西交软件 820 专业课资料

资料名称： 数据结构课件（朱利原版课件）

资料团队：软件堂堂团队

联系电话：15229238541

联系扣扣：871729782

建议利用时间：全程

软件团队版权所有，禁止传播
电话：15229238541

数据结构

朱利

西安交通大学软件学院

2013年3月

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

算法与数据结构

参考教材：《数据结构(C语言版)》。严蔚敏，吴伟民编著。清华大学出版社。

参考文献：

1 《数据结构》 。张选平，雷咏梅 编， 严蔚敏 审。机械工业出版社。

2 《数据结构与算法分析》。Clifford A. Shaffer著，张　铭，刘晓丹 译。电子工业出版社。

3 《数据结构习题与解析(C语实言版)》。李春葆。清华大学出版社。

4 《数据结构与算法》。夏克俭 编著。国防工业出版社。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

第1章　绪 论

目前，计算机已深入到社会生活的各个领域，其应用已不再仅仅局限于科学计算，而更多的是用于控制，管理及数据处理等非数值计算领域。计算机是一门研究用计算机进行信息表示和处理的科学。这里面涉及到两个问题：信息的表示，信息的处理。

信息的表示和组织又直接关系到处理信息的程序的效率。随着应用问题的不断复杂，导致信息量剧增与信息范围的拓宽，使许多系统程序和应用程序的规模很大，结构又相当复杂。因此，必须分析待处理问题中的对象的特征及各对象之间存在的关系，这就是数据结构这门课所要研究的问题。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

计算机求解问题的一般步骤

编写解决实际问题的程序的一般过程：

- 如何用数据形式描述问题?—即由问题抽象出一个适当的数学模型;
- 问题所涉及的数据量大小及数据之间的关系;
- 如何在计算机中存储数据及体现数据之间的关系?
- 处理问题时需要对数据作何种运算?
- 所编写的程序的性能是否良好?

上面所列举的问题基本上由数据结构这门课程来回答。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

1.1 数据结构及其概念

《算法与数据结构》是计算机科学中的一门综合性专业基础课。是介于数学、计算机硬件、计算机软件三者之间的一门核心课程，不仅是一般程序设计的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

1.1.1 数据结构的例子

例1：电话号码查询系统

设有一个电话号码薄，它记录了N个人的名字和其相应的电话号码，假定按如下形式安排：(a_1, b_1)，(a_2, b_2)，…(a_n, b_n)，其中$a_i, b_i(i=1，2…n)$ 分别表示某人的名字和电话号码。本问题是一种典型的表格问题。如表1-1，数据与数据成简单的一对一的线性关系。

姓名	电话号码
陈海	13612345588
李四锋	13056112345

表1-1 线性表结构

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播

电话：15229238541

例2：磁盘目录文件系统

磁盘根目录下有很多子目录及文件，每个子目录里又可以包含多个子目录及文件，但每个子目录只有一个父目录，依此类推：

本问题是一种典型的树型结构问题，如图**1-1**，数据与数据成一对多的关系，是一种典型的非线性关系结构—树形结构。

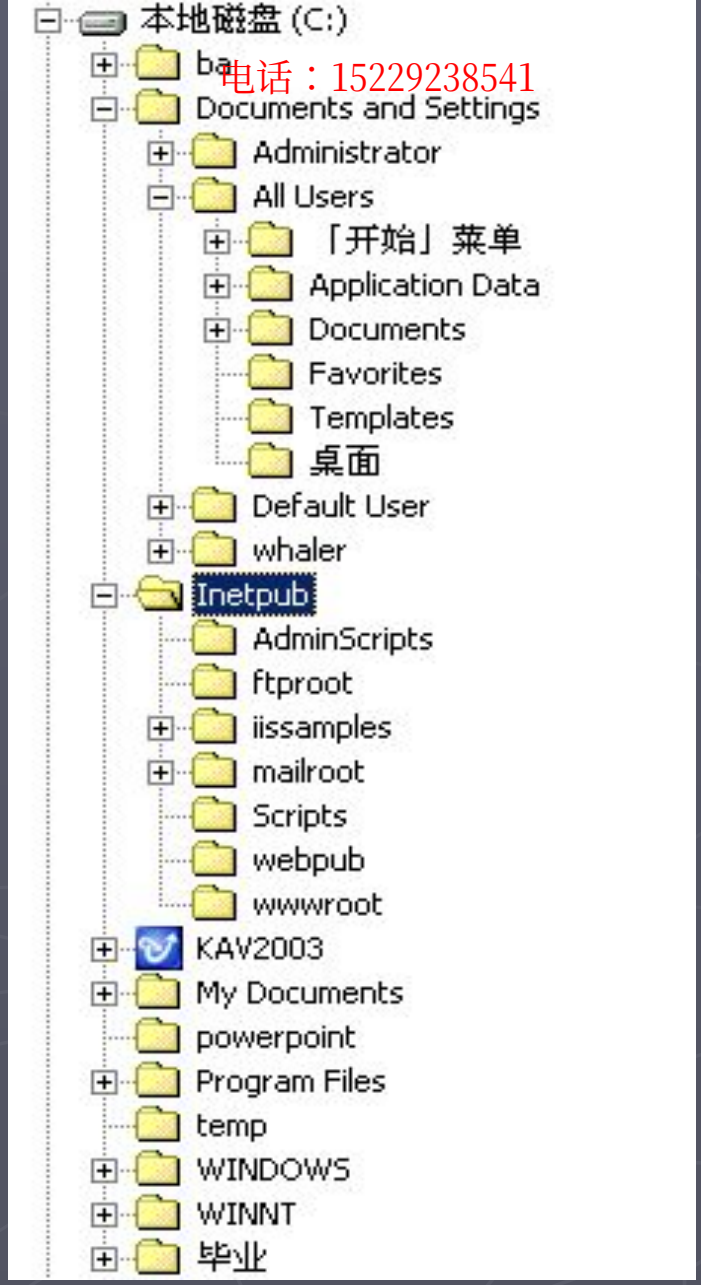


图1-1 树形结构

鸣谢 软件学院研究生会

00187229742

软件团队版权所有，禁止传播
电话：15229238541

例3：交通网络图

从一个地方到另外一个地方可以有多条路径。本问题是一种典型的网状结构问题，数据与数据成多对多的关系，是一种非线性关系结构。

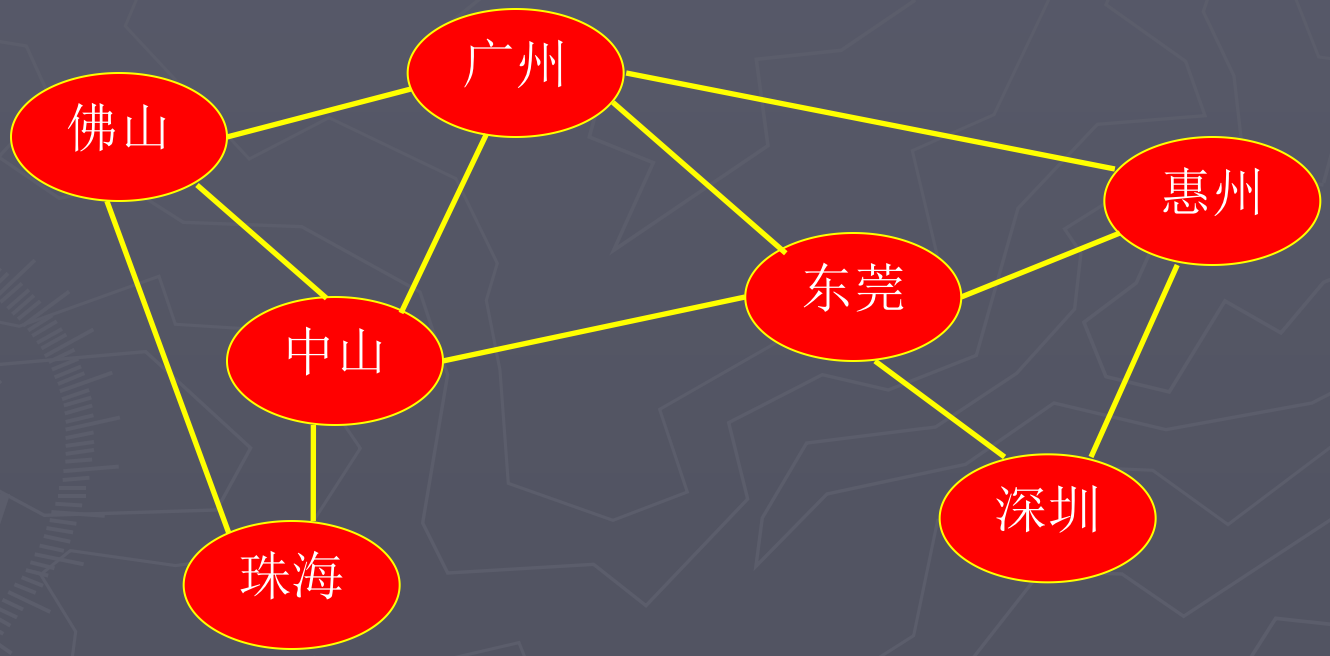


图1-2　网状结构

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

1.1.2 基本概念和术语

数据(Data)：是客观事物的符号表示。在计算机科学中指的是所有能输入到计算机中并被计算机程序处理的符号的总称。

数据元素(Data Element)：是数据的基本单位，在程序中通常作为一个整体来进行考虑和处理。

一个数据元素可由若干个数据项(Data Item)组成。数据项是数据的不可分割的最小单位。数据项是对客观事物某一方面特性的数据描述。

数据对象(Data Object)：是性质相同的数据元素的集合，是数据的一个子集。如字符集合C={'A','B','C',...}。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

数据结构(Data Structure)：是指相互之间具有(存在)一定联系(关系)的数据元素的集合。元素之间的相互联系(关系)称为逻辑结构。数据元素之间的逻辑结构有四种基本类型，如图1-3所示。

① 集合：结构中的数据元素除了“同属于一个集合”外，没有其它关系。

② 线性结构：结构中的数据元素之间存在一对一的关系。

③ 树型结构：结构中的数据元素之间存在一对多的关系。

④ 图状结构或网状结构：结构中的数据元素之间存在多对多的关系。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

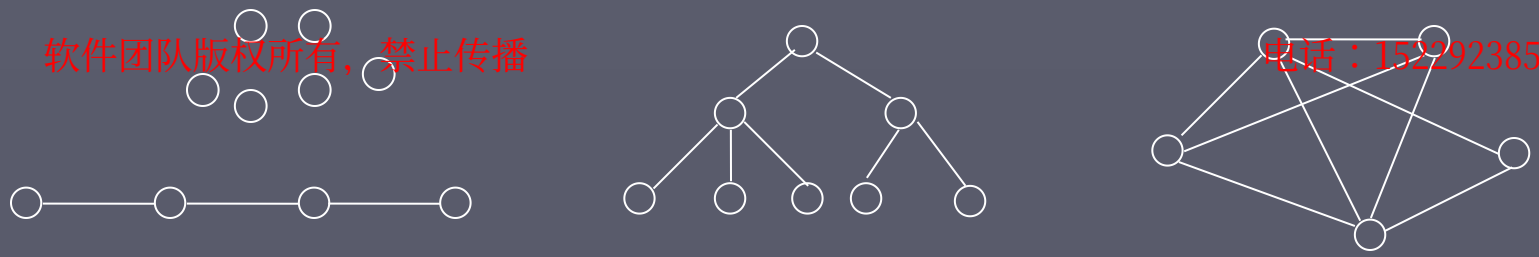


图1-3　　四类基本结构图

1.1.3 数据结构的形式定义

数据结构的形式定义是一个二元组：

Data-Structure=(D，S)

其中：D是数据元素的有限集，S是D上关系的有限集。

例2：设数据逻辑结构B=（K，R）

K={k_1, k_2, …, k_9}

R={ <k_1, k_3>，<k_1, k_8>，<k_2, k_3>，<k_2, k_4>，<k_2, k_5>，<k_3, k_9>，<k_5, k_6>，<k_8, k_9>，<k_9, k_7>，<k_4, k_7>，<k_4, k_6>}

鸣谢：软件学院研究生会
QQ：871729782

软件版权所有，禁止传播 电话：13429268541

数据元素之间的关系可以是元素之间代表某种含义的自然关系，也可以是为处理问题方便而人为定义的关系，这种自然或人为定义的“关系”称为数据元素之间的逻辑关系，相应的结构称为逻辑结构。

1.1.4 数据结构的存储方式

数据结构在计算机内存中的存储包括数据元素的存储和元素之间的关系的表示。

元素之间的关系在计算机中有两种不同的表示方法：顺序表示和非顺序表示。由此得出两种不同的存储结构：顺序存储结构和链式存储结构。

- 顺序存储结构：用数据元素在存储器中的相对位置来表示数据元素之间的逻辑结构(关系)。

鸿博软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

- 链式存储结构：在每一个数据元素中增加一个存放另一个元素地址的指针(pointer)，用该指针来表示数据元素之间的逻辑结构(关系)。

例：设有数据集合A={3.0,2.3,5.0,-8.5,11.0}，两种不同的存储结构。

- 顺序结构：数据元素存放的地址是连续的；
- 链式结构：数据元素存放的地址是否连续没有要求。

数据的逻辑结构和物理结构是密不可分的两个方面，一个算法的设计取决于所选定的逻辑结构，而算法的实现依赖于所采用的存储结构。

在C语言中，用一维数组表示顺序存储结构；用结构体类型表示链式存储结构。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

数据结构的三个组成部分：

逻辑结构： 数据元素之间逻辑关系的描述

D_S=（D，S）

存储结构： 数据元素在计算机中的存储及其逻辑关系的表现称为数据的存储结构或物理结构。

数据操作： 对数据要进行的运算。

本课程中将要讨论的三种逻辑结构及其采用的存储结构如图1-4所示。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

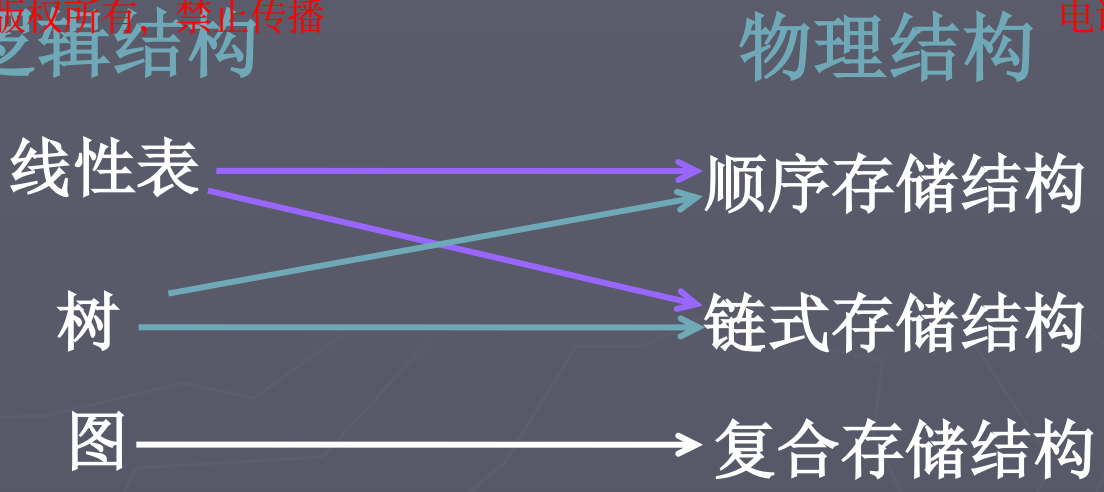


图1-4　逻辑结构与所采用的存储结构

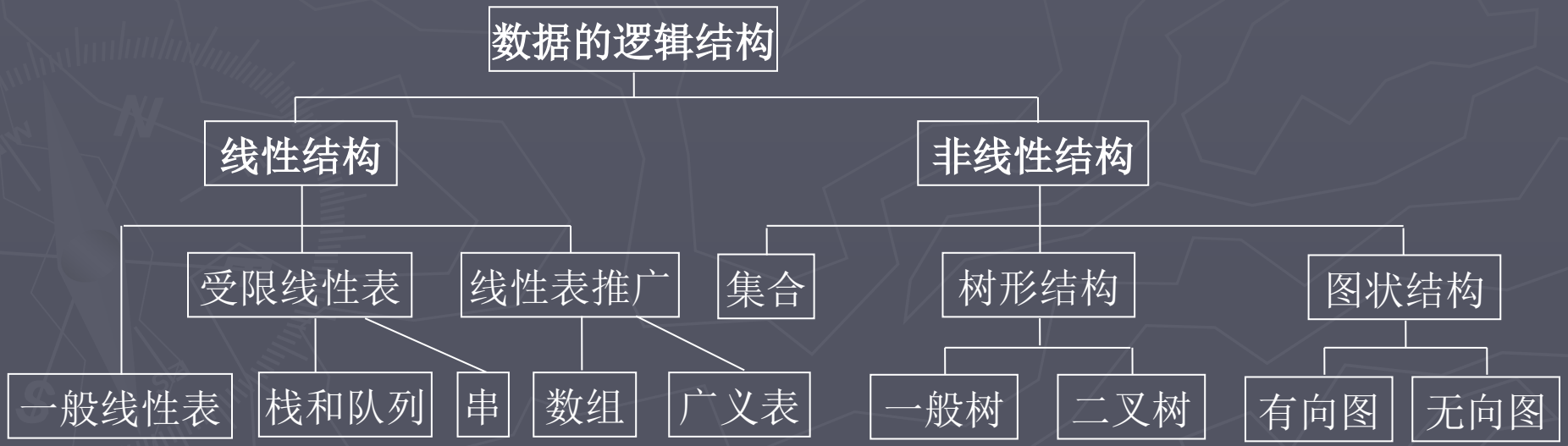


图1-5　数据逻辑结构层次关系图

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

1.1.5 数据类型

数据类型(Data Type)：指的是一个值的集合和定义在该值集上的一组操作的总称。

数据类型是和数据结构密切相关的一个概念。 在C语言中数据类型有：基本类型和构造类型。

数据结构不同于数据类型，也不同于数据对象，它不仅要描述数据类型的数据对象，而且要描述数据对象各元素之间的相互关系。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

1.1.6 数据结构的运算

数据结构的主要运算包括：

⑴ 建立(Create)一个数据结构；

⑵ 消除(Destroy)一个数据结构；

⑶ 从一个数据结构中删除(Delete)一个数据元素；

⑷ 把一个数据元素插入(Insert)到一个数据结构中；

⑸ 对一个数据结构进行访问(Access)；

⑹ 对一个数据结构(中的数据元素)进行修改(Modify)；

⑺ 对一个数据结构进行排序(Sort)；

鸣谢 软件学院研究生会 QQ：871729782

⑻ 对一个数据结构进行查找(Search)

软件团队版权所有，禁止传播
电话：15229238541

1.2 抽象数据类型

抽象数据类型(Abstract Data Type，简称ADT)：是指一个数学模型以及定义在该模型上的一组操作。

ADT的定义仅是一组逻辑特性描述，与其在计算机内的表示和实现无关。因此，不论ADT的内部结构如何变化，只要其数学特性不变，都不影响其外部使用。

ADT的形式化定义是三元组：ADT=(D，S，P)

其中：D是数据对象，S是D上的关系集，P是对D的基本操作集。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

ADT的一般定义形式是：

ADT <抽象数据类型名>{

数据对象： <数据对象的定义>

数据关系： <数据关系的定义>

基本操作： <基本操作的定义>

} ADT <抽象数据类型名>

- 其中数据对象和数据关系的定义用伪码描述。
- 基本操作的定义是：

<基本操作名>(<参数表>)

初始条件： <初始条件描述>

操作结果： <操作结果描述>

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

– 初始条件：描述操作执行之前数据结构和参数应满足的条件;若不满足，则操作失败，返回相应的出错信息。

– 操作结果：描述操作正常完成之后，数据结构的变化状况和 应返回的结果。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

1.3 算法分析初步

1.3.1 算法

算法(Algorithm)：是对特定问题求解方法(步骤)的一种描述，是指令的有限序列，其中每一条指令表示一个或多个操作。

算法具有以下五个特性

① 有穷性： 一个算法必须总是在执行有穷步之后结束，且每一步都在有穷时间内完成。

② 确定性：算法中每一条指令必须有确切的含义。不存在二义性。且算法只有一个入口和一个出口。

③ 可行性： 一个算法是能行的。即算法描述的操作都可以通过已经实现的基本运算执行有限次来实现。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

④ **输入**：一个算法有零个或多个输入，这些输入取自于某个特定的对象集合。

⑤ **输出**：一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。

一个算法可以用多种方法描述，主要有：使用自然语言描述；使用形式语言描述；使用计算机程序设计语言描述。

算法和程序是两个不同的概念。一个计算机程序是对一个算法使用某种程序设计语言的具体实现。算法必须可终止意味着不是所有的计算机程序都是算法。

在本门课程的学习、作业练习、上机实践等环节，算法都用C语言来描述。在上机实践时，为了检查算法是否正确，应编写成完整的C语言程序。

鸣谢 软件学院团委编 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

1.3.2 算法设计的要求

评价一个好的算法有以下几个标准

① 正确性(Correctness)： 算法应满足具体问题的需求。

② 可读性(Readability)： 算法应容易供人阅读和交流。可读性好的算法有助于对算法的理解和修改。

③ 健壮性(Robustness)： 算法应具有容错处理。当输入非法或错误数据时，算法应能适当地作出反应或进行处理，而不会产生莫名其妙的输出结果。

④ 通用性(Generality)： 算法应具有一般性，即算法的处理结果对于一般的数据集合都成立。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15225238541

⑤ 效率与存储量需求：效率指的是算法执行的时间；存储量需求指算法执行过程中所需要的最大存储空间。一般地，这两者与问题的规模有关。

1.3.3 算法效率的度量

算法执行时间需通过依据该算法编制的程序在计算机上运行所消耗的时间来度量。其方法通常有两种：

事后统计：计算机内部进行执行时间和实际占用空间的统计。

问题：必须先运行依据算法编制的程序；依赖软硬件环境，容易掩盖算法本身的优劣；没有实际价值。

事前分析：求出该算法的一个时间界限函数。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

与此相关的因素有：

- 依据算法选用何种策略；
- 问题的规模；
- 程序设计的语言；
- 编译程序所产生的机器代码的质量；
- 机器执行指令的速度；

撇开软硬件等有关部门因素，可以认为一个特定算法“运行工作量”的大小，只依赖于问题的规模（通常用n表示），或者说，它是问题规模的函数。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法分析应用举例

算法中基本操作重复执行的次数是问题规模n的某个函数，其时间量度记作 T(n)=O(f(n))，称作算法的渐近时间复杂度(Asymptotic Time complexity)，简称时间复杂度。

一般地，常用最深层循环内的语句中的原操作的执行频度(重复执行的次数)来表示。

“O”的定义： 若f(n)是正整数n的一个函数，则 O(f(n))表示∃ M≥0 ，使得当$n \geq n_0$时，$|f(n)| \leq M |f(n_0)|$。

表示时间复杂度的阶有：

O(1)：常量时间阶　　O (n)：线性时间阶

鸣谢 软件学院研究生会 QQ：877729782

软件团队版权所有，禁止传播 电话：15229238541

$O(n^k)$： k≥2，k次方时间阶

例 1　两个n阶方阵的乘法

```
for(i=1，i<=n; ++i)
    for(j=1; j<=n; ++j)
        {   c[i][j]=0 ;
            for(k=1; k<=n; ++k)
                c[i][j]+=a[i][k]*b[k][j] ; }
```

由于是一个三重循环，每个循环从1到n，则总次数为：$n\times n\times n=n^3$　时间复杂度为$T(n)=O(n^3)$

例 2　{++x; s=0 ;}

将x自增看成是基本操作，则语句频度为 1，即时间复杂度为$O(1)$。

嗨搜 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

如果将s=0也看成是基本操作，则语句频度为2，其时间复杂度仍为O(1)，即常量阶。

例3
```
for(i=1; i<=n; ++i)
        { ++x; s+=x ; }
```

语句频度为：2n，其时间复杂度为：O(n)，即为线性阶。

例4
```
for(i=1; i<=n; ++i)
    for(j=1; j<=n; ++j)
        { ++x; s+=x ; }
```

语句频度为：$2n^2$，其时间复杂度为：$O(n^2)$，即为平方阶。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

定理： 若$A(n)=a_m n^m+a_{m-1} n^{m-1}+\ldots+a_1 n+a_0$是一个m次多项式，则$A(n)=O(n^m)$

例 5

```
for(i=2;i<=n;++i)
    for(j=2;j<=i-1;++j)
        {++x; a[i,j]=x; }
```

语句频度为：$1+2+3+\ldots+n-2=(1+n-2)\times(n-2)/2$

$=(n-1)(n-2)/2$

$=n^2-3n+2$

∴时间复杂度为$O(n^2)$，即此算法的时间复杂度为平方阶。

- 一个算法时间为O(1)的算法，它的基本运算执行的次数是固定的。因此，总的时间由一个常数（即零次多项式）来限界。而一个时间为$O(n^2)$的算法则

鸣谢：软件学院研究生会 QQ：871729783

软件团队版权所有，禁止传播 电话：15229238541

以下六种计算算法时间的多项式是最常用的。其关系为：

$O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)$

- 指数时间的关系为：

$O(2^n)<O(n!)<O(n^n)$

当n取得很大时，指数时间算法和多项式时间算法在所需时间上非常悬殊。因此，只要有人能将现有指数时间算法中的任何一个算法化简为多项式时间算法，那就取得了一个伟大的成就。

- 有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

例1：素数的判断算法。

```
Void prime( int n)
   /*  n是一个正整数  */
   {   int i=2 ;
       while ( (n% i)!=0 && i*1.0< sqrt(n) )
       i++ ;
       if (i*1.0>sqrt(n) )
          printf(“&d 是一个素数\n”, n);
       else
          printf(“&d 不是一个素数\n”, n);
   }
```

嵌套的最深层语句是i++；其频度由条件((n% i)!=0 && i*1.0< sqrt(n)) 决定，显然i*1.0< sqrt(n) ，时间复杂度O(n1/2)

鹏城 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

例2：冒泡排序法。

```
Void bubble_sort(int a[]，int n)
  {   change=false;
     for (i=n-1; change=TURE; i>1 && change;
     --i)
        for (j=0; j<i; ++j)
           if (a[j]>a[j+1])
              {    a[j] ←→a[j+1] ;
           change=TURE ; }
  }
```

- 最好情况：0次
- 最坏情况：1+2+3+…+n-1=n(n-1)/2
- 平均时间复杂度为： $O(n^2)$

四海 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

1.3.4 算法的空间分析

空间复杂度(Space complexity)：是指算法编写成程序后，在计算机中运行时所需存储空间大小的度量。记作： S(n)=O(f(n))

其中： n为问题的规模(或大小)

该存储空间一般包括三个方面：

- 指令常数变量所占用的存储空间;
- 输入数据所占用的存储空间;
- 辅助(存储)空间。

一般地，算法的空间复杂度指的是辅助空间。

- 一维数组a[n]： 空间复杂度 O(n)
- 二维数组a[n][m]： 空间复杂度 O(n*m)

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

习题一

1 简要回答术语：数据，数据元素，数据结构，数据类型。

2 数据的逻辑结构？数据的物理结构？逻辑结构与物理结构的区别和联系是什么？

3 数据结构的主要运算包括哪些？

4 算法分析的目的是什么？算法分析的主要方面是什么？

5 分析以下程序段的时间复杂度，请说明分析的理由或原因。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播

(1)

```
Sum1( int n )
    {   int p=1, sum=0, m ;
        for (m=1; m<=n; m++)
            {  p*=m ; sum+=p ;  }
        return (sum) ;
    }
```

(2)

```
Sum2( int n )
    {   int sum=0, m, t ;
        for (m=1; m<=n; m++)
            {   p=1 ;
                for (t=1; t<=m; t++)  p*=t ;
                sum+=p ;
            }
        return (sum) ;
    }
```

(3) 递归函数

```
fact( int n )
    {   if (n<=1)  return(1) ;
        else return( n*fact(n-1)) ;
    }
```

电话：15229238541

鸣谢 软件学院研究生会

QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

第2章　线性表

线性结构是最常用、最简单的一种数据结构。而线性表是一种典型的线性结构。其基本特点是线性表中的数据元素是有序且是有限的。在这种结构中：

① 存在一个唯一的被称为“第一个”的数据元素；

② 存在一个唯一的被称为“最后一个”的数据元素；

③ 除第一个元素外，每个元素均有唯一一个直接前驱；

④ 除最后一个元素外，每个元素均有唯一一个直接后继。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

2.1 线性表的逻辑结构

2.1.1 线性表的定义

线性表(Linear List)：是由n(n≧0)个数据元素(结点)a_1，a_2，…a_n组成的有限序列。该序列中的所有结点具有相同的数据类型。其中数据元素的个数n称为线性表的长度。

当n=0时，称为空表。

当n>0时，将非空的线性表记作：(a_1，a_2，…a_n)

a_1称为线性表的第一个(首)结点，a_n称为线性表的最后一个(尾)结点。

呜谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229228541

a_1，a_2，…a_{i-1}都是a_i($2 \leqslant i \leqslant n$)的前驱，其中$a_{i-1}$是$a_i$的直接前驱；

a_{i+1}，a_{i+2}，…a_n都是a_i($1 \leqslant i \leqslant n-1$)的后继，其中$a_{i+1}$是$a_i$的直接后继。

2.1.2 线性表的逻辑结构

线性表中的数据元素a_i所代表的具体含义随具体应用的不同而不同，在线性表的定义中，只不过是一个抽象的表示符号。

- 线性表中的结点可以是单值元素(每个元素只有一个数据项)。

例1: 26个英文字母组成的字母表：（A，B，C，…，Z）

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15249238541

例2： 某校从1978年到1983年各种型号的计算机拥有量的变化情况：(6，17，28，50，92，188）

例3： 一副扑克的点数 (2，3，4，…，J，Q，K，A)

- 线性表中的结点可以是记录型元素，每个元素含有多个数据项，每个项称为结点的一个域。每个元素有一个可以唯一标识每个结点的数据项组，称为关键字。

例4： 某校2001级同学的基本情况：
{('2001414101'，'张里户'，'男'，06/24/1983)，('2001414102'，'张化司'，'男'，08/12/1984) …，('2001414102'，'李利辣'，'女'，08/12/1984) }

- 若线性表中的结点是按值(或按关键字值)由小到大(或由大到小)排列的，称线性表是有序的。

鸣谢 软件学院班会 0187126782

软件团队版权所有，禁止传播 电话：15229238541

- 线性表是一种相当灵活的数据结构，其长度可根据需要增长或缩短。
- 对线性表的数据元素可以访问、插入和删除。

2.1.3 线性表的抽象数据类型定义

ADT List{

数据对象：$D = \{ a_i \mid a_i \in ElemSet,\ i=1,2,\ldots,n,\ n \geqq 0 \}$

数据关系：$R = \{ \langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D,\ i=2,3,\ldots,n \}$

基本操作：

InitList(&L)

操作结果：构造一个空的线性表L；

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

ListLength(L)

初始条件：线性表L已存在；

操作结果：若L为空表，则返回TRUE，否则返回FALSE；

….

GetElem(L, i, &e)

初始条件：线性表L已存在，1≦i≦ListLength(L)；

操作结果：用e返回L中第i个数据元素的值；

ListInsert (L, i, &e)

初始条件：线性表L已存在，1≦i≦ListLength(L) ；

操作结果：在线性表L中的第i个位置插入元素e；

…

} ADT List

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

2.2 线性表的顺序存储

2.2.1 线性表的顺序存储结构

顺序存储：把线性表的结点按逻辑顺序依次存放在一组地址连续的存储单元里。用这种方法存储的线性表简称顺序表。

顺序存储的线性表的特点：

- 线性表的逻辑顺序与物理顺序一致;
- 数据元素之间的关系是以元素在计算机内“物理位置相邻”来体现。

设有非空的线性表：$(a_1, a_2, \ldots a_n)$。顺序存储如图2-1所示。

鸿翔软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

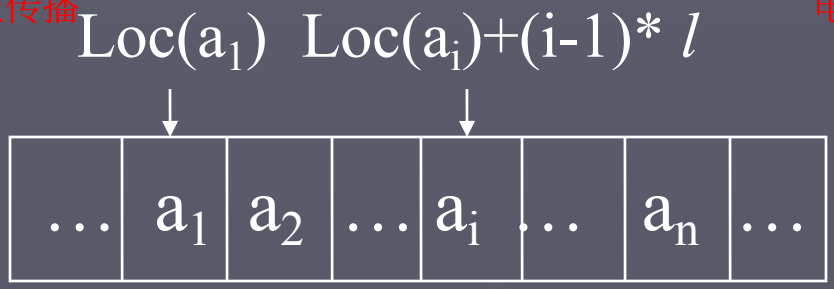


图2-1　线性表的顺序存储表示

在具体的机器环境下：设线性表的每个元素需占用*l*个存储单元，以所占的第一个单元的存储地址作为数据元素的存储位置。则线性表中第i+1个数据元素的存储位置$LOC(a_{i+1})$和第i个数据元素的存储位置$LOC(a_i)$之间满足下列关系：

$$LOC(a_{i+1})=LOC(a_i)+l$$

线性表的第i个数据元素a_i的存储位置为：

$$LOC(a_i)=LOC(a_1)+(i-1)*l$$

呜谢 软件学院研究生会
00：871729782

软件团队版权所有，禁止传播 电话：15229338541

在高级语言(如C语言)环境下：数组具有随机存取的特性，因此，借助数组来描述顺序表。除了用数组来存储线性表的元素之外，顺序表还应该有表示线性表的长度属性，所以用结构类型来定义顺序表类型。

```
#define  OK   1
#define  ERROR   -1
#define  MAX_SIZE  100
typedef  int  Status ;
typedef  int  ElemType ;
typedef  struct  sqlist
   {   ElemType  Elem_array[MAX_SIZE] ;
       int    length ;
   } SqList ;
```

鸣谢：软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

2.2.2 顺序表的基本操作

顺序存储结构中，很容易实现线性表的一些操作：初始化、赋值、查找、修改、插入、删除、求长度等。以下将对几种主要的操作进行讨论。

1 顺序线性表初始化

```
Status Init_SqList( SqList *L )
  {  L->elem_array=( ElemType
  * )malloc(MAX_SIZE*sizeof( ElemType ) ) ;
     if ( !L -> elem_array ) return  ERROR ;
     else {   L->length= 0 ;   return OK ; }
  }
```

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

2 顺序线性表的插入

在线性表 $L=(a_1, \ldots a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$ 中的第i(1≦i≦n)个位置上插入一个新结点e，使其成为线性表：

$L=(a_1, \ldots a_{i-1}, e, a_i, a_{i+1}, \ldots, a_n)$

实现步骤

（1）将线性表L中的第i个至第n个结点后移一个位置。

（2）将结点e插入到结点a_{i-1}之后。

（3）线性表长度加1。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
Status Insert_SqList(Sqlist *L，int i ， ElemType e)
  {  int j ;
    if ( i<0||i>L->length-1)  return  ERROR ;
    if (L->length>=MAX_SIZE)
      {   printf("线性表溢出!\n"); return ERROR ; }
    for ( j=L->length–1; j>=i-1; --j )
      L->Elem_array[j+1]=L->Elem_array[j];
        /* i-1位置以后的所有结点后移 */
    L->Elem_array[i-1]=e;   /* 在i-1位置插入结点 */
    L->length++ ;
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

时间复杂度分析

在线性表L中的第i个元素之前插入新结点，其时间主要耗费在表中结点的移动操作上，因此，可用结点的移动来估计算法的时间复杂度。

设在线性表L中的第i个元素之前插入结点的概率为P_i，不失一般性，设各个位置插入是等概率，则$P_i=1/(n+1)$，而插入时移动结点的次数为n-i+1。

总的平均移动次数：$E_{insert}=\sum p_i*(n-i+1)$ $(1\leqslant i\leqslant n)$

$\therefore E_{insert}=n/2$ 。

即在顺序表上做插入运算，平均要移动表上一半结点。当表长n较大时，算法的效率相当低。因此算法的平均时间复杂度为O(n)。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

3 顺序线性表的删除

在线性表 $L=(a_1, \ldots a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$ 中删除结点$a_i(1 \leqq i \leqq n)$，使其成为线性表：

$$L= (a_1, \ldots a_{i-1}, a_{i+1}, \ldots, a_n)$$

实现步骤

（1）将线性表L中的第i+1个至第n个结点依此向前移动一个位置。

（2）线性表长度减1。

算法描述

```
ElemType  Delete_SqList(Sqlist *L，int i)
  { int  k；  ElemType  x；
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

```
if  (L->length==0)
   {  printf("线性表L为空!\n"); return
   ERROR;  }
else if ( i<1||i>L->length )
   {  printf("要删除的数据元素不存在!\n") ;
      return ERROR ; }
   else  {  x=L->Elem_array[i-1] ;   /*保存结
   点的值*/
      for ( k=i ;  k<L->length ; k++)
           L->Elem_array[k-1]=L-
      >Elem_array[k];
                  /*  i位置以后的所有结点前移  */
      L->length--;  return (x);
      }
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

时间复杂度分析

删除线性表L中的第i个元素，其时间主要耗费在表中结点的移动操作上，因此，可用结点的移动来估计算法的时间复杂度。

设在线性表L中删除第i个元素的概率为P_i，不失一般性，设删除各个位置是等概率，则$P_i=1/n$，而删除时移动结点的次数为n-i。

则总的平均移动次数： $E_{delete}=\sum p_i*(n-i)$ $(1 \leqq i \leqq n)$

$\therefore E_{delete}=(n-1)/2$ 。

即在顺序表上做删除运算，平均要移动表上一半结点。当表长n较大时，算法的效率相当低。因此算法的平均时间复杂度为O(n)。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

4 顺序线性表的查找定位删除

在线性表 $L=(a_1, a_2, \ldots, a_n)$ 中删除值为x的第一个结点。

实现步骤

(1) 在线性表L查找值为x的第一个数据元素。

(2) 将从找到的位置至最后一个结点依次向前移动一个位置。

(3) 线性表长度减1。

算法描述

```
Status  Locate_Delete_SqList(Sqlist *L, ElemType x)
  /* 删除线性表L中值为x的第一个结点  */
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

```
while (i<L->length)     /*查找值为x的第一个结点*/
  {  if (L->Elem_array[i]!=x ) i++ ;
     else
       { for ( k=i+1; k< L->length; k++)
             L->Elem_array[k-1]=L->Elem_array[k];
         L->length--;  break ;
       }
  }
if (i>L->length)
  {   printf("要删除的数据元素不存在!\n");
     return ERROR ;  }
return OK;
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

时间复杂度分析

时间主要耗费在数据元素的比较和移动操作上。

首先，在线性表L中查找值为x的结点是否存在;

其次，若值为x的结点存在，且在线性表L中的位置为i，则在线性表L中删除第i个元素。

设在线性表L删除数据元素概率为P_i，不失一般性，设各个位置是等概率，则$P_i=1/n$。

- 比较的平均次数： $E_{compare}=\sum p_i*i \quad (1\leqq i\leqq n)$

$\therefore E_{compare}=(n+1)/2$ 。

- 删除时平均移动次数：$E_{delete}=\sum p_i*(n-i) \quad (1\leqq i\leqq n)$

$\therefore E_{delete}=(n-1)/2$ 。 平均时间复杂度：$E_{compare}+E_{delete}=n$，即为$O(n)$

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

2.3 线性表的链式存储

2.3.1 线性表的链式存储结构

链式存储：用一组任意的存储单元存储线性表中的数据元素。用这种方法存储的线性表简称线性链表。

存储链表中结点的一组任意的存储单元可以是连续的，也可以是不连续的，甚至是零散分布在内存中的任意位置上的。

链表中结点的逻辑顺序和物理顺序不一定相同。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

为了正确表示结点间的逻辑关系，在存储每个结点值的同时，还必须存储指示其直接后继结点的地址(或位置)，称为指针(pointer)或链(link)，这两部分组成了链表中的结点结构，如图2-2所示。

链表是通过每个结点的指针域将线性表的n个结点按其逻辑次序链接在一起的。

每一个结只包含一个指针域的链表，称为单链表。

为操作方便，总是在链表的第一个结点之前附设一个头结点(头指针)head指向第一个结点。头结点的数据域可以不存储任何信息(或链表长度等信息)。

data：数据域，存放结点的值。next：指针域，存放结点的直接后继的地址。

图2-2 链表结点结构

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

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。

例1、线性表L=(bat，cat，eat，fat，hat)

其带头结点的单链表的逻辑状态和物理存储方式如图2-3所示。

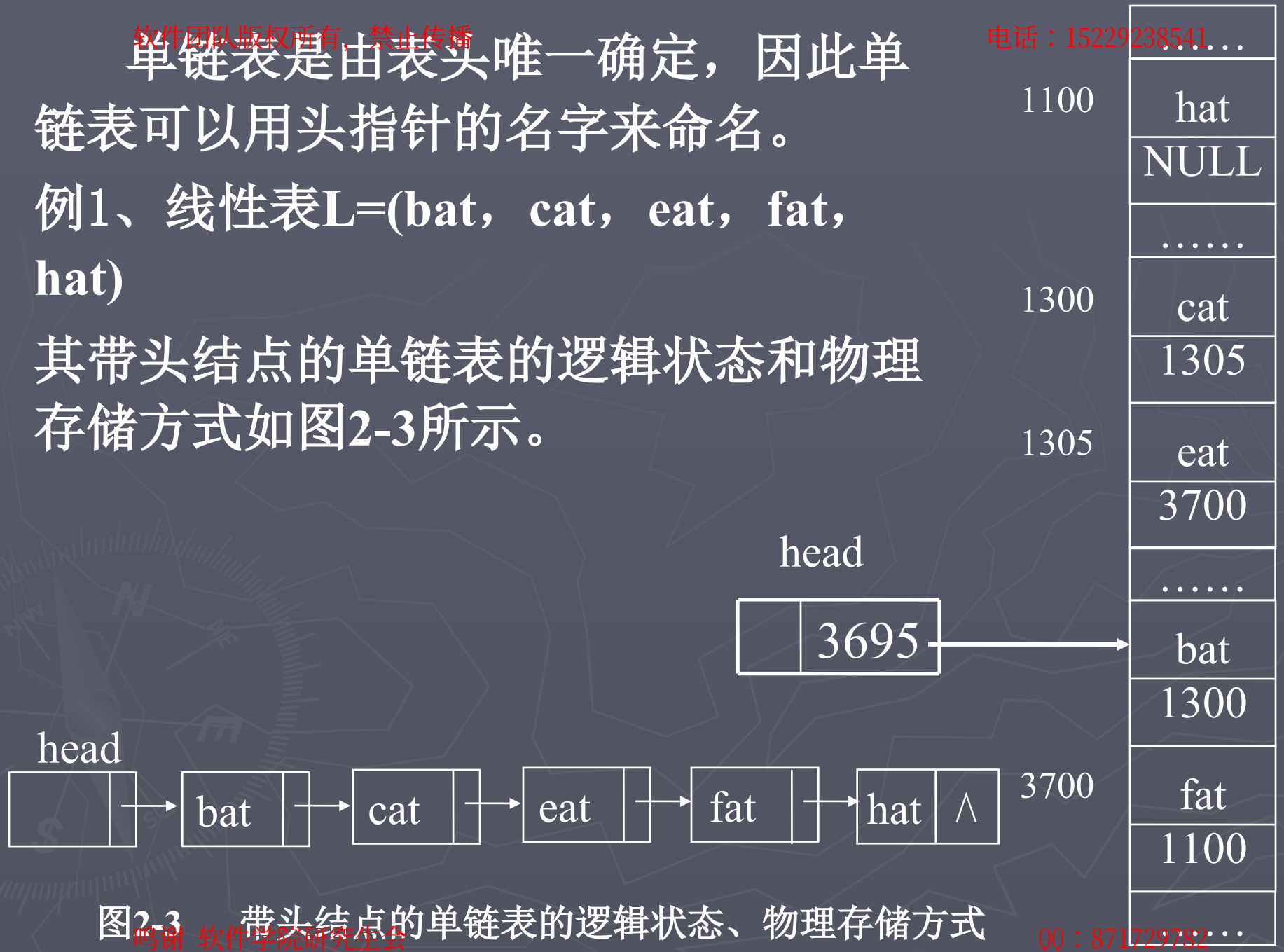


图2-3　带头结点的单链表的逻辑状态、物理存储方式

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

1 结点的描述与实现

C语言中用带指针的结构体类型来描述

```
typedef  struct  Lnode
  {  ElemType  data;     /*数据域，保存结点的值 */
     struct   Lnode  *next;     /*指针域*/
  }LNode;        /*结点的类型 */
```

2 结点的实现

结点是通过动态分配和释放来的实现，即需要时分配，不需要时释放。实现时是分别使用C语言提供的标准函数：malloc()，realloc()，sizeof()，free()。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

动态分配　p=(LNode*)malloc(sizeof(LNode));

函数malloc分配了一个类型为LNode的结点变量的空间，并将其首地址放入指针变量p中。

动态释放　free(p)；

系统回收由指针变量p所指向的内存区。P必须是最近一次调用malloc函数时的返回值。

3　最常用的基本操作及其示意图

(1) 结点的赋值

```
LNode  *p;
p=(LNode*)malloc(sizeof(LNode));
p->data=20; p->next=NULL ;
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

(2) 常见的指针操作

① q=p；

② q=p->next；

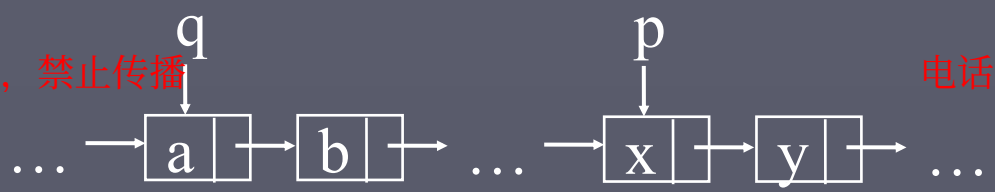
③ p=p->next；

④ q->next=p；

(a)

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541



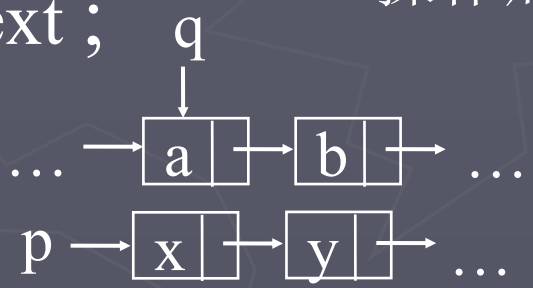
操作前

(b)

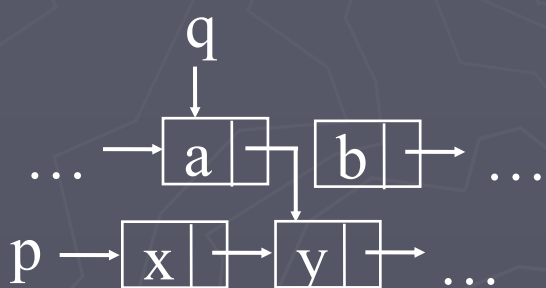
操作后

⑤ q->next=p->next ;

(a)



操作前



操作后

(b)

操作前

操作后

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

2.3.2 单线性链式的基本操作

1 建立单链表

假设线性表中结点的数据类型是整型，以32767作为结束标志。动态地建立单链表的常用方法有如下两种：头插入法，尾插入法。

(1) 头插入法建表

从一个空表开始，重复读入数据，生成新结点，将读入数据存放到新结点的数据域中，然后将新结点插入到当前链表的表头上，直到读入结束标志为止。即每次插入的结点都作为链表的第一个结点。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
LNode  *create_LinkList(void)
  /*  头插入法创建单链表,链表的头结点head作为返回值  */
  {    int data ;
     LNode *head, *p;
     head= (LNode  *) malloc( sizeof(LNode));
     head->next=NULL;        /*  创建链表的表头结点 head  */
     while (1)
       {   scanf("%d", &data) ;
          if (data==32767)  break ;
          p= (LNode  *)malloc(sizeof(LNode));
          p->data=data;     /*  数据域赋值  */
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

```
        p->next=head->next; head->next=p;
            /* 钩链，新创建的结点总是作为第一个结点
        */
    }
  return (head);
}
```

(2) 尾插入法建表

头插入法建立链表虽然算法简单，但生成的链表中结点的次序和输入的顺序相反。若希望二者次序一致，可采用尾插法建表。该方法是将新结点插入到当前链表的表尾，使其成为当前链表的尾结点。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
LNode  *create_LinkList(void)
   /*  尾插入法创建单链表,链表的头结点head作为返回值  */
  {   int data ;
     LNode *head, *p, *q;
     head=p=(LNode  *)malloc(sizeof(LNode));
     p->next=NULL;        /*  创建单链表的表头结点 head  */
     while (1)
        {    scanf("%d",& data);
           if (data==32767)  break ;
           q= (LNode  *)malloc(sizeof(LNode));
           q->data=data;     /*  数据域赋值  */
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15[illegible]

```
        /*钩链，新创建的结点总是作为最后一个结点*/
    }
  return (head);
}
```

无论是哪种插入方法，如果要插入建立的单线性链表的结点是n个，算法的时间复杂度均为O(n)。

对于单链表，无论是哪种操作，只要涉及到钩链(或重新钩链)，如果没有明确给出直接后继，钩链(或重新钩链)的次序必须是“先右后左”。

软件团队版权所有，禁止传播
电话：15229238541

2 单链表的查找

(1) **按序号查找** 取单链表中的第i个元素。

对于单链表，不能象顺序表中那样直接按序号i访问结点，而只能从链表的头结点出发，沿链域next逐个结点往下搜索，直到搜索到第i个结点为止。因此，**链表不是随机存取结构**。

设单链表的长度为n，要查找表中第i个结点，仅当1≦i≦n时，i的值是合法的。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
ElemType  Get_Elem(LNode *L ,  int  i)
  {   int j ;  LNode *p;
     p=L->next;  j=1;      /*  使p指向第一个结点  */
     while  (p!=NULL && j<i)
        {   p=p–>next;  j++;  }        /*  移动指针p，j
        计数  */
     if  (j!=i)  return(-32768) ;
     else      return(p->data);
        /*   p为NULL 表示i太大;  j>i表示i为0  */
  }
```

移动指针p的频度：

i<1时：0次；i∈[1,n]：i-1次；i>n：n次。

鸣谢：软件学院研究生会 QQ：871729782

时间复杂度：O(n)

软件团队版权所有，禁止传播
电话：15229238541

(2) 按值查找

按值查找是在链表中，查找是否有结点值等于给定值key的结点？若有，则返回首次找到的值为key的结点的存储位置；否则返回NULL。查找时从开始结点出发，沿链表逐个将结点的值和给定值key作比较。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
LNode  *Locate_Node(LNode *L，int key)
/*  在以L为头结点的单链表中查找值为key的第一个结点  */
   {   LNode *p=L–>next;
       while（ p!=NULL&& p–>data!=key)
       p=p–>next;
       if（p–>data==key）  return p;
       else
          {    printf(“所要查找的结点不存在!!\n”);
             retutn(NULL);
          }
   }
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

3 单链表的插入

插入运算是将值为e的新结点插入到表的第i个结点的位置上，即插入到a_{i-1}与a_i之间。因此，必须首先找到a_{i-1}所在的结点p，然后生成一个数据域为e的新结点q，q结点作为p的直接后继结点。

算法描述

```
void  Insert_LNode(LNode *L，int i，ElemType e)
   /*  在以L为头结点的单链表的第i个位置插入值为e的结点 */
  {  int  j=0; LNode *p，*q;
     p=L–>next ;
     while  ( p!=NULL&& j<i-1)
```

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

```
    if (j!=i-1)   printf("i太大或i为0!!\n");
    else
      {  q=(LNode *)malloc(sizeof(LNode));
         q->data=e;   q->next=p->next;
         p->next=q;
      }
  }
```

设链表的长度为n，合法的插入位置是1≦i≦n。算法的时间主要耗费移动指针p上，故时间复杂度亦为O(n)。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

4 单链表的删除

(1) 按序号删除

删除单链表中的第i个结点。

为了删除第i个结点a_i，必须找到结点的存储地址。该存储地址是在其直接前趋结点a_{i-1}的next域中，因此，必须首先找到a_{i-1}的存储位置p，然后令p–>next指向a_i的直接后继结点，即把a_i从链上摘下。最后释放结点a_i的空间，将其归还给“存储池”。

设单链表长度为n，则删去第i个结点仅当1≦i≦n时是合法的。则当i=n+1时，虽然被删结点不存在，但其前趋结点却存在，是终端结点。故判断条件之一是p–>next!=NULL。显然此算法的时间复杂度也是O(n)

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

算法描述

```
void  Delete_LinkList(LNode *L,  int i)
   /*  删除以L为头结点的单链表中的第i个结点   */
  {  int  j=1;  LNode *p,  *q;
     p=L;  q=L->next;
     while  ( p->next!=NULL&& j<i)
        {  p=q;  q=q–>next;  j++;  }
     if  (j!=i)     printf(“i太大或i为0!!\n ”);
     else
        {  p–>next=q–>next;   free(q);    }
  }
```

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

(2) 按值删除

删除单链表中值为key的第一个结点。

与按值查找相类似，首先要查找值为key的结点是否存在？若存在，则删除；否则返回NULL。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

算法描述

```
void  Delete_LinkList(LNode *L，int key)
  /*  删除以L为头结点的单链表中值为key的第一个结点  */
  {    LNode *p=L,  *q=L–>next;
     while（ q!=NULL&& q–>data!=key)
       {  p=q;  q=q–>next;   }
     if  (q–>data==key)
       {  p->next=q->next;  free(q);   }
     else
       printf(“所要删除的结点不存在!!\n”);
  }
```

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229383511

算法的执行与形参key有关，平均时间复杂度为O(n)。

从上面的讨论可以看出，链表上实现插入和删除运算，无需移动结点，仅需修改指针。解决了顺序表的插入或删除操作需要移动大量元素的问题。

变形之一：

删除单链表中值为key的所有结点。

与按值查找相类似，但比前面的算法更简单。

基本思想：从单链表的第一个结点开始，对每个结点进行检查，若结点的值为key，则删除之，然后检查下一个结点，直到所有的结点都检查。

软件团队版权所有，禁止传播
电话：15229238541

算法描述

```
void  Delete_LinkList_Node(LNode *L，int key)
  /* 删除以L为头结点的单链表中值为key的第一个结点 */
  {    LNode *p=L, *q=L–>next;
     while ( q!=NULL)
       { if (q–>data==key)
            { p->next=q->next; free(q); q=p->next; }
          else
            { p=q; q=q–>next;  }
       }
  }
```

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

变形之一：

删除单链表中所有值重复的结点，使得所有结点的值都不相同。

与按值查找相类似，但比前面的算法更复杂。

基本思想：从单链表的第一个结点开始，对每个结点进行检查：检查链表中该结点的所有后继结点，只要有值和该结点的值相同，则删除之；然后检查下一个结点，直到所有的结点都检查。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
void  Delete_Node_value(LNode *L)
  /*  删除以L为头结点的单链表中所有值相同的结点  */
  {    LNode *p=L->next, *q, *ptr;
     while（ p!=NULL）  /*  检查链表中所有结点  */
       {   *q=p, *ptr=p–>next;
          /*  检查结点p的所有后继结点ptr  */
          while (ptr!=NULL)
             {  if (ptr–>data==p->data)
                   {  q->next=ptr->next;  free(ptr);
                      ptr=q->next;  }
                else  {  q=ptr;  ptr=ptr–>next;   }
             }
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

```
        p=p->next ;
    }
}
```

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

5 单链表的合并

设有两个有序的单链表，它们的头指针分别是La、Lb，将它们合并为以Lc为头指针的有序链表。合并前的示意图如图2-4所示。

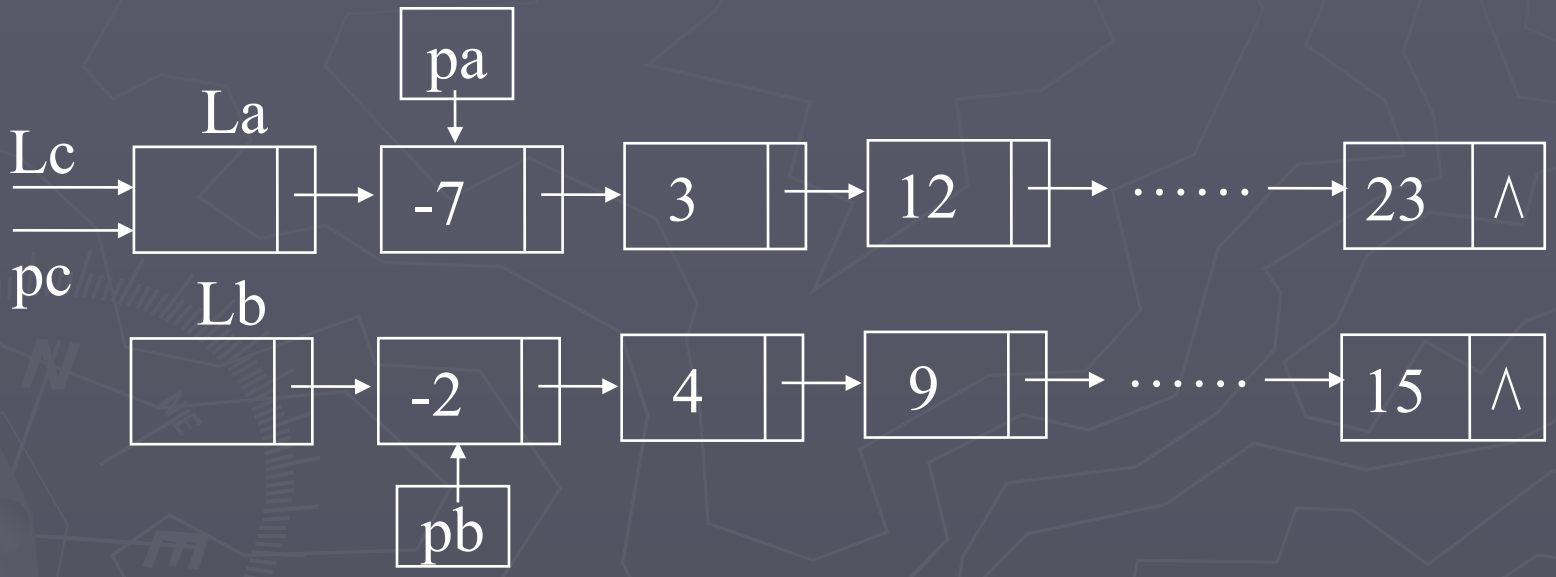


图2-4　两个有序的单链表La，Lb的初始状态

QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

合并了值为-7，-2的结点后示意图如图2-5所示。

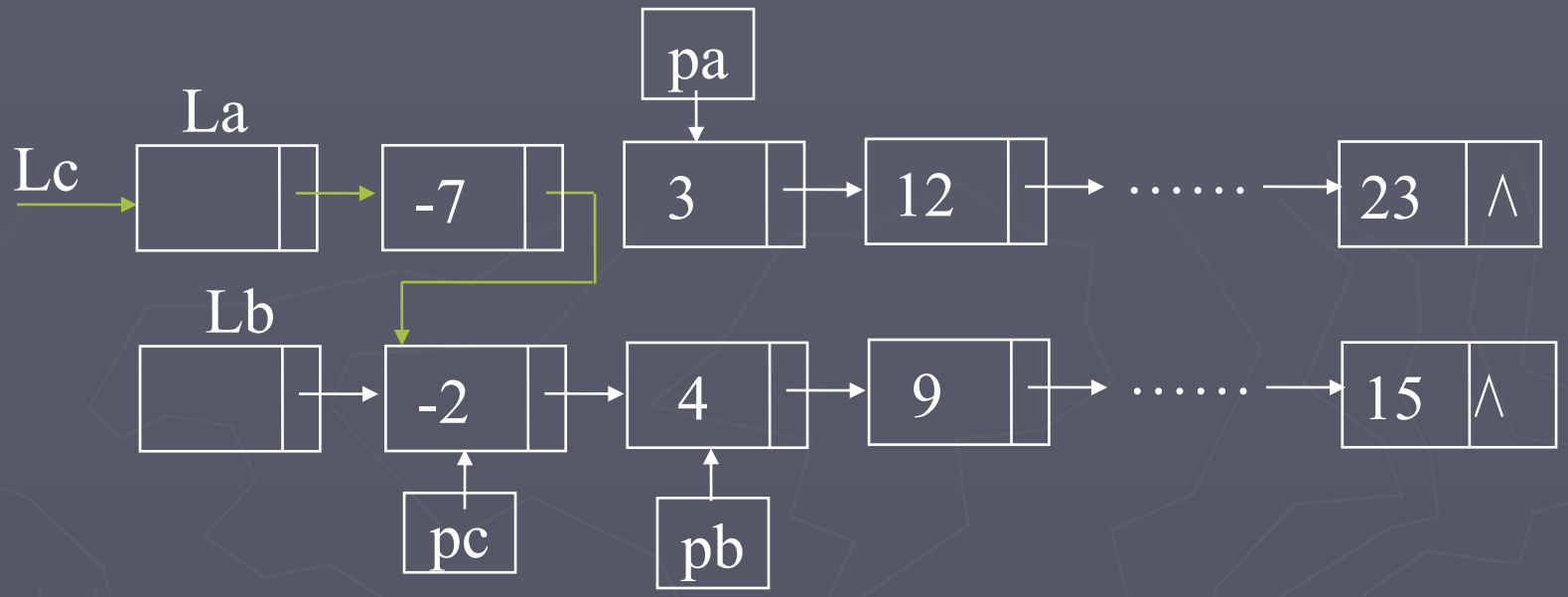


图2-5　合并了值为-7，-2的结点后的状态

算法说明

算法中pa，pb分别是待考察的两个链表的当前结点，pc是合并过程中合并的链表的最后一个结点。

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
LNode  *Merge_LinkList(LNode *La,  LNode *Lb)
   /*  合并以La, Lb为头结点的两个有序单链表  */
  {   LNode *Lc,  *pa ,  *pb ,  *pc, *ptr ;
     Lc=La ;  pc=La  ;   pa=La->next ;  pb=Lb->next  ;
     while (pa!=NULL      && pb!=NULL)
       {  if  (pa->data<pb->data)
            {   pc->next=pa ;  pc=pa ;   pa=pa->next  ;  }
          /*  将pa所指的结点合并，pa指向下一个结点  */
          if  (pa->data>pb->data)
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

```
        if (pa->data==pb->data)
          {  pc->next=pa; pc=pa;  pa=pa->next ;
             ptr=pb;pb=pb->next;
          free(ptr);  }
          /* 将pa所指的结点合并，pb所指结点删除 */
       }
     if (pa!=NULL) pc->next=pa;
    else  pc->next=pb;   /*将剩余的结点链上*/
    free(Lb);
    return(Lc);
  }
```

算法分析

西谢 软件学院研究生会 QQ：871729782

若La、Lb两个链表的长度分别是m、n，则链表

软件团队版权所有，禁止传播
电话：15229238541

2.3.3 循环链表

循环链表(Circular Linked List)：是一种头尾相接的链表。其特点是最后一个结点的指针域指向链表的头结点，整个链表的指针域链接成一个环。

从循环链表的任意一个结点出发都可以找到链表中的其它结点，使得表处理更加方便灵活。

图2-6是带头结点的单循环链表的示意图。

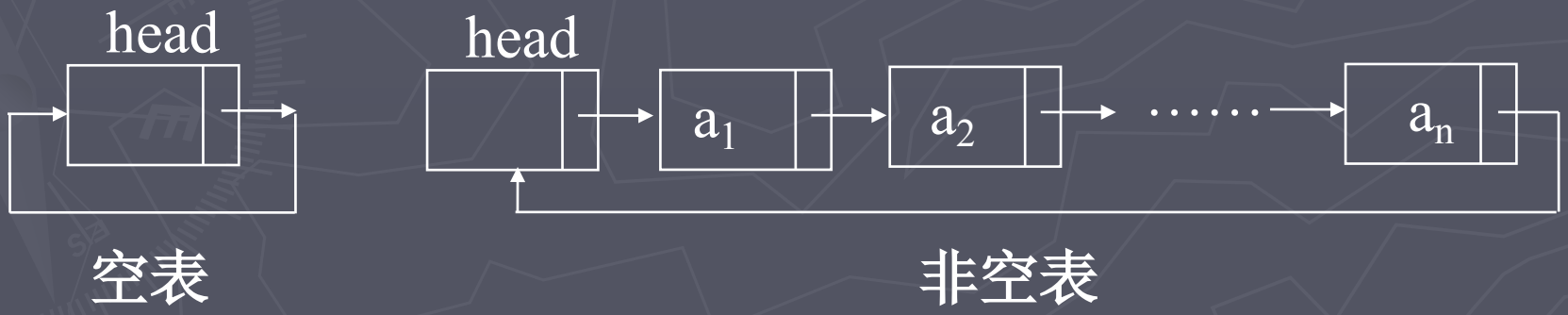


图2-6　单循环链表示意图

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

循环链表的操作

对于单循环链表，除链表的合并外，其它的操作和单线性链表基本上一致，仅仅需要在单线性链表操作算法基础上作以下简单修改：

(1) 判断是否是空链表：head->next==head ；

(2) 判断是否是表尾结点：p->next==head ；

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

2.4 双向链表

双向链表(Double Linked List)：指的是构成链表的每个结点中设立两个指针域：一个指向其直接前趋的指针域prior，一个指向其直接后继的指针域next。这样形成的链表中有两个方向不同的链，故称为双向链表。

和单链表类似，双向链表一般增加头指针也能使双链表上的某些运算变得方便。

将头结点和尾结点链接起来也能构成循环链表，并称之为双向循环链表。

双向链表是为了克服单链表的单向性的缺陷而引入的。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

1 双向链表的结点及其类型定义

双向链表的结点的类型定义如下。其结点形式如图2-7所示，带头结点的双向链表的形式如图2-8所示。

```
typedef struct DuInode
    {    ElemType  data ;
         struct DuInode  *prior , *next ;
    }DulNode ;
```

图2-7　双向链表结点形式

空双向链表　　非空双向链表

图2-8　带头结点的双向链表形式

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15222333**

双向链表结构具有对称性，设p指向双向链表中的某一结点，则其对称性可用下式描述：

(p->prior)->next=p=(p->next)->prior ;

结点p的存储位置存放在其直接前趋结点p->prior的直接后继指针域中，同时也存放在其直接后继结点p->next的直接前趋指针域中。

2 双向链表的基本操作

(1) 双向链表的插入 将值为e的结点插入双向链表中。插入前后链表的变化如图2-9所示。

图2-9 双向链表的插入

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229585341

① 插入时仅仅指出直接前驱结点，钩链时必须注意先后次序是：“先右后左”。部分语句组如下：

```
S=(DulNode *)malloc(sizeof(DulNode));
S->data=e;
S->next=p->next;   p->next->prior=S;
p->next=S;  S->prior=p;    /* 钩链次序非常重要 */
```

② 插入时同时指出直接前驱结点p和直接后继结点q，钩链时无须注意先后次序。部分语句组如下：

```
S=(DulNode *)malloc(sizeof(DulNode));
S->data=e;
p->next=S;       S->next=q;
S->prior=p;       q->prior=S;
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

(2) 双向链表的结点删除

设要删除的结点为p，删除时可以不引入新的辅助指针变量，可以直接先断链，再释放结点。部分语句组如下：

```
p->prior->next=p->next;
p->next->prior=p->prior;
free(p);
```

注意：

与单链表的插入和删除操作不同的是，在双向链表中插入和删除必须同时修改两个方向上的指针域的指向。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

2.5 一元多项式的表示和相加

1 一元多项式的表示

一元多项式 $p(x)=p_0+p_1x+p_2x^2+\ \dots\ +p_nx^n$，由n+1个系数唯一确定。则在计算机中可用线性表(p_0，p_1，p_2，…，p_n)表示。既然是线性表，就可以用顺序表和链表来实现。两种不同实现方式的元素类型定义如下：

(1) 顺序存储表示的类型

```
typedef struct
  { float  coef;   /*系数部分*/
    int    expn;   /*指数部分*/
  } ElemType ;
```

(2) 链式存储表示的类型

```
typedef struct ploy
  { float coef ;    /*系数部分*/
    int   expn ;    /*指数部分*/
    struct ploy  *next ;
  } Ploy ;
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

2 一元多项式的相加

不失一般性，设有两个一元多项式：

$P(x)=p_0+p_1x+p_2x^2+ \dots +p_nx^n$，

$Q(x)=q_0+q_1x+q_2x^2+ \dots +q_mx^m$ （$m<n$）

$R(x)=P(x)+ Q(x)$

$R(x)$由线性表$R((p_0+q_0)，(p_1+q_1)，(p_2+q_2)，\dots，(p_m+q_m)，\dots，p_n)$唯一表示。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

(1) 顺序存储表示的相加

线性表的定义

```
typedef  struct
   {   ElemType  a[MAX_SIZE];
       int    length;
   }Sqlist  ;
```

用顺序表示的相加非常简单。访问第5项可直接访问：**L.a[4].coef ， L.a[4].expn**

(2) 链式存储表示的相加

当采用链式存储表示时，根据结点类型定义，凡是系数为0的项不在链表中出现，从而可以大大减少链表的长度。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播
电话：15229238541

一元多项式相加的实质是：

- 指数不同： 是链表的合并。
- 指数相同： 系数相加，和为0，去掉结点，和不为0，修改结点的系数域。

算法之一：

就在原来两个多项式链表的基础上进行相加，相加后原来两个多项式链表就不在存在。当然再要对原来两个多项式进行其它操作就不允许了。

鸣谢 软件学院研究生会
QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

算法描述

```
Ploy *add_ploy(ploy *La， ploy *Lb)
  /* 将以La，Lb为头指针表示的一元多项式相加 */
  {  ploy *Lc,*pc,*pa,*pb,*ptr;  float x;
     Lc=pc=La;pa=La->next;pb=Lb->next;
     while (pa!=NULL&&pb!=NULL)
       { if (pa->expn<pb->expn)
           { pc->next=pa;pc=pa;pa=pa-
         >next;  }
             /* 将pa所指的结点合并，pa指向下一个结点
         */
         if (pa->expn>pb->expn)
```

鸣谢 软件学院研究生会 QQ：871729782

软件团队版权所有，禁止传播 电话：15229238541

```
else
  { x=pa->coef+pb->coef;
    if (abs(x)<=1.0e-6)
        /* 如果系数和为0，删除两个结点 */
       { ptr=pa; pa=pa->next;
free(ptr);
         ptr=pb; pb=pb->next;
free(ptr); }
      else    /* 如果系数和不为0，修改其中一
个结点的系数域，删除另一个结点 */
       {  pc->next=pa; pa->coef=x;
          pc=pa; pa=pa->next;
          ptr=pb; pb=pb->next;
```

鸣谢 软件学院研究生会 QQ：871729782