演讲比赛流程管理系统

1. 演讲比赛程序需求

1.1 比赛规则

- 学校举行一场演讲比赛，共有12个人参加。比赛共两轮，第一轮为淘汰赛，第二轮为决赛。
- 每名选手都有对应的编号，如10001~10012
- 比赛方式:分组比赛，每组6个人;
- 第一轮分为两个小组，整体按照选手编号进行抽签后顺序演讲。
- 十个评委分别给每名选手打分，去除最高分和最低分，求的平均分为本轮选手的成绩
- 当小组演讲完后，淘汰组内排名最后的三个选手，前三名晋级，进入下一轮的比赛。
- 第二轮为决赛，前三名胜出
- 每轮比赛过后需要显示晋级选手的信息

1.2 程序功能

- 开始演讲比赛:完成整届比赛的流程，每个比赛阶段需要给用户一个提示，用户按任意键后继续下一个阶段
- 查看往届记录I查看之前比赛前三名结果，每次比赛都会记录到文件中，文件用.csv后缀名保存
- 清空比赛记录:将文件中数据清空
- 退出比赛程序:可以退出当前程序

2. 项目创建

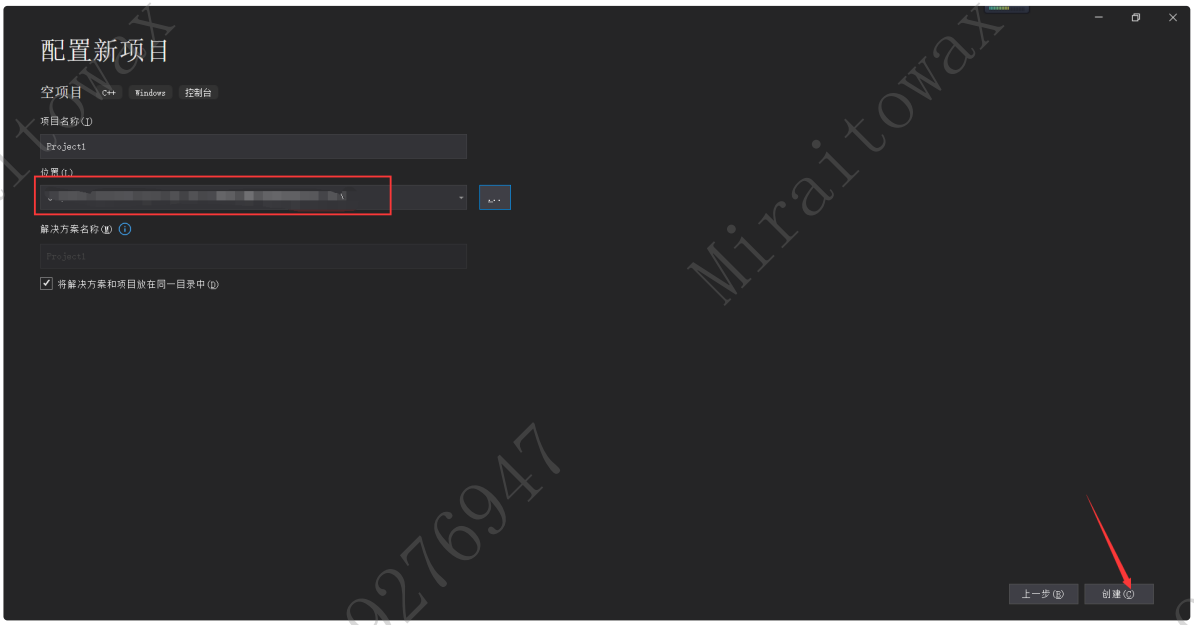
创建项目步骤如下:

- 创建新项目
- 添加文件

2.1 创建项目

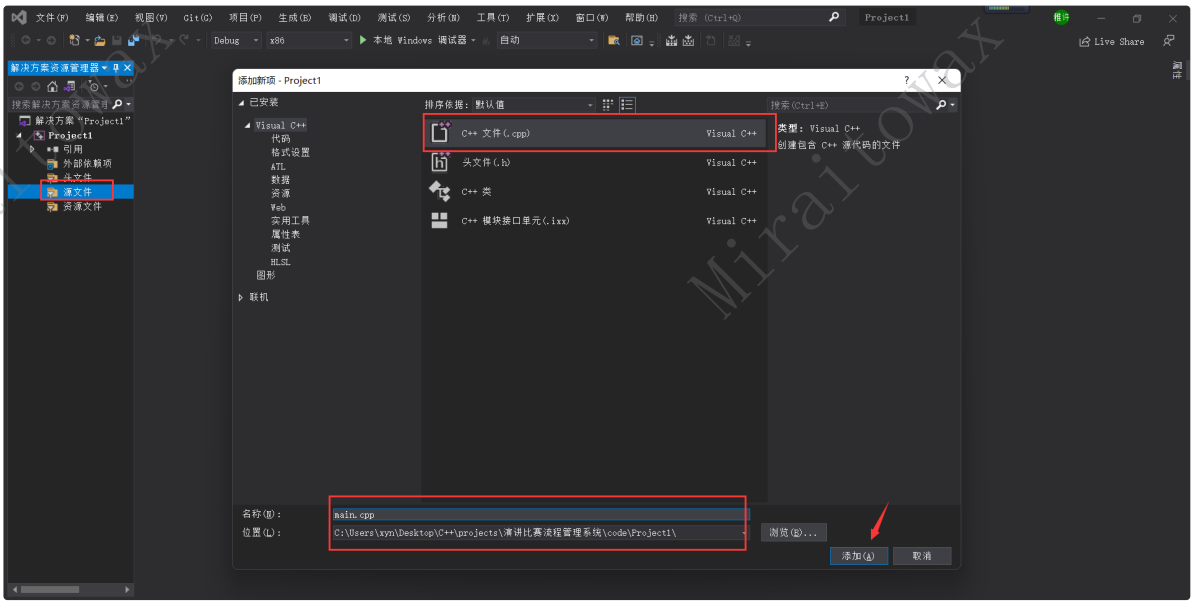
- 打开Visual Studio 2019，点击创建新项目，创建新的C++项目

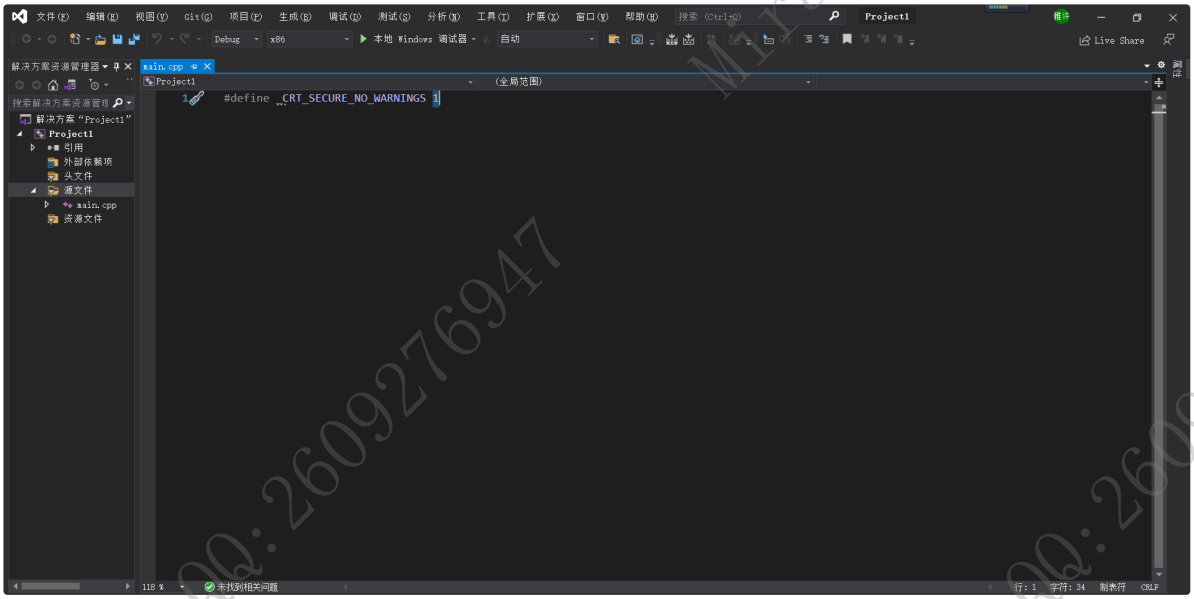
- 填写项目名称以及选取项目路径，点击确定生成项目



2.2 添加文件

- 右键源文件，进行添加文件操作
- 填写文件名称，点击添加





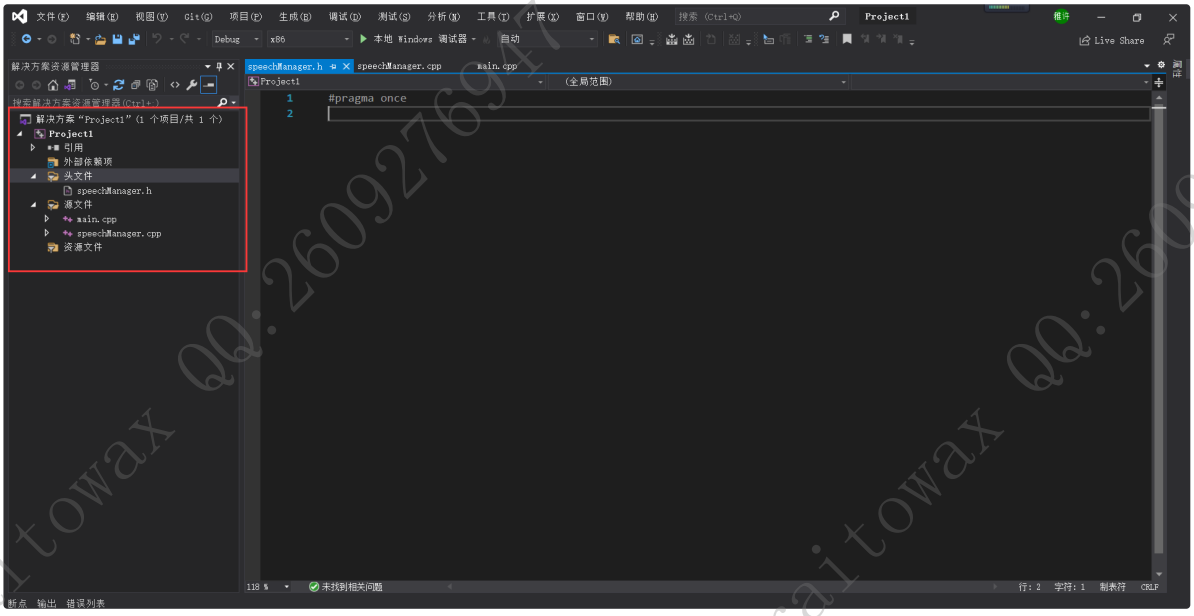
3. 创建管理类

功能描述：

- 提供菜单界面与用户交互
- 对演讲比赛流程进行控制
- 与文件的读写交互

3.1 创建文件

- 在文件和源文件的文件夹下分别创建 `speechManager.h` 和 `speechManager.cpp` 文件



3.2 头文件实现

在 `SpeechManager.h` 中设计管理类

```cpp
#pragma once
#include<iostream>
using namespace std;

//设计演讲管理类
class SpeechManager {
public:
    //构造函数
    SpeechManager();
    //析构函数
    ~SpeechManager();
};
```

3.3 源文件实现

在SpeechManager.cpp中将构造函数与析构函数空实现补全

```cpp
#define _CRT_SECURE_NO_WARNINGS 1
#include"speechManager.h"
//构造函数
SpeechManager::SpeechManager() {

}
//析构函数
SpeechManager::~SpeechManager() {

}
```

4. 菜单功能

功能描述：与用户的沟通界面

4.1 添加成员函数

- 在管理类SpeechManager.h中添加成员函数void show_Menu();

```cpp
//菜单功能
void show_Menu();
```

4.2 菜单功能实现

- 在管理类SpeechManager.cpp中实现show_Menu()函数

```
//菜单功能
void SpeechManager::show_Menu() {
    cout << "********************************************" << endl;
    cout << "************* 欢迎参加演讲比赛 *************" << endl;
    cout << "************* 1. 开始演讲比赛  *************" << endl;
    cout << "************* 2. 查看往届记录  *************" << endl;
    cout << "************* 3. 清空比赛记录  *************" << endl;
    cout << "************* 0. 退出比赛程序  *************" << endl;
    cout << "********************************************" << endl;
    cout << endl;
}
```

4.3 测试代码功能

- 在main.cpp中测试菜单功能

```
//创建管理类对象
SpeechManager sm;
sm.show_Menu();
return 0;
```

5. 退出功能

功能描述：实现退出程序

5.1 提供功能接口

- 在main函数中提供分支选择，提供每个功能接口

```
//退出系统
void exitSystem();
```

5.2 实现退出功能

在SpeechManager.h中提供退出系统成员函数void exitSystem();

在SpeechManager.cpp中提供具体的功能实现

```cpp
//退出系统
void SpeechManager::exitSystem() {
    cout << "欢迎下次使用" << endl;
    system("pause");
    exit(0);
}
```

5.3 测试功能

在main函数分支0选项中，调用退出程序的接口

```cpp
sm.exitSystem();
```

6. 演讲比赛功能

6.1 功能分析

比赛流程分析：

抽签→开始演讲比赛→显示第一轮比赛结果→抽签→开始演讲比赛→显示前三名结果→保存分数

6.2 创建选手类

- 选手类的属性包含：选手姓名、分数
- 头文件中创建`speaker.h`文件，并添加代码

```cpp
#pragma once
#include<iostream>
using namespace std;

//选手类
class Speaker {
public:
    string m_Name;
    double m_Score[2];//分数 最多两轮得分
};
```

6.3 比赛

6.3.1 成员属性添加

- 在SpeechManager.h中添加属性

```
//成员属性
//保存第一轮比赛选手编号容器
vector<int>v1;
//第一轮晋级选手编号容器
vector<int>v2;
//胜出前三名选手编号容器
vector<int>vVectory;
//存放编号以及对应具体选手容器
map<int, Speaker>m_Speaker;
//存放比赛轮数
int m_Index;
```

6.3.2 初始化属性

- 在SpeechManager.h中提供开始比赛的成员函数void initSpeech();

```
//初始化容器和属性
void initSpeech();
```

- 在SpeechManager.cpp中实现void initSpeech();

```
//初始化容器和属性
void SpeechManager::initSpeech() {
    //容器都置空
    this->v1.clear();
    this->v2.clear();
    this->vVectory.clear();
    this->m_Speaker.clear();
    //初始化比赛轮数
    this->m_Index = 1;
}
```

- SpeechManager构造函数中调用void initSpeech();

```cpp
SpeechManager::SpeechManager() {
    //初始化容器和属性
    this->initSpeech();
}
```

6.3.3 创建选手

- 在speechManager.h中提供开始比赛的成员函数void createSpeaker();

```cpp
//创建12名选手
void createSpeaker();
```

- 在speechManager.cpp中实现void createSpeaker();

```cpp
//创建12名选手
void SpeechManager::createSpeaker() {
    string nameSeed = "ABCDEFGHIJKL";
    for (int i = 0; i < nameSeed.size(); i++) {
        string name = "选手";
        name += nameSeed[i];
        Speaker sp;
        sp.m_Name = name;
        //两轮得分均赋为0
        for (int j = 0; j < 2; j++) {
            sp.m_Score[j] = 0;
        }
        //创建选手编号 并且放入到v1容器中
        this->v1.push_back(i + 10001);
        //选手编号以及对应选手 放入到map容器中
        this->m_Speaker.insert(make_pair(i + 10001, sp));
    }
}
```

- SpeechManager类的构造函数中调用void createSpeaker();

```cpp
//创建12名选手
this->createSpeaker();
```

- 测试在main函数中，可以创建完管理对象后，使用下列代码测试12名选手的初始状态

```cpp
//测试12名选手创建
for (map<int, Speaker>::iterator it = sm.m_Speaker.begin(); it != sm.m_Speaker.end(); it++) {
    cout << "选手编号: " << it->first << " " << "姓名: " << it->second.m_Name << " " << "分数: " << it->second.m_Score[0] << endl;
}
```

6.3.4 开始比赛成员函数添加

- 在 SpeechManager 中提供开始比赛的成员函数 void startSpeech();
- 该函数功能是主要控制比赛的流程

```cpp
//开始比赛 比赛整个流程控制函数
void startSpeech();
```

- 在 SpeechManager.cpp 中将 startSpeech 的空实现先写入
- 将整个比赛的流程写到函数中

```cpp
//开始比赛 比赛整个流程控制函数
void SpeechManager::startSpeech() {
    //第一轮开始比赛
    //1. 抽签

    //2. 比赛

    //3. 显示晋级结果

    //第二轮开始比赛
    //1. 抽签

    //2. 比赛

    //3. 显示最终结果

    //4. 保存分数到文件中
}
```

6.3.5 抽签

功能描述：

- 正式比赛前，所有选手的比赛顺序需要打乱，只需要将存放选手编号的容器打乱次序即可
- 在SpeechManager.h中提供抽签的成员函数void speechDraw();

```
//抽签
void speechDraw();
```

- 在SpeechManager.cpp中实现成员函数void speechDraw();

```
//抽签
void SpeechManager::speechDraw() {
    cout << "第 <<" << this->m_Index << ">> 轮选手正在抽签" << endl;
    cout << "---------------------------------------" << endl;
    cout << "抽签后的演讲顺序如下: " << endl;
    if (this->m_Index == 1) {
        //第一轮比赛
        random_shuffle(v1.begin(), v1.end());
        for (vector<int>::iterator it = v1.begin(); it != v1.end(); it++)
{
            cout << *it << " ";
        }
        cout << endl;
    }
    else {
        //第一轮比赛
        random_shuffle(v2.begin(), v2.end());
        for (vector<int>::iterator it = v2.begin(); it != v2.end(); it++)
{
            cout << *it << " ";
        }
        cout << endl;
    }
    cout << "---------------------------------------" << endl;
    system("pause");
    cout << endl;
}
```

- 在main比赛控制流程中，调用抽签函数

```
sm.startSpeech();
```

6.3.6 开始比赛

- 在speechManager.h中提供比赛的成员函数void speechManager();

```
//开始比赛
void speechContest();
```

- 在speechManager.cpp中实现成员函数void speechContest();

```
//开始比赛
void SpeechManager::speechContest() {
    cout << "-------------第" << this->m_Index << "轮比赛开始--------------
" << endl;
    //准备临时容器存放小组成绩
    multimap<double, int, greater<double>>groupScore;
    int num = 0;//记录人员的个数 6人一组
    vector<int>v_Src;//比赛选手容器
    if (this->m_Index == 1) {
        v_Src = v1;
    }
    else {
        v_Src = v2;
    }
    //遍历所有选手进行比赛
    for (vector<int>::iterator it = v_Src.begin(); it != v_Src.end();
it++) {
        num++;
        //评委打分
        deque<double>d;
        for (int i = 0; i < 10; i++) {
            double score = (rand() % 401 + 600) / 10.f;//60~100
            d.push_back(score);
            //cout << score << " ";
        }
```

```
        //cout << endl;
        sort(d.begin(), d.end(), greater<double>());
        d.pop_front();//去除最高分
        d.pop_back();//去除最低分
        double sum = accumulate(d.begin(), d.end(), 0.0f);
        double avg = sum / (double)d.size();//平均分
        ////打印平均分
        //cout << "编号: " << *it << " " << "姓名: " << this->m_Speaker[*it].m_Name << " " << "平均分: " << avg << endl;
        //将平均分放入到map容器中
        this->m_Speaker[*it].m_Score[this->m_Index - 1] = avg;
        //将打分数据放入临时小组容器中
        groupScore.insert(make_pair(avg, *it));//key是得分 value是具体选手编号
        //每六人取出前三名
        if (num % 6 == 0) {
            cout << "第" << num / 6 << "小组比赛名次如下: " << endl;
            for (multimap<double, int, greater<double>>::iterator it = groupScore.begin(); it != groupScore.end(); it++) {
                cout << "编号: " << it->second << " " << "姓名: " << this->m_Speaker[(*it).second].m_Name << " " << "成绩: " << this->m_Speaker[it->second].m_Score[this->m_Index - 1] << endl;
            }
            //取走前三名
            int count = 0;
            for (multimap<double, int, greater<double>>::iterator it = groupScore.begin(); it != groupScore.end() && count < 3; it++,count++) {
                if (this->m_Index == 1) {
                    v2.push_back((*it).second);
                }
                else {
                    vVectory.push_back((*it).second);
                }
            }
            groupScore.clear();
```

```
            cout << endl;
        }
    }
    cout << "-------------第" << this->m_Index << "轮比赛完毕---------------
" << endl;
    system("pause");
}
```

- 测试 在`startSpeech`函数中添加

```
//2. 比赛
this->speechContest();
```

6.3.7 显示比赛分数

- 在`speechManager.h`中提供显示分数的成员函数`void showScore();`

```
//显示得分
void showScore();
```

- 在`speechManager.cpp`中实现成员函数`void showScore();`

```
//显示得分
void SpeechManager::showScore() {
    cout << "-------------第" << this->m_Index << "轮比晋级选手: ------------
---" << endl;
    vector<int>v;
    if (this->m_Index == 1) {
        v = v2;
    }
    else {
        v = vVectory;
    }
    for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
        cout << "选手编号: " << *it << " " << "姓名: " << this-
>m_Speaker[*it].m_Name << " " << "得分: " << this-
>m_Speaker[*it].m_Score[this->m_Index - 1] << endl;
    }
    cout << endl;
```

```
    system("pause");
    system("cls");
    this->show_Menu();
}
```

- 在 `startSpeech` 比赛流程控制函数中，调用显示比赛分数函数

```
//3. 显示晋级结果
this->showScore();
```

6.3.8 第二轮比赛

第二轮比赛流程同第一轮，只是比赛的轮+1，其余流程不变

- 在 `startSpeech` 比赛流程控制的函数中，加入第二轮的流程

```
//开始比赛 比赛整个流程控制函数
void SpeechManager::startSpeech() {
    //第一轮开始比赛
    //1. 抽签
    this->speechDraw();
    //2. 比赛
    this->speechContest();
    //3. 显示晋级结果
    this->showScore();
    //第二轮开始比赛
    this->m_Index++;
    //1. 抽签
    this->speechDraw();
    //2. 比赛
    this->speechContest();
    //3. 显示最终结果
    this->showScore();
    //4. 保存分数到文件中
}
```

6.4 保存分数

功能描述：

- 将每次演讲比赛的得分记录到文件中

功能实现：

- 在speechManager.h中国添加保存记录的成员函数void saveRecord();

```
//保存分数
void saveRecord();
```

- 在speechManager.cpp中实现成员函数void saveRecord();

```
//保存分数
void SpeechManager::saveRecord() {
    ofstream ofs;
    ofs.open("speech.csv", ios::out, ios::app);//用追加的方式写文件
    //将每个选手的数据写入到文件中
    for (vector<int>::iterator it = vVectory.begin(); it ≠ vVectory.end(); it++) {
        ofs << *it << "," << this→m_Speaker[*it].m_Score[1] << ",";
    }
    ofs << endl;
    //关闭文件
    ofs.close();
    cout << "记录已经保存" << endl;
}
```

- 在startSpeech比赛流程控制函数中，调用函数

```
//4. 保存分数到文件中
this→saveRecord();
cout << "本届比赛完毕" << endl;
system("pause");
system("cls");
```

7. 查看记录

7.1 读取记录分数

- 在speechManager.h中添加保存记录的成员函数void loadRecord();
- 添加判断文件是否为空的标志bool fileIsEmpty;
- 添加往届记录的容器map<int,vector<string>m_Record>;
- 其中m_Record中key代表第几届，value记录具体的信息

```
//读取记录
void loadRecord();
//判断文件是否为空
bool fileIsEmpty;
//存放往届记录的容器
map<int, vector<string>>m_Record;
```

- 在speechManager.cpp中实现成员函数void loadRecord();

```
//读取记录
void SpeechManager::loadRecord() {
    ifstream ifs("speech.csv", ios::in);//读文件
    if (!ifs.is_open()) {
        this->fileIsEmpty = true;
        cout << "文件不存在" << endl;
        ifs.close();
        return;
    }
    char ch;
    ifs >> ch;
    if (ifs.eof()) {
        cout << "文件为空" << endl;
        this->fileIsEmpty = true;
        ifs.close();
        return;
    }
    //文件不为空
    this->fileIsEmpty = false;
    ifs.putback(ch);//将上面读取的单个字符放回来

    string data;
    int index = 0;
```

```
    while (ifs >> data) {
        //cout << data << endl;
        vector<string>v;//存放6个string字符串
        int pos = -1;
        int start = 0;
        while (true) {
            pos = data.find(",", start);
            if (pos == -1) {
                //没有找到
                break;
            }
            string temp = data.substr(start, pos - start);
            //cout << temp << endl;
            v.push_back(temp);
            start = pos + 1;
        }
        this->m_Record.insert(make_pair(index, v));
        index++;
    }
    ifs.close();
    //for (map<int, vector<string>>::iterator it = m_Record.begin(); it != m_Record.end(); it++) {
    //  cout << it->first << " " << "冠军编号: " << it->second[0] << " " << "分数:" << it->second[1] << endl;
    //}
}
```

- 在 `speechManager` 构造函数中调用获取往届记录函数

```
//加载往届记录
this->loadRecord();
```

7.2 查看记录功能

- 在 `speechManager.h` 中添加保存记录的成员函数 `void showRecord();`

```
//显示往届记录
void showRecord();
```

- 在speechManager.cpp中实现成员函数void showRecord();

```
//显示往届记录
void SpeechManager::showRecord() {
    for (int i = 0; i < this->m_Record.size(); i++) {
        cout << "第" << i + 1 << "届" << " "
            << "冠军编号: " << this->m_Record[i][0] << "得分: " << this->m_Record[i][1] << " "
            << "亚军编号: " << this->m_Record[i][2] << "得分: " << this->m_Record[i][3] << " "
            << "季军编号: " << this->m_Record[i][3] << "得分: " << this->m_Record[i][4] << endl;
    }
    system("pause");
    system("cls");
}
```

7.3 bug解决

1. 查看往届记录，若文件不存在或为空，并未表示

解决方式：在showRecord函数中，开始判断文件状态并加以判断

```
//显示往届记录
void SpeechManager::showRecord() {
    if (this->fileIsEmpty) {
        cout << "文件为空或者文件不存在" << endl;
    }
    else {
        for (int i = 0; i < this->m_Record.size(); i++) {
            cout << "第" << i + 1 << "届" << " "
                << "冠军编号: " << this->m_Record[i][0] << "得分: " << this->m_Record[i][1] << " "
                << "亚军编号: " << this->m_Record[i][2] << "得分: " << this->m_Record[i][3] << " "
```

```
                << "季军编号: " << this->m_Record[i][3] << "得分: " << this->m_Record[i][4] << endl;
            }
        }
        system("pause");
        system("cls");
    }
```

2. 若记录为空或者不存在，比完赛后仍然提示记录为空

解决方式： saveRecord 中更新文件为空的标志

```
//更改文件的状态
this->fileIsEmpty = false;
```

3. 比完赛后查不到本届比赛的记录，没有实时更新

解决方式：比赛完毕之后，所有数据重置

在 startSpeech 函数中，添加以下代码

```
//重置比赛 获取记录
//初始化容器和属性
this->initSpeech();
//创建12名选手
this->createSpeaker();
//加载往届记录
```

4. 初始化时，并没有初始化记录容器

解决方式： initSpeech 中添加初始化记录容器

```
//将记录的容器清空
this->m_Record.clear();
```

5. 每次记录都是一样的

```
//随机数种子
srand((unsigned int)time(NULL));
```

8. 清空记录

8.1 清空记录功能实现

- 在`speechManager.h`中添加保存记录的成员函数`void clearRecord();`

```
//清空记录
void clearRecord();
```

- 在`speechManager.cpp`中实现成员函数`void clearRecord();`

```
/清空记录
void SpeechManager::clearRecord() {
    cout << "是否确定清空文件? " << endl;
    cout << "1、是" << endl;
    cout << "2、否" << endl;
    int select = 0;
    cin >> select;
    if (select == 1) {
        //确认清空
        ofstream ofs("speech.csv", ios::trunc);
        ofs.close();
        //初始化容器和属性
        this->initSpeech();
        //创建12名选手
        this->createSpeaker();
       //加载往届记录
        this->loadRecord();
        cout << "清空成功" << endl;
    }
    system("pause");
    system("cls");
}
```

- 至此本案例