

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 网安2003班**

**学 号： U202012043**

**姓 名： 范启航**

**指导教师： 张云鹤**

**报告日期： 2020年12月27日**

**网络空间安全学院**

**目□□录**

[**1□□□表达式和标准输入输出实验 1**](#_Toc404837920)

[1.1□□实验目的 1](#_Toc404837921)

[1.2□□实验内容 1](#_Toc404837922)

[1.3□□实验小结 1](#_Toc404837923)

[**2□□□流程控制实验 2**](#_Toc404837924)

[2.1□□实验目的 1](#_Toc404837921)

[2.2□□实验内容 1](#_Toc404837922)

[2.3□□实验小结 1](#_Toc404837923)

[**3□□□函数与程序结构实验 3**](#_Toc404837929)

[3.1□□实验目的 1](#_Toc404837921)

[3.2□□实验内容 1](#_Toc404837922)

[3.3□□实验小结 1](#_Toc404837923)

[**4□□□编译预处理实验 4**](#_Toc404837934)

[4.1□□实验目的 1](#_Toc404837921)

[4.2□□实验内容 1](#_Toc404837922)

[4.3□□实验小结 1](#_Toc404837923)

[**5□□□数组实验 5**](#_Toc404837938)

[5.1□□实验目的 1](#_Toc404837921)

[5.2□□实验内容 1](#_Toc404837922)

[5.3□□实验小结 1](#_Toc404837923)

[**6□□□指针实验 6**](#_Toc404837943)

[6.1□□实验目的 1](#_Toc404837921)

[6.2□□实验内容 1](#_Toc404837922)

[6.3□□实验小结 1](#_Toc404837923)

[**7□□□结构与联合实验 7**](#_Toc404837948)

[7.1□□实验目的 1](#_Toc404837921)

[7.2□□实验内容 1](#_Toc404837922)

[7.3□□实验小结 1](#_Toc404837923)

[**8□□□文件实验 8**](#_Toc404837953)

[8.1□□实验目的 1](#_Toc404837921)

[8.2□□实验内容 1](#_Toc404837922)

[8.3□□实验小结 1](#_Toc404837923)

[**参考文献 9**](#_Toc404837957)

# 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)

表7.2.1各表达式的值

|  |  |  |  |
| --- | --- | --- | --- |
| **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’ |

**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 \*/

}

**解答：**

（1）错误修改：

1）第7行：更改函数声明为void create\_list(struct s\_list \*\*headp, int \*p);

2) 第12行：将head的地址传入函数，应改为create\_list(&head,s);

3) 第20行：函数原型改为void create\_list(struct s\_list \*\*headp,int \*p)

接收head的地址从而通过间访来改变main中head的值

4) 第36行：通过间访来headp改变main中head,应改为：\*headp = loc\_head;

（2）错误修改后运行结果：

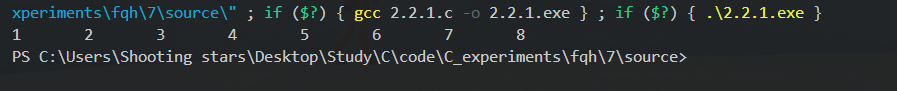


图7-2-1程序运行示意图

（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：

代码如下：

#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);//?更改函数声明

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 \*/  //?将head的地址传入函数

    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)//?更改函数原型，接收head的地址

{

    int i = 0;

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

    while(\*p != 0)

    {

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

        loc\_head->data = \*p++;

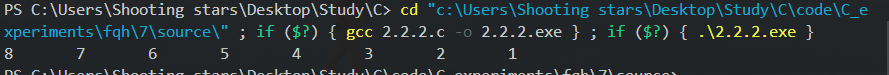
        loc\_head->next = tail;

        tail = loc\_head; }

    \*headp = loc\_head;

}

运行结果示意图：



7-2-2运行结果示意图

**7.3 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，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( int n)  
{  
printf(“the function %d is called!\n”,n);  
}

**解答：**

流程图：



图7-3-1程序设计流程图

代码如下：

#include <stdio.h>

#include <stdlib.h>

struct bits

{

    unsigned bit0:1;

    unsigned bit1:1;

    unsigned bit2:1;

    unsigned bit3:1;

    unsigned bit4:1;

    unsigned bit5:1;

    unsigned bit6:1;

    unsigned bit7:1;

};

void f0(int n);

void f1(int n);

void f2(int n);

void f3(int n);

void f4(int n);

void f5(int n);

void f6(int n);

void f7(int n);

int main()

{

    void (\*p\_fun[8])(int n);

    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;

    struct bits b;

    unsigned int n;

    scanf("%u", &n);

    b.bit0 = n &0x1;

    n >>= 1;

    b.bit1 = n&0x1;

    n >>= 1;

    b.bit2 = n&0x1;

    n >>= 1;

    b.bit3 = n&0x1;

    n >>= 1;

    b.bit4 = n&0x1;

    n >>= 1;

    b.bit5 = n&0x1;

    n >>= 1;

    b.bit6 = n&0x1;

    n >>=1;

    b.bit7 = n&0x1;

    if(b.bit0)

    p\_fun[0](0);

    if(b.bit1)

    p\_fun[1](1);

    if(b.bit2)

    p\_fun[2](2);

    if(b.bit3)

    p\_fun[3](3);

    if(b.bit4)

    p\_fun[4](4);

    if(b.bit5)

    p\_fun[5](5);

    if(b.bit6)

    p\_fun[6](6);

    if(b.bit7)

    p\_fun[7](7);

    return 0;

}

void f0(int n)

{

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

}

void f1(int n)

{

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

}

void f2(int n)

{

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

}

void f3(int n)

{

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

}

void f4(int n)

{

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

}

void f5(int n)

{

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

}

void f6(int n)

{

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

}

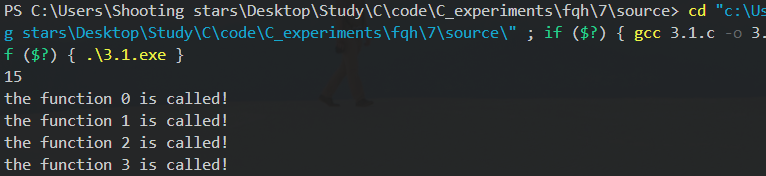
void f7(int n)

{

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

}

运行结果：



7-3-2运行结果示意图

（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。

解答：

流程图：



7-3-3程序设计流程图（1）



7-3-4程序设计流程图（2）



7-3-5程序设计流程图（3）

代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

// struct构建

typedef struct \_Node

{

    char ID[30];

    char name[30];

    int Score[4];

    int total;

    double average;

    struct \_Node\* next;

} Node;

typedef struct \_List

{

    Node \*head;

    Node \*tail;

} List;

//函数声明

void input\_info(List \*list);

void output\_info(List \*list);

void change(List \*list);

void average(List \*list);

void output\_more(List \*list);

int main()

{

    int  n, flag = 1;

    void (\*p)(List \*);

    List list;

    list.head = NULL;

    list.tail = NULL;

    while(1)

    {

        //系统界面

        printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

        printf("\*\*\*\*\*      班级成绩系统      \*\*\*\*\*\n");

        printf("1:输入每个学生的各项信息\n");

        printf("2:输出每个学生的各项信息\n");

        printf("3:修改指定学生的指定数据项的内容\n");

        printf("4:统计每个同学的平均成绩\n");

        printf("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\n");

        printf("输入其他字符退出本系统\n");

        printf("请输入功能代码:\n");

        scanf("%d", &n);

        switch (n)

        {

            case 1 :p =input\_info;break;

            case 2 :p =output\_info; break;

            case 3 :p = change; break;

            case 4 :p = average; break;

            case 5 :p = output\_more; break;

            default :printf("正常退出\n"); flag = 0;

            break;

        }

        if(!flag)

        break;

        p(&list);

    }

    return 0;

}

void input\_info(List \*list)

{

    int i=0,flag;

    Node \*p;

    while(1)

    {

        printf("输入功能代码：\n");

        printf("1：继续录入学生信息\n");

        printf("0：结束信息录入\n");

        scanf("%d", &flag);

        if(!flag)

        break;

        i++;

        p = (Node \*)malloc(sizeof(Node));

        printf("请输入第%d个学生学号:\n", i);

        scanf("%s", p->ID);

        printf("请输入学生姓名：\n");

        scanf("%s", p->name);

        printf("请输入学生英语成绩：\n");

        scanf("%d", &p->Score[0]);

        printf("请输入学生高等数学成绩：\n");

        scanf("%d", &p->Score[1]);

        printf("请输入学生普通物理成绩：\n");

        scanf("%d", &p->Score[2]);

        printf("请输入学生C语言程序设计成绩：\n");

        scanf("%d", &p->Score[3]);

        if(!(list->head))

        {

            list->head = p;

            list->tail = p;

        }

        else

        {

            list->tail->next = p;

            list->tail = p;

        }

    }

    if(i>0)

        list->tail->next = NULL;

}

void output\_info(List \*list)

{

    int i;

    Node \*p;

    for(p = list->head, i = 1; p; p = p->next,i++)

    {

        printf("第%d个学生学号:%s\n", i,p->ID);

        printf("姓名:%s\n", p->name);

        printf("英语成绩:         %d\n", p->Score[0]);

        printf("高等数学成绩:     %d\n", p->Score[1]);

        printf("普通物理成绩:     %d\n", p->Score[2]);

        printf("C语言程序设计成绩:%d\n", p->Score[3]);

    }

}

void change(List \*list)

{

    char s[30], t[30];

    int n,temp, flag;

    printf("请输入要修改的学生的学号\n");

    scanf("%s", s);

    Node \*p = list->head;

    while(p && strcmp(s,p->ID))

    p = p->next;

    if(p)

    {

        while(1)

        {

            printf("请输入要修改的项目:\n");

            printf("1:学号\n");

            printf("2:姓名\n");

            printf("3:英语成绩\n");

            printf("4:高等数学成绩\n");

            printf("5:普通物理成绩\n");

            printf("6:C语言程序设计成绩\n");

            printf("输入其他字符退出\n");

            scanf("%d", &n);

            printf("请输入要修改的内容:\n");

            switch (n)

            {

            case 1: scanf("%s", t);

                    strcpy(p->ID, t);break;

            case 2: scanf("%s", t);

                    strcpy(p->name, t);break;

            case 3: scanf("%d", &temp);

                    p->Score[0] = temp;break;

            case 4: scanf("%d", &temp);

                    p->Score[1] = temp;break;

            case 5: scanf("%d", &temp);

                    p->Score[2] = temp; break;

            case 6: scanf("%d", &temp);

                    p->Score[3] = temp;break;

            default: flag = 0;

                    break;

            }

            if(!flag)

                break;

        }

    }

    else printf("输入错误，无此学生\n");

}

void average(List \*list)

{

    Node \*p;

    int i,n;

    for(p = list->head; p; p = p->next)

    {

        p->total = 0;

        for(i = 0; i<4; i++)

        p->total += p->Score[i];

        p->average = p->total/4.0;

    }

void output\_more(List \*list)

{

    average(list);

    int i;

    Node \*p;

    for(p = list->head, i = 1; p; p = p->next,i++)

    {

        printf("第%d个学生学号:%s\n", i,p->ID);

        printf("姓名:%s\n", p->name);

        printf("英语成绩:         %d\n", p->Score[0]);

        printf("高等数学成绩:     %d\n", p->Score[1]);

        printf("普通物理成绩:     %d\n", p->Score[2]);

        printf("C语言程序设计成绩:%d\n", p->Score[3]);

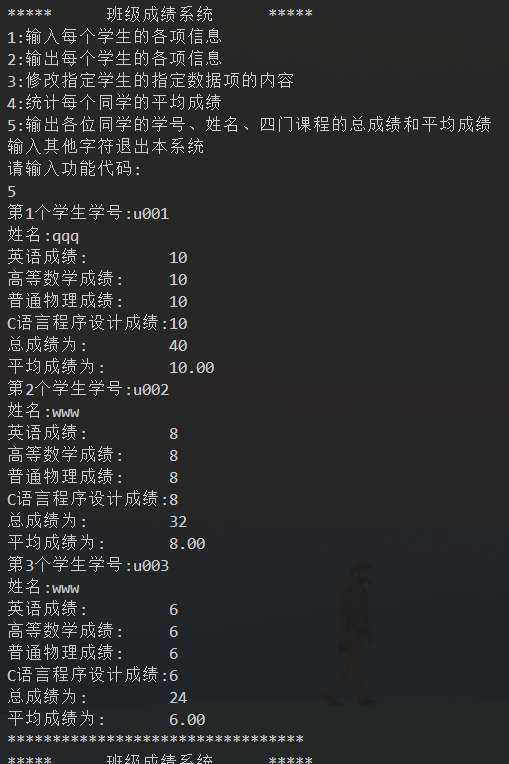
        printf("总成绩为:         %d\n", p->total);

        printf("平均成绩为:       %.2f\n", p->average);

    }

}

运行结果：



7-3-6运行结果示意图

**7.4．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。

解答：

（1）:

排序部分代码如下：

void sort(List \*list)

{

    Node \*p, \*temp;

    temp = (Node \*)malloc(sizeof(Node));

    int len, i,j,k;

    //统计学生

    for(p = list->head,len = 0; p ; p = p->next,len++);

    //冒泡排序

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

    {

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

        {

            if(p->average > p->next->average)

            {

                //学号交换

                strcpy(temp->ID, p->ID);

                strcpy(p->ID, p->next->ID);

                strcpy(p->next->ID, temp->ID);

                //姓名交换

                strcpy(temp->name, p->name);

                strcpy(p->name, p->next->name);

                strcpy(p->next->name, temp->name);

                //各项成绩交换

                for(k = 0; k<4; k++)

                {

                    temp->Score[k] = p->Score[k];

                    p->Score[k] = p->next->Score[k];

                    p->next->Score[k] = temp->Score[k];

                }

                //总成绩交换

                temp->total = p->total;

                p->total = p->next->total;

                p->next->total = temp->total;

                //平均成绩交换

                temp ->average = p->average;

                p->average = p->next->average;

                p->next->average = temp->average;

            }

        }

    }

    free(temp);

}

（2）

排序部分代码如下：

void sort(List \*list)

{

    average(list);

    Node \*p,\*prior,\*after,\*q;

    int len, i,j,k;

    //统计学生

    for(p = list->head,len = 0; p ; p = p->next,len++);

    //冒泡排序

    for(i = 0; i<len-1; i++)

    {

        p = list->head;

        if(p->average > p->next->average)

        {

            after = p->next->next;

            list->head = p->next;

            list->head->next = p;

            p->next = after;

        }

        for(j = 1,prior = list->head, p = prior ->next; j<len-i-1;j++)

        {

            if(p->average > p->next->average)

            {

                q = p->next;

                after = q->next;

                prior->next = q;

                q->next = p;

                p->next = after;

                prior = prior->next;

            }

            else

            {

                p = p->next;

                prior = prior->next;

            }

        }

    }

}

（3）

结构声明部分：

typedef struct \_Node

{

    char ID[30];

    char name[30];

    int Score[4];

    int total;

    double average;

    struct \_Node\* prior;

    struct \_Node\* next;

} Node;

链表创建：

        if(!(list->head))

        {

            list->head = p;

            list->tail = p;

            p ->prior = list->tail;

        }

        else

        {

            list->tail->next = p;

            p->prior = list->tail;

            list->tail = p;

        }

## 7.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在本次实验中主要应用到了结构，联合和字段的知识，在使用结构时需要主要各个运算符的优先级，并对其进行判断才能读懂他人的程序，使自己的程序更加简洁明了。在使用结构构建链表过程中，要灵活运用指针进行操作。在必要时可以多创建一些指针来进行灵活的操作。在对链表排序是用交换数据域与交换指针域两种方法，当数据域较少时，选择交换数据域更加方便，而当数据域较多时，往往选择交换指针域使代码更加简洁，运行效率更高。因此掌握好两种方法并灵活应用能够使程序更加高效。

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计,北京： 科学出版社,2013

[2] 李开,卢萍,曹计昌. C语言实验与课程设计, 北京：科学出版社,2011