

专题 2（数据处理类）：世界杯小组赛积分统计系统

【应用背景】世界杯小组赛积分统计

2026 年美加墨世界杯是第 23 届世界杯足球赛，由美国、加拿大和墨西哥联合主办，将于 2026 年 7 月 19 日举行。让我们先对 2022 卡塔尔世界杯进行一次精彩回顾吧。



世界杯小组赛采用积分制，每个小组 4 个球队，共进行 6 场比赛，每个球队胜一场球积 3 分，平一场球积 1 分，负一场球得 0 分，最后小组内根据各球队的积分排名决定进入下一轮比赛的球队。

本程序的任务是简单模拟 2022 世界杯小组赛的积分统计系统，统计每个球队的进球数、失球数、胜利场数、平局场数、失败场数、积分，然后计算排名。通过读入文件数据，可以查询各小组的积分排名以及各球队的赛事信息等，并存储至相应文件中。

各问摘要：

第1问，基础问答，单项选择题。

第2问，程序调试，实现全部赛事信息和所有球队进/失球数的表示。

第3问，程序规范，程序代码的模块化处理。

第4问，二级集成，通过数据文件读取全部赛事信息，并实现球队查找。

第5问，三级应用，实现按小组及积分排序的功能，并计算球队在本小组的排名。

第6问：应用提升，动态数组应用，赛事信息查找，系统功能集成，将球队所有信息写入文件。

【第 1 问，基础问答，具体答题在考试客户端中单项选择题部分】

【第 2 问，程序调试，实现全部赛事信息和所有球队进/失球数的表示】

下列程序中提供了若干小组的赛事信息，存放在结构体match类型的数组中，结构体match定义为：

```
struct match
{
    char group;           //小组
    char team1[20];       //球队1
    char team2[20];       //球队2
```

```

    int goal1;           //球队1的进球数
    int goal2;           //球队2的进球数
};

```

程序实现了全部赛事信息和所有球队进/失球数的表示，其中，球队比赛成绩存放在结构体team类型的数组中，结构体team类型的定义为：

```

struct team
{
    char group;           //小组
    char team[20];        //球队
    int goal;             //进球数
    int lost;             //失球数
    int win;              //胜利场数
    int tie;              //平局场数
    int defeat;           //失败场数
    int score;            //积分
    int rank;             //排名
};

```

程序中包含 3 个错误，请按题中的功能要求，打开 C:\KS\WorldCup02Err.c，调试并修改该程序(在所修改语句后加“/* */”或“//_”作为标记)，使其运行能得到正确的结果。修改后的程序保存为 C:\KS\WorldCup02.c。

运行结果示例：

```

小组      赛事      比分
A          Qatar:Ecuador  0:2
A          Senegal:Holland 0:2
A          Qatar:Senegal  1:3
A          Holland:Ecuador 1:1
A          Holland:Qatar  2:0
A          Ecuador:Senegal 1:2
B          England:Iran  6:2
B          USA:Wales    1:1
B          Wales:Iran   0:2
B          England:USA   0:0
B          Wales:England 0:3
B          Iran:USA     0:1

小组  球队      进/失
A     Qatar    1/7
A     Senegal  5/4
A     Holland  5/1
A     Ecuador  4/3
B     Iran     4/7
B     England  9/2
B     USA      2/1
B     Wales    1/6

```

包含错误的程序如下：

```

#include<stdio.h>
#include<string.h>
#include<stdlib.h>

```

```

struct team
{

```

```

char group;           //小组
char team[20];        //球队
int goal;             //进球数
int lost;             //失球数
int win;              //胜利场数
int tie;              //平局场数
int defeat;           //失败场数
int score;            //积分，胜一场球积 3 分，平一场球积 1 分，负一场球得 0 分
int rank;             //排名
};

```

```

struct match
{
    char group;           //小组
    char team1[20];        //球队 1
    char team2[20];        //球队 2
    int  goal1;            //球队 1 的进球数
    int  goal2;            //球队 2 的进球数
}

```

```

int main()
{
    struct team t[8] = {{'A',"Qatar"},{'A',"Senegal"},{'A',"Holland"},{'A',"Ecuador"},
                        {'B',"Iran"},{'B',"England"},{'B',"USA"},{'B',"Wales"}};
    struct match m[12] = {{'A',"Qatar","Ecuador",0,2}, {'A',"Senegal","Holland",0,2},
                          {'A',"Qatar","Senegal",1,3}, {'A',"Holland","Ecuador",1,1},
                          {'A',"Holland","Qatar",2,0}, {'A',"Ecuador","Senegal",1,2},
                          {'B',"England","Iran",6,2}, {'B',"USA","Wales",1,1},
                          {'B',"Wales","Iran",0,2}, {'B',"England","USA",0,0},
                          {'B',"Wales","England",0,3}, {'B',"Iran","USA",0,1}};

    int i,j,flag;

    // 获取所有球队进/失球数
    for(i=0; i<8; i++)
    {
        t[i].goal=t[i].lost=1;
        for(j=0; j<12; j++) //本循环体无错
        {
            flag=0;
            if (strcmp(t[i].team, m[j].team1)==0) flag=1;
            if (strcmp(t[i].team, m[j].team2)==0) flag=2;
            switch (flag)
            {
                case 1: t[i].goal=t[i].goal+m[j].goal1; t[i].lost=t[i].lost+m[j].goal2; break;

```

```

        case 2: t[i].goal=t[i].goal+m[j].goal2; t[i].lost=t[i].lost+m[j].goal1;
    }
}
}

//输出所有比赛
printf("  小组          赛事          比分\n");
for(i=0;i<12;i++);
{
    printf("    %c    %10s:%-10s %d:%d\n",
           m[i].group,m[i].team1,m[i].team2,m[i].goal1,m[i].goal2);
}
putchar('\n');

//输出所有球队进/失球数
printf("  小组  球队      进/失\n");
for(i=0;i<8;i++)
{
    printf("    %c    %-10s %d/%d\n",t[i].group,t[i].team,t[i].goal,t[i].lost);
}
return 0;
}

```

【第 3 问，程序规范，程序代码的模块化处理】

在第 2 问的基础上，修改程序，并将修改后的程序另保存为 **C:\KS\WorldCup03.c**。

(1) 通过函数 **get_team()** 获取球队的进/失球数、胜/平/负局数、以及积分。函数原型为：

```
void get_team(struct team t[], int n1, struct match m[], int n2);
```

其中，结构体数组 **t** 中已经保存 **n1** 个球队的信息，结构体数组 **m** 中已经保存 **n2** 场赛事。

(2) 通过函数 **display_match()** 输出赛事信息。函数原型为：

```
void display_match(struct match m[], int n);
```

其中，结构体数组 **m** 中已经保存 **n** 场赛事信息。

(3) 通过函数 **display_team()** 输出球队比赛成绩（进/失球数、胜/平/负局数、积分）。函数原型为：

```
void display_team(struct team t[], int n);
```

其中，结构体数组 **t** 中已经保存 **n** 个球队的信息。

(4) 修改 **main()** 函数，**main()** 函数只起声明数据和调用函数的功能。**main()** 函数中，结构体声明语句如第 2 问，各调用函数语句如下：

```

get_team(t, 8, m, 12);
display_match(m,12);
display_team(t,8);

```

输出结果示例：

小组	赛事	比分
A	Qatar:Ecuador	0:2
A	Senegal:Holland	0:2
A	Qatar:Senegal	1:3
A	Holland:Ecuador	1:1
A	Holland:Qatar	2:0
A	Ecuador:Senegal	1:2
B	England:Iran	6:2
B	USA:Wales	1:1
B	Wales:Iran	0:2
B	England:USA	0:0
B	Wales:England	0:3
B	Iran:USA	0:1

小组	球队	进/失	胜/平/负	积分
A	Qatar	1/7	0/0/3	0
A	Senegal	5/4	2/0/1	6
A	Holland	5/1	2/1/0	7
A	Ecuador	4/3	1/1/1	4
B	Iran	4/7	1/0/2	3
B	England	9/2	2/1/0	7
B	USA	2/1	1/2/0	5
B	Wales	1/6	0/1/2	1

【第 4 问，二级集成，通过文件数据读取全部赛事信息，并实现球队查找。】

在第 3 问的基础上修改程序，完成如下功能，并将修改后的程序另保存为 C:\KS\WorldCup04.c。

(1) 赛事信息可以通过读取文本文件 match.txt 得到，假设 match.txt 中包含不超过 100 个赛事信息。文件中的数据和结构体 match 对应，分别为：小组、球队 1、球队 2、球队 1 的进球数、球队 2 的进球数。示例如下：

```

match - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
A Qatar Ecuador 0 2
A Senegal Holland 0 2
A Qatar Senegal 1 3
A Holland Ecuador 1 1
A Holland Qatar 2 0
A Ecuador Senegal 1 2
B England Iran 6 2
B USA Wales 1 1
B Wales Iran 0 2
B England USA 0 0
B Wales England 0 3
B Iran USA 0 1

```

读取数据文件 match.txt 的 read()函数原型为：

```
int read(struct match m[]);
```

其中，结构体数组 m 用于存放读取到的赛事信息，函数返回读取到的信息记录数，如果文件打开不成功，输出“文件打开失败！”，程序运行结束。

提示:如果采用 fscanf 语句,因为输入语句中第一个格式符为%c,需要在格式符中加入'\n',具体格式为: fscanf(fp, "%c%s%d%d\n",).

说明:球队初始信息仍然通过结构体数组初始化获取。

(2) 通过函数 search_team()查询某球队的信息。函数原型为：

```
void search_team(struct team t[], int n, char team[ ]);
```

其中，结构体数组 t 中保存了 n 个球队的信息，函数功能是查询球队名称为 team 的球队信息，如果找到，输出该球队信息，如果没找到，输出“没有找到该球队”。

(3) 修改 main() 函数，调用各函数，主函数只起声明数据和调用函数的功能。

输出结果示例1：

```

小组    赛事    比分
A      Qatar:Ecuador    0:2
A      Senegal:Holland    0:2
A      Qatar:Senegal    1:3
A      Holland:Ecuador    1:1
A      Holland:Qatar    2:0
A      Ecuador:Senegal    1:2
B      England:Iran    6:2
B      USA:Wales    1:1
B      Wales:Iran    0:2
B      England:USA    0:0
B      Wales:England    0:3
B      Iran:USA    0:1

小组  球队  进/失  胜/平/负  积分
A    Qatar    1/7    0/0/3    0
A    Senegal    5/4    2/0/1    6
A    Holland    5/1    2/1/0    7
A    Ecuador    4/3    1/1/1    4
B    Iran    4/7    1/0/2    3
B    England    9/2    2/1/0    7
B    USA    2/1    1/2/0    5
B    Wales    1/6    0/1/2    1

请输入想要查询的球队:USA

小组  球队  进/失  胜/平/负  积分
B    USA    2/1    1/2/0    5

```

输出结果示例2:

```

小组    赛事    比分
A      Qatar:Ecuador    0:2
A      Senegal:Holland    0:2
A      Qatar:Senegal    1:3
A      Holland:Ecuador    1:1
A      Holland:Qatar    2:0
A      Ecuador:Senegal    1:2
B      England:Iran    6:2
B      USA:Wales    1:1
B      Wales:Iran    0:2
B      England:USA    0:0
B      Wales:England    0:3
B      Iran:USA    0:1

小组  球队  进/失  胜/平/负  积分
A    Qatar    1/7    0/0/3    0
A    Senegal    5/4    2/0/1    6
A    Holland    5/1    2/1/0    7
A    Ecuador    4/3    1/1/1    4
B    Iran    4/7    1/0/2    3
B    England    9/2    2/1/0    7
B    USA    2/1    1/2/0    5
B    Wales    1/6    0/1/2    1

请输入想要查询的球队:Qutor
没有找到该球队

```

【第 5 问，三级应用，实现按小组及积分排序的功能，并计算球队在本小组的排名】

在第 4 问的基础上修改程序，完成如下功能，并将修改后的程序另保存为 C:\KS\WorldCup05.c。

(1) 通过函数 score_sort() 把球队按照小组升序和积分降序排序：首先按小组的升序排序，同一小组内按照积分降序排序。函数原型为：

```
void score_sort(struct team t[], int n);
```

其中，结构体数组 t 中已经保存 n 个球队的信息。

(2) 通过函数 get_rank() 得到各球队在小组内的排名。函数原型为：

```
void get_rank(struct team t[], int n);
```

其中，结构体数组 t 中已经保存 n 个球队的信息。排名结果如下图所示：

小组	球队	进/失	胜/平/负	积分	排名
A	Holland	5/1	2/1/0	7	1
A	Senegal	5/4	2/0/1	6	2
A	Ecuador	4/3	1/1/1	4	3
A	Qatar	1/7	0/0/3	0	4
B	England	9/2	2/1/0	7	1
B	USA	2/1	1/2/0	5	2
B	Iran	4/7	1/0/2	3	3
B	Wales	1/6	0/1/2	1	4

提示：在排序的基础上求排名。

(3) 修改第 3 问中的函数 display_team() 输出球队信息时，能输出球队的排名。函数原型仍为：

```
void display_team(struct team t[], int n);
```

其中，结构体数组 t 中已经保存 n 个球队的信息。

(4) 修改第 4 问中的函数 search_team()，输出找到的球队信息时，能输出球队的排名，其他

要求不变。

(5) 修改 main() 函数，main() 函数只起声明数据和调用函数的功能。

输出结果示例：

小组	赛事	比分
A	Qatar:Ecuador	0:2
A	Senegal:Holland	0:2
A	Qatar:Senegal	1:3
A	Holland:Ecuador	1:1
A	Holland:Qatar	2:0
A	Ecuador:Senegal	1:2
B	England:Iran	6:2
B	USA:Wales	1:1
B	Wales:Iran	0:2
B	England:USA	0:0
B	Wales:England	0:3
B	Iran:USA	0:1

小组	球队	进/失	胜/平/负	积分	排名
A	Holland	5/1	2/1/0	7	1
A	Senegal	5/4	2/0/1	6	2
A	Ecuador	4/3	1/1/1	4	3
A	Qatar	1/7	0/0/3	0	4
B	England	9/2	2/1/0	7	1
B	USA	2/1	1/2/0	5	2
B	Iran	4/7	1/0/2	3	3
B	Wales	1/6	0/1/2	1	4

请输入想要查询的球队:USA

小组	球队	进/失	胜/平/负	积分	排名
B	USA	2/1	1/2/0	5	2

【第 6 问，三级提升，动态数组应用，赛事信息查找，系统功能集成，将球队所有信息写入文件】

在第 5 问的基础上修改程序，完成如下功能，并将修改后的程序另保存为 C:\KS\WorldCup06.c。

(1) 通过函数 count() 计算文件 match.txt 中的记录个数。函数原型为：

```
int count(char *filename);
```

函数返回文件 filename 中的记录个数。

(2) main() 中声明存储赛事的数组时，采用动态数组声明，步骤如下：调用 count() 函数计算文件 match.txt 中的记录数，然后根据该记录数声明动态数组。

动态数组使用举例：

```
int main()
{
    int *A;
    int n,i;
    printf("请输入数组大小: ");
    scanf("%d",&n);
    A=(int *)malloc(sizeof(int)*n); //动态申请了大小为 n 的数组
    printf("请输入数组元素: ");
    for(i=0;i<n;i++)
        scanf("%d",&A[i]);
    .....
}
```

(3) 通过函数 search_match() 查询某球队的赛事信息。函数原型为：

```
int search_match(struct match m[], int n, struct match m_work[], char team[ ]);
```


其中，结构体数组 `m` 中已保存 `n` 条赛事信息，函数功能是查询球队名称为 `team` 的赛事信息，并保存至结构体数组 `m_work` 中，函数返回值为查询到的赛事信息数。

注意该函数中不输出查找到的比赛信息，查找到的球队比赛信息存储在结构体数组 `m_work` 中，在 `main` 函数中通过调用 `display_match()` 函数输出。

(4) 通过函数 `save()` 保存所有球队信息到 `team.txt` 数据文件中。函数原型为：

```
void save(struct team t[], int n);
```

其中，结构体数组 `t` 中已保存 `n` 条球队信息。

`team.txt` 中的数据如下：

*team - 记事本					
文件(F)	编辑(E)	格式(O)	查看(V)	帮助(H)	
A Holland	5/1	2/1/0	7	1	
A Senegal	5/4	2/0/1	6	2	
A Ecuador	4/3	1/1/1	4	3	
A Qatar	1/7	0/0/3	0	4	
B England	9/2	2/1/0	7	1	
B USA	2/1	1/2/0	5	2	
B Iran	4/7	1/0/2	3	3	
B Wales	1/6	0/1/2	1	4	

(5) 设置菜单，包含如下功能：

1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出

程序开始时，需要先调用 `count()` 函数计算记录个数，然后创建动态数组，再调用 `read()` 函数从文件读入信息以获取全部赛事信息，并通过 `get_team()` 函数获取所有球队的信息（包括球队的进/失球数、胜/平/负局数、积分、排名）；然后显示菜单，根据输入的功能号，调用相应函数；当输入功能号 0（退出）时，需要调用 `save()` 函数将球队的数据写入到文件 `team.txt` 中。

输出结果示例1：

```

1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):1
小组      赛事      比分
A      Qatar:Ecuador    0:2
A      Senegal:Holland  0:2
A      Qatar:Senegal    1:3
A      Holland:Ecuador  1:1
A      Holland:Qatar    2:0
A      Ecuador:Senegal  1:2
B      England:Iran     6:2
B      USA:Wales        1:1
B      Wales:Iran       0:2
B      England:USA     0:0
B      Wales:England   0:3
B      Iran:USA        0:1

```

输出结果示例2：

```

1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):2
小组  球队  进/失  胜/平/负  积分  排名
A  Holland  5/1    2/1/0    7      1
A  Senegal  5/4    2/0/1    6      2
A  Ecuador  4/3    1/1/1    4      3
A  Qatar    1/7    0/0/3    0      4
B  England  9/2    2/1/0    7      1
B  USA      2/1    1/2/0    5      2
B  Iran     4/7    1/0/2    3      3
B  Wales    1/6    0/1/2    1      4

```


输出结果示例3:

```
-----
1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):3
请输入想要查询的球队:USA
-----


| 小组 | 球队  | 进/失 | 胜/平/负 | 积分 | 排名 |
|----|-----|-----|-------|----|----|
| B  | USA | 2/1 | 1/2/0 | 5  | 2  |


-----
1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):4
请输入想要查询的球队:Wales
-----


| 小组 | 赛事            | 比分  |
|----|---------------|-----|
| B  | USA:Wales     | 1:1 |
| B  | Wales:Iran    | 0:2 |
| B  | Wales:England | 0:3 |


```

输出结果示例4:

```
-----
1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):4
请输入想要查询的球队:Qutor
-----
没有找到和该球队相关的赛事
-----
1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):5
输入错误
-----
1. 赛事信息
2. 积分排名
3. 查询球队信息
4. 查询赛事信息
0. 退出
-----
请输入你的选择(0-4):0
谢谢, 再见!
```