计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：朱龙康 学号：1824120002 专业：计算机科学与技术\_ 年级：\_2018级

课程：数据结构 主讲教师：\_袁彩虹\_ 辅导教师：\_\_袁彩虹\_\_

实验时间：\_\_2020\_年 \_3\_月 \_\_27\_日 \_上\_午\_8\_时至\_10\_时，实验地点\_\_\_\_\_\_\_

实验题目： 栈和队列的操作

实验目的： 掌握顺序栈和链栈的基本操作，循环队列和链队列的基本操作，为后续学习打下基础，以便更好地达成后续更高层次的课程目标。

实验环境（硬件和软件） CodeBlocks

实验内容：

*1）编程实现顺序栈和链栈的基本操作：建栈，取栈顶元素，入栈，出栈；*

*2）编程实现循环队列和链队列的基本操作：建队列，取队头元素，入队，出队。*

实验步骤：

1）**顺序栈的定义和基本操作**

//定义顺序栈的存储结构

typedef struct {

int \*base;//栈底指针

int \*top;//栈尾指针

int stacksize;//栈的最大容量 用于判断是否栈满

}SqStack;

//1.初始化

bool InitStack(SqStack &s){

s.base=new int[MAXSIZE];//为顺序栈分配空间

if(!s.base)return false;//分配失败

s.top=s.base;//空栈时 top=base

s.stacksize=MAXSIZE;

return true;//分配成功

}

//2.入栈

bool Push(SqStack &s,int e){

if(s.top-s.base==s.stacksize)return false;//1.先判断是否满

\*s.top=e;//元素压入栈顶

s.top++;//顶指针加一

}

//3.出栈

bool Pop(SqStack &s,int&m){

if(s.top==s.base)return false;//栈空

s.top--;//顶指针减1 此处和入栈相反

m=\*s.top;//取栈顶元素

}

//4.取栈顶元素

int GetTop(SqStack s){

if(s.top!=s.base)return \*(s.top-1);//栈非空

else return NULL;//栈空 返回null

}

**2）链栈的定义和基本操作**

//---------链栈存储结构

typedef struct StackNode{

int data;//数据域

struct StackNode \*next;//指针域

}StackNode,\*LinkStack;

//--------1.将链表头作为栈顶

//---------2.不用像单链表那样设置头结点

//----------1.初始化

bool InitStack(LinkStack &s){

s=NULL;

return true;

}

//----------2.入栈

bool Push(LinkStack &s,int e){

StackNode \*p=new StackNode;//先为e创建一个节点

p->data=e;//赋值给该节点

p->next=s;//新节点插入

s=p;//修改栈顶指针

return true;

}

//----------3.出栈

bool Pop(LinkStack&s,int &m){

if(s==NUll)return false;//判断空

m=s->data;//保存栈顶元素

StackNode\*p=s;//创建节点临时保存栈顶节点

s=s->next;//修改栈顶指针

delete p;//释放节点

return true;

}

//---------4.取栈顶元素

int GetTop(LinkStack l){

if(l==null)return null;

else return l->data;//返回栈顶元素

}

**3）循环队列的定义和基本操作**

//-------队列的顺序存储结构

#define MAXSIZE 50//队列最大长度

typedef struct

{

double \*base;//存储空间的基地址

int front;//头指针

int rear;//尾指针

}SqQueue;

//-----------1.初始化

bool InitQueue(SqQueue &q){

q.base=new double[MAXSIZE];//申请空间

if(!q.base)return false;//申请失败

q.front=q.rear=0;//队列空 头尾指针置零

return true;

}

//-----------2.求队长

int GetqLength(SqQueue q){

return (q.rear-q.front+MAXSIZE)%MAXSIZE;

}

//-----------3.入队

bool InQueue(SqQueue &q,double e){

if((q.rear+1)%MAXSIZE==q.front)return false;//队满

q.base[q.rear]=e;//将e入队

q.rear=(q.rear+1)%MAXSIZE;//尾指针移动

return true;

}

//-----------4.出队

bool OutQueue(SqQueue &q,double &m){

if(q.rear==q.front)return false;//队空

m=q.base[q.front];//将m传给参数带出

q.front=(q.front+1)%MAXSIZE;//头指针改变

return true;

}

//-----------5.取队头元素

double GetFront(SqQueue s){

if(s.front==s.rear)return NULL;//队空

else return s.base[s.front];//返回头指针所指元素

}

**3）链队列的定义和基本操作**

//-----结点的存储结构

typedef struct QNode

{

int data;//数据域

struct QNode \*next;//指针域

}QNode,\*QueuePtr;

//-------链队的存储结构

typedef struct

{

QueuePtr rear;//尾指针

QueuePtr front;//头指针

}LinkQueue;

//初始化

bool InitQueue(LinkQueue &q){

q.front=q.rear=new QNode;//生成头结点，头尾指针都指向该节点

q.front->next=NULL;//头结点next域为空

return true;

}

//-----入队

bool InQueue(LinkQueue &q,int e){

QueuePtr p=new QNode;//生成新节点

p->data=e;//新节点的数据为入队数据

p->next=NULL;//新节点的next赋空

q.rear->next=p;//尾节点next指向新节点

q.rear=p;//尾指针后移

return true;

}

//----------出队

bool OutQueue(LinkQueue &q,int &m){

if(q.front==q.rear)return false;//空队

Queueptr p=q.front->next;//建立新指针指向首元节点，用于临时保存该节点

m=q.front->next->data;//保存要删除节点的数据

q.front->next=p->next;//头结点的next指向首元节点的后一个节点

if(q.rear=p)q.rear=q.front;//若尾节点就是首元节点，为了防止删除尾指针，需要将尾指针前移

delete p;//释放删除的节点空间

return true;

}

//取队顶元素

int GetFront(LinkQueue q){

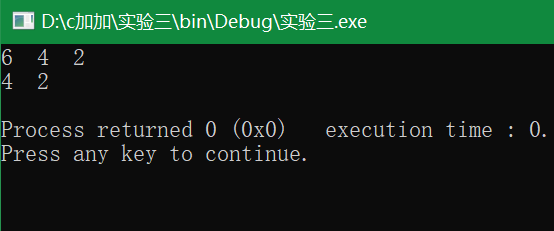
if(q.front==q.rear)return NULL;//队空

return q.front->next->data;//返回队顶数据

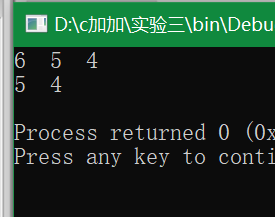
}

实验数据记录：

顺序栈 入栈6 4 2 出栈6

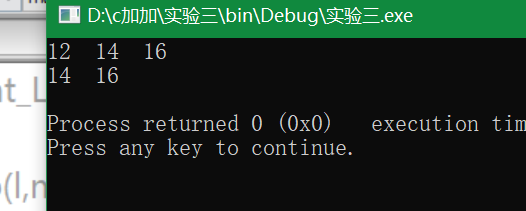


链栈 入栈 6 5 4 出栈 6



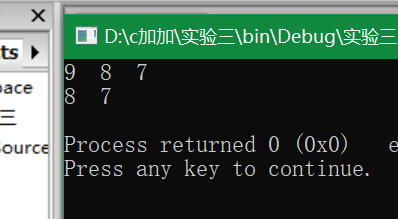
循环队列 入队 12 14 16

出队 12



链队列 入队9 8 7

出队 9



问题讨论：