

停车场管理

专业：计算机科学与技术 班级：23052322 姓名：张逸轩 学号：23051214

1. 实验目的

实际实现栈和队列，并且用于模拟。增加对栈和队列的认知。

2. 实验过程(实验方案、流程、程序等)(参考书上的格式需要写详细)

使用 c 语言，先设计栈和队列这两个数据结构如下：

```
1  typedef struct {
2      int *data;
3      int front;
4      int rear;
5      int size;
6  }tQueue;
7
8  tQueue *create_queue(int size);
9  void push_queue(tQueue *q, int data);
10 int pop_queue(tQueue *q);
11 bool is_empty_queue(tQueue *q);
12 bool is_full_queue(tQueue *q);
13 void free_queue(tQueue *q);
```



```
1  typedef struct {
2      int *data;
3      int top; //top means used size
4      int size;
5  } tStack;
6
7  tStack *create_stack(int size);
8  void push_stack(tStack *s, int data);
9  int pop_stack(tStack *s);
10 bool is_empty_stack(tStack *s);
11 bool is_full_stack(tStack *s);
12 void free_stack(tStack *s);
```

实现代码如下：



```
1  #include "queue.h"
2
3  tQueue* create_queue(int size)
4  {
5      tQueue* queue = (tQueue*)malloc(sizeof(tQueue));
6      queue->data = (int*)malloc(sizeof(int) * size);
7      queue->front = 0;
8      queue->rear = 0;
9      queue->size = size;
10     return queue;
11 }
12
13 bool is_full_queue(tQueue* queue)
14 {
15     return (queue->rear + 1) % queue->size == queue->front;
16 }
17
18 bool is_empty_queue(tQueue* queue)
19 {
20     return queue->front == queue->rear;
21 }
22
23 void push_queue(tQueue* queue, int data)
24 {
25     if (is_full_queue(queue))
26     {
27         printf("Queue is full\n");
28         return;
29     }
30     queue->data[queue->rear] = data;
31     queue->rear = (queue->rear + 1) % queue->size;
32 }
33
34 int pop_queue(tQueue* queue)
35 {
36     if (is_empty_queue(queue))
37     {
38         printf("Queue is empty\n");
39         return -1;
40     }
41     int data = queue->data[queue->front];
42     queue->front = (queue->front + 1) % queue->size;
43     return data;
44 }
45
46 void free_queue(tQueue* queue)
47 {
48     free(queue->data);
49     free(queue);
50 }
```



```
1  #include "stack.h"
2
3  tStack* create_stack(int size)
4  {
5      tStack *s = (tStack *)malloc(sizeof(tStack));
6      s->data = (int *)malloc(size * sizeof(int));
7      s->top = -1;
8      s->size = size;
9      return s;
10 }
11
12 void push_stack(tStack *s, int data)
13 {
14     if (s->top == s->size - 1)
15     {
16         printf("tStack is full\n");
17         return;
18     }
19     s->top++;
20     s->data[s->top] = data;
21 }
22
23 int pop_stack(tStack *s)
24 {
25     if (s->top == -1)
26     {
27         printf("tStack is empty\n");
28         return -1;
29     }
30     int data = s->data[s->top];
31     s->top--;
32     return data;
33 }
34
35 bool is_empty_stack(tStack *s)
36 {
37     return s->top == -1;
38 }
39
40 bool is_full_stack(tStack *s)
41 {
42     return s->top == s->size - 1;
43 }
44
45 void free_stack(tStack *s)
```

然后参考题目意思，进行两种模拟：

```
1 void car_arrive(int id, int car_time)
2 {
3     if(is_full_stack(ins))
4     {
5         push_queue(waits, id);
6     }
7     else
8     {
9         push_stack(ins, id);
10        arrive_time[id] = car_time;
11    }
12    printf("id: %d arrive: %d\n", id, car_time);
13 }
14
15 void car_departure(int id, int car_time)
16 {
17     int now_id = -1;
18     while(now_id != id)
19     {
20         now_id = pop_stack(ins);
21         //printf("now_id: %d\n", now_id);
22         if(now_id != id)
23         {
24             push_stack(outs, now_id);
25         }
26     }
27     leave_time[id] = car_time;
28     while(!is_empty_stack(outs))
29     {
30         now_id = pop_stack(outs);
31         push_stack(ins, now_id);
32     }
33     while(!is_empty_queue(waits) && !is_full_stack(ins))
34     {
35         now_id = pop_queue(waits);
36         push_stack(ins, now_id);
37         arrive_time[now_id] = car_time;
38     }
39     printf("id: %d departure: %d\n", id, car_time);
40 }
```

最后统计结果：

```

1 void calc()
2 {
3     for(int i = 0; i < CARS_NUM; i++)
4     {
5         if(arrive_time[i] != -1)
6         {
7             if(leave_time[i] != -1)
8             {
9                 printf("id: %d arrive: %d departure: %d stay: %d\n", i, arrive_time[i], leave_time[i], leave_time[i] - arrive_time[i]);
10            }
11            else
12            {
13                printf("id: %d arrive: %d departure: NONE stay: still staying\n", i, arrive_time[i]);
14            }
15        }
16    }
17 }

```

实际项目结构如图：

```

hewo@hewo-thinkpad ~/CS/hdu/hdu-data-structure/lab1 } main tree
.
├── input
├── lab1
├── lab1.docx
├── Makefile
├── readme.md
├── src
│   ├── common.h
│   ├── main.c
│   ├── queue
│   │   ├── queue.c
│   │   └── queue.h
│   └── stack
│       ├── stack.c
│       └── stack.h

```

4 directories, 11 files

其中 input 提供数据，lab1 为可执行程序。

main.c 如下：

```

1  #include "common.h"
2  #include "stack/stack.h"
3  #include "queue/queue.h"
4
5  #define CARS_NUM 1005
6
7  int n;
8  tStack *ins;
9  tStack *outs;
10 tQueue *waits;
11
12 int arrive_time[1005];
13 int leave_time[1005];
14
15 void init()
16 {
17     ins = create_stack(n);
18     outs = create_stack(n);
19     waits = create_queue(CARS_NUM);
20     memset(arrive_time, -1, sizeof(arrive_time));
21     memset(leave_time, -1, sizeof(leave_time));
22 }
23
24 void car_arrive(int id, int car_time)
25 {
26     if(is_full_stack(ins))
27     {
28         push_queue(waits, id);
29     }
30     else
31     {
32         push_stack(ins, id);
33         arrive_time[id] = car_time;
34     }
35     printf("id: %d arrive: %d\n", id, car_time);
36 }
37
38 void car_departure(int id, int car_time)
39 {
40     int now_id = -1;
41     while(now_id != id)
42     {
43         now_id = pop_stack(ins);
44         //printf("now_id: %d\n", now_id);
45         if(now_id != id)
46         {
47             push_stack(outs, now_id);
48         }
49     }
50     leave_time[id] = car_time;
51     while(!is_empty_stack(outs))
52     {
53         now_id = pop_stack(outs);
54         push_stack(ins, now_id);
55     }
56     while(!is_empty_queue(waits) && !is_full_stack(ins))
57     {
58         now_id = pop_queue(waits);
59         push_stack(ins, now_id);
60         arrive_time[now_id] = car_time;
61     }
62     printf("id: %d departure: %d\n", id, car_time);
63 }
64
65 void calc()
66 {
67     for(int i = 0; i < CARS_NUM; i++)
68     {
69         if(arrive_time[i] != -1)
70         {
71             if(leave_time[i] != -1)
72             {
73                 printf("id: %d arrive: %d departure: %d stay: %d\n", i, arrive_time[i], leave_time[i], leave_time[i] - arrive_time[i]);
74             }
75             else
76             {
77                 printf("id: %d arrive: %d departure: NONE stay: still staying\n", i, arrive_time[i]);
78             }
79         }
80     }
81 }
82
83 int main()
84 {
85     scanf("%d\n", &n);
86     init();
87     while(1)
88     {
89         char type;
90         int id, car_time;
91         scanf("%c %d %d", &type, &id, &car_time);
92         //printf("type: %c id: %d time: %d\n", type, id, car_time);
93         if(type == 'E')
94         {
95             printf("OVER\n");
96             break;
97         }
98         else if(type == 'A')
99         {
100             car_arrive(id, car_time);
101         }
102         else if(type == 'D')
103         {
104             car_departure(id, car_time);
105         }
106     }
107     calc();
108 }

```

3. 实验结果及结果分析

```
hewo@hewo-thinkpad ~/CS/hdu/hdu-data-structure/lab1 主 main ./lab1 < input
id: 1 arrive: 5
id: 2 arrive: 10
id: 1 departure: 15
id: 3 arrive: 20
id: 4 arrive: 25
id: 5 arrive: 30
id: 2 departure: 35
id: 4 departure: 40
OVER
id: 1 arrive: 5 departure: 15 stay: 10
id: 2 arrive: 10 departure: 35 stay: 25
id: 3 arrive: 20 departure: NONE stay: still staying
id: 4 arrive: 35 departure: 40 stay: 5
id: 5 arrive: 40 departure: NONE stay: still staying
```

实际结果如上图，可以实现停车场的模拟，（上文是输入数据的可读化），最后统计每个车在停车场的的时间（未离去则告知）

4. 实验总结

学习了数据结构 栈 和队列的实现，实际操作发现了容易出错的点位，同时明白了栈和队列在实际中的用途。