实验4队列的应用

[ 发布人： [易法令](https://www.scholat.com/flyi)   截止时间：2024-10-22 23:59:59  ]

一、实验目的

1．掌握队列的顺序存储结构

2．掌握队列先进先出运算原则在解决实际问题中的应用

二、实验内容

1.仿照教材顺序循环队列的例子，设计一个只使用队头指针和计数器的顺序循环队列抽象数据类型。其中操作包括:初始化、入队列、出队列、判断队列是否非空。编写主函数，验证所设计的顺序循环队列的正确性。

以下是队列操作函数的定义:

(1)QueueInitiate(Q) 初始化队列Q

(2)QueueNotEmpty(Q) 队列Q非空否

(3)QueueAppend(Q,x) 入队列，在队列Q的队尾插入数据元素x。

(4)QueueDelete(Q,d) 出队列，把队列Q的队头元素删除并由参数d带回。

提示:队尾的位置可由队头指针与计数器进行求解，请思考它们之间的关系，同时还要考虑如何实现循环队列（可借助求模运算）。

2.利用以上队列函数，编写算法（用函数表示算法）计算杨辉三角，并打印对应的数值（图形）。

三、实验源代码

typedef struct

{

    ElemType queue[MaxQueueSize];

    int front;   /\*队头指针\*/

    int count;  /\*计数器\*/

}SequenceQueue;

/\*初始化循环队列Q\*/

void QueueInitiate(SequenceQueue \*Q)

{

    Q->front = 0;

    Q->count = 0;

}

/\*判断队列是否为空\*/

int QueueNotEmpty(SequenceQueue Q)

{

    /\*空队列返回0，否则返回1\*/

    if (Q.count == 0)

    {

        return 0;

    }

    else

    {

        return 1;

    }

}

/\*入队列，在队尾插入元素x\*/

int QueueAppend(SequenceQueue \*Q, ElemType x)

{

    /\*成功入队返回1，否则返回0\*/

    if (Q->count == MaxQueueSize)

    {

        printf("Queue is Full!");

        return 0;

    }

    else

    {

        Q->queue[(Q->front + Q->count) % MaxQueueSize] = x;

        Q->count++;

        return 1;

    }

}

/\*出队列，把队列Q的队头元素删除并由参数d带回。\*/

int QueueDelete(SequenceQueue \*Q, ElemType \*d)

{

    /\*成功出队返回1，失败返回0\*/

    if (Q->count == 0)

    {

        printf("Queue is emtpy!");

        return 0;

    }

    else

    {

        \*d = Q->queue[Q->front];

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

        Q->count--;

        return 1;

    }

}

/\*获取队头元素并赋值给d\*/

int QueueGet(SequenceQueue \*Q, ElemType \*d)

{

    /\*成功返回1,失败返回0\*/

    if (Q->count == 0)

    {

        printf("Queue is emtpy, no data can get!");

        return 0;

    }

    else

    {

        \*d = Q->queue[Q->front];

        return 1;

    }

}

#include <stdio.h>

#define MaxQueueSize 100

typedef int ElemType;

#include "SequenceQueue.h"

void YangHuiTriangle(int n);

int main(void)

{

    /\*验证所设计的顺序循环队列的正确性\*/

    SequenceQueue Q;

    //初始化队列

    QueueInitiate(&Q);

    //入队列操作

    QueueAppend(&Q, 10);

    QueueAppend(&Q, 20);

    QueueAppend(&Q, 30);

    // 判断队列是否非空

    if (QueueNotEmpty(Q))

    {

        printf("Queue is not empty.\n");

    }

    else

    {

        printf("Queue is empty.\n");

    }

    //获取队头元素

    int a;

    QueueGet(&Q, &a);

    printf("Queue head num is %d\n", a);

    // 出队列操作

    int d;

    while (QueueNotEmpty(Q))

    {

        if (QueueDelete(&Q, &d))

        {

            printf("Dequeued: %d\n", d);

        }

    }

    // 判断队列是否非空

    if (QueueNotEmpty(Q))

    {

        printf("Queue is not empty.\n");

    }

    else

    {

        printf("Queue is empty.\n");

    }

    int N;

    printf("Input the num of line:");

    scanf("%d", &N);    /\*用户输入需要打印杨辉三角的行数\*/

    YangHuiTriangle(N);

    return 0;

}

void YangHuiTriangle(int n)

{

    int temp, x;    /\*temp用于临时存储队列中的元素，x用于获取队列中的元素\*/

    SequenceQueue myQueue;

    QueueInitiate(&myQueue);    /\*初始化的队列\*/

    QueueAppend(&myQueue, 1);   /\*第一行第一个元素1加入队列\*/

    for (int m = 2; m <= n; m++)

    {

        QueueAppend(&myQueue, 1);   /\*每行的第一个元素总是1\*/

        /\*打印和计算除了每行首尾元素之外的其他元素\*/

        for (int i = 1; i <= m - 2; i++)

        {

            QueueDelete(&myQueue, &temp);   /\*取出队列的第一个元素(上一行的元素)\*/

            printf("%6d", temp);

            QueueGet(&myQueue, &x);     /\*获取队列的当前第一个元素(下一行需要计算的元素之一)\*/

            temp += x;      /\*计算当前元素的值(上一行的两个元素之和)\*/

            QueueAppend(&myQueue, temp);    /\*将计算后的元素加入队列\*/

        }

        /\*打印每行的最后一个元素(它总是1的前一个元素)\*/

        QueueDelete(&myQueue, &x);

        printf("%6d", x);

        /\*每行的最后一个元素总是1\*/

        QueueAppend(&myQueue, 1);

        printf("\n");

    }

    /\*打印队列中的剩余元素\*/

    while (QueueNotEmpty(myQueue))

    {

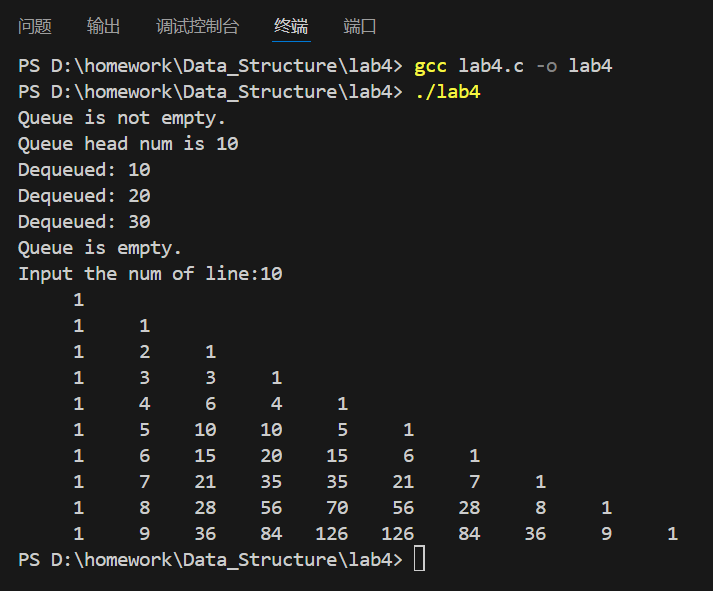
        QueueDelete(&myQueue, &x);

        printf("%6d", x);

    }

}

四、实验结果（测试数据）



五、实验心得

答：本次实验中，我掌握了队列的顺序存储结构，掌握了队列先进先出运算原则在解决实际问题中的应用。并学会了用顺序循环队列来设计杨辉三角。在实现顺序循环队列的过程中，我遇到了如何正确计算队尾位置和实现循环的问题，但通过仔细思考和多次调试，最终成功解决了这些问题。在利用队列计算杨辉三角的过程中，我体会到了队列在解决实际问题中的应用价值。虽然本次实验中采用的方法效率较低，但它展示了队列在算法设计中的灵活性和实用性。