#define \_CRT\_SECURE\_NO\_WARNINGS 1

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<stdio.h>

#include<stdlib.h>

// 定义链表结点结构

typedef struct Node

{

int data;

struct Node\* next;

} LinkNode;

// 创建包含 n 个结点的链表

LinkNode\* Create(LinkNode\* L, int n)

{

int i;

LinkNode\* p;

// 为链表头结点分配内存

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

L->next = NULL;

LinkNode\* tail = L;

// 循环创建 n 个结点，并将其添加到链表中

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

{

p = (LinkNode\*)malloc(sizeof(LinkNode));

p->data = i;

p->next = NULL;

tail->next = p;

tail = p;

}

return L;

}

// 反转链表

LinkNode\* back(LinkNode\* L)

{

// 如果链表为空或只有一个结点，则无需反转

if (L == NULL || L->next == NULL)

{

return L;

}

LinkNode\* pre = NULL;

LinkNode\* p = L->next;

LinkNode\* q = L->next->next;

// 遍历链表，反转指针方向

while (q != NULL)

{

p->next = pre;

pre = p;

p = q;

q = q->next;

}

p->next = pre;

L->next = p;

return L;

}

int main()

{

LinkNode\* p;

LinkNode\* L = NULL;

// 创建包含 6 个结点的链表

L = Create(L, 6);

printf("初始链表: ");

// 打印初始链表的元素

p = L->next;

while (p != NULL)

{

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

p = p->next;

}

printf("\n");

// 反转链表

L = back(L);

printf("逆置后的链表: ");

// 打印逆置后链表的元素

p = L->next;

while (p != NULL)

{

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

p = p->next;

}

printf("\n");

return 0;

}