**车辆管理系统之设计报告**

**题目：车辆管理系统之设计报告**

**班级：计算机大类1701班**

**组长：魏锐颖2017011152**

**组员：周峻宇2017011155**

**向娜2017011153**

**王伟红2017011151**

**指导老师：马旭平**

**日期：2018.6.28**

1. **总体设计**

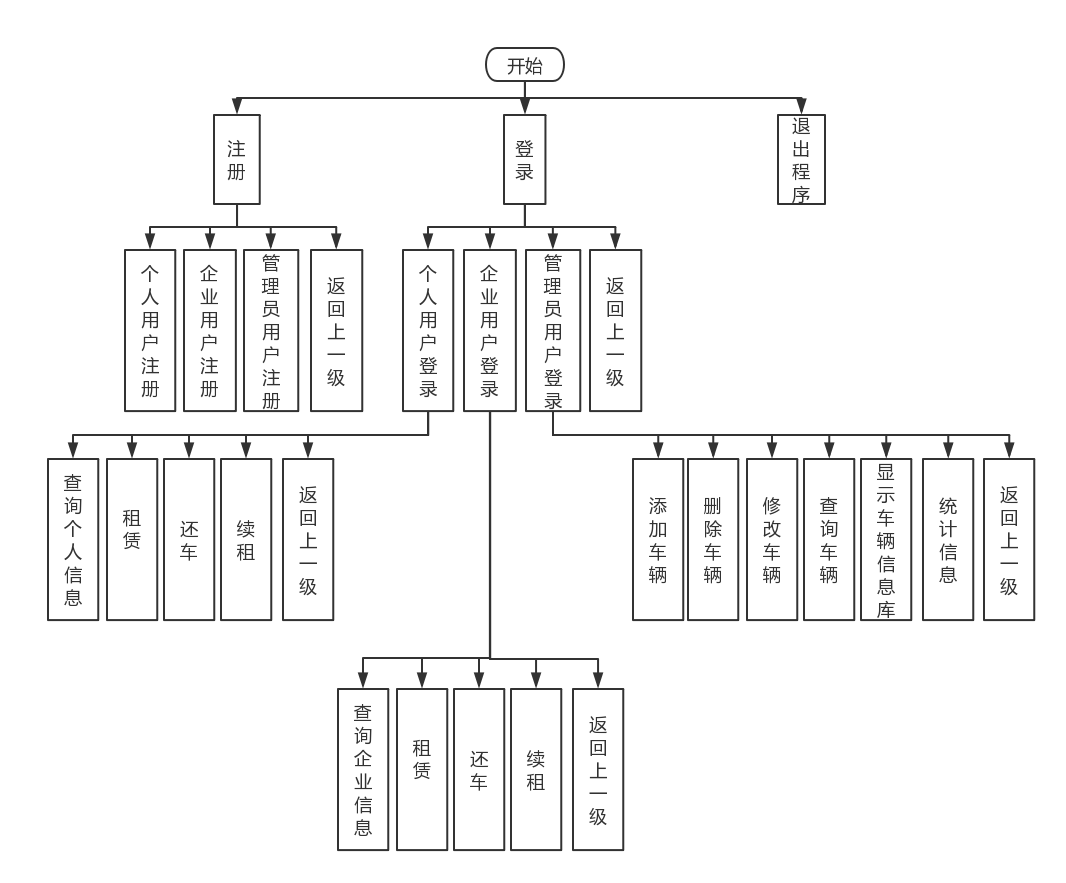
**1.1系统功能分析**

本车辆管理系统主要分为两个大的模块，一个是客户用来租赁车辆的租借模块，一个是管理员用来管理车辆的信息模块。进入系统后通过MainMenu()选择注册/登录，选择后通过RegMenu(),LoginMenu()函数来选择注册/登录角色，通过RegPerson(), RegCompany(), RegAdmin()函数来确认角色是否注册成功。通过LoginPerson(), LoginCompany(), LoginAdmin()函数进行角色的登录。其中个人/企业为客户，管理员管理车辆。进入界面后通过PersonMenu(), CompanyMenu(), AdminMenu()来显示角色的主界面。（1）客户租赁车辆的模块通过PrintInfo(), BorrowAuto(), BackAuto(), BorrowLong()来查询个人/企业信息，租赁，还车，续租。（2）管理员的管理车辆信息模块通过调用AddAuto(), DelAuto(), ModAuto()函数来实现对三种车辆信息的添加，删除，修改，通过AutoSeacherMenu()函数选择查询车辆的方式，再通过SeacherAutoByProduct(), SeacherAutoByID(), SeacherAutoByType()函数来从制造公司/编号/类型来对车进行信息查询，通过PrintAllAutos()函数输出所查询所有车辆信息，通过TotalAuto()函数进行车辆数量统计。通过LoadData(), SaveData()函数实现数据的加载和保存。

**1.2系统功能模块划分和设计**

本系统定义了两个模块，(1)定义User类，一级派生类PersonalInfo类，二级派生类Admin类，Client类，三级派生类由Client类派生的Personal类和Company类，(2)还定义了Auto类及它的派生类Bus类，Car类，Truck类。除此之外，还有Rent()类。并定义了RegMenu(),LoginMenu(), LoadData(), SaveData()等函数。

**1.3系统功能模块图**

****

**1.4类的设计**

**1.4.1**

User类——PersonalInfo类——（1）Admin类（2）Client类——（2.1）Personal类（2.2）Company类

**1.4.2**

Auto类——Car类，Bus类，Truck类

**1.4.3**

Rent类

**1.5选题要求**

**1.5.1 封装性：**

用类进行封装,User类,PersonalInfo类,Admin类,Client类,Personal类,Company类,Auto类,Car类,Bus类,Truck类,Rent类.

**1.5.2 三级或三级以上派生：**

User类——PersonalInfo类——（1）Admin类（2）Client类——（2.1）Personal类（2.2）Company类

**1.5.3 多态性及虚函数：**

多态性：

相同的对象收到不同的消息或者不同的对象收到相同的消息时产生的不同的实现动作。

用户选择数字，每个不同的数调用的函数不一样。

void MainMenu()

{

int select;

do

{

system("cls");

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* 车辆管理系统 \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << "1. 注册" << endl;

cout << "2. 登录" << endl;

cout << "3. 退出系统" << endl;

select = InputIntInRange("\n请选择:", 1, 3);

switch(select)

{

case 1:

RegMenu();

break;

case 2:

LoginMenu();

break;

case 3:

break;

}

}while (select != 3);

}

虚函数：

Class Auto

{

virtual void add() = 0;

virtual void modify() = 0;

virtual void PrintInfo() = 0;

};

**1.5.4 文件：**

void LoadData()

{

fstream file;

file.open("data.txt", ios::in);

if (file.is\_open())

{

file >> g\_nAdmin;

int i;

for (i = 0; i < g\_nAdmin; i++)

{

file >> g\_admin[i];

}

file >> g\_nPersonal;

for (i = 0; i < g\_nPersonal; i++)

{

file >> g\_personal[i];

}

file >> g\_nCompany;

for (i = 0; i < g\_nCompany; i++)

{

file >> g\_company[i];

}

int nSize;

file >> nSize;

for (i = 0; i < nSize; i++)

{

int nType;

file >> nType;

if (nType == 1)

{

Car \*car = new Car();

file >> \*car;

g\_auto.push\_back(car);

}

else if (nType == 2)

{

Truck \*truck = new Truck();

file >> \*truck;

g\_auto.push\_back(truck);

}

else if (nType == 3)

{

Bus \*bus = new Bus();

file >> \*bus;

g\_auto.push\_back(bus);

}

}

}

file.close();

}

void SaveData()

{

fstream file;

file.open("data.txt", ios::out);

if (file.is\_open())

{

file << g\_nAdmin << "\t";

int i;

for (i = 0; i < g\_nAdmin; i++)

{

file << g\_admin[i] << "\t";

}

file << g\_nPersonal << "\t";

for (i = 0; i < g\_nPersonal; i++)

{

file << g\_personal[i] << "\t";

}

file << g\_nCompany << "\t";

for (i = 0; i < g\_nCompany; i++)

{

file << g\_company[i] << "\t";

}

int nSize = g\_auto.size();

file << nSize << "\t";

for (i = 0; i < nSize; i++)

{

int nType = g\_auto[i]->GetType();

file << nType << "\t";

if (nType == 1)

{

file << \*((Car\*)g\_auto[i]) << "\t";

}

else if (nType == 2)

{

file << \*((Truck\*)g\_auto[i]) << "\t";

}

else if (nType == 3)

{

file << \*((Bus\*)g\_auto[i]) << "\t";

}

}

}

file.close();

}

**1.5.5 数组及指针:**

数组：

class Auto

{

protected:

char color[20]; // 颜色

char brands[20]; // 品牌

char id[20]; // 编号

char product[40]; // 制造公司

int nType; // 类别 1:轿车 2:卡车 3:大客车

bool bRent; // 是否被租赁

};

class Admin : public PersonalInfo

{

private:

char idcard[20];

};

等。

代码中运用很多数组来定义信息性质。

指针：

if (nType == 1)

{

Car \*car = new Car();

file >> \*car;

g\_auto.push\_back(car);

}

else if (nType == 2)

{

Truck \*truck = new Truck();

file >> \*truck;

g\_auto.push\_back(truck);

}

else if (nType == 3)

{

Bus \*bus = new Bus();

file >> \*bus;

g\_auto.push\_back(bus);

}

**1.5.6 静态成员：**

class Personal : public Client

{

public:

static int m\_nMax; // 个人账号数量限制

};

class Company : public Client

{

public:

static int m\_nMax; // 企业账号数量限制

};

等。

**1.5.7共用数据的保护：**

// 输入整数

int InputInt(char const\* msg);

// 输入一个范围内的整数

int InputIntInRange(char const \* msg, int l, int r);

// 输入一个浮点数

double InputDouble(char const\* msg);

等。

**1.5.8 友元或类模板：**

重载了运算符“>>”,“<<”。

friend istream& operator>>(istream& in, Personal& gc)

{

in >> (Client&)gc;

in >> gc.sex >> gc.age >> gc.title >> gc.idcard;

if (gc.num > 0)

{

for (int i = 0; i < gc.num; i++)

{

in >> gc.rent[i];

}

}

return in;

}

friend ostream& operator<<(ostream& out, const Personal& gc)

{

out << (const Client&)gc << "\t";

out << gc.sex << "\t" << gc.age << "\t" << gc.title << "\t" << gc.idcard;

if (gc.num > 0)

{

for (int i = 0; i < gc.num; i++)

{

out << "\t" << gc.rent[i];

}

}

return out;

}

**2.详细设计及实现**

**2.1类的实现**

**2.1.1User类的实现**

class User

{

protected:

char id[20]; // 账号

char psw[20]; // 密码

public:

User();

User(char chID[], char chPsw[]);

char\* GetID();

void SetID(char ch[]);

char\* GetPsw();

void SetPsw(char ch[]);

// 输入操作符重载

friend istream& operator>>(istream& in, User& gc)

{

in >> gc.id >> gc.psw;

return in;

}

// 输出操作符重载

friend ostream& operator<<(ostream& out, const User& gc)

{

out << gc.id << "\t" << gc.psw;

return out;

}

};

**2.1.2PersonalInfo类的实现**

class PersonalInfo : public User

{

protected:

char name[20]; // 姓名

char company[40]; // 公司

char phone[20]; // 联系电话

public:

PersonalInfo(char chID[], char chPsw[]);

PersonalInfo();

virtual void PrintInfo() = 0;

char\* GetName();

void SetName(char ch[]);

char\* GetCompany();

void SetCompany(char ch[]);

char\* GetPhone();

void SetPhone(char ch[]);

friend istream& operator>>(istream& in, PersonalInfo& gc)

{

in >> (User&)gc;

in >> gc.name >> gc.company >> gc.phone;

return in;

}

friend ostream& operator<<(ostream& out, const PersonalInfo& gc)

{

out << (const User&)gc << "\t";

out << gc.name << "\t" << gc.company << "\t" << gc.phone;

return out;

}

};

**2.1.3Admin类的实现**

class Admin : public PersonalInfo

{

private:

char idcard[20];

public:

static int m\_nMax; // 管理员数量限制

Admin();

Admin(char chID[], char chPsw[]);

char\* GetIDCard();

void SetIDCard(char ch[]);

void PrintInfo();

void Reg();

friend istream& operator>>(istream& in, Admin& gc)

{

in >> (PersonalInfo&)gc;

in >> gc.idcard;

return in;

}

friend ostream& operator<<(ostream& out, const Admin& gc)

{

out << (const PersonalInfo&)gc << "\t";

out << gc.idcard;

return out;

}

};

**2.1.4Client类的实现**

class Client : public PersonalInfo

{

protected:

int credit; // 信誉分值

int num; // 当前租赁的车数量

public:

Client();

Client(char chID[], char chPsw[]);

int GetCredit();

void SetCredit(int n);

int GetNum();

void SetNum(int n);

friend istream& operator>>(istream& in, Client& gc)

{

in >> (PersonalInfo&)gc;

in >> gc.credit >> gc.num;

return in;

}

friend ostream& operator<<(ostream& out, const Client& gc)

{

out << (const PersonalInfo&)gc << "\t";

out << gc.credit << "\t" << gc.num;

return out;

}

};

**2.1.5Personal类的实现**

class Personal : public Client

{

private:

bool sex; // true:男 false:女

int age; // 年龄

char title[20]; // 职位

char idcard[20]; // 身份证

Rent rent[2];

public:

static int m\_nMax; // 个人账号数量限制

Personal();

Personal(char chID[], char chPsw[]);

bool GetSex();

void SetSex(bool b);

int GetAge();

void SetAge(int n);

char\* GetTitle();

void SetTitle(char ch[]);

char\* GetIDCard();

void SetIDCard(char ch[]);

void PrintInfo();

void BorrowAuto();

void BackAuto();

void BorrowLong();

friend istream& operator>>(istream& in, Personal& gc)

{

in >> (Client&)gc;

in >> gc.sex >> gc.age >> gc.title >> gc.idcard;

if (gc.num > 0)

{

for (int i = 0; i < gc.num; i++)

{

in >> gc.rent[i];

}

}

return in;

}

friend ostream& operator<<(ostream& out, const Personal& gc)

{

out << (const Client&)gc << "\t";

out << gc.sex << "\t" << gc.age << "\t" << gc.title << "\t" << gc.idcard;

if (gc.num > 0)

{

for (int i = 0; i < gc.num; i++)

{

out << "\t" << gc.rent[i];

}

}

return out;

}

void Reg();

};

**2.1.6Company类的实现**

class Company : public Client

{

private:

char email[20]; // 身份证

Rent rent[3]; // 租赁信息

public:

static int m\_nMax; // 企业账号数量限制

Company();

Company(char chID[], char chPsw[]);

char\* GetEmail();

void SetEmail(char ch[]);

void PrintInfo();

void BorrowAuto();

void BackAuto();

void BorrowLong();

friend istream& operator>>(istream& in, Company& gc)

{

in >> (Client&)gc;

in >> gc.email;

if (gc.num > 0)

{

for (int i = 0; i < gc.num; i++)

{

in >> gc.rent[i];

}

}

return in;

}

friend ostream& operator<<(ostream& out, const Company& gc)

{

out << (const Client&)gc << "\t";

out << gc.email;

if (gc.num > 0)

{

for (int i = 0; i < gc.num; i++)

{

out << "\t" << gc.rent[i];

}

}

return out;

}

void Reg();

};

**2.1.7Auto类（抽象基类）的实现**

class Auto

{

protected:

char color[20]; // 颜色

char brands[20]; // 品牌

char id[20]; // 编号

char product[40]; // 制造公司

int nType; // 类别 1:轿车 2:卡车 3:大客车

bool bRent; // 是否被租赁

public:

static int m\_nMax; // 最多车辆数量

Auto();

Auto(char chColor[], char chBrands[], char chID[], char chProduct[]);

char\* GetColor();

void SetColor(char ch[]);

char\* GetBrands();

void SetBrands(char ch[]);

char\* GetID();

void SetID(char ch[]);

char\* GetProduct();

void SetProduct(char ch[]);

int GetType();

void SetType(int n);

bool GetRentStatu();

void SetRentStatu(bool b);

virtual void add() = 0;

virtual void modify() = 0;

virtual void PrintInfo() = 0;

friend istream& operator>>(istream& in, Auto& gc)

{

in >> gc.color >> gc.brands >> gc.id >> gc.product >> gc.nType >> gc.bRent;

return in;

}

friend ostream& operator<<(ostream& out, const Auto& gc)

{

out << gc.color << "\t" << gc.brands << "\t" << gc.id << "\t" << gc.product << "\t" << gc.nType << "\t" << gc.bRent;

return out;

}

};

**2.1.8Car类的实现**

class Car : public Auto

{

private:

char type[20]; // 微型车 小型车 紧凑型车 中型车 中大型车 大型车

public:

Car();

Car(char chColor[], char chBrands[], char chID[], char chProduct[]);

char\* GetCarType();

void SetCarType(char ch[]);

void add();

void modify();

void PrintInfo();

int GetType();

friend istream& operator>>(istream& in, Car& gc)

{

in >> (Auto&)gc;

in >> gc.type;

return in;

}

friend ostream& operator<<(ostream& out, const Car& gc)

{

out << (const Auto&)gc << "\t";

out << gc.type;

return out;

}

};

**2.1.9 Bus类的实现**

class Bus : public Auto

{

private:

int seats;

public:

Bus();

Bus(char chColor[], char chBrands[], char chID[], char chProduct[]);

int GetSeats();

void SetSeats(int n);

void add();

void modify();

void PrintInfo();

int GetType();

friend istream& operator>>(istream& in, Bus& gc)

{

in >> (Auto&)gc;

in >> gc.seats;

return in;

}

friend ostream& operator<<(ostream& out, const Bus& gc)

{

out << (const Auto&)gc << "\t";

out << gc.seats;

return out;

}

};

**2.1.10Truck类的实现**

class Truck : public Auto

{

private:

double weight; // 载重容量

public:

Truck();

Truck(char chColor[], char chBrands[], char chID[], char chProduct[]);

double GetWeight();

void SetWeight(double d);

void add();

void modify();

void PrintInfo();

int GetType();

friend istream& operator>>(istream& in, Truck& gc)

{

in >> (Auto&)gc;

in >> gc.weight;

return in;

}

friend ostream& operator<<(ostream& out, const Truck& gc)

{

out << (const Auto&)gc << "\t";

out << gc.weight;

return out;

}

};

**2.1.11Rent类的实现**

class Rent

{

private:

char carID[20]; // 车辆ID

char userID[20]; // 用户账号

char backDate[20]; // 归还日期

public:

Rent();

Rent(char chCarID[20], char chUserID[20], char chBackDate[20]);

char\* GetCarID();

void SetCarID(char ch[]);

char\* GetUserID();

void SetUserID(char ch[]);

char\* GetBackDate();

void SetBackDate(char ch[]);

bool Overdue();

void AddRent();

void PrintInfo();

friend istream& operator>>(istream& in, Rent& gc)

{

in >> gc.carID >> gc.userID >> gc.backDate;

return in;

}

friend ostream& operator<<(ostream& out, const Rent& gc)

{

out << gc.carID << "\t" << gc.userID << "\t" << gc.backDate;

return out;

}

};

**2.2 功能模块分析**

**2.2.1 模块一 主菜单**

void MainMenu()

{

int select;

do

{

system("cls");

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* 车辆管理系统? \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

cout << "1. 注册" << endl;

cout << "2. 登录" << endl;

cout << "3. 退出系统" << endl;

select = InputIntInRange("\n请选择:", 1, 3);

switch(select)

{

case 1:

RegMenu();

break;

case 2:

LoginMenu();

break;

case 3:

cout<<"感谢您的使用!"<<endl;

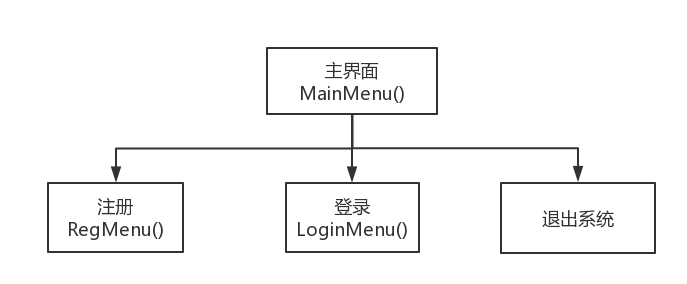
break;

}

}while (select != 3);

|  |  |  |
| --- | --- | --- |
| **主菜单选择** | | |
| **Case：1** | **Case：2** | **Case：3** |
| **注册** | **登录** | **退出系统** |

}

****

**2.2.2模块二 注册界面**

void RegMenu()

{

int select;

do

{

system("cls");

cout << "1. 个人用户注册" << endl;

cout << "2. 企业用户注册" << endl;

cout << "3. 管理员用户注册" << endl;

cout << "4. 返回上一级菜单 << endl;

select = InputIntInRange("\n请选择:", 1, 4);

switch(select)

{

case 1:

RegPerson();

break;

case 2:

RegCompany();

break;

case 3:

RegAdmin();

break;

case 4:

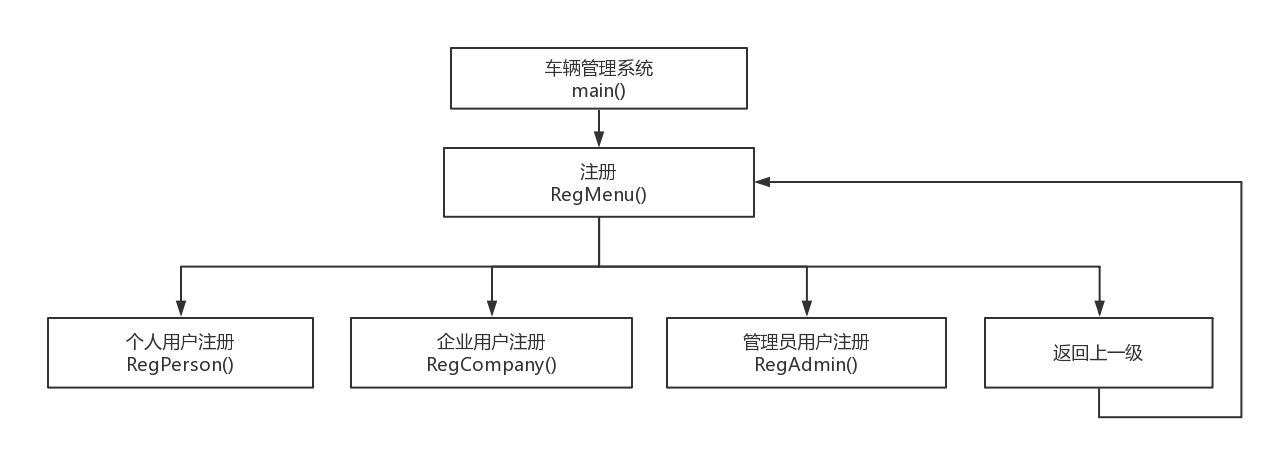
break;

}

}while (select != 4);

}

|  |  |  |  |
| --- | --- | --- | --- |
| **注册界面** | | | |
| **Case：1** | **Case：2** | **Case：3** | **Case：4** |
| **个人用户注册** | **企业用户注册** | **管理员注册** | **返回上一级** |

****

**2.2.3模块三 登录**

void LoginMenu()

{

int select;

do

{

system("cls");

cout << "1. 个人用户登录" << endl;

cout << "2. 企业用户登录" << endl;

cout << "3. 管理员用户登录" << endl;

cout << "4. 返回上一级菜单 << endl;

select = InputIntInRange("\n请选择:", 1, 4);

switch(select)

{

case 1:

LoginPerson();

break;

case 2:

LoginCompany();

break;

case 3:

LoginAdmin();

break;

case 4:

break;

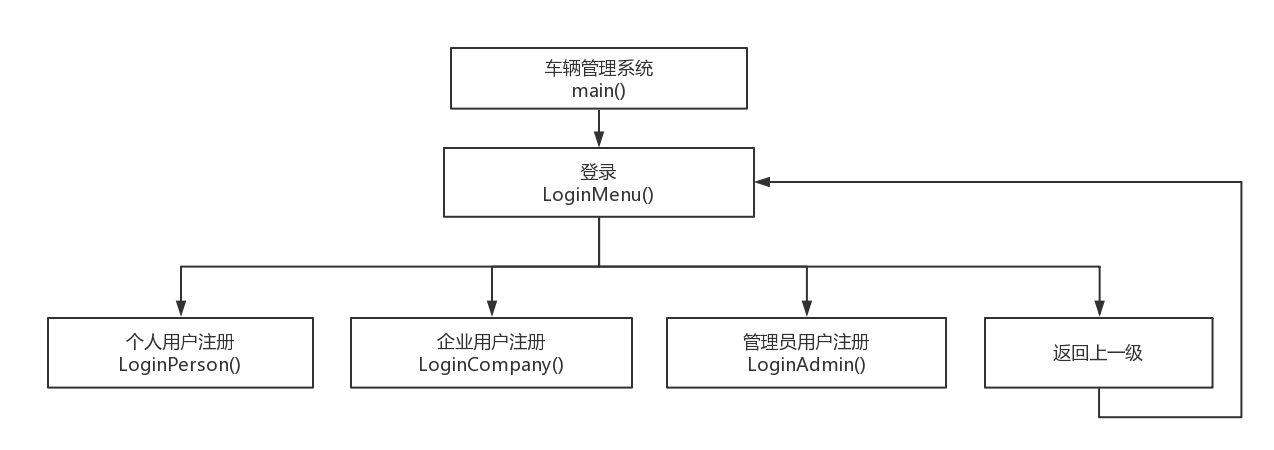
}

system("pause");

}while (select != 4);

}

|  |  |  |  |
| --- | --- | --- | --- |
| **登录界面** | | | |
| **Case：1** | **Case：2** | **Case：3** | **Case：4** |
| **个人用户登录** | **企业用户登录** | **管理员登录** | **返回上一级** |

****

**2.2.4模块四 用户界面**

**2.2.4.1 个人用户界面**

void PersonMenu()

{

int select;

do

{

cout << endl;

cout << "1. 查询个人信息" << endl;

cout << "2. 租赁" << endl;

cout << "3. 还车" << endl;

cout << "4. 续租" << endl;

cout << "5. 返回上一级菜单 << endl;

select = InputIntInRange("\n请选择:", 1, 5);

switch(select)

{

case 1:

g\_personal[g\_countIndex].PrintInfo();

break;

case 2:

g\_personal[g\_countIndex].BorrowAuto();

break;

case 3:

g\_personal[g\_countIndex].BackAuto();

break;

case 4:

g\_personal[g\_countIndex].BorrowLong();

break;

case 5:

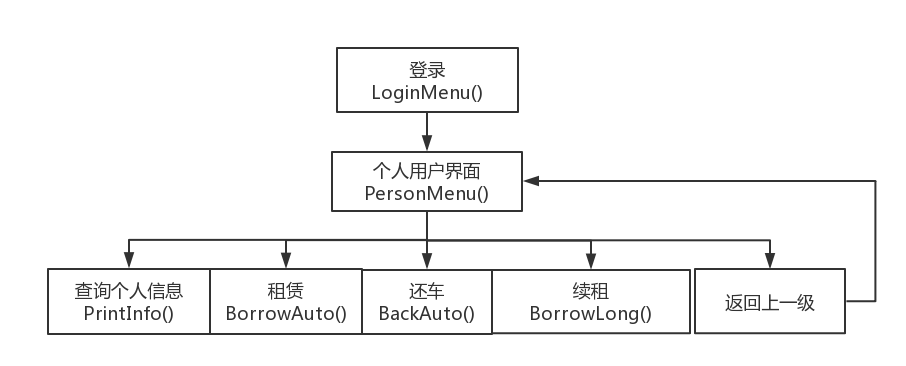
break;

}

}while (select != 5);

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **个人用户界面** | | | | |
| **Case：1** | **Case：2** | **Case：3** | **Case：4** | **Case：5** |
| **查询个人信息** | **租赁** | **还车** | **续租** | **返回上一级** |

****

**2.2.4.2 企业用户界面**

void CompanyMenu()

{

int select;

do

{

cout << endl;

cout << "1. 查询公司信息" << endl;

cout << "2. 租赁" << endl;

cout << "3. 还车" << endl;

cout << "4. 续租" << endl;

cout << "5. 返回上一级菜单 << endl;

select = InputIntInRange("\n请选择:", 1, 5);

switch(select)

{

case 1:

g\_company[g\_countIndex].PrintInfo();

break;

case 2:

g\_company[g\_countIndex].BorrowAuto();

break;

case 3:

g\_company[g\_countIndex].BackAuto();

break;

case 4:

g\_company[g\_countIndex].BorrowLong();

break;

case 5:

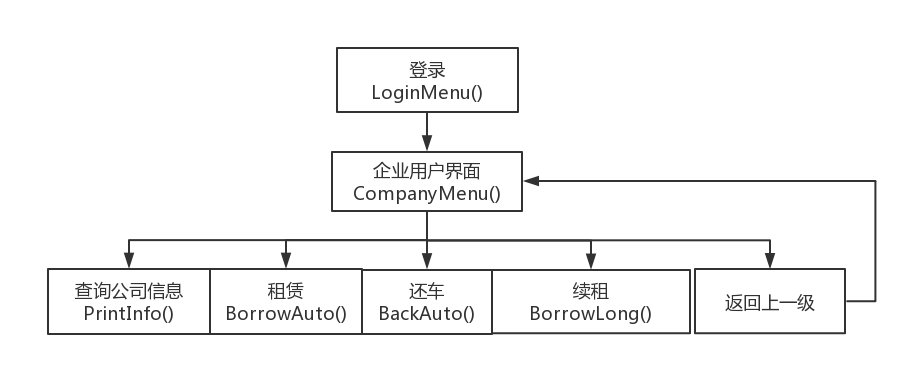
break;

}

}while (select != 5);

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **企业用户界面** | | | | |
| **Case：1** | **Case：2** | **Case：3** | **Case：4** | **Case：5** |
| **查询公司信息** | **租赁** | **还车** | **续租** | **返回上一级** |

****

**2.2.4.3 管理员用户界面**

void AdminMenu()

{

int select;

do

{

system("cls");

cout << "1. 添加车辆" << endl;

cout << "2. 删除车辆" << endl;

cout << "3. 修改车辆" << endl;

cout << "4. 查询车辆" << endl;

cout << "5. 显示车辆信息库" << endl;

cout << "6. 统计信息" << endl;

cout << "7. 返回上一级菜单 << endl;

select = InputIntInRange("\n请选择:", 1, 7);

switch(select)

{

case 1:

AddAuto();

break;

case 2:

DelAuto();

break;

case 3:

ModAuto();

break;

case 4:

AutoSearchMenu();

break;

case 5:

PrintAllAutos();

break;

case 6:

TotalAuto();

break;

case 7:

break;

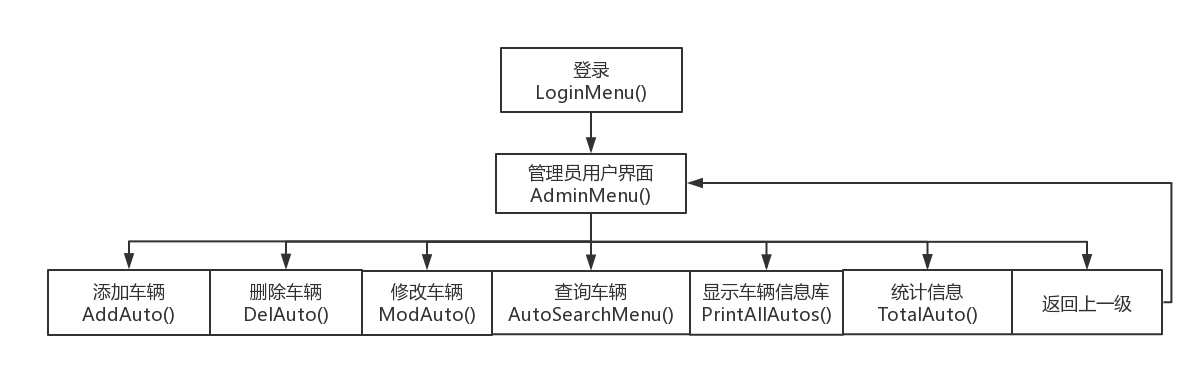
}

system("pause");

}while (select != 7);

}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **管理员用户界面** | | | | | | |
| **Case：1** | **Case：2** | **Case：3** | **Case：4** | **Case：5** | **Case：6** | **Case：7** |
| **添加车辆** | **删除车辆** | **修改车辆** | **查询车辆** | **显示车辆信息库** | **统计信息** | **返回上一级** |

****

**2.2.5 查询车辆界面**

void AutoSearchMenu()

{

int select;

do

{

cout << endl;

cout << "1. 按车辆制造公司查询" << endl;

cout << "2. 按编号查询" << endl;

cout << "3. 按类别查询" << endl;

cout << "4. 返回上一级菜单 << endl;

select = InputIntInRange("\n请选择:", 1, 4);

switch(select)

{

case 1:

SearchAutoByProduct();

break;

case 2:

SearchAutoByID();

break;

case 3:

SearchAutoByType();

break;

case 4:

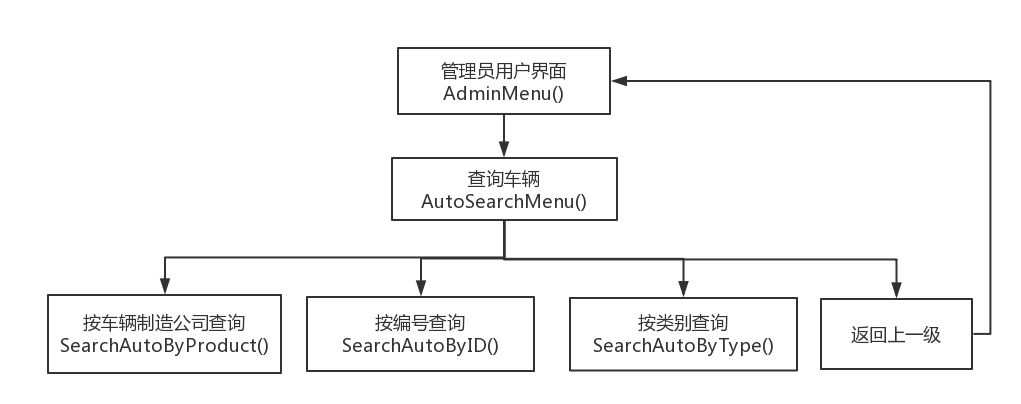
break;

}

}while (select != 4);

|  |  |  |  |
| --- | --- | --- | --- |
| **查询车辆界面** | | | |
| **Case：1** | **Case：2** | **Case：3** | **Case：4** |
| **按车辆制造公司查询** | **按编号查询** | **按类别查询** | **返回上一级** |

}

****

**3.参考资料**

陈明 C++面向对象程序设计教程 清华大学出版社

郑伟 C++程序设计 西安电子科技大学出版社

安金梁 Visual C++程序设计与项目实践 电子工业出版社

朱金付 C++课程设计 清华大学出版社

谭浩强 C++面向对象程序设计 清华大学出版社