# 实验报告

**题目**：车厢调度问题，求出所有可能由此输出的长度为n的车厢序列

**班级**：电信1808 **姓名**：喻越 **学号**：U201813467 **完成日期**：2019.04.08

**一、需求分析**

1. **输入的形式和输入值的范围：**输入要调度车厢总数n由键盘输入，要求为正整数，以回车符运行程序（程序设定n=<100）。
2. **输出的形式：**模拟程序以用户和计算机的对话方式执行，即在计算机终端显示提示信息” 请输入要调度车厢总数n=”后，由用户在键盘上输入模拟程序中规定的运算命令；相应的输入数据跟运算结果显示在其后。以” 所有可能序列：”的形式输出。
3. **程序所能达到的功能**
   1. 创建及初始化栈
   2. 栈的基础操作：入栈和出栈
   3. 利用递归求出并输出可能的车厢序列
4. **测试数据**

输入：3 //要调度车厢总数

输出：所有可能序列：

3 2 1

2 3 1

2 1 3

1. 3 2

1 3 2

**二、概要设计**

1. **所有抽象数据类型的定义**：

为实现上述功能，需要一个抽象数据类型：栈

链式线性表

ADT Stack{

数据对象：D={ai|ai∈ElemSet, i=1,2, …,n, n≥0}

数据关系：R1={<ai-1,ai>|ai-1,ai∈D, i=1,2, …,n }

           约定an端为栈顶，a1端为栈底。

基本操作：

   InitStack( &S )

      操作结果：构造一个空栈S。

   DestroyStack ( &S )

初始条件：栈S已存在。

      操作结果：销毁栈S。

ClearStack( &S )

     初始条件：栈S已存在。

操作结果：将S清为空栈。

  StackEmpty( S )

初始条件：栈S已存在。

操作结果：若S为空栈，则返回TRUE，否则返回FALSE。

StackLength( S )

初始条件：栈S已存在。

操作结果：返回S的数据元素个数，即栈的长度。

GetTop( S, &e )

初始条件：栈S已存在且非空。

操作结果：用e返回S的栈顶元素。

Push( &S, e )

初始条件：栈S已存在。

操作结果：插入元素e为新的栈顶元素。

Pop( &S, &e )

初始条件：栈S已存在且非空。

操作结果：删除S的栈顶元素，并用e返回其值

StackTraverse( S, visit() )

初始条件：栈S已存在且非空。

操作结果：从栈底到栈顶依次对S的每个数据元素调用函数visit()。一旦visit()失败，则操作失败。

1. **本程序包含四个模块**
   1. 主程序模块

**int main()**

**{**

初始化栈；

**scanf(n);**

运行核心程序**schedule;**

**return 0;**

**}**

* 1. 栈建立模块——实现栈的建立
  2. 栈基础操作模块——实现栈的入栈和出栈
  3. 车厢调度单元模板——用递归函数处理车厢调度并输出处理结果

**各模块之间的调用关系：**

主程序模块

车厢调度单元模块

t main()

{

int c[20],i,first\_m;

linklist L;

printf("请输入游戏者密码:");

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

{

scanf("%d",&c[i]);

}

L=creatlist(c,N);

printf("请输入初始m值：");

scanf("%d",&first\_m);

game(L,N,first\_m);

return 0;

}

栈基础操作模块

1. **详细设计**

**1.各数据类型**

**int n;** //元素类型

**int outline[maxsize]；** //数组类型

**struct stack** //栈结构类型

**{**

**int data[maxsize];**

**int top;**

**};**

1. **主程序算法**

**int main()**

**{**

**int outline[maxsize];**

**struct stack \*s=(struct stack\*)malloc(sizeof(struct stack));**

**s->top=0;** //在主函数中进行栈的建立和初始化操作

**printf("请输入要调度车厢总数n=");**

**scanf("%d",&n);**

**push(s,1);**

**printf("所有可能序列：\n");**

**schedule(s,1,outline,0);**

**return 0;**

**}**

**4.栈的基础操作（入栈、出栈）的算法**

**void push(struct stack \*ptrs,int item)**

**{**

**ptrs->top++;**

**ptrs->data[ptrs->top]=item;**

**}**

**int pop(struct stack \*ptrs)**

**{**

**int a;**

**a=ptrs->data[ptrs->top];**

**ptrs->top--;**

**return a;**

**}**

1. **车厢调度递归程序**

/\*代码思路源于百度\*/

**void schedule(struct stack \*ptrs,int x,int line[n],int y)**

**{**

**int i,m;**

**if(x<n)**

**{**

**push(ptrs,x+1);**

**schedule(ptrs,x+1,line,y);**

**pop(ptrs);**

**}**

**if(ptrs->top!=0)**

**{**

**m=pop(ptrs);**

**line[y]=m;**

**y++;**

**schedule(ptrs,x,line,y);**

**push(ptrs,m);**

**}**

**if(x==n&&ptrs->top==0)**

**{**

**for(i=0; i<y; i++)**

**{**

**printf("%d\t",line[i]);**

**}**

**printf("\n");**

**}**

**}**

**函数的调用关系反映了模拟程序的层次关系：**

**main**

**schedule**

**push pop**

1. **调试分析**

**1.** **问题解决及经验与体会**

①.在建立push函数时，错将“ptrs->data[ptrs->top]=item”写成“item=ptrs->data[ptrs->top]”导致程序输出结果完全错误；

1. .在建立schedule函数时，错将“void schedule(struct stack \*ptrs,int x,int line[n],int y)”中的所有“，”写成“；”，导致程序编译时出现大量报错，以后应减少低级错误的发生；
2. .关键程序递归函数不会写，无法正确分析问题，最后借助网络资料得以完成。
3. **算法的时空分析**

程序push和pop的时间复杂度均为O（n）,程序schedule的时间复杂

为O（n^2）,整个程序的时间复杂度O（n^2）。

1. **用户使用说明**

1..程序设定要调度的总车厢数不得超过100

2..用户在出现"请输入要调度车厢总数n="时，键入一个正整数，并以"enter"结束输入

3.在出现"请输入要调度车厢总数n="后键入正整数，使用enter运行程序即会出现如下结果：

所有可能序列：

（下面是排列好的数字）

1. **测试结果**
2. 键入“3（回车）”

输出：

所有可能序列：

3 2 1

2 3 1

2 1 3

1 3 2

1 2 3

1. 键入“6（回车）”

输出：

所有可能序列：

4 3 2 1

3 4 2 1

3 2 4 1

3 2 1 4

2 4 3 1

2 3 4 1

2 3 1 4

2 1 4 3

2 1 3 4

1 4 3 2

1 3 4 2

1 3 2 4

1 2 4 3

1 2 3 4

**附录**

**源程序文件名清单：**

**stdio.h**

**stdlib.h**  //运行malloc和free函数的源程序