

## 实验 7 结构与联合实验

### 一、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。

### 二、实验题目及要求

#### 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, 'A', u}, {100, 'B', 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	$(++p) \rightarrow x$	100	100
2	$p++, p \rightarrow c$	B	B
3	$*p++ \rightarrow t, *p \rightarrow t$	U, x	U, x
4	$* ( ++p ) \rightarrow t$	x	x
5	$* ++p \rightarrow t$	x	V
6	$++ *p \rightarrow t$	V	V

#### 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 */
}

```

**解答:**

1. 运行结果为空白
2. 修改后:

```

#include "stdio.h"
#include "stdlib.h"
struct s_list {
    int data; /* 数据域 */
    struct s_list *next; /* 指针域 */

```

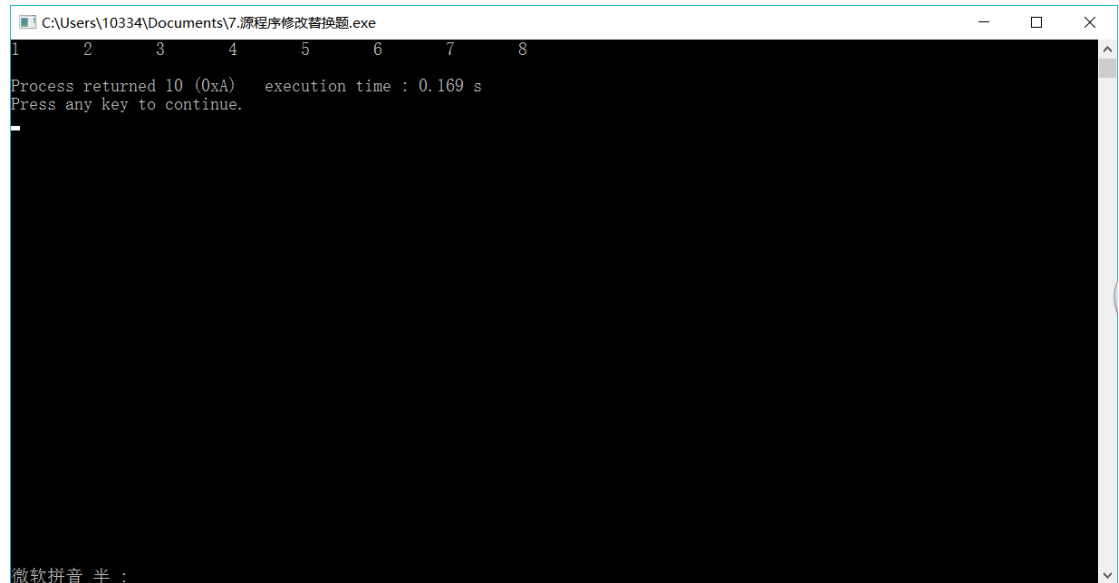
```

};
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 */
}

```

### 3. 运行结果:



4. 微软拼音 半 :

- (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.

```
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 */
        loc_head->last = NULL; /* loc_head 指向空 */
        tail = loc_head; /* tail 指向第一个结点 */
        while (*p) {
            loc_head = (struct s_list *)malloc(sizeof(struct s_list)); /*
新创建一个节点
            loc_head->last=tail; /* 新节点的 last 指向上一个节点 */
            tail = loc_head; /* tail 指向新节点, 为下一次赋值 last 做准备 */
            tail->data = *p++; /* 向新创建的结点的数据域赋值 */
        }
        //tail->last = 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. 运行结果:

```
C:\Users\10334\Documents\7.源程序修改替换题.exe
8 7 6 5 4 3 2 1
Process returned 10 (0xA)   execution time : 0.154 s
Press any key to continue.

微软拼音 半 :
```

### 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:

- 声明 8 个中断函数, 记为 isri。
- 声明一个联合为字段赋值, 联合占 2 个字节, 中断字段也占两个字节, 而且存储是从低位往高位存储的, 低 8 位分别对应 8 个 bit 结构, 高 8 位为 rsv, 因此为 all 赋值后, all 的低 8 位的值就是 bit 的值, all 的高 8 位的值就是 rsv 的值
- 令结构指针指向相应的中断函数。
- 依次访问每个字段, 使用 if 语句判断是否满足要求, 若满足则调用相应函数。

2) 程序清单

```
/*字段和联合的配合使用*/
#include<stdio.h>
/*定义 8 个中断提示函数*/
void isr0(void)
{
    printf("The Interrupt Service Routine isr0 is called!");
}
```

```

void isr1(void)
{
    printf("The Interrupt Service Routine isr1 is called!");
}
void isr2(void)
{
    printf("The Interrupt Service Routine isr2 is called!");
}
void isr3(void)
{
    printf("The Interrupt Service Routine isr3 is called!");
}
void isr4(void)
{
    printf("The Interrupt Service Routine isr4 is called!");
}
void isr5(void)
{
    printf("The Interrupt Service Routine isr5 is called!");
}
void isr6(void)
{
    printf("The Interrupt Service Routine isr6 is called!");
}
void isr7(void)
{
    printf("The Interrupt Service Routine isr7 is called!");
}
/*中断字段, bit_i 对应 isr_i,如果 bit_i 为 1, 则引用 isr_i,rsv 凑够 1 个字节*/
struct ISR_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;
};
/*使用联合可以为字段结构赋值, 联合占 2 个字节, 中断字段也占两个字节,
而且存储是从低位往高位存储的, 低 8 位分别对应 8 个 bit 结构, 高 8 位为 rsv,
因此为 all 赋值后, all 的低 8 位的值就是 bit 的值,
all 的高 8 位的值就是 rsv 的值*/
union ISR_REG {
    unsigned short all;
    struct ISR_BITS bit;
};

```

```

void main(void)
{
    /*声明函数指针， 分别 p_isr[i]指向 isri*/
    void(*p_isr[8])(void);
    p_isr[0] = isr0;
    p_isr[1] = isr1;
    p_isr[2] = isr2;
    p_isr[3] = isr3;
    p_isr[4] = isr4;
    p_isr[5] = isr5;
    p_isr[6] = isr6;
    p_isr[7] = isr7;
    union ISR_REG isr_reg;
    int N,i;//N 组输入
    scanf("%d", &N);
    for (i = 0; i < N; i++)
    {
        scanf("%hd", &isr_reg.all);
        printf("%d:\n", isr_reg.all);
        /*分别判断各 bit 位的数， 如果为 1 则调用对应的中断函数*/
        if (isr_reg.bit.bit0) { p_isr[0](); putchar('\n'); }
        if (isr_reg.bit.bit1) { p_isr[1](); putchar('\n'); }
        if (isr_reg.bit.bit2) { p_isr[2](); putchar('\n'); }
        if (isr_reg.bit.bit3) { p_isr[3](); putchar('\n'); }
        if (isr_reg.bit.bit4) { p_isr[4](); putchar('\n'); }
        if (isr_reg.bit.bit5) { p_isr[5](); putchar('\n'); }
        if (isr_reg.bit.bit6) { p_isr[6](); putchar('\n'); }
        if (isr_reg.bit.bit7) { p_isr[7](); putchar('\n'); }
        /*错误： 引用函数指针是格式为 p ( )， 括号里面填参数， 但不能没有括
号!!! */
        putchar('\n');
    }
    //system("pause");
}
3) 测试
(a) 测试数据：
    5
    56 0 255 8 89
(b) 运行结果：

```

```
CAUsers\10354\Documents\untitled6.exe
0 0.255 8 89
86:
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr4 is called!
The Interrupt Service Routine isr5 is called!
87:
88:
The Interrupt Service Routine isr0 is called!
The Interrupt Service Routine isr1 is called!
The Interrupt Service Routine isr2 is called!
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr4 is called!
The Interrupt Service Routine isr5 is called!
The Interrupt Service Routine isr6 is called!
The Interrupt Service Routine isr7 is called!
89:
The Interrupt Service Routine isr3 is called!
90:
The Interrupt Service Routine isr0 is called!
The Interrupt Service Routine isr3 is called!
The Interrupt Service Routine isr4 is called!
The Interrupt Service Routine isr6 is called!
Process returned 5 (0x5)   execution time : 2.070 s
Press any key to continue.
```

(c)

- (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：

- a) 定义学生信息结构 STUDENT\_INFO
- b) 主函数负责调用信息输入、信息处理、信息输出函数
- c) 子函数处理信息：
  - i. struct STUDENT\_INFO \* input(int N); 信息输入函数，返回链表的头部地址,N 代表 N 组输入，使用先进先出链表存储数据。并且采用了 if 语句判断 N，只有在至少有一组输入时才创建链表。
  - ii. void output(const struct STUDENT\_INFO \*p); 信息输出函数，接受链表的头部地址，遍历到 next 指向空字符为止，依次输出学生信息
  - iii. void modify(struct STUDENT\_INFO \*p); 信息修改函数，修改指定学生（使用学号定位）的指定成绩，接收链表的头部地址，通过 C 库中的 strcmp()函数比较学号，定位要修改信息的学生，使用结构指针指向存储该学生信息的结构，接下来同样用 strcmp()定位要修改的科目，找到后就可进行更改，修改数据数字可以直接赋值，修改字符串则使用 strcpy()。
  - iv. void SumAndAvg(const struct STUDENT\_INFO \*p,float \* ptof,int subject\_num);平均成绩和总成绩的计算，参数包括指向头链表的指针 p，存储平均成绩的地址 avg，学生人数 num\_of\_student，使用结构指针遍历学生，计算学生成绩之和。
  - v. output()将信息输出。
- d) 程序结束运行。



## 2) 程序清单

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
/*定义结构*/
struct STUDENT_INFO
{
    char ID_card[15];
    char name[20];
    float ENG, MATH, PHY, C;
    struct STUDENT_INFO *next;
};
/*信息输入函数，返回链表的头部地址,N 代表 N 组输入*/
struct STUDENT_INFO * input(int N);
/*信息输出函数，接受链表的头部地址，遍历到 next 指向空字符为止，依次输出
学生信息*/
void output(const struct STUDENT_INFO *p);
/*信息修改函数，修改指定学生（使用学号定位）的指定成绩，接收链表的头部
地址*/
void modify(struct STUDENT_INFO *p);
/*平均和总成绩计算*/
void SumAndAvg(const struct STUDENT_INFO *p,float * ptof,int subject_num);
/*以平均成绩排序*/
void Sort(struct STUDENT_INFO *p,float * avg,int num_of_student);
/*排序后输出*/
void sort_output(const struct STUDENT_INFO *p,float *avg);
int main(void)
{
    int N;
    float avg[20];//存储平均成绩
    scanf("%d", &N);
    getchar();
    struct STUDENT_INFO *pt;
    pt = input(N);
    output(pt);
    modify(pt);
    printf("Alter:\n");
    output(pt);
    printf("SumAndAvg:\n");
    SumAndAvg(pt, avg, 4);
    putchar('\n');
    printf("Sort:\n");
    Sort(pt, avg, N);
    sort_output(pt,avg);
    //system("pause");

    return 0
}
struct STUDENT_INFO * input(int N)
```

```

{
    int i;
    struct STUDENT_INFO *head, *tail, *p;
    if (N)
    {
        head = (struct STUDENT_INFO *)malloc(sizeof(struct STUDENT_INFO));
        tail = head;
        scanf(" %s %s %f %f %f %f", tail->ID_card, tail->name, &tail->ENG,
&tail->MATH, &tail->PHY, &tail->C);
        getchar();
        for (i = 2; i <= N; i++)
        {
            tail->next = (struct STUDENT_INFO *)malloc(sizeof(struct
STUDENT_INFO));
            tail = tail->next;
            scanf(" %s %s %f %f %f %f", tail->ID_card, tail->name, &tail->ENG,
&tail->MATH, &tail->PHY, &tail->C);
        }
            tail->next = NULL;

        }
        else return 0;
        return head;
    }
}

void output(const struct STUDENT_INFO *p)
{
    printf("%-15s%-20s%-10s%-10s%-10s%-10s\n", "ID", "Name", "English",
"Math", "Physics", "C");
    while (p)
    {
        printf("%-15s%-20s%-10.2f%-10.2f%-10.2f%-10.2f\n", p->ID_card,
p->name, p->ENG, p->MATH, p->PHY, p->C);
        p = p->next;
    }
    putchar('\n');
}

void modify(struct STUDENT_INFO *p)
{
    int N, i;
    char ID[15];
    char subject[20];
    float new_score;
    struct STUDENT_INFO *origin;
    origin = p;
    scanf("%d", &N);
    for (i = 1; i <= N; i++, p=origin)
    {
        scanf("%s %s %f", ID, subject, &new_score);
        //printf("%s %s %f", ID, subject, new_score);
    }
}

```

```

/*寻找学生*/
while (strcmp(ID,p->ID_card))
{
    p = p->next;
}
/*寻找科目*/
if (!strcmp("English", subject)) p->ENG=new_score;
else if (!strcmp("Math", subject)) p->MATH = new_score;
else if (!strcmp("Physics", subject)) p->PHY = new_score;
else if (!strcmp("C", subject)) p->C = new_score;
}
}
void SumAndAvg(const struct STUDENT_INFO *p,float *ptof,int subject_num)
{
    float sum;
    int i=0;

    printf("%-15s%-20s%-10s%-10s\n", "ID", "Name", "SUM", "AVG");
    while (p)
    {
        sum = ptof[i] = 0;
        sum = p->C + p->MATH + p->PHY + p->ENG;
        *ptof = sum / subject_num;
        printf("%-15s%-20s%-10.2f%-10.2f\n", p->ID_card, p->name, sum, *ptof);
        ptof++;
        p = p->next;
    }
}
void Sort(struct STUDENT_INFO *p, float * avg, int num_of_student)
{
    int i, j, k;
    int count;
    struct STUDENT_INFO *pt = p;
    for (i = 0; i < num_of_student; i++, p=p->next, pt = p)
    {
        k = i;
        for (j = i; j < num_of_student; j++)
        {
            if (avg[j] < avg[k])
                k = j;
        }
        if (k != i)
        {
            float temp;
            temp = avg[i];
            avg[i] = avg[k];
            avg[k] = temp;
            for (count = 0; count < k - i; count++)
                pt = pt->next;//此时 pt 指向 avg 最小的结构
            struct STUDENT_INFO *p1, *p2, ptemp;

```

```

        p1 = p->next;
        p2 = pt->next;
        ptemp = *p;
        *p = *pt;
        *pt = ptemp;
        pt->next = p2;
        p->next = p1;
    }

}

}

void sort_output(const struct STUDENT_INFO *p,float * avg)
{
    printf("%-15s%-20s%-10s\n", "ID", "Name","AVG");
    while (p)
    {
        printf("%-15s%-20s%-10.2f\n", p->ID_card, p->name,*avg);
        p = p->next;
        avg++;
    }
    //putchar('\n');
}

```

## 2) 测试

### (a) 测试数据:

5//输入输出

U20140101 ZhangChuanChao 85 86 87 88

U20140126 MaiDouDou 99 99 99 99

U20140158 XiaoDouDou 56 85 89 59

U20140312 DaoDaoDog 84 89 65 100

U20140359 XiDaDa 88.8 88.8 88.8 88.8

3//修改数据

U20140101 Math 95.6

U20140359 C 100

U20140359 English 100

### (b) 测试结果:

```
C:\Users\10158\Documents\visual studio 2015\Projects\c语言程序设计\c语言程序设计.exe
9
20140101 Math 95.6
20140359 C 100
20140359 English 100
Enter:
ID      Name      English  Math  Physics  C
20140101 ZhengChuanChao 85.00  95.60  87.00  88.00
20140126 MaDouDou  99.00  99.00  99.00  99.00
20140158 XiaoDouDou  56.00  85.00  89.00  59.00
20140312 DaChaoDe  84.00  89.00  65.00  100.00
20140359 XiDaDa  100.00  88.80  88.80  100.00
SumAndAvg:
ID      Name      SUM      AVG
20140101 ZhengChuanChao 355.60  88.90
20140126 MaDouDou  396.00  99.00
20140158 XiaoDouDou  289.00  72.25
20140312 DaChaoDe  338.00  84.50
20140359 XiDaDa  377.00  94.40
Sort:
ID      Name      AVG
20140158 XiaoDouDou  72.25
20140312 DaChaoDe  84.50
20140101 ZhengChuanChao 88.90
20140359 XiDaDa  94.40
20140126 MaDouDou  99.00
请按任意键继续. . .
请按任意键继续. . .
```

#### 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。

(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。

(3) 采用双向链表重做编程设计题第(2)题。

#### 四、实验总结

本次实验相对较难, 编写的程序代码也较长。通过这次实验, 我熟悉和掌握了结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。其中, 成绩处理这一题充分体现了函数式编程的优点, 将一个大问题化为多个小问题, 分部调试, 哪里出了问题就修改相应的函数即可, 而不必审阅整个程序, 大大提高了效率, 节省了时间。另外, 程序修改和替换这道题让我掌握的先进先出链表和先进后出链表的声明, 尤其要注意的是在使用函数声明先进先出链表时函数的形参要声明为指向指针的指针, 该指针指向已声明的结构, 再引用该函数时要将头指针的地址作为实参传递, 而不能传递它的值。如程序修改替换这一题, 传递了 head 存储的值 (NULL), 因此不能正确声明链表, 同样也无法对 head 进行操作。