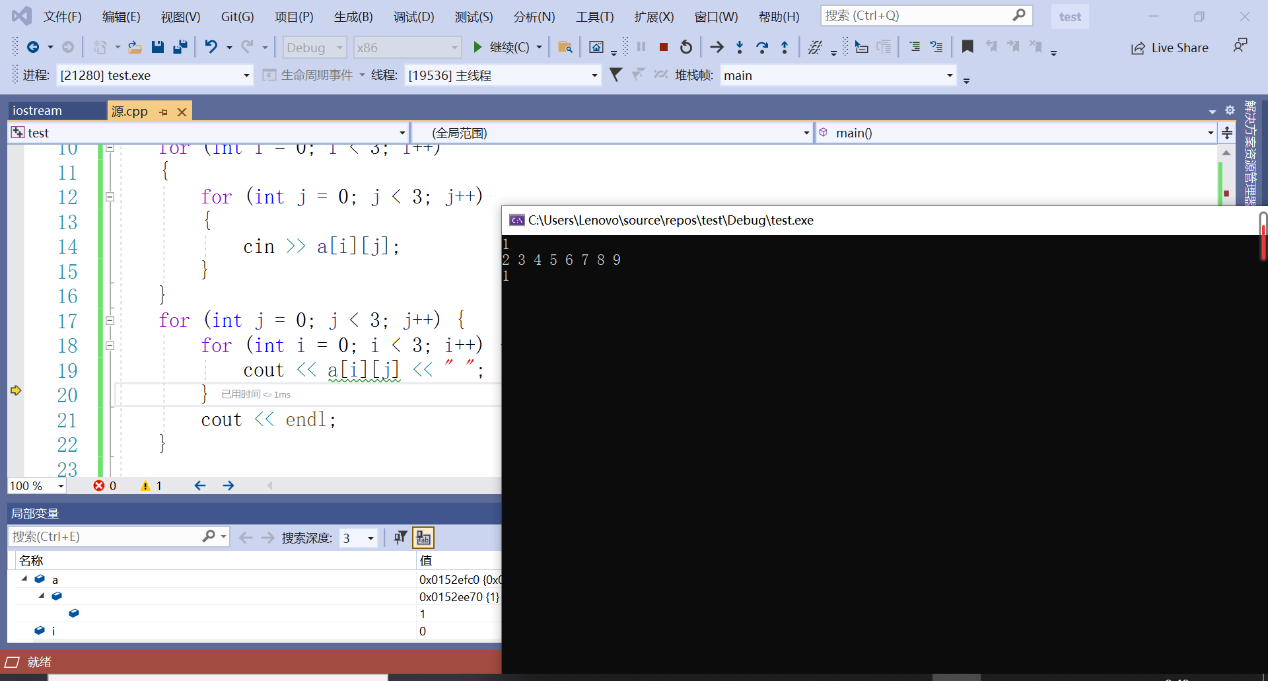
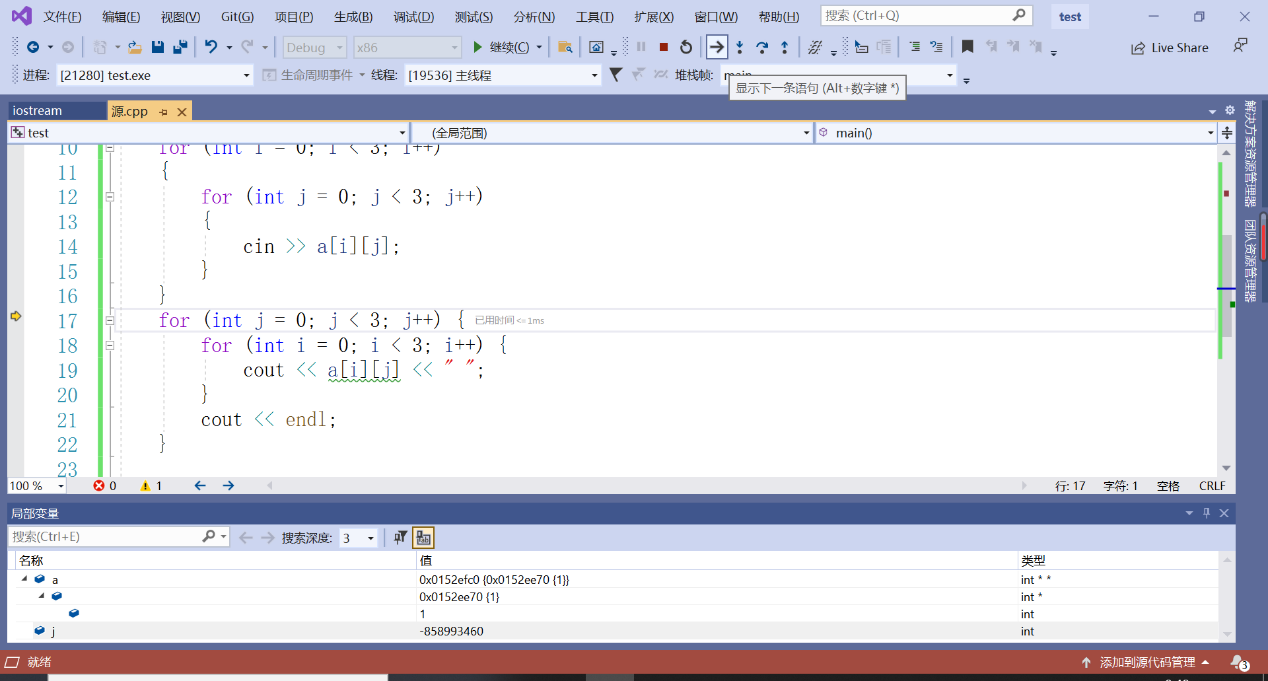
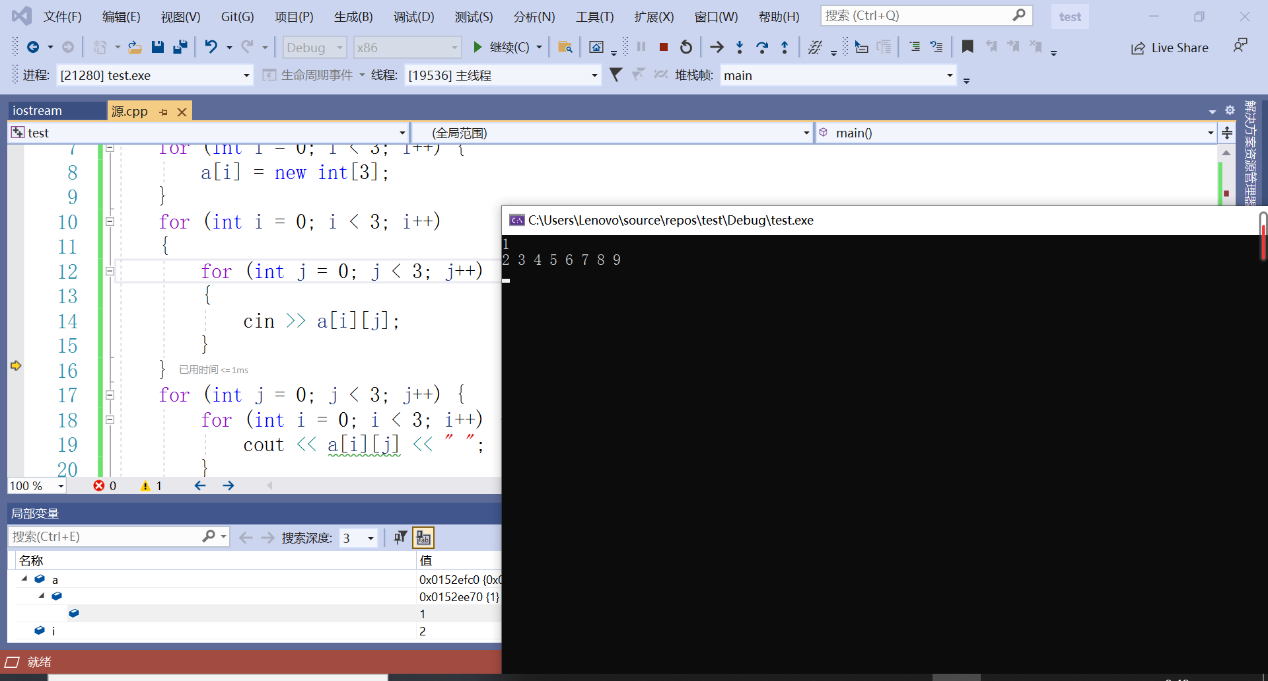
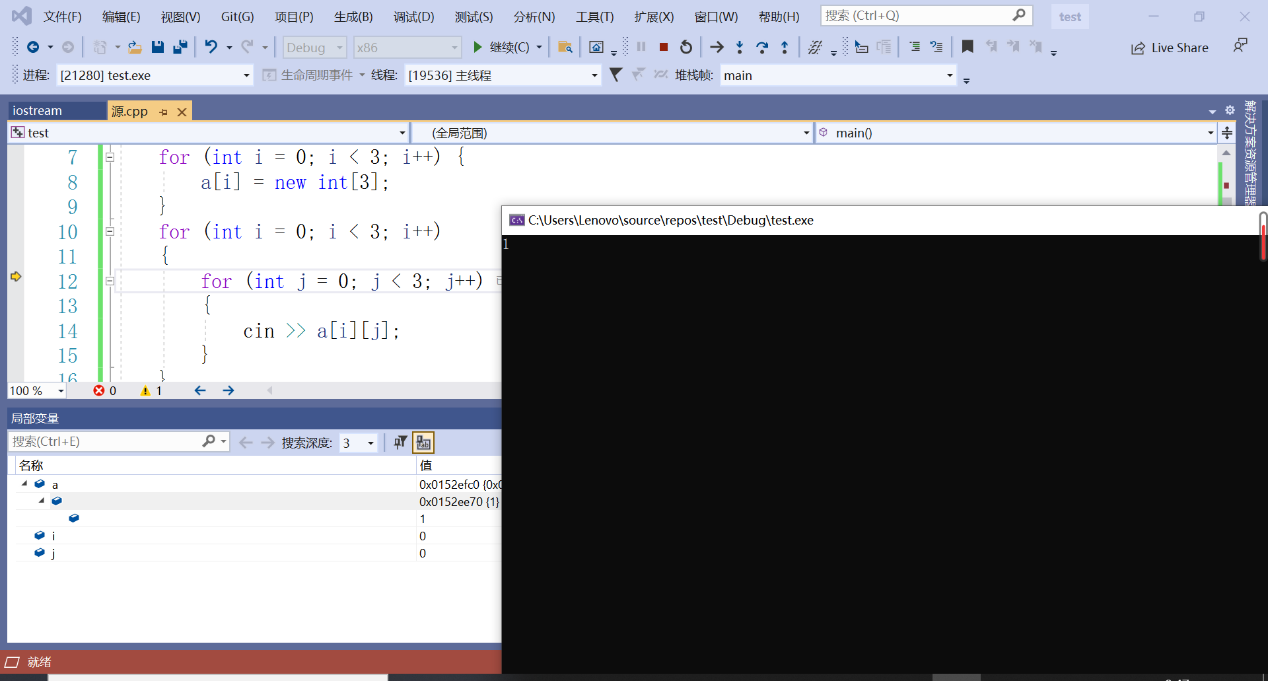
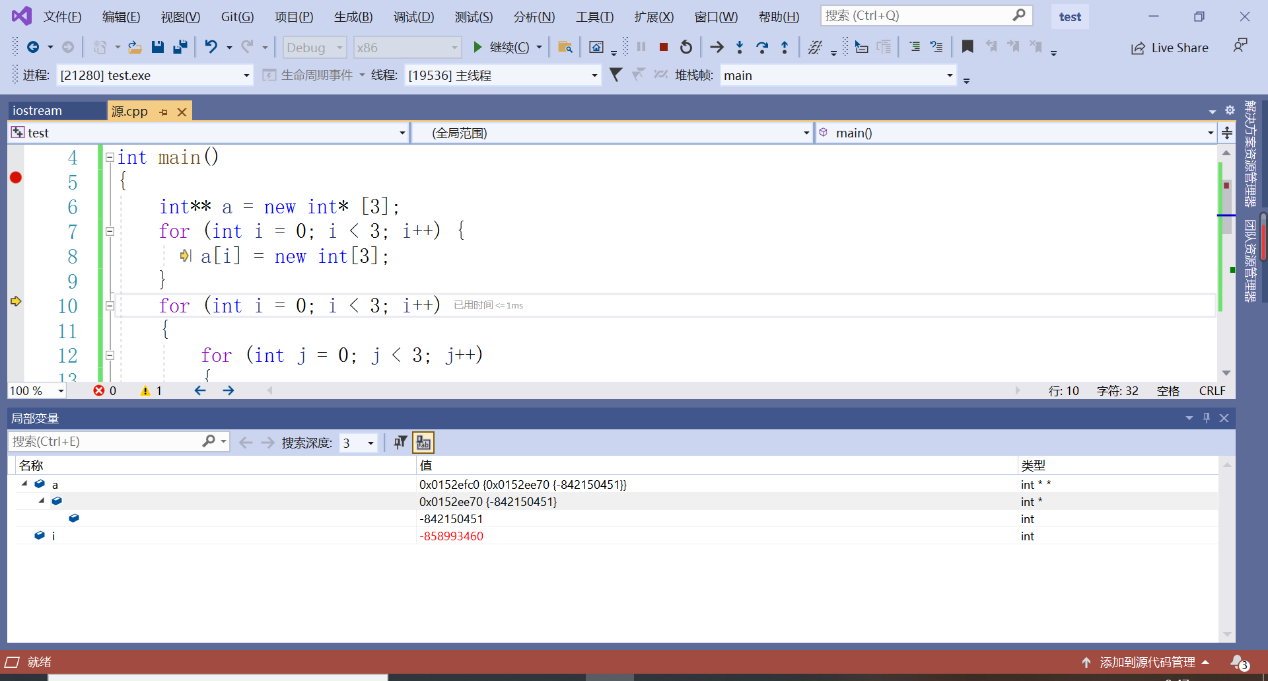
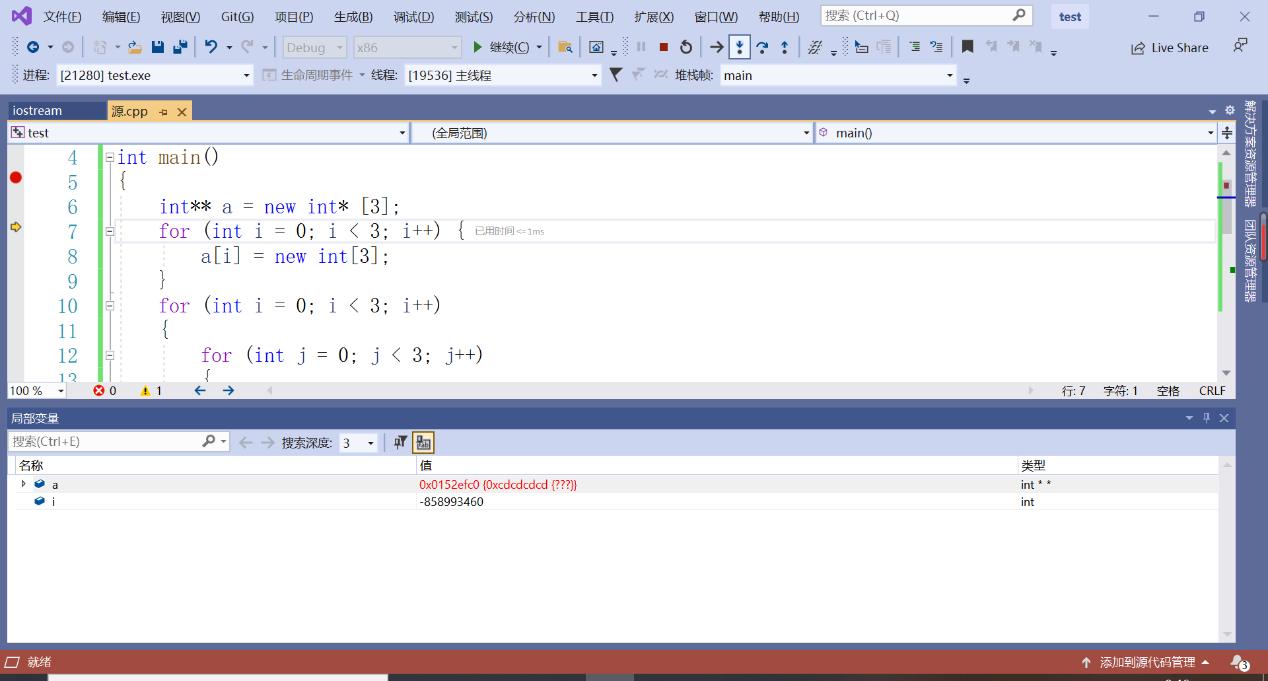
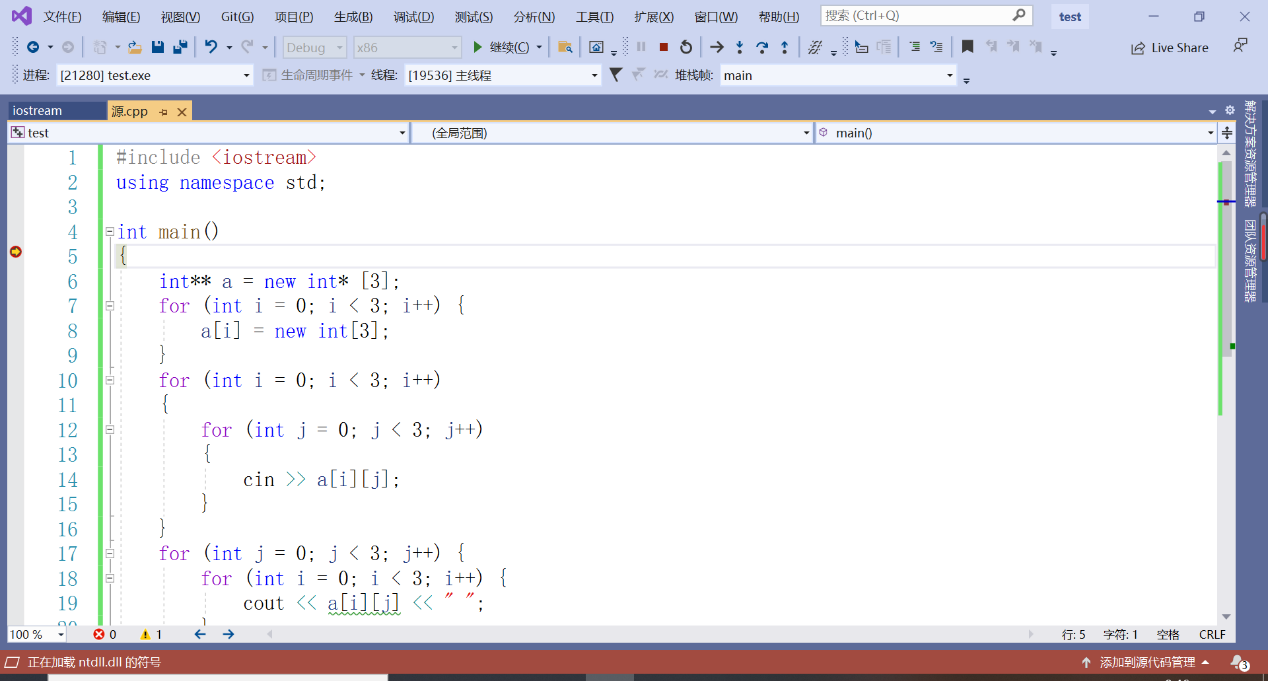
**2.1 结果**



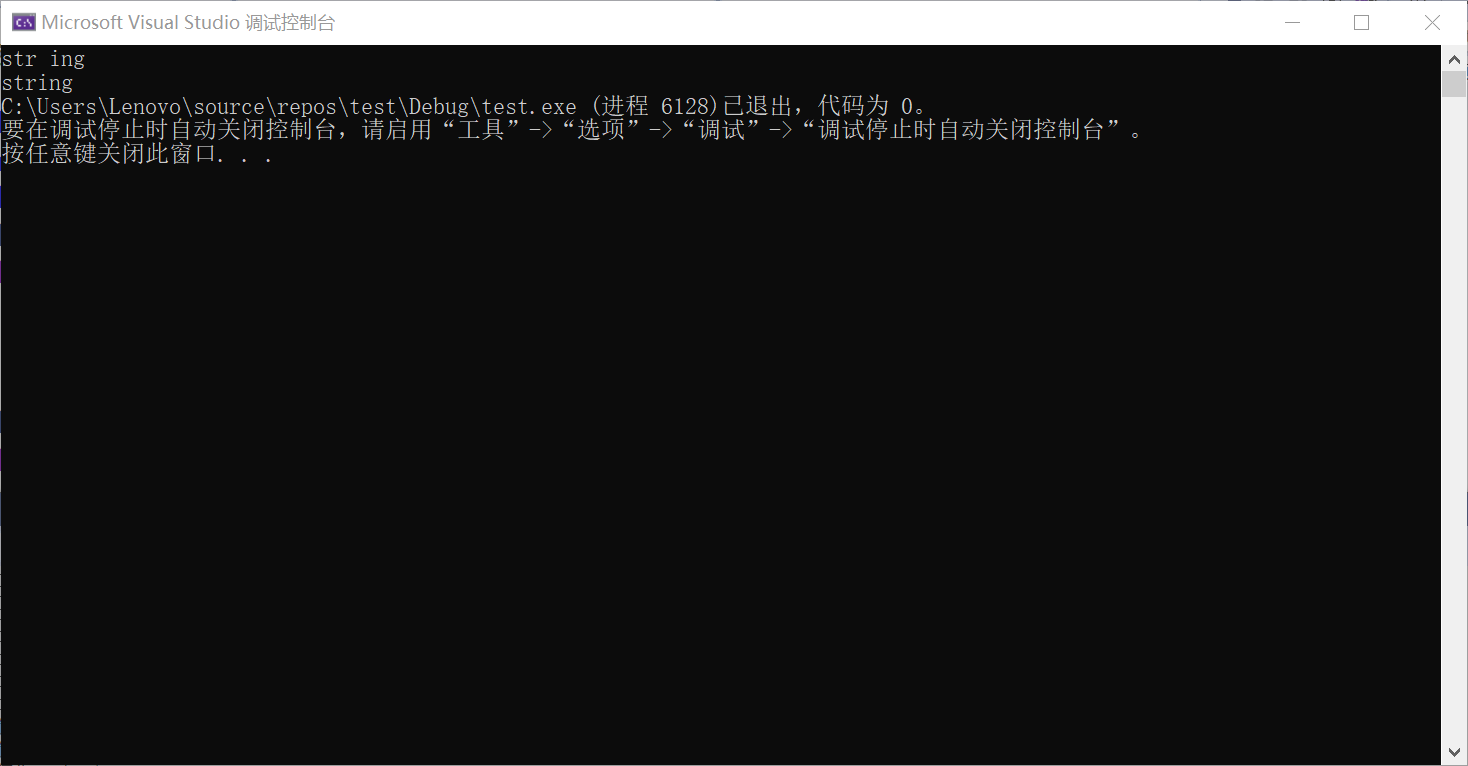
**2.2 结果**



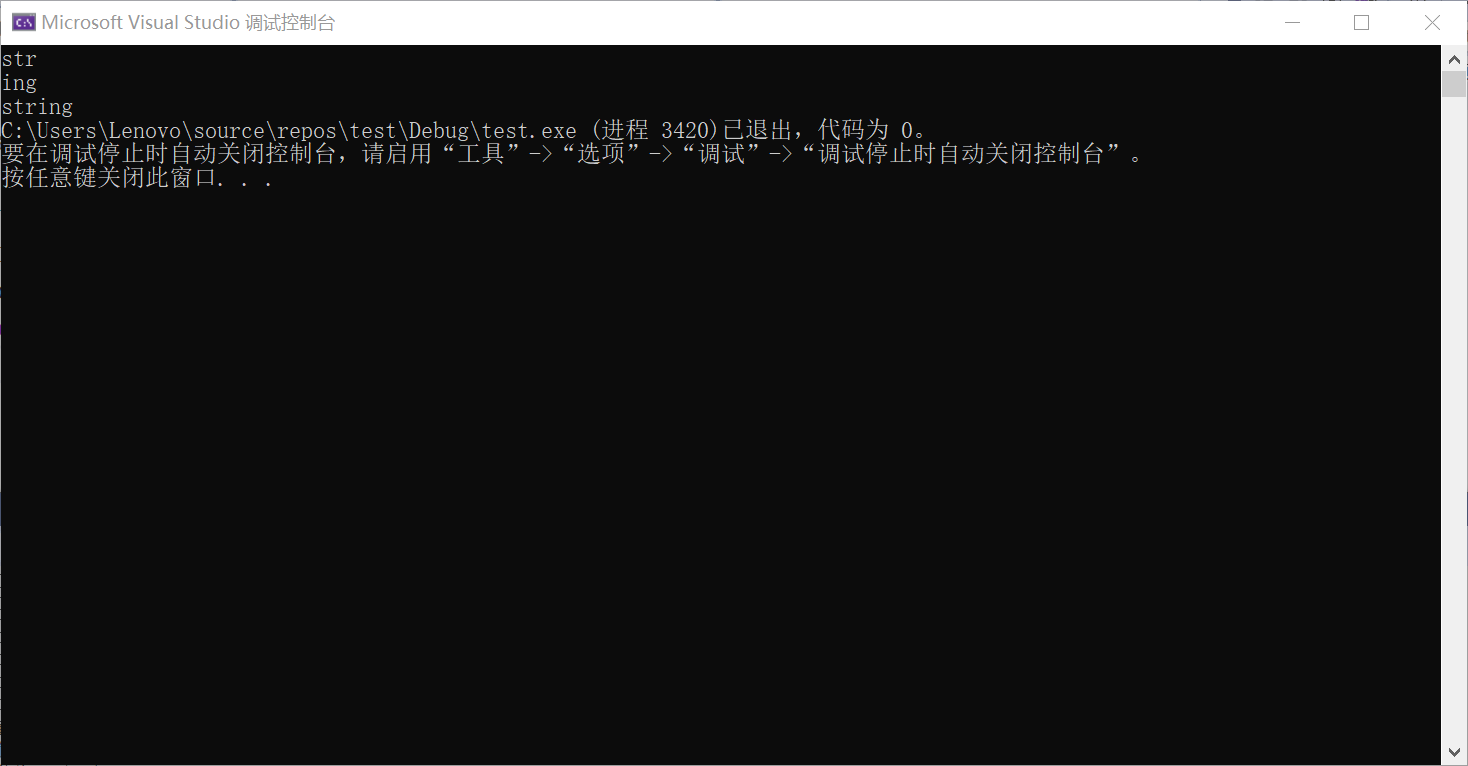
**2.2 debug**



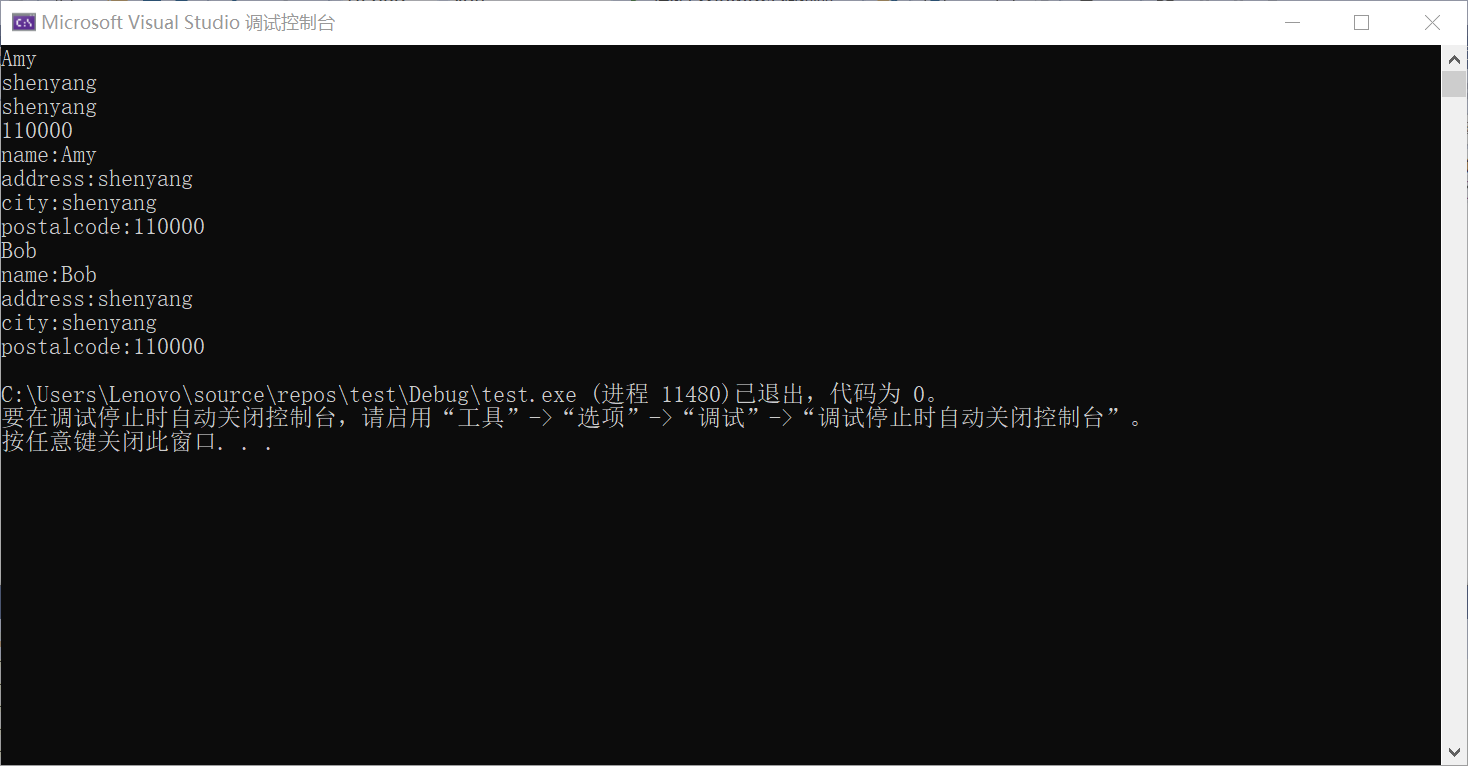
**2.3 结果**



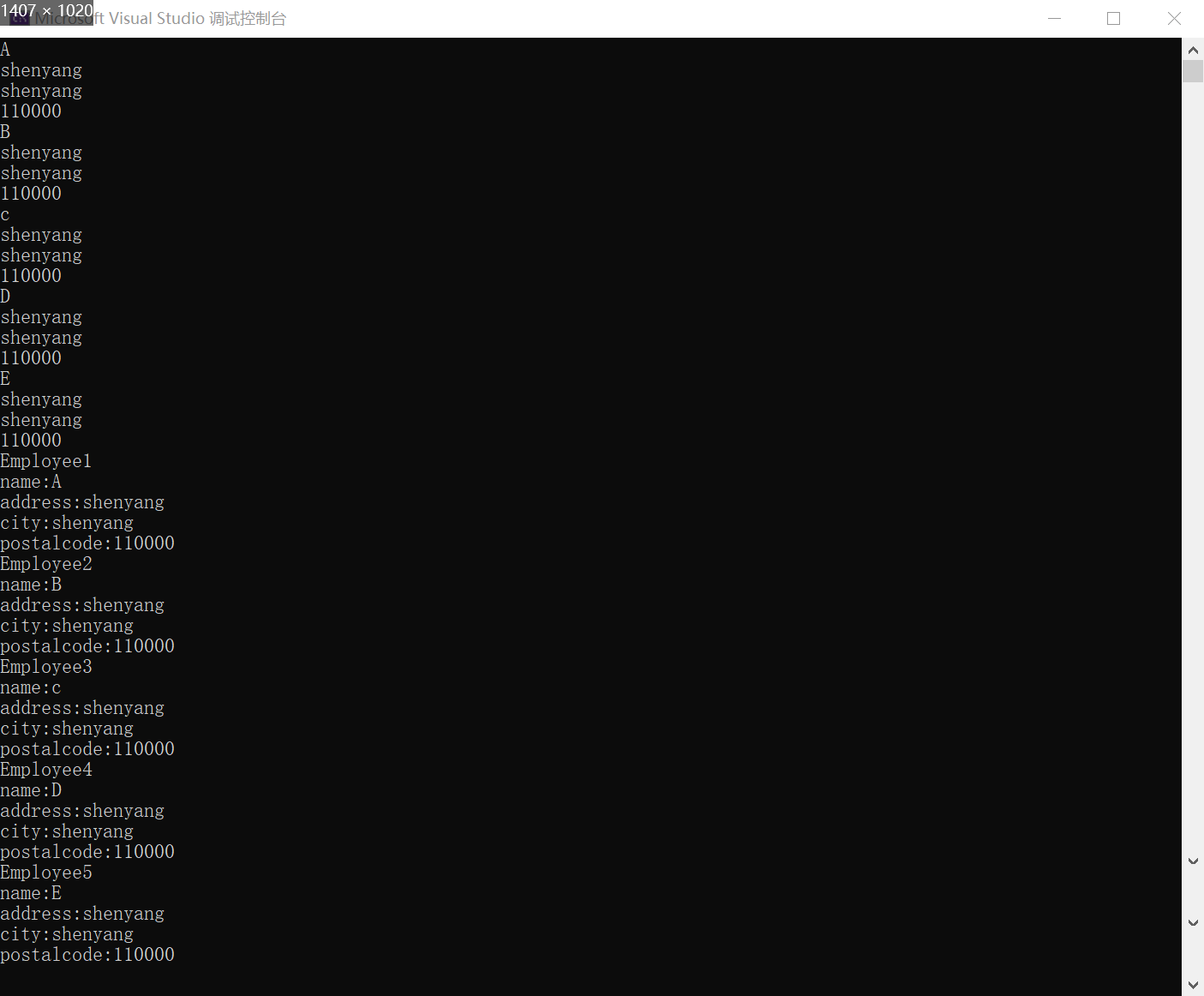
**2.4 结果**



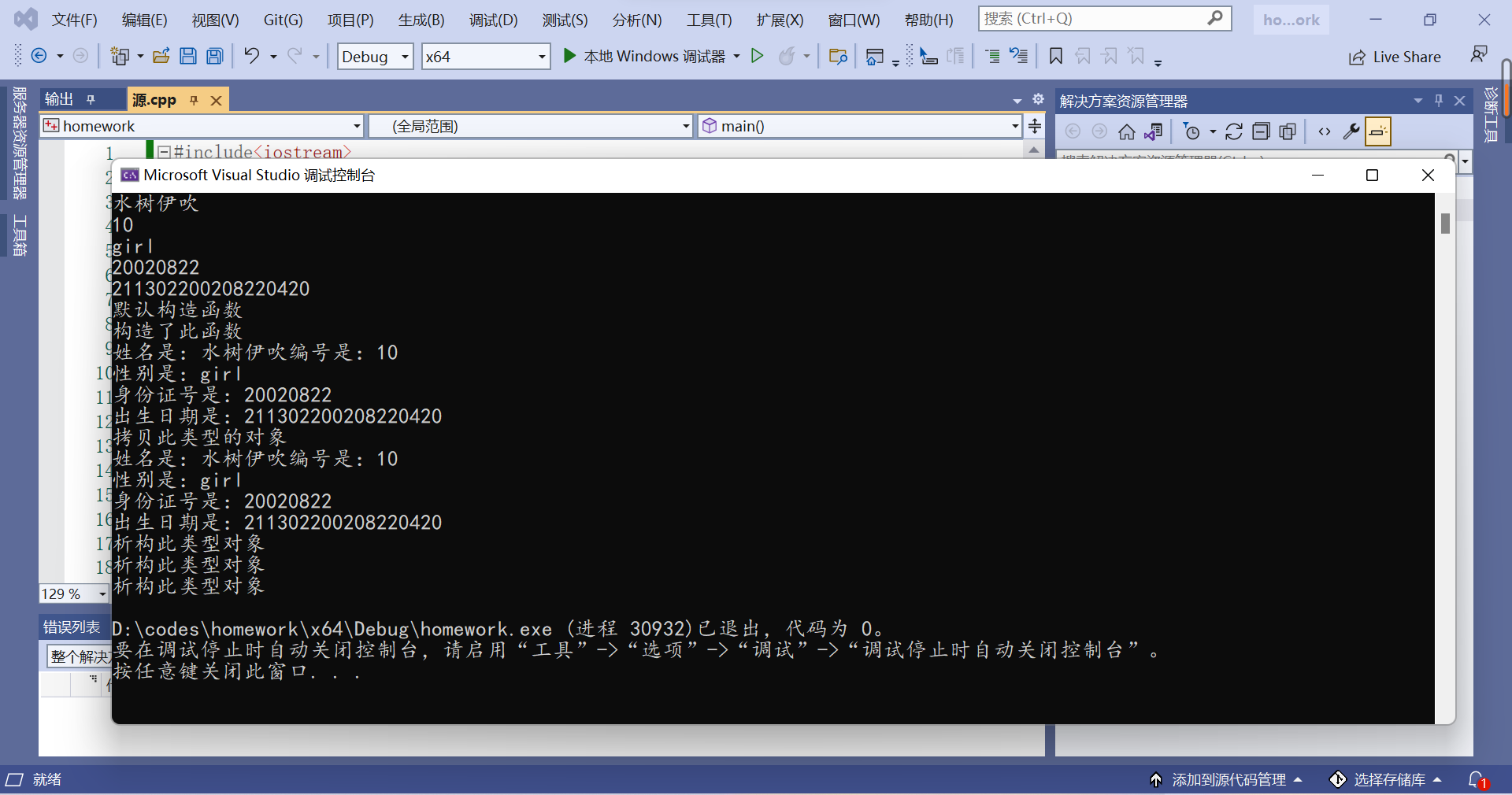
**2.5 结果**



**2.6 结果**



**2.7（选做）结果**



**6.心得体会：学到了什么；遇到的问题及解决方法等（可结合思考题）**

**思考题：**

1如何存储和处理字符串？

存储和处理可以用字符串数组，或者用string类处理，然后就可以用string类的函数，还有字符串的相关函数进行处理。

2头文件<string.h>和头文件<string>有何区别？

<string>和<string.h>是两个独立的头文件

string.h中包含一些常用的c字符串处理函数，比如strcmp，strlen等，操作的是char \*类型的变量。

<string>中是c++的头文件，包含了string类的操作。

但是<string>中的函数是包含在std空间之中的，如果要使用它，要加上using namespace std处理。

3有几种方法来表示和处理数组元素？

使用数组名和下标

使用数组名和指针运算

使用指针变量

**心得体会：**

1.通过本次实验课，更加熟练了数组和字符数组的用法。

2.其中第二道利用动态分配来实现，一开始对于二维数组的动态分配的知识并不熟悉，但是在查阅书籍之后，明白了应该先new一个二维指针，然后对于每一个一维指针再new一个int数组，以此达到题目的目的。

3.在动态内存分配实现动态数组时，通过Debug观察指针的内容及其所指的对象的内容。

4.更加熟悉了标准C++库（STL中的string）字符串以及相应函数的方法.

**实验三**

**1.实验目的**1.学习定义和使用类的继承关系，定义派生类2.熟悉不同继承方式下对基类成员的访问控制3.学习利用虚基类解决二义性问题

**2.实验要求**1.定义一个基类Animal，有私有整型成员变量age，构造其派生类dog，在其成员函数SetAge（int n）中直接给age赋值，看看会有什么问题，把age改为公有成员变量，还会有问题吗？编程尝试，并在本次报告的心得中说明对该问题的理解。2.定义一个基类BaseClass,有整型变量Number，构造其派生类DerivedClass（其中派生类有一基类BaseClass对象的私有成员，即 ），观察构造函数和析构函数的执行情况（在构造函数和析构函数中用cout输出提示信息）。 3.定义一个车（vehicle）基类，具有Maxspeed、Weight等成员，run、stop等成员函数，由此派生出自行车（bicycle）类、汽车（motorcar）类。自行车（bicycle）类有高度（Height）等属性，汽车（motorcar）类有座位数（SeatNun）等属性。从bicycle和motorcar派生出摩托车（motorbicycle）类，在继承过程中，注意将适当的类设置为虚基类。在主函数中调用vehicle类的run和stop函数，编程完成各个类的定义。如果不设置为虚基类，会有什么问题？（在本次报告的心得中说明对该问题的理解）

4.以学校职工为基类，派生出教师类和管理人员类，又从教师类和管理人员类共同派生出教师管理人员类。

【输出形式】

Zhao is a Staff 20 years old

Zhang is a Teacher 30 years old, Lecture

Wang is a management 50 years old, dean

Li is a Teacher management 40 years old, Peofessor,

department head

**3.实验内容（各题目的题干）**

1.编写程序定义基类Animal，成员变量age定义为私有的。构造派生类dog，在其成员函数SetAge（int n）中直接对age赋值时，会出现类似以下的错误提示： error C2248 : ‘age’ : cannot access private member declared in class ‘Animal’ error C2248 : ‘age’ : cannot access private member declared in class ‘Animal’ 把age改为公有成员变量后重新编译就可以了。程序名：lab3\_1.cpp。2.编写程序定义一个基类BaseClass，构造其派生类DerivedClass，在构造函数和析构函数中用cout输出提示信息，观察构造函数和析构函数的执行情况。程序名：lab3\_2.cpp。3.用debug功能跟踪程序lab3\_2的执行过程，观察基类和派生类的构造函数和析构函数的执行情况（需要有证明完成debug跟踪观察基类和派生类的构造函数和析构函数的执行情况的截屏，放在本次报告的运行结果部分）。4.编写程序定义一个车（vehicle）基类，由此派生出自行车（bicycle）类、汽车（motorcar）类，注意把vehicle派生为虚基类。再从bicycle和motorcar派生出摩托车（motorcycle）类，在main()函数中测试这个类。程序名:lab3\_3.cpp。 编译成功后，把vehicle设置为非虚基类，再编译一次，此时系统报错，无法编译成功。这是因为若不把vehicle设置为虚基类，会出现二义性错误，程序不能成功编译。

5.以学校职工为基类，派生出教师类和管理人员类，又从教师类和管理人员类共同派生出教师管理人员类。

【输出形式】

Zhao is a Staff 20 years old

Zhang is a Teacher 30 years old, Lecture

Wang is a management 50 years old, dean

Li is a Teacher management 40 years old, Peofessor,

department head

**4.源程序（加注释）3.1**

1. #include<iostream>
2. **using** **namespace** std;
3. **class** Animal {
4. **public**:
5. **int** age;
6. };
8. **class** dog :**public** Animal
9. {
10. **public**:
11. **void** SetAge(**int** n)
12. {
13. **this**->age = n;
14. }
16. **int** GetAge(**void**)
17. {
18. **return** **this**->age;
19. }
20. };
22. **int** main()
23. {
24. dog d;
25. d.SetAge(1);
26. cout<<d.GetAge()<<endl;
27. **return** 0;
28. }

**3.2**

1. #include <iostream>
2. **using** **namespace** std;
4. **class** BaseClass
5. {
6. **public**:
7. **int** Number;
8. BaseClass()
9. {
10. cout << "Construct BaseClass" << endl;
11. }
12. ~BaseClass()
13. {
14. cout << "Destruct BaseClass" << endl;
15. }
16. };
18. **class** DerivedClass :**public** BaseClass
19. {
20. **private**:
21. BaseClass b;
22. **public**:
23. DerivedClass()
24. {
25. cout << "Construct DerivedClass" << endl;
26. }
27. ~DerivedClass()
28. {
29. cout << "Destruct DerivedClass" << endl;
30. }
31. };
33. **int** main()
34. {
35. DerivedClass obj(1, 2);
36. **return** 0;
37. }

**3.3**

1. #include<iostream>
2. **using** **namespace** std;
3. **class** vehicle
4. {
5. **public**:
6. **void** run(){
7. cout << "vehicle run" << endl;
8. };
9. **void** stop(){
10. cout << "vehicle stop" << endl;
11. };
12. **private**:
13. **int** Maxspeed, Weight;
14. };

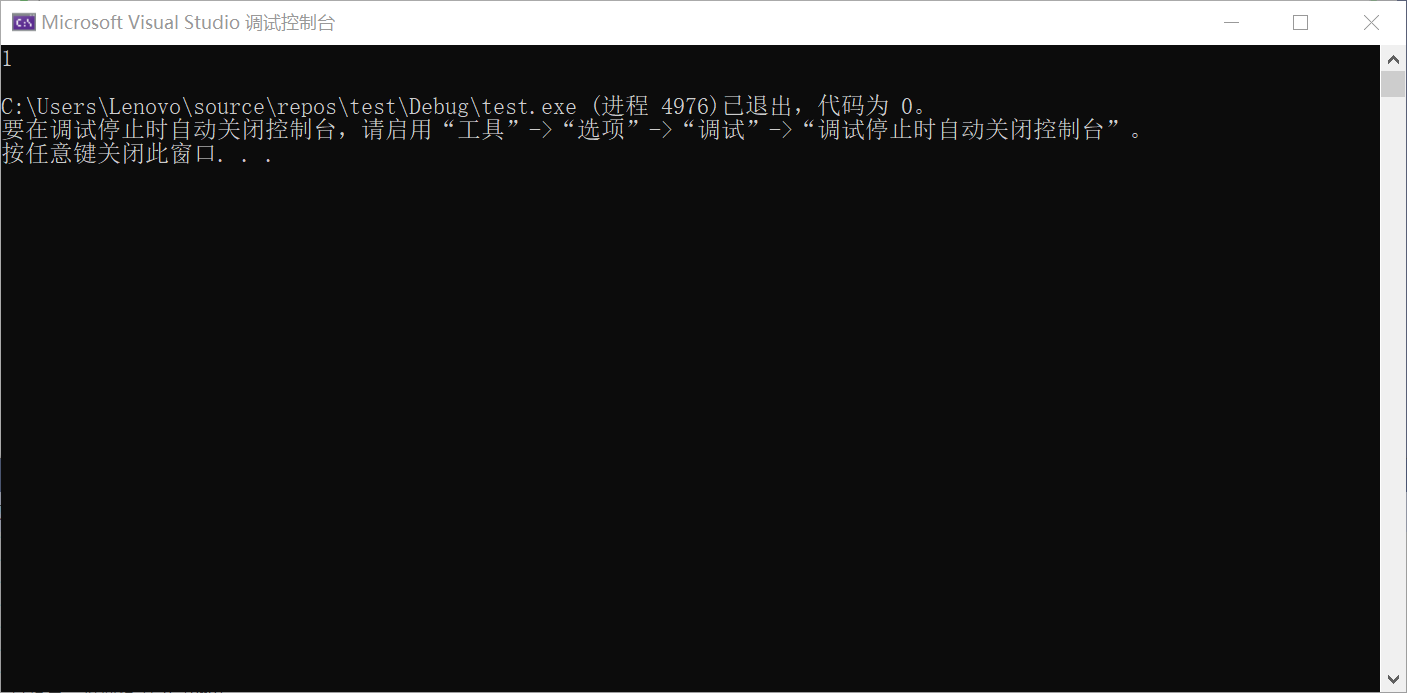
17. **class** bicycle :**virtual** **public** vehicle
18. {
19. **private**:
20. **int** Height;
21. };
22. **class** motorcar :**virtual** **public** vehicle
23. {
24. **private**:
25. **int** SeatNun;
26. };
27. **class** motorbicycle :**public** motorcar, **public** bicycle
28. {
29. };
31. **int** main()
32. {
33. motorbicycle m;
34. m.run();
35. m.stop();
36. **return** 0;
37. }

**3.4**

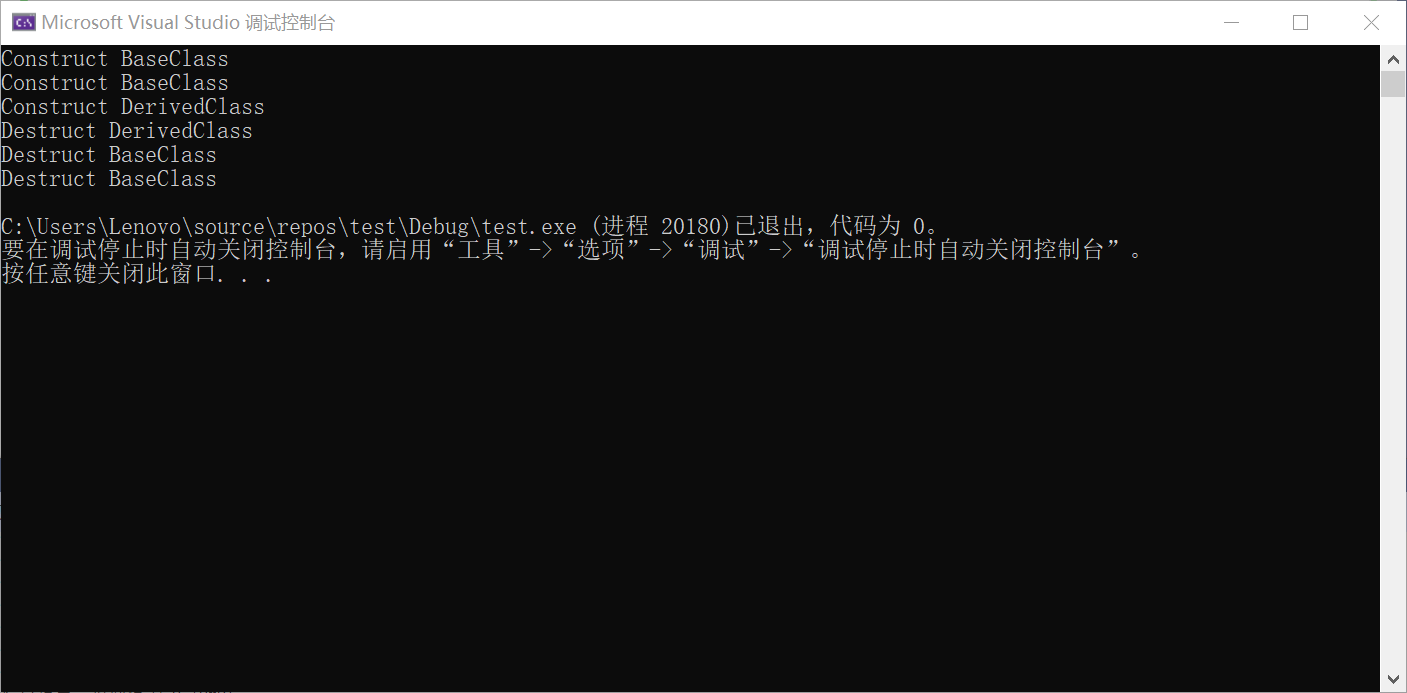
1. #include  <iostream>
2. **using**  **namespace**  std;
3. #include<cstring>
4. **class**  CStaff
5. {
6. **protected**:
7. **int**  number;
8. **char**  name[10];
9. **int**  age;
10. **public**:
11. CStaff(**int**  num, **const**  **char**\* na, **int**  a)
12. {
13. number = num;
14. age = a;
15. strcpy(name, na);
16. }
17. **void**  Display()
18. {
19. cout << name << "  is  a  Staff  " << age << "  yeas  old" << endl;
20. }
21. };
23. **class** CTeacher : **virtual** **public** CStaff
24. {
25. **protected**:
26. **char**  zch[20];          //职称
27. **public**:
28. CTeacher(**int**  num, **const**  **char**\* na, **int**  a, **const**  **char**\* zc);
29. **void**  Display()
30. {
31. cout << name << "  is  a  Teacher  " << age << "  yeas  old,  " << zch << endl;
32. }
33. };
34. CTeacher::CTeacher(**int**  num, **const**  **char**\* na, **int**  a, **const**  **char**\* zc):CStaff(num, na, a)
35. {
36. strcpy(zch, zc);
37. }
39. **class** CManagement : **virtual** **public** CStaff
40. {
41. **protected**:
42. **char**  zw[50];                          //职务
43. **public**:
44. CManagement(**int**  num, **const**    **char**\* na, **int**  a, **const**  **char**\* z);
45. **void**  Display()
46. {
47. cout << name << "  is  a  management  " << age << "  yeas  old,  " << zw << endl;
48. }
49. };
50. CManagement::CManagement(**int**  num, **const**    **char**\* na, **int**  a, **const**  **char**\* z):CStaff(num, na, a)
51. {
52. strcpy(zw, z);
53. }
55. **class** CTeacherManagement :**public** CTeacher,**public** CManagement
56. {
57. **public**:
58. CTeacherManagement(**int**  num, **const**  **char**\* na, **int**  a, **const**  **char**\* zc, **const**  **char**\* z);
59. **void**  Display()
60. {
61. cout << name << "  is  a  Teacher  management  " << age << "  yeas  old,  "
62. << zch << ",  " << zw << endl;
63. }
64. };
66. CTeacherManagement::CTeacherManagement(**int**  num, **const**  **char**\* na, **int**  a, **const**  **char**\* zc, **const**  **char**\* z): CStaff(num, na, a), CManagement(num,na,a,z), CTeacher(num,na,a,zc){}

69. **int**  main()
70. {
71. CStaff      s1(101, "Zhao", 20);
72. CTeacher  t1(102, "Zhang", 30, "Lecture");
73. CManagement  m1(103, "Wang", 50, "dean");
74. CTeacherManagement  tm1(104, "Li", 40, "Peofessor", "department  head");
75. s1.Display();
76. t1.Display();
77. m1.Display();
78. tm1.Display();
79. **return**  0;
80. }

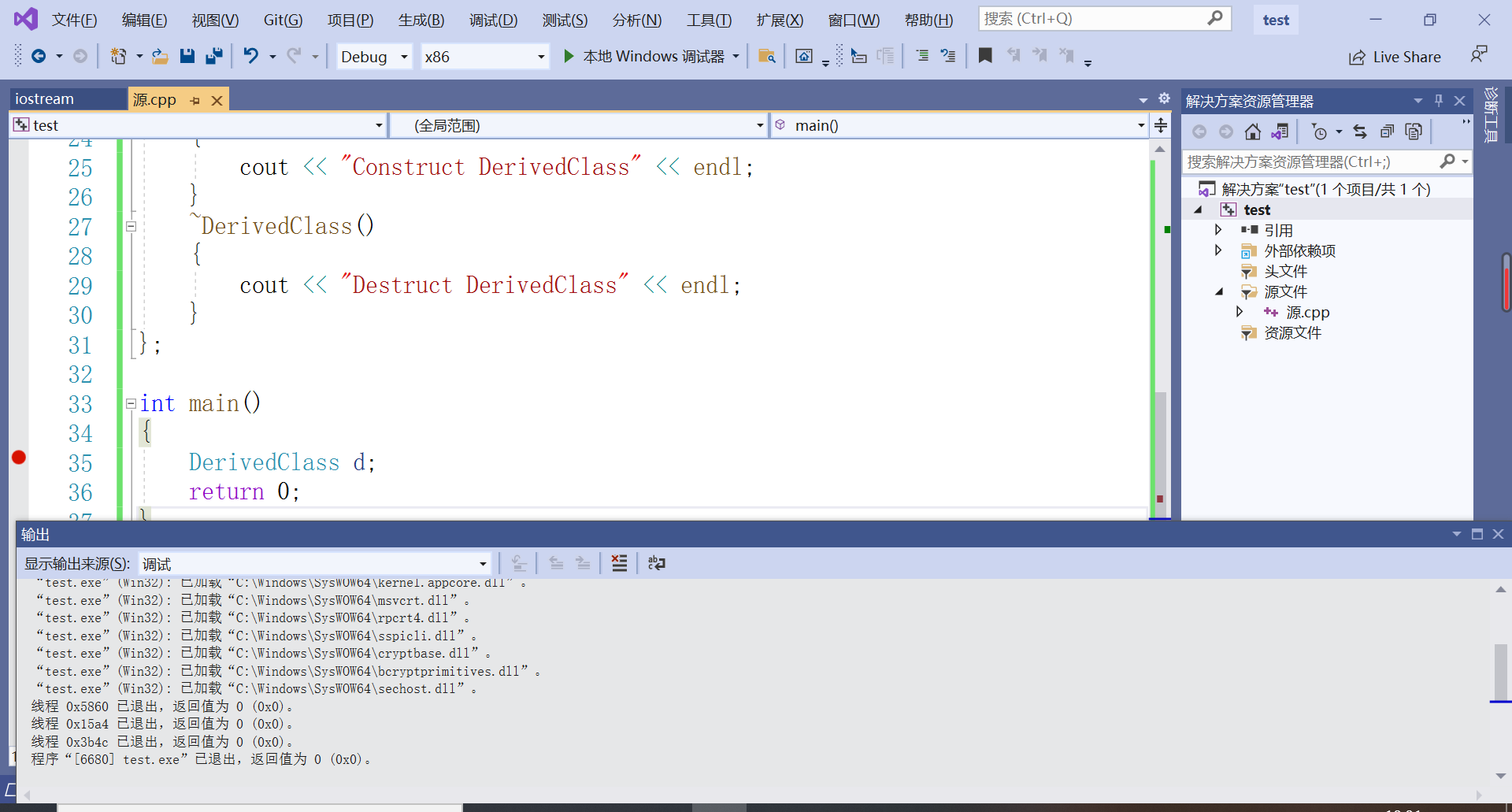
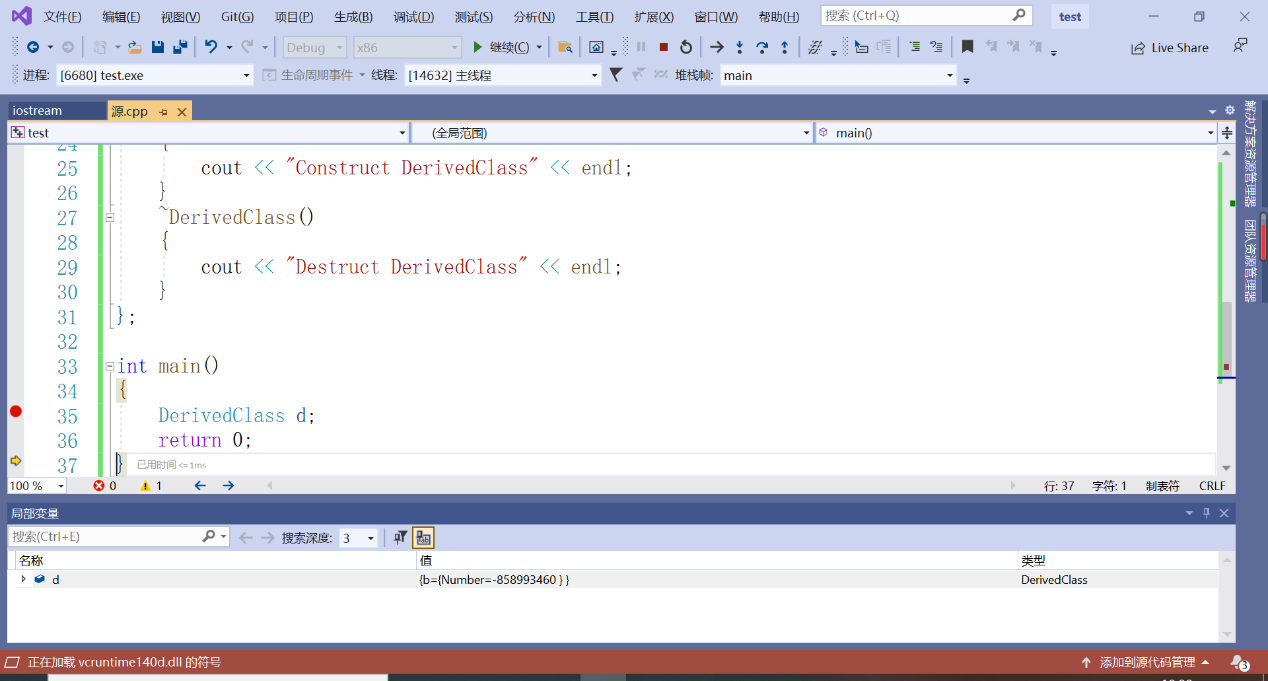
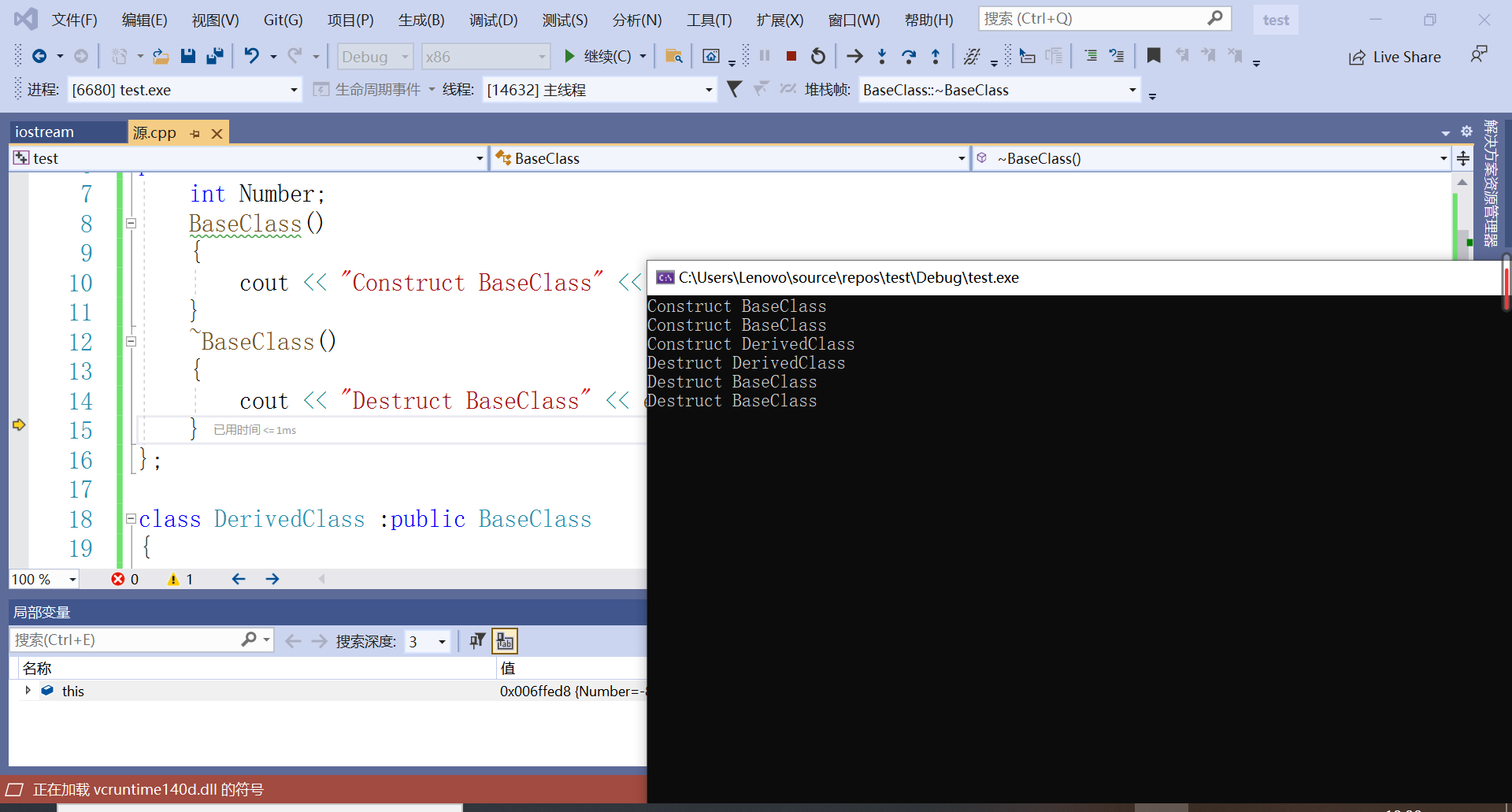
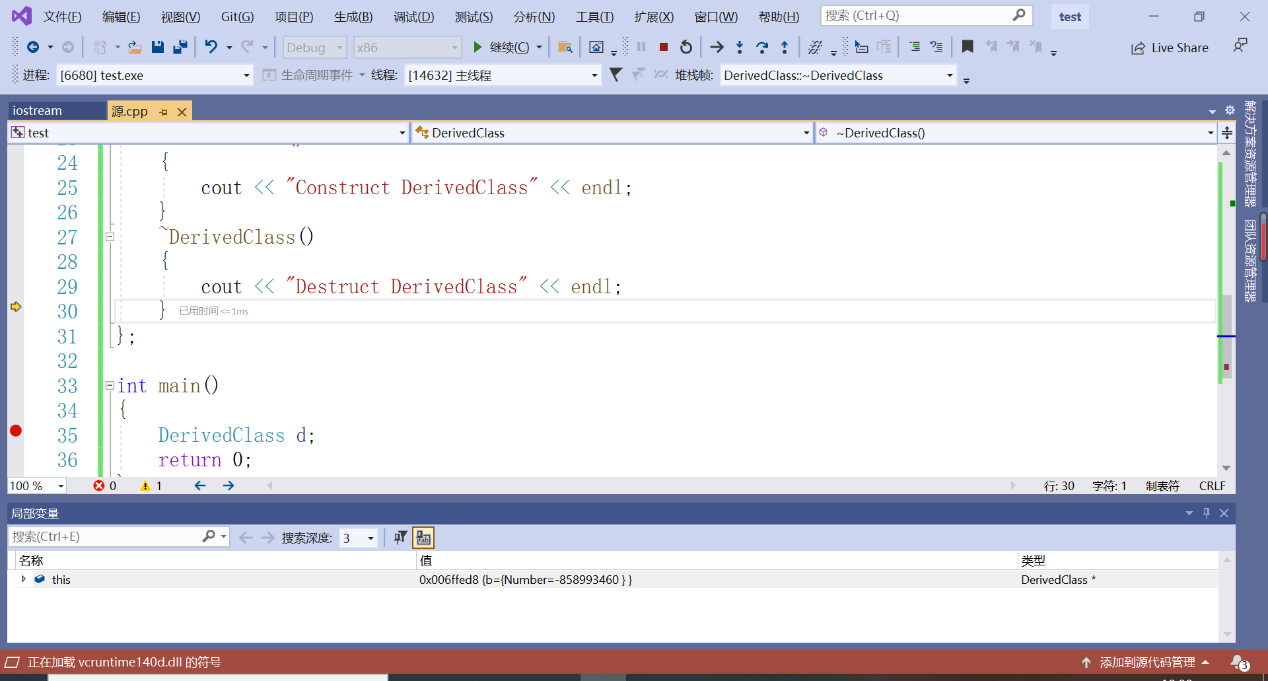
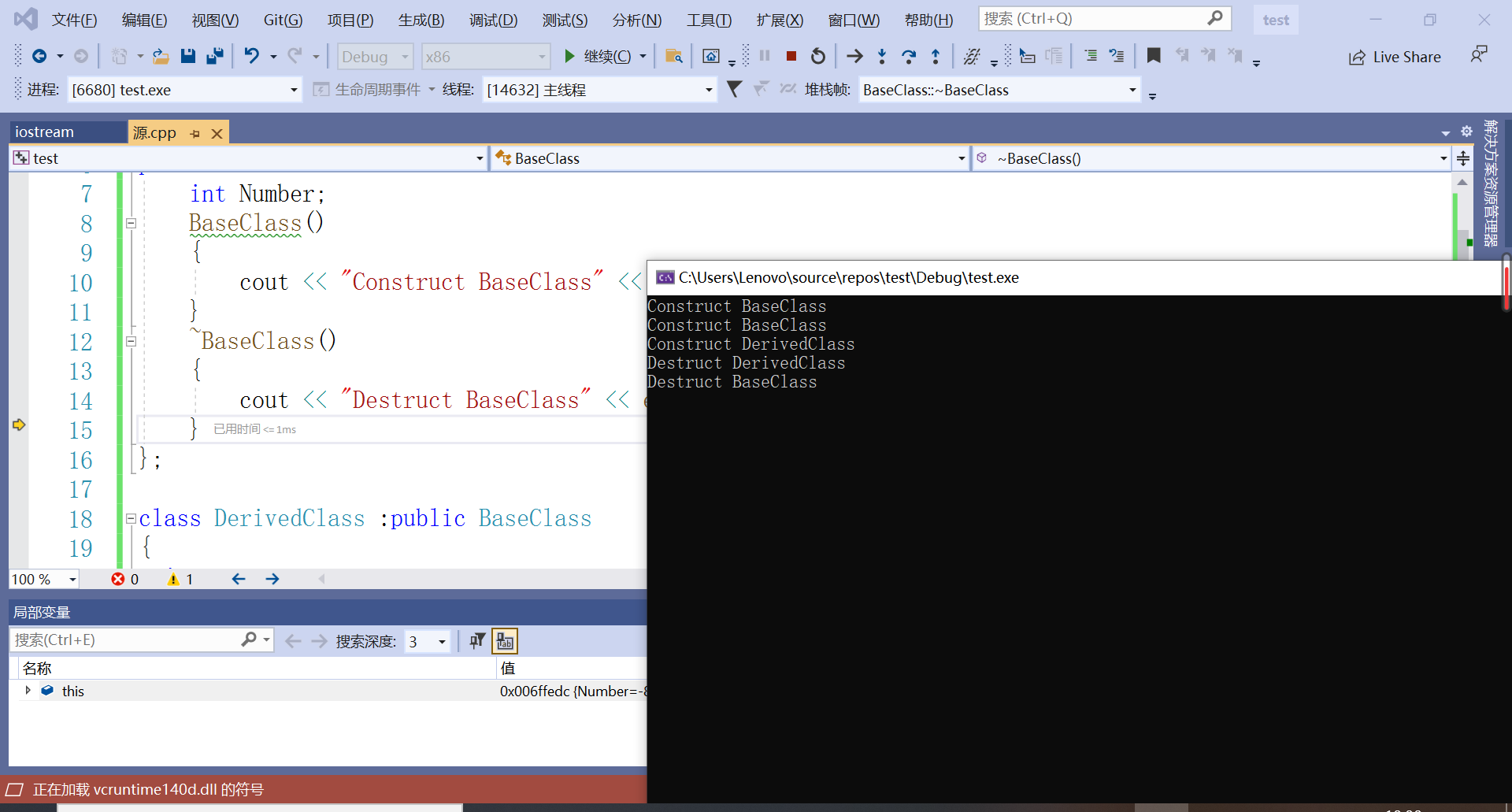
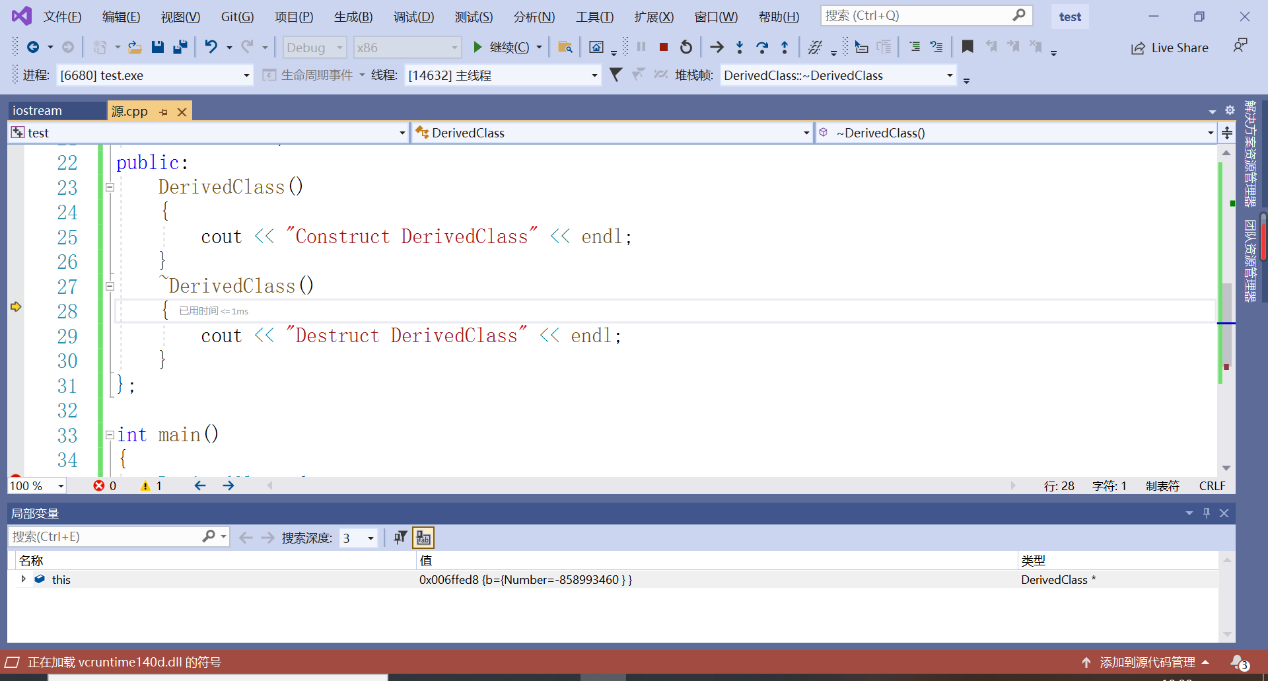
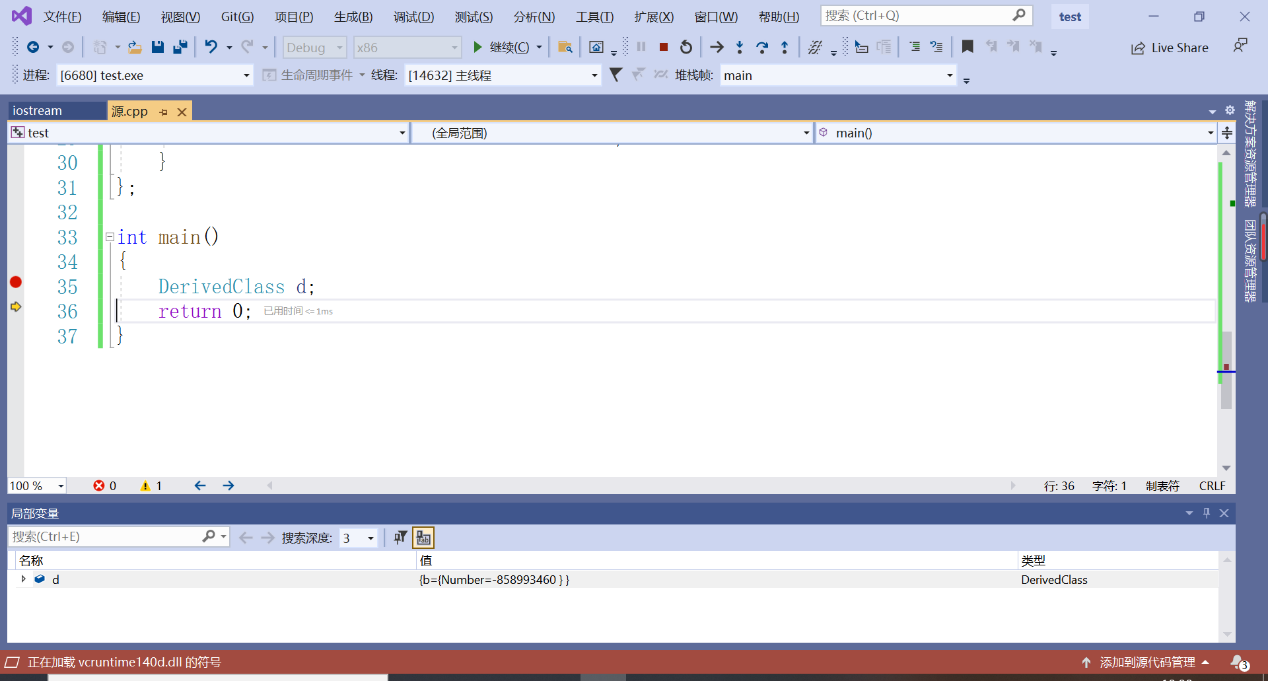
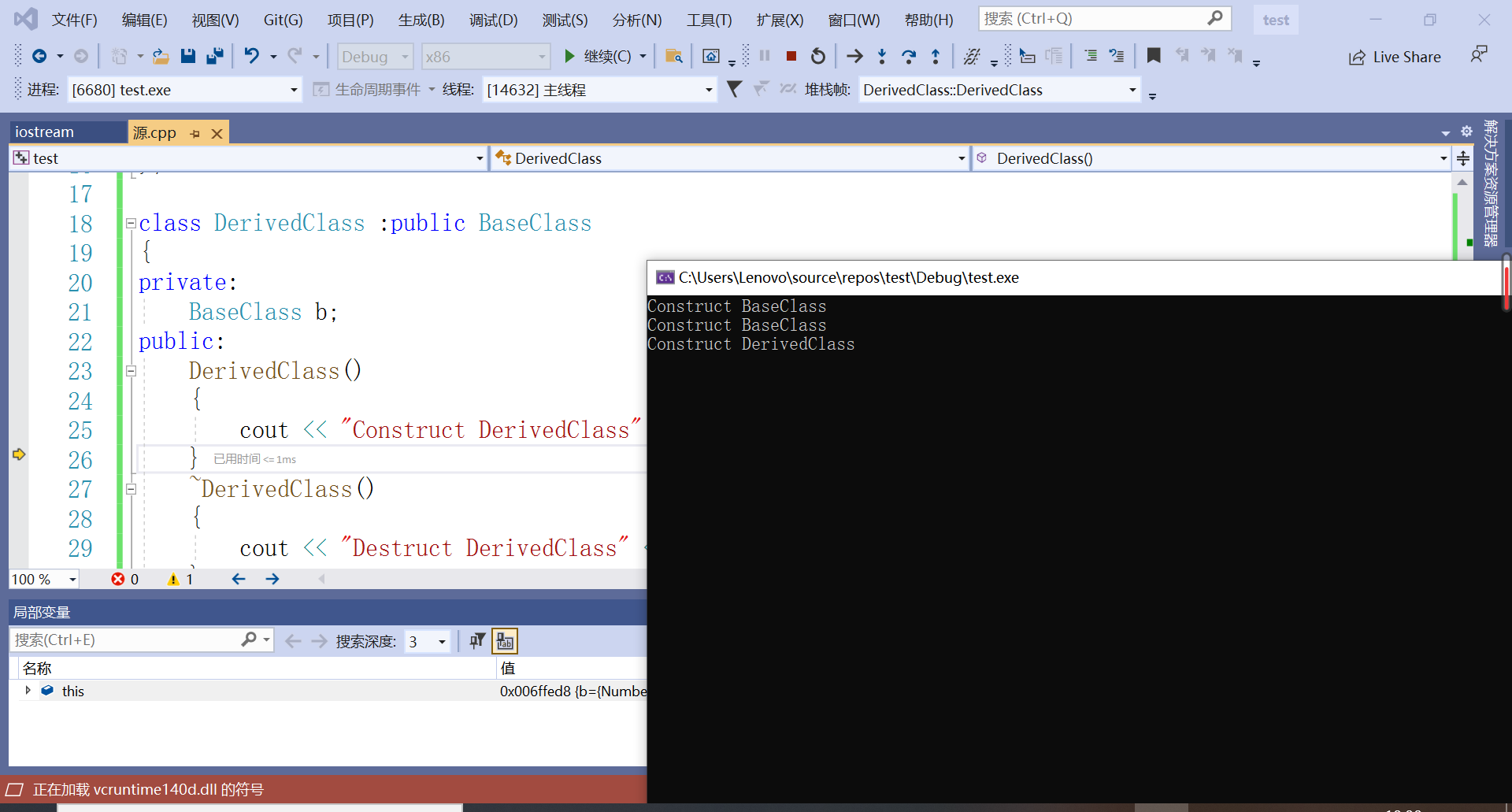
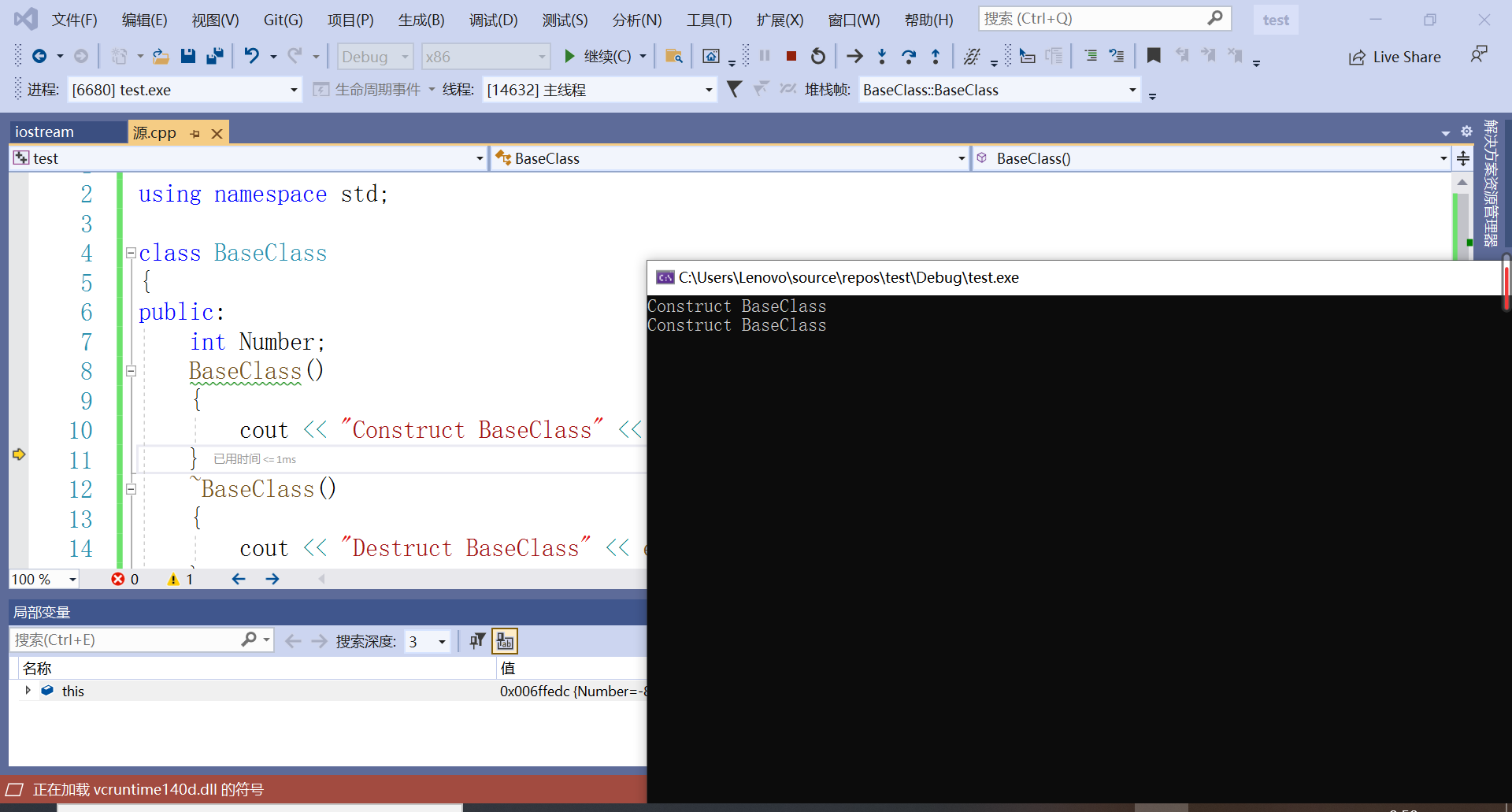
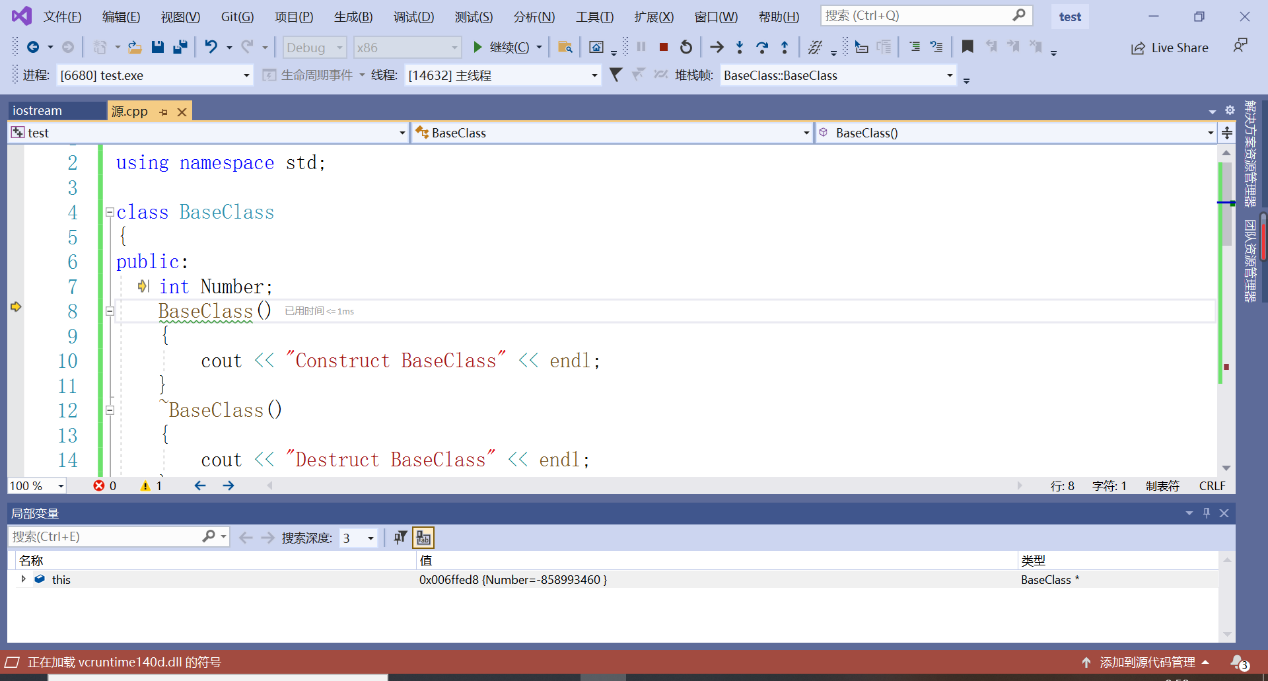
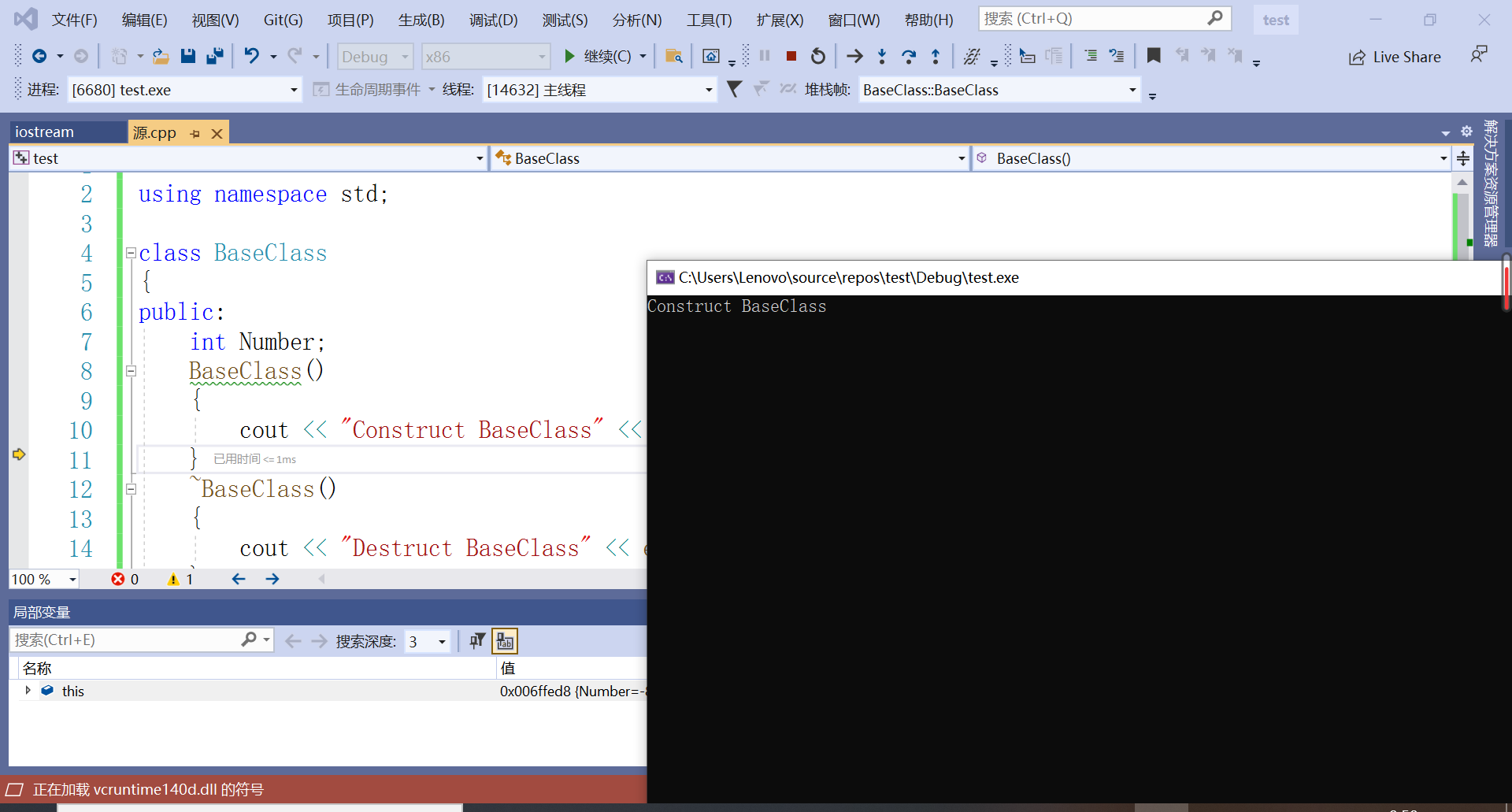
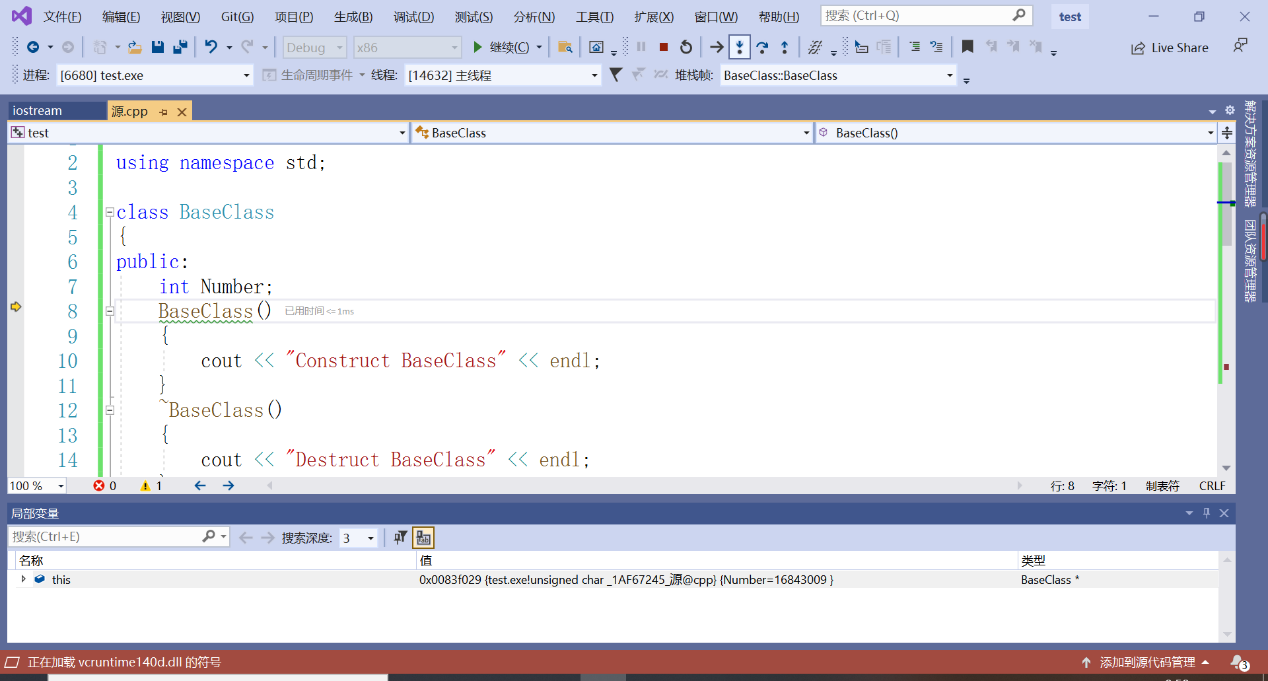
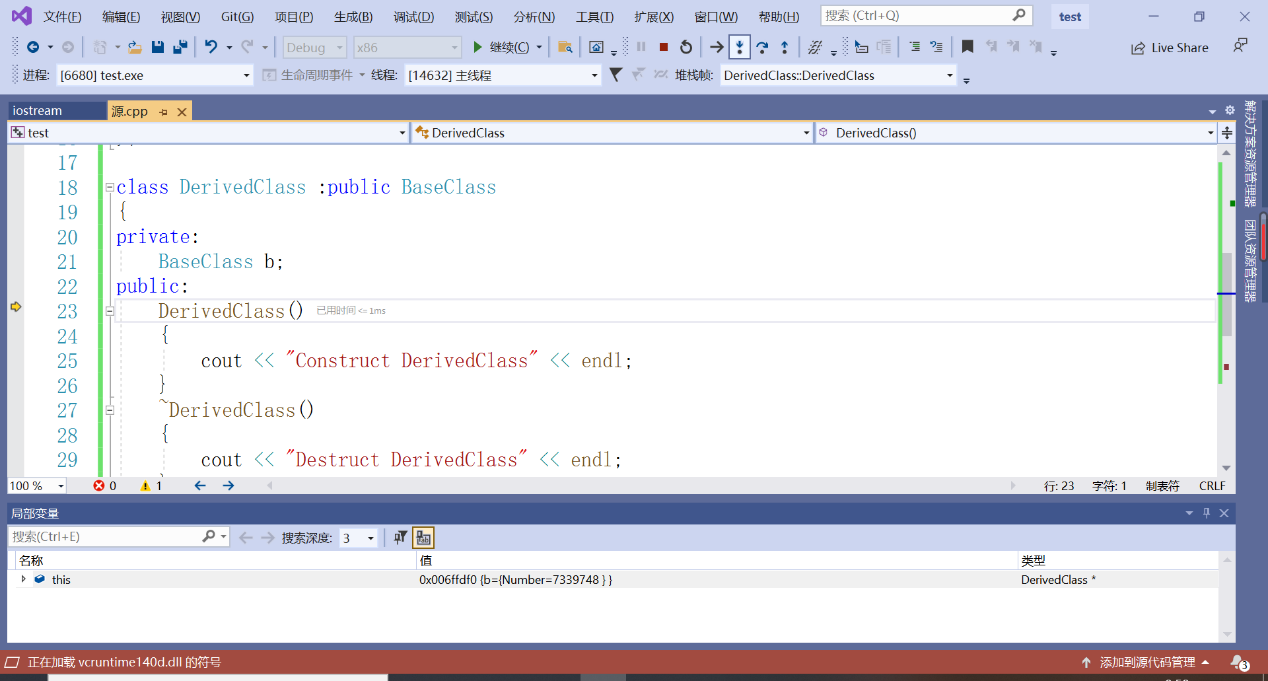
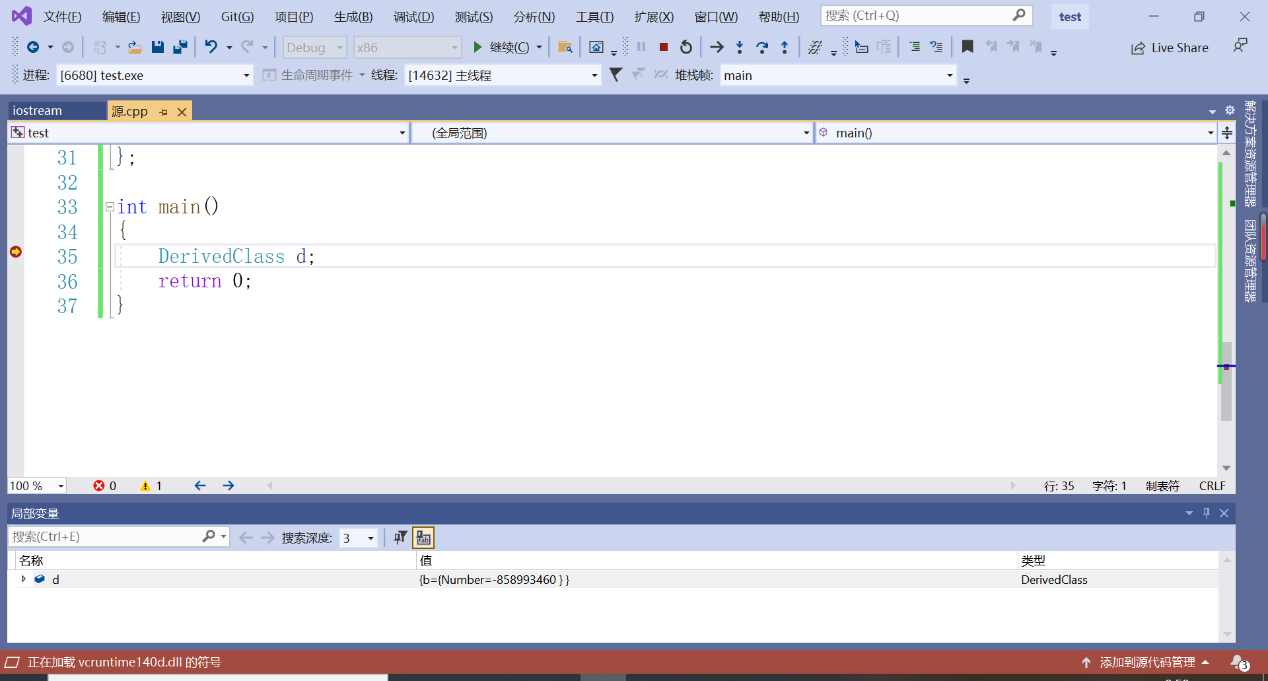
**5.运行结果（贴图） 第三次实验截屏要求：除题目运行结果截屏外，还需要有debug截屏。第二题需要有debug部分的截屏，第二题按照调试操作步骤截屏。3.1 结果**



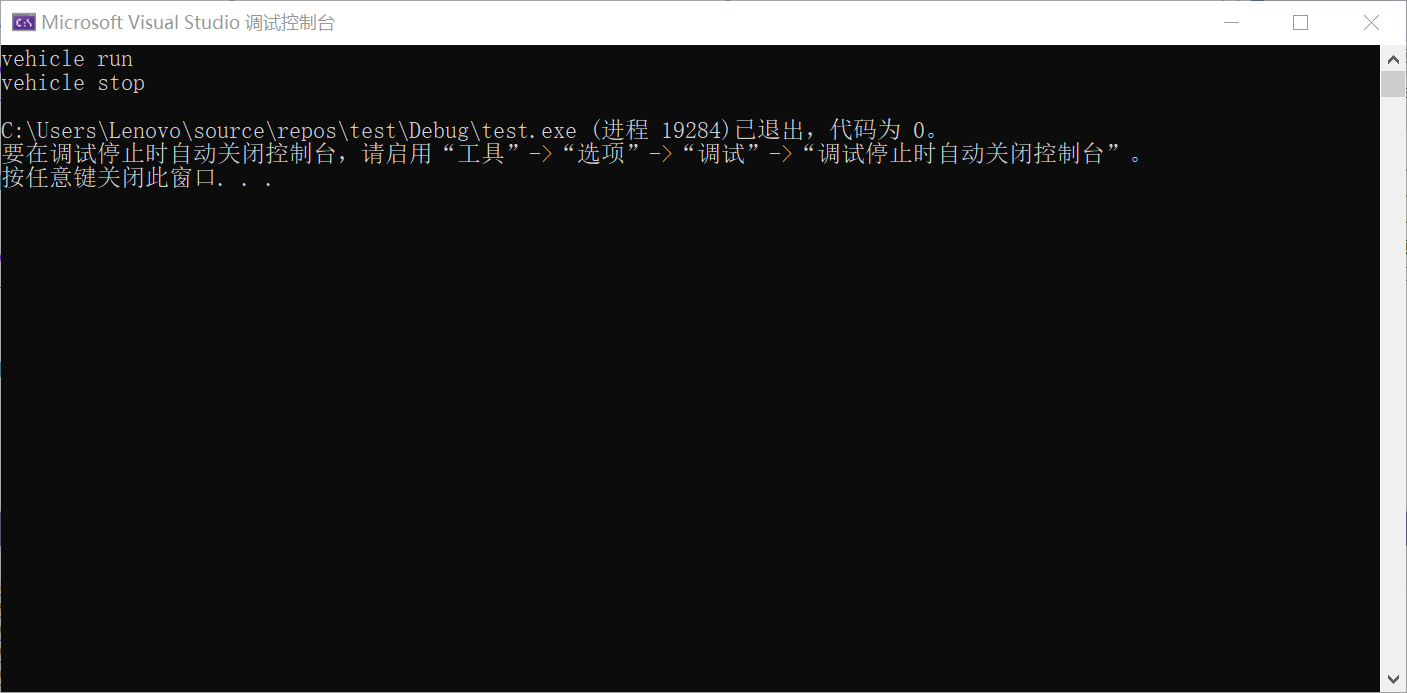
**3.2 结果**



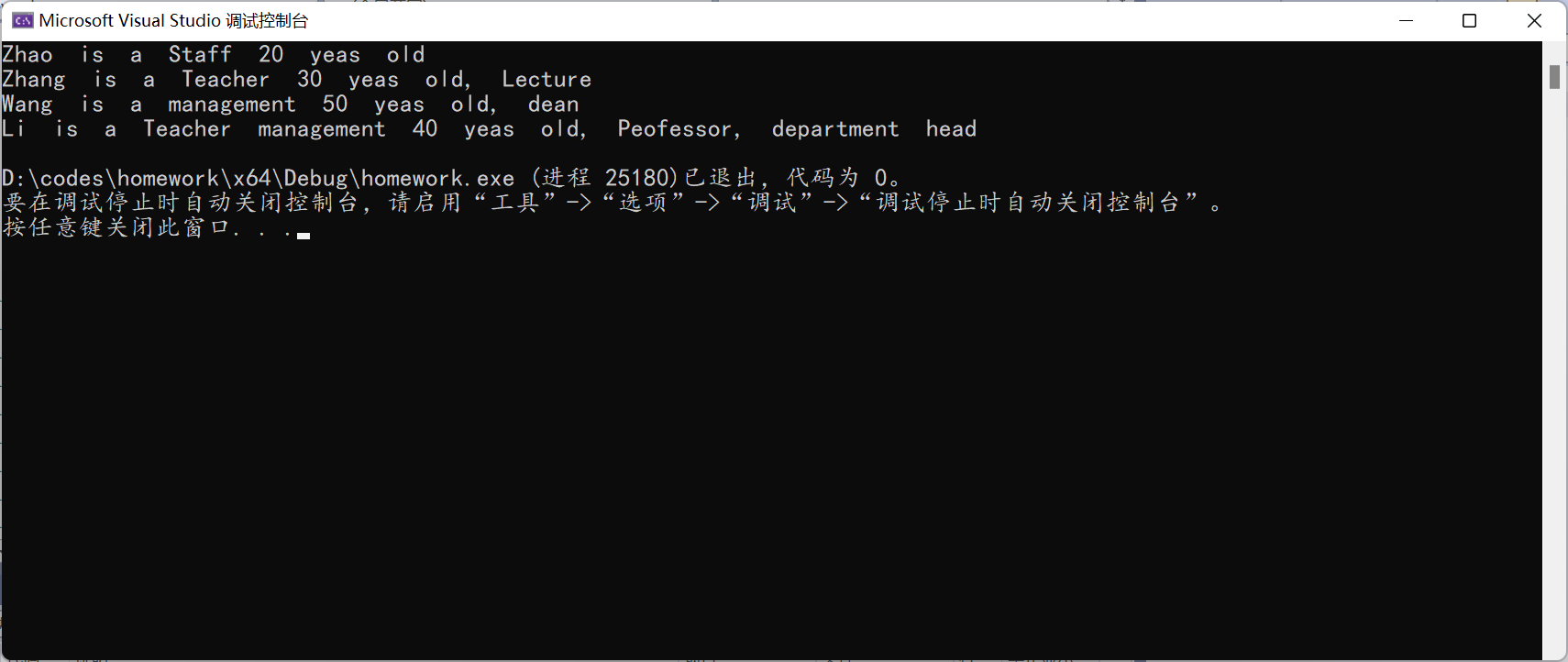
**3.2 debug**



**3.3 结果**



**3.4结果**



1. **心得体会：学到了什么；遇到的问题及解决方法等（可结合思考题）**

**思考题：**

(1)如何在已有的类的基础上设计新的类？

对已用的类进行继承处理，然后再继承的前提之下再声明自己所需要的新的函数和新的数据成员。

(2)基类和派生类对象的构造顺序是怎样的？

先按照继承的顺序构造基类，然后如果派生类还有对象成员，就先构造这些对象，最后再对派生类的其他数据成员进行构造处理。

(3)如何利用虚基类解决二义性问题？

在对底层的基类进行继承的时候，我们在继承的类型前面加上virtual处理，声明此次继承是虚基类。在最远的派生类的继承之中也使用virtual继承基类，这样基类就不会出现重复拷贝继承的问题而出错了。

**心得体会：**1.通过此次实验课，更加熟悉了类的继承关系的知识点，明白了派生类“是”一个基类的用意。2.熟悉不同继承方式（public,private,protected）下对基类成员的访问控制，并更加熟悉了不同继承方式各基类成员属性继承之后的属性。

3.在面对菱形继承的时候，明白了可以用虚基类解决二义性问题。注意到了在继承时，构造函数也要继承。不然会报错没有可用的基类默认构造函数。

4.一开始对于虚基类的知识点并不熟悉，所以最后一题一直卡壳，不过通过查阅书籍，学习虚基类的知识，最后过关。

**实验四**

**1.实验目的**

1.掌握运算符重载的方法2.学习使用虚函数实现动态多态性

**2.实验要求**

1.定义Point类，有坐标\_x，\_y两个成员变量；对Point类重载“＋＋”(自增)、“――”(自减)运算符，实现对坐标值的改变。

2. 定义一个时间Time类，包含三个属性:hour, minute和second

要求通过运算符重载实现如下功能:

时间输入输出(>>、<<)；

时间增加减少若干(+=、-=)

时间前、后自增加/减少1秒(++、--)

3. 定义一个车(vehiele)基类，有Run、Stop等成员函数，由此派生出自行车(bicycle)类、汽车(motorcar)类，从bicycle和motorcar派生出摩托车(motorcycle)类，它们都有Run、Stop等成员函数。观察虚函数的作用。

4. （选做）对实验2中的People类重载“＝＝”运算符和“－”运算符，“＝＝”运算符判断两个people类对象的id属性是否相等；“－”运算符实现People类对象的赋值操作。

**3.实验内容（各题目的题干）**1.编写程序定义Point类，在类中定义整型的私有成员变量\_x、\_y，定义成员函数Point& operator++()；Point operator++(int)；以实现对Point类重载“++”(自增)运算符，定义成员函数Point＆ operator－－()；Point operator－－(int)；以实现对Point类重载“－－”(自减)运算符，实现对坐标值的改变。程序名：1ab4\_1．cpp。2.编写程序定义一个时间Time类，包含三个属性hour, minute和second,通过运算符重载实现如下功能:时间输入输出(>>、<<)；时间增加减少若干(+=、-=) ,例：Time& operator+=(const Time&);Time& operator-=(const Time&)；时间前、后自增加/减少1秒(++、--),前自增例：Time& operator++(); 后自增例：Time operator++(int)；程序中输入固定为两个Time实例(time1，time2),依次输出以下表达式的值time1 += (time2++);time1 -= time2;++time2;time2 += (time1--);--time1;time2 -= time1;程序名：lab4\_2.cpp。

3.编写程序定义一个车(vehicle)基类，有Run、Stop等成员函数，由此派生出自行车(bicycle)类、汽车(motorcar)类，从bicycle和motorcar派生出摩托车(motorcycle)类，它们都有Run、Stop等成员函数。在main()函数中定义vehicle、bicycle、motorcar、motorcycle的对象，调用其Run()、Stop()函数，观察其执行情况。再分别用vehicle类型的指针来调用这几个对象的成员函数，看看能否成功；把Run、Stop定义为虚函数，再试试看。程序名：lab4\_3.cpp。

4. （选做）对实验2中的People类重载“＝＝”运算符和“－”运算符，“＝＝”运算符判断两个people类对象的id属性是否相等；“－”运算符实现People类对象的赋值操作。

**4.源程序（加注释）4.1**

1. #include <iostream>
2. **using** **namespace** std;
4. **class** Point
5. {
6. **private**:
7. **int** \_x;
8. **int** \_y;
9. **public**:
10. // ++point
11. Point& operator++()
12. {
13. \_x++;
14. \_y++;
15. **return** \***this**;
16. }
17. // point++
18. Point operator++(**int** a)
19. {
20. Point tmp = \***this**;
21. \_x++;
22. \_y++;
23. **return** tmp;
24. }
25. // --point
26. Point& operator--()
27. {
28. \_x--;
29. \_y--;
30. **return** \***this**;
31. }
32. //point--
33. Point operator--(**int** a)
34. {
35. Point tmp = \***this**;
36. \_x--;
37. \_y--;
38. **return** tmp;
39. }
40. Point(**int** a, **int** b) :\_x(a), \_y(b)
41. {
43. }
44. **void** show(**const** Point &p)
45. {
46. printf("(%d,%d)\n", p.\_x, p.\_y);
47. }
48. };
50. **int** main()
51. {
52. **int** a, b;
53. cin >> a >> b;
54. Point p(a, b);
56. p.show(p);
57. p.show(p.operator++(10));
58. p.show(p.operator++());
59. p.show(p.operator--(10));
60. p.show(p.operator--());
61. **return** 0;
62. }

**4.2**

1. #include <iostream>
2. #include <cstdlib>
3. #include <cmath>
4. **using** **namespace** std;
6. **class** Time
7. {
8. **friend** ostream& operator<< (ostream& out, Time& T);
9. **friend** istream& operator>> (istream& in, Time& T);
10. **private**:
11. **int** hour;
12. **int** minute;
13. **int** second;
15. **public**:
16. Time(**int** a, **int** b, **int** c)
17. {
18. hour = a;
19. minute = b;
20. second = c;
21. }
22. Time() : hour(0), minute(0), second(0) {}
23. Time& operator++()
24. {
25. second++;
26. **if** (second >= 60) { second = 0; minute++; }
27. **if** (minute >= 60) { minute = 0; hour++; }
28. **if** (hour >= 24) { hour = hour % 24; }
29. **return** \***this**;
30. }
31. Time operator++(**int**)
32. {
33. Time temp = \***this**;
34. second++;
35. **if** (second >= 60) { second = 0; minute++; }
36. **if** (minute >= 60) { minute = 0; hour++; }
37. **if** (hour >= 24) { hour = hour % 24; }
38. **return** temp;
39. }
40. Time& operator--()
41. {
42. second--;
43. **if** (second < 0) { minute--; second = 59; }
44. **if** (minute < 0) { hour--; minute = 59; }
45. **if** (hour < 0) { hour += 24; }
46. **return** \***this**;
47. }
48. Time operator--(**int**)
49. {
50. Time temp = \***this**;
51. second--;
52. **if** (second < 0) { minute--; second = 59; }
53. **if** (minute < 0) { hour--; minute = 59; }
54. **if** (hour < 0) { hour += 24; }
55. **return** temp;
56. }
57. Time& operator+=(**const** Time& T)
58. {
59. **this**->second += T.second;
60. **if** (**this**->second >= 60)
61. {
62. **this**->second -= 60;
63. **this**->minute++;
64. }
65. **this**->minute += T.minute;
66. **if** (**this**->minute >= 60)
67. {
68. **this**->minute -= 60;
69. **this**->hour++;
70. }
71. **this**->hour += T.hour;
72. **if** (**this**->hour >= 24)
73. {
74. **this**->hour = **this**->hour % 24;
75. }
76. **return** \***this**;
77. }
78. Time& operator-=(**const** Time& T)
79. {
80. **if** (**this**->second < T.second)
81. {
82. **this**->minute--;
83. **this**->second += 60;
84. }
85. **this**->second -= T.second;
86. **if** (**this**->minute < T.minute)
87. {
88. **this**->hour--;
89. **this**->minute += 60;
90. }
91. **this**->minute-= T.minute;
92. **if** (**this**->hour < T.hour)
93. {
94. **this**->hour += 24;
95. }
96. **this**->hour -= T.hour;
97. **return** \***this**;
98. }
100. };
102. ostream& operator<< (ostream& out, Time& T)
103. {
104. **if** (T.hour < 10)
105. {
106. out << "0";
107. }
108. out << T.hour << ":";
109. **if** (T.minute < 10)
110. {
111. out << "0";
112. }
113. out << T.minute << ":";
114. **if** (T.second < 10)
115. {
116. out << "0";
117. }
118. out << T.second;
119. **return** out;
120. }
122. istream& operator>> (istream& in, Time& T)
123. {
124. in >> T.hour >> T.minute >> T.second;
125. **return** in;
126. }

129. **int** main()
130. {
131. Time t1;
132. Time t2;
133. cin >> t1;
134. cin >> t2;
135. t1 += (t2++);
136. cout << t1 << endl;
137. t1 -= t2;
138. cout << t1 << endl;
139. ++t2;
140. cout << t2 << endl;
141. t2 += (t1--);
142. cout << t2 << endl;
143. --t1;
144. cout << t1 << endl;
145. t2 -= t1;
146. cout << t2;
147. **return** 0;
148. }

**4.3**

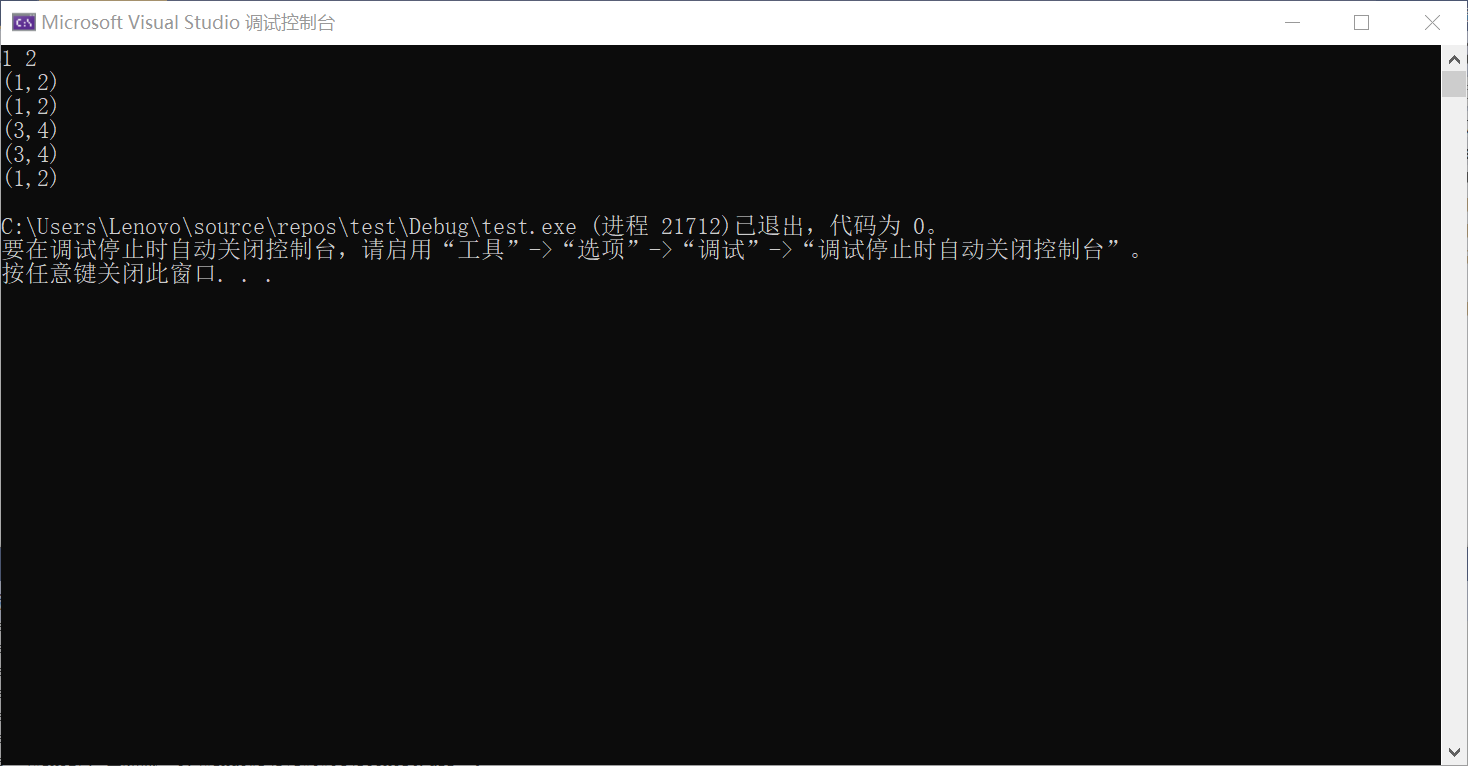
1. #include <iostream>
2. **using** **namespace** std;
4. **class** vehicle
5. {
6. **public**:
7. **virtual** **void** run()
8. {
9. cout << "vehicle run!" << endl;
10. }
11. **virtual** **void** stop()
12. {
13. cout << "vehicle stop!" << endl;
14. }
15. };
17. **class** bicycle :**virtual** **public** vehicle
18. {
19. **public**:
20. **void** run()
21. {
22. cout << "bicycle run!" << endl;
23. }
24. **void** stop()
25. {
26. cout << "bicycle stop!" << endl;
27. }
28. };
30. **class** motorcar :**virtual** **public** vehicle
31. {
32. **public**:
33. **void** run()
34. {
35. cout << "motorcar run!" << endl;
36. }
37. **void** stop()
38. {
39. cout << "motorcar stop!" << endl;
40. }
41. };
43. **class** motorbicycle :**public** bicycle, **public** motorcar
44. {
45. **public**:
46. **void** run()
47. {
48. cout << "motorbicycle run!" << endl;
49. }
50. **void** stop()
51. {
52. cout << "motorbicycle stop!" << endl;
53. }
54. };
56. **int** main()
57. {
58. vehicle v;
59. bicycle b;
60. motorcar mc;
61. motorbicycle mb;
63. v.run();
64. b.run();
65. mc.run();
66. mb.run();
68. vehicle\* pv = &v;
69. pv->run();
71. pv = &b;
72. pv->run();
74. pv = &mc;
75. pv->run();
77. pv = &mb;
78. pv->run();
79. **return** 0;
80. }

**4.4 选做题：**

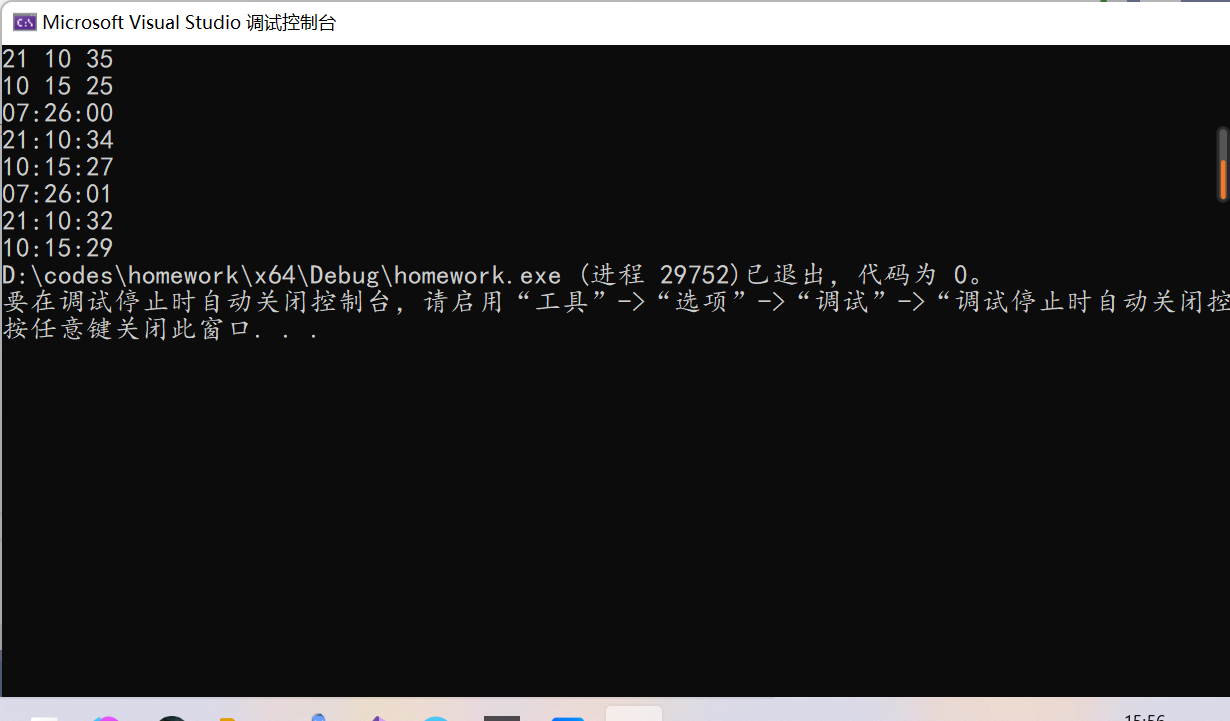
1. #include<iostream>
2. #include<math.h>
3. #include<string>
4. #include<iomanip>
5. #include<cstring>
6. **using** **namespace** std;
7. **class** people
8. {
9. **public**:
10. people()
11. {
12. cout<<"默认构造函数"<<endl;
13. }
14. people(**char** n[],**char** a[],**char** b[],string d,**char** c[])
15. {
16. strcpy(name,n);
17. strcpy(number,a);
18. strcpy(sex,b);
19. strcpy(id,c);
20. birthday=d;
21. cout<<"构造了此函数"<<endl;
22. }
23. people(people&a)
24. {
25. strcpy(name,a.name);
26. strcpy(number,a.number);
27. strcpy(sex,a.sex);
28. strcpy(id,a.id);
29. birthday=a.birthday;
30. cout<<"拷贝此类型的对象"<<endl;
31. }
32. ~people()
33. {
34. cout<<"析构此类型对象"<<endl;
35. }
36. **void** show()
37. {
38. cout<<"姓名是："<<name;
39. cout<<"编号是："<<number<<endl<<"性别是："<<sex<<endl;
40. cout<<"身份证号是："<<id<<endl<<"出生日期是："<<birthday<<endl;
41. }
42. people operator-(people&a)//对于赋值的操作
43. {
44. **if**(**this**==&a)
45. **return** \***this**;
46. **else**
47. {
48. strcpy(name,a.name);
49. strcpy(number,a.number);
50. strcpy(sex,a.sex);
51. strcpy(id,a.id);
52. birthday=a.birthday;
53. }
54. **return** \***this**;
55. }
56. **bool** operator==(people&a)//对于等于的操作的函数重载
57. {
58. **int** x=0;
59. **if**(**this**==&a)
60. **return** **true**;
61. **else**
62. {
63. **if**(strcmp(name,a.name)==0)
64. x++;
65. **if**(strcmp(number,a.number)==0)
66. x++;
67. **if**(strcmp(sex,a.sex)==0)
68. x++;
69. **if**(strcmp(id,a.id)==0)
70. x++;
71. **if**(birthday==a.birthday)
72. x++;
73. }
74. **if**(x==5)
75. **return** **true**;
76. **else**
77. **return** **false**;
78. }
79. **private**:
80. **char** name[11];
81. **char** number[7];
82. **char** sex[3];
83. string birthday;
84. **char** id[16];
85. };
86. **int** main ()
87. {
88. **char** n[11];
89. **char** a[7];
90. **char** b[3];
91. string d;
92. **char** c[16];
93. cin>>n>>a>>b>>c>>d;
94. people people1;
95. people people2(n,a,b,d,c);
96. people2.show();
97. people people3(people2);
98. people3.show();
99. people1-people2;
100. **if**(people1==people2)
101. cout<<"他们相等"<<endl;
102. **if**(people3==people2)
103. cout<<"他们相等"<<endl;
104. }

**5.运行结果（贴图）**

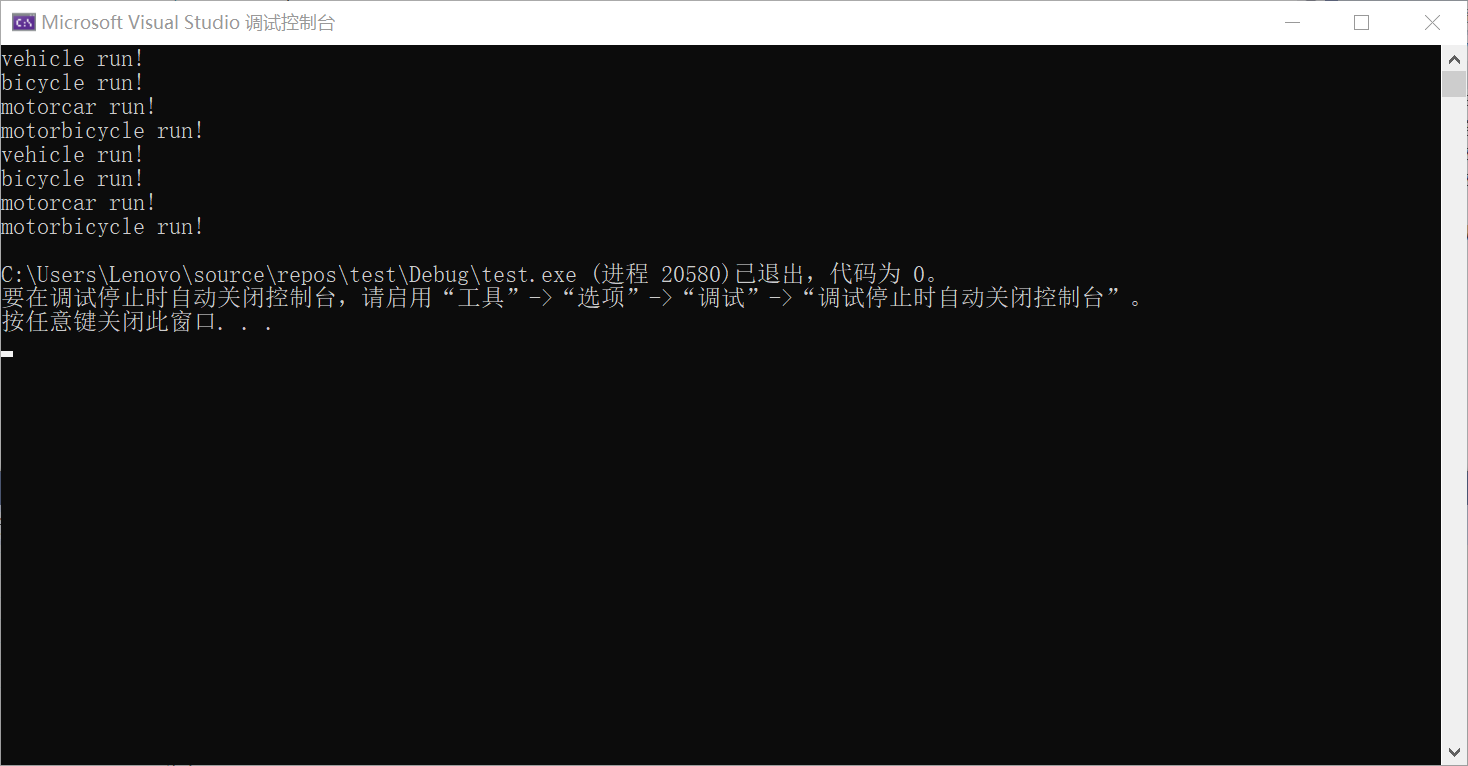
**4.1 结果**



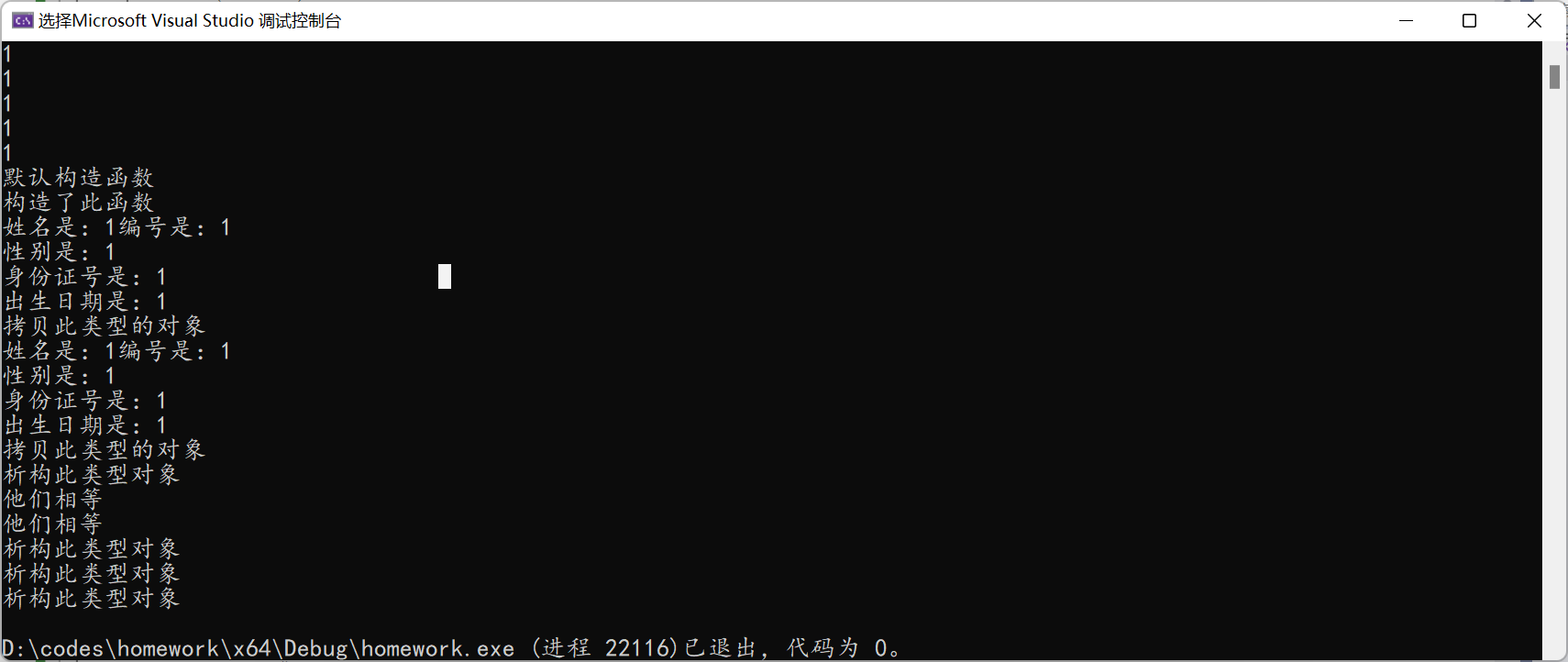
**4.2 结果**



**4.3 结果**



**4.4 结果（选做）：**



1. **心得体会：学到了什么；遇到的问题及解决方法等（可结合思考题）**

**思考题：**

1.如何将一个运算符重载为类的成员函数？

在类的内部声明定义重载运算符的函数，确定返回的类型，运算符重载进行的操作。如果返回的是类的类型，类型名+operator+你用重载的运算符（）{内部操作}

2.如何将一个运算符重载为类的友元函数？

首先这个函数要定义在类的内部，在函数前面定义为friend处理就可以了。Friend 类型名+operator+你用重载的运算符（）{内部操作}，但在这个时候要注意参数的个数处理问题。

3.如何实现运行时刻的多态？

在基类与派生类中定义函数原型相同的两个虚函数，然后定义基 类的指针，用基类的指针指向不同的派生类对象，通过虚函数即 可实现运行时多态。

**心得体会：**

1.通过本次实验课，对于运算符重载有了更加深层次的理解。

2.对于自增自减运算符的重载从理论到实践，更加深刻的体会了：在成员函数内重载自增自减运算符，前置不需要参数，而后置需要一个虚参数。

3. 在面对菱形继承的时候，学习使用虚函数实现动态多态性。

4. 更加深刻熟悉了虚函数、虚继承的语法和关系。