**实验报告**

题目：模拟车厢调度问题

班级：电信1809 姓名：刘俊逸 学号：U201813492 完成日期：2019.4.8

1. 需求分析
2. 本实验的栈其实是一个数组，他存储先后进入该栈的数，然后输出各个不同的序列。
3. 演示程序以用户和计算机的对话方式执行，即在计算机终端上显示“提示信息”之后，由用户在键盘上输入演示程序中规定的运算命令；相应的输入数据和运算结果显示在其后。
4. 程序执行的命令包括：
5. 构造栈指针
6. 构造push函数
7. 构造pop函数
8. 构造process函数
9. 构造主函数
10. 测试数据

当n=4时，输出序列为：

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

1. 概要设计

为实现上述程序功能，用栈的顺序存储结构模拟该过程。

1. 详细设计
2. 设置栈指针并初始化

struct N

{

int data[max];

int top;

}s;//设置一个栈指针

int n;//车厢的个数

void initstack()

{

s.top=-1;

}

1. 构造push函数

void push(int q)//元素q进栈

{

s.top++;

s.data[s.top]=q;

}

1. 构造pop函数

int pop()//出栈

{

int temp;

temp=s.data[s.top];

s.top--;

return temp;

}

1. 判断栈是否空

int emp()//判断栈是否空

{

if(s.top==-1)

return 1;

else

return 0;

}

1. 构造process函数

void process(int pos,int path[],int a)//处理位置为pos的元素

{

int m,i;

if(pos<n)//编号进栈递归，一个数进栈后，有两种处理方式：要么出栈，要么下一个数进栈

{

push(pos+1);//当前元素进栈后下一元素进栈

process(pos+1,path,a);//处理下一个元素，返回表示下一元素进栈的情况处理完了

pop();//下一个元素处理完后pop掉，准备处理直接出栈

}

if(!emp())//递归处理出栈，一个数出栈后，有两种处理方式：要么进栈，要么下一个数出栈

{

m=pop();

path[a]=m;

a++;

process(pos,path,a);//出栈后下一个元素继续进栈

push(m);

}

if(pos==n&&emp())//输出一种可能的情况

{

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

{

printf("%d ",path[i]);

}

printf("\n");

}

}

1. 构造主函数

void main()

{

int path[max];

printf("输入要调度的车厢总数:");

scanf("%d",&n);

initstack();

push(1);

printf("输出所有序列\n");

process(1,path,0);//从1开始递归处理所有元素

}

1. 调试分析

在调试过程中，主要是在process这个函数设计过程中遇到不少麻烦，而这主要的原因就在于平时对递归的调用算法没太钻研所致，在此过程中借助于课本上以及网上查到的关于递归的资料才算勉强完成这个函数

1. 测试结果
2. n=1：序列：1
3. n=2: 序列：1 2；2 1
4. n=3：序列：1 2 3；1 3 2；2 1 3；2 3 1；3 2 1