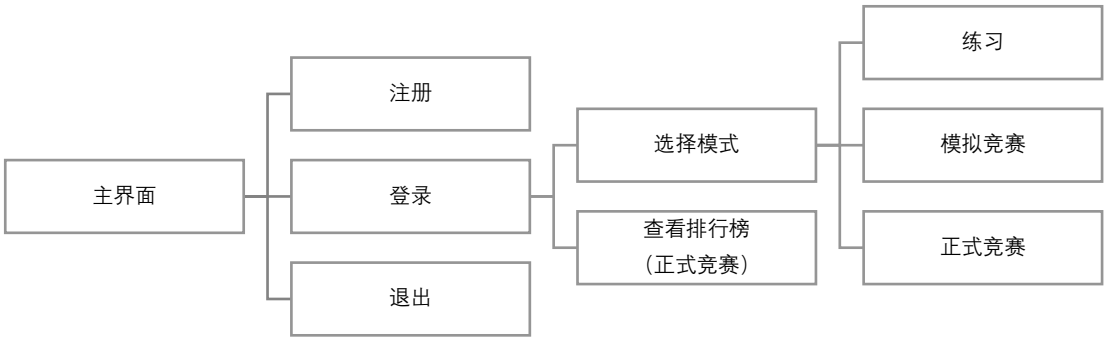
郑凯琳 205220025

项目四：实验报告

框架设计 & 程序流程



1. **存储和管理用户信息**
   - User 结构体
     - username：用户的用户名，用于标识用户。
     - password：用户的密码，用于验证用户身份。

```
struct User
{
    string username;
    string password;
    //int score;
    //int answerTime;
};
```

   - 全局变量
     - users：一个 vector 容器，用于存储多个用户对象。
     - usersfile：一个字符串变量，表示用户文件的名称和路径。

```
// 全局变量
vector<User> users;
string usersfile = "users.txt";
```

2. **重要的辅助函数**
   - `wchar_t* to_wchar(string text)`
     - 将 string 转换成 wchar
     - outtextxy 函数是针对 C-String（以空字符 '\0' 结尾的字符数组）的，需要将字符串以 C-String 的形式传递给 str 参数
   - `class EasyTextBox`
     - 实现文本框控件
     - 在绘图窗口中创建和显示文本框，并获取用户输入的文本
     - 实现依赖于绘图库和消息处理机制，通过处理鼠标点击和字符输入消息来实现文本输入和控件的交互
   - `class EasyButton`
     - 实现按钮控件
     - 在绘图窗口中创建按钮，并根据用户的点击来触发相应的事件处理函数
     - 通过使用该控件，用户可以实现按钮的显示、交互和事件响应功能。

3. **题目（问题）：QuestionBank.h**
   - 题目结构体

```
// 结构体表示题目
struct Question
{
    wchar_t content[1000];      // 题目内容
    wchar_t optionA[1000];      // 选项 A
    wchar_t optionB[1000];      // 选项 B
    wchar_t optionC[1000];      // 选项 C
    wchar_t answer[1000];       // 答案
};
```

   - 题目对象的创建函数
     - 用于创建并返回一个新的题目对象
     - 函数接受题目的内容、选项和答案作为参数
     - 函数内部首先创建了一个新的 Question 结构体对象 new_ques
     - 最后返回新创建的题目对象 new_ques

```
vector<Question> questions;

Question new_Question(
    const wchar_t* content,
    const wchar_t* optionA,
    const wchar_t* optionB,
    const wchar_t* optionC,
    const wchar_t* answer
) {
    Question new_ques = {};
    wcscpy_s(new_ques.content, content);
    wcscpy_s(new_ques.optionA, optionA);
    wcscpy_s(new_ques.optionB, optionB);
    wcscpy_s(new_ques.optionC, optionC);
    wcscpy_s(new_ques.answer, answer);
    return new_ques;
}
```

   - 题目容器和题目内容存储
     - 名为 questions 的容器（vector）来存储题目对象
     - questions 可以用于存储多个题目，每个题目都是一个 Question 结构体对象
     - 另外，代码中还定义了几个容器（vector）：
       - `vector<const wchar_t*> question_content`：用于存储题目的内容
       - `vector<const wchar_t*> question_optionA`：用于存储选项 A
       - `vector<const wchar_t*> question_optionB`：用于存储选项 B
       - `vector<const wchar_t*> question_optionC`：用于存储选项 C
       - `vector<const wchar_t*> question_answer`：用于存储答案

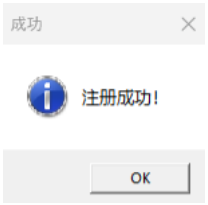
4. **首页界面**　　`void Init()`
   - 读取用户信息文件 “users.txt”：
     - 创建一个 User 对象来存储这些信息。
     - 将 User 对象添加到名为 users 的容器中。
   - 初始化图形界面：
   - 绘制界面元素：

- 加载背景图片并创建一个 IMAGE 对象 bgimg，图片文件名为 "init.jpg"。

- 处理用户的鼠标操作：
  - 进入一个无限循环，不断监听鼠标消息。
  - 获取鼠标消息，并判断是否为左键按下的消息（WM_LBUTTONDOWN）。
  - 如果用户点击了注册选项的区域，调用 Register 函数进行注册操作，并跳出循环。
  - 如果用户点击了登录选项的区域，调用 Login 函数进行登录操作，并跳出循环。
  - 如果用户点击了退出选项的区域，调用 closegraph 函数关闭图形界面，程序结束。

5. **注册界面** `void Register()`

- 用户可以在界面上输入用户名和密码，并通过点击注册按钮进行注册操作。
- 代码中的控件类 EasyTextBox 和 EasyButton 封装了文本框和按钮的绘制、交互和事件处理，简化了界面开发过程。
- 同时，使用鼠标消息处理函数 getmessage 和判断控件的点击位置，实现了用户与界面的交互。
  - ❖ 注册成功




Project4
用户名：
apps
密　码：
abcd1234
注册
返回

- 注册失败


错误
用户名已存在!
OK


Project4
用户名：
wingga
密　码：
qwertyuio
注册
返回

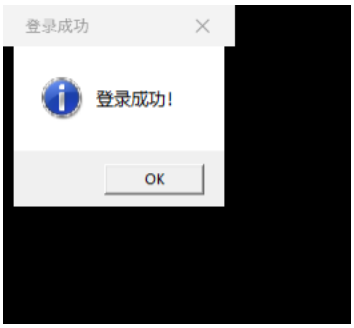
- 注册失败


错误
密码长度必须至少为八位!
OK

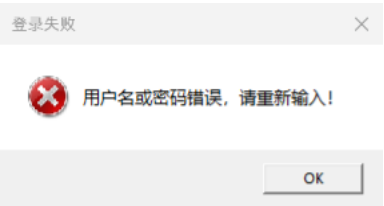
6. **登录界面**　　void Login()

- 用户可以在界面上输入用户名和密码，并通过点击登录按钮进行验证操作。
- 代码中的控件类 EasyTextBox 和 EasyButton 封装了文本框和按钮的绘制、交互和事件处理，简化了界面开发过程。
- 同时，使用鼠标消息处理函数 getmessage 和判断控件的点击位置，实现了用户与界面的交互。
- 登录成功后，将弹出登录成功的消息框并跳转到选择界面；
  登录失败则弹出登录失败的消息框并保留在登录界面。

❖ 登录成功



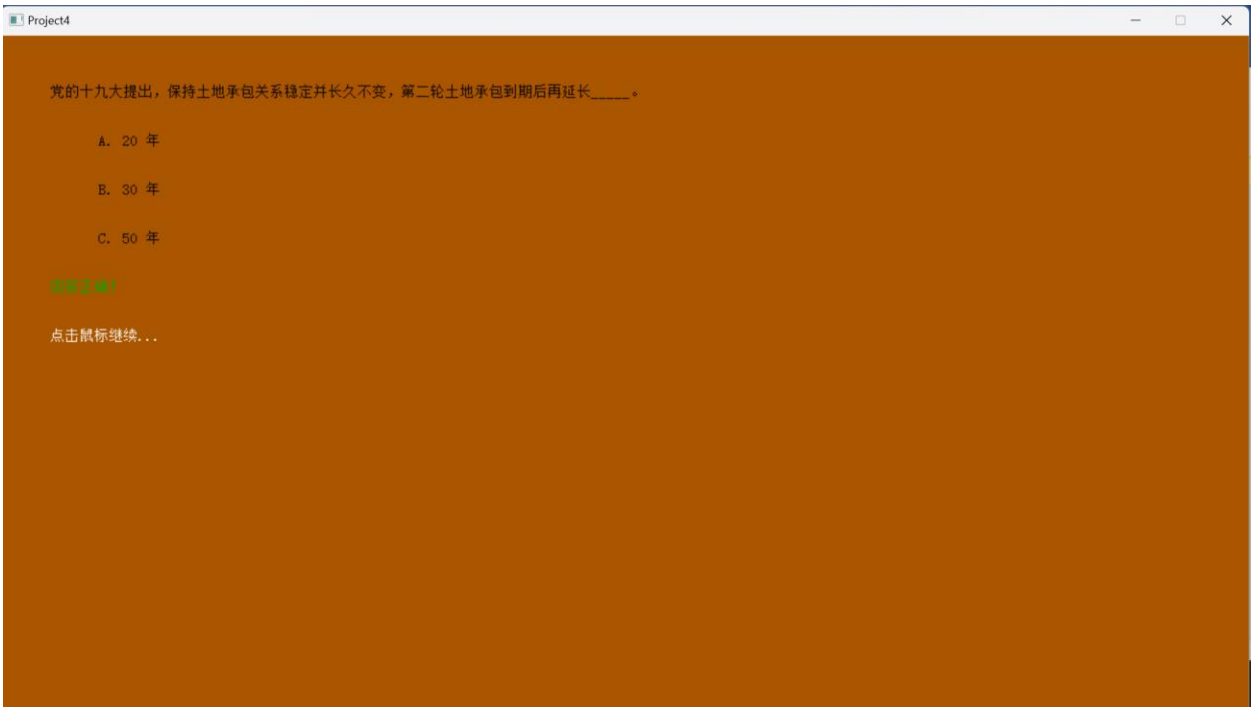
❖ 登录失败

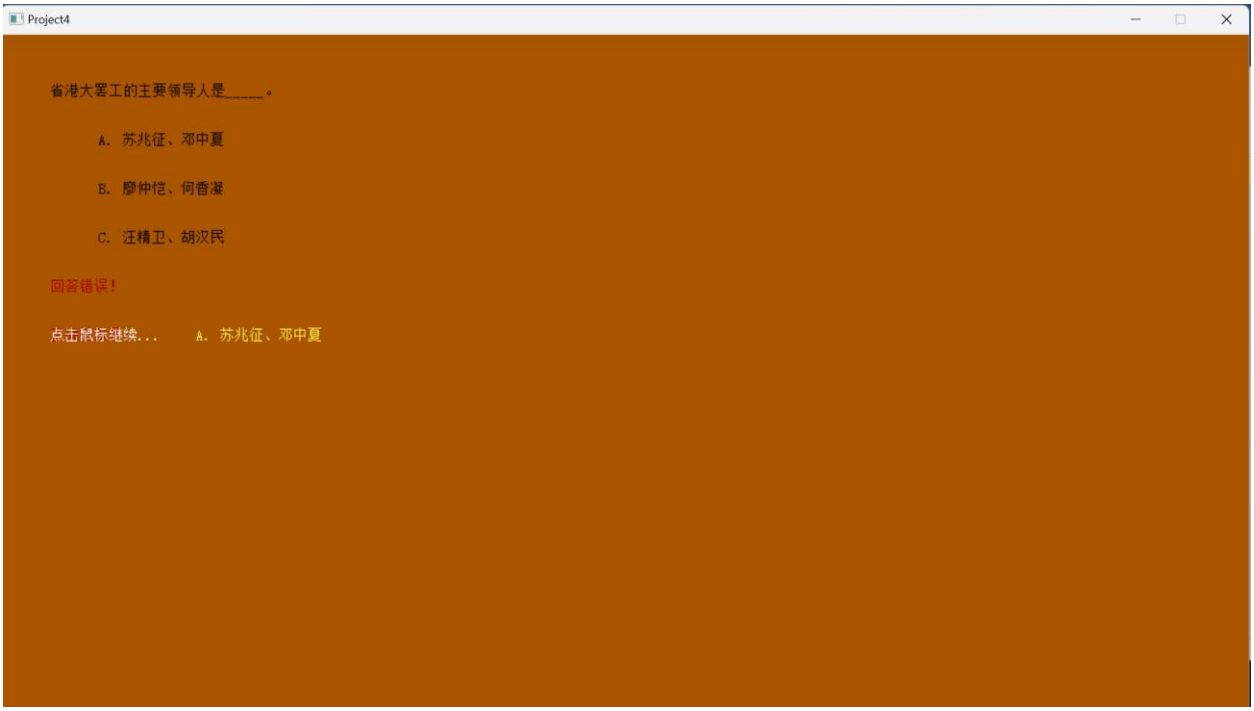


7. **选择模式界面** `void Mode()`
   - 用户可以在界面上选择不同的模式进行操作。
   - 界面使用图形绘制了标题和选项，并通过监听鼠标消息实现了用户与界面的交互。
   - 根据用户的选择，执行相应的操作函数。
     这样的界面设计可以提供更多功能和交互选项，增强了用户体验。
   - 代码中的控件类 EasyTextBox 和 EasyButton 封装了文本框和按钮的绘制、交互和事件处理，简化了界面开发过程。

8. 练习模式　　`void Exercise()`
   用户可以在绘图窗口中随机选择一道题目进行练习，并给出答案的反馈。
   - 随机选择题目：
     - SelectQuestion 函数用于从题目列表中随机选择一道题目。
     - 使用 srand 函数初始化随机数种子，使用 rand 函数生成一个随机的题目索引。
     - 返回随机选择的题目对象。
   - 练习模式：
     - 调用 InitQues 函数进行一些初始化操作。

- 使用 initgraph 函数初始化绘图窗口大小。
- 进入一个无限循环，在每次循环开始前清空绘图窗口。
- 随机选择一道题目，并绘制题目内容和选项。
- 绘制退出按钮。
- 监听鼠标消息，等待用户点击鼠标。
- 如果用户点击了退出按钮，调用 Mode 函数返回到选择模式界面。
- 检查用户的答案，并根据答案的正确性绘制结果。
- 绘制结果文本和提示文本。
- 再次监听鼠标消息，等待用户点击鼠标继续。





9. 模拟竞赛 & 正式竞赛

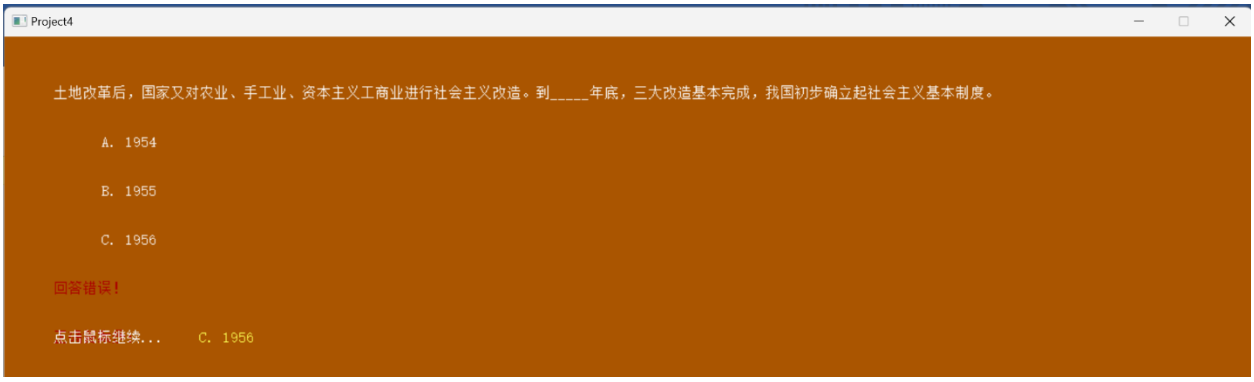
```
void MockExam(int questionCount, int timeLimit)
void Exam(int questionCount, int timeLimit)
```

用户需要在规定的时间内回答一定数量的题目，并在竞赛结束后显示答题结果。

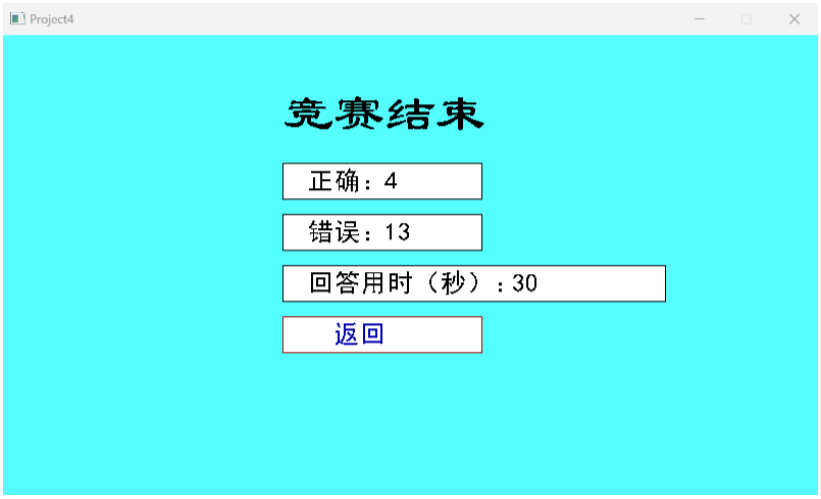
- 绘制竞赛界面：
    - DrawCompetition 函数用于绘制竞赛界面，包括题目内容、选项和倒计时。

- 使用 settextcolor 和 settextstyle 函数设置文本颜色和样式。
- 使用 outtextxy 函数在指定位置输出文本。
- 使用 fillrectangle 函数绘制选项区域的矩形。

- 显示竞赛结果：
  - DisplayResult 函数用于显示竞赛结果。
  - 使用 initgraph 函数初始化图形界面。
  - 使用 settextcolor、settextstyle、outtextxy 和 fillrectangle 函数绘制界面元素，包括标题、正确数量、错误数量和答题用时。
  - 监听鼠标消息，如果用户点击返回按钮，则调用 Mode 函数返回到选择模式页面。
- 模拟竞赛 & 正式竞赛：
  - MockExam 函数用于实现模拟竞赛的逻辑。// Exam 函数用于实现正式竞赛的逻辑。
  - 调用 InitQues 函数进行一些初始化操作。
  - 使用 initgraph 函数初始化绘图窗口大小。
  - 计算竞赛的结束时间，根据题目数量和时间限制计算。
  - 初始化正确数量、错误数量和答题用时的变量。
  - 进入一个循环，在每次循环开始前清空绘图窗口。
  - 随机选择一道题目，并绘制竞赛界面，显示倒计时。
  - 监听鼠标消息，等待用户点击鼠标。
  - 检查用户的答案，并根据答案的正确性绘制结果。
  - 统计正确数量、错误数量和答题用时。
  - 监听鼠标消息，等待用户点击鼠标继续。
  - 更新题目数量和答题用时的变量。
  - 关闭绘图窗口。
  - 调用 DisplayResult 函数显示竞赛结果。




Project4
新文化运动的两大口号是_____。
A. 民主和科学
B. 民权和平等
C. 自由和民主
回答正确!
点击鼠标继续...


Project4
竞赛结束
正确：4
错误：13
回答用时（秒）：30
返回