|  |  |
| --- | --- |
| **编号:** | **第1章第1次** |

****

信息科学与工程学院实验报告

《面向对象程序设计》

**Object-Oriented Programming**

|  |  |
| --- | --- |
| 姓名： | 颜丙超 |
| 学号： | 202311000415 |
| 班级： | 计工本2301 |
| 教师: | 张庆科 |
| 时间： | 2024年10月18日 |

**《面向对象程序设计》实验报告**

**报告要求：**实验报告包含实验目的、实验内容、实验过程（详细操作流程）、实验结果（程序运行结果高清截图）、实验分析总结五个部分。报告中若涉及代码程序，请在附录部分提供完整程序源码及源码托管地址(基于Highlight软件导入源码)。报告撰写完毕后请将PDF格式版本上传到坚果云作业提交系统。

1. **实验目的**

掌握指向常量指针的用法

掌握C++重载函数的基本用法

掌握C++内联函数的基本用法

掌握C++带有默认形参值函数的用法

**二、实验内容**

1. 实验一之中，需要定义三角形的三条边，利用函数进行面积计算
2. 实验二需要对重载函数的特点和用法进行分析，求出最大公约数
3. 实验三需要将将其中一个设置为内联函数。要求在主函数内调用各函数10000次，计算比较两种函数各自的执行时间，基于实验结果分析内联函数和普通函数的区别

4.建立VS项目，编写一个C++程序，定义函数calculateAre(...)，用于计算不同形状的面积。 函数接受两个参数：形状类型（std::string，默认值为"circle"）和相关的尺寸（double，默认值为1.0）。根据形状类型，计算并返回对应的面积。

**三、实验过程**

1.实验一定义triangleArea函数：函数接受一个const double\*类型的参数，表示指向double类型常量的指这确保了通过该指针不能修改它所指向的数组内容。使用指针解引用来获取数组中的三个元素，即三角形的三条边长。使用海伦公式计算三角形的面积。返回计算得到的面积。在main函数中：定义一个包含三条边长的double类型数组sides。

提示用户输入三条边长，并使用循环从标准输入读取这些值。调用triangleArea函数，并将结果存储在area变量中。输出计算得到的三角形面积。

1. 实验二

头文件：

int gcd(int a, int b);：计算两个整数的最大公约数，使用欧几里得算法。

int commonDivisor(int x1, int x2, int x3);：计算三个整数的最大公约数，通过递归调用gcd函数。

int commonDivisor(int x1, int x2, int x3, int x4);：计算四个整数的最大公约数，通过递归调用commonDivisor(int, int, int)和gcd函数。

主函数：

int main();：程序的入口点，负责读取用户输入并调用commonDivisor函数计算最大公约数，然后输出结果。

1. 实验三

内联函数和普通函数之间的性能差异可能非常小或无法测量。然而，在实际应用中，内联函数可能会在某些情况下提供性能优势，这取决于编译器的优化能力、函数体的复杂性和调用频率。因此，在决定是否使用内联函数时，应该综合考虑这些因素。

1. 实验四

double calculateArea(const std::string& shapeType, double a, double b = 1.0, double c = 1.0);：这是一个函数声明，但由于后面紧接着给出了一个同名但参数不同的函数定义，这个声明实际上是多余的，且可能会引起编译器的警告或错误（函数重载时，同名函数的参数列表必须不同）。

double calculateArea(const std::string& shapeType, double a, double b, double c)：这是函数的实际定义。它根据传入的图形类型（shapeType）和相应的边长（a, b, c），计算并返回图形的面积。然而，这个函数的参数设计并不理想，因为它为所有图形类型使用了相同的参数列表，这导致了对于圆形和矩形来说，c参数是多余的。

主函数（main）

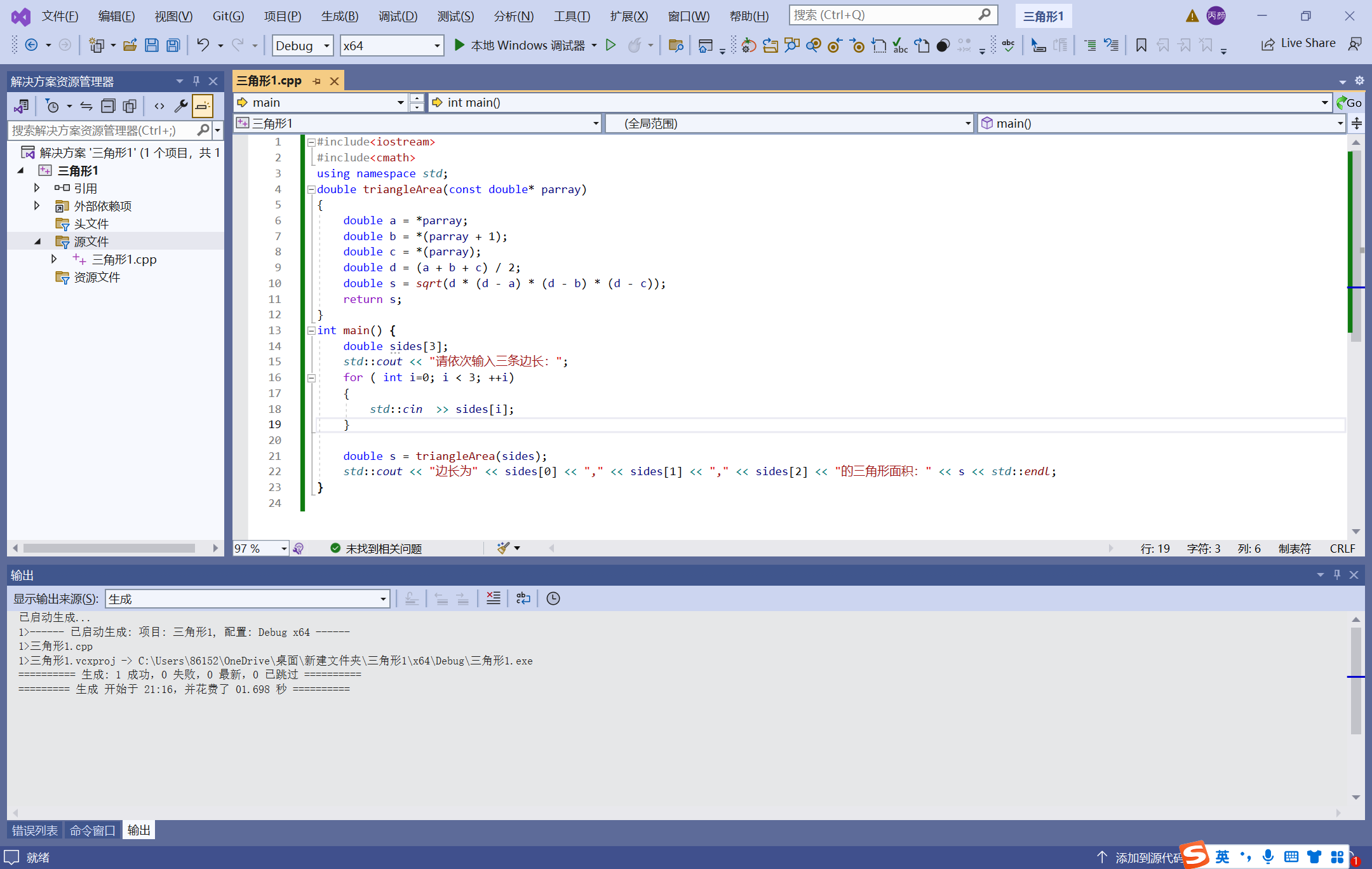
定义了字符串shape和双精度浮点数a, b, c，其中b和c被初始化为2.0。这是不理想的，因为初始值应该根据用户的输入来确定，特别是对于矩形和三角形。

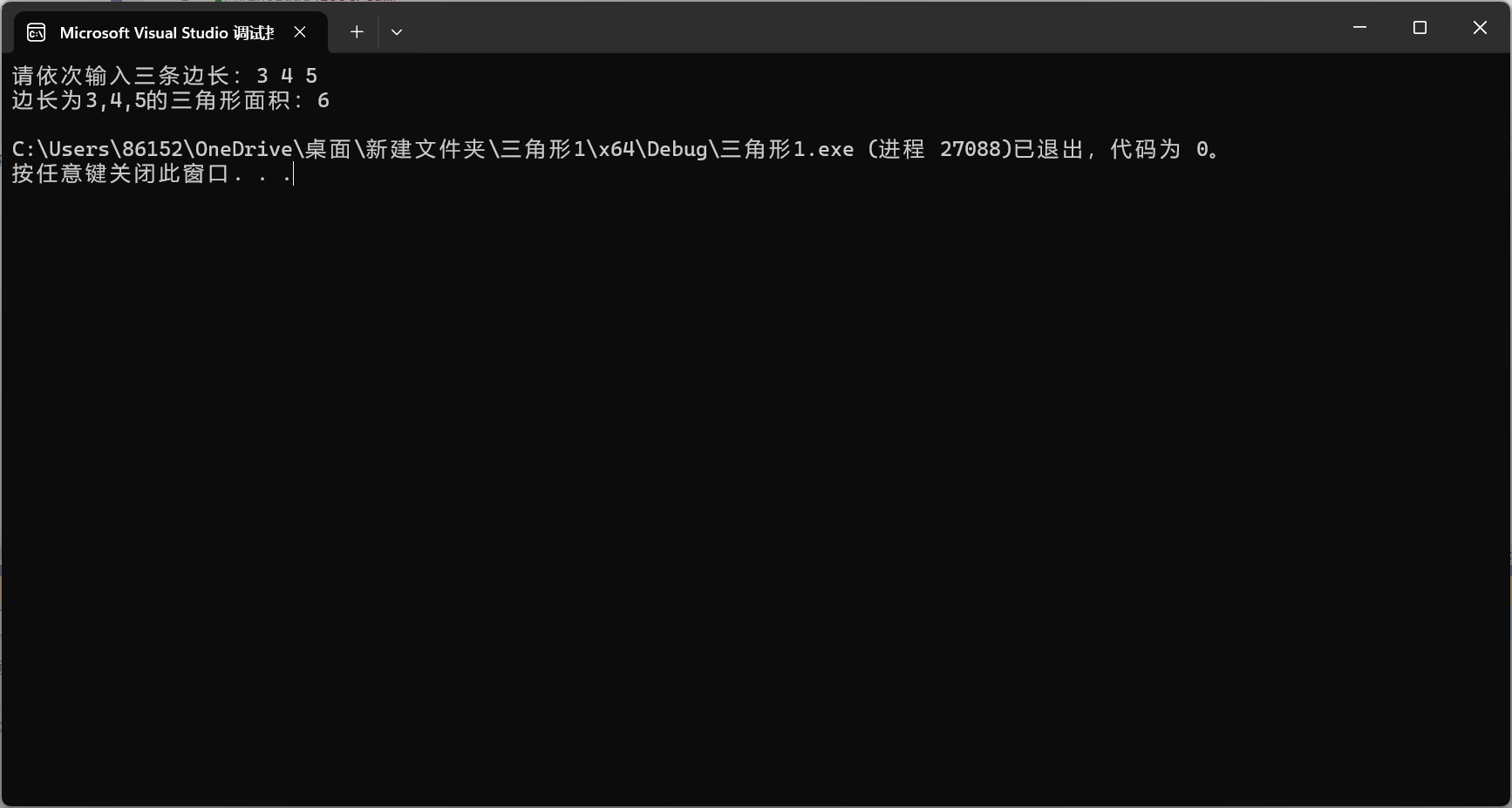
通过std::cin读取用户选择的图形类型（通过choice变量），然后根据选择设置shape字符串，并提示用户输入相应的边长。

使用switch语句根据用户的选择调用calculateArea函数，并输出计算结果。

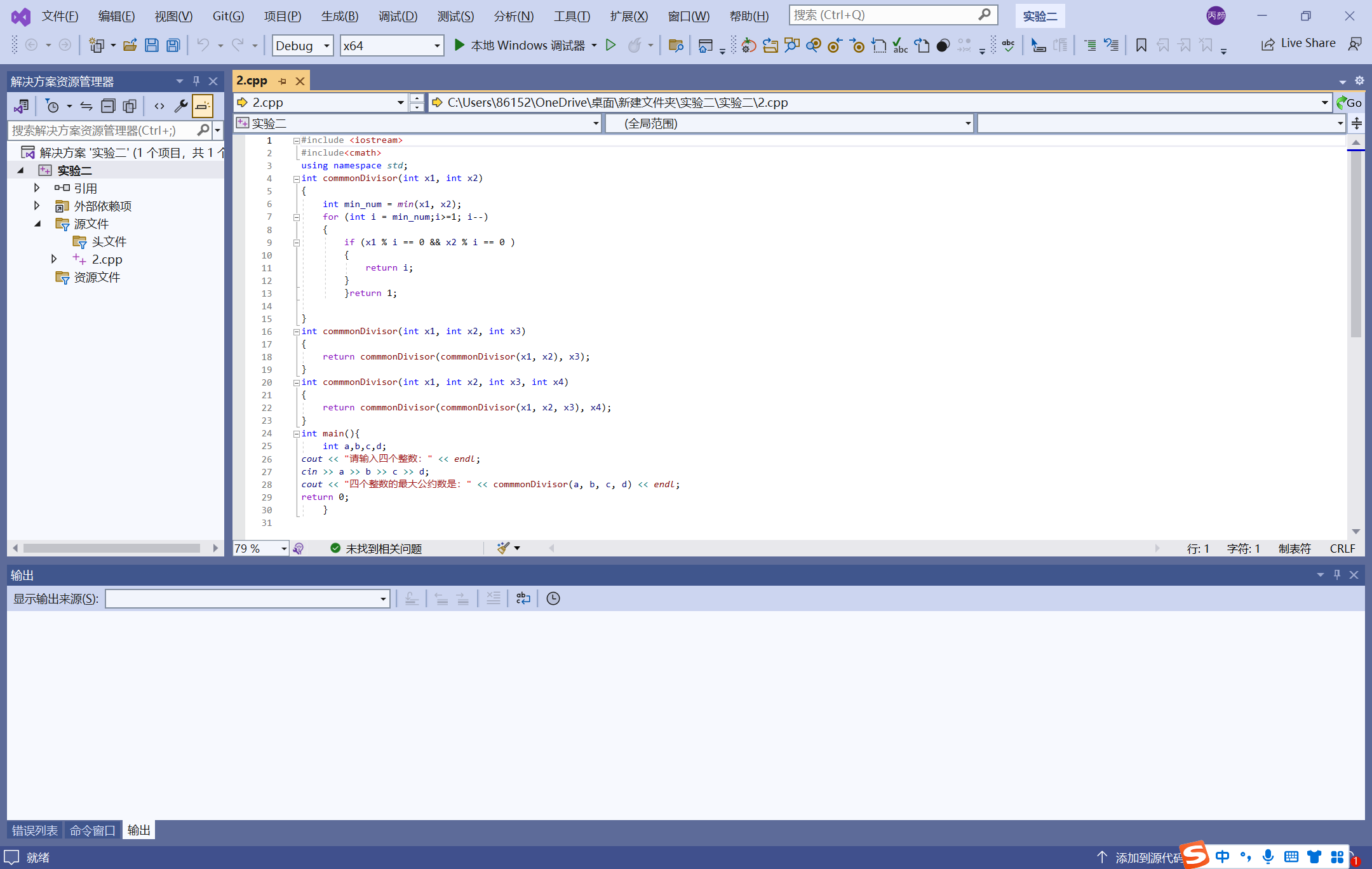
**四、实验结果**

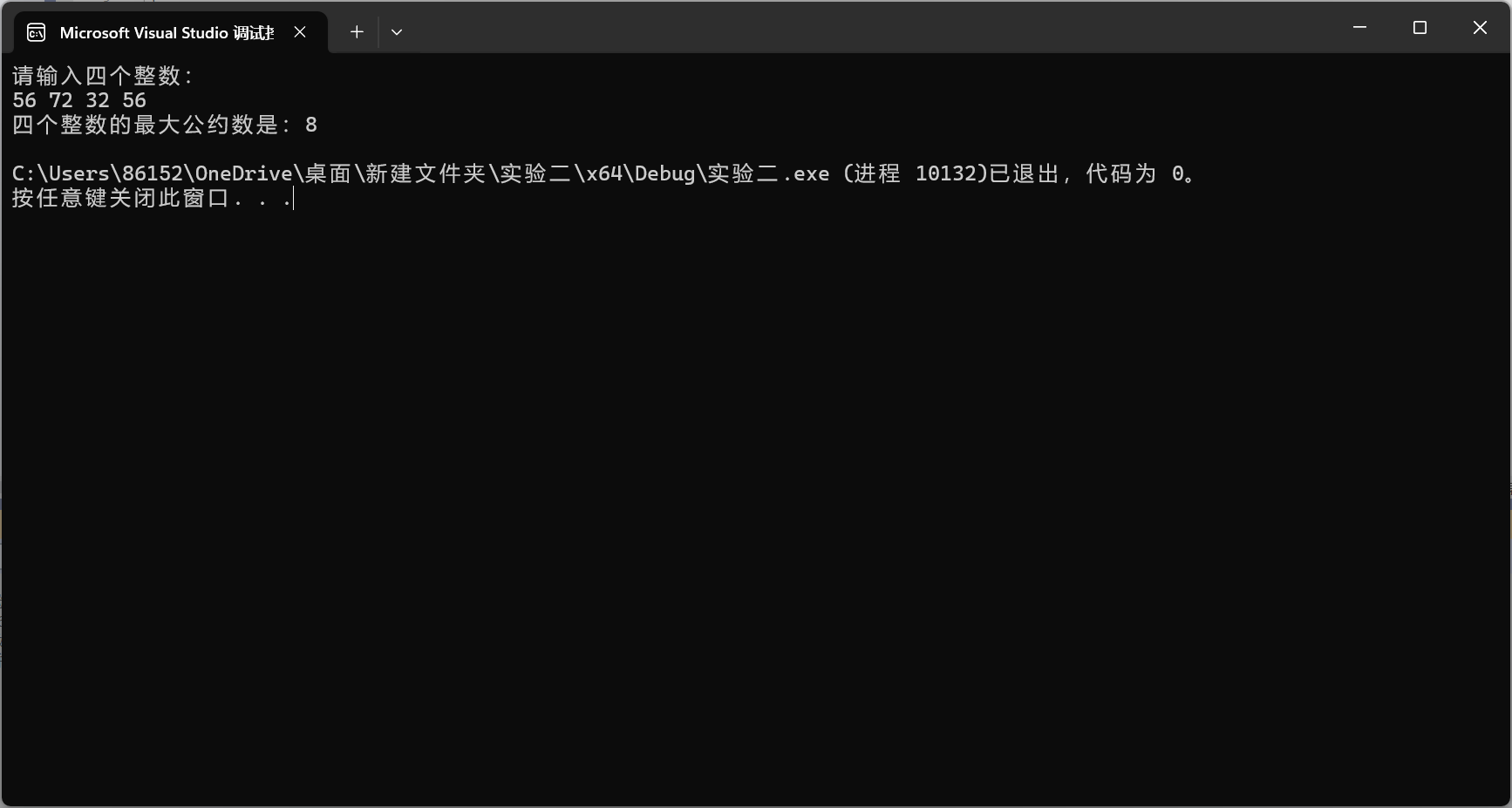
实验一



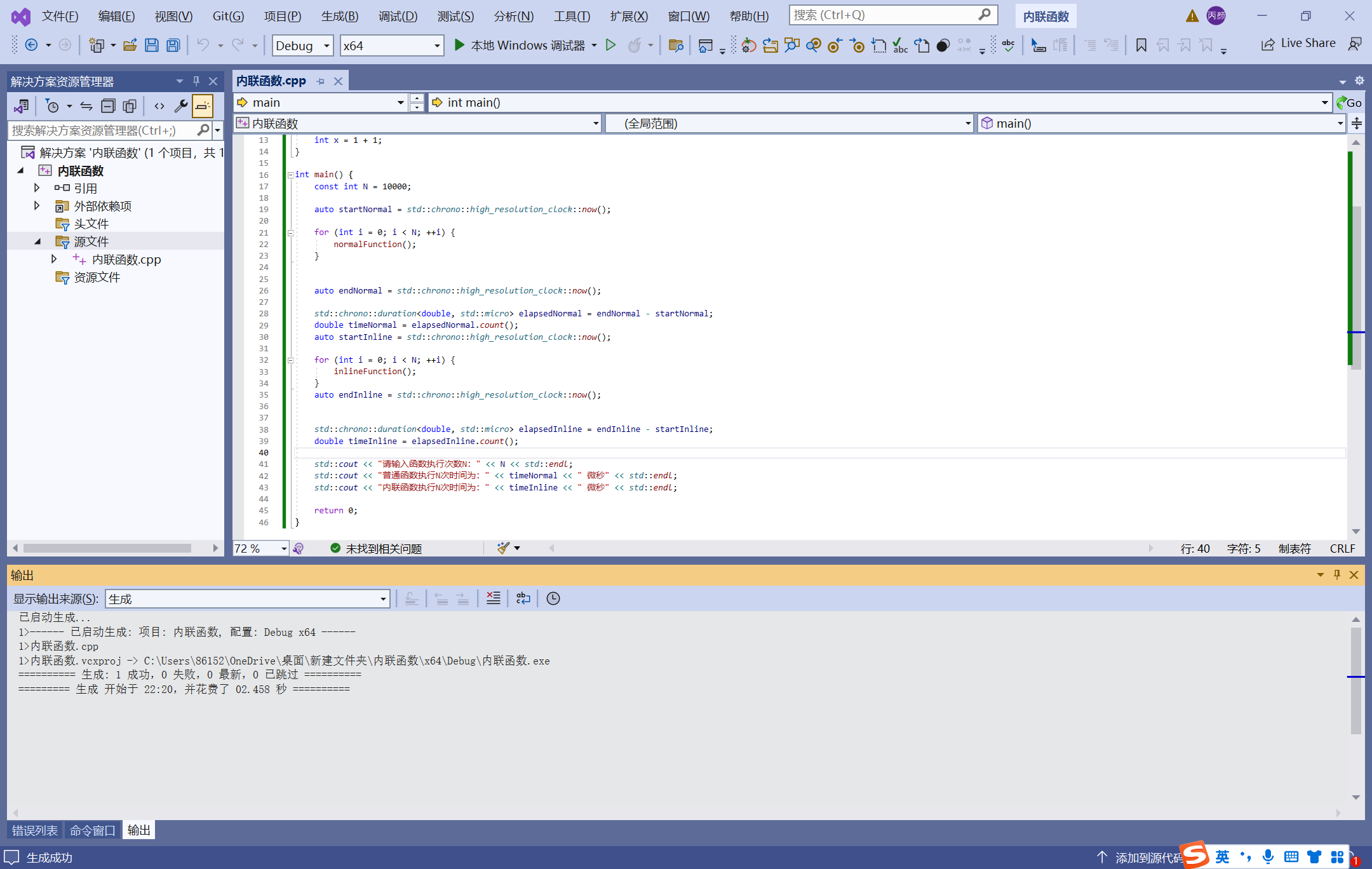


实验二



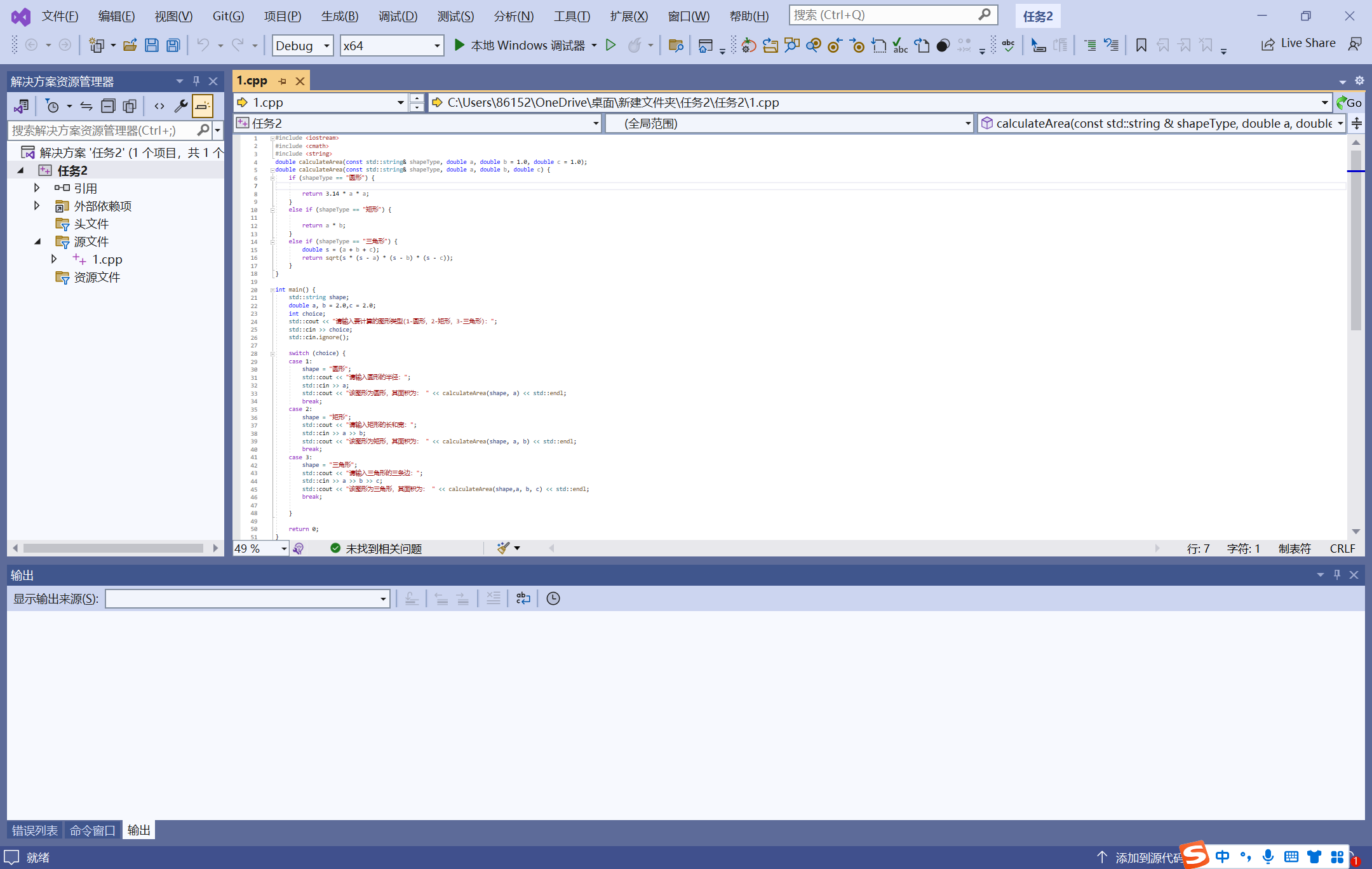


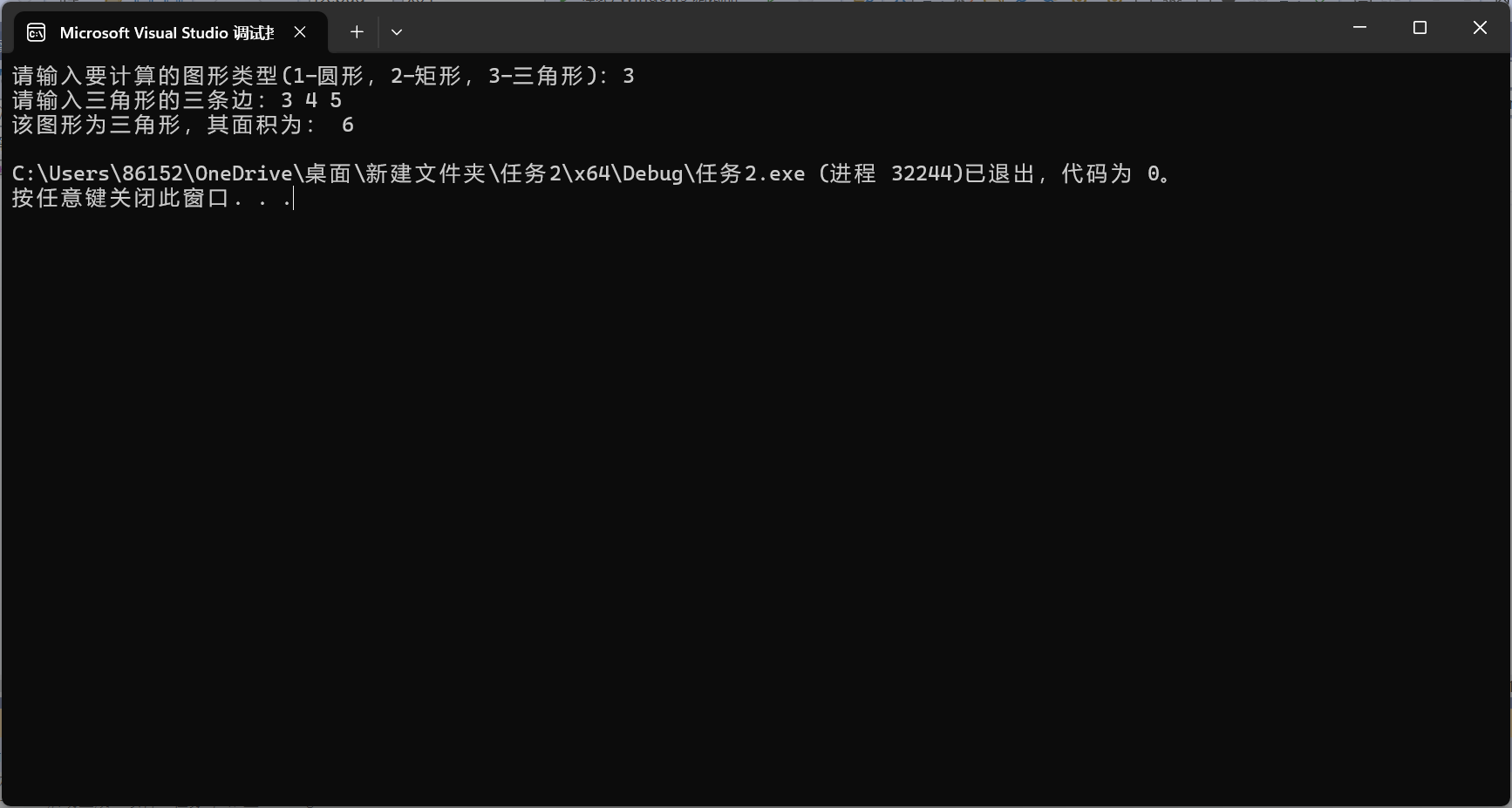
实验三

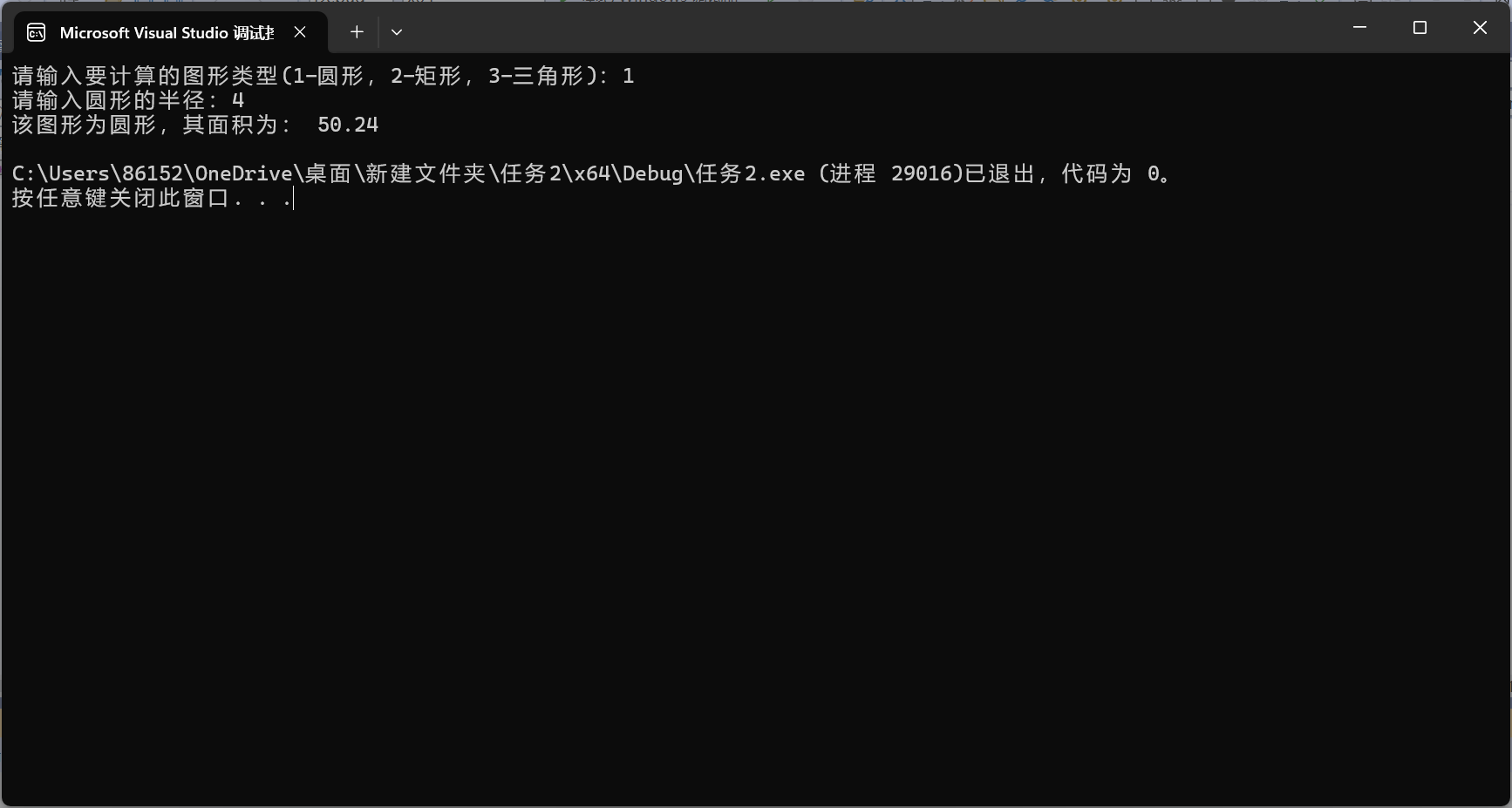




实验四







**五、实验总结**

实验一代码

01 **#include<iostream>**

02 **#include<cmath>**

03 **using namespace** std**;**

04 **double** triangleArea**(const double\*** parray**)**

05 **{**

06 **double** a **= \***parray**;**

07 **double** b **= \*(**parray **+** **1);**

08 **double** c **= \*(**parray**+** **2);**

09 **double** d **= (**a **+** b **+** c**) /** **2;**

10 **double** s **=** sqrt**(**d **\* (**d **-** a**) \* (**d **-** b**) \* (**d **-** c**));**

11 **return** s**;**

12 **}**

13 **int** main**()**

14 **{**

15 **double** sides**[3];**

16 std**::**cout **<<** "���������������߳���"**;**

17 **for** **(** **int** i**=0;** i **<** **3; ++**i**)**

18 **{**

19 std**::**cin **>>** sides**[**i**];**

20 **}**

21

22 **double** s **=** triangleArea**(**sides**);**

23 std**::**cout **<<** "�߳�Ϊ" **<<** sides**[0] <<** "," **<<** sides**[1] <<** "," **<<** sides**[2] <<** "��������������" **<<** s **<<** std**::**endl**;**

24 **}**

25

实验二代码

01 **#include <iostream>**

02 **#include<cmath>**

03 **using namespace** std**;**

04 **int** commmonDivisor**(int** x1**,** **int** x2**)**

05 **{**

06 **int** min\_num **=** min**(**x1**,** x2**);**

07 **for** **(int** i **=** min\_num**;** i**>=1;** i**--)**

08 **{**

09 **if** **(**x1 **%** i **==** **0** **&&** x2 **%** i **==** **0** **)**

10 **{**

11 **return** i**;**

12 **}**

13 **}**

14 **return** **1;**

15

16 **}**

17 **int** commmonDivisor**(int** x1**,** **int** x2**,** **int** x3**)**

18 **{**

19 **return** commmonDivisor**(**commmonDivisor**(**x1**,** x2**),** x3**);**

20 **}**

21 **int** commmonDivisor**(int** x1**,** **int** x2**,** **int** x3**,** **int** x4**)**

22 **{**

23 **return** commmonDivisor**(**commmonDivisor**(**x1**,** x2**,** x3**),** x4**);**

24 **}**

25 **int** main**()**

26 **{**

27 **int** a**,**b**,**c**,**d**;**

28 cout **<<** "�������ĸ�������" **<<** endl**;**

29 cin **>>** a **>>** b **>>** c **>>** d**;**

30 cout **<<** "�ĸ�������������Լ���ǣ�" **<<** commmonDivisor**(**a**,** b**,** c**,** d**) <<** endl**;**

31 **return** **0;**

32 **}**

33

实验三代码

01 **#include<iostream>**

02 **#include<cmath>**

03 **#include <chrono>**

04 **using namespace** std**;**

05 **#include <chrono>**

06

07 **void** normalFunction**()**

08 **{**

09

10 **int** x **=** **1** **+** **1;**

11 **}**

12 inline **void** inlineFunction**()**

13 **{**

14

15 **int** x **=** **1** **+** **1;**

16 **}**

17

18 **int** main**()**

19 **{**

20 **const int** N **=** **10000;**

21

22 auto startNormal **=** std**::**chrono**::**high\_resolution\_clock**::**now**();**

23

24 **for** **(int** i **=** **0;** i **<** N**; ++**i**)**

25 **{**

26 normalFunction**();**

27 **}**

28

29

30 auto endNormal **=** std**::**chrono**::**high\_resolution\_clock**::**now**();**

31

32 std**::**chrono**::**duration**<double,** std**::**micro**>** elapsedNormal **=** endNormal **-** startNormal**;**

33 **double** timeNormal **=** elapsedNormal**.**count**();**

34 auto startInline **=** std**::**chrono**::**high\_resolution\_clock**::**now**();**

35

36 **for** **(int** i **=** **0;** i **<** N**; ++**i**)**

37 **{**

38 inlineFunction**();**

39 **}**

40 auto endInline **=** std**::**chrono**::**high\_resolution\_clock**::**now**();**

41

42

43 std**::**chrono**::**duration**<double,** std**::**micro**>** elapsedInline **=** endInline **-** startInline**;**

44 **double** timeInline **=** elapsedInline**.**count**();**

45

46 std**::**cout **<<** "�����뺯��ִ�д���N��" **<<** N **<<** std**::**endl**;**

47 std**::**cout **<<** "��ͨ����ִ��N��ʱ��Ϊ��" **<<** timeNormal **<<** " ΢��" **<<** std**::**endl**;**

48 std**::**cout **<<** "��������ִ��N��ʱ��Ϊ��" **<<** timeInline **<<** " ΢��" **<<** std**::**endl**;**

49

50 **return** **0;**

51 **}**

实验四代码

01 **#include <iostream>**

02 **#include <cmath>**

03 **#include <string>**

04 **double** calculateArea**(const** std**::**string**&** shapeType**,** **double** a**,** **double** b **=** **1.0,** **double** c **=** **1.0);**

05 **double** calculateArea**(const** std**::**string**&** shapeType**,** **double** a**,** **double** b**,** **double** c**)**

06 **{**

07 **if** **(**shapeType **==** "圆形"**)**

08 **{**

09

10 **return** **3.14** **\*** a **\*** a**;**

11 **}**

12 **else if** **(**shapeType **==** "矩形"**)**

13 **{**

14

15 **return** a **\*** b**;**

16 **}**

17 **else if** **(**shapeType **==** "三角形"**)**

18 **{**

19 **double** s **= (**a **+** b **+** c**)/2;**

20 **return** sqrt**(**s **\* (**s **-** a**) \* (**s **-** b**) \* (**s **-** c**));**

21 **}**

22 **}**

23

24 **int** main**()**

25 **{**

26 std**::**string shape**;**

27 **double** a**,** b **=** **2.0,**c **=** **2.0;**

28 **int** choice**;**

29 std**::**cout **<<** "请输入要计算的图形类型(1-圆形，2-矩形，3-三角形)："**;**

30 std**::**cin **>>** choice**;**

31 std**::**cin**.**ignore**();**

32

33 **switch** **(**choice**)**

34 **{**

35 **case** **1:**

36 shape **=** "圆形"**;**

37 std**::**cout **<<** "请输入圆形的半径："**;**

38 std**::**cin **>>** a**;**

39 std**::**cout **<<** "该图形为圆形，其面积为： " **<<** calculateArea**(**shape**,** a**) <<** std**::**endl**;**

40 **break;**

41 **case** **2:**

42 shape **=** "矩形"**;**

43 std**::**cout **<<** "请输入矩形的长和宽："**;**

44 std**::**cin **>>** a **>>** b**;**

45 std**::**cout **<<** "该图形为矩形，其面积为： " **<<** calculateArea**(**shape**,** a**,** b**) <<** std**::**endl**;**

46 **break;**

47 **case** **3:**

48 shape **=** "三角形"**;**

49 std**::**cout **<<** "请输入三角形的三条边："**;**

50 std**::**cin **>>** a **>>** b **>>** c**;**

51 std**::**cout **<<** "该图形为三角形，其面积为： " **<<** calculateArea**(**shape**,**a**,** b**,** c**) <<** std**::**endl**;**

52 **break;**

53

54 **}**

55

56 **return** **0;**

57 **}**