

**课程: 算法与数据结构**

**停车场管理系统课程设计**

**学院 信息工程学院**

**专业 计算机科学与技术**

**班级 信2121-1**

**姓名 张瀛煜，沈召才，王宁**

**组号 第15组**

**学号 202105721124, 1128,1117**

**任课教师 杨昌玉**

**学年学期 2021-2022-1学期**

2021 年 12 月 23 日

**任 务 书**

|  |
| --- |
| 一、设计目的、内容及要求  1、设计目的  设停车场内只有一个可停放n辆汽车的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列，若停车场内已停满n辆汽车，则后来的汽车只能在门外的便道上等候，一旦有车开走，则排在便道上的第一辆车即可开入；当停车场内某辆车离开时，在其之后开入的车辆必须先退出停车场让路，待该车辆开出大门外，其他车辆再按照原次序进入停车场，每辆停放在停车场的在其离开停车场时必须按其停留的时间长短缴纳费用。  2、设计内容及要求  运用数据结构知识，要求以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入的输入数据的方式进行模拟管理。输入1，表示车辆到达；输入2，表示车辆离开；输入3，表示显示出停车场内及便道上的停车情况；输入4，表示退出系统。车辆到达操作，需输入汽车牌照号码及到达时刻；车辆离开操作，需输入汽车在停车场的位置及离开时刻，且应输出汽车在停车场内停留的时间和应缴纳的费用，要求处理的数据元素包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻； |
| 二、预期的设计成果  记录车辆到达的车牌号、到达时间、车辆停放位置、车辆的驶离时间、驶离位置、车牌号，并根据到达时间和驶离时间计算停车费用。同时也根据提示显示便道信息。使用栈和队列两种数据结构实现了模拟停车场的管理，其中为了模拟车辆的收费等信息，定义车辆类型。 |
| 三、进程安排  12月20日上午：查阅资料，确定分组、扮演角色和选题；  12月20日下午：开始文档，代码的编写；  12月21日全天：根据不同的角色，展开讨论，分工完成自己负责的任务；  12月23日上午：答辩验收。 |

**目 录**

[一、任务分配 1](#_Toc91072278)

[二、简介 1](#_Toc91072279)

[三、算法说明 2](#_Toc91072280)

[四、测试结果 4](#_Toc91072285)

[五、分析与探讨 12](#_Toc91072287)

[附录 源代码 13](#_Toc91072288)

# 一、任务分配

·程序员：张瀛煜。主要任务：负责算法的设计，并完成源代码的编写。

·测试员：沈召才。主要任务：负责设计测试用程序，并对实验结果进行整理分析，最后完成实验报告的第四、第五部分内容，即测试结果与分析探讨部分。

·文档员：王宁。主要任务：负责撰写实验报告的第一、第二、第三部分内容，即实验内容简介与算法描述。同时完成整个文档的整合，使整篇报告排版、文字风格统一。

# 二、简介

2.1问题描述

设停车场内只有一个可停放n辆汽车的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列，若停车场内已停满n辆汽车，则后来的汽车只能在门外的便道上等候，一旦有车开走，则排在便道上的第一辆车即可开入；当停车场内某辆车离开时，在其之后开入的车辆必须先退出停车场让路，待该车辆开出大门外，其他车辆再按照原次序进入停车场，每辆停放在停车场的在其离开停车场时必须按其停留的时间长短缴纳费用。

2.2 设计要求

运用数据结构知识，要求以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入的输入数据的方式进行模拟管理。输入1，表示车辆到达；输入2，表示车辆离开；输入3，表示显示出停车场内及便道上的停车情况；输入4，表示退出系统。车辆到达操作，需输入汽车牌照号码及到达时刻；车辆离开操作，需输入汽车在停车场的位置及离开时刻，且应输出汽车在停车场内停留的时间和应缴纳的费用，要求处理的数据元素包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻；

2.3 课程设计的任务

该系统主要实现以下几个功能：

1.车辆到达信息登记：包括车牌号，到达时间，在停车场停放的位置等；

2.车辆离开记录：包括离开时间，在停车场停留时间，停车费用等；

3.信息显示：显示停车场内和便到内的汽车到达时机急位置等基本信息；

2.4 课程设计的思想

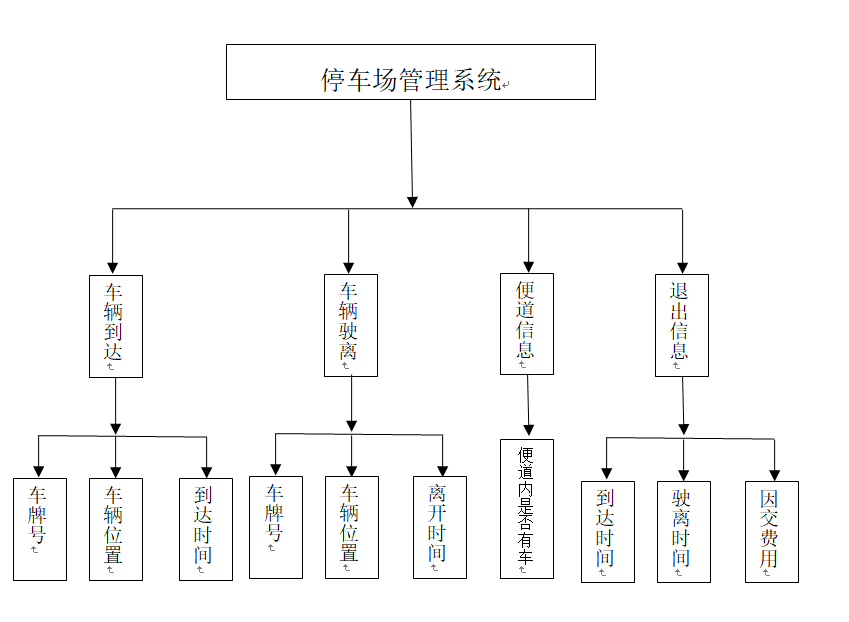
该程序应用栈来完成的车辆进停车场和出停车场就是一个数据进栈和出栈的过程。如果停车场（栈）满了，就将车辆放在便道（队列）里；在车辆离开时计算出使用费用，并将便道第一辆车停到该车位置。

3.数据结构

本设计使用的数据结构是顺序栈和链式队列

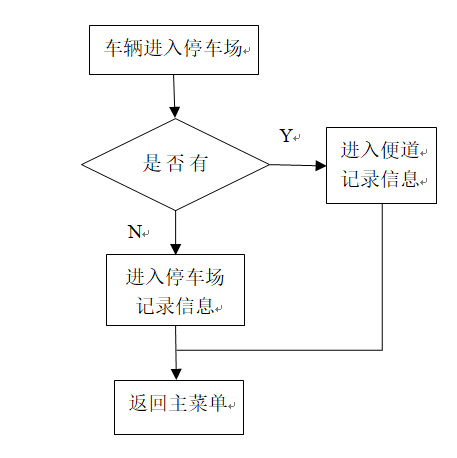
# 三、算法说明

3.1、系统总体设计



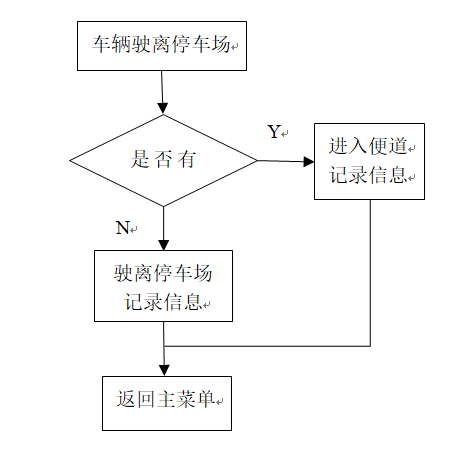
3.2、流程图及系统测试

3.2.1、进入停车场



模拟停车场车辆进出时需要输入车辆的信息，包括车牌号码及进入与离开时刻，因此可以定义一个时间节点类型和一个车辆信息节点类型，在顺序栈及链式队列中定义节点类型为车辆信息节点类型。

3.2.2、驶离停车场



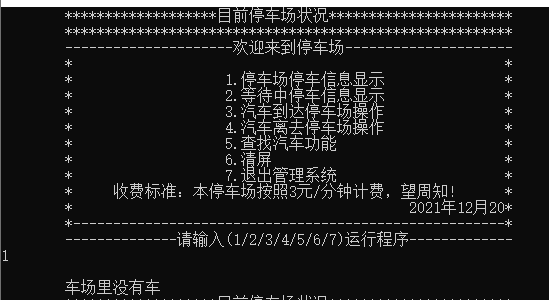
车辆的离开，则需另一个栈，要给离去的汽车让路而从停车场退出来的汽车临时停放，也用顺序栈实现，车辆离开后需要检查便道内是否有车辆等待，若有等待车辆则进行便道内的车辆进入停车场的操作，即将车辆信息节点进入栈操作，输入当前时间后开始计费，最后进行出队操作，表示车辆离开便道已进入停车场。

# 四、测试结果

4.1、测试比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试编号 | 测试概述 | 测试步骤 | 测试结果 | 错误原因 |
| 1 | 停车场停车信息显示 | 先输入3，汽车到达停车场，在输入1查看停车信息 | 位置 到达时间 车牌号 | 无 |
| 2 | 等待中停车信息显示 | 先输入3，汽车到达停车场，在重复执行，直到停车场已满，在输入2查看等待停车信息 | 第1辆等待的车辆 | 无 |
| 3 | 汽车到达停车场操作 | 直接输入3 | 输入车牌号(例：鲁A11111): | 无 |
| 4 | 汽车离去停车场操作 | 直接输入4 | 请输入车在车场的位置 | 无 |
| 5 | 查找汽车功能 | 直接输入5 | 请输入要搜索的车牌号： | 无 |

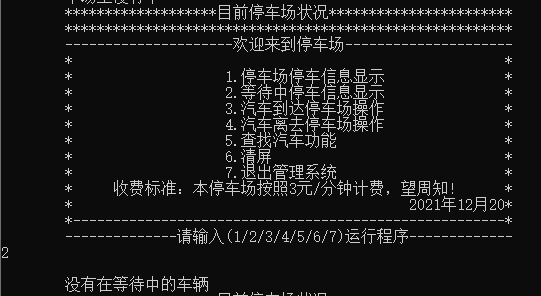
测试输入 1



正确输出 : 位置 到达时间 车牌号

实际输出 : 车场里没有车

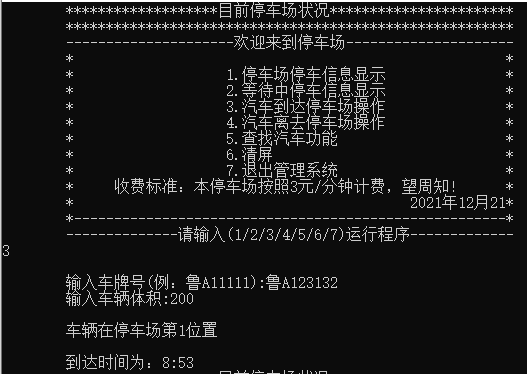
测试输入 2



正确输出 : 第1辆等待的车辆

实际输出 : 车场里没有车

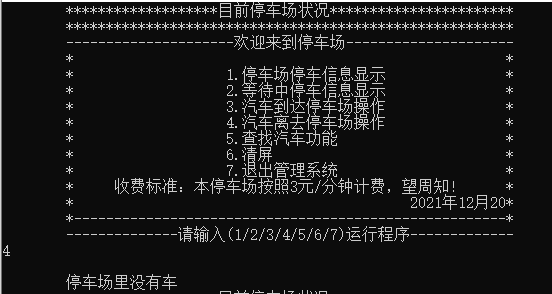
测试输入 3



正确输出 : 输入车牌号(例：鲁A11111)： 输入车辆体积：

实际输出 : 输入车牌号(例：鲁A11111): 输入车辆体积：

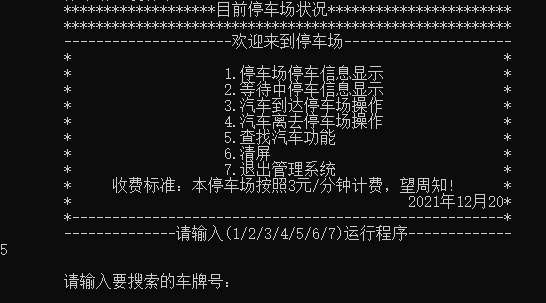
测试输入 4



正确输出 : 请输入车在车场的位置

实际输出 : 停车场里没有车

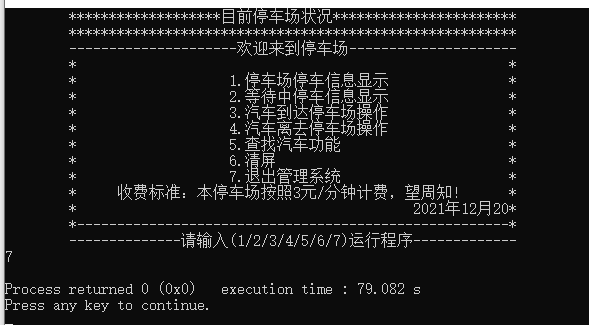
测试输入 5



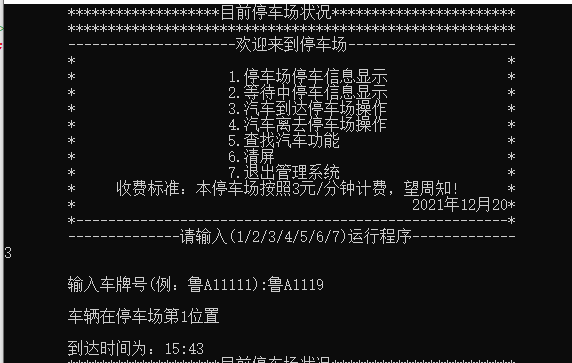
正确输出 : 请输入要搜索的车牌号：

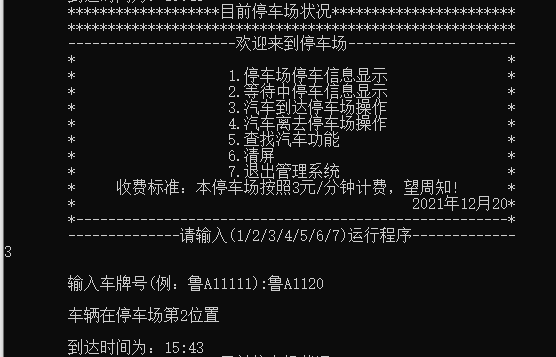
实际输出 : 请输入要搜索的车牌号：

测试输入 7



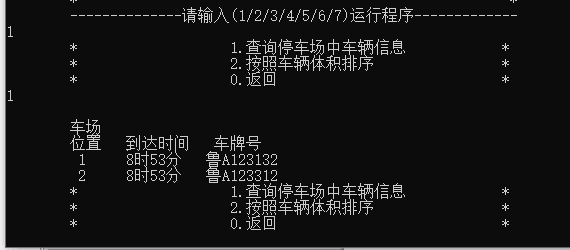
测试车辆进去停车场 1



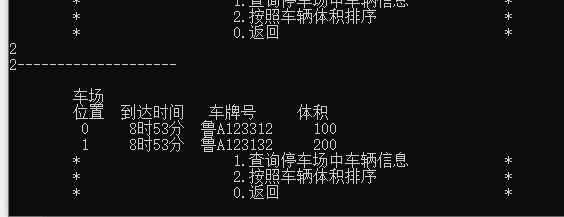




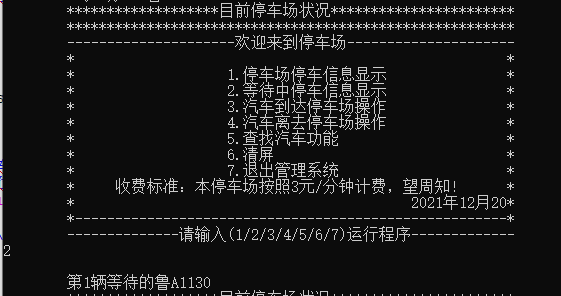
测试停车场停车信息显示



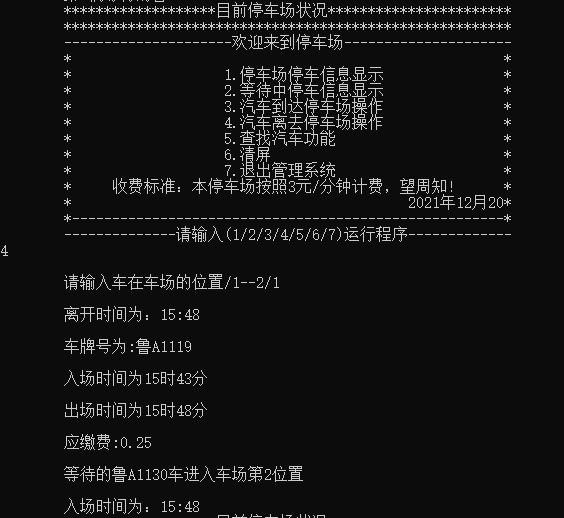
按照体积排序

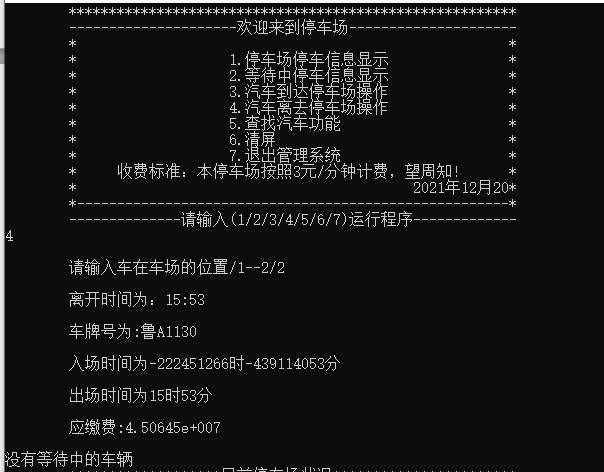


测试等待中停车信息显示

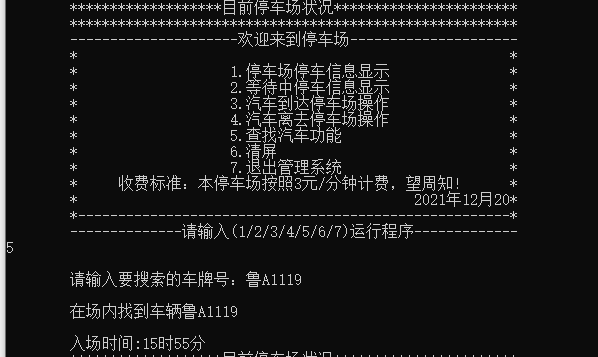


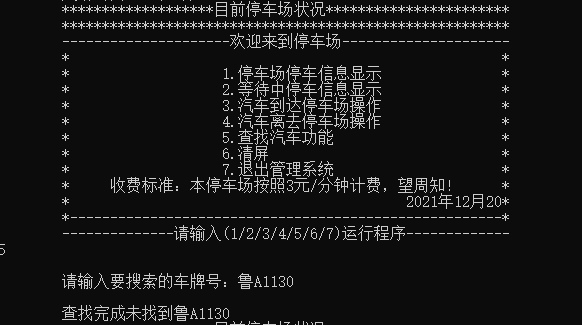
测试车辆驶出停车场





测试查找汽车功能





需要注意的是，测试员的任务不是提供几组简单的数据让程序员容易通过，从而宣称该程序是正确的，而应该是千方百计设计“刁难”的数据，想办法让所测试的程序暴露出问题，这样才能真正帮助程序员完成正确的程序。

# 五、分析与探讨

主要使用了栈和队列两种数据结构，实现了模拟停车场的管理，其中为了模拟车辆的查询和收费定义了车辆信息结点类型。车辆的进入与离开用到了栈和队列的基本操作。

本程序中主要有以下函数：Arrival是车辆进入停车场的函数，主要运用了入栈和入队的操作；Leavel是车辆离开的函数，包含了出栈和出队的操作；showWaitCar和showInPark和findCarInfo主要运用了队列的循环。

通过这段时间的课程设计，我们组的各个成员对计算机的应用，数据结构的应用以及C++语言的使用都有了更深的了解。尤其是C++语言的进步让我深刻的感受到任何所学的知识都需要实践，没有实践就无法真正理解这些知识以及掌握它们。在理论学习和上机实践的各个环节中，通过自主学习和请教老师和同学，我收获了很多。当然也遇到不少的问题，也正是因为这些问题引发的思考给我带了收获。从当初不喜欢上机写程序到现在能主动写程序，从当初拿着程序不只如何下手到现在知道如何分析问题，如何用专业知识解决实际问题的转变，我发现无论是专业知识还是动手能力，自己都有很大程度的提高。在这段时间里，我对栈和队列等使用更加熟悉，头文件不能顶格，逐渐形成了较好的编程习惯。在杨老师的指导帮助下，同学们课余时间的讨论中，这些问题都一一得到了解决。在程序的调试能力上，无形中得到了许多的提高。例如：头文件的使用，变量和数组的范围问题，定义变量时出现的问题等等。  
 回顾此次停车场系统课程设计，这几天我仍感触很深，从选题到定稿，从理论到实践，从程序员，测试员和文档员的共同商量与互帮互助下，在整整一星期的日子里，我们终于完成了这项课程设计，但是可以说是学到很多很多的东西。无论是小组内的共同合作还是以后在工作中的经验，不仅可以巩固了以前所学过的知识，而且学到了很多在书本上所没有学到过的知识。通过这次课程设计之后，一定要把以前所学过的知识重新温故，在以后会有更多的课程设计，努力为了将来打好基础，机会是留给有准备的人的，况且现在计算机发展迅速，作为IT中的初学者，一定学好并继承前人的精神。最后杨老师很认真的对待我们的答辩，这是一种很负责任的表现。让我们真正的了解和掌握一些基本知识，在以后的工作中做到游刃有余。这次课程设计终于顺利完成了，在设计中遇到了很多编程问题，最后在杨老师的辛勤指导下，终于迎刃而解。同时，在老师的身上我们都学得到很多实用的知识，在次我们衷心的表示感谢！

# 附录 源代码

1. #include <iostream>
2. #include <malloc.h>
3. #include <stdio.h>
4. #include <string>
5. #include <string.h>
6. #include <list>
7. #include <windows.h>
8. #include<algorithm>
9. using namespace std;
10. */\*\*停车场容量\*/*
11. #define MAXNUM 3
12. #define OK 1
13. #define ERROR 0
14. SYSTEMTIME sys;
15. double price = 0.05;
16. */\*\*时间\*/*
17. typedef struct time
18. {
19. int hour;
20. int min;
21. } Time;
22. */\*\*车辆信息\*/*
23. typedef struct node
24. {
25. char car\_num[10];
26. double volume;
27. Time inTime;
28. Time outTime;
29. } CarNode;
30. */\*\*停车场内车辆信息\*/*
31. typedef struct node2
32. {
33. CarNode \*stack[MAXNUM + 1];
34. int top;
35. } SeqStackCar;
36. typedef struct car
37. {
38. CarNode \*data;    *//车辆信息*
39. struct car \*next; *//指向下一个的指针域*
40. } QueueNode;
41. typedef struct Node
42. {
43. QueueNode \*head; *//头车*
44. QueueNode \*rear; *//尾车*
45. } LinkQueueCar;
46. void StackInit(SeqStackCar \*seq)
47. {
48. seq->top = 0;
49. for (int i = 0; i <= MAXNUM; i++)
50. {
51. seq->stack[seq->top] = NULL;
52. }
53. }
54. int QueueInit(LinkQueueCar \*q)
55. {
56. q->head = (QueueNode \*)malloc(sizeof(QueueNode));
57. if (q->head != NULL)
58. {
59. q->head->next = NULL;
60. q->rear = q->head;
61. return OK;
62. }
63. else
64. {
65. return ERROR;
66. }
67. }
68. void Print(CarNode \*p, int room)
69. {
70. int in\_h, in\_m, out\_h, out\_m;
71. in\_h = p->inTime.hour;
72. in\_m = p->inTime.min;
73. GetLocalTime( &sys );
74. cout << "\n\t离开时间为：" << sys.wHour << ":" <<sys.wMinute << endl;
75. p->outTime.hour = sys.wHour;
76. p->outTime.min = sys.wMinute;
77. cout << "\n\t车牌号为:" << p->car\_num << endl;
78. cout << "\n\t入场时间为" << p->inTime.hour << "时" << p->inTime.min << "分\n";
79. cout << "\n\t出场时间为" << p->outTime.hour << "时" << p->outTime.min << "分\n";
80. out\_h = p->outTime.hour;
81. out\_m = p->outTime.min;
82. cout << "\n\t应缴费:" << ((out\_h - in\_h) \* 60 + (out\_m - in\_m)) \* price << endl;
83. free(p);
84. }
85. bool matchCarNum(char \*incar)
86. {
87. string firstlocation = "京津沪渝冀豫云辽黑湘皖鲁新苏浙赣鄂桂甘晋蒙陕吉闽贵粤青藏川宁琼使领";
88. string regional = "QWERTYUIOPASDFGHJKLZXCVBNM";
89. string a = incar;
90. if (firstlocation.find(a.substr(0, 2)) <= firstlocation.length() && regional.find(incar[2]) <= firstlocation.length())
91. {
92. return true;
93. }
94. else
95. {
96. cout << "\t车牌号错误" << endl;
97. return false;
98. }
99. }
100. int Arrival(SeqStackCar \*enter, LinkQueueCar \*q)
101. {
102. CarNode \*p;
103. QueueNode \*t;
104. p = (CarNode \*)malloc(sizeof(CarNode));
105. memset(p->car\_num, 0, 10);
106. while (true)
107. {
108. cout << "\n\t输入车牌号(例：鲁A11111):";
109. cin >> p->car\_num;
110. if (matchCarNum(p->car\_num))
111. break;
112. }
113. cout << "\t输入车辆体积:";
114. cin >> p->volume;
115. if (enter->top < MAXNUM)
116. {
117. enter->top++;
118. cout << "\n\t车辆在停车场第" << enter->top << "位置\n";
119. GetLocalTime( &sys );
120. cout << "\n\t到达时间为：" << sys.wHour << ":" << sys.wMinute << endl;
121. p->inTime.hour = sys.wHour;
122. p->inTime.min = sys.wMinute;
123. enter->stack[enter->top] = p;
124. return OK;
125. }
126. else
127. {
128. cout << "\n\t车场已满需要等待!\n";
129. t = (QueueNode \*)malloc(sizeof(QueueNode));
130. t->data = p;
131. t->next = NULL;
132. q->rear->next = t;
133. q->rear = t;
134. return OK;
135. }
136. }
137. void Leave(SeqStackCar \*enter, SeqStackCar \*temp, LinkQueueCar \*wite)
138. {
139. int room;
140. CarNode \*p, \*t;
141. QueueNode \*q;
142. *//判断是否有车*
143. if (enter->top > 0)
144. {
145. while (true)
146. {
147. cout << "\n\t请输入车在车场的位置/1--" << enter->top << "/";
148. cin >> room;
149. if (room >= 1 && room <= enter->top)
150. {
151. break;
152. }
153. }
154. while (enter->top > room)
155. {
156. temp->top++;
157. temp->stack[temp->top] = enter->stack[enter->top];
158. enter->stack[enter->top] = NULL;
159. enter->top--;
160. }
161. p = enter->stack[enter->top];
162. enter->stack[enter->top] = NULL;
163. enter->top--;
164. while (temp->top >= 1)
165. {
166. enter->top++;
167. enter->stack[enter->top] = temp->stack[temp->top];
168. enter->stack[temp->top] = NULL;
169. temp->top--;
170. }
171. Print(p, room);
172. if ((wite->head != wite->rear) && enter->top < MAXNUM)
173. {
174. q = wite->head->next;
175. t = q->data;
176. enter->top++;
177. cout << "\n\t等待的" << (t->car\_num) << "车进入车场第" << (enter->top) << "位置\n";
178. GetLocalTime(&sys);
179. cout << "\n\t入场时间为：" << sys.wHour << ":" << sys.wMinute << endl;
180. p->inTime.hour = sys.wHour;
181. p->inTime.min = sys.wMinute;
182. wite->head->next = q->next;
183. if (q == wite->rear)
184. {
185. wite->rear = wite->head;
186. }
187. enter->stack[enter->top] = t;
188. free(q);
189. }
190. else
191. {
192. cout << "\n没有等待中的车辆\n";
193. }
194. }
195. else
196. {
197. cout << "\n\t停车场里没有车\n";
198. }
199. }
200. int showWaitCar(LinkQueueCar \*wite)
201. {
202. if ((wite->head != wite->rear))
203. {
204. CarNode \*t;
205. QueueNode \*q = wite->head->next;
206. int i = 1;
207. while (q != NULL)
208. {
209. t = q->data;
210. cout << "\n\t第" << i << "辆等待的" << (t->car\_num) << endl;
211. q = q->next;
212. i++;
213. }
214. return OK;
215. }
216. else
217. {
218. cout << "\n\t没有在等待中的车辆\n";
219. return OK;
220. }
221. }
222. void showinPark(SeqStackCar \*S)
223. {
224. if (S->top > 0)
225. {
226. cout << "\n\t车场";
227. cout << "\n\t位置   到达时间   车牌号\n";
228. for (int i = 1; i <= S->top; i++)
229. {
230. cout << "\t " << i;
231. cout << "     " << S->stack[i]->inTime.hour << "时" << S->stack[i]->inTime.min << "分";
232. cout << "   " << S->stack[i]->car\_num << endl;
233. }
234. }
235. else
236. {
237. cout << "\n\t车场里没有车" << endl;
238. }
239. }
240. int findCarInfo(SeqStackCar \*enter, LinkQueueCar \*wite)
241. {
242. cout << "\n\t请输入要搜索的车牌号：";
243. char findcarnum[10];
244. cin >> findcarnum;
245. for (int i = 1; i <= enter->top; i++)
246. {
247. int result = strcmp(findcarnum, enter->stack[i]->car\_num);
248. if (result == 0)
249. {
250. cout << "\n\t在场内找到车辆" << findcarnum << endl;
251. cout << "\n\t入场时间:" << enter->stack[i]->inTime.hour << "时" << enter->stack[i]->inTime.min << "分\n";
252. return OK;
253. }
254. }
255. if ((wite->head != wite->rear))
256. {
257. CarNode \*t;
258. QueueNode \*q = wite->head->next;
259. int i = 1;
260. while (q != NULL)
261. {
262. t = q->data;
263. int result = strcmp(findcarnum, t->car\_num);
264. if (result == 0)
265. {
266. cout << "\n\t在等待队列中找到车辆" << findcarnum << endl;
267. return OK;
268. }
269. q = q->next;
270. i++;
271. }
272. }
273. cout << "\n\t查找完成未找到" << findcarnum << endl;
274. return ERROR;
275. }
276. bool cmp(CarNode a,CarNode b)
277. {
278. if  (a.volume<b.volume)
279. {
280. return true;
281. }
282. else
283. {
284. if(a.volume==b.volume)
285. {
286. if  (a.volume <b.volume)
287. return true;
288. }
289. }
290. return false;
291. }
292. void orderByVolume(SeqStackCar \*S)
293. {
294. CarNode arrCar[S->top];
295. for (int i = 1; i <= S->top; i++)
296. {
297. CarNode carnode;
298. strcpy(carnode.car\_num, S->stack[i]->car\_num);
299. carnode.volume = S->stack[i]->volume;
300. carnode.inTime = S->stack[i]->inTime;
301. carnode.outTime = S->stack[i]->outTime;
302. arrCar[i-1] = carnode;
303. }
304. cout << "\n\t车场";
305. cout << "\n\t位置  到达时间   车牌号     体积\n";
306. sort(arrCar+0,arrCar+S->top,cmp);
307. for (int i = 0; i < S->top; i++)
308. {
309. CarNode  c = arrCar[i];
310. cout << "\t " << i;
311. cout << "     " <<c.inTime.hour << "时" << c.inTime.min << "分";
312. cout << "  " << c.car\_num;
313. cout << "     " << c.volume << endl;
314. }
315. }
316. void Welcome()
317. {
318. GetLocalTime(&sys);
319. cout << ("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*目前停车场状况\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
320. cout << ("\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
321. cout << ("\t---------------------欢迎来到停车场---------------------\n");
322. cout << ("\t\*                                                      \*\n");
323. cout << ("\t\*                   1.停车场停车信息显示               \*\n");
324. cout << ("\t\*                   2.等待中停车信息显示               \*\n");
325. cout << ("\t\*                   3.汽车到达停车场操作               \*\n");
326. cout << ("\t\*                   4.汽车离去停车场操作               \*\n");
327. cout << ("\t\*                   5.查找汽车功能                     \*\n");
328. cout << ("\t\*                   6.清屏                             \*\n");
329. cout << ("\t\*                   7.退出管理系统                     \*\n");
330. cout << "\t\*     收费标准：本停车场按照" << (60 \* price) << "元/分钟计费，望周知!      \*\n";
331. cout << "\t\*                                          " << sys.wYear << "年" << sys.wMonth << "月" << sys.wDay << "\*\n";
332. cout << "\t\*------------------------------------------------------\*\n";
333. cout << "\t--------------请输入(1/2/3/4/5/6/7)运行程序-------------\n";
334. }
335. void showFrom2(SeqStackCar \*S)
336. {
337. bool flage = true;
338. int choose;
339. while (flage)
340. {
341. cout << ("\t\*                   1.查询停车场中车辆信息            \*\n");
342. cout << ("\t\*                   2.按照车辆体积排序                \*\n");
343. cout << ("\t\*                   0.返回                            \*\n");
344. while (true)
345. {
346. cin >> choose;
347. if (choose >= 0 && choose <= 2)
348. {
349. break;
350. }
351. else
352. {
353. cout << "\t--------------请输入(1/2/0/)运行程序---------------\n";
354. }
355. }
356. switch (choose)
357. {
358. case 1:
359. showinPark(S);
360. break;
361. case 2:
362. orderByVolume(S);
363. break;
364. case 0:
365. flage = false;
366. break;
367. default:
368. break;
369. }
370. }
371. }
372. bool flage = true;
373. int main()
374. {
375. SeqStackCar enter, temp;
376. LinkQueueCar wait;
377. StackInit(&enter);
378. StackInit(&temp);
379. QueueInit(&wait);
380. int choose;
381. while (flage)
382. {
383. Welcome();
384. cin >> choose;
385. if (choose >= 1 && choose <= 7)
386. {
387. flage = true;
388. switch (choose)
389. {
390. */\*\* 1.停车场停车信息显示\*/*
391. case 1:
392. showFrom2(&enter);
393. break;
394. */\*\*2.等待中停车信息显示 \*/*
395. case 2:
396. showWaitCar(&wait);
397. break;
398. */\*\*3.汽车到达停车场操作\*/*
399. case 3:
400. Arrival(&enter, &wait);
401. break;
402. */\*\*4.汽车离去停车场操作\*/*
403. case 4:
404. Leave(&enter, &temp, &wait);
405. break;
406. */\*\*5.查找汽车功能 \*/*
407. case 5:
408. findCarInfo(&enter, &wait);
409. break;
410. */\*\*6.清屏 \*/*
411. case 6:
412. system("cls");
413. break;
414. */\*\*7.退出系统 \*/*
415. case 7:
416. exit(0);
417. break;
418. default:
419. break;
420. }
421. }
422. else
423. {
424. cout << "\t--------------输入错请重新运行---------------\n";
425. flage = false;
426. return main();
427. }
428. }
429. return 0;
430. }