

**《面向对象程序设计实践（C++）》课程实验报告**



**题目: 物流管理平台设计**

**姓 名 鄭毓恒**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**班 级 2020211302**

**学 号 2020211262**

**任课教师 双锴**

**2022年 6 月**

**题目一：物流业务管理子系统和用户管理系统子系统**

**题目要求**

设计平台支持新用户注册或老用户登录，登录后用户可以修改账户密码、管理余额、发送快递、接收快递以及查询快递。平台也支持物流公司管理员登录，管理员可以查看所有用户信息和所有快递的信息，并支持查询。题目要求用户和快递数据持久化。

**高层数据信息设计**

全局常量定义

#define RECEIVE 1 //已接收

#define NOT\_RECEIVE 0 //未接收

#define SPLIT '.' //文件数据分隔符

全局数据结构定义

/\*

\* 快递寄/收件人信息

\* ID用户号

\* name用户姓名

\* addr地址

\* phone电话

\*/

typedef struct

{

string ID;

string name;

string addr;

string phone;

}PackUserInfo;

全局变量定义

User\* user = NULL; //存储登录中的用户的信息

Admin\* admin = NULL; //存储登录中的管理员的信息

vector<Pack> packList; //相关快递列表

vector<User> allUserList; //所有用户列表，仅管理员可用

fstream userInfoFile; //读写用户信息文件流

fstream adminInfoFile; //读写管理员信息文件流

fstream packInfoFile; //读写快递信息文件流

**关键类的设计**

**用户类**

用户类成员变量包括用户ID、姓名、电话等用户信息，用户可进行的操作由不同成员函数执行。成员函数的执行需要全局变量packList, userInfoFile和packInfoFile。

|  |  |
| --- | --- |
| **成员变量** | **用户信息** |
| public | |
| string ID | 用户ID |
| string name | 用户姓名 |
| string phone | 用户电话 |
| string addr | 用户地址 |
| int balance | 用户余额 |
| private | |
| string pw | 用户密码 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| User() | 默认构造函数 |
| User(const string& id, const string& ipw, const string& iname, const string& iphone, const string& iaddr, const int& ibalance) | 用于注册或从文件读取输入的构造函数 |
| ~User() | 默认析构函数 |
| void regis() | 用户注册 |
| void login() | 用户登录 |
| void output() const | 输出用户信息 |
| private | |
| void loadUserPacklist() | 读取与该用户相关的快递，存在向量packList中 |
| void userInterface() | 用户界面 |
| void userInfoInterface() | 用户个人信息管理界面 |
| void userPackInterface() | 用户物流信息管理界面 |
| void changePW() | 更改密码 |
| void checkBalance() const | 查看余额 |
| void topUp() | 充值 |
| void regis(const string& id) | 在登录时使用不存在用户ID时使用，注册id |
| void login(const string& id) | 在注册时使用已存在用户ID时使用，登录id |
| int checkPwValid(const string& newPw) const | 检查newPw是否符合密码格式要求，符合则返回1，否则返回0 |
| void sendPack() | 发送快递 |
| void recvPack() const | 接收快递 |
| void searchPack() const | 查询快递 |
| void loadPacketUserInfo(PackUserInfo& sender) const | 加载寄件人信息 |
| void loadPacketUserInfo(const streampos& loc, PackUserInfo& recver) const | 加载收件人信息，streampos loc是寄件人信息在用户信息文件的位置 |
| void searchPackBySender() const | 根据寄件人信息查询快递 |
| void searchPackBySenderID() const | 根据寄件人用户ID查询快递 |
| void searchPackBySenderName() const | 根据寄件人姓名查询快递 |
| void searchPackBySenderPhone() const | 根据寄件人电话查询快递 |
| void searchPackBySenderAddr() const | 根据寄件人地址查询快递 |
| void searchPackByRecver() const | 根据收件人信息查询快递 |
| void searchPackByRecverID() const | 根据收件人用户ID查询快递 |
| void searchPackByRecverName() const | 根据收件人姓名查询快递 |
| void searchPackByRecverPhone() const | 根据收件人电话查询快递 |
| void searchPackByRecverAddr() const | 根据收件人地址查询快递 |
| void searchPackBySendTime() const | 根据发送时间查询快递 |
| void searchPackByPackID() const | 根据快递号查询快递 |
| int checkUnrecvPack() const | 输出未接收快递 |
| streampos searchID() const | 从文件中查找该对象id的用户信息，返回在文件中的位置，不存在则返回-1 |
| streampos searchID(const string& id) const | 从文件中查找用户ID为id的用户信息，返回在文件中的位置，不存在则返回-1 |
| void writeToFile(int mode) const | 将用户信息写入文件，mode=0表示写入新用户信息，mode=1表示更改原有用户的信息 |

**管理员类**

管理员类成员变量包括管理员ID、姓名、余额等信息，管理员可进行的操作由不同成员函数执行。成员函数的执行需要全局变量packList, allUserList, userInfoFile, adminInfoFile和packInfoFile。

|  |  |
| --- | --- |
| 成员变量 | 管理员信息 |
| public | |
| string ID | 管理员ID |
| string name | 管理员姓名 |
| int balance | 余额。余额是所有管理员共用的，单独存储在文件的第一行 |
| private |  |
| string pw | 管理员密码 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Admin() | 默认构造函数 |
| ~Admin() | 默认析构函数 |
| void login() | 管理员登录 |
| private | |
| void adminInterface() | 管理员界面 |
| void loadAllPacklist() const | 管理员有权访问所有快递，从文件读取所有快递 |
| void loadAllUserList() const | 管理员有权访问所有用户，从文件读取所有用户 |
| void adminUserInterface() const | 管理员管理用户信息界面 |
| void adminPackInterface() const | 管理员管理物流信息界面 |
| void checkAllUser() const | 查看所有用户 |
| void checkAllPack() const | 查看所有快递 |
| void checkBalance() const | 查询管理员余额 |
| void searchUser() const | 查询用户 |
| void searchPack() const | 查询快递 |
| void searchUserByID() const | 以用户ID查询用户 |
| void searchUserByName() const | 以用户姓名查询用户 |
| void searchUserByPhone() const | 以用户电话查询用户 |
| void searchUserByAddr() const | 以用户地址查询用户 |
| void searchPackBySender() const | 根据寄件人信息查询快递 |
| void searchPackBySenderID() const | 根据寄件人用户ID查询快递 |
| void searchPackBySenderName() const | 根据寄件人姓名查询快递 |
| void searchPackBySenderPhone() const | 根据寄件人电话查询快递 |
| void searchPackBySenderAddr() const | 根据寄件人地址查询快递 |
| void searchPackByRecver() const | 根据收件人信息查询快递 |
| void searchPackByRecverID() const | 根据收件人用户ID查询快递 |
| void searchPackByRecverName() const | 根据收件人姓名查询快递 |
| void searchPackByRecverPhone() const | 根据收件人电话查询快递 |
| void searchPackByRecverAddr() const | 根据收件人地址查询快递 |
| void searchPackBySendTime() const | 根据发送时间查询快递 |
| void searchPackByPackID() const | 根据快递号查询快递 |

**快递类**

快递类成员变量包括快递号、快递状态、发送时间等信息，快递可进行的操作由不同成员函数执行。成员函数的执行需要全局变量packList, adminInfoFile和packInfoFile。

|  |  |
| --- | --- |
| 成员变量 | 快递信息 |
| public | |
| string packID | 快递号，由寄件人用户号和发送时间构成 |
| int state | 接收状态 |
| SYSTEMTIME sendTime | 发送时间 |
| SYSTENTIME recvTime | 接收时间 |
| PackUserInfo sender | 寄件人信息 |
| PackUserInfo recver | 收件人信息 |
| string packInfo | 快递详情 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Pack() | 默认构造函数 |
| Pack(const string& ipackID, const int& istate, const SYSTEMTIME& isendTime, const SYSTEMTIME& irecvTime, const PackUserInfo& isender, const PackUserInfo& irecver, const string& ipackInfo) | 用于读取文件输入的构造函数 |
| Pack(const string& ipackID, const SYSTEMTIME& isendTime, const PackUserInfo& isender, const PackUserInfo& irecver, const string& ipackInfo) | 用于寄件的构造函数 |
| ~Pack() | 默认析构函数 |
| void output() const | 输出快递信息 |
| void sendPack() const | 发送快递 |
| void recvPack() | 接收快递 |

**总体设计**

用户、快递和管理员信息分别存储在userInfo.txt、packInfo.txt和adminInfo.txt三个文件当中，三个文本文件存放在名为data的文件夹中，程序运行时必须把data文件夹放到相同目录当中。

文件userInfo.txt存储所有用户信息，每一行为一个用户的信息。用户信息按用户ID、密码、姓名、电话、地址、余额这个顺序存储在同一行，每一项用’.’分隔开。文件packInfo.txt存储所有快递信息，每一行为一个快递的信息。快递信息按快递号、接收状态、发送时间、接收时间（如果快递未接收则没有接收时间）、寄件人信息、收件人信息、快递详情这个顺序存储在同一行，每一项用’.’分隔开。文件adminInfo.txt存储所有管理员信息，除了文件第一行外，每一行为一个管理员的信息。文件第一行存储管理员余额，所有管理员共用一个余额。管理员信息按管理员ID、密码、姓名这个顺序存储在同一行，每一项用’.’分隔开。

程序执行时，首先打开这三个文件，然后让用户表明身份。如果是新用户则调用User::regis()；如果是老用户则调用User::login()；如果是管理员则调用Admin::login()。

**用户使用时**

新用户注册时首先要求输入新用户ID，要求长度为8-12个字符，只能包含数字、大小写英文字母以及下划线。输入并确定输入合法后，利用searchID()函数检查有无和老用户的ID重合，如果没有重复则继续注册，如果有重复则询问是否用输入的ID并调用login()进行登录，或者重新输入ID。然后要求用户输入密码，密码只能含有大小写字母和数字，长度在8-10个字符，利用checkPwValid()函数检查密码的合法性。然后让用户输入自己的姓名、电话和地址，最后调用writeToFile()函数将新用户信息写入文件，再调用userInterface()函数进入用户界面。

老用户登录时首先要求输入用户ID，检查输入的合法性以及是否为已存在的用户。如果不是则询问是否用输入的ID并调用regis()进行注册，或者重新输入ID，如果是则从文件获取ID的密码，要求用户输入密码。用户输入正确的密码后，加载用户的姓名、电话等其他信息，再调用userInterface()函数进入用户界面。

进入用户界面后，首先调用loadUserPacklist()函数加载所有与该用户相关的快递信息。该函数从文件中读取快递，并将收件人或寄件人为当前用户的快递存入packList中。接着，用户界面提示用户选择进入管理个人信息界面或物流功能界面，分别调用userInfoInterface()函数和userPackInterface()函数。

进入个人信息管理界面后，用户可选择更改密码、查询余额和充值，将分别调用changePW()、checkBalance()和TopUp()函数。除了checkBalance()函数直接输出用户余额外，其他两个均需要输入密码。输入正确密码后，再输入要改的密码或充值金额，检查合法性后调用writeToFile()函数及时写入文件。

在物流功能界面，用户可选择发送快递、接收快递和查询快递，将分别调用sendPack()、recvPack()和searchPack()函数。

发送快递首先检查用户的余额是否足够，足够则要求用户输入收件人ID，再检查合法性和ID是否存在。如果输入的ID合法，则调用LoadPacketUserInfo()从文件读入收件人的信息，然后让用户输入快递详情。快递号利用寄件人ID和精准到秒的发送时间组成，而快递信息中的发送时间只精准到分钟。接着，构造一个Pack类对象，然后调用Pack::sendPack()发送快递。该函数将新发送的快递写入文件和加入packList后，再更改管理员文件中的余额，返回。最后，用户余额减去运费，再调用writeToFile()函数更改文件中的信息。

接受快递首先从packList中检查是否有未接收的快递，有就全部输出。然后，用户可输入要接收的快递的快递号，检查合法性和快递号是否存在。如果输入合法，调用Pack::recvPack()进行接收。该函数将更改快递状态并添加接收时间，然后更改文件中信息。

查询快递支持寄件人、收件人、发送时间和快递号多种查询，需要用户选择查询方式。只需要在packList中查询快递，不需要访问文件。

**管理员使用时**

管理员不能注册，只能登录。登录时首先要求输入ID，然后检查输入的ID是否在文件中存在，存在则提示用户输入密码，再对输入的密码进行判断。如果密码正确，登录成功，加载管理员姓名和余额，然后调用adminInterface()函数进入管理员界面。

进入管理员界面后，首先调用loadAllPacklist()和loadAllUserList()函数加载所有快递信息和用户信息。这两个函数从文件中读取信息，存入packList和allUserList当中。接着，界面提示管理员选择进入管理用户信息界面或管理物流信息界面，分别调用adminInfoInterface()函数和adminPackInterface()函数。

进入管理用户信息界面后，提示管理员可选择查看所有用户或按条件进行查询，并分别调用checkAllUser()和searchUser()函数。查询用户信息只需要访问allUserList，不需访问文件。查询用户支持用户ID、姓名、电话和地址四种查询方式，需要管理员选择。

进入管理物流信息界面后，提示管理员可选择查看所有快递、查看余额或按条件查询快递，并分别调用checkAllPack()、checkBalance()和searchPack()函数。查询快递只需要访问packList，不需访问文件。查询快递支持寄件人、收件人、发送时间和快递号多种查询，需要管理员选择查询方式。

**实现中的重要问题和解决方案**

程序要求支持多种方式查询用户和快递，如果在文件中进行查找则代码较为复杂。因此，登录完成后预先读入可能需要的用户和快递信息，存储进allUserList和packList中，简化查找部分代码。

在程序中，需要多次对文件流指针进行移动，可用seekg()函数。测试过程中发现，当文件流指针到达文件结尾EOF时，无法使用seekg()函数。因此，每次使用seekg()函数前，调用clear()函数清楚文件流指针状态。

程序中使用cin、getline、getchar等多种输入方式，一次输入后的缓冲区残留可能会影响下一次输入，如cin后换行符遗留在缓冲区内，影响接着的getline。因此，输入后使用ignore()函数，使用参数LLONG\_MAX和’\n’，表示跳过缓冲区LLONG\_MAX个字节或直到换行符为止（将跳过换行符）。

**题目二：快递员任务管理子系统**

**题目要求**

在题目一的基础上加上快递员的角色和功能。管理员可添加或删除快递员以及为快递分配快递员。快递员可以进行揽收和查询快递。同时，为快递进行分类，每一类有不同的运费计算方式。快递除了未接收和已接收，加上未揽收状态，查询快递的方式要求新增按快递状态查询。这一部分报告主要对新增或变更部分进行说明。

**高层数据信息定义**

**全局常量定义**

#define NOT\_PICK 0 //未揽收

#define NORMAL 0

#define FRAGILE 1

#define BOOK 2

**全局数据结构定义**

/\*

\* 快递所需的快递员信息

\* 快递员ID

\* name快递员姓名

\* phone快递员电话

\*/

typedef struct

{

string ID;

string name;

string phone;

}PackPostmanInfo;

**全局变量定义**

Postman\* postman = NULL; //存储登录中的快递员的信息

vector<Postman> allPostmanList; //所有快递员列表，仅管理员可用

fstream postmanInfoFile; //读写快递员信息文件流

**关键类的设计**

**用户类**

用户可使用的功能增加了按快递状态查询快递，因此新增相关的成员函数。

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| private | |
| void searchPackByState() const | 根据快递状态查询快递 |
| int checkUnrecvPackIn() const | 输出寄给当前用户的未接收快递 |
| int checkUnpickPackIn() const | 输出寄给当前用户的未揽收快递 |
| int checkRecvPackIn() const | 输出寄给当前用户的已接收快递 |
| int checkUnrecvPackOut() const | 输出当前用户寄出的未接收快递 |
| int checkUnpickPackOut() const | 输出当前用户寄出的未揽收快递 |
| int checkRecvPackOut() const | 输出当前用户寄出的已接收快递 |

**管理员类**

管理员类新增了与管理快递员相关以及按快递状态查询快递的成员函数。

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| private | |
| void loadAllPostmanList() const | 管理员有权查看所有快递员，从文件中读取并将所有快递员信息存储到allPostmanList |
| void adminPostmanInterface() const | 管理员管理快递员信息界面 |
| void searchPackByState() const | 根据快递状态查询快递 |
| int outputUnrecvPack() const | 输出未接收快递 |
| int outputUnpickPack() const | 输出未揽收快递 |
| int outputRecvPack() const | 输出已接收快递 |
| void checkAllPostman() const | 查看所有快递员 |
| void addPostman() const | 新增快递员 |
| void deletePostman() const | 删除快递员 |
| int searchPostman(const string& id) const | 查看ID为参数id的快递员是否存在，是则返回在allPostman向量的下标，否则返回-1 |
| int checkPwValid(const string& newPw) const | 检查输入的新密码是否符合格式要求，是则返回1，否则返回0 |
| void addPostmanToFile() const | 向快递员信息文件中写入新快递员信息 |
| void deletePostmanFromFile(const string& id) const | 从快递员信息文件删除ID为参数id的快递员信息 |
| void assignPack() const | 为快递分配快递员 |
| int checkUnassignPack() const | 输出未分配快递员的快递 |
| void searchPostman() const | 查询快递员 |
| void searchPostmanByID() const | 以ID查询快递员 |
| void searchPostmanByName() const | 以姓名查询快递员 |
| void searchPostmanByPhone() const | 以电话查询快递员 |
| void showAllPackAssignedToOne() const | 输出分配给指定一个快递员的所有快递 |
| void searchPackAssignedToOne() const | 在指定的一个快递员负责的快递中查找快递 |
| void searchAssignPackBySender(const string& id) const | 在指定的一个快递员负责的快递中根据寄件人信息查询快递 |
| void searchAssignPackBySenderID(const string& id) const | 在指定的一个快递员负责的快递中根据寄件人用户ID查询快递 |
| void searchAssignPackBySenderName(const string& id) const | 在指定的一个快递员负责的快递中根据寄件人姓名查询快递 |
| void searchAssignPackBySenderPhone(const string& id) const | 在指定的一个快递员负责的快递中根据寄件人电话查询快递 |
| void searchAssignPackBySenderAddr(const string& id) const | 在指定的一个快递员负责的快递中根据寄件人地址查询快递 |
| void searchAssignPackByRecver(const string& id) const | 在指定的一个快递员负责的快递中根据收件人信息查询快递 |
| void searchAssignPackByRecverID(const string& id) const | 在指定的一个快递员负责的快递中根据收件人用户ID查询快递 |
| void searchAssignPackByRecverName(const string& id) const | 在指定的一个快递员负责的快递中根据收件人姓名查询快递 |
| void searchAssignPackByRecverPhone(const string& id) const | 在指定的一个快递员负责的快递中根据收件人电话查询快递 |
| void searchAssignPackByRecverAddr(const string& id) const | 在指定的一个快递员负责的快递中根据收件人地址查询快递 |
| void searchAssignPackBySendTime(const string& id) const | 在指定的一个快递员负责的快递中根据发送时间查询快递 |
| void searchAssignPackByPackID(const string& id) const | 在指定的一个快递员负责的快递中根据快递号查询快递 |
| void searchAssignPackByState(const string& id) const | 在指定的一个快递员负责的快递中根据快递状态查询快递 |
| int outputUnrecvAssignPack(const string& id) const | 在指定的一个快递员负责的快递中输出未接收快递 |
| int outputUnpickAssignPack(const string& id) const | 在指定的一个快递员负责的快递中输出未揽收快递 |
| int outputRecvAssignPack(const string& id) const | 在指定的一个快递员负责的快递中输出已接收快递 |

**快递员类**

快递员类成员变量包括ID、姓名、电话等快递员信息，快递员可进行的操作由不同成员函数执行。成员函数的执行需要全局变量packList, postmanInfoFile和packInfoFile。

|  |  |
| --- | --- |
| 成员变量 | 快递员信息 |
| public | |
| string ID | 快递员ID |
| string pw | 密码 |
| string name | 姓名 |
| string phone | 电话 |
| int balance | 余额 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Postman() | 默认构造函数 |
| Postman(const string& id, const string& ipw, const string& iname, const string& iphone, const int& ibalance) | 从文件输入或管理员添加快递员时使用的构造函数 |
| ~Postman() | 默认析构函数 |
| void login() | 登录 |
| void output() const | 输出快递员信息 |
| private | |
| void loadPostmanPackList() const | 从文件读取当前快递员负责的快递，存储在packList中 |
| void postmanInterface() | 快递员界面 |
| void pickUp() | 揽收快递 |
| void checkBalance() const | 查看余额 |
| void searchPack() const | 查询快递 |
| void searchPackBySender() const | 根据寄件人信息查询快递 |
| void searchPackBySenderID() const | 根据寄件人用户ID查询快递 |
| void searchPackBySenderName() const | 根据寄件人姓名查询快递 |
| void searchPackBySenderPhone() const | 根据寄件人电话查询快递 |
| void searchPackBySenderAddr() const | 根据寄件人地址查询快递 |
| void searchPackByRecver() const | 根据收件人信息查询快递 |
| void searchPackByRecverID() const | 根据收件人用户ID查询快递 |
| void searchPackByRecverName() const | 根据收件人姓名查询快递 |
| void searchPackByRecverPhone() const | 根据收件人电话查询快递 |
| void searchPackByRecverAddr() const | 根据收件人地址查询快递 |
| void searchPackBySendTime() const | 根据发送时间查询快递 |
| void searchPackByPackID() const | 根据快递号查询快递 |
| void searchPackByState() const | 根据快递状态查询快递 |
| int outputUnrecvPack() const | 输出未接收快递 |
| int outputUnpickPack() const | 输出未揽收快递 |
| int outputRecvPack() const | 输出已接收快递 |
| void updateBalanceInFile() const | 揽收快递后，在快递员信息文件中更新余额 |
| streampos searchID() const | 返回当前快递员的信息在快递员信息文件中的位置 |

**普通快递类**

题目一中的快递类在题目二被改为普通快递类，并且是其他快递类的父类。新增了揽收时间、快递员信息等成员变量和一些成员函数。

|  |  |
| --- | --- |
| 成员变量 | 快递信息 |
| public | |
| PackPostmanInfo postman | 快递员信息 |
| SYSTEMTIME pickUpTime | 揽收时间 |
| int type | 快递类型 |
| int amount | 快递重量/数量 |
| int price | 运费 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| void pickUp() | 揽收快递 |
| void assignPostman(const PackPostmanInfo& postmanInfo) | 为快递分配快递员，参数为快递员信息 |
| void deletePostman() | 删除快递的快递员信息 |
| virtual void getPrice() | 计算快递的运费 |

**易碎品类**

公有继承了普通快递类，主要区别在对运费计算的方法。

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Fragile() | 默认构造函数 |
| Fragile(const string& id, const SYSTEMTIME& sTime, const PackUserInfo& send, const PackUserInfo& recv, const int& ty, const int& am, const string& info) | 用于寄件的构造函数 |
| ~Fragile() | 默认析构函数 |
| void getPrice() | 计算快递的运费 |

**书籍类**

公有继承了普通快递类，主要区别在对运费计算的方法。

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Book() | 默认构造函数 |
| Book(const string& id, const SYSTEMTIME& sTime, const PackUserInfo& send, const PackUserInfo& recv, const int& ty, const int& am, const string& info) | 用于寄件的构造函数 |
| ~Book() | 默认析构函数 |
| void getPrice() | 计算快递的运费 |

**总体设计**

快递员信息存储在postmanInfo.txt文件中，文件同样放在data文件夹中。文件postmanInfo.txt存储所有快递员信息，每一行为一个快递员的信息。快递员信息按快递员ID、密码、姓名、电话和余额这个顺序存储在同一行，每一项用’.’分隔开。

文件packInfo.txt存储快递信息的方式有变动。快递信息按快递号、接收状态、发送时间、接收时间（如果快递未接收则没有接收时间）、寄件人信息、收件人信息、快递员信息、揽收时间（如果快递未揽收则没有揽收时间）、快递类型、快递重量、运费、快递详情这个顺序存储在同一行，每一项用’.’分隔开。

程序执行时，首先打开所需文件，然后让用户表明身份。如果是新用户则调用User::regis()；如果是老用户则调用User::login()；如果是管理员则调用Admin::login()；如果是管理员则调用Postman::login()。

**快递员使用时**

快递员不能注册，只能登录。登录时首先要求输入ID，然后调用searchID()检查输入的ID是否在文件中存在，存在则提示用户输入密码，再对输入的密码进行判断。如果密码正确，登录成功，加载快递员其他信息，然后调用postmanInterface()函数进入快递员界面。

进入快递员界面后，首先调用loadPostmanPackList()函数，将所有该快递员负责的快递从文件读入packList中。然后，快递员可以选择揽收快递、查询快递和查看余额，并分别调用pickUp()、searchPack()和checkBalance()。

揽收函数时，先调用outputUupickPack()，输出所有未揽收的快递，再让快递员输入要揽收的快递的快递号。程序接着对输入的合法性和输入的快递号是否存在进行判断，如果存在则将运费的一半（向上取整）转给快递员余额，调用updateBalanceInFile()及时在文件中更改，然后调用Pack::pickUp()进行揽收。该函数将快递状态改为“未接收”，并获取当前时间作为揽收时间，然后将更新后的快递信息内容写入文件。

查询余额函数直接输出余额，而查询快递函数会让快递选择查询方式，并根据快递员的选择调用其他函数完成查询功能。查询快递只需要在packList中进行查找，不需对文件进行任何操作。

**用户使用时**

代码在用户发送快递方面更改了题目一的代码，其他基本一致。用户在确认收件人后，需要选择快递类型，然后再输入快递重量或书籍本数，最后输入快递详情。快递在调用Pack::sendPack()之前，先调用Pack::getPrice()计算快递运费，然后判断用户余额是否足够，足够的话才调用Pack::sendPack()发送快递。

**管理员使用时**

在管理员登录完毕后，在管理员界面，将会调用loadAllPostmanList()，从文件中把所有快递员信息读入allPostmanList中。然后管理员可选择管理快递员信息，并调用adminPostmanInterface()函数进入管理快递员信息界面。

管理快递员信息界面中，管理员可选择添加快递员、删除快递员、查看所有快递员和查询快递员，并分别调用addPostman()、deletePostman()、checkAllPostman()和searchPostman()函数。

新增快递员时，管理员先输入快递员ID，判断其合法性，调用searchPostman()函数判断ID是否已经存在。然后输入快递员的密码，判断其合法性。最后，输入快递员姓名和电话，将快递员信息加入allPostmanList，并调用addPostmanToFile()函数写入文件。

删除快递员时，判断其合法性，调用searchPostman()函数判断ID是否已经存在。如果存在则检查该快递员是否有未完成的快递，如有，管理员可选择放弃删除或继续删除。继续删除的话，这些未完成的快递任务会一一调用Pack::deletePostman()函数，删除快递的快递员信息，将状态改为未揽收。最后，调用deletePostmanFromFile()从文件中删除快递员。

在管理快递信息界面，管理员还可选择为快递分配快递员，调用assignPack()函数。该函数调用checkUnassignPack()函数输出所有未被分配快递员的快递，然后管理员输入要分配的快递，判断其合法性后，再输入快递员ID，判断其合法性后，调用Pack::assignPostman()函数，将快递员信息写入快递信息，完成分配。

其余新增的查询快递员和快递函数只需要在所需的全局变量中进行查找，不需要对文件进行操作。

**实现中的重要问题和解决方案**

由于题目要求快递只能有三个状态，不存在一个状态表示快递未被分配快递员，因此将未被分配快递员的快递信息中的每一项快递员信息用”???”表示。

不同的快递类型有不同的运费计算方式，因此，首先将普通快递类设为父类，其余类型快递公有继承普通快递。父类中的getPrice()函数为虚函数，其他类型快递根据自己的运费计算方式重定义该函数，pack\*类型的对象也可使用其他类型的getPrice()函数。

管理员和负责快递的快递员对运费进行五五分成。如果运费是奇数，则管理员收到运费/2，向下取整；快递员收到运费/2，向上取整。不需使用浮点数，简化程序。

**题目三：物流管理系统（网络版）**

**题目要求**

在题目一、二的基础上，将物流管理平台修改为网络版。题目要求采用传统CS结构，客户端和服务器端为不同进程，两端之间使用socket通信。

**服务器端**

**高层数据信息设计**

**全局常量定义**

题目二的基础上新增

#define SEND\_SPLIT '|' //发送数据分隔符

**全局变量定义**

fstream userInfoFile; //读写用户信息文件流

fstream adminInfoFile; //读写管理员信息文件流

fstream packInfoFile; //读写快递信息文件流

fstream postmanInfoFile; //读写快递员信息文件流

//定义发送缓冲区和接受缓冲区

char send\_buf[1000];

char recv\_buf[1000];

//定义服务端socket，接受请求socket

SOCKET server;

SOCKET client;

删去了题目二的全局向量和对象。

**关键类的设计**

**用户类**

在服务器端，当客户端使用者身份为用户。使用用户类的成员函数完成工作。

|  |  |
| --- | --- |
| 成员变量 | 作用 |
| public | |
| streampos locInFile | 当前登录客户端的用户的信息在文件的位置 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| User() | 默认构造函数 |
| ~User() | 默认析构函数 |
| void recvFromClient(const string& req) | 从客户端收到用户的请求后调用此函数进行判断 |
| private | |
| void checkExistId(const string& id) const | 检查用户ID id是否存在，存在发送1，否则发送0 |
| void sendPw(const string& id) const | 发送用户ID id的密码 |
| void getInfo(const string& id) | 发送用户ID id的用户信息 |
| void changePW(const string& pw) const | 更改用户ID id的密码 |
| void changeBalance(const string& balance) const | 更改用户ID id的余额 |
| void sendPack(const string& packInfo) const | 发送快递 |
| void recvPack(const string& packInfo) const | 接收快递 |
| void loadPacketUserInfo(const string& id) const | 加载用户ID为id的收件人信息 |
| void newUser(const string& userInfo) | 向文件添加新用户 |
| void loadUserPackList(const string& id) const | 从文件中读取与用户ID为id的用户相关的快递并发送 |
| streampos searchID(const string& id) const | 返回用户ID为id的用户信息在文件中的位置 |
| void writeToFile(int mode, const string& info) const | 将用户信息写入文件，mode=0写入新用户信息，mode=1更改某用户的密码，mode=1更改某用户的余额 |

**快递员类**

在服务器端，当客户端使用者身份为快递员。使用快递员类的成员函数完成工作。

|  |  |
| --- | --- |
| 成员变量 | 作用 |
| public | |
| streampos locInFile | 当前登录客户端的快递员的信息在文件的位置 |

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Postman() | 默认构造函数 |
| ~Postman() | 默认析构函数 |
| void recvFromClient(const string& req) | 从客户端收到快递员的请求后调用此函数进行判断 |
| private | |
| void checkExistId(const string& id) const | 检查快递员ID id是否存在，存在发送1，否则发送0 |
| void sendPw(const string& id) const | 发送快递员ID id的密码 |
| void getInfo(const string& id) | 发送快递员ID id的信息 |
| void pickUp(const string& info) const | 揽收快递 |
| void updateBalanceInFile(const string& balance) const | 更改快递员ID id的余额 |
| void loadPostmanPackList(const string& id) const | 从文件读取当前快递员负责的快递并发送 |
| streampos searchID(const string& id) const | 返回当前登录的快递员的信息在文件中的位置 |

**管理员类**

在服务器端，当客户端使用者身份为管理员。使用管理员类的成员函数完成工作。

|  |  |
| --- | --- |
| 成员函数 | 函数功能 |
| public | |
| Admin() | 默认构造函数 |
| ~Admin() | 默认析构函数 |
| void recvFromClient(const string& req) const | 从客户端收到管理员的请求后调用此函数进行判断 |
| private | |
| void checkExistId(const string& id) const | 检查管理员ID id是否存在，存在发送1，否则发送0 |
| void sendPw(const string& id) const | 发送管理员ID id的密码 |
| void getInfo(const string& id) | 发送管理员ID id的信息 |
| void addPostman(const string& postmanInfo) const | 往文件新增快递员信息 |
| void deletePostmanFromPack(const string& packId) const | 删除文件中快递的快递员信息 |
| void deletePostman(const string& id) const | 从文件中删除快递员 |
| void assignPack(const string& info) const | 为快递分配快递员 |
| void loadAllUserList() const | 管理员有权查看所有用户，提前从文件中读取并将所有用户信息并发送 |
| void loadAllPostmanList() const | 管理员有权查看所有快递员，提前从文件中读取并将所有快递员信息并发送 |
| void loadAllPackList() const | 管理员有权查看所有快递，提前从文件中读取并将所有快递信息并发送 |
| streampos searchID(const string& id) const | 返回ID为id的管理员信息在文件中的位置 |

**客户端**

**高层数据信息设计**

**全局常量定义**

题目二的基础上新增

#define SEND\_SPLIT '|' //发送数据分隔符

**全局变量定义**

//定义发送缓冲区和接受缓冲区

char send\_buf[1000];

char recv\_buf[1000];

//定义服务端socket，接受请求socket

SOCKET server;

删去了题目二的全局文件流变量。

**关键类的设计**

在客户端，类的定义与题目二的大致相同。删去了使用文件流读写的代码和成员函数，改为向服务器端发出请求和数据。例如，用户类中，删去了void writeToFile(int mode) const函数，改为在其他位置，将数据发送给服务器端。服务器端接收后，写入文件。

**总体设计**

客户端发给服务器端的请求有固定的格式。请求包括用户身份、请求类型和数据三部分，以’.’分隔开来。用户身份为单个字符，0表示管理员，1表示用户，2表示快递员。请求类型也是单个字符，为数字，表示需要进行的操作。数据为请求的操作所需的数据，可以有多项数据，用’.’分隔。

以下为不同类的每个操作的操作码和所需数据。

**用户类**

|  |  |  |
| --- | --- | --- |
| 所需操作 | 操作码 | 所需数据 |
| 检查用户ID是否存在 | 0 | 用户ID |
| 发送用户ID的密码 | 1 | 用户ID |
| 发送用户ID的用户信息 | 2 | 用户ID |
| 更改用户的密码 | 3 | 新密码 |
| 更改用户的余额 | 4 | 新余额 |
| 发送快递 | 5 | 运费、快递信息（文件存储格式） |
| 接收快递 | 6 | 快递号、接收时间 |
| 加载某个用户的收件人信息 | 7 | 用户ID |
| 向文件写入新用户 | 8 | 用户信息（文件存储格式） |
| 从文件中读取与用户相关的快递并发送 | 9 | 用户ID |

**快递员类**

|  |  |  |
| --- | --- | --- |
| 所需操作 | 操作码 | 所需数据 |
| 检查快递员ID是否存在 | 0 | 快递员ID |
| 发送快递员的密码 | 1 | 快递员ID |
| 发送快递员的信息 | 2 | 快递员ID |
| 揽收快递 | 3 | 快递号、揽收时间 |
| 更新余额 | 4 | 新余额 |
| 从文件读取快递员负责的快递并发送 | 5 | 快递员ID |

**管理员类**

|  |  |  |
| --- | --- | --- |
| 所需操作 | 操作码 | 所需数据 |
| 检查管理员ID是否存在 | 0 | 管理员ID |
| 发送管理员的密码 | 1 | 管理员ID |
| 发送管理员的信息 | 2 | 管理员ID |
| 添加新快递员 | 3 | 快递员信息（文件存储格式） |
| 删除快递的快递员信息 | 4 | 快递号 |
| 删除快递员 | 5 | 快递员ID |
| 为快递分配快递员 | 6 | 快递号，快递员信息 |
| 从文件读取所有用户信息并发送 | 7 | 无 |
| 从文件读取所有快递员信息并发送 | 8 | 无 |
| 从文件读取所有快递信息并发送 | 9 | 无 |

当客户端需要从文件读取数据或者需要将数据写入文件，先产生一个相应操作的请求，然后发送给服务器端进行下一步处理。服务器接收到请求后，先读取第一部分，也就是客户端用户身份，来决定应该调用哪个类的recvFromClient()函数。在相应的recvFromClient()函数中，再根据操作码决定应该调用哪个成员函数，并利用数据，完成需要的操作。

**当用户使用时**

当客户端需要判断某个用户ID是否存在，将发送请求给服务器端。服务器端利用User:: checkExistId()函数，检查文件中是否存在该ID。有则发送1，没有则发送0。

当客户端进行登录时，需要指定用户的密码，将发送请求给服务器端。服务器端利用User:: sendPw()函数，从文件中找到ID的密码，然后发送给客户端。完成登录后，需要指定用户的个人信息，客户端将发送请求给服务器端。服务器端利用User:: getInfo()函数，从文件中找到该用户的个人信息，然后发送给客户端。

用户在客户端注册完毕后，将新用户信息发送给服务器端，服务器端调用User::newUser()函数，将新用户信息写入文件。调用newUser()或getInfo()函数，则代表用户在客户端注册或登录完毕，并可能将使用其他功能。可在服务器端对User::locInFile进行赋值，存储当前登录客户端的用户的信息在文件的位置，提高效率。

用户在客户端需要将相关快递信息读入packList，将请求发送给服务器端。服务器端调用User:: loadUserPackList()函数，将所有相关快递信息发送给客户端。

用户在客户端进行充值或修改密码后，将请求连同新密码或新余额发送给服务器端。服务器端将新密码或新余额写入用户文件。

用户发送快递时，客户端发送的请求的数据包括运费和快递信息。运费作为额外单独的一项是为了方便更改管理员文件中的余额，快递信息可直接写入快递文件。用户接收快递时，客户端发送的请求除了快递号用来指定快递外，还包括接收时间。这是为了避免客户端接收和服务器端接收快递的时间不一样。

**快递员使用时**

快递员揽收快递时，客户端发送的请求包括快递号和揽收时间。快递号用来找到指定的快递，揽收时间是为了避免两端揽收的时间不一样。其余功能与用户使用时大致相同，不再赘述。

**管理员使用时**

管理员新增快递员时，客户端发送的请求内含快递员信息，服务器端可直接将请求的数据部分写入文件。删除快递员时，请求只需包括快递员ID，服务器端便可找到要删除的快递员信息进行删除。

删除快递员时可能需要删除一些快递的快递员信息，客户端只需要在请求中数据包括快递号，服务器端便会找到相应快递进行操作。而为快递分配快递员时，需要快递号和快递员信息，从文件找出目标快递并写入快递员信息。其余功能与用户使用时大致相同，不再赘述。

**实现中的重要问题和解决方案**

使用socket通信时，因为以为一次send()对应另一端的一次recv()，一开始对两端的send()和recv()的安排较为随意，也没有规定缓冲区大小。结果导致socket通信出现问题，程序无法执行。后面规定了缓冲区大小为1000个字节，并且两端尽量保持recv()和send()交替使用，避免可能出现的错误，如一次recv()接收了另一端的多次send()，其中包括不需要的数据。

一开始编写两端代码时，没有妥善计划，两端耦合度过高，收发数据混乱，处理接收数据过于复杂，程序无法使用。因此，重新定义所需的类，简化服务器端的工作，并制定一个统一的请求格式。服务器端只关心接收到的请求，而不在乎客户端正在进行什么操作。

如loadUserPackList()这种发送多项快递信息时，有可能超出缓冲区的大小（1000字节）。因此，服务器端不断发送数据，最后发送一个空串表示发送结束。客户端不断接收数据，并不断向后添加到一个string类型的变量，直到收到空串。客户端再从该变量读取一项项的信息，存入相应向量中。每一项信息之间用’|’分隔。

在socket通信中，需要recv\_buf和send\_buf接收和发送缓冲区。这两个变量不能为string类型，只能是char类型。定义为string类型可能使用更加方便，但在实践中出现了错误，无法读取。使用char类型只要借助库函数，也不会复杂。

**实验总结**

通过本次实验，对C++的面向对象编程有了更深刻的理解。定义每一个类时，都需要考虑它的属性和行为，才能定义成员变量以及函数，达到题目要求。同时，程序数据需要长久保存，需要频繁对文件进行操作，也对文件流读写的使用更加熟练。在完成网络版的题目三中，也自学了socket的原理以及如何使用socket进行通信。

本次实验代码量较多，完成本次实验在学习C++各方面的特性和编写程序都花费了不少时间，代码也主要使用简单算法，效率不高。但是，因为是单人完成，还需使用如socket通信等课外知识，实验充分锻炼了我的编程能力，提高了知识储备，使我受益匪浅。