# 实验四 队列的存储与操作

**一、实验目的**

1．理解队列是限定只能在队头进行删除操作在队尾进行插入操作的线性表；

2．理解队列的存储结构特点，掌握队列的存储分配要点；

3．掌握队列的基本操作及实现，深刻领会队列操作的先进先出特征，并能正确分析其时间复杂度，知道队列性能优于普通线性表，以及队列的常用情形。

**二、实验内容**

课堂上完成顺序队列，链队作为课后练习。

1. 定义顺序队列的存储结构：（私有）数据封装（一维）**数组（**存储数据元素，位置/下标表示数据元素之间的逻辑关系，不占用存储空间）、**队头和队尾**指针（整型变量，存储队头和队尾下标），队列的头尾在数组存储空间内循环移动；本次实现队列的对头指针的下一个位置存储表中第一个数据元素，队尾指针指向表中最后一个元素。

2. 顺序队列的基本操作

（1）初始化顺序队列；

（2）数据元素入队；

（3）数据元素出队；

（4）读队头元素；

（5）判定队列空/满；

（6）销毁顺序队列。

**三、算法思想和时间复杂度**

顺序队列在通过除以数组规模取余数（模运算）实现环（首尾相接）。

1. 初始化顺序队列：队头和队尾为低端0；

2. 数据元素入队列：队列不满，先增加队尾指针，再存储数据元素，时间复杂度为O（1）；

3. 数据元素出队列：队列不空，先增加对头指针，再读数据元素时间复杂度为O（1）；

4. 读队头元素：队列不空，读数据元素，不动队头指针，时间复杂度为O（1）；

5. 判定队列空/满操作：front等于rear/front等于（rear+1）%Queue\_Size。

**四、参考代码**

**1. 队列的存储结构和操作接口**

#ifndef SeqList\_H

#define SeqList\_H

const int QueueSize=100; //???

template <class T>

class CirQueue

{

public:

CirQueue() {front=rear=QueueSize-1;} //???

~CirQueue() {}

void EnQueue(T x);

T DeQueue();

T GetQueue();

int Empty(){return front==rear? 1: 0;} //???

private:

T data[QueueSize];

int front,rear;

};

#endif

**2. 队列操作的实现**

#include "CirQueue.h"

/\*

\*前置条件：队列已存在

\*输 入：元素x

\*功 能：入队操作，将元素x插入到对尾

\*输 出：若插入不成功，抛出异常

\*后置条件：如果插入成功，队尾增加一个元素

\*/

template <class T>

void CirQueue<T>::EnQueue(T x)

{

if((rear+1)%QueueSize==front) //???

throw"上溢";

rear=(rear+1)%QueueSize; //???

data[rear]=x; //???

}

/\*

\*前置条件：队列已存在

\*输 入：无

\*功 能：出队操作，删除队头元素

\*输 出：若删除成功，返回被删元素值，否则，抛出删除异常

\*后置条件：如果删除成功，队头减少一个元素

\*/

template <class T>

T CirQueue<T>::DeQueue()

{

if(rear==front) throw"下溢";

front=(front+1)%QueueSize; //???

return data[front]; //???

}

/\*

\*前置条件：队列已存在

\*输 入：无

\*功 能：读取队头元素

\*输 出：若队列不空，返回队头元素

\*后置条件：队列不变

\*/

template <class T>

T CirQueue<T>::GetQueue()

{

int i;

if(rear==front) //???

throw"下溢";

i=(front+1)%QueueSize; //???

return data[i]; //???

}

**3. 调试运行**

#include <iostream> //引用输入输出流库函数的头文件

using namespace std;

#include "CirQueue.cpp"

void main( )

{

CirQueue<int>a; //???

int e;

try

{

cout<<"执行入队操作(请输入5个队列元素):\n"<<endl;

for(int i=0;i<5;i++) //???

{

cin>>e; //???

a.EnQueue(e); //???

}

}

catch(char\* wrong)

{

cout << wrong; //???

}

e = a.GetQueue(); //???

cout<<"队头元素为:"<<e<<endl;

try

{

cout<<"执行出队操作(输入出队元素个数)\n"<<endl;

cin>>e; //???

for(int i=0;i<e;i++) //???

{

cout<<a.DeQueue()<<"已出队"<<endl;

}

}

catch(char\* wrong)

{

cout << wrong; //???

}

try

{

e = a.GetQueue(); //???

cout<<"队头元素为:"<<e<<endl;

}

catch(char\* wrong)

{

cout << wrong; //???

}

if(a.Empty()) //???

cout<<"队列为空"<<endl;

else

cout<<"队列非空"<<endl;

system("pause");

}

**五、队列的应用**

队列的应用主要有数据缓冲、程序/进程（线程）调度、树的层次遍历和图的广度优先遍历等。