

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： CS2208**

**学 号： U202215595**

**姓 名： 郭凯锐**

**指导教师： 毛伏兵**

**报告日期： 2022年11月30日**

**计算机科学与技术学院**

目 录

[目 录 II](#_Toc121408445)

[**1 实验6 指针程序设计实验 1**](#_Toc121408446)

[1.1 程序改错与跟踪调试 1](#_Toc121408447)

[1.2 程序完善与修改替换 3](#_Toc121408448)

[1.3 程序设计 6](#_Toc121408449)

[1.4小结 12](#_Toc121408450)

[**2 实验7 结构与联合 13**](#_Toc121408451)

[2.1 表达式求值的程序验证 13](#_Toc121408452)

[2.2 源程序修改替换 14](#_Toc121408453)

[2.3 程序设计 16](#_Toc121408454)

[2.4小结 26](#_Toc121408455)

[参考文献 27](#_Toc121408456)

# **1 实验6 指针程序设计实验**

## 1.1 程序改错与跟踪调试

在下面所给的源程序中，函数 strcopy(t, s) 的功能是将字符串 s 复制给字符串 t，并且返回串 t 的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:  
programming↙（键盘输入）  
programming  
Input a string again:  
language↙ （键盘输入）  
language

源程序：



图1-1-1-1 源程序

解答：

第 5 行的字符数组定义错误，正确形式为：

#define maxn 10005

char s1[maxn] , s2[maxn] , s3[maxn];

第 12 行的语句错误，正确形式为：

strcopy(s3, s2);

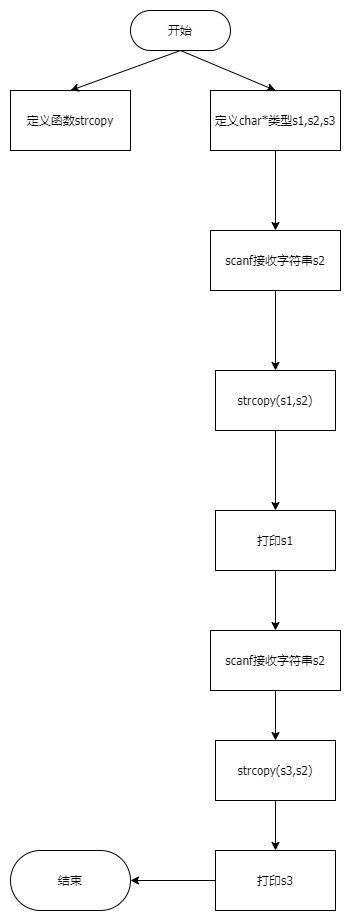


图1-1-1-2 流程图

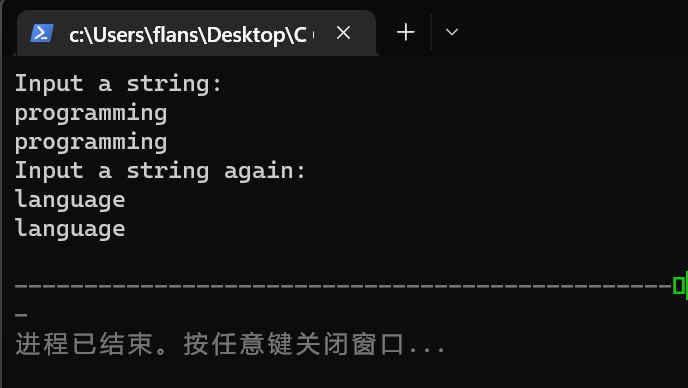


图 1-1-2 源程序改错题的程序运行结果示意图

## 1.2 程序完善与修改替换

（1）下面程序中函数 strsort 用于对字符串进行升序排序，在主函数中输入 N 个字符串（字符串长度不超过 49）存入通过 malloc 动态分配的存储空间，然后调用 strsort 对这 N 个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

源程序：



图1-2-1程序完善源程序

替换后的程序如下图所示：



图1-2-2 修改之后的程序



图6-2-3 运行结果

②数组作为函数参数其本质类型是指针。例如，对于形参 char \*s []，编译器将其解释为 char \*\*s，两种写法完全等价。请用二级指针形参重写 strsort 函数，并且在该函数体的任何位置都不允许使用下标引用。

替换后的程序如下所示：



图1-2-4 替换之后的程序



图1-2-5运行结果

## 1.3 程序设计

1. 指针取出每个字节：

编写一个程序，从整形变量的高字节开始，依次取出每字节的高4位和低4位并以十六进制数字字符的形式进行显示，要求通过指针取出每字节。

解答：

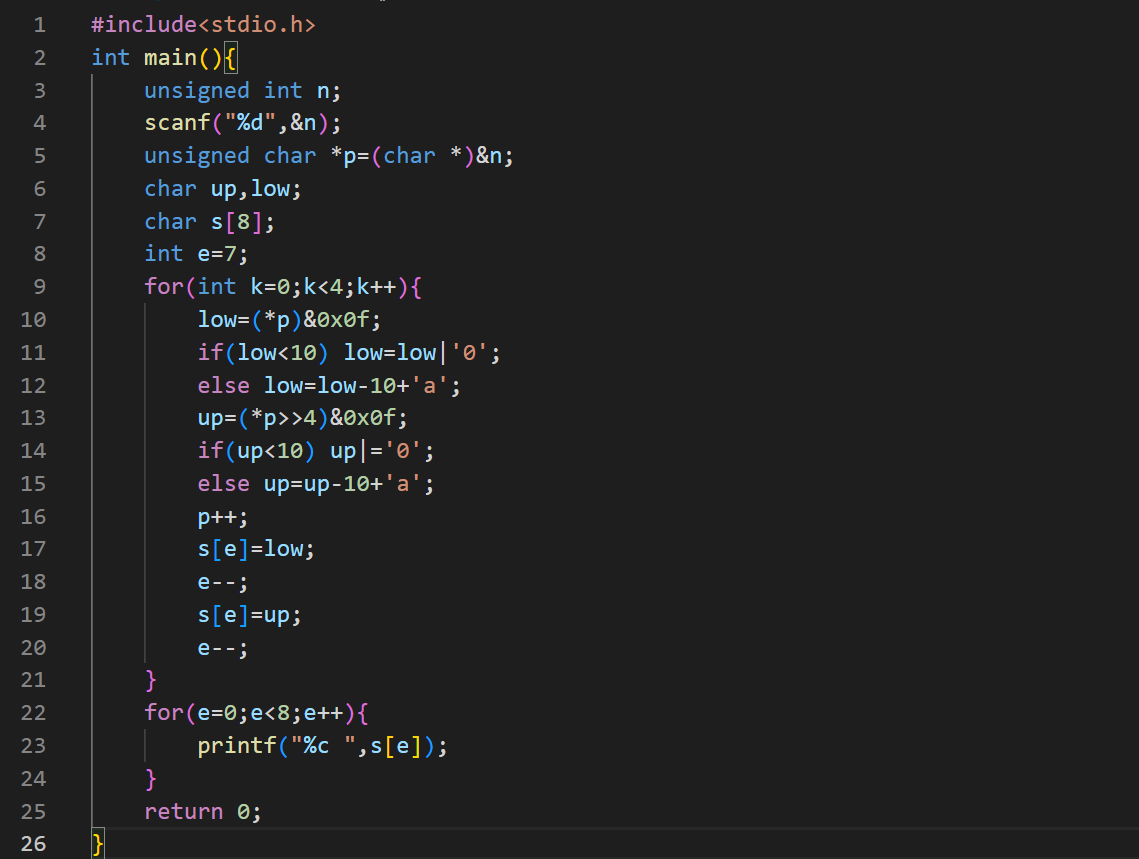


图1-3-1 程序代码

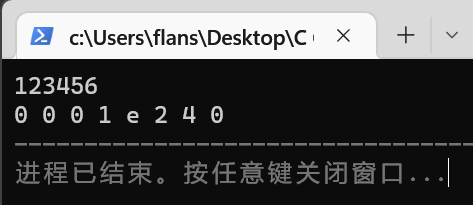


图1-3-2 运行结果

2.删除重复字符：

定义函数RemoveSame(a,n)，去掉有n个元素的有序整数序列a中的重复元素，返回去重后序列的长度。

解答：

.

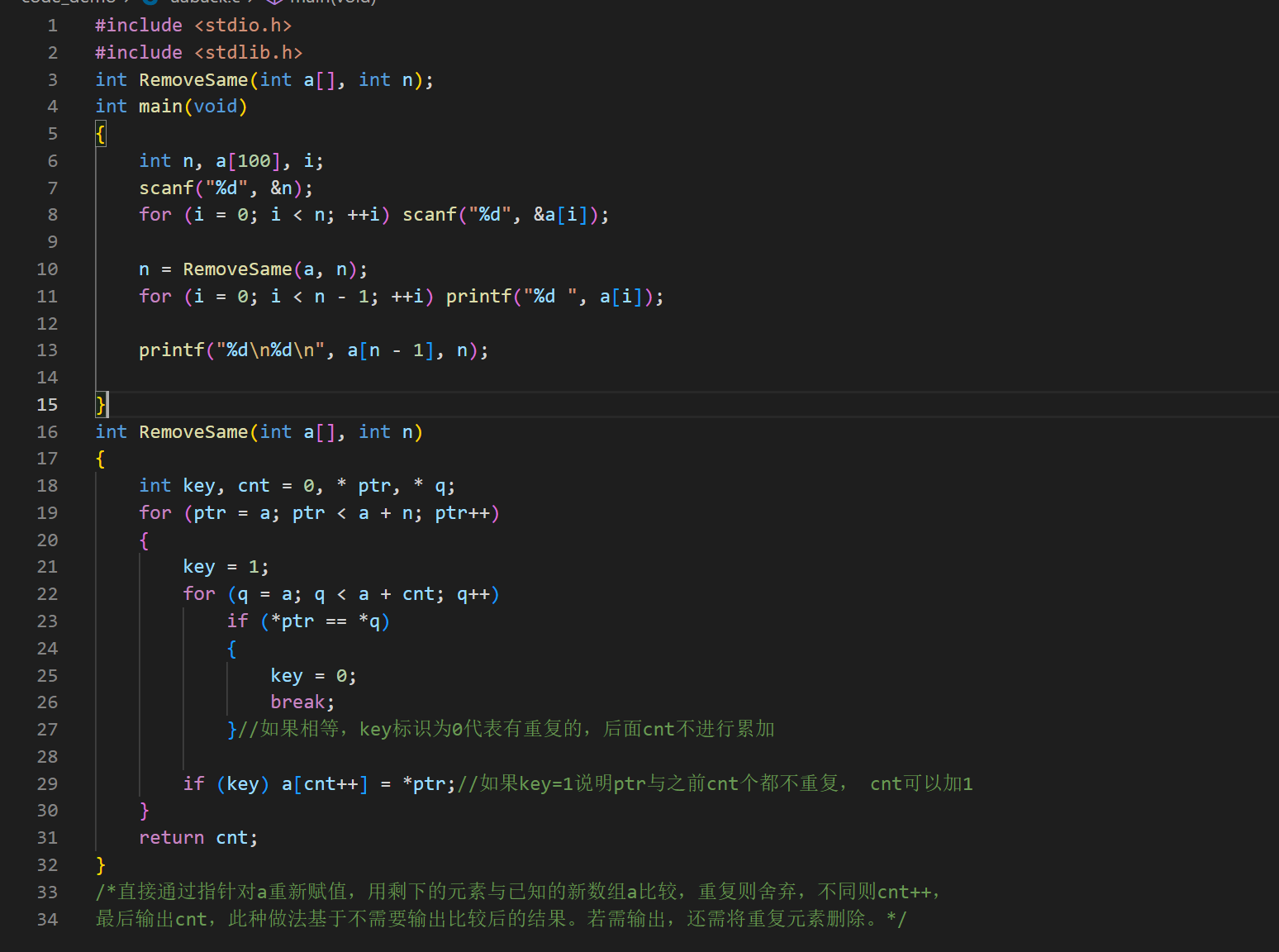


图1-3-3 程序代码



图1-3-4 运行结果

3.旋转图像：

输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像，输出原始矩阵逆时针旋转90°后的矩阵。

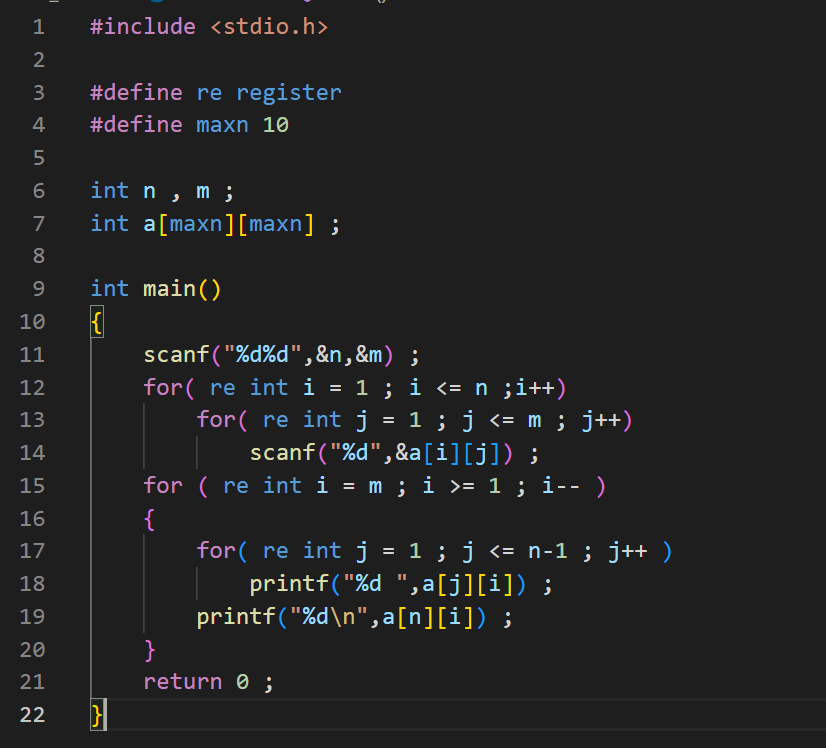


图1-3-5 程序代码

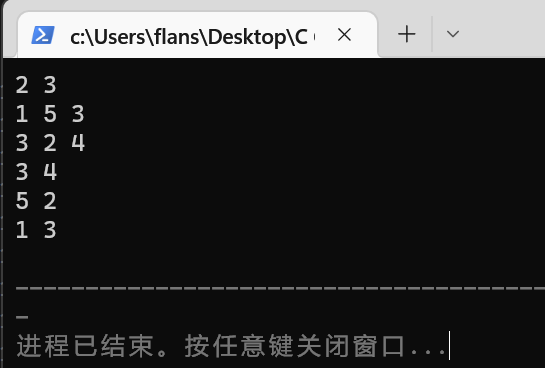


图1-3-6 运行结果

4. 命令行实现对N个整数排序:

解答：

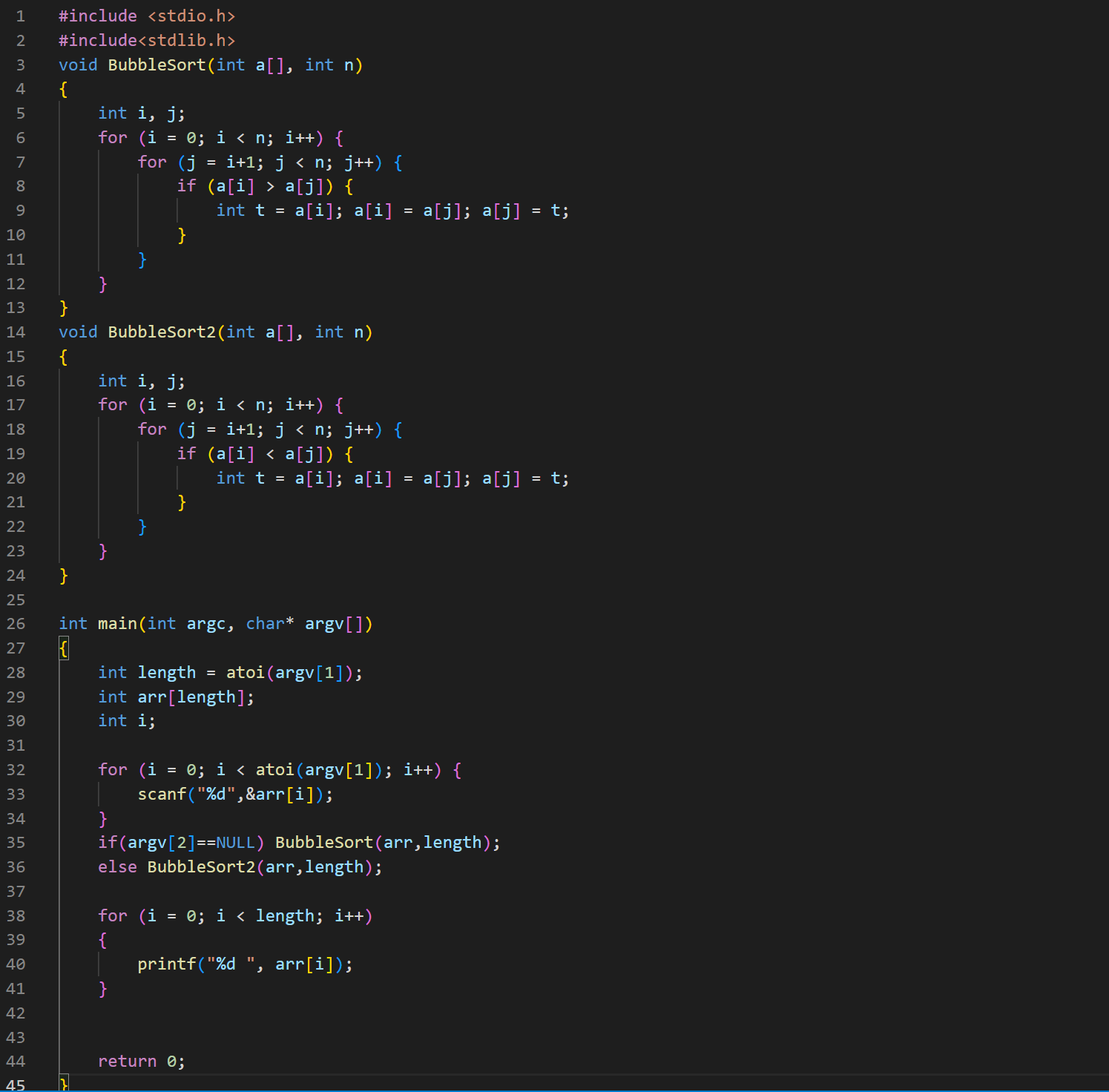


图1-3-7 程序代码

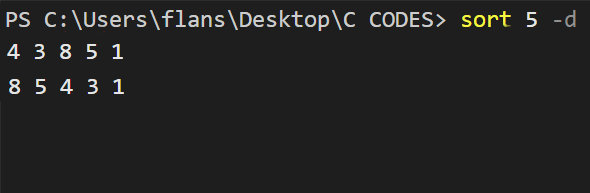


图1-3-6 运行结果

5.删除子串：

解答：

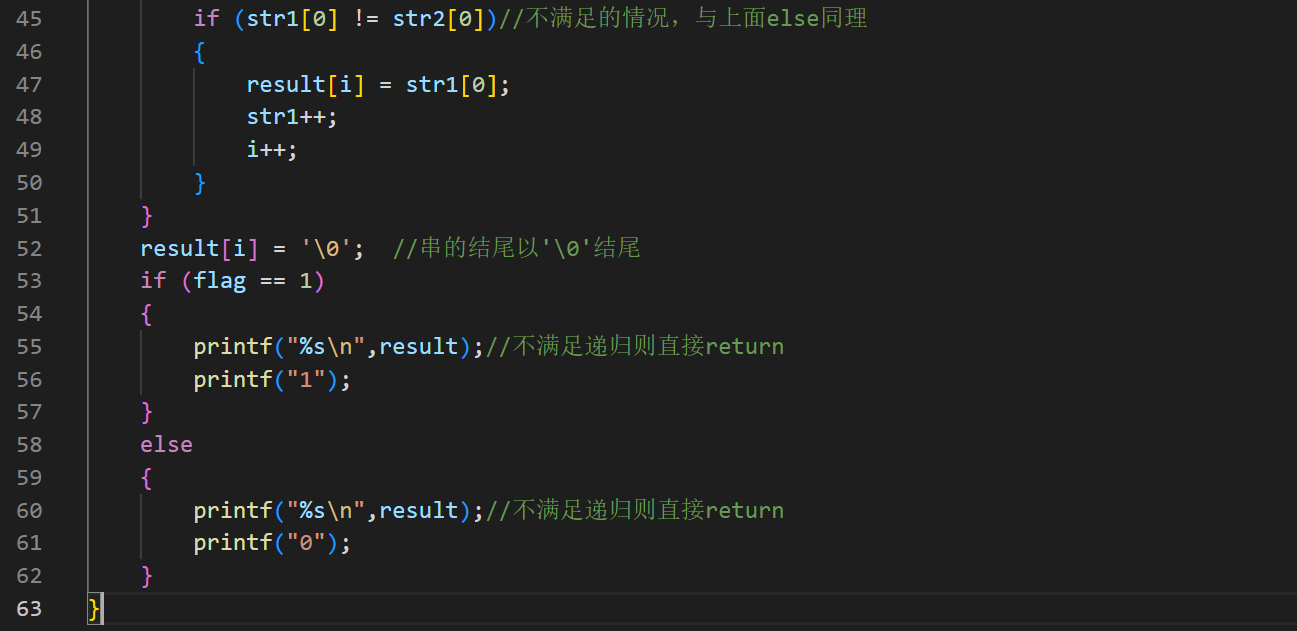


图1-3-9 程序代码

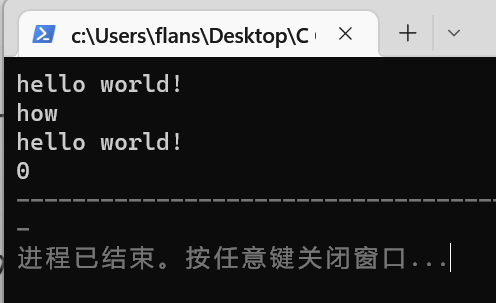


图1-3-10 运行结果

\*\*6. 求两个不超过200位的非负整数积：

解答:

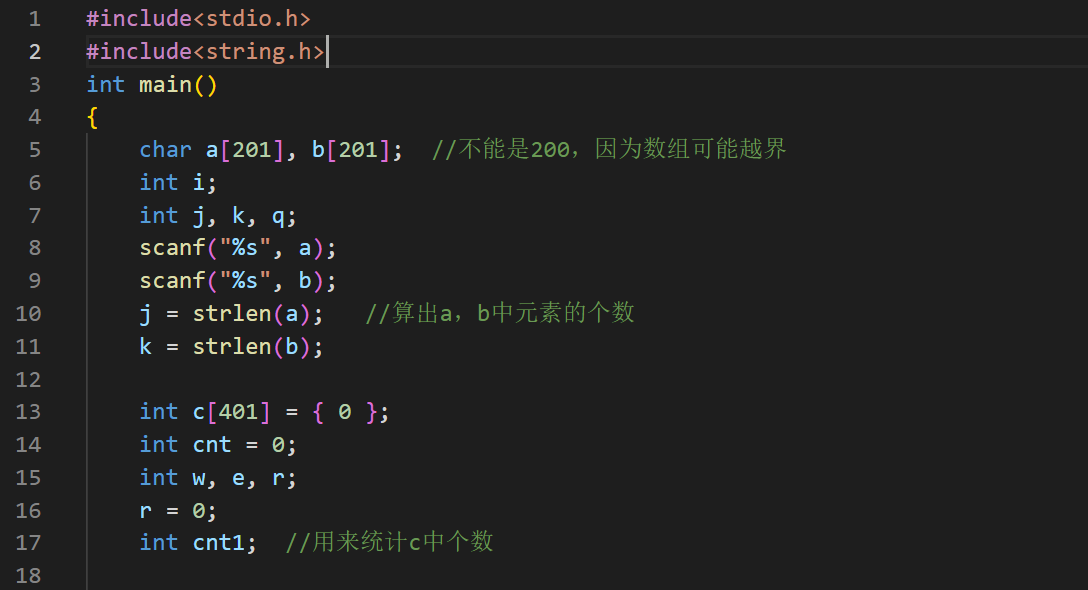


图1-3-11 程序代码

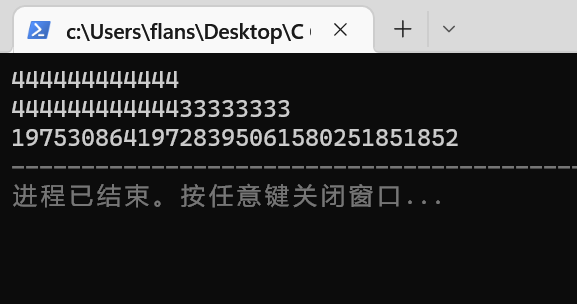


图1-3-12 运行结果

## 1.4小结

指针是C语言学习和使用的重难点。本次实验主要练习了指针的使用，包括指针变量的声明和引用，利用指针引用所指变量，作为函数参数和返回值的指针的使用，复杂类型指针的使用等。

指针常常被称为C语言的精华，我认识到了它的重要意义。借由指针，C语言可以对计算机内存进行直接操作，更加贴近底层，大大提高了执行效率。但同时，这也要求我们设计程序时格外重视对指针的使用，防止出现野指针，内存泄漏等情况。

# 2 实验7 结构与联合

## 2.1 表达式求值的程序验证

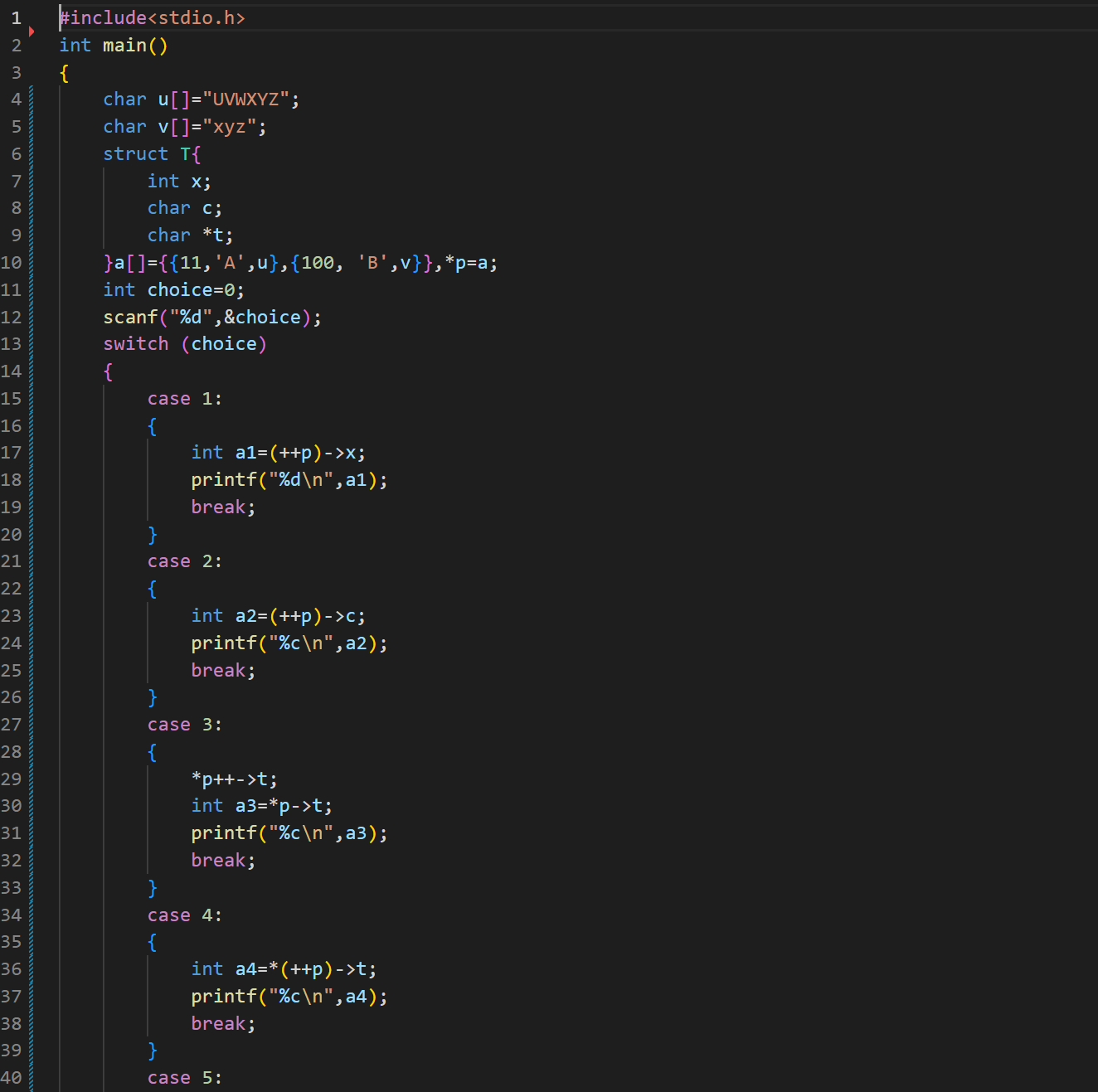


图2-1-1 程序代码

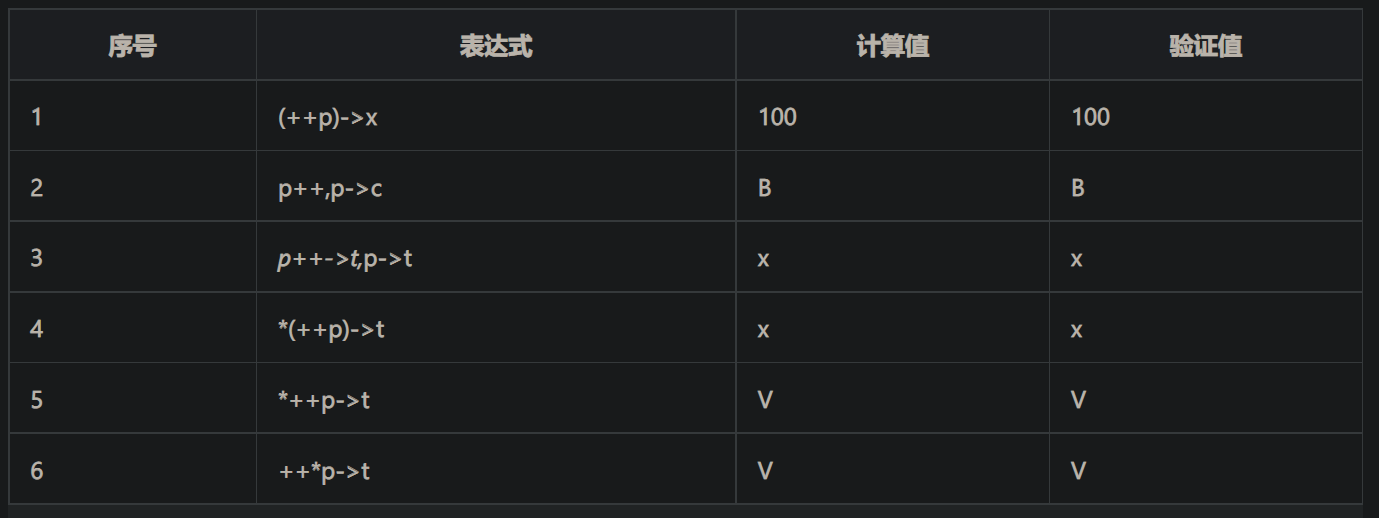


图2-1-2 计算结果

## 2.2 源程序修改替换

（1）给定一批整数，以 0 为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

修改后的程序：

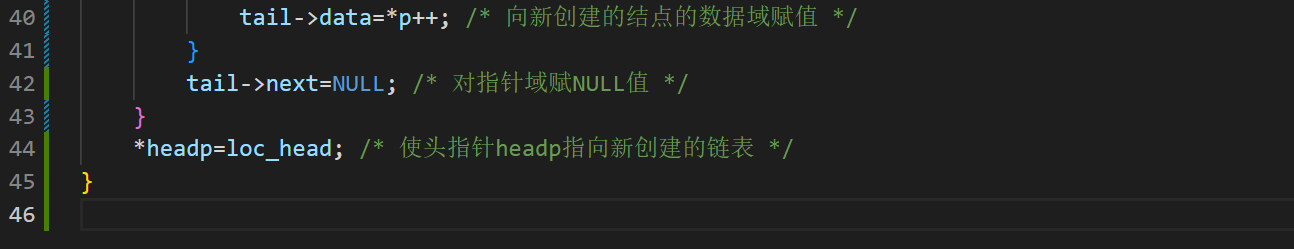
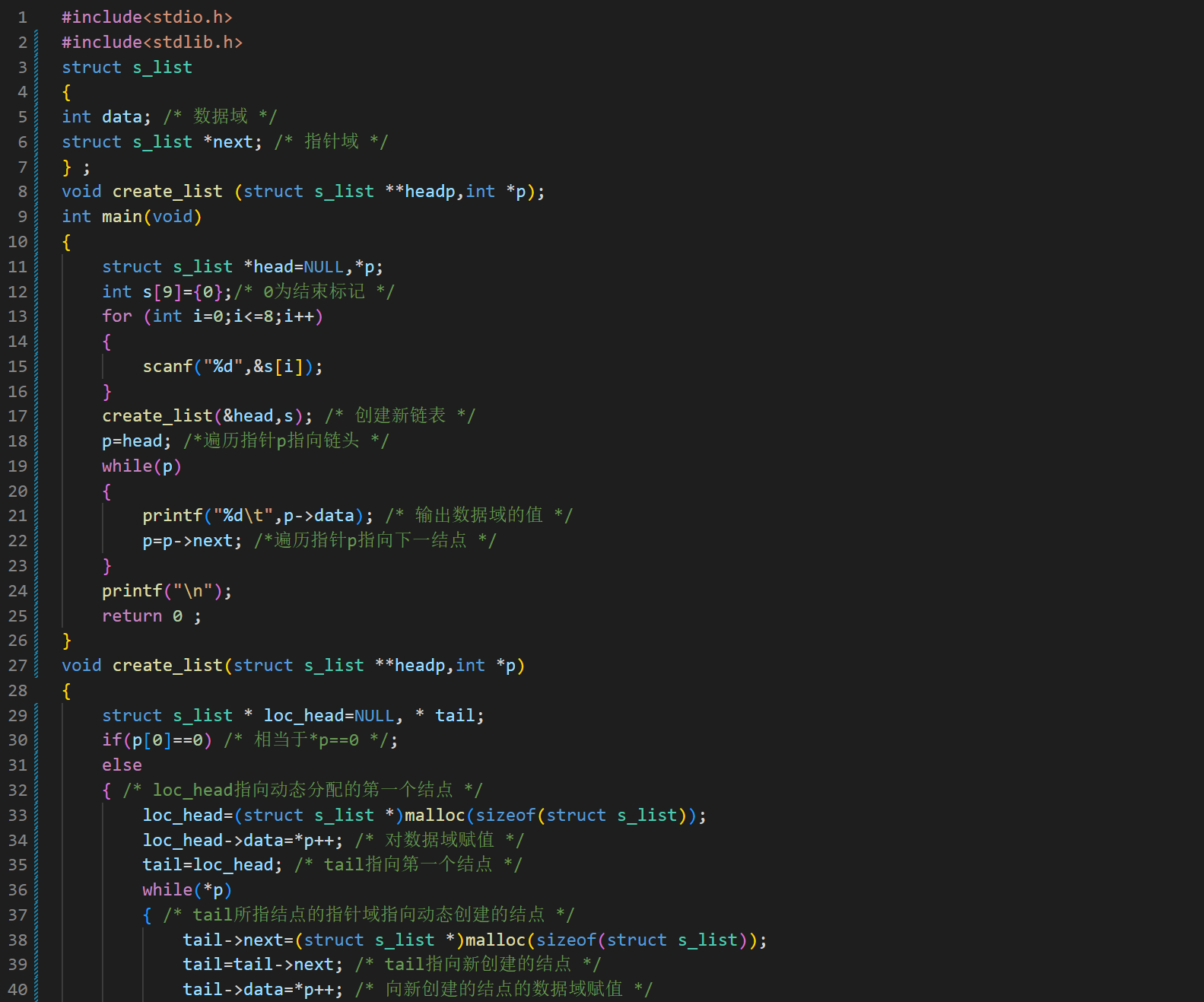


图2-2-1 程序代码

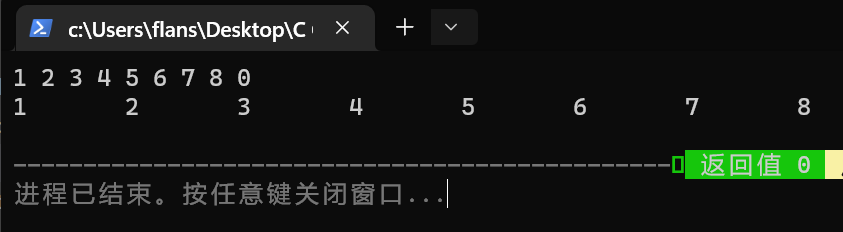


图2-2-2 运行结果

（2）修改替换create\_list函数，将其建成一个后进先出的链表。

修改后的程序：

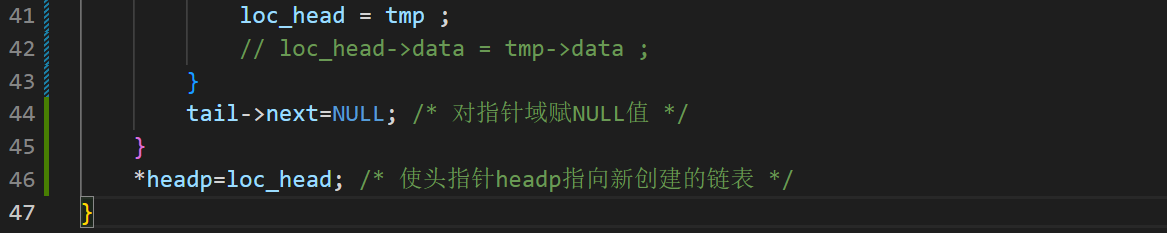
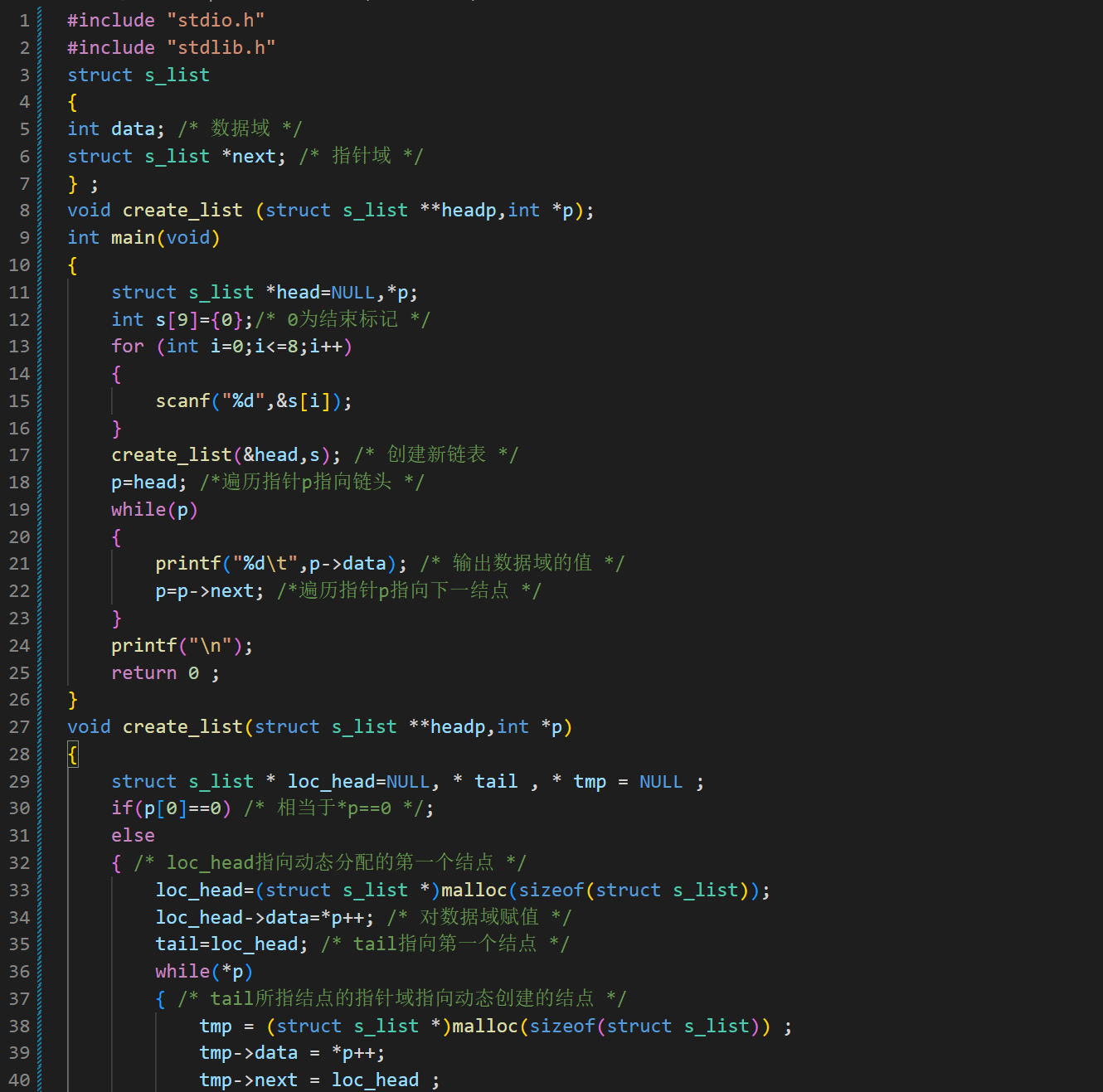


图2-2-3 程序代码



图2-2-4 运行结果

## 2.3 程序设计

（1）设计一个字段结构 struct bits, 它将一个 8 位无符号字节从最低位到最高位声明为 8 个字段，依次为 bit0, bit1,..., bit7，且 bit0 优先级最高。同时设计8个函数，第 i 个函数以 biti(i=0, 1, ..., 7)为参数，并且在函数体内输出 biti 的值。将 8 个函数的名字存入一个函数指针数组 p\_fun。如果 bit0 为 1, 调用 p\_fun[0] 指向的函数。如果 struct bits 中有多位为 1，则根据优先级从高到低顺序依次调用函数指针数组 p\_fun 中相应元素指向的函数。

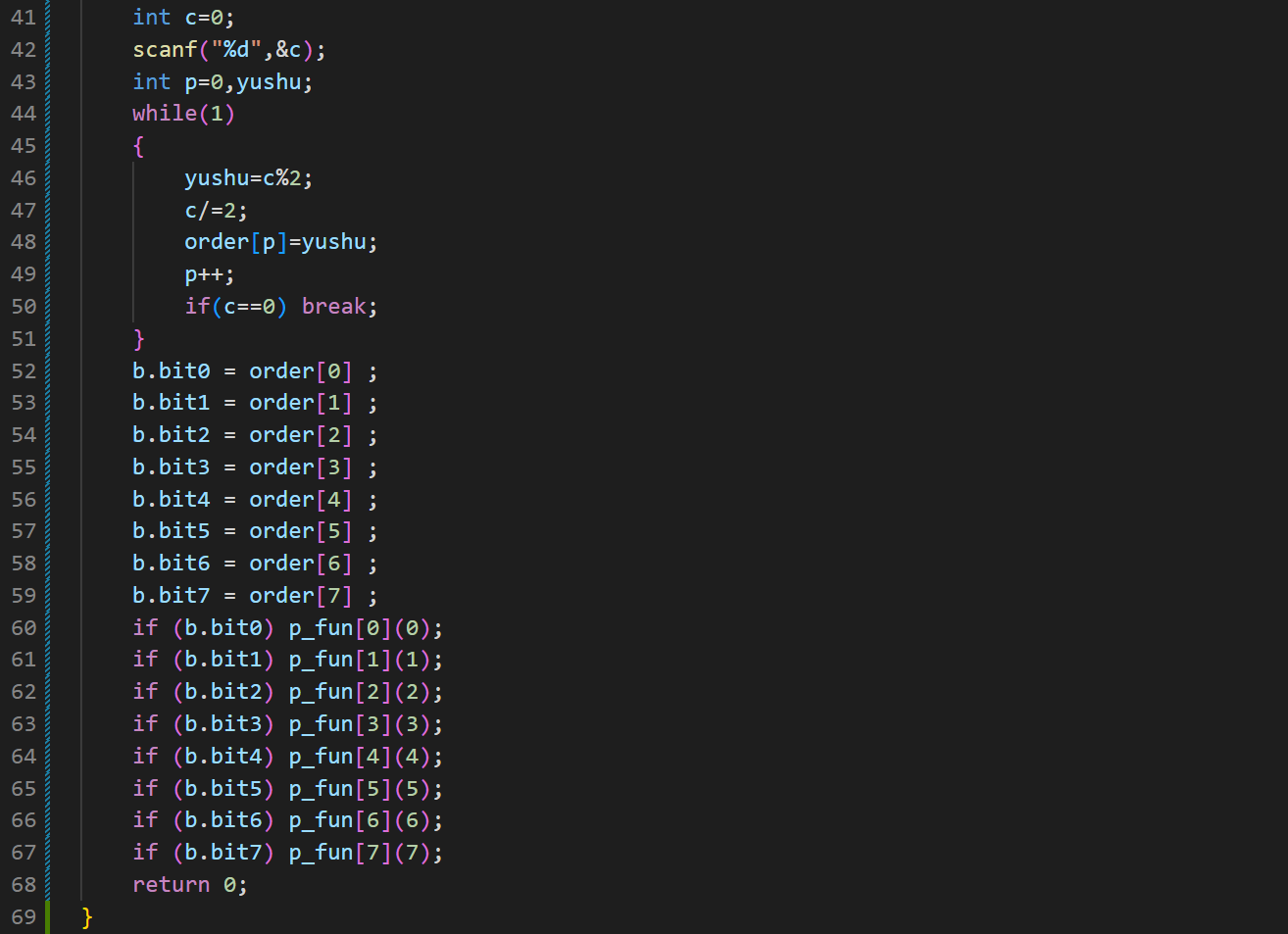
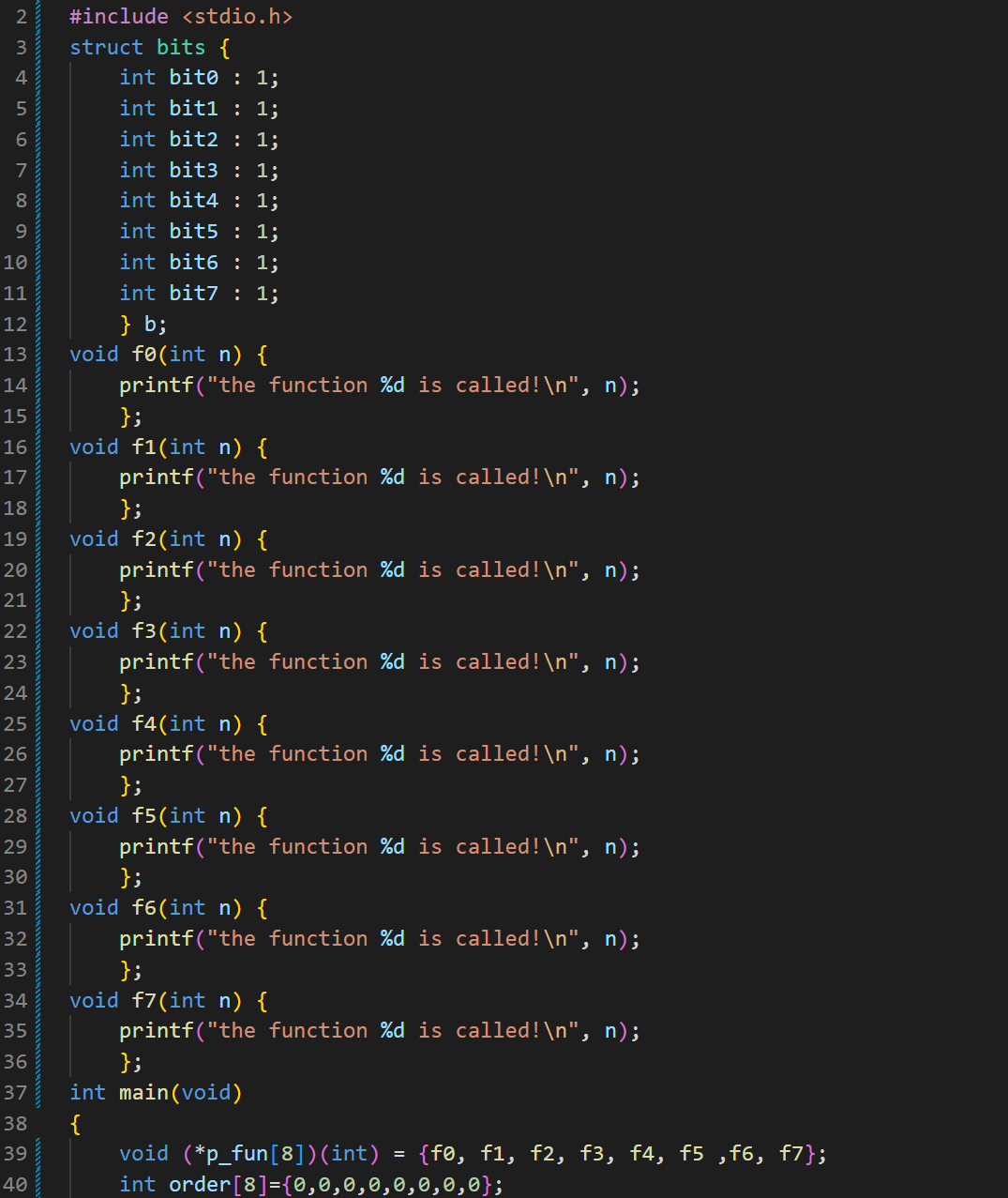


图2-2-5 程序代码

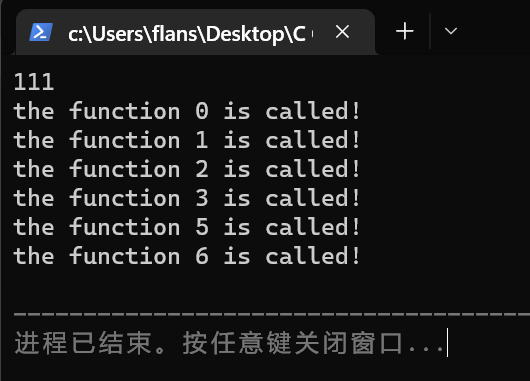


图2-2-6 运行结果

（2）用单向链表建立一张班级成绩单，包括每个学生的学号、姓名、英语、高等数学、普通物理、C语言程序设计4门课程的成绩。

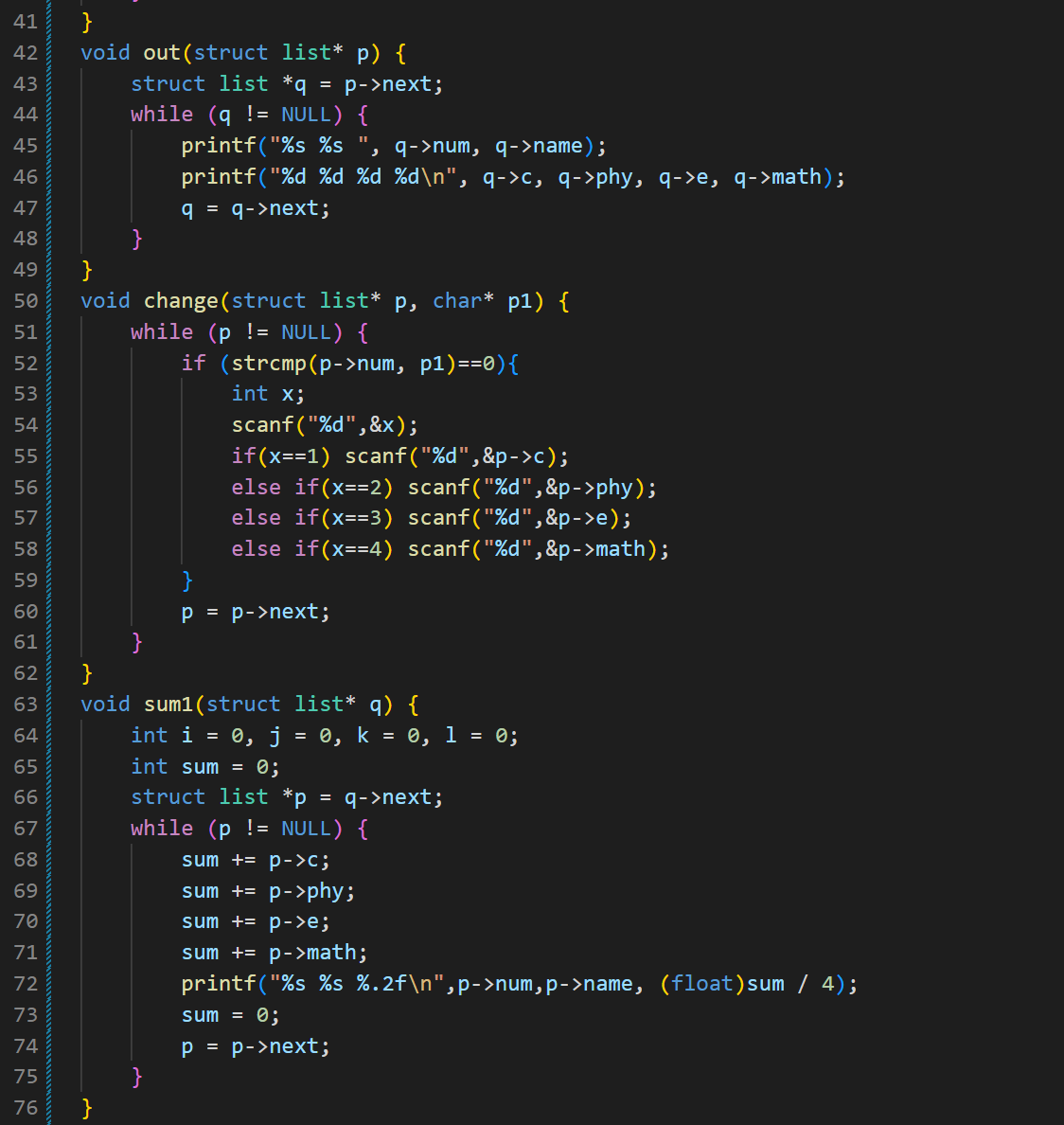
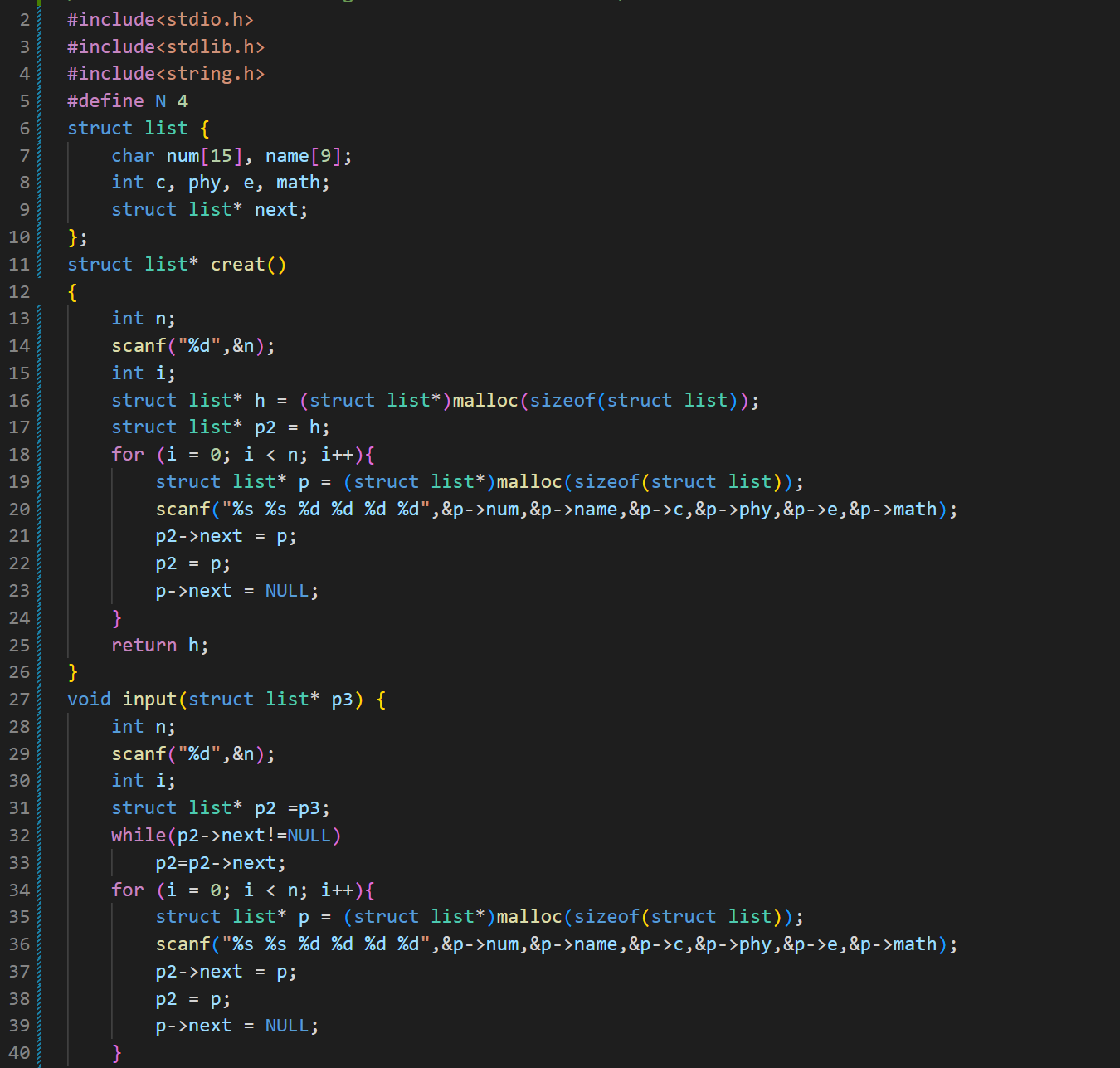


图2-2-7 程序代码

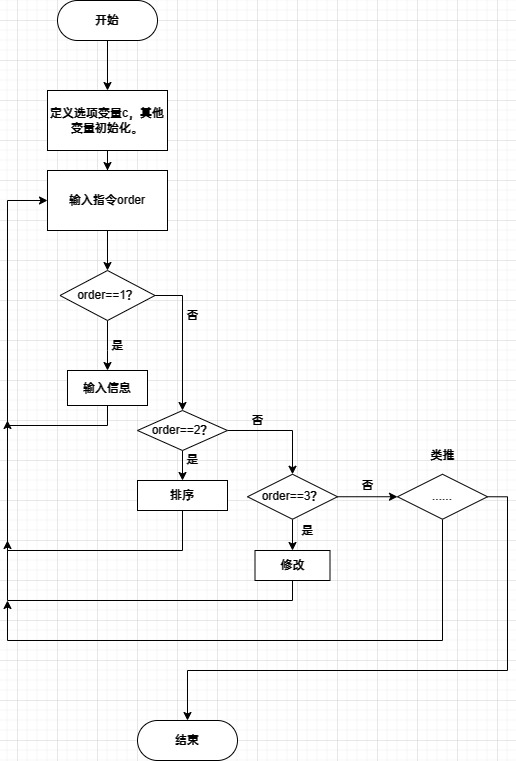


图2-2-8-1 流程图

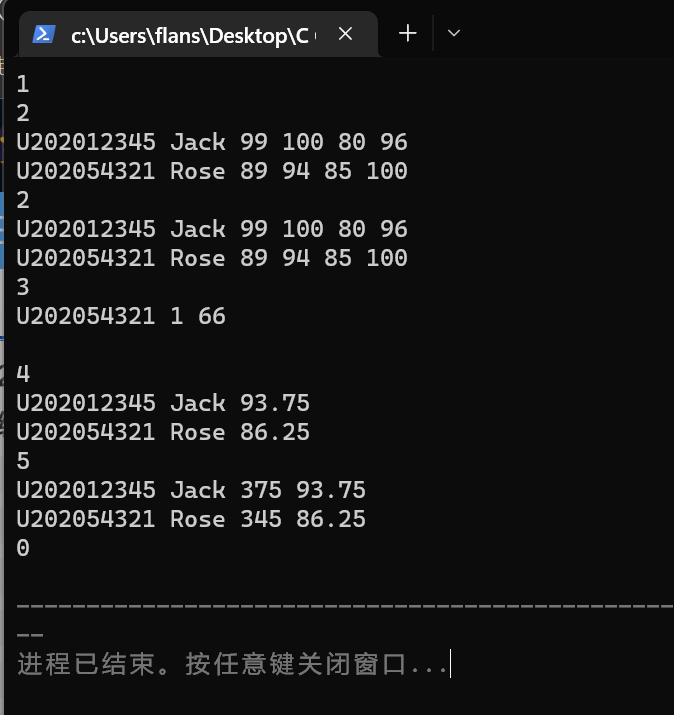


图2-2-8-2 运行结果

（3）对程序设计的第二题增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。

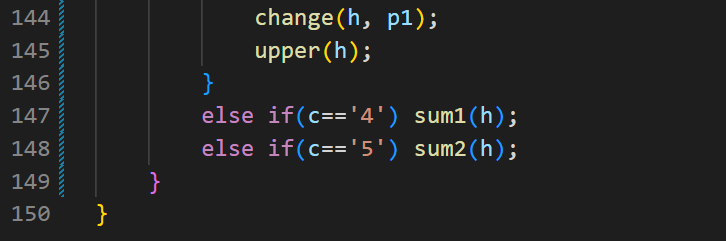
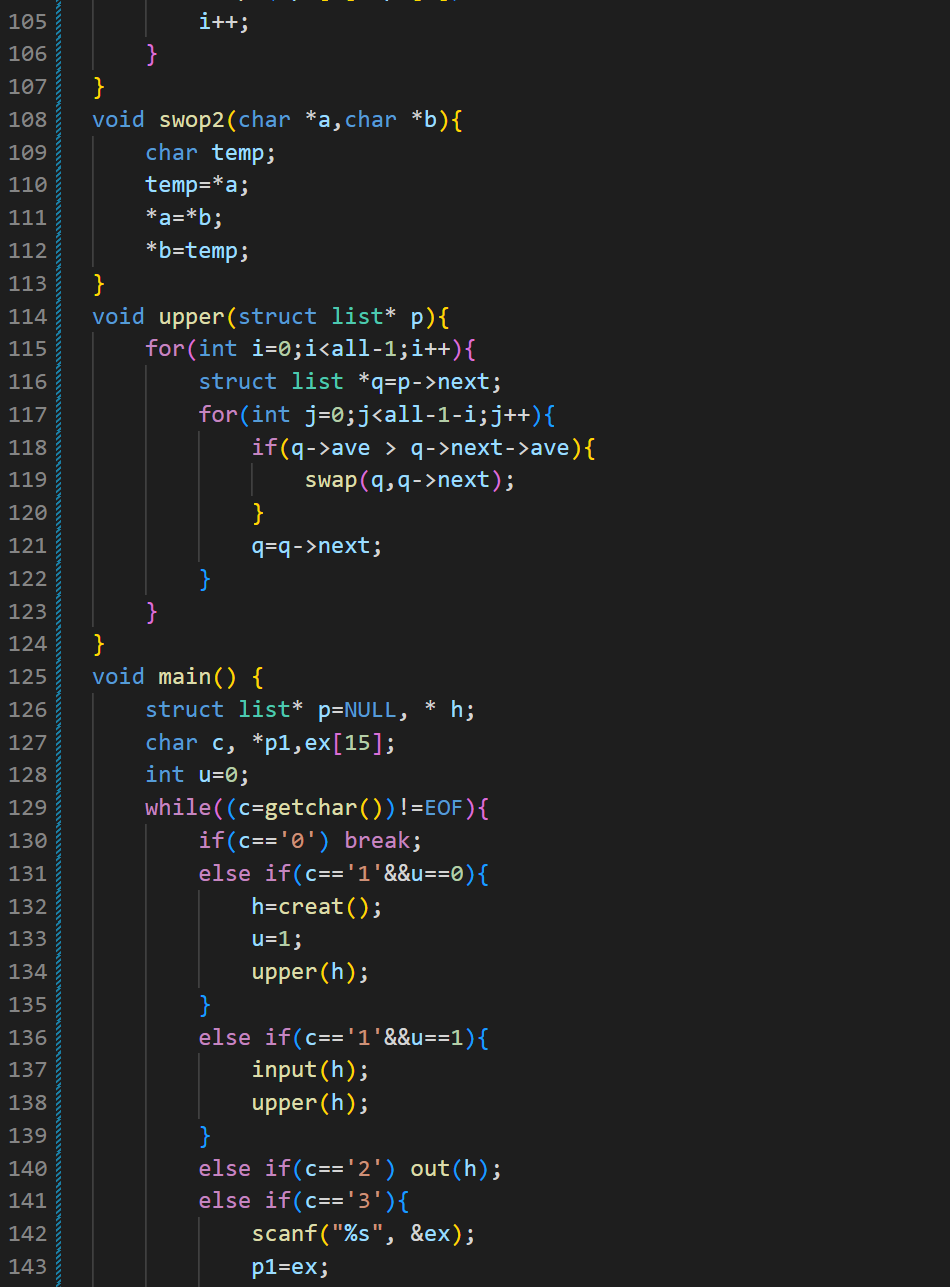
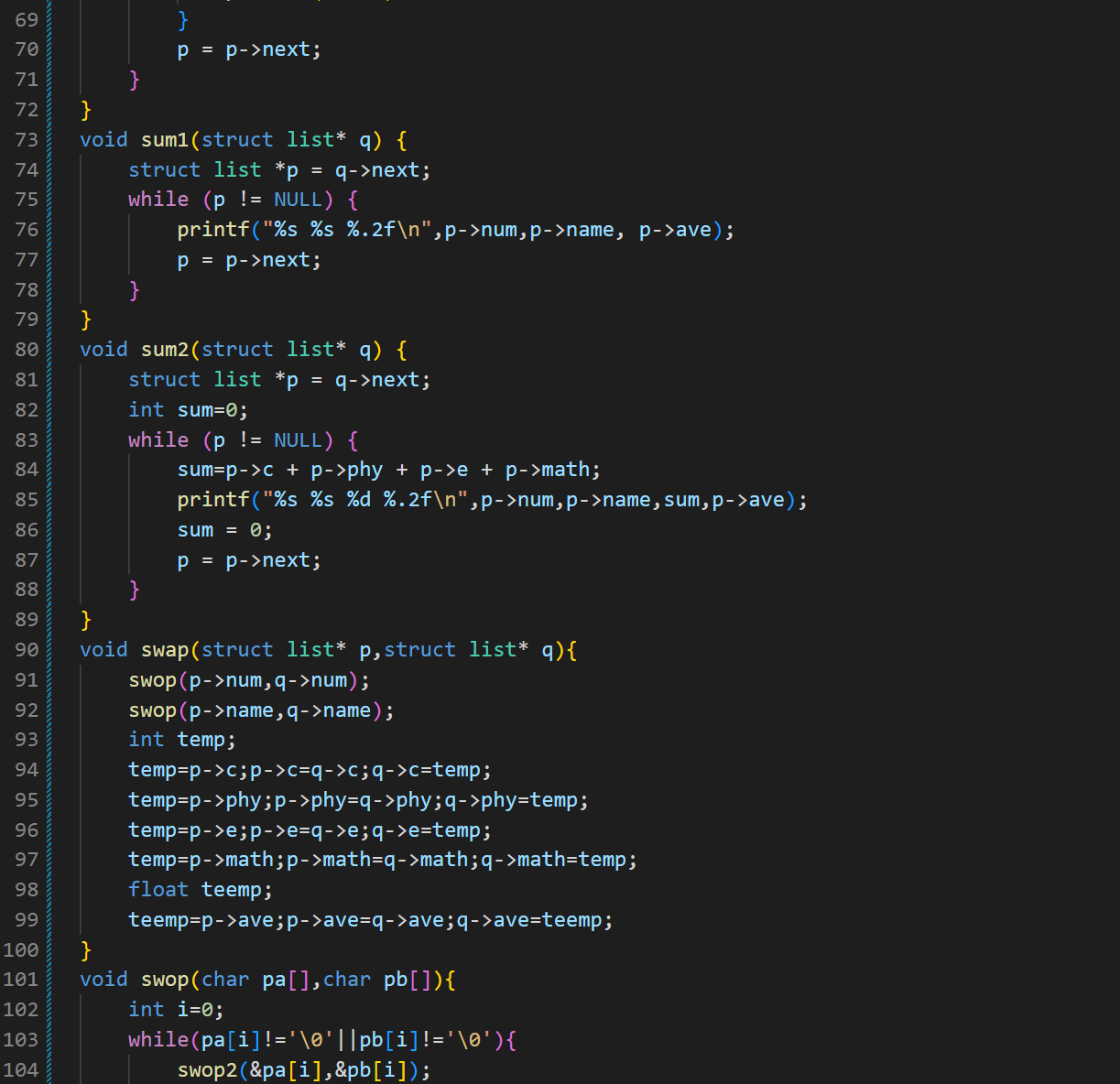
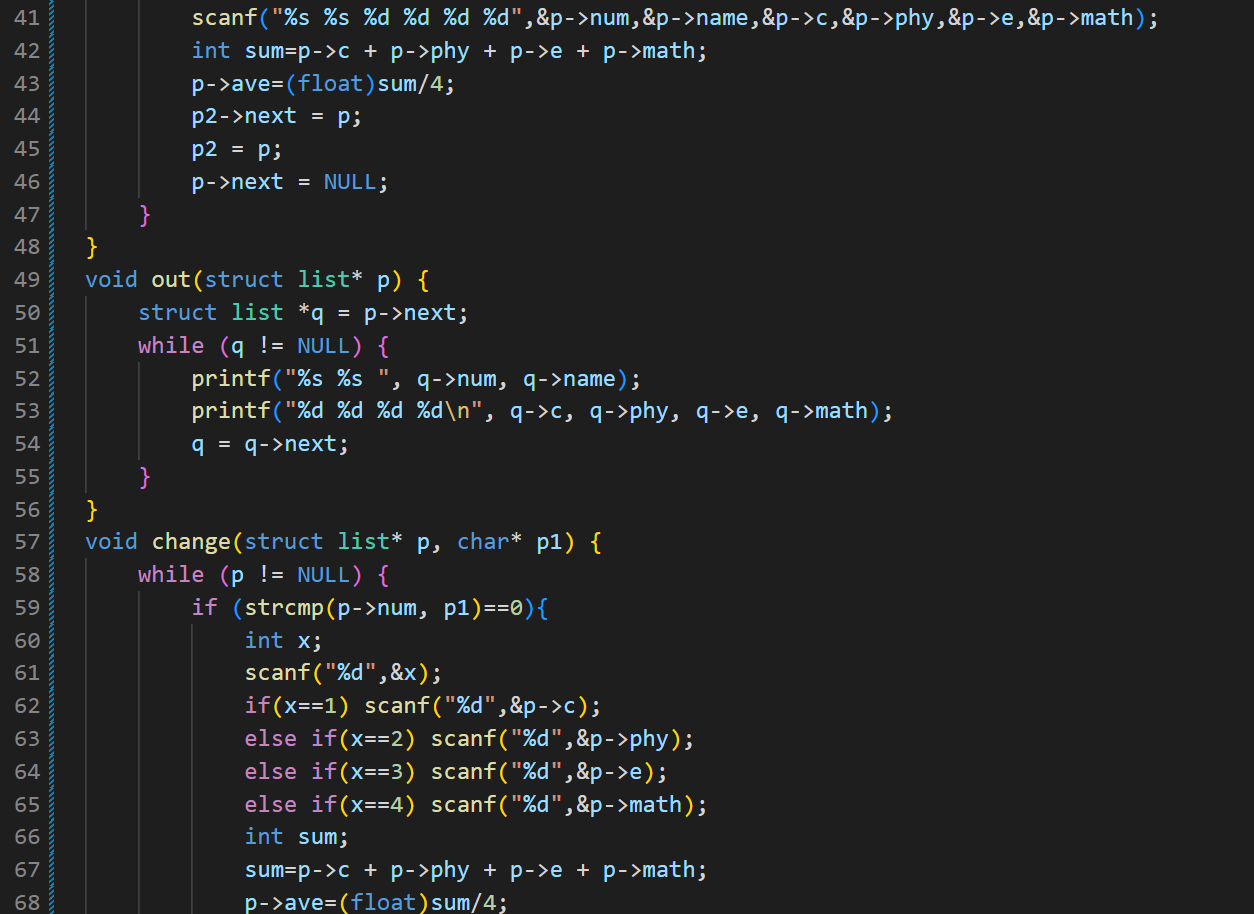
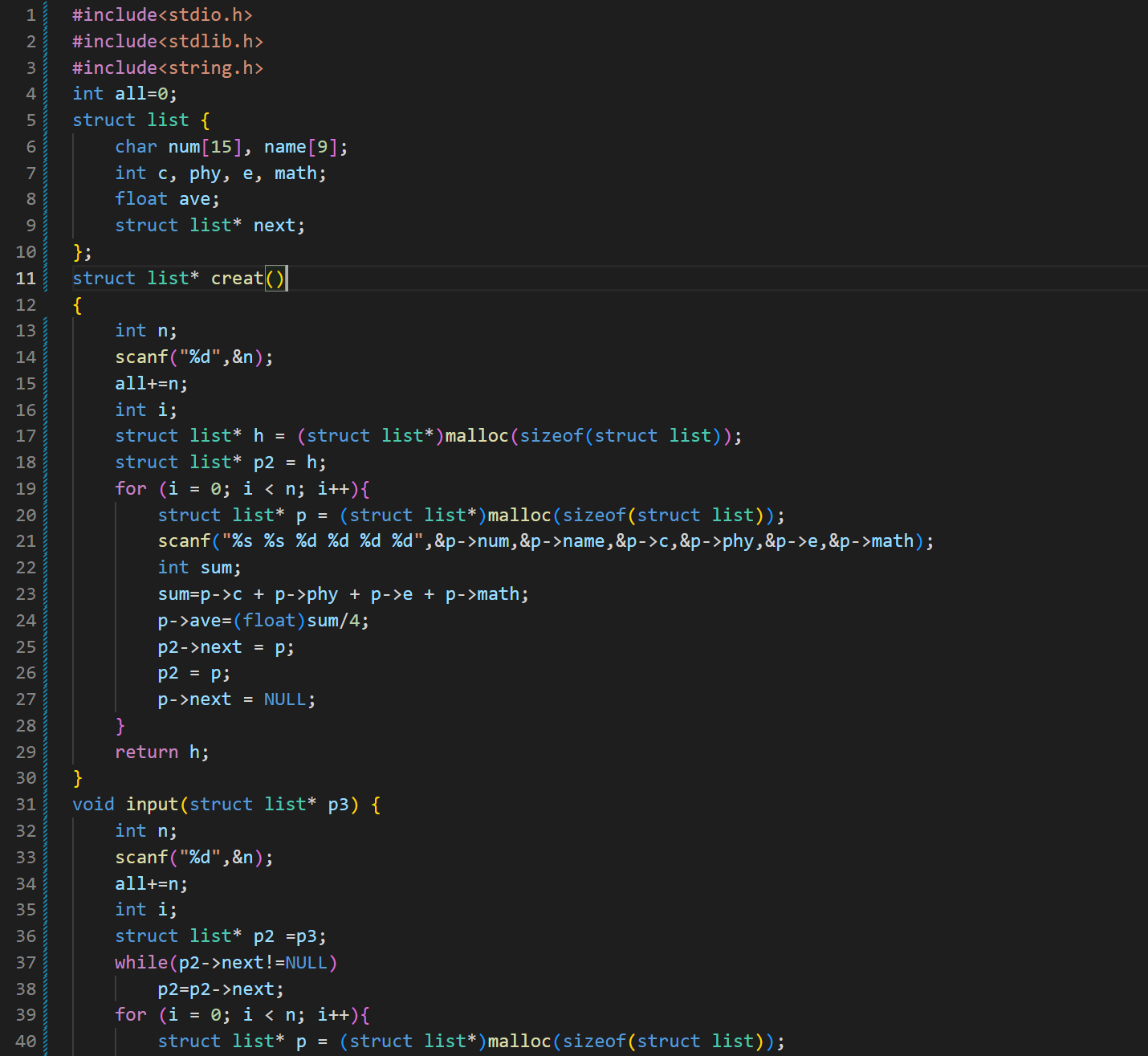


图2-2-9 程序代码

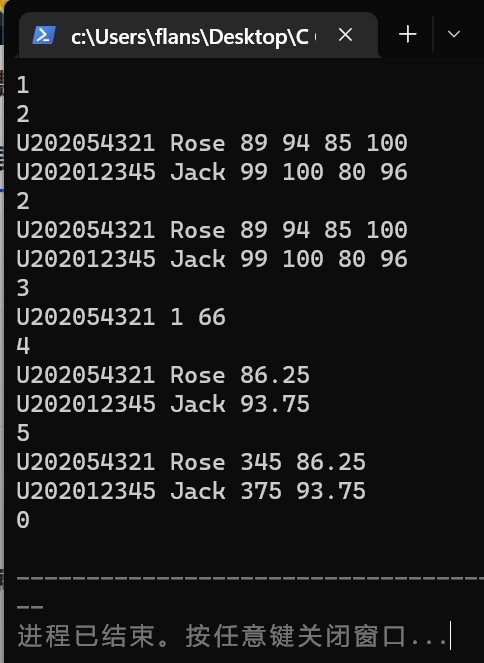


图2-2-10 运行结果

(4) 回文字符串是正读和反读都相同的字符串，例如“abba”和“abcba”是回文字符串。设计程序判断输入的任意长度的字符串是否为回文字符串。提示：由于要求字符串长度任意，所以用单链表存储字符串，即判断一个单链表是否为回文链表。

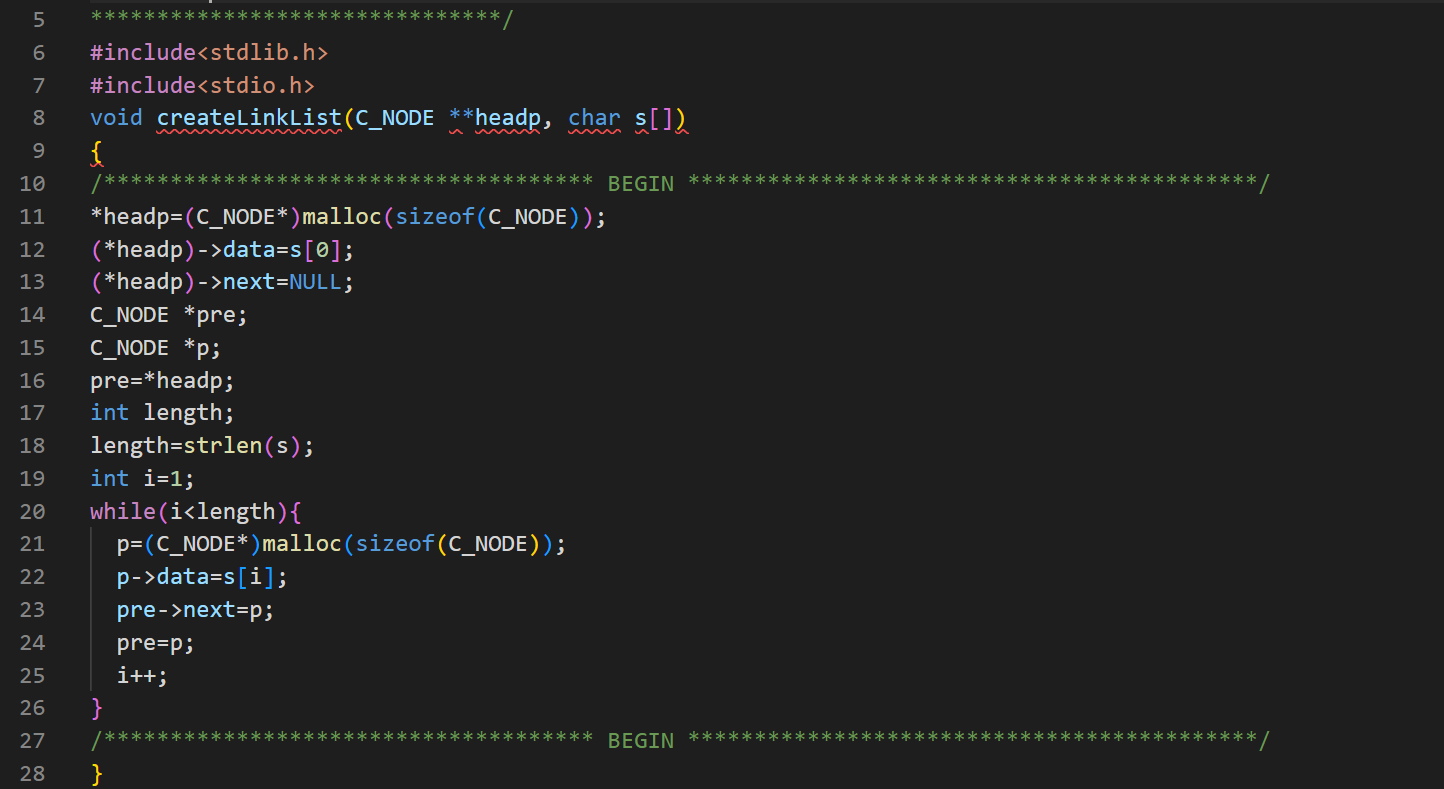


图2-2-11 程序代码

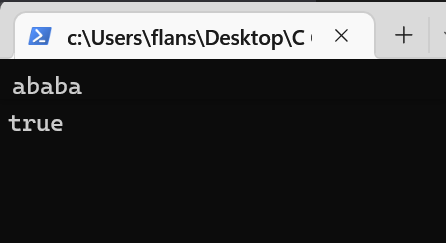


图2-2-12 运行结果

(5) 利用值栈对逆波兰表达式进行求值。逆波兰表达式从键盘输入，其中的运算符仅包含加、减、乘、除4种运算，表达式中的数都是十进制数，用换行符结束输入。由于逆波兰表达式的长度不限，所以值栈要用后进先出链表实现。

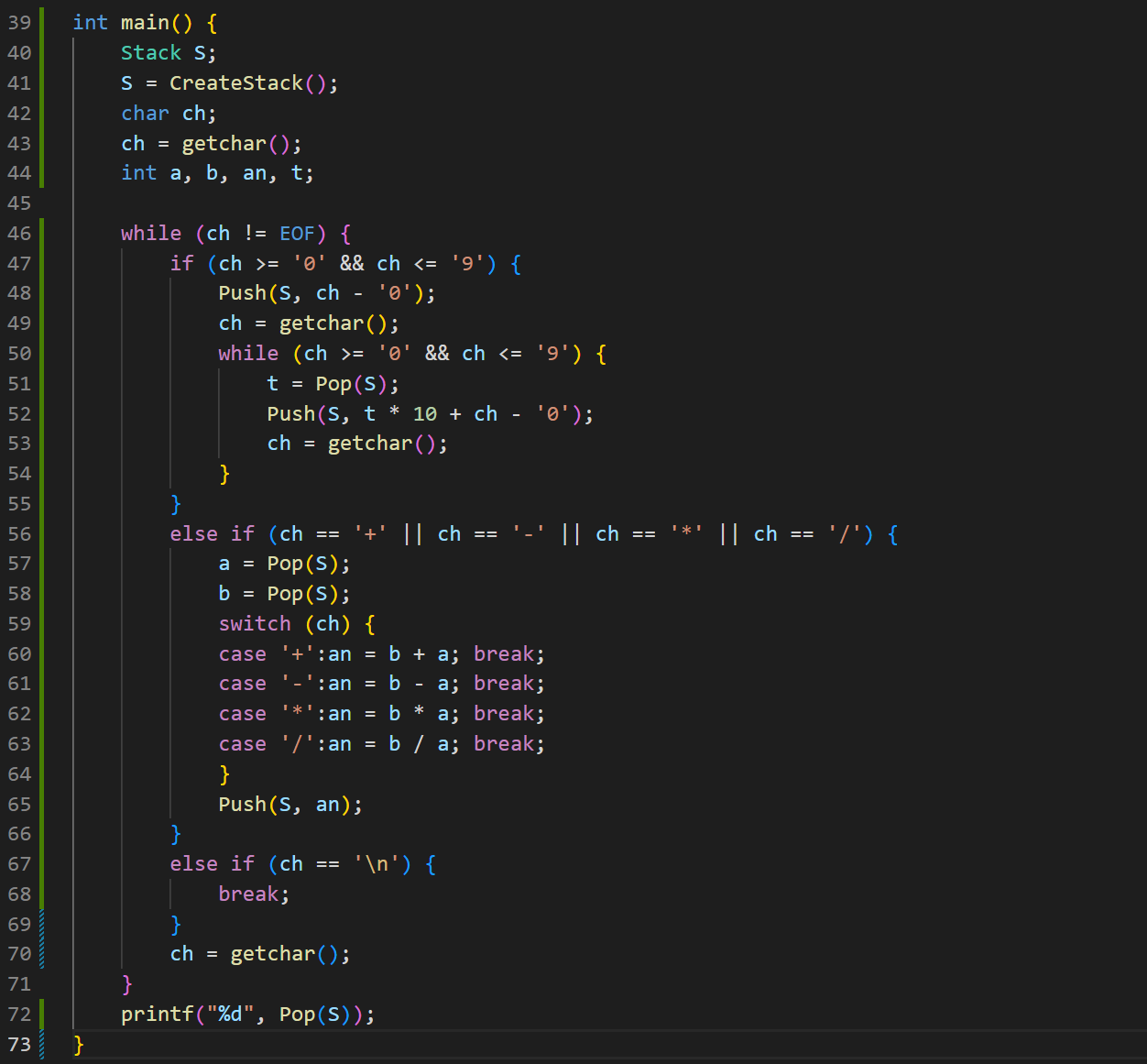


图2-2-13 程序代码

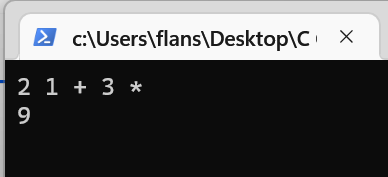


图2-2-14 运行结果

## 2.4小结

本次实验主要学习了结构体、结构体指针的使用，字段结构的声明和使用，联合的声明和使用，链表的声明和使用。结构体是C语言中一种重要的构造类型，借助结构体，可以实现链表等多种复杂的数据类型，突破数组仅能存储相同类型数据的限制。结构体和结构指针的应用，也可以达到接近面对对象语言中“对象”的效果。

在实际运用中，我发现字段结构类型变量的使用具有一定的局限性，如：当比特数小于一字节时，无法通过键访，指针等方式进行引用，进而进行遍历操作。另外，链表在进行遍历和删除操作时，虽然效率比压缩数组高很多，但是将对记数造成很大影响，（例如排序时）必须对引入的计数变量进行更复杂的操作。

# 参考文献

[1] 卢萍,李开,王多强，甘早斌. C语言程序设计典型题解与实验指导,北京：清华大学出版社,2019