

華中科技大學

課程實驗報告

課程名稱： C 語言程序設計實驗

專業班級：

學 號：

姓 名：

指導教師：

報告日期： 2022.12.28

網絡空間安全學院

目 录

5 数组实验	错误!未定义书签。
5.1 实验目的	错误!未定义书签。
5.2 实验内容	错误!未定义书签。
5.3 实验小结	19
7 结构与联合实验.....	错误!未定义书签。
7.1 实验目的	20
7.2 实验内容	20
7.3 实验小结	41
实验总结.....	41
参考文献.....	41

5 数组程序设计实验

5.1 实验目的

- (1) 掌握数组的说明、初始化和使用。
- (2) 掌握一维数组作为函数参数时实参和形参的用法。
- (3) 掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。
- (4) 掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

5.2 实验内容

5.2.1 源程序改错与跟踪调试

在下面所给的源程序中，函数 `strcate(t,s)` 的功能是将字符串 `s` 连接到字符串 `t` 的尾部；函数 `strdelc(s,c)` 的功能是从字符串 `s` 中删除所有与给定字符 `c` 相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgrmingLnguge

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

(1) 单步执行源程序。进跟踪进入 `strcate` 时，观察字符数组 `t` 和 `s` 中的内容，分析结果是否正确。当单步执行光条刚落在第二个 `while` 语句所在行时，`i` 为何值？`t[i]` 为何值？分析该结果是否存在问题。当单步执行光条落在 `strcate` 函数块结束标记即右花括号 “}” 所在行时，字符数组 `t` 和 `s` 分别为何值？分析是否实现了字符串连接。

(2) 跟踪进入函数 `strdelc` 时，观察字符数组 `s` 中的内容和字符 `c` 的值，分析结果是否正确。单步执行 `for` 语句过程中，观察字符数组 `s`, `j` 和 `k` 值的变化，分析该结果是否存在问题。当单步执行光条落在 `strdelc` 函数块结束标记 “}” 所在行时，字符串 `s` 为何值？分析是否实现了所要求的删除操作。

```

/*实验 5-1 程序改错与跟踪调试题程序*/
#include <stdio.h>
void strcat(char [],char []);
void strdelc(char [],char );
int main(void)
{
    char a[]="Language", b[]="Programming";
    printf("%s %s\n", b,a);
    strcat(b,a); printf("%s %s\n",b,a);
    strdelc(b, 'a');    printf("%s\n",b);
    return 0;
}
void strcat(char t[],char s[])
{
    int i = 0, j = 0;
    while(t[i++] );
    while((t[i++] = s[j++]) != '\0');
}
void strdelc(char s[], char c)
{
    int j,k;
    for(j=k=0; s[j] != '\0'; j++)
        if(s[j] != c) s[k++] = s[j];
}

```

解答：

(1) 在单步执行过程中，观察以下变量值：

- 1) 当单步执行光条刚落在第二个 while 语句所在行时，i 为 12，t[i]为 0，该结果存在问题。
- 2) 当单步执行光条落在 strcat 函数块结束标记即右花括号 “}” 所在行时，字符数组 t 和 s 分别为 Programming 和 uage，未实现字符串连接。
- 3) 当单步执行光条落在 strdelc 函数块结束标记 “}” 所在行时，字符串 s 为 ProgrmingLngugeage。

(2) 基于以上的单步执行观察结果，说明还存在如下逻辑错误：

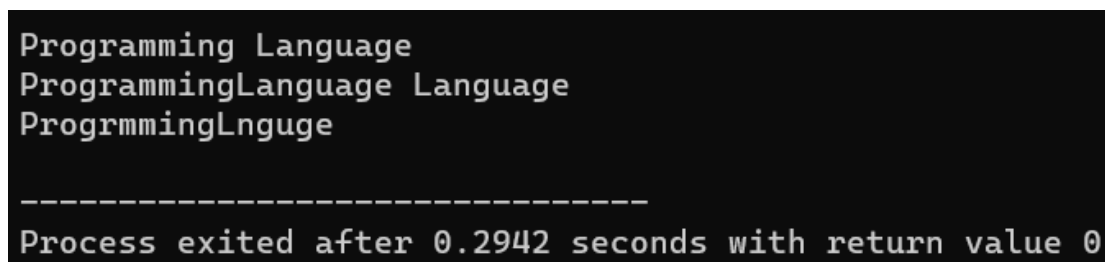
- 1) 数组 b 的内存不够存储 ab 中的字符串，应该修改成：b[n],n>16 or 数组 ab 间加数组 c[m],m>1。(因编译器版本不同结果不同)
- 2) while(t[i++])结束时 i 仍自加，应该修改成：其后加一行 i--;。

3) 循环字符串赋值结尾没有结束符, 应该修改成: 最后添加 `s[k]=0;` 。

修改后, 源程序清单如下:

```
#include<stdio.h>
void strcate(char [],char []);
void strdelc(char [],char );
int main(void)
{
    char a[]="Language", c[2],b[]="Programming";
    //或者扩大 b 的空间 即 b[n]n>16
    printf("%s %s\n", b,a);
    strcate(b,a);printf("%s %s\n",b,a);
    strdelc(b, 'a');    printf("%s\n",b);
    return 0;
}
void strcate(char t[],char s[])
{
    int i = 0, j = 0;
    while(t[i++] );
    i--; //运算多加了 1, 要减去
    while((t[i++] = s[j++]) != '\0');
}
void strdelc(char s[], char c)
{
    int j,k;
    for(j=k=0; s[j] != '\0'; j++)
        if(s[j] != c) s[k++] = s[j];
    s[k]=0; //结尾添加结束符
}
```

(3) 错误修改后运行结果如图 5-1 所示。



```
Programming Language
ProgrammingLanguage Language
ProgrmmingLnguge
-----
Process exited after 0.2942 seconds with return value 0
```

图 5-1 实验 1-1 修改后运行结果截图

5.2.2 源程序完善和修改替代

(1) 下面的源程序用于求解瑟夫问题: M 个人围成一圈, 从第一个人开始依次从 1 至 N 循环报数, 每当报数为 N 时报数人出圈, 直到圈中只剩下一个人为止。

①请在源程序中的下划线处填写合适的代码来完善该程序。

```
#include<stdio.h>
#define M 10
#define N 3
int main(void)
{
    int a[M], b[M]; /* 数组 a 存放圈中人的编号, 数组 b 存放出圈人的编号 */
    int i, j, k;
    for(i = 0; i < M; i++) /* 对圈中人按顺序编号 1—M */
        a[i] = i + 1;
    for(i = M, j = 0; i > 1; i--){
        /* i 表示圈中人个数, 初始为 M 个, 剩 1 个人时结束循环; j 表示当前报数人的位置 */
        for(k = 1; k <= N; k++) /* 1 至 N 报数 */
            if(++j > i - 1) j = 0; /* 最后一个人报数后第一个人接着报, 形成一个圈 */
        b[M-i] = j ? _____: _____; /* 将报数为 N 的人的编号存入数组 b */
        if(j)
            for(k = --j; k < i-1; k++) /* 压缩数组 a, 使报数为 N 的人出圈 */
                _____;
    }
    for(i = 0; i < M-1; i++) /* 按次序输出出圈人的编号 */
        printf(“%6d”, b[i]);
    printf(“%6d\n”, a[0]); /* 输出圈中最后一个人的编号 */
    return 0;
}
```

②上面的程序中使用数组元素的值表示圈中人的编号, 故每当有人出圈时都要压缩数组, 这种算法不够精炼。如果采用做标记的办法, 即每当有人出圈时对相应数组元素做标记, 从而可省掉压缩数组的时间, 这样处理效率会更高一些。请采用做标记的办法修改程序, 并使修改后的程序与原程序具有相同的功能。

解答:

(1) 填写合适代码完善程序:

- 1) `b[M-i] = j ? a[j-1]:a[i-1];`
- 2) `for(k = --j; k < i-1; k++) a[k]=a[k+1];`

(2) 用做标记的办法修改程序:

1) 算法流程如图 5-2 所示。

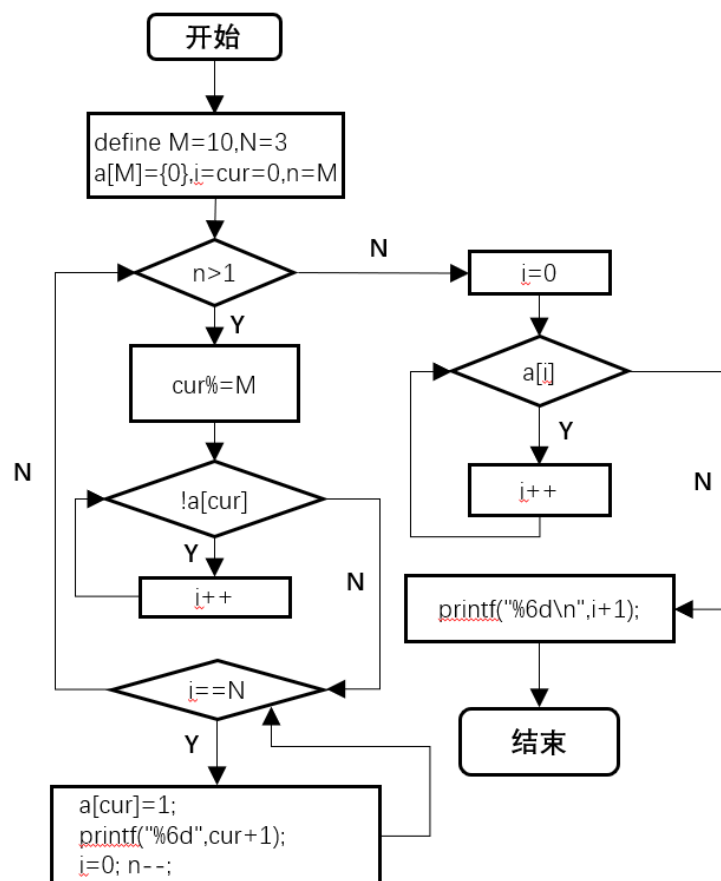


图 5-2 实验 2-1 改进后程序流程图

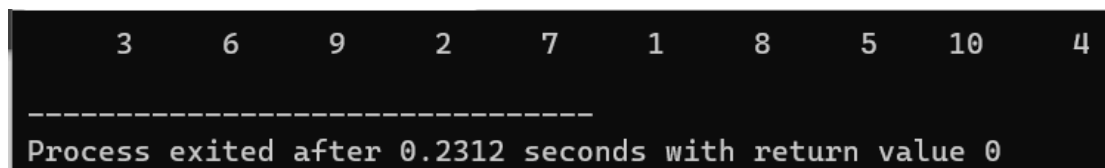
2) 源程序清单

```

#include<stdio.h>
#define M 10
#define N 3
int main(void)
int main(void)
{
    int a[M]={0}; //全标记为 0
    int i,cur,n; //i 计数， cur 人的位置， n 为剩余人数
    for(i=cur=0,n=M;n>1;cur++){
        cur%=M; //形成闭环
        if(!a[cur]) i++; //为 0 则计数
        if(i==N){
            a[cur]=1; //出圈标记为 1
            printf("%6d",cur+1);
            i=0;
            n--;
        }
    }
    for(i=0;a[i];i++); //寻找胜利的人
}
    
```

```
    printf("%6d\n",i+1);
return 0;
}
```

3) 对应测试数据的运行结果截图



```

    3      6      9      2      7      1      8      5     10      4
-----
Process exited after 0.2312 seconds with return value 0

```

图 5-3 实验 2-2 改善后程序运行结果截图

5.2.3 程序设计

(1) 输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

解答：

1) 解题思路：

1.输入 n

2.根据 n 的正负判断数据的符号位输出形式，再通过 for 循环将其依次存入数组 ch 中，运算是通过移位实现的。

3. 最后通过循环输出结果

4. 结束

2) 源程序清单

```
#include<stdio.h>
int main()
{
    int n;
    int i;
    char ch[32];
    scanf("%d",&n);
    for (i=0;i<32;i++)
    if(n>=0) ch[32-i-1]=(n>>i)%2+48;
    else ch[32-i-1]=-(n>>i)%2+48;
    for (i=0;i<32;i++)
    printf("%c",ch[i]);
    return 0;
}
```

3) 测试

(a) 测试数据：

验证正负,所以选择 1, -1, 0。

如表 2-1 所示。

表 5-1 编程题 1 的测试数据

测试用例	程序输入	理论结果
	n	
用例 1	1	00000000000000000000000000000001
用例 2	-1	11111111111111111111111111111111
用例 3	0	00000000000000000000000000000000

(b) 对应测试用例 1 的运行结果如图 5-4 所示。

[illegible]

图 5-4 程序设计题 1 的测试用例 1 的运行结果

对应测试用例 2 的运行结果如图 5-5 所示。

[illegible]

图 5-5 程序设计题 1 的测试用例 2 的运行结果

对应测试用例 3 的运行结果如图 5-6 所示。

[illegible]

图 5-6 程序设计题 1 的测试用例 3 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)编写一个 C 程序,要求采用模块化程序设计思想,将相关功能用函数实现,并提供菜单选项(0-4,其中 0 为退出)。对应菜单选项 1-4,该程序具有以下功能:

①“成绩输入”,输入 n 个学生的姓名和 C 语言课程的成绩。②“成绩排序”,将成绩按从高到低的次序排序,姓名同时进行相应调整。成绩相同的,按照输入先后次序排列。③“成绩输出”,输出排序后所有学生的姓名和 C 语言课程的成绩。④“成绩查找”,输入一个 C 语言课程成绩值,用二分查找进行搜索。如果查找到有该成绩,则输出该成绩学生的姓名和 C 语言课程的成绩;否则,输出提

示 “not found!”。

解答：

1) 解题思路：

1.输入菜单编号，用 abcd 的值作为执行操作的判断。

2.分别用两个数组完成“成绩输入”，设计一个冒泡法函数完成“成绩排序”，直接用循环在主函数中实现“成绩输出”，设计另一个二分法函数完成“成绩查找”，再在主函数中按顺序执行输出的语句。

3. 在主函数中通过 switch-case, 并单个使用 scanf 确保菜单选项的连续输入，并用 while 循环实现当值为 0 时跳出的操作。

4. 结束

2) 程序清单

```
#include<stdio.h>
#include<string.h>
void star (int a[],char b[][10],int n)
{
    int i,j,t;
    char m[10];
    for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++)
            if (a[j]<a[j+1]){
                t=a[j];a[j]=a[j+1];a[j+1]=t;
                strcpy(m,b[j]); //把 b 作为一维数组将字符串换位置
                strcpy(b[j],b[j+1]);
                strcpy(b[j+1],m);
            }
}

int Binary(int a[],int x,int n) //二分法
{
    int low=n-1,high=0,mid;
    while(low>=high){
        mid=(low+high)/2;
        if(x<a[mid]) high=mid+1;
        else if(x>a[mid])
            low=mid-1;
        else return mid;
    }

    return -1;
}

int main()
{
    int n,i,j;
```

```

int m,k;
int a,b,c,d;
int mid;
scanf("%d",&m);
scanf("%d",&n);
int score[n];
char name[n][10];
while(m){
switch(m){
case 1:
for(i=0;i<n;i++) {
scanf("%s %d",&name[i],&score[i]);
}
scanf("%d",&m);a=1;break;
case 2:
star(score,name,n);scanf("%d",&m);b=2;break;
case 3:
c=1;scanf("%d",&m);break;
case 4:
scanf("%d",&k);
mid=Binary(score,k,n);
if(mid!=-1)d=1;else d=2;
scanf("%d",&m);
break;
}
}
if(a==1) printf("%d records were input!\n",n);
if(b==2) printf("Reorder finished!\n");
if(c==1) for(i=0;i<n;i++) printf("%s %d\n",name[i],score[i]);
if(d==1) printf("%s %d\n",name[mid],score[mid]);
else printf("not found!");
return 0;
}

```

3) 测试

(a) 测试数据:

先输入菜单数 1，再输入人数，接着是姓名和成绩，根据菜单通过由 1-4 执行，依次测试对应菜单功能的实现，最后输入 0 结尾。

(b) 菜单 1 的运行结果如图 5-7 所示。

```

1
6
Jack 95
Mike 90
Joe 75
Andy 95
Rose 89
Sophia 7
0
6 records were input!
-----
Process exited after 60.01 seconds with return value 0

```

图 5-7 程序设计题 2 的测试菜单 1 的运行结果

菜单 1 和 2 的运行结果如图 5-8 所示。

```

1
6
Jack 95
Mike 90
Joe 75
Andy 95
Rose 89
Sophia 77
2
0
6 records were input!
Reorder finished!
-----
Process exited after 6.359 seconds with return value 0

```

图 5-8 程序设计题 2 的测试菜单 1 和 2 的运行结果

菜单 1，2，3 的运行结果如图 5-9 所示。

```

1
6
Jack 95
Mike 90
Joe 75
Andy 95
Rose 89
Sophia 77
2
3
0
6 records were input!
Reorder finished!
Jack 95
Andy 95
Mike 90
Rose 89
Sophia 77
Joe 75
-----
Process exited after 14.11 seconds with return value 0

```

图 5-9 程序设计题 2 的测试菜单 1，2，3 的运行结果

菜单 1, 2, 3, 4 的运行结果如图 5-10 所示。

```

1
6
Jack 95
Mike 90
Joe 75
Andy 95
Rose 89
Sophia 77
2
3
4
89
0
6 records were input!
Reorder finished!
Jack 95
Andy 95
Mike 90
Rose 89
Sophia 77
Joe 75
Rose 89
-----
Process exited after 10.2 seconds with return value 0
    
```

图 5-10 程序设计题 2 的测试菜单 1, 2, 3, 4 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。

(3) 求解 N 皇后问题，即在 $N \times N$ 的棋盘上摆放 N 个皇后，要求任意两个皇后不能在同一行、同一列、同一对角线上。输入棋盘的大小 N (N 取值 1-10)，如果能满足摆放要求，则输出所有可能的摆放法的数量，否则输出“无解”

解答：

1) 解题思路：

1.定义全局变量 size, flag, result_num, 以及数组 a, 和常量 maxsize。

2.分别用不同的函数执行操作。

2.1 check 函数用来检查竖直以及斜对角是否符合，符合则返回 1。

2.2 solve 函数用来执行重要过程，用递归改变行和用循环改变列，最后结束条件为最后一行也满足，再使次数增加。

2.3 主函数用来调用函数和对无解进行判断，再进行输出。

3.结束

2) 程序清单

```
#include <stdio.h>
```

```

#define maxsize 20
int a[maxsize][maxsize]={0};
int size;
int flag;
int result_num=0;
int check(int row,int col);
void solve(int row);
int main(){scanf("%d",&size);int a;
solve(0);
if (flag!=1) printf("无解") ;
else printf("%d",result_num);
return 0;
}
void solve(int row)
{
    int i,j;
    if(row>=size) {result_num++;
flag=1;}
else{
for(i=0;i<size;i++){
    if(check(row,i)){a[row][i]=1;
    solve(row+1);
    a[row][i]=0;}
}
}
}
int check(int row,int col)
{int ok=1,i;
for(i=0;i<row&&ok;i++){
    if(a[i][col]==1)
    ok=0;
}
for(i=1;row-i>=0&&col-i>=0&&ok;i++){
    if(a[row-i][col-i]==1) ok=0;
}
for(i=1;row-i>=0&&col+i<size&&ok;i++){
    if(a[row-i][col+i]==1) ok=0;
}
return ok;
}
}

```

3) 测试

(a) 测试数据:

测试一个有解值和无解值。

如表 5-2 所示。

表 5-2 编程题 3 的测试数据

测试 用例	程 序 输 入	理 论 结 果
	N	

用例 1	1	1
用例 2	2	无解

(b) 对应测试用例 1 的运行结果如图 5-11 所示。

```
1
1
-----
Process exited after 2.276 seconds with return value 0
```

图 5-11 程序设计题 3 的测试用例 1 的运行结果

对应测试用例 2 的运行结果如图 5-12 所示。

```
2
无解
-----
Process exited after 5.39 seconds with return value 0
```

图 5-12 程序设计题 3 的测试用例 2 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。

(4) 实现一个铁路购票系统的简单分配算法，用来处理一节车厢的座位分配。假设一节车厢有 20 排，每一排有 5 个座位，用 A、B、C、D、F 表示，第一排是 1A、1B、1C、1D、1F，第二排是 2A、2B、2C、2D、2F，以此类推，第 20 排是 20A、20B、20C、20D、20F。购票时，每次最多够 5 张，座位的分配策略是：如果这几张票能安排在同一排相邻座位，则应该安排在编号最小的相邻座位；否则，应该安排在编号最小的几个空座位中（不考虑是否相邻）。

解答：

1) 解题思路：

1. 围绕 `empty[flag]` 进行，这是表明某一行剩余的座位数

2. 一开始要通过循环将所需票数和空座位比较，能放就放，并及时对空座位数进行修改，不能放就比较下一行，当二十行都排完了，其他不需相邻的票需要对已经排位了的和剩余数量进行统计，通过自减和判断条件实现。

3. 注意临界条件，在输出的时候对何时输出空格要进行判断，以及在数字转字符运算的时候对除数是否加一进行判断。

4. 结束

2) 程序清单

```
#include<stdio.h>
#define N 100
```

```

int empty[20]; //记录每一排空余的座位数
void assign_seat(int ticket[],int p)
{
    int i,j,flag=-1,start,count=0;
    for(i=0;i<20;i++)
    {
        if(empty[i]>=p)
        {
            flag=i; //判断是否可以安排在同一编号的相邻座位
            break;
        }
    }
    if(flag>=0) //若可以
    {
        start=flag*5+(5-empty[flag]); //可以购买的第一张票的座位
        for(i=0;i<p;i++)
        {
            char c;
            int s1,s2;
            s1=ticket[start+i]/5;
            s2=ticket[start+i]%5;
            if(s2) s1++;
            switch(s2){
                case 1: c='A';break;
                case 2: c='B';break;
                case 3: c='C';break;
                case 4: c='D';break;
                case 0: c='F';break;
            }
            if(i<(p-1)) printf("%d%c ",s1,c);
            else printf("%d%c",s1,c);
            ticket[start+i]=0; //重置为 0
        }
        printf("\n");
        empty[flag]-=p; //修改该排剩余座位数
    }
    else
    {
        for(i=0;i<20;i++)
        {
            if(empty[i])
            {
                start=i*5+(5-empty[i]);
                while(empty[i])
                {
                    char c;
                    int s1,s2;
                    s1=ticket[start]/5;
                    s2=ticket[start]%5;

```



```

        if(s2) s1++;
    switch(s2){
        case 1: c='A';break;
        case 2: c='B';break;
        case 3: c='C';break;
        case 4: c='D';break;
        case 0: c='F';break;
    }
    count++; //记录已购买的票数
    if(count!=p) printf("%d%c ",s1,c);
    else printf("%d%c",s1,c);
    empty[i]--;
    ticket[start]=0;
    start++;
}
if(count==p)
{
    printf("\n");
    break;
}
}
}
}
}

int main()
{
    int n,i,number[N],ticket[100];
    scanf("%d",&n);
    for(i=0;i<n;i++) scanf("%d",&number[i]);
    for(i=0;i<100;i++) ticket[i]=i+1;
    for(i=0;i<20;i++) empty[i]=5;
    for(i=0;i<n;i++) assign_seat(ticket,number[i]);
    return 0;
}

```

3) 测试

(a) 测试数据:

测试一组在 20 组以内且有的可以在一行，一组在 20 组以外且必须要返回。

如表 5-3 所示。

表 5-3 编程题 4 的测试数据

测试 用例	程 序 输 入		理 论 结 果
	number[n]	n	
用例 1	2 5 4 2	4	1A 1B 2A 2B 2C 2D 2F

			3A 3B 3C 3D 1C 1D
用例 2	4 5 3 4 5 5 5 5 5 5 4 4 4 4 4 4 3 3 3	21	1A 1B 1C 1D 2A 2B 2C 2D 2F 3A 3B 3C 4A 4B 4C 4D 5A 5B 5C 5D 5F 6A 6B 6C 6D 6F 7A 7B 7C 7D 7F 8A 8B 8C 8D 8F 9A 9B 9C 9D 9F 10A 10B 10C 10D 10F 11A 11B 11C 11D 11F 12A 12B 12C 12D 13A 13B 13C 13D 14A 14B 14C 14D 15A 15B 15C 15D 16A 16B 16C 16D 17A 17B 17C 17D 18A 18B 18C 18D 19A 19B 19C 20A 20B 20C 1F 3D 3F

(b) 对应测试用例 1 的运行结果如图 5-13 所示。

```

4
2 5 4 2
1A 1B
2A 2B 2C 2D 2F
3A 3B 3C 3D
1C 1D

-----
Process exited after 7.809 seconds with return value 0
    
```

图 5-13 程序设计题 4 的测试用例 1 的运行结果

对应测试用例 2 的运行结果如图 5-14 所示。

```

21
4 5 3 4 5 5 5 5 5 5 4 4 4 4 4 4 3 3 3
1A 1B 1C 1D
2A 2B 2C 2D 2F
3A 3B 3C
4A 4B 4C 4D
5A 5B 5C 5D 5F
6A 6B 6C 6D 6F
7A 7B 7C 7D 7F
8A 8B 8C 8D 8F
9A 9B 9C 9D 9F
10A 10B 10C 10D 10F
11A 11B 11C 11D 11F
12A 12B 12C 12D
13A 13B 13C 13D
14A 14B 14C 14D
15A 15B 15C 15D
16A 16B 16C 16D
17A 17B 17C 17D
18A 18B 18C 18D
19A 19B 19C
20A 20B 20C
1F 3D 3F
-----
Process exited after 4.009 seconds with return value 0

```

图 5-14 程序设计题 4 的测试用例 2 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。

(5) 将数组中指定的两段数据交换。输入 n 个整数到数组 a 中，再输入正整数 $m1$ 、 $n1$ 、 $m2$ 、 $n2$ ($0 \leq m1 \leq n1 < m2 \leq n2 < n$)，将数组中由 $m1$ 、 $n1$ 指定的一段数据和由 $m2$ 、 $n2$ 指定的一段数据交换位置，其它数据位置不变，输出重新排列后的数组元素。 **要求：**1. 将交换数组中两段数据的功能定义为函数。2. 所有的操作都在数组 a 上完成，不允许使用其它数组。

解答：

1) 解题思路：

1.输入数字，并利用循环存入数组，这里数组 a 需要扩大内存空间。

2.使用多个函数完成程序

2.1 函数 `change` 用于两端数据交换，主要细节在于有些值在循环中会改变，需要一开始确定，其次一些字节距离的使用需要严谨判断。

2.2 主函数用于输入，输出，以及调用函数。

3. 结束

2) 程序清单

```

#include<stdio.h>
void change(int a[],int m1,int n1,int m2,int n2,int n)
{

```

```

int i=n1-m1+1,b=m2-n1-1,c=m1,e=n,j,k;
for(;m1<=m2-1;m1++) a[n++]=a[m1];
for(;m2<=n2;m2++) a[c++]=a[m2];
for(k=0;k<b;e++,k++) a[c++]=a[e+i];
for(j=0;j<i;e++,j++) a[c++]=a[e-b];
}
int main()
{
    int n,i,m1,n1,m2,n2;
    scanf("%d",&n);
    int a[100]={0};
    for(i=0;i<n;i++) scanf("%d",&a[i]);
    scanf("%d %d %d %d",&m1,&n1,&m2,&n2);
    change(a,m1,n1,m2,n2,n);
    for(i=0;i<n;i++) printf("%d ",a[i]);
    return 0;
}

```

3) 测试

(a) 测试数据:

选择一组有重复数字的，一组没有。

如表 5-4 所示。

表 5-4 编程题 5 的测试数据

测试 用例	程 序 输 入				理 论 结 果
	a[100]	m1 n1 m2 n2	n		
用例 1	1 2 3 4 5 6 7	1 2 4 6	7	1 5 6 7 4 2 3	
用例 2	1 2 3 4 5 6 7	0 2 6 6	7	7 4 5 6 1 2 3	

(b) 对应测试用例 1 的运行结果如图 5-15 所示。

```

7
1 2 3 4 5 6 7
1 2 4 6
1 5 6 7 4 2 3
-----
Process exited after 23.1 seconds with return value 0

```

图 5-15 程序设计题 5 的测试用例 1 的运行结果

对应测试用例 2 的运行结果如图 5-16 所示。

```

7
1 2 3 4 5 6 7
0 2 6 6
7 4 5 6 1 2 3
-----
Process exited after 19.95 seconds with return value 0

```

图 5-16 程序设计题 2 的测试用例 2 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。

5.3 实验小结

开始的改错，第一个的问题固然是数组 `b` 原先的内存空间不够，再连接 `a` 的字符串会造成溢出，但是对于数组 `ab` 的存储顺序和字节数我产生了疑问。首先关于存储顺序，搜索知道了堆栈的“先入后出”的存储和释放模式，先从内存高位存起，所以先定义的数组 `a` 存储在较高位，但是一个数组内部的元素则是从低到高依次连续存储，所以与按逻辑低到高的顺序调用吻合。其次是字节数的问题，在我的编译器上对于数组 `b` 如果在十六个字符以内系统会分配十六个字节，超出十六在三十二之间则分配三十二个字节，以此类推，所以对于 `a` 取地址取得的字符串，并不是按我们所想的从第二个字母开始，但是我用另一个版本的编译器则是系统未自行增加或者补充字节，直接是从第二个字母开始，我开始搜集资料，知道了“内存对齐”的概念，不同的编译器会有不同的处理方式，而且与所使用的其他变量也有关系，这里便不再赘述。

接下来对于瑟夫问题，这是一个很有趣的问题且有很多有效且多样的方法，但是这个问题确实很绕，总是看了后面忘了前面，填空的时候需要不断地结合上下的代码反复验证数值。在改写程序的时候想到了课上的方法，思索改写的方式，因为需要输出淘汰的人员名单，便需要在改变时同时输出，有了上面的代码，同时我还思索是不是仍然能用题给的双数组的模式进行修改，原理类似只是需要注意判断条件，判断该号码出圈的两个条件成立时，需要报数的号码自减。

对于 OJ 的任务确实花的时间比较多，任务一，我一开始只考虑到要一位一位对字节数进行存储，后面运行发现对于负数的存储会出现问题，全部都是斜杠，尝试用十进制的运算模式添加了负号，虽然答案对了，但是对于电脑二进制运算模式还有疑问，搜集了资料 and 与同学讨论了一下，更深入理解了二进制的对符号位的定义和运算。任务二，需要的功能函数很多，就是一个个尝试，在对于二维数组存储姓名的数组内容的改变存在了问题，想要整个字符串和下一组交换，尝试了一下只打一个中括号，发现可以运行，后面去搜了一下原因发现二维数组是可以作为一维数组调用，对这个印象很深，有时候多尝试是很重要的，然后要善于利用互联网获取自己需要的信息。任务三，N 皇后看懂了课上的例子就问题不大了，就将打印棋盘的程序删去，将其改为计数，再用 `flag` 判断是否有解即可。任务四，购票的输出比较麻烦，显示在上面的是我最初的想法将其转换成数字再转换成字符，比较麻烦，所以代码也比较长，而且本身题目限制条件也比较多，比如结尾不能有空格，就需要充分结合判断条件进行输出，还有一种二维数组的方式这里不再说明，代码会更加简便，所呈现的这种一定不是最简单且最好理解的，但是这里秉持初心吧。任务五，一开始看到题目就马上想到用多个数组进行部分储存再拼接，往后看发现不可以用其它数组，就想到在一个数组中完成，只能把数组 `a` 的内存扩大，然后其中一个麻烦的点在于其中的变量一直在改变会影响判断条件，需要开始设置多个变量储存传入变量起始值，最后反复调试和修改也完成了。

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

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	$*p++ \rightarrow t, *p \rightarrow t$	x	x
4	$*(++p) \rightarrow t$	x	x
5	$*++p \rightarrow t$	V	V
6	$++*p \rightarrow t$	V	V

解答：

- 1) 表达式的值如上。
- 2) 源程序清单

```
#include<stdio.h>
char u[]="UVWXYZ";
char v[]="xyz";
struct T{
    int x;
    char c;
```

```

char *t;
}a[]={11,'A',u},{100,'B',v}},*p=a;
int main()
{
// printf("(++p)->x=%d",(++p)->x);
// p++;
// printf("p->c=%c",p->c);
// *p++->t;
// printf("*p->t=%c",*p->t);
// printf("*(++p)->t=%c",*(++p)->t);
// printf("**++p->t=%c",**++p->t);
// printf("+++p->t=%c",+++p->t);
return 0;
}

```

运行结果如图 7-1 所示。

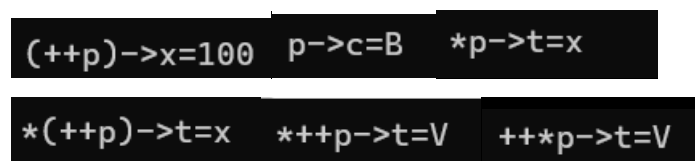


图 7-1 实验 1-1 运行结果截图

7.2.2 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) 在函数 creat_list 中，需要被操作的是头指针的地址值，所以参数类型不应该是 struct s_list *headp 而是 struct s_list **headp，同样在引用和赋值的时候也需要相应的改变类型。
- (2) 先进后出的链表不变，和（1）相同。改变后续链表指向实验题目要求。

源程序清单如下：

```

1) #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) #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;
    struct s_list *temp;//临时指针，用来指向新创建的结点

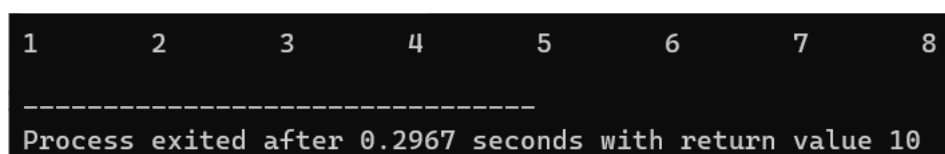
```

```

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;
}

```

(3) 修改后运行结果如图 7-2 所示。



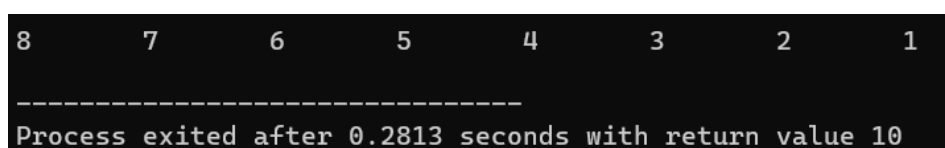
```

1      2      3      4      5      6      7      8
-----
Process exited after 0.2967 seconds with return value 10

```

图 7-2 实验 2-1 修改后运行结果截图

修改替换后运行结果如图 7-3 所示。



```

8      7      6      5      4      3      2      1
-----
Process exited after 0.2813 seconds with return value 10

```

图 7-3 实验 2-2 修改替换后运行结果截图

7.2.3 程序设计

(1) 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。除菜单 0 为退出外，菜单 1-5 分别实现下列功能：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：

1.调用菜单 1 创建链表， 把链表头指针返回

2.建立相应函数， 包含所需功能

2.1 out 函数输出每个学生的各项信息。

2.2 change 函数修改指定学生的指定数据项的内容。

2.3 average 函数统计每个同学的平均成绩（保留 2 位小数）。

2.4 sum 函数输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

3. 用主函数进行不同菜单的输入和调用，以及最后信息输出的一系列执行。

4. 释放 malloc 空间，程序结束

2) 程序清单

```
#include<stdio.h>
#include<stdlib.h>
struct stu{
    int number;
    char name[8];
    int English,math,physics,C;
    struct stu* next;
};
int num;
struct stu* qq;
struct stu* creat()
{
    scanf("%d",&num);
    int i;
    struct stu* h=(struct stu*)malloc(sizeof(struct stu));
    struct stu* p2=h;
    for(i=0;i<num;i++)
    {
        struct stu* p=(struct stu*)malloc(sizeof(struct stu));

        scanf("%d%s%d%d%d%d",&p->number,p->name,&p->English,&p->math,&p->
physics,&p->C);
        p2->next=p;
        p2=p;
        p->next=NULL;
    }
    return h;
}

void out(struct stu* p)
{
    struct stu* q=p->next;
    while(q!=NULL){

        printf("%d %s %d %d %d %d\n",q->number,q->name,q->English,q->math,q->ph
ysics,q->C);
        q=q->next;
    }
}

void change(struct stu* p)
{

```

```

    qq=p->next;
    int number,n,i;
    scanf("%d",&number);
    scanf("%d",&n);
    for(i=0;i<number-2021001;i++) qq=qq->next;
    switch(n){
        case 0:scanf("%s",qq->name);break;
        case 1:scanf("%d",&qq->English);break;
        case 2:scanf("%d",&qq->math);break;
        case 3:scanf("%d",&qq->physics);break;
        case 4:scanf("%d",&qq->C);break;
    }

}

void average(struct stu* p)
{
    int sum=0;
    struct stu* q=p->next;
    while(q!=NULL){
        sum=q->C+q->physics+q->English+q->math;
        printf("%d %s %.2f\n",q->number,q->name,sum/4.0);
        sum=0;
        q=q->next;
    }
}

void sum(struct stu* q)
{
    int sum=0;
    struct stu* p=q->next;
    while(p!=NULL){
        sum=p->C+p->physics+p->English+p->math;
        printf("%d %s %d %.2f\n",p->number,p->name,sum,sum/4.0);
        sum=0;
        p=p->next;
    }
}

void main()
{
    struct stu* h,* p;
    int n,a,b,c,d,e;
    a=b=c=d=e=0;
    scanf("%d",&n);
    while(n)
    {
        switch(n){

```

```

        case 1:h=creat();scanf("%d",&n);a=1;break;
        case 2:scanf("%d",&n);b=1;break;
        case 3:change(h);scanf("%d",&n);c=1;break;
        case 4:scanf("%d",&n);d=1;break;
        case 5:scanf("%d",&n);e=1;break;
    }
}
if(a) printf("完成了%d 位同学的成绩输入\n",num);
if(b) out(h);
if(c)
printf("%d %s %d %d %d %d\n",qq->number,qq->name,qq->English,qq->math,qq->
physics,qq->C);
if(d) average(h);
if(e) sum(h);
while(h!=NULL){
    p=h->next;
    free(h);
    h=p;
}
}

```

3) 测试

(a) 测试数据:

测试菜单 345。

如表 7-1 所示。

表 7-1 程序设计题 1 的测试数据

测试 用例	程 序 输 入	理 论 结 果
	n	
用例 1	3 2021003 0 Joan	2021003 Joan 77 86 90 75
用例 2	4	2021001 Jack 91.00 2021002 Mike 80.00 2021003 Joe 84.50
用例 3	5	2021001 Jack 364 91.00 2021002 Mike 320 80.00 2021003 Joe 328 82.00

(b) 对应测试测试用例 1 的运行结果如图 7-4 所示。

```

1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
3
2021003 0 Joan
0
完成了3位同学的成绩输入
2021003 Joan 77 86 90 75
-----
Process exited after 9.302 seconds with return value 0

```

图 7-4 程序设计题 1 的测试用例一的运行结果

对应测试用例 2 的运行结果如图 7-5 所示。

```
1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
4
0
完成了3位同学的成绩输入
2021001 Jack 91.00
2021002 Mike 80.00
2021003 Joe 82.00
-----
Process exited after 1.805 seconds with return value 0
```

图 7-5 程序设计题 1 的测试用例二的运行结果

对应测试用例 3 的运行结果如图 7-6 所示。

```
1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
5
0
完成了3位同学的成绩输入
2021001 Jack 364 91.00
2021002 Mike 320 80.00
2021003 Joe 328 82.00
-----
Process exited after 1.658 seconds with return value 0
```

图 7-6 程序设计题 1 的测试用例三的运行结果

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)在实验 7-1 的基础上增加菜单选择项 6: ⑥增加按照平均成绩进行升序(0)及降序(1)排序的函数，写出用交换结点数据域的方法排序的函数，排序可指定用选择法(0)或冒泡法(1)。

解答:

1) 解题思路:

- 1.其他菜单运行同上一个程序一样，增加一个结构数组 q，和函数 roll。
- 2.数组 q 用来存数据，便于在排序直接进行数据域的改变。
- 3.通过输入 m,n 进行升序降序，冒泡法选择法的选择。
4. 用结构数组 q 进行输出，释放 malloc 内存，程序结束。

2) 程序清单

```
#include<stdio.h>
#include<stdlib.h>
struct stu{
    int number;
    char name[8];
    int English,math,physics,C;
    struct stu* next;
};
int num;
struct stu* qq;
struct stu q[10];
double a[10];

struct stu* creat()
{
    scanf("%d",&num);
    int i;
    struct stu* h=(struct stu*)malloc(sizeof(struct stu));
    struct stu* p2=h;
    for(i=0;i<num;i++)
    {
        struct stu* p=(struct stu*)malloc(sizeof(struct stu));
        scanf("%d%s%d%d%d%d",&p->number,p->name,&p->English,&p->math,&p->
physics,&p->C);
        q[i].number=p->number;
        int j;
        for(j=0;j<8;j++)
            q[i].name[j]=p->name[j];
        q[i].English=p->English;
        q[i].math=p->math;
        q[i].physics=p->physics;
        q[i].C=p->C;
        p2->next=p,p2=p,p->next=NULL;
    }
    return h;
}

void out(struct stu* p)
{
    struct stu* q=p->next;
    while(q!=NULL){
        printf("%d %s %d %d %d %d\n",q->number,q->name,q->English,q->math,q->ph
ysics,q->C);
        q=q->next;
    }
}

void change(struct stu* p)
```

```

{
    qq=p->next;
    int number,n,i;
    scanf("%d",&number);
    scanf("%d",&n);
    for(i=0;i<number-2021001;i++) qq=qq->next;
    switch(n){
        case 0:scanf("%s",qq->name);break;
        case 1:scanf("%d",&qq->English);break;
        case 2:scanf("%d",&qq->math);break;
        case 3:scanf("%d",&qq->physics);break;
        case 4:scanf("%d",&qq->C);break;
    }
}

void average(struct stu* p)
{
    int sum=0;
    struct stu* q=p->next;
    while(q!=NULL){
        sum=q->C+q->physics+q->English+q->math;
        printf("%d %s %.2f\n",q->number,q->name,sum/4.0);
        sum=0;
        q=q->next;
    }
}

void sum(struct stu* q)
{
    int sum=0;
    struct stu* p=q->next;
    while(p!=NULL){
        sum=p->C+p->physics+p->English+p->math;
        printf("%d %s %d %.2f\n",p->number,p->name,sum,sum/4.0);
        sum=0;
        p=p->next;
    }
}

void roll(struct stu* p)
{
    struct stu* pp=p->next,qq;
    int i=0,sum;
    while(pp!=NULL){
        sum=pp->C+pp->physics+pp->English+pp->math;
        a[i++]=sum/4.0;
        sum=0;
        pp=pp->next;
    }
    int n,m;

```



```

scanf("%d%d",&n,&m);
int j,k;
double t;
if(m){
    for(i=num-1;i>=0;i--){
        for(j=1,k=0;j<=i;j++,k++){
            if(n?a[j]>a[k]:a[j]<a[k]){
                t=a[j],qq=q[j],a[j]=a[k],q[j]=q[k],a[k]=t,q[k]=qq;
            }
        }
    }
}
if(!m){
    int max,min;
    for(i=0;i<num;i++){
        for(max=i,j=i;j<num;j++){
            if(n?a[j]>a[max]:a[j]<a[max]) max=j;
        }
        t=a[i],a[i]=a[max],a[max]=t,qq=q[i],q[i]=q[max],q[max]=qq;
    }
}
}

void main()
{
    struct stu* h,* p;
    int n,a1,b,c,d,e,f;
    a1=b=c=d=e=f=0;
    scanf("%d",&n);
    while(n)
    {
        switch(n){
            case 1:h=creat();scanf("%d",&n);a1=1;break;
            case 2:scanf("%d",&n);b=1;break;
            case 3:change(h);scanf("%d",&n);c=1;break;
            case 4:scanf("%d",&n);d=1;break;
            case 5:scanf("%d",&n);e=1;break;
            case 6:roll(h);scanf("%d",&n);f=1;break;
        }
    }
    if(a1) printf("完成了%d 位同学的成绩输入\n",num);
    if(b) out(h);
    if(c)
        printf("%d %s %d %d %d %d\n",qq->number,qq->name,qq->English,qq->math,qq->
        physics,qq->C);
    if(d) average(h);
    if(e) sum(h);
    if(f) {
        int i;
        for(i=0;i<num;i++){

```

```

        printf("%d %s %.2f\n",q[i].number,q[i].name,a[i]);
    }
}
while(h!=NULL){
    p=h->next;
    free(h);
    h=p;
}
}

```

3) 测试

(a) 测试数据:

测试选择法升序和冒泡法降序。

如表 7-2 所示。

表 7-2 程序设计题 2 的测试数据

测试用例	程序输入		理论结果
	m	n	
用例 1	0	0	2021002 Mike 80.00 2021003 Joe 82.00 2021001 Jack 91.00
用例 2	1	1	2021001 Jack 91.00 2021003 Joe 82.00 2021002 Mike 80.00

(b) 对应测试用例 1 的运行结果如图 7-7 所示。

```

1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
6
0 0
0
完成了3位同学的成绩输入
2021002 Mike 80.00
2021003 Joe 82.00
2021001 Jack 91.00
-----
Process exited after 14.07 seconds with return value 0

```

图 7-7 程序设计题 2 的测试用例一的运行结果

对应测试用例 2 的运行结果如图 7-8 所示。

```

1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
6
1 1
0
完成了3位同学的成绩输入
2021001 Jack 91.00
2021003 Joe 82.00
2021002 Mike 80.00
-----
Process exited after 12.82 seconds with return value 0

```

图 7-8 程序设计题 2 的测试用例二的运行结果

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。

(3) 将实验 7-1 的菜单选择项 6 修改为：⑥增加按照平均成绩进行升序（0）及降序（1）排序的函数，写出用**交换结点指针域**的方法排序的函数，排序可指定用选择法（0）或冒泡法（1）。

解答：

1) 解题思路：

1.其他菜单运行同上一个程序一样，结构增加 ave 将平均分记录，便于之后进行比较，以及函数 struct student* bubble_sort 和 struct student * select_sort。

2.函数以结构指针作为返回值，在排序进行指针域的改变，便于输出。

3.通过输入 m,n 进行升序降序，冒泡法选择法的选择。

4.进行输出，释放 malloc 内存，程序结束。

2) 程序清单

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct student
{
    char num[8];
    char name[21];
    int scores[4];
    float ave;
    struct student* next;
}stu,*head,*p,*p1,*pp;
int sum[10];
struct student* bubble_sort(struct student *head,int m);
struct student * select_sort(struct student *head,int n,int m);
int main()
{

```

```

int num,i,j;
int a=0,b=0,c=0,d=0,e=0,f=0;
scanf("%d",&i);
while(i)
{
    switch(i)
    {
        case 1:
            scanf("%d",&num);
            head=(struct student*)malloc(sizeof(struct student));
            p1=head;
            for (j=0;j<num;j++)
            {
                p=(struct student*)malloc(sizeof(struct student));
                scanf("%s%s%d%d%d%d",&p->num,p->name,&p->scores[0],
                    &p->scores[1],&p->scores[2],&p->scores[3]);
                p1->next=p;
                p1=p;
                p->next=NULL;
            }
            a=1;
            head=head->next;
            scanf("%d",&i);break;
        case 2:
            b=1;
            scanf("%d",&i);break;
        case 3:{
            int k,new_score;
            char new_name[21],Num[8];
            scanf("%s%d",Num,&k);
            if (k!=0)
                scanf("%d",&new_score);
            else
                scanf("%s",new_name);
            for (j=0,p=head;j<num;j++)
            {
                if (strcmp(p->num,Num)==0)
                {
                    switch(k)
                    {
                        case 0:strcpy(p->name,new_name);break;
                        case 1:p->scores[0]=new_score;break;
                        case 2:p->scores[1]=new_score;break;
                        case 3:p->scores[2]=new_score;break;
                        case 4:p->scores[3]=new_score;break;
                    }
                    pp=p;
                    break;
                }
                p=p->next;
            }
        }
    }
}

```

```

    }
    c=1;
    scanf("%d",&i);break;}
case 4:
    for (j=0,p=head;j<num;j++)
    {
        p->ave=(p->scores[0]+p->scores[1]+p->scores[2]+p->scores[3])/4.0;
        p=p->next;
    }
    d=1;
    scanf("%d",&i);break;
case 5:
    for(j=0,p=head;j<num;j++)
    {
        sum[j]=p->scores[0]+p->scores[1]+p->scores[2]+p->scores[3];
        p=p->next;
    }
    for (j=0,p=head;j<num;j++)
    {
        p->ave=(p->scores[0]+p->scores[1]+p->scores[2]+p->scores[3])/4.0;
        p=p->next;
    }
    e=1;
    scanf("%d",&i);break;
case 6:
    for (j=0,p=head;j<num;j++)
    {
        p->ave=(p->scores[0]+p->scores[1]+p->scores[2]+p->scores[3])/4.0;
        p=p->next;
    }
    int m,n;
    scanf("%d%d",&m,&n);
    if (!n) head=select_sort(head,num,m);
    if (n) head=bubble_sort(head,m);
    f=1;
    scanf("%d",&i);break;
}
}
if (a==1)
printf("完成了%d 位同学的成绩输入\n",num);
if (b==1)
{
    for (j=0,p=head;j<num;j++)
    {
        printf("%s %s %d %d %d %d\n",p->num,p->name,p->scores[0],
            p->scores[1], p->scores[2],p->scores[3]);
        p=p->next;
    }
}
}

```

```

if (c==1)
printf("%s %s %d %d %d %d\n",pp->num,pp->name,pp->scores[0],
      pp->scores[1], pp->scores[2],pp->scores[3]);
if (d==1)
{
    for (j=0,p=head;j<num;j++)
    {
        printf("%s %s %.2f\n",p->num,p->name,p->ave);
        p=p->next;
    }
}
if (e==1)
{
    for (j=0,p=head;j<num;j++)
    {
        printf("%s %s %d %.2f\n",p->num,p->name,sum[j],p->ave);
        p=p->next;
    }
}
if (f==1)
{
    for (j=0,p=head;j<num;j++)
    {
        printf("%s %s %.2f\n",p->num,p->name,p->ave);
        p=p->next;
    }
}
while(head)
{
    p1=head;
    head=head->next;
    free(p1);
}
return 0;
}
}

struct student* bubble_sort(struct student *head,int m){
    struct student *tail=NULL,*q,*p=(struct student*)malloc(sizeof(struct student));
    for(p->next=head,head=p;head->next!=tail;tail=q){
        for(p=head,q=p->next;q->next!=tail;p=p->next,q=p->next){
            if(m?p->next->ave<q->next->ave:p->next->ave>q->next->ave){
                p->next=q->next;
                q->next=q->next->next;
                p->next->next=q;
            }
        }
        head=head->next;
        free(p);
        return head;
    }
}

```

```

struct student * select_sort(struct student *head,int n,int m)
{
    int i,j;
    struct student *p1;
    struct student *max=head,*turn=head,*preturn=NULL,*premax=NULL;
    for (i=0;i<n-1;i++)
    {
        for (j=0,p1=max->next;j<n-i-1;j++,p1=p1->next)
        {
            if (m?p1->ave>max->ave:p1->ave<max->ave)
            {
                premax=max;
                max=p1;
            }
        }
        if (turn!=max)
        {
            if (turn==head)
                head=max;
            else preturn->next=max;
            struct student* temp=temp=(struct student*)malloc(sizeof(struct student));
            temp=max->next;
            if (turn->next==max)
            {
                max->next=turn;
                turn->next=temp;
            }
            else{
                max->next=turn->next;
                premax->next=turn;
                turn->next=temp;
            }
        }
        preturn=max;
        max=max->next;
        turn=max;
    }
    return head;
}

```

3) 测试

(a) 测试数据:

测试选择法升序和冒泡法降序。

如表 7-3 所示。

表 7-3 程序设计题 3 的测试数据

测试	程序输入	理论结果
----	------	------

用例	m	n	
用例 1	0	0	2021002 Mike 80.00 2021003 Joe 82.00 2021001 Jack 91.00
用例 2	1	1	2021001 Jack 91.00 2021003 Joe 82.00 2021002 Mike 80.00

(b) 对应测试用例 1 的运行结果如图 7-9 所示。

```

1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
6
0 0
0
完成了3位同学的成绩输入
2021002 Mike 80.00
2021003 Joe 82.00
2021001 Jack 91.00
-----
Process exited after 15.1 seconds with return value 0

```

图 7-9 程序设计题 3 的测试用例一的运行结果

对应测试用例 2 的运行结果如图 7-10 所示。

```

1
3
2021001 Jack 90 92 87 95
2021002 Mike 85 70 75 90
2021003 Joe 77 86 90 75
6
1 1
0
完成了3位同学的成绩输入
2021001 Jack 91.00
2021003 Joe 82.00
2021002 Mike 80.00
-----
Process exited after 5.815 seconds with return value 0

```

图 7-10 程序设计题 3 的测试用例二的运行结果

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

(4) 约瑟夫问题。N 个人围成一圈并从 1 到 N 编号，从编号为 1 的人循序依次从 1 到 M 报数，报数为 M 的人退出圈子，后面紧挨着的人接着从 1 到 M 报数。如此下去，每次从 1 报到 M，都会有一个人退出圈子，直到圈子里剩下最后一个人，游戏结束。要求：用单向循环链表存放圈中人信息，编程求解约瑟夫问题，输出依次退出圈子的人和最后留在圈中人的编号。

解答：

1) 解题思路:

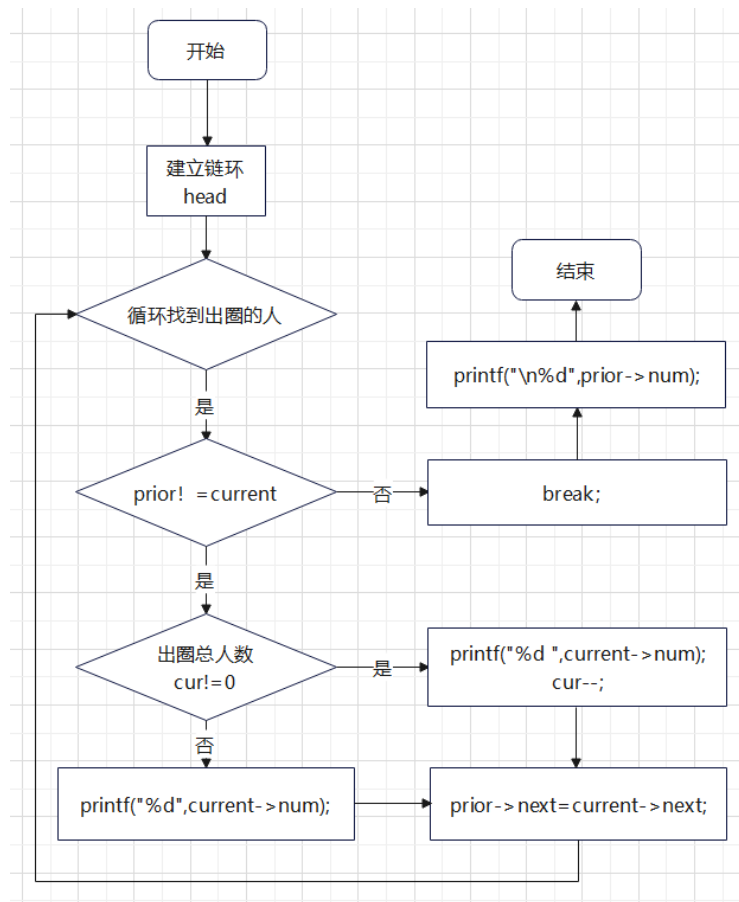


图 7-11 程序设计题 4 流程图

2) 程序清单

```

#include<stdio.h>
#include<stdlib.h>
struct people{
    int num;
    struct people* next;
}*head,*p,*p2,*prior,* current;
void main()
{
    int N,M;
    scanf("%d%d",&N,&M);
    int i,cur=N-2;
    head=(struct people*)malloc(sizeof(struct people));
    p2=head;
    for(i=1;i<=N;i++)
    {
        p=(struct people*)malloc(sizeof(struct people));
        p->num=i;
        p2->next=p;
        p2=p;
        if(i<N) p->next=NULL;
        else p->next=head->next;
    }
}
    
```

```

    }
    prior=head;
    while(1){
        current=prior->next;
        for(i=1;i<M;i++) {
            prior=prior->next;
            current=current->next;
        }
        if(prior!=current){
            if(cur){
                printf("%d ",current->num);
                cur--;
            }else{
                printf("%d",current->num);
            }
        } else break;
        prior->next=current->next;
    }
    printf("\n%d",prior->num);
    while(head!=NULL){
        p=head;
        free(p);
        head=head->next;
    }
}

```

3) 测试

(a) 测试数据：

测试两组不同数据的情况。

如表 7-4 所示。

表 7-4 程序设计题 4 的测试数据

测试用例	程序输入		理论结果
	M	N	
用例 1	9	3	3 6 9 4 8 5 2 7 1
用例 2	20	3	3 6 9 12 15 18 15 10 14 19 4 11 17 7 16 8 2 13 20

(b) 对应测试用例 1 的运行结果如图 7-12 所示。

```

9 3
3 6 9 4 8 5 2 7
1
-----
Process exited after 3.192 seconds with return value 2

```

图 7-12 程序设计题 4 的测试用例一的运行结果

对应测试用例 2 的运行结果如图 7-13 所示。

```
20 3
3 6 9 12 15 18 1 5 10 14 19 4 11 17 7 16 8 2 13
20
-----
Process exited after 3.061 seconds with return value 3
```

图 7-13 程序设计题 4 的测试用例二的运行结果

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

7.3 实验小结

在实验题对不同运算符和标识符组合的结果进行猜测和运行，让我对于运算符优先级和结合性进一步了解。

程序设计题反复在考察对结构体的运用来创建链表，一开始经常会不知道怎么链表就断开了，后面尝试用画图去慢慢理解，一遍遍加强记忆，逐渐明白如何使用链表的传递和结点的生成。并了解到交换结点数据域和指针域的区别，进行反复尝试和修改，收获颇丰。

实验总结

这些题目的代码有些很长，限制条件和要求又比较繁琐，需要多个函数的设计执行和调用，过程略微复杂，所呈现在上面的代码不一定是最优解，但也是我不断修改和思索地结果。中间也不断遇到一些细节的问题，需要不断地调试，修改，运行，很磨砺耐心和提高对代码的使用和读写能力。

实验课是个很好地锻炼自己阅读和运用代码能力的形式，也确实每次需要花费不少时间，但每次做完都会收获满满，既会欣喜于自己的成长，也感谢那个为了弄懂一个点不断地去探讨以及搜集资料的自己，坚持而且不愿放弃，这可能就是对知识的渴求吧，接下来也会不断去努力，希望接下来能看到更好的自己！

参考文献

- [1] 卢萍,李开,王多强等. C 语言程序设计, 北京: 清华大学出版社, 2021
- [2] 卢萍,李开,王多强等. C 语言程序设计典型题解与实验指导, 北京: 清华大学出版社, 2019