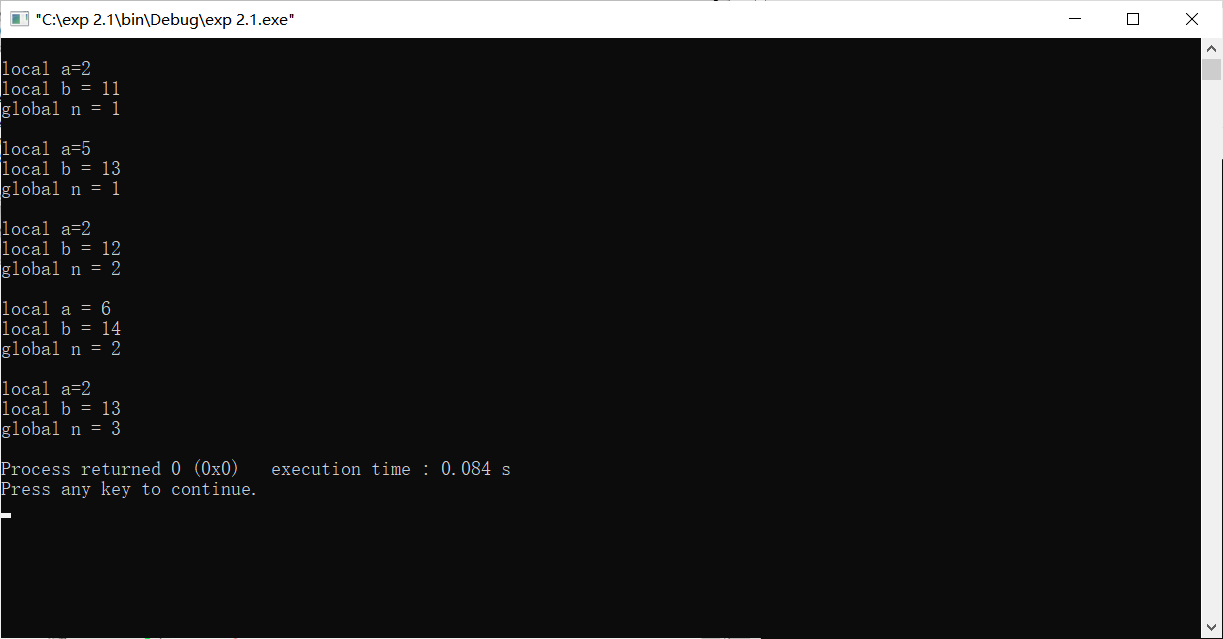
**C++实验二**

**李嘉康**

注：题目对应的源码在文件里分别对应exp 2.1,exp 2.2.........

### 题目1 分析程序运行结果



**运行结果如图：**

**得出结果的原因(以及实验要求)：**

A=5（主函数里的局部变量定为A），作为形参x调入func，a++等于2，b++等于11，n++=1，return 13给b；A++ = 6，再作为x调入func，a++ = 2，b作为func里的静态局部变量，b++ = 12，return 后为14.n++ = 2；

再调用func，b++ = 13，a++ = 2，n++ =3；

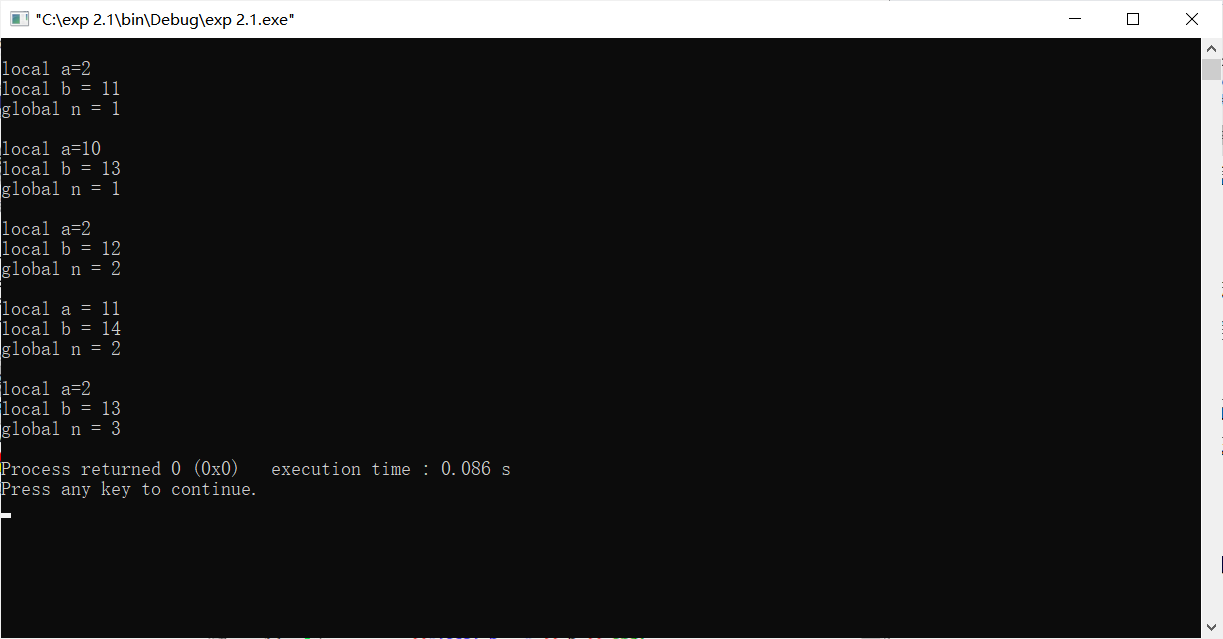
全局变量为n，局部变量有a,b等，局部静态变量有b（func里的）

局部变量有效范围只在作用域里，全局变量作用域在全局，任意处更改都能影响到它的值，局部静态变量每次在所在域里被调用的时候值不会被重新赋值。

**当调用func函数时，系统会给形参变量在栈中分配相应的内存空间，即整形变量,和指针变量，同时将实参赋值给对应的形参，即使定义了相同的全局变量或者指针变量。但他们代表的不是相同的内存空间，所以他们互不影响。**

**函数可能在程序中多次调用，因为形参需要用来存储实参对应的值，正是由于实参值具有不确定性，因此不能指定形参的值。函数的参数传递是赋值过程，则形参必须有特定的存储空间才能接受实参的值。在函数的实参只要满足赋值形参的赋值要求都可以。**

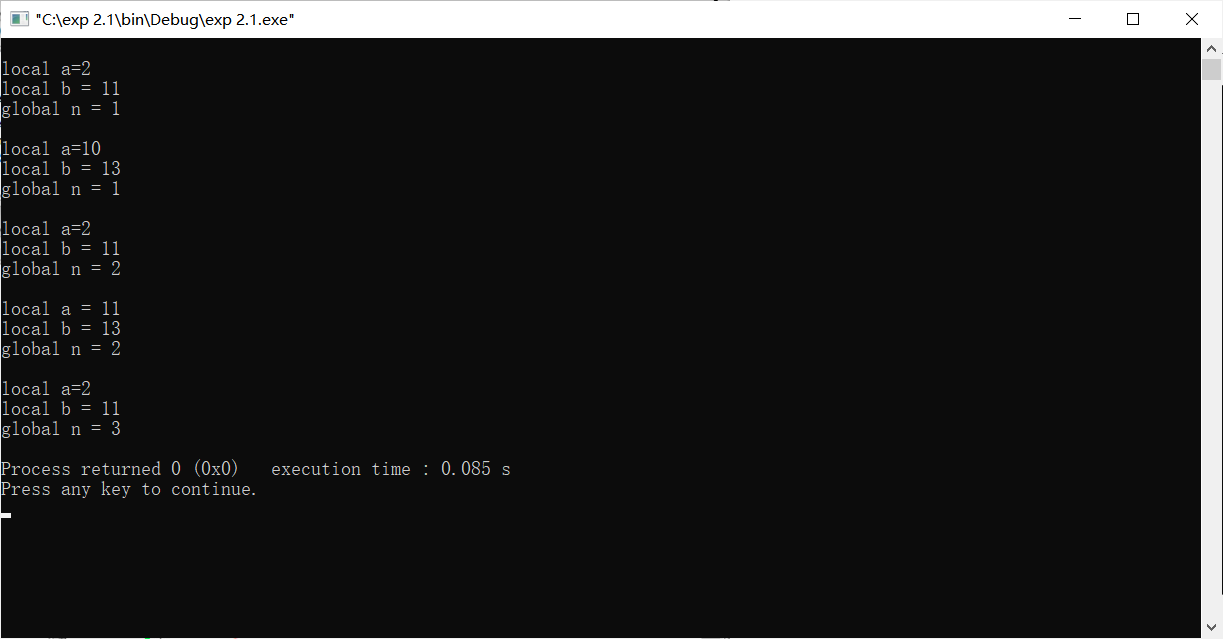
**测试数据：**



分析结果同上，只是a作为主函数里的局部变量的值改变了。

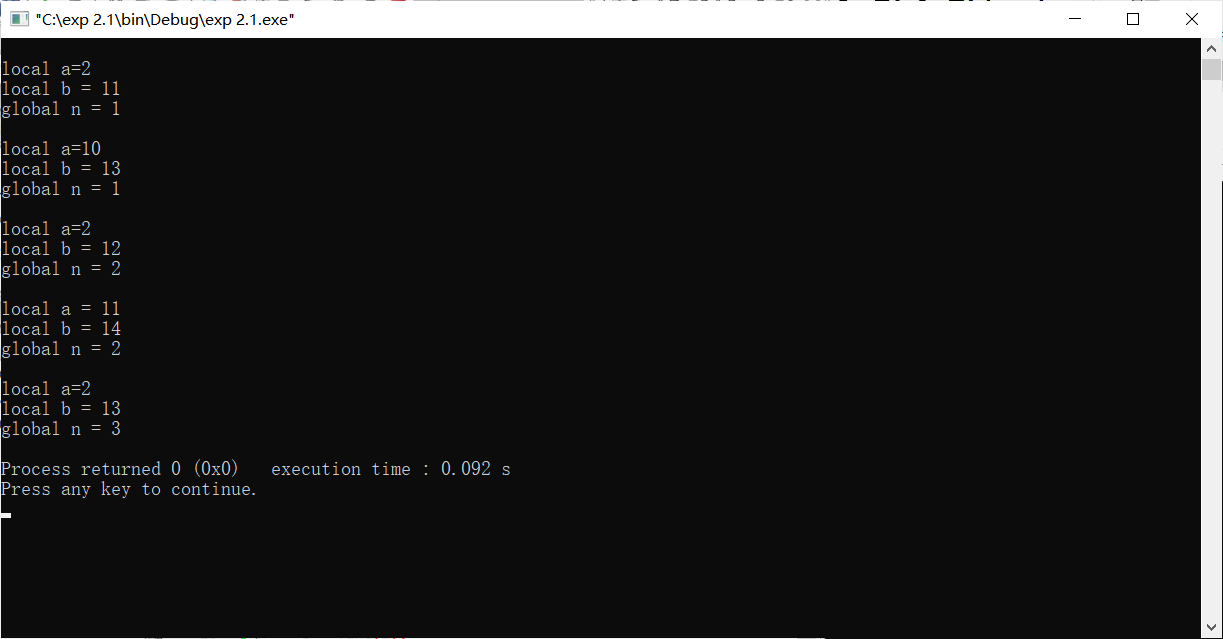
**思考与扩展：**

1.改为int后



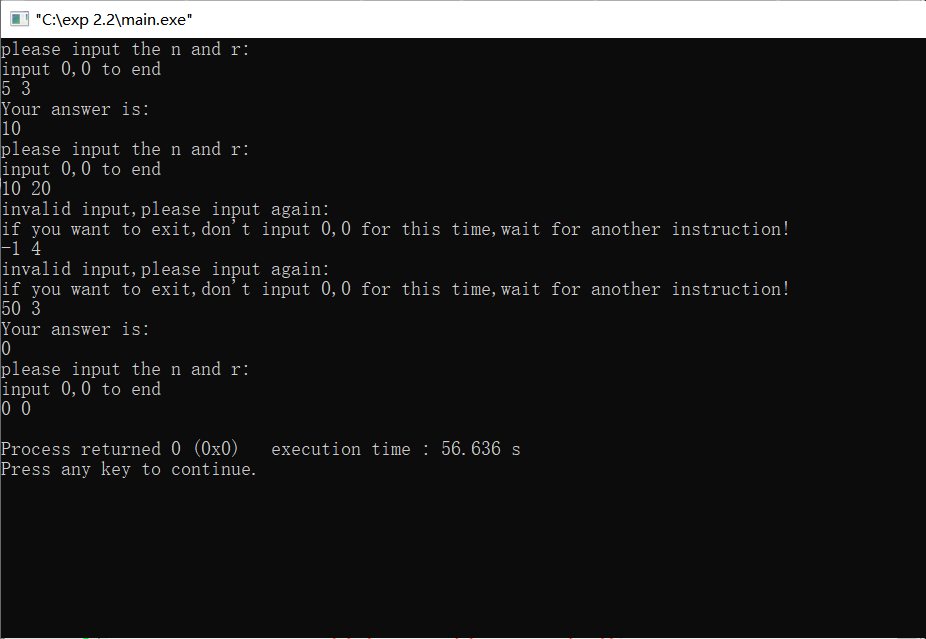
可以看到在func里打印出来的b值一直是11，没有改变，每次调用func都会被重新赋值。

1. 移到第三行后



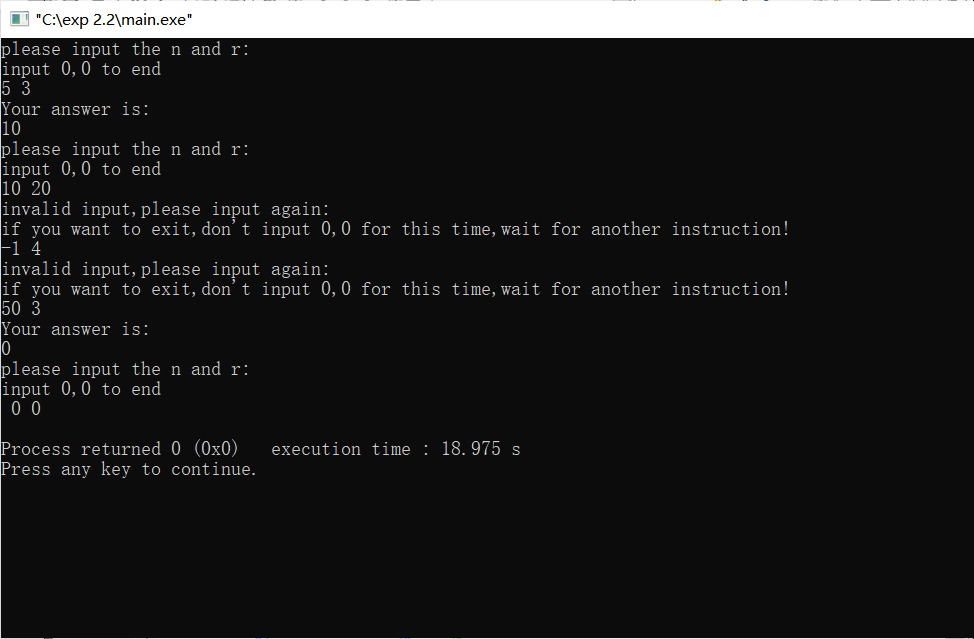
a,b变成了全局变量，实际也没改变什么，因为在func里面的a,b的值不会考虑全局a，b（重名的情况下）。

### 题目2 函数的递归调用和多文件结构



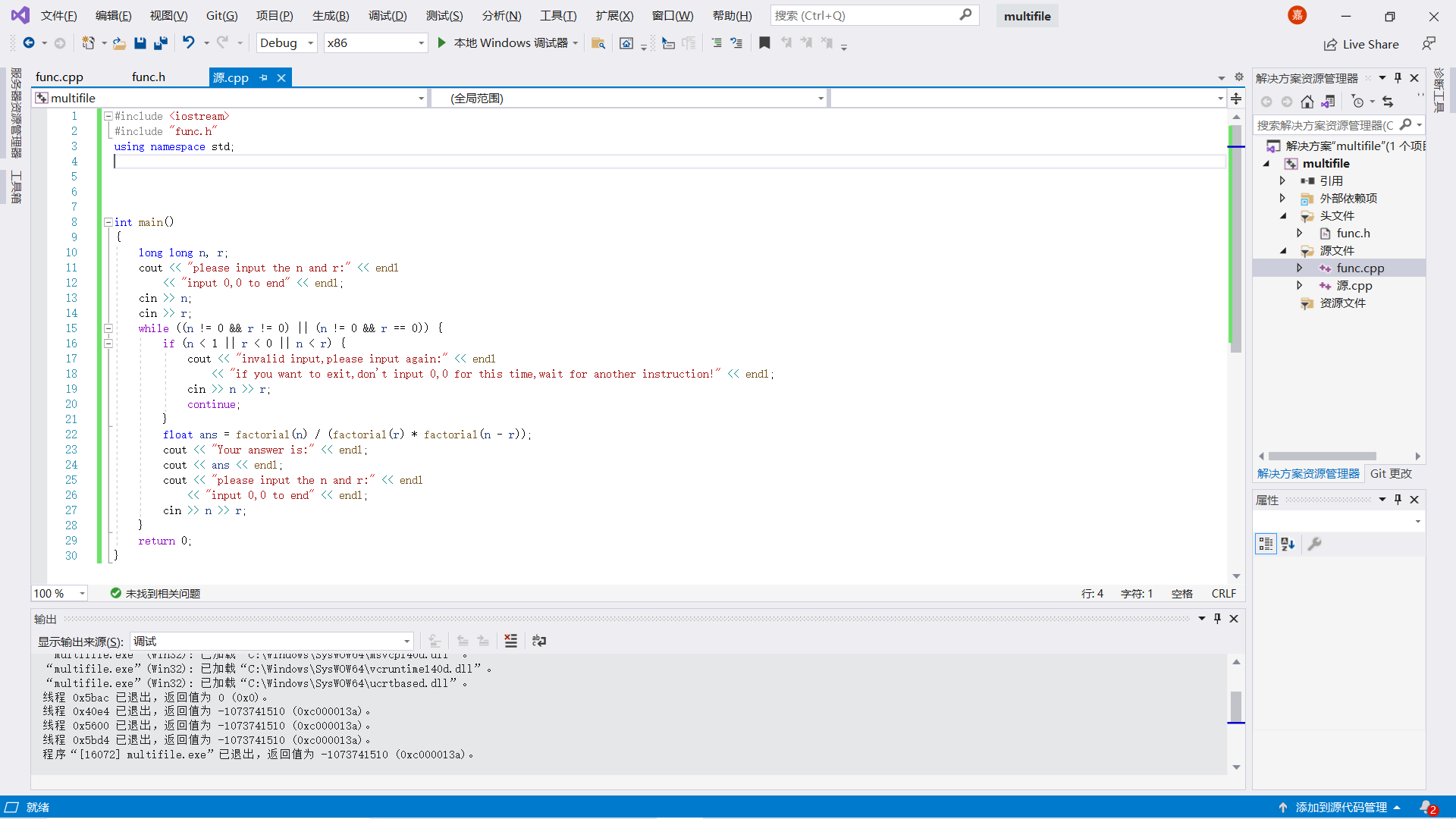
递归初步运行截图如上。50 3 结果为0的原因是codeblocks无法处理过长位数。

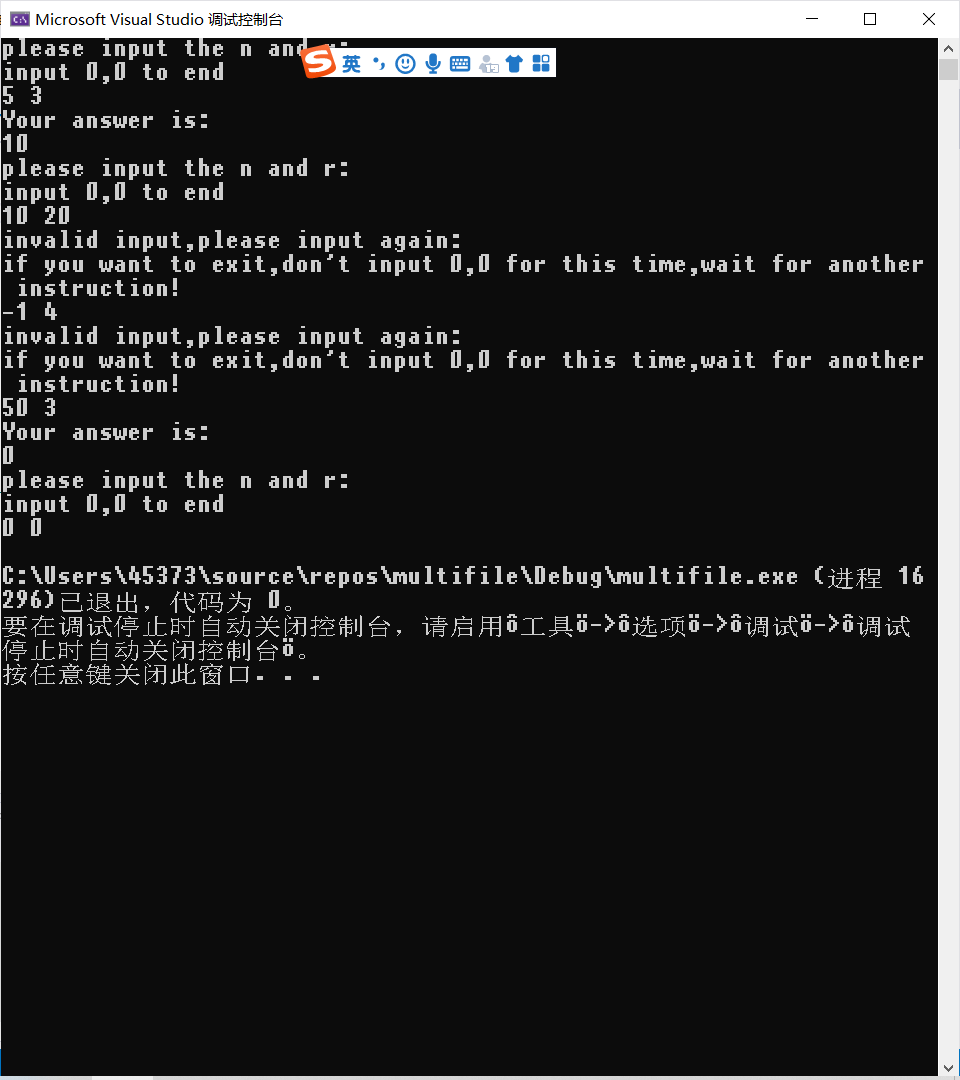
非递归结果如下：



运行multifile里面的main.cpp

截图如下:





思考与扩展：

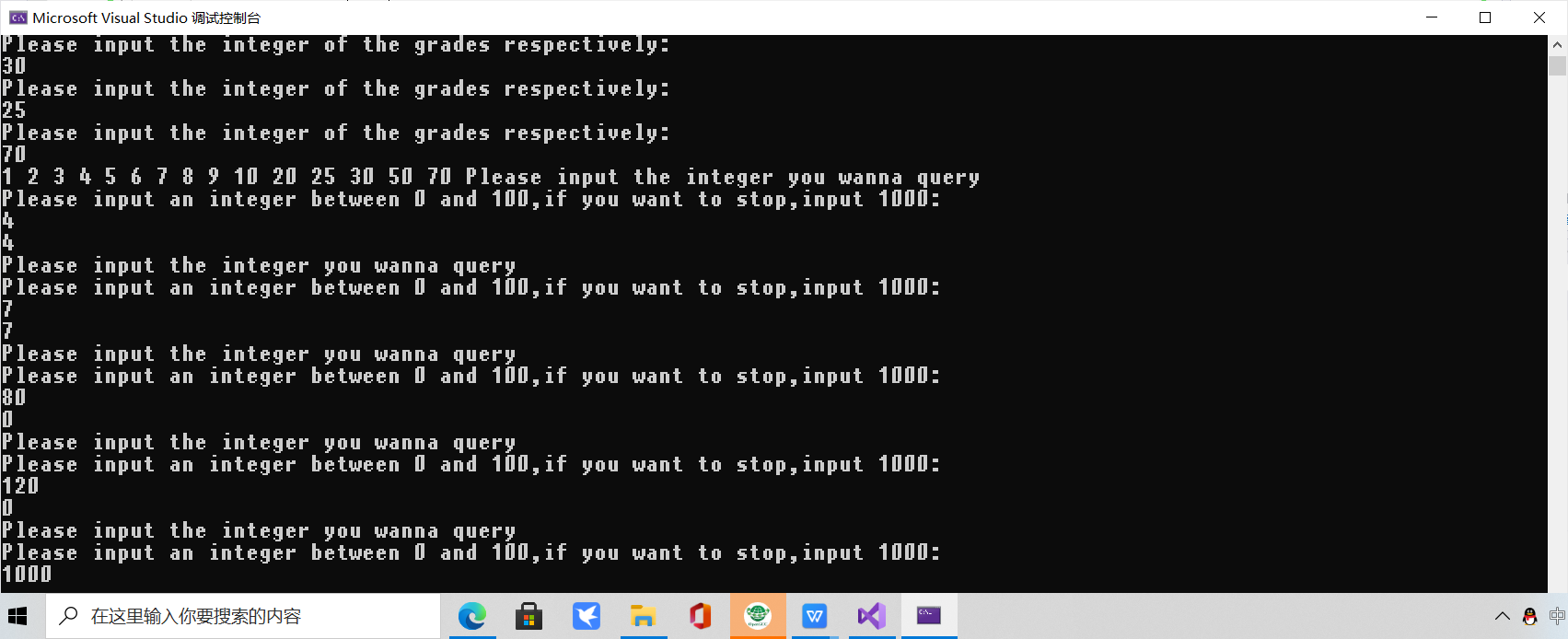
1. codeblocks断点似乎在这里不起作用:( 不过断点流程已很清楚
2. 有终止判断if(....),以及常规执行步骤
3. Windows： 1024K bytes(Release Version)，故n大概在200-300左右就开始stack overflow了

4.在将一个程序分解成若干个文件时，需要考虑标识符在其他文件中的可见性

5.在开发较大程序时，通常将其分解为多个源程序文件，每个较小的程序用一个源程序文件建立，因此此处更好的识别，与修改。

### 题目3 排序查找

运行结果如下图：



如图排完序后的数组是：1,2,3,4,5,6,7,8,9,10,20,25,30,50,70

测试了几个数据，4,7,80,120 //注意这里如果返回0是指没有找到。

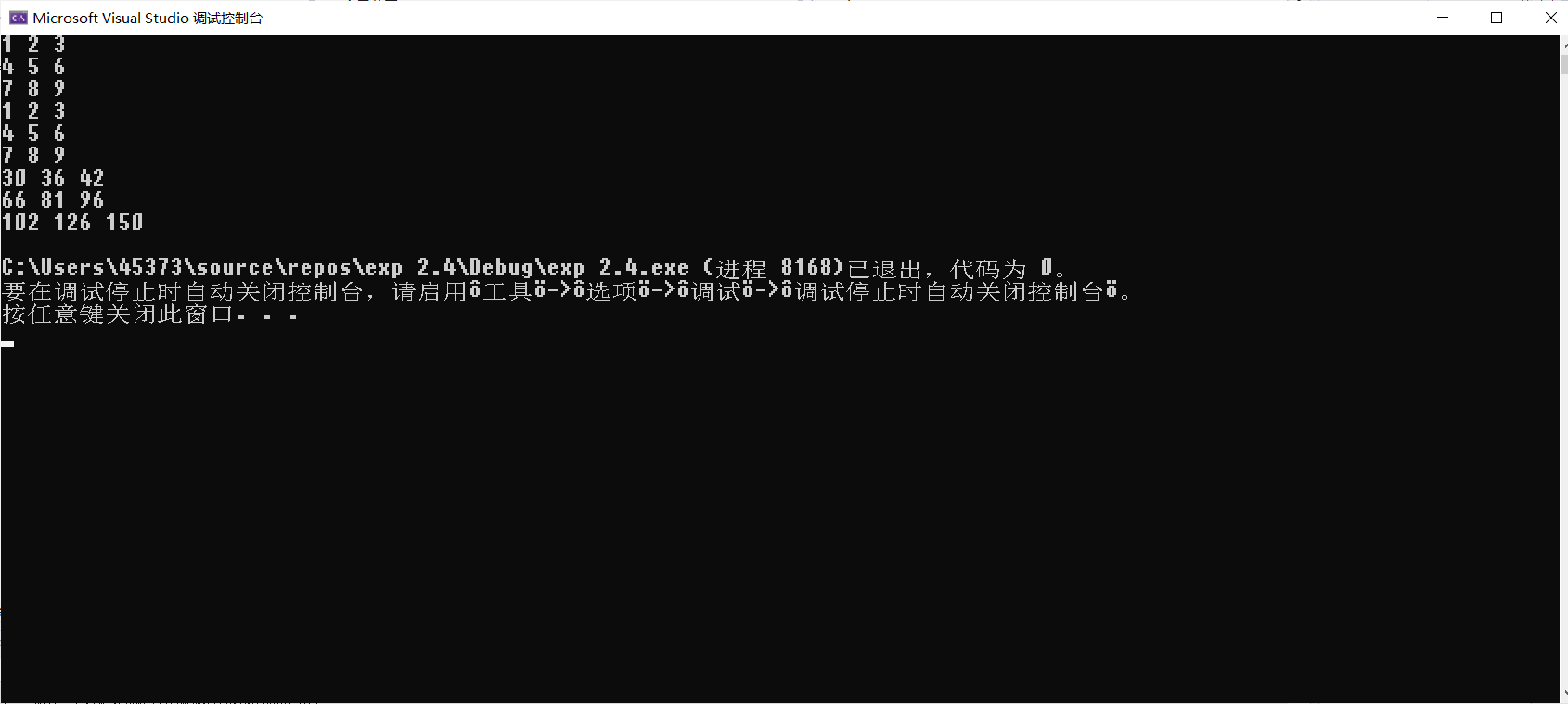
思考与扩展1：

冒泡排序与选择排序都是属于O（n²）的，但是选择排序不稳定，实际情况很有可能远小于n²，故总体来看还是选择排序好。

2：当然是折半查找，这把查找总体时间复杂度log2n，顺序查找是O（n）。

3：数组名的本质是地址，即指针，若数组名作为形参，必须写成(int array[])的形式。

### 题目4 矩阵相乘

运行结果如图：

思考与扩展1：

输入的时候改为这个：

for (int i = 0; i < 4; i++)

for (int j = 0; j < 3; j++)

cin >> A[i][j];

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

for (int j = 0; j < 2; j++)

cin >> B[i][j];

输出的时候改为这个：

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 2; j++)

{

cout << C[i][j] << " ";

}

cout << endl;

}

函数改为这个：

void matrix\_mul(int A[4][3],int B[3][2]) {//矩阵相乘函数

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 2; j++)

{

for (int k = 0; k < 3; k++)

{

C[i][j] += A[i][k] \* B[k][j];

}

}

}

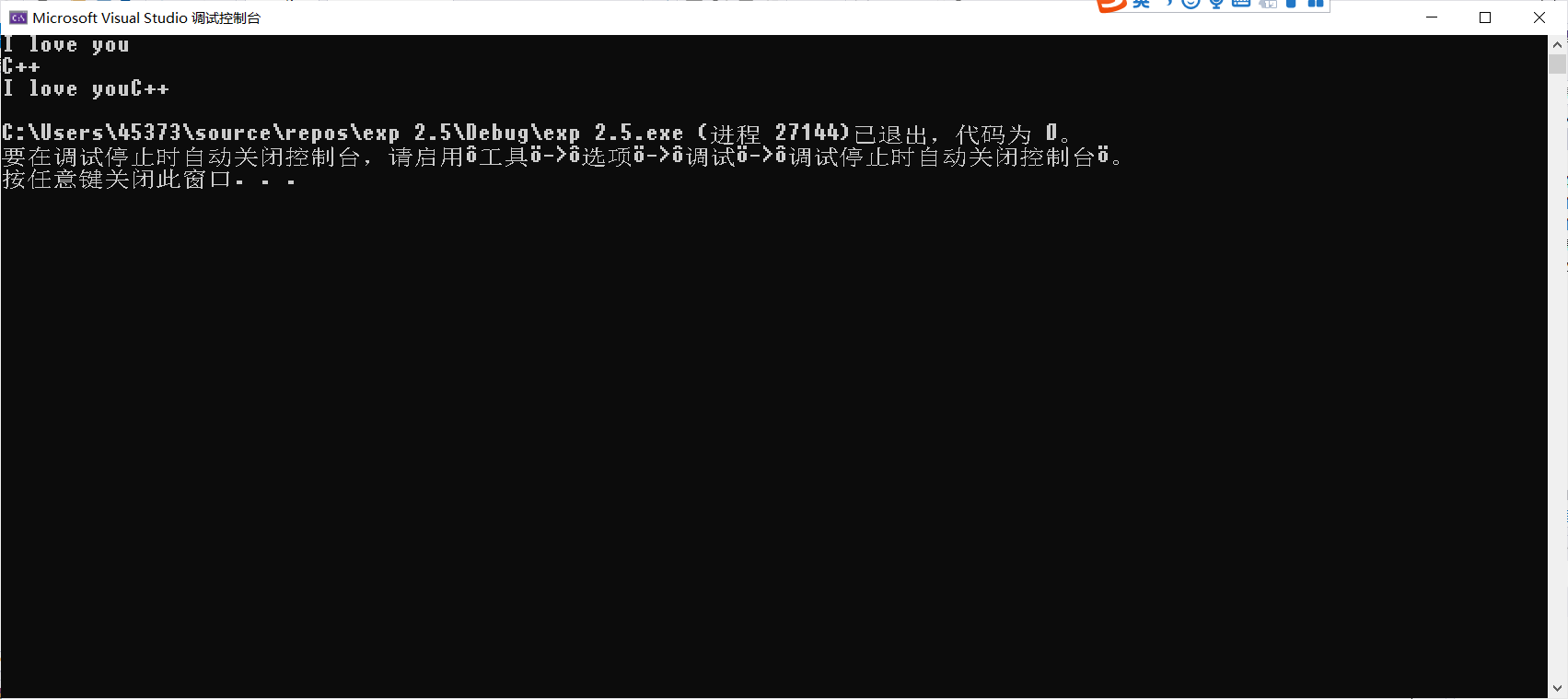
}

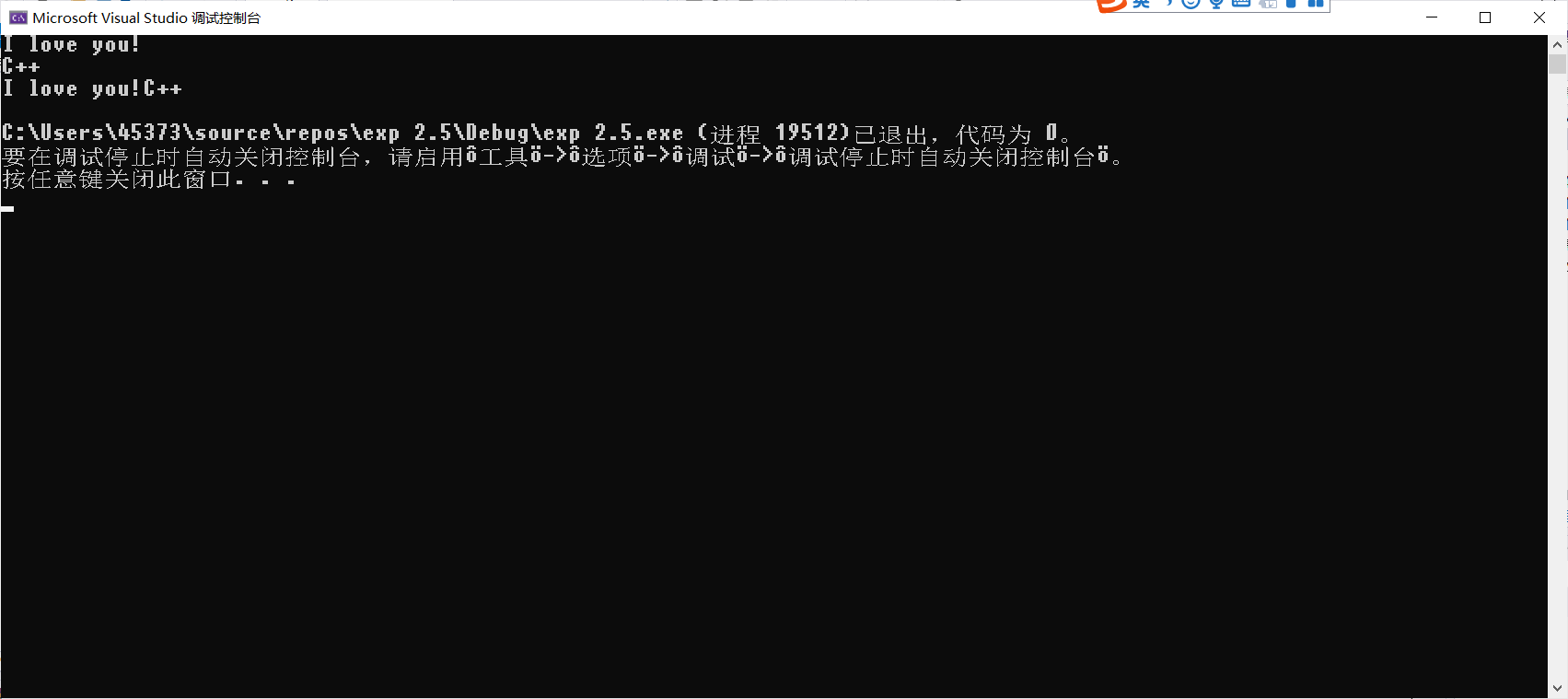
思考与扩展2：多维数组，传入函数的实参仍然是指针，只不过是指向数组的指针。形参而言，必须规定清楚第二维的（或者后几维）的大小。

### 题目5 字符串连接

运行结果如图：

字符数组：



Strcat：

String:

和以上俩个一样。

显然方法一用数组模拟最为复杂，其次是方法二用函数，最简单的还是直接用c++提供的string可以直接加减最为方便。

思考与扩展：

* 对于字符数组，其长度是固定的，其中任何一个数组元素都可以为 null 字符。因此，字符数组不一定是字符串。
* 对于字符串，它必须以 null 结尾，其后的字符不属于该字符串。字符串一定是字符数组，它是最后一个字符为 null 字符的字符数组。