//双链表基本运算算法

/\*利用rand函数获得随机数，动态数组存储元素，调用函数创建双链表L1，初始化L2

调用函数获得元素，去模2判断是否为偶数，将偶数插入链表L2，存储偶数的位置，最后从L1后面逆序删除偶数\*/

#include <stdio.h>

#include<time.h>

#include<stdlib.h>

#include <malloc.h>

typedef int ElemType;

typedef struct DNode //定义双链表结点类型

{

ElemType data;

struct DNode\* prior; //指向前驱结点

struct DNode\* next; //指向后继结点

} DLinkNode;

void CreateListF(DLinkNode\*& L, ElemType a[], int n)

//头插法建双链表

{

DLinkNode\* s;

L = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建头结点

L->prior = L->next = NULL;

for (int i = 0; i < n; i++)

{

s = (DLinkNode\*)malloc(sizeof(DLinkNode));//创建新结点

s->data = a[i];

s->next = L->next; //将结点s插在原开始结点之前,头结点之后

if (L->next != NULL) L->next->prior = s;

L->next = s; s->prior = L;

}

}

void CreateListR(DLinkNode\*& L, ElemType a[], int n)

//尾插法建双链表

{

DLinkNode\* s, \* r;

L = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建头结点

L->prior = L->next = NULL;

r = L; //r始终指向终端结点,开始时指向头结点

for (int i = 0; i < n; i++)

{

s = (DLinkNode\*)malloc(sizeof(DLinkNode));//创建新结点

s->data = a[i];

r->next = s; s->prior = r; //将结点s插入结点r之后

r = s;

}

r->next = NULL; //尾结点next域置为NULL

}

void InitList(DLinkNode\*& L)

{

L = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建头结点

L->prior = L->next = NULL;

}

void DestroyList(DLinkNode\*& L)//释放链表

{

DLinkNode\* pre = L, \* p = pre->next;

while (p != NULL)

{

free(pre);

pre = p;

p = pre->next;

}

free(pre);

}

bool ListEmpty(DLinkNode\* L)

{

return(L->next == NULL);

}

int ListLength(DLinkNode\* L)

{

DLinkNode\* p = L;

int i = 0;

while (p->next != NULL)

{

i++;

p = p->next;

}

return(i);

}

void DispList(DLinkNode\* L)//打印链表

{

DLinkNode\* p = L->next;

while (p != NULL)

{

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

p = p->next;

}

printf("\n");

}

bool GetElem(DLinkNode\* L, int i, ElemType& e)

{

int j = 0;

DLinkNode\* p = L;

if (i <= 0) return false; //i错误返回假

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

{

j++;

p = p->next;

}

if (p == NULL)

return false;

else

{

e = p->data;

return true;

}

}

int LocateElem(DLinkNode\* L, ElemType e)

{

int n = 1;

DLinkNode\* p = L->next;

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

{

n++;

p = p->next;

}

if (p == NULL)

return(0);

else

return(n);

}

bool ListInsert(DLinkNode\*& L, int i, ElemType e)//插入结点

{

int j = 0;

DLinkNode\* p = L, \* s;

if (i <= 0) return false; //i错误返回假

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

{

j++;

p = p->next;

}

if (p == NULL) //未找到第i-1个结点

return false;

else //找到第i-1个结点p

{

s = (DLinkNode\*)malloc(sizeof(DLinkNode)); //创建新结点s

s->data = e;

s->next = p->next; //将结点s插入到结点p之后

if (p->next != NULL)

p->next->prior = s;

s->prior = p;

p->next = s;

return true;

}

}

bool ListDelete(DLinkNode\*& L, int i, ElemType& e)//删除结点

{

int j = 0;

DLinkNode\* p = L, \* q;

if (i <= 0) return false; //i错误返回假

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

{

j++;

p = p->next;

}

if (p == NULL) //未找到第i-1个结点

return false;

else //找到第i-1个结点p

{

q = p->next; //q指向要删除的结点

if (q == NULL)

return false; //不存在第i个结点

e = q->data;

p->next = q->next; //从单链表中删除\*q结点

if (p->next != NULL) p->next->prior = p;

free(q); //释放q结点

return true;

}

}

int main() {

srand((unsigned)time(NULL));

int n;//定义元素总数

n = rand() % 7 + 1;//获得随机数

int\* p, \* p1;//定义动态数组

p = (int\*)malloc(sizeof(int) \* n);

p1 = (int\*)malloc(sizeof(int) \* n);

int i;

printf("元素总数为:%d\n", n);

printf("元素为:");

for (int i = 0; i < n; i++) {

p[i] = rand() % 10;//获得随机元素

p1[i] = 0;//初始化数组

printf("%d ", p[i]);

}

printf("\n");

DLinkNode\* L1, \* L2;//创建头指针

CreateListR(L1, p, n);//调用函数，创建双链表

printf("双链表L1为:");

DispList(L1);//打印链表L1

InitList(L2);//调用函数

i = 1;

int j = 1, temp;

while (i <= n) {

if (GetElem(L1, i, temp)) {

if (temp % 2 == 0) {//判断是否为偶数

ListInsert(L2, j, temp);//插入链表

j++;

p1[j - 1] = i;//存储L1中偶数位置

}

}

i++;

}

for (; j> 1; j--) {

ListDelete(L1, p1[j - 1], temp);//从链表L1最后开始删除偶数

}

printf("仅为偶数的双链表L2:");

DispList(L2);//打印链表L1

printf("删去偶数的双链表L1:");

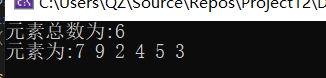
DispList(L1);//打印链表L2

DestroyList(L1);//释放链表L1内存

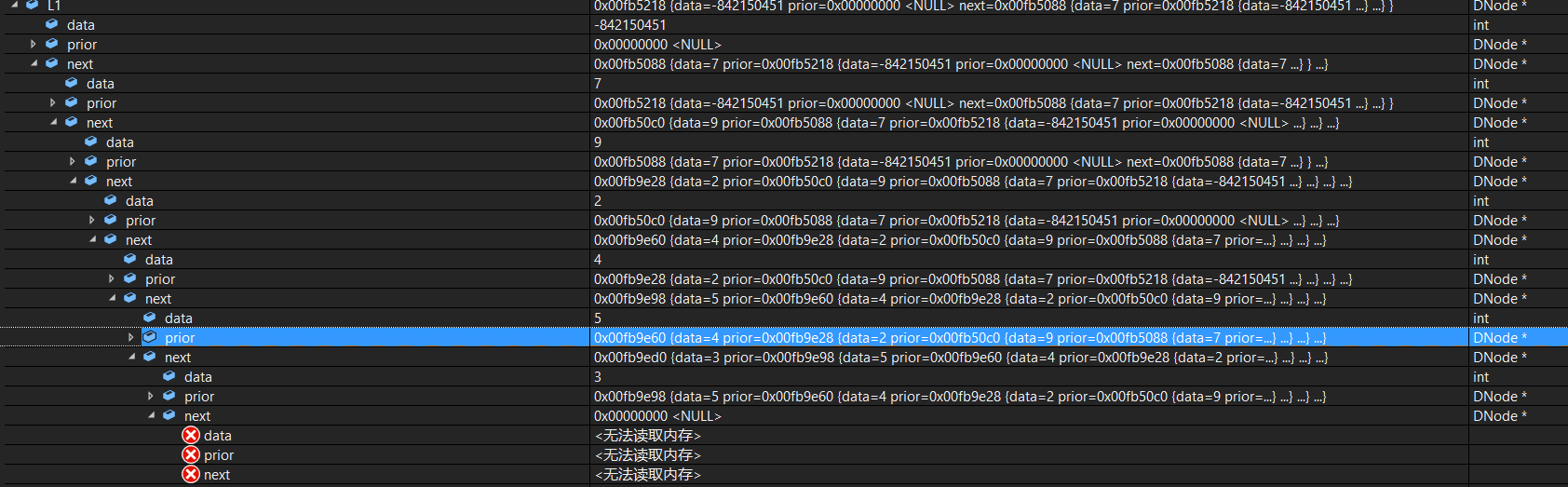
DestroyList(L2);//释放链表L2内存

return 0;

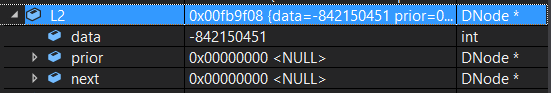
}



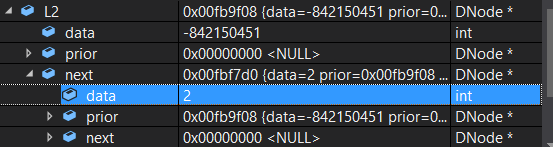
创建链表L1



链表L2初始化



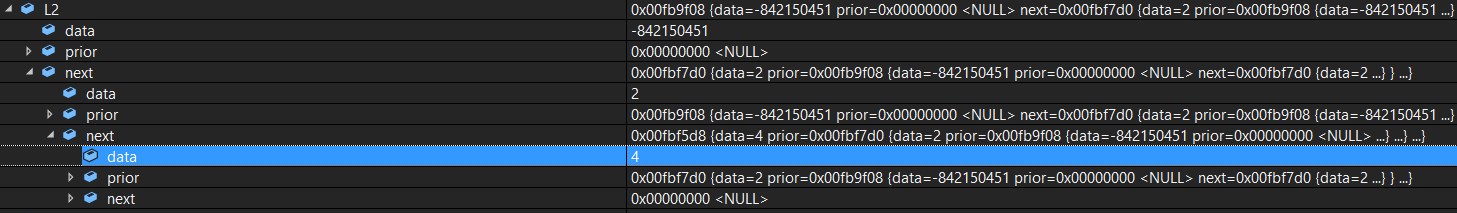
插入偶数2



元素2在L1中的位置



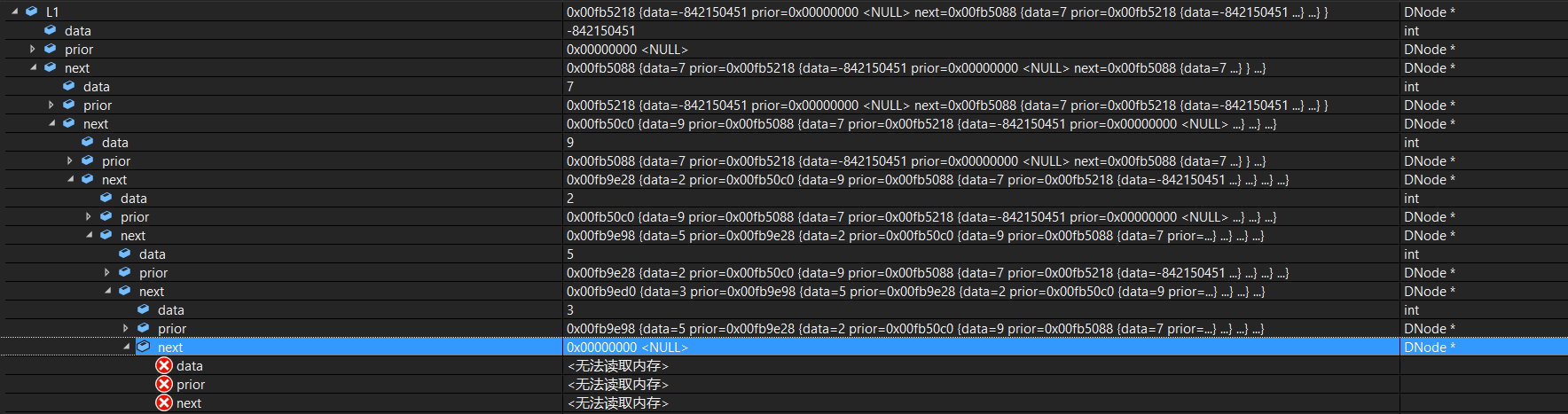
插入偶数4



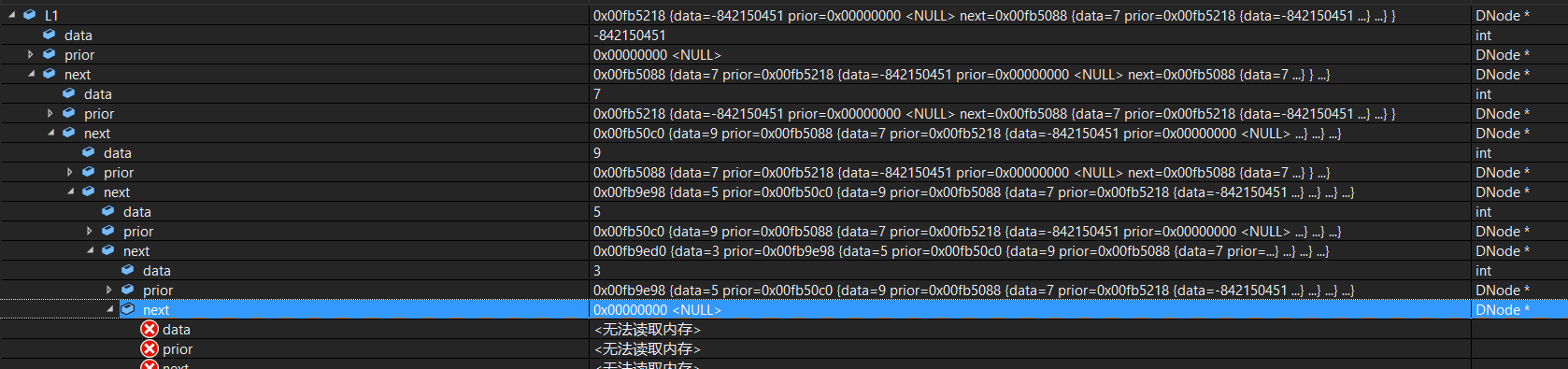
元素4在L1中的位置



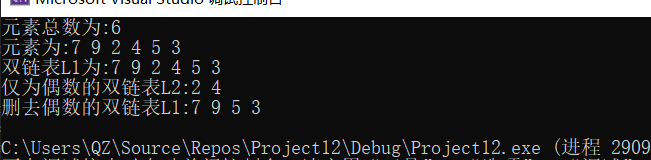
删除4



删除2



打印链表L1，L2



释放内存

