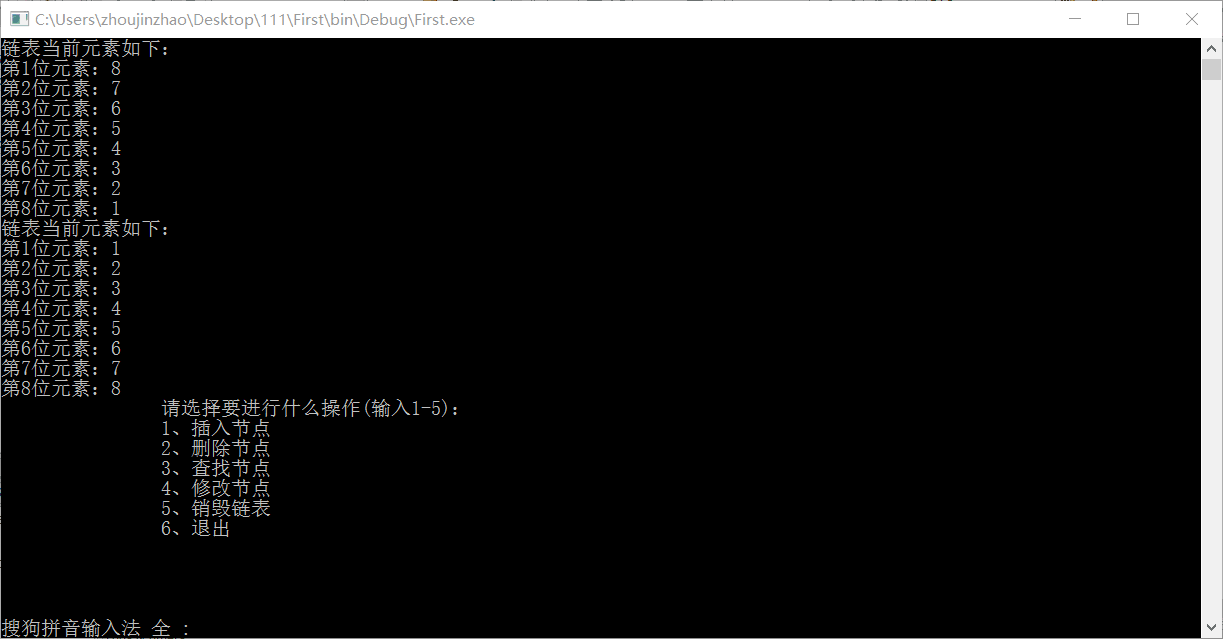
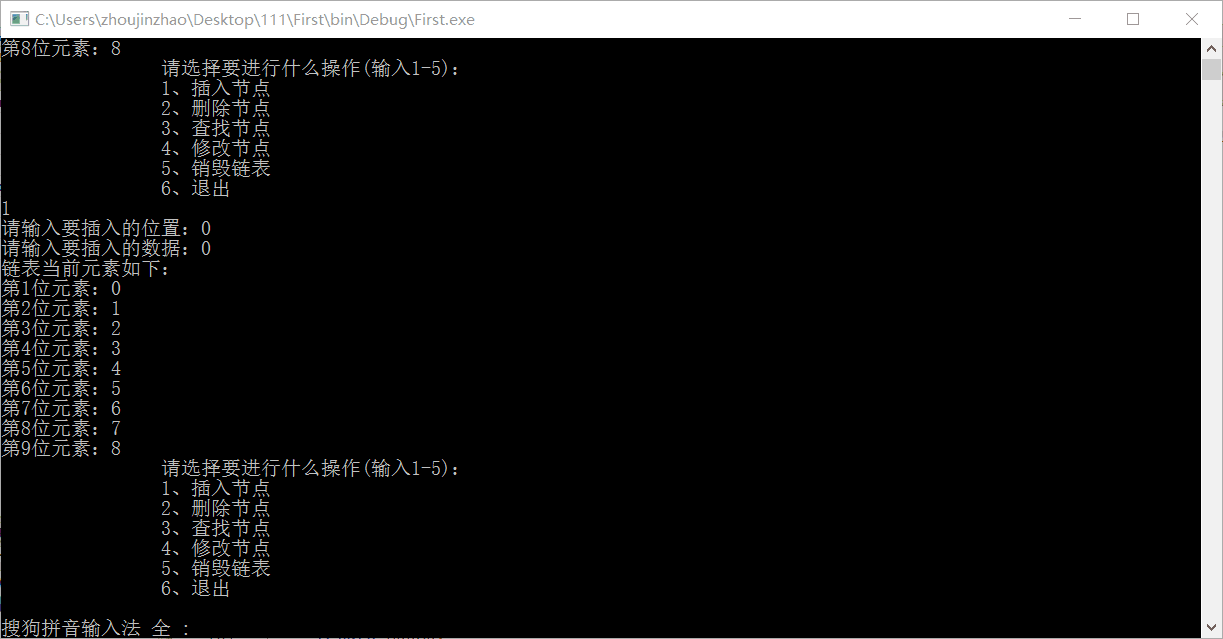
单链表

1. **运行截图**
2. 生成链表和链表排序：

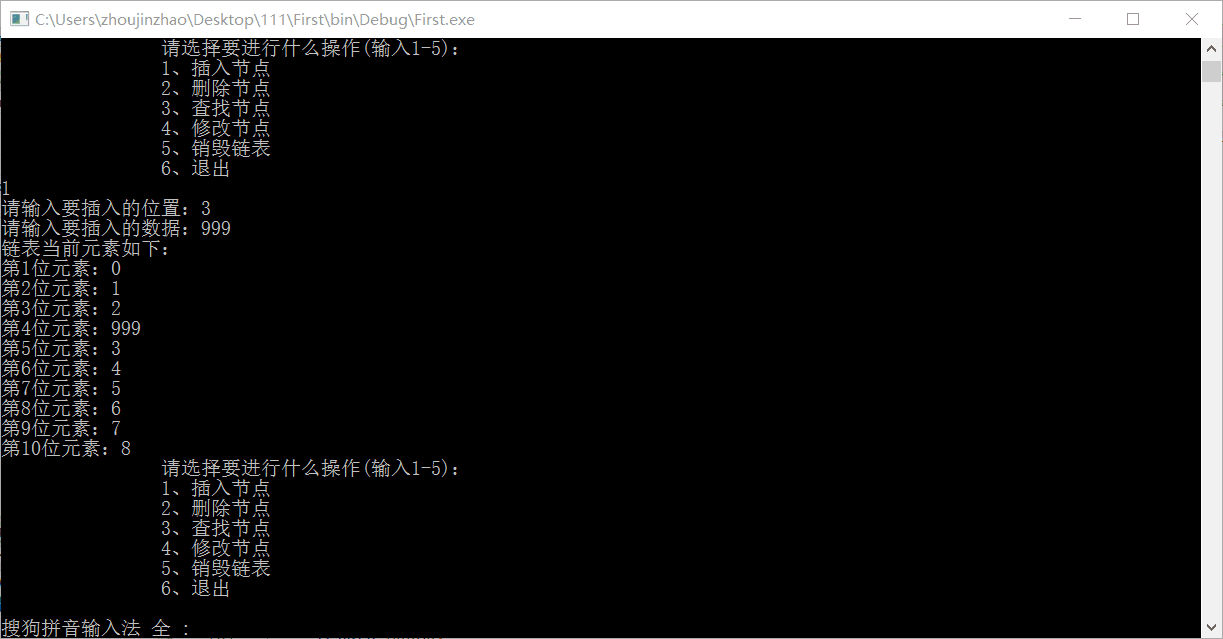


(2)插入节点：

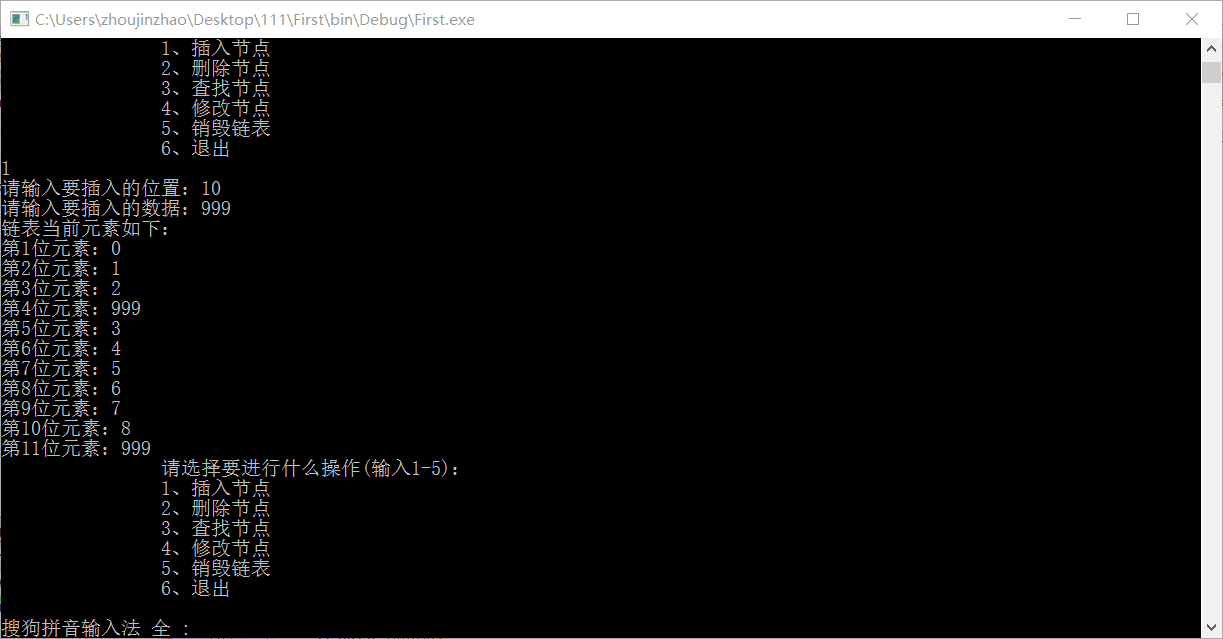
0位置（链表开头）



中间位置



尾部

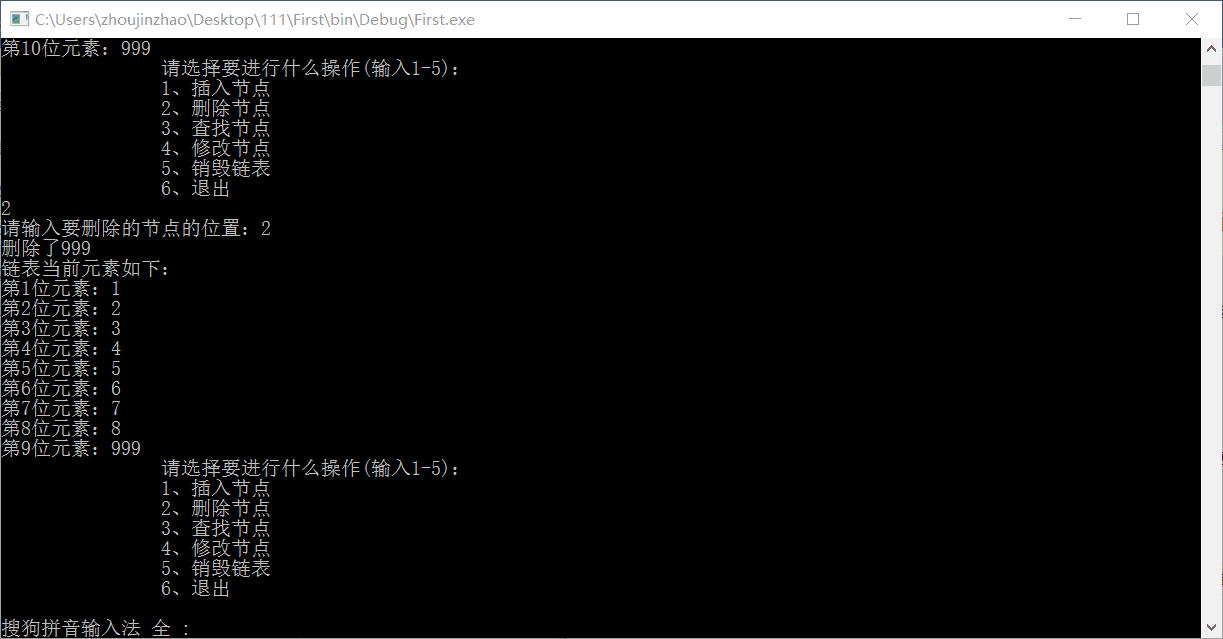


(2)删除节点：

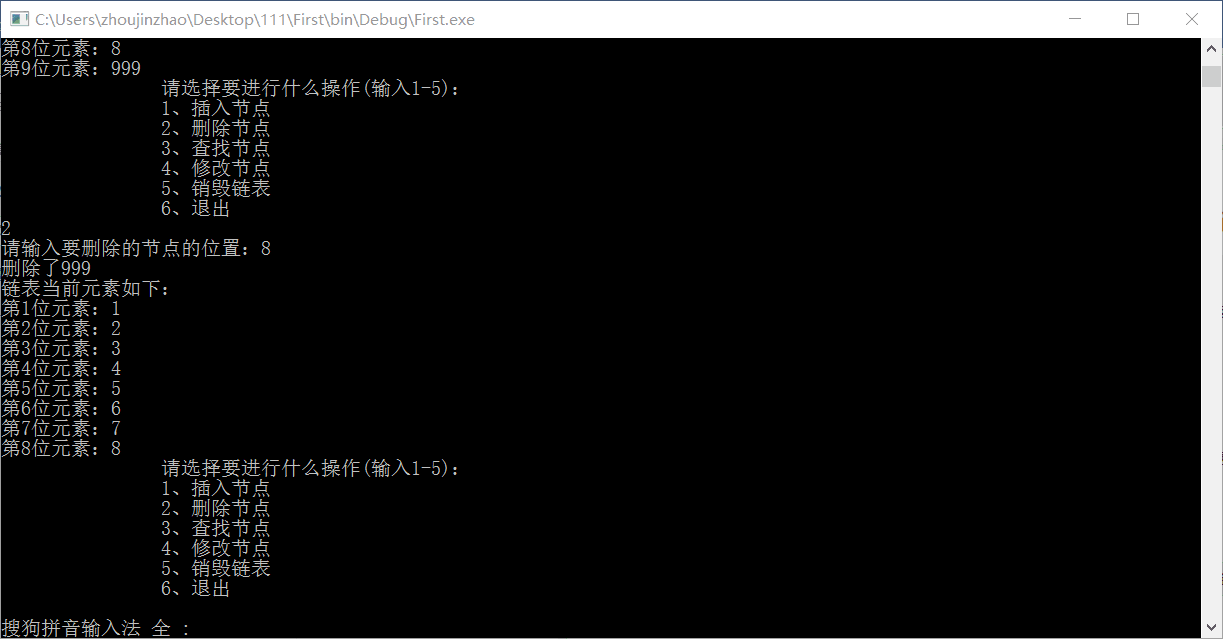
链表第一位



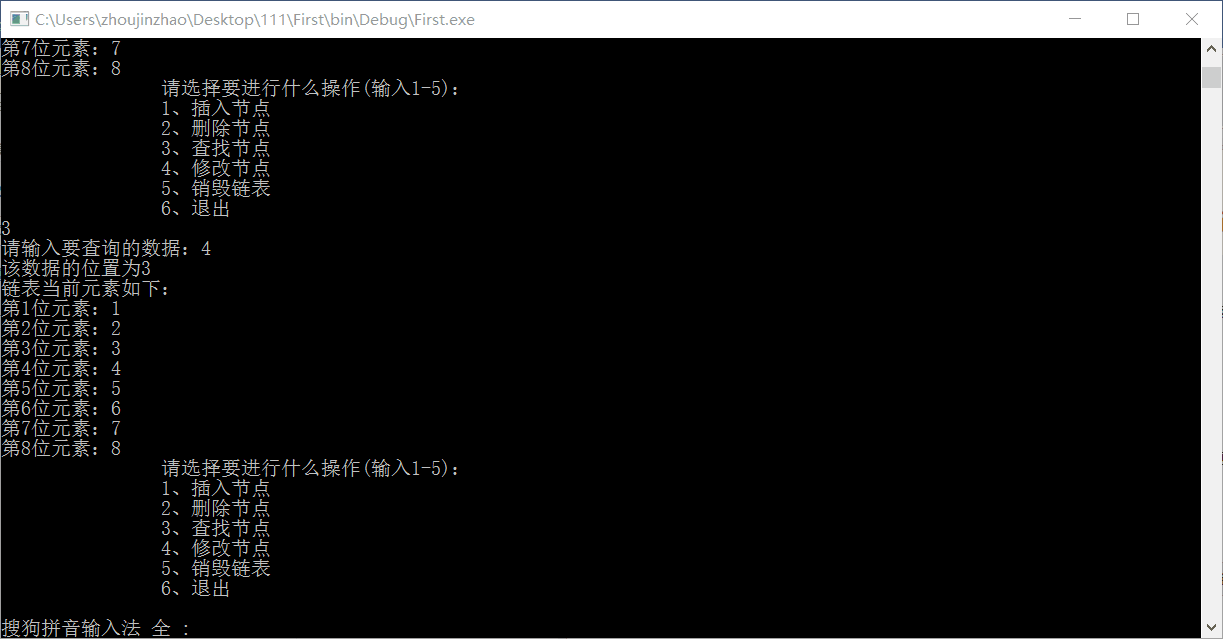
中间



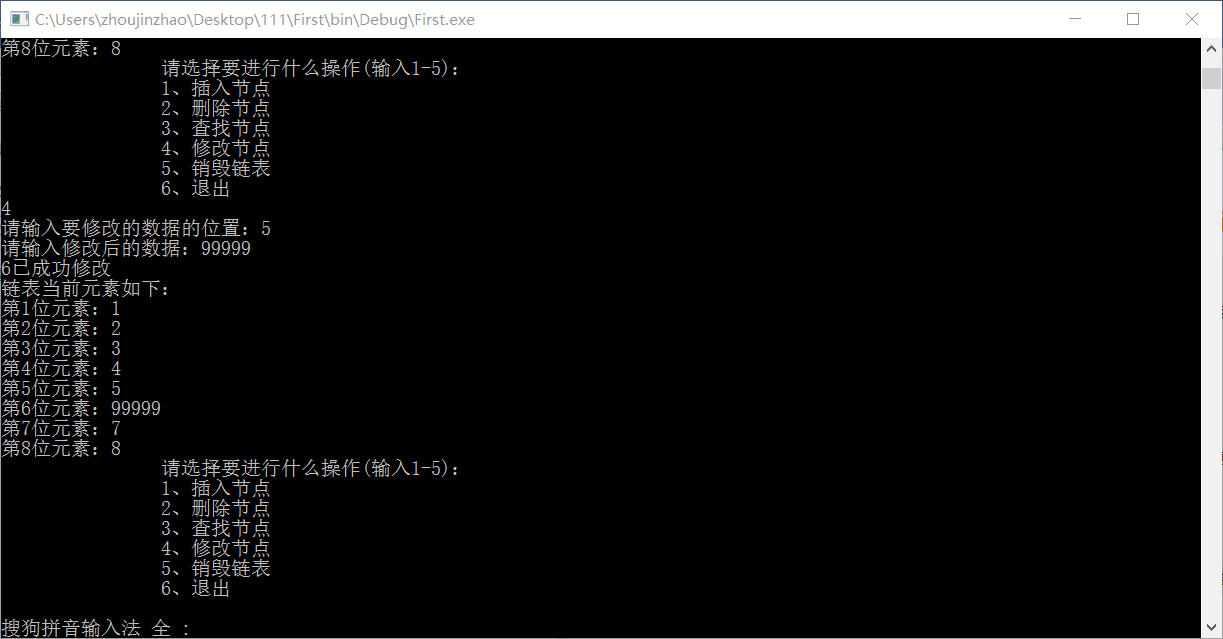
尾部



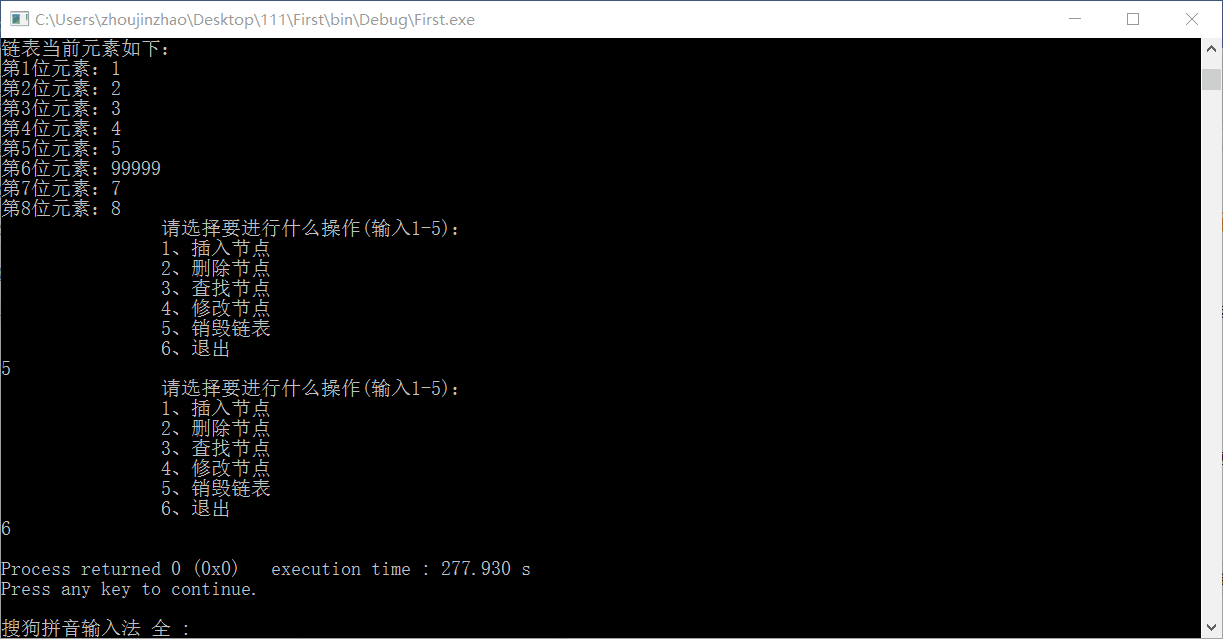
(3)查询数据：



(4)修改节点数据：



(5)销毁链表:



**二、源代码**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node \*next;

}Node, \*ptr\_Node;

typedef enum Status {

SUCCESS, ERROR

}Status;

//根据数组生成一条链表，成功则返回头节点，失败返回 NULL

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

{

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

int i=0;

ptr\_Node p,q,head;

p=q=(ptr\_Node)malloc(sizeof(Node));

head=NULL;

p->data=\*arr; //读取数组数据

while(i<n)

{

if(i==0)

{

head=p; //当i=0时读取到头指针

}

else

{

q->next=p; //当i!=0时，构建链接

}

q=p;i++;

p=(ptr\_Node)malloc(sizeof(Node));

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

}

//销毁给定的链表

void destroy(ptr\_Node head)

{

ptr\_Node 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

}

//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(ptr\_Node \*head, ptr\_Node 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\_Node p;

node=(ptr\_Node)malloc(sizeof(Node));//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;

}

//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 //删除成功返回 SUCCESS，其他情况返回 ERROR

Status delete(ptr\_Node \*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\_Node 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;

}

//在链表中查找节点值与 data 相等的节点，并返回找到的第一个节点的前一个节点的位置 //（例：头节点相等，返回 0），没找到或者其他情况返回-1

ptr\_Node search(ptr\_Node head, int data)

{

if(head==NULL) return -1; //当头指针为NULL

ptr\_Node p,q;

p=head;int ad=0,k=0; //ad为位置 k用来判断是否找到指定数据

while(p!=NULL)

{

if(p->data==data) //找到指定数据时

{

q=p;k++;break; //q为指定数据的地址（我写这里备用的，可以用来修改）

} //找到指定数据k值变为1

ad++;

p=p->next;

}

if(k==0) return -1; //没有找到指定数据

return ad; //返回指定数据的位置

}

//将链表中 index 位点后面的结点的值修改为(\*data)，将原值保存到(\*data) //修改成功返回 SUCCESS，其他情况返回 ERROR

Status edit(ptr\_Node head, int index, int \*data)

{

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

ptr\_Node p,q;

p=head;

int n=0,temp; //n为节点位置 temp用来交换

while(p!=NULL)

{

if(n==index)

{

temp=p->data;p->data=\*data;\*data=temp;//进行交换

return SUCCESS;

}

n++; //位置+1

p=p->next; //指向下一节点

}

return ERROR;

}

//将链表结点值按照一定的格式输出（每行多少个自己定义，格式越漂亮越好）

void print(ptr\_Node head)

{

if(head==NULL) return;

ptr\_Node 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

}

}

//此题选做,实现单链表按照节点值大小升序排序

Status sort(ptr\_Node \*head)

{

ptr\_Node p,q,k;int temp;

p=\*head;

while(p!=NULL)

{

q=p;k=p;

while(q!=NULL)

{

if((q->data)<(k->data)) k=q;

q=q->next;

}

if((k->data)<(p->data))

{

temp=p->data;p->data=k->data;k->data=temp;

}

p=p->next;

}

}

int main()

{

int a[8]={8,7,6,5,4,3,2,1}; //数组

ptr\_Node head,node; //node为插入的新节点

int index,temp,choice;

head=create(a,8); //转化数组为链表

print(head); //输出当前链表

sort(&head);

print(head);

while(1)

{

printf("\t\t请选择要进行什么操作(输入1-5)：\n");

printf("\t\t1、插入节点\n");

printf("\t\t2、删除节点\n");

printf("\t\t3、查找节点\n");

printf("\t\t4、修改节点\n");

printf("\t\t5、销毁链表\n");

printf("\t\t6、退出\n");

scanf("%d",&choice); //输入要选择的操作

if(choice<0||choice>6)

{

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(&head,node,index)==1) printf("插入失败！\n"); //插入函数

print(head);break;

case 2: printf("请输入要删除的节点的位置：");

scanf("%d",&index); //index为要删除的节点的前一个位置

if(delete(&head,index,&temp)==1) print("删除失败!\n"); //删除函数

else printf("删除了%d\n",temp);

print(head);break;

case 3: printf("请输入要查询的数据：");

scanf("%d",&index); //index为要查询的数据

int ad=search(head,index); //定义ad接收查询函数返回的结果

if(ad==-1) printf("该数据不存在\n");

else printf("该数据的位置为%d\n",ad);

print(head);break;

case 4: printf("请输入要修改的数据的位置：");

scanf("%d",&index); //index为要修改的数据的位置

printf("请输入修改后的数据：");

scanf("%d",&temp); //temp为要改成的数据

int ed=edit(head,index,&temp);//定义ed接收修改函数返回的结果

if(ed==1) printf("修改错误！\n");

else printf("%d已成功修改\n",temp);

print(head);break;

case 5: destroy(head);break; //销毁链表

case 6: return 0; //结束程序

}

}

return 0;

}