## 第一部分 必做题

### 第1题：菜鸟智慧系统

#### 数据结构：

采用了结构体存储包裹的相关数据，如下：

typedef struct pakege{

string name;

string packCode;

string phone;

string size;

int date;

}pak;

双向链表节点的类如下：

class Node {

public:

pak data;

Node \*prev;

Node \*next;

Node(pak data) : data(data), prev(nullptr), next(nullptr) {}

};

#### 算法设计思想：

将双向链表相关的数据成员（头尾指针，节点个数，节点最大个数）以及相关函数（添加删除节点和遍历链表等）写在一个类DoublyLinkedList的里面。方便系统的相关编写。大中小三个货架分别对应三个双向链表。用一个全局变量Date表示日期。

在主函数中，先使用loadPack函数读取data.txt文件里的数据到链表中。然后使用menu函数输出菜单并接受命令进行操作，当一天结束后会使用updatePack函数清除过期包裹。退出系统则会调用savePack函数将链表的数据写回至data.txt文件中。

在menu函数中输出菜单，程序使用者可选择进入管理员系统、用户系统、结束今天或退出主系统。管理员系统会调用admin函数，用户系统会调用users函数。

在admin函数中，管理员可以手动或自动存包裹，或者获取相关统计信息。手动存包裹调用packNotAuto函数，该函数在管理员输入相关信息后，生成取件码，并使用DoublyLinkedList类的添加节点（添加在链表尾部）insertAtTail函数添加包裹到相对应链表中。自动存包裹调用packAuto函数读取input.txt文件里的数据添加包裹，过程类似手动存包裹，统计则调用analyse函数展示相关信息。

在users函数中，用户可以按照电话号码取包裹或者用取件码取包裹。前者调用pickByPhone函数，后者调用pickByCode函数。在pickByPhone函数中，用户输入电话号码后，程序现在查找链表中是否存在对应电话号码，然后调用双向链表的删除节点成员函数。在pickByCode函数中，程序会根据输入的取件码查找对应的包裹的电话号码，然后再根据电话号码删除对应的所有包裹。如果输入的电话号码或者取件码找不到对应包裹会给出相关提示。

源程序：

#include <bits/stdc++.h>

using namespace std;

#define MAX\_PERSON 30

int Date = 0;

typedef struct pakege{

string name;

string packCode;

string phone;

string size;

int date;

}pak;

class Node {

public:

pak data;

Node \*prev;

Node \*next;

Node(pak data) : data(data), prev(nullptr), next(nullptr) {}

};

class DoublyLinkedList {

public:

Node \*head;

Node \*tail;

int MaxOfList;

int NumOfList = 0;

DoublyLinkedList(int M) : MaxOfList(M), head(nullptr), tail(nullptr) {}

void traverse() {

Node \*temp = head;

while (temp) {

cout << "姓名："<<temp->data.name << " 电话："<<temp->data.phone<<" 日期（第几天）："<<temp->data.date<<" 取件码"<<temp->data.packCode<<" 货架编号："<<temp->data.size<<endl;

temp = temp->next;

}

}

void insertAtHead(pak data) {

if (NumOfList+1 <= MaxOfList){

Node \*newNode = new Node(data);

NumOfList++;

if (!head) {

head = newNode;

tail = newNode;

return;

}

head->prev = newNode;

newNode->next = head;

head = newNode;

}

else{

cout<<"货架已满"<<endl;

}

}

void insertAtTail(pak data) {

if (NumOfList+1 <= MaxOfList){

Node \*newNode = new Node(data);

NumOfList++;

if (!tail) {

head = newNode;

tail = newNode;

return;

}

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

else {

cout<<"货架已满"<<endl;

}

}

void deleteAtHead() {

if (!head) {

return;

}

if (head == tail) {

NumOfList=0;

delete head;

head = nullptr;

tail = nullptr;

return;

}

Node \*temp = head;

head = head->next;

head->prev = nullptr;

NumOfList--;

delete temp;

}

void deleteAtTail() {

if (!tail) {

return;

}

if (head == tail) {

NumOfList=0;

delete tail;

head = nullptr;

tail = nullptr;

return;

}

Node \*temp = tail;

tail = tail->prev;

tail->next = nullptr;

NumOfList--;

delete temp;

}

/\*

bool search(pak data) {

Node \*temp = head;

while (temp) {

if (temp->data == data) {

return true;

}

temp = temp->next;

}

return false;

}

\*/

bool searchPhone(string phoneNeedToSearch) {

Node \*temp = head;

while (temp) {

if (temp->data.phone == phoneNeedToSearch) {

return true;

}

temp = temp->next;

}

return false;

}

bool searchCode(string codeNeedToSearch) {

Node \*temp = head;

while (temp) {

if (temp->data.packCode == codeNeedToSearch) {

return true;

}

temp = temp->next;

}

return false;

}

string findPhoneByCode(string codeNeedToSearch){

Node \*temp = head;

string result;

while (temp) {

if (temp->data.packCode == codeNeedToSearch) {

result = temp->data.phone;

return result;

}

temp = temp->next;

}

}

void deleteNode(pak data) {

Node \*temp = head;

while (temp) {

if (temp->data.phone == data.phone) {

if (temp == head) {

deleteAtHead();

return;

} else if (temp == tail) {

deleteAtTail();

return;

} else {

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

delete temp;

NumOfList--;

return;

}

}

temp = temp->next;

}

}

void deleteNodeByPhone(string phoneDel) {

Node \*temp = head;

while (temp) {

if (temp->data.phone == phoneDel) {

cout << "取到取件码为"<<temp->data.packCode<<"的包裹\n";

if (temp == head) {

deleteAtHead();

return;

} else if (temp == tail) {

deleteAtTail();

return;

} else {

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

delete temp;

NumOfList--;

return;

}

}

temp = temp->next;

}

}

void deleteOutdate(int date) {

Node \*temp = head;

while (temp) {

if (temp->data.date + 2 <= date) {

if (temp == head) {

deleteAtHead();

return;

} else if (temp == tail) {

deleteAtTail();

return;

} else {

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

delete temp;

NumOfList--;

return;

}

}

temp = temp->next;

}

}

};

DoublyLinkedList Max(50);

DoublyLinkedList Mid(100);

DoublyLinkedList Min(500);

void packNotAuto(){

pak packTemp;

string codePart;

cout<<"请输入要存放的货架编号: 1(大) 2(中) 3(小)\n";

cin >> packTemp.size;

if (packTemp.size=="1"){

codePart = std::to\_string(Max.NumOfList+1);

}

else if (packTemp.size=="2"){

codePart = std::to\_string(Mid.NumOfList+1);

}

else if (packTemp.size=="3"){

codePart = std::to\_string(Min.NumOfList+1);

}

cout<<"请输入包裹主人电话号码\n";

cin >> packTemp.phone;

cout<<"请输入包裹主人姓名\n";

cin >> packTemp.name;

packTemp.date = Date;

packTemp.packCode = packTemp.size + codePart;

if (packTemp.size=="1"){

Max.insertAtTail(packTemp);

}

else if (packTemp.size=="2"){

Mid.insertAtTail(packTemp);

}

else if (packTemp.size=="3"){

Min.insertAtTail(packTemp);

}

cout<<"存放包裹完成\n";

}

void packAuto(){

ifstream inputPack;

inputPack.open("input.txt");

while (!inputPack.eof()){

pak packTemp;

string codePart;

inputPack >> packTemp.size;

if (packTemp.size=="1"){

codePart = std::to\_string(Max.NumOfList+1);

}

else if (packTemp.size=="2"){

codePart = std::to\_string(Mid.NumOfList+1);

}

else if (packTemp.size=="3"){

codePart = std::to\_string(Min.NumOfList+1);

}

//cout<<"请输入包裹主人电话号码\n";

inputPack >> packTemp.phone;

//cout<<"请输入包裹主人姓名\n";

inputPack >> packTemp.name;

packTemp.date = Date;

packTemp.packCode = packTemp.size + codePart;

if (packTemp.size=="1"){

Max.insertAtTail(packTemp);

}

else if (packTemp.size=="2"){

Mid.insertAtTail(packTemp);

}

else if (packTemp.size=="3"){

Min.insertAtTail(packTemp);

}

}

cout<<"存放包裹完成\n";

inputPack.close();

}

void analyse(){

cout<<"以下为包裹总信息的相关数据分析\n";

cout<<"现在大货架包裹总数为："<<Max.NumOfList<<endl;

cout<<"现在中货架包裹总数为："<<Mid.NumOfList<<endl;

cout<<"现在小货架包裹总数为："<<Min.NumOfList<<endl;

cout<<"按y显示详细信息\n";

string command;

cin >> command;

if (command == "y"||command == "Y") {

cout<<"大货架包裹具体信息如下：\n";

Max.traverse();

cout<<"中货架包裹具体信息如下：\n";

Mid.traverse();

cout<<"小货架包裹具体信息如下：\n";

Min.traverse();

}

}

void admin(){

while(1){

char command;

cout<<"输入相关数字进行相关操作\n";

cout<<"\*\*\*\*\*\*\*1.手动存包裹\*\*\*\*\*\n";

cout<<"\*\*\*\*\*\*\*2.自动存包裹\*\*\*\*\*\n";

cout<<"\*\*\*\*\*\*\*3.获取分析信息\*\*\*\n";

cout<<"\*\*\*\*\*\*\*4.退出管理员系统\*\n";

cin>> command;

if (command == '1') {

packNotAuto();

}

else if (command =='2') {

packAuto();

}

else if (command =='3') {

analyse();

}

else if (command =='4') break;

else cout<<"输入指令无效，请重新输入\n";

}

}

void pickByPhone(){

string phoneOfUser;

cout<<"请输入电话号码取件\n";

while (1){

cin >> phoneOfUser;

if (Max.searchPhone(phoneOfUser)||Mid.searchPhone(phoneOfUser)||Min.searchPhone(phoneOfUser)){

while (Max.searchPhone(phoneOfUser)){

Max.deleteNodeByPhone(phoneOfUser);

}

while (Mid.searchPhone(phoneOfUser)){

Mid.deleteNodeByPhone(phoneOfUser);

}

while (Min.searchPhone(phoneOfUser)){

Min.deleteNodeByPhone(phoneOfUser);

}

cout<<"相关的所有包裹已经取出\n";

break;

}

else{

cout<<"电话号码不存在或不存在该号码所对应的包裹\n";

break;

}

}

}

void pickByCode(){

string codeOfUser;

string phoneOfUser;

cout<<"请取件码取件\n";

while (1){

cin >> codeOfUser;

if (Max.searchCode(codeOfUser)||Mid.searchCode(codeOfUser)||Min.searchCode(codeOfUser)){

if (Max.searchCode(codeOfUser)){

phoneOfUser = Max.findPhoneByCode(codeOfUser);

while (Max.searchPhone(phoneOfUser)){

Max.deleteNodeByPhone(phoneOfUser);

}

}

if (Mid.searchCode(codeOfUser)){

phoneOfUser = Mid.findPhoneByCode(codeOfUser);

while (Mid.searchPhone(phoneOfUser)){

Mid.deleteNodeByPhone(phoneOfUser);

}

}

if (Min.searchCode(codeOfUser)){

phoneOfUser = Min.findPhoneByCode(codeOfUser);

while (Min.searchPhone(phoneOfUser)){

Min.deleteNodeByPhone(phoneOfUser);

}

}

cout<<"电话号为"<<phoneOfUser<<"的相关的所有包裹已经取出\n";

break;

}

else{

cout<<"取件码不存在或不存在该取件码所对应的包裹\n";

break;

}

}

}

void users(){

while(1){

char command;

cout<<"输入相关数字进行相关操作\n";

cout<<"\*\*\*\*\*\*\*1.电话号取包裹\*\*\*\n";

cout<<"\*\*\*\*\*\*\*2.取件码取包裹\*\*\*\n";

cout<<"\*\*\*\*\*\*\*3.退出用户系统\*\*\*\n";

cin>> command;

if (command == '1') {

pickByPhone();

}

else if (command =='2') {

pickByCode();

}

else if (command =='3') break;

else cout<<"输入指令无效，请重新输入\n";

}

}

bool menu(){

while(1){

char command;

cout<<"输入相关数字表示您的身份\n";

cout<<"\*\*\*\*\*\*\*1.管理员\*\*\*\*\*\*\*\*\*\n";

cout<<"\*\*\*\*\*\*\*2.用户 \*\*\*\*\*\*\*\*\*\n";

cout<<"\*\*\*\*\*\*\*3.结束今天\*\*\*\*\*\*\*\n";

cout<<"\*\*\*\*\*\*\*4.退出系统\*\*\*\*\*\*\*\n";

cin>> command;

if (command == '1') {

admin();

}

else if (command =='2') {

users();

}

else if (command =='3') break;

else if (command =='4') return false;

else cout<<"输入指令无效，请重新输入\n";

}

return true;

}

void loadPack(){

ifstream inputPack;

inputPack.open("data.txt");

while (!inputPack.eof()){

pak packTemp;

string codePart;

inputPack >> packTemp.size;

//cout<<"请输入包裹主人电话号码\n";

inputPack >> packTemp.phone;

//cout<<"请输入包裹主人姓名\n";

inputPack >> packTemp.name;

packTemp.date = Date;

inputPack >> packTemp.packCode;

if (packTemp.size=="1"){

Max.insertAtTail(packTemp);

}

else if (packTemp.size=="2"){

Mid.insertAtTail(packTemp);

}

else if (packTemp.size=="3"){

Min.insertAtTail(packTemp);

}

}

cout<<"加载完成\n";

inputPack.close();

}

void updatePack(){

Max.deleteOutdate(Date);

Min.deleteOutdate(Date);

Mid.deleteOutdate(Date);

}

void savePack(){

ofstream savePack;

savePack.open("data.txt");

while (Max.head){

savePack << Max.head->data.size <<" ";

//cout << Max.head->data.size <<" ";

//cout<<"请输入包裹主人电话号码\n";

savePack << Max.head->data.phone<<" ";

//cout<<"请输入包裹主人姓名\n";

savePack << Max.head->data.name<<" ";

//packTemp.date = Date;

savePack << Max.head->data.packCode<<endl;

Max.deleteAtHead();

}

while (Mid.head){

savePack << Mid.head->data.size <<" ";

//cout<<"请输入包裹主人电话号码\n";

savePack << Mid.head->data.phone<<" ";

//cout<<"请输入包裹主人姓名\n";

savePack << Mid.head->data.name<<" ";

//packTemp.date = Date;

savePack << Mid.head->data.packCode<<endl;

Mid.deleteAtHead();

}

while (Min.head){

savePack << Min.head->data.size <<" ";

//cout<<"请输入包裹主人电话号码\n";

savePack << Min.head->data.phone<<" ";

//cout<<"请输入包裹主人姓名\n";

savePack << Min.head->data.name<<" ";

//packTemp.date = Date;

savePack << Min.head->data.packCode<<endl;

Min.deleteAtHead();

}

cout<<"保存完成\n";

savePack.close();

}

int main() {

bool needToExit;

loadPack();

while(1){

cout<<"现在是第"<<Date<<"天"<<endl;

cout<<"已清除过期包裹\n";

needToExit = menu();

if (!needToExit) break;

Date++;

updatePack();

cout<<"今天结束\n";

}

savePack();

return 0;

}

#### 测试数据和结果：

##### 测试数据

###### Data.txt

每行从左到右依次是货架编码、电话号码，姓名和取件码

1 18907765205 lxy 11

2 15278666682 ww 21

3 18077691108 wzc 31

3 15949390045 yzw 32

###### Input.txt

每行从左至右依次是货架编码电话号码和姓名

1 18077691108 wzc

2 17058582345 wzx

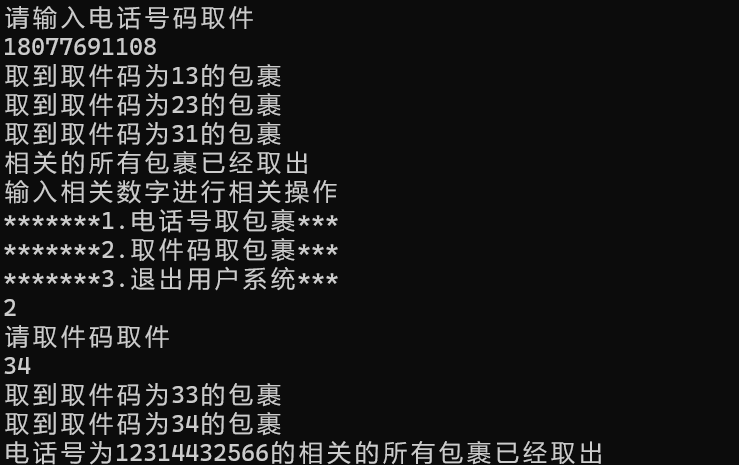
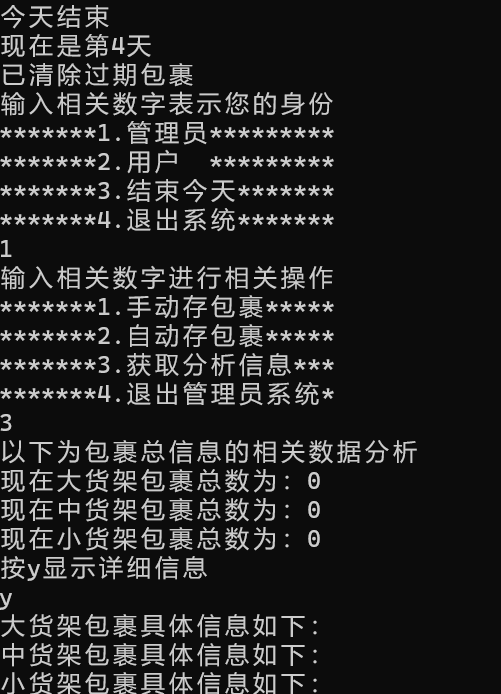
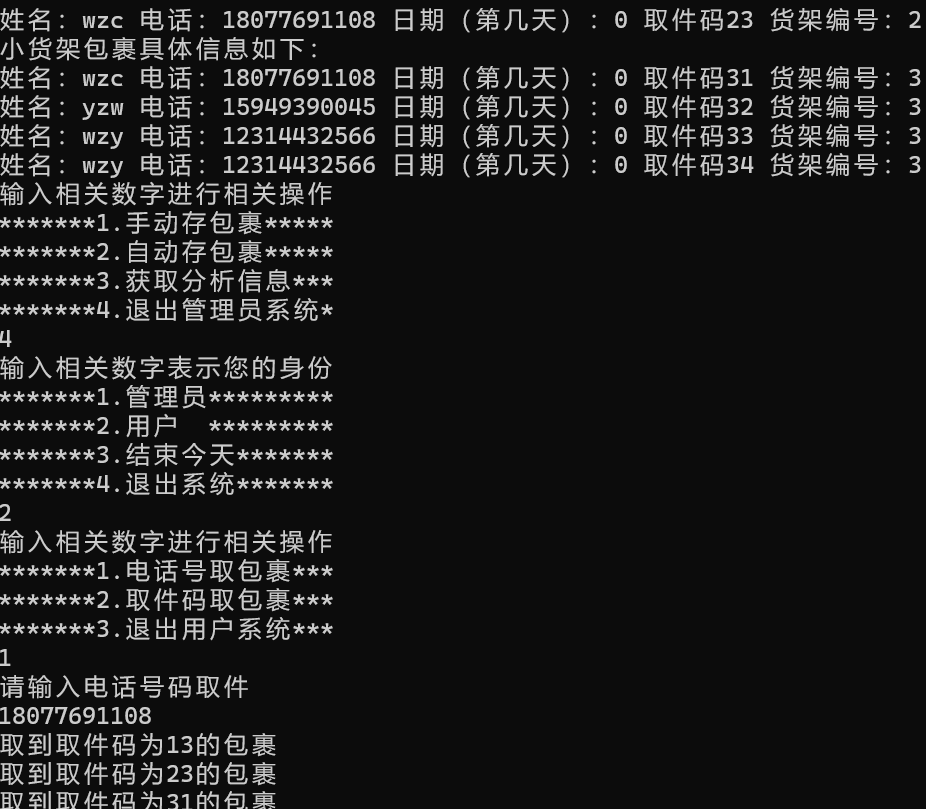
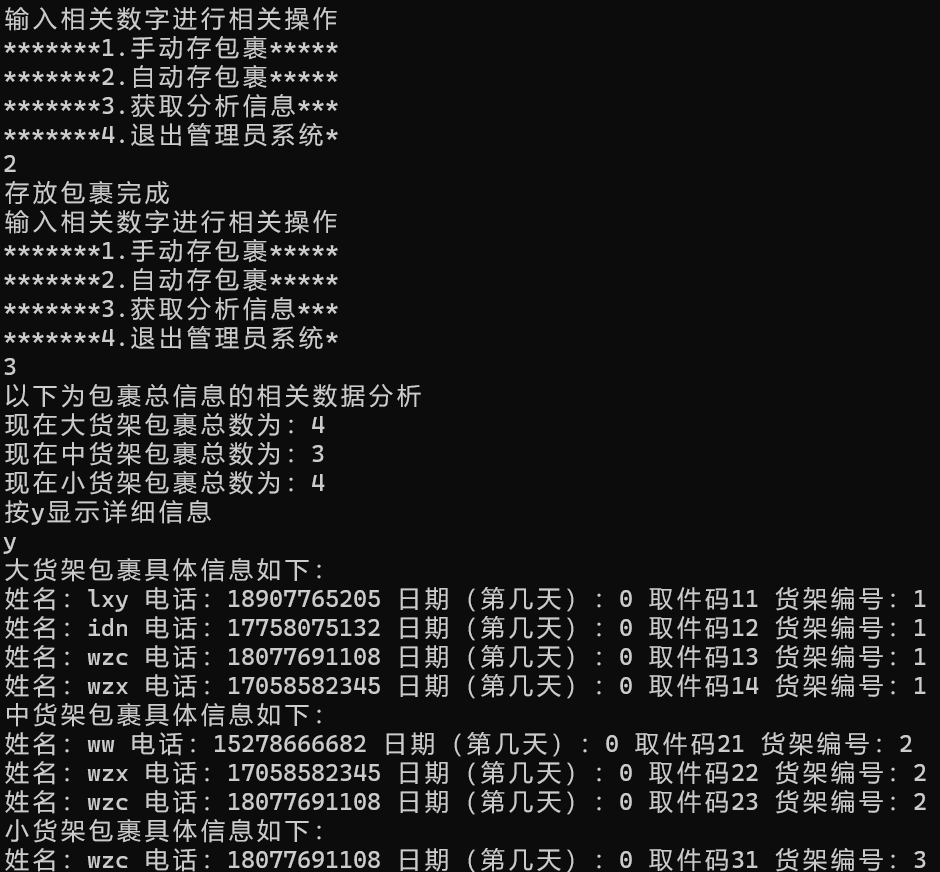
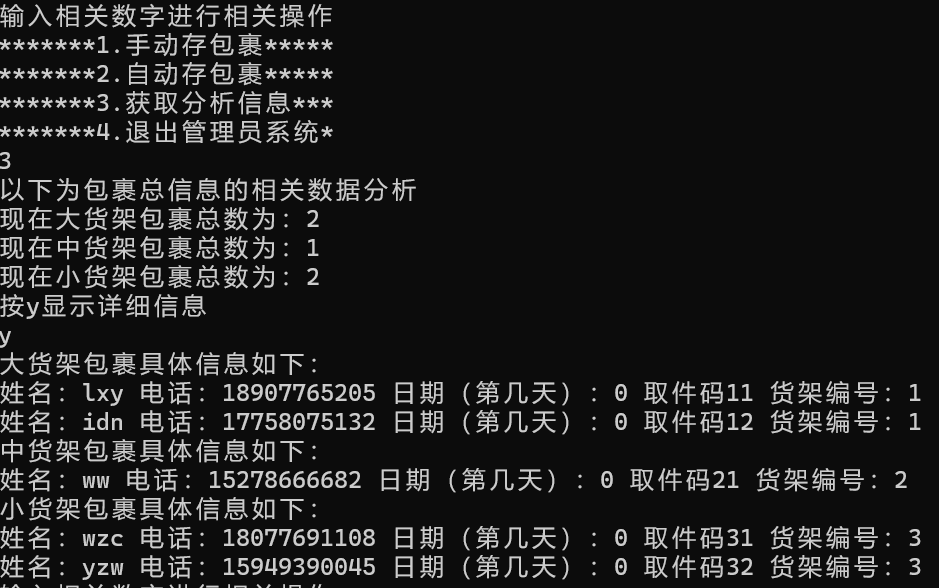
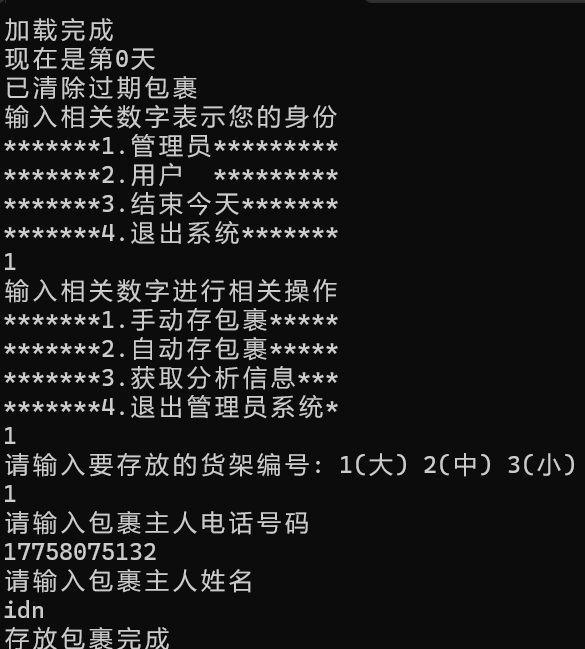
3 12314432566 wzy

1 17058582345 wzx

2 18077691108 wzc

3 12314432566 wzy

##### 测试结果



#### 时间复杂度：

存包裹O(1)

电话号码取包裹O（N）（N最大为所有包裹数）

取件码取包裹O（2N）

清除过期包裹O (N)

#### 反思：

部分操作可以抽象为一个函数被反复调用，减少代码量。深刻理解了线性表中的双向链表的相关操作。

#### 改进：

#### 该题代码行：

505行