# 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101~112
- 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
- 10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：完成整界比赛的流程，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：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：将文件数据清空
- 退出比赛程序：可以退出当前程序

## 1.3 程序效果图

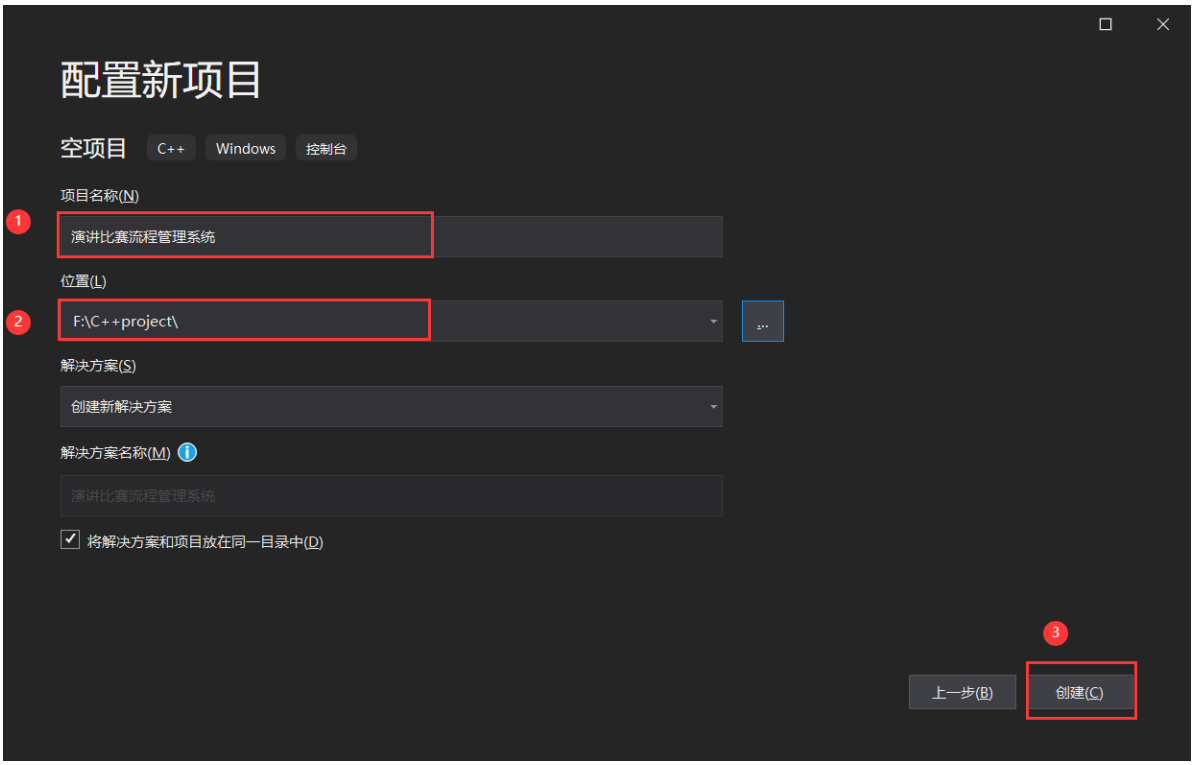
# 2、项目创建

创建项目步骤如下：

- 创建新项目
- 添加文件

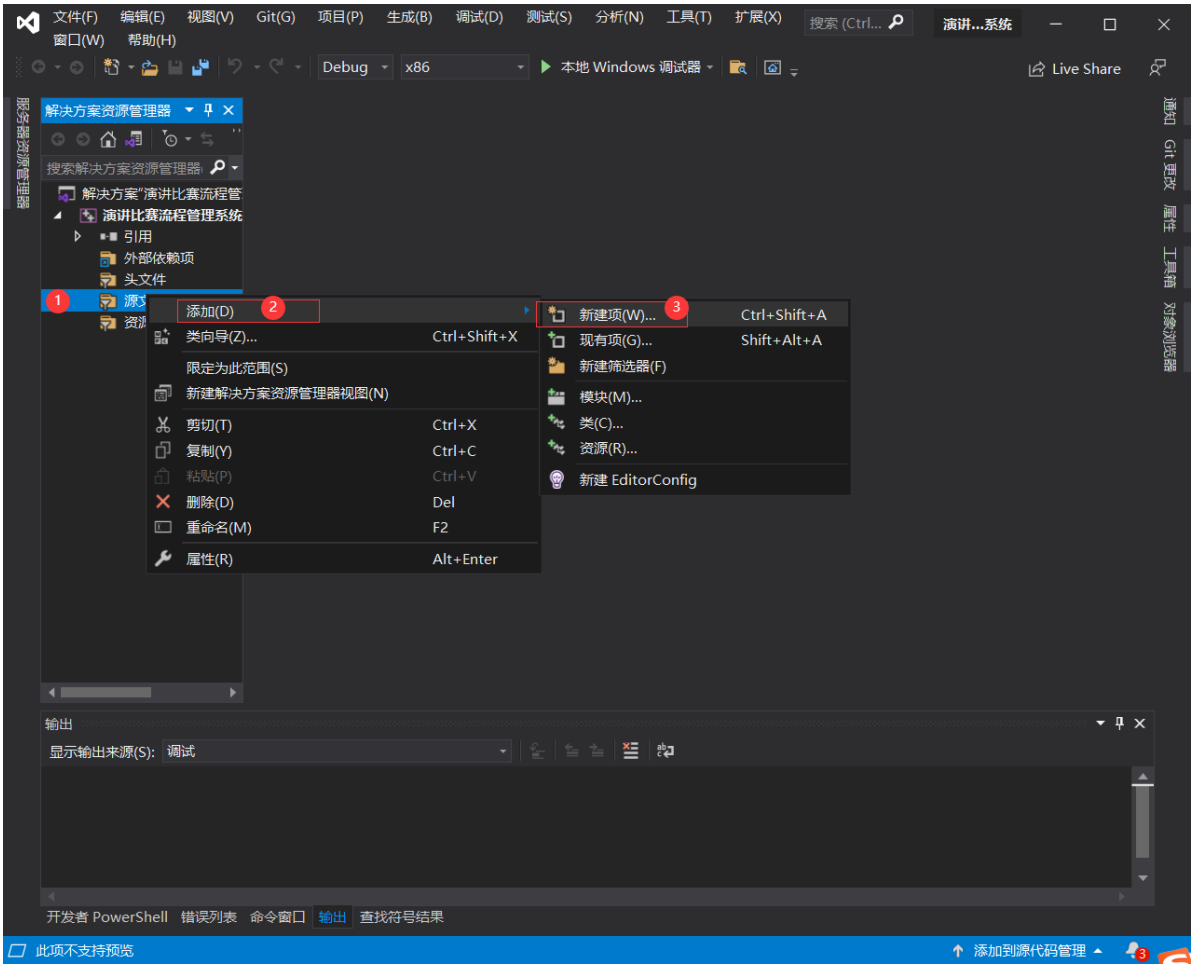
## 2.1 创建项目

- 打开VS2019点击创建新项目

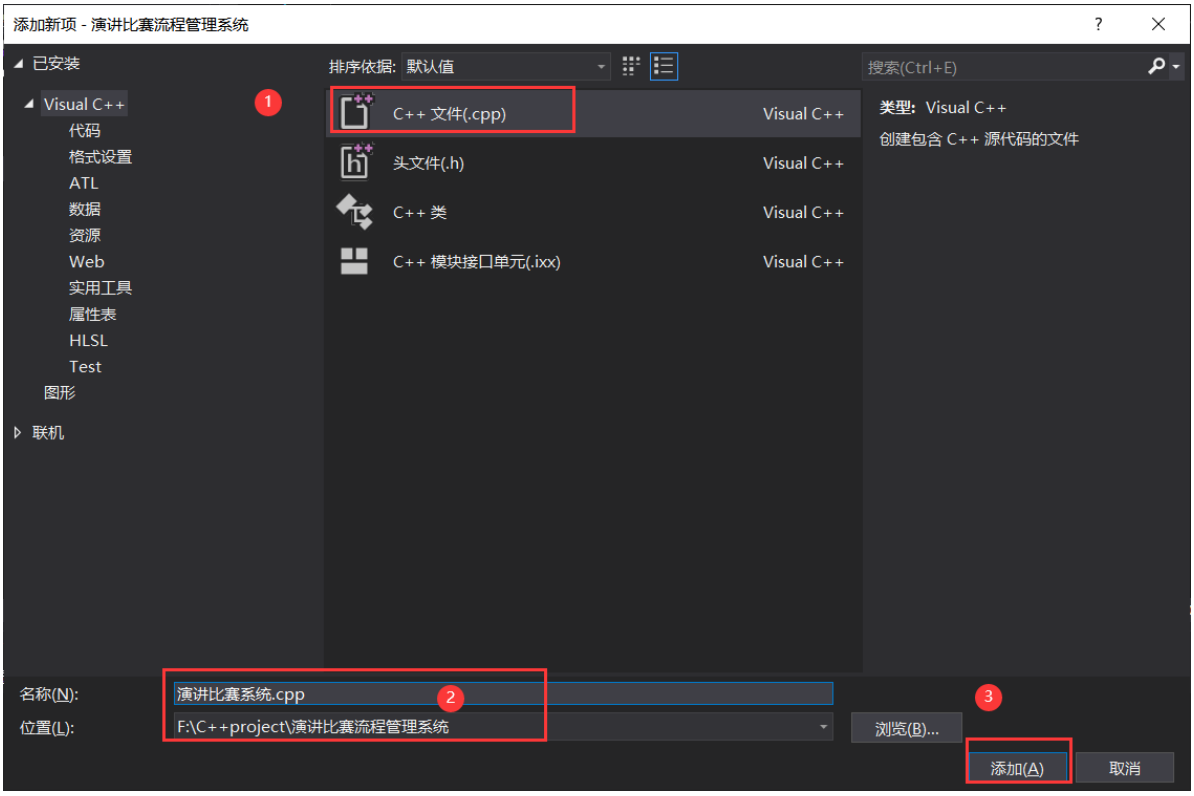



## 2.2 添加文件

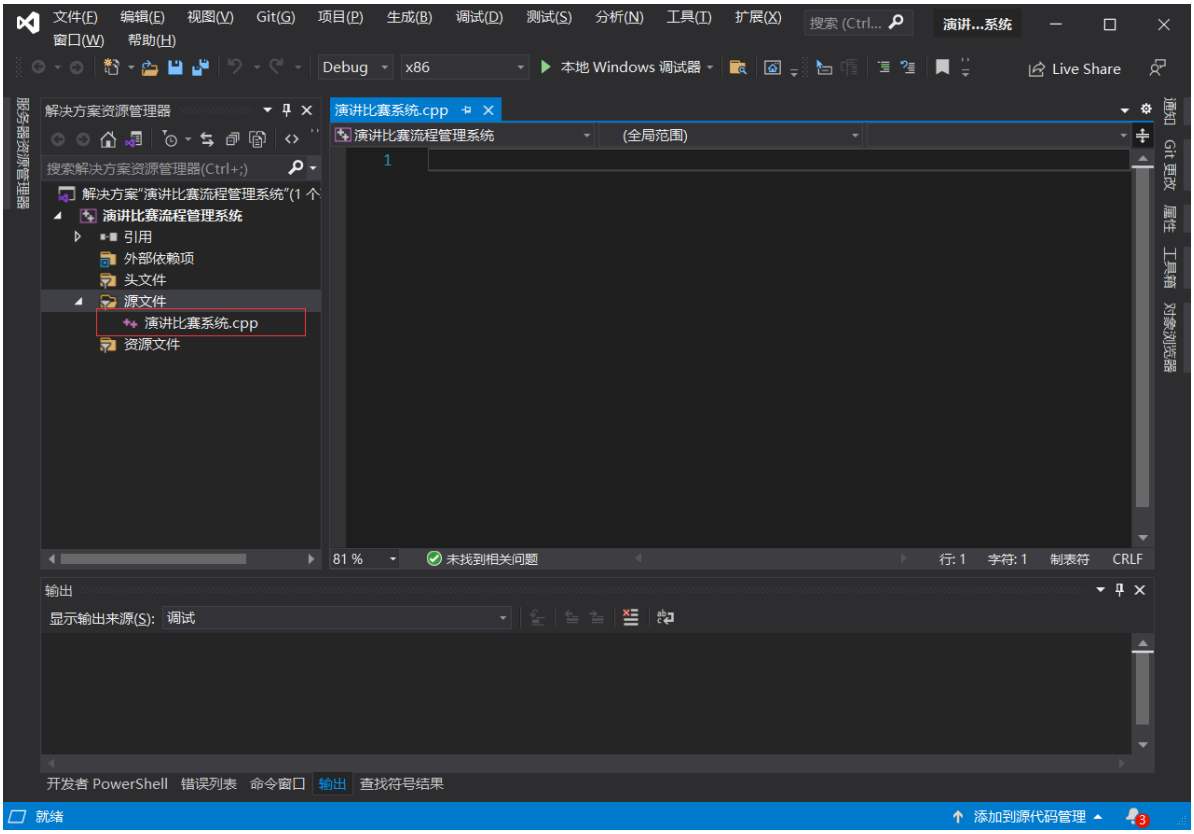
1、右键点击源文件，进行添加文件操作




2、填写文件名称和路径，点击添加




3、生成文件，效果如下




至此，项目已创建完毕

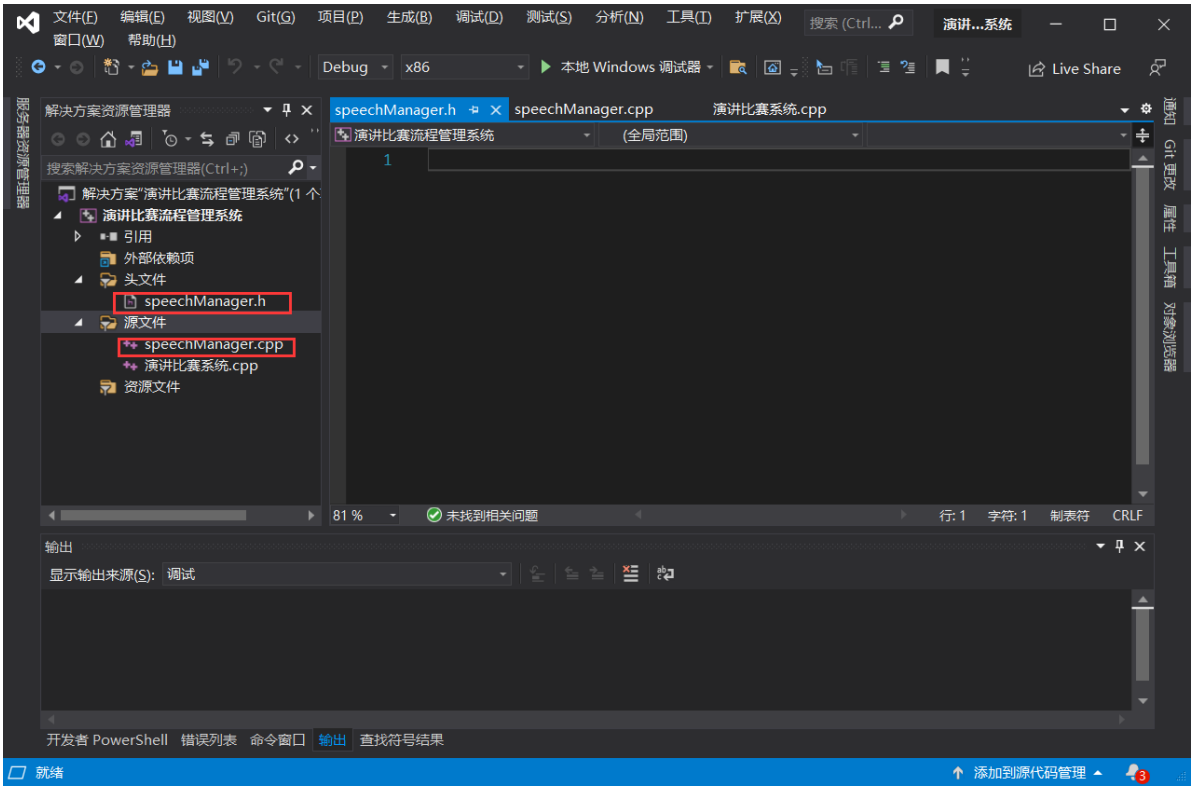
# 3、创建管理类

**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

- 在头文件和源文件的文件夹下分别创建speechManager.h 和 speechManger.cpp 文件




## 3.2 头文件实现

在speechManager.h中设计管理类

代码如下：

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

//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中将构造和析构函数空实现补全

```
#include"speechManager.h"

//构造函数
SpeechManager::SpeechManager()
{

}

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

}
```

至此演讲管理类已创建完毕

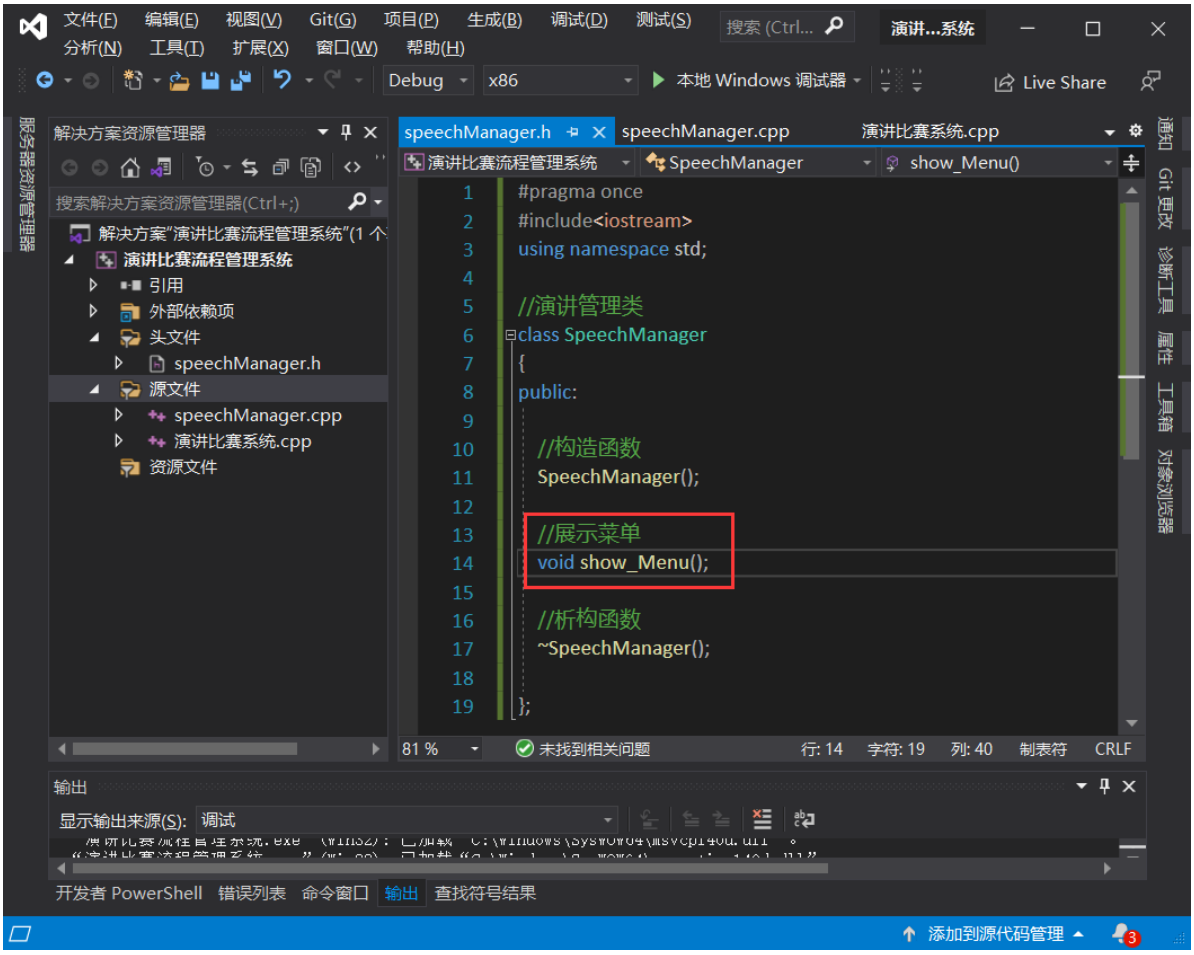
# 4、 菜单功能

**功能描述：**

- 实现与用户沟通界面

## 4.1 添加成员函数

在管理类speechManager.h中添加成员函数 void showMenu();



## 4.2 菜单功能实现

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

```
//展示菜单
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;
}
```

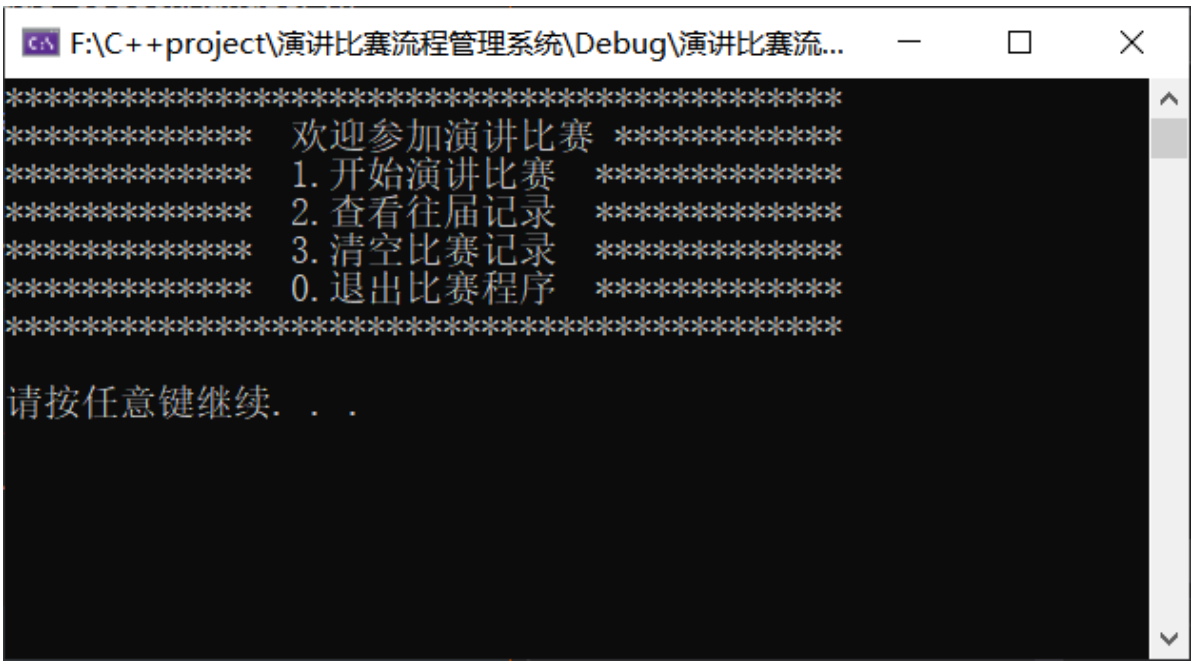
## 4.3 测试菜单功能

- 在演讲比赛流程管理系统.cpp 中测试菜单功能

代码如下：

```
#include<iostream>
using namespace std;
#include"speechManager.h"

int main()
{
    SpeechManager sm;
    sm.show_Menu();

    system("pause");
    return 0;
}
```

- 运行效果如图：




菜单界面搭建完毕

# 5、 退出功能

## 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

**示例代码：**

```
#include<iostream>
using namespace std;
#include"speechManager.h"

int main()
{
    SpeechManager sm;

    sm.show_Menu();

    int choice = 0;  //接收用户选择

    while (true)
    {
        sm.show_Menu();

        cout << "请输入你的选择：" << endl;
        cin >> choice;
        switch (choice)
        {
        case 1: //开始比赛
            break;
```

```
        case 2://查看记录
            break;

        case 3://清空记录
            break;

        case 0://退出系统
            break;

        default:system("cls"); //清屏
            break;
        }
    }

    system("pause");
    return 0;
}
```

## 5.2 实现退出功能

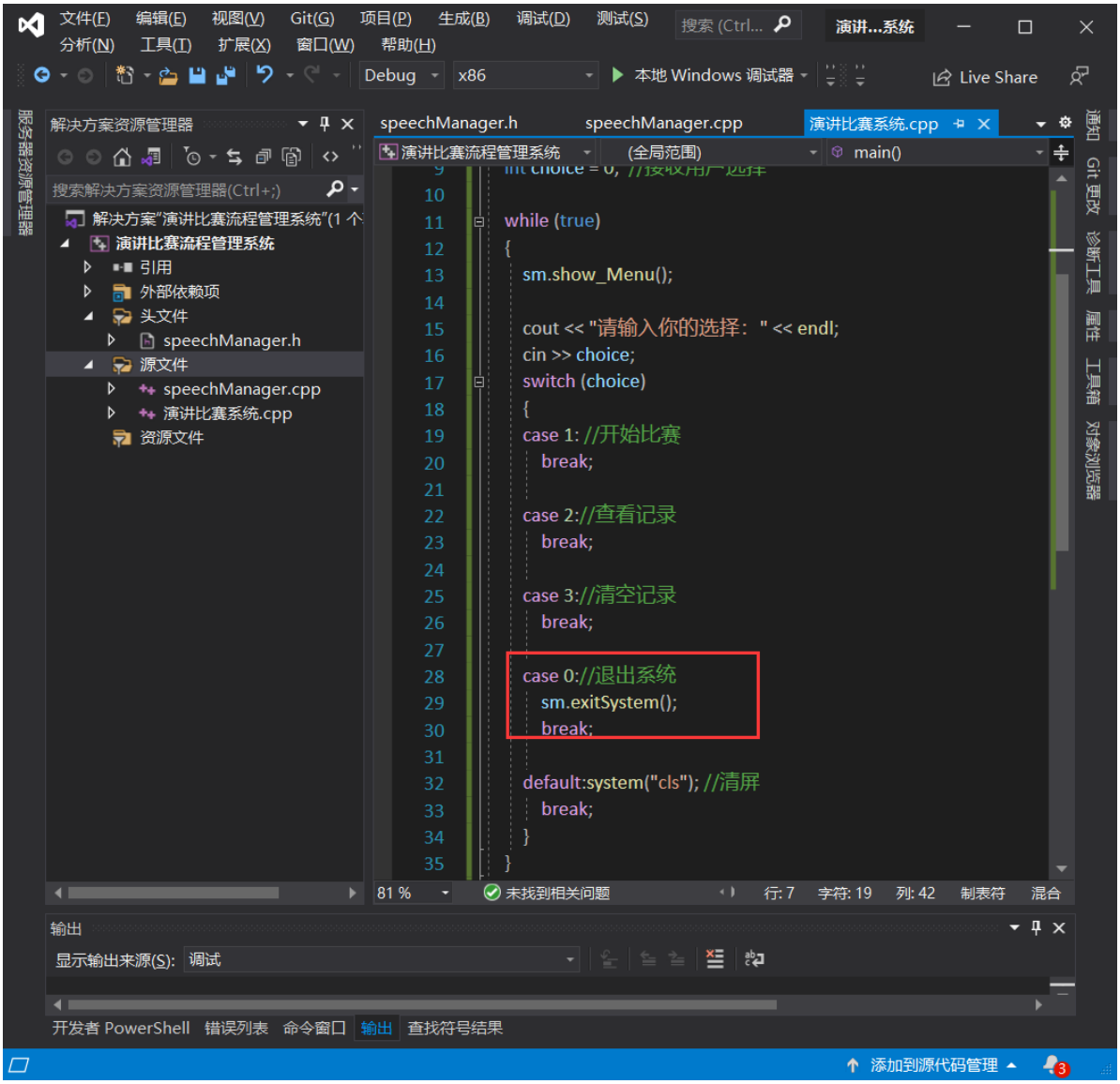
在speechManager.h中提供退出系统的成员函数：void exitSystem();

在speechManager.cpp中提供具体的功能实现：

```
//退出函数实现
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，跳跃退出程序接口




运行测试效果如图：


## 6、 演讲比赛功能

## 6.1 功能分析

比赛分析：

抽签 -> 开始比赛 -> 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

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

```
#pragma onec  //防止头文件重复定义
#include<iostream>
using namespace std;

//选手类
class Speaker
{
public:

    string m_name;  //姓名
    double m_Score[2];  //分数  最多有两轮得分

};
```

## 6.3 比赛

### 6.3.1 成员属性添加

- 在speechManager.h 中添加属性

```
        //可以采用容器嵌套技术
        //成员属性
        //保存第一轮比赛选手编号容器 12人
        vector<int>v1;

        //保存第一轮晋级选手编号容器 6人
        vector<int>v2;

        //胜出前三名选手编号容器 3 人
        vector<int>vVictory;

        //存放编号，以及对应具体学生容器
        map<int, Speaker>m_Speaker;

        //存放比赛轮数
        int m_Index;
```

"const_pointer": 不是 "std::allocator_traits<_Alloc>" 的成员

原因：容器调用错误把map写成vector

### 6.3.2 初始化属性

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

```
//容器初始化操作
void initSpeech();
```

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

```
void SpeechManager::initSpeech()
{
    //容器都置空
    this->v1.clear();
    this->v2.clear();
    this->vVictory.clear();
    this->m_Speaker.clear();
    this->m_Index = 1;
}
```

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

```
//构造函数
SpeechManager::SpeechManager()
{
    //容器初始化操作
    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();

```
//初始化创建12名选手
void createSpeaker();
```

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

```
//初始化创建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;
```

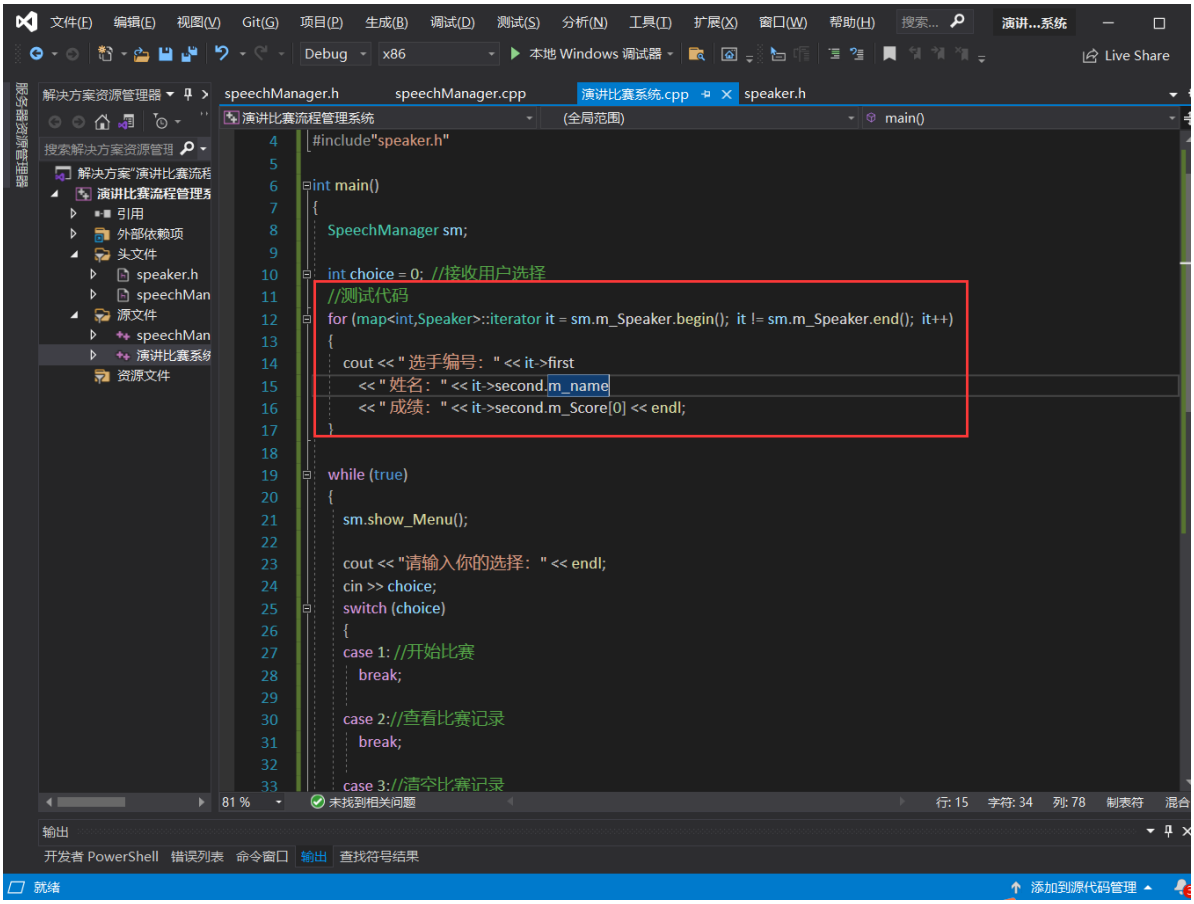
```
        for (int i = 0; i < 2; i++)
        {
            sp.m_Score[i] = 0;
        }

        //12名选手编号
        this->v1.push_back(i + 1001);

        //选手编号   以及对应的选手 存放到map容器中
        this->m_Speaker.insert(make_pair(i + 1001, sp));
    }
}
```

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

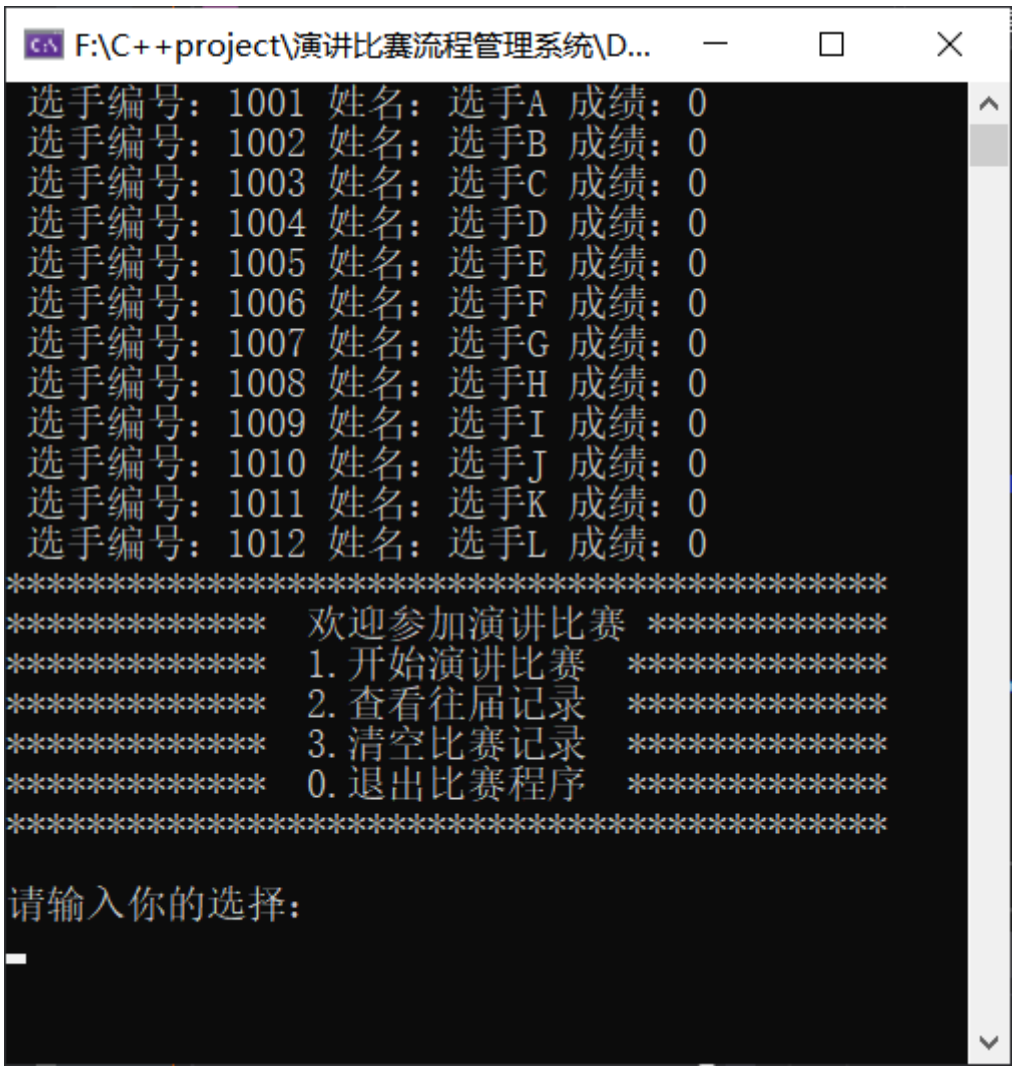
```
//构造函数
SpeechManager::SpeechManager()
{
    //容器初始化操作
    this->initSpeech();

    //创建选手
    this->createSpeaker();
}
```

- 测试程序，在main函数中，可以在创建完管理对象后，使用下列代码测试12名选手初始状态。

```
    //测试代码
    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
            << " 成绩： " << it->second.m_Score[0] << endl;
    }
```


#include"speaker.h"
int main()
{
SpeechManager sm;
int choice = 0; //接收用户选择
//测试代码
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
<< " 成绩： " << it->second.m_Score[0] << endl;
}
while (true)
{
sm.show_Menu();
cout << "请输入你的选择： " << endl;
cin >> choice;
switch (choice)
{
case 1: //开始比赛
break;
case 2://查看比赛记录
break;
case 3://清空比赛记录

- 测试效果如图所示：




测试完毕，符合预期效果，可以把测试代码删除或者注释起来。

### 6.3.4 开始比赛成员函数添加

- 在speechManager.h 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

```
        //开始比赛 - 比赛流程控制
        void startSpeech();
```

- 在speechManager.cpp中将startSpeech的空实现先写入
- 我们可以先将整个比赛的流程写到函数中

```
//开始比赛 - 比赛流程控制
void startSpeech()
{
    //第一轮比赛
    //1、抽签

    //2、比赛

    //3、显示晋级结果

    //第二轮比赛

```

```
    //1、抽签

    //2、比赛

    //3、显示最后结果

    //4、保存分数

}
```

### 6.3.5 抽签

**功能描述：**

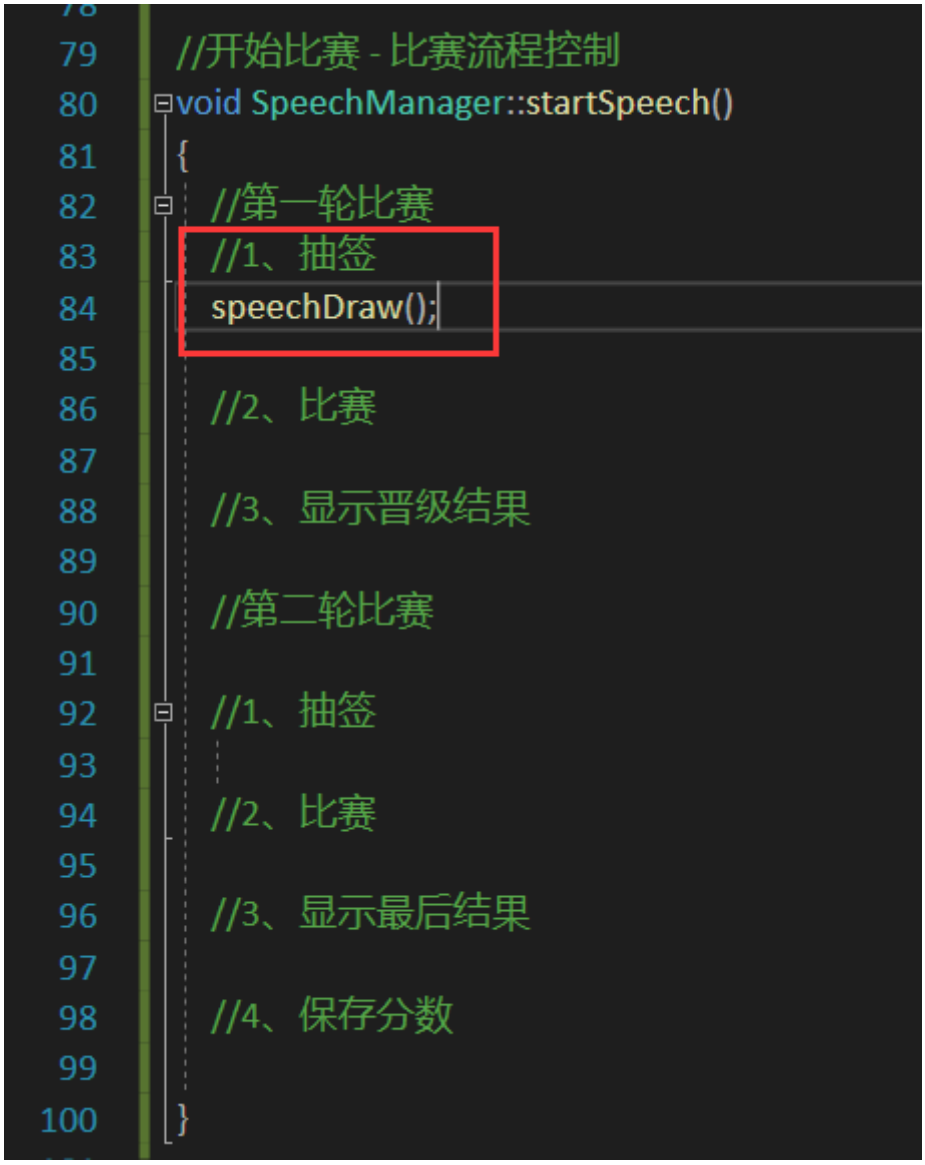
- 正式比赛前，所有选手的比赛顺序需要打乱，我们只需要将存放选手编号的容器再次打乱次序即可。
- 在speechManager.h 中提供抽签的成员函数 void speechDraw();

```
        //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 << " >> 轮比赛选手正在抽签" << endl;
    cout << " ----------------------------------------------------"<<endl;
    cout << " 抽签后以及顺序如下： " << endl;
    if (this->m_Index == 1)
    {
        random_shuffle(v1.begin(),v1.end());
        for (vector<int>::iterator it = v1.begin(); it != v1.end(); it++)
        {
            cout << *it << " ";
        }
        cout << endl;
    }
    else
    {
        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startSpeech比赛流程控制函数中，调用抽签函数

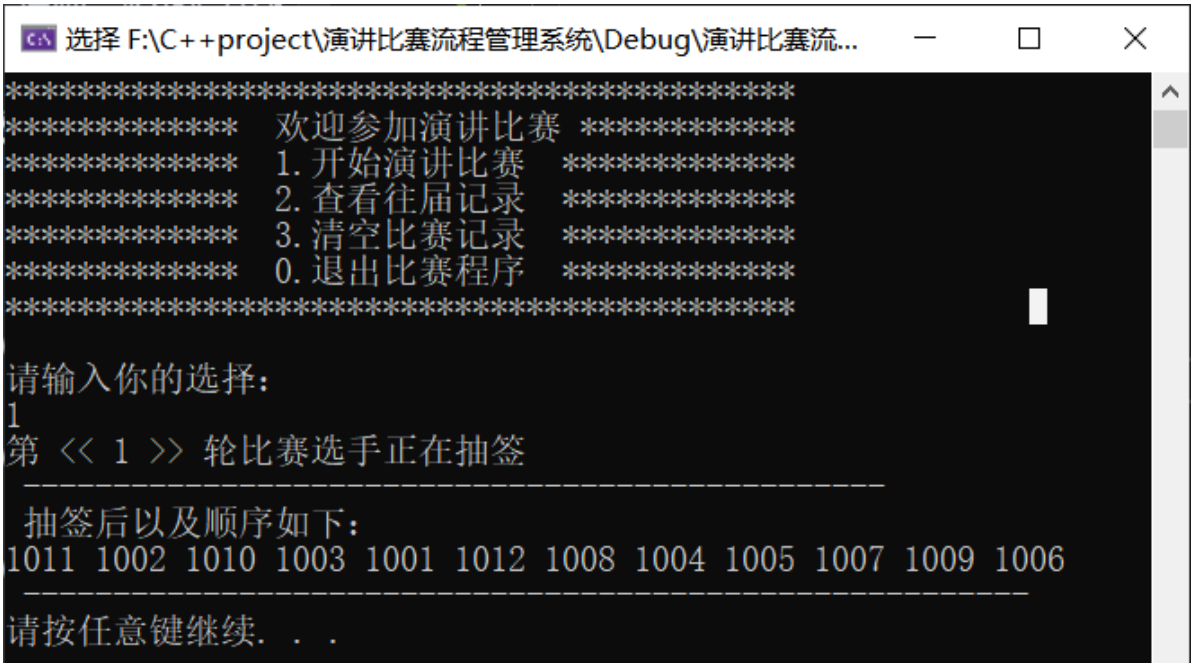


```
79  //开始比赛 - 比赛流程控制
80  void SpeechManager::startSpeech()
81  {
82      //第一轮比赛
83      //1、抽签
84      speechDraw();
85
86      //2、比赛
87
88      //3、显示晋级结果
89
90      //第二轮比赛
91
92      //1、抽签
93
94      //2、比赛
95
96      //3、显示最后结果
97
98      //4、保存分数
99
100 }
```

- 在main函数中，分支1选项中，调用开始比赛的接口
- 测试

```
while (true)
{
    sm.show_Menu();

    cout << "请输入你的选择： " << endl;
    cin >> choice;
    switch (choice)
    {
    case 1: //开始比赛
        sm.startSpeech();
        break;

    case 2://查看比赛记录
        break;

    case 3://清空比赛记录
        break;

    case 0://退出系统
        sm.exitSystem();
        break;
```

- 测试结果



### 6.3.6 开始比赛

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

```
    //比赛
    void speechContest();
```

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

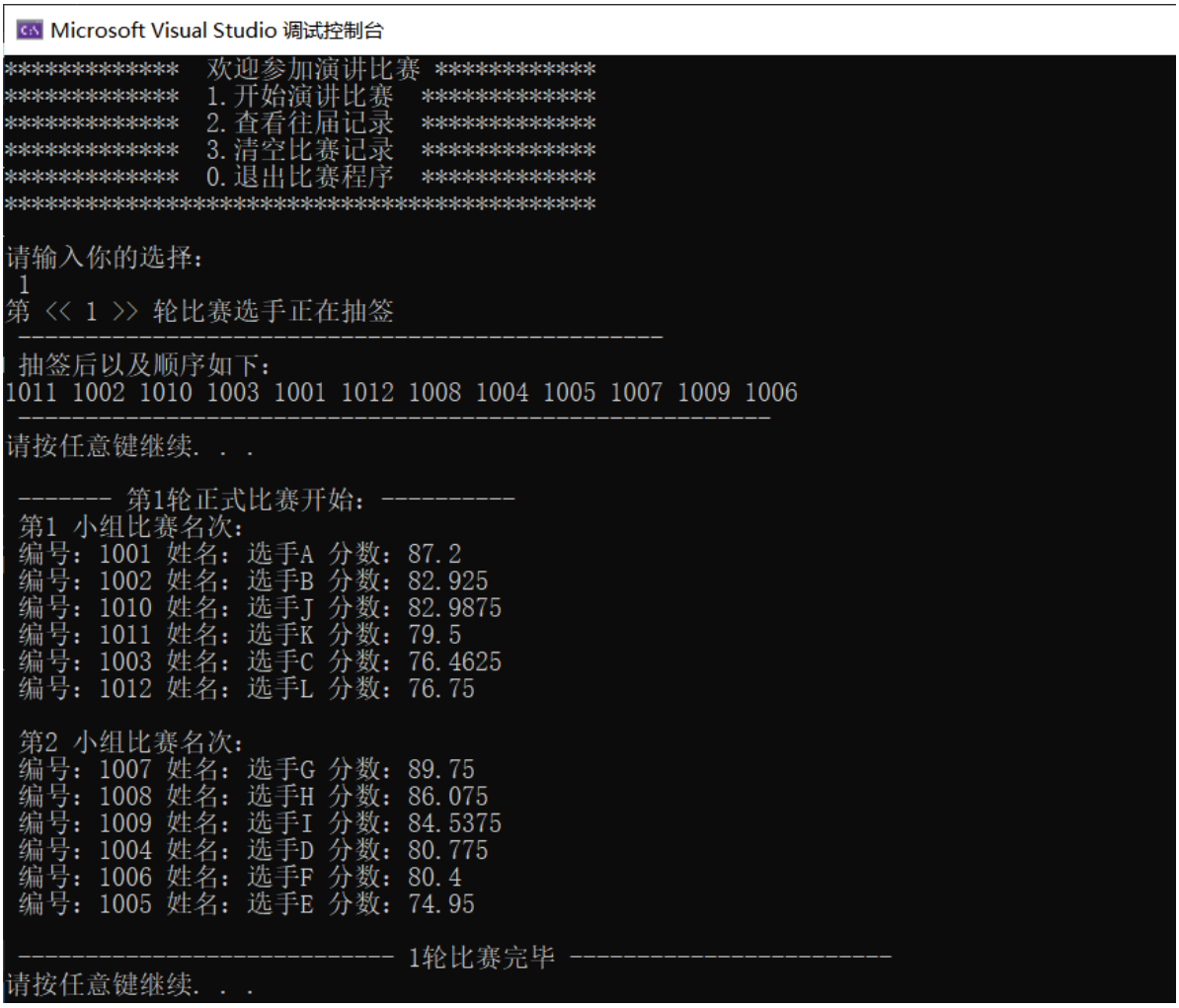
```
//2、比赛
void SpeechManager::speechContest()
{
    cout << " ------- 第" << this->m_Index << "轮正式比赛开始：----------" << endl;
    multimap<double, int, greater<int>>groupScore;  //临时容器，保存key分数 value选手编号

    int num = 0; //记录人员数，6个为1组

    vector<int>v_Src;   //比赛的人员容器
    if (this->m_Index == 1)
    {
        v_Src = v1;
    }
    else
    {
        v_Src = v2;
    }

    //遍历所有参赛选手
    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; //600~1000
            //cout << score << " ";
            d.push_back(score);
        }
        //cout << endl;
        sort(d.begin(), d.end(), greater<double>());  //排序
        d.pop_back();   //去除最低分
        d.pop_front();    //去除最高分

        double sum = accumulate(d.begin(),d.end(),0.0f);
        double avg = sum / d.size();

        //打印平均分
        //cout << "编号：  " << *it << " 姓名：" << this->m_Speaker[*it].m_name << " 获取平均分：" << avg << endl;

        //将平均分放入到map容器里
        this->m_Speaker[*it].m_Score[this->m_Index-1] = avg;

        //将打分数据放入到临时小组容器中
        groupScore.insert(make_pair(avg, *it));    //key是得分，value是具体选手编号
```

```
        //每6人取出前三名
        if (num%6 == 0)
        {
            cout << " 第" << num / 6 << " 小组比赛名次： " << 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
                    << " 分数： " << 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; count++,it++)
            {
                if (this->m_Index == 1)
                {
                    v2.push_back((*it).second);
                }
                else
                {
                    vVictory.push_back((*it).second);
                }
            }

            //容器清空
            cout << endl;
            groupScore.clear();
        }
    }
    cout <<" ---------------------------- "<<this->m_Index<< "轮比赛完毕 ---------
---------------"<<endl;
    system("pause");
}
```

- 在startSpeech比赛流程控制的函数中，调用比赛函数

```
while (true)
{
    sm.show_Menu();

    cout << "请输入你的选择： " << endl;
    cin >> choice;
    switch (choice)
    {
    case 1: //开始比赛
        sm.startSpeech();
        break;

    case 2://查看比赛记录
        break;

    case 3://清空比赛记录
        break;

    case 0://退出系统
        sm.exitSystem();
        break;
```

- 再次运行代码，测试比赛




### 6.3.7 显示比赛分数

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

```
        //3、显示晋级结果
        void showScore();
```

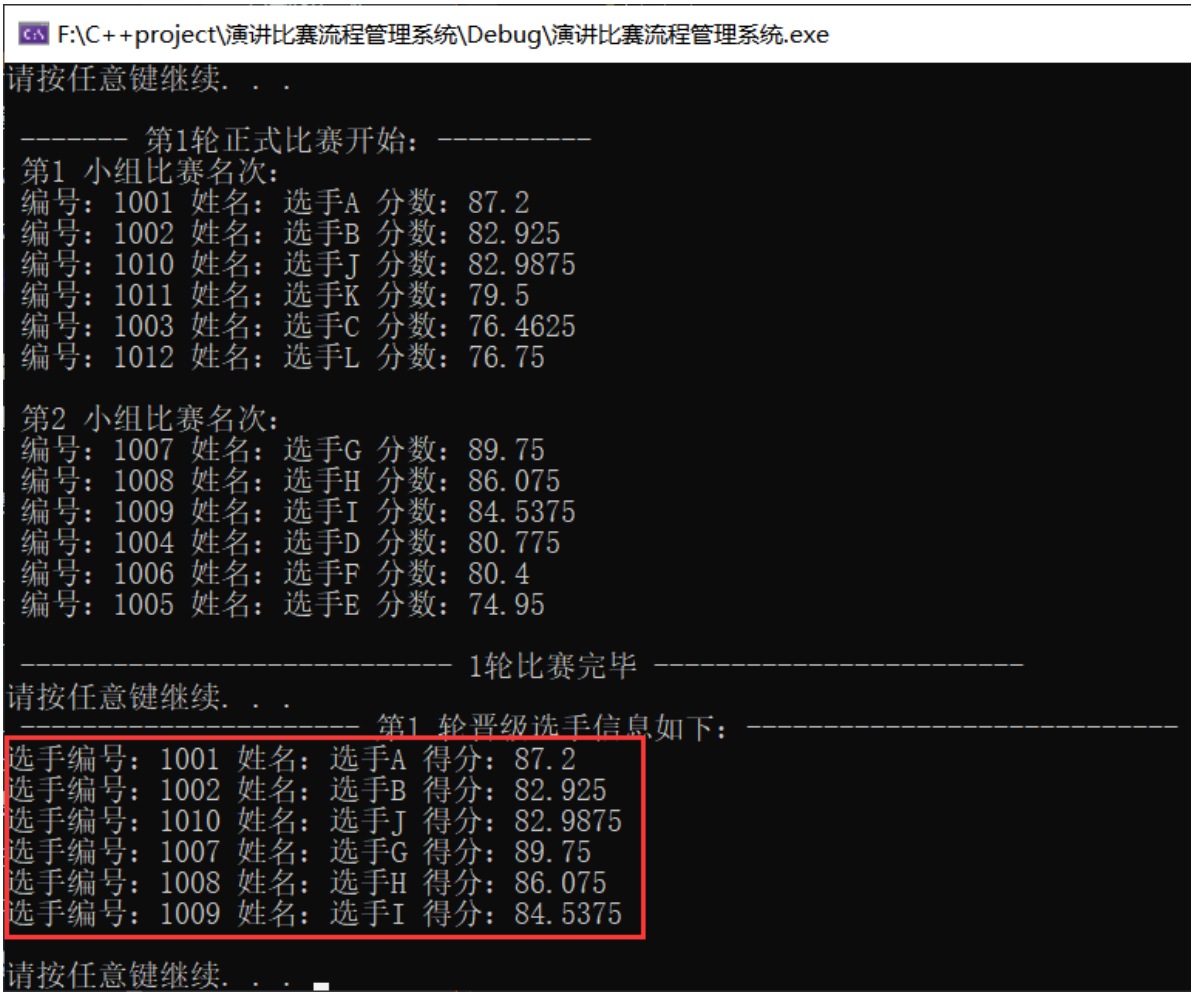
- 在speechManager.cpp 中实现成员函数 void showScore();

```
//3、显示晋级结果
void SpeechManager::showScore()
{
    cout << " ---------------------- 第" << this->m_Index << " 轮晋级选手信息如下：-
---------------------------" << endl;

    vector<int>v;
    if (this->m_Index == 1)
    {
        v = v2;
    }
    else
    {
        v = vVictory;
```

```
    }

    for (vector<int>::iterator it = v.begin(); it != v.end(); it++)
    {
        cout << "选手编号：" << *it << " 姓名：" << 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比赛流程控制的函数中，校园显示比赛分数函数


- 运行代码，测试效果



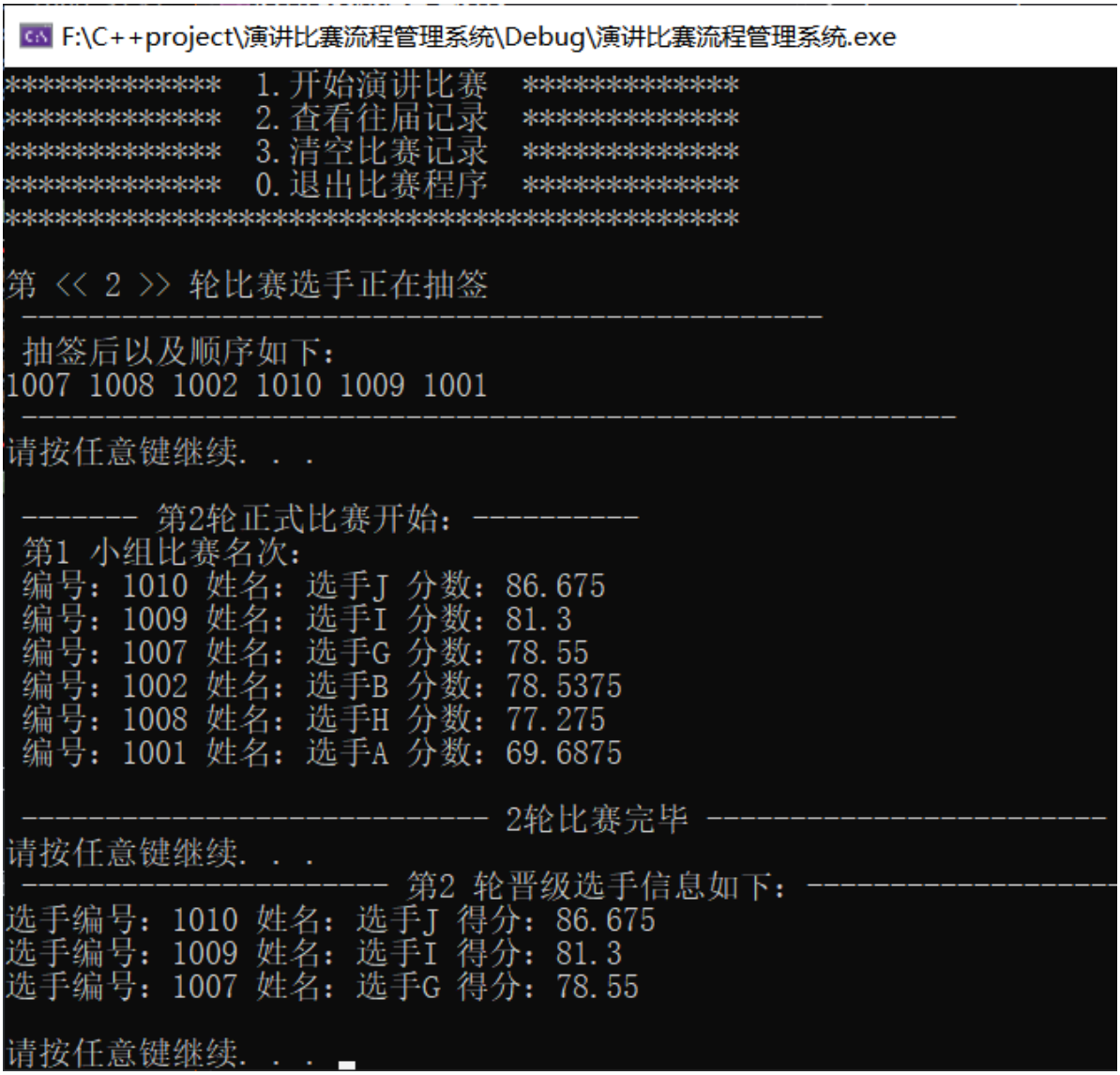

### 6.3.8 第二轮比赛

在第二轮比赛流程中，m_Index++，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

```cpp
//开始比赛 - 比赛流程控制
void SpeechManager::startSpeech()
{
    //第一轮比赛
    //1、抽签
    this->speechDraw();

    //2、比赛
    this->speechContest();

    //3、显示晋级结果
    this->showScore();

    //第二轮比赛
    this->m_Index++;

    //1、抽签
    this->speechDraw();

    //2、比赛
    this->speechContest();

    //3、显示最后结果
    this->showScore();

    //4、保存分数

}
```

测试，将整个比赛流程都跑通




## 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 中添加保存记录的成员函数， void saveRecord();

```
    //保存记录
    void saveRecord();
```

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

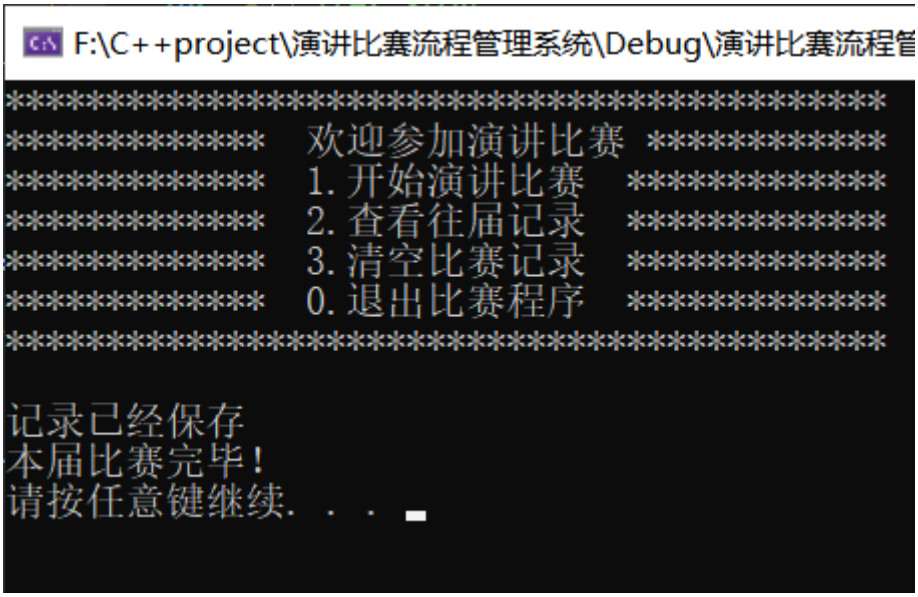
```
//保存记录
void SpeechManager::saveRecord()
{
    ofstream ofs;
    ofs.open("speech.csv",ios::out | ios::app );  //用追加方式写文件

    for (vector<int>::iterator it = vVictory.begin(); it != vVictory.end();
it++)
    {
        ofs << *it << "," << this->m_Speaker[*it].m_Score[1] << ",";
```

```
    }
    ofs << endl;
    cout << "记录已经保存" << endl;
}
```

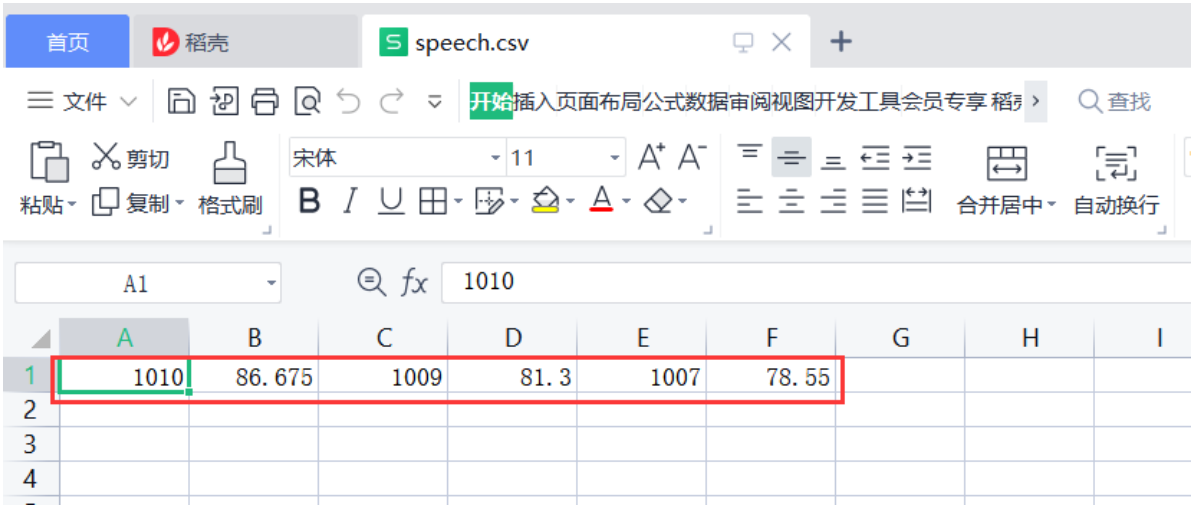
- 在startSpeech比赛流程控制的函数中，最后调用保存记录分数函数

```
86      //2、比赛
87      this->speechContest();
88
89      //3、显示晋级结果
90      this->showScore();
91
92      //第二轮比赛
93      this->m_Index++;
94
95      //1、抽签
96      this->speechDraw();
97
98      //2、比赛
99      this->speechContest();
100
101     //3、显示最后结果
102     this->showScore();
103
104     //4、保存分数
105     this->saveRecord();
106     cout << "本届比赛完毕！ " << endl;
107     system("pause");
108     system("cls");
109 }
110
```

- 测试，整个比赛完毕后保存情况




打开文件 speech.csv ,里面保存了前三名选手的变化以及得分




至此，整个演讲比赛功能制作完毕!

# 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;`

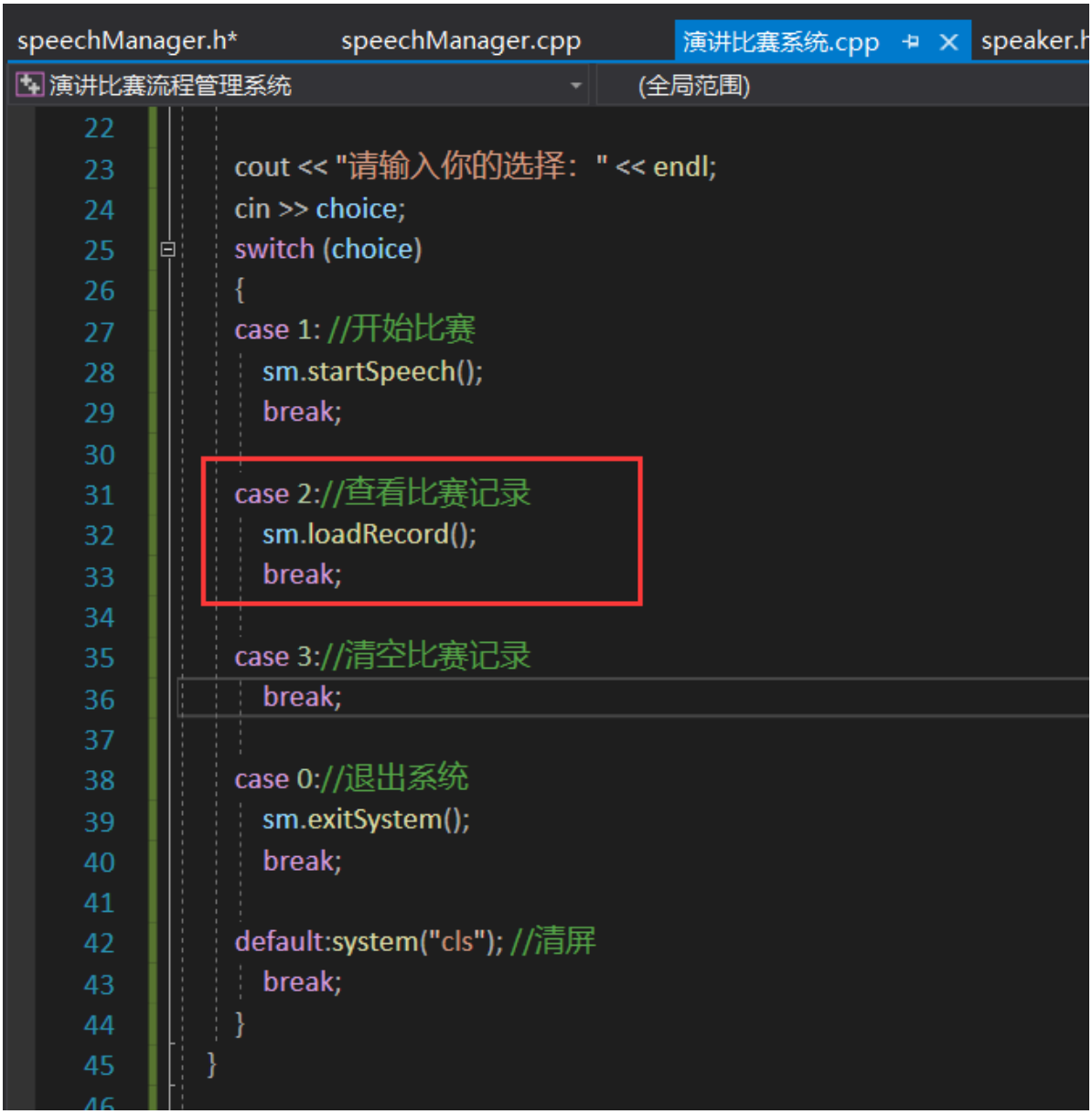
其中m_Record 中的key代表第几届，value记录具体的信息

```
        //保存记录
        void saveRecord();

        //读取记录
        void loadRecord();

        //判断文件是否为空
        bool fileIsEmpty;

        //往届记录
        map<int, vector<string>> mRecord;
```

- 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

    //文件不存在情况
    if (!ifs.is_open())
    {
        this->fileIsEmpty = true;
        cout << "文件不存在" << endl;
        ifs.close();
        return;
    }

    //文件清空清空
    char ch;
    ifs >> ch;
    if (ifs.eof())
    {
        cout << " 文件为空 " << endl;
        this->fileIsEmpty = true;
        ifs.close();
        return;
    }

    //文件存在并且有数据
    this->fileIsEmpty = false;
    ifs.putback(ch);  //将上面读取的单个字符放回来

    string data;
    while (ifs>>data)
    {
        cout << data << endl;
    }
    ifs.close();
}
```

- 在SpeechManager构造函数中调用获取往届记录函数




## 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();

```
//显示往届得分
void showRecord();
```

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

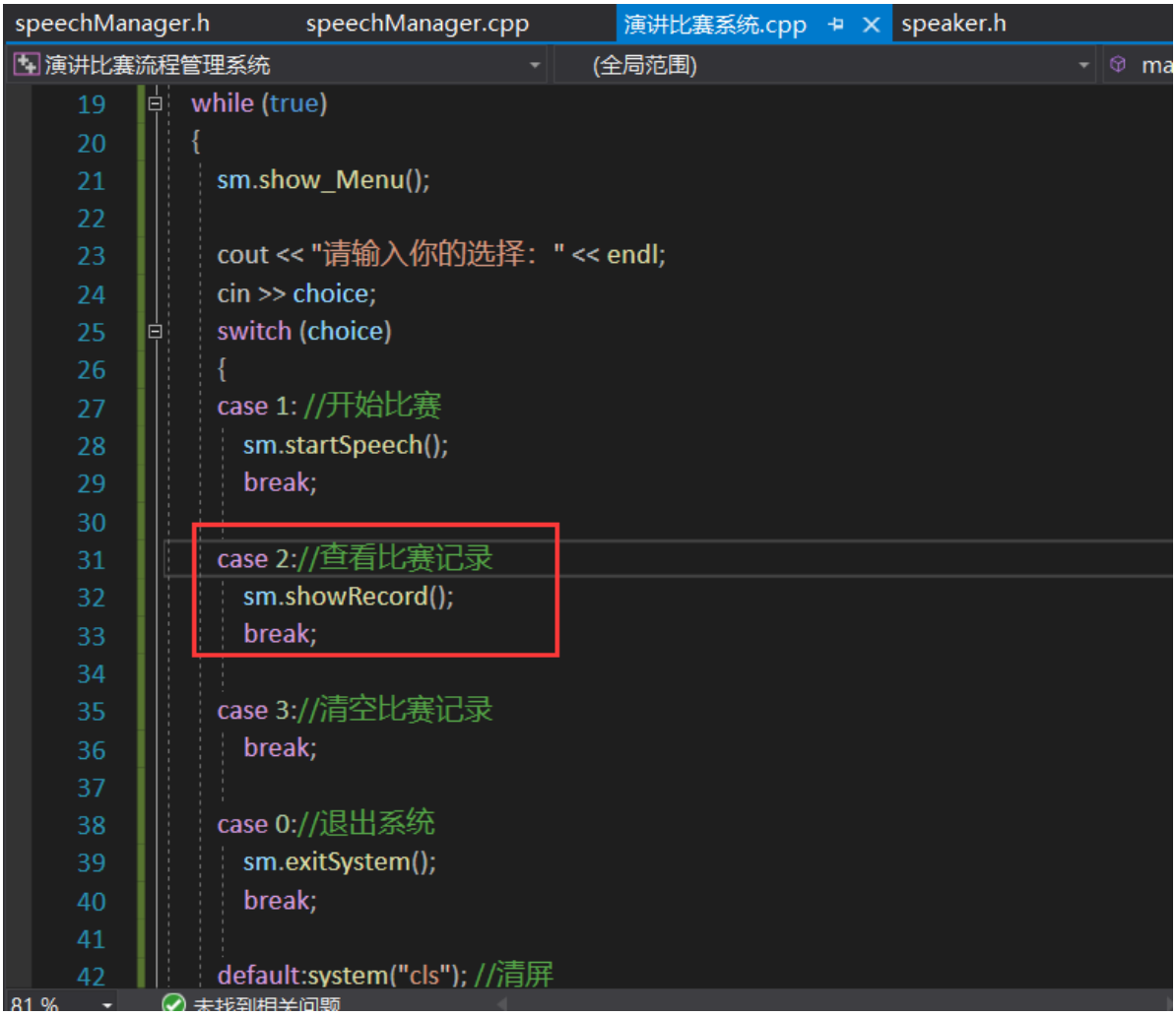
```
//显示往届得分
void showRecord();
```

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

```
//显示往届得分
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][4] << " 得分：" << this-
>m_Record[i][5] << endl;
    }
    system("pause");
    system("cls");
}
```

## 7.3 测试功能

在main函数分支 2 选项中，调用查看记录的接口




显示效果：

```
F:\C++project\演讲比赛流程管理系统\Debug\演讲比赛流程管理系统.exe
*********************************************
*************  欢迎参加演讲比赛 *************
*************  1.开始演讲比赛  *************
*************  2.查看往届记录  *************
*************  3.清空比赛记录  *************
*************  0.退出比赛程序  *************
*********************************************

请输入你的选择：
2
第1届冠军编号：1010 得分：86.675 亚军编号：1009 得分：81.3 季军编号：1007 得分：78.55
第2届冠军编号：1010 得分：86.675 亚军编号：1009 得分：81.3 季军编号：1007 得分：78.55
请按任意键继续. . .
```

## 7.4 bug解决

目前程序中有几处bug未解决：

1. 查看往届记录，若文件不存在或为空，并未提示

解决方式：在showRecord函数中，开始判断文件状态并加以判断

```
347  //显示往届得分
348  void SpeechManager::showRecord()
349  {
350      if (this->fileIsEmpty)
351      {
352          cout << "文件为空或者文件不存在！" << endl;
353      }
354      else
355      {
356          for (int i = 0; i < this->m_Record.size(); i++)
357          {
358              cout << "第" << i + 1 << "届"
359                  << "冠军编号：" << this->m_Record[i][0] << " 得分：" << this->m_Record[i][1] << " "
360                  << "亚军编号：" << this->m_Record[i][2] << " 得分：" << this->m_Record[i][3] << " "
361                  << "季军编号：" << this->m_Record[i][4] << " 得分：" << this->m_Record[i][5] << endl;
362          }
363      }
364
365      system("pause");
366      system("cls");
367  }
```

2. 若记录为空或者不存在，比完赛后依然提示记录为空

解决方式：saveRecord中更新文件为空的标志

```cpp
//保存记录
void SpeechManager::saveRecord()
{
    ofstream ofs;
    ofs.open("speech.csv",ios::out | ios::app ); //用追加方式写文件

    for (vector<int>::iterator it = vVictory.begin(); it != vVictory.end(); it++)
    {
        ofs << *it << "," << this->m_Speaker[*it].m_Score[1] << ",";
    }
    ofs << endl;
    cout << "记录已经保存" << endl;

    //有记录了，文件不为空
    this->fileIsEmpty = false;
}
```

3. 比赛完后查不到本届的记录，没有实时更新

解决方式：比赛完毕后，所有数据重置

```cpp
    //3、显示最后结果
    this->showScore();

    //4、保存分数
    this->saveRecord();

    //重置比赛，获取记录
    //容器初始化操作
    this->initSpeech();

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

    //加载往届记录
    this->loadRecord();

    cout << "本届比赛完毕！ " << endl;
    system("pause");
    system("cls");
}
```

4. 在初始化时，没有初始化记录容器

解决办法：initSpeech中添加，初始化记录容器

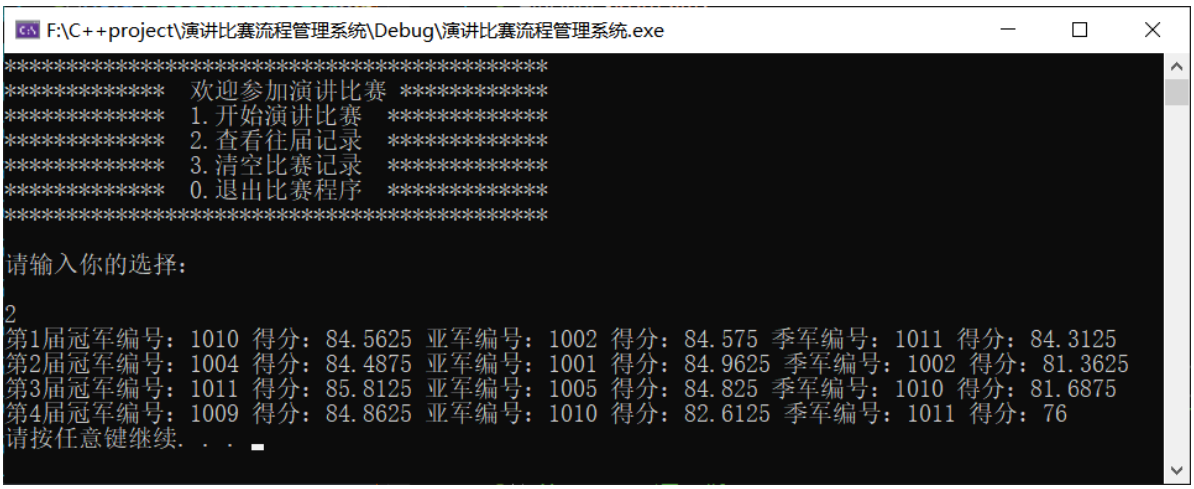
```
47  void SpeechManager::initSpeech()
48  {
49      //容器都置空
50      this->v1.clear();
51      this->v2.clear();
52      this->vVictory.clear();
53      this->m_Speaker.clear();
54      this->m_Index = 1;  //初始化比赛轮数
55
56      //将记录的容器也清空
57      this->m_Record.clear();
58  }
```

5. 每次记录都是一样的

解决办法：在main函数开始处添加随机种子

```
    srand((unsigned int)time(NULL));
```

所有bug解决后，测试：




# 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();

```
//清空文件
void clearRecord();
```

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

```
//清空文件
void SpeechManager::clearRecord()
{
```

```
    cout << "是否确定清空文件？" << endl;
    cout << " 1、确定" << endl;
    cout << " 2、返回" << 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();

        //重新创建12名选手
        this->createSpeaker();

        //加载往届记录
        this->loadRecord();

        cout << "清空成功" << endl;
    }

    system("pause");
    system("cls");

}
```

## 8.2 测试清空

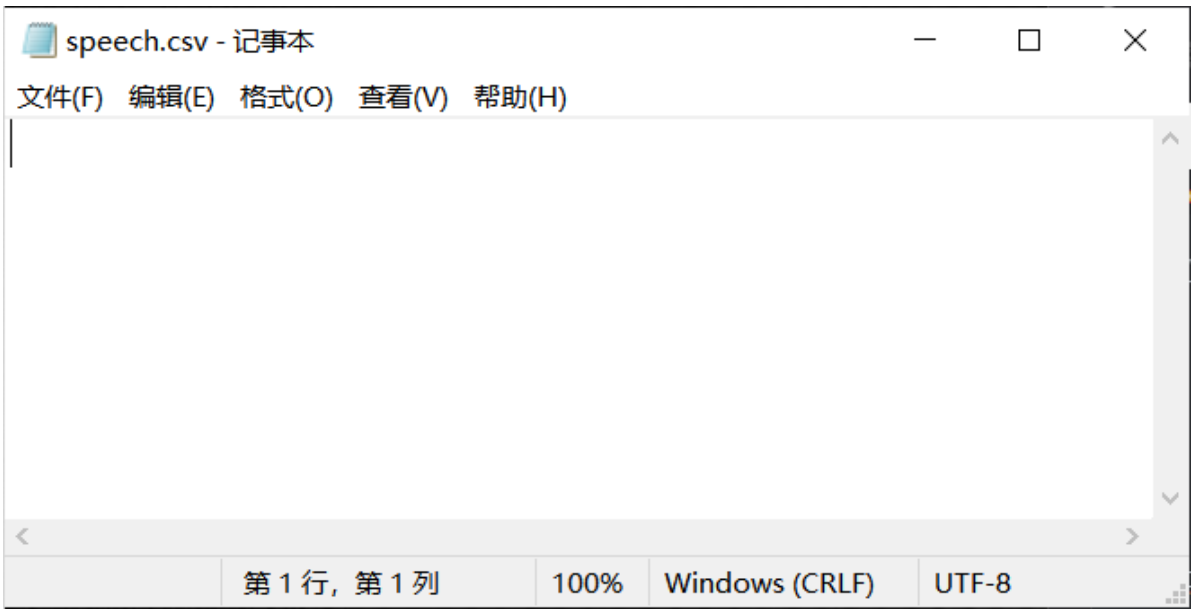
在main函数分支 3 选项中，调用清空比赛记录的接口

```
speechManager.h    speechManager.cpp    演讲比赛系统.cpp    speaker.h
演讲比赛流程管理系统                    (全局范围)
22      {
23          sm.show_Menu();
24
25          cout << "请输入你的选择： " << endl;
26          cin >> choice;
27          switch (choice)
28          {
29          case 1: //开始比赛
30              sm.startSpeech();
31              break;
32
33          case 2://查看比赛记录
34              sm.showRecord();
35              break;
36
37          case 3://清空比赛记录
38              sm.clearRecord();
39              break;
40
41          case 0://退出系统
42              sm.exitSystem();
43              break;
44
45          default:system("cls"); //清屏
```

运行程序，测试清空记录：

```
F:\C++project\演讲比赛流程管理系统\Debug\...
**********************************************
**********  欢迎参加演讲比赛 **********
**********  1. 开始演讲比赛  **********
**********  2. 查看往届记录  **********
**********  3. 清空比赛记录  **********
**********  0. 退出比赛程序  **********
**********************************************

请输入你的选择：
3
是否确定清空文件？
 1、确定
 2、返回
1
清空成功
请按任意键继续. . .
```

speech.csv中记录也为空


speech.csv - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
第 1 行，第 1 列
100%
Windows (CRLF)
UTF-8

本案例到此结束