
程序设计基础及语言 A

实验指导手册

东南大学

计算机科学与工程学院

软件学院

人工智能学院

网络空间安全学院

2019 年 9 月

目 录

第一部分 实验环境的使用方法.....	1
1.1 目的.....	1
1.2 Visual C++2010 Express Edition Manual.....	1
1.2.1 First Run.....	1
1.2.2 New project.....	2
1.2.3 Add new source file.....	4
1.2.4 Run Program in Dos Prompt.....	8
1.2.5 Run program in IDE.....	10
1.2.6 用 Ctrl+F5 运行	14
1.2.7 First Debug.....	15
第二部分 实验作业.....	22
Lab1 Introduction to C++ Programming.....	22
Lab2 Introduction to Classes and Objects.....	27
Lab3 Control Statements: Part I.....	29
Lab4 Control Statements II	31
Lab5 Function and Recursion I.....	36
Lab6 Function and Recursion II.....	38
Lab7 Array	错误！未定义书签。
Lab8 Pointers and Pointer-based Strings.....	47

第一部分 实验环境的使用方法

Visual C++ 2010（简称 VC++）是 Microsoft 公司开发的基于 C/C++ 的集成开发工具，它是 Visual Studio 中功能强大、代码效率最高的开发工具。

利用 Visual C++ 可以采用两种方式编写 Win32 应用程序：

第一种：基于 Windows API 的编程方式（又称为控制台编程方式）。

第二种：基于 MFC 的 C++ 编程方式。

本节介绍 VC++ 语言的最基本的编程环境和控制台编程方式，通过简单实例介绍 VC++ 的基本使用方法，使学生尽快学会和掌握如何在 VC++ 环境下，采用控制台编程方式进行课程实验。

1.1 目的

1. 了解 Visual C++ 2010 的特点。
2. 熟悉 Visual C++ 2010 的控制台开发环境。
3. 学习和掌握 Visual C++ 2010 的编写控制台程序的方法

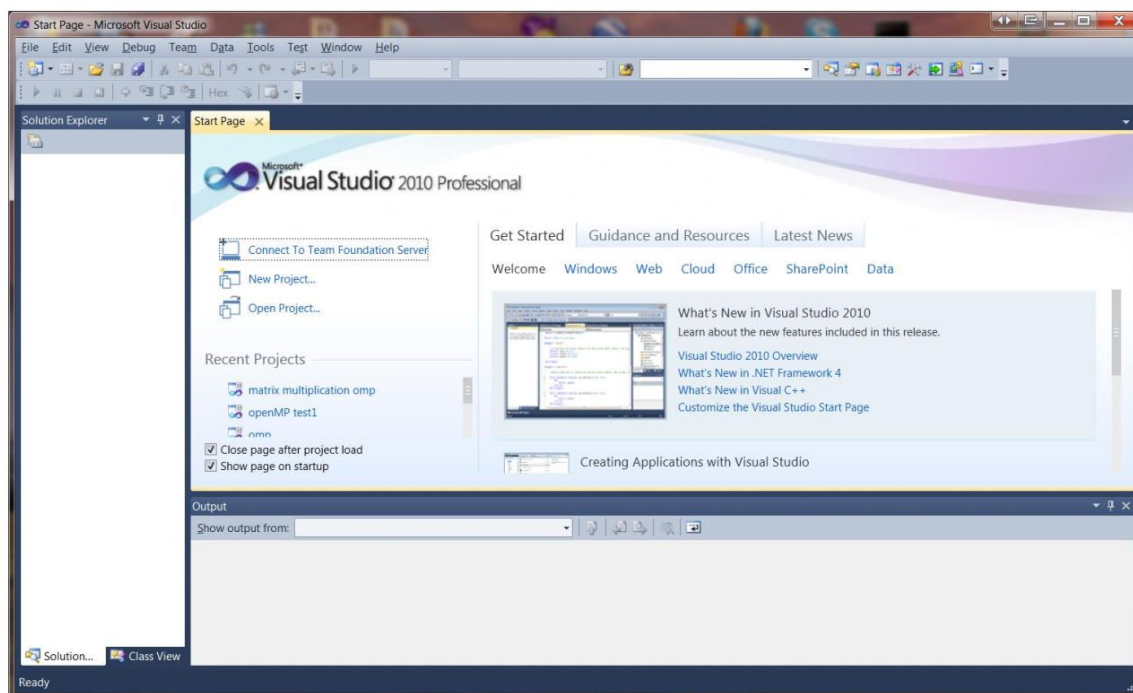
1.2 Visual C++ 2010 Express Edition Manual

Created with a trial version of ScreenSteps

1.2.1 First Run



点击 windows 左下角，开始 - 程序，找到新安装的 Microsoft Visual Studio 2010。Visual Studio 2010 的主窗口如下所示：

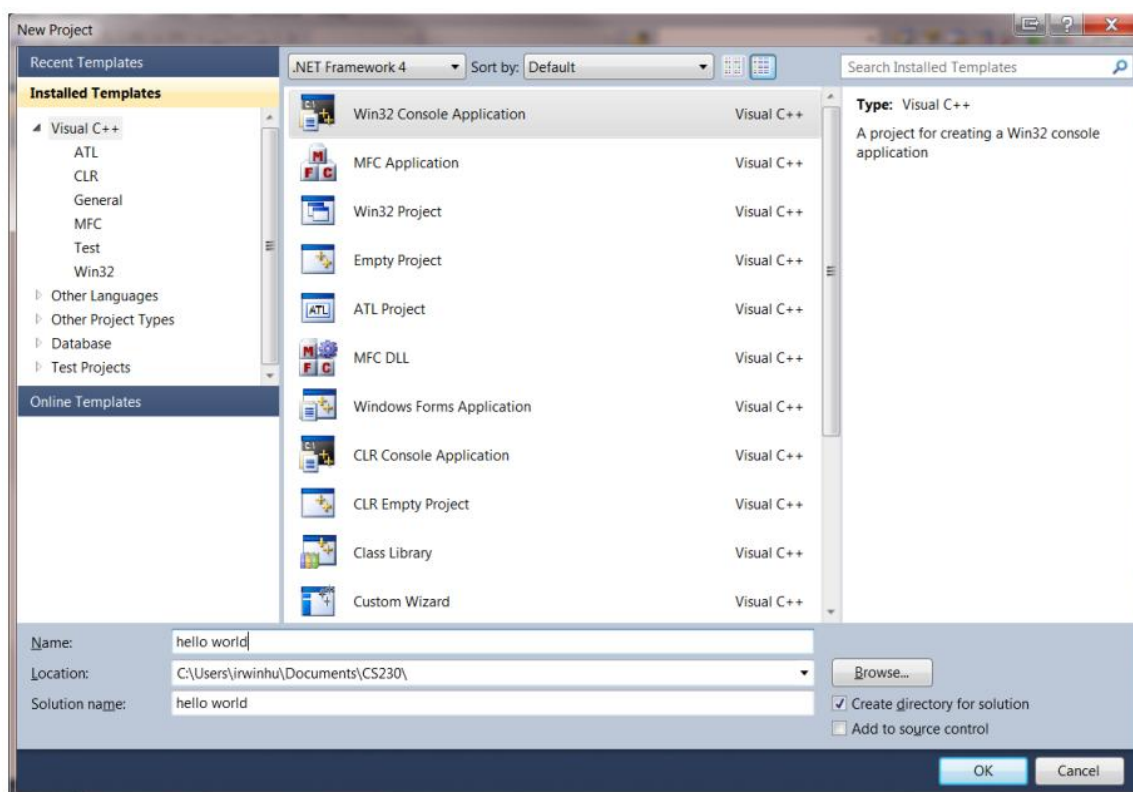


注意：

如果没有显示左边的“解决方案资源管理器”窗口，请单击菜单栏中的“视图”，然后单击“解决方案资源管理器”来显示它。

1.2.2 New project

在菜单中，点击 File – New – Projects 来显示 New project 对话框。

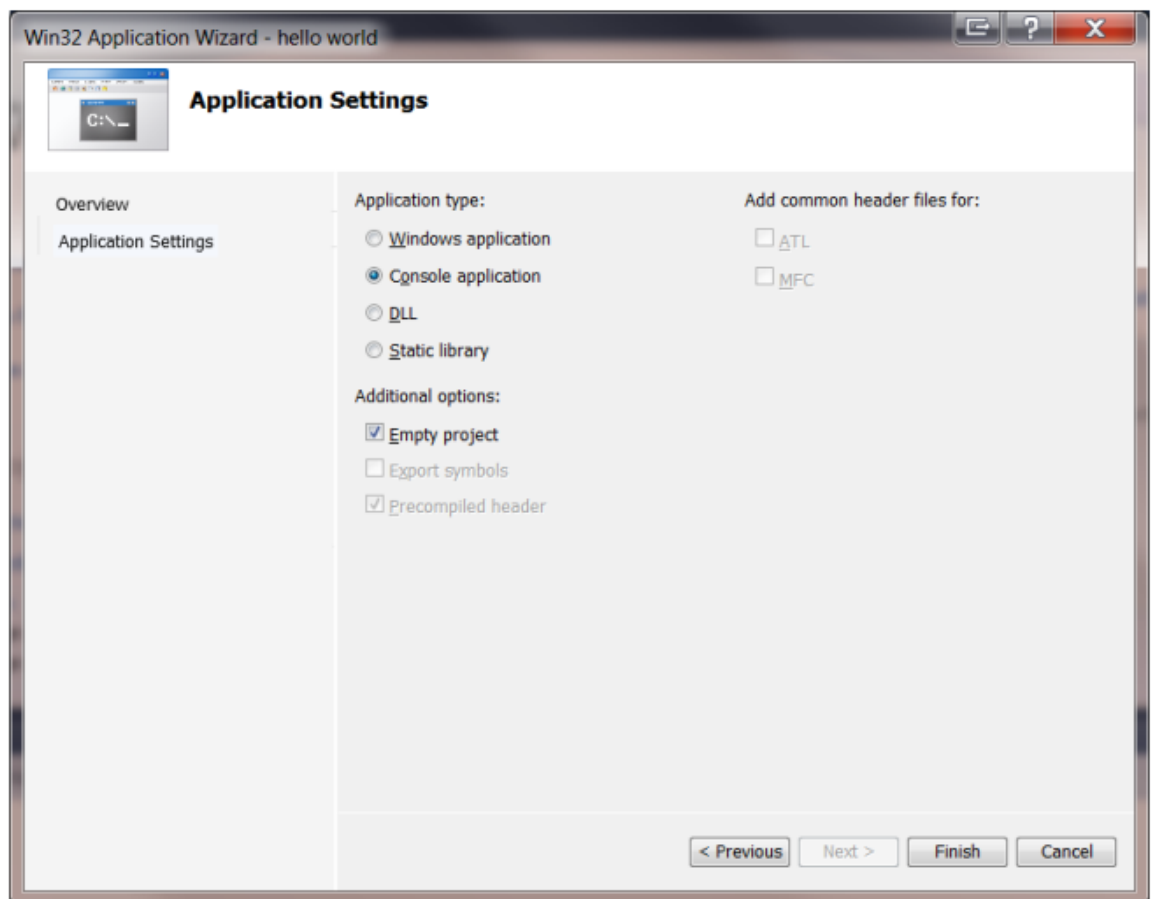


在 **New** 对话框中，通过单击已经安装的模板窗格中的 **Visual C++** 和中间窗格中的 **Win32 控制台应用程序** 进行选择。然后在名称框中输入文件名字（例如，**hello world** 如图所示），选择一个位置或文件夹用于储存项目文件。

单击 **OK** 按钮显示 **Win32 应用程序向导——hello world** 窗口如下图所示：



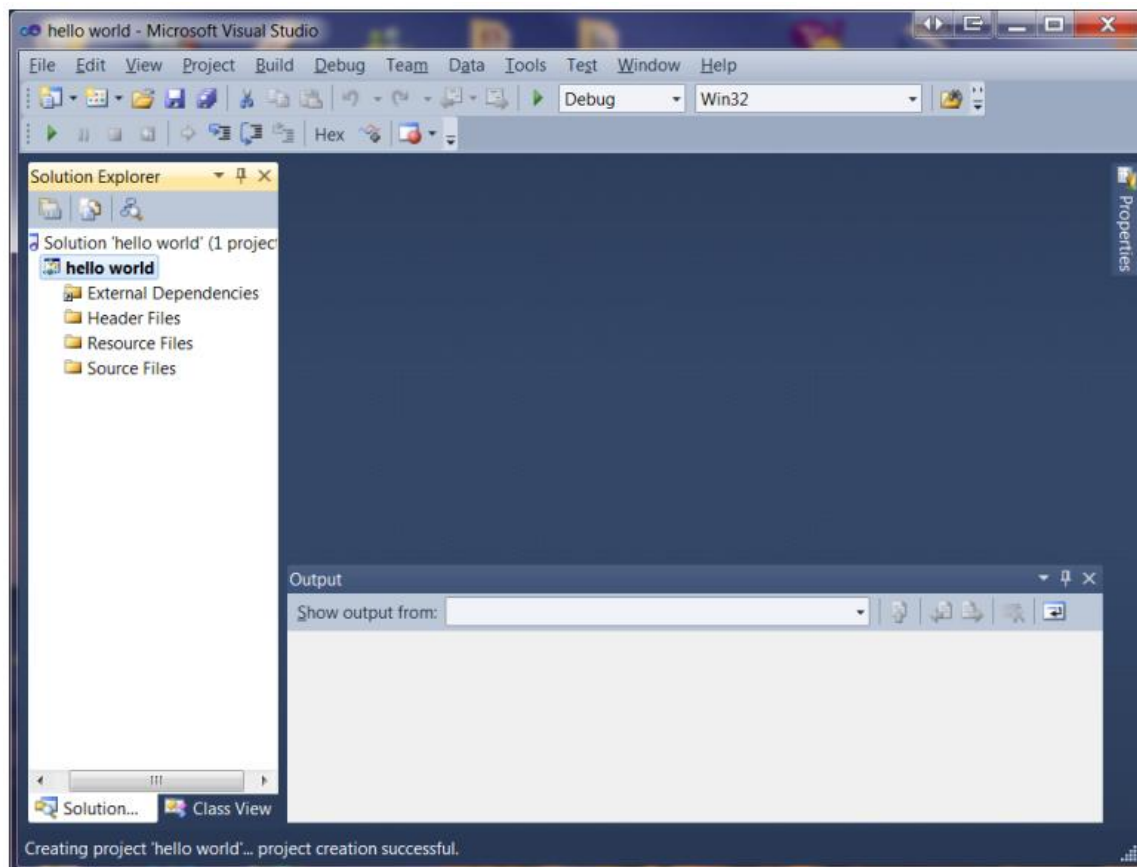
单击 **Next** 按钮，弹出如下对话框：



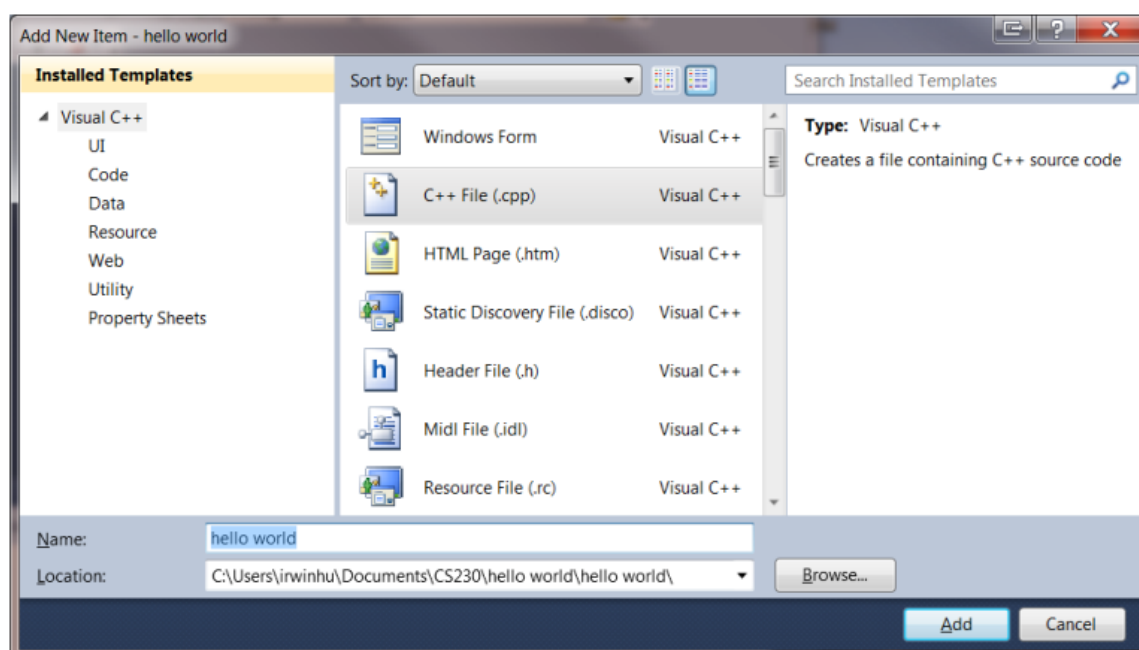
选择如上所示的空项目框，并单击 **Finish** 按钮继续下一步。

1.2.3 Add new source file

现在系统显示一下窗口。

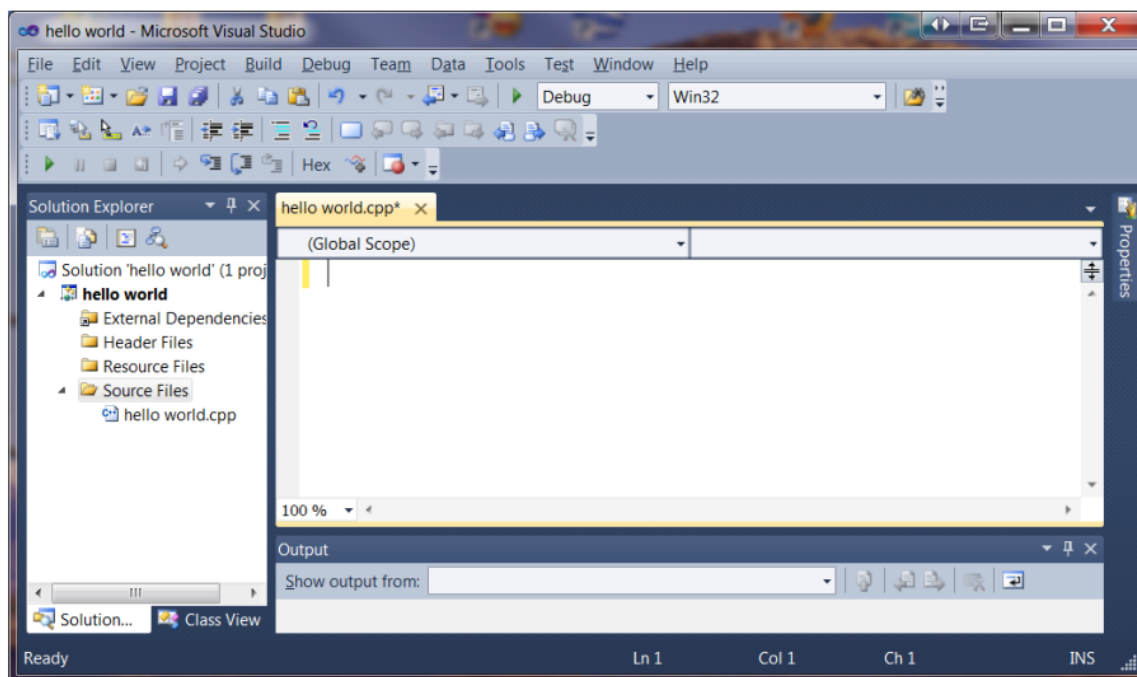


右键单击 Solution Explorer 中的 Source Files, 在弹出菜单中, 单击 Add 然后 New Item。Add New Item – hello world 对话框将会如下显示。

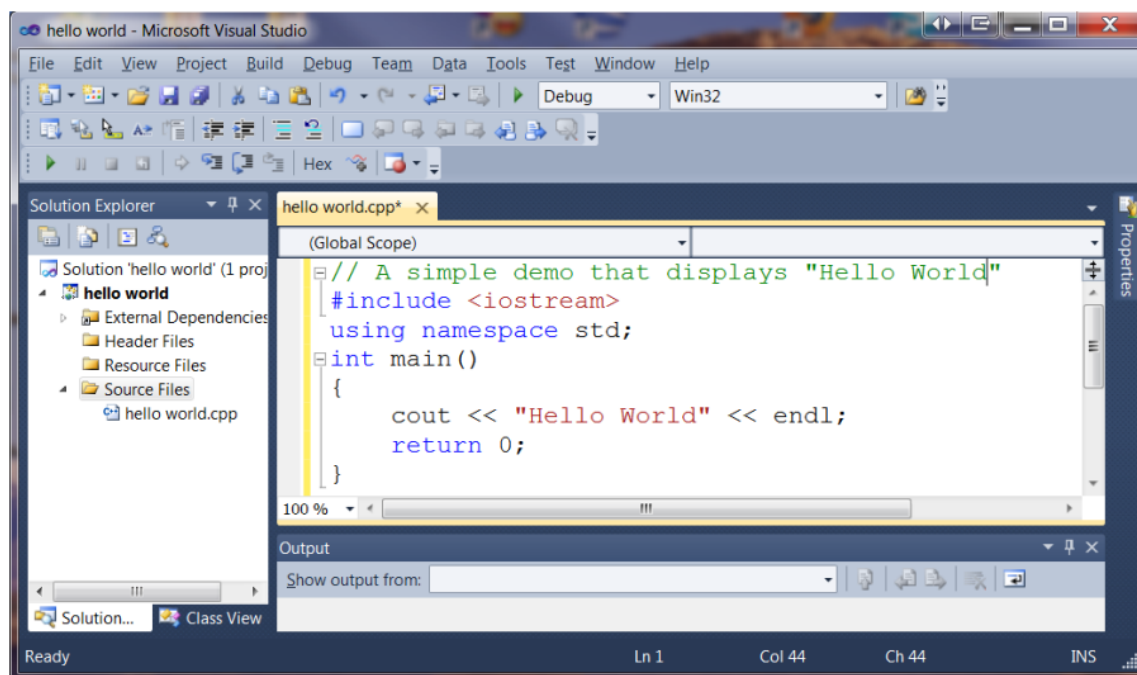


通过单击中间窗格中的 C++ File (.cpp), 选择它, 并输入一个任意文件名 (例如, **hello work**, 这里选择了我们用于项目的相同名称)。单击 **Add** 继续下一步。

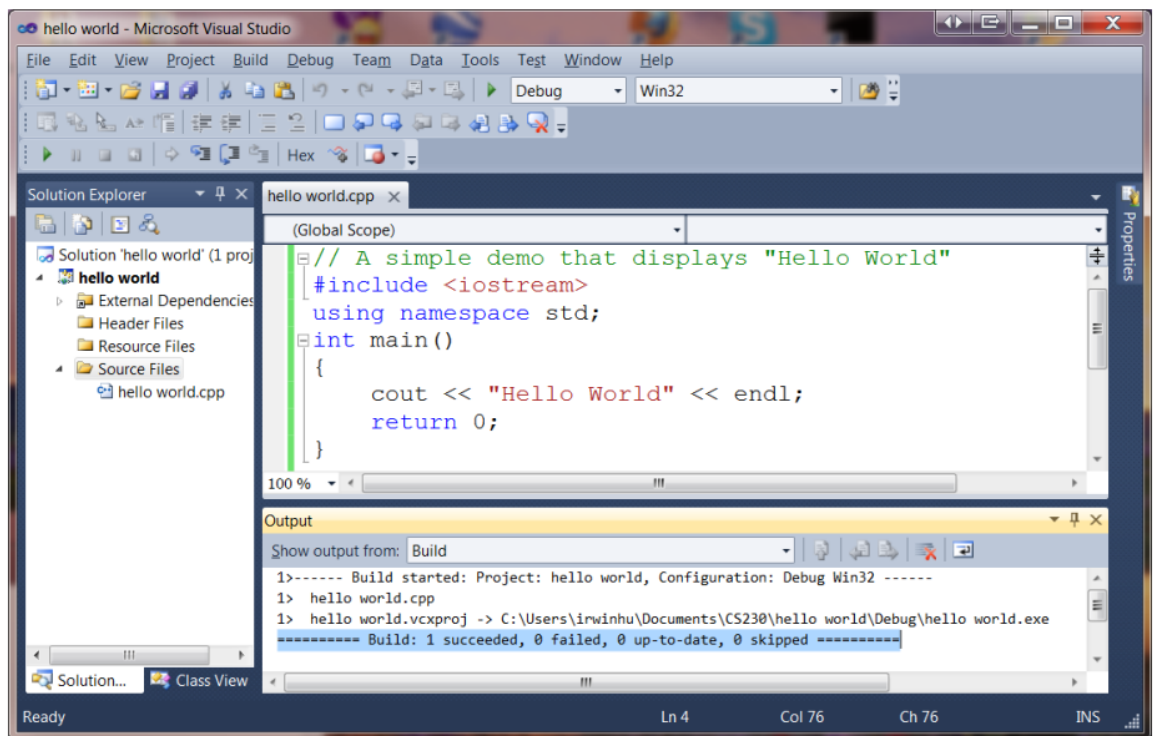
系统显示如下窗口。此时我们刚添加的 `cpp` 文件中，`hello word.cpp` 卡片中的编辑区是空白的，供我们输入 C++ 源代码。



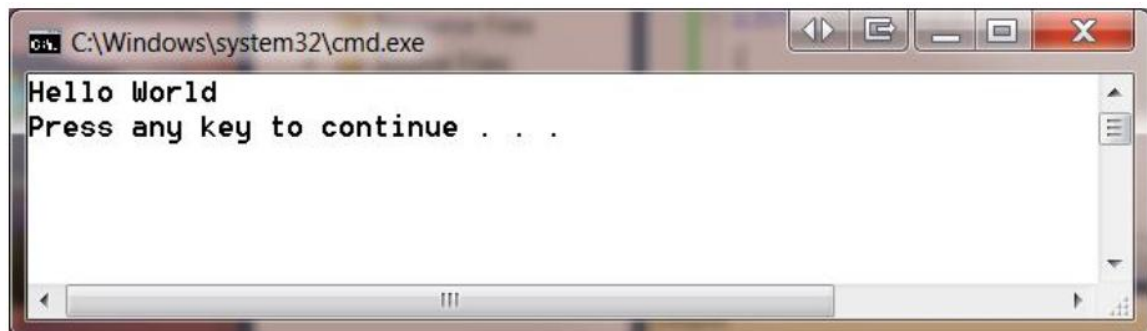
输入源代码后的同一窗口：



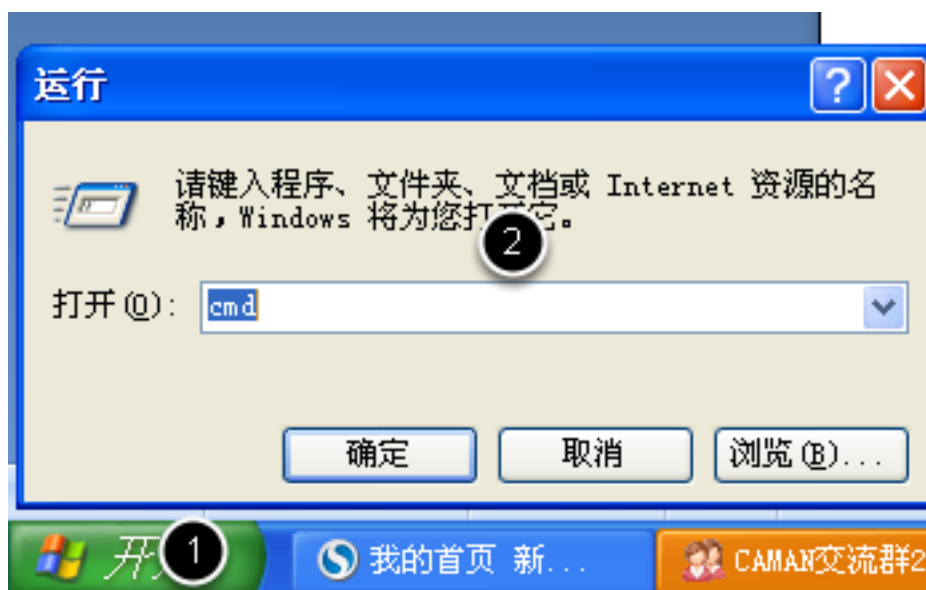
要在单个步骤中编译，链接，加载和执行程序，请单击菜单栏中的 `debug`，然后单击 `Start Without Debugging`。如果项目中没有错误，则输出窗口中显示 `====Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====`。



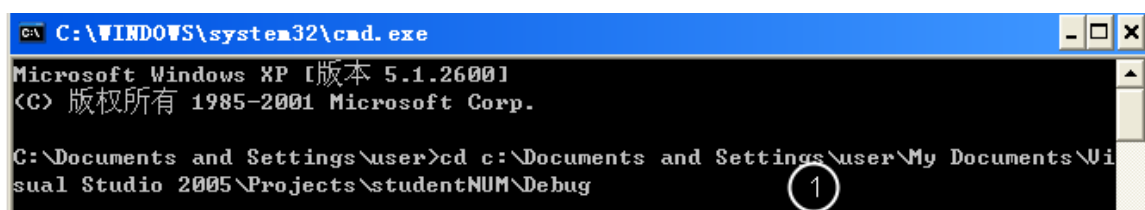
程序输出将显示在一个单独的窗口中（C:\Windows\system32\cmd.exe）：



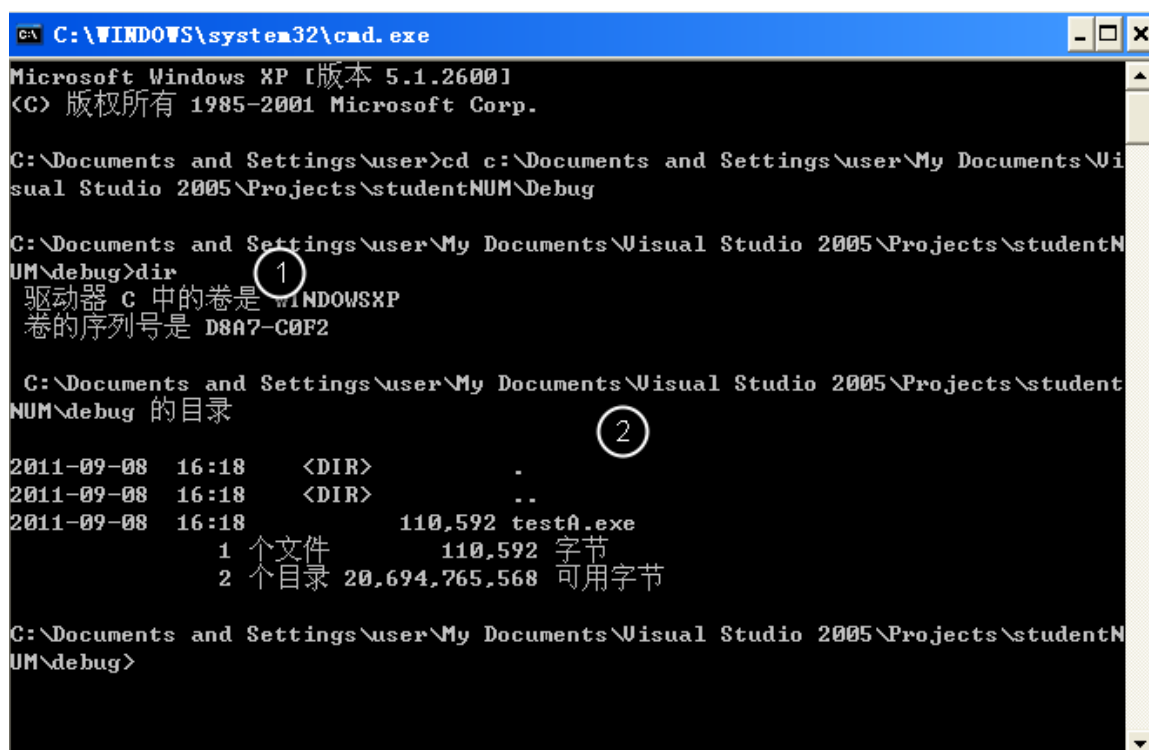
1.2.4 Run Program in Dos Prompt



点击开始->运行->输入 cmd,按回车键。



输入以上命令，`cd` 是更改路径的命令。当然，可以用复制+粘贴命令，把刚才的路径信息复制下来，如果不会就认真输入。



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\user>cd c:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\Debug

C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug>dir
驱动器 C 中的卷是 WINDOWSXP
卷的序列号是 D8A7-C0F2

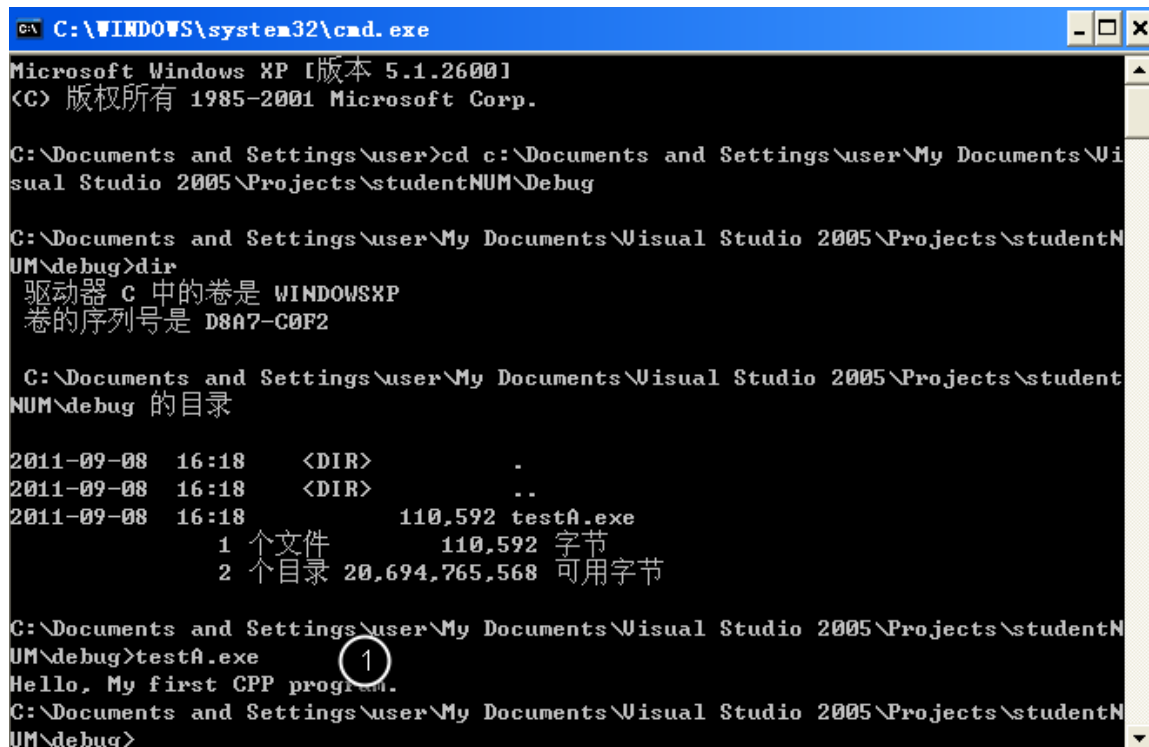
C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug 的目录

2011-09-08 16:18 <DIR>      .
2011-09-08 16:18 <DIR>      ..
2011-09-08 16:18             110,592 testA.exe
                1 个文件             110,592 字节
                2 个目录 20,694,765,568 可用字节

C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug>

```

争取变更路径后，用 `dir` 命令，可以显示当前目录中的文件。确认刚才编译的结果存在。如果不存在，可能是没有编译成功，或者路径错误。



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\user>cd c:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\Debug

C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug>dir
驱动器 C 中的卷是 WINDOWSXP
卷的序列号是 D8A7-C0F2

C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug 的目录

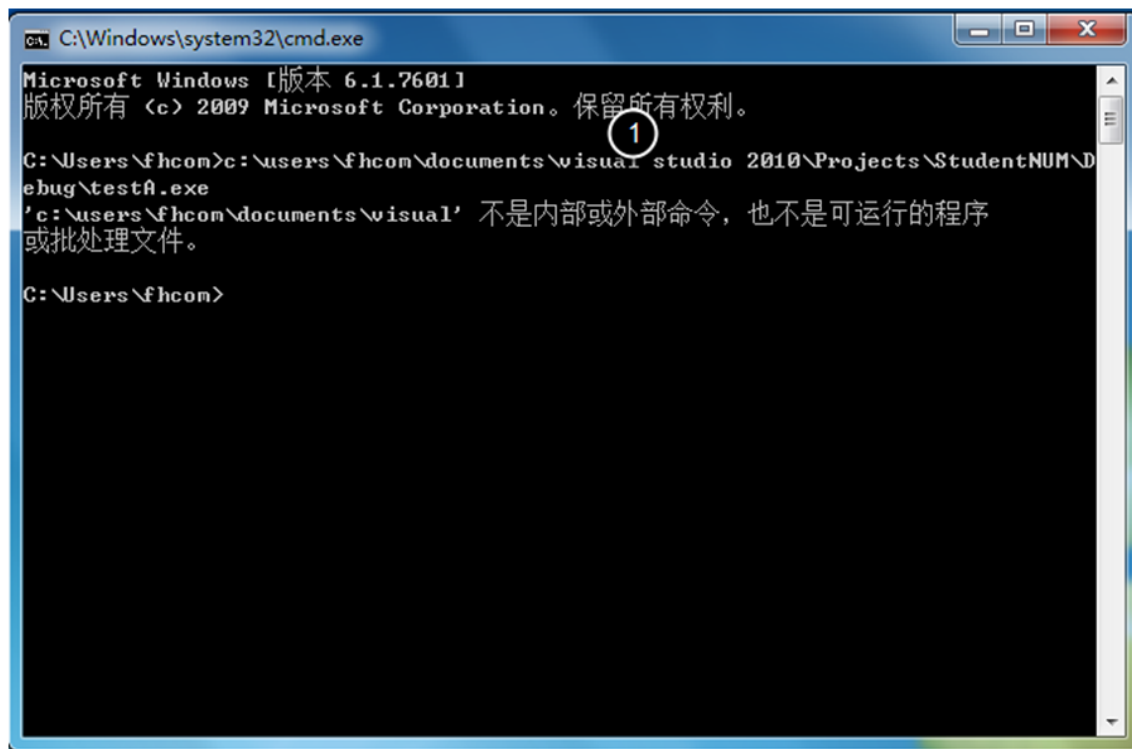
2011-09-08 16:18 <DIR>      .
2011-09-08 16:18 <DIR>      ..
2011-09-08 16:18             110,592 testA.exe
                1 个文件             110,592 字节
                2 个目录 20,694,765,568 可用字节

C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug>testA.exe
Hello, My first CPP program.

C:\Documents and Settings\user\My Documents\Visual Studio 2005\Projects\studentNUM\debug>

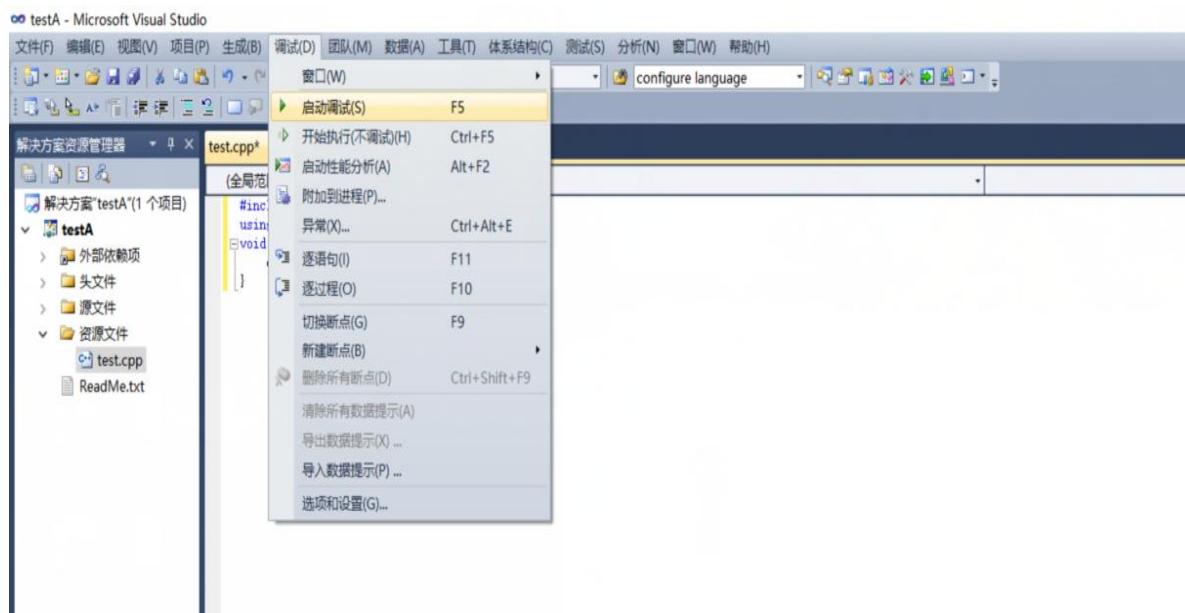
```

输入 `testA.exe`（或者 `hello world.exe` 取决于前边自己输入的文件名），回车键执行该程序。可以看到 `Hello` 信息，表面程序可以正常的编译执行。

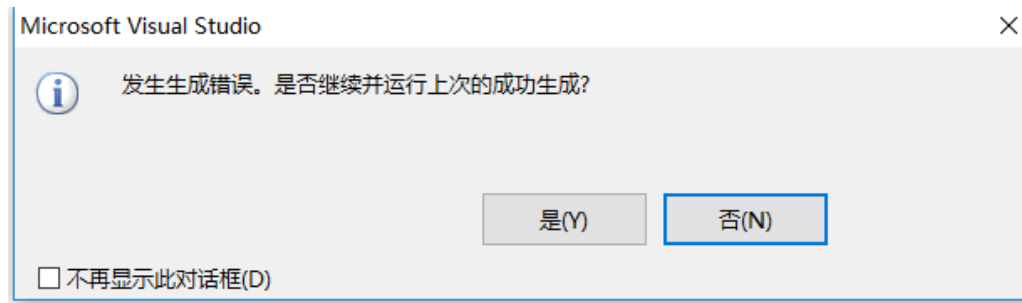


如果想偷懒，在 DOS 窗口下，直接复制路径和执行程序名，会出现错误。

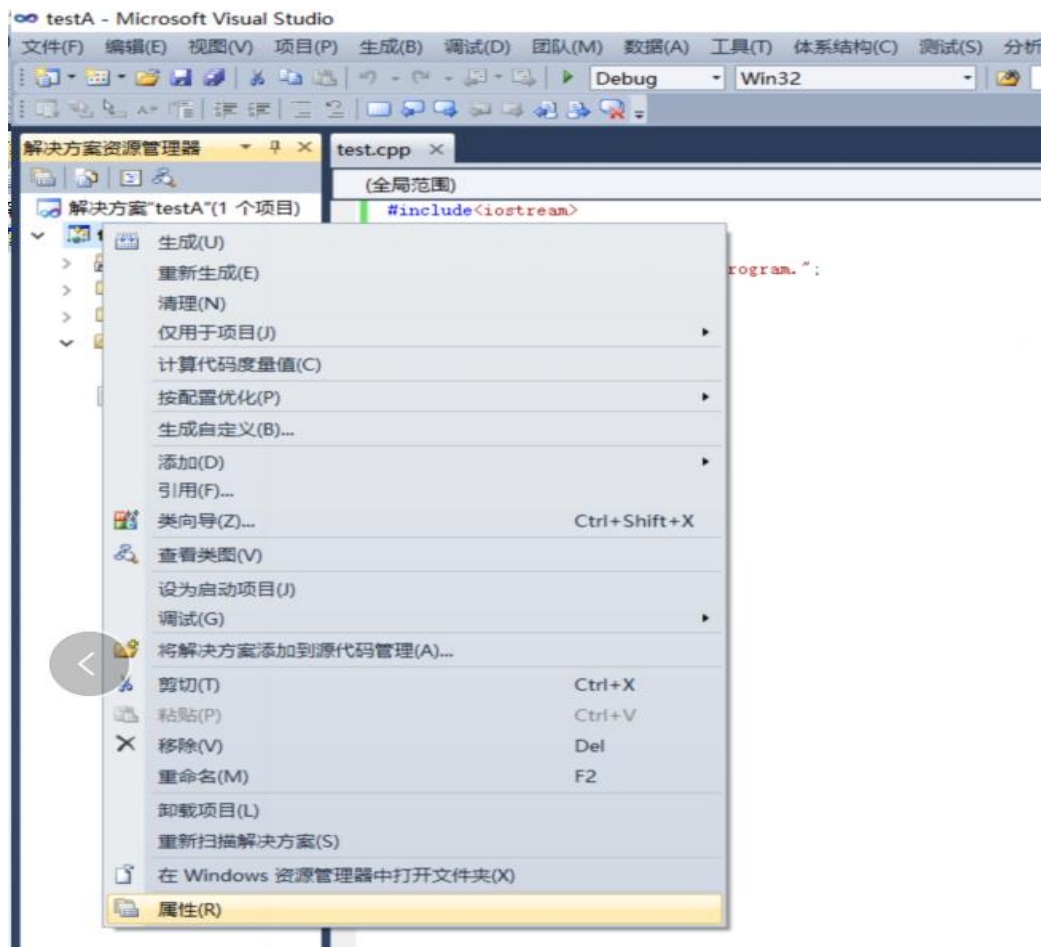
1.2.5 Run program in IDE



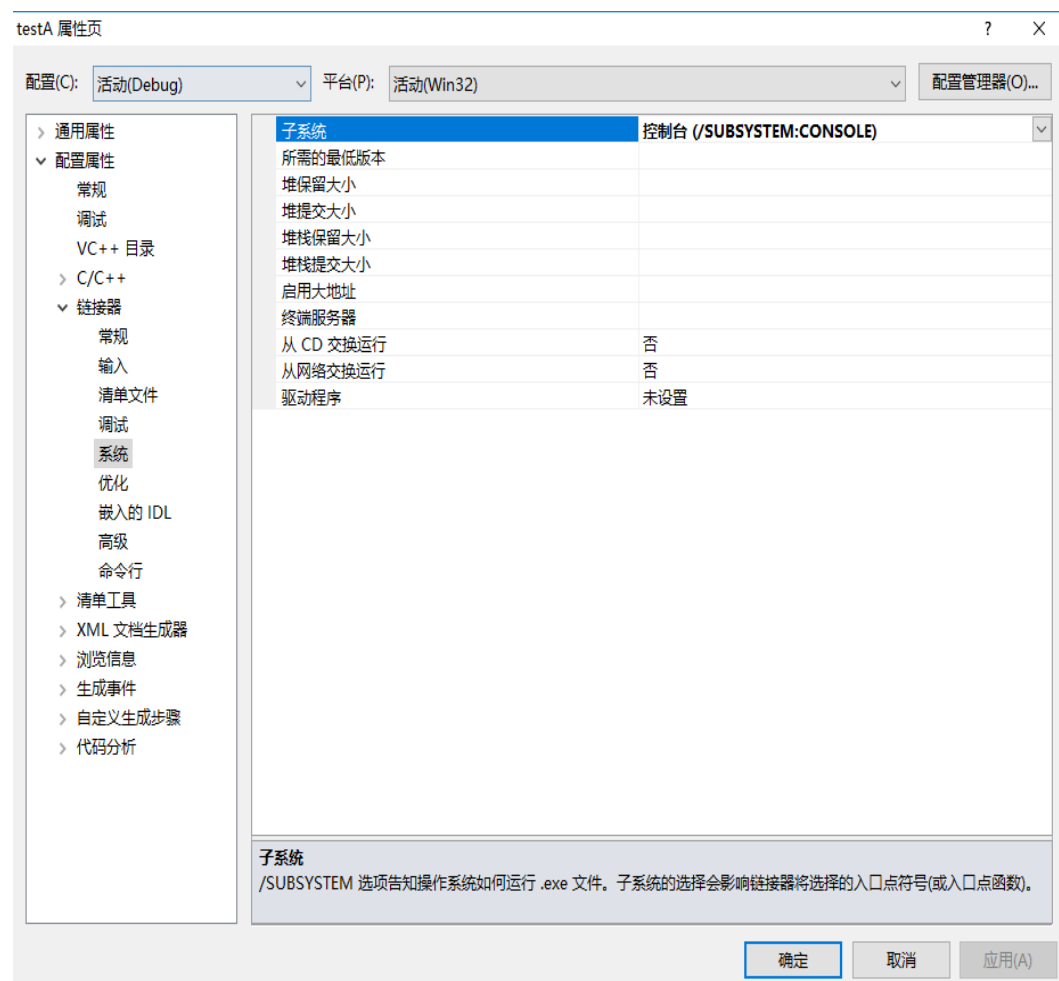
当然，大家也可以用 F5，利用调试方式去运行一个程序



程序不能正常运行。

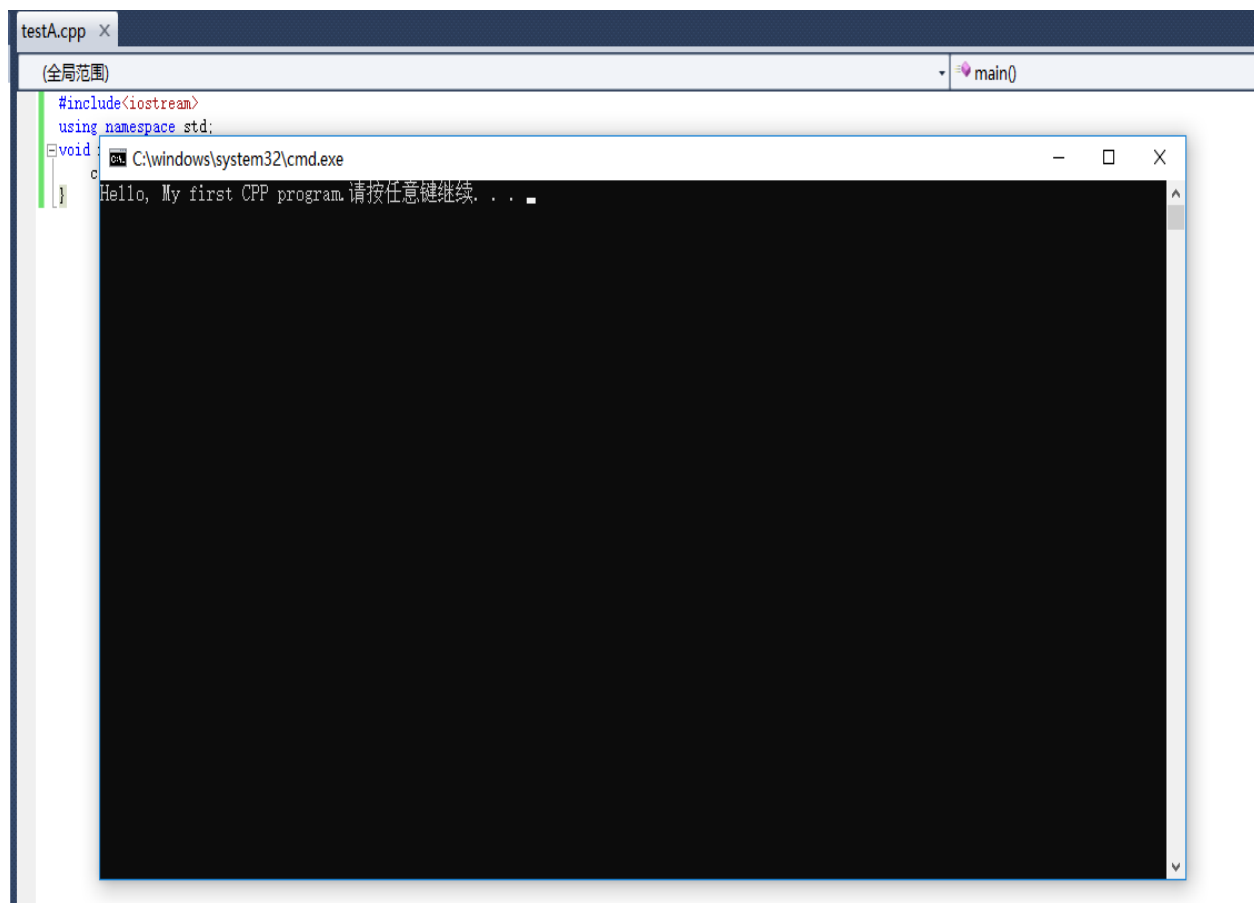


右击 testA 选择 properties



Linker->SubSystem 选择下拉框里的 Console

1.2.6 用 Ctrl+F5 运行



接下来，用 **Ctrl + F5**，来执行程序，就可以看到希望的黑色窗口了。注意 如果使用 **F5** 来执行还是看不见运行结果的。

两种执行方式的区别在于：

F5 run with debug

Ctrl + F5 run without debug.

如果你要自己看运行结果就用 **Ctrl+F5**

如果想 **Debug** 你的 **code**，看错误出在哪就用 **F5**。

你可以再 **code** 最左边你想要开始的 **debug** 的地方设一个 **breakpoint**，当你按 **F5**，你的程序会跳到那里，你可以用 **F10** 或者 **F11** 来查看，每行代码的运行结果

1.2.7 First Debug

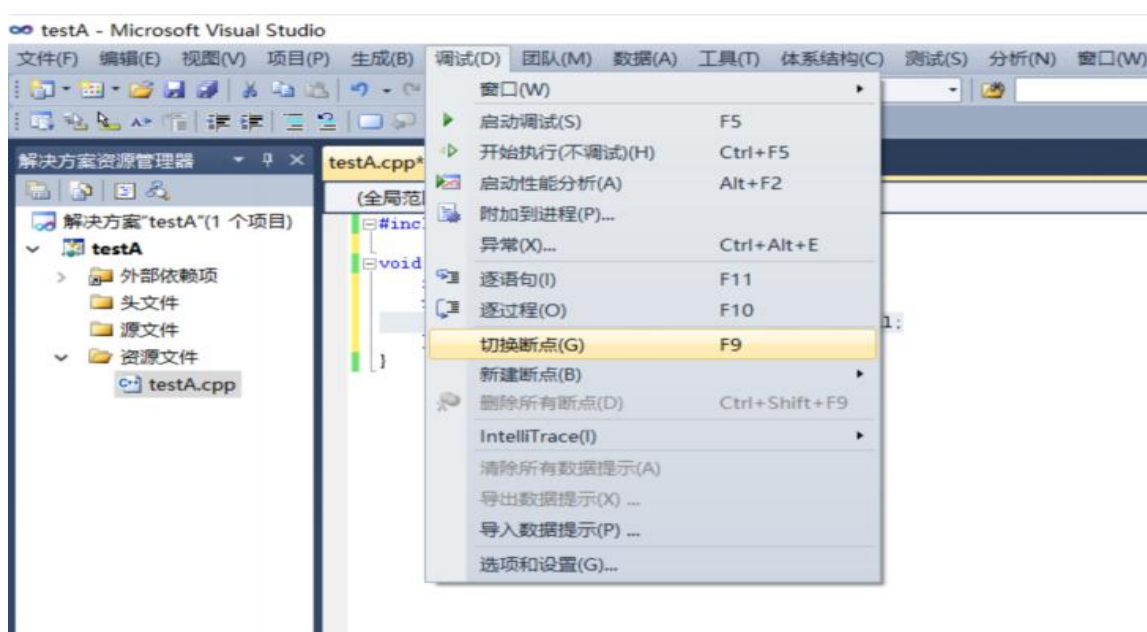


学习了解 Debug 功能。修改源代码为：

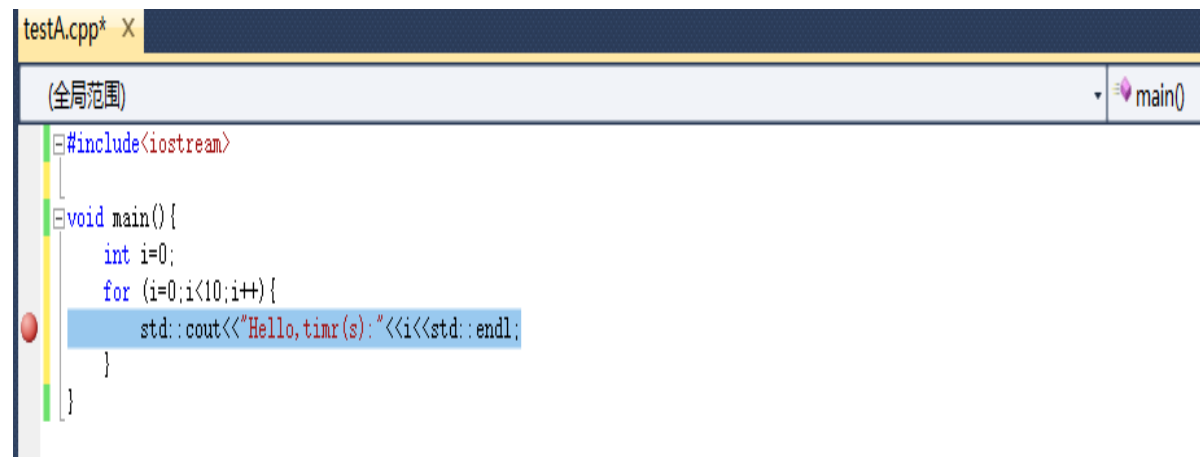
```
# include<iostream>
```

```
void main() {
    int i = 0;
    for (i = 0; i < 10; i++) {
        std::cout<<"Hello, time(s):"<< i << std::endl;
    }
}
```

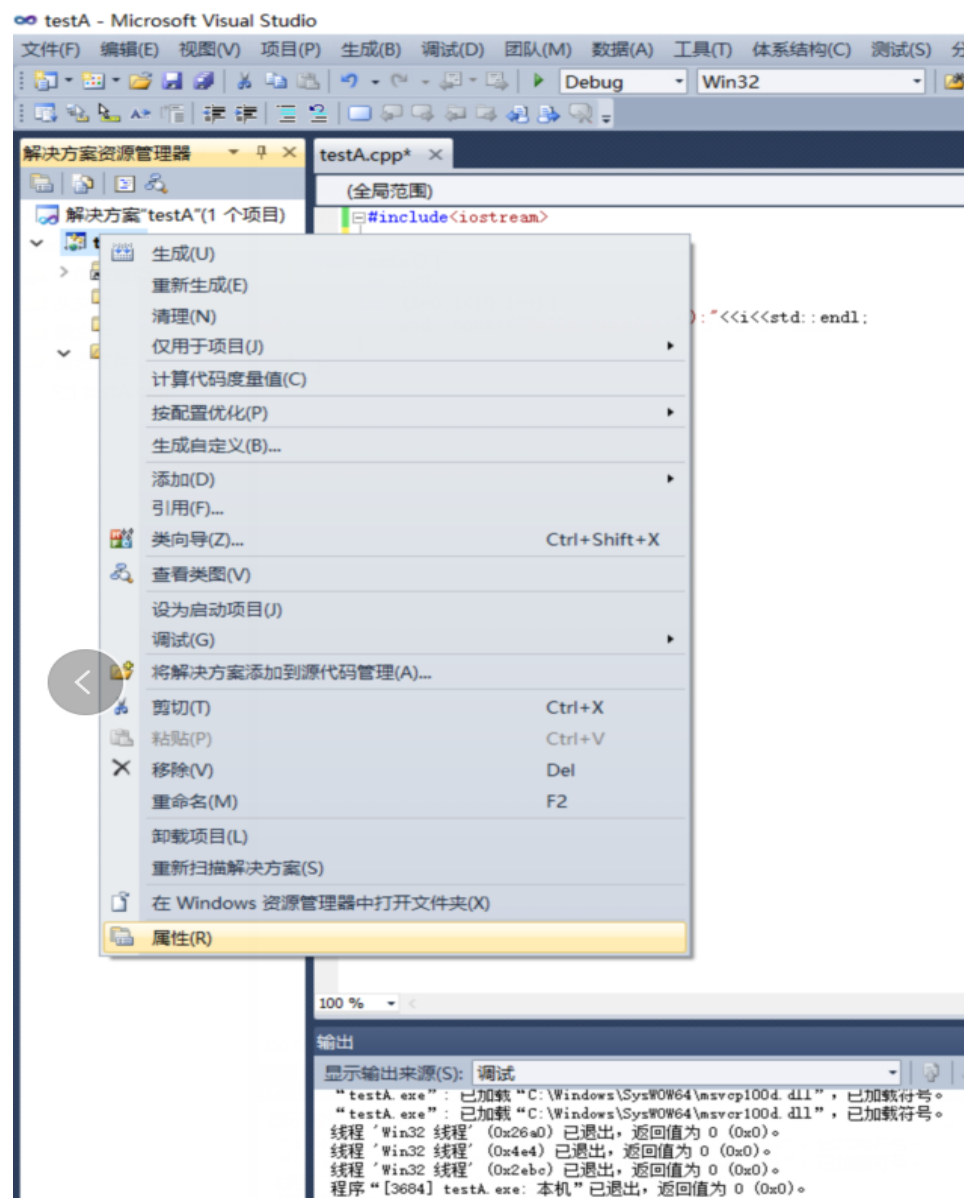
选择 `std::cout<<"Hello, time(s):"<< i << std::endl;` 这句话。



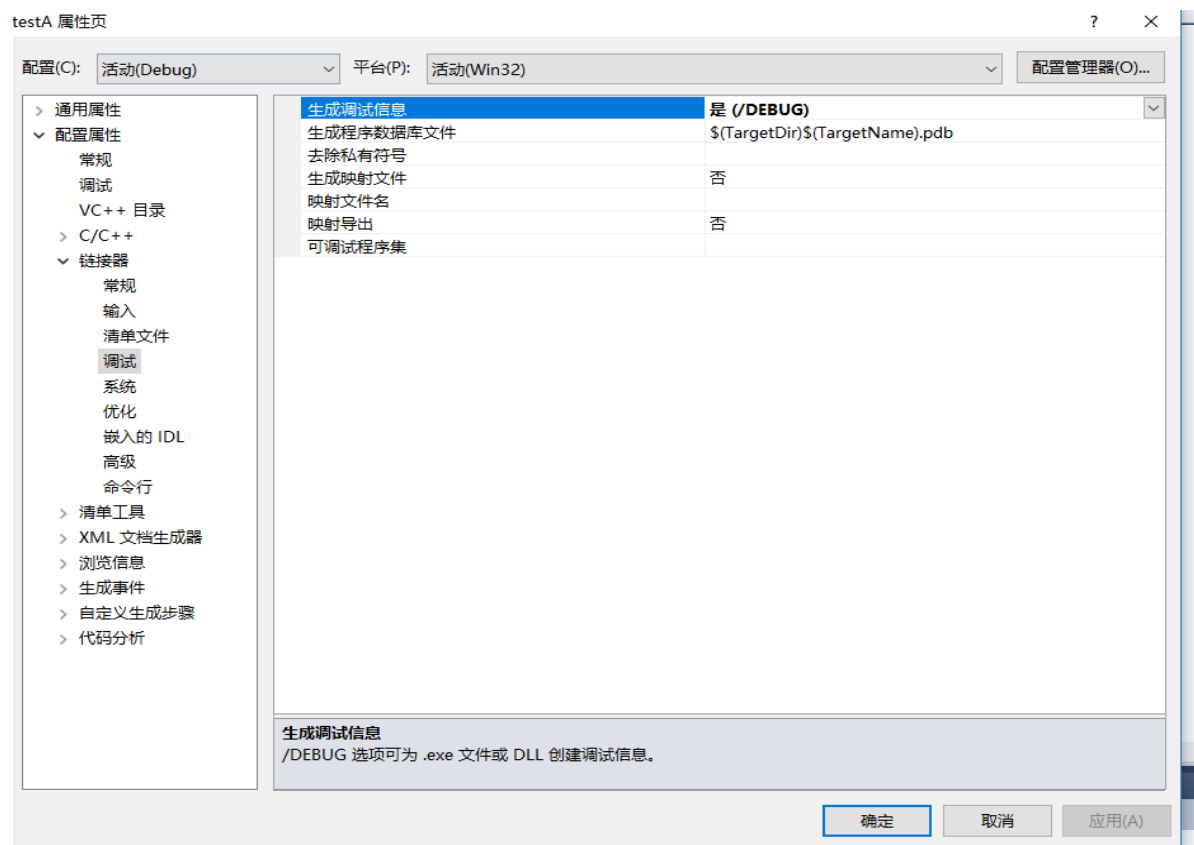
在菜单中选择 **Debug->Toggle Breakpoint**，当然，直接按 **F9** 也可以。



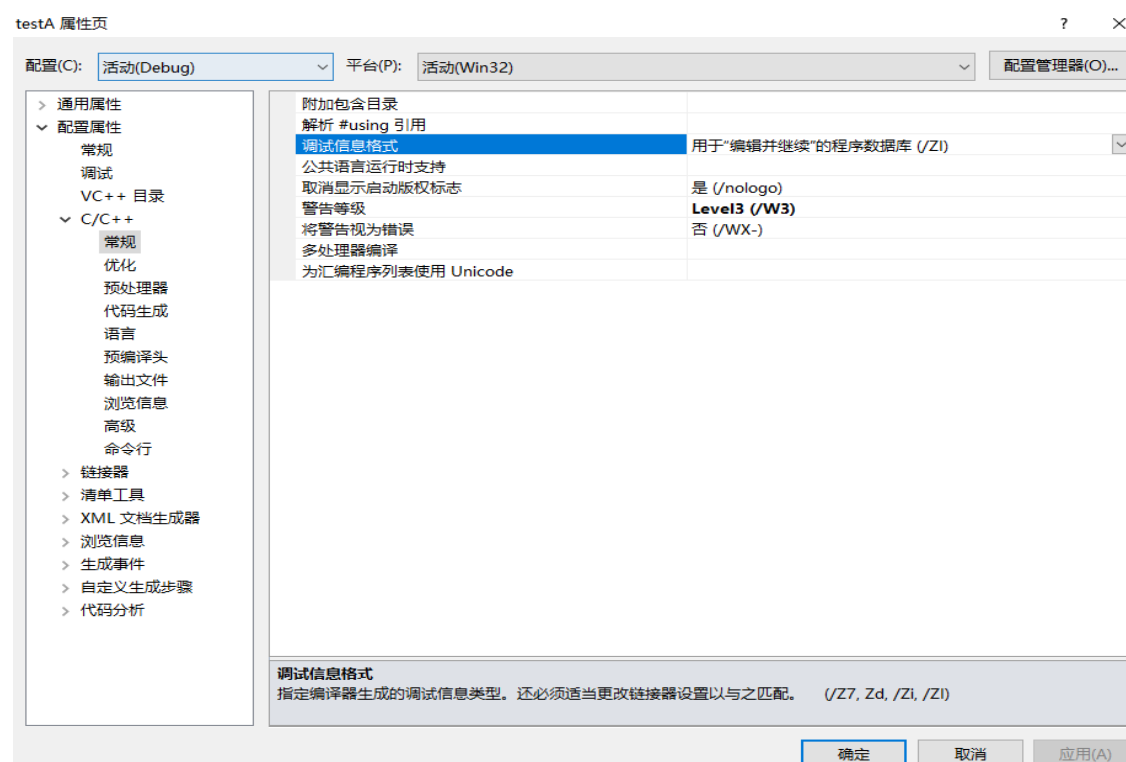
则在这句话前面出现了一个 红色 的标记，称为 断点。



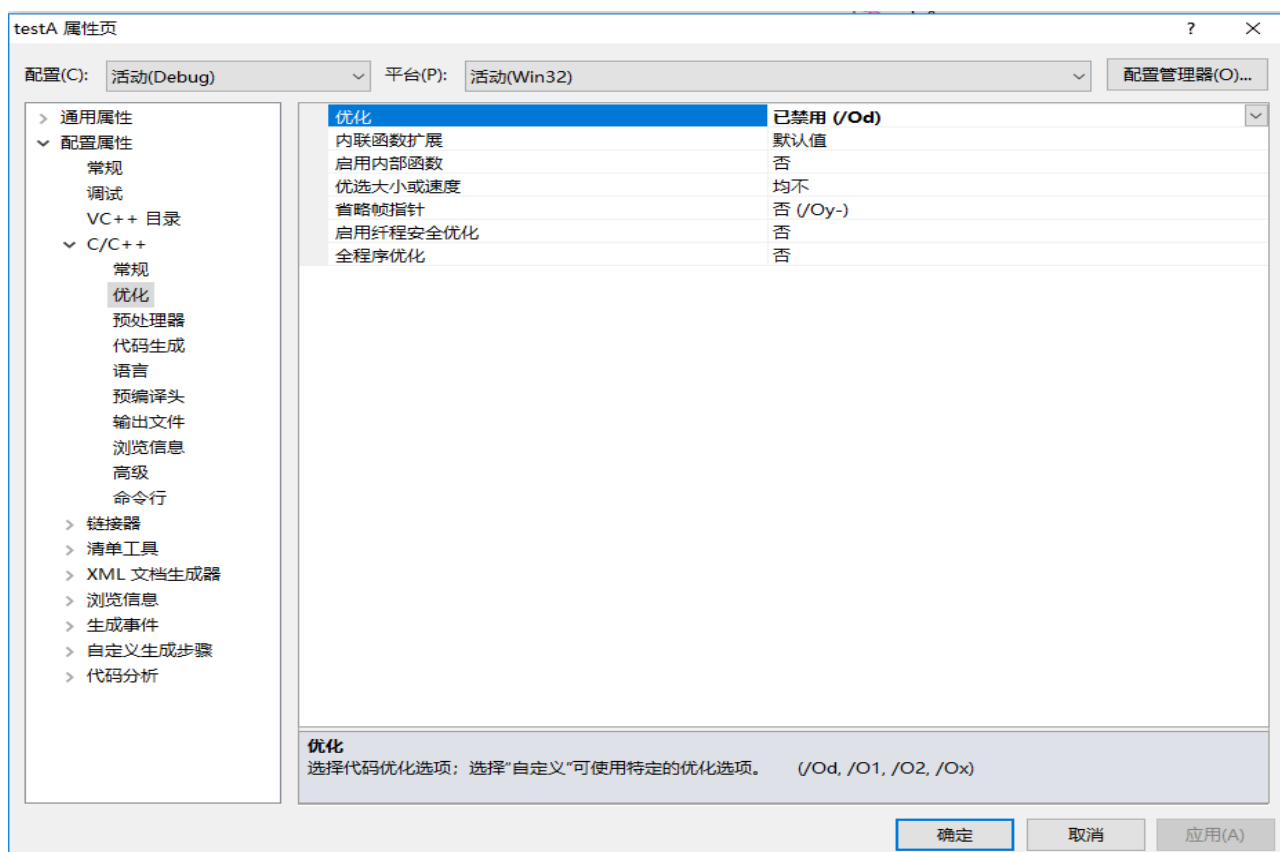
调试之前需要 更改一些属性。



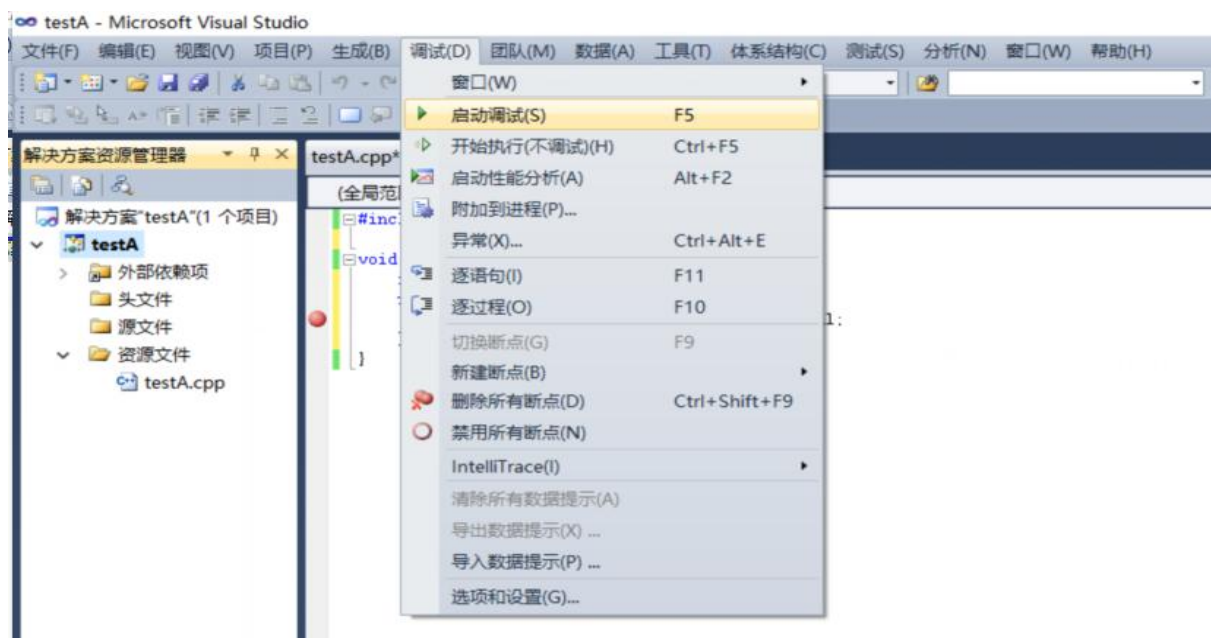
Linker->Debugger->Generate Debug Info 改为 Yes。



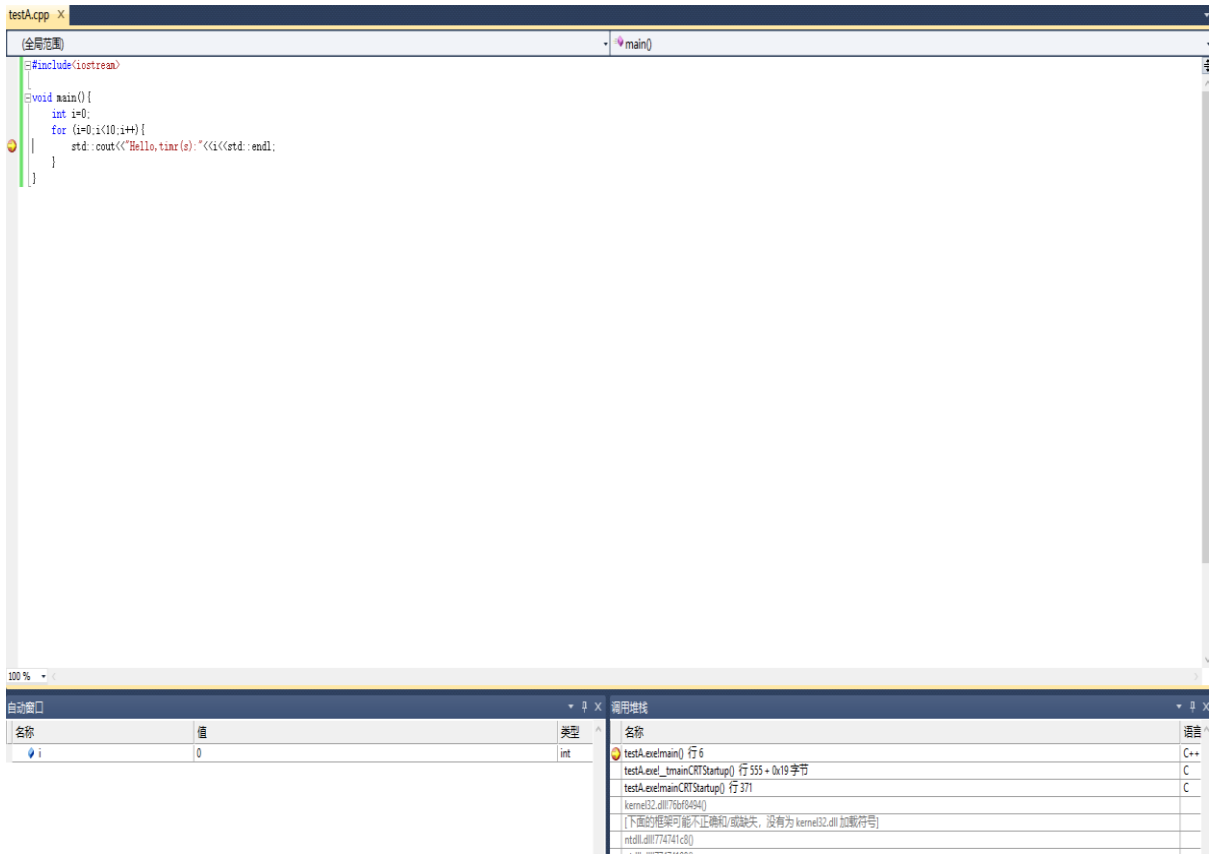
C/C++->General->Debug Information Format->ZI



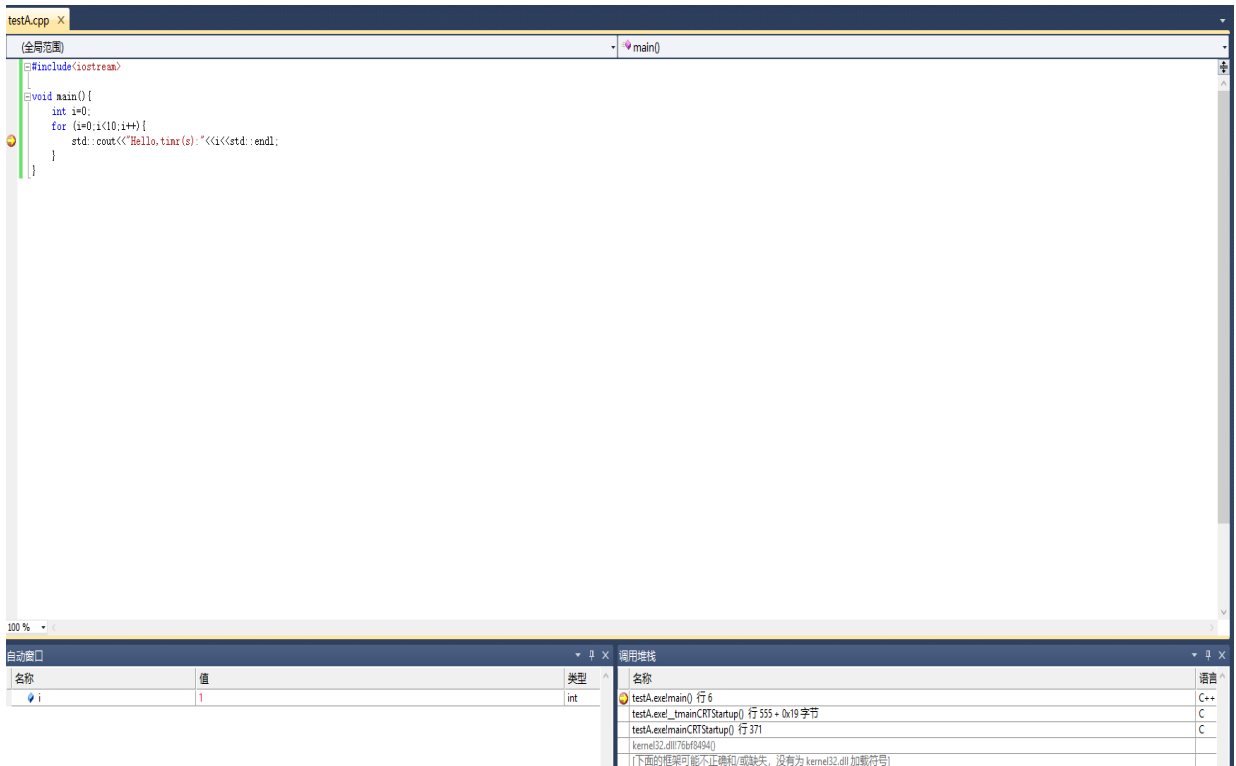
C/C++->Optimization->Optimization->Disabled



在菜单中选择 调试 -- 启动调试，当然，直接按 F5 也可以。



程序执行到断点所在位置并停顿下来， 下面的自动窗口，给出了代码中变量的值， 可以看见变量 `i = 0`。



可以按绿色按钮（F5）继续直到再次执行到断点，也可以按 F10，按照过程跟踪。这里选择 F5，继续执行，直到下个断点。当程序再次停顿下来时，变量 $i=1$ ，已经完成了一次循环。程序中可以设置多个断点，关于 F5，F10，F11 的区别，请进一步学习。

第二部分 实验作业

Lab1 Introduction to C++ Programming

Objectivities:

1. To write simple computer programs in C++.
2. To write simple input and output statements.
3. To use fundamental types.
4. Basic computer memory concepts.
5. To use arithmetic operators.
6. The precedence of arithmetic operators.
7. To write simple decision-making statements

Experiment

- **Ex1:**

输入书上 p35 例 fig02_03.cpp, 熟悉编程环境。

- **Ex2:**

Description of the Problem

Write a program that inputs three integers from the keyboard, and prints the sum, average, product, smallest and largest of these numbers. The screen dialogue should appear as follows: [Note: 13, 27 and 14 are input by the user.]

Sample Output

```
Input three different integers: 13 27 14
Sum is 54
Average is 18
Product is 4914
Smallest is 13
Largest is 27
```

Template

```
//EX2: numbercompare.cpp
2 #include <iostream> // allows program to perform input and output
3
4 using std::cout; // program uses cout
5 using std::endl; // program uses endl
```



```
6 using std::cin; // program uses cin
7
8 int main()
9 {
10 int number1; // first integer read from user
11 int number2; // second integer read from user
12 int number3; // third integer read from user
13 int smallest; // smallest integer read from user
14 int largest; // largest integer read from user
15
16 cout << "Input three different integers: "; // prompt
17 /* Write a statement to read in values for number1, number2 and
18 number3 using a single cin statement */
19
20 largest = number1; // assume first integer is largest
21
22 /* Write a statement to determine if number2 is greater than
23 largest. If so assign number2 to largest */
24
25 /* Write a statement to determine if number3 is greater than
26 largest. If so assign number3 to largest */
27
28 smallest = number1; // assume first integer is smallest
29
30 /* Write a statement to determine if number2 is less than
31 smallest. If so assign number2 to smallest */
32
33 /* Write a statement to determine if number3 is less than
34 smallest. If so assign number3 to smallest */
35
36 /* Write an output statement that prints the sum, average,
37 product, largest and smallest */
38
39 return 0; // indicate successful termination
40
41 } // end main
```

Problem-Solving Tips

1. Prompt the user to input three integer values. You will use a single cin statement to read all three values.
2. Sometimes it is useful to make an assumption to help solve or simplify a problem. For example, we assume number1 is the largest of the three values and assign it to largest. You will use if statements to determine whether number2 or number3 are larger.

3. Using an if statement, compare largest to number2. If the content of number2 is larger, then store the variable's value in largest.
4. Using an if statement, compare largest to number3. If the content of number3 is larger, then store the variable's value in largest. At this point you are guaranteed to have the largest value stored in largest.
5. Perform similar steps to those in Steps 2–4 to determine the smallest value.
6. Write a cout statement that outputs the sum, average, product (i.e., multiplication), largest and smallest values.
7. Be sure to follow the spacing and indentation conventions mentioned in the text.
8. If you have any questions as you proceed, ask your lab instructor for assistance.

Follow-Up Questions and Activities

1. Modify your solution to use three separate cin statements rather than one. Write a separate prompt for each cin.
2. Does it matter whether < or <= is used when making comparisons to determine the smallest integer? Which did you use and why?

● Ex3:

Description of the Problem

Write a program that reads in two integers and determines and prints whether the first is a multiple of the second. [Hint: Use the modulus operator.]

Sample Output

Enter two integers: 22 8 22 is not a multiple of 8

Problem-Solving Tips

9. The input data consists of two integers, so you will need two int variables to store the input values.
10. Use cin to read the user input into the int variables.
11. Use an if statement to determine whether the first number input is a multiple of the second number input. Use the modulus operator, %. If one number divides into another evenly, the modulus operation results in 0. If the result is 0, display a message indicating that the first number is a multiple of the second number.
12. Use an if statement to determine whether the first number input is not a multiple of the second number input. If one number does not divide into another evenly, the modulus operation results in a non-zero value. If non-zero, display a message indicating that the first number is not a multiple of the second.
13. Be sure to follow the spacing and indentation conventions mentioned in the text.
14. If you have any questions as you proceed, ask your lab instructor for assistance.

● Ex4:

Problem Description

Write a program that inputs a five-digit number, separates the number into its individual digits and prints the digits separated from one another by three spaces each. [Hint: Use integer division and the modulus operator.] For example, if the user inputs 42339, the program should print what is shown in the the sample output.

Sample Output

```
4 2 3 3 9
```

Template

```
1 // ex4
2 #include <iostream> // allows program to perform input and output
3
4 using std::cout; // program uses cout
5 using std::endl; // program uses endl
6 using std::cin; // program uses cin
7
8 int main()
9 {
10  int number; // integer read from user
11
12  cout << "Enter a five-digit integer: "; // prompt
13  cin >> number; // read integer from user
14
15  /* Write a statement to print the left-most digit of the
16  5-digit number */
17  /* Write a statement that changes number from 5-digits
18  to 4-digits */
19  /* Write a statement to print the left-most digit of the
20  4-digit number */
21  /* Write a statement that changes number from 4-digits
22  to 3-digits */
23  /* Write a statement to print the left-most digit of the
24  3-digit number */
25  /* Write a statement that changes number from 3-digits
26  to 2-digits */
27  /* Write a statement to print the left-most digit of the
28  2-digit number */
29  /* Write a statement that changes number from 2-digits
30  to 1-digit */
31  cout << number << endl;
32
```

```
33 return 0; // indicate successful termination
34
35 } // end main
```

Problem-Solving Tips

1. The input data consists of one integer, so you will use an int variable (number) to represent it. Note that the description indicates that one five-digit number is to be input—not five separate digits.
2. You will use a series of statements to “break down” the number into its individual digits using modulus (%) and division (/) calculations.
3. After the number has been input using cin, divide the number by 10000 to get the leftmost digit. Why does this work? In C++, dividing an integer by an integer results in an integer. For example, 42339 / 10000 evaluates to 4 because 10000 divides evenly into 42339 four times. The remainder 2339 is truncated.
4. Change the number to a 4-digit number using the modulus operator. The number modulus 10000 evaluates to the integer remainder—in this case, the right-most four digits. For example, 42339 % 10000 results in 2339. Assign the result of this modulus operation to the variable that stores the five-digit number input.
5. Repeat this pattern of division and modulus reducing the divisor by a factor of 10 each time (i.e., 1000, 100, 10). After the number is changed to a four-digit number, divide/modulus by 1000. After the number is changed to a three-digit number, divide/modulus by 100. And so on.
6. Be sure to follow the spacing and indentation conventions mentioned in the text.
7. If you have any questions as you proceed, ask your lab instructor for assistance.

Follow-Up Questions and Activities

1. What happens when the user inputs a number which has fewer than five digits? Why? What is the output when 1763 is entered?
2. The program you completed in this lab exercise inputs a number with multiple digits and separates the digits. Write the inverse program, a program which asks the user for three one-digit numbers and combines them into a single three-digit number. [Hint: Use multiplication and addition to form the three-digit number.]

Lab2 Introduction to Classes and Objects and Strings

Objectivities

1. How to define a class and use it to create an object
2. How to implement a class's behaviors as member functions
3. How to implement a class's attributes as data members
4. How to call a member function of an object to perform a task.
5. The differences between data members of a class and local variables of a function.
6. How to use a constructor to initialize that an object's data is initialized when the object is created.
7. How to engineer a class to separate its interface from its implementation and encourage reuse.
8. How to use objects of class string.

Experiments

● Ex1(p.84--3.11)

Description of the Problem

Modify class GradeBook (Figs. 3.11-3.12) as follows:

- (a) Include a second string data member that represents the course instructor's name.
- (b) Provide a *set* function to change the instructor's name and a *get* function to retrieve it.
- (c) Modify the constructor to specify course name and the instructor's name parameters.
- (d) Modify function `displayMessage` to output the welcome message and course name, then the string "This course is presented by: " followed by the instructor's name.

Use the modified class in a test program that demonstrates the class's new capabilities.

Sample Output

```
Welcome to the grade book for
CS101 Introduction to C++ Programming!
This course is presented by: Sam Smith
Changing instructor name to Judy Jones
Welcome to the grade book for
CS101 Introduction to C++ Programming!
This course is presented by: Judy Jones
```

Problem-Solving Tips

1. In class GradeBook, declare a string data member to represent the instructor's name.
2. Declare a public *set* function for the instructor's name that does not return a value and takes a string as a parameter. In the body of the *set* function, assign the parameter's value to the data member that represents the instructor's name.
3. Declare a public *get* function that returns a string and takes no parameters. This member function should return the instructor's name.

4. Modify the constructor to take two string parameters. Assign the parameter that represents the instructor's name to the appropriate data member.
5. Add a cout statement to member function displayMessage to output the value of the data member you declared earlier.
6. Be sure to follow the spacing and indentation conventions mentioned in the text.
7. If you have any questions as you proceed, ask your lab instructor for help.

● **Ex2(p.85--3.14)**

Sample Output

Employee 1: Bob Jones; Yearly Salary: 34500 Employee 2: Susan Baker; Yearly Salary: 37800 Increasing employee salaries by 10% Employee 1: Bob Jones; Yearly Salary: 37944 Employee 2: Susan Baker; Yearly Salary: 41580

Problem-Solving Tips

1. Class Employee should declare three data members.
2. The constructor must declare three parameters, one for each data member. The value for the salary should be validated to ensure it is not negative.
3. Declare a public set and get functions for each data member. The set functions should not return values and should each specify a parameter of a type that matches the corresponding data member (string for first name and last name, int for the salary). The get functions should receive no parameters and should specify a return type that matches the corresponding data member.
4. When you call the constructor from the main function, you must pass it three arguments that match the parameters declared by the constructor.
5. Giving each employee a raise will require a call to the get function for the salary to obtain the current salary and a call to the set function for the salary to specify the new salary.
6. Be sure to follow the spacing and indentation conventions mentioned in the text.
7. If you have any questions as you proceed, ask your lab instructor for help.

● **Ex3(p.85--3.15)**

Lab3 Control Statements: Part I

Objectivities:

1. Basic problem-solving techniques.
2. To develop algorithm through the process of top-down, stepwise refinement.
3. Counter-controlled repetition and sentinel-controlled repetition.
4. To use the **if** and **if...else** selection statements to choose among alternative actions.
5. To use the **while** repetition statement to execute statements in a program repeatedly.

Experiment

● Ex1(P126-- 4.17)

The process of finding the largest number (i.e., the maximum of a group of numbers) is used frequently in computer applications. For example, a program that determines the winner of a sales contest inputs the number of units sold by each salesperson. The salesperson who sells the most units wins the contest.

Your program should use three variables, as follows:

- a) counter: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed).
- b) number: The current number input to the program.
- c) largest: The largest number found so far.

Requirement:

1. Write a pseudocode program.
2. Write a C++ program
 - a) To input 10 numbers **from keyboard** by the user.
 - b) To use a **while** and **if** statement to determine the largest number.
 - c) To **print** the largest number.
3. Execute the program.

● Ex2(P128-4.26)

A Palindrome is a number or text phrase that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611.

Requirement:

1. Write a C++ program
 - a) To read in a five-digit integer.
 - b) To use a **while** and **if...else** statement to determine whether it is a palindrome.
 - c) To **Output** “the five-digit integers is a palindrome!” or “It is not a palindrome!”.
2. Execute the program.

✓ **Hint:**

Use the division and modulus operators to separate the number into its individual digits.

- **Ex3(P128-4.27)**

- **Ex4(P128-4.34)**

Lab4 Control Statements: Part2; Logical Operators

Objective

1. The essentials of counter-controlled repetition.
2. To use the for and do...while repetition statements to execute statements in a program repeatedly.
3. To understand multiple selection using the switch selection statement.
4. To use the break and continue program control statements to alter the flow of control.
5. To use the logical operators to form complex conditional expressions in control statements.
6. To avoid the consequences of confusing the equality and assignment operators.

Experiments

- **Ex1(p.162—5.6)**

Description of the Problem

Write a program that uses a for statement to calculate and print the average of several integers.

Assume the last value read is the sentinel **9999**. A typical input sequence might be

10 8 11 7 9 9999 indicating that the program should calculate the average of all the values preceding **9999**.

Sample Output

Enter integers (9999 to end):

10 8 11 7 9 9999

The average is: 9

Template

```

1  // Lab 1: IntegerAverage.cpp
2  // Calculate the average of several integers.
3
4  #include <iostream>
5  using std::cin;
6  using std::cout;
7  using std::endl;
8
9  int main()
10 {
11     int value; // current value
12     int count = 0; // number of inputs
13     int total; // sum of inputs
14
```

```

15 // prompt for input
16 cout << "Enter integers (9999 to end):" << endl;
17 cin >> value;
18
19 // loop until sentinel value read from user
20 /* Write a for header to initialize total to 0
21    and loop until value equals 9999 */
22 {
23     /* Write a statement to add value to total */
24     /* Write a statement to increment count */
25
26     cin >> value; // read in next value
27 } // end for
28
29 // if user entered at least one value
30 if (count != 0 )
31     cout << "\nThe average is: "
32     << /* Convert total to a double and divide it by count */ << endl;
33 else
34     cout << "\nNo values were entered." << endl;
35
36 return 0; // indicate program ended successfully
37 } // end main

```

Problem-Solving Tips

1. When used for sentinel-controlled repetition, a **for** loop can be written much like a **while** loop, using the same loop-continuation condition as a **while** loop.
2. When performing sentinel-controlled repetition, a **for** loop does not need to increment any counter variable. But it can still initialize a variable if so desired.

Follow-up Question

1. Modify the program to perform counter-controlled repetition with a **for** loop. Assume that the first integer entered by the user represents the number of subsequent integers that the user will input to be averaged.

Sample output:

```

Enter integers (first integer should be the number of subsequent integers):
5 12 7 4 19 6
The average is: 9.6

```

● **Ex2(p.164—5.20)**

Description of the Problem

A right triangle can have sides that are all integers. A set of three integer values for the sides of a

right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Find all Pythagorean triples for *side1*, *side2* and *hypotenuse* all no larger than 500. Use a triple-nested for loop that tries all possibilities. This is an example of “brute force computing.” You will learn in more advanced computer-science courses that there are many interesting problems for which there is no known algorithmic approach other than using sheer brute force.

Sample Output

Side 1	Side 2	Side3
3	4	5
5	12	13
6	8	10
7	24	25
...		
319	360	481
320	336	464
325	360	485
340	357	493
A total of 386 triples were found.		

Template

```

1 // Lab 2: pythagorean.cpp
2 // Find Pythagorean triples using brute force computing.
3 #include <iostream>
4 using std::cout;
5 using std::endl;
6
7 int main()
8 {
9     int count = 0; // number of triples found
10    long int hypotenuseSquared; // hypotenuse squared
11    long int sidesSquared; // sum of squares of sides
12
13    cout << "Side 1\tSide 2\tSide3" << endl;
14
15    // side1 values range from 1 to 500
16    /* Write a for header for side1 */
17    {
18        // side2 values range from current side1 to 500
19        /* Write a for header for side2 */
20        {
21            // hypotenuse values range from current side2 to 500
22            /* Write a for header for hypotenuse */

```

```

23     {
24         // calculate square of hypotenuse value
25         /* Write a statement to calculate hypotenuseSquared */
26
27         // calculate sum of squares of sides
28         /* Write a statement to calculate the sum of the sides Squared */
29
30         // if (hypotenuse)^2 = (side1)^2 + (side2)^2,
31         // Pythagorean triple
32         if ( hypotenuseSquared == sidesSquared )
33         {
34             // display triple
35             cout << side1 << '\t' << side2 << '\t'
36                 << hypotenuse << '\n';
37             count++; // update count
38         } // end if
39     } // end for
40 } // end for
41 } // end for
42
43 // display total number of triples found
44 cout << "A total of " << count << " triples were found." << endl;
45 return 0; // indicate successful termination
46 } // end main

```

Problem-Solving Tips

1. This program does not require any input from the user.
2. This program can take a significant amount of time to run, depending on your computer's processor speed. If you have a CPU monitor available on your system, it is worth taking a look at it when this program executes.
3. Do not be concerned that you are trying values that do not seem to make sense, such as a 1–1–500 triangle.
4. Remember that brute-force techniques try all possible values.
5. The formula for the Pythagorean Theorem is $\text{hypotenuse}^2 = (\text{side } 1)^2 + (\text{side } 2)^2$.
6. To avoid producing duplicate Pythagorean triples, start the second for loop at $\text{side2} = \text{side1}$ and the third for loop at $\text{hypotenuse} = \text{side2}$. This way, when a Pythagorean triple is found, side1 will be the shortest side of the triangle and hypotenuse will be the longest side.

Follow-up Question

1. How many times did this program execute the innermost for loop? Add another counter to the program that counts the number of times this loop iterates. Declare a new variable of type **long**, named *loopCounter* and initialize it to 0. Then add a statement in the innermost for statement that increments *loopCounter* by 1. Before exiting the program, print the value of *loopCounter*. Do the numbers match?

2. Add a **break** statement to the program inside the innermost for loop. This **break** statement should be called after the 20th Pythagorean triple is found. Explain what happens to the program after the 20th triple is found. Are all three for loops exited, or just the innermost one? What happens when the **break** statement is placed inside the middle loop? The outermost loop?
3. Add a **continue** statement to the program that prevents a Pythagorean triple from being found when `side1` is equal to 8. Using your solution to *Follow-Up Question 1*, calculate how many times this new program executes the innermost for loop. Explain why the **continue** statement affected the output.
4. Explain why a **long** variable is used for *hypotenuseSquared* and *sideSquared*. Modify the program so that they are both of type **short** instead of type **long**. Rerun the program. What happens?

- **Ex3(p.163--5.23、 5.24)**
- **Ex4(p.99—5.28)**

Lab5 Function and Recursion-I

Objective

1. To construct programs modularly from functions.
2. Master the method of function definition, function declaration and parameter passing.
3. Master the use of random number generation mechanism to achieve simulation technology
4. Write C++ programs: function calls and recursive calls.
5. Understand the concept of scope and how identifiers are restricted to specific areas of the program.
6. Understand the difference between a pass by value call and a pass by reference call in parameter passing, when to apply a pass by value call and when to apply a pass by reference call.

Experiments

● Ex1(p.224—6.23)

(Square of any character) Write a function that displays at the left margin of the screen a solid square of whatever character is contained in character parameter fillCharacter while the side is specified in the integer parameter side.

For example: side=5, fillCharacter='*'

```
*****
*****
*****
*****
*****
```

● Ex2(p.224—6.28)

(Perfect Numbers) An integer is said to be a perfect number if the sum of its factors, including 1 (but not the number itself), is equal to the number. For example, 6 is a perfect number, because $6 = 1 + 2 + 3$. Write a function perfect that determines whether parameter number is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect. Challenge the power of your computer by testing numbers much larger than 1000.

Output:

```

Perfect integers between 1 and 1000:
6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14
496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248
Press any key to continue_

```

• Ex3(p.224—6.29)

(PrimeNumbers) An integer is said to be prime if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.

1. Write a function that determines whether a number is prime.
2. Use this function in a program that determines and prints all the prime numbers between 2 and 10,000. How many of these numbers do you really have to test before being sure that you have found all the primes?
3. Initially, you might think that $n/2$ is the upper limit for which you must test to see whether a number is prime, but you need only go as high as the square root of n . Why? Rewrite the program, and run it both ways. Estimate the performance improvement.

Output:

```

The prime numbers from 1 to 10000 are:
 2   3   5   7   11  13  17  19  23  29
31  37  41  43  47  53  59  61  67  71
73  79  83  89  97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541

9293 9311 9319 9323 9337 9341 9343 9349 9371 9377
9391 9397 9403 9413 9419 9421 9431 9433 9437 9439
9461 9463 9467 9473 9479 9491 9497 9511 9521 9533
9539 9547 9551 9587 9601 9613 9619 9623 9629 9631
9643 9649 9661 9677 9679 9689 9697 9719 9721 9733
9739 9743 9749 9767 9769 9781 9787 9791 9803 9811
9817 9829 9833 9839 9851 9857 9859 9871 9883 9887
9901 9907 9923 9929 9931 9941 9949 9967 9973
Total of 1229 prime numbers between 1 and 10000.
Press any key to continue

```

• Ex4(p.224—6.33)

(Coin Tossing) Write a program that simulates coin tossing. The program should call a function flip that takes no arguments and returns 0 for heads and 1 for tails. Call the function 10, 100, 1000, 10000 and 100000 times and output the percentage of tails (or heads).

Lab6 Function and Recursion-II

Objective

1. To construct programs modularly from functions.
2. Master the method of function definition, function declaration and parameter passing.
3. Master the use of random number generation mechanism to achieve simulation technology
4. Write C++ programs: function calls and recursive calls.
5. Understand the concept of scope and how identifiers are restricted to specific areas of the program.
6. Understand the difference between a pass by value call and a pass by reference call in parameter passing, when to apply a pass by value call and when to apply a pass by reference call.

Experiments

● Ex1(p.226—6.36)

Write a recursive function $\text{power}(\text{base}, \text{exponent})$ that, when invoked, returns $\text{base}^{\text{exponent}}$

For example, $\text{power}(3, 4) = 3 * 3 * 3 * 3$. Assume that exponent is an integer greater than or equal to 1. Hint: The recursion step would use the relationship

$$\text{base}^{\text{exponent}} = \text{base} \cdot \text{base}^{\text{exponent} - 1}$$

and the terminating condition occurs when exponent is equal to 1, because $\text{base}^1 = \text{base}$

Output:

Enter a base and an exponent: 3 5
3 raised to the 5 is 243

● Ex2(p.226—6.37)

(Least Common Multiple: Recursive Solution) Write a recursive version of the function lcm from Fig 6.29. Compare this version with the iterative version developed in Exercise 6.31.

● Ex3(p.228—6.41)

(Recursive Greatest Common Divisor) The greatest common divisor of integers x and y is the largest integer that evenly divides both x and y . Write a recursive function gcd that returns the greatest common divisor of x and y , defined recursively as follows: If y is equal to 0, then $\text{gcd}(x, y)$ is x ; otherwise, $\text{gcd}(x, y)$ is $\text{gcd}(y, x \% y)$, where $\%$ is the modulus operator.

[Note: For this algorithm, x must be larger than y.]

Output:

Enter two integers: 30 10

Greatest common divisor of 30 and 10 is 10

● Ex4

Write function template TwoSmallest, and find the two minimum values of the three elements. Requirements:

- 1) For function template TwoSmallest:
 - a) the prototype of the function is `int TwoSmallest(T a, T b, T c, T &s1, T &s2);`
 - b) the parameters s1 and s2 of the function are the minimum value and the sub-small value obtained respectively;
 - c) the number of minimum values returned by the function (if the two minimum values are equal, the function returns 1)
- 2) design function template `void ShowTwoSmallest(int smallnum, T s1, T s2)`, output two minimum output, where: smallnum is the number of the minimum value;
- 3) in the main program, use menu selection and call function template to complete the comparison test of integer, real number, character and string, and output the results, such as:

```

*****
1.  int
2.  double
3.  char
4.  string
0.  exit
*****
Inout choice: 4
String1:abc
String2:abc
String3:123
The smallest number : 123
The second smallest number : abc
*****
1.  int
2.  double
3.  char
4.  string
0.  exit
*****
Inout choice: _

```

4) use the standard C++string class to process strings

5) the following test example can be used to verify the correctness of the program:

a) 1 2 -1

● Output:

The smallest number : -1

The second smallest number: 1

b) 'a' 'a' '0'

● Output:

The smallest number: 0

The second smallest number: a

c) "aaa" "aaa" "aaa"

● Output:

The smallest number: aaa

d) 3.2 -1.1 -1.1

● Output:

The smallest number: -1.1

The second smallest number: 3.2

Lab7 Array

Objectivities

1. Using rand to generate random numbers and using srand to seed the random-number generator.
2. Declaring, initializing and referencing arrays.
3. The follow-up questions and activities also will give you practice:
4. Remembering that arrays begin with subscript 0 and recognizing off-by-one errors.
5. Preventing array out-of-bounds errors.
6. Using two-dimensional arrays.

Experiments

- **Ex1 (Duplicate Elimination with array)**

Problem Description

Use a one-dimensional array to solve the following problem: Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, validate it and store it in the array only if it is not a duplicate of a number already read. After reading all the values, display only the unique values that the user entered. Provide for the “worst case” in which all 20 numbers are different. Use the smallest possible array to solve this problem.

Problem-Solving Tips

Compare every value input to all existing array elements. If it is a duplicate, set a flag variable to 1. This flag should be used to determine whether it is necessary to print the value. Use a counter variable to keep track of the number of elements entered into the array and the array position where the next value should be stored.

Sample output:

```

Enter 20 integers between 10 and 100:
10
5
Invalid number.
20
30
40
50
60
70
80
90
100
110
Invalid number.
10
Duplicate number.
11
22
33
44
55
66
77
88
99
45

The nonduplicate values are:
10 20 30 40 50 60 70 80 90 100 11 22 33 44 55 66 77 88 99 45

```

● Ex2(Dice Rolling)

Problem Description

Write a program that simulates the rolling of two dice. The sum of the two values should then be calculated. [Note: Each die can show an integer value from 1 to 6, so the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums]. The following table shows the 36 possible combinations of the two dice. Your program should roll the two dice {3600, 36000, 360000, 36000000} times. Use a one-dimensional array to tally the number of times each possible sum appears. Print the results in a tabular format. Also, determine if the totals are reasonable (i.e., there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7).

SUM	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Table. The 36 possible outcomes of rolling two dice

Output Sample:

	Sum	Total	Expected	Actual
2	999025	2.778%	2.775%	
3	2001526	5.556%	5.560%	
4	3000750	8.333%	8.335%	
5	3999635	11.111%	11.110%	
6	4997824	13.889%	13.883%	
7	5998453	16.667%	16.662%	
8	5001051	13.889%	13.892%	
9	4000069	11.111%	11.111%	
10	3002868	8.333%	8.341%	
11	1998632	5.556%	5.552%	
12	1000167	2.778%	2.778%	

请按任意键继续...

● Ex3(Bubble Sorting Algorithm)

Problem Description

In the bubble sort algorithm, smaller values gradually “bubble” their way upward to the top of the array like air bubbles rising in water, while the larger values sink to the bottom. The bubble sort makes several passes through the array. On each pass, successive pairs of elements are compared. If a pair is in increasing order (or the values are identical), we leave the values as they are. If a pair is in decreasing order, their values are swapped in the array.

Write a program that sorts an array of 10 integers using bubble sort.

Data items in original order

2 6 4 8 10 12 89 68 45 37

Data items in ascending order

2 4 6 8 10 12 37 45 68 89

Template

```
// bubblesort.cpp
// This program sorts an array's values into ascending order.
#include <iostream>
using std::cout;
using std::endl;

#include <iomanip>
#include <array>
using std::setw;

int main()
{
    const int arraySize = 10; // size of array a
    array<int, arraySize> a = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};
    int hold; // temporary location used to swap array elements
```

```

cout << "Data items in original order\n";
// output original array
for ( int i = 0; i < arraySize; i++ )
    cout << setw( 4 ) << a[ i ];
// bubble sort
// loop to control number of passes
/* Write a for header to loop for one iteration less than the size
of the array */
{
    // loop to control number of comparisons per pass
    /* Write a for header to iterate j from 0 and keep looping while j is less than arraySize - 1 */
    {
        // compare side-by-side elements and swap them if
        // first element is greater than second element
        /* Write an if statement to test if element j is greater than element j + 1 */
        {
            /* Write code to swap the values in element j and element j + 1, using hold as temporary
            storage */
        } // end if
    } // end for
} // end for
cout << "\nData items in ascending order\n";
// output sorted array
for ( int k = 0; k < arraySize; k++ )
    cout << setw( 4 ) << a[ k ];
cout << endl;
return 0; // indicates successful termination
} // end main

```

Fig. L 7.3 | bubblesort.cpp.

Problem-Solving Tips

1. Each “bubbling” pass through the array brings one element, the i^{th} up to its correct position. This means that the program will require $arraySize - 1$ passes through the array to sort the entire array.
2. Each bubbling pass will look at each pair of adjacent elements and swap them if they are not already in sorted order.
3. To swap two elements, the value of one element will have to be stored in a temporary storage variable while the value of the other element is placed in the first, and then the second element can be replaced with the temporary storage value.

Follow-Up Question and Activity

1. This bubble sort algorithm is inefficient for large arrays. Make the following simple modifications to improve the performance of the bubble sort:
 - a) After the first pass, the largest number is guaranteed to be in the highest-numbered element of the array; after the second pass, the two highest numbers are “in place,” and so on. Instead of making nine comparisons on every pass, modify the bubble sort to make eight comparisons on the second pass, seven on the third pass, and so on.
 - b) The data in the array may already be in the proper order or near-proper order, so why make nine passes if fewer will suffice? Modify the sort to check at the end of each pass if any swaps have been made. If none have been made, then the data must already be in the proper order, so the program should terminate. If swaps have been made, then at least one more pass is needed.

- **Ex4(Finding Prime Numbers)**

Problem Description

A prime integer is any integer greater than 1 that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:

1. Create an array with all elements initialized to 1 (true). Array elements with prime subscripts will remain. All other array elements will eventually be set to zero. You will ignore elements 0 and 1 in this exercise.
2. Starting with array subscript 2, every time an array element is found whose value is 1, iterate through the remainder of the array and set to zero every element whose subscript is a multiple of the subscript for the element with value 1. For array subscript 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for array subscript 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (subscripts 6, 9, 12, 15, etc.); and so on.
3. When this process is complete, the array elements that are still set to one indicate that the subscript is a prime number. These subscripts can then be printed. Write a program that uses an array of 1000 elements to determine and print the prime numbers between 2 and 999.

Output:

2	3	5	7	11	13	17	19
23	29	31	37	41	43	47	53
59	61	67	71	73	79	83	89
97	101	103	107	109	113	127	131
137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223
227	229	233	239	241	251	257	263
269	271	277	281	283	293	307	311
313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457
461	463	467	479	487	491	499	503
509	521	523	541	547	557	563	569
571	577	587	593	599	601	607	613
617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719
727	733	739	743	751	757	761	769
773	787	797	809	811	821	823	827
829	839	853	857	859	863	877	881
883	887	907	911	919	929	937	941
947	953	967	971	977	983	991	997

- **Ex5 (Eight Queens P275-7.24)---附加题**

Lab8 Pointers and Pointer-based Strings

Objectives

1. To understand the concept of pointer, master pointer-based indirect organization method of data.
2. To master the basic of declare and use pointer in C++, the basic operation of pointer.
3. To master the relationship between pointer and array, the pointer-based array operation.
4. To master the relationship between pointer and string, the pointer-based string operation.
5. To master the use of standard string library fuction.
6. To master the relationship between pointer and function, the pointer-based function operation.

Experiments

● Ex1

1. To run the following program and analysis its result.

```
#include <iostream>
using std::cout;
using std::endl;

void main()
{
    int a[]={ 1,3,5,7,9};
    int *p[]={ a,a+1,a+2,a+3,a+4};
    int *p1=a;

    cout<<"Test 1:-----"<<endl;
    cout<<a[4]<<"\t"<<*(a+2)<<"\t"<<*p[1]<<**(p+1)<<"\t"<<*(p+1)+a[2]<<"\t"
        <<*(p+4)-*(p+0)<<"\t"<<*(a+3)%a[4]<<endl;

    cout<<"Test 2:-----"<<endl;
    cout<<*(&p1)<<"\t"<<*(p1+2)<<"\t"<<*(&p1+4)<<*(p1+8)<<"\t"<<endl;

    cout<<"Test 3:-----"<<endl;
    cout<<sizeof(p1)<<"\t"<<sizeof(*p1)<<"\t"<<&p1<<"\t"<<*p1<<endl;
}
```

2. To declare a function to swap two integer as following every prototype, and analysis carefully their executing.

```
void swap(int a, int b); // declare a local identifier temp
void swap(int a, int b); // declare a local identifier temp*
```

```
void swap(int* a, int* b); // declare a local identifier temp
void swap(int* a, int* b); // declare a local identifier temp*
void swap(int& a, int& b); // declare a local identifier temp
```

3. To declare a function to get the sum of two integer as following every prototype, and analysis carefully their executing.

```
int getsum(int a, int b); // declare a local identifier temp, return temp
int* swap(int a, int b); // declare a local identifier temp, return the address of temp
void swap(int a, int b); // declare a local identifier temp*, return temp
void swap(int a, int b); // declare a local identifier temp*, return the content of temp
void swap(int a, int b); // declare a local identifier temp&, return temp
```

● Ex2

At first, we declare a 3-by-5 two-dimensional array. Then transpose it and output result by pointer-based operation.

● Ex3

Store two user input strings in two character arrays. Connect them, change the punctuation(标点) and convert the lowercase letters into uppercase letters for output as follows:

(注意: 不允许使用字符串标准库函数, 只能使用字符数组)

Please input two strings:

Hello C++!

Happy New Year, 2019!

Output:

HELLO C++, HAPPY NEW YEAR, 2019!

● Ex4

Write a function **paren_check()** to check the parentheses (小括号) in an arithmetic expression. When scanning the arithmetic expression from the first character to the last one, the number of right parentheses is **equal** to the number of left parentheses. If this condition is satisfied, **paren_check()** will return 1, otherwise return 0.

For example, given the expression $((a+1)*b/(c+d))$, **paren_check()** will return 1. However, given the expression $((a+1)*b/c+d)$, **paren_check()** will return 0.

Use a char array to save the arithmetic expression.

The prototype of **paren_check()** is defined as follows,

```
bool paren_check(char* expr);
```