#include<iostream>

using namespace std;

class Queue

{

public:

//构造函数：初始化队列，假设队列元素都为简单的整型数据

//C++实现 //C语言实现

Queue(int QueueCapacity); //InitQueue(&Q) 创建队列

virtual ~Queue(); //DestroyQueue(&Q) 销毁队列

void ClearQueue(); //ClearQueue(&Q) 清空队列

bool QueueEmpty() const; //QueueEmpty(Q) 判断队列是否为空

bool QueueFull() const; //QueueFull(Q) 判断队列是否为满

int QueueLength() const; //QueueLength(Q) 队列长度

bool EnQueue(int element); //EnQueue(&Q,element) 新元素入队

bool DeQueue(int &element); //DeQueue(&Q,&element)首元素出队

void QueueTraverse(); //QueueTranverse(Q) 遍历队列

private:

int \*m\_pQueue; //队列数组指针

int m\_iQueueLen; //队列元素的个数

int m\_iQueueCapacity; //队列的数组容量

int m\_iHead;

int m\_iTail;

};

Queue::Queue(int QueueCapacity)

{

m\_iQueueCapacity = QueueCapacity; //队列容量，初始化时给定

m\_pQueue = new int[m\_iQueueCapacity]; //申请空间

m\_iHead = 0; //队列头的位置

m\_iTail = 0; //队列尾的位置

m\_iQueueLen = 0; //队列初始长度

}

Queue::~Queue()

{

delete []m\_pQueue;

m\_pQueue = NULL; //释放空间

}

void Queue::ClearQueue() //清空队列只需将头尾、长度置为0

{

m\_iHead = 0;

m\_iTail = 0;

m\_iQueueLen = 0;

}

bool Queue::QueueEmpty() const //判断队列是否为空：队列长度等于0即为空

{

if (m\_iQueueLen == 0)

{

return true;

}

else

{

return false;

}

//m\_iQueueLen == 0 ? true : false;

}

bool Queue::QueueFull() const //判断队列是否为满：队列长度等于队列申请的空间 即为满

{

if (m\_iQueueLen == m\_iQueueCapacity)

{

return true;

}

else

{

return false;

}

}

int Queue::QueueLength() const //求队列的长度

{

return m\_iQueueLen;

}

bool Queue::EnQueue(int element) //元素入队：先判断队列是否未满

{

if (QueueFull())

{

cout << "queue is full!" << endl;

return false;

}

else

{

m\_pQueue[m\_iTail] = element; //将入队元素的值赋给队尾

m\_iTail++; //队尾后移 1

m\_iTail = m\_iTail % m\_iQueueCapacity; //队尾的下标变成 m\_iTail % m\_iQueueCapacity

m\_iQueueLen++; //队列长度加 1

return true;

}

}

bool Queue::DeQueue(int &element) //元素出队：先判断队列是否为空

{

if (QueueEmpty())

{

cout << "queue is empty!" << endl;

return false;

}

else

{

element = m\_pQueue[m\_iHead]; //将要出队的队列头的值赋给给定的变量

m\_iHead++; //队列头后移 1

m\_iHead = m\_iHead % m\_iQueueCapacity; //头的下标更新为 m\_iHead % m\_iQueueCapacity

m\_iQueueLen--; //队列长度减 1

return true;

}

}

void Queue::QueueTraverse() //遍历对列打印出各元素的值

{

for (int i = m\_iHead; i < m\_iQueueLen + m\_iHead; i++) //i 从头开始访问，直到访问完所有元素，长度为m\_iQueueLen + m\_iHead

{

cout << m\_pQueue[i % m\_iQueueCapacity] << endl;

}

}

int main()

{

Queue q(10);

int i;

for (i = 0; i < 10; i++)

q.EnQueue(i);

q.EnQueue(10);

q.QueueTraverse();

for (i = 0; i < 10; i++)

q.DeQueue(i);

q.DeQueue(i);

q.QueueTraverse();

}