

本节内容

单链表

建立



单链表的建立

尾插法

头插法



如果给你很多个数据元素 (ElemType) , 要把它们存到一个单链表里边, 咋 neng 呢?

Step 1: 初始化一个单链表

Step 2: 每次娶一个数据元素, 插入到表尾/表头

本节探讨带头
结点的情况

尾插法建立单链表

```
typedef struct LNode{  
    ElemType data;  
    struct LNode *next;  
}LNode, *LinkList;  
  
//初始化一个单链表 (带头结点)  
bool InitList(LinkList &L) {  
    L = (LNode *) malloc(sizeof(LNode)); //分配一个头结点  
    if (L==NULL) //内存不足, 分配失败  
        return false;  
    L->next = NULL; //头结点之后暂时还没有节点  
    return true;  
}  
  
void test(){  
    LinkList L; //声明一个指向单链表的指针  
    //初始化一个空表  
    InitList(L);  
    //.....后续代码.....  
}
```

//定义单链表结点类型
//每个节点存放一个数据元素
//指针指向下一个节点



尾插法建立单链表

```
//在第 i 个位置插插入元素 e (带头结点)  
bool ListInsert(LinkList &L, int i, ElemtType e){  
    if(i<1)  
        return false;  
    LNode *p; //指针p指向当前扫描到的结点  
    int j=0; //当前p指向的是第几个结点  
    p = L; //L指向头结点, 头结点是第0个结点 (不存数据)  
    while (p!=NULL && j<i-1) { //循环找到第 i-1 个结点  
        p=p->next;  
        j++;  
    }  
    if(p==NULL) //i值不合法  
        return false;  
    LNode *s = (LNode *)malloc(sizeof(LNode));  
    s->data = e;  
    s->next=p->next;  
    p->next=s; //将结点s连到p之后  
    return true; //插入成功  
}
```

尾插法建立单链表:

初始化单链表

设置变量 length 记录链表长度

While 循环 {

每次取一个数据元素 e;
ListInsert (L, length+1, e) 插到尾部;
length++;

}

每次都从头开始之后遍历，
时间复杂度为 $O(n^2)$



length=3

设置一个
表尾指针

↑
r

尾插法建立单链表

```
//后插操作：在p结点之后插入元素 e
bool InsertNextNode (LNode *p, ElemType e){
    if (p==NULL)
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    if (s==NULL) //内存分配失败
        return false;
    s->data = e; //用结点s保存数据元素e
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true;
}
```



设置一个表尾指针

后插操作

尾插法建立单链表

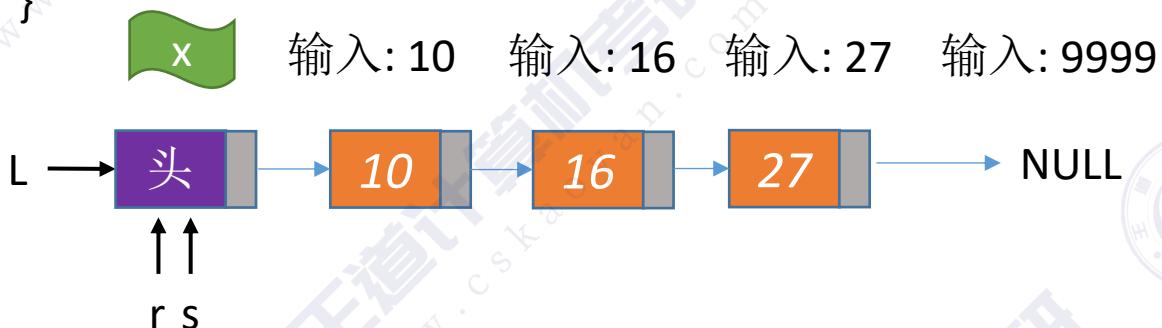
```
LinkList List_TailInsert(LinkList &L){    //正向建立单链表
    int x;                                //设ElemType为整型
    L=(LinkList)malloc(sizeof(LNode));      //建立头结点
    LNode *s,*r=L;                        //r为表尾指针
    scanf("%d",&x);                      //输入结点的值
    while(x!=9999){                      //输入9999表示结束
        s=(LNode *)malloc(sizeof(LNode));  //在r结点之后插入元素x
        s->data=x;
        r->next=s;
        r=s;
        scanf("%d",&x);
    }
    r->next=NULL;                        //尾结点指针置空
    return L;
}
```

在r结点之后插入元素 x

//r指向新的表尾结点

永远保持 r 指向
最后一个结点

时间复杂度: $O(n)$

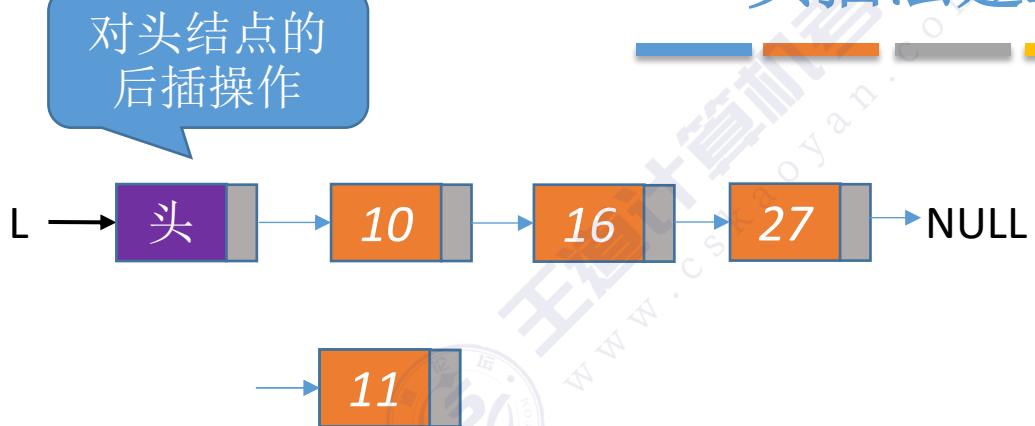


如果不带头结点呢?



动口不动手

头插法建立单链表



```
//后插操作: 在p结点之后插入元素 e
bool InsertNextNode (LNode *p, ElemType e){
    if (p==NULL)
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    if (s==NULL)    //内存分配失败
        return false;
    s->data = e;      //用结点s保存数据元素e
    s->next=p->next;
    p->next=s;        //将结点s连到p之后
    return true;
}
```

头插法建立单链表:

初始化单链表

While 循环 {
 每次取一个数据元素 e;
 InsertNextNode (L, e);
}

头插法建立单链表

```
LinkList List_HeadInsert(LinkList &L){ //逆向建立单链表
    LNode *s;
    int x;
    L=(LinkList)malloc(sizeof(LNode)); //创建头结点
    L->next=NULL; //初始为空链表
    scanf("%d",&x); //输入结点的值
    while(x!=9999){ //输入9999表示结束
        s=(LNode*)malloc(sizeof(LNode)); //创建新结点
        s->data=x;
        s->next=L->next;
        L->next=s; //将新结点插入表中, L为头指针
        scanf("%d",&x);
    }
    return L;
}
```



如果去掉这一句呢？

养成好习惯，只要是
初始化单链表，都先
把头指针指向 NULL

```
//后插操作: 在p结点之后插入元素 e
bool InsertNextNode (LNode *p, ElemtType e){
    if (p==NULL)
        return false;
    LNode *s = (LNode *)malloc(sizeof(LNode));
    if (s==NULL) //内存分配失败
        return false;
    s->data = e; //用结点s保存数据元素e
    s->next=p->next;
    p->next=s; //将结点s连到p之后
    return true;
}
```

如果不带头结点呢？

一定要动手鸭



知识回顾与重要考点

头插法、尾插法：核心就是初始化操作、指定结点的后插操作

注意设置一个指向表尾结点的指针



头插法的重要应用：链表的逆置

动手试一试：给你一个 LinkList L ,
如何逆置？

