# 7结构与联合实验

## 7.1 实验目的

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。在实验过程中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。了解字段结构和联合的用法。

## 7.2 必做题

### 7.2.1 表达式求值

**【题目】**

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 |  |  |
| 2 | p++,p->c |  |  |
| 3 | \*p++->t,\*p->t |  |  |
| 4 | \*(++p)->t |  |  |
| 5 | \*++p->t |  |  |
| 6 | ++\*p->t |  |  |

**【计算值】**

表7-1 实验七必做题7.2.1计算值表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 表达式 | 计算值 | 验证值 |
| 1 | (++p)->x | 100 |  |
| 2 | p++,p->c | B |  |
| 3 | \*p++->t,\*p->t | x |  |
| 4 | \*(++p)->t | x |  |
| 5 | \*++p->t | V |  |
| 6 | ++\*p->t | V |  |

**【程序代码】**

#include<stdio.h>

int main()

{

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\n", ((++p)->x)); p = a;

printf("%c\n", (p++, p->c)); p = a;

printf("%c\n", (\*p++->t, \*p->t)); p = a;

printf("%c\n", (\*(++p)->t)); p = a;

printf("%c\n", (\*++p->t)); p = a; p->t--;

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

return 0;

}

**【验证值】**

表7-2 实验七必做题7.2.1验证值表

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

**程序运行图如下图。**

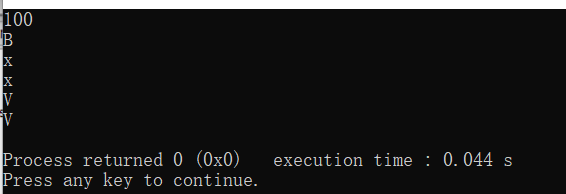


图7-1 实验七必做题7.2.1程序运行图

### 7.2.2 源程序修改替换

**【题目】**

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。

**【错误分析及修正】**

由于C语言使用值传递的方式，故在函数中对形参headp赋值并不能改变其实参的大小，因此该参数没有必要，以返回值的形式返回地址。所以参数类型不应该是struct s\_list \*headp而是struct s\_list \*\*headp，同样的在引用和赋值的时候也需要相应的改变类型。

**【修改后程序】**

(1)

#include<stdio.h>

#include<stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

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

int main(void) {

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

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

creat\_list(&head, s);

p = head;

while(p) {

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

p = p -> next; }

printf("\n");

return 0;

}

void creat\_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;

}

(2)

#include <stdio.h>

#include <stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

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

int main()

{

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

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

creat\_list(&head,s);

p=head;

while(p){

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

p=p->next; }

printf("\n");

return 0;

}

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

{

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

struct s\_list\*temp;

if(p[0]==0 ;

else {

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

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

tail=loc\_head;

while(\*p){

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

temp->next=loc\_head;

loc\_head=temp;

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

tail->next=NULL; }

\*headp=loc\_head;

}

**【运行结果】**

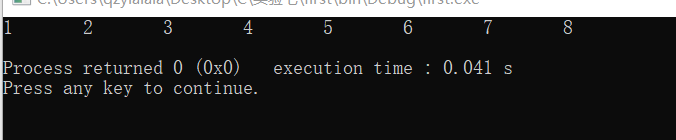


图7-2 实验七必做题7.2.2（1）运行结果

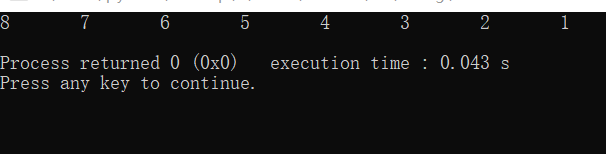


图7-3 实验七必做题7.2.2（2）运行结果

### 7.2.3 字段函数

**【题目】**

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);

}

**【算法流程图】**

本程序为简单的顺序执行。依照题意设置字段结构和函数，优先级可以通过函数出现位置来实现。其他不变，照常输入。

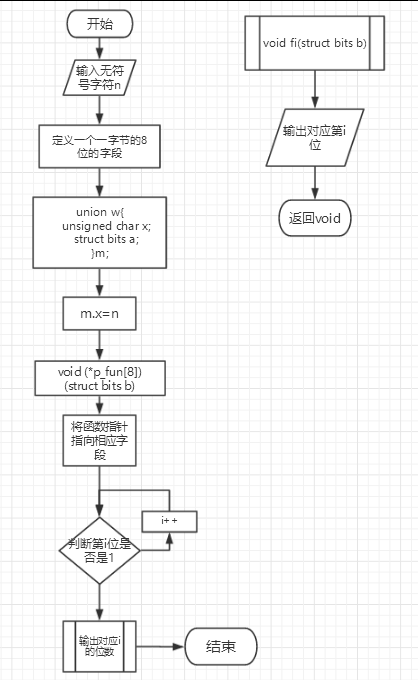


图7-4 实验七必做题7.2.3算法流程图

**【程序清单】**

#include<stdio.h>

struct bits{

unsigned int bit0:1,bit1:1, bit2:1, bit3:1, bit4:1, bit5:1, bit6:1, bit7:1;

};

union w{

unsigned char x;

struct bits a;

}m;

void f0(struct bits b)

{

printf("the function 1 is called! %d\n",b.bit0);

}

void f1(struct bits b)

{

printf("the function 2 is called! %d\n",b.bit1);

}

void f2(struct bits b)

{

printf("the function 3 is called! %d\n",b.bit2);

}

void f3(struct bits b)

{

printf("the function 4 is called! %d\n",b.bit3);

}

void f4(struct bits b)

{

printf("the function 5 is called! %d\n",b.bit4);

}

void f5(struct bits b)

{

printf("the function 6 is called! %d\n",b.bit5);

}

void f6(struct bits b)

{

printf("the function 7 is called! %d\n",b.bit6);

}

void f7(struct bits b)

{

printf("the function 8 is called! %d\n",b.bit7);

}

int main()

{

void (\*p\_fun[8])(struct bits b);

unsigned char n;

printf("Pleas enter a char:");

scanf("%c",&n);

m.x=n;//将其存进联合中

p\_fun[0]=f0;//分别使函数指针指向不同字段

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(m.a.bit0)

p\_fun[0](m.a);

if(m.a.bit1)

p\_fun[1](m.a);

if(m.a.bit2)

p\_fun[2](m.a);

if(m.a.bit3)

p\_fun[3](m.a);

if(m.a.bit4)

p\_fun[4](m.a);

if(m.a.bit5)

p\_fun[5](m.a);

if(m.a.bit6)

p\_fun[6](m.a);

if(m.a.bit7)

p\_fun[7](m.a);

printf("\nPress any key to quit...");

getch();

return 0;

}

**【测试数据、测试理论结果及测试结果】**

根据输入的字符不同种类设计如下三组样例，理论结果与实际结果一致。

1.输入a；

预测输出为

the function 1 is called! 1

the function 6 is called! 1

the function 7 is called! 1

2.输入4；

预测输出为

the function 3 is called! 1

the function 5 is called! 1

the function 6 is called! 1

3.输入？；

预测输出为

the function 1 is called! 1

the function 2 is called! 1

the function 6 is called! 1

the function 8 is called! 1

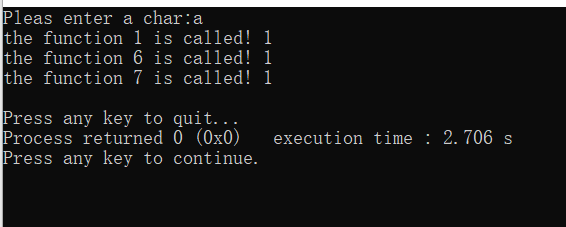


图7-5 实验七必做题7.2.3样例1测试结果

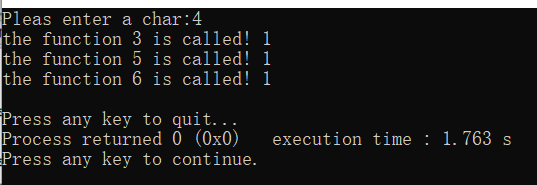


图7-6 实验七必做题7.2.3样例2测试结果

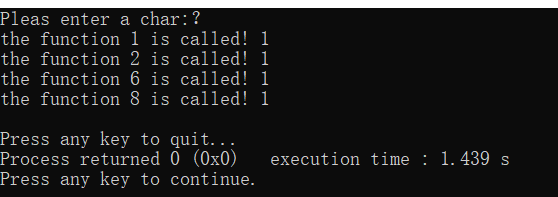


图7-7 实验七必做题7.2.3样例3测试结果

### 7.2.4 成绩统计

**【题目】**

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。

**【算法流程图】**

首先创建一个链表，并将数据输入链表，然后遍历链表输出。之后输入做的更改，遍历链表寻找对应数据，并修改它的值。之后遍历链表算出和，与之前所记录下的总数据数相比得到平均数。最后通过选择排序进行排序后输出。如图所示。

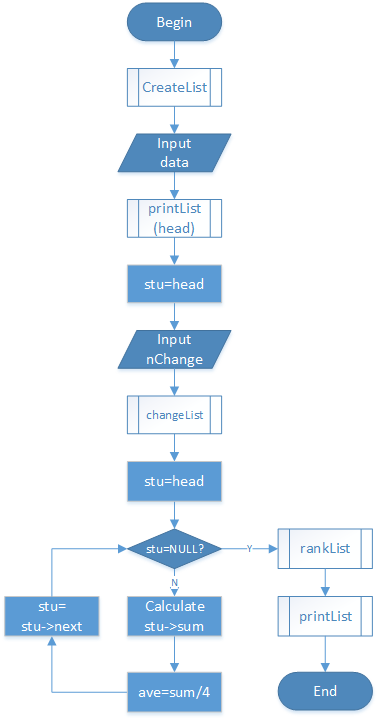


图7-8 实验七 必做题7.2.4算法流程图

**【程序清单】**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int flag = 0;

typedef struct \_Stu {

char stuNum[20];

char name[30];

float engGrade;

float mathGrade;

float phyGrade;

float CGrade;

float ave;

float sum;

struct \_Stu \*next;

}Stu;

Stu \*createList(int);

void printList(Stu \*stu);

int main()

{

int i, n, nChange;

scanf("%d", &n);

getchar();

Stu \*stu = createList(n);

Stu \*head = stu;

char waste[4];

//input the list

for (i = 0; i < n; i++)

{

scanf("%s%s", stu->stuNum, stu->name);

scanf("%f", &stu->engGrade);

scanf("%f", &stu->mathGrade);

scanf("%f", &stu->phyGrade);

scanf("%f", &stu->CGrade);

fgets(waste, 4, stdin);

stu = stu->next;

}

//print out the list

printList(head);

//change the partiicular contents. Take change the C grade of Harry for example

stu = head;//initiate the pointer stu

scanf("%d", &nChange);

char \*lesson[4] = { "英语","数学","物理","C" };

char changeLesson[15], changeNum[20];

float changeScore;

while (nChange-- > 0)

{

stu = head;

scanf("%s %s %f", changeNum, changeLesson, &changeScore);

while (stu)

{

if (strcmp(stu->stuNum, changeNum) == 0)

{

if (strcmp(changeLesson, "英语") == 0)

stu->engGrade = changeScore;

if (strcmp(changeLesson, "数学") == 0)

stu->mathGrade = changeScore;

if (strcmp(changeLesson, "物理") == 0)

stu->phyGrade = changeScore;

if (strcmp(changeLesson, "C") == 0)

stu->CGrade = changeScore;

break;

}

stu = stu->next;

}

}

printf("Alter:\n");

printList(head);

//calciulate the average scores of each student

float sum[4] = { 0 };

for (stu = head; stu != NULL; stu = stu->next)

{

stu->sum = stu->engGrade + stu->CGrade + stu->mathGrade + stu->phyGrade;

stu->ave = stu->sum / 4;

}

//print out the sum and average

printf("总分和平均分:\n");

printf("%-15s%-20s%-10s%-10s\n", "ID", "名字", "总分", "平均分");

stu = head;

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->sum, stu->ave);

stu = stu->next;

}

//sort the number

Stu \*begin, \*end;

for (begin = head; begin != NULL; begin = begin->next)

for (end = begin; end != NULL; end = end->next)

if (end->ave < begin->ave)

{

char tmpNum[20], tmpName[20];

strcpy(tmpNum, end->stuNum);

strcpy(end->stuNum,begin->stuNum);

strcpy(begin->stuNum, tmpNum);

strcpy(tmpName,end->name);

strcpy(end->name,begin->name);

strcpy(begin->name,tmpName);

float tmpScore = end->ave;

end->ave = begin->ave;

begin->ave=tmpScore;

}

printf("\nSort:\n");

printf("%-15s%-20s%-10s\n", "ID", "名字", "平均分");

stu = head;

while (stu)

{

printf("%-15s%-20s%-10.2f\n",

stu->stuNum, stu->name, stu->ave);

stu = stu->next;

}

printf("\n");

return 0;

}

void printList(Stu \*stu)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "名字", "英语", "数学", "物理", "C");

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->engGrade, stu->mathGrade, stu->phyGrade, stu->CGrade);

stu = stu->next;

}

printf("\n");

}

Stu \*createList(int stuNum)

{

static int num;

static Stu \*head;

static Stu \*tail;

if (flag == 0)

{

num = stuNum;

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

tail = head;

flag = 1;

}

if (num-- < 2)

{

tail->next = NULL;

return head;

}

else {

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

tail = tail->next;

return createList(num);

}

}

**【测试数据、测试理论结果及测试结果】**

5

U201614547 Qw 85 86 87 88

U201614848 Er 89 90 91 92

U201615049 Ty 56 85 89 59

U201615252 Ui 84 89 64 100

U201616260 Op 89.5 90 85 74

使用3次测试修改，分别为：

3

U201614547 数学 95.6

U201615049 C 100

U201616260 英语 100

第一次输入后，预计输出为 ：

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 86.00 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 59.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 89.50 90.00 85.00 74.00第二次输入后，预计输出为：

Alter:

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 95.60 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 100.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 100.00 90.00 85.00 74.00

总分和平均分

ID 名字 总分 平均分

U201614547 Qw 355.60 88.90

U201614848 Er 362.00 90.50

U201615049 Ty 330.00 82.50

U201615252 Ui 337.00 84.25

U201616260 Op 349.00 87.25

Sort:

ID 名字 平均分

U201615049 Ty 82.50

U201615252 Ui 84.25

U201616260 Op 87.25

U201614547 Qw 88.90

U201614848 Er 90.50

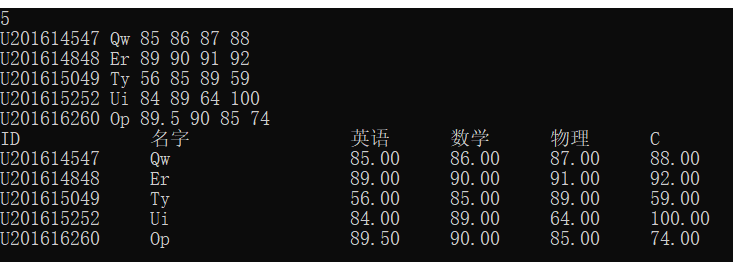


图7-9 实验七必做题7.2.4测试结果（1）

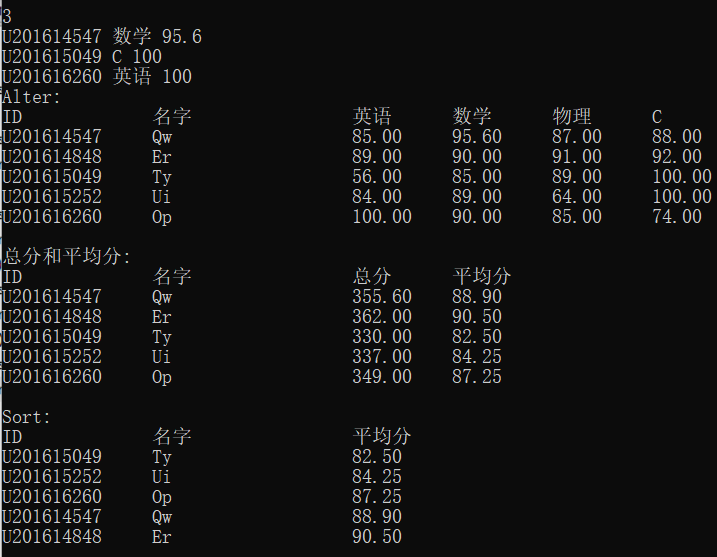


图7-10 实验七必做题7.2.4测试结果（2）

## 7.3 选做题

### 7.3.1 双向链表

**【题目】**

采用双向链表重做编程设计题中的第（2）题。

**【算法流程图】**

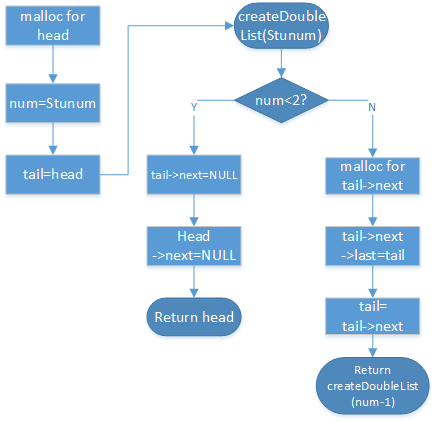


图7-11 实验七选做题双向链表算法流程图

**【程序清单】**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct \_Stu {

char stuNum[20];

char name[30];

float engGrade;

float mathGrade;

float phyGrade;

float CGrade;

float ave;

float sum;

struct \_Stu \*next;

struct \_Stu \*last;

}Stu;

Stu \*createDoubleList(int);

void printList(Stu \*stu);

int main()

{

int i, n, nChange;

scanf("%d", &n);

getchar();

Stu \*stu = createDoubleList(n);

Stu \*head = stu, \*rst;

char waste[4];

//input the list

for (i = 0; i < n; i++)

{

scanf("%s%s", stu->stuNum, stu->name);

scanf("%f", &stu->engGrade);

scanf("%f", &stu->mathGrade);

scanf("%f", &stu->phyGrade);

scanf("%f", &stu->CGrade);

fgets(waste, 4, stdin);

stu = stu->next;

}

printList(head);

//change the partiicular contents. Take change the C grade of Harry for example

stu = head;//initiate the pointer stu

scanf("%d", &nChange);

char \*lesson[4] = { "English","Math","Physics","C" };

char changeLesson[15], changeNum[20];

float changeScore;

while (nChange-- > 0)

{

stu = head;

scanf("%s %s %f", changeNum, changeLesson, &changeScore);

while (stu)

{

if (strcmp(stu->stuNum, changeNum) == 0)

{

if (strcmp(changeLesson, "English") == 0)

stu->engGrade = changeScore;

if (strcmp(changeLesson, "Math") == 0)

stu->mathGrade = changeScore;

if (strcmp(changeLesson, "Physics") == 0)

stu->phyGrade = changeScore;

if (strcmp(changeLesson, "C") == 0)

stu->CGrade = changeScore;

break;

}

stu = stu->next;

}

}

printf("Alter:\n");

printList(head);

//calculate the average scores of each student

float sum[4] = { 0 };

for (stu = head; stu != NULL; stu = stu->next)

{

stu->sum = stu->engGrade + stu->CGrade + stu->mathGrade + stu->phyGrade;

stu->ave = stu->sum / 4;

}

//print out the sum and average

printf("SumAndAvg:\n");

printf("%-15s%-20s%-10s%-10s\n", "ID", "Name", "SUM", "AVG");

stu = head;

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->sum, stu->ave);

stu = stu->next;

}

return 0;

}

//to print out the list

void printList(Stu \*stu)

{

printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "Name", "English", "Math", "Physics", "C");

while (stu)

{

printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n",

stu->stuNum, stu->name, stu->engGrade, stu->mathGrade, stu->phyGrade, stu->CGrade);

stu = stu->next;

}

printf("\n");

}

Stu \*createDoubleList(int stuNum)

{

static Stu \*head= (Stu\*)malloc(sizeof(Stu));

static int num = stuNum;

static Stu \*tail = head;

if (num-- < 2)

{

tail->next = NULL;

head->last = NULL;

return head;

}

else

{

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

tail->next->last = tail;

tail = tail->next;

return createDoubleList(num);

}

}

**【测试数据、测试理论结果及测试结果】**

5

U201614547 Qw 85 86 87 88

U201614848 Er 89 90 91 92

U201615049 Ty 56 85 89 59

U201615252 Ui 84 89 64 100

U201616260 Op 89.5 90 85 74

使用3次测试修改，分别为：

3

U201614547 数学 95.6

U201615049 C 100

U201616260 英语 100

第一次输入后，预计输出为 ：

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 86.00 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 59.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 89.50 90.00 85.00 74.00第二次输入后，预计输出为：

Alter:

ID 名字 英语 数学 物理 C

U201614547 Qw 85.00 95.60 87.00 88.00

U201614848 Er 89.00 90.00 91.00 92.00

U201615049 Ty 56.00 85.00 89.00 100.00

U201615252 Ui 84.00 89.00 64.00 100.00

U201616260 Op 100.00 90.00 85.00 74.00

总分和平均分

ID 名字 总分 平均分

U201614547 Qw 355.60 88.90

U201614848 Er 362.00 90.50

U201615049 Ty 330.00 82.50

U201615252 Ui 337.00 84.25

U201616260 Op 349.00 87.25

Sort:

ID 名字 平均分

U201615049 Ty 82.50

U201615252 Ui 84.25

U201616260 Op 87.25

U201614547 Qw 88.90

U201614848 Er 90.50

实际输出完全符合预测。

## 7.4 小结

通过表达式求值进一步加深了对指针与结构体的理解，可以更熟练地计算指针与结构体的表达式；通过源程序修改替换进一步理解了性参与实参的区别；程序设计第二题，则是学生信息录入表，并用链表来记录。而链表就是结构的一种比较普遍的应用，通过这个题目我对创建链表更为熟悉，也对链表的输入输出有了更深的印象，所以通过成绩统计以及之后的构建单项链表、构建双向链表进一步熟悉了关于链表数据结构的有关操作；通过查阅资料学习了宏定义中#与##运算符的作用，并尝试用其简化代码，取得了一定效果。