计算机与信息工程学院实验报告

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

姓名：杨佳森 学号：\_2112080106 专业：数据科学与大数据技术年级：\_2021\_

课程： 数据结构 主讲教师：\_刘成\_\_\_\_ 辅导教师：\_ 无\_\_\_\_\_\_

实验时间：2023\_年 4月 11\_日 下\_午\_17\_时至\_19\_时，实验地点\_计算机大楼213\_

实验题目： 链队列的基本操作

实验目的： 通过该实验，使学生理解链队列的构造特点并灵活应用，掌握链队基本操作的编程实现，认识栈是在一端进行插入，在另一端进行删除集中操作的线性结构，掌握队列的“先入先出”操作特点，知道判断队列空和满的条件，进一步熟悉C语言中指针操作。

实验环境（硬件和软件） PC和vscode

实验内容：

用链式存储结构，实现教材定义的队列的基本操作。

实验步骤：

1、初始化队列

void Init(LinkQueue q) {

q.front = q.rear = nullptr;

cout << "初始化成功！" << endl;

}

2、销毁队列

void Destory(LinkQueue q) {

while(q.front) {

q.rear = q.front->next;

delete q.front;

q.front = q.rear;

}

cout << "已销毁队列！" << endl;

}

3、清空队列

int Clear(LinkQueue q) {

while(!Empty(q)) Del(q);

cout << "成功清空！" << endl;

return 0;

}

4、判断队列是否为空

bool Empty(LinkQueue q) {

return q.front == q.rear;

}

5、返回队列中元素个数

int GetLength(LinkQueue q) {

QueueNode \*p = q.front;

int length = 0;

while(p != nullptr) {

length++;

p = p->next;

}

return length;

}

6、返回队列队头元素

int GetTop(LinkQueue q) {

if(Empty(q)) {

cout << "该链队列为空！" << endl;

return 0;

}

return q.front->data;

}

7、插入新的队尾元素

void Add(LinkQueue q, int data) {

QueueNode \*p = new QueueNode;

p->data = data;

p->next = nullptr;

if(Empty(q)) q.front = q.rear = p;

else {

q.rear->next = p;

q.rear = p;

}

}

8、删除队头元素

int Del(LinkQueue q) {

//先判断队列是否为空

if(Empty(q)) {

cout << "该链队列为空！" << endl;

return 0;

}

QueueNode \*p = q.front;

int data = p->data;

q.front = q.front->next;

if(q.front == nullptr) q.rear = nullptr;

delete p;

return data;

}

9、初始化并创建队列

void foundqueue(LinkQueue q) {

Init(q);

int n;

cout << "请输入元素个数：";

cin >> n;

cout << "请输入元素：";

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

int data;

cin >> data;

Add(q, data);

}

}

10、输出队列元素

void traverse(LinkQueue q) {

int length = GetLength(q);

if(length > 0) {

cout << "链队列的元素为: ";

QueueNode \*p = q.front->next;

while(p) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

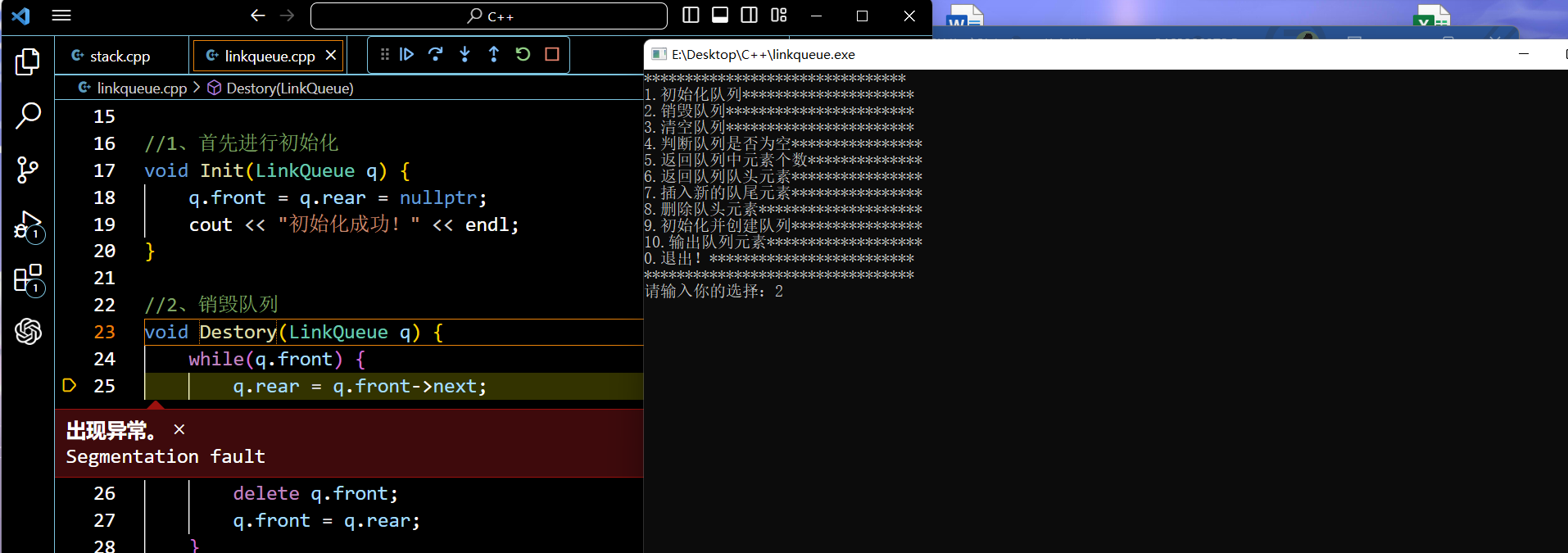
}

else cout << "顺序栈为空！" << endl;

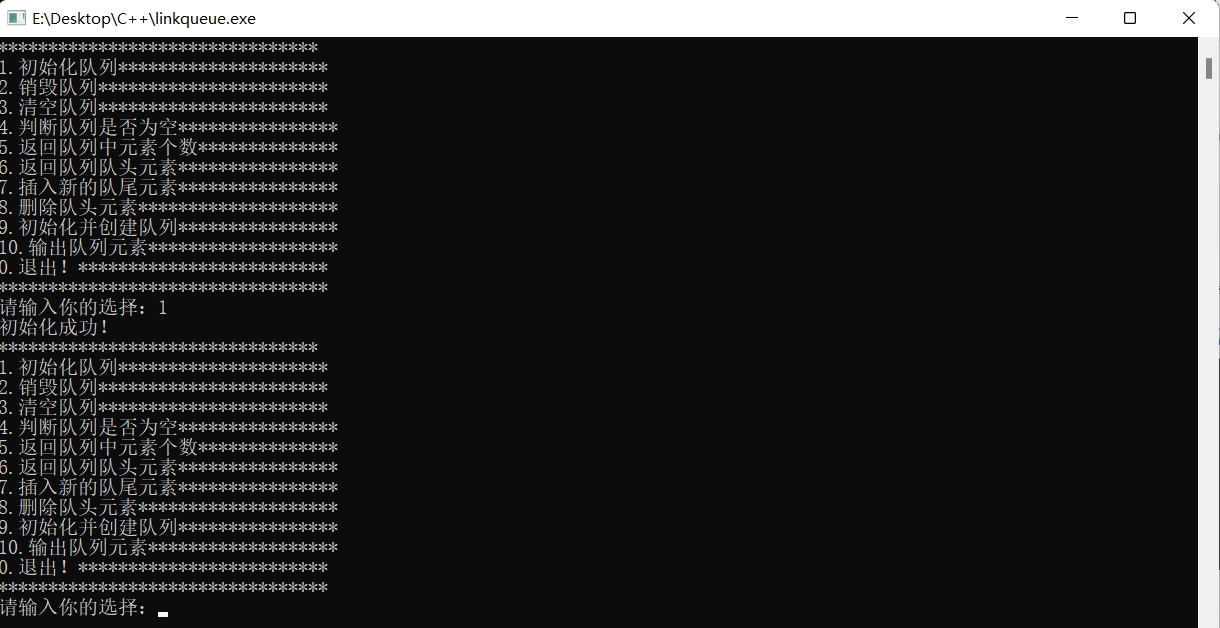
}

实验数据记录：

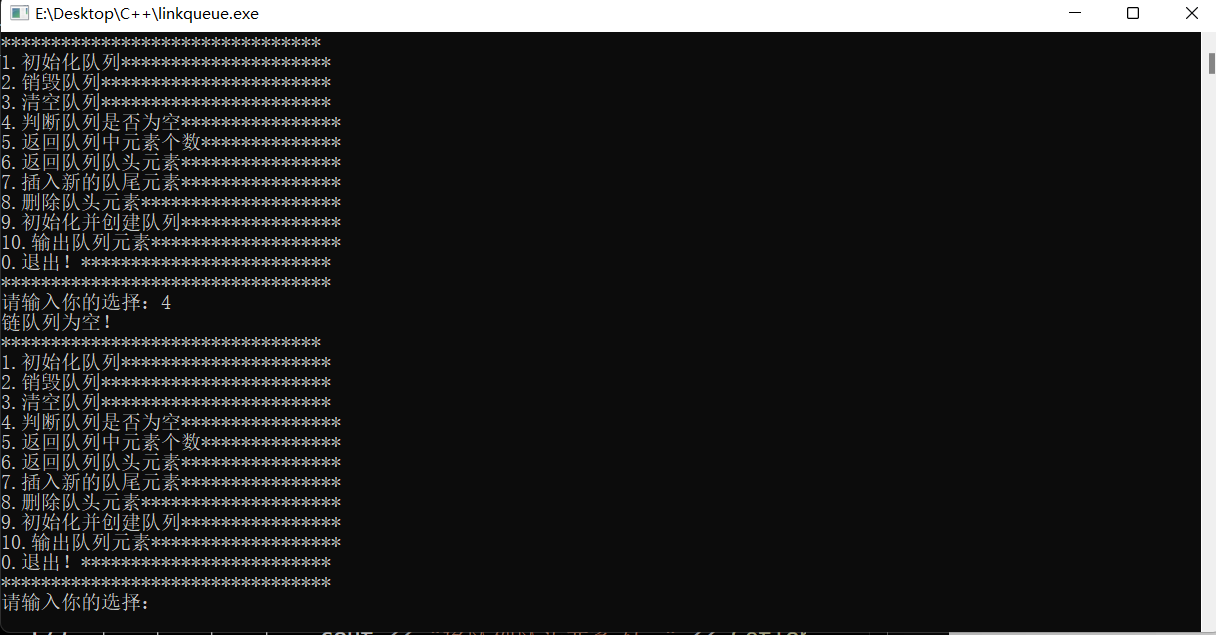
1、没有进行初始化前进行其他操作，程序是否能控制住：



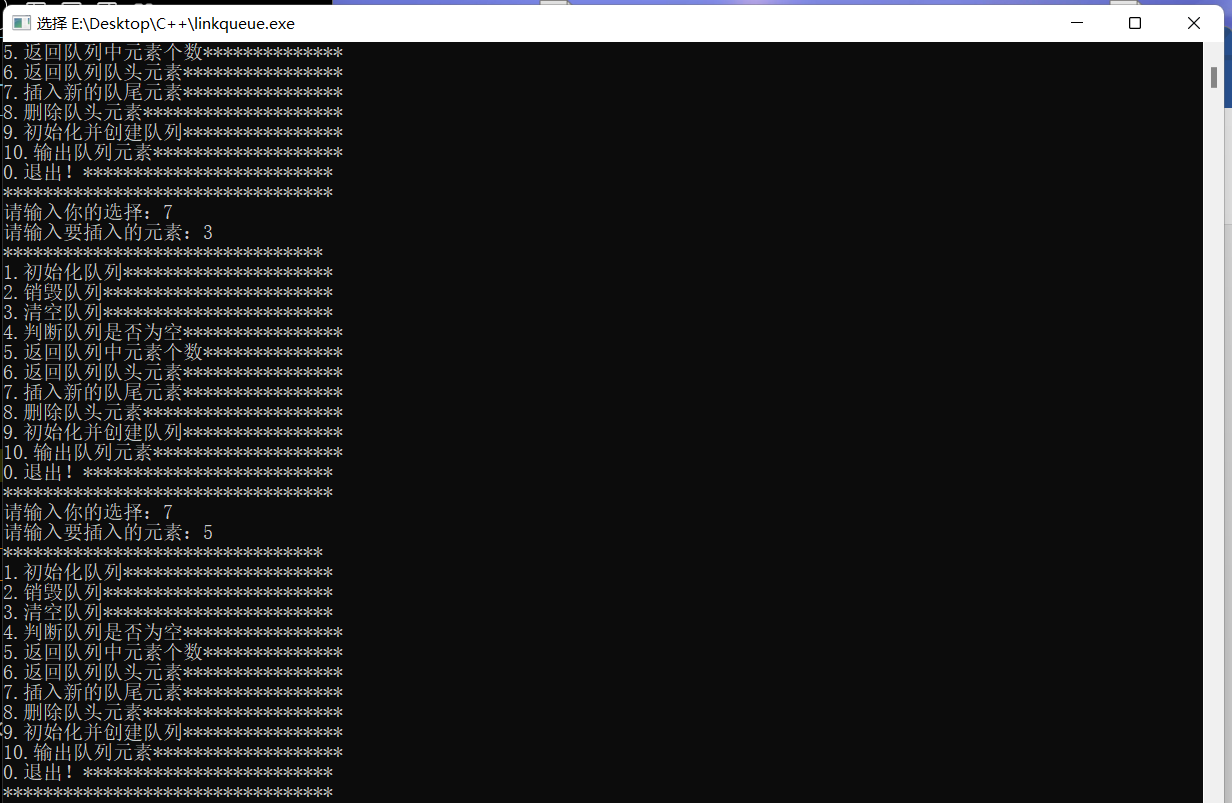
2、初始化一个队列



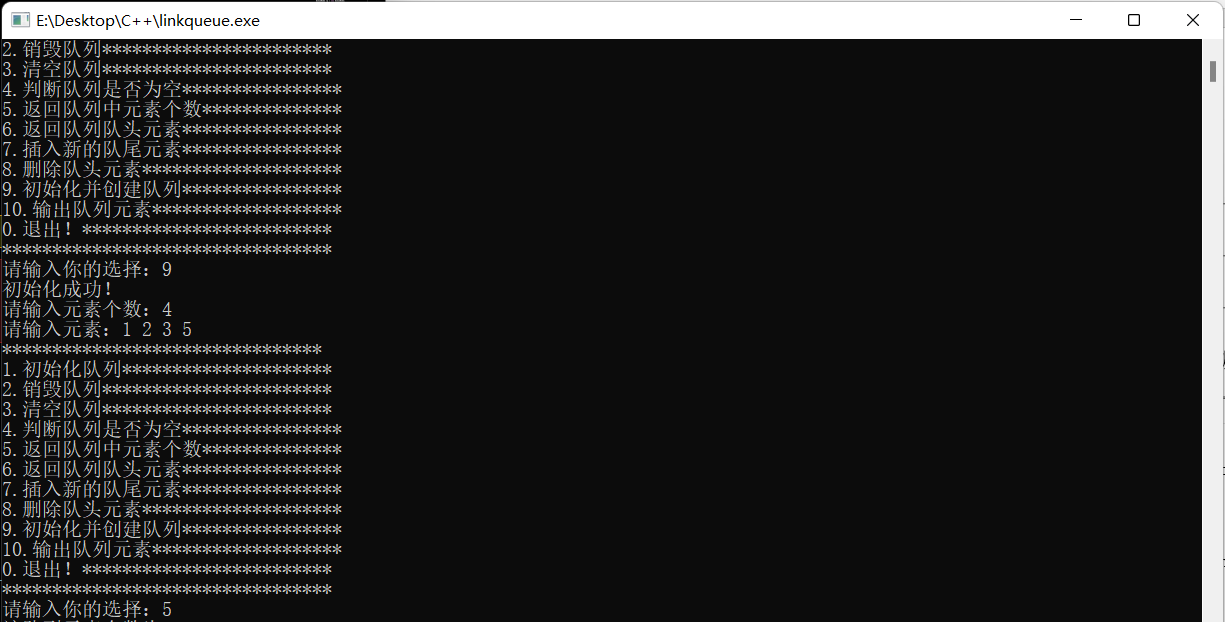
3、判断队列是否为空：



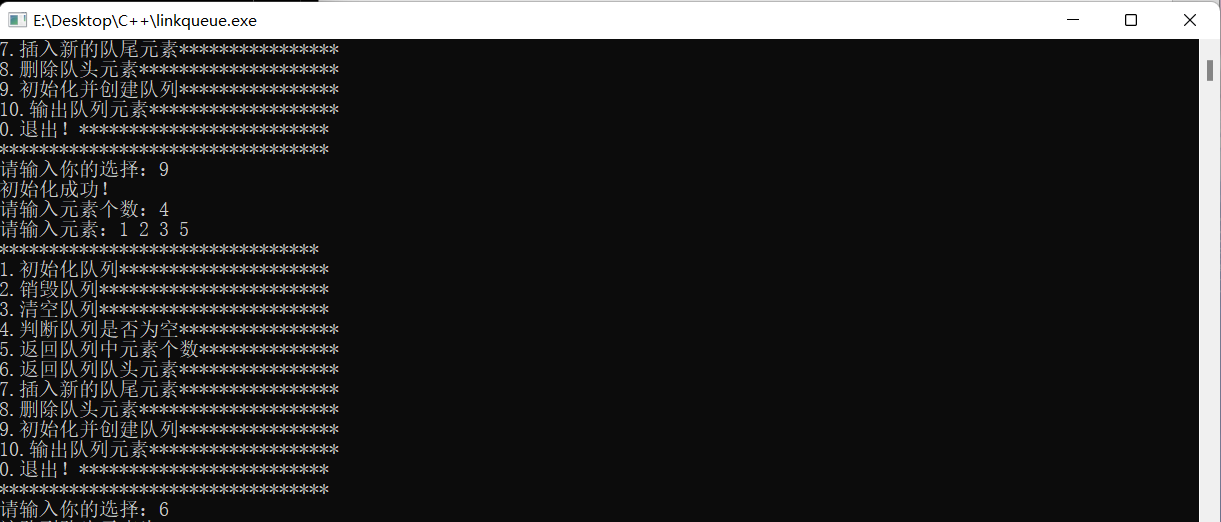
4、4个数入队，1，2，3，5；



5、输出队列长度



6、取队头元素，在判断栈是否为空，在判断栈长度



7、出队，在判断栈长度



8、销毁栈，在做其他操作，判断程序是否能控制



问题讨论：

1、队列可以从头部或者尾部进行遍历，但不能同时遍历，无需开辟空间，因为在遍历的过程中不影响数据结构。

源代码：



#include <iostream>

using namespace std;

typedef struct Node {

int data; //节点数据域

struct Node \* next; //节点指针域

}QueueNode;

//声明一个结构体来定义上面这个结构体的两个指针

typedef struct {

QueueNode \*front;

QueueNode \*rear;//分别指向队首和队尾的指针

}LinkQueue;

//1、首先进行初始化

void Init(LinkQueue q) {

q.front = q.rear = nullptr;

cout << "初始化成功！" << endl;

}

//2、销毁队列

void Destory(LinkQueue q) {

while(q.front) {

q.rear = q.front->next;

delete q.front;

q.front = q.rear;

}

cout << "已销毁队列！" << endl;

}

//4、判断队列是否为空

bool Empty(LinkQueue q) {

return q.front == q.rear;

}

//5、返回队列中元素个数

int GetLength(LinkQueue q) {

QueueNode \*p = q.front;

int length = 0;

while(p != nullptr) {

length++;

p = p->next;

}

return length;

}

//6、返回队列队头元素

int GetTop(LinkQueue q) {

if(Empty(q)) {

cout << "该链队列为空！" << endl;

return 0;

}

return q.front->data;

}

//7、插入新的队尾元素

void Add(LinkQueue q, int data) {

QueueNode \*p = new QueueNode;

p->data = data;

p->next = nullptr;

if(Empty(q)) q.front = q.rear = p;

else {

q.rear->next = p;

q.rear = p;

}

}

//8、删除队头元素

int Del(LinkQueue q) {

//先判断队列是否为空

if(Empty(q)) {

cout << "该链队列为空！" << endl;

return 0;

}

QueueNode \*p = q.front;

int data = p->data;

q.front = q.front->next;

if(q.front == nullptr) q.rear = nullptr;

delete p;

return data;

}

//9、初始化并创建队列

void foundqueue(LinkQueue q) {

Init(q);

int n;

cout << "请输入元素个数：";

cin >> n;

cout << "请输入元素：";

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

int data;

cin >> data;

Add(q, data);

}

}

//10、输出队列元素

void traverse(LinkQueue q) {

int length = GetLength(q);

if(length > 0) {

cout << "链队列的元素为: ";

QueueNode \*p = q.front->next;

while(p) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

else cout << "顺序栈为空！" << endl;

}

//3、清空队列

int Clear(LinkQueue q) {

while(!Empty(q)) Del(q);

cout << "成功清空！" << endl;

return 0;

}

void show() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1.初始化队列\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "2.销毁队列\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "3.清空队列\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "4.判断队列是否为空\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "5.返回队列中元素个数\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "6.返回队列队头元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "7.插入新的队尾元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "8.删除队头元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "9.初始化并创建队列\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "10.输出队列元素\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "0.退出！\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

int main()

{

int choice;

LinkQueue q;

while(1)

{

show();

cout << "请输入你的选择：";

cin >> choice;

switch(choice)

{

case 1:

Init(q);

break;

case 2:

Destory(q);

break;

case 3:

Clear(q);

break;

case 4:

if(Empty(q)) cout << "链队列为空！" << endl;

else cout << "链队列不为空！" << endl;

break;

case 5:

cout << "该队列元素个数为：" << GetLength(q) << endl;

break;

case 6:

cout << "该队列队头元素为：" << GetTop(q) << endl;

break;

case 7:

int data;

cout << "请输入要插入的元素：";

cin >> data;

Add(q, data);

break;

case 8:

cout << "已删除队头元素，该元素为：" << Del(q) << endl;

break;

case 9:

foundqueue(q);

break;

case 10:

traverse(q);

break;

case 0:

exit(0);

break;

default:

cout << "输入有误，请重新输入！" << endl;

break;

}

}

}