数据结构实验报告  
实验名称:实验二——题目5  
学生姓名:蒲亚雄  
班级:电信1808  
学号:U201813473  
日期: 2019年4月15日  
1.实验要求  
利用队列结构实现车厢重排问题。车厢重排问题如下:  
一列货车共有n节车厢,每个车厢都有自己的编号,编号范围从1~n。给定任意次序的车厢,通过转轨站将车厢编号按顺序重新排成1~n。转轨站共有k个缓冲轨,缓冲轨位于入轨和出轨之间。开始时,车厢从入轨进入缓冲轨,经过缓冲轨的重排后,按1~n的顺序进入出轨。缓冲轨按照先进先出方式,编写一个算法,将任意次序的车厢进行重排,输出每个缓冲轨中的车厢编号。  
2. 程序分析  
将每个轨道视为一个队列,每一个车厢视为一个结构体的结点,结点存储车厢编号以及下一节车厢的地址。用尾插法建立队列,并根据队列队尾入队,队头出队的特点实现结点的出队以及入队。由于车厢在重排过程中将会频繁进行入队以及出队的操作,如果采用顺序存储结构,则在结点出入队时是操作将会十分不方便,还会占用大量多余的空间和时间,故选用链式存储结构,可以直接调动结点,十分简洁。  
重排过程比如:编号为3的车厢进入缓冲轨1,则下一个编号小于3的车厢则必须进入下一个缓冲轨2,而编号大于3的车厢则进入缓冲轨1,排在3号车厢的后面。在把车厢c移至缓冲轨是,车厢c应该移动到这样的缓冲轨中:该缓冲轨中队尾车厢的编号小于c;如果有多个缓冲轨满足这一条件,则选择对位车厢编号最大的缓冲轨,否则选择一个空的  
缓冲轨。这样,出轨的时候才可以按照从小到大的顺序重新编排。2.1 存储结构  
NO 1  
next1  
NO n  
^  
next1  
NO 2  
next2  
………  
front rear  
2.2 关键算法分析  
1. 移动车厢算法(车厢要从a队列移至b队列)  
·自然语言描述  
(1)判断a队列是否为空,为空则输出“队列下溢”,提示出  
错。  
(2)否则,设置工作指针p指向a队列的队头元素,  
(3)将b的尾指针指向next域指向p;  
(4)b队列的尾指针后移,指向p指向的结点;  
(5)a头结点指向p后一个结点  
(6)p的next域置为空,完成一次车厢的移动  
(7)a队列车厢数减1,b队列车厢数加1  
·伪代码描述  
两个队列,车厢要从a队列移至b队列;  
(1)bus\* p=a.front->next;  
(2)if(!p) cout<<"Underflow"<<endl;  
(3)b.rear->nex  
  
我的手机 2019/4/15 23:01:50  
  
(3)b.rear->next=p;  
(4)b.rear=b.rear->next;  
(5)a.front->next=p->next;  
(6)p->next=NULL;  
(7)if(!(a.front->next))a.rear=a.front;  
(8)a.Number--;b.Number++;  
图示  
p  
a  
next2  
NO 2  
next1  
NO n  
^  
NO 1  
^  
⑵  
a.rear  
b.front  
b.rear  
b.rear  
a.front  
⑴  
b  
NO 1  
next1  
NO 2  
next1  
NO n  
p  
next2  
算法时间复杂度O(1),空间复杂度S(1)。  
2. 重排车厢算法  
·自然语言描述  
next\_out记录下一个要输出的车厢号,初始值为1;Num\_hc个缓冲轨。  
(1)依次取出入轨中的每一个车厢编号,若编号与next\_out相等,则将该  
车厢移动到出轨,next\_out增一,并输出该步骤。此时完成一次调  
度,函数返回值为true  
(2)若入轨中没有符合要求的车厢,考察每一个缓冲轨队列的第一个车厢  
编号,若与next\_out相等,则将该车厢移入出轨,next\_out增一,并输出该步骤。此时完成一次调度,函数返回值为true  
(3)若入轨和缓冲轨的队友元素没有标号为next\_out的车厢,则考察每个  
缓冲轨的尾车厢编号,找到小于入轨中第一个车厢编号的的最大队尾元素所在的缓冲轨并输出该步骤。此时完成一次调度,函数返回值为true  
(4)如果这样的缓冲轨存在,则将入轨中的第一个车厢移至该缓冲轨并输  
出该步骤。此时完成一次调度,函数返回值为true  
(5)如果没有这样的缓冲轨,但有多于一个空缓冲轨,则把入轨中的第一  
个车厢移至一个空缓冲轨并输出该步骤。此时完成一次调度,函数返回值为true  
(6)如果没有空缓冲轨,则车厢无法重排,算法结束,函数返  
回值为false  
(7)如果入轨中没有车厢,则排序完成,算法结束,函数返回值为  
false  
·伪代码描述  
1. 如果入轨中的车厢编号等于next\_out  
1.1 将该车厢移到出轨  
1.2 next\_out++  
1.3 输出该步骤,函数返回值为true  
2. 否则,考察缓冲队列  
For(int i=1;i<Num\_hc;i++)  
2.1 取队列i的队头元素NO  
2.2 如果NO==next\_out,则  
2.2.1  
  
我的手机 2019/4/15 23:02:04  
  
(3)b.rear->next=p;  
(4)b.rear=b.rear->next;  
(5)a.front->next=p->next;  
(6)p->next=NULL;  
(7)if(!(a.front->next))a.rear=a.front;  
(8)a.Number--;b.Number++;  
图示  
p  
a  
next2  
NO 2  
next1  
NO n  
^  
NO 1  
^  
⑵  
a.rear  
b.front  
b.rear  
b.rear  
a.front  
⑴  
b  
NO 1  
next1  
NO 2  
next1  
NO n  
p  
next2  
算法时间复杂度O(1),空间复杂度S(1)。  
2. 重排车厢算法  
·自然语言描述  
next\_out记录下一个要输出的车厢号,初始值为1;Num\_hc个缓冲轨。  
(1)依次取出入轨中的每一个车厢编号,若编号与next\_out相等,则将该  
车厢移动到出轨,next\_out增一,并输出该步骤。此时完成一次调  
度,函数返回值为true  
(2)若入轨中没有符合要求的车厢,考察每一个缓冲轨队列的第一个车厢  
编号,若与next\_out相等,则将该车厢移入出轨,next\_out增一,并输出该步骤。此时完成一次调度,函数返回值为true  
(3)若入轨和缓冲轨的队友元素没有标号为next\_out的车厢,则考察每个  
缓冲轨的尾车厢编号,找到小于入轨中第一个车厢编号的的最大队尾元素所在的缓冲轨并输出该步骤。此时完成一次调度,函数返回值为true  
(4)如果这样的缓冲轨存在,则将入轨中的第一个车厢移至该缓冲轨并输  
出该步骤。此时完成一次调度,函数返回值为true  
(5)如果没有这样的缓冲轨,但有多于一个空缓冲轨,则把入轨中的第一  
个车厢移至一个空缓冲轨并输出该步骤。此时完成一次调度,函数返回值为true  
(6)如果没有空缓冲轨,则车厢无法重排,算法结束,函数返  
回值为false  
(7)如果入轨中没有车厢,则排序完成,算法结束,函数返回值为  
false  
·伪代码描述  
1. 如果入轨中的车厢编号等于next\_out  
1.1 将该车厢移到出轨  
1.2 next\_out++  
1.3 输出该步骤,函数返回值为true  
2. 否则,考察缓冲队列  
For(int i=1;i<Num\_hc;i++)  
2.1 取队列i的队头元素NO  
2.2 如果NO==next\_out,则  
2.2.1 将队列i的队头元素移到出轨,并输出  
2.2.2 next\_out++  
2.2.3 输出该步骤以及缓冲轨的情况,函数返回值true  
3. 如果入轨和缓冲轨的队头元素没有编号为next\_out的车厢,  
则  
3.1 求小于入轨中第一车车厢编号的最大队尾元素所在的队列  
编号nextH  
3.2 求第一个空缓冲轨的编号vacantH  
3.3 如果nextH存在,则把入轨中的第一个车厢移至缓冲轨  
nextH,函数返回值为true,输出该步骤以及缓冲轨情况  
3.4 如果nextH不存在,但有空缓冲轨,则把入轨中的第一个车  
厢移至一个空缓冲轨函数返回至为true;否则车厢无法完成  
重排,算法结束,函数返回值为false  
4. 如果如果入轨中没有车厢,则重排完成,算法结束,函数  
返回值为false  
算法的时间复杂度为O(n),空间复杂度为S(4)  
2.3 其他  
程序中的车厢移动函数包括两个,为重载函数,但算法是一样的。  
3. 程序运行结果  
开始  
输入车厢数目  
输入车厢排序  
创建入轨队列  
输入缓冲轨数目  
创建缓冲轨队列(数组)  
输出重排前的序列  
进行车厢重排,并随时跟踪缓冲轨中车厢情况  
输出重排后的序列  
结束  
测试条件:问题规模的数量级是1,车厢数量为5,车厢排序为:4 2 1 5 3,缓冲轨数量为2,运行结果如下:  
  
当缓冲轨数目为1时,运行结果如下:  
  
测试结论:程序可以完成程序所要求的车厢重排功能,并能随时跟踪缓冲轨中的车厢情况;当所给条件无法完成重排时,会给用户提  
示。  
4.程序

#include<stdio.h>   
#define MaxLen 100   
struct snode{   
int data[MaxLen];   
int top;   
}s;//定义一个栈指针   
int n;//定义输入序列总个数   
void Initstack()   
{   
s.top=-1;   
}   
void push(int q)//元素n进栈   
{   
s.top++;   
s.data[s.top]=q;   
}   
int pop()//出栈   
{   
int temp;   
temp=s.data[s.top];   
s.top--;   
return temp;   
}   
int Emptys()//判断栈空   
{   
if(s.top==-1)   
return 1;   
else   
return 0;   
}   
void process(int pos,int path[],int curp)//当前处理位置pos的元素   
{   
int m,i;   
if(pos<n)//编号进栈递归   
{   
push(pos+1);//当前元素进栈后下一个元素继续进栈   
process(pos+1,path,curp); //处理下一个元素，返回表明下一个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(!Emptys())//递归处理出栈  
{   
m=pop();   
path[curp]=m;   
curp++;   
process(pos,path,curp);//出栈后处理下一个素继续进栈  
push(m);   
}   
if(pos==n&&Emptys())//输出一种可能的方案   
{   
for(i=0;i<curp;i++)   
printf("%2d",path[i]);   
printf("\n");   
}   
}   
void main()   
{   
int path[MaxLen];   
printf("输入要调度车厢总数:");   
scanf("%d",&n);   
Initstack();   
push(1);   
printf("所有输出序列:\n");   
process(1,path,0); //从1 开始，递归处理所有元素   
}

 5总结  
原来程序中重排的函数只能调用一次调度,要完成整个车厢的重排工作,必须要实现循环,故将重排函数的返回值设为bool型,通过do-while完成这个循环,设置了一个变量judge来决定循环的终止条件。如果排序完成则返回true,未完成则返回false。  
这次试验使我熟悉了链队列的特点以及操作方法,并且体会到了其在实际运用中的重要性。同时,车厢要完成重排需要考虑到各种情况,在编写算法的过程中锻炼了我的逻辑思维能力和考虑问题的周密性。  
下一步需要改进的是:如果车厢数量较多,用户在输入时可能会出现错误,如果在重排之前编写一个能够检验用户输入情况是否有错的函数就会使程序更加完美。同时算法中使用了许多if嵌套的语句,使得程序的可读性变差,也增加了时间复杂度。