信息科学与技术学院，21 级电子信息类专业

程 序 设 计 实 践 报 告

学　　号：_____210800229_____

姓　　名：_____杨佳宇_____

设计题目：

1._____运动会分数统计_____

2._____停车场管理系统_____

3._____　迷宫问题求解_____

指导教师：_____齐金鹏_____

成　　绩：_____

完成日期：_____2022.6.19_____

设计题目

题一： 运动会分数统计

任务：

参加运动会有 n 个学校，学校编号为 1……n。比赛分成 m 个男子项目和 w 个女子项目。项目编号为男子 1～m，女子 m+1～m+w。不同的项目取前五名或前三名积分；取前五名的积分分别为：7、5、3、2、1，前三名的积分分别为：5、3、2；哪些项目取前五名或前三名由学生自己设定。（m<=20, n<=20）

基本功能要求：

1)可以输入各个项目的前三名或前五名的成绩；

2)能统计各学校总分；

3)可以按学校编号、学校总分、男女团体总分排序输出.

4)可以按学校编号查询学校某个项目的情况；

5)可以按项目编号查询取得前三或前五名的学校。

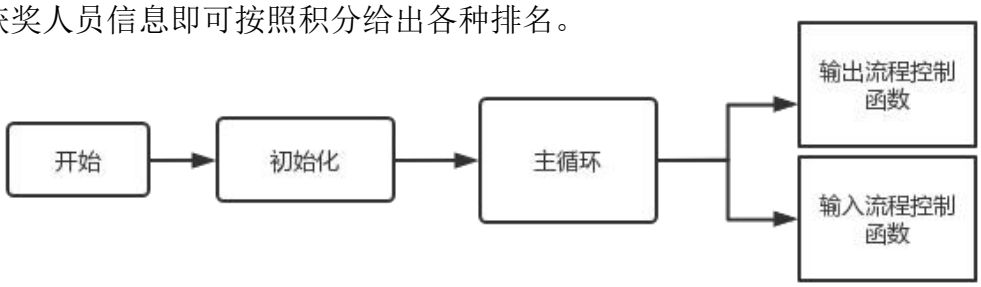
额外附加部分：

1.将比赛有关信息存入文件，无需每次运行都要输入相关信息，并且支持对已保存信息进行修改。

2.可任意指定赋分机制，同时也支持默认规则，也可以后续对规则进行修改。

需求分析：

首先需要构建顺序表储存相关信息，如参赛学校的名称，获得的分数，比赛项目，及其相关的赋分规则，以及参赛选手信息。然后需要输入比赛相关信息，如参数学校，比赛项目的相关信息，至此基础信息完备。在项目结束后输入相关获奖人员信息即可按照积分给出各种排名。



通过设立流程标志控制对应的界面进行控制，通过 switch 语句进行代码分片在不同的界面实现不同的操作逻辑。

概要设计：

```
  //学校信息
struct School
{
    char name[20];    //学校名称
    uint16_t score;   //学校总分
    uint16_t score_m; //学校男子总分
    uint16_t score_w; //学校女子总分
} School[20];
//学生信息
struct Student
{
    char name[20];   //学生姓名
    uint8_t sex;     //性别
    char school[20]; //所属学校
    float points;    //成绩
    uint8_t score;   //对学校的积分贡献
    uint8_t event;   //参与项目编号
} Student[40];
//项目信息
struct Event
{
    char name[20];     //项目名称
    uint8_t category;  //项目类别（男子/女子）
    uint8_t type;      //选取前几名积分
    uint8_t score[20]; //各名次贡献的积分
} Event[20];
void Input_proc(void);     //输入流程控制函数
void Output_proc(void);    //输出流程控制函数
void Sort(uint16_t arr[]); //冒泡排序函数
void init(void);           //初始化
void filewrite(void);     //退出时保存信息
```

详细设计：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
uint8_t ug_state = 0;
```

```c
int sys_state = 0; //主流程标志
uint8_t key = 0;   //菜单选择标志
int school_num, event_male_num, event_female_num;
int student_num; //基本数量信息
uint8_t i, m, n, k;
FILE *fp;
char name[20]; //临时变量
//定义默认积分规则
int count[20] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0};
uint8_t default0[20] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0};
uint8_t default3[20] = {5, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0};
uint8_t default5[20] = {7, 5, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0};
//定义性别枚举提高可读性
enum sex
{
    male,
    female
};
//学校信息
struct School
{
    char name[20];    //学校名称
    uint16_t score;   //学校总分
    uint16_t score_m; //学校男子总分
    uint16_t score_w; //学校女子总分
} School[20];
//学生信息
struct Student
{
    char name[20];   //学生姓名
    uint8_t sex;     //性别
    char school[20]; //所属学校
    float points;    //成绩
    uint8_t score;   //对学校的积分贡献
    uint8_t event;   //参与项目编号
} Student[40];
//项目信息
struct Event
{
    char name[20];     //项目名称
```

```
    uint8_t category;  //项目类别（男子/女子）
    uint8_t type;      //选取前几名积分
    uint8_t score[20]; //各名次贡献的积分
} Event[20];
void Input_proc(void);     //输入流程控制函数
void Output_proc(void);    //输出流程控制函数
void Sort(uint16_t arr[]); //冒泡排序函数
void init(void);           //初始化
void filewrite(void);      //退出时保存信息
int main(void)
{
    init();
    while (1)
    {
        system("cls");
        Output_proc();
        Input_proc();
        if (ug_state == 4)
        {
            filewrite();//保存信息
            break;//跳出主循环
        }
    }
    return 0;
}
void Output_proc(void)
{
    switch (ug_state)
    {
    case 0:
        printf("\t\t\t\t 运动会统计系统\n");
        printf("\t\t1.比赛信息输入\n");
        printf("\t\t2.比赛结果输入\n");
        printf("\t\t3.比赛结果查询\n");
        printf("\t\t4.退出\n");
        break;
    case 1:
        printf("\t\t\t\t 运动会统计系统\n");
        if (sys_state)
        {
            printf("\t\t1.修改积分规则\n");
            printf("\t\t2.重置比赛信息\n");
            printf("\t\t3.返回\n");
        }
```

```
        else
            printf("\t\t有多少所学校参赛:");
        break;
    case 2:
        printf("\t\t\t\t运动会统计系统\n");
        printf("\t\t学校:");
        for (i = 0; i < school_num; i++)
        {
            if (i % 5 == 0)
            {
                printf("\n");
                printf("\t\t  ");
            }

            printf(" %d.%s", (i + 1), School[i].name);
        }
        printf("\n");
        printf("\t\t项目:");
        for (i = 0; i < event_male_num + event_female_num; i++)
        {
            if (i % 5 == 0)
            {
                printf("\n");
                printf("\t\t  ");
            }
            printf(" %d.%s", (i + 1), Event[i].name);
        }
        printf("\n");
        printf("\t\t输入项目编号:");
        break;
    case 3:
        printf("\t\t\t\t运动会统计系统\n");
        printf("\t\t1.按学校编号排序\n");
        printf("\t\t2.按学校总分排序\n");
        printf("\t\t3.男子团体总分排序\n");
        printf("\t\t4.女子团体总分排序\n");
        printf("\t\t5.返回\n");
        break;
    }
}
void Input_proc(void)
{
    switch (ug_state)
    {
```

```
    case 0:
        scanf("%d", &key);
        ug_state = key;
        break;
    case 1:
        if (!sys_state)
        {
            scanf("%d", &school_num);
            while (getchar() != '\n')
                continue;
            for (i = 0; i < school_num; i++)
            {
                printf("\t\t 第%d 所学校是:", i + 1);
                gets(School[i].name);
            }
            printf("\t\t 男子项目个数是:");
            scanf("%d", &event_male_num);
            printf("\t\t 女子项目个数是:");
            scanf("%d", &event_female_num);
            printf("\t\t 按照男子项目在前，女子项目在后输入各项目名
称,并指定积分规则\n");
            printf("\t\t 默认可选前 3 名(5,3,2)或前 5 名积分
(7,5,3,2,1)，也可以任意指定\n");
            for (i = 0; i < (event_female_num + event_male_num);
i++)
            {
                while (getchar() != '\n')
                    continue;
                printf("\t\t 第%d 个项目的名称:", (i + 1));
                gets(Event[i].name);
                Event[i].category = (i < event_male_num) ? male :
female;
                printf("\t\t 选取前几名计算积分:");
                scanf("%d", &n);
                Event[i].type = n;
                if (n == 3 || n == 5)
                {
                    printf("\t\t 是否采用默认积分方式(y/n):");
                    while (getchar() != '\n')
                        continue;
                    scanf("%c", &m);
                    if (m == 'y' || m == 'Y')
                    {
                        if (n == 3)
```

```
                        memcpy(Event[i].score, default3, 20);
                    else
                        memcpy(Event[i].score, default5, 20);
                }
                else
                {
                    memcpy(Event[i].score, default3, 20);
                    printf("\t\t 请依次输入前%d 名对应积分(用空格隔开):", n);
                    for (k = 0; k < n; k++)
                    {
                        scanf("%d", &Event[i].score[k]);
                    }
                }
            }
            else
            {
                memcpy(Event[i].score, default0, 20);
                printf("\t\t 请依次输入前%d 名对应积分(用空格隔开):", n);
                for (k = 0; k < n; k++)
                {
                    scanf("%d", &Event[i].score[k]);
                }
            }
            sys_state = 1;
            key = 0;
            ug_state = key;
        }
    }
    else
    {
        scanf("%d", &key);
        if (key == 1)
        {
            printf("\t\t 要修改的项目编号是:");
            scanf("%d", &i);
            printf("\t\t 选取前几名计算积分:");
            scanf("%d", &n);
            Event[i].type = n;
            if (n == 3 || n == 5)
            {
                printf("\t\t 是否采用默认积分方式(y/n):");
                while (getchar() != '\n')
```

```
                    continue;
                scanf("%c", &m);
                if (m == 'y' || m == 'Y')
                {
                    if (n == 3)
                        memcpy(Event[i].score, default3, 20);
                    else
                        memcpy(Event[i].score, default5, 20);
                }
                else
                {
                    memcpy(Event[i].score, default3, 20);
                    printf("\t\t 请依次输入前%d 名对应积分(用空格隔开):", n);
                    for (k = 0; k < n; k++)
                    {
                        scanf("%d", &Event[i].score[k]);
                    }
                }
            }
            else
            {
                memcpy(Event[i].score, default0, 20);
                printf("\t\t 请依次输入前%d 名对应积分(用空格隔开):", n);
                for (k = 0; k < n; k++)
                {
                    scanf("%d", &Event[i].score[k]);
                }
            }
        }
        else if (key == 2)
        {
            printf("\t\t 请确认是否重置(y/n):");
            while (getchar() != '\n')
                continue;
            scanf("%c", &m);
            if (m == 'y' || m == 'Y')
            {
                sys_state = 0;
            }
        }
        else
            key = 0;
```

```
            key = 0;
            ug_state = key;
        }
        break;
    case 2:
        if (sys_state)
        {
            scanf("%d", &key);
            if (count[key - 1] == key)
            {
                printf("\t\t 已经记录数据，是否覆盖(y/n)");
                while (getchar() != '\n')
                    continue;
                scanf("%c", &m);
                if (m == 'y' || m == 'Y')
                {
                    count[key - 1] = key;
                }
                else
                {
                    key = 2;
                    ug_state = key;
                    break;
                }
            }
            count[key - 1] = key;
            printf("\t\t 请按照名次依次输入项目%d %s 的前%d 名运动员
的姓名，校名及成绩\n", key, Event[key - 1].name, Event[key -
1].type);
            for (i = 0; i < Event[key - 1].type; i++)
            {
                while (getchar() != '\n')
                    continue;
                printf("\t\t 第%d 名:\n", i + 1);
                printf("\t\t 姓名:");
                gets(Student[student_num++].name);
                printf("\t\t 校名:");
                gets(Student[student_num - 1].school);
                m = 0;
                for (k = 0; k < school_num; k++)
                {
                    if (strcmp(Student[student_num - 1].school,
School[k].name) == 0)
                    {
```

```
                        School[k].score += Event[key -
1].score[i];
                        if (Event[key - 1].category == male)
                        {
                            School[k].score_m += Event[key -
1].score[i];
                        }
                        else
                        {
                            School[k].score_w += Event[key -
1].score[i];
                        }
                        m = 1;
                        n = k;
                    }
                }
                while (m != 1)
                {
                    printf("\t\t请正确输入\n");
                    gets(Student[student_num - 1].school);
                    m = 0;
                    for (k = 0; k < school_num; k++)
                    {
                        if (strcmp(Student[student_num -
1].school, School[k].name) == 0)
                        {
                            School[k].score += Event[key -
1].score[i];
                            if (Event[key - 1].category == male)
                            {
                                School[k].score_m += Event[key -
1].score[i];
                            }
                            else
                            {
                                School[k].score_w += Event[key -
1].score[i];
                            }
                            m = 1;
                            n = k;
                        }
                    }
                }
                printf("\t\t成绩:");
```

```
                scanf("%f", &Student[student_num - 1].points);
                Student[student_num - 1].event = key;
                Student[student_num - 1].sex = Event[key -
1].category;
                Student[student_num - 1].score = Event[key -
1].score[i];
            }
        }
        else
        {
            printf("请先补全比赛信息");
        }
        key = 0;
        ug_state = key;
        break;
    case 3:
    {
        scanf("%d", &key);
        n = 1;
        uint16_t arr[20] = {0};
        if (key == 1)
        {
            i = 0;
            printf("\t\t\t按学校编号\n");
            printf("\t\t序号\t学校\t积分\n");
            for (; i < school_num; i++)
                printf("\t\t%d\t%s\t%d\n", n++, School[i].name,
School[i].score);
        }
        else
        {
            if (key == 2)
            {
                printf("\t\t\t按学校总分\n");
                for (i = 0; i < school_num; i++)
                    arr[i] = School[i].score;
            }
            else if (key == 3)
            {
                printf("\t\t\t按男子团体总分\n");
                for (i = 0; i < school_num; i++)
                    arr[i] = School[i].score_m;
            }
            else if (key == 4)
```

```
            {
                printf("\t\t\t按女子团体总分\n");
                for (i = 0; i < school_num; i++)
                    arr[i] = School[i].score_w;
            }
            else if (key == 5)
            {
                key = 0;
                ug_state = key;
                break;
            }
            Sort(arr);
            printf("\t\t排名\t学校\t积分\n");
            for (k = 0; k < school_num; k++)
            {
                for (m = 0; m < school_num; m++)
                    if (School[m].score == arr[k] && (n<=5))
                    {
                        printf("\t\t%d\t%s\t%d\n", n++,
School[m].name, School[m].score);
                    }
            }
        }
        printf("\t\t1.返回\n");
        scanf("%d", &key);
        key = 3;
        ug_state = key;
    }
    break;
    }
}
void init(void)
{
    fp = fopen("a.txt", "rb");
    fread(School, sizeof(struct School), 20, fp);
    fread(Student, sizeof(struct Student), 40, fp);
    fread(Event, sizeof(struct Event), 20, fp);
    fread(&school_num, sizeof(int), 1, fp);
    fread(&event_male_num, sizeof(int), 1, fp);
    fread(&event_female_num, sizeof(int), 1, fp);
    fread(&sys_state, sizeof(int), 1, fp);
    fread(&student_num, sizeof(int), 1, fp);
    fread(count, sizeof(int), 20, fp);
    fclose(fp);
```

```
    for (i = 0; i < 20; i++)
    {
        School[i].score = 0;
        School[i].score_m = 0;
        School[i].score_w = 0;
    }
}
void Sort(uint16_t arr[])
{
    int j, tem, s;
    for (s = 0; s < school_num - 1; s++)
    {
        int count = 0;
        for (j = 0; j < school_num - 1 - s; j++)
        {
            if (arr[j] < arr[j + 1])
            {
                tem = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = tem;
                count = 1;
            }
        }
        if (count == 0)
            break;
    }
}
void filewrite()
{
    fp = fopen("a.txt", "wb");
    fwrite(School, sizeof(struct School), 20, fp);
    fflush(fp);
    fwrite(Student, sizeof(struct Student), 40, fp);
    fflush(fp);
    fwrite(Event, sizeof(struct Event), 20, fp);
    fflush(fp);
    fwrite(&school_num, sizeof(int), 1, fp);
    fflush(fp);
    fwrite(&event_male_num, sizeof(int), 1, fp);
    fflush(fp);
    fwrite(&event_female_num, sizeof(int), 1, fp);
    fflush(fp);
    fwrite(&sys_state, sizeof(int), 1, fp);
    fflush(fp);
```

```
    fwrite(&student_num, sizeof(int), 1, fp);
    fflush(fp);
    fwrite(count, sizeof(int), 20, fp);
    fflush(fp);
    fclose(fp);
}
```

调试分析：

a,在连续使用 fwrite()函数进行文件写入时，需要使用 fflush()清除文件缓存区，不然会导致，之前的内容被大量重复的写入文件造成文件存储读取异常。

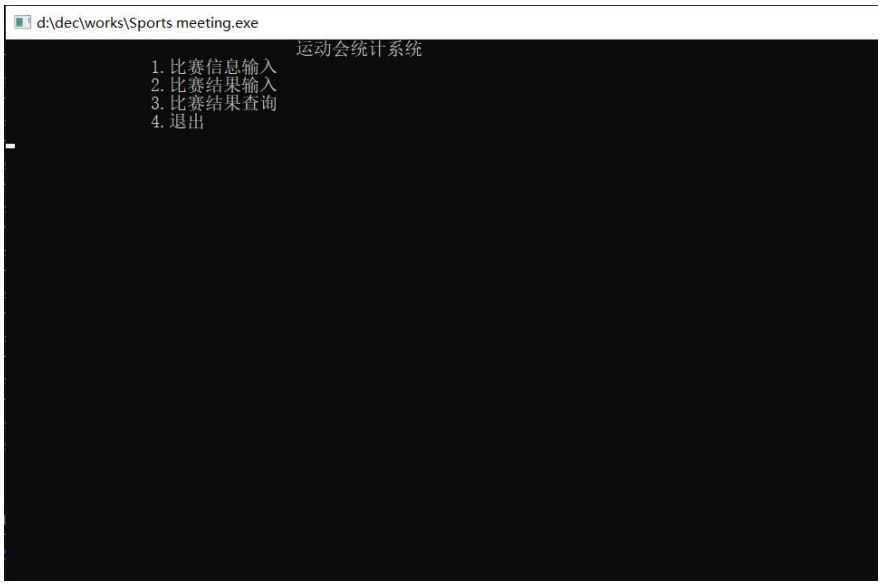
b.编程时追求控制窗口内的布局，这在实际开发中没有任何意义，应该注意避免。

c.结构体数组是一个很好的顺序存储结构，可以很好的进行内容管理，但需要注意结构体名除了在刚刚定义时，否则均不能作为左值直接复制，可采用 memcpy()函数予以解决这个问题。

用户手册：

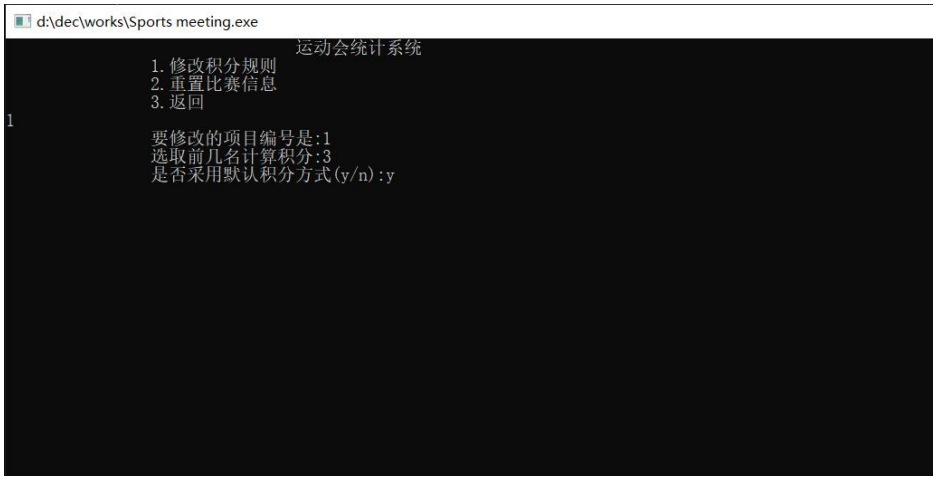
（1）演示程序的运行环境为 Windows10 系统，Visual Code Studio 中的 MinGW64 中运行。执行文件为：Sports meeting.exe

（2）进入演示程序后即显示 DOS 形式的界面：

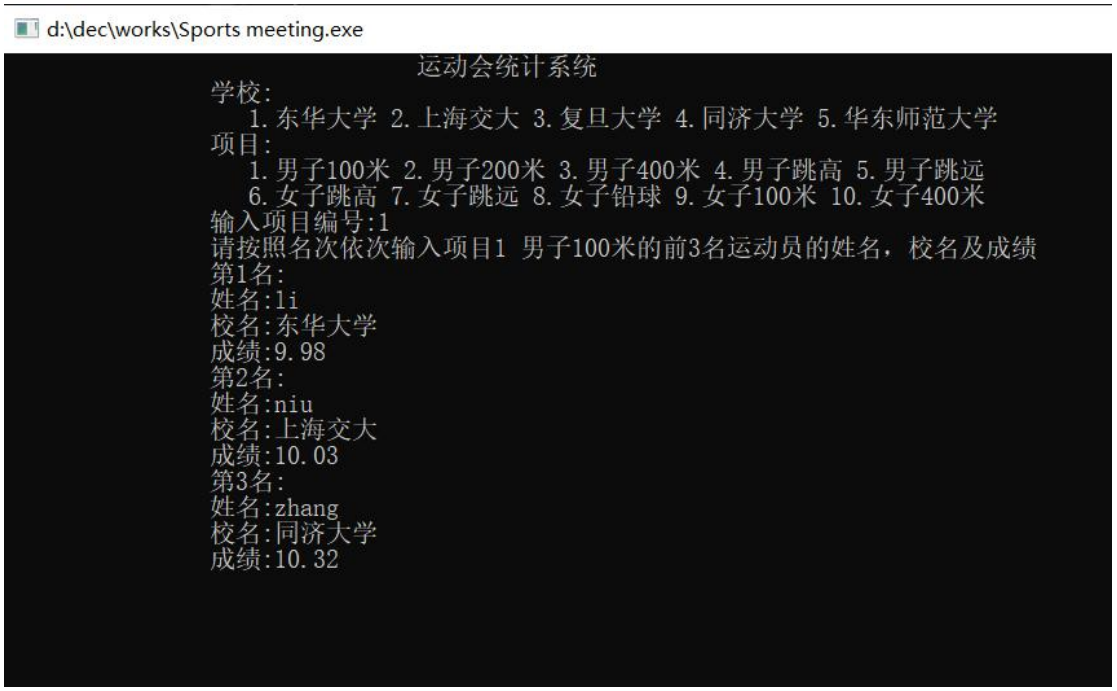


（3）输入 1 后回车进入比赛信息输入界面，按提示输入相关数据：

如若已经有信息，将显示如下：



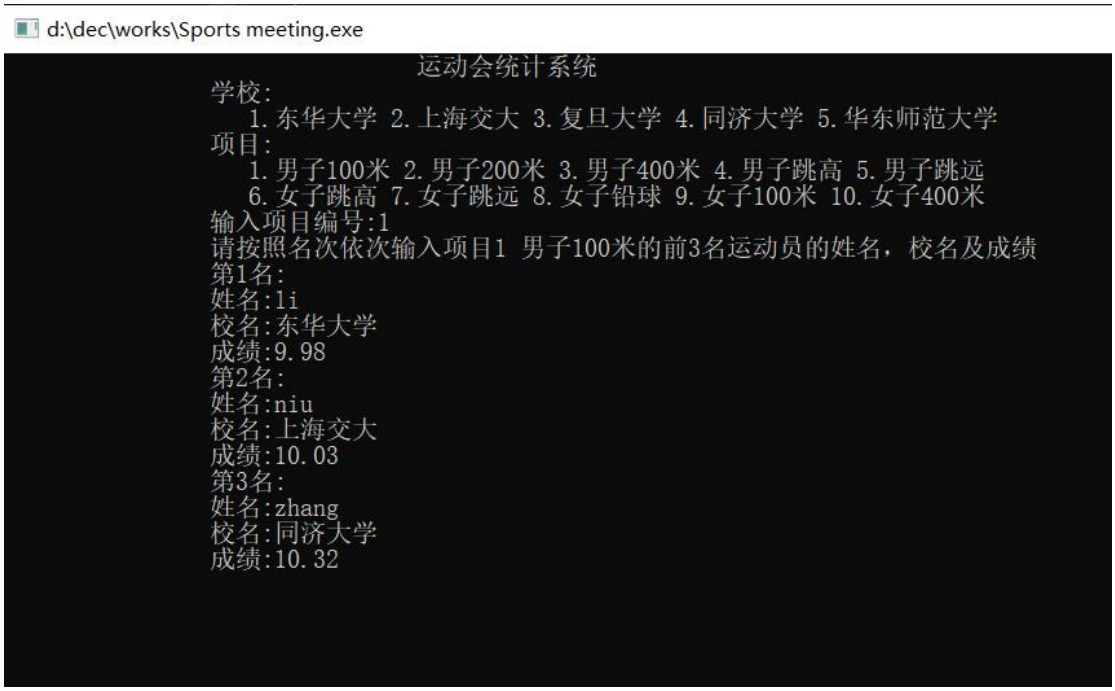
（4）信息输入完后返回主界面，输入 2 回车进入比赛结果输入界面：



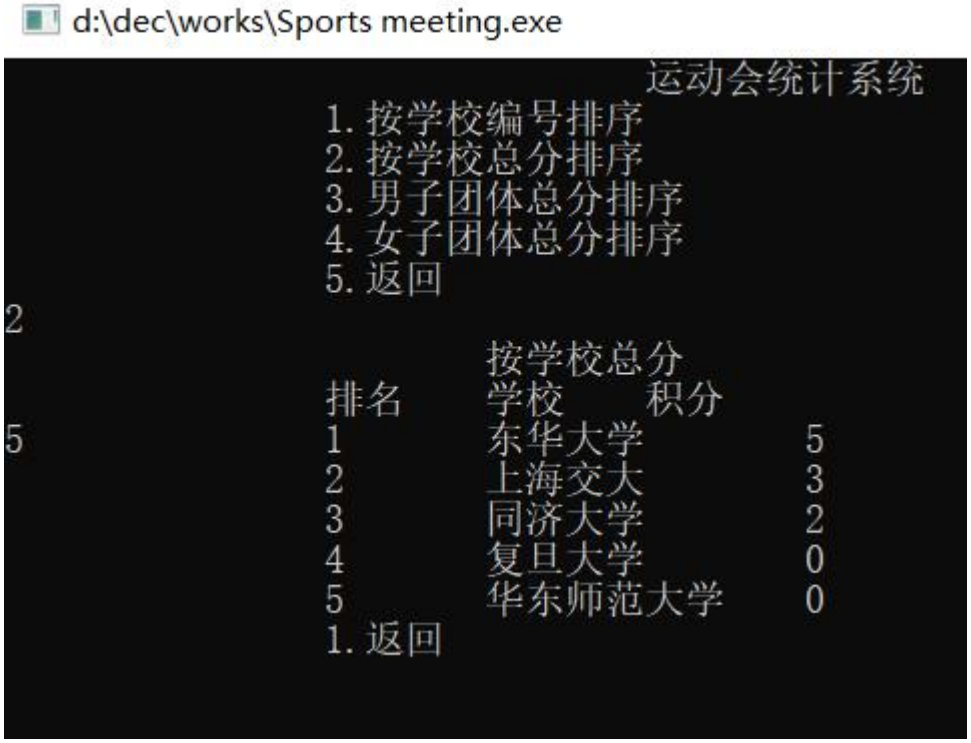
（5）一切信息录入后可进入，就可以进入比赛结果查询界面进行结果查询：

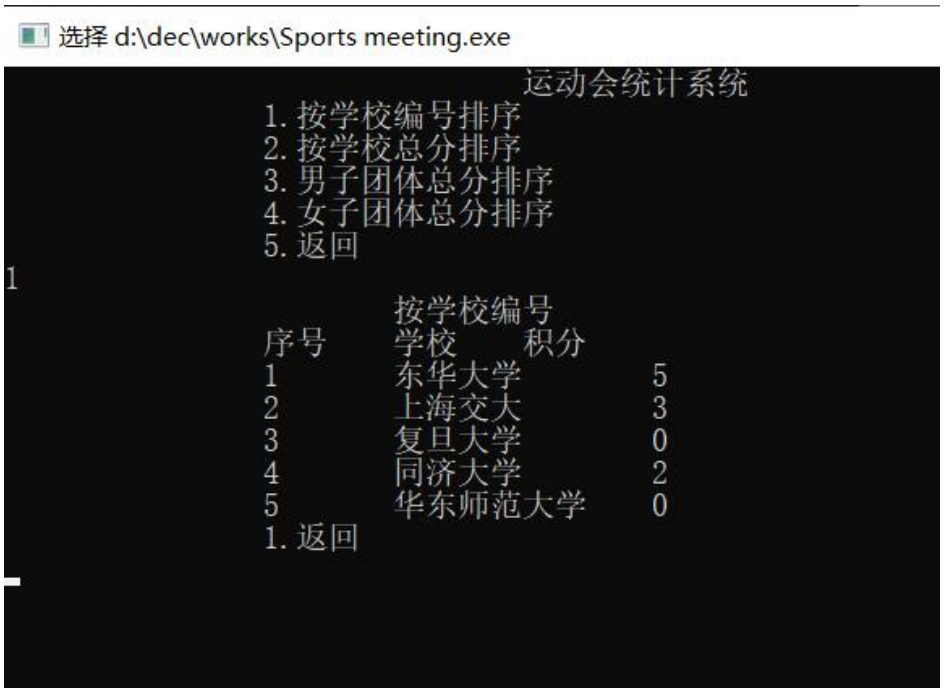
测试结果（截图）：

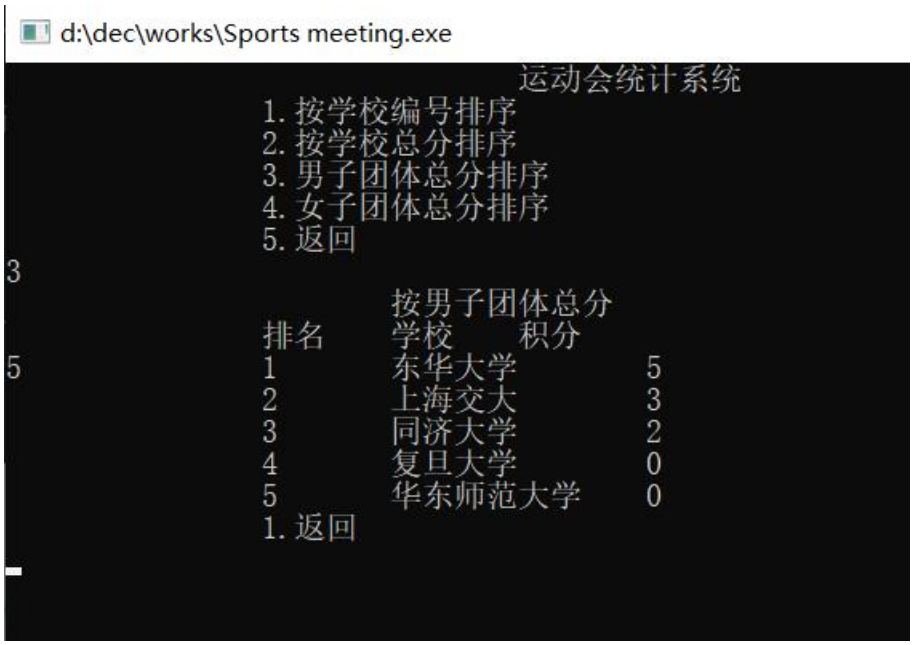
（1）录入如图数据



（2）可获得：







功能要素很多这里就不一一展示了

设计题目

题二： 停车场管理系统

任务：

设计一个停车场管理系统，模拟停车场的运作，此程序具备以下功能：

（1）若车辆到达，则显示汽车在停车场内或者便道上的停车位置；

（2）若车辆离去，则显示汽车在停车场内停留的时间和应缴纳的费用（在便道上停留的时间不收费）

基本功能要求：

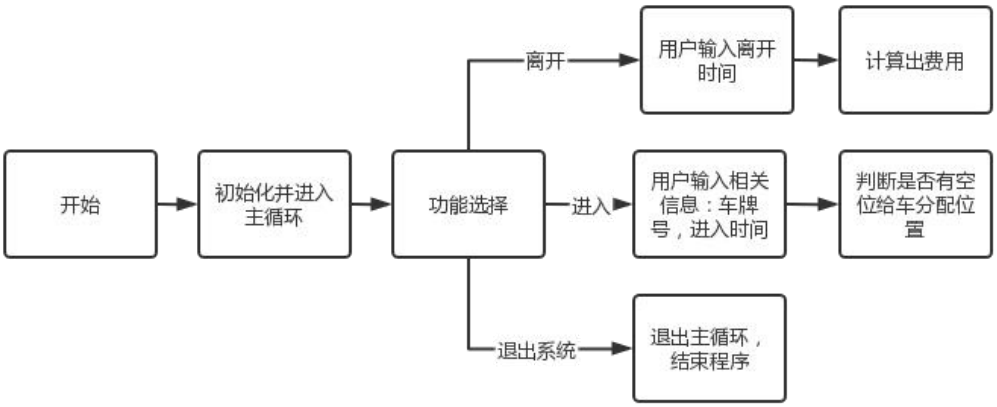
（1）要求以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入和输入数据序列进行模拟管理。

（2）要求处理的数据元素包括三个数据项：汽车“到达”或“离去”信息，汽车牌照号码及到达或离去的时刻。

（3）要求栈以顺序结构实现，队列以链表实现。

需求分析：

主要流程为由用户输入进入停车场的车辆的信息，例如车牌号和进入时间以及离开时间，然后程序会自动生成所需要缴纳的费用。



由于没有外部的辅助设备，帮忙输入相关信息，这里让用户输入的方式来进行完成。

概要设计：

```
typedef struct
{
    int hour;
    int min;
} time;
typedef struct
{
    int num;
    int position;
    time t;
    float money;
} Car;
typedef struct Node
{
    Car data;
    struct Node *next;
} CQueueNode;
typedef struct
{
    Car elem[NUM + 1];
    int top;
} Stack;
typedef struct
{
    CQueueNode *front;
    CQueueNode *rear;
} LinkQueue;
void InitQueue(LinkQueue *Q);          //初始化队列
int EnterQueue(LinkQueue *Q, Car *t); //进队
void InitStack(Stack *S);              //初始化栈
void Push(Stack *S, Car *r);           //压栈
int IsEmpty(Stack *S);                 //判断栈空
int IsFull(Stack *S);                  //判断栈满
int GetTop(Stack *S, Car *n);
int DeleteQueue(LinkQueue *Q, Car *x);
void CarIn(Stack *S, LinkQueue *Q, Car *r);
void CostCalculate(Car *r, int h, int m);
void CarOut(Stack *S, Stack *S0, Car *r, LinkQueue *Q);
```

详细设计：

```
#include <stdio.h>
#include <stdlib.h>
#define NUM 20
typedef struct
{
    int hour;
    int min;
} time;
typedef struct
{
    int num;
    int position;
    time t;
    float money;
} Car;
typedef struct Node
{
    Car data;
    struct Node *next;
} CQueueNode;
typedef struct
{
    Car elem[NUM + 1];
    int top;
} Stack;
typedef struct
{
    CQueueNode *front;
    CQueueNode *rear;
} LinkQueue;
void InitQueue(LinkQueue *Q);          //初始化队列
int EnterQueue(LinkQueue *Q, Car *t); //进队
void InitStack(Stack *S);              //初始化栈
void Push(Stack *S, Car *r);           //压栈
int IsEmpty(Stack *S);                 //判断栈空
int IsFull(Stack *S);                  //判断栈满
int GetTop(Stack *S, Car *n);
int DeleteQueue(LinkQueue *Q, Car *x);
void CarIn(Stack *S, LinkQueue *Q, Car *r);
void CostCalculation(Car *r, int h, int m);
void CarOut(Stack *S, Stack *S0, Car *r, LinkQueue *Q);
```

```c
int main(void)
{
    int n, m, i = 1, j, flag = 0;
    Car c[10];
    Stack S, S0;
    LinkQueue Q;    //便道
    InitStack(&S);  //堆栈 S
    InitStack(&S0); //临时堆栈 S0
    InitQueue(&Q);
    while (1)
    {
        printf("\t\t\t\t 欢迎停车");
        printf("\n\t\t 请选择:\n");
        printf("\n\t\t 1 :进入停车场");
        printf("\n\t\t 2 :离开停车场");
        printf("\n\t\t 3 :退出系统\n");
        printf("\n");
        scanf("%d", &m);
        switch (m)
        {
        case 1:
            printf("\n\t\t 请输入车牌号:");
            scanf("%d", &c[i].num);
            printf("\n\t\t 请输入到达/离开时间(形如 2:00):");
            scanf("%d:%d", &c[i].t.hour, &c[i].t.min);
            CarIn(&S, &Q, &c[i]);
            i++; //车辆的情况
            break;
        case 2:
            printf("\n\t\t 请输入车牌号:");
            scanf("%d", &n);
            for (j = 0; j < 10; j++)
                if (n == c[j].num)
                    break;
            printf("\n\t\t 请输入到达/离开时间(形如 2:00):");
            scanf("%d:%d", &c[j].t.hour, &c[j].t.min);
            CarOut(&S, &S0, &c[j], &Q); //车辆的情况
            break;
        case 3:
            flag = 1;
            break;
        default:
            printf("\n\t\t 请输入 1 或 2 或 3\n");
        }
```

```
        if (flag)
            break; //结束程序
    }
    return 0;
}
void InitQueue(LinkQueue *Q)
{
    Q->front = (CQueueNode *)malloc(sizeof(CQueueNode));
    if (Q->front != NULL)
    {
        Q->rear = Q->front;
        Q->front->next = NULL;
    }
}
int EnterQueue(LinkQueue *Q, Car *t)
{
    CQueueNode *NewNode;
    NewNode = (CQueueNode *)malloc(sizeof(CQueueNode));
    if (NewNode != NULL)
    {
        NewNode->data.num = t->num;
        NewNode->data.t.hour = t->t.hour;
        NewNode->data.t.min = t->t.min;
        NewNode->next = NULL;
        Q->rear->next = NewNode;
        Q->rear = NewNode;
        return 1;
    }
    else
        return 0;
}
void InitStack(Stack *S)
{
    S->top = 0;
}
void Push(Stack *S, Car *r) //便道中的车入库
{
    S->top++;
    S->elem[S->top].num = r->num;
    r->position = S->elem[S->top].position = S->top;
    S->elem[S->top].t.hour = r->t.hour;
    S->elem[S->top].t.min = r->t.min;
}
int IsEmpty(Stack *S) //判断车库是否为空
```

```
{
    return (S->top == 0 ? 1 : 0);
}
int IsFull(Stack *S) //判断车库是否为满
{
    return (S->top == NUM ? 1 : 0);
}
int GetTop(Stack *S, Car *n) //车离开车库
{
    n->num = S->elem[S->top].num;
    n->position = S->elem[S->top].position;
    n->t.hour = S->elem[S->top].t.hour;
    n->t.min = S->elem[S->top].t.min;
    return 1;
}
int DeleteQueue(LinkQueue *Q, Car *x)
{
    CQueueNode *p;
    if (Q->front == Q->rear)
        return 0;       //判断便道为空
    p = Q->front->next; //将便道中的车放入车库
    Q->front->next = p->next;
    if (Q->rear == p)
        Q->rear = Q->front;
    x->num = p->data.num;
    x->t.hour = p->data.t.hour;
    x->t.min = p->data.t.min;
    free(p); //释放临时指针
    return 1;
}
void CarIn(Stack *S, LinkQueue *Q, Car *r)
{
    if (IsFull(S))
    {
        printf("停车场已满，请在便道中等待");
        EnterQueue(Q, r); //车进入便道
    }
    else
    {
        Push(S, r);
        printf("\n\t\t 所在位置 %d", r->position); //打印车的位置
    }
}
void CostCalculation(Car *r, int h, int m)
```

```
{
    if (m > r->t.min)
    {
        r->t.min += 60;
        r->t.hour -= 1;
    }
    h = r->t.hour - h;
    m = r->t.min - m;
    printf("\n\t\t 停车 %d 小时 %d 分钟\n", h, m);
    printf("每小时收费 10 元\n");
    h = h * 20;
    m = h + m;
    r->money = 0.5 * m;
    printf("请支付金额%.2f 元\n", r->money); //输出车主应付金额
}
void CarOut(Stack *S, Stack *S0, Car *r, LinkQueue *Q)
{
    int tag = S->top;
    Car x;
    if (IsEmpty(S))
        printf("不存在该车辆");
    else
    {
        for (; r->num != S->elem[tag].num && tag > 0; tag--)
        {
            Push(S0, &S->elem[tag]);
            S->top--;
        }
        if (r->num == S->elem[tag].num)
        {
            CostCalculation(r, S->elem[tag].t.hour,
S->elem[tag].t.min);
            S->top--;
            for (; S0->top > 0; S0->top--)
                Push(S, &S0->elem[S0->top]);
            if (S->top < NUM && Q->front != Q->rear) //判断车库是
否有此车，有就找到此车，然后退出
            {
                DeleteQueue(Q, &x);
                Push(S, &x);
            }
        }
        else if (tag == 0) //过道中的车无需收车费
        {
```

```
            printf("没有进入停车场支付金额 0 元");
            for (; S0->top > 0; S0->top--)
                Push(S, &S0->elem[S0->top]);
        }
    }
}
```

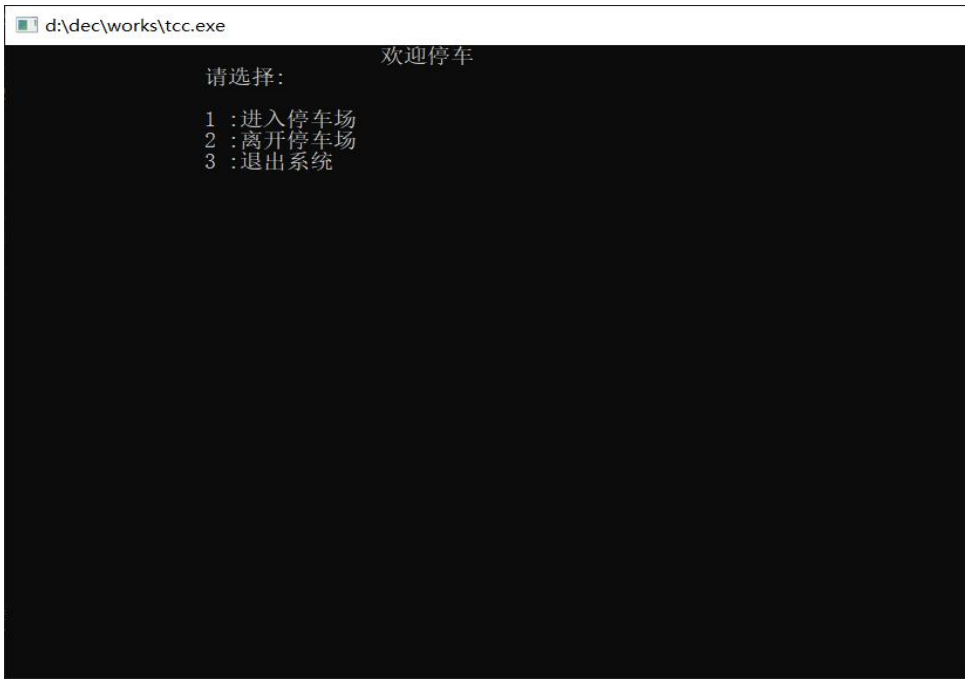
调试分析：

a. 由于上一个题目做的很丰富，非常耗时，这个题目尝试对程序进行精简，立足于题目要求，不再增加额外的内容导致画蛇添足。
b. 刚开始时很茫然，对于队和栈的相关知识遗忘了很多，通过查阅资料和复习，我最终还是克服了这些问题，但现存的问题依旧不好解决，如：代码太啰嗦，可读性还有待提高。

用户手册：

（1）演示程序的运行环境为 Windows10 系统，Visual Code Studio 中的 MinGW64 中运行。执行文件为：tcc.exe。

（2）进入演示程序后即进入控制台：



（3）输入 1 输入相应信息后，即可完成停车操作

（4）输入 2 离开即可计算出费用。

（5）输入 3 即可退出系统。

测试结果（截图）：

（1）输入 1，车牌号 123456，以及到达时间 2:00
（2）输入 1，车牌号 654321，以及到达时间 3:00

```
d:\dec\works\tcc.exe
                                欢迎停车
                请选择:

                1 :进入停车场
                2 :离开停车场
                3 :退出系统

1

                请输入车牌号:123456

                请输入到达/离开时间(形如2:00):2:00

                所在位置
1                               欢迎停车
                请选择:

                1 :进入停车场
                2 :离开停车场
                3 :退出系统

1

                请输入车牌号:654321

                请输入到达/离开时间(形如2:00):3:00

                所在位置
2                               欢迎停车
                请选择:
```

(3) 输入 2,输入车牌号 123465，输入离开时间 3:00
(4) 输入 2,输入车牌号 654321，输入离开时间 4:00

```
d:\dec\works\tcc.exe
                2 :离开停车场
                3 :退出系统

2

                请输入车牌号:123456

                请输入到达/离开时间(形如2:00):3:00

                停车 1 小时 0 分钟
每小时收费 10 元
请支付金额10.00 元
                                欢迎停车
                请选择:

                1 :进入停车场
                2 :离开停车场
                3 :退出系统

2

                请输入车牌号:654321

                请输入到达/离开时间(形如2:00):4:00

                停车 1 小时 0 分钟
每小时收费 10 元
请支付金额10.00 元
                                欢迎停车
                请选择:
```

设计题目

题三： 迷宫问题求解

任务：

迷宫问题是取自心理学的一个古典实验。实验中，把一只老鼠从一个没有顶的大盒子的门放入，在盒中设置了许多墙，对行进的方向形成了多处阻挡。盒子仅仅有一个出口，在出口处放置了一块奶酪，吸引老鼠在迷宫中寻找道路以到达出口。重复对老鼠进行上述实验，看老鼠能在多久找到出口。

请设计一个算法实现迷宫问题求解。

基本功能要求：

实现迷宫通路的找出。

需求分析：

想象一只真实的老鼠在迷宫中的场景，它需要一步步的试错，不断地找到新的可以通行的地方，同时为了避免陷入死循环，绕圈子，我们需要记住来时的路径，不走重复的路径。这可以通过栈来实现，将我们的选择不断压入栈中，如果陷入了死局，我们就删去栈顶，重新寻找，直到所有可能被穷尽或找到出口，为了避免死胡同，陷入循环，我们规定试错的方向为顺时针方向。

概要设计：

```
typedef struct
{
    int ord;         //路径中走过的序号
    int seat[2];    //坐标
    int di;         //从这一块走向下一块的方向
} SElemType;
Status Mazepath(int *start, int *end);//主要的逻辑函数
void NextPos(int curpos[2], int x, int y, int di);
Status Pass(int x, int y);//判断当前块可否通行
```

详细设计：

```
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#define OVERFLOW -2
#define SIZE_H 6 //迷宫矩阵的行数
#define SIZE_L 5 //迷宫矩阵的列数
int curstep = 1;
int map_flag[6][5] = {{0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0,
0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}}; //足
迹标注
int maze[6][5] = {{0, 1, 0, 0, 0}, {1, 0, 1, 1, 0}, {1, 0, 1, 1,
0}, {1, 1, 0, 1, 1}, {0, 0, 1, 0, 1}, {1, 1, 1, 0, 0}};     //
定义迷宫
typedef int Status;
typedef struct
{
    int ord;         //路径中走过的序号
    int seat[2];    //坐标
    int di;         //从这一块走向下一块的方向
} SElemType;
#define STACK_INIT_SIZE 10 /* 存储空间初始分配量 */
#define STACKINCREMENT 2   /* 存储空间分配增量 */
typedef struct SqStack
{
    SElemType *base; /* 在栈构造之前和销毁之后，base 的值为 NULL */
    SElemType *top;  /* 栈顶指针 */
    int stacksize;   /* 当前已分配的存储空间，以元素为单位 */
} SqStack;           /* 顺序栈 */
SqStack S;
Status InitStack(SqStack &S)
{ /* 构造一个空栈 S */
    S.base = (SElemType *)malloc(STACK_INIT_SIZE *
sizeof(SElemType));
    if (!S.base)
        exit(OVERFLOW); /* 存储分配失败 */
    S.top = S.base;
    S.stacksize = STACK_INIT_SIZE;
    return 1;
}
Status Pop(SqStack &S, SElemType &e)
{ /* 若栈不空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK;否则返回
ERROR */
```

```
    if (S.top == S.base)
        return 0;
    e = *--S.top;
    return 1;
}
Status Push(SqStack &S, SElemType e)
{                                       /* 插入元素 e 为新的栈顶元
素 */
    if (S.top - S.base >= S.stacksize) /* 栈满，追加存储空间 */
    {
        S.base = (SElemType *)realloc(S.base, (S.stacksize +
STACKINCREMENT) * sizeof(SElemType));
        if (!S.base)
            exit(OVERFLOW); /* 存储分配失败 */
        S.top = S.base + S.stacksize;
        S.stacksize += STACKINCREMENT;
    }
    *(S.top)++ = e;
    return 1;
}
Status StackEmpty(SqStack S)
{ /* 若栈 S 为空栈，则返回 TRUE，否则返回 FALSE */
    if (S.top == S.base)
        return 1;
    else
        return 0;
}
void FootPrint(int *curpos)
{
    map_flag[curpos[0]][curpos[1]] = 1;
    printf("经过(%d,%d),", curpos[0], curpos[1]);
}
void MarkPrint(int *seat)
{
    if (map_flag[seat[0]][seat[1]] != 1)
    {
        map_flag[seat[0]][seat[1]] = -1;
        printf("到(%d,%d),此路不通，掉头!\n", seat[0], seat[1]);
    }
    else
        map_flag[seat[0]][seat[1]] = -1;
}
Status Pass(int x, int y)
{
```

```
    if (maze[x][y] != 1 && map_flag[x][y] == 0)
        return 1;
    else
        return 0;
}
void NextPos(int curpos[2], int x, int y, int di)
{
    int m = curpos[0];
    int n = curpos[1];
    if (di == 1)
    {
        curpos[0] = x;
        curpos[1] = y + 1;
    }
    else if (di == 2)
    {
        curpos[0] = x + 1;
        curpos[1] = y + 1;
    }
    else if (di == 3)
    {
        curpos[0] = x + 1;
        curpos[1] = y;
    }
    else if (di == 4)
    {
        curpos[0] = x + 1;
        curpos[1] = y - 1;
    }
    else if (di == 5)
    {
        curpos[0] = x;
        curpos[1] = y - 1;
    }
    else if (di == 6)
    {
        curpos[0] = x - 1;
        curpos[1] = y - 1;
    }
    else if (di == 7)
    {
        curpos[0] = x - 1;
        curpos[1] = y;
    }
```

```
    else if (di == 8)
    {
        curpos[0] = x - 1;
        curpos[1] = y + 1;
    }
    if (curpos[0] < 0 || curpos[0] > 5)
        curpos[0] = m;
    if (curpos[1] < 0 || curpos[1] > 4)
        curpos[1] = n;
}
Status Mazepath(int *start, int *end)
{
    int curpos[2];
    SElemType e;
    InitStack(S);
    curpos[0] = start[0];
    curpos[1] = start[1];
    curstep = 1;
    do
    {

        if (Pass(curpos[0], curpos[1]))
        {
            FootPrint(curpos);
            e.ord = curstep;
            e.seat[0] = curpos[0];
            e.seat[1] = curpos[1];
            e.di = 1;
            Push(S, e);
            if ((curpos[0] == end[0]) && (curpos[1] == end[1]))
                return 1;
            NextPos(curpos, curpos[0], curpos[1], 1);
            curstep++;
        }
        else
        {
            if (!StackEmpty(S))
            {
                Pop(S, e);
                while (e.di == 8 && !StackEmpty(S))
                {
                    MarkPrint(e.seat);
                    Pop(S, e);
                }
```

```
                if (e.di < 8)
                {
                    e.di++;
                    Push(S, e);
                    NextPos(curpos, e.seat[0], e.seat[1], e.di);
                }
            }
        }
    } while (!StackEmpty(S));
    return 0;
}
int main(void)
{
    int i = 0,j = 0;
    int start[2] = {0, 0};
    int end[2] = {5, 4};
    if (Mazepath(start, end) != 1)
        printf("\n 迷宫无解");
    else
    {
        printf("抵达出口\n");
        printf("路径如图，“1”标识迷宫的解，“-1”标识试错的路径\n");
    }
    for (i = 0; i < SIZE_H; i++)
    {
        for (j = 0; j < SIZE_L; j++)
        {
            printf("%d  ", map_flag[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

调试分析：

a. 中间遇到了一个棘手的问题，出现了路径的无限循环，经过打点 debug，最终判断问题在于我没有标记已经走过的路径，导致在两个块之间反复跳动。 ‘

b. 还有一个很致命的问题在于，我在转换方向时，当只需要改变单独改变 x 或 y 坐标时，我没有给另一个坐标复位，导致程序天马行空，做出了很多出乎意料的举动。最终通过打断点一步步 debug 才查到该漏洞。

c. 栈的先进后出的特性真的十分方便有用。

用户手册：

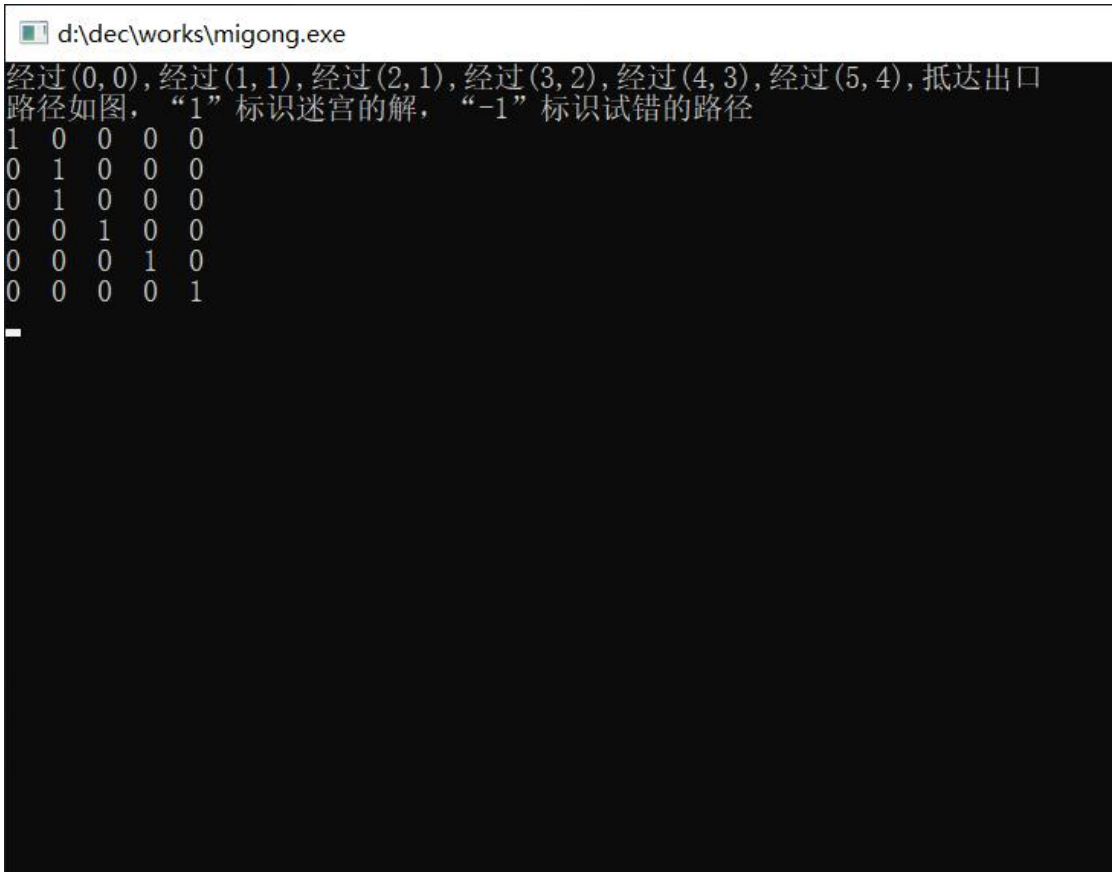
1)演示程序的运行环境为 Windows10 系统,Visual Code Studio 中的 MinGW64 中运行。执行文件为：migong.exe

（2）进入演示程序后即显示 DOS 形式的界面：

该解为参考测试数据的解，可以改变代码中对迷宫的定义来验证

定义如下：

```
#define SIZE_H 6 //迷宫矩阵的行数
#define SIZE_L 5 //迷宫矩阵的列数
int maze[6][5] = {{0, 1, 0, 0, 0},
                  {1, 0, 1, 1, 0},
                  {1, 0, 1, 1, 0},
                  {1, 1, 0, 1, 1},
                  {0, 0, 1, 0, 1},
                  {1, 1, 1, 0, 0}};     //定义迷宫
```

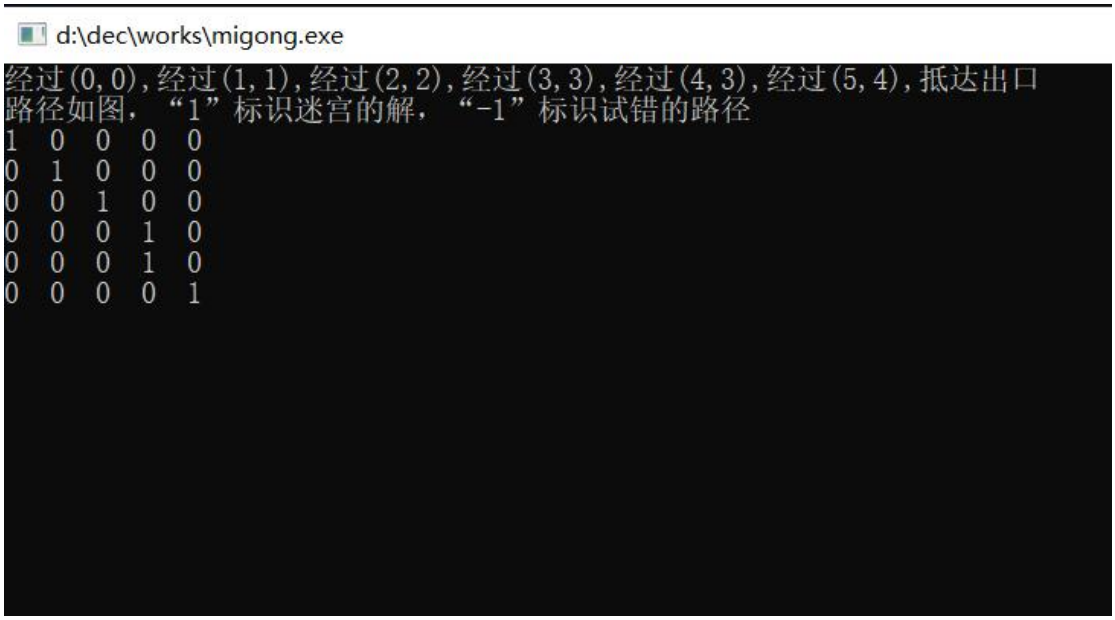


测试结果（截图）：

手动修改迷宫的代码定义:

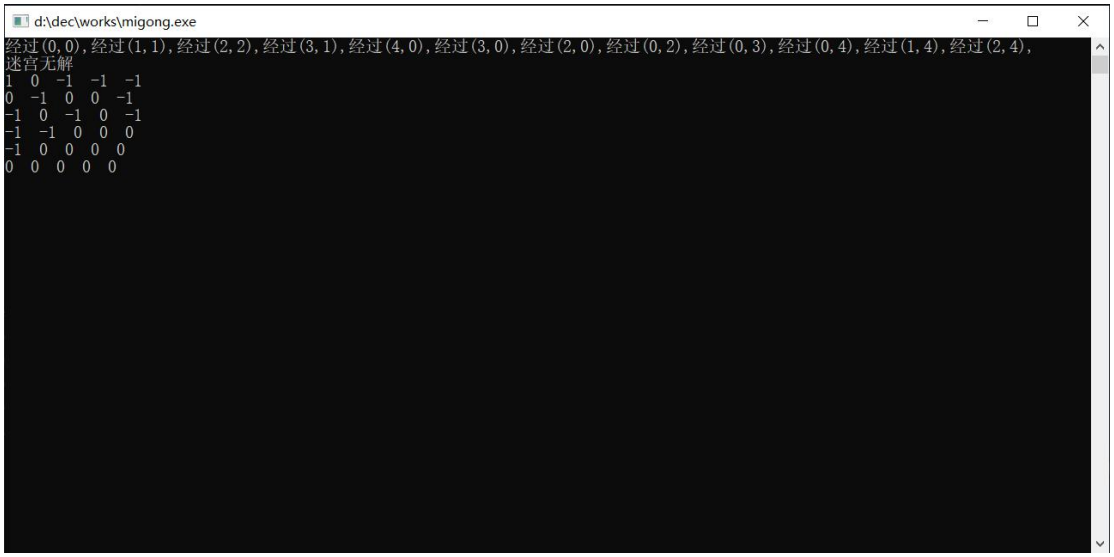
(1)修改为

```
#define SIZE_H 6 //迷宫矩阵的行数
#define SIZE_L 5 //迷宫矩阵的列数
int maze[6][5] = {{0, 1, 0, 0, 0},
                  {1, 0, 1, 1, 0},
                  {1, 0, 0, 1, 0},
                  {1, 0, 1, 0, 1},
                  {0, 0, 1, 0, 1},
                  {1, 1, 1, 0, 0}};     //定义迷宫
```



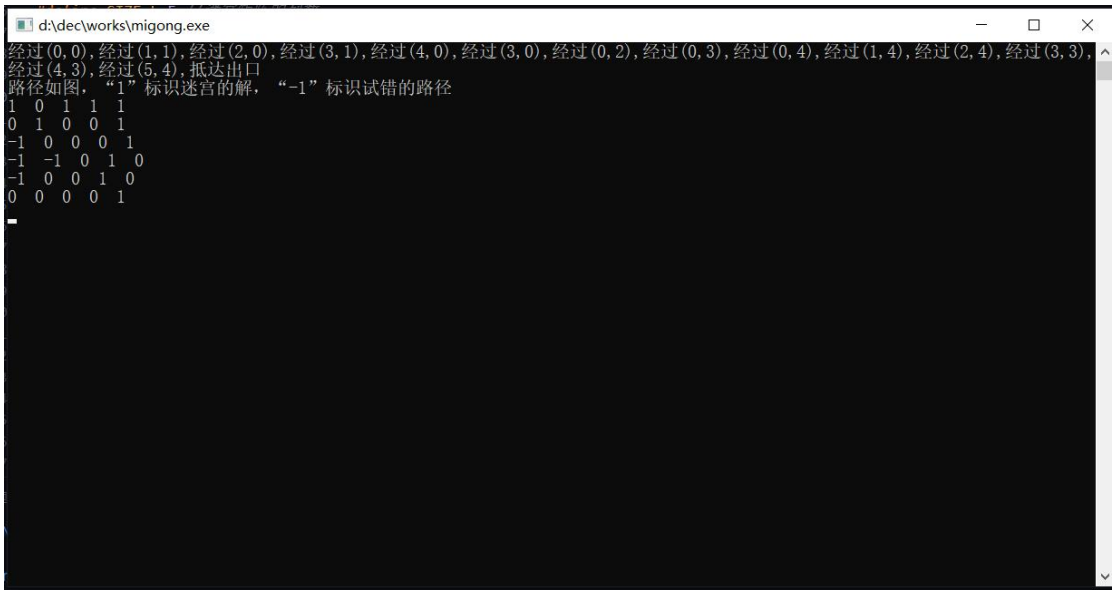
(2) 修改为

```
#define SIZE_H 6 //迷宫矩阵的行数
#define SIZE_L 5 //迷宫矩阵的列数
int maze[6][5] = {{0, 1, 0, 0, 0},
                  {1, 0, 1, 1, 0},
                  {0, 1, 0, 1, 0},
                  {0, 0, 1, 1, 1},
                  {0, 1, 1, 0, 1},
                  {1, 1, 1, 0, 0}};     //定义迷宫
```



(3) 修改为:

```
#define SIZE_H 6 //迷宫矩阵的行数
#define SIZE_L 5 //迷宫矩阵的列数
int maze[6][5] = {{0, 1, 0, 0, 0},
                  {1, 0, 1, 1, 0},
                  {0, 1, 1, 1, 0},
                  {0, 0, 1, 0, 1},
                  {0, 1, 1, 0, 1},
                  {1, 1, 1, 0, 0}};     //定义迷宫
```



总结与分析：

写程序一定要有耐心，代码量越大出错的可能性就越高，熟练的运用数据结构的相关知识，可以减少出错的机会，使得代码的层次更加的清晰完善，避免不必要的麻烦。