**实验2**

**实验题目：利用循环队列打印杨辉三角**

**实验目的：**

1）进一步熟悉利用Code::Blocks，VC++等集成编译环境调试C/C++代码；

2）理解掌握循环队列的基本操作，例如：循环队列初始化、入队、出队、判断队空、判断队满以及求队列长度等；

3）掌握如何利用循环队列解决实际应用问题。

**实验要求：**

阅读所提供的代码，根据提示补全缺少的部分，完成下列功能：

1）遍历循环队列，输出循环队列中的元素值；

2）根据输入的行数，打印相应的杨辉三角；

**实验主要步骤：**

1. 分析、理解给出的程序，根据提示填写缺少部分的代码；
2. 调试程序，并且编译通过；
3. 设你的学号末尾数字为x，将mod(x, 5) + 5作为输入行数，获得程序输出结果（即打印在屏幕上的杨辉三角）。

备注：杨辉三角是除了每一行的第一个元素和最后一个元素是1，其他元素的值是上一行与之相邻的两个元素之和。

**课后实验报告提交要求：**

1. 提交时将所有函数写在一个源文件(.c文件或者.cpp文件)中；
2. 完成实验报告中的问题；

程序代码（请同学们***不改动已提供的代码***，填写空缺部分的代码）:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include "stdio.h"

#include "malloc.h"

#define OK 1

#define OVERFLOW -1

#define ERROR 0

#define QMAXSIZE 23//定义长度，长度>=输出行数+3

typedef int ElemType;

typedef int Status;

typedef struct

{

ElemType\* base; //初始化的动态分配存储空间

int front; //头指针，队列不空指向队列头元素

int rear; //尾指针，队列不空指向队列尾元素下一位置

}SqQueue;

Status InitQueue(SqQueue& Q)

{//构造空队列

Q.base = new ElemType[QMAXSIZE];

if (!Q.base)

return OVERFLOW;

Q.front = Q.rear = 0;

return OK;

}

Status QueueLength(SqQueue Q)

{//获取队列长度

return (Q.rear - Q.front + QMAXSIZE) % QMAXSIZE;

}

void GetHead(SqQueue Q, ElemType& e)

{//返回队头元素

if (Q.front == Q.rear)

e = 0;

else

e = Q.base[Q.front];

}

Status EnQueue(SqQueue& Q, ElemType e)

{//插入元素

if ((Q.rear + 1) % QMAXSIZE == Q.front)

return ERROR;

Q.base[Q.rear] = e;

Q.rear = (Q.rear + 1) % QMAXSIZE;

return OK;

}

Status DeQueue(SqQueue& Q, ElemType& e)

{//删除元素

if (Q.front == Q.rear)

return ERROR;

e = Q.base[Q.front];

Q.front = (Q.front + 1) % QMAXSIZE;

return OK;

}

Status QueueEmpty(SqQueue& Q)

{//判断队列是否为空

if (Q.front == Q.rear)

return OK;

else

return ERROR;

}

void PrintQueue(SqQueue Q, ElemType n)

{//遍历循环队列，并打印所有元素

int count = 1;

while (count <= n)

{

ElemType e;

DeQueue(Q, e);

printf("%d \t", e);

count++;

}

printf("\n");

}

void YangHui(int n)

{//杨辉三角

SqQueue Q;

ElemType j, s, t;

printf("第1行：1\n");

InitQueue(Q);

EnQueue(Q, 0); /\*开始\*/

EnQueue(Q, 1); /\*第1行\*/

EnQueue(Q, 1);

for (j = 2; j < n; j++) //从第二行开始循环到n-1行

{

EnQueue(Q, 0); /\*第j行的结束符\*/

printf("第%d行：", j);

do

{

DeQueue(Q, s);

GetHead(Q, t);

if (t)

printf("%d\t", t); /\*非0输出，否则换行\*/

else

printf("\n");

EnQueue(Q, s + t);

} while (t != 0); /\*遇到结束符前循环\*/

}

DeQueue(Q, s); //输出最后一行

printf("第%d行：", j);

PrintQueue(Q, j);

}

main()

{

ElemType n;

while (true)

{

printf("请输入输出的行数(小于等于%d)：", QMAXSIZE - 3);

scanf("%d", &n);

if (n <= (QMAXSIZE - 3) && n > 1)

{

YangHui(n);

break;

}

else

printf("输入错误，请重新输入\n");

}

}

**实验问题：**

1. 解释加阴影部分代码所实现的功能；
2. 分析函数YangHui() 的时间复杂度。
3. 判断元素是否为0，不为0则输出，属于该层，还未结束；

为0则表示该层结束，换行。

1. O（n^2）

分析如下：第n行中有n个元素，加上0可视为n+1个元素，输入n+1个元素并且判断n+1个元素，但由于从第二层开始，可视为判断2+3+4……+n，根据等差数列求和，可得时间复杂度为O（n^2）

**实验结果：**

根据实验步骤中的提示，将屏幕输出内容截取，并粘贴到word文档中。

