实习报告：2.3题 车厢调度

实习报告

题目：假设停在铁路调度站入口处的车厢序列的编号次序为1，2，3，…，n。设计一个程序，求出所有可能输出的长度为n的车厢序列。

班级：电信1809 姓名：文竞宇 学号：U201813487 完成日期：2019.4.8

**一、需求分析**

1.用编号依次为1，2，3，……，n表示停在铁路调度站入口处的车厢序列。

2.用一个栈形象地表示为火车的调度站。

3.利用栈先进后出的性质，结合递归和回溯算法，实现编号1…n的车厢的所有可能的序列。

**二、概要设计**

1.设定栈的抽象数据类型定义：

**ADT** Stack{

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

**数据关系：**R1={<,> | ,∈D, i=2，……，n}

**基本操作：**

InitStack(&S)

操作结果：构造一个空栈S。

Push(&S,e);

初始条件：栈S已存在。

操作结果：在栈S的栈顶插入新的栈顶元素e。

Pop(&S,e);

初始条件：栈S已存在。

操作结果：删除S的栈顶元素，并以e返回其值。

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。

}**ADT** Stack

2.本程序包括两个模块：

* + 1. 初始化数据——输入总数——初始化栈和序列
    2. 显示所有的序列——递归调用——输出所有结果

**三、详细设计**

1.栈类型

struct {

SElemType \*base;

SElemType \*top;

int stacksize;

}SqStack;

栈的基本操作设置如下：

Status InitStack(SqStack &S)

{

S.base = (SElemType\*)malloc(STACK\_INIT\_SIZE \* sizeof(SElemType));

if (!S.base)

exit(OVERFLOW);

S.top = S.base;

S.stacksize = STACK\_INIT\_SIZE;

}//初始化

Status StackEmpty(SqStack S)

{

if (S.top == S.base)

return ERROR;

return OK;

}//若S为空栈，则返回OK；否则返回ERROR

Status Push(SqStack &S, SElemType e)

{

if (S.top - S.base >= S.stacksize)

{

S.base = (SElemType\*)realloc(S.base, (S.stacksize + STACKINCREMENT) \* sizeof(SElemType));

if (!S.base)

exit(OVERFLOW);

S.top = S.base + S.stacksize;

S.stacksize += STACKINCREMENT;

}

\*S.top++ = e;

return OK;

}//若分配内存成功，则在S的栈顶插入新的栈顶元素e，并返回OK；否则栈不变，并返回ERROR

Status Pop(SqStack &S, SElemType &e)

{

if (S.top == S.base)

return ERROR;

e = \*--S.top;

return OK;

}//若栈不空，则删除S的栈顶元素并以e带回其值，并返回OK；否则栈不变，并返回ERROR

2.求所有序列的伪码算法：

Status Move(SqStack temp, int In[], int Out[], int i,int j,int n)

{

if (i < n) //入栈

{

Push(temp, In[i]);

Move(temp, In, Out, i + 1, j, n);

Pop(temp, Out[j]);

}

if (!StackEmpty(temp)) //出栈

{

Pop(temp, Out[j]);

Move(temp, In, Out, i, j + 1, n);

Push(temp, Out[j]);

}

if (j >= n && StackEmpty(temp))

{

printf("Out---->");

PrintArray(Out, n);

}

return OK;

}

3.主函数和其他函数的伪码算法：

int main()

{

int In[N];

int i;

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

In[i] = i + 1;

printf("In <----");

PrintArray(In, N);

SqStack temp;

InitStack(temp);

int Out[N];

Move(temp, In, Out, 0, 0, N);

return 0;

}

Status PrintArray(int \*s, int n)

{

for (int i = 0; i < n; i++)

printf("%5d", s[i]);

printf("\n");

return OK;

}

**四、调试分析**

1.本次作业比较困难，有一个递归算法，即求出所有的情况，所以调试过程不太顺利。

2.栈的操作比较简单。

3.输入与输出数据类型选择数组可以比较方便地存储。

4.本题中主要算法：Move的时间复杂度为。

**五、用户手册（略）**

**六、测试结果（略）**

**七、附录**

源程序文件名清单：

Training 2 Train Management.cpp //主函数