# 兰顿蚂蚁第四次工作报告

202430551743 软件1班 严嘉鹏

1. **页面跳转**

上一周的代码实现了让用户选择两个模式之一开始游戏，但是只能玩一次，即每个模式参与一次后自动退出游戏。本周进行改进。

**实现功能：**无论是参与了自由模式还是闯关模式，每轮游戏结束后，用户都有3个选择：1、再玩一次本模式(可循环) 2、退出当前模式回到主页面 3、退出游戏。同时，由于我的游戏记录模块规定在运行时添加游戏记录，所以意外的实现了输出同一登录时间下的每轮游戏记录。

**实现手段：**没有添加新的函数接口，而是简单使用while里嵌套switch实现游戏可不断反复进行。

**遇到问题：**退出游戏，既要跳出switch,又要跳出while,但是break只能跳出一次switch，思考过后，在全局声明了一个bool变量作为while里的条件判断，在switch的break前改变bool值即可一起跳出。

**代码如下：**

bool shouldExit = false;

while (!shouldExit)

{

std::cout << "请输入接下来的操作：" << std::endl;

std::cout << "1、再玩一次本模式" << std::endl;

std::cout << "2、退出该模式并回到主页面 " << std::endl;

std::cout << "3、退出游戏 " << std::endl;

std::cin >> input;

switch (input)

{

case 1:

{

freedommode.freeMode();

break;

}

case 2:

{

GameHomepage homepage2;

homepage2.beginGame();

shouldExit = true;

break;

}

case 3:

{

std::cout << "游戏已退出，欢迎下次再来！" << std::endl;

shouldExit = true;

break;

}

}

}

1. **黑框框里的字体变色**

由于暂时没有可视化界面，用户的回答正误在较多的输出中不明显，为了便于观察，将“恭喜您回答正确”字体调为绿色、“很遗憾，回答错误”调为红色。

**实现功能：**黑框框里的字体变色。

**实现手段：**在VS2022里调用<windows.h>里的函数

**代码如下：**

#include <iostream>

#include <windows.h>

int main() {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

// 设置文本颜色为红色

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED);

std::cout << "This is red text." << std::endl;

// 设置文本颜色为绿色

SetConsoleTextAttribute(hConsole, FOREGROUND\_GREEN);

std::cout << "This is green text." << std::endl;

// 设置文本颜色为黄色

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED | FOREGROUND\_GREEN);

std::cout << "This is yellow text." << std::endl;

// 设置文本颜色为蓝色

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

std::cout << "This is blue text." << std::endl;

// 恢复默认颜色

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

return 0;

}

1. **注册与登录界面的调整**

上一周的代码对于主页面的用户注册与登录表达不够清晰，而且没有考虑到注册时用户名已存在、用户第一次登录密码错误、用户忘记密码的情况，本周进行改进。

**实现功能:**

原代码逻辑：用户输入账户名和密码->程序判断是否注册->程序判断登录是否成功。

改进后：进入主页面后有2个选择：1、注册，2、登录->若选注册，会检查用户名是否已存在；若选登录，每个用户有3次密码输入机会，每次输入密码错误后，会有两个选择：1、重新输入（3次密码输入不对直接退出游戏），2、忘记密码（提示用户输入新的密码后用新密码登录）

**实现手段:**

由于用户的注册和登录对于每种游戏模式都要使用，所以我只要重写用户登录这个函数接口，并且写一个忘记密码的函数，再结合switch提供选择就行。

**遇到问题：**

刚开始，我在书写忘记密码这个函数时，先用ofstream outfine(filename)的方式打开文件，再重新传入新的密码。然后运行发现该用户之前的游戏记录全没了，而且无法找回，这种不加后缀的文件打开方式默认为ios：：out(新输入内容会直接覆盖原内容)。所以，我需要修改第一行（密码），同时保留该用户的游戏记录。没有合适的文件打开方式，只能求助AI。AI提供的思路也简单，那就是，先用字符数组将原来的游戏记录保存，再修改第一行的密码即可。

利用了getline函数，总结这个函数的知识点：std::istream& getline (std::istream& is, std::string& str); std::getline 是 C++ 标准库 <string> 中提供的一个函数，它的作用是从输入流 is 中读取一行文本，并将其存储到字符串 str 中。这里的输入流 is 可以是标准输入流 std::cin，也可以是文件输入流 std::ifstream，在使用 std::getline 从文件输入流（如 std::ifstream）中读取内容时，文件输入流对象维护着一个内部的文件指针，该指针会记录当前读取的位置。每次调用 std::getline 函数时，它会从文件指针当前所在的位置开始读取，直到遇到换行符 \n 为止，将读取到的内容（不包含换行符）存储到 str 中，然后将文件指针移动到换行符之后的位置，也就是下一行的起始位置。

**代码如下:**

bool User::forgotPassword()

{

std::string newPassword;

std::cout << "请输入新的密码（该密码直接替换原密码，不会影响您的游戏记录）" << std::endl;

newPassword=getpassword();

std::string filename = username + ".txt";

std::ifstream inFile(filename);

if (!inFile.is\_open())

{

std::cout << "无法打开文件" << std::endl;

return false;

}

std::vector<std::string>lines;//用来存储新的密码和原来的游戏记录

std::string line;//用于ifstream的指针标记读取行数

//读取第一行（密码）

if (std::getline(inFile, line))

{

lines.push\_back(newPassword);

}

//读取剩余行（用户原来的游戏记录）

while (std::getline(inFile, line))

{

lines.push\_back(line);

}

inFile.close();

std::ofstream outFile(filename);

if (!outFile.is\_open())

{

std::cout << "无法打开文件进行写入" << std::endl;

return false;

}

//将修改后的内容重新传入文件

for (const auto& l : lines)

{

outFile << l << std::endl;

}

outFile.close();

std::cout << "密码修改成功，请使用新密码登录" << std::endl;

return true;

}

1. **隐藏用户输入的密码并修改打印用户游戏记录的函数**

结合实际，一般的app里涉及到输入密码的环节，为了保护用户隐私，会将用户输入的密码转为“\*”的形式呈现，这里效仿；

**实现功能：**隐藏用户输入的密码；不在游戏记录里输出用户的密码（文件第一行）

**实现手段：**调用<conio.h>，处理光标和退格键；调用一次getline()函数却不输出，只下调一行文件指针，实现跳过输出文件第一行（密码）。

**遇到问题：**需要学习一个常识，空格位置有一个光标，输出的一个字符会出现在原空格位置（若输出空格就可以替换原光标位置处的字符），然后光标会自动向后挪一个位置（光标和字符都占一个位置）。

**代码如下：**

std::string getPassword()

{

std::string password;

char ch;

while ((ch = \_getch()) != '\r')//用户按下回车键时停止输入

{

if (ch == '\b')//处理退格键

{

if (!password.empty())

{

password.pop\_back();//删除最后一个字符

std::cout << "\b \b";//回退光标

}

}

else

{

password += ch;

std::cout << '\*';

}

}

std::cout << std::endl;

return password;

}

void User::printAllRecordsFromFile(const std::string& \_username) {

std::ifstream readFile(\_username + ".txt");

if (!readFile.is\_open()) {

std::cerr << "无法打开文件读取游戏记录。" << std::endl;

return;

}

std::string line;

std::cout << "从文件中读取的用户 " << \_username << " 的所有记录：" << std::endl;

// 跳过第一行（密码）

if (std::getline(readFile, line))

{

// 不打印第一行

}

while (std::getline(readFile, line)) {

std::cout << line << std::endl;

}

readFile.close();

}

1. **实验心得**

对函数接口有了更深的理解，以前认为它只是可以封装常用的代码，这一次通过修改userLogin()函数定义，就解决了用户密码错误和忘记密码的问题，最重要的是，其他地方的代码几乎没动。如果不是改函数接口的话，要想实现同样的功能，又要使用switch语句写很多条件，还要复制粘贴到多个位置。

1. **下周计划**

1自由模式的游戏记录：存档（记录电脑随机生成的蚁行图和用户的作答时间和正确率）

2用户注销以及删除某游戏记录

3闯关模式选择性保存：重新设置保存接口，用户在某关结束后可选择保存与否

4制作闯关模式排行榜：设置小数格式，某条形如：用户名，作答时间，正确率，登录时间。

难点：确定如何排名（已经设置了3关，每关又设置了3种难度模式）和书写排序算法

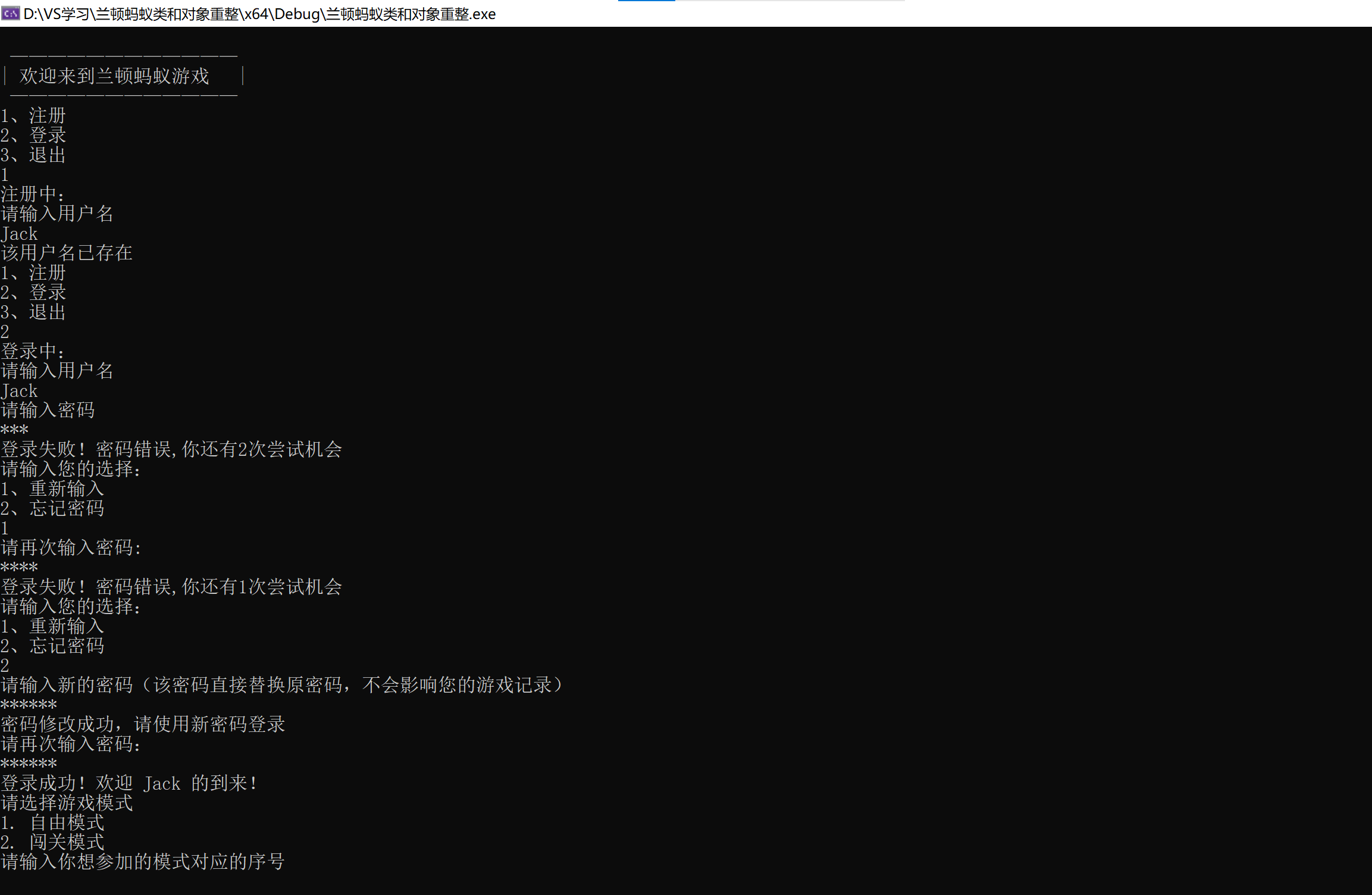
5制作自由模式题库：期望实现不同用户将自己得到的有意思的题目上传至题库（自由模式是电脑随机生成有效题目），供其他用户查看并尝试

6构思并完成趣味模式类图：如何修改兰顿蚂蚁的规则或棋盘组成，使之适用于某些道具

难点：由于是回溯性游戏，如果中间通过道具使游戏变简单，那么初始棋盘也要更新，说白了，使用道具后，要同步更新答案。

1. **实验成果**

改进后的用户注册与登录



页面跳转与字体变色



改进后的游戏记录（隐藏了用户密码），只显示登录时间的游戏记录来自自由模式，因为还没有做自由模式的游戏记录，以及还没有解决删除记录的操作，所以暂时这样输出

