**第一部分 第1.3题：停车场管理**

**实验报告**

题目：设计一个一元稀疏多项式简单计算器。

班级：F1702128 姓名：赵伟基 学号：517021910883 完成日期：2018-10-28

1. **需求分析**
2. 本问题要求用栈模拟停车场，以队列模拟车厂外的便道，通过输入车辆状态（到达，离开停车场，离开便道），汽车牌照号码以及到达或者离开的时刻来对停车场进行模拟管理显示停车位置，停留时间和应缴的费用。
3. 输入需求：

程序对输入状态用A，D，C，Q，E等字母表示，能够接受5个指令到达指令（‘A’，车牌号，到达时间）；离开停车场（‘D’，车牌号，离开时间）；离开便道（‘Q’，车牌号）；查询车场及便道信息（‘C’）以及退出程序（‘E’，0 ，0）。程序对其他未定义的状态输入有容错性，能够滤去。

1. 输出需求：

程序窗口作为与用户交互的界面，应有对用户进行操作提示，对于到达状态应更新并显示停车场的状态；对于离开停车场（便道）应更新停车场（便道）的状态并显示，对于离开停车场还应显示车辆牌号，到达离开时间以及应缴费用等信息。

1. 测试数据

假设停车场的容纳量为n，设n=2，输入数据为：(‘A’,1,5)，(‘A’,2,10)， (‘D’,1,15)，(‘A’,3,20)，(‘A’,4,25)，(‘A’,5,30)，(‘D’,2,35)， (‘D’,4,40)，(‘E’,0,0)。其中‘A’表示到达（Arrival）；‘D’表示离去（Departure）； ‘E’表示输入结束（End）。

1. **概要设计**

为实现上述功能，需要有一个结构体来存储车辆具体的信息；还需要一个顺序结构栈来作为停车场，以及一个顺序结构栈作为用来临时停放为让路而退出的汽车的空间。需要一个以链表结构队列来作为便道。因此需要定义车辆结构体，停车场栈，链队中汽车结点结构体和便道队列四种结构。此外声明一个停车场类对着四个结构体进行统一管理。

1. Struct car 车辆结构体

数据对象：车牌号—id，到达时间—reachtime，离开时间—lefttime

1. Struct StackCar 停车场栈

数据对象：栈顶-top，一系列car结构体指针-Carstack[]；

1. Struct QCarNode 链队中汽车结构体

数据对象：car类data-车辆信息，结点

1. Struct LinkQueueCar 便道队列

数据对象：QCarNode类的头指针，尾指针

1. CarSystem 类

数据对象：StackCar \*parking\_lot—停车场

StackCar \*leaveLine –临时存储场

LinkQueueCar \*waitQueue –便道

基本操作：

1. CarSystem()：

构造函数：对parking\_lot leaveLine 和waitQueue进行初始化。构造空栈、空队列

1. ~ CarSystem()：

析构函数：释放parking\_lot、 waitQueue的空间，销毁栈和队列

1. Arrival（）

到达模块：模拟车辆到达，若停车场未满，则将车辆信息填写在停车场栈中，若停车场已满，则将车辆信息填写在便道队列中。

1. Leave()

离开模块：模拟停车场车辆离开，若车辆不在停车场尾部，则将该车后部车辆存入临时栈中，删除该车后按原次序返回，再判断便道中有无车辆，若有，模拟便道车辆进入停车场，依次向前。然后进行缴费模块。

1. Charge(car \*p,int time)

缴费模块：给出离开车辆的车牌号，到达离开时间，以及应缴费用。应缴费用=（离开时间-到达时间）\*100

1. ShowInfo（）

显示模块：显示停车场和便道车位情况。

1. Carstack（）

停车场车位显示，图形化界面

1. Carqueue（）

便道车位显示，图形化界面

1. QueueCarLeave（） //补充题内容

模拟便道车辆直接离开

1. main模块：

while（不输入退出字符）

{

指令分辨；

指令实现；

}

1. **详细设计**

**3.1 parking\_lot\_manage.h**

//头文件 完成声明工作

#ifndef parking\_lot\_manage\_h

#define parking\_lot\_manage\_h

#include <iostream>

using namespace std;

#define maxm 2

struct car // 车辆结构体

{

int id;

int reachtime;

int lefttime;

};

struct StackCar //停车场栈

{

int top;

car \*CarStack[maxm + 1];

};

struct QCarNode //链队中的汽车结点结构体

{

car \*data; //汽车信息

QCarNode \*next;

};

struct LinkQueueCar //便道

{

QCarNode \*head; //对头指针

QCarNode \*rear; //队尾指针

};

class CarSystem{

StackCar \*parking\_lot, \*leaveLine; //停车场栈，车辆出停车场时的临时栈

LinkQueueCar \*waitQueue; //便道

public:

CarSystem(); //构造函数

~CarSystem(); //析构函数

void Arrival(); //车辆进站

void Leave(); //车辆出站

void charge(car \*p,int time); //离开车辆缴费相关信息

void ShowInfo(); //显示车位情况

void Carstack(); //停车场中车位状况

void Carqueue(); //便道中车位状况

void QueueCarLeave(); //便道中车辆离开

};

#endif

**3.2 parking\_lot\_manage.cpp**

//CarSystem类函数具体实现

#include <iostream>

#include "parking\_lot\_manage.h"

using namespace std;

void CarSystem::Arrival()

{

car \*car1;

QCarNode \*car2;

int time;

int id;

cin>>id>>time;

car1=new car;

car1->id=id;

car1->reachtime=time;

if(parking\_lot->top<maxm) //停车场未满 车入场

{

parking\_lot->top++;

parking\_lot->CarStack[parking\_lot->top]=car1;

}

else //停车场已满 车入便道

{

car2=new QCarNode;

car2->data=car1;

car2->next=NULL;

waitQueue->rear->next=car2;

waitQueue->rear=car2;

}

}

void CarSystem::Leave()

{

int id,time,item=-1;

cin>>id>>time;

car\*p;

QCarNode\*q;

for(int i=1;i<maxm+1;i++) //检索车辆所在车场位置

if( parking\_lot->CarStack[i]->id==id)

item=i;

if(parking\_lot->top > 0)

{

while (parking\_lot->top>item) //车辆不再车场出口处 后面车辆倒出进入临时栈

{

leaveLine->top++;

leaveLine->CarStack[leaveLine->top] = parking\_lot->CarStack[parking\_lot->top];

parking\_lot->CarStack[parking\_lot->top] = NULL;

parking\_lot->top--;

}

p=parking\_lot->CarStack[parking\_lot->top];

parking\_lot->CarStack[parking\_lot->top] = NULL;

parking\_lot->top--;

//车辆删除

while(leaveLine->top>0) //车辆倒回停车场

{

parking\_lot->top++;

parking\_lot->CarStack[parking\_lot->top]=leaveLine->CarStack[leaveLine->top];

leaveLine->CarStack[leaveLine->top]=NULL;

leaveLine->top--;

}

charge(p,time); //输出收费信息

if(waitQueue->head!=waitQueue->rear&&parking\_lot->top<maxm)

//便道车辆进入

{

parking\_lot->top++;

parking\_lot->CarStack[parking\_lot->top]=waitQueue->head->next->data;

parking\_lot->CarStack[parking\_lot->top]->reachtime=time;

q=waitQueue->head->next;

waitQueue->head->next=q->next;

if(q==waitQueue->rear)

waitQueue->rear=waitQueue->head;

delete q;

}

}

else cout<<"停车场无车";

}

void CarSystem::charge(car \*p, int time)

{

int chargemoney;

p->lefttime=time;

chargemoney=(p->lefttime-p->reachtime)\*100;

cout<<"\n-----------缴费信息----------"<<endl;

cout<<"\n该车的车牌号为"<<p->id<<endl;

cout<<"\n该车的到达时间为"<<p->reachtime<<endl;

cout<<"\n该车的离开时间为"<<p->lefttime<<endl;

cout<<"\n该车应缴纳费用"<<chargemoney<<"元"<<endl;

delete p;

}

CarSystem::CarSystem()

{

parking\_lot=new StackCar;

parking\_lot->top=0;

parking\_lot->CarStack[0]=NULL;

leaveLine=new StackCar;

leaveLine->top=0;

leaveLine->CarStack[0]=NULL;

waitQueue=new LinkQueueCar;

waitQueue->head=new QCarNode;

waitQueue->rear=waitQueue->head;

}

CarSystem::~CarSystem()

{

QCarNode \*p;

int rec;

while(parking\_lot->top!=0)

{

rec=parking\_lot->top;

--parking\_lot->top;

delete parking\_lot->CarStack[rec];

};

while (waitQueue->head!= waitQueue->rear)

{

p = waitQueue->head;

waitQueue->head = waitQueue->head->next;

delete p;

}

}

void CarSystem::Carstack() //车场车位显示

{

int i;

cout<<endl<<"----------停车场信息更新-------------";

if (parking\_lot->top>0)

{

cout << "\n车位号\t" << " 到达时间 \t" << "车牌号\n";

for (i = 1; i <= parking\_lot->top; i++)

{

cout << " " << i << " \t";

cout <<parking\_lot->CarStack[i]->reachtime<<" \t"<< parking\_lot->CarStack[i]->id;

cout<<endl;

}

}

else cout<<"\n车场无车";

cout<<"\n------------------------------------\n";

}

void CarSystem::Carqueue() //便道车位显示

{

QCarNode \*p;

p=waitQueue->head->next;

cout<<endl<<"------------便道信息更新-------------"<<endl;

if(waitQueue->head!=waitQueue->rear)

{

cout<<"队头（显示车牌号）"<<endl;

while(p!=NULL)

{

cout<<p->data->id<<endl;

p=p->next;

}

cout<<"队尾";

}

else cout<<"便道中无车";

cout<<"\n------------------------------------\n";

}

void CarSystem::QueueCarLeave() //车辆离开车道

{

int id;

cin>>id;

QCarNode \*p,\*q;

if(waitQueue->head!=waitQueue->rear)

{

p=waitQueue->head->next;

q=waitQueue->head;

while(p->data->id!=id) //检索对应车牌号的车

{

q = p;

p = p->next;

}

q->next=p->next;

cout<<"车牌号为"<<p->data->id<<"的车辆已离开便道"<<endl;

delete p;

}

else cout<<"便道中无车"<<endl;

}

void CarSystem::ShowInfo()

{

char str;

cout << "请输入查看列表:" << endl;

cout << "--- 1.停车场车位状况-->请按 p ---" << endl;

cout << "--- 2.便道停车位状况-->请按 q ---" << endl;

cout << "--- 3.返回主菜单 -->请按 b ---" << endl;

cin>>str;

switch(str){

case 'p':Carstack();break;

case 'q':Carqueue();break;

case 'b':break;

default:break;

}

}

**3.3 main.cpp**

#include <iostream>

#include "parking\_lot\_manage.h"

using namespace std;

void showMenu()；

int main()

{

char command='A';

showMenu()；

CarSystem CAR;

while (command!='E')

{

cin>>command;

switch (command){

case 'A':

CAR.Arrival();

CAR.Carstack();

break;

case 'D':

CAR.Leave();

CAR.Carstack();

break;

case 'C':

CAR.ShowInfo();

break;

case 'Q':

CAR.QueueCarLeave();

CAR.Carqueue();

break;

case 'E':break;

default: cout<<"输入不合乎规范";

}

}

return 0;

}

void showMenu()

{

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

cout << "\*\*\*\*\*\* 停 车 场 管 理 系 统 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* A:车辆到达车场，请输入车牌号和到达时间 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* D:车辆离开车场，请输入车牌号和离开时间 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* Q:车辆离开便道，请输入车牌号 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* C:查询功能 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* E:退出程序 \*\*\*\*\*\*" << endl;

cout << "\n\t\n\t\t请选择：";

}

1. **调试分析**
2. 算法复杂度分析
3. 时间复杂度

O（n）：Leave（），~CarSystem（），Carstack（），Carqueue（），

QueueCarLeave（）,main();

O（1）：CarSystem（），ShowInfo（），Charge(),Arrival();

算法复杂度为O(n)的函数中部分函数由于需要遍历或查找parking\_lot或leaveLine的元素，main函数实行遍历主要是为了完善功能。

1. 空间复杂度 O（n）

构造函数中提前声明若干指针。

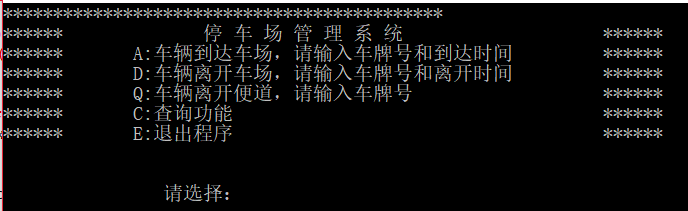
2．一些函数传参可以优化，该程序完全使用类的公有成员函数，对停车场类包装不成熟，导致数据安全性没有保证。

1. **用户手册**
2. 本程序使用的 Code::Blocks 16.01 IDE，程序以项目（project）方式组织，





1. 依次点击菜单Build->build and run；可见程序菜单，依照菜单格式键入便可完成系列操作。



1. **测试结果**

执行(‘A’,1,5)：构建停车场车辆 id=1，reachtime=5，显示停车场信息；

执行（‘A’,2,10）：构建停车场车辆 id=2，reachtime=10，显示停车场信息；

执行（‘D’,1,15）：车辆1离开，显示缴费信息，显示更新停车场的信息；

执行(‘A’,3,20)：构建停车场车辆 id=3，reachtime=20，显示停车场信息；

执行(‘A’,4,25)：构建便道车辆 id=4，reachtime=25，显示停车场信息；

执行(‘A’,5,30)：构建便道车辆 id=5，reachtime=30，显示停车场信息；

执行(‘D’,2,35)： 车辆2离开，显示缴费信息，车辆4进场，reachtime=35，显示停车场信息。

执行(‘D’,4,40)：车辆4离开，显示缴费信息，车辆5进场，reachtime=40，显示停车场信息。

执行(‘E’,0,0)：程序结束

1. **附录**

源程序文件名清单：

main.cpp //主程序和目录函数实现

parking\_lot\_manage.h //头文件

parking\_lot\_manage.cpp//头文件函数实现