西安财经学院 信息学院

姓名：程驰

学号：1931054041

班级：计本1902

指导教师：李薇

成绩：

数据结构 实验报告

实验名称 队列的基本操作

实验日期：2020年10月16日

**一、实验目的：**

1．复习C语言程序设计中的知识。

2．掌握队列的顺序和链式存储结构的表示和实现方法。

3．掌握队列的基本操作的算法实现。

**二、实验内容：**

1. 建立循环队列。

2. 在循环队列上实现入队、出队等操作。

3. 建立链队列。

4. 在链队列上实现入队、出队等操作。

5. 用队列实现杨辉三角形的打印。（选做）

三、实验要求：

1.编写实现循环队列的基本算法的函数，并在此基础上设计一个主程序完成如下功能：

⑴初始化循环队列Q（数据类型任选）;

⑵入队；

⑶出队；

⑷求队列的元素个数；

⑸遍历输出循环队列Q；

⑹置空队列。

2. 编写实现顺序栈的基本算法（初始化、入栈、出栈、取栈顶元素、遍历、置空等）的函数，并在此基础上设计一个主程序完成如下功能：

⑴初始化链队列Q（数据类型任选）;

⑵入队；

⑶出队；

⑷求队列的元素个数；

⑸遍历输出循环队列Q；

⑹置空队列。

3. 任选一种队列的存储结构，实现杨辉三角形的打印。（选做）

四、实验步骤：

1.循环队列操作

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

#define MAXSIZE 10

typedef struct{

int data[MAXSIZE];

int front,rear;

}CSeQueue;

//队列的初始化

CSeQueue \*Init\_Queue(){

CSeQueue \*q;

q = (CSeQueue \*)malloc(sizeof(CSeQueue));

q->front = q->rear = 0;

return q;

}

//判队满

int Full\_Queue(CSeQueue \*q){

if((q ->rear + 1) % MAXSIZE == q ->front){

return 1;

}else

return 0;

}

//入队

int In\_Queue(CSeQueue \*q,int x){

if(Full\_Queue(q)){

printf("队满");

return 0;

}

else{

q ->rear = (q ->rear + 1) % MAXSIZE;

q ->data[q ->rear] = x;

return 1;

}

}

// 判队空

int Empty\_Queue(CSeQueue \*q){

if(q ->front == q ->rear)

return 1;

else

return 0;

}

//出队

int Out\_Queue(CSeQueue \*q,int \*x){

if(Empty\_Queue(q)){

printf("队空");

return 0;

}

else {

q ->front = (q ->front + 1) % MAXSIZE;

\*x = q ->data[q ->front];

return 1;

}

}

//打印输出队列中的元素

void printf\_Queue(CSeQueue \*q){

int i = q ->front;

printf("队列中的元素是:");

while(i != q ->rear){

printf("%d ",q ->data[i+1]);

i ++;

i = i % MAXSIZE;

}

}

//求队列的长度

int Queue\_Length(CSeQueue \*q){

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

}

int main(){

//队列初始化

CSeQueue \*q;

q = Init\_Queue();

//插入第一个元素

printf("将元素1插入到队尾后的结果:\n");

In\_Queue(q,1);

printf\_Queue(q);

printf("循环队列长度=%d\n",Queue\_Length(q));

// 插入第二个元素

printf("将元素2插入到队尾后的结果:\n");

In\_Queue(q,2);

printf\_Queue(q);

printf("循环队列长度=%d\n",Queue\_Length(q));

// 插入第三个元素

printf("将元素3插入到队尾后的结果:\n");

In\_Queue(q,3);

printf\_Queue(q);

printf("循环队列长度=%d\n",Queue\_Length(q));

// 删除第一个元素

int a;

Out\_Queue(q,&a);

printf("出队的元素是:%d\n",a);

printf\_Queue(q);

printf("循环队列长度=%d\n",Queue\_Length(q));

}

2.链队列操作

typedef int ElemType;

typedef struct Queue{

struct Queue\*next;

int data;

}ListQueue; //接口

struct Queue\*front=NULL;//指向队列首元节点的指针

struct Queue\*rear=NULL;//指向队列尾节点的指针

int length=0; //控制队列长度，为了方便第一次入队

ListQueue \*createQueue() //初始化

{

ListQueue \*s=(ListQueue \*)malloc(sizeof(ListQueue));

if(!s)

{

printf("内存分配失败！");

exit(ERROR);

}

s->next=NULL;

length++;

return s;

}

void PushQueue(ListQueue \*s) //入队

{

ListQueue \*p=(ListQueue\*)malloc(sizeof(ListQueue));

if(!p) //创建失败

{

printf("内存分配失败！");

exit(ERROR);

}

printf("请输入你要数据的数据：");

int a;

scanf("%d",&a);

p->data=a;

p->next=NULL;

if(length==1) //第一次入队

{

front=rear=p;

s->next=front;

}

else //第n+1次入队（n>=1)

{

rear->next=p;

rear=p;

}

length++; //控制是否为第一次入队，初始化时length已经为1；

}

void PopQueue(ListQueue\*s) //出队

{

if(front==NULL||rear==NULL)

printf("该队列已空！");

int v;

v=front->data;

s->next=front->next;

free(front);

front=s->next;

printf("该出队的数据为：");

printf("%d",v);

}

void Display(ListQueue\*s) //遍历队列

{

ListQueue\*v=s->next;

while(v)

{

printf("%d->",v->data);

v=v->next;

}

}

void DestoryQueue(ListQueue \*s) //清空队列并且销毁头节点

{

ListQueue \*v=s->next;

while(v)

{

front=v->next;

free(v);

v=front;

}

printf("该队列已清空！");

free(s);

printf("\n");

printf("头节点释放成功！");

}

int main()

{

ListQueue \*t=createQueue();

PushQueue(t);

PushQueue(t);

PushQueue(t);

printf("该数据如下");

Display(t);

printf("\n");

printf("出队！");

PopQueue(t);

printf("\n");

printf("余下数据如下：");

printf("\n");

Display(t);

printf("\n");

DestoryQueue(t);

return 0;

}

2.链队列操作

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<iostream>

#include<cstdio>

#include<cstdlib>

#define ERROR 0

using namespace std;

typedef int ElemType;

typedef struct Queue{

struct Queue\*next;

int data;

}ListQueue; //接口

struct Queue\*front=NULL;//指向队列首元节点的指针

struct Queue\*rear=NULL;//指向队列尾节点的指针

int length=0; //控制队列长度，为了方便第一次入队

ListQueue \*createQueue() //初始化

{

ListQueue \*s=(ListQueue \*)malloc(sizeof(ListQueue));

if(!s)

{

printf("内存分配失败！");

exit(ERROR);

}

s->next=NULL;

length++;

return s;

}

void PushQueue(ListQueue \*s) //入队

{

ListQueue \*p=(ListQueue\*)malloc(sizeof(ListQueue));

if(!p) //创建失败

{

printf("内存分配失败！");

exit(ERROR);

}

printf("请输入需要的数据：");

int a;

scanf("%d",&a);

p->data=a;

p->next=NULL;

if(length==1) //第一次入队

{

front=rear=p;

s->next=front;

}

else //第n+1次入队（n>=1)

{

rear->next=p;

rear=p;

}

length++; //控制是否为第一次入队，初始化时length已经为1；

}

void PopQueue(ListQueue\*s) //出队

{

if(front==NULL||rear==NULL)

printf("该队列已空！");

int v;

v=front->data;

s->next=front->next;

free(front);

front=s->next;

printf("该出队的数据为：");

printf("%d",v);

}

void Display(ListQueue\*s) //遍历队列

{

ListQueue\*v=s->next;

while(v)

{

printf("%d->",v->data);

v=v->next;

}

}

void DestoryQueue(ListQueue \*s) //清空队列并且销毁头节点

{

ListQueue \*v=s->next;

while(v)

{

front=v->next;

free(v);

v=front;

}

printf("该队列已清空！");

free(s);

printf("\n");

printf("头节点释放成功！");

}

int main()

{

ListQueue \*t=createQueue();

PushQueue(t);

PushQueue(t);

PushQueue(t);

printf("该数据如下");

Display(t);

printf("\n");

printf("出队！");

PopQueue(t);

printf("\n");

printf("余下数据如下：");

printf("\n");

Display(t);

printf("\n");

DestoryQueue(t);

return 0;

}

3.杨辉三角

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<iostream>

#include<cstdio>

#include<cstdlib>

#define MAXSIZE 100

typedef struct

{

int data[MAXSIZE];

int front,rear;

}SeqQueue,\*PSeqQueue;

//初始化

PSeqQueue Init\_SeqQueue()

{

PSeqQueue Q;

Q=(PSeqQueue)malloc(sizeof(SeqQueue));

if(Q)

{

Q->front=0;

Q->rear=0;

}

return Q;

}

//判空

int Empty\_SeqQueue(PSeqQueue Q)

{

if(Q&&Q->front==Q->rear)

return 1;

else

return 0;

}

//进队操作

int In\_SeqQueue(PSeqQueue Q,int x)

{

if((Q->rear+1)%MAXSIZE==Q->front)

{

printf("队满！");

return 0;

}

else

{

Q->rear=(Q->rear+1)%MAXSIZE;

Q->data[Q->rear]=x;

return 1;

}

}

//出队操作

int Out\_SeqQueue(PSeqQueue Q,int \*x)

{

if(Empty\_SeqQueue(Q))

{

printf("队空！");

return 0;

}

else

{

Q->front=(Q->front+1)%MAXSIZE;

\*x=Q->data[Q->front];

return 0;

}

}

//取队头元素

int Front\_SeqQueue(PSeqQueue Q,int \*x)

{

if(Q->front==Q->rear)

{

printf("队空！");

return -1;

}

else

{

\*x=Q->data[Q->front+1]%MAXSIZE;

return 1;

}

}

//销毁队列

void Destroy\_SeqQueue(PSeqQueue \*Q)

{

if(\*Q)

free(\*Q);

\*Q=NULL;

}

void yanghui\_trangle(int n)

{

PSeqQueue Q;

Q=(PSeqQueue)malloc(sizeof(SeqQueue));

if(Q)

{

Q->front=0;

Q->rear=0;

}

In\_SeqQueue(Q,1);

In\_SeqQueue(Q,1);//

int i;

for(i=1;i<=n;i++)

{

printf("\n");

for(int k=0;k<=40-4\*i;k+=2)

printf(" ");//控制输出格式

In\_SeqQueue(Q,0);

int t;

int s=0;

for(int j=1;j<=i+2;j++)

{

Out\_SeqQueue(Q,&t);

In\_SeqQueue(Q,s+t);//将前两项的和放入队尾

s=t;

if(j!=i+2)

printf("%4d",s);//遇到0的时候不输出，进行下一行

}

}

printf("\n");

Destroy\_SeqQueue(&Q);

}

int main()

{

printf("请输入要打印杨辉三角几行：");

int n;

scanf("%d",&n);

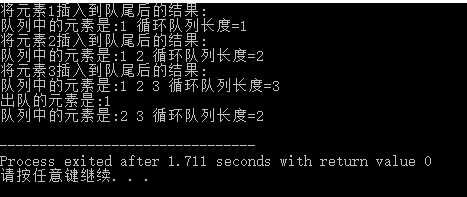
yanghui\_trangle(n);

return 0;

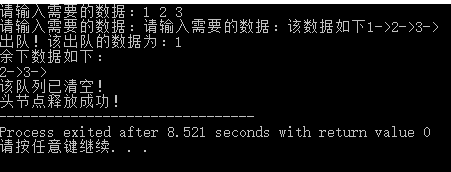
}

五、实验结果：

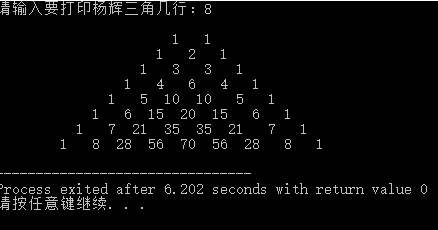
1.循环队列操作



2.链队列操作



3.杨辉三角



六、实验总结：

通过这次实验，我对队列的基本操作与简单应用认识更为深刻。