双向链表

1. 运行截图
2. 生成双向链表

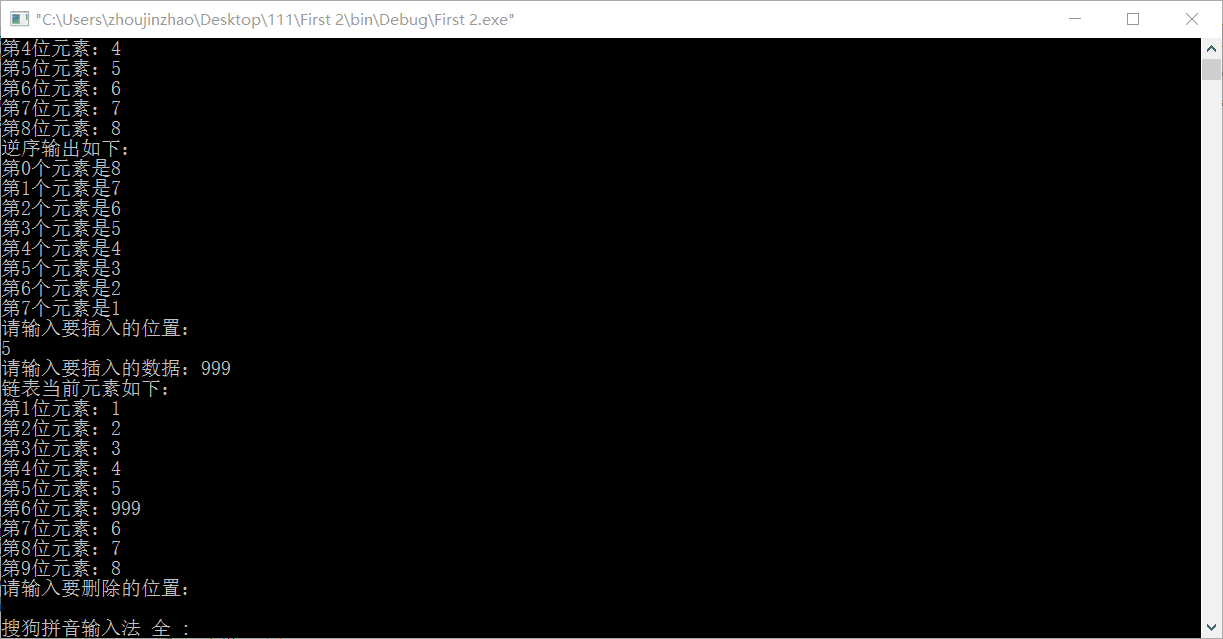


1. 插入节点

Index=0时

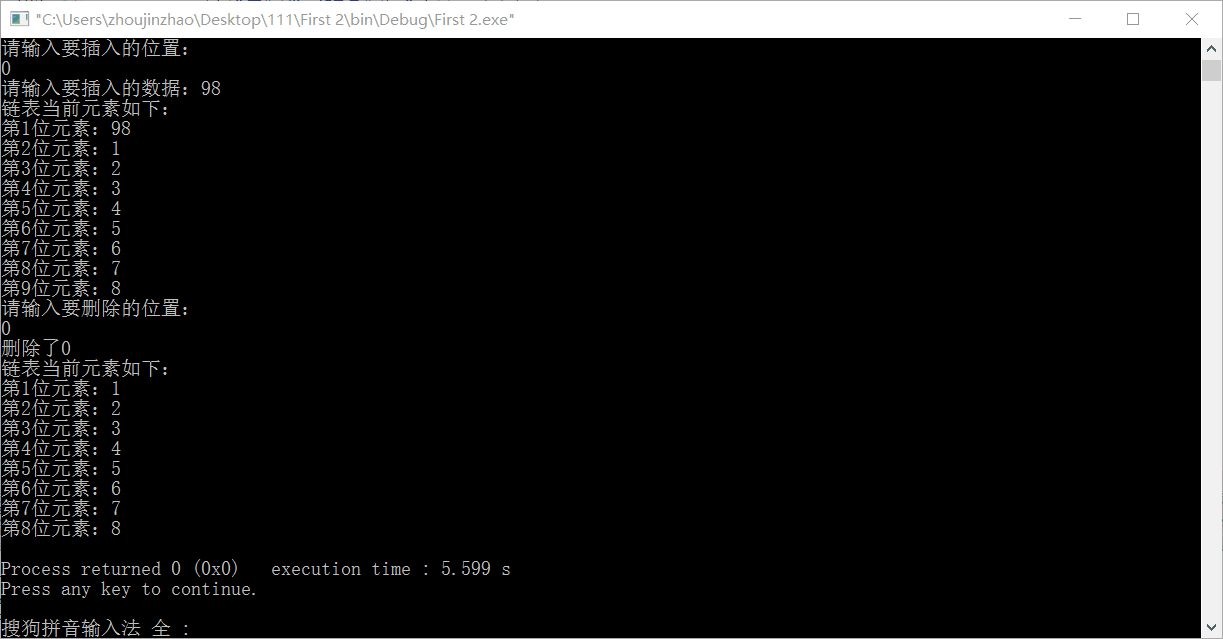


Index！=0时

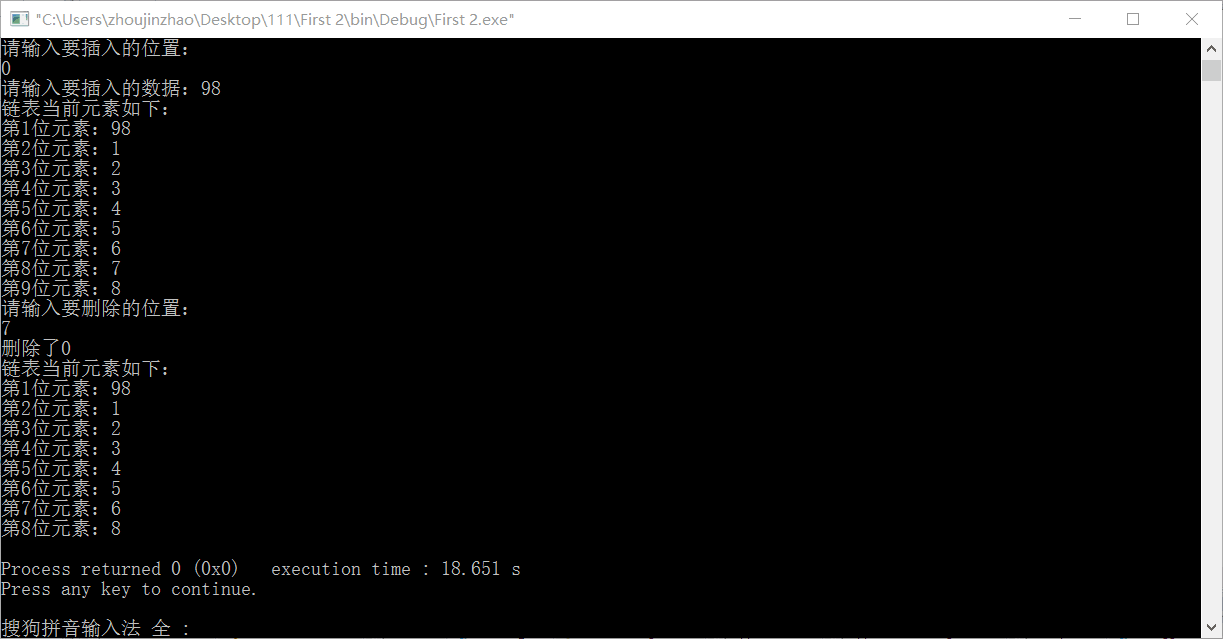


1. 删除节点

Index=0时



Index！=0时



1. 源代码

#include <stdio.h>

#include <stdlib.h>

/\*\* \* 双向链表操作 \*/

typedef struct TNode {

int data;

struct TNode \*next;

struct TNode \*pre;

}TNode, \*ptr\_TNode;

typedef enum Status {

SUCCESS,ERROR

}Status;

//生成双向链表

ptr\_TNode create(int \*arr, int n)

{

if(arr==NULL) return NULL; //若数组为空数组，返回NULL

int i=0;

ptr\_TNode p,q,head;

p=q=(ptr\_TNode)malloc(sizeof(TNode));

head=NULL;

p->data=\*arr; //读取数组数据

while(i<n)

{

if(i==0)

{

p->pre=NULL;//逆向链接的尾节点

head=p; //当i=0时读取到头指针

}

else

{

p->pre=q; //逆向链接

q->next=p; //当i!=0时，构建链接

}

q=p;i++;

p=(ptr\_TNode)malloc(sizeof(TNode));

p->data=\*(arr+i); //读取下一元素

}

q->next=NULL; //最后一位指针指向NULL

return(head); //返回头指针

}

ptr\_TNode tails(ptr\_TNode head)//返回尾节点

{

ptr\_TNode p;

p=head;

while(p->next!=NULL)

{

p=p->next;

}

return p;

}

//双向链表,删除第 index 位节点后面的节点，将删除的结点的值保存到（\*data）中

Status delete\_T(ptr\_TNode \*head, int index, int \*data)

{

if(head==NULL||\*head==NULL||index<0) return ERROR; //当头指针为NULL或index值不合理时

ptr\_TNode p,q;

p=\*head;

if(index==0) //删除第一个节点

{

\*data=p->data;

p=p->next;q=\*head;\*head=p;

free(q);return SUCCESS;

}

int n=0; //n表示节点位置

while(p!=NULL)

{

if(n==index-1) //找到指定位置后

{ //删除操作

\*data=p->next->data;

q=p->next;

p->next=p->next->next;

free(q);return SUCCESS; //释放空间并返回SUCCESS

}

n++; //位置+1

p=p->next; //指向下一节点

}

return ERROR;

}

//双向链表,在第 index 位后面插入 node 结点 //插入成功返回 SUCCESS，失败返回 ERROR

Status insert\_T(ptr\_TNode \*head, ptr\_TNode node, int index)

{

if(head==NULL||\*head==NULL||index<0) return ERROR; //头指针为NULL时和index值不合理时

ptr\_TNode p;

node=(ptr\_TNode)malloc(sizeof(TNode));//开辟新空间

printf("请输入要插入的数据：");

scanf("%d",&node->data); //输入要插入的节点的数据

p=\*head;

if(index==0)

{

node->next=\*head;\*head=node;//插入到第一个节点

return SUCCESS;

}

int n=0; //n表示节点位置

while(p!=NULL)

{

if(n==index-1) //找到位置后

{ //插入操作

node->next=p->next;

p->next=node;

return SUCCESS;

}

n++; //位置+1

p=p->next; //指向下一节点

}

return ERROR;

}

//输出链表 顺序

void print\_next(ptr\_TNode head)//按顺序输出

{

if(head==NULL) return;

ptr\_TNode p;

p=head;

int b=1;//表示第几个元素

printf("链表当前元素如下：\n");

while(p!=NULL)

{

printf("第%d位元素：%d\n",b++,p->data);//输出

p=p->next;//指向下一节点

}

}

//输出链表 逆序

void print\_pre(ptr\_TNode tail)//从尾节点开始输出（逆序）

{

ptr\_TNode q;

q=tail;

int b=0;

printf("逆序输出如下：\n");

while(q!=NULL)

{

printf("第%d个元素是%d\n",b++,q->data);

q=q->pre;

}

}

//销毁给定的链表

void destroy(ptr\_TNode head)

{

ptr\_TNode p,q;

p=head;

while(p)

{

q=p->next; //q指向下一指针

free(p); //释放p的空间

p=q; //试p指向q

}

head=NULL; //让头指针指向BULL

}

int main()

{

int b[8]={1,2,3,4,5,6,7,8},index,data,choice;//index为位置，

ptr\_TNode head,tail,node;

head=create(b,8); //生成双向链表

tail=tails(head); //读取尾节点

print\_next(head); //顺序输出

print\_pre(tail); //逆序输出

while(1)

{

printf("\t\t请选择要进行什么操作(输入1-3)：\n");

printf("\t\t1、插入节点\n");

printf("\t\t2、删除节点\n");

printf("\t\t3、退出\n");

scanf("%d",&choice); //输入要选择的操作

if(choice<0||choice>3)

{

printf("输入错误！\t");continue;

}

switch(choice)

{

case 1: printf("请输入要插入的位置：");

scanf("%d",&index); //index为要插入的位置

if(insert\_T(&head,node,index)==1) printf("未成功插入！\n");//插入函数

else print\_next(head);break;

case 2: printf("请输入要删除的位置：\n"); //输入删除位置

scanf("%d",&index);

if(delete\_T(&head,index,&data)==1) printf("没有该节点\n");//data存储删除数据，ed接手删除函数返回结果

else printf("删除了%d\n",data);

print\_next(head);break;

case 3: destroy(head);return 0; //销毁链表//结束程序

}

}

return 0;

}