template<class T>

class arrayQueue

{

public:

arrayQueue(int initialCapacity = 10)

{//构造函数

if(initialCapacity<0) std::cerr<<"队列长度必须大于0！"<<std::endl;

else{

Queue=new T[initialCapacity];

arrayLength=initialCapacity;

qFront=qSize=0;

}

}

~arrayQueue() {delete [] Queue;}

bool empty() const

{

if(qSize==0) return true;

else return false;

}

int size() const

{

return qSize;

}

T& front()

{

if(empty()!=true)

return Queue[qFront]; //这里qFront记录的就是队列首元素的位置

else

{ std::cerr<<"队列为空"<<std::endl; exit(1); }

}

T& back()

{

if(empty()!=true)

return Queue[(qFront+qSize-1)%arraylength];

else

{ std::cerr<<"队列为空"<<std::endl; exit(1); }

}

void pop()

{//从队首删除元素

T \*p=&front(); //这里已经判断了队列是否为空

qFront=(qFront+1)%arrayLength;

(\*p).~T();

qSize--;

}

void push(const T& ele)

{//从队尾添加元素

if( qSize==arrayLength-1 )

{//队列将满，加倍数组长度

T \*newQueue=new T[2\*arrayLength];

int start=qFront;

if(start==0||start==1)

{//未形成环

copy(Queue+start,Queue+arrayLength,newQueue);

}

else

{//形成了环

copy(Queue+start,Queue+arrayLength,newQueue);

//复制第2段(start,队列末端，新队列起点）

copy(Queue,Queue+qSize-(arrayLength-start),newQueue+(arrayLength-start)); //复制第1段（原队列首端，qback,新队列第arraylength-start个位置）

}

qFront=0; //重新设置首游标

arrayLength=arrayLength\*2;

delete [] Queue;

Queue=newQueue;

}

//把元素插入队列的尾部

Queue[(qFront+qSize)%arrayLength]=ele;

qSize++;

}

private:

int qFront; //队列中第一个元素

int Qsize; //队列的中元素个数

int arrayLength; //队列的容量

T \*Queue; //队列元素数组

};

template<class T>

class arrayQueue

{

public:

arrayQueue(int initialCapacity = 10)

{//构造函数

if(initialCapacity<0) std::cerr<<"队列长度必须大于0！"<<std::endl;

else{

Queue=new T[initialCapacity];

arrayLength=initialCapacity;

qFront=qBack=0;

lastOp=0;

}

}

~arrayQueue() {delete [] Queue;}

bool empty() const

{

if(qFront==qBack&&lastOp!=1) return true;

else return false;

}

int size() const

{

return (arrayLength+qBack-qFront)%arrayLength;

}

T& front()

{

if(empty()!=true)

return Queue[(qFront+1)%arrayLength];

else

{ std::cerr<<"队列为空"<<std::endl; exit(1); }

}

T& back()

{

if(empty()!=true)

return Queue[qBack];

else

{ std::cerr<<"队列为空"<<std::endl; exit(1); }

}

void pop()

{//从队首删除元素

T \*p=&front(); //这里已经判断了队列是否为空

qFront=(qFront+1)%arrayLength;

(\*p).~T();

lastOp=2;

}

void push(const T& ele)

{//从队尾添加元素

if( (qBack+1)%arrayLength==qFront )

{//队列将满，加倍数组长度

T \*newQueue=new T[2\*arrayLength];

int start=(qFront+1)%arrayLength;

if(start==0||start==1)

{//未形成环

copy(Queue+start,Queue+qBack+1,newQueue);

}

else

{//形成了环

copy(Queue+start,Queue+arrayLength,newQueue);

//复制第2段(start,队列末端，新队列起点）

copy(Queue,Queue+qBack+1,newQueue+(arrayLength-start));

//复制第1段（原队列首端，qback,新队列第arraylength-start个位置）

}

qFront=(arrayLength)\*2-1;

qBack=arrayLength-1-1; //重新设置首尾游标

arrayLength=arrayLength\*2;

delete [] Queue;

Queue=newQueue;

}

//把元素插入队列的尾部

qBack=(qBack+1)%arrayLength;

Queue[qBack]=ele;

lastOp=1;

}

private:

int qFront; //队列中第一个元素的前一个未知

int qBack; //队列最后一个元素的位置

int arrayLength; //队列的容量

int lastOp; //0代表未操作，1代表push,2代表pop

T \*Queue; //队列元素

};