# 实验7 结构与联合实验

## 7.1 实验目的

1．通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。

2．通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。

3．了解字段结构和联合的用法。

## 7.2 实验内容

**1．表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

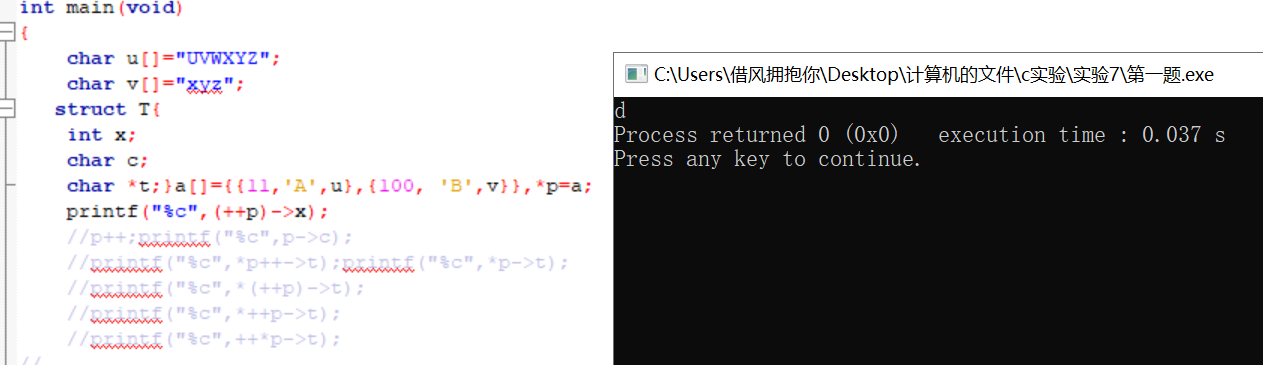
}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

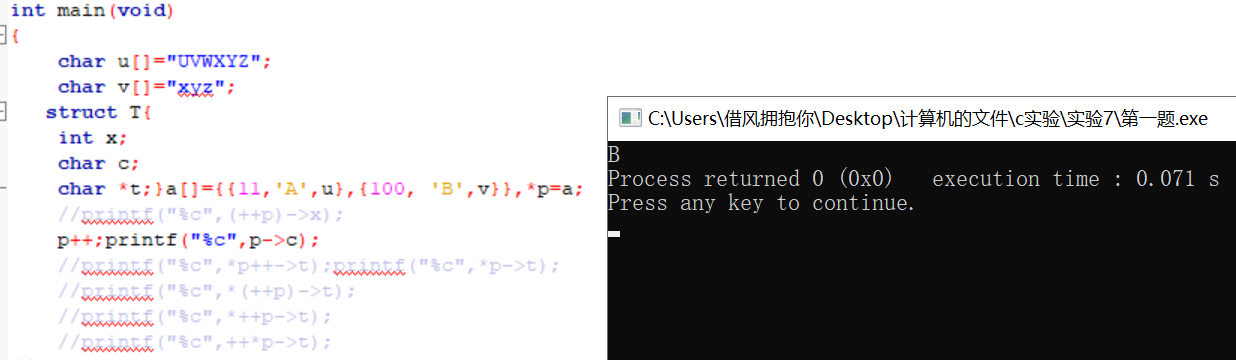
请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

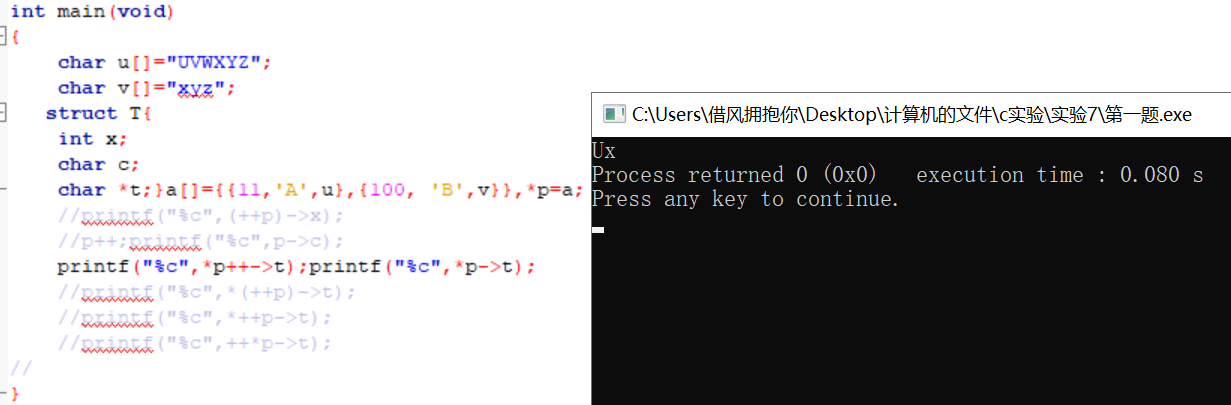
|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | ‘B’ | ‘B’ |
| 3 | \*p++->t,\*p->t | ‘UVWXYZ’ | ‘U’ |
| 4 | \*(++p)->t | ‘xyz’ | ‘x’ |
| 5 | \*++p->t | ‘V’ | ‘V’ |
| 6 | ++\*p->t | ‘y’ | ‘V’ |

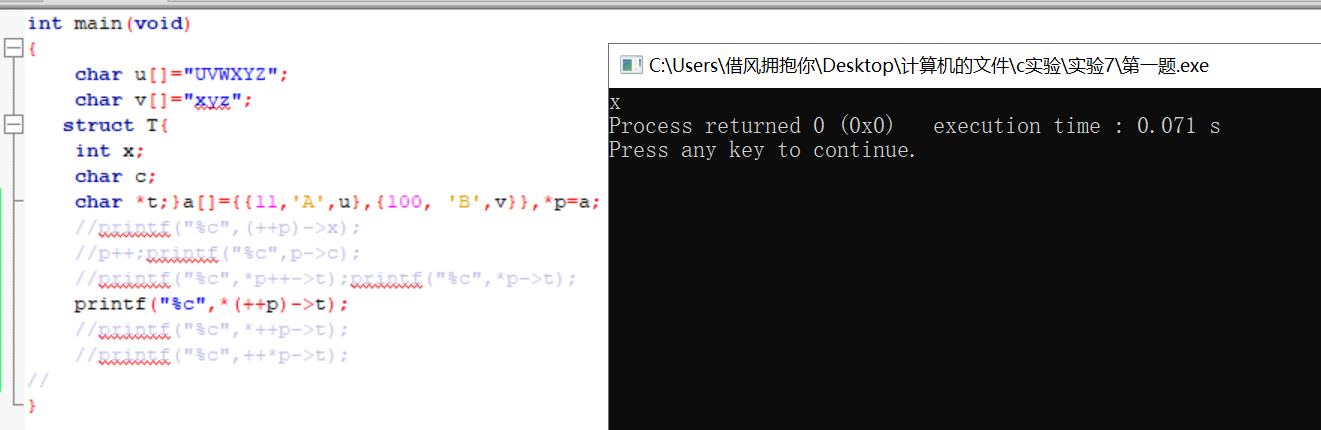
**解答：**

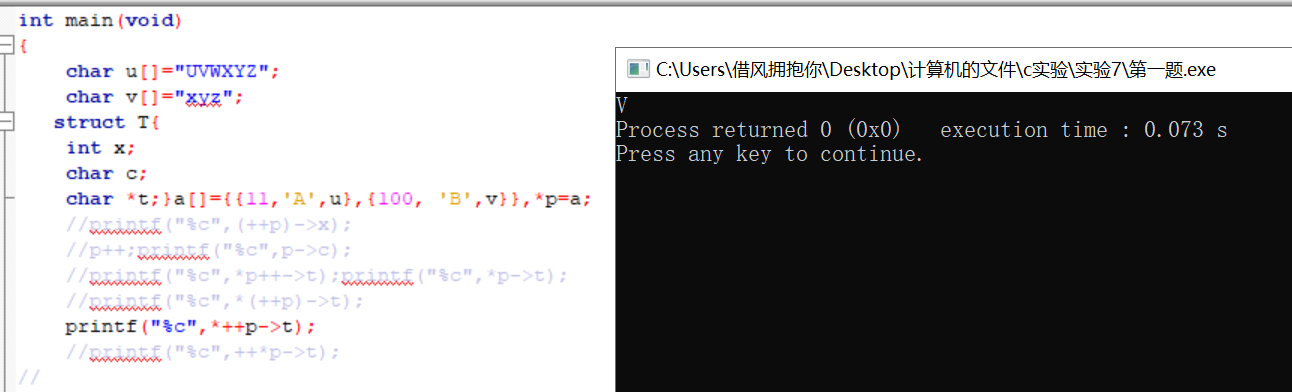
验证值的截图：

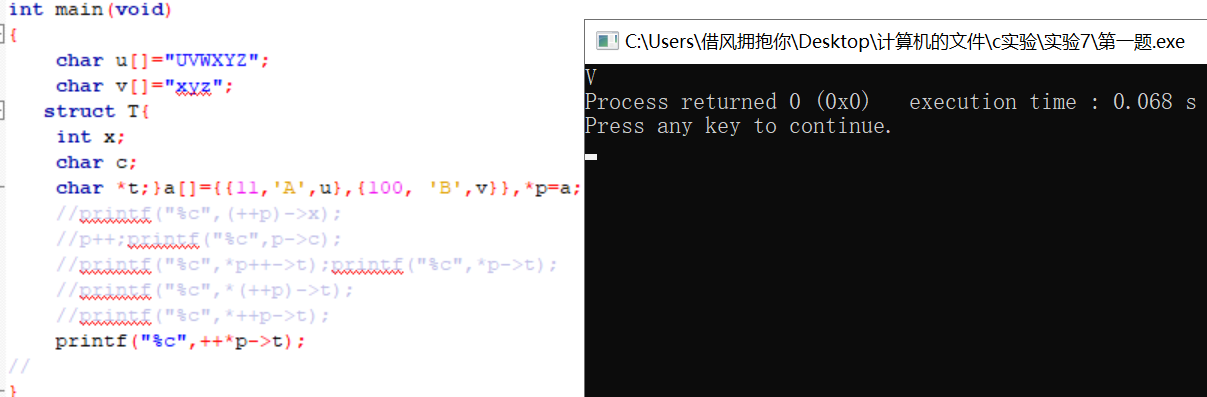
 **序号一的验证值截图**

 **序号二的验证值截图**

 **序号三的验证值截图**

 **序号四的验证值截图**

 **序号五的验证值截图**

 **序号六的验证值截图**

### 2. 源程序修改替换

给定一批整数，以0作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的指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。

1. 源程序中存在什么样的错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

void create\_list(struct s\_list \*headp,int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

（2）修改替换create\_list函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

**解答：**

1. 根据debug测试结果，发现在 create list（head，s）函数前后 head 指针的值没有任何变化，得出结论：在create list 函数中并没有将head指针指向创建的链表表头。分析得：head指针变量的值当作实参传给形参headp，但并没有在head指针变量的地址上进行操作，改变head指针的指向。

修改措施：

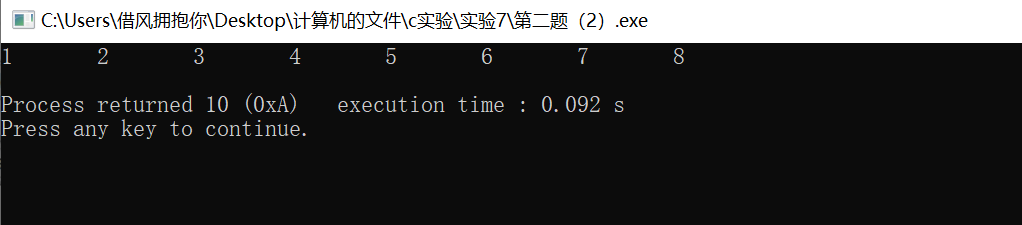
1. Main函数中调用的函数返回值赋予p：

p=create\_list(head,s); /\* 创建新链表 \*/

1. create\_list()尾加入返回值

return loc\_head;

修改后截图：



1. 解题思路：

建立链表时，从头指针开始，创建的下一个指针temp的temp->next指向前一个建立的指针，使得头指针始终指向最后建立的节点。

程序清单：

#include "stdio.h"

#include "stdlib.h"

struct s\_list

{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*\*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(&head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

void create\_list(struct s\_list \*\*headp,int \*p)

{

struct s\_list \* loc\_head,\*pre,\*t=NULL,\*h;

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++;

pre=loc\_head;

while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

h=(struct s\_list \*)malloc(sizeof(struct s\_list));

h->next=pre;

h->data=\*p++;

pre=h;

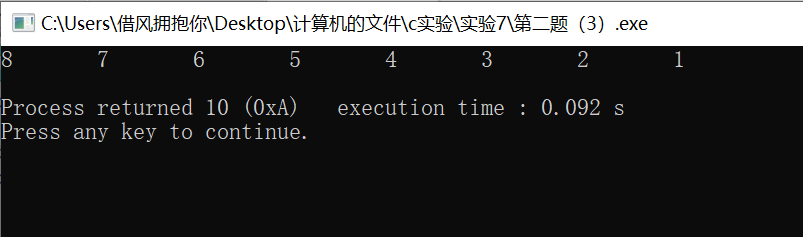
}

loc\_head->next=NULL;

}

\*headp=pre; /\* 使头指针headp指向新创建的链表 \*/

}

运行截图：  


### 3． 编程设计题

（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向最高位声明为8个字段，各字段依次为bit0, bit1, …, bit7，且bit0的优先级最高。同时设计8个函数，第i个函数以biti(i=0,1,2,…,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依次调用函数指针数组p\_fun中相应元素指向的函数。8个函数中的第0个函数可以设计为：

void f0(struct bits b)

{

Printf(“the function %d is called!\n”,b);

}

**解答：**

1. 解题思路：
2. 建立一个字段结构，每个字段宽度为一（一位）
3. 建立一个数组指针函数void （\*p[8]）(struct bits b);

同时给p[8]指定指向函数f0，f1，f2，f3，f4，f5，f6，f7

1. 根据输入的字符转化为二进制并赋予字段结构的每个字段
2. 根据每个字段进行输出函数。

2）源程序清单

#include "stdio.h"

#include "stdlib.h"

struct bits

{

unsigned short int bit0:1;

unsigned short int bit1:1;

unsigned short int bit2:1;

unsigned short int bit3:1;

unsigned short int bit4:1;

unsigned short int bit5:1;

unsigned short int bit6:1;

unsigned short int bit7:1;

};

void f0(struct bits b);

void f1(struct bits b);

void f2(struct bits b);

void f3(struct bits b);

void f4(struct bits b);

void f5(struct bits b);

void f6(struct bits b);

void f7(struct bits b);

int main(void)

{

int i=0;

struct bits simple1={1,0,1,0,1,0,1,1};

struct bits \*p=&simple1;

void (\*p\_fun[8])(unsigned short int);

p\_fun[0]=f0;

p\_fun[1]=f1;

p\_fun[2]=f2;

p\_fun[3]=f3;

p\_fun[4]=f4;

p\_fun[5]=f5;

p\_fun[6]=f6;

p\_fun[7]=f7;

if(p->bit0) p\_fun[0](p->bit0);

if(p->bit1) p\_fun[1](p->bit1);

if(p->bit2) p\_fun[2](p->bit2);

if(p->bit3) p\_fun[3](p->bit3);

if(p->bit4) p\_fun[4](p->bit4);

if(p->bit5) p\_fun[5](p->bit5);

if(p->bit6) p\_fun[6](p->bit6);

if(p->bit7) p\_fun[7](p->bit7);

return 0;

}

void f0(struct bits b)

{

printf("0the function %d is called!\n",b);

}

void f1(struct bits b)

{

printf("1the function %d is called!\n",b);

}

void f2(struct bits b)

{

printf("2the function %d is called!\n",b);

}

void f3(struct bits b)

{

printf("3the function %d is called!\n",b);

}

void f4(struct bits b)

{

printf("4the function %d is called!\n",b);

}

void f5(struct bits b)

{

printf("5the function %d is called!\n",b);

}

void f6(struct bits b)

{

printf("6the function %d is called!\n",b);

}

void f7(struct bits b)

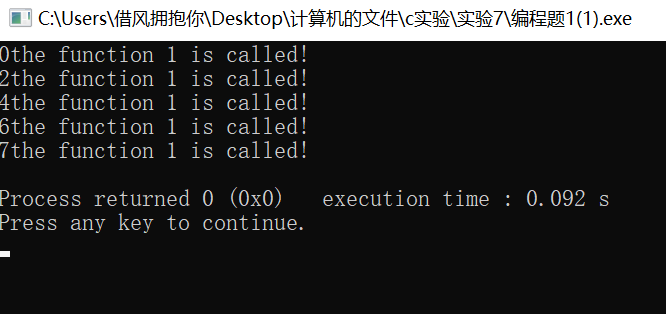
{

printf("7the function %d is called!\n",b);

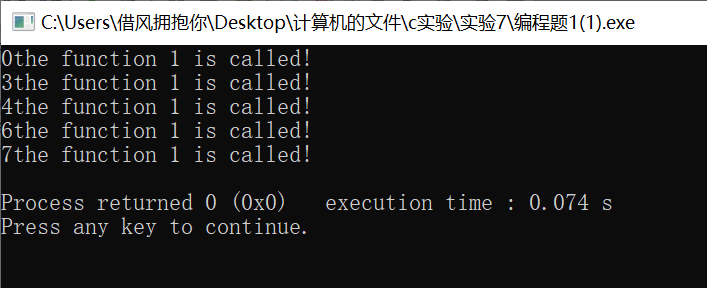
}

3）测试

|  |  |  |  |
| --- | --- | --- | --- |
| **用例** | **二进制** | **理论输出** | **实际输出** |
| 用例1 | 1,0,1,0,1,0,1,1 | 0,2,4,6,7 |  |
| 用例2 | 1,0,0,1,1,0,1,1 | 0,3,4,6,7 |  |



用例一截图



用例二截图

（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语言程序设计四门课程的成绩。用函数编程实现下列功能：

(1) 输入每个学生的各项信息。

(2) 输出每个学生的各项信息。

(3) 修改指定学生的指定数据项的内容。

(4) 统计每个同学的平均成绩（保留2位小数）。

(5) 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

(6) 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。

(7) 进一步写出用交换结点指针域的方法升序排序的函数。

**\*本题将选作一置入其中：**

**解答：**

1) 解题思路：

建立一个记录学生信息的结构体和一个学生结构体（该结构体包含学生信息结构体变量，同时有指向下一个学生的指针next。

依次输入学生学号,姓名,成绩

当输入学生学号为零时结束输入

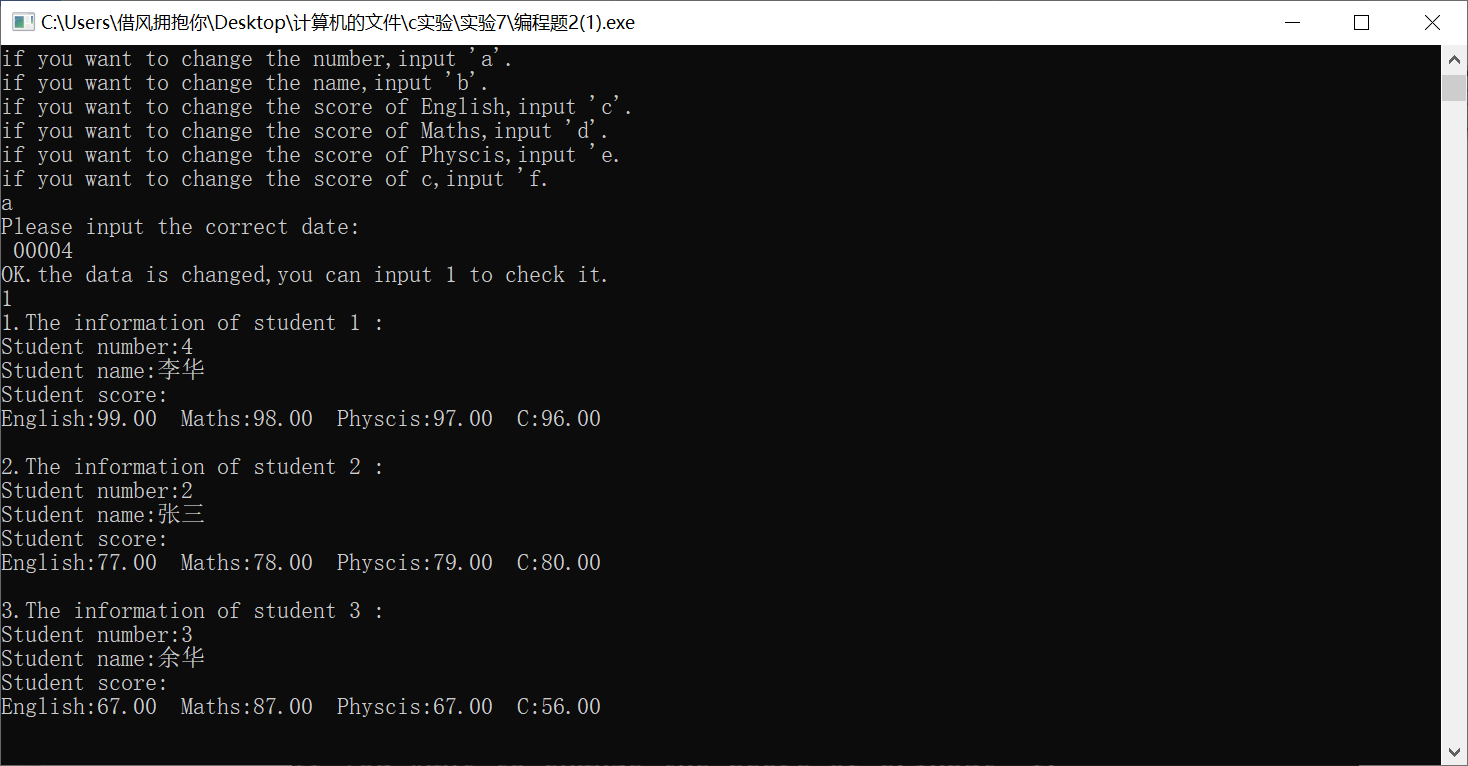
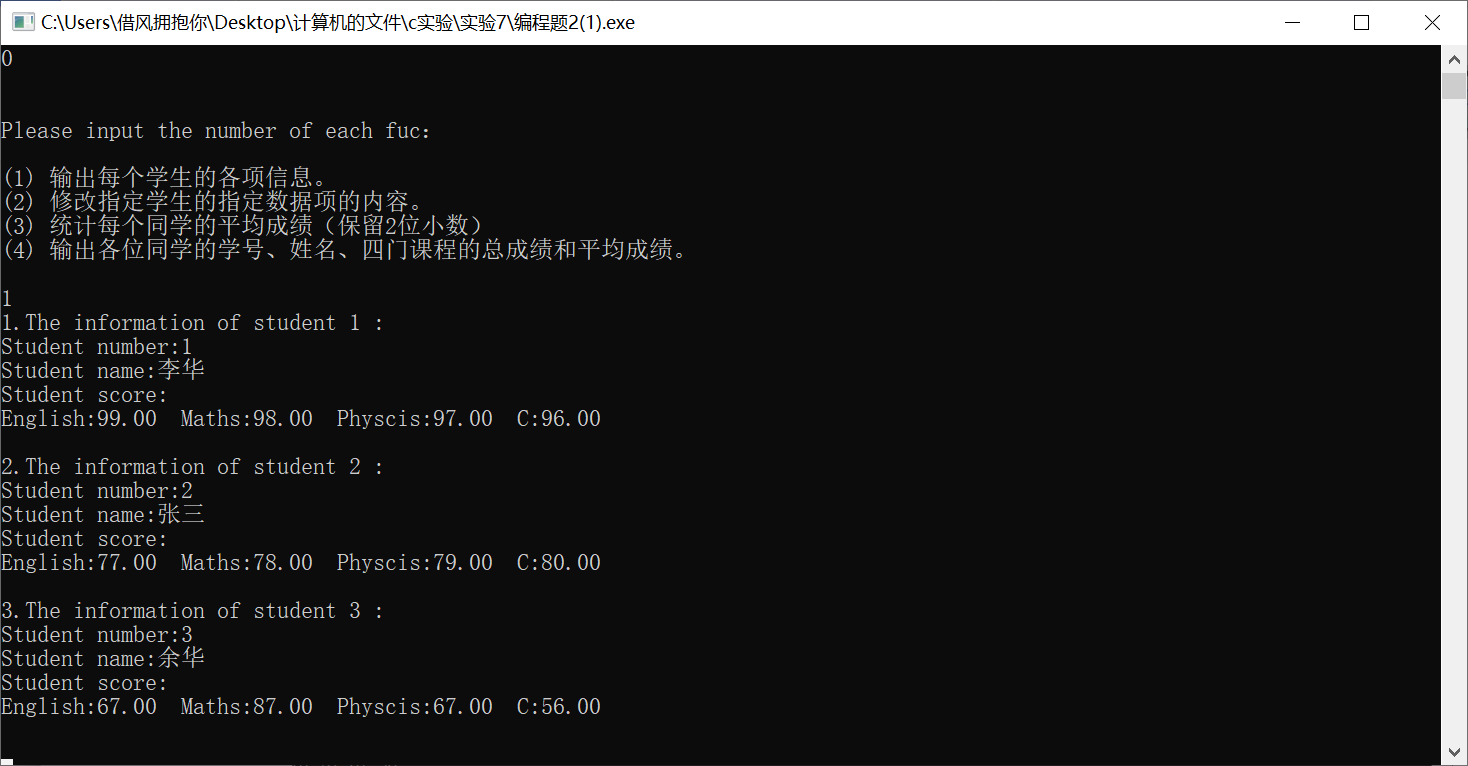
指针域的排序是将两个节点的顺序通过指针的转化来变化，新建立两个临时的指针变量来存储指向当前节点的p指针，和指向p->next->next的指针q，经过变化，是两节点的连接顺序改变。

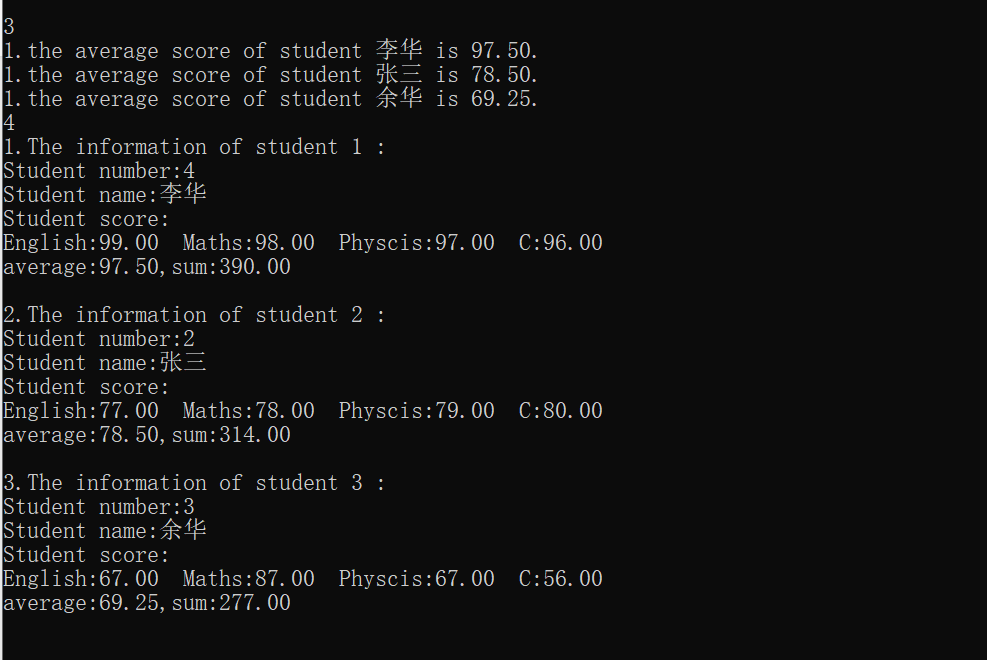
2）程序清单

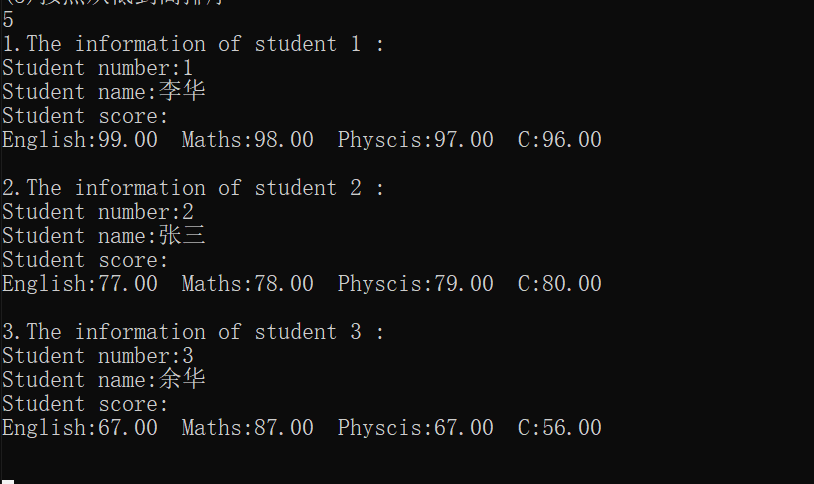
程序过长（放置在文件中，略）

程序运行截图：









## 1.3 实验小结

本次实验的难点在于结构体指针变量的理解和链表的建立与遍历，尤其需要第一题的对指针更加充分的了解，一开始对于它的了解有点不太熟悉，后来通过实践进一步加深了理解。

第三题的难点在于结构体的构建以及链表的交换数据，以及更改链表的数值，在不断的摸索之后更加充分了解了双向链表的删除增添。以及链表的排序。

经过这次试验，我收获匪浅。