# 数据结构实验二

**专业：计算机科学与技术 年级：18级**

**学号： E01814234 姓名：郭佳明**

**实验日期： 2019-4-8 报告提交时间：**

**得分： 评阅人：**

目录（按住ctrl并单击可直接访问）

所有代码已经通过上机测试并上传到GitHub

网址链接: https://github.com/LaPhilosophie/Data\_Structure\_Lab\_Work\_2

[数据结构实验二 1](#_Toc6835792)

[顺序栈的相关操作 2](#_Toc6835793)

[实验要求： 2](#_Toc6835794)

[源代码 2](#_Toc6835795)

[运行结果截图 5](#_Toc6835796)

[链栈的相关操作 6](#_Toc6835797)

[实验要求： 6](#_Toc6835798)

[源代码 6](#_Toc6835799)

[运行结果截图 9](#_Toc6835800)

[循环队列的相关操作 10](#_Toc6835801)

[实验要求： 10](#_Toc6835802)

[源代码 10](#_Toc6835803)

[运行结果截图 13](#_Toc6835804)

[链队列的相关操作 14](#_Toc6835805)

[实验要求： 14](#_Toc6835806)

[源代码 14](#_Toc6835807)

[运行结果截图 17](#_Toc6835808)

[实现进制转换（栈） 18](#_Toc6835809)

[实验要求 18](#_Toc6835810)

[源代码 18](#_Toc6835811)

[运行结果截图 20](#_Toc6835812)

# 顺序栈的相关操作

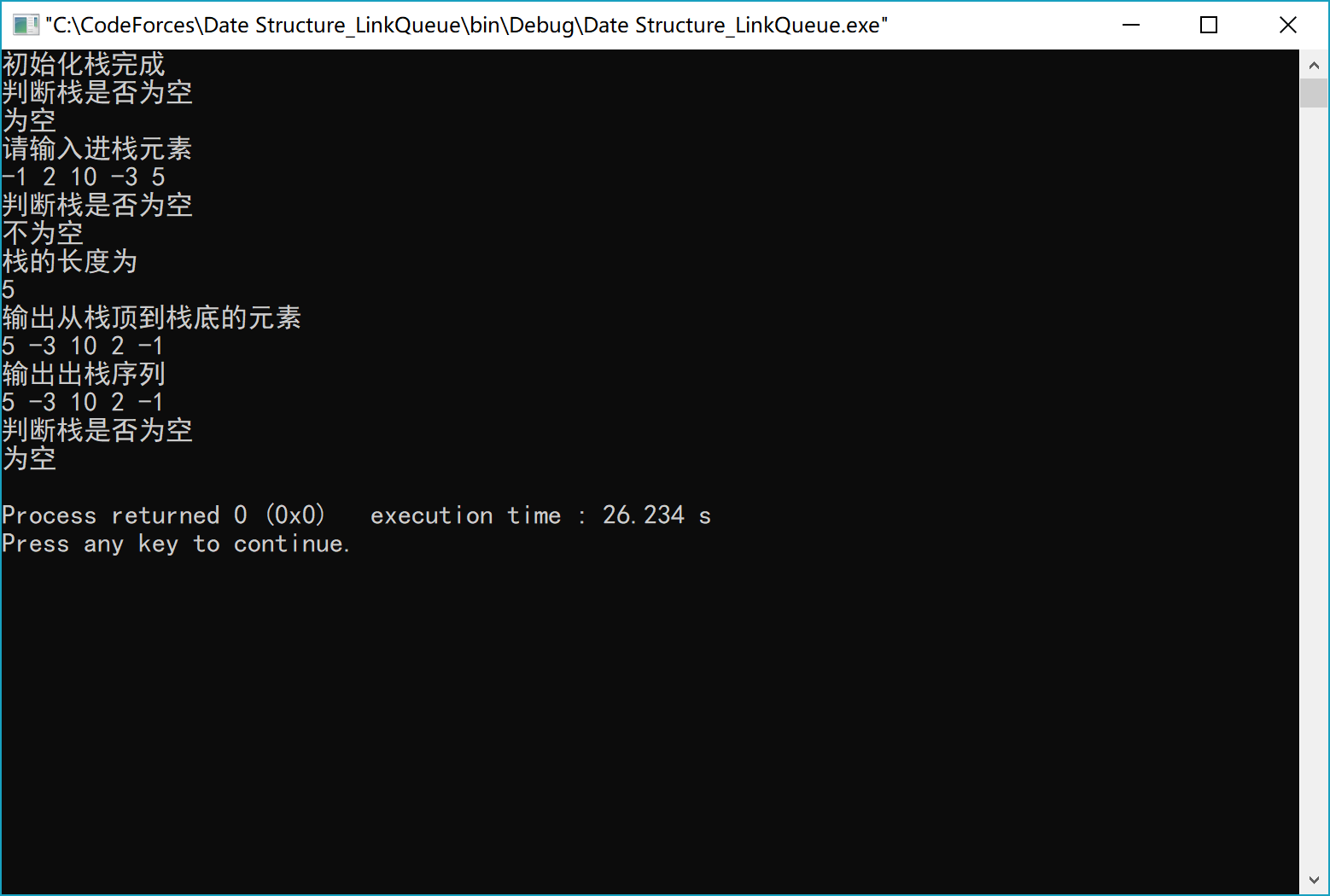
## 实验要求：

|  |
| --- |
| /\* |
|  | 2.1 实现顺序栈的基本运算 |
|  | 编写一个程序，实现顺序栈的各种基本运算，并在此基础上设计一个主程序完成如下功能： |
|  | （1）初始化栈s； |
|  | （2）判断栈s是否为空； |
|  | （3）依次进栈元素-1,2,10,-3,5； |
|  | （4）判断栈s是否为空； |
|  | （5）输出栈长度； |
|  | （6）输出从栈顶到栈底的元素；（仅仅输出） |
|  | （7）输出出栈序列； （出栈） |
|  | （8）判断栈s是否为空。 |
|  | \*/ |

## 源代码

|  |  |
| --- | --- |
|  | #include <bits/stdc++.h> |
|  | using namespace std; |
|  | #define MAXSIZE 100 |
|  | #define ERROR -1 |
|  | #define OK 1 |
|  | typedef struct |
|  | { |
|  | int \*base; |
|  | int \*top; |
|  | int stacksize; |
|  | }SqStack; |
|  | typedef int Status; |
|  | Status InitStack(SqStack &S) |
|  | { |
|  | S.base=new int [MAXSIZE]; |
|  | if(!S.base) |
|  | exit(OVERFLOW); |
|  | S.top=S.base; |
|  | S.stacksize=MAXSIZE; |
|  | return OK; |
|  | } |
|  | Status Push(SqStack &S,int e) |
|  | { |
|  | if(S.top-S.base==S.stacksize) |
|  | return ERROR; |
|  | \*S.top++=e; |
|  | return OK; |
|  | } |
|  | Status Pop(SqStack &S,int &e) |
|  | { |
|  | if(S.top==S.base) |
|  | return ERROR; |
|  | e=\*--S.top; |
|  | return OK; |
|  | } |
|  | int GetTop(SqStack S)//返回栈顶元素，不修改指针 |
|  | { |
|  | if(S.top!=S.base) |
|  | return \*(S.top-1); |
|  | } |
|  | void IsEmpty(SqStack S) |
|  | { |
|  | if(S.top==S.base) |
|  | cout<<"为空"<<endl; |
|  | else |
|  | cout<<"不为空"<<endl; |
|  | } |
|  | void CreatStack(SqStack &S) |
|  | { |
|  | int e; |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | cin>>e; |
|  | Push(S,e); |
|  | } |
|  | } |
|  | int StackLength(SqStack S) |
|  | { |
|  | return S.top-S.base; |
|  | } |
|  | void Traverse(SqStack S) |
|  | { |
|  | int \*p; |
|  | p=S.top-1; |
|  | while(p>=S.base) |
|  | { |
|  | cout<<\*p<<" "; |
|  | p--; |
|  | } |
|  | cout<<endl; |
|  | } |
|  | void PopAll(SqStack &S) |
|  | { |
|  | int e; |
|  | while(S.top!=S.base) |
|  | { |
|  | Pop(S,e); |
|  | cout<<e<<" "; |
|  | } |
|  | cout<<endl; |
|  | } |
|  | int main() |
|  | { |
|  | SqStack S; |
|  | InitStack(S); |
|  |  |
|  | cout<<"初始化栈完成"<<endl; |
|  |  |
|  | cout<<"判断栈是否为空"<<endl; |
|  | IsEmpty(S); |
|  |  |
|  | cout<<"请输入进栈元素"<<endl; |
|  | CreatStack(S); |
|  |  |
|  | cout<<"判断栈是否为空"<<endl; |
|  | IsEmpty(S); |
|  |  |
|  | cout<<"栈的长度为"<<endl; |
|  | cout<<StackLength(S)<<endl; |
|  |  |
|  | cout<<"输出从栈顶到栈底的元素"<<endl; |
|  | Traverse(S); |
|  |  |
|  | cout<<"输出出栈序列"<<endl; |
|  | PopAll(S); |
|  |  |
|  | cout<<"判断栈是否为空"<<endl; |
|  | IsEmpty(S); |
|  | return 0; |
|  | } |
|  |  |

## 运行结果截图



# 链栈的相关操作

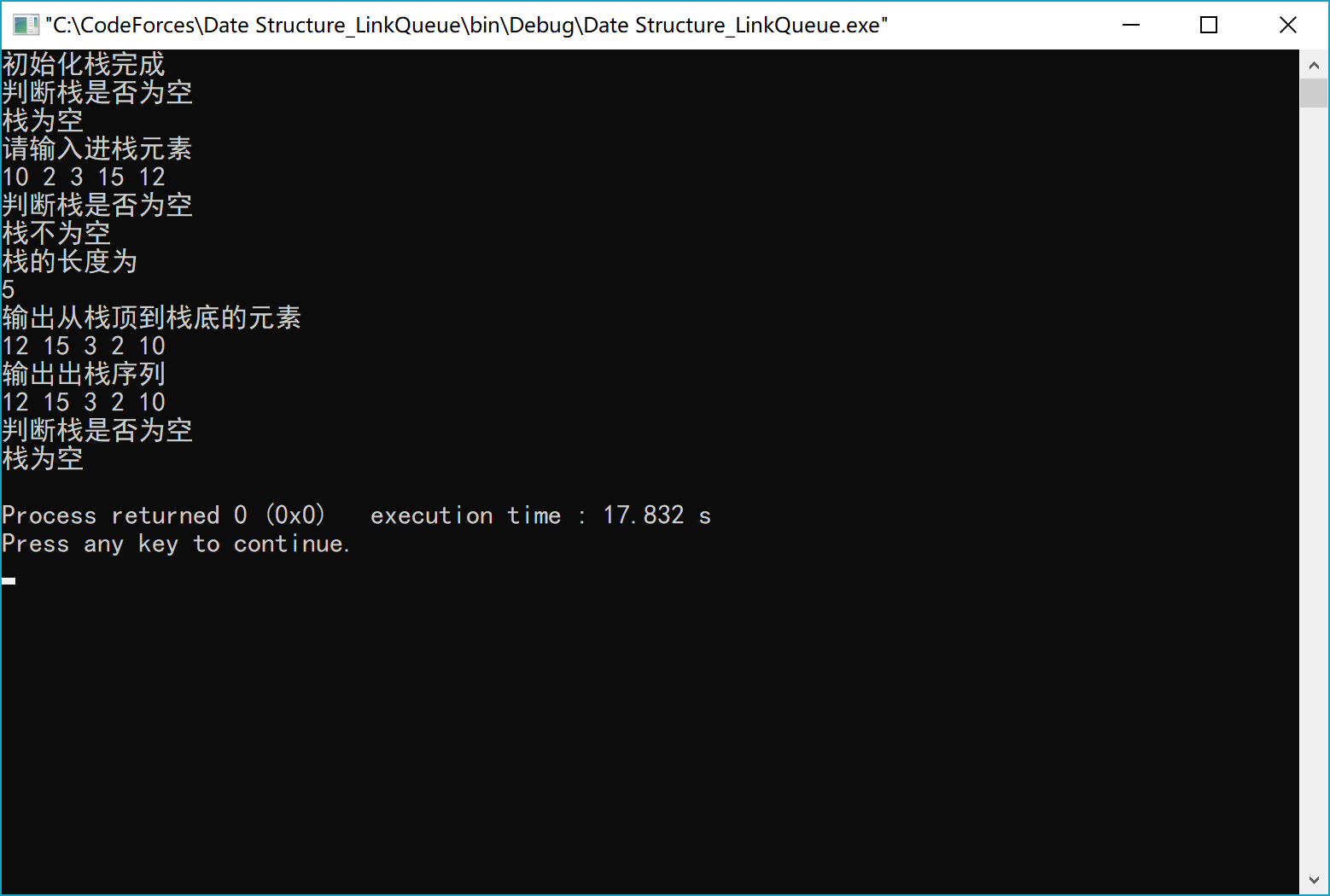
## 实验要求：

|  |
| --- |
| /\* |
|  | 编写一个程序，实现链栈的各种基本运算，并在此基础上设计一个主程序完成如下功能： |
|  | （1）初始化栈s； |
|  | （2）判断栈s是否为空； |
|  | （3）依次进栈元素10,-2,10,-3,15,12；10 2 3 15 12 |
|  | （4）判断栈s是否为空； |
|  | （5）输出栈长度； |
|  | （6）输出从栈顶到栈底的元素； |
|  | （7）输出出栈序列； |
|  | （8）判断栈s是否为空。 |
|  | \*/ |

## 源代码

|  |
| --- |
|  |
|  | #include <bits/stdc++.h> |
|  | using namespace std; |
|  | #define OK 1 |
|  | #define ERROR -1 |
|  | typedef struct StackNode |
|  | { |
|  | int data; |
|  | struct StackNode \*next; |
|  | }StackNode,\*LinkStack; |
|  | typedef int Status; |
|  | Status InitStack(LinkStack &S) |
|  | { |
|  | S=NULL; |
|  | return OK; |
|  | } |
|  | Status Push(LinkStack &S,int e) |
|  | { |
|  | LinkStack p; |
|  | p=new StackNode; |
|  | p->data=e; |
|  | p->next=S; |
|  | S=p; |
|  | return OK; |
|  | } |
|  | Status Pop(LinkStack &S,int &e) |
|  | { |
|  | if(S==NULL) |
|  | return ERROR; |
|  | LinkStack p; |
|  | e=S->data; |
|  | p=S; |
|  | S=S->next; |
|  | delete p; |
|  | return OK; |
|  | } |
|  | int GetHead(LinkStack S) |
|  | { |
|  | if(S!=NULL) |
|  | return S->data; |
|  | } |
|  | void IsEmpty(LinkStack S) |
|  | { |
|  | if(S==NULL) |
|  | cout<<"栈为空"<<endl; |
|  | else |
|  | cout<<"栈不为空"<<endl; |
|  | } |
|  | int StackLength(LinkStack S) |
|  | { |
|  | int j=0; |
|  | LinkStack p; |
|  | p=S; |
|  | while(p) |
|  | { |
|  | j++; |
|  | p=p->next; |
|  | } |
|  | return j; |
|  | } |
|  | void CreatStack(LinkStack &S) |
|  | { |
|  | for(int i=0;i<5;i++) |
|  | { |
|  | int e; |
|  | cin>>e; |
|  | Push(S,e); |
|  | } |
|  | } |
|  | void Traverse(LinkStack S) |
|  | { |
|  | if(S==NULL) |
|  | return ; |
|  | else |
|  | { |
|  | cout<<S->data<<" "; |
|  | Traverse(S->next); |
|  | } |
|  | } |
|  | void PopAll(LinkStack &S) |
|  | { |
|  | int e; |
|  | while(S!=NULL) |
|  | { |
|  | Pop(S,e); |
|  | cout<<e<<" "; |
|  | } |
|  | cout<<endl; |
|  | } |
|  | int main() |
|  | { |
|  | LinkStack S; |
|  | InitStack(S); |
|  |  |
|  | cout<<"初始化栈完成"<<endl; |
|  |  |
|  | cout<<"判断栈是否为空"<<endl; |
|  | IsEmpty(S); |
|  |  |
|  | cout<<"请输入进栈元素"<<endl; |
|  | CreatStack(S); |
|  |  |
|  | cout<<"判断栈是否为空"<<endl; |
|  | IsEmpty(S); |
|  |  |
|  | cout<<"栈的长度为"<<endl; |
|  | cout<<StackLength(S)<<endl; |
|  |  |
|  | cout<<"输出从栈顶到栈底的元素"<<endl; |
|  | Traverse(S); |
|  | cout<<endl; |
|  |  |
|  | cout<<"输出出栈序列"<<endl; |
|  | PopAll(S); |
|  |  |
|  | cout<<"判断栈是否为空"<<endl; |
|  | IsEmpty(S); |
|  |  |
|  | return 0; |
|  | } |

## 运行结果截图



# 循环队列的相关操作

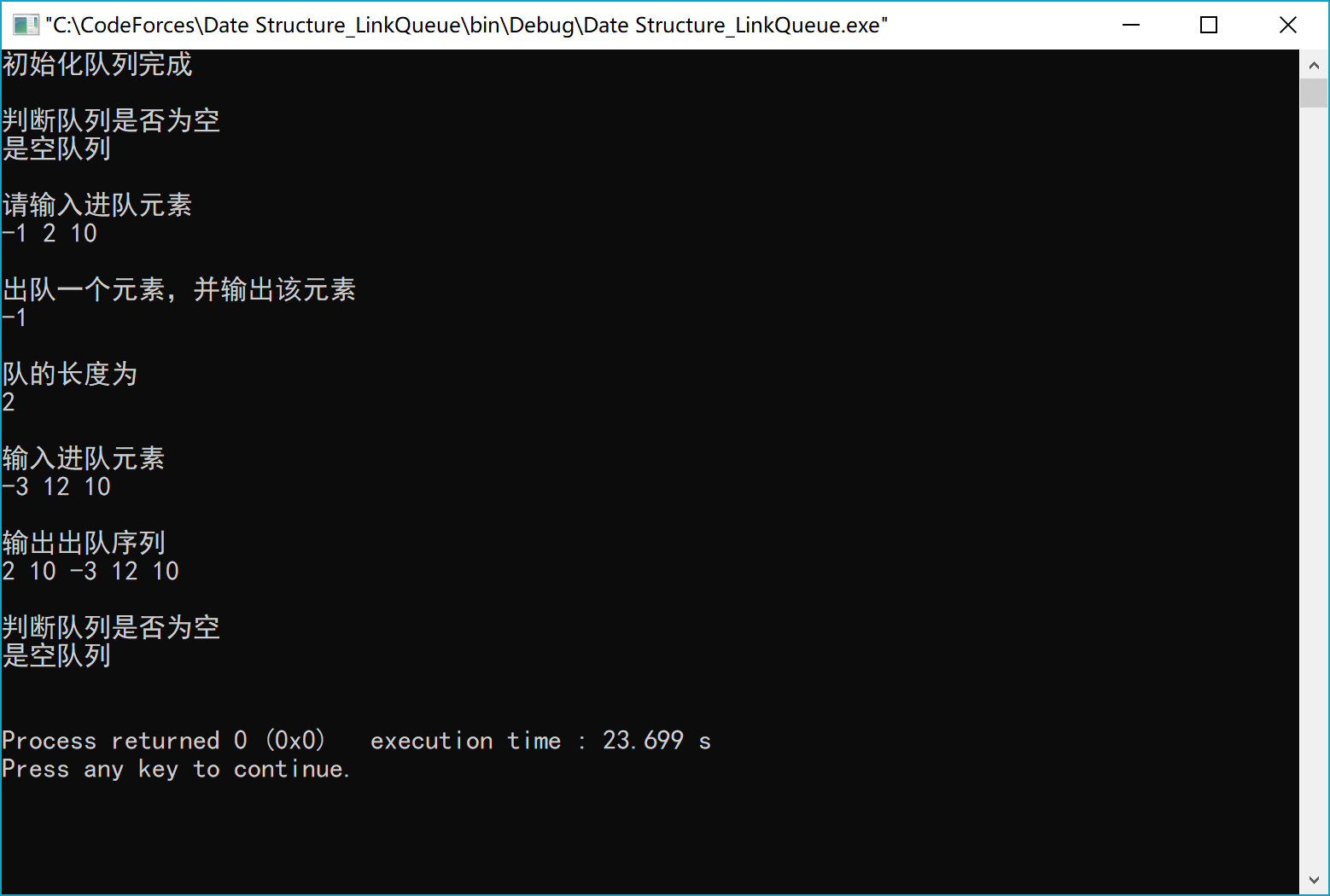
## 实验要求：

|  |
| --- |
| /\* |
|  | 编写一个程序，实现循环队列的各种基本运算，并在此基础上设计一个主程序完成如下功能： |
|  | （1）初始化队列q； |
|  | （2）判断队列q是否为空； |
|  | （3）依次进队列元素-1,2,10； |
|  | （4）出队一个元素，并输出该元素； |
|  | （5）输出队列的长度（元素个数）； |
|  | （6）依次进队元素-3,12,10； |
|  | （7）输出出队序列； |
|  | （8）判断队列q是否为空。 |
|  | \*/ |

## 源代码

|  |  |
| --- | --- |
|  | #include <bits/stdc++.h> |
|  | using namespace std; |
|  | #define OK 1 |
|  | #define ERROR -1 |
|  | #define MAXSIZE 100 |
|  | typedef int Status; |
|  | typedef struct |
|  | { |
|  | int \*base; |
|  | int Front; |
|  | int rear; |
|  | }SqQueue; |
|  | Status InitQueue(SqQueue &Q) |
|  | { |
|  | Q.base=new int[MAXSIZE]; |
|  | if(!Q.base) |
|  | exit(OVERFLOW); |
|  | Q.Front=Q.rear=0; |
|  | return OK; |
|  | } |
|  | int QueueLength(SqQueue Q) |
|  | { |
|  | return (Q.rear-Q.Front+MAXSIZE)%MAXSIZE; |
|  | } |
|  | Status EnQueue(SqQueue &Q,int e) |
|  | { |
|  | if((Q.rear+1)%MAXSIZE==Q.Front) |
|  | return ERROR; |
|  | Q.base[Q.rear]=e; |
|  | Q.rear=(Q.rear+1)%MAXSIZE; |
|  | return OK; |
|  | } |
|  | int DeQueue(SqQueue &Q) |
|  | { |
|  | if(Q.Front==Q.rear) |
|  | return ERROR; |
|  | int e=Q.base[Q.Front]; |
|  | Q.Front=(Q.Front+1)%MAXSIZE; |
|  | return e; |
|  | } |
|  | int GetHead(SqQueue &Q) |
|  | { |
|  | if(Q.Front!=Q.rear) |
|  | return Q.base[Q.Front]; |
|  | } |
|  | void IsEmpty(SqQueue Q) |
|  | { |
|  | if(Q.Front==Q.rear) |
|  | cout<<"是空队列"<<endl; |
|  | else |
|  | cout<<"不是空队列"<<endl; |
|  | } |
|  | void Creat\_SqQueue(SqQueue &Q,int n) |
|  | { |
|  | int e,i; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | cin>>e; |
|  | EnQueue(Q,e); |
|  | } |
|  | } |
|  | void Traverse(SqQueue &Q) |
|  | { |
|  | while(Q.Front!=Q.rear) |
|  | cout<<DeQueue(Q)<<" "; |
|  | cout<<endl; |
|  | } |
|  | int main() |
|  | { |
|  | SqQueue Q; |
|  | cout<<"初始化队列完成"<<endl; |
|  | InitQueue(Q);cout<<endl; |
|  |  |
|  | cout<<"判断队列是否为空"<<endl; |
|  | IsEmpty(Q);cout<<endl; |
|  |  |
|  | cout<<"请输入进队元素"<<endl; |
|  | Creat\_SqQueue(Q,3);cout<<endl; |
|  |  |
|  | cout<<"出队一个元素，并输出该元素"<<endl; |
|  | cout<<DeQueue(Q)<<endl;cout<<endl; |
|  |  |
|  | cout<<"队的长度为"<<endl; |
|  | cout<<QueueLength(Q)<<endl;cout<<endl; |
|  |  |
|  | cout<<"输入进队元素"<<endl; |
|  | Creat\_SqQueue(Q,3);cout<<endl; |
|  |  |
|  | cout<<"输出出队序列"<<endl; |
|  | Traverse(Q);cout<<endl; |
|  |  |
|  | cout<<"判断队列是否为空"<<endl; |
|  | IsEmpty(Q);cout<<endl; |
|  | return 0; |
|  | } |

## 运行结果截图



# 链队列的相关操作

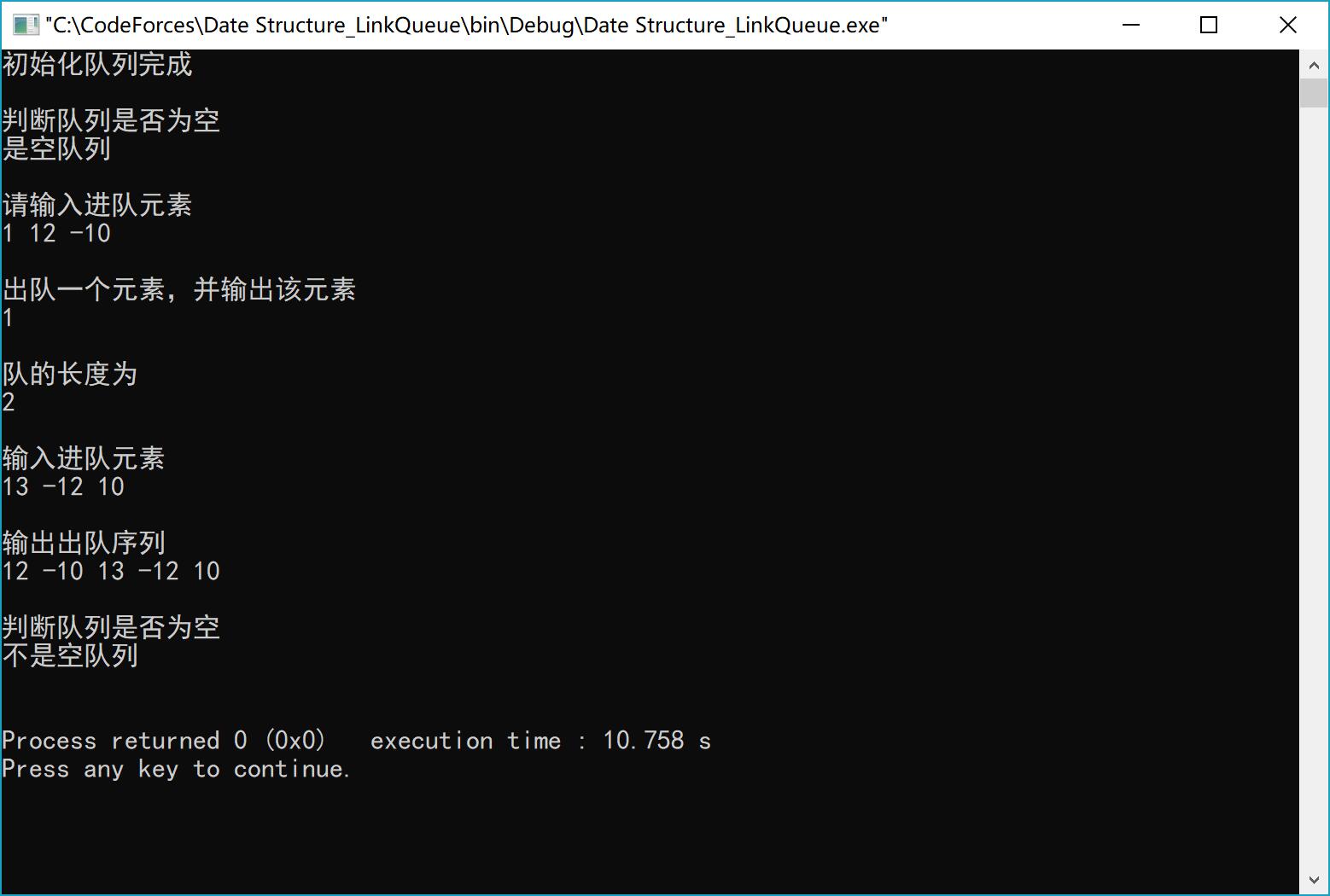
## 实验要求：

|  |
| --- |
| /\* |
|  | 编写一个程序，实现链队列的各种基本运算，并在此基础上设计一个主程序完成如下功能： |
|  | （1）初始化队列q； |
|  | （2）判断队列q是否为空； |
|  | （3）依次进队列元素1,12,-10； |
|  | （4）出队一个元素，并输出该元素； |
|  | （5）输出队列的长度（元素个数）； |
|  | （6）依次进队元素13,-12,10； |
|  | （7）输出队列长度； |
|  | （7）输出出队序列。 |
|  | \*/ |

## 源代码

|  |
| --- |
|  |
|  | #include <bits/stdc++.h> |
|  | #define MAXSIZE 100 |
|  | #define ERROR -1 |
|  | #define OK 1 |
|  | using namespace std; |
|  | typedef int Status; |
|  | typedef struct QNode |
|  | { |
|  | int data; |
|  | struct QNode \*next; |
|  | }QNode,\*QueuePtr; |
|  | typedef struct |
|  | { |
|  | QueuePtr Front; |
|  | QueuePtr rear; |
|  | }LinkQueue; |
|  | Status InitQueue(LinkQueue &Q) |
|  | { |
|  | Q.Front=Q.rear=new QNode; |
|  | Q.Front->next=NULL; |
|  | return OK; |
|  | } |
|  | Status EnQueue(LinkQueue &Q,int e) |
|  | { |
|  | QueuePtr p; |
|  | p=new QNode; |
|  | p->data=e; |
|  | p->next=NULL; |
|  | Q.rear->next=p; |
|  | Q.rear=p; |
|  | return OK; |
|  | } |
|  | void DeQueue(LinkQueue &Q) |
|  | {//删除Q的队头元素，用e返回值 |
|  | int e; |
|  | QueuePtr p; |
|  | p=Q.Front->next; |
|  | e=p->data; |
|  | Q.Front->next=p->next; |
|  | if(Q.rear==p) |
|  | Q.rear=Q.Front; |
|  | delete p; |
|  | cout<<e<<endl; |
|  | } |
|  | int GetHead(LinkQueue Q) |
|  | { |
|  | if(Q.Front!=Q.rear) |
|  | return Q.Front->next->data; |
|  | } |
|  | void IsEmpty(LinkQueue Q) |
|  | { |
|  | if(Q.Front==Q.rear) |
|  | cout<<"是空队列"<<endl; |
|  | else |
|  | cout<<"不是空队列"<<endl; |
|  | } |
|  | void Creat\_linkQueue(LinkQueue &Q,int n) |
|  | { |
|  | for(int i=0;i<n;i++) |
|  | { |
|  | int e; |
|  | cin>>e; |
|  | EnQueue(Q,e); |
|  | } |
|  | } |
|  | void Traverse(LinkQueue Q) |
|  | { |
|  | QueuePtr p; |
|  | p=Q.Front; |
|  | while(p!=Q.rear) |
|  | { |
|  | p=p->next; |
|  | cout<<p->data<<" "; |
|  | } |
|  | cout<<endl; |
|  | } |
|  | int QueueLength(LinkQueue Q) |
|  | { |
|  | QueuePtr p; |
|  | p=Q.Front; |
|  | int j=0; |
|  | while(p!=Q.rear) |
|  | { |
|  | j++; |
|  | p=p->next; |
|  | } |
|  | return j; |
|  | } |
|  | int main() |
|  | { |
|  | LinkQueue Q; |
|  | cout<<"初始化队列完成"<<endl; |
|  | InitQueue(Q);cout<<endl; |
|  |  |
|  | cout<<"判断队列是否为空"<<endl; |
|  | IsEmpty(Q);cout<<endl; |
|  |  |
|  | cout<<"请输入进队元素"<<endl; |
|  | Creat\_linkQueue(Q,3);cout<<endl; |
|  |  |
|  | cout<<"出队一个元素，并输出该元素"<<endl; |
|  | DeQueue(Q);cout<<endl; |
|  |  |
|  | cout<<"队的长度为"<<endl; |
|  | cout<<QueueLength(Q)<<endl;cout<<endl; |
|  |  |
|  | cout<<"输入进队元素"<<endl; |
|  | Creat\_linkQueue(Q,3);cout<<endl; |
|  |  |
|  | cout<<"输出出队序列"<<endl; |
|  | Traverse(Q);cout<<endl; |
|  |  |
|  | cout<<"判断队列是否为空"<<endl; |
|  | IsEmpty(Q);cout<<endl; |
|  | return 0; |
|  | } |
|  |  |

## 运行结果截图



# 实现进制转换（栈）

## 实验要求

|  |
| --- |
| /\* |
|  | 借助栈实现数制转换。要求将非负十进制整数N转换为二进制输出。 |
|  | \*/ |

## 源代码

|  |  |
| --- | --- |
|  | #include <bits/stdc++.h> |
|  | using namespace std; |
|  | #define MAXSIZE 100 |
|  | #define ERROR -1 |
|  | #define OK 1 |
|  | typedef struct |
|  | { |
|  | int \*base; |
|  | int \*top; |
|  | int stacksize; |
|  | }SqStack; |
|  | typedef int Status; |
|  | Status InitStack(SqStack &S) |
|  | { |
|  | S.base=new int [MAXSIZE]; |
|  | if(!S.base) |
|  | exit(OVERFLOW); |
|  | S.top=S.base; |
|  | S.stacksize=MAXSIZE; |
|  | return OK; |
|  | } |
|  | Status Push(SqStack &S,int e) |
|  | { |
|  | if(S.top-S.base==S.stacksize) |
|  | return ERROR; |
|  | \*S.top++=e; |
|  | return OK; |
|  | } |
|  | Status Pop(SqStack &S,int &e) |
|  | { |
|  | if(S.top==S.base) |
|  | return ERROR; |
|  | e=\*--S.top; |
|  | return OK; |
|  | } |
|  | Status StackEmpty(SqStack S) |
|  | { |
|  | if(S.top==S.base) |
|  | return 1; |
|  | return 0; |
|  | } |
|  | void converse(int N) |
|  | { |
|  | SqStack S; |
|  | InitStack(S); |
|  | while(N) |
|  | { |
|  | Push(S,N%8); |
|  | N=N/8; |
|  | } |
|  | while(!StackEmpty(S)) |
|  | { |
|  | int e; |
|  | Pop(S,e); |
|  | cout<<e; |
|  | } |
|  | } |
|  | int main() |
|  | { |
|  | cout<<"请输入需要进制转换的数字"<<endl;cout<<endl; |
|  |  |
|  | int N; |
|  |  |
|  | cin>>N; |
|  |  |
|  | converse(N); |
|  |  |
|  | return 0; |
|  | } |

## 运行结果截图

