石家莊鐵道大學
SHIJIAZHUANG TIEDAO UNIVERSITY

在线开放课程

线性表

单链表的基本操作
——初始化、查找

主讲 ： 刘辉

目录

一、单链表存储结构的定义

- **单链表存储结构的定义**

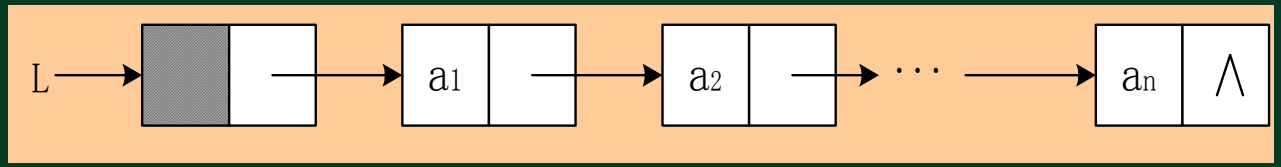
```
typedef struct LNode{
    ElemType   data;       //数据域
    struct  LNode  *next;   //指针域
}LNode, *LinkList;
// *LinkList为Lnode类型的指针
```

LNode *p ⟷ **LinkList p**

一、单链表存储结构的定义

- **单链表存储结构的定义**
  - **指针变量：**表示结点地址
  - **结点变量：**表示一个结点

LNode *p



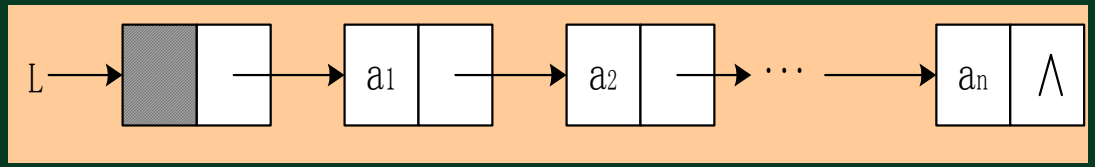
若$p->data=a_i$，则$p->next->data=a_{i+1}$

一、单链表存储结构的定义

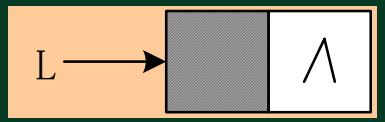
单链表存储结构的定义

- 单链表是由表头唯一确定，因此单链表可以用头指针的名字来命名
- 若头指针名是L，则把**链表**称为**表L**

非空表



空表



二、单链表的基本操作

- 初始化线性表L　InitList(L)
- 销毁线性表L　DestoryList(L)
- 清空线性表L　ClearList(L)
- 求线性表L的长度　ListLength(L)
- 判断线性表L是否为空　IsEmpty(L)

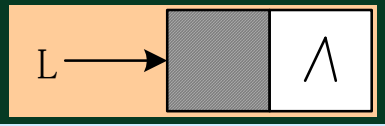
二、单链表的基本操作

- 获取线性表L中的某个数据元素内容
  GetElem(L,i,e)
- 检索值为e的数据元素　LocateELem(L,e)
- 在线性表L中插入一个数据元素
  ListInsert(L,i,e)
- 删除线性表L中第i个数据元素
  ListDelete(L,i,e)

三、单链表的初始化等操作

```
typedef struct LNode{
    ElemType  data;
    struct  LNode  *next;
}LNode, *LinkList
```

- **构造一个空表**



【算法思想】

（1）生成新结点作头结点，用头指针L指向头结点。

（2）头结点的指针域置空。

【算法描述】

```
Status InitList_L(LinkList &L){
  L=new LNode;
  L->next=NULL;
  return OK;
}
```

三、单链表的初始化等操作

```
typedef struct LNode{
    ElemType  data;
    struct  LNode  *next;
}LNode, *LinkList
```

- 销毁

【算法描述】

```
Status DestroyList_L(LinkList &L){
    LinkList  p;
    while(L)
    {  p=L;
       L=L->next;
       delete p;
    }
  return OK;
}
```

三、单链表的初始化等操作

```
typedef struct LNode{
    ElemType  data;
    struct  LNode  *next;
}LNode, *LinkList
```

- 清空

【算法描述】

```
// 将L重置为空表
Status ClearList(LinkList & L){
  LinkList p,q;
  p=L->next;  //p指向第一个结点
  while(p)    //没到表尾
    { q=p; p=p->next; delete q; }
  L->next=NULL;  //头结点指针域为空
  return OK;
}
```

三、单链表的初始化等操作

```
typedef struct LNode{
    ElemType  data;
    struct  LNode  *next;
}LNode, *LinkList
```

- **求表长**

【算法描述】

```
//返回L中数据元素个数
 int  ListLength_L(LinkList L){
LinkList p;
   p=L->next;        //p指向第一个结点
    i=0;
    while(p){         //遍历单链表，统计结点数
        i++;   p=p->next;   }
   return i;
 }
```

三、单链表的初始化等操作

```
typedef struct LNode{
    ElemType  data;
    struct  LNode  *next;
}LNode, *LinkList
```

- **判断表是否为空**

【算法描述】

```
//若L为空表，则返回1，否则返回0
int ListEmpty(LinkList L){
 if(L->next)  //非空
    return 0;
  else
    return 1;
 }
```

四、单链表的查找

- 顺序存储结构中如何查找第i个元素？
- 链表的查找：要从链表的头指针出发，顺着链域next逐个结点往下搜索，直至搜索到第i个结点为止。因此，链表不是随机存取结构
- **按序号查找：**查找第i个结点的值
- **按值查找：**查找 值为**的结点

四、单链表的查找

- **按序号查找**

【算法思想】

（1）从第1个结点（L->next）顺链扫描，用指针p指向当前扫描到的结点，p初值p = L->next。

（2）用 *j* 做计数器，累计当前扫描过的结点数，*j* 初值为1。

（3）当p指向扫描到的下一结点时，计数器 *j* 加1。

（4）当*j* = *i*时，p所指的结点就是要找的第*i*个结点。

四、单链表的查找

- **按序号查找**

【算法描述】

```
Status GetElem_L(LinkList L,int i,ElemType &e){
    p=L->next;j=1;   //初始化
    //向后扫描，直到p指向第i个元素或p为空
    while(p && j<i ){
       p=p->next;     ++j;      }
     if( !p  ||  j>i )   return ERROR;   //第i个元素不存在
     e=p->data;   //取第i个元素
     return OK;
}/  /GetElem_L
```

四、单链表的查找

- **按值查找—查找值为e的结点**

【算法思想】

（1）从第一个结点起，依次和e相比较。

（2）如果找到一个其值与e相等的数据元素，则返回其在链表中的“位置”；

（3）如果查遍整个链表都没有找到其值和e相等的元素，则返回“NULL”。

四、单链表的查找

- **按值查找—查找值为e的结点**

【算法描述】

```
LNode  *LocateELem_L (LinkList L，Elemtype e) {
  p=L->next;
  while(p &&p->data!=e)
       p=p->next;
  //返回L中值为e的数据元素的位置，查找失败返NULL
  return p;
}
```

小结

- 掌握单链表存储结构的定义方式
- 了解单链表的常用基本操作
- 掌握单链表初始化等操作的实现方式
- 掌握单链表查找算法的实现方式

谢谢！