如何管理我的 mhy 账号

2024 年 9 月 23 日

1 项目概述

1.1 项目内容及要求

- **情景要求**

小陈是一名原神玩家，她经常玩游戏，并且开了不同的账号。不过小陈一直有一个困惑：“我这么多的账号，有一些号的细节经常会搞混，能不能想一个办法，像提瓦特小助手那样，管理我的 mhy 账号呢？”

- 思路设计

题目要求使用链表的链表。那我们针对账号，可以分为两个层次。

第一层链表是针对多个账号设计的链表，第二层链表是针对一个账号里面不同角色的链表管理。通过三层结构：data、node、link 来进行书写。

对于第一层链表，我们定义了链表类：Accout，节点类：AccoutNode，数据类：AccoutData，详细解释参考 2 类的设计.

对于第二层链表，我们定义了链表类：Hero，节点类：Node，数据类：Data，详细解释参考 2 类的设计.

1.2 研究人员及分工

序号	学号	姓名	角色及具体贡献
1	23121538	郭咏钦	● 组长 ● PPT 制作 ● 报告撰写 ● 项目框架制作
2	23121537	王旭	Accout 类的完善
3	19121751	武辰龙	主程序测试的书写
4	23121994	肖雅文	Hero 类的完善

2 类的设计

2.1 Hero链表

- Data 结构体

Data 结构体只定义了一些数据：代表角色名字的 name，代表抽取此角色消耗的原石数量 stone。

```
std::string name;
    int stones;
```

- Node 结点类

Node 结点类包含的属性是数据 Data 结构体和指针 Next，用来指向下一个结点。

```
Data data;
Node* next;
```

Node 结点类只拥有两个方法，一个是 Node 结点类的初始化，另一个是友元函数的声明。

```
Node(const Data& data);     //Node 类的初始化，只输入数据，指针为空指针
friend class Hero;     //友元函数的申明
```

- Hero 链表

Hero 链表类包括两个属性，一个是结点类的头指针 head，另一个是结点的个数 num

```
Node* head;
int num;
```

Hero 链表类有六个方法，分别是类的初始化、链表的展示、链表的清空（删）、链表的连接新结点（增）、结点的查找（查）、结点个数的统计。

```
Hero(int n, const Data* datas); //类的初始化

void showList();    //链表的展示

void freeList();    //清空链表

void append(const Data& data);  //结点的增加

bool find(const std::string& name); //结点的数据查找

int heronums() const;   //结点的个数
```

这六个方法，展示了数据结构的一些基本操作，并且很好的满足了项目需求。

2.2 Accout链表

Accout 链表是第一层的结构，其中的结点数据包含了第二层链表 Hero 链表。

- AccoutData 结构体

AccoutData 定义了两个数据，其中之一为结点的账号 UID，另一个便是第二层链表，也就是这个账号的详细信息。

```
Hero link;
int UID;
```

- AccoutNode 结点类

  AccoutNode 结点类属性包含数据 AccoutDate，和指针 next 指向下一个结点。

```
AccoutData data;
AccoutNode* next;
```

  AccoutNode 结点类方法包括了类的初始化和友元函数的定义。

```
AccoutNode(const AccoutData& data);     //类的初始化，只输入数据，指针为空指针
friend class Accout;     //友元函数的申明
```

- Accout 链表类

  Accout 链表类同 Hero 链表类一样，属性为头结点 head 和结点数量 num。

```
AccoutNode* head;
int num;
```

Accout 方法有 7 个，分别是初始化、展示链表、清空链表、追加结点（增）、查找结点（查）、删除结点（删）以及返回结点的个数。

```
Accout(AccoutData* arr, int n); //类的初始化

void showAccoutList();  //展示链表

void freeLink();    //清空链表

void append(const AccoutNode& node);    //追加结点

Hero find(const int& UID);  //查找结点

void deleteAccout(const int& UID);  //删除结点

int accoutnums() const; //结点的个数
```

3 测试情况

3.1 测试样例设计

3.1.1 基本功能测试

测试结果请看视频（单击可访问）

3.2 测试结果对程序的改进情况

- 优化了查找