Contents

I. 引言

1.1 问题背景及意义

经过多年的发展，中国铁路客票发售和预订系统已成为覆盖全国的超大型分布式售票网络；12306 互联网售票系统是其中一个最主要的售票渠道，自上线以来，以便捷性迅速得到广泛认可，越来越多的旅客通过互联网购票，互联网售票量呈大幅增长态势，售票高峰期的大量并发请求给互联网售票系统带来的压力也屡创新高。[1]

铁路客票系统是指中国铁路客运的制票系统，包括铁路客票发售和预订。铁路客票系统包括查询旅客列车时刻表、票价、列车正晚点、改签，退签，余票查询，选座，点餐，支付等众多功能。铁路客票系统流量巨大，最高日访问量 **1500** 亿次，高峰日平均 1 秒承受 **170** 多万次点击。

因此，铁路客票系统功能众多，流量巨大，设计购票系统最基础功能和算法，并加以改进，在基础领域实现改进和突破，从而实现整体的提升，具有重要意义。

1.2 售票信息管理系统

本文通过 c++ 及数据结构，实现了售票信息的界面设计与实现，售票信息管理系统包含登录注册功能，并分为**普通用户**和**后台管理员**界面，普通用户具有购票，查询，充值，查询历史订单等功能；后台管理员可以添加，修改，删除，查询售票信息。售票信息管理系统关闭时将用户信息与售票信息保存至二**进制**文件中，打开时读取用户信息和售票信息。

II. 任务实现过程

2.1 信息封装与隐藏

2.1.1 文档结构

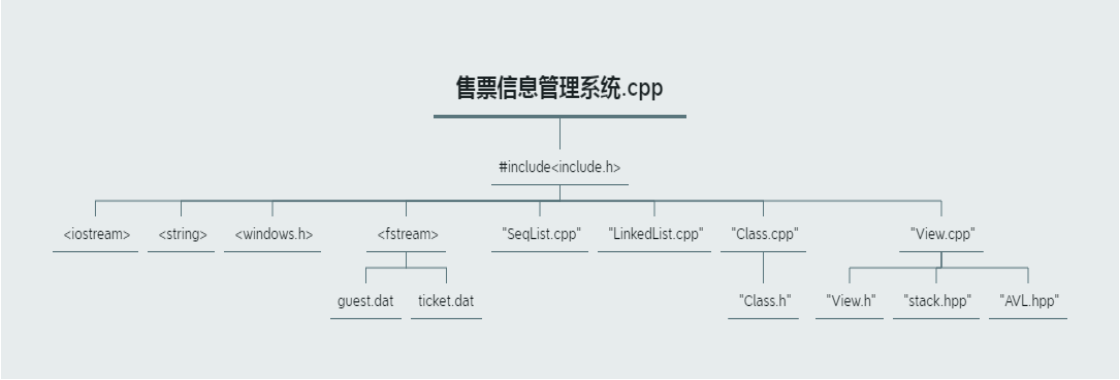

Figure 1 文档结构

图片 1 展示了售票信息管理系统的文档结构，笔者将所有的头文件放入了 <include> 文件中，将不同的功能模块分成不同的文件，便于代码修改和分工。其中，**view 类**存放了所有的界面，**class 类**存放了用户类和售票信息类。导入了 **fstream** 以实现和外部二进制文件 **guest.dat**，**ticket.dat** 进行 **IO** 操作。

```cpp
#include "include.h"            //包含所有头文件的头文件
using namespace std;

//******************控制台*********************
void setconsole()
{
    SetConsoleTitle(L"售票信息管理系统");
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), BACKGROUND_GREEN);
}

//******************主函数*************************
int main()
{
    setconsole();    //控制台修饰
     View view;
     view.input();  //输入数据
     view.MainView();  //主界面
}
```

Figure 2 主函数：通过模块分文件，使主函数简洁易读。

图片 2 中通过 **setconsole** 函数简单的对控制台进行了美化。函数定义在头文件 windows.h 中。**view** 是一个界面对象。

2.1.2 类

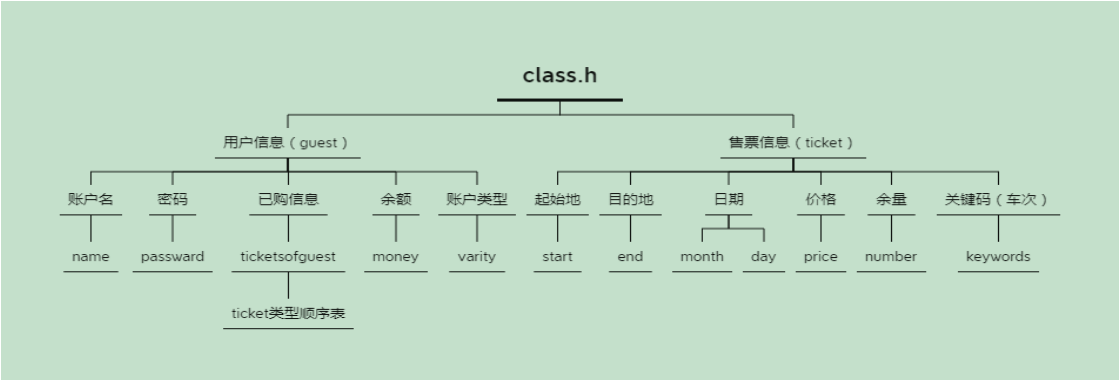

Figure 3 Class.h 结构

class.h 给出了售票信息系统最重要的两个类：存储用户信息的 **guest 类**，存储售票信息的 **ticket 类**。

ticket 类中车次用整数表示，形成了售票信息的唯一关键码，便于后续的 **AVL** 树搜索结构的实现。日期包含日和月两部分。(见附录 1)

guest 类中 **varity** 是一个整数，用于表示当前账户是普通用户还是后台管理员。**ticketsofguest** 是一个顺序表，顺序表中的**每个元素都是 ticket 类型的数据**，用于存储已经购买车票的信息。（见附录 2）

2.2 文件输入与输出

售票信息管理系统采用二进制保存信息和读取信息。用户信息保存到 guest.dat 中，售票信息保存到 ticket.dat 中。代码见附录 4。

2.3 逻辑结构与存储结构

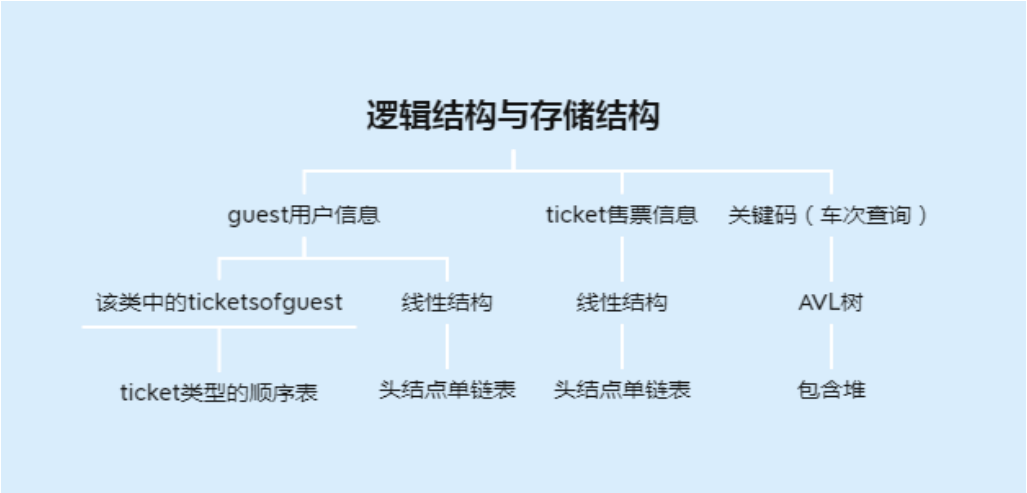

Figure 4 逻辑结构与存储结构

图 4 展示了本系统使用的逻辑结构与存储结构，其中 guest 类嵌套了顺序表，**顺序表中的每个元素都是 ticket 类型的对象**。关键码查询时构造了 **AVL** 树。

2.4 运算符重载

本系统为了满足查询，输出的要求，对 guest，ticket 中的运算符进行了重载。

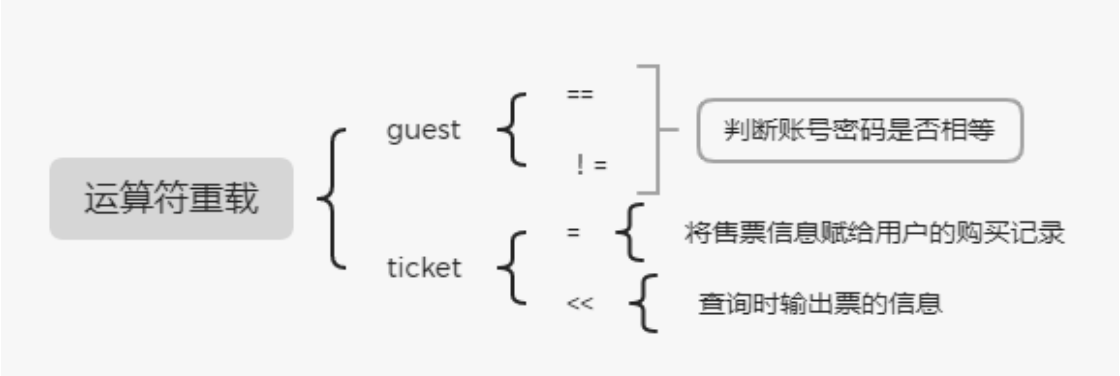

Figure 5 guest 与 ticket 的重载

其中 « 在 print 函数中使用，直接输出售票信息的信息，代码见附录 3.

2.5 界面设计

界面包括主界面，注册和登录界面，注册时分为普通用户和后台管理员，根据账户类型的不同，登录时进入不同的界面。其中查询使用的是同一个界面，返回时通过判断账户类型返回相应界面，提供三种方式的查询，车次查询构造 **AVL** 树进行查询。图 6 给出所有界面之间的关系。

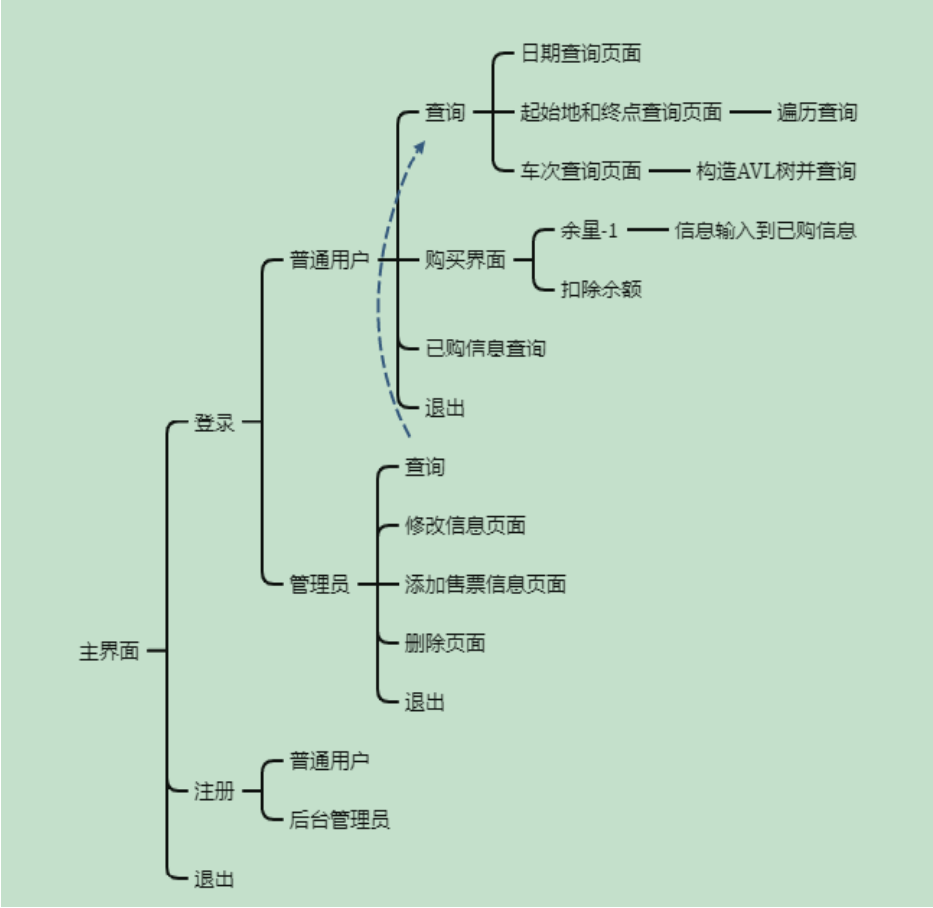

Figure 6 界面导图

2.5.1 登录注册界面设计

用户在注册时需要选择账户类型，普通用户或者后台管理员，系统界面 **view** 中含有一个当前账户的指针，系统会将账户信息记录到指针指向的 **varity**，在登录时会根据 **varity** 的值进入相应的界面。注意到，普通用户和后台管理员用了同一个查询界面，因此，从查询界面返回普通用户或者后台管理员界面时，也会根据该变量的值做出相应的“选择”。

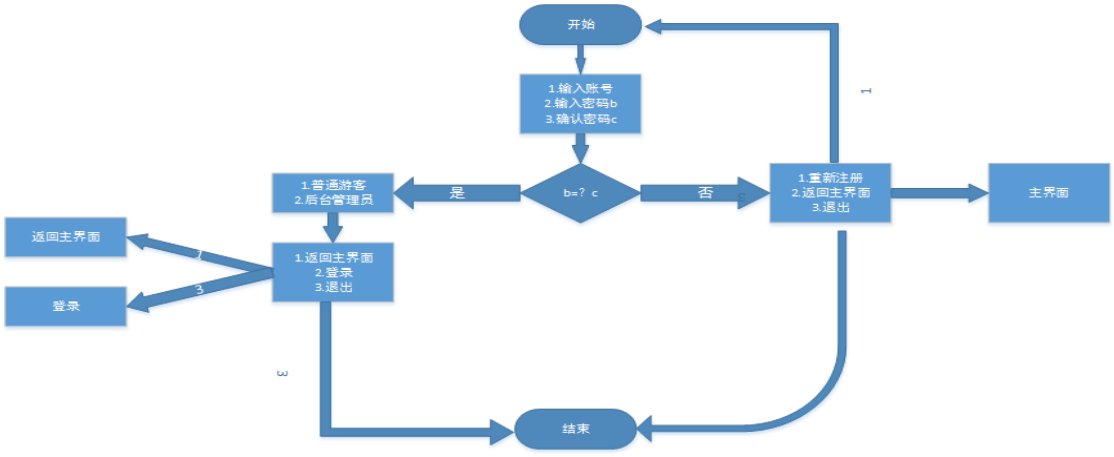

Figure 7 注册界面

图 7 给出了一个注册界面的流程图，其他界面类似，注册界面代码见附录 4。类似地，我们做出所有界面。

2.6 搜索功能的实现

本系统提供了三种方式的搜索：按日期搜索，按初始地终点搜索，按车次搜索。其中车次搜索使用了 **ticket 类 AVL 树**。

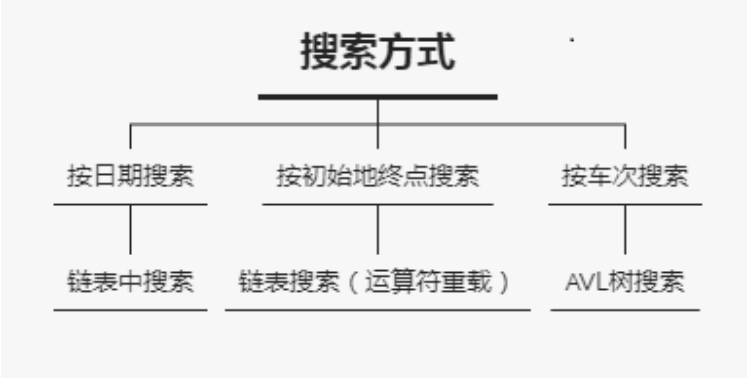

Figure 8 注册界面

III. 系统运行结果及使用说明

3.1 系统运行结果

通过不断调试与修改，售票信息管理系统实现了售票信息的界面设计与实现，售票信息管理系统包含登录注册功能，并分为**普通用户**和**后台管理员**界面，普通用户具有购票，查询，充值，查询历史订单等功能；后台管理员可以添加，修改，删除，查询售票信息。售票信息管理系统关闭时将用户信息与售票信息保存至二**进制**文件中，打开时读取用户信息和售票信息。

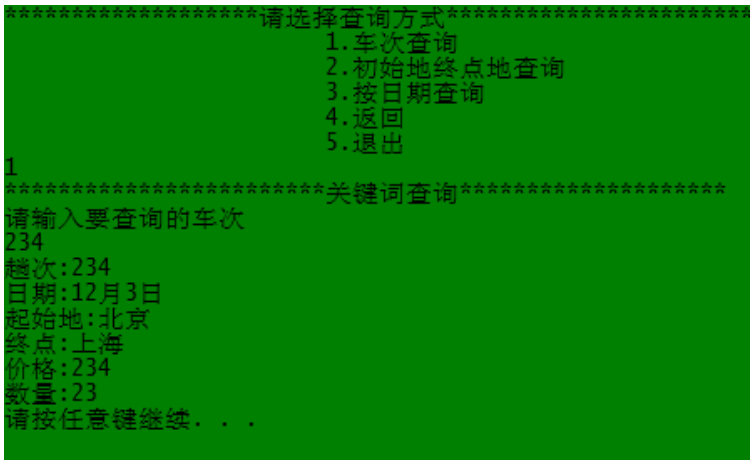

Figure 9 查询的一个例子

3.2 使用说明

1. 在使用之前，请先进行**注册**，普通游客没有删除和添加的权利；后台管理员没有购票的权利。
2. 如果你是普通游客，再购票之前，请先进行**充值**，以保证有充足的余额购票。
3. 在购票或删除之前请先**查询**要购买的车次。日期分为月份和号数。
4. 请不要直接关闭系统，请使用界面菜单进行退出，否则你本次的信息不会保存。

IV. 结论与改进

4.1 运算符重载重载

当我们遇到一些复杂结构时候，为了能够对对象进行操作，我们需要对其进行运算符重载。

运算符重载包括类内重载和类外重载，若参数用 value 表示，目数用 number 表示，类内重载因为含有 this 指针

$$value = number - 1$$

, 而在类外

$$value = number$$

4.2 模板类产生的类链表

本售票信息管理系统采用了模板类链表，而这里的 T 是一个类，因此我们遇到了这样的写法

```
current->data.ticketsofguest.insert(tickets.locate(i)->data);
current->data.money -= tickets.locate(i)->data.price;
```

Figure 10 对象的对象

其实我们只要把模板 T 看作是一个整体，当把它单独拿出来的时候，再把它拆分开，对它进行操作。这正是模板类的精髓之处。

4.3 序列化输出类链表

正是因为类链表中含有类顺序表，导致数据结构变得复杂，因此二进制输入输出不能将 guest 中的顺序表输出，本系统仅仅将用户信息输出，购票历史没有输出，这正是本系统的缺陷，也是未来的改进方向。

序列化 (Serialization) 是将对象的状态信息转换为可以存储或传输的形式的过程。在序列化期间，对象将其当前状态写入到临时或持久性存储区。以后，可以通过从存储区中读取或反序列化对象的状态，重新创建该对象。通过序列化方式将两个链表数据输入到一个文件中是未来的改进方向。

V. References

[1] 李杨, 李雯, 胡志鹏, 戴琳琳.12306 互联网售票系统余票数据一致性保障技术方案研究 [J]. 铁路计算机应用,2021,30(04):11-16.

[2] Y.Daniel Liang, Introduction to Programming, 北京：机械工业出版社, 2020.

[3] 殷人昆, 数据结构：用面向对象方法与 C++ 语言描述, 北京：清华大学出版社，2007.6

VI. Appendix

Listing 1: ticket 类

```
class ticket
{
public:
int keywords;
int month;
int day;
char start[10];
char end[10];
int price;
int number;
public:
bool ifkeywords(int n);
void print();
friend ostream& operator<<(ostream& os, const ticket& c);
};
```

Listing 2: guest 类

```
class guest
{
public:
char name[10];
char passward[15];
int money;
SeqList<ticket> ticketsofguest;
int varity;
public:
void print();
};
```

Listing 3: guest 和 ticket 的运算符重载

```
//**************************************************guest运算符重载*****************************************
bool operator==(guest a, guest b)
{
return (!strcmp(a.name, b.name) && !strcmp(a.passward, b.passward));
}

bool operator!=(guest& a, guest& b)
{
return strcmp(a.name, b.name) || strcmp(a.passward, b.passward);
}


//**************************************************ticket运算符重载*****************************************
```

```
void operator&=(ticket& a, ticket& b)
{
a.keywords=b.keywords;
a.month=b.month;
a.day = b.day;
strcpy_s(a.start,b.start);
strcpy_s(a.end,b.end);
a.price=b.price;
a.number = 1;
}

ostream&operator<<(ostream& os, const ticket & c)
{
os << "车次:" << c.keywords << endl;
os << "日期:" << c.month << "月" << c.day << "日" << endl;
os << "出发地:" << c.start << endl;
os << "终点:" << c.end << endl;
os << "价格:" << c.price << endl;
return os;
}
```

Listing 4: 注册界面代码

```
void View::signinview()
{
while(1)
{
system("cls");
cout << "请输入账号" << endl;
char a[10];
char b[15];
char c[15];
cin >> a;
cout << "请输入密码" << endl;
cin >> b;
cout << "确认密码" << endl;
cin >> c;
if (!(strcmp(b,c)))
{
cout << "请选择注册类型" << endl;
cout << "         1.普通游客" << endl;
cout << "         2.后台管理员" << endl;
int d;
cin >> d;
guest current;
strcpy_s(current.name,a);
strcpy_s(current.password, b);
current.varity = d;
guests.insert(0,current);
```

```
system("cls");
cout << "********************注册成功!**********************" << endl;
cout << "      1.返回主界面" << endl;
cout << "      2.登录" << endl;
cout << "      3.退出" << endl;
cout << "************************************************" << endl;
int n;
cin >> n;

switch (n)
{
case 1:MainView(); break;
case 2:loginview(); break;
case 3:shutdownview();
default:signinview();
}

break;
}
else
{
system("cls");
cout << "********************两次输入密码不一致，是否重新注册**********************"
    << endl;
cout << "         1.重新注册" << endl;
cout << "         2.返回主界面" << endl;
cout << "         3.退出" << endl;
cout << "************************************************" << endl;
int n;
cin >> n;

switch (n)
{
case 1:signinview(); break;
case 2:MainView(); break;
case 3:shutdownview();
default:loginview();
}

}
}
}
```