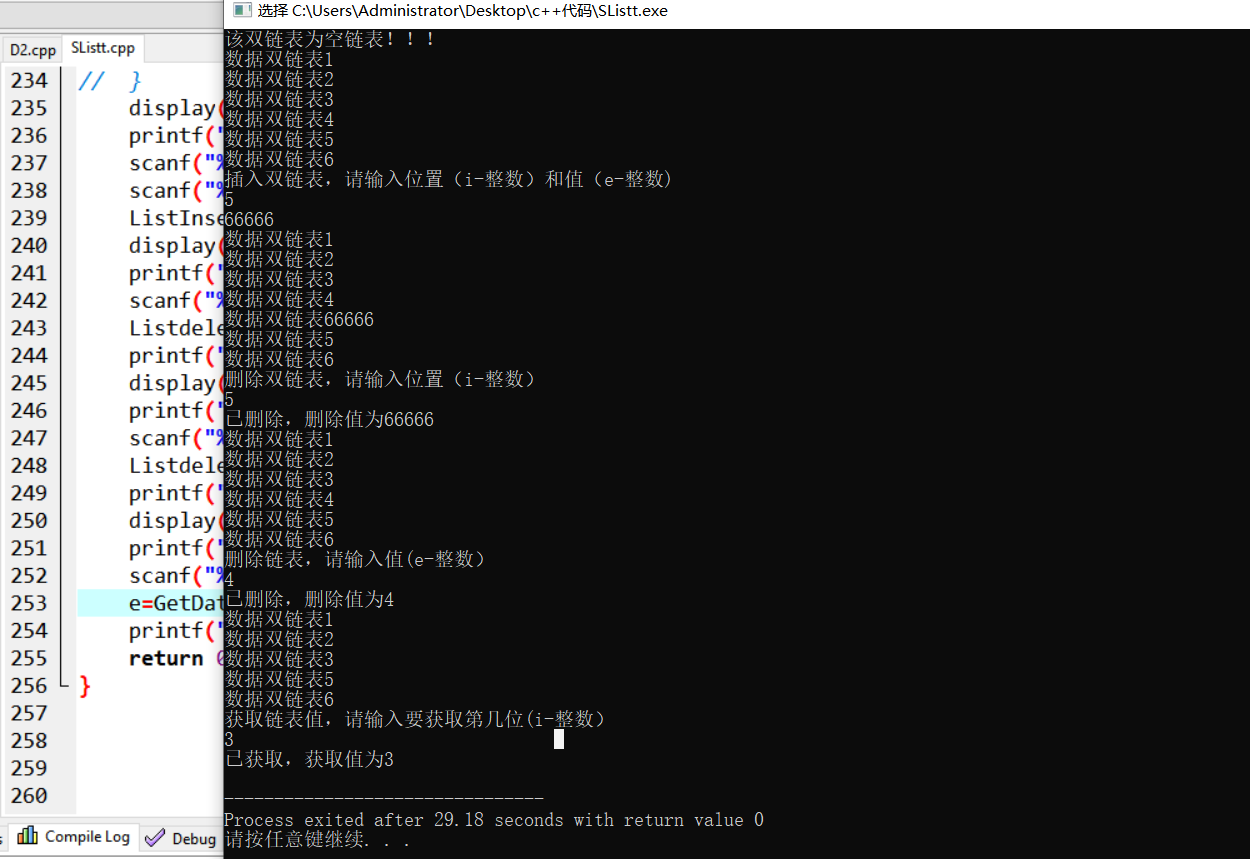
双链表



#include<stdio.h>

#include<malloc.h>

typedef struct LNode{

int data;

struct LNode \*next,\*prior;

}LNode,\*LinkList;

//定义一个获取数据的函数

int GetDate(LinkList L,int i) {

int j=1;

LNode \*p=L->next;

if(i==0){

return 0;

}

if(i<1)

return NULL;

while(p!=NULL && j<i){

p=p->next;

j++;

}

return p->data;

}

//初始化一个不带头节点单链表函数

bool InitList(LinkList &L){

L=NULL;

return true;

}

//初始化一个带头节点单链表函数

bool InitList1(LinkList & L){

L=(LNode \*)malloc(sizeof(LNode));

L->next=NULL;

L->prior=NULL;

return true;

}

//判断单链表是否为空的函数

bool Empty(LinkList L){

if(L->next==NULL)

return true;

else

return false;

}

//插入函数操作

bool ListInsert(LinkList &L,int i,int e){

if(i<1){

return false;

}

//如果不带头节点的话，

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// if(i==1){

// LNode\*s=(LNode \*)malloc(sizeof(LNode));

// s->data=e;

// L->s;

// return true;

// }

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LNode \*p;

int j=0;

p=L;

while(p!=NULL&&j<i-1){

p=p->next;

j++;

}

// printf("%d",j);

if(p==NULL)

return false;

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

s->data=e;

// printf("%d",e);

if(p->next==NULL) {

p->next=s;

s->prior=p;

s->next=NULL;

}

else{

s->next=p->next;

p->next->prior=s;

p->next=s;

s->prior=p;

}

return true;

}

//在p节点之后插入元素e

bool InsertNextNode(LNode \*p,int e) {

if(p==NULL){

return false;

}

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

s->data=e;

s->next=p->next;

p->next=s;

s->prior=p;

s->next->prior=s;

return true;

}

//在p节点之前插入元素e

bool InsertPriorNode(LNode \*p,int e) {

if(p==NULL){

return false;

}

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

s->data=e;

s->prior=p->prior;

p->prior->next=s;

s->next=p;

p->prior=s;

// s->data=p->data;

// p->data=e;

// s->next=p->next;

// p->next=s;

// return true;

}

//删除操作，按值查找

bool Listdelete1(LinkList &L,int &e){

LNode\* p;

// LNode\* q;

p=L;

if(p==NULL)

return false;

while(p->next!=NULL&&p->data!=e){

p=p->next;

}

if(p==NULL)

return false;

if(p->next==NULL&&p->data==e){

free(p);

return true;

}

else{

e=p->data;

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

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

// q=p->next;

// e=p->data;

// p->data=q->data;

// if(q==NULL){

// free(q);

// return q;

// }

free(p);

return true;

}

}

//删除操作，按位置删除

bool Listdelete(LinkList &L,int i,int & e){

LNode\* p,\*q;

int j=0;

p=L;

if(p==NULL)

return false;

while(p->next!=NULL&&j<i-1){

p=p->next;

j++;

}

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

if(p==NULL){

return false;

}

if(p->next==NULL){

return false;

}

q=p->next;

e=p->next->data;

// printf("%d\n",e);

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

q->prior=p;

free(q);

return true;

}

//显示函数

void display(LinkList L) {

if(L->next!=NULL){

while((L->next)!=NULL){

L=L->next;

printf("数据双链表%d\n",L->data);

}

}

else{

printf("数据双链表空");

}

}

int main(){

int i=0;

int e=0;

LinkList L;

InitList1(L);

if(Empty(L)==true){

printf("该双链表为空链表！！！\n");

//printf("111111111");

}

if(Empty(L)==false){

printf("该双链表不为空链表！！！\n");

//printf("44444444");

}

ListInsert(L,1,1);

ListInsert(L,2,2);

ListInsert(L,3,3);

ListInsert(L,4,4);

ListInsert(L,5,5);

ListInsert(L,6,6);

// if(Empty(L)){

// printf("该链表为空链表！！！\n");

// }

// if(!Empty(L)){

// printf("该链表不为空链表！！！\n");

// //printf("44442222222222222222244");

// }

display(L);

printf("插入双链表，请输入位置（i-整数）和值（e-整数)\n");

scanf("%d",&i);

scanf("%d",&e);

ListInsert(L,i,e);

display(L);

printf("删除双链表，请输入位置（i-整数）\n");

scanf("%d",&i);

Listdelete(L,i,e);

printf("已删除，删除值为%d\n",e);

display(L);

printf("删除链表，请输入值(e-整数）\n");

scanf("%d",&e);

Listdelete1(L,e);

printf("已删除，删除值为%d\n",e);

display(L);

printf("获取链表值，请输入要获取第几位(i-整数）\n");

scanf("%d",&i);

e=GetDate(L,i);

printf("已获取，获取值为%d\n",e);

return 0;

}