搭建框架代码--以项目2为例

在任务1中我们已经初步建立了项目的数据结构，现在需要利用已有的类及成员函数搭建**可正确运行的项目框架**，要求实现**功能的选择以及功能界面的跳转**。在搭建框架代码的同时完成为先前定义的类成员函数**填充完整的参数和返回值类型**

回顾任务1的指导中，我们定义了3类对象，管理员`admin`、用户`user`以及整个系统`My12306`。首先做一个简单回顾：

```
// admin.h
class admin
{
    private:
        static vector<train> trains; // 注意，添加static修饰后，车次信息trains由所有管理员对象共享
        static vecotr<ticket> tickets; // 同理所有管理员都能查看全部的旅客购票记录
    public:
        admin();//构造函数声明
        void import();//用于从文件导入车次信息、购票记录，分别存储到trains和tickets中；
        void save();//用于保存车次、购票记录到文件
        void read();//手动导入车次信息
        void add();//添加车次信息
        void del();//删除
        void revise();//修改
        void query_shift();//按班次号查询余票
        void query_station();//按起始站查询余票
        void sell_ticket();//根据要求售票，更新余票信息
        void refund_ticket();//根据要求退票，更新余票信息
}

//user.h
class user
{
    private:
        vector<ticket> tickets; // 每个旅客只能查看自己的购票记录,不需要static修饰
    public:
        user();//构造函数声明
        void load();//用于加载该用户的订单
        void save();//用于保存该用户的订单
        void book_ticket();//旅客购票，完成订单信息记录
        void cancel_ticket();//旅客取消订单，删除订单信息
        void show_tickets();//订单浏览
}
```

上述定义对原有的`admin`和`user`定义进行了修改，特别是使用了类的静态成员这一特性，因为注意到管理员的职责比较同质化，故除了成员变量，每个成员函数也可使用`static`修饰。此处我们暂时不考虑拓展功能中“用户系统”的身份鉴别问题，且可认为基础功能中只有一个管理员，一个旅客，两者身份可以无痕切换不被察觉。简言之，`admin`类和`user`类的定义只是简单地给原本面向过程中的一大堆功能函数进行了分类封装。

接下来我们回顾整个项目的类`My12306`

```
// My12306.h
class My12306
{
    private:
        admin adminster;
        user user;
    public:
        My12306();//构造函数
        void main_menu();//主界面
        void load_train();//车次信息录入界面
        void modify_train();//操作班次信息界面
        void check_train();//查询班次信息界面
        void ticket_menu();//票务系统界面
        void output();//导出界面
}
```

显然，界面跳转和功能选择并不属于`admim`和`user`中任何一类对象的职责，且理所应当在`My12306`中实现，下面给出部分代码，作为大家搭建框架代码的参考

```
//My12306.cpp
void My12306::main_menu(){
    {
    cout << "\t\t\t-----------------------------\t\t\t" << endl
        << "\t\t\t|       1.录入班次信息        |\t\t\t" << endl
        << "\t\t\t|       2.操作班次信息        |\t\t\t" << endl
        << "\t\t\t|       3.查询班次信息        |\t\t\t" << endl
        << "\t\t\t|       4.售票退票系统        |\t\t\t" << endl
        << "\t\t\t|       5.导出票务信息        |\t\t\t" << endl
        << "\t\t\t|       6.退出系统            |\t\t\t" << endl
        << "\t\t\t-----------------------------\t\t\t" << endl;
    int n;
loop:cout << "请输入您选择的功能: ";
    cin >> n; cout << endl;
    switch (n)
    {
    case 1:load_train(); goto loop;
    case 2:modify_train(); goto loop;
    case 3:check_train(); goto loop;
    case 4:ticket_menu(); goto loop;
    case 5:output(); goto loop;
    case 6: cout << "已经退出系统! " << endl; break;
    default:goto loop;
    }
}
```

以上是主界面的实现，相信对大家而言也没什么难度，注意代码中使用`goto`语句，为了简化代码量，当然并不建议大家平时编程时任意使用`goto`跳转

```
//My12306.cpp
void My12306::load_train(){
    cout << "\t\t\t---------------------------\t\t\t" << endl
```

```
        << "\t\t\t|       1.从文件导入          |\t\t\t" << endl
        << "\t\t\t|       2.手动添加            |\t\t\t" << endl
        << "\t\t\t|       3.返回主菜单          |\t\t\t" << endl
        << "\t\t\t----------------------------\t\t\t" << endl;
loop:cout << "请输入您选择的功能: ";
    cin >> n; cout<<endl;
    switch (n)
    {
        case 1:
            // do some input prompt and get input
            admin.import(/*input*/);
            // do some output prompt
            goto loop;
        case 2:
            // do some input prompt and get input
            admin.read(/*input*/);
            // do some output prompt
            goto loop;
        case 3:
            break;
        default:
            goto loop;
    }
}

void My12306::modify_train(){
    cout << "\t\t\t----------------------------\t\t\t" << endl
        << "\t\t\t|       1.删除班次            |\t\t\t" << endl
        << "\t\t\t|       2.修改班次            |\t\t\t" << endl
        << "\t\t\t|       3.返回主菜单          |\t\t\t" << endl
        << "\t\t\t----------------------------\t\t\t" << endl;
loop:cout << "请输入您选择的功能: ";
    cin >> n; cout<<endl;
    switch (n)
    {
        case 1:
            // do some input prompt and get input
            // 譬如提示输入班次号，作为参数传递给admin.del()
            admin.del(/*input*/);
            // do some output prompt
            goto loop;
        case 2:
            // do some input prompt and get input
            admin.revise(/*input*/);
            // do some output prompt
            goto loop;
        case 3:
            break;
        default:
            goto loop;
    }
}

void My12306::check_train(){
    cout << "\t\t\t----------------------------\t\t\t" << endl
```

```
        << "\t\t\t|       1.按班次号查询         |\t\t\t" << endl
        << "\t\t\t|       2.按出发达到站查询     |\t\t\t" << endl
        << "\t\t\t|       3.返回主菜单           |\t\t\t" << endl
        << "\t\t\t----------------------------\t\t\t" << endl;
loop:cout << "请输入您选择的功能：";
    cin >> n; cout<<endl;
    switch (n)
    {
        case 1:
            // do some input prompt and get input
            // 譬如提示输入班次号，作为参数传递给admin.query_shift()
            admin.query_shift(/*input*/);
            // do some output prompt
            // 可以在query_shift打印信息，也可以将班次信息作为返回值在output prompt进行打印，个人建议后者
            goto loop;
        case 2:
            // do some input prompt and get input
            admin.revquery_station(/*input*/);
            // do some output prompt
            goto loop;
        case 3:
            break;
        default:
            goto loop;
    }
}

void My12306::ticket_menu(){
    cout << "\t\t\t----------------------------\t\t\t" << endl
        << "\t\t\t|       1.售票               |\t\t\t" << endl
        << "\t\t\t|       2.退票               |\t\t\t" << endl
        << "\t\t\t|       3.订单浏览           |\t\t\t" << endl
        << "\t\t\t|       4.返回主菜单         |\t\t\t" << endl
        << "\t\t\t----------------------------\t\t\t" << endl;
loop:cout << "请输入您选择的功能：";
    cin >> n; cout<<endl;
    switch (n)
    {
        case 1:
            buy_choice: cout<<"若根据班次购买请输入1；若根据出发到达站购买请输入2："<<endl;
            cin>>n;cout<<endl;
            if(1 == n){
                // do some input prompt and get input
                admin.query_shift(/*input*/);//查询余票
            }
            else(2 == n){
                // do some input prompt and get input
                admin.query_station(/*input*/);//查询余票
            }
            else
                goto buy_choice;
            // 显示余票数，获取旅客是否购票以及购票数量的input
            admin.sell_ticket(/*input*/);//确认售票,返回订单信息
            user.book_ticket(/*ticket info*/);//旅客保存订单信息
```

```
            // do some output prompt
            goto loop;
        case 2:
            // do some input prompt and get input
            // 譬如获取订单号
            uesr.cancel_ticket(/*input*/);//确认退票,返回订单信息
            admin.refund_ticket(/*ticket info*/);//系统恢复余票信息
            // do some output prompt
            goto loop;
        case 3:
            user.show_tickets();//显示旅客的订单信息
            goto loop;
        case 4:
            break;
        default:
            goto loop;
    }
}
```

以上代码基本把整个项目的框架搭建起来，且是保证可以执行的。显然`admin`和`user`中的函数成员并没有具体实现，我们可以简单地在这些函数内部打印被调用信息（`in xxx function`），用于**检查框架运行时的执行路径**是否正确。

另外，一个完整的框架还需要把一些**输入输出提示完成**（上述代码中统一用注释标记），这些输入输出提示用于获取从键盘输入的运行时信息，同时也可以帮助帮助我们确定`admin`和`user`中函数成员的参数和返回值类型。

为函数成员**填充完整的参数和返回值类型**后，类的定义才算彻底完成，同时每个函数的具体功能也得到了明确，从而可以开始下一步，具体函数功能的实现。