本节主题：

线性表的应用

应用问题：表的自然连接

- 问题
  - 有表A，m1行、n1列
  - 有表B，m2行、n2列
  - 求自然连接结果

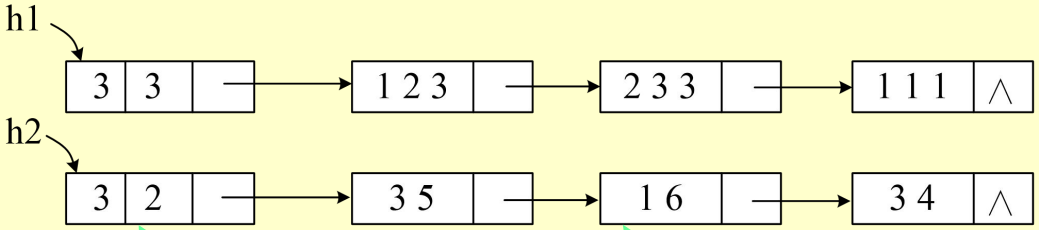
$$C = A \underset{i=j}{\triangleright\triangleleft} B$$

连接条件：表A的第i列与表B的第j列相等

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 5 \\ 1 & 6 \\ 3 & 4 \end{bmatrix}$$

$$C = A \underset{3=1}{\triangleright\triangleleft} B = \begin{bmatrix} 1 & 2 & 3 & 3 & 5 \\ 1 & 2 & 3 & 3 & 4 \\ 2 & 3 & 3 & 3 & 5 \\ 2 & 3 & 3 & 3 & 4 \\ 1 & 1 & 1 & 1 & 6 \end{bmatrix}$$

数据结构设计：顺序表和链表混合使用！！！



$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 3 & 5 \\ 1 & 6 \\ 3 & 4 \end{bmatrix}$$

```
typedef struct Node2
{
   int Row,Col;
   DList *next;
} HList;
//记录行、列数及首行地址
```

```
typedef int ElemType;
```

```
#define MaxCol  10
typedef struct Node1
{
   ElemType data[MaxCol];
   struct Node1 *next;
} DList;
//每行作为一个数据结点
```

用顺序表存储每一行的数据

$$C = A \underset{3=1}{\triangleright\!\triangleleft} B = \begin{bmatrix} 1 & 2 & 3 & 3 & 5 \\ 1 & 2 & 3 & 3 & 4 \\ 2 & 3 & 3 & 3 & 5 \\ 2 & 3 & 3 & 3 & 4 \\ 1 & 1 & 1 & 1 & 6 \end{bmatrix}$$

- 由于每个表的行数不确定，为此用单链表作为表的存储结构。
- 每行中的数据个数也是不确定的，采用顺序存储结构，便于随机查找行中的数据。
- 链表中的结点数，以及每行的数据数，在头结点中可以获得。

设计运算算法

- **CreateTable(HList *&h)**
  - 交互式创建多项式单链表。
  - 采用尾插法建表方法创建单链表,用户先输入表的行数和列数,然后输入各行的数据
- **DestroyTable(HList *&h)**
  - 销毁多项式单链表 。
- **DispTable (HList *h)**
  - 输出多项式单链表。
- **LinkTable(HList *h1,HList *h2,HList *&h)**
  - 实现两个多项式单链表的连接运算。

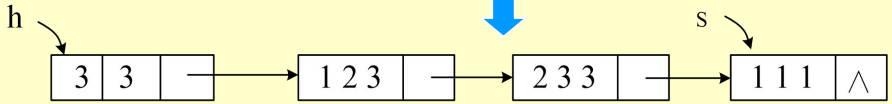
```
#define MaxCol  10
typedef struct Node2
{
    int Row,Col;
    DList *next;
} HList;
typedef int ElemType;
typedef struct Node1
{
    ElemType data[MaxCol];
    struct Node1 *next;
} DList;
```

CreateTable(HList *&h)：交互式创建多项式单链表

```
void CreateTable(HList *&h)
{
    int i,j;
    DList *r,*s;
    h=(HList *)malloc(sizeof(HList));
    printf("表的行数,列数:");
    scanf("%d%d",&h->Row,&h->Col);
    h->next = NULL;
    for (i=0; i<h->Row; i++)
    {
        printf("  第%d行:",i+1);
        s=(DList *)malloc(sizeof(DList));
        for (j=0; j<h->Col; j++)
            scanf("%d",&s->data[j]);
        if (h->next==NULL)
            h->next=s;
        else
            r->next=s;
        r=s;
    }
    r->next=NULL;
}
```

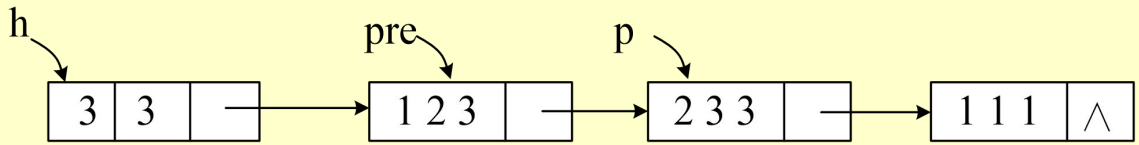
采用尾插法建表

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$



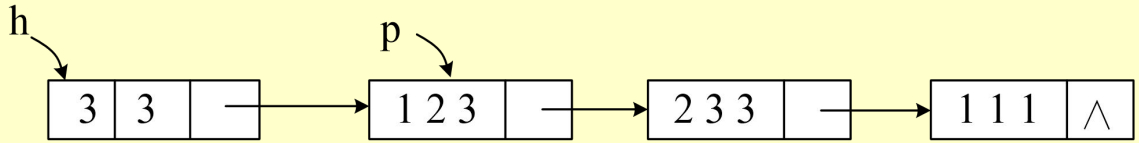
销毁单链表算法

```
void DestroyTable(HList *&h)
{
    DList *pre=h->next,*p=pre->next;
    while (p!=NULL)
    {
        free(pre);
        pre=p;
        p=p->next;
    }
    free(pre);
    free(h);
}
```



输出单链表算法

```
void DispTable(HList *h)
{
    int j;
    DList *p=h->next;
    while (p!=NULL)
    {
        for (j=0; j<h->Col; j++)
            printf("%4d",p->data[j]);
        printf("\n");
        p=p->next;
    }
}
```



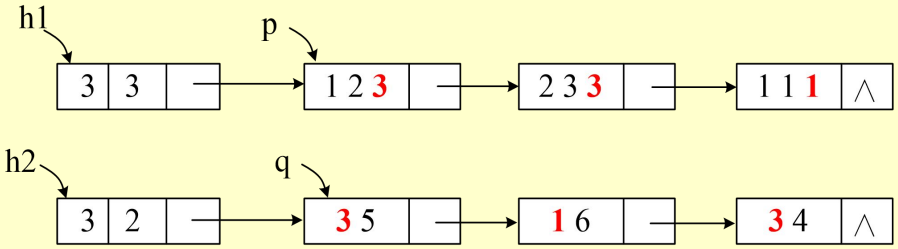
表连接运算算法

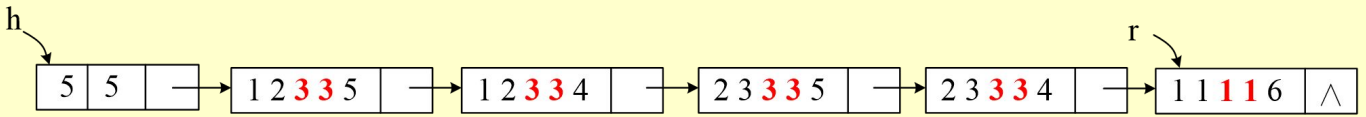
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 3 \\ 1 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 5 \\ 1 & 6 \\ 3 & 4 \end{bmatrix}$$

- 先输入两个表连接的列序号f1和f2
- 扫描单链表h1，对于h1的每个节点
  - 扫描单链表h2，连接条件成立，在结果单链表h中添加一个新节点
  - 连接条件的表示：

    `p->data[f1-1]==q->data[f2-1]`
- h也是采用尾插法加入新结点

$$C = A \underset{3=1}{\triangleright\!\triangleleft} B = \begin{bmatrix} 1 & 2 & 3 & 3 & 5 \\ 1 & 2 & 3 & 3 & 4 \\ 2 & 3 & 3 & 3 & 5 \\ 2 & 3 & 3 & 3 & 4 \\ 1 & 1 & 1 & 1 & 6 \end{bmatrix}$$

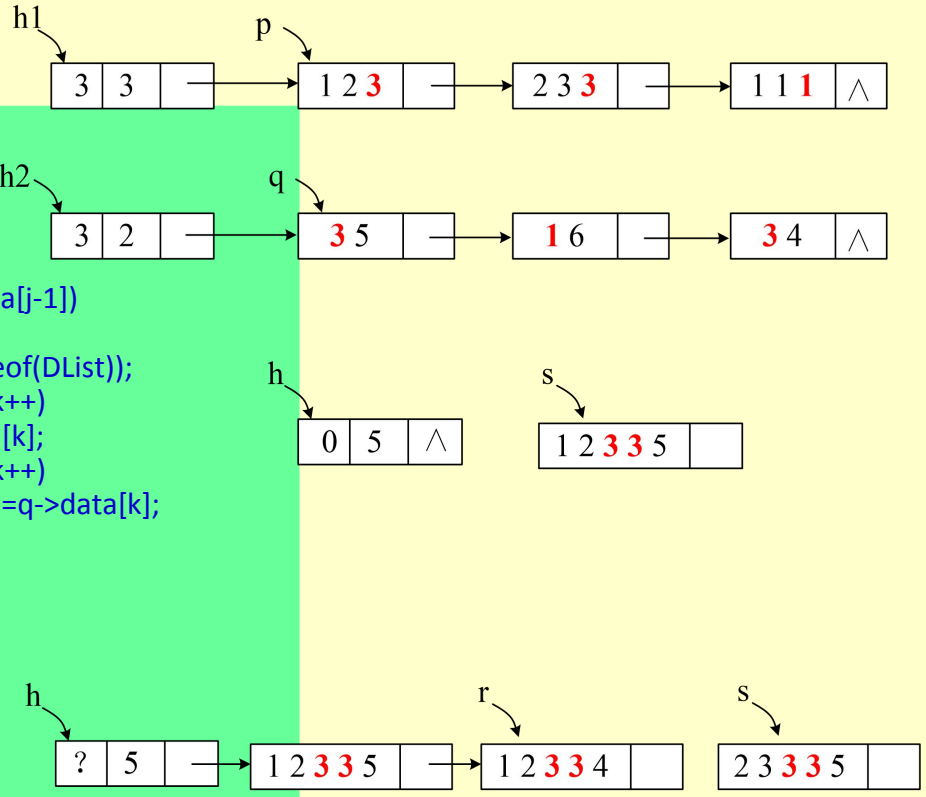




算法描述

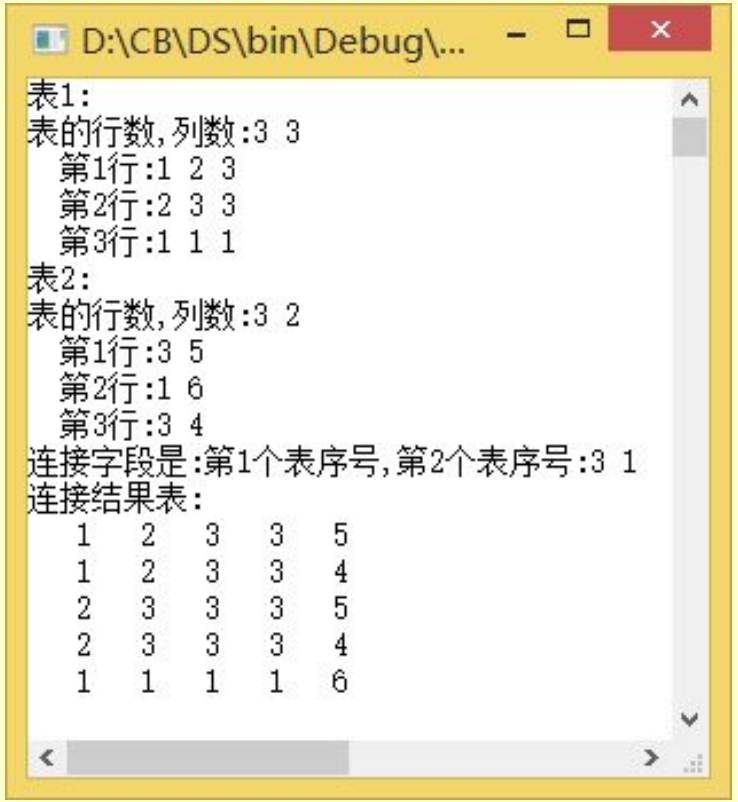
```
void LinkTable(HList *h1, HList *h2, HList *&h)
{
    int i,j,k;
    DList *p=h1->next,*q,*s,*r;
    printf("两个连接字段: ");
    scanf("%d%d",&i, &j);
    h=(HList *)malloc(sizeof(HList));
    h->Row=0;
    h->Col=h1->Col+h2->Col;
    h->next=NULL;
    //运算
    r->next=NULL;
}
```

```
while (p!=NULL)
{
    q=h2->next;
    while (q!=NULL)
    {
        if (p->data[i-1]==q->data[j-1])
        {
            s=(DList *)malloc(sizeof(DList));
            for (k=0; k<h1->Col; k++)
                s->data[k]=p->data[k];
            for (k=0; k<h2->Col; k++)
                s->data[h1->Col+k]=q->data[k];
            if (h->next==NULL)
                h->next=s;
            else
                r->next=s;
            r=s;
            h->Row++;
        }
        q=q->next;
    }
    p=p->next;
}
```



主函数

```
int main()
{
    HList *h1,*h2,*h;
    printf("表1:\n");
    CreateTable(h1);
    printf("表2:\n");
    CreateTable(h2);
    LinkTable(h1,h2,h);
    printf("连接结果表:\n");
    DispTable(h);
    DestroyTable(h1);
    DestroyTable(h2);
    DestroyTable(h);
}
```



思考题

- 体会数据结构中求解问题的一般过程。