**《数据结构》上机报告**

**2018 年 10 月 20 日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名： | 赵得泽 | 学号： | 1753642 | 班级： | 电子二班 | 得分： |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验题目** | | **队列** | | |
| **问题描述** | | 队列是限定在表的一端插入，另一端删除的线性表。队列的特点是先进先出（FIFO)。 **循环队列**是队列的**顺序存储结构**。**队列的链式存储结构**需要两个分别指示队头和队尾的指针，保证**入队和出队的操作时间复杂度都是O(1)**。 | | |
| **基本要求** | | 1. 练习循环队列的基本操作，包括入队、出队、判队空、判队满、队列的遍历。   2. 练习链队列的基本操作，包括入队、出队、判队空、判队满、队列的遍历。 | | |
| **已完成基本内容（序号）：** | 1，2 | |
| **选做要求** | |  | | |
| **已完成选做内容（序号）** | |  |
| **数据结构**  **设计** | | **循环队列数据结构：**  **typedef int Status;**  **typedef int QElemType;**  **class SqQueue**  **{**  **protected:**  **QElemType \*base;**  **int front;**  **int rear;**  **public:**  **int n;**  **SqQueue();**  **~SqQueue();**  **Status EnQueue(QElemType e);**  **Status DeQueue(QElemType &e);**  **Status QueueEmpty();**  **Status QueueFull();**  **void PrintQueue();**  **};**  循环队列也就是顺序队列，即线性表的顺序存储结构；它是用一组地址连续的存储单元依次存放从队列头到队列尾的元素。在本数据结构的设计过程中，**front**和**rear**分别作队列的头指针和尾指针进行队列元素的入队和出队操作  **base**则作为一个动态数组，为其分配空间，进行数据元素的存储。其他函数作为该类的成员函数执行入队、出队、判断队空、队满，打印队列等操作。  **链队列数据结构：**  **typedef int Status;**  **typedef int QElemType;**  **class QNode**  **{**  **public:**  **QElemType data;**  **QNode \*next;**  **};**  **class LinkQueue :public QNode**  **{**  **protected:**  **QNode \*front;**  **QNode \*rear;**  **int count;**  **public:**  **int n;**  **LinkQueue();**  **~LinkQueue();**  **Status EnQueue(QElemType e);**  **Status DeQueue(QElemType &e);**  **Status QueueEmpty();**  **void PrintQueue();**  **};**  链队列也就是线性表的链式存储结构，实际是带头结点的线性链表。在本数据结构的设计过程中**QNode**为结点类型，有数据域**data**存储队列数据和指针域**next**指向下一个结点；**LinkQueue**为链队列类型，其中有两个指针**front**和**rear**分别指向队头和队尾，进行队列数据的操作； | | |
| **功能(函数)说明** | | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：循环队列的初始化  说明：循环队列的初始化要将头结点和尾结点都初始化为0，并且头结点等于尾结点，为base分配空间\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  SqQueue::SqQueue()  {  base = new QElemType[MAXSIZE];  if (!base) exit(OVERFLOW);  front = rear = 0;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：循环队列的销毁  说明：将初始化时候的base数组delete即可；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  SqQueue::~SqQueue()  {  delete base;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：循环队列的入队  输入参数：入队元素e  说明：入队的时候要判断队列是否为空，若为空则不入队，否则，从队尾入队。但  是循环队列的入队为了防止溢出，应该rear = (rear + 1) %队列的长度;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status SqQueue::EnQueue(QElemType e)  {  if (QueueFull())  {  cout << "Queue is Full" << endl;  return ERROR;  }  base[rear] = e;  rear = (rear + 1) % (n + 1);  return OK;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：循环队列的出队  输出参数：出队元素  说明：出队的时候判断队列是否为空，若为空，则不出队；否则，从队头出队。  但是出队的时候为了防止溢出应该front = (front + 1) % 队列的长度;  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status SqQueue::DeQueue(QElemType &e)  {  if (QueueEmpty())  {  cout << "Queue is Empty" << endl;  return ERROR;  }  e = base[front];  cout << e << endl;  front = (front + 1) % (n + 1);  return OK;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：判断循环队列是否为空  说明：当队头等于队尾时，循环队列为空；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status SqQueue::QueueEmpty()  {  if (front == rear) return TRUE;  return FALSE;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：判断循环队列是否为满  说明：循环队列需要浪费一个空间，即一个空间不存数据来判断队满，当rear指向该空间时，若(rear + 1) % 队列的长度=front，则可知队满；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status SqQueue::QueueFull()  {  if ((rear + 1) % (n + 1) == front)  return TRUE;  return FALSE;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：链队列的初始化  说明：头结点和尾结点都是分配空间，并且头结点等于尾结点，头结点的next指针  为空；count用来计算当前队列的长度，初始化为0；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  LinkQueue::LinkQueue()  {  front = rear = new QNode;  if (!front) exit(OVERFLOW);  count = 0;  front->next = NULL;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：链队列的销毁  说明：利用循环，从头结点到尾结点，一一销毁  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  LinkQueue::~LinkQueue()  {  while (front)  {  rear = front->next;  delete front;  front = rear;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：链队列的入队  输入参数：入队元素e  说明：链队列入队从队尾进入若队满则不进入，否则，新建一个指针p申请一个结点类型空间，将入队元素赋给指针，然后连接到尾结点，尾结点后移一位等于结点p；实际长度加一（count++）  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status LinkQueue::EnQueue(QElemType e)  {  QNode \*p = new QNode;  if (count >= n)  {  cout << "Queue is Full" << endl;  return ERROR;  }  if (!p) exit(OVERFLOW);  p->data = e; p->next = NULL;  rear->next = p;  rear = p;  count++;  return OK;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：链队列的出队  输出参数：出队元素e  说明：链队列出队先判断队列是否为空，若为空则不出队；否则，定义一个新节点  指向队头结点，将队头结点的数据域赋给它，然后队头元素向前移一次，队列实际  长度减一（count--）。  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status LinkQueue::DeQueue(QElemType &e)  {  if (QueueEmpty())  {  cout << "Queue is Empty" << endl;  return ERROR;  }  QNode \*p;  p = front->next;  e = p->data;  cout << e << endl;  front->next = p->next;  if (rear == p) rear = front;  delete p;  count--;  return OK;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  功能：判断链队列是否为空  说明：头结点指向尾结点，则说明链队列为空；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Status LinkQueue::QueueEmpty()  {  if (front == rear) return TRUE;  return FALSE;  } | | |
| **开发环境** | | Win10,vs2017,C++高级程序语言设计 | | |
| **调试分析** | | **循环队列：**    **链队列：** | | |
| **心得体会** | 在本次实验的设计中，我充分领悟到了队列的含义和意义。队列主要包括顺序结构（循环队列）和链式结构（链队列），这两种结构的区别就是循环队列会在最初定义时有最大容量限定，而链队列则是在任意一块空间存储，不需要限定容量，可以随时进行容量的扩增。如果在题目要求下进行队列长度的限制，而且两者的长度相等时，两者完成相同的功能会得到相同的结果。  即它们的功能是有一定相同之处的。  但是对于循环队列来说，使用全部的空间，则会使队空的条件和队满的条件一致，即front=rear，但是我们想，也许少用一个空间，让rear指针指向该未使用的空间，则队满条件就变成了**(rear+1)%QueueSize=front**,而队空条件依然为**front=rear**，这样就很好的区分了队空和队满，还有就是循环队列的如队和出队操作都使用了**指针=（指针+1）%QueueSize**，这样也很好的避免了队列的“假溢出”，执行了入、出队操作。  相比之下，链式队列就是和**带头结点**的链表一样进行正常操作即可，不过要考虑到队列的实际意义——**队尾进，队头出**。 | | | |