面向对象程序设计项目报告

简易游戏账号管理系统的设计与实现

姓　　名：Ziggy Xuan

班　　级：

学　　号：

指 导 教 师：

2025 年 4 月

基于 C++的游戏账号管理系统设计与实现

摘要:

本文详细介绍了一个基于 C++语言开发的游戏账号管理系统。该系统提供完整的用户与管理员账户管理功能，支持普通玩家（Gamer）与管理员（Administrador）两种角色。玩家能够执行查看装备列表、购买装备等操作；管理员可查看详细的登录日志，管理账户状态如冻结或解冻。系统通过 MD5 算法实现密码加密，并利用安全模块（SecurityModule）监控账户异常登录行为，从而保障系统的安全性与稳定性。

关键词：

C++；账户管理；MD5 加密；游戏管理系统；安全登录；文件管理

目录

一、需求功能描述

随着网络游戏规模的持续扩大，账号安全和便捷管理已成为提升玩家体验与运营效率的关键。本项目基于 C++语言开发，融合面向对象思想，构建了一套完整的游戏账号管理系统。

为了实现用户与管理员账号的分级管理、装备查看与购买、以及安全登录控制等功能，系统划分为两个主要角色：普通用户（Gamer）和管理员（Administrador），并分别设计对应的权限与操作界面。

为了实现用户登录与注册功能，系统采用文件存储方式管理账号信息，并通过读取账号信息文件验证用户名与密码的正确性。对于用户登录失败的情况，系统支持实时注册，注册时系统会自动为用户创建包含初始余额和默认装备的用户数据文件，从而实现完整的注册-登录-使用流程。

为了实现密码保护机制，系统引入了 开源的 MD5 加密算法，对用户密码进行加密后保存到文件中。在用户登录时，通过对输入密码进行加密后与文件中已保存的密文进行比对，从而保障了账号信息的安全性。

为了实现装备查看与购买功能，系统设计了用户专属的数据文件，记录其用户名、账户余额和装备清单。在程序运行过程中，读取并解析该数据文件内容用于展示用户已有装备。同时，为了支持购买功能，系统设计了一个装备商店，通过文件读取预设装备信息并展示在控制台，用户可选择装备购买。系统会判断用户是否已有该装备以及余额是否充足，确保购买逻辑的合理性，并更新用户数据文件以反映变更。

为了实现用户登录安全控制功能，系统引入了登录失败次数统计机制。具体做法是：每当用户输入错误密码一次，系统会将失败记录写入非法登录日志文件。如果在同一天内同一账号失败次数达到三次，系统将自动将该账号写入冻结账户列表中，并在后续登录时阻止其进入系统，提示账号已被冻结。而在管理员界面中，管理员可以选择查看所有用户的登录日志，也可以查看并管理被冻结的账号。如果要解冻账户，需要管理员输入待解冻的账号的账号名，成功输入后，系统便会自动删除其在冻结列表和非法登录日志中的相关记录，从而恢复正常登录状态。

本系统整体采用面向对象设计思想，用账户类（Account）作为基类，用普通用户与管理员分别作为子类实现各自的功能模块。通过模块化的设计方式，实现了程序的可维护性和可复用性。

二、类设计描述

2.1 系统设计思路概述

为提高代码复用性与便于后续维护，本系统采用面向对象设计。整体分为五个核心类：

Account（基类）：定义了 alidatePassword、operMenu 等通用接口。

Gamer（玩家子类）：继承自 Account，重写登录验证、菜单显示与装备管理逻辑，负责 “查看

装备”“购买装备” 两大功能。

Administrador（管理员子类）：同样继承自 Account，提供查看登录日志、冻结与解冻账户的操作接口，同时调用了 SecurityModule 中的管理账号安全的方法 。

SecurityModule（安全模块）：记录登录失败日志、是否冻结账户检测功能，实现了自动冻结和手动解冻流程。

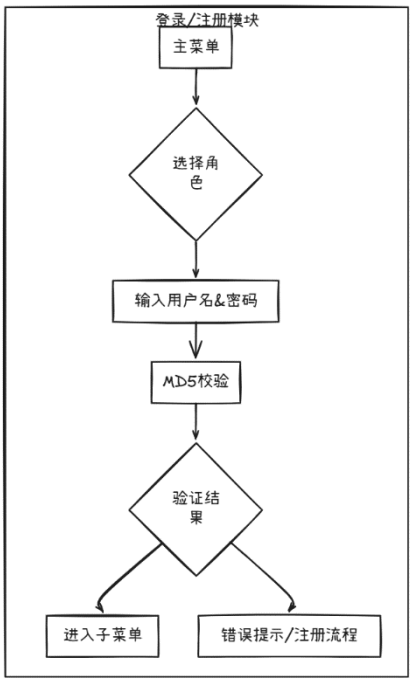
MD5 加密工具：采用了开源代码，通过封装单向哈希算法，在所有密码写入账号文件前均调用 md5()函数，让密码以加密形式存储，杜绝了明文泄露 。

2.2 系统功能架构

用户登录与注册模块

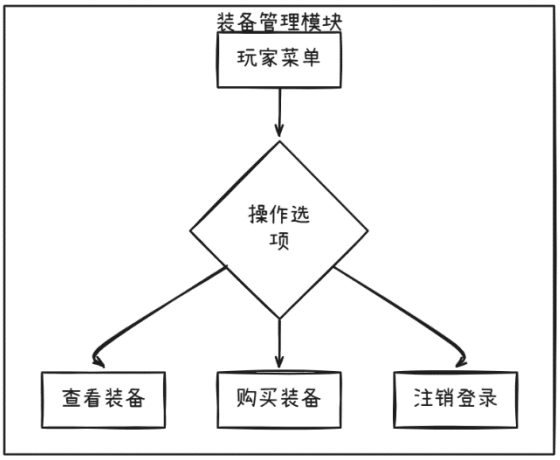
在 main.cpp 中，识别玩家（type=1）或管理员（type=2）的类型，调用相应的 alidatePassword 方法进行身份校验；Gamer 若不存在可注册，自动创建账号文件及默认数据。

装备管理模块（Gamer）



showequipment() 读取并展示玩家已有装备列表；

buyEquipment() 载入商店物品清单，包含了校验余额与防止重复购买的逻辑，完成购买后更新

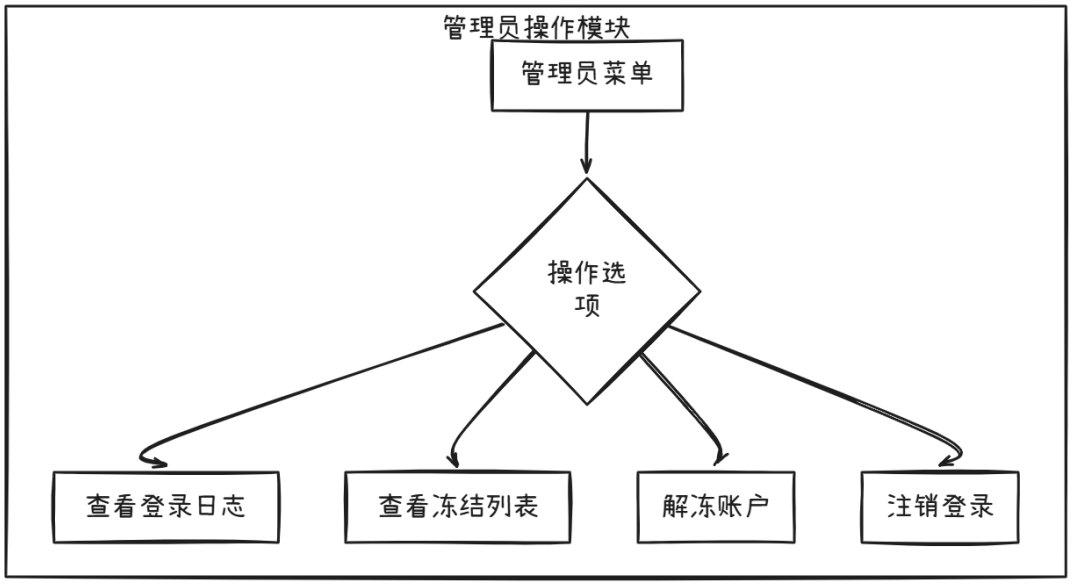


本地用户文件，实现了数据私有化。

管理员操作模块（Administrador）

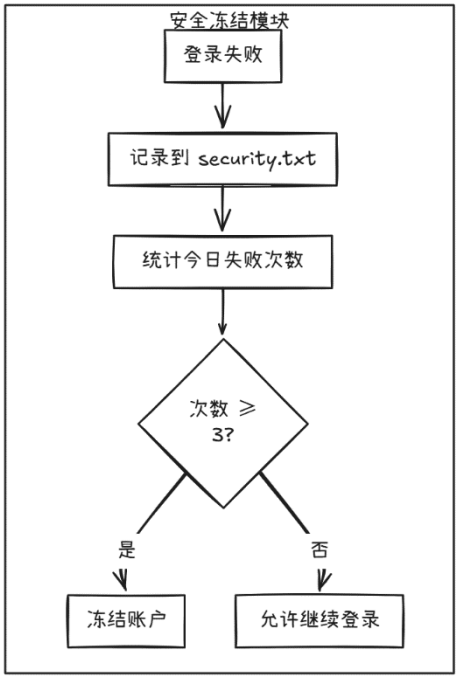
checkLoginAttempts() 展示成功登录日志；

tackleillegalaccount() 展示失败登录记录与冻结列表，并调用 unlockedAccount() 解冻指定用户。



安全控制模块（SecurityModule）

在每次登录前校验 security.txt，统计当日同一用户失败次数，超过三次写入 locked.txt；提供一键冻结与解冻接口，避免了程序被恶意攻击和数据破坏。



2.3 文件数据设计

accounts.txt：

```
+----------------------+
|    accounts.txt      |
+----------------------+
| username             |  ← 登录时用于身份匹配
| MD5(password)        |  ← 单向加密，杜绝明文存储
| type (GAMER/ADMIN)   |  ← 区分玩家与管理员权限
+----------------------+
```

accounts.txt 用于存储所有账号的基础信息。每一行记录一个账号，包含用户名、经 MD5 加密后的密码，以及身份类型（"GAMER" 或 "ADMIN"），以此实现用户权限的区分。在用户注册时，程序会在 LoginIn() 函数中将新玩家的信息写入该文件，并通过 std::ofstream 追加方式保存；而在登录过程中，调用 Account::alidatePassword() 方法，用 std::ifstream 的输入流操作逐字段读取文件内容，按用户名和类型进行匹配，再比对加密后的密码，实现身份的验证与权限识别。

login.txt：

```
+----------------------+
|     login.txt        |
+----------------------+
```

```
| username              |  ← 操作人
| timestamp             |  ← 登录时间（字符串形式）
+-----------------------+
```

login.txt 文件用来记录用户成功登录的信息，包含了用户名和登录的时间。该文件的写入也发生在 LoginIn() 函数中，用户成功通过身份验证后，程序使用 std::ctime() 获取当前时间，并以追加方式写入该文件。管理员可通过调用 SecurityModule::checkLoginAttempts() 查看登录日志。

security.txt：

```
+----------------------+
|    security.txt      |
+----------------------+
| username             |
| timestamp            |
+----------------------+
```

为了统计失败的登录行为，设计了 security.txt 文件。该文件结构与 login.txt 类似，记录尝试登录但密码错误的用户名和时间。在 LoginIn() 的失败分支中，该文件会被追加一条失败记录。系统在用户登录时会调用 SecurityModule::CheckLoginAttemptsAndLock() 函数读取此文件，借助 strftime() 获取当日日期，并统计该用户在当天的失败次数，一旦达到三次即自动触发锁定操作，从而有效防止账户被破解。

locked.txt：

```
+----------------------+
|    locked.txt        |
+----------------------+
| username             |  ← 每行一个被冻结用户名
+----------------------+
```

被锁定的账号将被记录在 locked.txt 文件中。在系统自动冻结时会进行文件写入。当管理员需要解冻账户时，调用 unlockedAccount() 方法会对该文件进行过重写，移除目标用户名，从而恢复登录权限。此外，管理员还可以通过 tackleillegalaccount() 方法读取该文件并展示。

user_data/<用户名>.txt：

```
+--------------------------------+
| user_data/<username>.txt       |
+--------------------------------+
| username                       |  ← 冗余字段，便于校验
| money                          |  ← 当前账户余额
```

| equipment list:　　| ← 分界关键字
|　1. 装备 A　　|
|　2. 装备 B　　|
+--------------------------------+

每个用户在注册成功后，系统会自动为其创建一个 user_data/<用户名>.txt 文件，作为其个人信息的存储单元。文件中包含用户名、当前账户余额以及一组装备清单。在 Gamer::showequipment() 函数中，通过读取该文件并输出装备信息。用户购买装备时，Gamer::buyEquipment() 会先在临时系统内存中更新余额和装备，然后重新写入整个文件内容，从而保证写入过程的安全性和稳定性。

三、核心算法描述

本章将全面介绍游戏账号管理系统的实现机制，包括主程序结构、玩家与管理员的操作流程、装备管理与安全模块的细节实现，以及系统在密码安全性上的具体处理。

3.1 总体实现方案

系统以 main.cpp 中的 main() 作为程序入口（）。核心流程如下：
主循环与菜单

```
while (true) {
  cout << "1.用户登录  2.管理员登录  3.退出系统" << endl;
  cin >> select;
  switch (select) {
    case 1: LoginIn(ACTD_FILE, 1); break;
    case 2: LoginIn(ACTD_FILE, 2); break;
    case 3: return 0;
  }
}
```

系统启动后，用户首先看到一个命令行形式的主菜单界面，菜单中提供三项可选操作，分别是“用户登录”、“管理员登录”以及“退出系统”。用户使用键盘输入所选项的编号（如输入 1 表示用户登录），系统接收用户输入，并将其存入变量 select 中。随后使用 switch-case 结构对输入内容进行分支判断，跳转到对应的功能处理流程。

登录与注册入口：LoginIn()

```
void LoginIn(string fileName, int type) {
  // 输入用户名、密码
  cin >> name >> pwd;
  if (type == 1) { // Gamer
    Gamer* gamer = new Gamer();
    int result = gamer->alidatePassword(name, pwd, fileName);
    // 根据 result 0/1/2/3 处理注册、登录成功与失败
  } else {        // Administrador
    Administrador* admin = new Administrador();
    int result = admin->alidatePassword(name, pwd, fileName);
    // 仅支持登录，返回结果 2 则进入管理员菜单
  }
}
```

当用户选择登录选项时，程序会调用 LoginIn() 函数进入身份验证流程。此函数接收两个参数：第一个参数是用于存储账户信息的文件路径，第二个参数为身份类型代号，其中“1”代表玩家身份，允许注册，“2”代表管理员身份，仅允许登录、禁止注册。根据身份不同，程序将调用对应的登录验证逻辑。若登录成功，将使用动态内存分配生成新的对象，并分别进入玩家菜单 GamerMenu() 或管理员菜单 AdministradorMenu()。

3.2 核心功能逻辑详述

3.2.1 玩家登录验证逻辑

玩家登录的关键流程定义在 Gamer.cpp 文件中。该文件中有一个名为 alidatePassword() 的函数，负责处理登录验证逻辑，其执行顺序如下：

```
int Gamer::alidatePassword(string name, string pwd, string fileName) {
  // (1) 冻结检查
  ifstream lockFile(LOCKED_FILE);
  while (getline(lockFile, line))
    if (line == "username:" + name) return 3;

  // (2) 连续失败次数检查
  if (security.CheckLoginAttemptsAndLock(name)) {
    security.lockAccount(name);
    return 3;
  }

  // (3) 密码比对
  ifstream ifs(fileName);
  while (ifs >> fName >> fPwd >> type) {
    if (fName == name && type=="GAMER")
      return (fPwd == md5(pwd)) ? 2 : 1;
  }
  return 0;
}
```

首先，系统会读取 locked.txt 文件，检查是否存在该账户的冻结记录。如果当前尝试登录的用户名在此文件中存在，系统将直接拒绝登录请求，返回表示“账户已被冻结”的状态码。

若账户未被冻结，程序将调用安全模块中的 CheckLoginAttemptsAndLock() 方法，统计用户当日的登录失败次数。判断依据是登录失败日志文件 SCUL_FILE 中是否存在三条以上的、属于该用户且日期为当天的记录。如果次数超过三次，则系统会自动将该用户账户写入 locked.txt 中，并终止本次登录流程。

如果上面的流程均顺利通过，程序会进行密码比对。系统会打开账户记录文件 accounts.txt，逐行读取每一条记录，匹配用户名与用户类型是否符合“GAMER”，然后将用户输入的密码通过 md5() 函数加密，与文件中存储的加密值进行比对。若两者一致，返回登录成功的标志；若密码不一致，则返回密码错误的状态码；若文件读取结束仍未找到匹配用户，则返回“账户不存在”的标志。

3.2.2 注册流程

当 alidatePassword 返回 0 时，LoginIn() 中执行：

```
if (result==0) {
  ofs.open(fileName, ios::app);
  ofs << name << " " << md5(pwd) << " GAMER" << endl;
  // 创建 user_data 文件夹及 <name>.txt
  userFile << "username: " << name << endl
           << "money: 5000" << endl
           << "equiment list:" << endl
           << "GAMER" << endl;
}
```

若玩家登录验证返回“用户不存在”状态，系统将询问用户是否注册新账号。若用户同意，程序将在 accounts.txt 文件末尾追加一条新记录，记录内容包括用户名、使用 MD5 加密后的密码以及身份标识“GAMER”。此外，系统还会在 user_data 目录下为该用户创建一个专属数据文件，命名为“用户名.txt”，并写入初始化的账户信息。文件包括用户名字段、初始资金（如 5000 金币）以及用于区分装备信息的标识“equiment list:”，该列表用于日后管理装备购买情况。文件最后写入关键字“GAMER”，作为装备读取的终止标记。

3.2.3 装备查看功能

```
void Gamer::showequipment() {
  ifstream file("user_data/"+username+".txt");
  bool show = false;
  while (getline(file, line)) {
    if (line == "equiment list:") { show = true; continue; }
    if (show && !line.empty()) cout << line << endl;
    if (line == "GAMER") break;
  }
}
```

玩家登录成功后，若选择“查看我的装备”，系统将调用 showequipment() 函数。该函数将打开 user_data/用户名.txt 文件，并逐行读取文件内容。程序将查找标记为“equiment list:”的行，从该行起，所有非空行都会被当作用户已拥有的装备进行展示，直到遇到关键字“GAMER”为止。此机制确保装备展示的准确性与完整性，并避免输出非装备内容。

3.2.4 装备购买功能

```
void Gamer::buyEquipment() {
  // (1) 载入商店物品
  ifstream store(IVNC_FILE);
  while (getline(store, line)) parse into equiment_store;
  // (2) 显示列表 & 余额，接收 select
  // (3) 合法性检查：边界、重复、余额
  // (4) 更新 money、ownedEquip
  // (5) 重写 user_data/<name>.txt（含新余额与装备）
}
```

当玩家选择“购买装备”功能时，系统将从商店文件中读取商品数据，并展示可选装备清单。装备信息通常按照“编号. 名称 $价格”的格式存储保存在内存中。

随后程序进入购买循环阶段。用户输入想要购买的装备编号后，系统将执行如下判断流程：首先检查该装备是否已被玩家拥有，若重复购买则提示不可重复购买；然后检查玩家账户余额是否足够购买该装备。若符合条件，则执行扣款操作，并将新购买的装备添加至装备列表中。

完成购买后，系统将重写玩家的数据文件，包括用户名、当前余额、装备列表等内容，从而实现对玩家资产状态的更新。

```
void SecurityModule::checkLoginAttempts() {
  ifstream file(LGIC_FILE);
  while (getline(file, line)) cout << line << endl;
}
void SecurityModule::tackleillegalaccount() {
  // 依次读取 security.txt、locked.txt 输出
  // 接收用户名，调用 unlockedAccount(name)
}
```

3.2.5 管理员功能

管理员登录后进入的菜单由 AdministradorMenu() 函数实现，菜单提供三项功能：查看登录日

志、管理冻结账号以及注销登录。

当管理员选择“查看登录日志”时，系统将读取 LGIC_FILE 文件内容，展示所有历史登录记录。

当管理员选择“管理冻结账号”时，系统会读取 SCUL_FILE（非法登录日志）和 locked.txt（冻结账户记录）两个文件，展示被判定为异常行为的用户记录，并提示管理员输入待解冻的用户名。输入后，程序将执行“解冻账户”操作，移除用户在两个文件中的所有记录，恢复其正常登录权限。

3.2.6 安全模块实现逻辑

安全模块的相关逻辑集中在 SecurityModule.cpp 文件中，主要处理登录失败检测、账户冻结与解冻等操作，是整个系统安全策略的核心部分。

在登录失败检测方面，系统通过调用 CheckLoginAttemptsAndLock() 函数实现对可疑账户的自动冻结。该函数使用 ctime 库函数 strftime() 提取当前日期，并遍历失败日志文件（security.txt），统计当天同一用户名的失败登录次数。

若计数器达到或超过 3 次，函数将返回 true，据此调用 lockAccount() 执行自动冻结操作。

账户冻结的实现逻辑十分简洁，通过向 locked.txt 文件追加一行 "username:<用户名>" 的记录来标识该账户已被冻结。

```
ofstream outFile(LOCKED_FILE, ios::app);
outFile << "username:" << name << endl;
```

当管理员执行解冻操作时，系统调用 unlockedAccount() 函数，从 locked.txt 和 security.txt 中剔除指定用户名的所有记录。

```
ifstream inFile(LOCKED_FILE);
ofstream outFile("temp_locked.txt");
while (getline(inFile, line)) {
    if (line.find(name) == string::npos) {
        outFile << line << endl;
    }
}
remove(LOCKED_FILE);
rename("temp_locked.txt", LOCKED_FILE);
```

日志记录与校验功能由 checkLoginAttempts() 和 tackleillegalaccount() 两个函数负责。

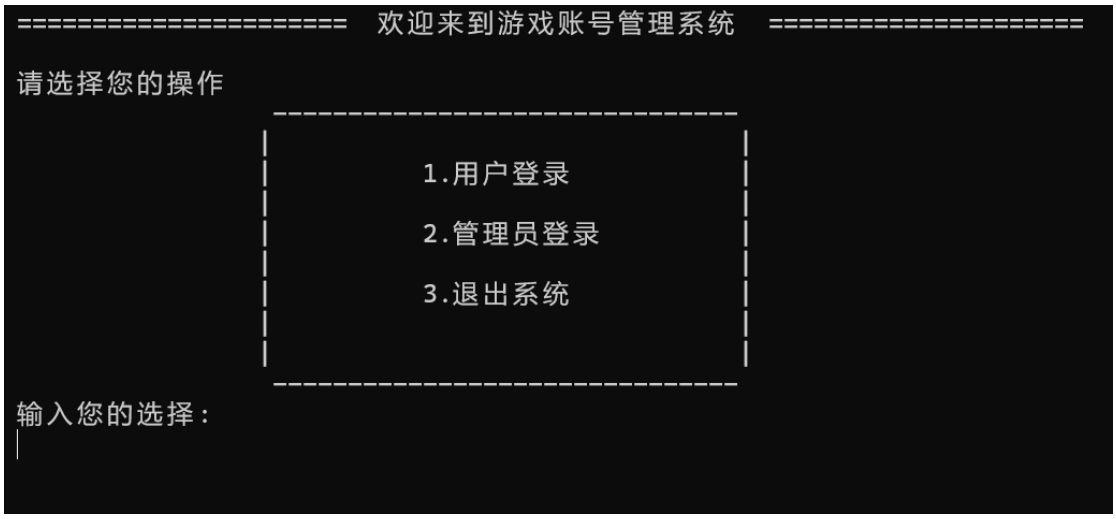
前者用于展示正常登录行为，读取 login.txt 并逐行输出；后者则汇总所有失败登录与被冻结账户，并提供解冻接口，帮助管理员进行账户操作。

3.3 密码加密机制

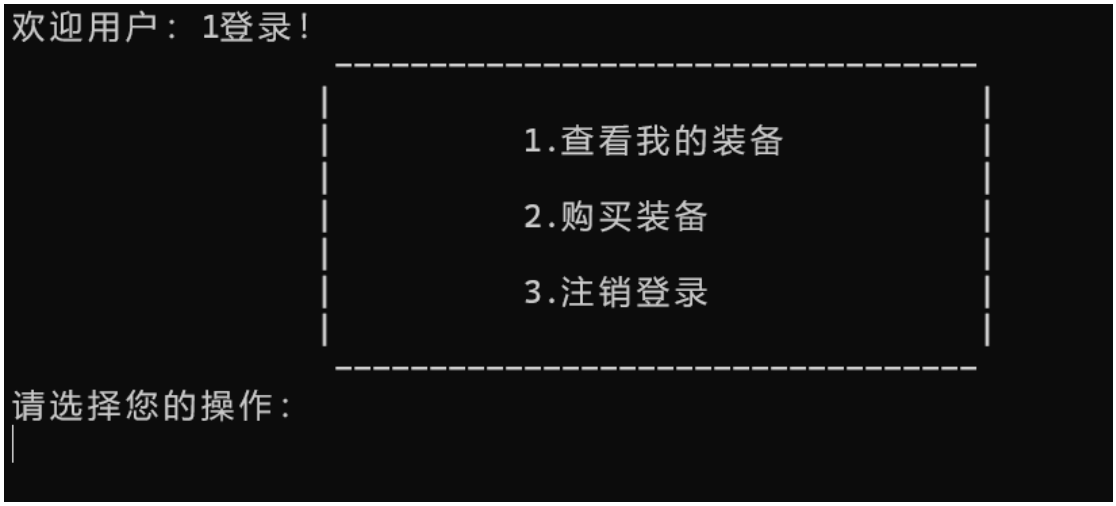
为了提升系统的安全性，所有密码在存储与校验过程中均不以明文形式出现，而是采用 MD5 哈希算法进行加密。该加密函数在 md5.cpp 中实现，通过四轮非线性变换（F、G、H、I 函数）和位移运算，对原始输入字符进行处理，最终生成长度为 32 位的十六进制字符串。用户登录时输入的密码会通过相同方式加密，再与账户文件中的密文进行比对，从而有效防止用户密码泄露。

四、系统实现效果

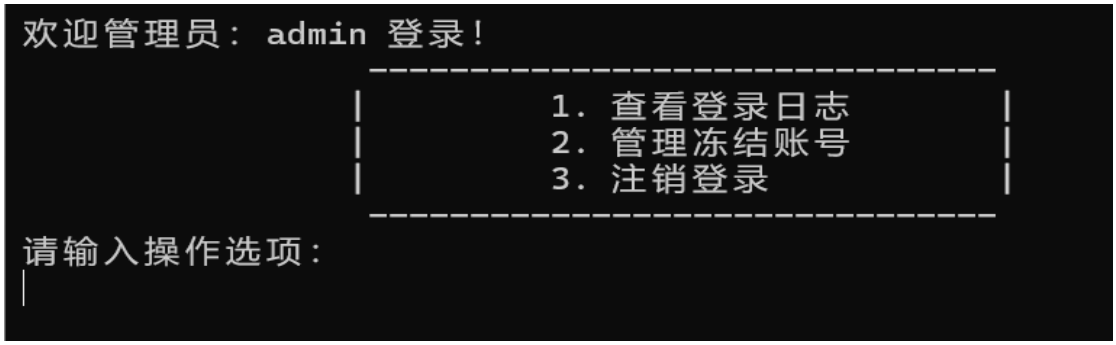
4.1 菜单展示



（程序主菜单）



（玩家主菜单）



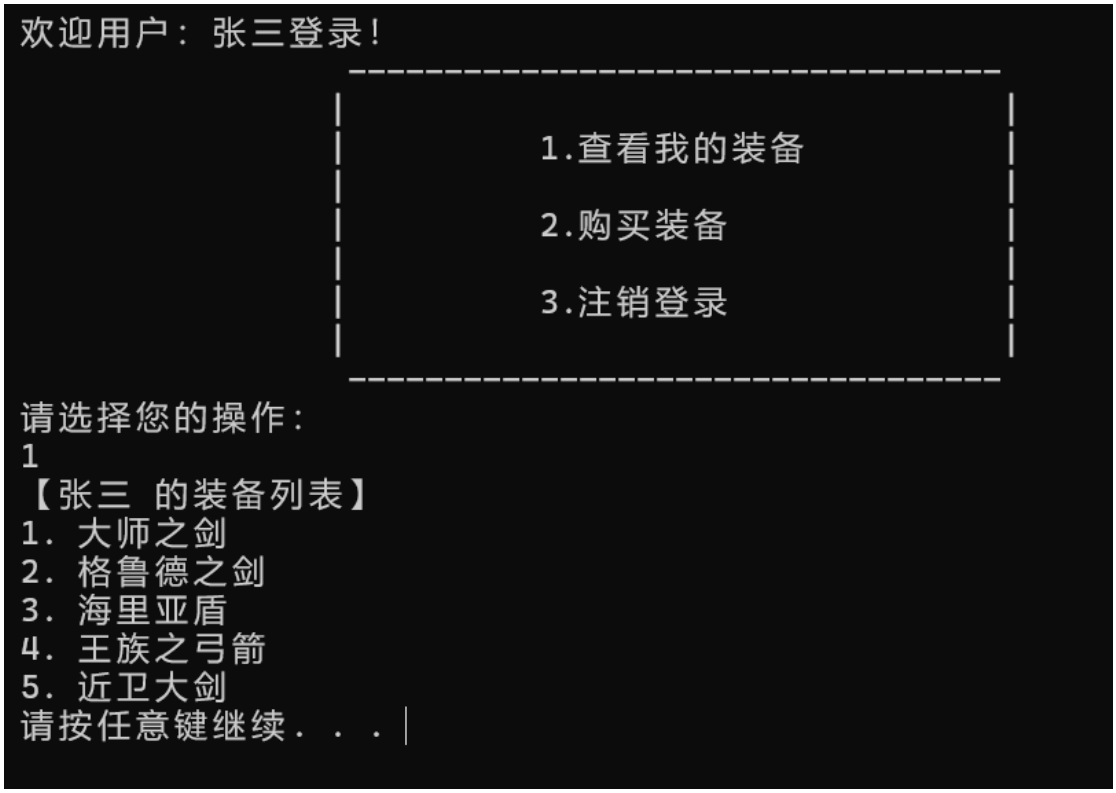
（管理员主菜单）

4.2 玩家注册和登录

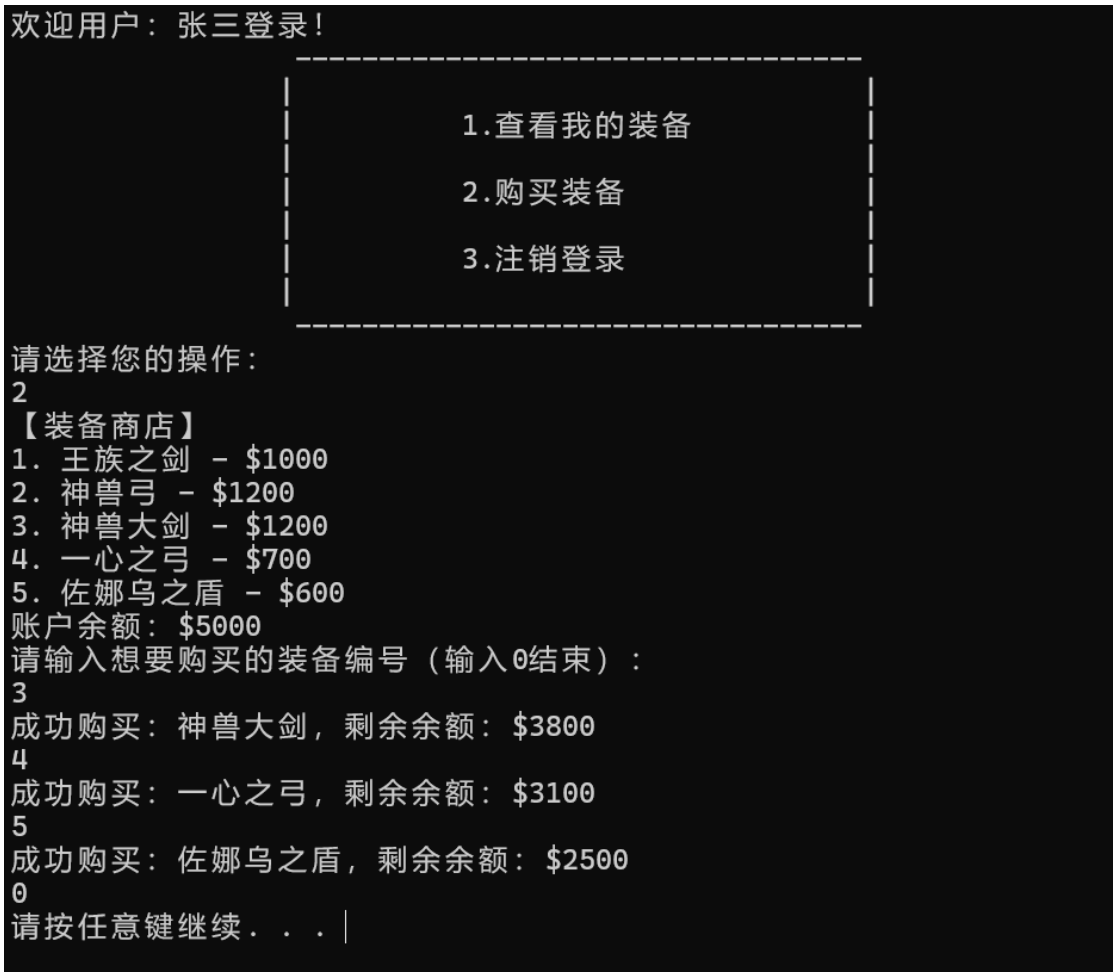
（玩家的注册）

（玩家成功登录）

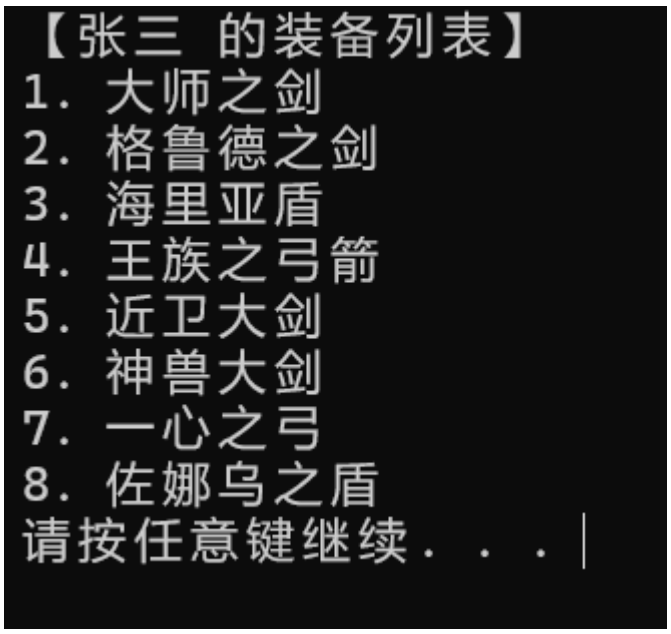
4.3 玩家功能展示



（玩家装备展示）

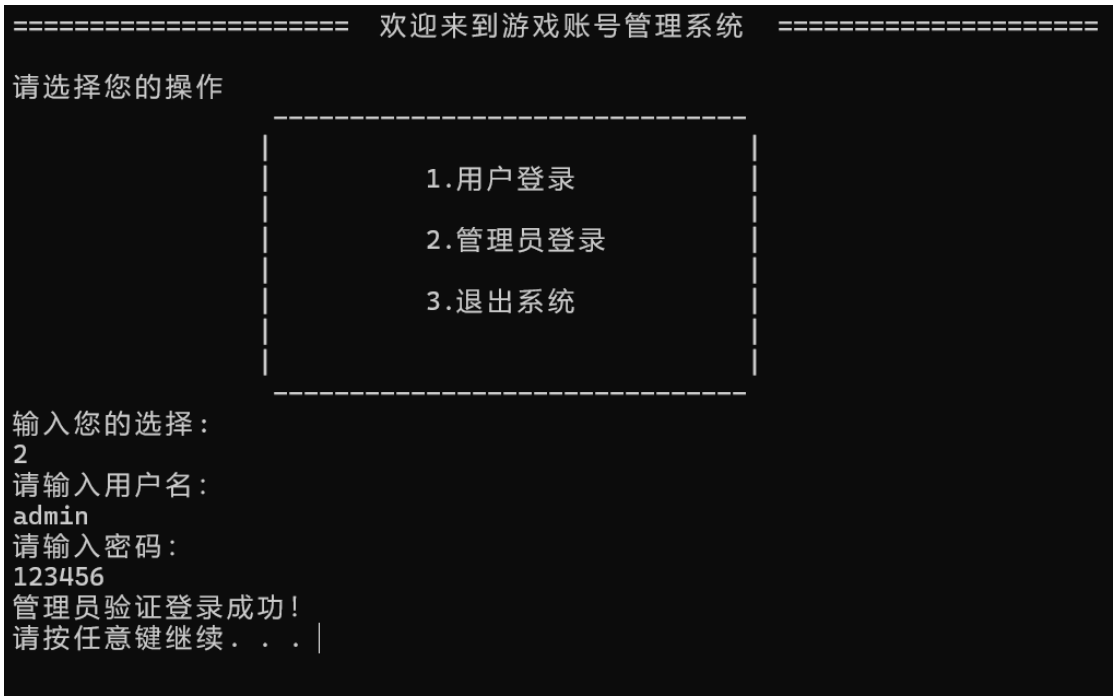


（玩家购买装备）

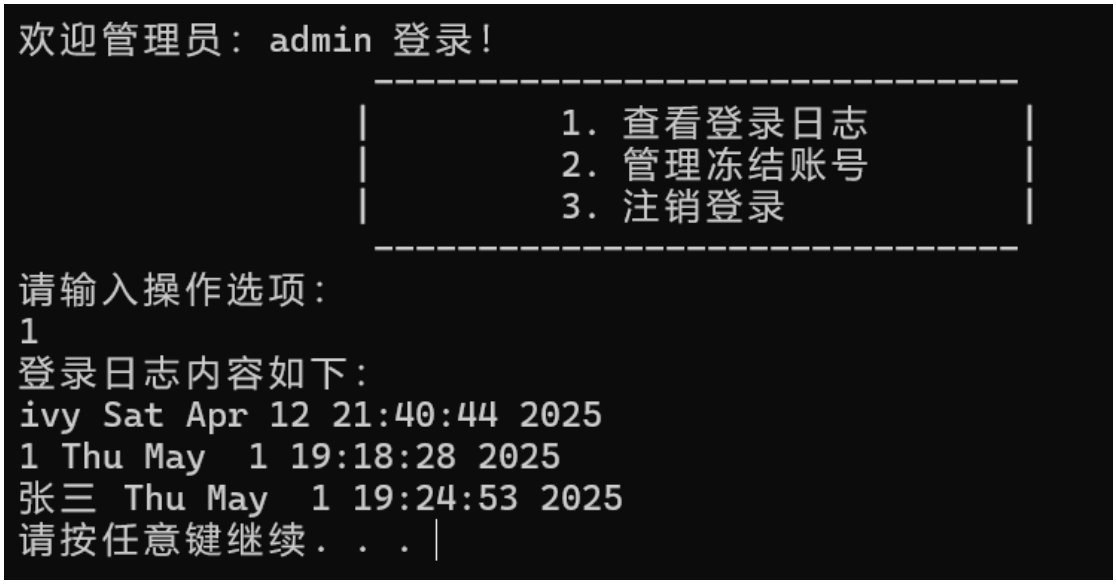


（更新列表展示）

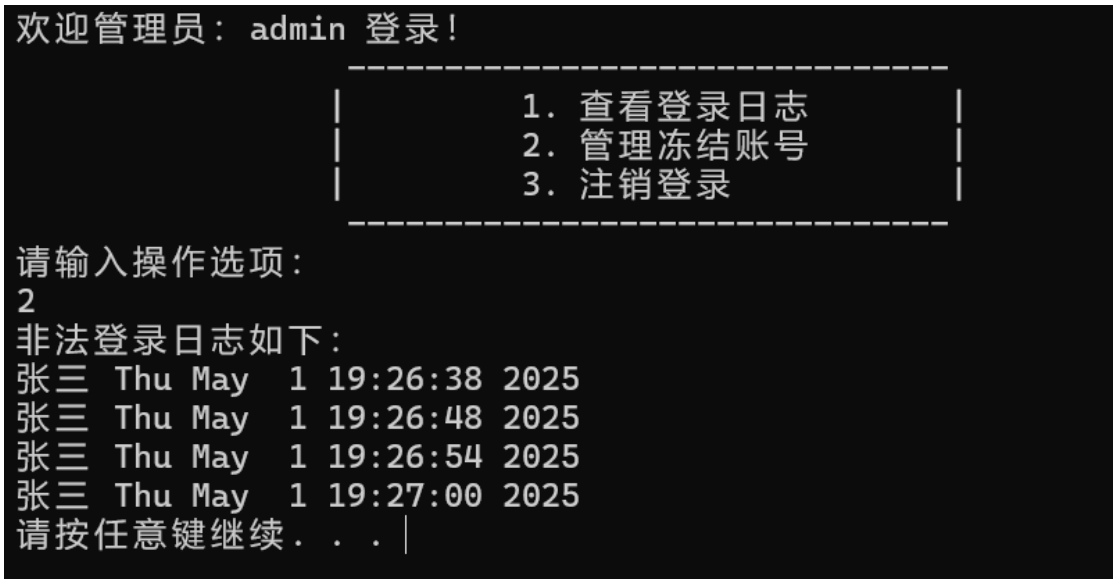
4.4 管理员的登录和功能



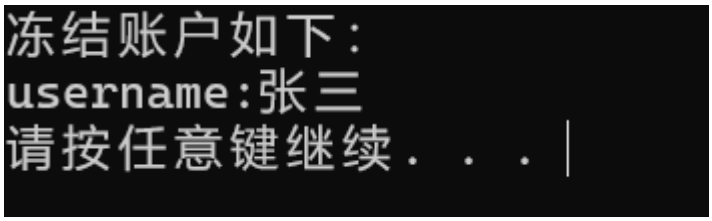
（管理员登录）

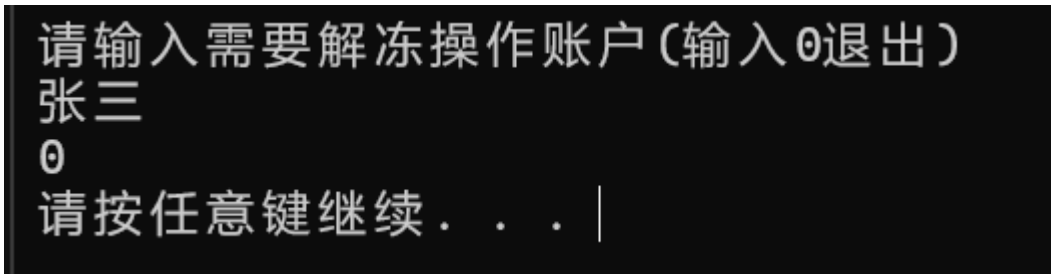


（查看登录日志）

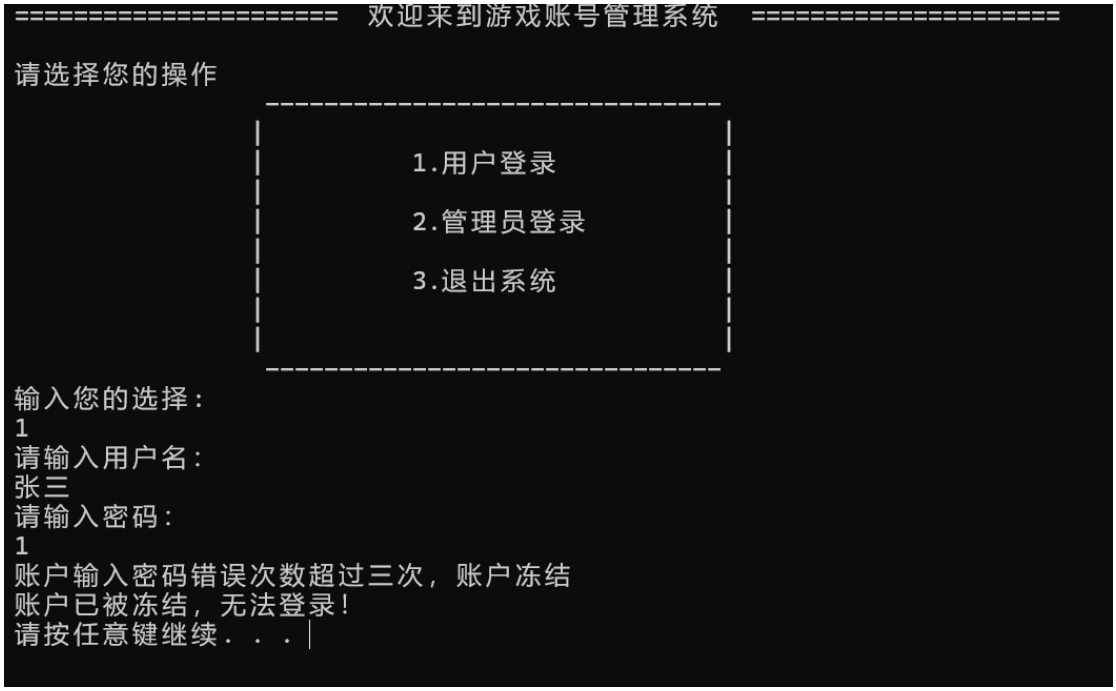

欢迎管理员: admin 登录!
1. 查看登录日志
2. 管理冻结账号
3. 注销登录
请输入操作选项:
2
非法登录日志如下:
张三 Thu May 1 19:26:38 2025
张三 Thu May 1 19:26:48 2025
张三 Thu May 1 19:26:54 2025
张三 Thu May 1 19:27:00 2025
请按任意键继续. . .

（查看非法登录日志）

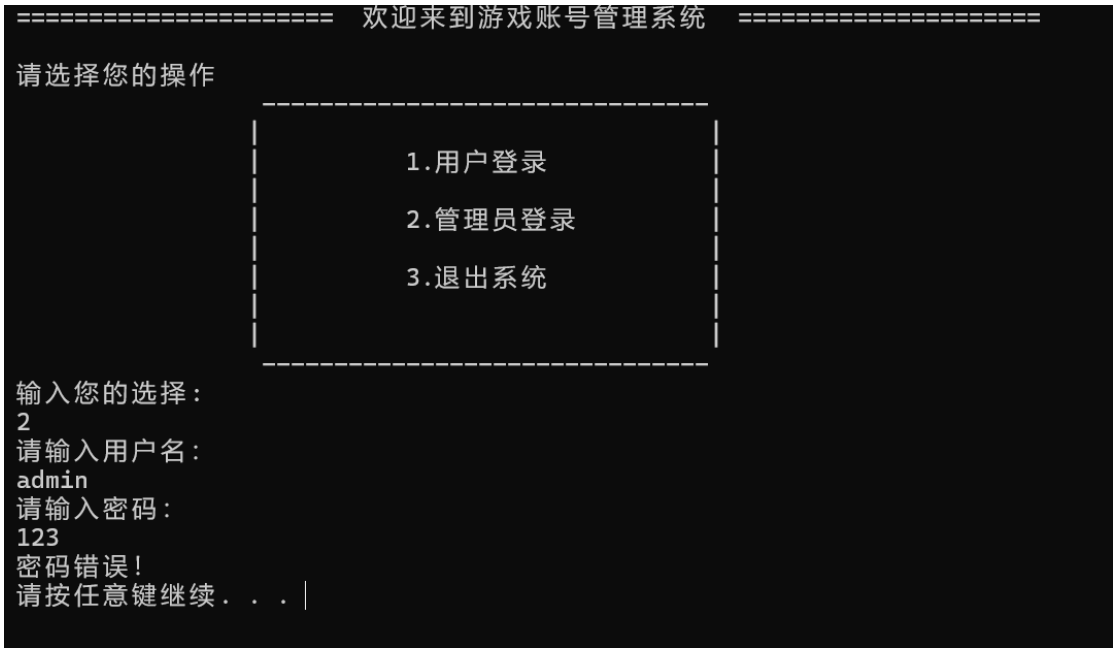

冻结账户如下:
username:张三
请按任意键继续. . .


请输入需要解冻操作账户(输入0退出)
张三
0
请按任意键继续. . .

4.5 错误情况展示



（玩家账户密码错误）



（管理员账户密码错误）

五、心得体会

本次项目为编写一个游戏账号管理系统，。这是我第一次以面向对象思想为核心进行模块化设计开发，虽然过程中遇到了不少挑战，但我们始终坚持不懈，不断尝试解决问题，最终完成了系统的搭建与调试。

在本次项目中，我们使用了类的继承与多态来设计用户和管理员的功能分层，利用文件 IO 技术完成了账户数据的保存与读取，实现了登录验证、装备购买、余额扣费、账户注册等多个功能模块。特别是在实现密码连续输错三次自动冻结账户功能时，我通过对日志文件进行分析比对，结合系统时间进行条件匹配，最终实现了自动封锁机制。这一功能的实现，极大地锻炼了我们对逻辑严密性的把握能力。

在实际开发中，我们也遇到了不少 Bug。例如登录失败日志格式不统一导致无法识别，冻结判断放置顺序不当导致功能失效等。通过这些问题，我进一步学会了如何调试程序、如何精细控制判断条件以及如何合理设计数据流程。

通过这次项目实践，我不仅巩固了 C++ 编程的核心知识，更深刻认识到将知识应用于实际系统开发中的复杂性与挑战性。程序从无到有的过程，既是一场技术训练，也是一种心智成长。回望项目的完成，不仅让我增强了信心，也让我意识到编程不只是冷冰冰的技术，更是一种不断追求逻辑严谨与创造美感的艺术。未来我将继续深入学习和实践，不断提升自身能力，迎接更多更复杂的开发挑战。

附录

Account.h:

```
#pragma once
#include<string>
#include<iostream>
#include<vector>
#include"md5.h"
using namespace std;
using psi = tuple<int,string,int>;
using pis = pair<int, string>;

class Account
{
public:
    virtual void operMenu() = 0;
    virtual int alidatePassword(string name,string pwd,string filename)=0;
    void  buyEquipment() {};

    int classification;
    string username;
    string password_hash;
    vector<pis> equipment_list;
    vector<psi>equiment_store;
    int money=5000;

};
```

Administrador.h:

```
#pragma once
#include<iostream>
#include "Account.h"
#include<fstream>
#include<tuple>
#include"SecurityModule .h"
using namespace std;



class Administrador :public Account
{
```

```
public:

   Administrador();


   Administrador(int id, string name, string pwd);

   virtual void operMenu();

   virtual int alidatePassword(string name, string pwd, string filename);

   SecurityModule security;

};
```

Gamer.h：

```
#include <io.h>
#include <fcntl.h>
#include"SecurityModule .h"


class Gamer :public Account
{
public:

   Gamer();

   Gamer(int index, string name, string pwd);

   virtual void operMenu();

   virtual int alidatePassword(string name, string pwd, string filename);

   void buyEquipment();

   void showequipment();

   SecurityModule security;
};

globalFile .h:
#pragma once
```

```
//账户文件
#define ACTD_FILE     "accountsdat.txt"
//登录成功记录文件
#define LGIC_FILE   "login_logcsv.txt"
//可购买装备
#define IVNC_FILE   "inventorycsv.txt"
//装备目录
#define EQUP_FILE  "equipment.txt"
//登录失败日志
#define SCUL_FILE     "securitylog.txt"
//临时文件
#define LOCKED_FILE  "locked.txt"
```

md5.h:

```
#ifndef MD5_H
#define MD5_H

#include <string>
#include <cstring>
#include <iostream>

class MD5 {
public:
    MD5();
    void update(const unsigned char* input, size_t length);
    void update(const char* input, size_t length);
    MD5& finalize();
    std::string hexdigest() const;
private:
    void transform(const unsigned char block[64]);
    void encode(unsigned char* output, const uint32_t* input, size_t length);
    void decode(uint32_t* output, const unsigned char* input, size_t length);

    bool finalized;
    unsigned char buffer[64];
    uint32_t state[4];
    uint32_t count[2];
    unsigned char digest[16];

    static const unsigned char PADDING[64];
};
```

```
std::string md5(const std::string str);

  #endif
```

SecurityModule .h：

```
#pragma once
#include<iostream>
#include<fstream>
#include<sstream>
#include<string>
#include"globalFile .h"
using namespace std;


class SecurityModule
{
public:
  void checkLoginAttempts() ;
  void lockAccount(const string& name) ;
  bool CheckLoginAttemptsAndLock(const string& name);
  void unlockedAccount(const string& name);
  void tackleillegalaccount();
  };
```

Administrador.cpp：

```
#include "Administrador.h"

Administrador::Administrador() {}

Administrador::Administrador(int id, string name, string pwd) {
  this->username = name;
  this->password_hash = md5(pwd);
  this->classification = id;
}

void Administrador::operMenu() {
  cout << "欢迎管理员：" << this->username << " 登录！" << endl;
  cout << "\t\t ------------------------------\n";
  cout << "\t\t|          1. 查看登录日志       |\n";
```

```
    cout << "\t\t|          2. 管理冻结账号        |\n";
    cout << "\t\t|          3. 注销登录            |\n";
    cout << "\t\t --------------------------------\n";
    cout << "请输入操作选项：" << endl;
}

int Administrador::alidatePassword(string name, string pwd, string fileName)
{
    ifstream ifs(fileName, ios::in);
    if (!ifs.is_open()) {
        cout << "无法打开文件" << endl;
        return 0;
    }

    string fName, fPwd, type;
    while (ifs >> fName >> fPwd >> type) {
        if (fName == name && type == "ADMIN") {
            if (fPwd == md5(pwd)) {
                return 2; // 登录成功
            }
            else {
                return 1; // 密码错误
            }
        }
    }
    return 0; // 不存在该管理员
  }
```

main.cpp：

```
#include"Account.h"
#include"SecurityModule .h"
#include"Gamer.h"
#include"globalFile .h"
#include"Administrador.h"
#include <direct.h> // 用于创建文件夹 mkdir()
#include <sys/stat.h> // 检查文件夹是否存在
#include<Windows.h>
#include<ctime>
using namespace std;

//进入玩家菜单
void GamerMenu(Account* &a_Gamer )
```

```
{
    while (true)
    {

        a_Gamer->operMenu();//展示菜单

        Gamer* Gam = (Gamer*) a_Gamer;//转换指针类型

        int select = 0;
        cin >> select;

        if (select == 1)
        {
            Gam->showequipment();

        }
        else if (select == 2)
        {
            Gam->buyEquipment();

        }
        else
        {

            delete Gam;
            cout << "注销成功" << endl;
            system("pause");
            system("cls");
            return;
        }

    }
}

void AdministradorMenu(Account*& a_Administrador)
{
    while (true)
    {
        //展示管理员菜单
        a_Administrador->operMenu();

        Administrador* Admin = (Administrador*)a_Administrador;//强制指针转换
```

```
    int select = 0;
        cin >> select;

        if (select == 1)
        {

            Admin->security.checkLoginAttempts();

        }
        else if (select == 2)
        {
            Admin->security.tackleillegalaccount();
        }
        else
        {
            delete Admin;
            cout << "注销成功" << endl;
            system("pause");
            system("cls");
            return;
        }

    }
}


void LoginIn(string fileName, int type)
{
    //父类指针 ，指向子类对象
    Account* person = NULL;

    //读文件
    ifstream ifs;
    ifs.open(fileName, ios::in);

    //判断文件是否存在
    if (!ifs.is_open())
    {
        cout << "文件不存在" << endl;
        ifs.close();
        return;
    }
```

```
    string name;
    string pwd;


    cout << "请输入用户名：" << endl;
    cin >> name;

    cout << "请输入密码：" << endl;
    cin >> pwd;

    if (type == 1)
    {
        Gamer* gamer = new Gamer();  // 临时 gamer 对象
        person = gamer;


        int result = person->alidatePassword(name, pwd, fileName);



        if (result == 2) {
            cout << "学生验证登录成功！" << endl;
            system("pause");
            system("cls");

            delete gamer;
            person = new Gamer(type, name, pwd);
            GamerMenu(person);


            time_t now = std::time(0);
            char* dt = std::ctime(&now); // 转为字符串格式
            dt[strlen(dt) - 1] = '\0'; // 去掉末尾的换行符
            ofstream loginFile(LGIC_FILE,ios::app);
            if (loginFile.is_open()) {
                loginFile << name << ' ' << dt << endl;
                loginFile.close();
            }
            return;
        }
        else if (result == 1) {
            cout << "密码错误！" << endl;

           time_t now = time(0);//获取当前的系统时间
```

```
            tm* ltm = localtime(&now);//将系统时间 now 转换为本地时间（tm 结构体）
            char dt[64];
            strftime(dt, sizeof(dt), "%a %b %d %H:%M:%S %Y", ltm);//使用 strftime 格式化时
间，生成字符串形式的时间，保存到字符数组 dt 中
            ofstream faileloginFile(SCUL_FILE, ios::app);
            if (faileloginFile.is_open()) {
                faileloginFile << name << ' ' <<dt<< endl;
                faileloginFile.close();
            }
            system("pause");
            system("cls");
            delete gamer;
            return;
        }
        else if(result==0) {
            cout << "账号未注册！" << endl;
            cout << "是否注册？Y/N：" << endl;
            char put_in;
            cin >> put_in;
            if (put_in == 'Y' || put_in == 'y') {
                ofstream ofs(fileName, ios::app);
                if (!ofs.is_open()) {
                    cout << "写入失败，无法打开文件" << endl;
                    delete gamer;
                    return;
                }
                ofs << name << " " << md5(pwd)<<" GAMER" << endl;
                ofs.close();
                cout << "注册成功，请重新登录！" << endl;
            }
            else {
                cout << "即将返回主菜单" << endl;

                system("pause");
                system("cls");
                return;
            }


            string folder = "user_data";
            string filename = folder + "/" + name + ".txt";

            ofstream userFile(filename);
            if (userFile.is_open()) {
```

```
                userFile << "username: " << name << endl;
                userFile << "money: " << 5000 << endl;
                userFile << "equiment list:" << endl;

                UINT orignalCP = GetConsoleCP();
                SetConsoleCP(CP_UTF8);
                ifstream eqFile(EQUP_FILE);
                string line;
                while (getline(eqFile, line)) {
                    userFile << line << endl;
                }

                eqFile.close();

                userFile << "GAMER" << endl;
                userFile.close();

                SetConsoleCP(orignalCP);

                cout << "注册成功！用户数据文件已创建：" << filename << endl;
                system("pause");
                system("cls");
                return;
            }
            else {
                cout << "无法创建用户专属文件。" << endl;

            }




        }
        else if(result==3)
        {
            cout << "账户已被冻结，无法登录！" << endl;
            system("pause");
            system("cls");
            delete gamer;
            return;
        }

}
```

```
        else if (type == 2)
    {
        Administrador* admin = new Administrador();
        person = admin;

        int result = person->alidatePassword(name, pwd, fileName);

        if (result == 2) {
            cout << "管理员验证登录成功！" << endl;
            system("pause");
            system("cls");



            delete admin;
            person = new Administrador(type, name, pwd);

            AdministradorMenu(person);



            return;
        }
        else if (result == 1) {
            cout << "密码错误！" << endl;
            system("pause");
            system("cls");
            delete admin;
            return;
        }
        else {
            cout << "管理员账号不存在！禁止注册管理员。" << endl;
            system("pause");
            system("cls");
            delete admin;
            return;
        }

    }

    cout << "验证登录失败！" << endl;
    system("pause");
    system("cls");

    return;
}
```

```
int main() {

    int select = 0;
    while (1) {
        cout << "=====================  欢迎来到游戏账号管理系统  ====================="
            << endl;
        cout << endl << "请选择您的操作" << endl;
        cout << "\t\t ------------------------------\n";
        cout << "\t\t|                              |\n";
        cout << "\t\t|          1.用户登录          |\n";
        cout << "\t\t|                              |\n";
        cout << "\t\t|          2.管理员登录        |\n";
        cout << "\t\t|                              |\n";
        cout << "\t\t|          3.退出系统          |\n";
        cout << "\t\t|                              |\n";
        cout << "\t\t|                              |\n";
        cout << "\t\t ------------------------------\n";
        cout << "输入您的选择: "<<endl;


        cin >> select;

        switch (select)
        {
        case 1:
            LoginIn(ACTD_FILE, 1);
            break;
        case 2:
            LoginIn(ACTD_FILE, 2);
            break;
        case 3:
            cout << "欢迎下一次使用" << endl;
            system("pause");
            return 0;
            break;
        default:
            cout << "输入有误，请重新选择！" << endl;
            system("pause");
            system("cls");
            break;
        }
}
```

```
}
```

Gamer.cpp:

```
#include"Gamer.h"



Gamer::Gamer(){
}


Gamer::Gamer(int index, string name, string pwd){

    this->username = name;
    this->password_hash = md5(pwd);
    this->classification= index;


    ifstream ifs;
    ifs.open(ACTD_FILE, ios::in);

    ifs.close();
}

//菜单界面
void Gamer::operMenu(){
    cout << "欢迎用户：" << this->username << "登录！" << endl;
    cout << "\t\t ---------------------------------\n";
    cout << "\t\t|                                 |\n";
    cout << "\t\t|          1.查看我的装备         |\n";
    cout << "\t\t|                                 |\n";
    cout << "\t\t|          2.购买装备             |\n";
    cout << "\t\t|                                 |\n";
    cout << "\t\t|          3.注销登录             |\n";
    cout << "\t\t|                                 |\n";
    cout << "\t\t ---------------------------------\n";
    cout << "请选择您的操作： " << endl;
}

int Gamer::alidatePassword(string name, string pwd, string fileName)
```

```
{
    // 检查该账户是否被锁定
    ifstream lockFile(LOCKED_FILE);
    string lockedLine;
    while (getline(lockFile, lockedLine)) {
        if (lockedLine == "username:" + name) {
            cout << "该账户已被冻结，无法登录。" << endl;
            return 3; // 表示账户已被冻结
        }
    }
    lockFile.close();

    ifstream ifs(fileName, ios::in);
    if (this->security.CheckLoginAttemptsAndLock(name)) {
        cout << "账户输入密码错误次数超过三次，账户冻结" << endl;
        this->security.lockAccount(name);
        return 3;
    }


   string fName, fPwd,type;
   while (ifs >> fName >> fPwd>>type) {
        if (fName == name) {
            if (fPwd == md5(pwd)&&type=="GAMER") {
                return 2; // 登录成功
            }
            else {
                return 1; // 密码错误
            }
        }

   }
   return 0; // 用户名不存在
}


void Gamer::showequipment() {


   // 用户数据文件路径
   string filepath = "user_data/" + this->username + ".txt";
   ifstream file(filepath);
   if (!file.is_open()) {
        cout << "无法打开用户装备文件: " << filepath << endl;
```

```
        return;
    }

    string line;
    int lineCount = 0;
    bool show = false;

    cout << "【" << this->username << " 的装备列表】" << endl;

    while (getline(file, line)) {
        lineCount++;

        if (line == "equiment list:") {
            show = true;
            continue;
        }
        if (line == "GAMER") {
            break;
        }

        if (show && !line.empty()) {
            cout << line << endl;
        }
    }

    file.close();


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

void Gamer::buyEquipment() {


    ifstream storeFile(IVNC_FILE);
    equiment_store.clear();  // 清空旧记录
    string line;
    while (getline(storeFile, line)) {
        stringstream ss(line);
        int index;
        string name, token;
        int price;
```

```
        ss >> index;
        ss.ignore(2);  // 跳过 ". "

        while (ss >> token) {
            if (token[0] == '$') {
                price = stoi(token.substr(1));
                break;
            }
            if (!name.empty()) name += " ";
            name += token;
        }

        equiment_store.push_back(make_tuple(index, name, price));
    }
    storeFile.close();

    UINT orignalCP = GetConsoleCP();


    cout << "【装备商店】" << endl;


    for (auto& item : equiment_store) {
        cout << get<0>(item) << ". " << get<1>(item) << " - $" << get<2>(item) << endl;
    }


    cout << "账户余额：$" << money << endl;
    cout << "请输入想要购买的装备编号（0结束）：" << endl;

    // 获取用户数据文件路径
    string userFile = "user_data/" + this->username + ".txt";

    vector<string> currentLines;
    ifstream inFile(userFile);


    string fileLine;
    while (getline(inFile, fileLine)) {
        currentLines.push_back(fileLine);
    }
    inFile.close();
```

```
// 查找已有装备
vector<string> ownedEquip;
bool inEquipSection = false;
for (size_t i = 0; i < currentLines.size(); i++) {
    if (currentLines[i] == "equiment list:") {
        inEquipSection = true;
        continue;
    }
    if (inEquipSection && !currentLines[i].empty()) {
        size_t pos = currentLines[i].find(". ");
        if (pos != string::npos) {
            ownedEquip.push_back(currentLines[i].substr(pos + 2));  // 装备名称
        }
    }
}


int select = 0;
while (cin >> select && select != 0) {
    if (select < 1 || select >(int)equiment_store.size()) {
        cout << "编号无效，请重新输入。" << endl;
        continue;
    }

    string itemName = get<1>(equiment_store[select - 1]);
    int itemPrice = get<2>(equiment_store[select - 1]);



    if (find(ownedEquip.begin(), ownedEquip.end(), itemName) != ownedEquip.end()) {
        cout << "你已拥有该装备，无法重复购买。" << endl;
        continue;
    }

    if (money < itemPrice) {
        cout << "余额不足，无法购买 " << itemName << endl;
        continue;
    }



    money -= itemPrice;
    ownedEquip.push_back(itemName);
```

```
        cout << "成功购买：" << itemName << "，剩余余额：$" << money << endl;
        SetConsoleCP(orignalCP);
    }

    // 更新文件内容
    ofstream outFile(userFile);

    outFile << " username: " << username << endl;
    outFile << "money: " << money << endl;
    outFile << "equiment list:" << endl;

    int count = 1;
    SetConsoleCP(CP_UTF8);
    for (const string& eq : ownedEquip) {
        outFile << count++ << ". " << eq << endl;
    }
    SetConsoleCP(orignalCP);

    outFile.close();
    system("pause");
    system("cls");
}
```

SecurityModule.cpp：

```
#include"SecurityModule .h"

void SecurityModule::unlockedAccount(const string& name) {
    // 删除 locked.txt 中的账户
    ifstream inFile(LOCKED_FILE);
    ofstream outFile("temp_locked.txt");

    string line;
    while (getline(inFile, line)) {
        if (line.find(name) == string::npos) {
            outFile << line << endl;
        }
    }
    inFile.close();
    outFile.close();
    remove(LOCKED_FILE);
    rename("temp_locked.txt", LOCKED_FILE);
```

```
    //删除 log.txt 中该账户的所有记

    ifstream logIn(SCUL_FILE);//创建一个 ifstream 输入文件流对象 logIn
    ofstream logOut("temp_security.txt");//创建一个 ofstream 输出文件流对象 logOut

    while (getline(logIn, line)) {
        if (line.find(name + " ") != 0) { // 检查是否是该账户的记录
            logOut << line << endl;
        }
    }
    logIn.close();
    logOut.close();
    remove(SCUL_FILE);
    rename("temp_security.txt", SCUL_FILE);
}

void SecurityModule::checkLoginAttempts() {
    ifstream file(LGIC_FILE);
    string line;
    cout << "登录日志内容如下：" << endl;
    while (getline(file, line)) {
        cout << line << endl;
    }

    file.close();
    system("pause");
    system("cls");

}

void SecurityModule::lockAccount(const string& name) {
    ofstream outFile(LOCKED_FILE, ios::app); // 以追加模式打开
    if (outFile.is_open()) {
        outFile<<"username:" << name << endl;
        outFile.close();
    }


}

bool SecurityModule::CheckLoginAttemptsAndLock(const string& name) {
    ifstream file(SCUL_FILE);
```

```
        // 获取今天的日期字符串（只保留年月日）
    time_t now = time(0);
    tm* ltm = localtime(&now);
    char today[20];
    strftime(today, sizeof(today), "%a %b %d", ltm); //确保格式

    int failCount = 0;
    string line;

    while (getline(file, line)) {
        // 判断是否以该用户名开头
        if (line.find(name + " ") == 0) {
            // 判断日期是否是今天
            if (line.find(today) != string::npos) {
                failCount++;
            }
        }
    }

    file.close();

    if (failCount >= 3) {
        return true;

    }
    else
    {
        return false;
    }
}

void SecurityModule::tackleillegalaccount() {
    ifstream securityfile(SCUL_FILE);
    string lines;
    cout << "非法登录日志如下：" << endl;
    while (getline(securityfile, lines)) {
        cout << lines << endl;
    }

    securityfile.close();
    system("pause");
    system("cls");

    ifstream lockedfile(LOCKED_FILE);
```

```
    string liness;
    cout << "冻结账户如下：" << endl;
    while (getline(lockedfile, liness)) {
        cout << liness << endl;
    }

    lockedfile.close();
    system("pause");
    system("cls");

    cout << "请输入需要解冻操作账户(输入0退出)" << endl;
    string unlockedusername;
    while (1) {
        if (unlockedusername == "0") break;
        cin >> unlockedusername;
        unlockedAccount(unlockedusername);
    }
    system("pause");
    system("cls");
}
```

md5.cpp：

```
#include "md5.h"
#include <cstdio>
#include <sstream>
#include <iomanip>

const unsigned char MD5::PADDING[64] = { 0x80 };

#define F(x, y, z) ((x & y) | (~x & z))
#define G(x, y, z) ((x & z) | (y & ~z))
#define H(x, y, z) (x ^ y ^ z)
#define I(x, y, z) (y ^ (x | ~z))

#define ROTATE_LEFT(x, n) ((x << n) | (x >> (32 - n)))

#define FF(a, b, c, d, x, s, ac) { a += F(b, c, d) + x + ac; a = ROTATE_LEFT(a, s); a += b; }
#define GG(a, b, c, d, x, s, ac) { a += G(b, c, d) + x + ac; a = ROTATE_LEFT(a, s); a += b; }
#define HH(a, b, c, d, x, s, ac) { a += H(b, c, d) + x + ac; a = ROTATE_LEFT(a, s); a += b; }
#define II(a, b, c, d, x, s, ac) { a += I(b, c, d) + x + ac; a = ROTATE_LEFT(a, s); a += b; }
```

```
MD5::MD5() : finalized(false) {
    count[0] = count[1] = 0;
    state[0] = 0x67452301;
    state[1] = 0xefcdab89;
    state[2] = 0x98badcfe;
    state[3] = 0x10325476;
}

void MD5::update(const unsigned char* input, size_t length) {
    size_t index = count[0] / 8 % 64;
    count[0] += (uint32_t)(length << 3);
    if (count[0] < (length << 3)) count[1]++;
    count[1] += (uint32_t)(length >> 29);

    size_t firstpart = 64 - index;
    size_t i = 0;
    if (length >= firstpart) {
        memcpy(&buffer[index], input, firstpart);
        transform(buffer);
        for (i = firstpart; i + 63 < length; i += 64)
            transform(&input[i]);
        index = 0;
    }
    memcpy(&buffer[index], &input[i], length - i);
}

void MD5::update(const char* input, size_t length) {
    update((const unsigned char*)input, length);
}

MD5& MD5::finalize() {
    static unsigned char bits[8];
    encode(bits, count, 8);

    size_t index = count[0] / 8 % 64;
    size_t padLen = (index < 56) ? (56 - index) : (120 - index);
    update(PADDING, padLen);
    update(bits, 8);

    encode(digest, state, 16);
    finalized = true;
    return *this;
}
```

```
void MD5::transform(const unsigned char block[64]) {
    uint32_t a = state[0], b = state[1], c = state[2], d = state[3], x[16];
    decode(x, block, 64);

    FF(a, b, c, d, x[0], 7, 0xd76aa478);
    FF(d, a, b, c, x[1], 12, 0xe8c7b756);
    FF(c, d, a, b, x[2], 17, 0x242070db);
    FF(b, c, d, a, x[3], 22, 0xc1bdceee);
    FF(a, b, c, d, x[4], 7, 0xf57c0faf);
    FF(d, a, b, c, x[5], 12, 0x4787c62a);
    FF(c, d, a, b, x[6], 17, 0xa8304613);
    FF(b, c, d, a, x[7], 22, 0xfd469501);
    FF(a, b, c, d, x[8], 7, 0x698098d8);
    FF(d, a, b, c, x[9], 12, 0x8b44f7af);
    FF(c, d, a, b, x[10], 17, 0xffff5bb1);
    FF(b, c, d, a, x[11], 22, 0x895cd7be);
    FF(a, b, c, d, x[12], 7, 0x6b901122);
    FF(d, a, b, c, x[13], 12, 0xfd987193);
    FF(c, d, a, b, x[14], 17, 0xa679438e);
    FF(b, c, d, a, x[15], 22, 0x49b40821);

    GG(a, b, c, d, x[1], 5, 0xf61e2562);
    GG(d, a, b, c, x[6], 9, 0xc040b340);
    GG(c, d, a, b, x[11], 14, 0x265e5a51);
    GG(b, c, d, a, x[0], 20, 0xe9b6c7aa);
    GG(a, b, c, d, x[5], 5, 0xd62f105d);
    GG(d, a, b, c, x[10], 9, 0x02441453);
    GG(c, d, a, b, x[15], 14, 0xd8a1e681);
    GG(b, c, d, a, x[4], 20, 0xe7d3fbc8);
    GG(a, b, c, d, x[9], 5, 0x21e1cde6);
    GG(d, a, b, c, x[14], 9, 0xc33707d6);
    GG(c, d, a, b, x[3], 14, 0xf4d50d87);
    GG(b, c, d, a, x[8], 20, 0x455a14ed);
    GG(a, b, c, d, x[13], 5, 0xa9e3e905);
    GG(d, a, b, c, x[2], 9, 0xfcefa3f8);
    GG(c, d, a, b, x[7], 14, 0x676f02d9);
    GG(b, c, d, a, x[12], 20, 0x8d2a4c8a);

    HH(a, b, c, d, x[5], 4, 0xfffa3942);
    HH(d, a, b, c, x[8], 11, 0x8771f681);
    HH(c, d, a, b, x[11], 16, 0x6d9d6122);
    HH(b, c, d, a, x[14], 23, 0xfde5380c);
    HH(a, b, c, d, x[1], 4, 0xa4beea44);
    HH(d, a, b, c, x[4], 11, 0x4bdecfa9);
```

```
    HH(c, d, a, b, x[7], 16, 0xf6bb4b60);
    HH(b, c, d, a, x[10], 23, 0xbebfbc70);
    HH(a, b, c, d, x[13], 4, 0x289b7ec6);
    HH(d, a, b, c, x[0], 11, 0xeaa127fa);
    HH(c, d, a, b, x[3], 16, 0xd4ef3085);
    HH(b, c, d, a, x[6], 23, 0x04881d05);
    HH(a, b, c, d, x[9], 4, 0xd9d4d039);
    HH(d, a, b, c, x[12], 11, 0xe6db99e5);
    HH(c, d, a, b, x[15], 16, 0x1fa27cf8);
    HH(b, c, d, a, x[2], 23, 0xc4ac5665);

    II(a, b, c, d, x[0], 6, 0xf4292244);
    II(d, a, b, c, x[7], 10, 0x432aff97);
    II(c, d, a, b, x[14], 15, 0xab9423a7);
    II(b, c, d, a, x[5], 21, 0xfc93a039);
    II(a, b, c, d, x[12], 6, 0x655b59c3);
    II(d, a, b, c, x[3], 10, 0x8f0ccc92);
    II(c, d, a, b, x[10], 15, 0xffeff47d);
    II(b, c, d, a, x[1], 21, 0x85845dd1);
    II(a, b, c, d, x[8], 6, 0x6fa87e4f);
    II(d, a, b, c, x[15], 10, 0xfe2ce6e0);
    II(c, d, a, b, x[6], 15, 0xa3014314);
    II(b, c, d, a, x[13], 21, 0x4e0811a1);
    II(a, b, c, d, x[4], 6, 0xf7537e82);
    II(d, a, b, c, x[11], 10, 0xbd3af235);
    II(c, d, a, b, x[2], 15, 0x2ad7d2bb);
    II(b, c, d, a, x[9], 21, 0xeb86d391);

    state[0] += a;
    state[1] += b;
    state[2] += c;
    state[3] += d;
}

void MD5::encode(unsigned char* output, const uint32_t* input, size_t length) {
    for (size_t i = 0, j = 0; j < length; i++, j += 4) {
        output[j] = (unsigned char)(input[i] & 0xff);
        output[j + 1] = (unsigned char)((input[i] >> 8) & 0xff);
        output[j + 2] = (unsigned char)((input[i] >> 16) & 0xff);
        output[j + 3] = (unsigned char)((input[i] >> 24) & 0xff);
    }
}

void MD5::decode(uint32_t* output, const unsigned char* input, size_t length) {
```

```
    for (size_t i = 0, j = 0; j < length; i++, j += 4) {
        output[i] = ((uint32_t)input[j]) |
            (((uint32_t)input[j + 1]) << 8) |
            (((uint32_t)input[j + 2]) << 16) |
            (((uint32_t)input[j + 3]) << 24);
    }
}

std::string MD5::hexdigest() const {
    if (!finalized) return "";

    std::ostringstream result;
    for (int i = 0; i < 16; ++i)
        result << std::hex << std::setw(2) << std::setfill('0') << (int)digest[i];
    return result.str();
}

std::string md5(const std::string str) {
    MD5 md5;
    md5.update(str.c_str(), str.length());
    md5.finalize();
    return md5.hexdigest();
}
```