

# 题目

设有一线性表，设计一个算法将线性表逆置，要求占用原线性表空间，使用顺序和单链表两种方法表示。

## 一、顺序表

### 1.数据结构

- 1. **结构说明**此时数据的逻辑结构为线性结构，存储结构为顺序存储。
- 2. **变量说明**使用typedef int ElemType;为int 起了一个别名，提高程序可读性，且便于修改。  
LIST\_SIZE为顺序表最大长度。  
Sqlist为构建的结构体变量名。  
n用于接收测试数据的顺序表长度。L为Sqlist类型的指针用于接收顺序表list的地址。  
first和last分别记录顺序表中第一个和最后一个元素的位置。
- 3. **函数说明**initlist (Sqlist \*L) 用于初始化顺序表，即将顺序表长度置0。  
createlist (Sqlist \* L, int n) 用于向顺序表中写入测试数据。  
swaplist (Sqlist \* L,int first,int last) 用于将顺序表逆置。  
printlist (Sqlist \* L) 用于输出顺序表。

### 2. 测试样例设计

- 使用1-10的自然数做测试样例。

### 3.核心算法

- 由于顺序表中的逻辑顺序与物理顺序一致，直接采用类似于数组的方式直接交换即可。

#### 当元素个数为奇数时

例如1 2 3 4 5,最后交换的是2 4，即 $i < j$ 。

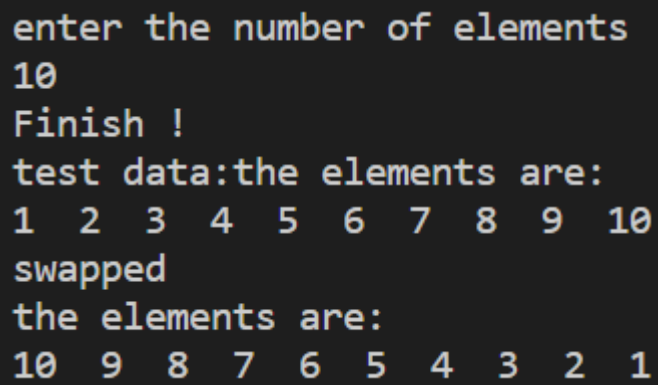
#### 当元素个数为偶数时

例如1 2 3 4 5 6，最后交换的是3 4，即 $i < j$ 。

综合下来就是  $i < j$ 。

```
//顺序存储
for(int i = first,j=last;i < j;i++,j--)
{
    temp = L->data[i];
    L->data[i] = L->data[j];
    L->data[j] = temp;
}
```

## 4. 运行结果截图



```
enter the number of elements
10
Finish !
test data:the elements are:
1 2 3 4 5 6 7 8 9 10
swapped
the elements are:
10 9 8 7 6 5 4 3 2 1
```

## 5. 完整代码

```

#include<stdio.h>
#define LIST_SIZE 1024
typedef int ElemType;
typedef struct
{
    ElemType data[LIST_SIZE];
    int last;
}Sqlist;
void initlist(Sqlist *L);//初始化顺序表
void createlist(Sqlist *L,int n);//写入测试样例
void swap(Sqlist *L,int first,int list);//交换函数
void printlist(Sqlist *L);//输出元素
int main()
{
    int n = 0;
    int first = 0,last;
    Sqlist *L = NULL;
    Sqlist list;
    L = &list;
    initlist(L);//初始化
    printf("enter the number of elements\n");
    scanf("%d",&n);
    last = n-1;
    createlist(L,n);//写入测试数据
    printf("test data:");
    printlist(L);//打印测试数据
    swap(L,first,last);
    printf("swapped\n");
    printlist(L);
    return 0;
}

void initlist(Sqlist *L)
{
    L->last = 0;
}
void createlist(Sqlist *L,int n)
{
    int i;
    L->last = n;
    for(i = 0;i < n;i++)
    {
        L->data[i]=i+1;
    }
    printf("Finish ! \n");
}
void swap(Sqlist *L,int first,int last)
{
    int temp = 0;
    for(int i = first,j = last;i < j;i++,j--)
    {
        temp = L->data[i];
        L->data[i] = L->data[j];
        L->data[j] = temp;
    }
}

```

```

    }
}
void printlist(Sqlist *L)
{
    int i;
    printf("the elements are:\n");
    for(i = 0; i < L->last; i++)
        printf("%d ", L->data[i]);
    putchar('\n');
}

```

## 二、单链表

### 1. 数据结构

1. **结构说明** 此时数据的逻辑结构是线性结构，存储结构为链式存储。
2. **变量说明** LNode与\*linklist为所构建的结构体的变量名,L为LNode类型的一个指针用于接收建立的单链表的头结点。  
n为测试数据的元素个数。  
head为LNode类型的指针，用于返回单链表的头结点。
3. **函数说明** createlist (n) 用于建立有n个结点的单链表。  
printlist (linklist L) 用于输出链表。  
swaplist(linklist L)用于逆置链表。

### 2. 测试样例

- 采用1-10的自然数作为测试样例。

### 3. 核心算法

- 由于是链式存储，只能单向读取，故选择使用头插法，将后继元素插入到前驱元素之前。

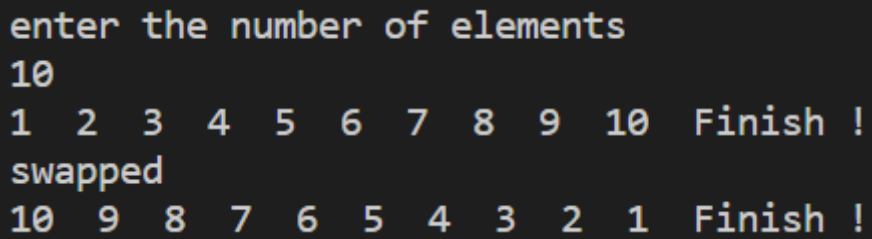
例：

假设这是一个单链表：a b c d e f，只需要将b插入到a之前即可。

即：第一次：b a c d e f 第二次：c b a d e f,依此类推直到f e d c b a。

```
//核心算法
void swaplist(linklist L)
{
    LNode *p,*q;
    p = L->next;
    L->next = NULL;
    while(p)
    {
        q = p;
        p = p->next; //后移
        q->next = L->next;
        L->next = q;
    }
}
```

#### 4. 运行结果截图



```
enter the number of elements
10
1 2 3 4 5 6 7 8 9 10 Finish !
swapped
10 9 8 7 6 5 4 3 2 1 Finish !
```

#### 5. 完整代码

```

#include<stdio.h>
#include<malloc.h>
typedef int ElemType;
typedef struct node
{
    int data;
    struct node *next;
}LNode,*linklist;//前者表示结点，后者表示一个链表

linklist createlist(int n);//建立单链表
void printlist(linklist L);//输出链表
void swaplist(linklist L);//逆置链表
int main()
{
    linklist L,C;
    int n;
    printf("enter the number of elements\n");
    scanf("%d",&n);
    L = createlist(n);
    printlist(L);
    swaplist(L);
    printf("swapped\n");
    printlist(L);
    return 0;
}

linklist createlist(int n)
{
    LNode *head,*p,*q;
    int i;
    head = (LNode*)malloc(sizeof(LNode));
    head->next = NULL;//头结点
    p = head;
    for(i = 0;i<n;i++)
    {
        q = (LNode*)malloc(sizeof(LNode));
        q->data = i+1;//用1-10的自然数作为测试样例
        p->next = q;
        p = q;
    }
    p->next = NULL;
    return head;//返回链表的首地址
}

void printlist(linklist L)
{
    L = L->next;//跳过头结点
    while(L)
    {
        printf("%d ",L->data);
        L = L->next;//依次向后移动
    }
    printf("Finish ! \n");
}

void swaplist(linklist L)

```

```
{
    LNode *p,*q;
    p = L->next;
    L->next = NULL;
    while(p)
    {
        q = p;
        p = p->next;//后移
        q->next = L->next;
        L->next = q;
    }
}
```