

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 密码202101**

**学 号： U202116003**

**姓 名： 侯竣**

**指导教师： 吴俊军**

**报告日期： 2022年12月29日**

**网络空间安全学院**

目录

**[5 数组程序设计实验](#_Toc24075_WPSOffice_Level1)** **[1](#_Toc24075_WPSOffice_Level1)**

[5.1 实验目的](#_Toc26910_WPSOffice_Level2) [1](#_Toc26910_WPSOffice_Level2)

[5.2 实验内容及要求](#_Toc7357_WPSOffice_Level2) [1](#_Toc7357_WPSOffice_Level2)

[5.3 实验小结](#_Toc30953_WPSOffice_Level2) [23](#_Toc30953_WPSOffice_Level2)

**[7 结构与联合实验](#_Toc26910_WPSOffice_Level1)** **[23](#_Toc26910_WPSOffice_Level1)**

[7.1 实验目的](#_Toc7357_WPSOffice_Level1) [23](#_Toc7357_WPSOffice_Level1)

[7.2 实验内容及要求](#_Toc30953_WPSOffice_Level1) [24](#_Toc30953_WPSOffice_Level1)

[7.3 实验小结](#_Toc2039_WPSOffice_Level1) [47](#_Toc2039_WPSOffice_Level1)

**[参考文献](#_Toc5645_WPSOffice_Level1)** **[48](#_Toc5645_WPSOffice_Level1)**

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 实验内容及要求

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\*/

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[]="Programming";
7. printf("%s %s\n", b,a);
8. strcate(b,a); printf("%s %s\n",b,a);
9. strdelc(b, 'a'); printf("%s\n",b);
10. return 0;
11. }
12. void strcate(char t[],char s[])
13. {
14. int i = 0, j = 0;
15. while(t[i++]) ;
16. while((t[i++] = s[j++] )!= '\0');
17. }
18. void strdelc(char s[], char c)
19. {
20. int j,k;
21. for(j=k=0; s[j] != '\0'; j++)
22. if(s[j] != c) s[k++] = s[j];
23. }

**解答**：

1. 问题回答：
2. 跟踪进入strcate时，字符数组t和s中的内容分别为"Programming"和"Language"。

当单步执行光条刚落在第二个while语句所在行时，i=12，t[i]='L'。是有问题的, 应该为’\0’, 即字符串结尾

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"Programming"和"Language"，显然未实现字符串连接。

1. 跟踪进入函数strdelc时，字符串s="Programming"，字符c='a'。

单步执行for语句过程中，字符串s, j和k值的变化如图5-1所示。

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"Progrmmingg"，如图5-2, 显然未实现删除功能。

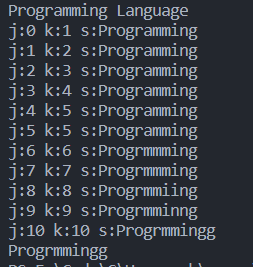


图5-1 字符数组s, j和k值截图



图5-2 实验5-1函数strdelc结束时字符串s截图

1. 错误修改：
2. 第6行，b数组后面空间不够，改为30。
3. 第15行后，添加i--。
4. 第22行后添加s[k]='\0'。

修改后源程序为:

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[30]="Programming";
7. printf("%s %s\n", b,a);
8. strcate(b,a); printf("%s %s\n",b,a);
9. strdelc(b, 'a'); printf("%s\n",b);
10. return 0;
11. }
12. void strcate(char t[],char s[])
13. {
14. int i = 0, j = 0;
15. while(t[i++]) ;
16. i--;
17. while((t[i++] = s[j++] )!= '\0');
18. }
19. void strdelc(char s[], char c)
20. {
21. int j,k;
22. for(j=k=0; s[j] != '\0'; j++)
23. if(s[j] != c) s[k++] = s[j];
24. s[k]='\0';
25. }
26. 错误修改后运行结果为:

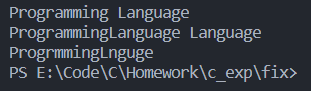


图5-3 实验5-1修改后运行结果截图

2. 源程序完善和修改替换

(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。

1. #include<stdio.h>
2. #define M 10
3. #define N 3
4. int main(void)
5. {
6. 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 \*/
7. int i, j, k;
8. for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/
9. a[i] = i + 1;
10. for(i = M, j = 0; i > 1; i--){
11. /\* 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 \*/
12. for(k = 1; k <= N; k++) /\* 1至N报数 \*/
13. 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 \*/
14. b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/
15. if(j)
16. for(k = --j; k < i-1; k++) /\* 压缩数组a，使报数为N的人出圈 \*/
17. \_\_\_\_\_\_\_\_\_\_\_\_\_\_;
18. }
19. for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/
20. printf(“%6d”, b[i]);
21. printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/
22. return 0;
23. }

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. 三处填空分别为：
2. a[j-1]
3. a[i-1]
4. a[k]=a[k+1]
5. （1）算法思路如下：
6. 先定义a，b两个一维数组，先循环给a编号
7. 再用for循环进行报号。
8. 利用for循环，每N个报一次，并通过是否等于0去除已经出圈
9. 将每次报号的人存入b数组，并将a数组中，报号者记为0。
10. 依次输出b数组的值, 即报号顺序, 最后输出a数组不为0的值, 即为最后一个出圈的人。
11. 源程序：

#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--)

{

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

if(a[j]==0) k--; //如果a[j]已经出圈，k--, 再往前走一步

if (++j > M - 1) j = 0;

}

b[M - i] = j ? a[j - 1] : a[M - 1]; //b[M-i]存放出圈人的编号

if(j)

a[j-1]=0; //a[j-1]出圈

else

a[M-1]=0; //a[M-1]出圈

}

for (i = 0; i < M - 1; i++) /\* 按次序输出出圈人的编号 \*/

printf("%6d", b[i]);

for(i=0;i<M-1;i++)

if(a[i]!=0) printf("%6d",a[i]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

1. 结果：



图5-4 实验5-2用例1的运行结果

1. 程序设计
2. 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. 实验思路如下：
2. 定义一个大小为33的字符数组a初始化为0, 用于存储结果
3. 定义常量mask 0x00000001。
4. 获取输入, 与mask循环相与得到二进制, 加上’0’存入a数组中保存结果, 并将num右移1位
5. 流程图

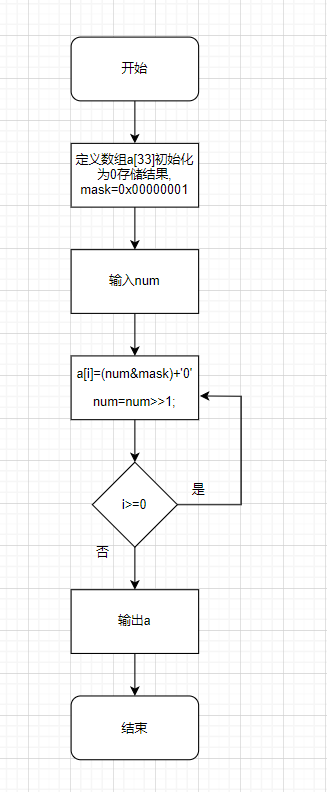


图5-5 算法流程图

1. 程序清单如下：

#include <stdio.h>

#define mask 0x00000001

int main() {

int num;

scanf("%d",&num);

char a[33]={0};

for(int i=31;i>=0;i--){

a[i]=(num&mask)+'0';

num=num>>1;

}

printf("%s",a);

return 0;

}

1. 测试。
2. 测试数据如表5-2所示。

表5-2 程序设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程 序 输 入 | 理论结果 | 实际结果 |
| x |
| 用例1 | 14 | 00000000000000000000000000001110 | 00000000000000000000000000001110 |
| 用例2 | -45 | 11111111111111111111111111010011 | 11111111111111111111111111010011 |
| 用例3 | 0 | 00000000000000000000000000000000 | 00000000000000000000000000000000 |

1. 运行结果:



图5-6 程序设计题1用例1的运行结果截图



图5-7 程序设计题1用例2的运行结果截图



图5-8 程序设计题1用例3的运行结果截图

1. 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。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. 实验思路如下：
2. 定义变量i，用于接收输入的菜单操作，定义变量n，用于表示需存储的数量，定义字符数组和整型数组用来存储姓名和成绩。
3. 定义InputScore函数，参数为n(输入人数)，成绩数组和名字数组, 循环读入。
4. 定义SortScore函数，参数为n, 成绩数组和名字数组，使用冒泡排序，将整型数组又小到大排序，同步更改名字数组。
5. 定义PrintScore函数，参数为n, 成绩数组和名字数组, 循环输出姓名与成绩。
6. 定义FindScore函数，二分查找输入的成绩，找到则输出成绩与对应的姓名，否则输出“not found!”。
7. 主函数读入各输入, 用switch分发到各函数进行操作
8. 程序清单如下：

#include <stdio.h>

#include "string.h"

// 成绩输入

void InputScore(int n,char b[][100],int a[])

{

int i;

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

scanf("%s%d",b[i],&a[i]);

printf("%d records were input!\n",i);

}

// 成绩排序

void SortScore(int n,char b[][100],int a[]){

int i,t;

char c[100];

for(i=0;i<n-1;i++)

{

for(int j=0;j<n-1;j++)

{

if (a[j]>=a[j + 1])

{

t = a[j], a[j] = a[j + 1], a[j + 1] = t;

strcpy(c, b[j]);

strcpy(b[j], b[j + 1]);

strcpy(b[j + 1], c);

}

}

}

printf("Reorder finished!\n");

}

// 成绩输出

void PrintScore(int n,char b[][100],int a[]){

for(int i=n-1;i>=0;i--){

printf("%s %d\n",b[i],a[i]);

}

}

// 寻找成绩

void FindScore(int x,int n,char b[][100],int a[])

{

int l=0, r=n-1, mid;

while(l<=r){

mid=(l+r)/2;

if(x<a[mid]) r=mid-1;

else if(x>a[mid]) {

l=mid+1;

}

else{

printf("%s %d\n", b[mid],a[mid]);

break;

}

}

if(x != a[mid]) printf("not found!\n");

}

int main(){

int i,n,k;

while ((scanf("%d",&i))!=0)

{

static int t;

if(i==1){

scanf("%d",&n);

}

t=n;

int a[20];

char b[20][100];

switch (i)

{

case 0:

return 0;

case 1:

InputScore(t,b,a);

break;

case 2:

SortScore(t,b,a);

break;

case 3:

PrintScore(t,b,a);

break;

case 4:

scanf("%d",&k);

FindScore(k,t,b,a);

break;

}

}

return 0;

}

1. 测试。
   1. 测试数据如表5-3所示。

表5-3 程序设计题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 输入0. | 结束程序。 |
| 用例2 | 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 |

* 1. 运行结果:



图5-9 程序设计题2用例1的运行结果截图

对应测试用例2的运行结果如下图所示。

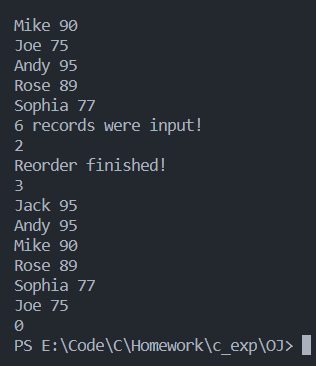


图5-10 程序设计题2用例2的运行结果截图

1. 求解N皇后问题，即在N×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. 实验思路如下：使用回溯法

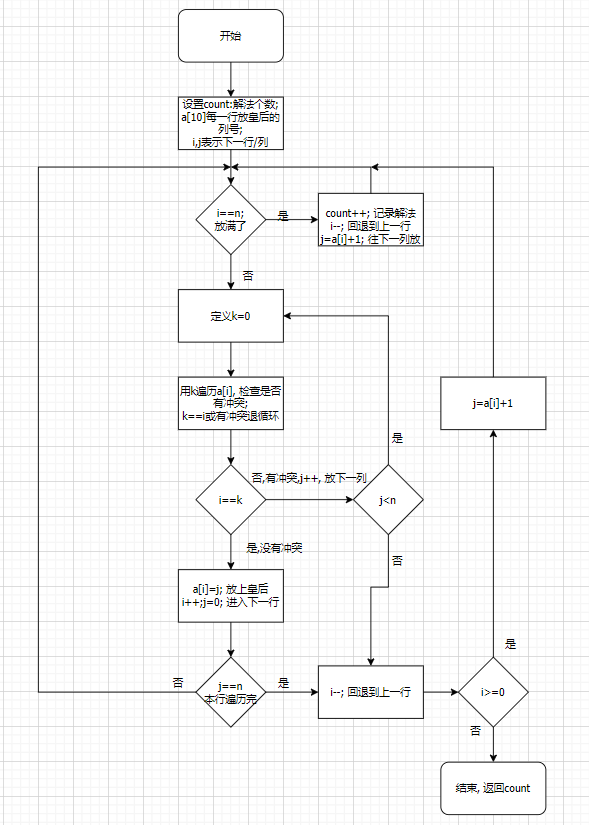


图5-11 算法设计思路

1. 源程序如下：

#include <stdio.h>

#include <math.h>

int NQueen(int n);

int main(void)

{

int n;

scanf("%d", &n);

if (n < 1 || n > 10){

printf("无解");

return 0;

}

int count = NQueen(n);

if (count == 0)

printf("无解");

else

printf("%d", count);

return 0;

}

int NQueen(int n){

int count = 0;

int a[10] = {0};

int i = 0, j = 0;

while (i >= 0){

if (i == n){

count++;

i--;

j = a[i] + 1;

continue;

}

for (; j < n; j++){

int k = 0;

for (; k < i; k++){

if (a[k] == j || abs(a[k] - j) == abs(k - i))

break;

}

if (k == i){

a[i] = j;

i++;

j = 0;

break;

}

}

if (j == n){

i--;

if (i >= 0)

j = a[i] + 1;

}

}

return count;

}

1. 测试。
   1. 测试数据如表5-4所示。

表5-4 程序设计题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程 序 输 入 | 理论结果 | 运行结果 |
| size |
| 用例1 | 1 | 1 | 1 |
| 用例2 | 2 | 无解！ | 无解！ |
| 用例3 | 4 | 2 | 2 |
| 用例4 | 5 | 10 | 10 |
| 用例5 | 7 | 40 | 40 |
| 用例6 | 10 | 724 | 724 |

* 1. 对应测试结果:

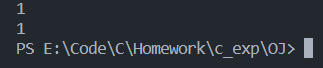


图5-12 程序设计题3用例1的运行结果截图

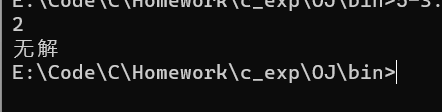


图5-13 程序设计题3用例2的运行结果截图

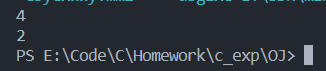


图5-14 程序设计题3用例3的运行结果截图

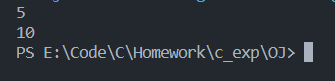


图5-15 程序设计题3用例4的运行结果截图

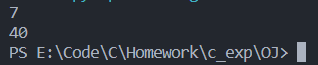


图5-16 程序设计题3用例5的运行结果截图

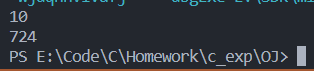


图5-17 程序设计题3用例6的运行结果截图

1. 本关任务：实现一个铁路购票系统的简单作为分配算法，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. 解答思路如下:

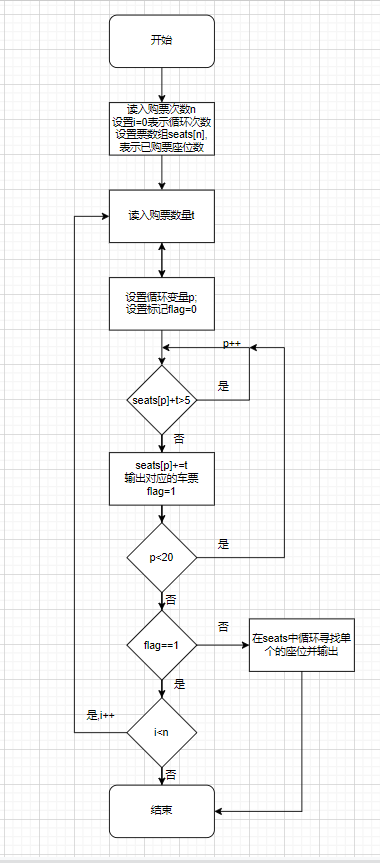


图5-18 算法设计思路

1. 源程序如下：

#include<stdio.h>

int main(){

int seats[20] = {0};

char SeatName[] = {'A','B','C','D','F'};

int n;

scanf("%d", &n);

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

{

int t = 0;

scanf("%d", &t);

int flag = 0; //是否找到了相邻座位

for(int p=0;p<20;p++){

if(seats[p]+t<=5){

seats[p]+=t;

flag = 1;

for(int j=0;j<t;j++){

printf("%d%c", p+1, SeatName[j+seats[p]-t]);

if(j!=t-1)

printf(" ");

}

printf("\n");

break;

}

}

if(flag==0){

for(int p=0;p<20;p++){

if(seats[p]<5){

while(seats[p]<5){

seats[p]++;

printf("%d%c ", p+1, SeatName[seats[p]-1]);

t--;

if(t==0)

return 0;

}

}

}

}

}

return 0;

}

1. 测试

表5-4 程序设计题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 4  5 4 4 2 | 1A 1B 1C 1D 1F  2A 2B 2C 2D  3A 3B 3C 3D  4A 4B |
| 用例2 | 21  4 5 3 4 5 5 5 5 5 5 5 4 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 |

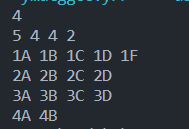


图5-19 测试用例1

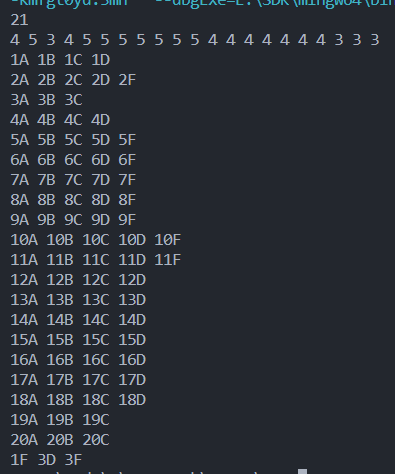


图5-20 测试用例2

1. 本关任务：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≤m1≤n1<m2≤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. 实验思路如下：
2. 判断n1-m1与n2-m2的大小, 选择比较短的一段移动到比较长的一段
3. 再移动较长段剩下未移动的部分
4. 源程序如下:

#include <stdio.h>

int main(){

int a[100],n,m1,n1,m2,n2;

scanf("%d",&n);

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

scanf("%d",&a[i]);

}

scanf("%d%d%d%d",&m1,&n1,&m2,&n2);

// 交换m1~n1和m2~n2的数据

int temp;

// 交换较短的一段

if(n1-m1<n2-m2){

for(int i=0;i<n1-m1+1;i++){

temp=a[m1+i];

a[m1+i]=a[m2+i];

a[m2+i]=temp;

}

}else{

for(int i=0;i<n2-m2+1;i++){

temp=a[m1+i];

a[m1+i]=a[m2+i];

a[m2+i]=temp;

}

}

// 移动剩下的数据

if(n1-m1<n2-m2){

for(int i=n1-m1+1;i<n2-m2+1;i++){

temp=a[m2+i];

for(int j=m2+i;j>m1+i;j--){

a[j]=a[j-1];

}

a[m1+i]=temp;

}

}else{

for(int i=n2-m2+1;i<n1-m1+1;i++){

temp=a[m1+i];

for(int j=m1+i;j<m2+i;j++){

a[j]=a[j+1];

}

a[m2+i-1]=temp;

}

}

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

printf("%d ",a[i]);

}

return 0;

}

1. 测试

表5-4 程序设计题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 7  1 2 3 4 5 6 7  1 2 4 6 | 1 5 6 7 4 2 3 |
| 用例2 | 1 2 3 4 5 6 7  1 1 4 5 | 1 5 6 3 4 2 7 |

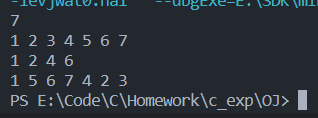


图5-21 测试用例1

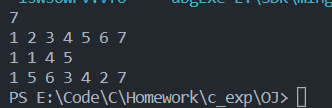


图5-22 测试用例2

## 5.3 实验小结

1. 首先对字符串处理函数的原理以及数组有较深体会, 要善于利用debug检查数组越界问题。

2. 第三题N皇后问题, 一般都会使用回溯法+递归解决N皇后问题, 这次尝试了使用非递归模式, 而是用循环解决问题, 富有挑战性

**实验七 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 实验内容及要求

**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,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 | ‘x’ | ‘x’ |
| 4 | \*(++p)->t | ‘x’ | ‘x’ |
| 5 | \*++p->t | ‘V’ | ‘V’ |
| 6 | ++\*p->t | ‘V’ | ‘V’ |

**解答：**

各表达式输出结果如下图所示：

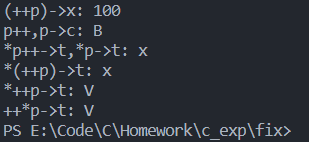


图7-1 程序验证题的运行结果

**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：

1 #include "stdio.h"

2 #include "stdlib.h"

3 struct s\_list{

4 int data; /\* 数据域 \*/

5 struct s\_list \*next; /\* 指针域 \*/

6 } ;

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

8 void main(void)

9 {

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

11 int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

12 create\_list(head,s); /\* 创建新链表 \*/

13 p=head; /\*遍历指针p指向链头 \*/

14 while(p){

15 printf("%d\t",p->data); /\* 输出数据域的值 \*/

16 p=p->next; /\*遍历指针p指向下一结点 \*/

17 }

18 printf("\n");

19 }

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

21 {

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

23 if(p[0]==0) /\* 相当于\*p==0 \*/

24 ;

25 else { /\* loc\_head指向动态分配的第一个结点 \*/

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

27 loc\_head->data=\*p++; /\* 对数据域赋值 \*/

28 tail=loc\_head; /\* tail指向第一个结点 \*/

29 while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

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

31 tail=tail->next; /\* tail指向新创建的结点 \*/

32 tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

33 }

34 tail->next=NULL; /\* 对指针域赋NULL值 \*/

35 }

36 headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

37 }

**解答：**

1. 错误修改：
2. 第7行，第20行，将参数中的\*headp改为双重指针\*\*headp, 才能达到修改headp的效果
3. 第12行，将create\_list(head,s);改为create\_list(&head,s); , 因为head是双重指针
4. 第36行，将headp改为\*headp。
5. 修改后的源程序清单如下：

#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);

int 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");

return 0;

}

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 \*/

}

1. 修改后的运行结果如图7-7所示。

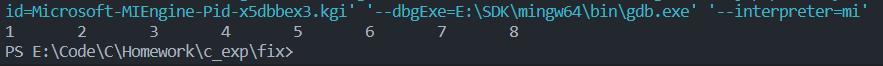


图7-2 程序改错题的运行结果

1. 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. 修改思路：令一个变量tmp存储新加入的节点, 令tmp->next=head, 再令head=tmp, 这样得到的就是后进先出的栈
2. 修改后的源程序清单如下：

#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);

int 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 \*/

}

p = head;

printf("\n");

return 0;

}

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

{

struct s\_list \*loc\_tail = NULL, \*head;

if (p[0] == 0); /\* 相当于\*p==0 \*/

else

{ /\* loc\_tail指向动态分配的最后一个结点 \*/

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

loc\_tail->data = \*p++; /\* 对数据域赋值 \*/

head = loc\_tail;

while (\*p)

{

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

tmp->next = head;

tmp->data = \*p++;

head = tmp;

}

loc\_tail->next = NULL; /\* 对指针域赋NULL值 \*/

}

\*headp = head; /\* 使头指针headp指向新创建的链表 \*/

}

1. 修改后的运行结果如图7-8所示：

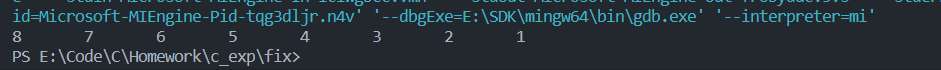


图7-3 程序修改题的运行结果

**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：
2. 定义结构体Student，包含学号，成绩等数据，并定义全局变量head表示学生链表。
3. 定义AddStudent函数，读入输入学生数n, 并申请空间创建学生链表。
4. 定义PrintAllStudent函数，利用while循环打印所有学生的数据。
5. 定义ModifyStudent函数，用于修改数据, 首先读入修改学生的学号, 再读入修改的项目, 用while遍历寻找并修改对应数据, 完成后打印修改后数据
6. 定义PrintAllAverage函数，利用while循环统计每个学生的平均成绩。
7. 定义PrintAllTotal函数，利用while循环输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。
8. 主函数中将6个函数利用switch串接，当输入为0时，释放内存, 执行return 0结束
9. 源程序代码清单如下：

#include <stdio.h>

#include "stdlib.h"

typedef struct Student {

int num;

char name[20];

int english, math, physics, c;

struct Student \*next;

}Student;

Student \*head = NULL;

void AddStudent();

void PrintAllStudent();

void ModifyStudent();

void PrintAllAverage();

void PrintAllTotal();

int main(){

int op; // 操作码

while(1){

scanf("%d", &op);

switch(op){

case 0: return 0;

case 1: AddStudent(); break;

case 2: PrintAllStudent(); break;

case 3: ModifyStudent(); break;

case 4: PrintAllAverage(); break;

case 5: PrintAllTotal(); break;

}

}

//释放内存

    Student \*p = head;

    while(p){

        Student \*q = p;

        p = p->next;

        free(q);

    }

return 0;

}

void AddStudent(){

int n = 0;

scanf("%d", &n);

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

Student \*s = (Student\*)malloc(sizeof(Student));

scanf("%d %s %d %d %d %d", &s->num, s->name, &s->english, &s->math, &s->physics, &s->c);

// 插到尾部

if (head == NULL) {

head = s;

s->next = NULL;

} else {

Student \*p = head;

while (p->next) {

p = p->next;

}

p->next = s;

s->next = NULL;

}

}

printf("完成了%d位同学的成绩输入\n", n);

}

void PrintAllStudent(){

Student \*p = head;

while(p){

printf("%d %s %d %d %d %d", p->num, p->name, p->english, p->math, p->physics, p->c);

}

}

void ModifyStudent(){

int num = 0;

scanf("%d", &num);

int op = 0; // 0: name, 1: english, 2: math, 3: physics, 4: c

scanf("%d", &op);

Student \*p = head;

while(p){

if (p->num == num) {

switch(op){

case 0: scanf("%s", p->name); break;

case 1: scanf("%d", &p->english); break;

case 2: scanf("%d", &p->math); break;

case 3: scanf("%d", &p->physics); break;

case 4: scanf("%d", &p->c); break;

}

break;

}

p = p->next;

}

//打印修改后的信息

printf("%d %s %d %d %d %d", p->num, p->name, p->english, p->math, p->physics, p->c);

}

void PrintAllAverage(){

Student \*p = head;

while(p){

printf("%d %s %.2f\n", p->num, p->name, (p->english + p->math + p->physics + p->c) / 4.0);

p = p->next;

}

}

void PrintAllTotal(){

Student \*p = head;

while(p){

printf("%d %s %d %.2f\n",

p->num, p->name,

p->english + p->math + p->physics + p->c,

(p->english + p->math + p->physics + p->c) / 4.0);

p = p->next;

}

}

1. 测试：
2. 测试数据如表7-1所示：

表7-1 程序设计题（1）测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89 | 完成了5位同学的成绩输入。 | 完成了5位同学的成绩输入。 |
| 用例2 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89  4 | 完成了5位同学的成绩输入。  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joe 82.00  2021004 Andy 94.75  2021005 Rose 88.50 | 完成了5位同学的成绩输入。  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joe 82.00  2021004 Andy 94.75  2021005 Rose 88.50 |
| 用例3 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89  5  0 | 完成了5位同学的成绩输入。  2021001 Jack 364 91.00  2021002 Mike 320 80.00  2021003 Joe 328 82.00  2021004 Andy 379 94.00  2021005 Rose 354 88.00  结束 | 完成了5位同学的成绩输入。  2021001 Jack 364 91.00  2021002 Mike 320 80.00  2021003 Joe 328 82.00  2021004 Andy 379 94.00  2021005 Rose 354 88.00  结束 |

b) 运行结果:

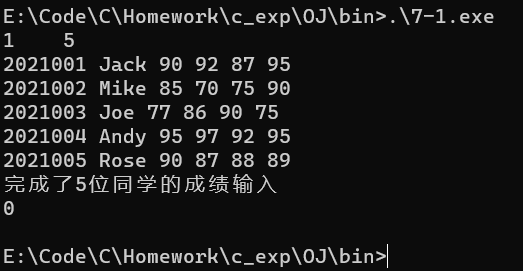


图7-4 程序设计题用例（1）的运行结果

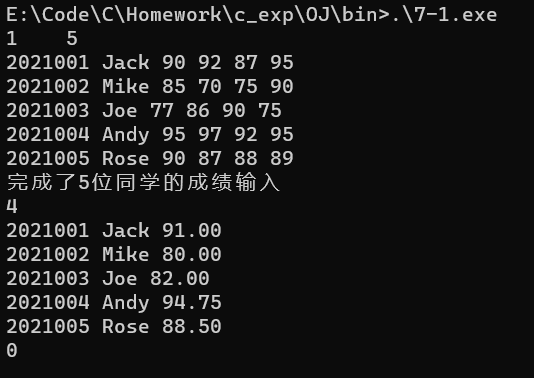


图7-5 程序设计题（1）用例2的运行结果

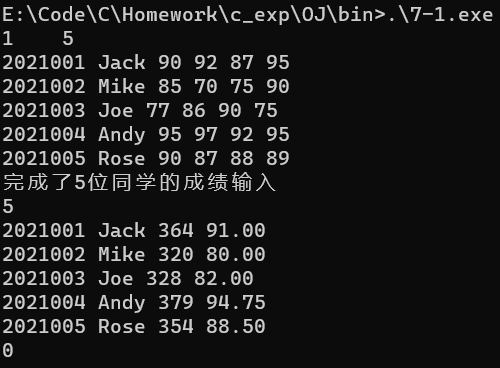


图7-6 程序设计题（1）用例3的运行结果

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）。

1. 解题思路:
2. 定义SortStudent函数, 读入升序以及排序方法要求并调用对应函数
3. 定义SelectSort函数, 即选择排序, 以升序为例:
   1. 定义数组p, 存储学生链表的数据, 定义变量i=0, 表示排序完的个数
   2. 遍历p数组, 找到总成绩最低的学生, 与第i位学生交换
   3. 再从剩下的n-i个学生中找最小的, 放到i处
   4. 重复b-c步, 直到i=n, 排序完成
   5. 根据p数组重新构建链表
4. 定义BubbleSort函数, 即冒泡排序, 以升序为例:
   1. 定义数组p, 存储学生链表的数据.
   2. 遍历p数组:
   3. 如果前一个学生比后一个学生总成绩高, 则交换二者, 否则不交换
   4. 重复b-c步直到没有学生可以交换, 排序完成.
   5. 根据p数组重新构建链表
5. 源程序清单如下:

#include <stdio.h>

#include "stdlib.h"

typedef struct Student {

    int num;

    char name[20];

    int english, math, physics, c;

    struct Student \*next;

}Student;

Student \*head = NULL;

int StudentCount = 0;

void AddStudent();

void PrintAllStudent();

void ModifyStudent();

void PrintAllAverage();

void PrintAllTotal();

void SortStudent();

void BubbleSort(int op);

void SelectSort(int op);

int main(){

    int op; // 操作码

    while(1){

        scanf("%d", &op);

        switch(op){

            case 0: return 0;

            case 1: AddStudent(); break;

            case 2: PrintAllStudent(); break;

            case 3: ModifyStudent(); break;

            case 4: PrintAllAverage(); break;

            case 5: PrintAllTotal(); break;

            case 6: SortStudent(); break;

        }

    }

    //释放内存

    Student \*p = head;

    while(p){

        Student \*q = p;

        p = p->next;

        free(q);

    }

    return 0;

}

void AddStudent(){

    int n = 0;

    scanf("%d", &n);

    StudentCount += n;

    for (int i = 0; i < n; i++) {

        Student \*s = (Student\*)malloc(sizeof(Student));

        scanf("%d %s %d %d %d %d", &s->num, s->name, &s->english, &s->math, &s->physics, &s->c);

        // 插到尾部

        if (head == NULL) {

            head = s;

            s->next = NULL;

        } else {

            Student \*p = head;

            while (p->next) {

                p = p->next;

            }

            p->next = s;

            s->next = NULL;

        }

    }

    printf("完成了%d位同学的成绩输入\n", n);

}

void PrintAllStudent(){

    Student \*p = head;

    while(p){

        printf("%d %s %d %d %d %d", p->num, p->name, p->english, p->math, p->physics, p->c);

    }

}

void ModifyStudent(){

    int num = 0;

    scanf("%d", &num);

    int op = 0; // 0: name, 1: english, 2: math, 3: physics, 4: c

    scanf("%d", &op);

    Student \*p = head;

    while(p){

        if (p->num == num) {

            switch(op){

                case 0: scanf("%s", p->name); break;

                case 1: scanf("%d", &p->english); break;

                case 2: scanf("%d", &p->math); break;

                case 3: scanf("%d", &p->physics); break;

                case 4: scanf("%d", &p->c); break;

            }

            break;

        }

        p = p->next;

    }

    //打印修改后的信息

    printf("%d %s %d %d %d %d", p->num, p->name, p->english, p->math, p->physics, p->c);

}

void PrintAllAverage(){

    Student \*p = head;

    while(p){

        printf("%d %s %.2f\n", p->num, p->name, (p->english + p->math + p->physics + p->c) / 4.0);

        p = p->next;

    }

}

void PrintAllTotal(){

    Student \*p = head;

    while(p){

        printf("%d %s %d %.2f\n",

                p->num, p->name,

                p->english + p->math + p->physics + p->c,

                (p->english + p->math + p->physics + p->c) / 4.0);

        p = p->next;

    }

}

void SortStudent(){

    int op = 0; // 0: 升序, 1: 降序

    scanf("%d", &op);

    int method = 0; // 0: 选择法, 1: 冒泡法

    scanf("%d", &method);

    if(method == 0){

        SelectSort(op);

    } else {

        BubbleSort(op);

    }

    PrintAllAverage();

}

void BubbleSort(int op){

    Student p[StudentCount]; // 临时数组

    Student \*q = head;

    for (int i = 0; i < StudentCount; i++) {

        p[i] = \*q;

        q = q->next;

    }

    for (int i = 0; i < StudentCount - 1; i++) {

        for (int j = 0; j < StudentCount - 1 - i; j++) {

            if (op == 0) {

                if (p[j].english + p[j].math + p[j].physics + p[j].c > p[j + 1].english + p[j + 1].math + p[j + 1].physics + p[j + 1].c) {

                    Student temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            } else {

                if (p[j].english + p[j].math + p[j].physics + p[j].c < p[j + 1].english + p[j + 1].math + p[j + 1].physics + p[j + 1].c) {

                    Student temp = p[j];

                    p[j] = p[j + 1];

                    p[j + 1] = temp;

                }

            }

        }

    }

    // 重新构建链表

    head = NULL;

    for (int i = 0; i < StudentCount; i++) {

        Student \*s = (Student\*)malloc(sizeof(Student));

        \*s = p[i];

        // 插到尾部

        if (head == NULL) {

            head = s;

            s->next = NULL;

        } else {

            Student \*p = head;

            while (p->next) {

                p = p->next;

            }

            p->next = s;

            s->next = NULL;

        }

    }

}

void SelectSort(int op){

    Student p[StudentCount]; // 临时数组

    Student \*q = head;

    for (int i = 0; i < StudentCount; i++) {

        p[i] = \*q;

        q = q->next;

    }

    for (int i = 0; i < StudentCount - 1; i++) {

        int min = i;

        for (int j = i + 1; j < StudentCount; j++) {

            if (op == 0) {

                if (p[j].english + p[j].math + p[j].physics + p[j].c < p[min].english + p[min].math + p[min].physics + p[min].c) {

                    min = j;

                }

            } else {

                if (p[j].english + p[j].math + p[j].physics + p[j].c > p[min].english + p[min].math + p[min].physics + p[min].c) {

                    min = j;

                }

            }

        }

        if (min != i) {

            Student temp = p[i];

            p[i] = p[min];

            p[min] = temp;

        }

    }

    // 重新构建链表

    head = NULL;

    for (int i = 0; i < StudentCount; i++) {

        Student \*s = (Student\*)malloc(sizeof(Student));

        \*s = p[i];

        // 插到尾部

        if (head == NULL) {

            head = s;

            s->next = NULL;

        } else {

            Student \*p = head;

            while (p->next) {

                p = p->next;

            }

            p->next = s;

            s->next = NULL;

        }

    }

}

1. 测试:
2. 表7-2 程序设计题2测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 1 5  2021001 Jack 90 92 87 95  2021002 Mike 85 70 75 90  2021003 Joe 77 86 90 75  2021004 Andy 95 97 92 95  2021005 Rose 90 87 88 89  6 0 1  0 | 完成了5位同学的成绩输入。  2021002 Mike 80.00  2021003 Joe 82.00  2021005 Rose 88.50  2021001 Jack 91.00  2021004 Andy 94.75 | 完成了5位同学的成绩输入。  2021002 Mike 80.00  2021003 Joe 82.00  2021005 Rose 88.50  2021001 Jack 91.00  2021004 Andy 94.75 |
| 用例2 | 1 5  2021001 Jack 90 92 87 95  2021002 Mike 85 70 75 90  2021003 Joe 77 86 90 75  2021004 Andy 95 97 92 95  2021005 Rose 90 87 88 89  6 1 0  0 | 完成了5位同学的成绩输入。  2021004 Andy 94.75  2021001 Jack 91.00  2021005 Rose 88.50  2021003 Joe 82.00  2021002 Mike 80.00 | 完成了5位同学的成绩输入。  2021004 Andy 94.75  2021001 Jack 91.00  2021005 Rose 88.50  2021003 Joe 82.00  2021002 Mike 80.00 |

1. 结果如图:

b) 运行结果:

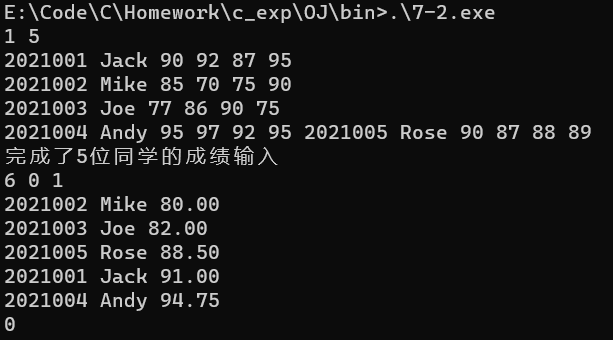


图7-7 程序设计题用例（1）的运行结果

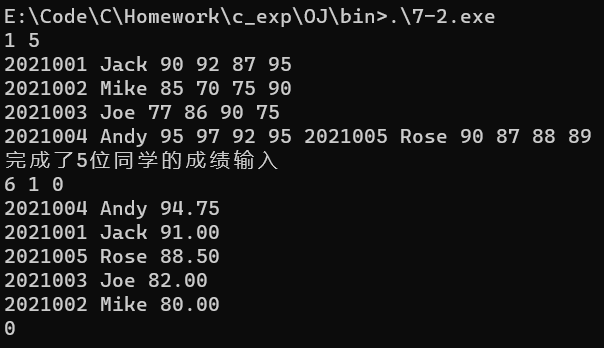


图7-8 程序设计题（1）用例2的运行结果

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）。

1. 解题思路:
2. 思路与上题一致, 仅修改数组p为存储指针域的数组, 其余一致
3. 源程序清单如下:

#include <stdio.h>

#include "stdlib.h"

typedef struct Student {

    int num;

    char name[20];

    int english, math, physics, c;

    struct Student \*next;

}Student;

Student \*head = NULL;

int StudentCount = 0;

void AddStudent();

void PrintAllStudent();

void ModifyStudent();

void PrintAllAverage();

void PrintAllTotal();

void SortStudent();

void BubbleSort(int op);

void SelectSort(int op);

int main(){

    int op; // 操作码

    while(1){

        scanf("%d", &op);

        switch(op){

            case 0: return 0;

            case 1: AddStudent(); break;

            case 2: PrintAllStudent(); break;

            case 3: ModifyStudent(); break;

            case 4: PrintAllAverage(); break;

            case 5: PrintAllTotal(); break;

            case 6: SortStudent(); break;

        }

    }

    //释放内存

    Student \*p = head;

    while(p){

        Student \*q = p;

        p = p->next;

        free(q);

    }

    return 0;

}

void AddStudent(){

    int n = 0;

    scanf("%d", &n);

    StudentCount += n;

    for (int i = 0; i < n; i++) {

        Student \*s = (Student\*)malloc(sizeof(Student));

        scanf("%d %s %d %d %d %d", &s->num, s->name, &s->english, &s->math, &s->physics, &s->c);

        // 插到尾部

        if (head == NULL) {

            head = s;

            s->next = NULL;

        } else {

            Student \*p = head;

            while (p->next) {

                p = p->next;

            }

            p->next = s;

            s->next = NULL;

        }

    }

    printf("完成了%d位同学的成绩输入\n", n);

}

void PrintAllStudent(){

    Student \*p = head;

    while(p){

        printf("%d %s %d %d %d %d", p->num, p->name, p->english, p->math, p->physics, p->c);

    }

}

void ModifyStudent(){

    int num = 0;

    scanf("%d", &num);

    int op = 0; // 0: name, 1: english, 2: math, 3: physics, 4: c

    scanf("%d", &op);

    Student \*p = head;

    while(p){

        if (p->num == num) {

            switch(op){

                case 0: scanf("%s", p->name); break;

                case 1: scanf("%d", &p->english); break;

                case 2: scanf("%d", &p->math); break;

                case 3: scanf("%d", &p->physics); break;

                case 4: scanf("%d", &p->c); break;

            }

            break;

        }

        p = p->next;

    }

    //打印修改后的信息

    printf("%d %s %d %d %d %d", p->num, p->name, p->english, p->math, p->physics, p->c);

}

void PrintAllAverage(){

    Student \*p = head;

    while(p){

        printf("%d %s %.2f\n", p->num, p->name, (p->english + p->math + p->physics + p->c) / 4.0);

        p = p->next;

    }

}

void PrintAllTotal(){

    Student \*p = head;

    while(p){

        printf("%d %s %d %.2f\n",

                p->num, p->name,

                p->english + p->math + p->physics + p->c,

                (p->english + p->math + p->physics + p->c) / 4.0);

        p = p->next;

    }

}

void SortStudent(){

    int op = 0; // 0: 升序, 1: 降序

    scanf("%d", &op);

    int method = 0; // 0: 选择法, 1: 冒泡法

    scanf("%d", &method);

    if(method == 0){

        SelectSort(op);

    } else {

        BubbleSort(op);

    }

    PrintAllAverage();

}

void BubbleSort(int op){

    Student \*p[StudentCount]; // 临时数组

    Student \*q = head;

    for (int i = 0; i < StudentCount; i++) {

        p[i] = q;

        q = q->next;

    }

    for (int i = 0; i < StudentCount - 1; i++) {

        for (int j = 0; j < StudentCount - 1 - i; j++) {

            if (op == 0) {

                if ((p[j]->english + p[j]->math + p[j]->physics + p[j]->c) > (p[j+1]->english + p[j+1]->math + p[j+1]->physics + p[j+1]->c)) {

                    Student \*t = p[j];

                    p[j] = p[j+1];

                    p[j+1] = t;

                }

            } else {

                if ((p[j]->english + p[j]->math + p[j]->physics + p[j]->c) < (p[j+1]->english + p[j+1]->math + p[j+1]->physics + p[j+1]->c)) {

                    Student \*t = p[j];

                    p[j] = p[j+1];

                    p[j+1] = t;

                }

            }

        }

    }

    // 重新连接

    for (int i = 0; i < StudentCount - 1; i++) {

        p[i]->next = p[i+1];

    }

    p[StudentCount-1]->next = NULL;

    head = p[0];

}

void SelectSort(int op){

    Student \*p[StudentCount]; // 临时数组

    Student \*q = head;

    for (int i = 0; i < StudentCount; i++) {

        p[i] = q;

        q = q->next;

    }

    for (int i = 0; i < StudentCount - 1; i++) {

        int min = i;

        for (int j = i + 1; j < StudentCount; j++) {

            if (op == 0) {

                if ((p[j]->english + p[j]->math + p[j]->physics + p[j]->c) < (p[min]->english + p[min]->math + p[min]->physics + p[min]->c)) {

                    min = j;

                }

            } else {

                if ((p[j]->english + p[j]->math + p[j]->physics + p[j]->c) > (p[min]->english + p[min]->math + p[min]->physics + p[min]->c)) {

                    min = j;

                }

            }

        }

        if (min != i) {

            Student \*t = p[i];

            p[i] = p[min];

            p[min] = t;

        }

    }

    // 重新连接

    for (int i = 0; i < StudentCount - 1; i++) {

        p[i]->next = p[i+1];

    }

    p[StudentCount-1]->next = NULL;

    head = p[0];

}

1. 测试:
2. 表7-3 程序设计题3测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 1 5  2021001 Jack 90 92 87 95  2021002 Mike 85 70 75 90  2021003 Joe 77 86 90 75  2021004 Andy 95 97 92 95  2021005 Rose 90 87 88 89  6 1 1  0 | 完成了5位同学的成绩输入。  2021004 Andy 94.75  2021001 Jack 91.00  2021005 Rose 88.50  2021003 Joe 82.00  2021002 Mike 80.00 | 完成了5位同学的成绩输入。  2021004 Andy 94.75  2021001 Jack 91.00  2021005 Rose 88.50  2021003 Joe 82.00  2021002 Mike 80.00 |
| 用例2 | 1 5  2021001 Jack 90 92 87 95  2021002 Mike 85 70 75 90  2021003 Joe 77 86 90 75  2021004 Andy 95 97 92 95  2021005 Rose 90 87 88 89  6 0 0  0 | 完成了5位同学的成绩输入。  2021002 Mike 80.00  2021003 Joe 82.00  2021005 Rose 88.50  2021001 Jack 91.00  2021004 Andy 94.75 | 完成了5位同学的成绩输入。  2021002 Mike 80.00  2021003 Joe 82.00  2021005 Rose 88.50  2021001 Jack 91.00  2021004 Andy 94.75 |

1. 结果如图:

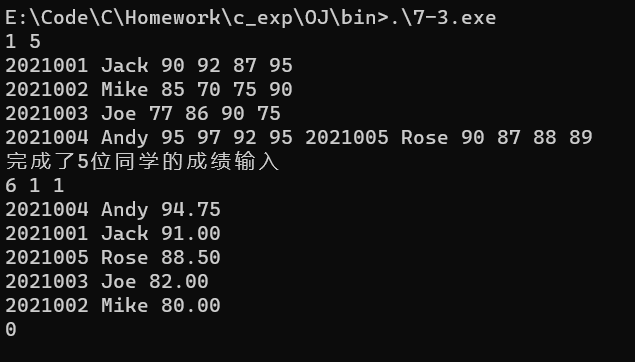


图7-11 程序设计题用例（1）的运行结果

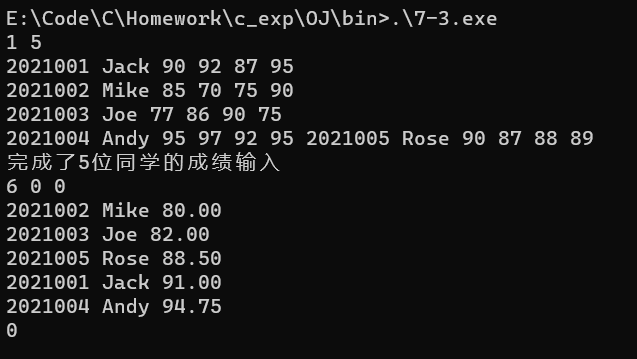


图7-10 程序设计题（1）用例2的运行结果

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。

1. 解题思路:
2. 定义一个单向链表存储人, NumOfPeople变量存储剩余人数.
3. 在while循环内:
4. 定义p, q, 分别指向上一个人以及当前人
5. 利用for循环报数m次
6. 得到出局的人q
7. 令此人出局, 并释放空间, 输出q的编号
8. 重复c-f步, 直到q->next!=q
9. 源程序清单如下:

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int num;

struct Node \*next;

}Node;

Node \*head = NULL;

int NumOfPeople = 0;

void AddNode(int n);

void Josephus(int m);

int main(){

int n, m;

scanf("%d %d", &n, &m);

NumOfPeople = n;

AddNode(n);

Josephus(m);

return 0;

}

void AddNode(int n){

// 虚构头结点, 单向循环链表

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

head->num = 0;

head->next = NULL;

Node \*p = head;

for (int i = 1; i <= n; i++) {

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

s->num = i;

s->next = NULL;

p->next = s;

p = s;

}

p->next = head->next;

}

void Josephus(int m){

//快慢指针

Node \*p = head;

Node \*q = head->next;

while(q->next != q){

for (int i = 1; i < m; i++) {

p = p->next;

q = q->next;

}

printf("%d", q->num);

if(NumOfPeople > 2){

printf(" ");

}

p->next = q->next;

free(q);

q = p->next;

NumOfPeople--;

}

printf("\n%d", p->num);

}

1. 测试:
2. 表7-4 程序设计题4测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 6 5 | 5 4 6 2 3  1 | 5 4 6 2 3  1 |
| 用例2 | 13 12345678 | 7 13 4 1 12 9 11 10 5 8 6 3  2 | 7 13 4 1 12 9 11 10 5 8 6 3  2 |

1. 结果如图:

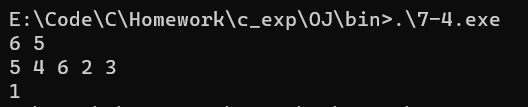


图7-11 程序设计题用例（1）的运行结果

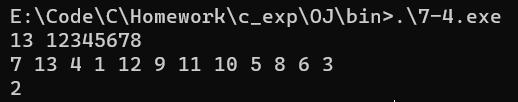


图7-10 程序设计题（1）用例2的运行结果

**三、实验小结**

本次实验的成绩管理系统, 一开始看到会觉得功能很复杂, 无从下手. 但利用自顶向下的设计方案, 采用模块化编程:先以链表为核心, 把各种功能看作是对链表的操作, 思路就变得十分清晰起来, 而且代码也比较美观.

另外, 在成绩排序时, 直接使用链表交换, 发现没有交换成功, 试了许多方案也无果而终. 这时突然想到能不能转化为数组, 这样就成为了经典的排序, 排序完之后再转换成链表, 尝试后成功了. 遇到问题时可以换一个角度, 看看能不能转化为自己会的方案.

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