## 实验五 Function and Recursion

## 学号: 09017423 姓名：杨彬

### 实验目的

Objective 1．掌握模块化构建程序的方法

2．掌握函数定义、函数声明、参数传递的方法

3．掌握利用随机数生成机制实现模拟技术

4．编写C++程序：函数调用和递归调用

5．理解作用域的概念，以及标识符是如何被限制在程序的特定区域中

6．理解参数传递中传值调用与传引用调用之间的区别，何时应用传值调用，何时应用传引用调用

### 二、实验内容

Exp1（英文）Use techniques similar to those developed in 6.22 and 6.23 to produce a program that graphs a wide range of shapes.

（中文）分别用三个函数实现绘制正方形(square)、菱形(diamond)、三角形(triangle)，在main函数中用循环结构嵌套switch结构实现对各种图形函数的调用测试。

Exp2（习题6.29）：

问题描述 (英文) (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.

(中文) 一个完全数是这样的一个整数，它的所有因子（包含1但不包含该数本身）的和恰好等于该数本身。举个例子，6是一个完全数，因为6=1+2+3。编写一个函数perfect(bool类型返回值)，用它来判断该函数的形式参数是否为一个完全数。在程序中调用该函数，找出1至1000这个范围内所有的完全数并打印出来。并且将这些找出来的完全数的因子打印出来，用来确认该数确实是一个完全数。你可以通过测试数值在1000以上的完全数来挑战

Exp33、（习题6.30）：

问题描述 (英文) (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. （中文）（质数问题）一个质数是这样的一个整数，它只能被1和该数本身整除。举个例子，2，3，4，7是质数，而4，6，8，9不是质数。 1.编写一个函数来判断一个数是否为质数。 2.在程序中调用这个函数，找出2至10000这个范围内所有的质数并且打印出来。在你确认找出所有质数之前你测试了多少次呢？ 3.最初，你可能认为判断一个数是否为质数的测试次数上限为n/2，但实际你需要的测试次数的上限只是n的平方根。这是为什么呢？重新编写这个程序，分别在这两种方式下运行，并观察出性能的改进。

Exp4

4、（习题6.32）：

问题描述 (英文) The greatest common divisor (GCD) of two integers is the largest integer that evenly divides each of the numbers. Write a function GCD that returns the greatest common divisor of two integers. (中文) 两个整型数的最大公约数(The greatest common divisor, GCD)定义为最大的整除这两个整数的数。编写gcd函数，返回两个整型数的最大公约数。

Exp5

5、（习题6.40）：

问题描述 (英文) Write a recursive function power ( base, exponent ) that, when invoked, returns base exponent For example, 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 base exponent = base · base exponent - 1 and the terminating condition occurs when exponent is equal to 1, because base1 = base （中文）编写一个递归函数power ( base, exponent )，当函数被调用时，返回 base exponent 例如，power( 3, 4 ) = 3 \* 3 \* 3 \* 3。假设指数exponent是一个大于或等于0的整数。提示：递归步骤使用关系式 base exponent = base · base exponent - 1 和当指数exponen=1时，递归终止条件发生，因为此时base1 = base。

Exp6

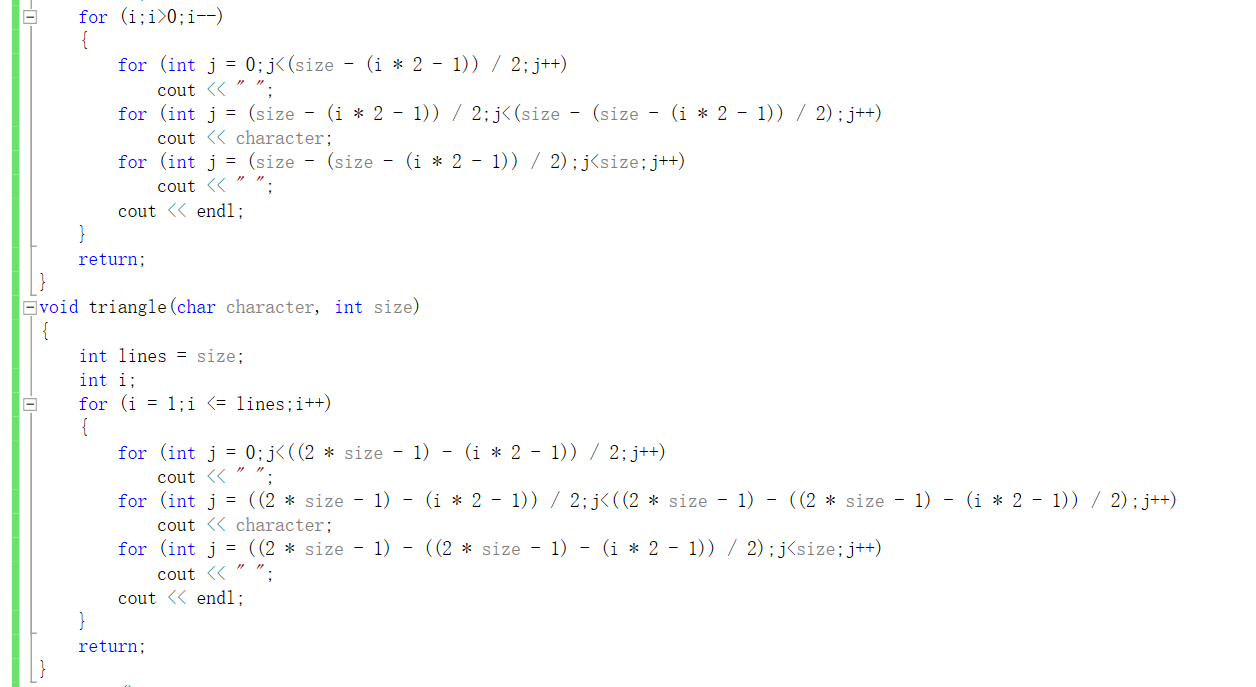
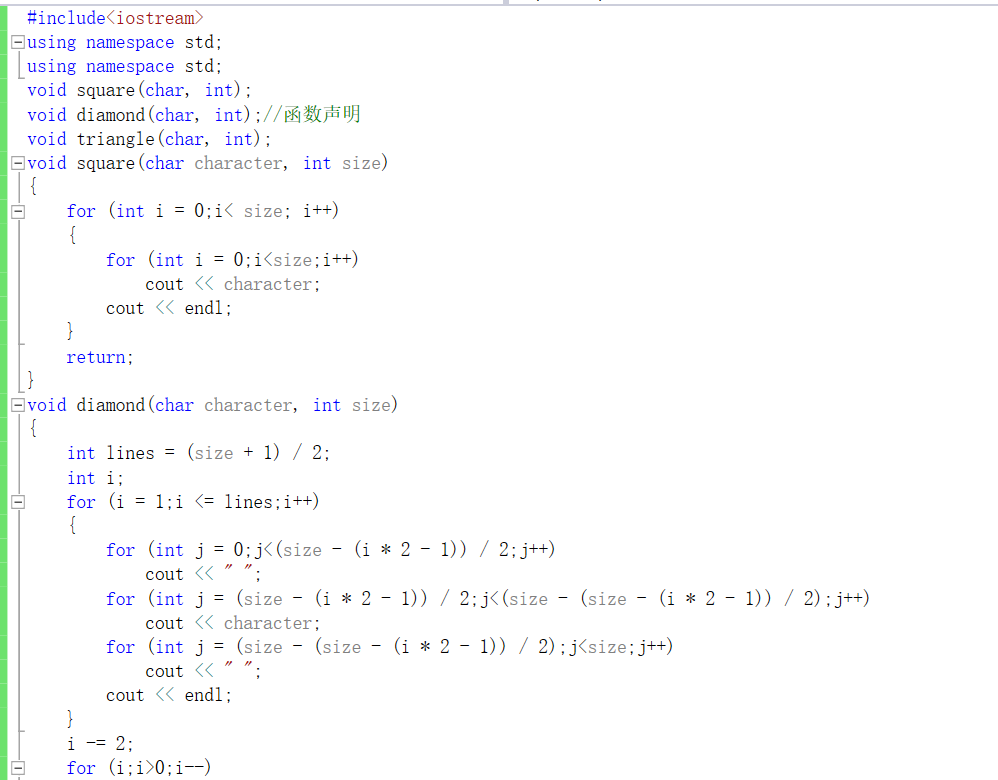
、（习题6.40）：

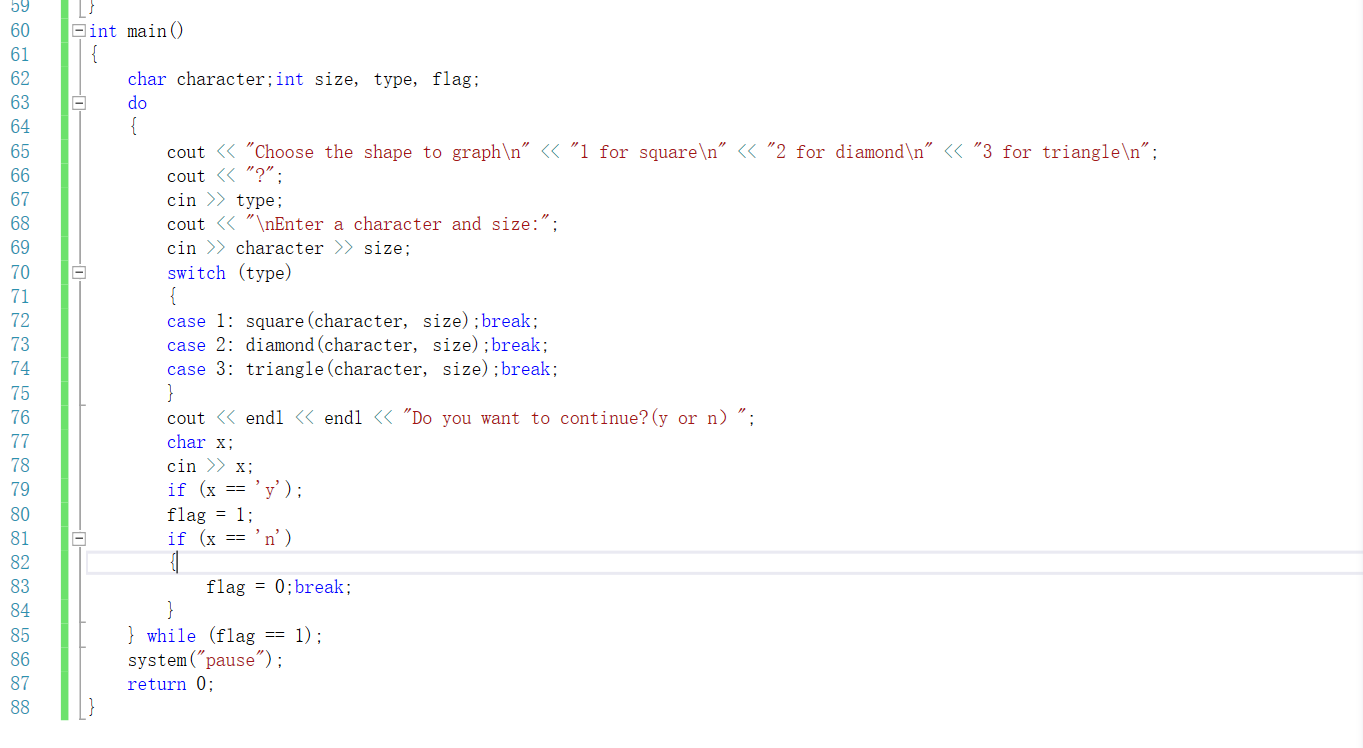
问题描述 (英文) (Fibonacci Series) The Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, ... begins with the terms 0 and 1 and has the property that each succeeding term is the sum of the two preceding terms. (a) Write a nonrecursive function fibonacci( n ) that calculates the nth Fibonacci number. (b) Determine the largest int Fibonacci number that can be printed on your system. Modify the program of part (a) to use double instead of int to calculate and return Fibonacci numbers, and use this modified program to repeat part (b). （中文）斐波那契数列 0, 1, 1, 2, 3, 5, 8, 13, 21, ... 从数值0和1开始，具有每一项斐波那契数列的值是前两项斐波那契数列的值之和的性质。（a）编写一个非递归函数fibonacci( n )计算第n项斐波那契数列的值。 （b）确定能打印输出在你系统中的最大的斐波那契数列的整型值。修改（a）部分的程序，使用double类型的值代替int型的值来计算和返回斐波那契数列的值，并且使用这修改过的程序来实现（b）部分的功能。

Exp 7

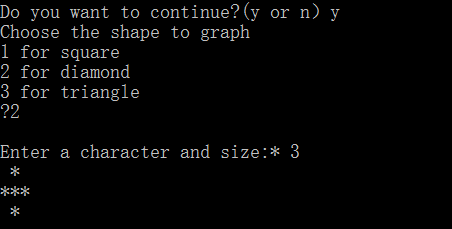
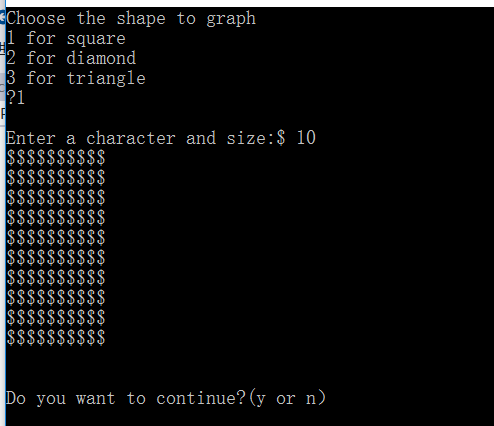
(英文)(Recursive Greatest Common Divisor) The greatest common divisor of integers x and y is the largest integer that evenly divides both x and y. Write a recursive function gcd that returns the greatest common divisor of x and y, defined recursively as follows: If y is equal to 0, then gcd( x, y ) is x; otherwise, gcd( x, y ) is gcd( y, x % y ), where % is the modulus operator. [Note: For this algorithm, x must be larger than y.] （中文）（递归法求最大公约数）整数x和y的最大公约数是能同时被x和y整除的最大整数。编写一个函数gcd（），返回x和y的最大公约数，定义递归如下：假如y=0，gcd( x, y )=x；否则，gcd( x, y )= gcd( y, x % y )，这里的%是求模运算符。[注：对于此算法，x必须大于y。]

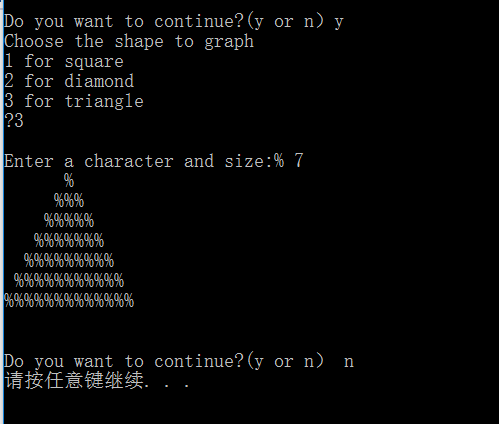
### 三、实验代码及结果：

Ex1

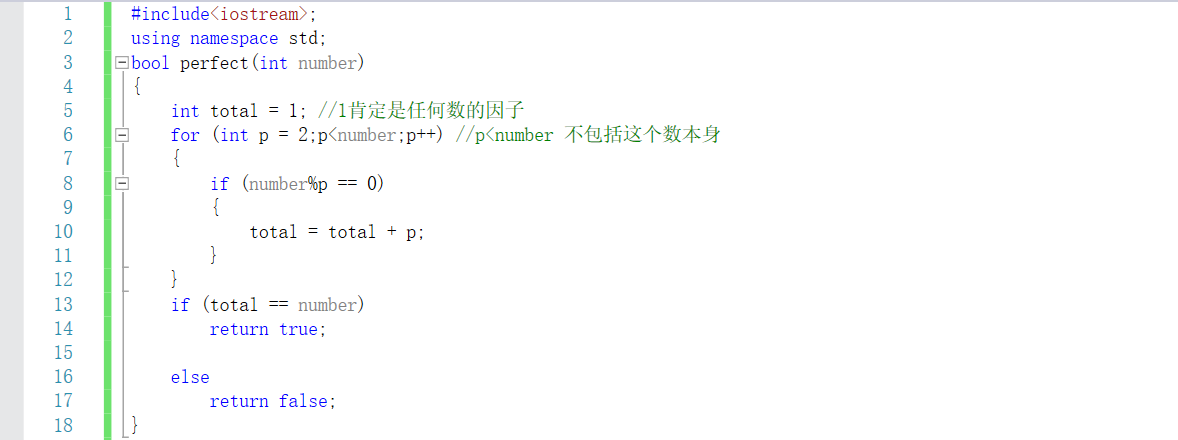


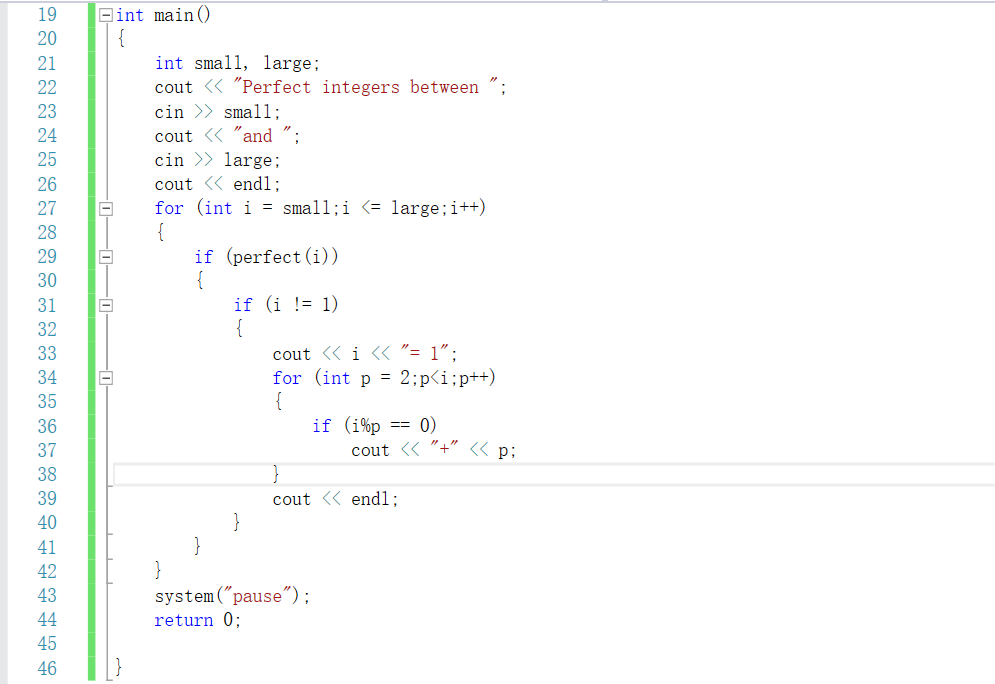
结果

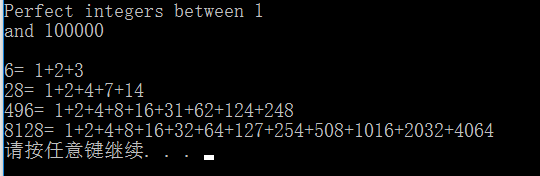
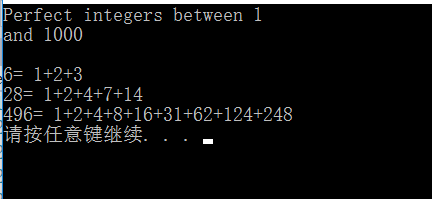




Exp2

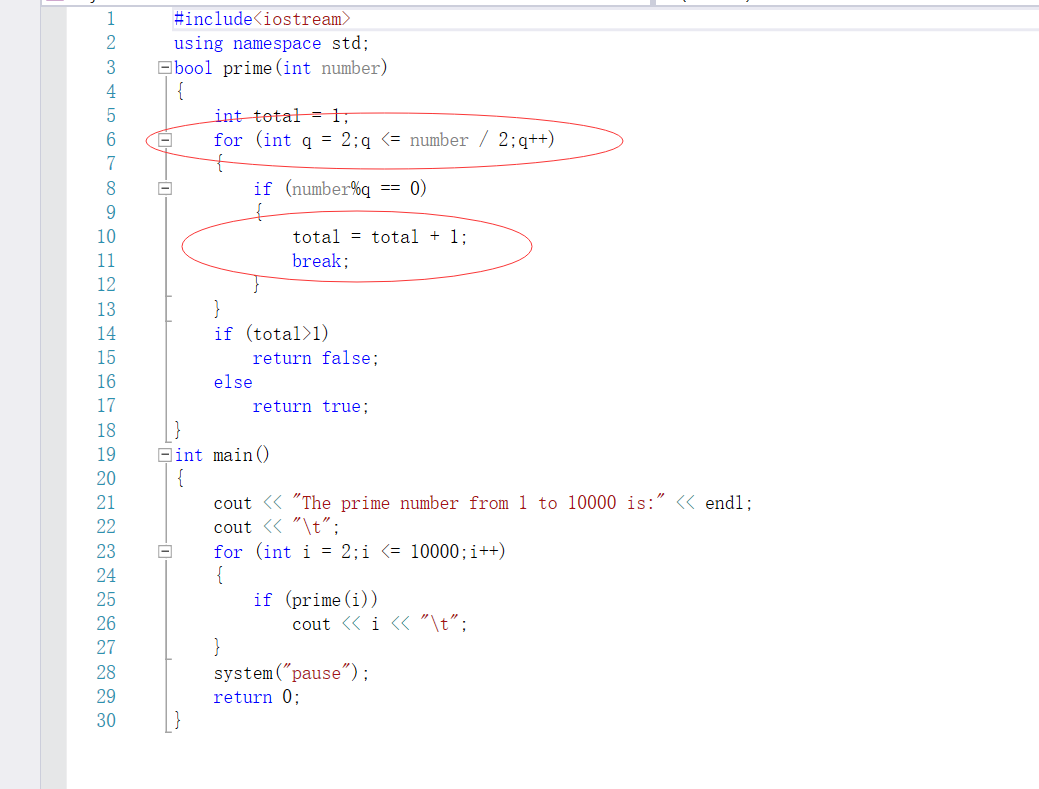


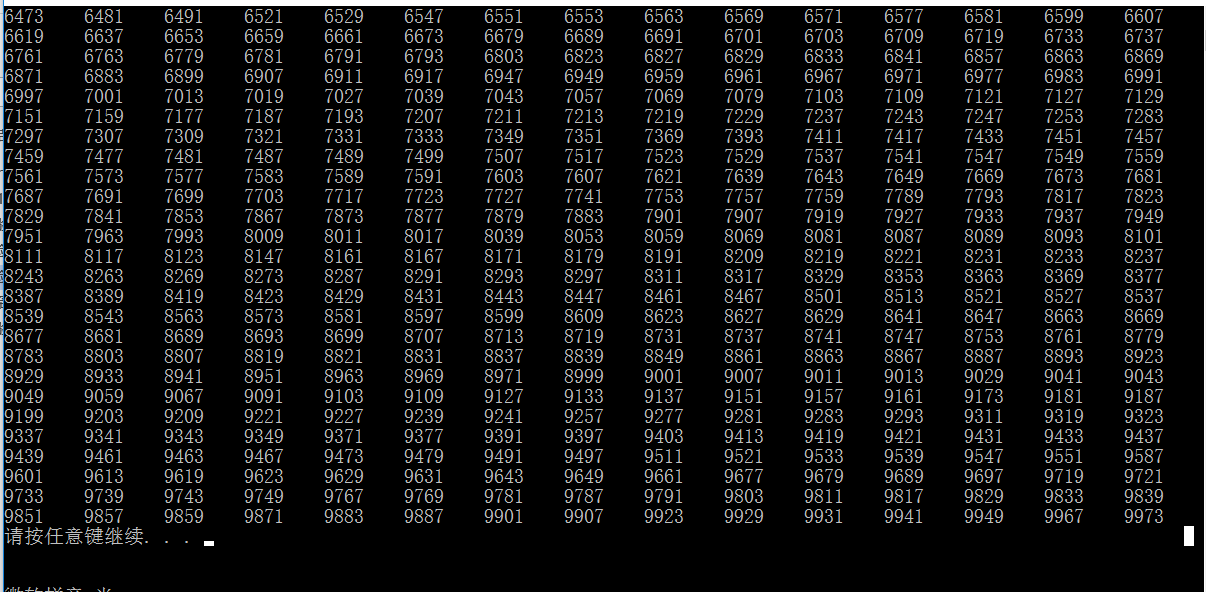


结果

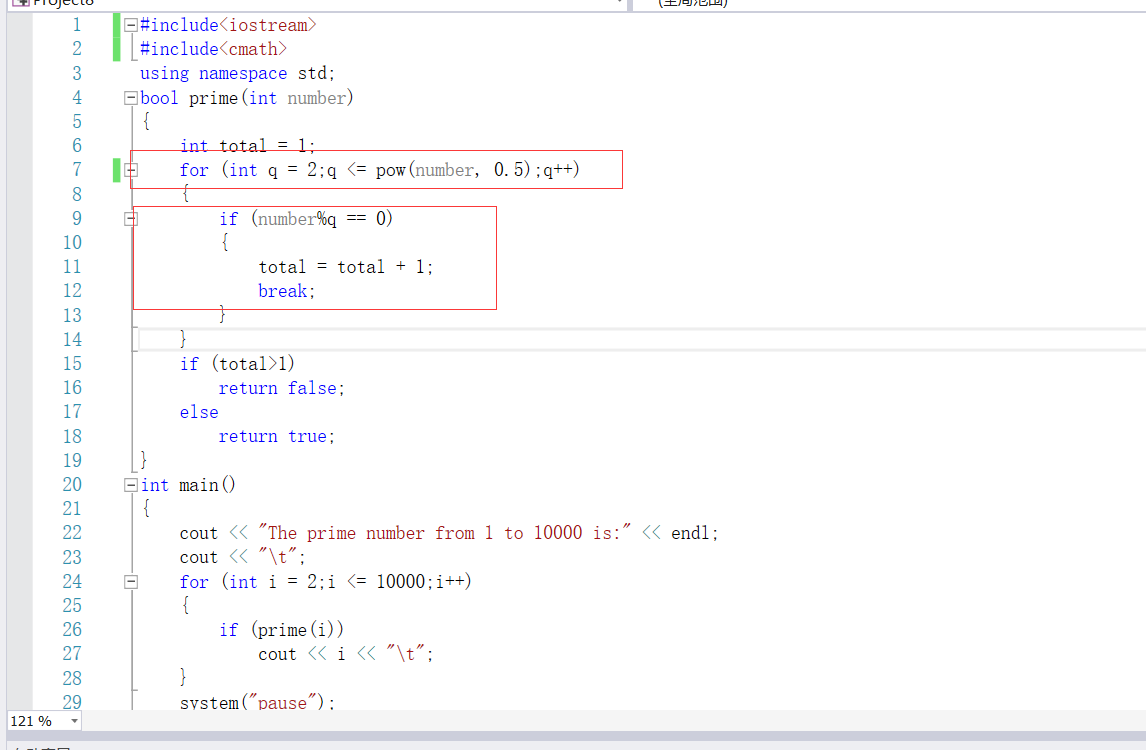
Exp3

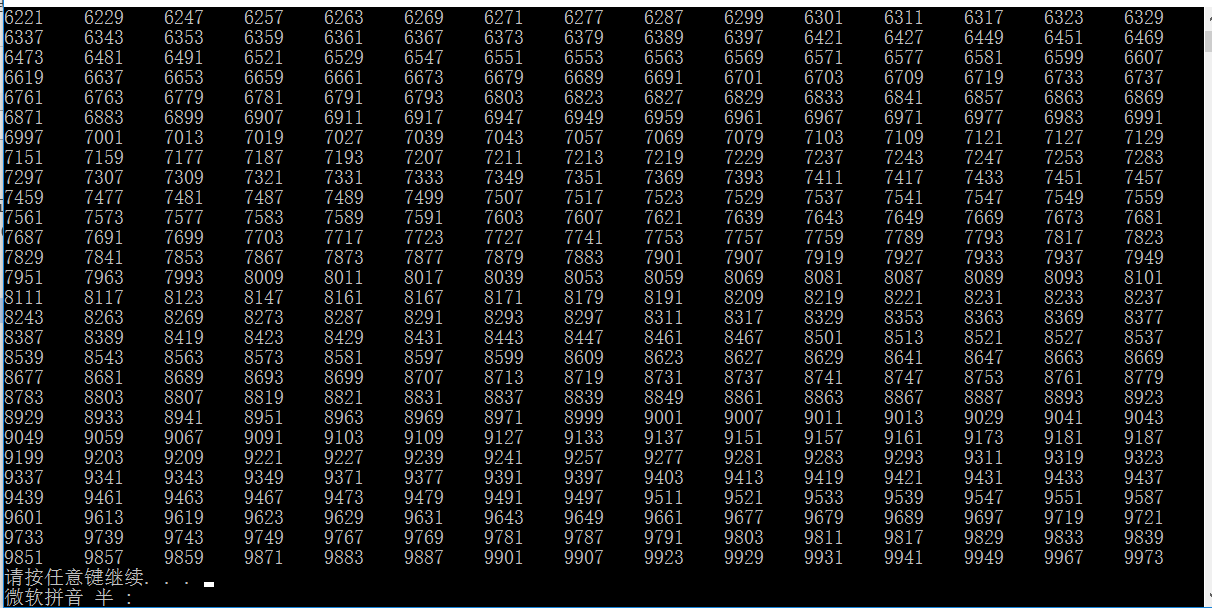
1. 版本1



结果

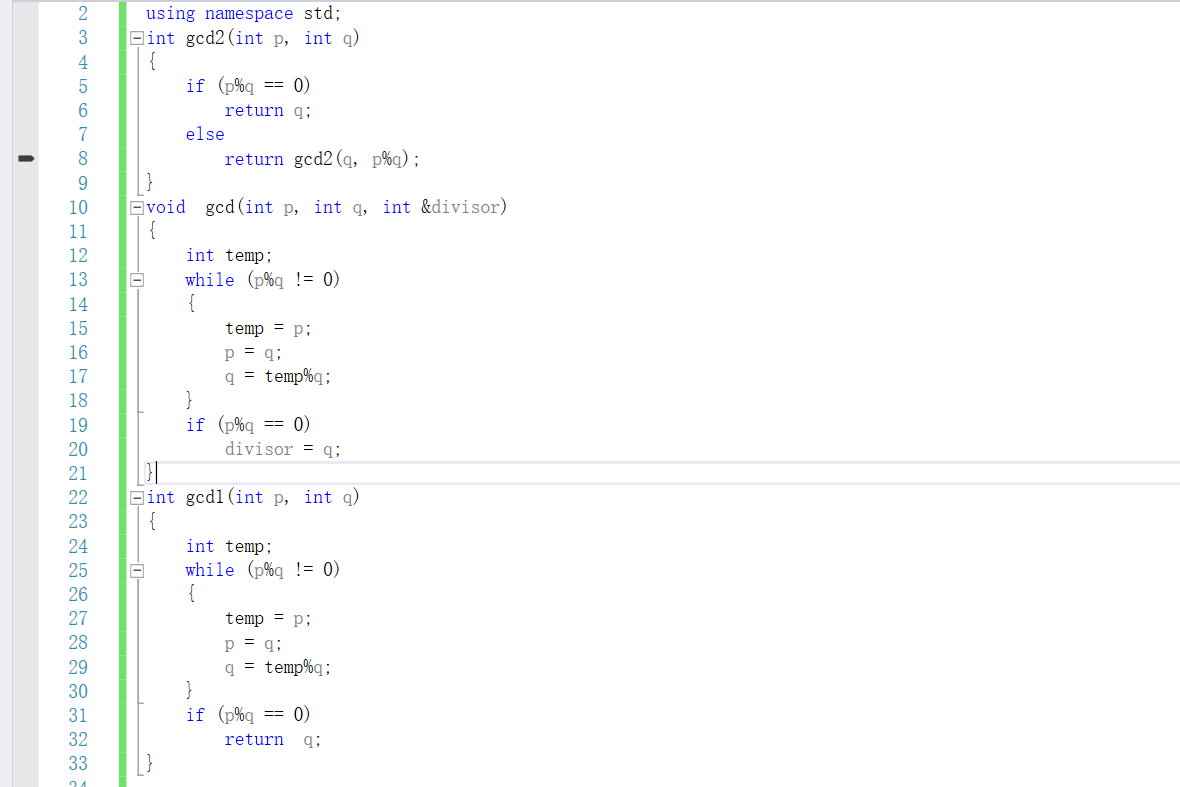
版本2

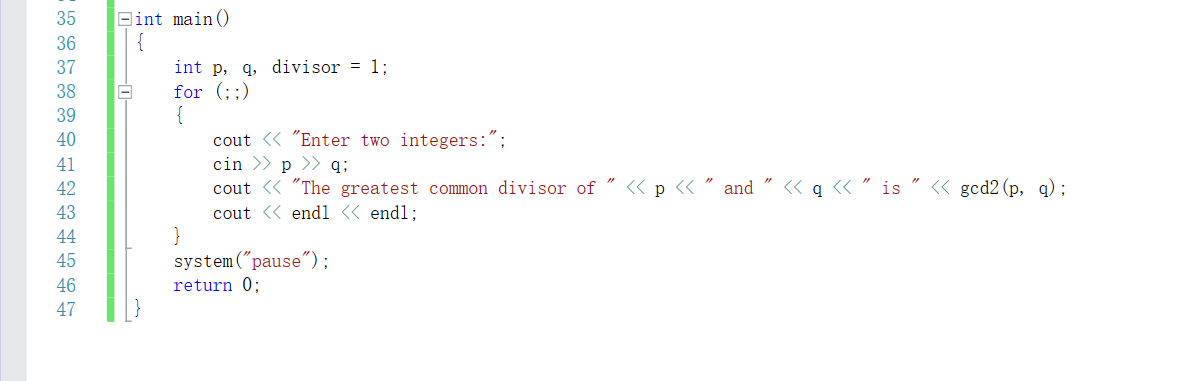


结果

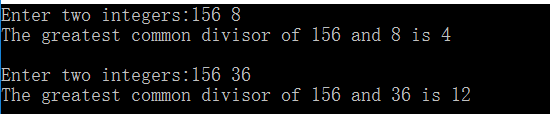
Exp 4

源代码



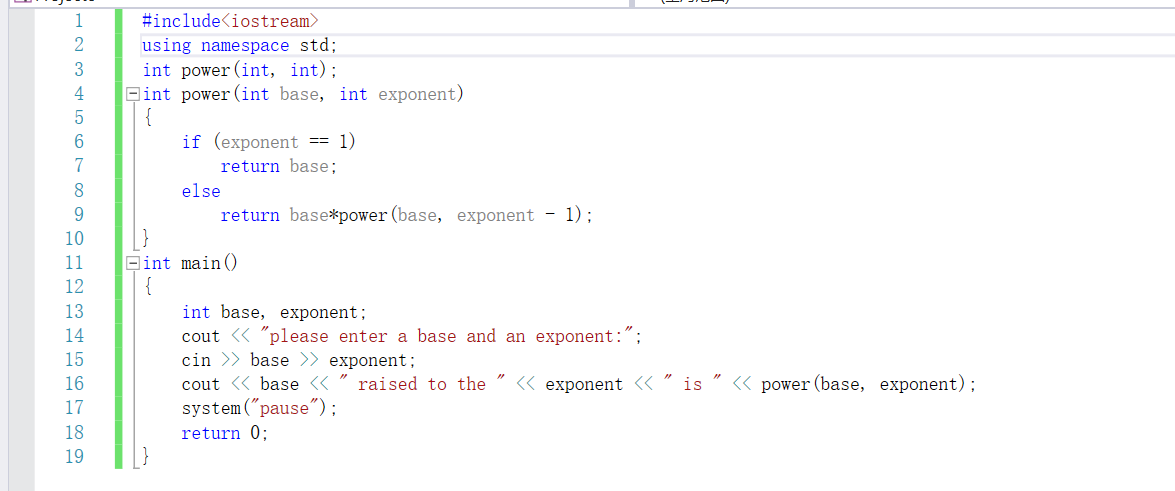


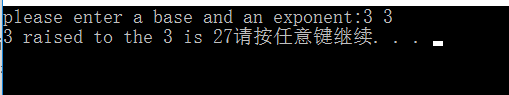
结果



Exp 5

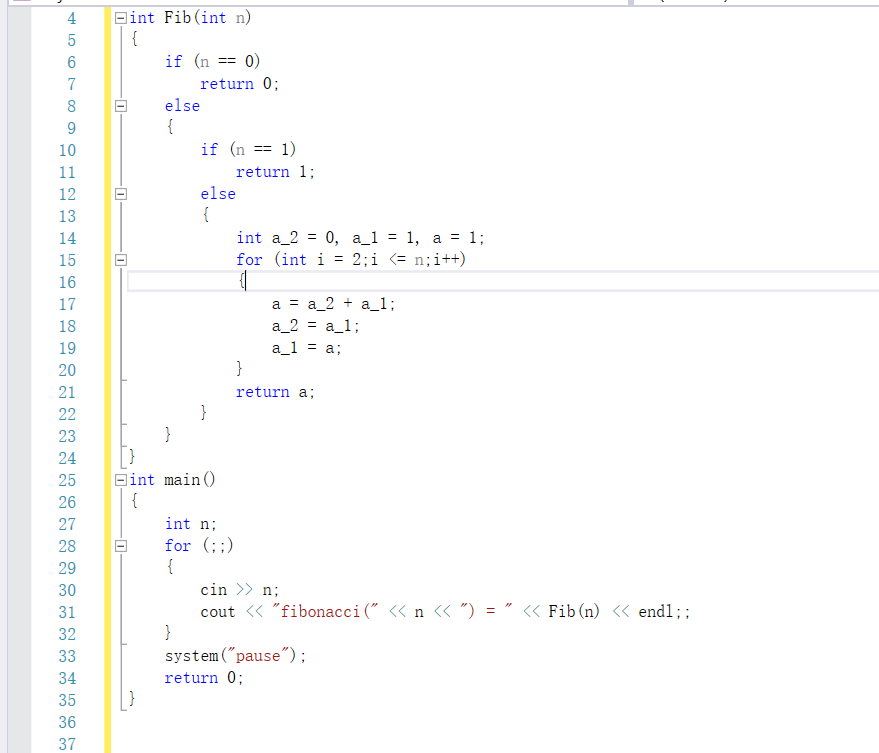
源代码



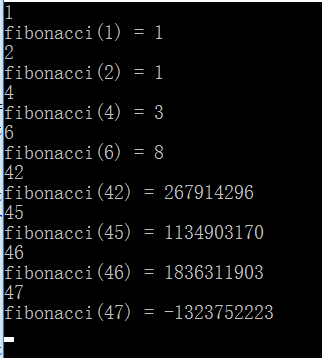


Exp6

源代码 6.1 int 型

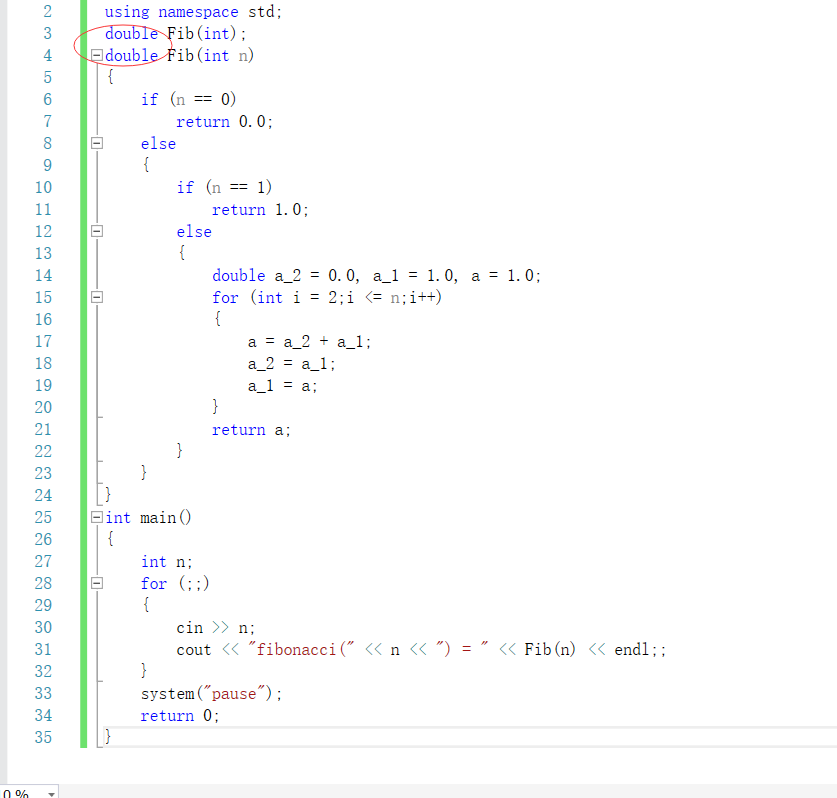


结果

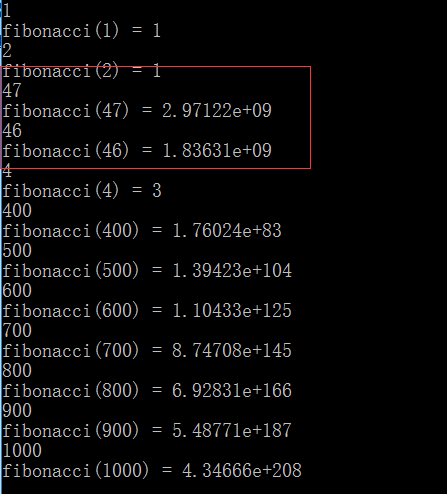
（第46个数最大）

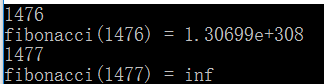
Double版本

源代码

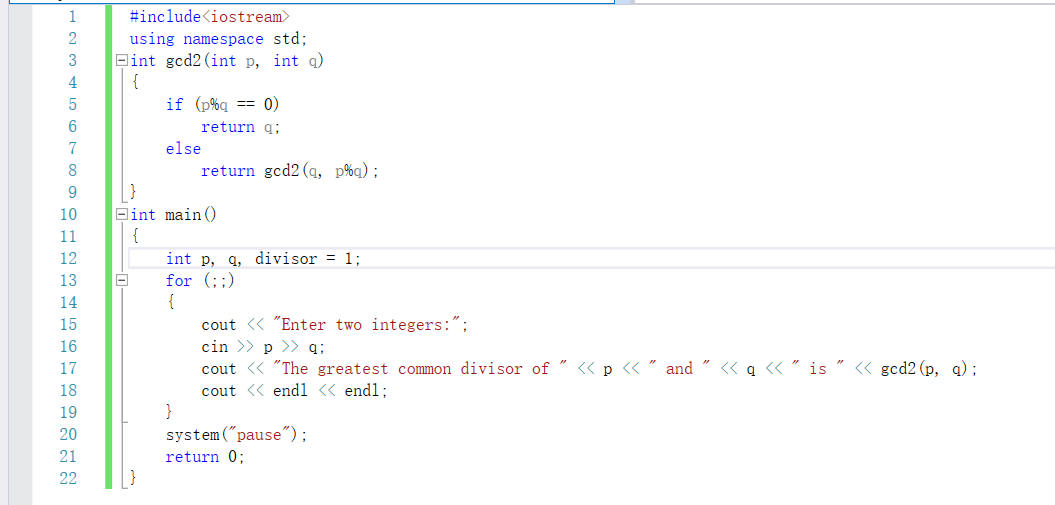


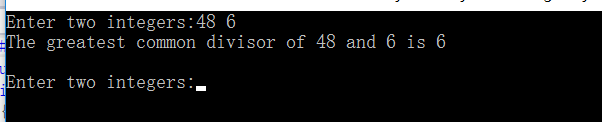
结果



（最大值）

Exp7





### 遇到的问题及解决过程

Exp1：nothing

Exp2:1一开始我的程序得到的结果是程序把所有的数极其因子的和都输出了出来。

2检查了程序发现我把输出部分放在了函数体内部。

3分析查找资料后知道了，在if（perfect(i)）的判断条件中，及时结果为加=假，但是函数perfect(i)仍然会被执行，也就是说函数内部的cout语句仍然还是被执行了。这就导致了不管是不是完全数，都会被打印出来。

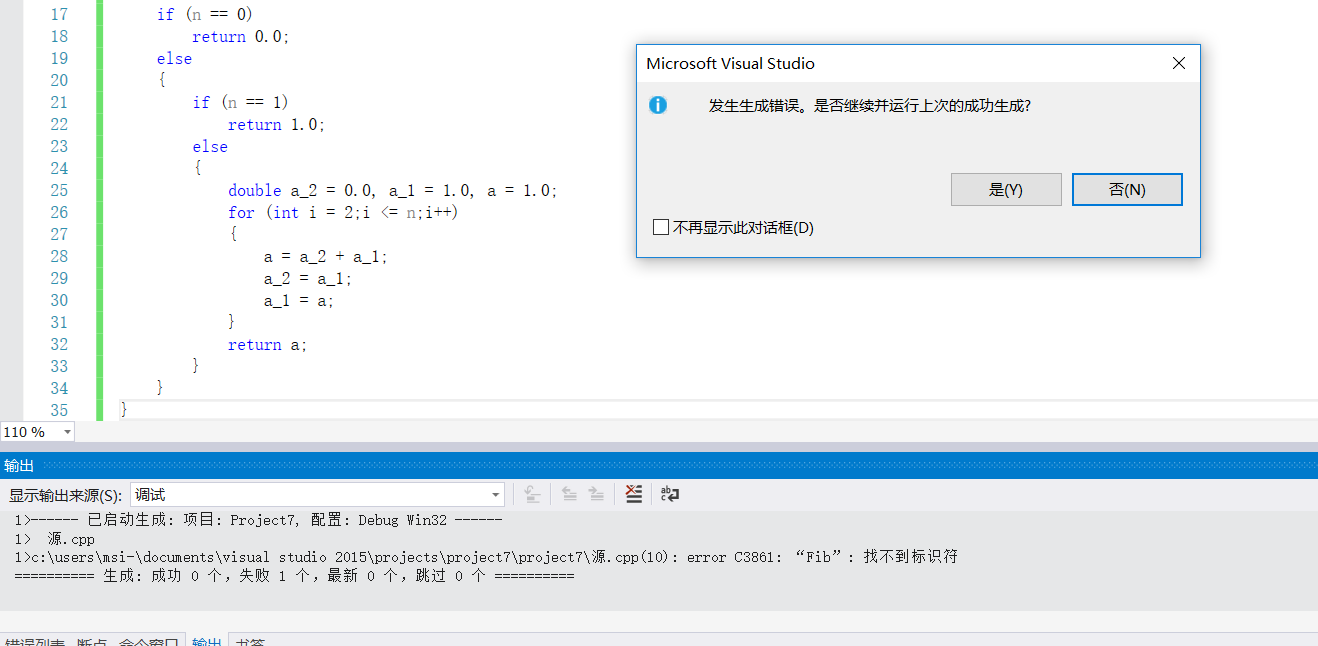
4最后我将输出语句放在perfect函数的外部，perfect函数仅仅作为判断是否输出这样就解决了这个问题。

Exp3:nothing

Exp4nothing

Exp5nothing

Exp6:这个实验中一开始我将Feb这个函数没有声明直接定义在main函数之后。结果导致错误



经过查书，我知道了如果一个函数要将其定义当成函数原型来使用的话，那么这个函数必须出现在所有的调用它的语句之前。否则就需要在一开始进行函数声明。

因此我加上函数声明之后就解决了问题。

Exp7 nothing

### 实验体会

Exp1: 打印图形的关键就是要弄清楚图形的规律，不如符号以及空格和循环语句中的count变量的函数关系。通过函数的方式，我第一次实践了模块化编程，函数可以节省大量的代码。让程序变得简洁而且明了，提高了程序的可读性

Exp2:函数只要被调用了不管是在哪里（在判断语句的条件部分）函数都会被执行一次（比如函数内的cout语句）而与判断结果的对错无关。在寻找一个数的因数的时候有一种方法就是可以采用%运算符，然后通过穷举法来找到所有的因子。

Exp3:这里因为一个数的因子要么是其平方数，要么是成对存在的，所以在控制调教中，只需要用这个数的平方根即可。这样可以减少运算量，提升性能。

Exp4 这里我采用了三种方法来实现这个算法，1 递归gcd2 2 迭代 gcd1 3通过引用来传递参数gcd 经过实验三种方法都可以的出结果。我得出了一下结论，1几乎很多算法，能用递归实现的，都可以用迭代法来实现。2通过引用传递参数，可以改变参数的原始的值，3善用调试工具来跟踪，程序中变量的变化情况。

Exp5通过这个实验我编写了一个递归程序，初步理解了递归函数的写法，应该包含两个部分，一个是函数能做的事（return base 简单的部分 ）另一个是函数不能做的事(return function())，同时递归函数还应该要有边界。

Exp6 这个实验一开始先采用递归的方法，然后将递归转化成了迭代，因此我初步尝试了，采用递归的思想，用非递归（迭代）的方法解决问题。对于不同类型能够得到的最大的斐波拉系数的思考：导致这样的结果的原因就是 int型和double型在内存中占用的空间不同int 型占用32位 8个字节，而double 型占用64位16个字节，因此double型能储存的数据范围更大。（这对于本台电脑）。这就告诉我一定要针对不同的数据，选择相应的数据类型。

Exp7 递归一定要分为几个部分 1 分解（简单的和复杂的部分 return base;return function()）2 边界（停止递归的最简单的情况）3 整合