队列

1.顺序队列

```
#define OK 1
#define ERROR 0
#define TRUE 1
#define FALSE 0
#define MAXSIZE 20 /* 存储空间初始分配量 */

typedef int Status;
typedef int QElemType; /* QElemType类型根据实际情况而定，这里假设为int */

/* 循环队列的顺序存储结构 */
typedef struct
{
    QElemType data[MAXSIZE];
    int front;      /* 头指针 */
    int rear;       /* 尾指针，若队列不空，指向队列尾元素的下一个位置 */
}SqQueue;

Status visit(QElemType c)
{
    printf("%d ",c);
    return OK;
}

/* 初始化一个空队列Q */
Status InitQueue(SqQueue *Q)
{
    Q->front=0;
```

```
    Q->rear=0;
    return  OK;
}

/* 将Q清为空队列 */
Status ClearQueue(SqQueue *Q)
{
    Q->front=Q->rear=0;
    return OK;
}

/* 若队列Q为空队列,则返回TRUE,否则返回FALSE */
Status QueueEmpty(SqQueue Q)
{
    if(Q.front==Q.rear) /* 队列空的标志 */
        return TRUE;
    else
        return FALSE;
}

/* 返回Q的元素个数，也就是队列的当前长度 */
int QueueLength(SqQueue Q)
{
    return  (Q.rear-Q.front+MAXSIZE)%MAXSIZE;
}

/* 若队列不空,则用e返回Q的队头元素,并返回OK,否则返回ERROR
*/
Status GetHead(SqQueue Q,QElemType *e)
{
    if(Q.front==Q.rear) /* 队列空 */
        return ERROR;
    *e=Q.data[Q.front];
    return OK;
}
```

```c
/* 若队列未满，则插入元素e为Q新的队尾元素 */
Status EnQueue(SqQueue *Q,QElemType e)
{
    if ((Q->rear+1)%MAXSIZE == Q->front)    /* 队列满的判断 */
        return ERROR;
    Q->data[Q->rear]=e;          /* 将元素e赋值给队尾 */
    Q->rear=(Q->rear+1)%MAXSIZE;/* rear指针向后移一位置， */
                                /* 若到最后则转到数组头部 */
    return  OK;
}

/* 若队列不空，则删除Q中队头元素，用e返回其值 */
Status DeQueue(SqQueue *Q,QElemType *e)
{
    if (Q->front == Q->rear)            /* 队列空的判断 */
        return ERROR;
    *e=Q->data[Q->front];               /* 将队头元素赋值给e */
    Q->front=(Q->front+1)%MAXSIZE;  /* front指针向后移一位置， */
                                    /* 若到最后则转到数组头部 */
    return  OK;
}

/* 从队头到队尾依次对队列Q中每个元素输出 */
Status QueueTraverse(SqQueue Q)
{
    int i;
    i=Q.front;
    while((i+Q.front)!=Q.rear)
    {
```

```
        visit(Q.data[i]);
        i=(i+1)%MAXSIZE;
    }
    printf("\n");
    return OK;
}

int main()
{
    Status j;
    int i=0,l;
    QElemType d;
    SqQueue Q;
    InitQueue(&Q);
    printf("初始化队列后，队列空否？%u(1:空 0:
否)\n",QueueEmpty(Q));

    printf("请输入整型队列元素(不超过%d个),-1为提前结束符:
",MAXSIZE-1);
    do
    {
        /* scanf("%d",&d); */
        d=i+100;
        if(d==-1)
            break;
        i++;
        EnQueue(&Q,d);
    }while(i<MAXSIZE-1);

    printf("队列长度为：%d\n",QueueLength(Q));
    printf("现在队列空否？%u(1:空 0:
否)\n",QueueEmpty(Q));
    printf("连续%d次由队头删除元素,队尾插入元
素:\n",MAXSIZE);
    for(l=1;l<=MAXSIZE;l++)
    {
```

```
            DeQueue(&Q,&d);
            printf("删除的元素是%d,插入的元素:%d
\n",d,l+1000);
            /* scanf("%d",&d); */
            d=l+1000;
            EnQueue(&Q,d);
        }
        l=QueueLength(Q);

        printf("现在队列中的元素为: \n");
        QueueTraverse(Q);
        printf("共向队尾插入了%d个元素\n",i+MAXSIZE);
        if(l-2>0)
            printf("现在由队头删除%d个元素:\n",l-2);
        while(QueueLength(Q)>2)
        {
            DeQueue(&Q,&d);
            printf("删除的元素值为%d\n",d);
        }

        j=GetHead(Q,&d);
        if(j)
            printf("现在队头元素为: %d\n",d);
        ClearQueue(&Q);
        printf("清空队列后, 队列空否? %u(1:空 0:
否)\n",QueueEmpty(Q));
        return 0;
}
```

2.链队列

```
#define OK 1
#define ERROR 0
#define TRUE 1
```

```c
#define FALSE 0
#define MAXSIZE 20 /* 存储空间初始分配量 */

typedef int Status;

typedef int QElemType; /* QElemType类型根据实际情况而定，这里假设为int */

typedef struct QNode    /* 结点结构 */
{
   QElemType data;
   struct QNode *next;
}QNode,*QueuePtr;

typedef struct          /* 队列的链表结构 */
{
   QueuePtr front,rear; /* 队头、队尾指针 */
}LinkQueue;

Status visit(QElemType c)
{
    printf("%d ",c);
    return OK;
}

/* 构造一个空队列Q */
Status InitQueue(LinkQueue *Q)
{
    Q->front=Q->rear=
(QueuePtr)malloc(sizeof(QNode));
    if(!Q->front)
        exit(OVERFLOW);
    Q->front->next=NULL;
    return OK;
}
```

```
/* 销毁队列Q */
Status DestroyQueue(LinkQueue *Q)
{
    while(Q->front)
    {
        Q->rear=Q->front->next;
        free(Q->front);
        Q->front=Q->rear;
    }
    return OK;
}

/* 将Q清为空队列 */
Status ClearQueue(LinkQueue *Q)
{
    QueuePtr p,q;
    Q->rear=Q->front;
    p=Q->front->next;
    Q->front->next=NULL;
    while(p)
    {
        q=p;
        p=p->next;
        free(q);
    }
    return OK;
}

/* 若Q为空队列,则返回TRUE,否则返回FALSE */
Status QueueEmpty(LinkQueue Q)
{
    if(Q.front==Q.rear)
        return TRUE;
    else
        return FALSE;
}
```

```c
/* 求队列的长度 */
int QueueLength(LinkQueue Q)
{
    int i=0;
    QueuePtr p;
    p=Q.front;
    while(Q.rear!=p)
    {
        i++;
        p=p->next;
    }
    return i;
}

/* 若队列不空,则用e返回Q的队头元素,并返回OK,否则返回ERROR
*/
Status GetHead(LinkQueue Q,QElemType *e)
{
    QueuePtr p;
    if(Q.front==Q.rear)
        return ERROR;
    p=Q.front->next;
    *e=p->data;
    return OK;
}


/* 插入元素e为Q的新的队尾元素 */
Status EnQueue(LinkQueue *Q,QElemType e)
{
    QueuePtr s=(QueuePtr)malloc(sizeof(QNode));
    if(!s) /* 存储分配失败 */
        exit(OVERFLOW);
    s->data=e;
    s->next=NULL;
```

```
    Q->rear->next=s;    /* 把拥有元素e的新结点s赋值给原队尾结点的后继，见图中① */
    Q->rear=s;      /* 把当前的s设置为队尾结点，rear指向s，见图中② */
    return OK;
}

/* 若队列不空,删除Q的队头元素,用e返回其值,并返回OK,否则返回ERROR */
Status DeQueue(LinkQueue *Q,QElemType *e)
{
    QueuePtr p;
    if(Q->front==Q->rear)
        return ERROR;
    p=Q->front->next;       /* 将欲删除的队头结点暂存给p，见图中① */
    *e=p->data;             /* 将欲删除的队头结点的值赋值给e */
    Q->front->next=p->next;/* 将原队头结点的后继p->next赋值给头结点后继，见图中② */
    if(Q->rear==p)      /* 若队头就是队尾，则删除后将rear指向头结点，见图中③ */
        Q->rear=Q->front;
    free(p);
    return OK;
}

/* 从队头到队尾依次对队列Q中每个元素输出 */
Status QueueTraverse(LinkQueue Q)
{
    QueuePtr p;
    p=Q.front->next;
    while(p)
    {
        visit(p->data);
        p=p->next;
```

```
    }
    printf("\n");
    return OK;
}

int main()
{
    int i;
    QElemType d;
    LinkQueue q;
    i=InitQueue(&q);
    if(i)
        printf("成功地构造了一个空队列!\n");
    printf("是否空队列? %d(1:空 0:否)
",QueueEmpty(q));
    printf("队列的长度为%d\n",QueueLength(q));
    EnQueue(&q,-5);
    EnQueue(&q,5);
    EnQueue(&q,10);
    printf("插入3个元素(-5,5,10)后,队列的长度
为%d\n",QueueLength(q));
    printf("是否空队列? %d(1:空 0:否)
",QueueEmpty(q));
    printf("队列的元素依次为: ");
    QueueTraverse(q);
    i=GetHead(q,&d);
    if(i==OK)
     printf("队头元素是: %d\n",d);
    DeQueue(&q,&d);
    printf("删除了队头元素%d\n",d);
    i=GetHead(q,&d);
    if(i==OK)
        printf("新的队头元素是: %d\n",d);
    ClearQueue(&q);
    printf("清空队列后,q.front=%u q.rear=%u q.front-
>next=%u\n",q.front,q.rear,q.front->next);
```

```
    DestroyQueue(&q);
    printf("销毁队列后,q.front=%u
q.rear=%u\n",q.front, q.rear);

    return 0;
}
```