结构体申明

```
#include<stdio.h>
#include<stdlib.h>


//数据结构定义，LNode一般是指单个节点，LinkList指的是所有节点
typedef struct LNode{
    int data;
    struct LNode *next;
}LNode, *LinkList;
```

头插法建立单链表

```
//采用头插法建立单链表，逆向建立单链表
LinkList List_HeadInsert(LinkList &L){
    LNode *s;                               //s节点存放需要插入的节点
    int x;
    L = (LinkList)malloc(sizeof(LNode));    //初始化头结点
    L->next = NULL;
    scanf("%d",&x);
    while(x!=9999){                         //输入9999代表结束
        s = (LNode *)malloc(sizeof(LNode)); //创建新的节点(指针节点)
        s -> data = x;                      //插入新的节点
        s -> next = L -> next;
        L -> next = s;
        scanf("%d",&x);                     //再次循环获取需要插入节点的值
    }
    return L;
}
```

尾插法建立单链表

```
//采用尾插法建立单链表，正向建立单链表
LinkList List_TailInsert(LinkList &L){
    LNode *s,*r = L;
    int x;
    L = (LinkList)malloc(sizeof(LNode));    //初始化头结点
    L->next = NULL;
    scanf("%d",&x);
    while(x!=9999){                         //输入9999代表结束
        s = (LNode *)malloc(sizeof(LNode)); //创建新的节点(指针节点)
        s -> data = x;                      //插入新的节点
        r  = s;
        r = s;                              //r指向s，再次回到链表尾端
        scanf("%d",&x);                     //再次循环获取需要插入节点的值
    }
    r -> next = NULL;
    return L;
}
```

按序号查找元素

```
//按序号查找结点值
LNode *GetElem(LinkList L, int i){
    int j = 1;
    LNode *p = L->next;                    //这里是带头结点的链表表示方法
    if(i == 0) return L;                   //i=0，返回头结点
    if(i < 1) return NULL;                 //i不在取值范围，返回NULL
    while(p && j < i){                     //保证p不为空，且未到所找的第i个节点
        p = p->next;
        j++;
    }
    return p;
}
```

按值查找

```
//按值查找
LNode *LocateElem(LinkList L, int value){
    LNode *p = L->next;
    while(p && p -> data != value){        //从第一个节点查找data域为e的节点
        p = p->next;
    }
    return p;
}
```

插入单链表第i个位置

```
//插入单链表第i个位置节点
void InsertLNode(LinkList &L,int i,int value){
    LNode *s = (LNode *)malloc(sizeof(LNode));  //创建新的节点(指针节点)
    LNode *p = GetElem(L,i-1);
    s -> data = value;                          //插入新的节点
    s ->next = p->next;
    p ->next = s;
}
```

前插操作

```
//前插操作，交换数据域即可
void FrontInsertLNode(LinkList &L,int i,int value){
    LNode *s = (LNode *)malloc(sizeof(LNode));  //创建新的节点(指针节点)
    LNode *p = GetElem(L,i-1);
    s -> data = value;                          //插入新的节点
    s ->next = p->next;
    p ->next = s;
    //交换数据域
    int temp = s->data;
    s->data = p->data;
    p->data = s->data;
}
```

删除第i个节点

```
//删除第i个节点
void DeleteILNode(LinkList &L, int i){
    LNode *p = GetElem(L, i-1);
    LNode *q = p->next;
    p -> next = q->next;
    free(q);
}
```

删除指定节点

```
//删除指定节点，时间复杂度为O(1)
void DeletePointedLNode(LinkList &p){
    LNode *q = p->next;                 //直接将后一个节点值赋给前一个节点
    p->data = q->data;
    p->next = q->next;                  //然后删除后一个节点即可
    free(q);
}
```