

杭州电子科技大学

HANGZHOU DIANZI UNIVERSITY

数据结构课程设计报告

(2021-2022-2 学期)



题目	车厢调度			
学院	理学院			
专业	信息与计算科学			
组号	第十六组	学号	姓名	分工
		20071226	童繁	进程图
		20071227	王瀚功	测试
		20071228	王赛豪	文案
		20071229	吴政豪	示意图
		20071230	武琦	代码
时间	2022 年 4 月 11 日			

目录

1. 需求分析.....	1
2. 概要设计.....	1
3. 详细设计.....	3
3.1 栈的表示与实现.....	3
3.1.1 栈的顺序存储表示.....	3
3.1.2 栈的基本操作函数.....	3
3.2 双栈实现车厢调度.....	5
3.3 车厢序列及其状态变化过程的输出.....	6
4. 功能测试.....	7
5. 研究思考.....	8
6. 心得体会.....	10
7. 附录	11

1. 需求分析

- (1) 设计一个程序, 求出所有可能由车厢系列编号为 $1, 2, \dots, n$ 输出的长度为 n 的车厢序列;
- (2) 利用栈的顺序结构 `SqStack` 之上实现栈的基本操作, 程序对栈的任何存取 (即更改, 读取和状态判别等操作) 必须借助于基本操作进行;
- (3) 利用双向栈存储结构实现调度站和输出序列这两个栈的空间共享, 并思考对于车厢序列长度 n , 两栈共享空间长度 m ;
- (4) 对于每个输出序列印出操作序列状态变化过程。

2. 概要设计

一个数的进栈以后, 有两种处理方式: 要么立刻出栈, 或者下一个数的进栈 (如果还有下一个元素)。其出栈以后, 也有两种处理方式: 要么继续出栈 (栈不为空), 或者下一个数的入栈。

要实现车厢调度, 需要建立栈的顺序结构来实现栈的基本操作, 并利用双向栈存储结构实现调度站和输出序列这两个栈的空间共享。最后, 输出所有可能的序列并印出每个输出序列状态变化的过程。

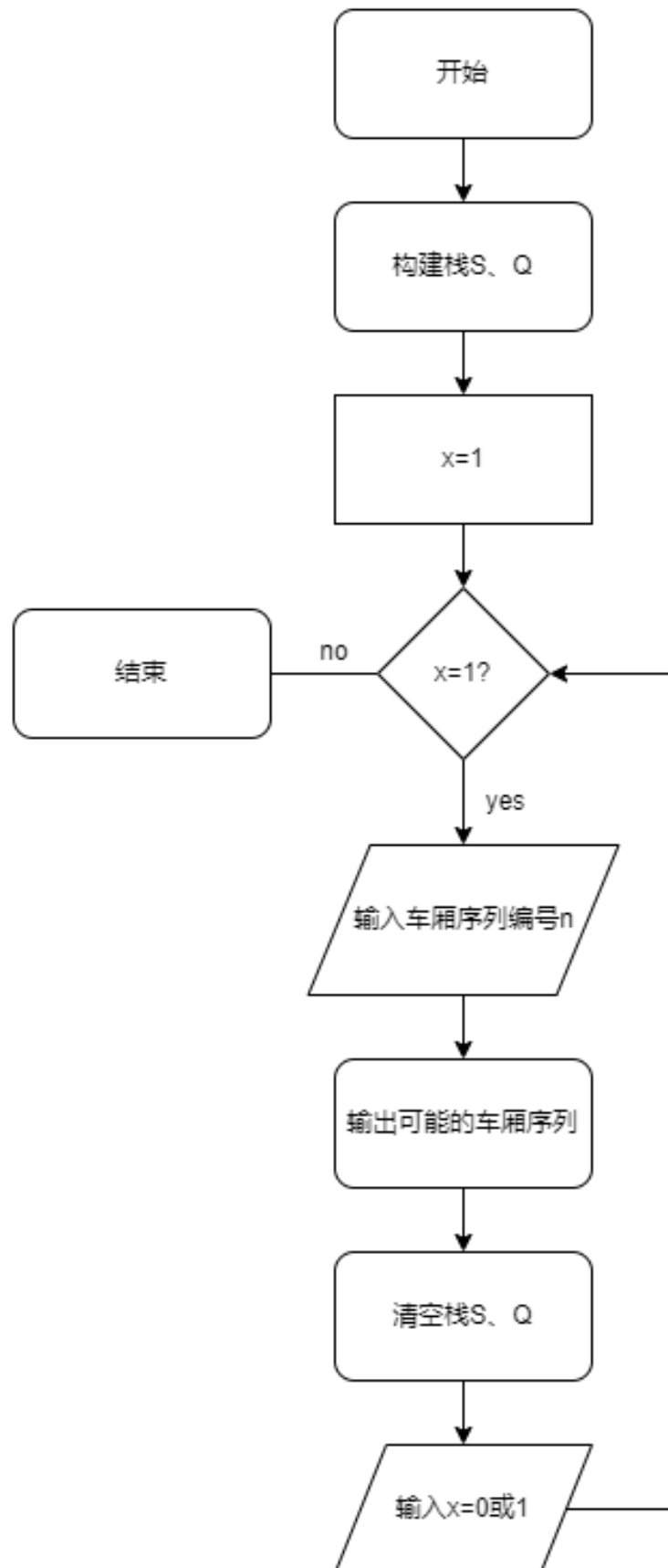


图 1: 主函数流程图

3. 详细设计

3.1 栈的表示与实现

3.1.1 栈的顺序存储表示

Listing 1: 栈的顺序存储

```
1 #define STACK_INIT_SIZE 100 //存储空间初始分配量
2 #define STACKINCREMENT 10 //存储空间分配增量
3 typedef int SElemType;
4 typedef int Status;
5 typedef struct
6 {
7     SElemType *base; //栈底指针
8     SElemType *top; //栈顶指针
9     int stacksize; //栈容量
10 }SqStack;
```

3.1.2 栈的基本操作函数

Listing 2: 栈的操作函数

```
1 Status InitStack(SqStack *S) //构造一个空栈S
2 {
3     S->base=(SElemType *)malloc(STACK_INIT_SIZE*sizeof(SElemType));
4     if(!S->base)
5         exit(OVERFLOW); //存储分配失败
6     S->top=S->base;
7     S->stacksize=STACK_INIT_SIZE;
8     return OK;
9 }
10
11 Status ClearStack(SqStack *S) //把S置为空栈
12 {
13     S->top=S->base;
14     return ERROR;
15 }
16
17 Status StackEmpty(SqStack *S) //若S为空栈, 则返回OK; 否则返回ERROR
18 {
19     if(S->base==S->top)
20         return OK;
21     else
22         return ERROR;
```

```
23 }
24
25 SElemType StackLength(SqStack *S)//返回S的元素个数，即栈的长度
26 {
27     return S->top-S->base;
28 }
29
30 Status Push(SqStack *S,SElemType e)//插入元素e为新的栈顶元素
31 {
32     if(S->top-S->base>=S->stacksize)//栈满，追加存储空间
33     {
34         S->base=(SElemType
35             *)realloc(S->base,(S->stacksize+STACKINCREMENT)*sizeof(SElemType));
36         if(S->base)
37             exit(OVERFLOW);//存储分配失败
38         S->top=S->base+S->stacksize;
39         S->stacksize+=STACKINCREMENT;
40     }
41     *(S->top++)=e;
42     return OK;
43 }
44
45 Status Pop(SqStack *S,SElemType
46     *e)//若栈不空，则删除S的栈顶元素，用e返回其值，并返回OK；否则返回ERROR
47 {
48     if(S->top==S->base)
49         return ERROR;
50     *e=*(--S->top);
51     return OK;
52 }
```

3.2 双栈实现车厢调度

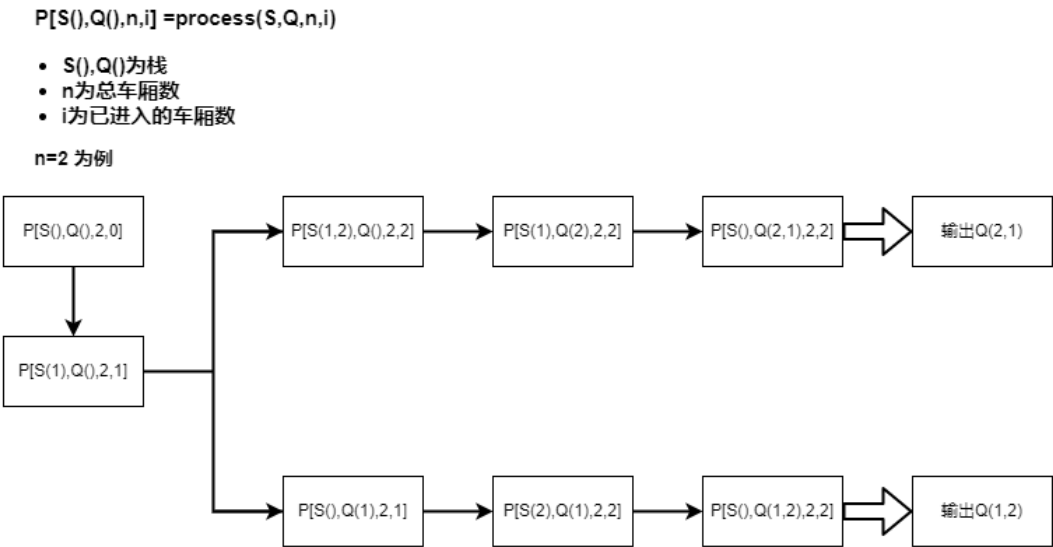


图 2: Process 函数示意图

Listing 3: 车厢调度函数

```
1 Status Process(SqStack *S,SqStack *Q,SElemType n,SElemType i)
2 {
3     //找出当前元素进栈后所有可能的操作
4     SElemType *e,a;
5     e=&a;
6     if(i<n)//编号进栈递归
7     {
8         Push(S,i+1);
9         Process(S,Q,n,i+1);
10        Pop(S,e);
11        i--;
12    }
13    if(!StackEmpty(S))//递归处理出栈
14    {
15        Pop(S,e);
16        Push(Q,a);
17        Process(S,Q,n,i+1);
18        Pop(Q,e);
```

```
19     Push(S,a);
20 }
21 if(StackLength(Q)==n&&StackEmpty(S))//输出可能的方案
22 {
23     printf("第%d种情况:",t);
24     Print(Q);
25     t++;
26 }
27 }
```

3.3 车厢序列及其状态变化过程的输出

Listing 4: 打印函数

```
1 Status Print(SqStack *S)//打印序列状态变化过程
2 {
3     SElemType *p,i,max=0;
4     p=S->base;
5     while(p!=S->top)//输出序列
6     {
7         printf("%d",*p);
8         p++;
9     }
10    printf(" 具体过程:");
11    p=S->base;
12    while(p!=S->top)//输出序列状态变化过程
13    {
14        for(i=max+1;i<=*p;i++)
15            printf("%d进 ",i);
16        printf("%d出 ",*p);
17        if(max<*p)
18            max=*p;
19        p++;
20    }
21    printf("\n");
22 }
```

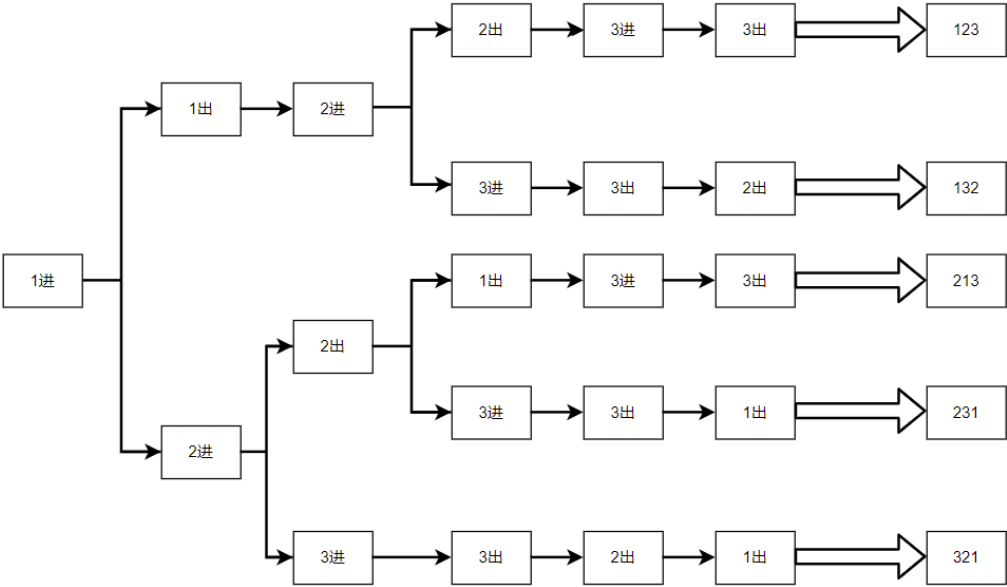



图 3: n=3 时的序列变化情况

4. 功能测试

分别取 n=1,2,3 和 4，测试程序：

```

请输入车厢序列编号:1
输出序号为:
第1种情况:1 具体过程:1进 1出
是否继续输入车厢序列编号: 1:继续 0:退出
1
请输入车厢序列编号:2
输出序号为:
第1种情况:21 具体过程:1进 2进 2出 1出
第2种情况:12 具体过程:1进 1出 2进 2出
是否继续输入车厢序列编号: 1:继续 0:退出
1
请输入车厢序列编号:3
输出序号为:
第1种情况:321 具体过程:1进 2进 3进 3出 2出 1出
第2种情况:231 具体过程:1进 2进 2出 3进 3出 1出
第3种情况:213 具体过程:1进 2进 2出 1出 3进 3出
第4种情况:132 具体过程:1进 1出 2进 3进 3出 2出
第5种情况:123 具体过程:1进 1出 2进 2出 3进 3出
是否继续输入车厢序列编号: 1:继续 0:退出
1
请输入车厢序列编号:4
输出序号为:
第1种情况:4321 具体过程:1进 2进 3进 4进 4出 3出 2出 1出
第2种情况:3421 具体过程:1进 2进 3进 3出 4进 4出 2出 1出
第3种情况:3241 具体过程:1进 2进 3进 3出 2出 4进 4出 1出
第4种情况:3214 具体过程:1进 2进 3进 3出 2出 1出 4进 4出
第5种情况:2431 具体过程:1进 2进 2出 3进 4进 4出 3出 1出
第6种情况:2341 具体过程:1进 2进 2出 3进 3出 4进 4出 1出
第7种情况:2314 具体过程:1进 2进 2出 3进 3出 1出 4进 4出
第8种情况:2143 具体过程:1进 2进 2出 1出 3进 4进 4出 3出
第9种情况:2134 具体过程:1进 2进 2出 1出 3进 3出 4进 4出
第10种情况:1432 具体过程:1进 1出 2进 3进 4进 4出 3出 2出
第11种情况:1342 具体过程:1进 1出 2进 3进 3出 4进 4出 2出
第12种情况:1324 具体过程:1进 1出 2进 3进 3出 2出 4进 4出
第13种情况:1243 具体过程:1进 1出 2进 2出 3进 4进 4出 3出
第14种情况:1234 具体过程:1进 1出 2进 2出 3进 3出 4进 4出
是否继续输入车厢序列编号: 1:继续 0:退出
0
请按任意键继续. . . █

```

图 4: 功能测试

5. 研究思考

对于车厢序列长度 n , 两栈共享空间长度 m 取 n 最合适。理由如下:

1. $m < n$

当双栈存入 m 个数据时, $s.top2 - s.top1 = 1$ (满栈), 无法存入 n 个数据。

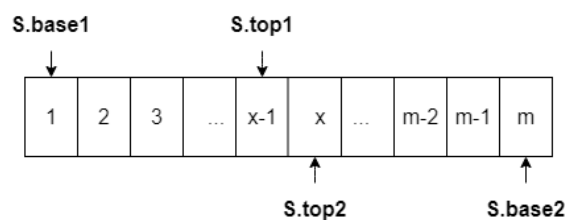


图 5: $m < n$ 时

2. $m = n$

当双栈存入 n 个数据时, $s.top2 - s.top1 = 1$ (满栈), 刚好存入 n 个数据。

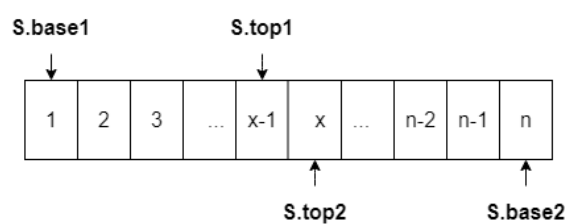


图 6: $m = n$ 时

3. $m < n$

当双栈存入 n 个数据时, $s.top2 - s.top1 > 1$ (满栈), 足够存入 n 个数据。

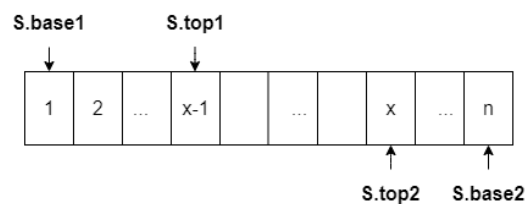


图 7: $m > n$ 时

故综上： $m=n$ 时能刚好处理 n 个数据，并且所用空间最小。

6. 心得体会

这次课程设计的心得体会通过实践我们的收获如下：

- 1、巩固和加深了对数据结构的理解，提高了综合运用本课程所学知识的能力。
- 2、培养了独立思考，深入研究，分析问题、解决问题的能力。
- 3、通过实际编译系统的分析设计、编程调试，掌握了应用软件的分析方法和工程设计方法。

总的来说，这次课程设计让我们组受益匪浅，对数据结构也有了进一步的理解和认识。

7. 附录

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  #define STACK_INIT_SIZE 100 //存储空间初始分配量
5  #define STACKINCREMENT 10 //存储空间分配增量
6  #define OVERFLOW -2
7  #define OK 1
8  #define ERROR 0
9
10 typedef int SElemType;
11 typedef int Status;
12 static SElemType t=1;
13 typedef struct
14 {
15     SElemType *base;//栈底指针
16     SElemType *top;//栈顶指针
17     int stacksize;//栈容量
18 }SqStack;
19
20 Status InitStack(SqStack *S)//构造一个空栈S
21 {
22     S->base=(SElemType *)malloc(STACK_INIT_SIZE*sizeof(SElemType));
23     if(!S->base)
24         exit(OVERFLOW);//存储分配失败
25     S->top=S->base;
26     S->stacksize=STACK_INIT_SIZE;
27     return OK;
28 }
29
30 Status ClearStack(SqStack *S)//把S置为空栈
31 {
32     S->top=S->base;
33     return ERROR;
34 }
35
36 Status StackEmpty(SqStack *S)//若S为空栈,则返回OK;否则返回ERROR
37 {
38     if(S->base==S->top)
39         return OK;
40     else
41         return ERROR;
42 }
43
44 SElemType StackLength(SqStack *S)//返回S的元素个数,即栈的长度
```

```
45 {
46     return S->top-S->base;
47 }
48
49 Status Push(SqStack *S,SElemType e)//插入元素e为新的栈顶元素
50 {
51     if(S->top-S->base>=S->stacksize)//栈满，追加存储空间
52     {
53         S->base=(SElemType
54             *)realloc(S->base,(S->stacksize+STACKINCREMENT)*sizeof(SElemType));
55         if(S->base)
56             exit(OVERFLOW);//存储分配失败
57         S->top=S->base+S->stacksize;
58         S->stacksize+=STACKINCREMENT;
59     }
60     *(S->top++)=e;
61     return OK;
62 }
63 Status Pop(SqStack *S,SElemType
64     *e)//若栈不空，则删除S的栈顶元素，用e返回其值，并返回OK；否则返回ERROR
65 {
66     if(S->top==S->base)
67         return ERROR;
68     *e=*(--S->top);
69     return OK;
70 }
71 Status Print(SqStack *S)//打印序列状态变化过程
72 {
73     SElemType *p,i,max=0;
74     p=S->base;
75     while(p!=S->top)//输出序列
76     {
77         printf("%d",*p);
78         p++;
79     }
80     printf(" 具体过程:");
81     p=S->base;
82     while(p!=S->top)//输出序列状态变化过程
83     {
84         for(i=max+1;i<=*p;i++)
85             printf("%d进 ",i);
86         printf("%d出 ",*p);
87         if(max<*p)
88             max=*p;
89         p++;
90     }
```

```
90     }
91     printf("\n");
92 }
93
94 Status Process(SqStack *S, SqStack *Q, SElemType n, SElemType
    i) //找出当前元素进栈后所有可能的操作
95 {
96     SElemType *e, a;
97     e = &a;
98     if (i < n) //编号进栈递归
99     {
100         Push(S, i+1);
101         Process(S, Q, n, i+1);
102         Pop(S, e);
103         i--;
104     }
105     if (!StackEmpty(S)) //递归处理出栈
106     {
107         Pop(S, e);
108         Push(Q, a);
109         Process(S, Q, n, i+1);
110         Pop(Q, e);
111         Push(S, a);
112     }
113     if (StackLength(Q) == n && StackEmpty(S)) //输出可能的方案
114     {
115         printf("第%d种情况:", t);
116         Print(Q);
117         t++;
118     }
119 }
120
121 SElemType main()
122 {
123     SElemType n, x=1;
124     SqStack *S, *Q, s, q;
125     S = &s;
126     Q = &q;
127     InitStack(S);
128     InitStack(Q);
129     while (x)
130     {
131         printf("请输入车厢序列编号:");
132         scanf("%d", &n);
133         printf("输出序号为:\n");
134         t=1;
135         Process(S, Q, n, 0);
```

```
136     ClearStack(S);
137     ClearStack(Q);
138     printf("是否继续输入车厢序列编号: 1:继续 0:退出\n");
139     scanf("%d",&x);
140 }
141 system("pause");
142 return ERROR;
143 }
```