# 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

7.2.1 表达式求值得程序验证题

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 | U，x | U，x |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

解答:

用于验证的源程序:

#include<stdio.h>

int main(void)

{

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,'A',u},{100, 'B',v}},\*p=a;

//printf("%d",((++p)->x));

//p++;printf("%d",p->c);

//printf("%d",\*p++->t);printf("%d",\*p->t);

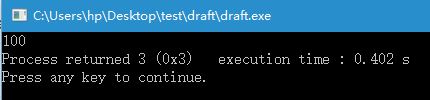
//printf("%d",\*(++p)->t);

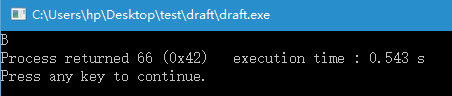
///printf("%d",\*++p->t);

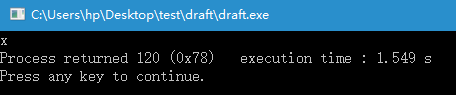
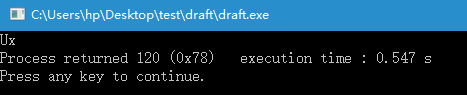
//printf("%c",++\*p->t);

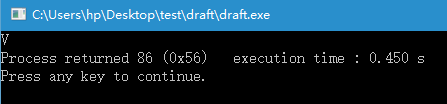
}

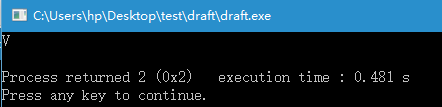
依次消去每行注释后程序运行结果如图:











验证了结果的正确性。

7.2.2 源程序修改替换题

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）函数void create\_list(struct s\_list \*headp,int \*p)的形参struct s\_list \*headp无法将创建的链表传递出函数，应使用void create\_list(struct s\_list \*\*headp,int \*p)的函数声明，结尾对地址赋值；

替换后的程序如下所示：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data;

struct s\_list \*next;

};

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};

create\_list(&head,s);

p=head;

while(p){

printf("%d\t",p->data);

p=p->next;

}

printf("\n");

}

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

{

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

if(p[0]==0)

;

else {

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

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

tail=loc\_head;

while(\*p){

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

tail=tail->next;

tail->data=\*p++;

}

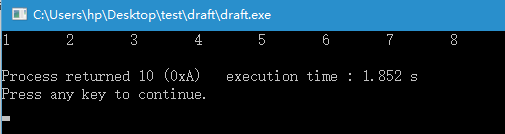
tail->next=NULL;

}

\*headp=loc\_head;

}

运行截图：



（2）修改函数void create\_list(struct s\_list \*\*headp,int \*p)可达到目的，将创建链表部分修改为先创建尾指针，随后逐步向前创建链表，最后创建头指针，即：

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

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

loc\_head->next=NULL;

while(\*p){

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

tail->next=loc\_head;

tail->data=\*p++;

loc\_head=tail;

修改后源程序：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data;

struct s\_list \*next;

};

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};

create\_list(&head,s);

p=head;

while(p){

printf("%d\t",p->data);

p=p->next;

}

printf("\n");

}

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

{

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

if (p[0]==0);

else {

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

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

loc\_head->next=NULL;

while(\*p){

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

tail->next=loc\_head;

tail->data=\*p++;

loc\_head=tail;

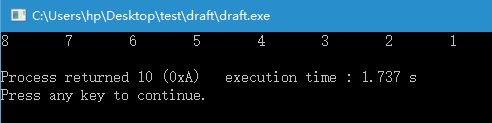
}

}

\*headp=loc\_head;

}

程序运行结果：



7.2.3 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. 提示并输入数据组数N；
3. 输入1个八位无符号字节isr\_reg，其中无符号字节isr\_reg以union形式储存，即

union ISR\_REG{

unsigned short all;

struct bits bit;

};

从而可以通过isr\_reg.bit.biti访问无符号字节八个字段的值；

1. 对无符号字节isr\_reg进行处理，具体流程转3.1；
   1. 设计八个函数用来输出函数被引用的情况，从而能够根据字段biti的值判断是否引用对应函数；
   2. 定义函数指针数组void(\*p\_fun[8])(void)指向八个函数从而便于使用函数；
   3. 通过结构变量bit与联合变量isr\_reg的同时使用，从而能够利用isr\_reg.bit.biti访问无符号字节的各个字段，并在访问后通过函数指针数组引用函数进行输出，体现biti的值；
2. 转2输入下一个八位无符号字节，直至处理完N个字节；
3. 结束。
4. 程序清单：

#include<stdio.h>

struct bits{

unsigned int bit0: 1;

unsigned int bit1: 1;

unsigned int bit2: 1;

unsigned int bit3: 1;

unsigned int bit4: 1;

unsigned int bit5: 1;

unsigned int bit6: 1;

unsigned int bit7: 1;

unsigned int rsv : 8;

};

union ISR\_REG{

unsigned short all;

struct bits bit;

};

void(\*p\_fun[8])(void);

void f0(void){

printf("the function 0 is called!\n");

}

void f1(void){

printf("the function 1 is called!\n");

}

void f2(void){

printf("the function 2 is called!\n");

}

void f3(void){

printf("the function 3 is called!\n");

}

void f4(void){

printf("the function 4 is called!\n");

}

void f5(void){

printf("the function 5 is called!\n");

}

void f6(void){

printf("the function 6 is called!\n");

}

void f7(void){

printf("the function 7 is called!\n");

}

int main(void)

{

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;

union ISR\_REG isr\_reg;

unsigned short N,i;

scanf("%d",&N);

for(i=0;i<N;i++){

scanf("%u",&isr\_reg.all);

printf("%u:\n",isr\_reg.all);

if(isr\_reg.bit.bit0) p\_fun[0]();

if(isr\_reg.bit.bit1) p\_fun[1]();

if(isr\_reg.bit.bit2) p\_fun[2]();

if(isr\_reg.bit.bit3) p\_fun[3]();

if(isr\_reg.bit.bit4) p\_fun[4]();

if(isr\_reg.bit.bit5) p\_fun[5]();

if(isr\_reg.bit.bit6) p\_fun[6]();

if(isr\_reg.bit.bit7) p\_fun[7]();

printf("\n");

}

}

1. 测试
2. 测试数据：

为方便分析测试结果，选择无符号字节分别为0、8、255。如表7-1所示。

表7-1 编程题7.2.3.(1)的测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入（数据组数N=3） | 理 论 结 果 |
| 八位无符号字节 |
| 用例1 | 0 | 没有函数被引用 |
| 用例2 | 8 | 函数f3被引用 |
| 用例3 | 255 | 函数f0、f1、f2、f3、f4、f5、f6、f7被引用 |

1. 对应测试用例的运行结果如图7-1所示。

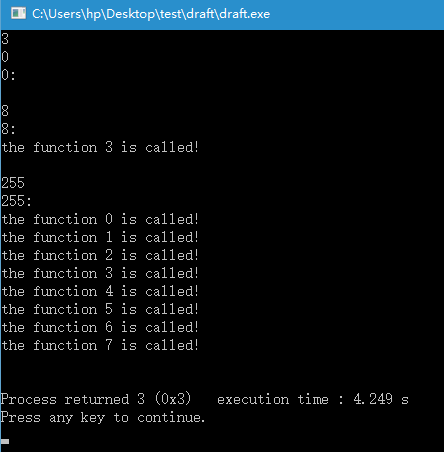


图7-1 编程题7.2.3.(1)的测试用例的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（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。

解答：

1. 解题思路：
2. 定义结构变量score\_tab并命名为studs，每个结构变量中包含学生的各项信息；
3. 创建头指针，即：head=(studs\*)malloc(sizeof(studs))，向头指针数据域中输入数据后，通过for循环继续创建后续指针并向其数据域中赋值，创建完链表后为尾指针赋值为空指针，从而完成要求（1）；
4. 输出表格首行，随后通过for循环遍历链表并输出，即

printf("ID Name English Math Physics C \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

从而完成要求（2）；

1. 自定义函数exchange，通过strcmp判断链表中对应学生所在位置，随后通过strcmp判断对应科目并修改其中数据，即

void exchange(char id[15],char subname[20],float grade)

{

studs \*tail;

tail=head;

while(tail){

if (!strcmp(tail->id,id)) break;

tail=tail->next;

}

if(!strcmp(subname,"English")) tail->score\_en=grade;

else if(!strcmp(subname,"Math")) tail->score\_math=grade;

else if(!strcmp(subname,"Physics")) tail->score\_phy=grade;

else if(!strcmp(subname,"C")) tail->score\_c=grade;

}

随后遍历链表输出从而完成要求（3）；

1. 定义浮点型变量sum、avg，并通过遍历链表完成平均成绩的计算及赋值，即：

float sum,avg;

tail=head;

while(tail){

tail->score\_sum=(tail->score\_en+tail->score\_math+tail->score\_phy+tail->score\_c);

tail->score\_avg=tail->score\_sum/4;

tail=tail->next;

}

从而完成要求（4）；

1. 通过5计算各同学的总成绩和平均成绩后，遍历链表输出指定数据从而完成要求（5），即：

printf("SumAndAvg:\nID Name SUM AVG \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_sum, tail->score\_avg);

tail=tail->next;

}

printf("\n");

1. 结束。
2. 程序清单：

#include<stdio.h>

#include<malloc.h>

#include<string.h>

typedef struct score\_tab{

char id[15];

char name[20];

float score\_en;

float score\_math;

float score\_phy;

float score\_c;

float score\_sum;

float score\_avg;

struct score\_tab \*next;

}studs;

studs \*head=NULL;

void exchange(char id[15],char subname[20],float grade)

{

studs \*tail;

tail=head;

while(tail){

if (!strcmp(tail->id,id)) break;

tail=tail->next;

}

if(!strcmp(subname,"English")) tail->score\_en=grade;

else if(!strcmp(subname,"Math")) tail->score\_math=grade;

else if(!strcmp(subname,"Physics")) tail->score\_phy=grade;

else if(!strcmp(subname,"C")) tail->score\_c=grade;

}

int main (void)

{

int N,i,j;

scanf("%d",&N);

studs \*tail;

head=(studs\*)malloc(sizeof(studs));

//input

scanf("%s%s%f%f%f%f", head->id, head->name, &head->score\_en, &head->score\_math, &head->score\_phy, &head->score\_c);

tail=head;

for(i=1;i<N;i++){

tail->next=(studs\*)malloc(sizeof(studs));

tail=tail->next;

scanf("%s%s%f%f%f%f", tail->id, tail->name, &tail->score\_en, &tail->score\_math, &tail->score\_phy, &tail->score\_c);

}

tail->next=NULL;

//output\_first

printf("ID Name English Math Physics C \n");

tail=head;

while(tail){ //or for(i=0;i<N;i++)

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

printf("\n");

//exchange\_the\_tab

int change\_num;

char change\_id[15];

char change\_subname[10];

float change\_grade;

scanf("%d",&change\_num);

while(change\_num--){

scanf("%s%s%f",change\_id,change\_subname,&change\_grade);

exchange(change\_id,change\_subname,change\_grade);

}

//output\_after\_exchange

printf("Alter:\nID Name English Math Physics C \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

printf("\n");

//calculate\_sum\_and\_avg

float sum,avg;

tail=head;

while(tail){

tail->score\_sum=(tail->score\_en+tail->score\_math+tail->score\_phy+tail->score\_c);

tail->score\_avg=tail->score\_sum/4;

tail=tail->next;

}

//output\_after\_cal

printf("SumAndAvg:\nID Name SUM AVG \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_sum, tail->score\_avg);

tail=tail->next;

}

printf("\n");

}

1. 测试
2. 测试数据

测试用例：依次输入五位同学的学号、姓名、四门科目的成绩，即：

第一位同学：学号U20140101、姓名ZhangChuanChao、英语成绩85、数学成绩86、物理成绩87、C语言成绩88；

第二位同学：学号U20140126、姓名MaiDouDou、英语成绩99、数学成绩99、物理成绩99、C语言成绩99；

第三位同学：学号U20140158、姓名XiaoDouDou、英语成绩56、数学成绩85、物理成绩89、C语言成绩59；

第四位同学：学号U20140312、姓名DaoDaoDog、英语成绩84、数学成绩89、物理成绩65、C语言成绩100；

第五位同学：学号U20140359、姓名XiDaDa、英语成绩88.8、数学成绩88.8、物理成绩88.8、C语言成绩88.8；

用以修改的数据（3组）：

U20140101 Math 95.6（将学号为U20140101的同学的数学成绩修改为95.6）;

U20140359 C 100（将学号为U20140359的同学的C语言成绩修改为100）;

U20140359 English 100（将学号为U20140359的同学的英语成绩修改为100）;

理论输出：5位学生的各项信息，修改后的五位学生的各项信息，五位学生的学号、姓名以及总分均分；

1. 对应测试测试用例的运行结果如图7-2所示。

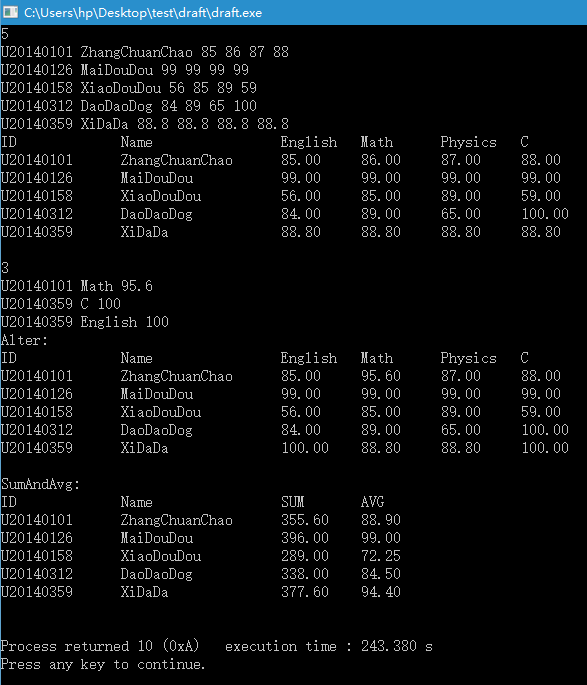


图7-2 编程题7.3.2.(2)的测试用例的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

7.2.4 选做题

（1）对编程设计题第（2）题的程序，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。

解答：

1. 解题思路：

自定义函数sort\_list\_data实现通过交换结点数据域从而升序排序链表内数据，即：

void sort\_list\_data(studs \*head)

{

studs \*p1=head,\*p2;

int len=0,i,j;

char char\_temp[20];

float temp;

while(p1){

len++;

p1=p1->next;

}

for(i=0,p1=head ; i<len-1 ; i++,p1=p1->next)

for(j=i+1,p2=p1->next ; j<len ; j++,p2=p2->next)

if(p1->score\_avg>p2->score\_avg){

//id

strcpy(char\_temp,p1->id);

strcpy(p1->id,p2->id);

strcpy(p2->id,char\_temp);

//name

strcpy(char\_temp,p1->name);

strcpy(p1->name,p2->name);

strcpy(p2->name,char\_temp);

//avg

temp=p1->score\_avg;

p1->score\_avg=p2->score\_avg;

p2->score\_avg=temp;

//sum

temp=p1->score\_sum;

p1->score\_sum=p2->score\_sum;

p2->score\_sum=temp;

//en

temp=p1->score\_en;

p1->score\_en=p2->score\_en;

p2->score\_en=temp;

//math

temp=p1->score\_math;

p1->score\_math=p2->score\_math;

p2->score\_math=temp;

//phy

temp=p1->score\_phy;

p1->score\_phy=p2->score\_phy'

p2->score\_phy=temp;

//c

temp=p1->score\_c;

p1->score\_c=p2->score\_c;

p2->score\_c=temp;

}

}

将其添加到编程题（2）的程序主函数之前。修改数据后引用函数并遍历链表输出即可得到升序排序后的链表。

1. 程序清单：

#include<stdio.h>

#include<malloc.h>

#include<string.h>

typedef struct score\_tab{

char id[15];

char name[20];

float score\_en;

float score\_math;

float score\_phy;

float score\_c;

float score\_sum;

float score\_avg;

struct score\_tab \*next;

}studs;

studs \*head=NULL;

void exchange(char id[15],char subname[20],float grade)

{

studs \*tail;

tail=head;

while(tail){

if (!strcmp(tail->id,id)) break;

tail=tail->next;

}

if(!strcmp(subname,"English")) tail->score\_en=grade;

else if(!strcmp(subname,"Math")) tail->score\_math=grade;

else if(!strcmp(subname,"Physics")) tail->score\_phy=grade;

else if(!strcmp(subname,"C")) tail->score\_c=grade;

}

void sort\_list\_data(studs \*head)

{

studs \*p1=head,\*p2;

int len=0,i,j;

char char\_temp[20];

float temp;

while(p1){

len++;

p1=p1->next;

}

for(i=0,p1=head ; i<len-1 ; i++,p1=p1->next)

for(j=i+1,p2=p1->next ; j<len ; j++,p2=p2->next)

if(p1->score\_avg>p2->score\_avg){

//id

strcpy(char\_temp,p1->id);

strcpy(p1->id,p2->id);

strcpy(p2->id,char\_temp);

//name

strcpy(char\_temp,p1->name);

strcpy(p1->name,p2->name);

strcpy(p2->name,char\_temp);

//avg

temp=p1->score\_avg;

p1->score\_avg=p2->score\_avg;

p2->score\_avg=temp;

//sum

temp=p1->score\_sum;

p1->score\_sum=p2->score\_sum;

p2->score\_sum=temp;

//en

temp=p1->score\_en;

p1->score\_en=p2->score\_en;

p2->score\_en=temp;

//math

temp=p1->score\_math;

p1->score\_math=p2->score\_math;

p2->score\_math=temp;

//phy

temp=p1->score\_phy;

p1->score\_phy=p2->score\_phy'

p2->score\_phy=temp;

//c

temp=p1->score\_c;

p1->score\_c=p2->score\_c;

p2->score\_c=temp;

}

}

int main (void)

{

int N,i,j;

scanf("%d",&N);

studs \*tail;

head=(studs\*)malloc(sizeof(studs));

//input

scanf("%s%s%f%f%f%f", head->id, head->name, &head->score\_en, &head->score\_math, &head->score\_phy, &head->score\_c);

tail=head;

for(i=1;i<N;i++){

tail->next=(studs\*)malloc(sizeof(studs));

tail=tail->next;

scanf("%s%s%f%f%f%f", tail->id, tail->name, &tail->score\_en, &tail->score\_math, &tail->score\_phy, &tail->score\_c);

}

tail->next=NULL;

//output\_first

printf("ID Name English Math Physics C \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

printf("\n");

//exchange\_the\_tab

int change\_num;

char change\_id[15];

char change\_subname[10];

float change\_grade;

scanf("%d",&change\_num);

while(change\_num--){

scanf("%s%s%f",change\_id,change\_subname,&change\_grade);

exchange(change\_id,change\_subname,change\_grade);

}

//output\_after\_exchange

printf("Alter:\nID Name English Math Physics C \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

printf("\n");

//calculate\_sum\_and\_avg

float sum,avg;

tail=head;

while(tail){

tail->score\_sum=(tail->score\_en+tail->score\_math+tail->score\_phy+tail->score\_c);

tail->score\_avg=tail->score\_sum/4;

tail=tail->next;

}

//output\_after\_cal

printf("SumAndAvg:\nID Name SUM AVG \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_sum, tail->score\_avg);

tail=tail->next;

}

printf("\n");

//Sort

sort\_list\_data(head);

//output\_after\_sort

printf("Sort:\nID Name AVG \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f\n",tail->id, tail->name, tail->score\_avg);

tail=tail->next;

}

printf("\n");

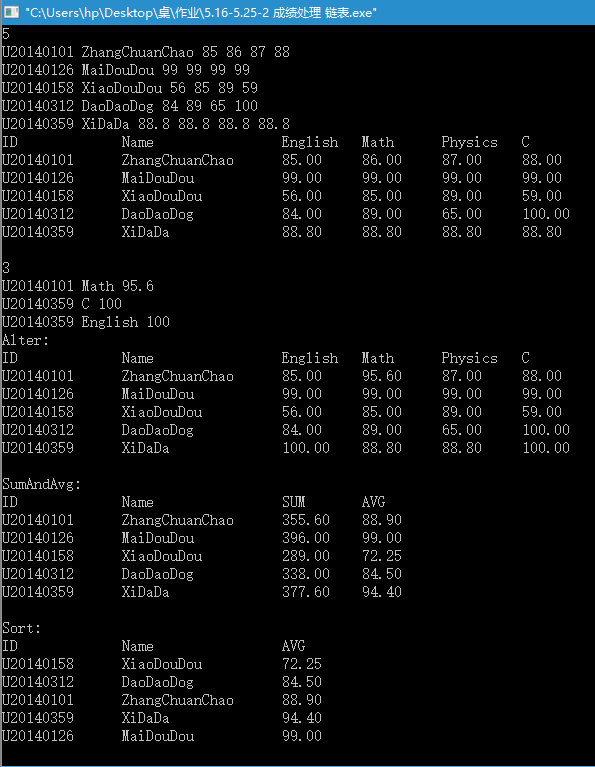
}

1. 测试：

测试用例与编程题（2）相同。

理论输出：尾部增加一个命名为Sort的输出，输出按平均分排序后的学生学号姓名以及平均成绩；

程序运行结果如图：



说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（2）对选做题第（1）题，进一步写出用交换结点指针域的方法升序排序的函数。

解答：

1. 解题思路：

自定义函数sort\_list\_point通过交换结点指针域来实现升序排序，即：

void sort\_list\_point(studs \*\*head)

{

studs \*prior1,\*prior2,\*p1,\*p2,\*t;

int i=0;

p1=\*head;

for(p2=p1->next,prior2=p1 ; p2!=NULL ; prior2=p2,p2=p2->next)

if(p1->score\_avg>p2->score\_avg){

prior2->next=p1;

t=p1->next;

p1->next=p2->next;

(\*head)=p2;

p2->next=t;

p1=p2;

}

for(prior1=\*head,p1=prior1->next ; p1->next!=NULL ; prior1=p1,p1=p1->next)

for(p2=p1->next,prior2=p1 ; p2!=NULL ; prior2=p2,p2=p2->next)

if(p1->score\_avg>p2->score\_avg){

t=p2->next;

prior1->next=p2;

prior2->next=p1;

p2->next=p1->next;

p1->next=t;

p1=p2;

}

}

将其添加到编程题（2）的程序主函数之前。修改数据后引用函数并遍历链表输出即可得到升序排序后的链表。

1. 程序清单：

#include<stdio.h>

#include<malloc.h>

#include<string.h>

typedef struct score\_tab{

char id[15];

char name[20];

float score\_en;

float score\_math;

float score\_phy;

float score\_c;

float score\_sum;

float score\_avg;

struct score\_tab \*next;

}studs;

studs \*head=NULL;

void exchange(char id[15],char subname[20],float grade)

{

studs \*tail;

tail=head;

while(tail){

if (!strcmp(tail->id,id)) break;

tail=tail->next;

}

if(!strcmp(subname,"English")) tail->score\_en=grade;

else if(!strcmp(subname,"Math")) tail->score\_math=grade;

else if(!strcmp(subname,"Physics")) tail->score\_phy=grade;

else if(!strcmp(subname,"C")) tail->score\_c=grade;

}

void sort\_list\_point(studs \*\*head)

{

studs \*prior1,\*prior2,\*p1,\*p2,\*t;

int i=0;

p1=\*head;

for(p2=p1->next,prior2=p1 ; p2!=NULL ; prior2=p2,p2=p2->next)

if(p1->score\_avg>p2->score\_avg){

prior2->next=p1;

t=p1->next;

p1->next=p2->next;

(\*head)=p2;

p2->next=t;

p1=p2;

}

for(prior1=\*head,p1=prior1->next ; p1->next!=NULL ; prior1=p1,p1=p1->next)

for(p2=p1->next,prior2=p1 ; p2!=NULL ; prior2=p2,p2=p2->next)

if(p1->score\_avg>p2->score\_avg){

t=p2->next;

prior1->next=p2;

prior2->next=p1;

p2->next=p1->next;

p1->next=t;

p1=p2;

}

}

int main (void)

{

int N,i,j;

scanf("%d",&N);

studs \*tail;

head=(studs\*)malloc(sizeof(studs));

//input

scanf("%s%s%f%f%f%f", head->id, head->name, &head->score\_en, &head->score\_math, &head->score\_phy, &head->score\_c);

tail=head;

for(i=1;i<N;i++){

tail->next=(studs\*)malloc(sizeof(studs));

tail=tail->next;

scanf("%s%s%f%f%f%f", tail->id, tail->name, &tail->score\_en, &tail->score\_math, &tail->score\_phy, &tail->score\_c);

}

tail->next=NULL;

//output\_first

printf("ID Name English Math Physics C \n");

tail=head;

while(tail){ //or for(i=0;i<N;i++)

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

printf("\n");

//exchange\_the\_tab

int change\_num;

char change\_id[15];

char change\_subname[10];

float change\_grade;

scanf("%d",&change\_num);

while(change\_num--){

scanf("%s%s%f",change\_id,change\_subname,&change\_grade);

exchange(change\_id,change\_subname,change\_grade);

}

//output\_after\_exchange

printf("Alter:\nID Name English Math Physics C \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_en, tail->score\_math, tail->score\_phy, tail->score\_c);

tail=tail->next;

}

printf("\n");

//calculate\_sum\_and\_avg

float sum,avg;

tail=head;

while(tail){

tail->score\_sum=(tail->score\_en+tail->score\_math+tail->score\_phy+tail->score\_c);

tail->score\_avg=tail->score\_sum/4;

tail=tail->next;

}

//output\_after\_cal

printf("SumAndAvg:\nID Name SUM AVG \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f%-10.2f\n",tail->id, tail->name, tail->score\_sum, tail->score\_avg);

tail=tail->next;

}

printf("\n");

//Sort

sort\_list\_point(&head);

//output\_after\_sort

printf("Sort:\nID Name AVG \n");

tail=head;

while(tail){

printf("%-15s%-20s%-10.2f\n",tail->id, tail->name, tail->score\_avg);

tail=tail->next;

}

printf("\n");

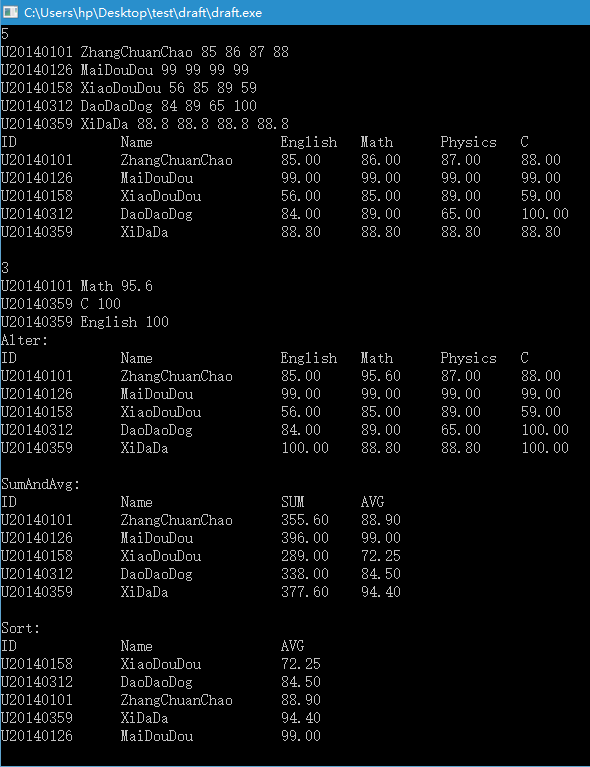
}

1. 测试：

测试用例与编程题（2）相同。

理论输出：尾部增加一个命名为Sort的输出，输出按平均分排序后的学生学号姓名以及平均成绩；

程序运行结果如图：



说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 7.3 自设题

**（1）**自设实验题目：右对齐输出两个链表中的数据

**（2）**实验目的：通过设计实验程序，理解链表存储数据的形式，熟练掌握对链表的相关操作。

**（3）**实验程序：

#include<stdio.h>

#include<malloc.h>

struct s\_list{

int data;

struct s\_list \*next;

};

void foo(struct s\_list \*a,struct s\_list \*b)

{

struct s\_list \*p, \*q, \*k, \*t, \*s;

for (p = a, q = b ; p && q ; p = p->next, q = q->next);

k = p?p:q;

t = p?a:b;

s = p?b:a;

for (; t; printf("%d ", t->data), t = t->next);

printf("\n");

for (; k; printf(" "), k = k->next);

for (; s; printf("%d ", s->data), s = s->next);

}

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

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

if(p[0]==0) ;

else {

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

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

tail=loc\_head;

while(\*p){

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

tail=tail->next;

tail->data=\*p++;

}

tail->next=NULL;

}

\*headp=loc\_head;

}

int main(void)

{ struct s\_list \*m=NULL,\*n=NULL;

int s[]={1,2,3,4,5,6,7,8,0},t[]={8,9,0};

create\_list(&m,s);

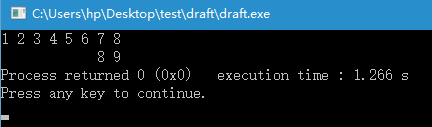
create\_list(&n,t);

foo(m,n);

}

**（4）**实验用例及程序运行结果：

实验用例:程序中构造的两个链表数据分别为1、2、3、4、5、6、7、8，8、9；



**（5）**实验结论：说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 7.4 实验小结

在实验中主要遇到的问题有：

1. 对链表的理解与处理有一定的难度，如在解决7.2.2源程序修改替换题中的问题（1）时，开始没有想到怎样把生成的链表传递出来；又如在解决同一题的问题（2）时并不能马上得到后入先出构建链表的思路，后都在查阅课本理解后完成了程序修改和替换。
2. 在7.2.3程序设计第二题（成绩处理）时，题目提出的要求比较多，完成这样稍微复杂一点的程序时各种各样的小错误都容易对程序造成影响且不容易发现，这就需要程序员对代码进行多次检验。在编程时完成一部分调试一部分，最终完成了程序。
3. 在完成选做题（2）对链表结点进行处理从而升序排列时遇到了一定困难，排序逻辑较复杂，翻书理解后解决问题。

体会：

1. 对链表进行操作时容易涉及到各种间访和取地址的操作，如7.2.2中的源程序修改替换题，既需要注意引用函数能否将函数内的内容导出，又需要注意使用链表完成一个问题可能有多种途径。多多练习才能更熟练地掌握并灵活使用链表；
2. 7.2.3中结构与联合的共同使用体现出C语言的灵活性。通过熟悉联合变量union在内存中的存储方式、引用union变量时的具体流程，再搭配struct结构变量在内存中的存储方式等一系列基础知识，能够巧妙地对数据进行各种处理。另外善于使用联合变量也能够有效的节省内存空间，从而能在大项目中起到重要作用；
3. 链表是C语言中的重点难点之一，善于使用链表可以有效的节省内存空间，也能够更加灵活地使用内存空间，但链表的灵活性也使其使用变得复杂起来。然而熟能生巧，多多练习相关题目便能够熟练掌握包括遍历链表、查找链表、排序链表等一系列的操作。