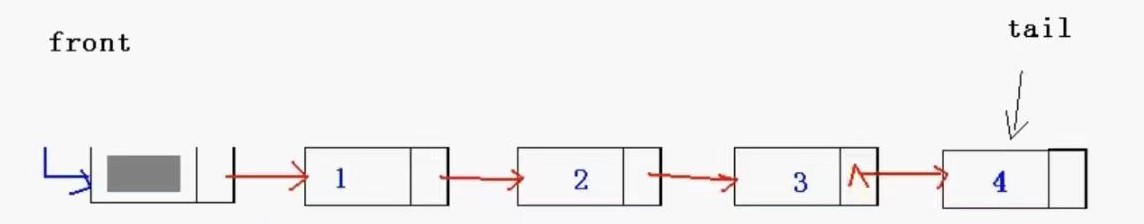
**队列的链式表示和实现**



**//队列的链式表示和实现**

**//单链表插入和删除操作的特殊情况**

**//只允许在队尾插入数据（入队）**

**//只允许在队头删除数据（出队）**

#include<stdio.h>

#include<assert.h>

#include<malloc.h>

#define ElemType int

typedef struct QueueNode {

ElemType data;

struct QueueNode\* next;

}QueueNode;

typedef struct LinkQueue {

QueueNode\* front;

QueueNode\* tail;

}LinkQueue;

void InitQueue(LinkQueue\*Q) {

QueueNode\* s = (QueueNode\*)malloc(sizeof(QueueNode));

assert(s != NULL);

Q->front = Q->tail = s;

Q->tail->next = NULL;

}

void EnQueue(LinkQueue\* Q, ElemType x) {

QueueNode\* s = (QueueNode\*)malloc(sizeof(QueueNode));

assert(s != NULL);

s->data = x;

s->next = NULL;

Q->tail->next = s;

Q->tail = s;

}

void ShowQueue(LinkQueue\* Q) {

QueueNode\* p = Q->front->next;

printf("Front:>");

while (p != NULL) {

printf("%d", p->data);

p = p->next;

}

printf("<:Tail.\n");

}

void DeQueue(LinkQueue\* Q) {

if (Q->front == Q->tail) {

return;

}

QueueNode\* p = Q->front->next;

Q->front->next = p->next;

free(p);

if (p = Q->tail) { //判断特殊情况 不用多说了

Q->tail = Q->front;

}

}

void GetHead(LinkQueue\* Q, ElemType\* v) {

if (Q->front == Q->tail) {

return;

}

QueueNode\* p = Q->front->next;

\*v = p->data; //\*v =Q->front->next->data 这样写不好 有个不成文的规定：链表里不超过二级指向

}

int Length(LinkQueue\* Q) {

int len = 0;

QueueNode\* p = Q->front->next;

while (p != NULL) {

len++;

p = p->next;

}

return len;

}

void ClearQueue(LinkQueue\* Q) {

if (Q->front == Q->tail) {

return;

}

QueueNode\* p = Q->front->next;

while (p != NULL) {

Q->front->next = p->next;

free(p);

p = Q->front->next;

}

Q->tail = Q->front;

}

void DestroyQueue(LinkQueue\* Q) {

ClearQueue(Q);

free(Q->front);

Q->front = Q->tail = NULL;

}

void main() {

LinkQueue Q;

InitQueue(&Q);

for (int i = 1; i <= 10; i++) {

EnQueue(&Q, i);

}

ShowQueue(&Q);

DeQueue(&Q);

ShowQueue(&Q);

printf("Len=%d\n", Length(&Q));

}