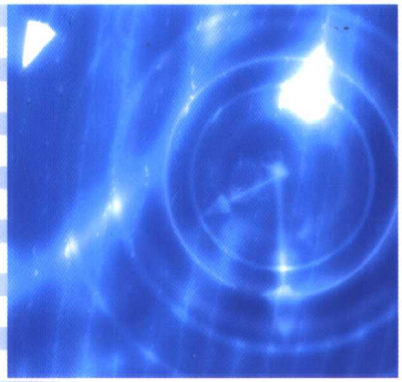
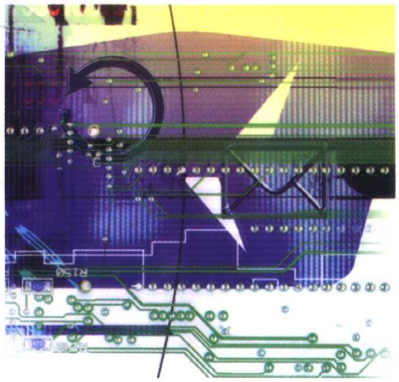
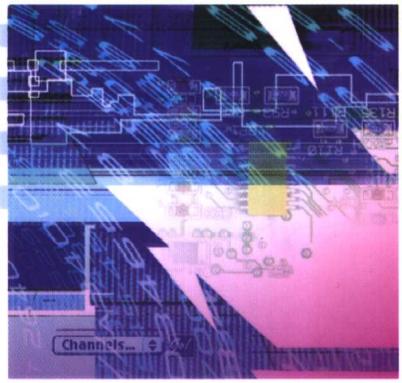
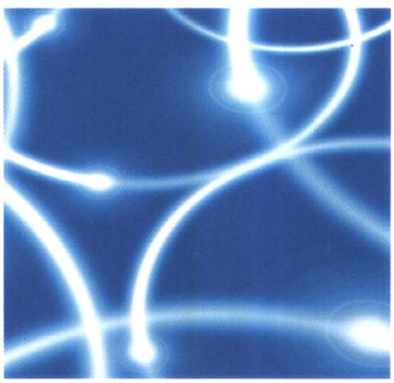
21世纪高等学校应用型教材

数据结构
(C语言版)

□ 胡学钢 主编

高等教育出版社
Higher Education Press

21世纪高等学校应用型教材

数据结构(C语言版)

胡学钢　主　编

高等教育出版社

内容提要

为满足不同专业和教学层次对数据结构课程的要求，本书从数据结构的核心内容和实际需要出发，以简明的方式介绍了数据结构的基本知识。全书共8章，主要内容包括：概论、线性表、栈、队列、数组、树、图、查找、排序和文件等。每章后都配有习题。

本书语言精炼，例题丰富，内容由浅入深，简明易懂，可作为本科非计算机专业、成人以及高职高专计算机相关专业的教材或参考书。

本书的配套电子教案可从高等教育出版社网站下载，网址为 http://www.hep.edu.cn 或 http://cs.hep.com.cn。此外，本书还配有实验指导和习题解答可供读者学习参考。

图书在版编目（CIP）数据

数据结构：C语言版/胡学纲主编. —北京：高等教育出版社，2004.4
ISBN 7-04-013292-3

Ⅰ.数… Ⅱ.胡… Ⅲ.①数据结构-高等学校：技术学校-教材②C语言-程序设计-高等学校：技术学校-教材 Ⅳ.①TP311.12②TP312

中国版本图书馆CIP数据核字（2003）第125617号

出版发行	高等教育出版社	购书热线	010-64054588
社　　址	北京市西城区德外大街4号	免费咨询	800-810-0598
邮政编码	100011	网　　址	http://www.hep.edu.cn
总　　机	010-82028899		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	中国农业出版社印刷厂		
开　　本	787×1092　1/16	版　　次	2004年1月第1版
印　　张	11.5	印　　次	2004年4月第2次印刷
字　　数	280 000	定　　价	16.00元

前　言

在计算机及应用专业及相关专业的课程中，数据结构是重要的专业基础课程，是提高软件设计水平以及学习后续课程所必需的基础。课程中介绍了软件设计中常见的线性表、串、栈和队列、数组、树和二叉树、图、文件等数据结构及其在计算机内存中的存储结构和各种操作的实现，介绍了软件设计中常用的排序和查找方法，并讨论有关运算的性能。通过对这些内容的学习，将使学生能熟练地掌握各种常用结构的特性和各种运算的实现方法及其性能，并能在实际应用中根据具体问题的要求设计出合理的数据结构和运算。

由于该课程中的许多内容较抽象，并缺乏有效的实验条件，使学习数据结构课程的难度较大。

为此，本书尽量以通俗易懂的语言方式介绍有关知识，并给出了必要的例题及其分析，以帮助读者理解知识点并掌握学习方法，易于激发读者的学习兴趣，可收到良好的学习效果。每章后都给出了习题，有助于读者及时巩固所学的知识。

第 1 章介绍了数据结构课程的研究内容、特点以及一些所需的基础知识和学习方法，有助于整个课程的学习。

第 2 章介绍了线性表和串的逻辑结构和运算，重点讨论了顺序表和链表结构及其基本运算实现，并给出了大量例题及分析讲解。

第 3 章介绍了栈、队列和数组这三种结构的逻辑结构和运算，顺序栈、链栈、顺序(循环)队列和链队列及其基本运算实现，介绍了数组的有关内容。

第 4 章介绍了树和森林的逻辑结构、存储结构和运算，介绍了二叉树的概念、性质、存储结构，重点讨论了二叉树的遍历运算方法及其应用，讨论了线索二叉树的概念、应用及相关算法，并给出了相应的例题及分析，另外还介绍了哈夫曼树的有关知识及其应用。

第 5 章介绍了图的有关概念，介绍了两种常用的存储结构，重点讨论了两种遍历算法的方法和应用，最后以通俗的语言介绍了最小生成树、拓扑排序和最短路径等几个图的应用问题及求解算法。

第 6 章所介绍的查找是软件设计中常用的基本运算，本章讨论了在顺序表、树表和散列表等几种表结构上的查找方法及其性能。

第 7 章重点讨论了各种排序的方法、算法及其性能分析。

第 8 章对关于文件方面的基本知识作了简要介绍。

本书由胡学钢主编。其中，第 1 章、第 4 章和第 5 章由胡学钢编写，第 2 章和第 8 章由范宝德编写，第 3 章由张晶编写，第 6 章由阙夏编写，第 7 章由周红鹃编写。全书由胡学钢最后统稿。

由于受水平及时间所限，书中如存在不足之处，恳请读者指正。

胡学钢

2003 年 12 月于合肥工大

策划编辑　雷顺加
责任编辑　萧　潇
封面设计　王凌波
责任印制　杨　明

目　录

第1章 概 论

本章导读

数据结构是计算机专业重要的专业基础课程，直接关系到后续课程的学习和软件设计水平的提高。本章首先通过实例来说明《数据结构》课程在软件设计中的作用以及在计算机专业课程中的地位，然后介绍与整个课程有关的概念、术语、算法及其描述语言和算法分析等。

1.1 数据结构的研究内容

数据结构这门课程在计算机专业中有什么作用？下面从应用计算机解决实际问题的过程来谈谈本课程在软件设计中的作用。

1.1.1 用计算机解决实际问题的过程

在用计算机解决实际问题时，一般要经过几个步骤：首先，对具体问题抽象出数学模型，然后针对数学模型设计出求解算法，最后编出程序上机调试，直至得到最终的解答。数值计算问题的数学模型一般可由数学方程或数学公式来描述。但是，对非数值计算问题，如图书资料的检索、职工档案管理、博弈游戏等问题，它们的数学模型无法用数学方程或数学公式来描述。在这类问题的处理对象中的各分量不再是单纯的数值型数据，更多的是字符、字符串及其他用编码表示的信息。因此，首要的问题是把处理对象中的各种信息按照其逻辑特性组织起来，再存储到计算机中。然后设计出求解算法，并编写出相应的程序。下面简述各环节的有关内容。

建立模型：一般情况下，实际应用问题可能是各种各样的，例如我们所熟悉的工资表的处理问题、学生成绩管理问题、电话号码查询问题等。这些问题无论是所涉及到的数据还是其操作要求，都可能存在一定的差异。尽管如此，许多应用问题之间还是具有一定的相似之处。例如，虽然工资表和学生成绩表的具体信息(栏目)不同，但如果将两个表中每个人的工资信息和成绩信息看做一个整体，则这两个表结构之间就有了某些共性。从操作方面来看，虽然对这两种表的操作存在差异，但也存在一些相同的基本操作。例如，查询一个人的工资信息和成绩信息，修改有关信息等。

正因为许多不同的问题之间存在的某些共性，使我们可以将一个具体的问题用这些共性的形式描述出来，这就是通常所说的**建立模型**。建立问题的模型通常包括所描述问题中的数据对象及其关系的描述、问题求解的要求及方法等方面。建立问题模型有这样的好处：因为所涉及到的许多基本模型在有关的课程中已有介绍，因此通过建立模型，就可以将一个具体的问题转换成所熟悉的模型，然后借助于这一模型来实现。数据结构、离散数学及许多数学课程中就介绍了许多模型。例如，要描述一个群体中个体之间的关系时，我们可以采用《数据结构》和《离散数学》中所介绍的图结构。要描述一个工程内的关系或进展情况时，我们可以采用《数据结构》中所介绍的 AOV 网或 AOE 网等。即使所建立的模型没有现成的求解方法，也相对易于构造求解方法。

构造求解算法：在建立了模型之后，一个具体的问题就转变成了一个用模型所描述的抽象问题。借助于这一模型以及已有的知识（例如数据结构中有关图结构的基本知识），我们可以相对容易地描述出原问题的求解方法，即算法。从某种意义上说，该算法不仅能实现原问题的求解，而且还可能实现许多类似的具体问题的求解，尽管这些具体问题的背景及其描述形式可能存在较大的差异。

选择存储结构：在构造出求解算法之后，就需要考虑在计算机上实现求解了。为此，需要做两方面的工作，其一是选择合适的存储结构，以便将问题所涉及到的数据（包括数据中的基本对象及对象之间的关系）存储到计算机中。不同的存储形式对问题的求解实现有较大的影响，所占用的存储空间也可能有较大的差异。

编写程序：在选择了存储结构之后，就可以进行实现求解的另一项工作，即编写程序了。存储形式和问题要求决定了编写程序的方法。

测试：在编写出完整的程序之后，需要经过测试才能交付使用。

例如，假定要编写一个计算机程序以查询某市或单位的私人电话。对任意给定的一个姓名，若此人装有电话，则要求迅速找到其电话号码，否则指出此人没有装电话。

解决此问题时，首先要构造一张电话号码登记表，表中每个登记项有两条信息：姓名及电话号码。在将众多的登记项合在一起构成表时，有多种不同的组织形式，查找的速度取决于表的结构及存储方式。

最简单的方式是把表中的信息，按照某种次序（如登记的次序）依次存储在计算机内一组连续的存储单元中。用高级语言表述，就是把整个表作为一个数组，表的每项（即一个人的姓名和电话号码）是数组的一个元素。查找时从表的第一项开始，依次查对姓名，直到找出指定的姓名或是确定表中没有要找的姓名为止。这种查找方法对于一个规模不大的单位或许是可行的，但对于一个有几十万乃至几百万个私人电话的城市就不实用了。因此，一种常用的做法是把这张表按姓氏或姓氏笔划排列，并另造一张姓名索引表。这有点像汉语字典的形式。对这种表的查找过程，可以先在索引表中查对姓氏，然后根据索引表中的地址到登记表中核查姓名，这样查找登记表时就无须查找其他姓氏的名字了。因此，在这种新的结构上产生的查找方法就会更有效。这两张表便是我们为解决电话号码查询问题而建立的数学模型。这类模型的主要操作是按照某个特定要求（如给定姓名）对登记表进行查询。诸如此类的还有人事档案管理、图书资料管理等。在这类文档管理的数学模型中，计算机处理的对象之间通常存在一种简单的线性关系，故这类数学模型可称为线性的数据结构。

数据结构课程涉及到上述求解过程中的大多数步骤，如下所示：

与建立模型的关系:数据结构课程中介绍了许多基本的数据结构模型及其运算实现。例如,线性表、栈和队列、树和二叉树、图、二叉排序树、堆等。通过学习,不仅可以掌握这些基本内容及其应用,还能根据实际问题选择合适的模型。

与算法设计的关系:课程中对每种结构都讨论了相应的基本运算的实现,并且其中的一些算法是非常经典的。掌握这些基本运算的实现方法有助于进行更为复杂的算法设计。

与选择存储结构的关系:课程中对每种结构都讨论了其具体存储结构及其对运算实现的影响。例如,在第 2 章中所介绍的对顺序表进行插入和删除运算,平均需要移动表中一半的元素,而采用链表结构则不需要移动元素。通过对这些内容的学习和比较,可使学生熟练地选择合理的存储结构。

与编程之间的关系:在实现各结构的算法编写时,涉及到许多具有代表性的设计方法。通过对这些方法的学习,有助于提高读者的编程技术的提高。

综上所述,数据结构对提高软件设计人员的设计水平有较大的影响。也正因为如此,这一课程在计算机专业课程中具有极其重要的作用。几乎绝大多数学校和研究单位的计算机专业研究生入学考试都将这一课程定为考试课程之一。

1.1.2 学习数据结构的意义

数据结构作为一门独立的课程,在美国是从 1968 年开始的。在这之前,它的某些内容曾在其他课程中有所阐述。1968 年,美国一些大学计算机系的教学计划中,虽然把数据结构规定为一门课程,但对其内容并做作明确规定。当时,数据结构几乎与图论,特别是与表、树的理论是同义语。随后,数据结构这个概念被扩充到包括网络、集合代数、格、关系等方面,从而变成现在称之为"离散结构"的内容。然而,由于数据必须在计算机中进行处理,因此,不仅要考虑数据本身的数学特性,而且还要考虑数据的存储结构,这就进一步扩大了数据结构的内容。

数据结构是计算机科学中一门综合性的专业基础课程。数据结构的研究不仅涉及计算机硬件(特别是编码理论、存储设备和存取方法等)的研究范围,而且和计算机软件的研究有着密切的关系。无论是编译程序还是操作系统都涉及到如何组织数据,使检索和存取数据更为方便。因此,可以认为数据结构是介于数学、计算机硬件和软件三者之间的一门核心课程。在计算机科学中,数据结构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。

目前,数据结构不仅是大学计算机专业的核心课程之一,而且是一些非计算机专业的主要选修课程之一。

随着计算机应用领域的扩大和软、硬件的发展,"非数值性问题" 显得越来越重要。据统计,当今处理非数值性问题占用了 90%以上的机器时间。从前面的例子可以看到,解决此类问题的关键已不再是数学方法的问题,而是设计出合适的数据结构。瑞士计算机科学家沃斯(N. Wirth)曾提出"算法 + 数据结构 = 程序"。可见,程序设计的实质是对实际问题选择一种好的数据结构,并设计一个好的算法。因此,若仅仅掌握几种计算机语言和程序设计方法,而缺乏数据结构知识,则难以应付众多复杂的课题,且不能有效地利用计算机。

1.2 基本术语

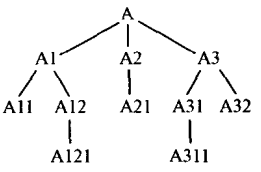
数据是指信息的载体，是能够输入到计算机中，并被计算机识别、存储和处理的符号的集合。数据的形式较多，例如我们前面所介绍的工资报表、学生成绩表、一个家族关系的表示形式、表示一个群体中个体之间关系的图形描述等，如图 1－1 所示。

编号	姓名	基本工资	奖金	…	…

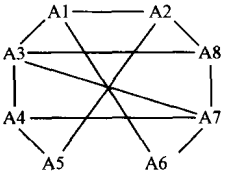
(a) 工资表示例

序号	学号	姓名	成绩	备注

(b) 成绩表示例



(c) 家族关系示例



(d) 群体间关系示例（连线表示相互认识关系）

图 1－1 数据示例

虽然这些数据的形式及运算存在较大的差异，但可以找出其中的共性：都是由若干个具有独立意义的个体所组成的，个体之间存在着某些关系。对这些数据的运算也有相似之处。例如，在家族关系数据中，组成数据的基本个体是人，其中的人与人之间存在着多种关系，例如父子关系、兄弟关系、祖先－后代关系等，其中有些关系是直接表示出来的，还有一些关系则是隐含的。对家族关系数据，通常要涉及到查询特定个体间的关系，插入和删除个体等。

数据结构课程中主要讨论这些具有共性的内容。为便于讨论，先给出有关概念。

数据元素：数据中具有独立意义的个体。例如工资表中的个人工资信息，成绩表中的学生成绩信息，家族关系中的个人等。在有些场合下，也称之为**元素、记录、结点、顶点**等。

字段(域)：虽然将具有独立意义的个体用元素来表示，但在许多情况下还需要特定个体的具体信息，因而涉及到了元素的字段信息。字段是对元素的详细描述，通常情况下，元素可能包含多个字段。例如，在图 1－1 所示的成绩表中，每个元素包括序号、学号、姓名、成绩、备注 5 个字段，分别描述了每个学生的有关信息。

数据结构：在图 1－1 所示的几个数据示例中，元素之间各自具有一定的构成形式，这些构成形式被称为数据结构。**数据结构**是指组成数据的元素之间的结构关系。在图 1－1 中，图(a)，图(b)中的元素依次排列，构成**线性关系**；图(c)中的元素是按树的形式构成**树型结构**；图(d)中的元素构成**图结构**。线性结构、树型结构和图结构是《数据结构》中几类常见的数据结构形式。如果数据中元素之间没有关系，则构成**集合**，这也是一种结构。

通常，称这几类结构为**逻辑结构**，因为只考虑了元素之间的逻辑关系，而没有考虑到其在计算机中的具体实现。

在用计算机解决涉及到数据结构的问题时，需要完成如下任务：

(1) 数据结构的存储：为所涉及的数据结构选择一种存储形式，并将其存储到计算机中，这样就得到了数据结构在内存中的**存储结构**(有时也称为**物理结构**)。一种逻辑结构可能会有多种存储结构。例如，可以采用顺序存储，也可采用连接形式的存储。不同存储结构上所实现的运算的性能可能有一定的差异。

(2) 数据结构**运算**的实现：在选择了数据结构的存储结构之后，就可以实现所给出的运算了。在本课程中，运算的实现一般是以算法的形式给出的，而算法大多以某种描述语言中的子程序的形式给出。由于本书主要以 C 语言作为算法描述语言，所以算法就以 C 语言中的函数形式给出。

由此可见，对一种数据结构，需要涉及到其逻辑结构、存储结构和运算三个方面。也就是说，对每种结构都要注意这三方面的联系。

由于不同的存储形式对算法的时间性能、空间性能等有较大影响，即使是相同的存储结构，也可能会存在不同的算法实现。为此，需要解决这样的问题：究竟何种存储结构更为合适？什么算法更有效？所此，需要对算法进行分析。通过分析，可以知道所实现的算法的性能及所选择的存储结构是否符合要求。有关算法分析部分将在本章的后面部分讨论。

1.3 算法描述及分析

1.3.1 算法描述语言概述

如前所述，对数据结构的运算实现是以算法的形式来描述的。什么是算法？这很难给出一个严格的定义。简单地说，算法就是某类问题的求解方法。下面给出一个关于算法的描述：算法就是一段程序，该程序段对给定的输入可在有限的时间内产生出确定的输出结果。

算法可采用多种描述语言来描述，例如自然语言、计算机语言或某些类语言。各种描述语言在对问题的描述能力方面存在一定的差异。例如，自然语言较为灵活，但不够严谨，而计算机语言虽然严格，但由于语法方面的限制，使得灵活性不足。因此，许多教材中采用的是以一种计算机语言为基础，适当添加某些功能或放宽某些限制而得到的一种类语言。这些类语言既具有计算机语言的严格性，又具有某些灵活性，同时也容易上机实现，因而被广泛接受。目前的许多《数据结构》教材采用类 PASCAL 语言、类 C++ 或类 C 语言作为算法描述语言。

本教材中选用的是以 C 语言为主体的算法描述语言，在 C 语言的基础上适当扩充一些功能或放宽某些限制。所涉及的算法以 C 语言的函数形式给出。

考虑到读者已经学过 C 语言，因而不再对 C 语言部分做详细介绍。下面补充一下所扩充的一些功能描述。

(1) 输入和输出语句：由于 C 语言中的输入和输出语句的形式对数据的类型有一定的限制，因此，本书中采用 C++ 中的独立于数据类型的输入和输出语句。

① 输入：cin>>x；其功能是读入从键盘输入的一个数，并赋给相同类型的变量 x.。其中变量 x 的类型可以是整型、浮点型、字符型等不同类型。

该语句可用下面的形式同时输入多个不同类型的变量。

```
cin>>x1>>x2>>x3>>x4>>x5;
```

② 输出：cout<<exp；其功能是将表达式 exp 的值输出到屏幕上。其中表达式 exp 的类型也可以是整型、浮点型、字符型等不同类型。

该语句可用下面的形式同时输出多个不同类型的表达式的值。

```
cout<<exp1<<exp2<<exp3<<exp4<<exp5;
```

(2) 最大和最小值函数 min 和 max

datatype min(datatype exp1, datatype exp2,…, datatype expn)：

返回表达式 expi(i=1,2,…,n)中的最小的值。其中元素类型 datatype 可以是各种类型。

datatype max(datatype exp1, datatype exp2,…, datatype expn)：

返回表达式 expi(i=1,2,…,n)中的最大的值。

(3) 交换变量的值：x1<==>x2；交换变量 x1 和 x2 的值。

(4) 注释：为简捷起见，本书中程序的注释采用 C++ 中的注释形式，即在双斜线“//”后面的内容就是注释的内容。例如，下面语句的右边就是一个注释。

```
A[i]=i*i;                    //此处为注释内容
```

1.3.2 算法分析

如前所述，为了了解算法的有关性能，需要对算法进行分析。通过分析，不仅可以知道算法的有关性能，而且由此还可知道所选择的存储结构是否符合要求。

衡量算法的主要性能指标包括时间性能、空间性能等。其中，时间性能是指运行算法所需的时间的度量，而空间性能则是指运行算法所需要的辅助空间的规模。在数据结构中，大多数算法分析是针对算法的时间性能进行的。算法的时间性能以时间复杂度来衡量。

为了便于比较算法的时间复杂度，不宜采用在某个具体机器上所运行的时间的形式来表示，而一般是以算法中**基本语句的执行次数**来衡量。然而，在实际应用时，精确计算基本语句的执行次数的很困难，同时也是不必要的，因为许多算法中语句的执行次数要取决于输入数据，可能会有多种复杂的情况。为便于计算，对这一时间复杂度大多采用一种近似的形式来描述，即采用基本语句执行次数的**数量级**来表示。此处所谓数量级是这样定义的：

如果变量 n 的函数 $f(n)$和 $g(n)$满足：$\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=$常数 $k(k\neq\infty,0)$，则称 $f(n)$和 $g(n)$是同一数量级的，并用 $f(n)=O(g(n))$的形式来表示。

由定义可知，两个函数为同一数量级，强调的是在 n 趋向无穷大时，两者“接近”。例如，$\dfrac{n(n+1)}{2}$和 n^2 是同一数量级的，因为$\lim\limits_{n\to\infty}\dfrac{\frac{n(n+1)}{2}}{n^2}=1/2$ 为一个常数。

通过求解算法中语句执行次数的数量级所得到的时间复杂度忽略了其中的“细微”部分，得到了时间性能的近似描述，这一描述有助于对算法时间性能的简要了解。

例 1.1 求解下列各程序段的时间复杂度。

(1) For (i=1; i<n; i++) x++;

解：虽然从循环语句的内部执行来看，该程序段中基本语句的执行数多于循环体的执行次数，但两者是同一数量级的，因此，该问题的求解可以以其循环体的执行次数的数量级的求解来实现。由于循环体执行 $n-1$ 次，因此其时间复杂度为 $O(n)$。

(2) For (i=1; i<n; i++)

For (j=1; j<=i; j++) {x++;}

解：该问题同样以其循环体的执行次数的数量级的求解来实现。由于是双重循环，并且内层循环的循环次数不是常数，因此其计算稍微有些麻烦。计算方法如下：

i=1 时，内层循环执行 1 次(j 从 1 到 1)；

i=2 时，内层循环执行 2 次(j 从 1 到 2)；

i=3 时，内层循环执行 3 次(j 从 1 到 3)；

…… ……

i=$n-1$ 时，内层循环执行 $n-1$ 次(j 从 1 到 $n-11$)；

因此，最内层循环体共执行 $n(n-1)/2$ 次，因而其时间复杂度为 $O(n^2)$。

(3) i=1;

while (i<n) i*=2;

解：对许多初学者来说，该问题的求解有些麻烦，求解方法如下：

由语句可知，循环次数和 i 之间的对应关系如下：

次数	1	2	3	4	…	k
I 值	2^1	2^2	2^3	2^4	…	2^k

假设最后一项即 $2^k=n$，则可知 $k=\log_2 n$。也就是说，循环次数 k 是 $\log_2 n$ 的数量级。即使 2^k 不是正好等于 n，k 也与 $\log_2 n$ 几乎相等。由此可知，该语句段的时间复杂度为 $O(\log_2 n)$。

本章小结

数据结构研究软件设计中的基本技术，是计算机专业重要的专业基础课程。课程中所研究的内容有助于实际问题求解的许多方面。

数据是计算机的操作对象，可以分解为元素的集合，每个元素中可能包含多个称为字段的描述信息。组成数据的元素之间按一定的结构形式组织起来，从而构成数据结构。

数据结构包含逻辑结构、存储结构和运算三个方面。数据结构的逻辑结构侧重描述元素之间的内在联系，不涉及数据结构的具体存储实现。数据结构的存储结构讨论数据的存储结构，不同的存储结构对运算的实现可能有较大的差异。数据结构的运算又可以分解为运算定义、运算实现即算法和算法分析三项内容。运算定义是从有关领域中提出的具有某些共性的问题描述。

运算实现是在选定的存储结构上实现所描述的运算，从而得到算法。算法分析通过对所实现的算法的时间性能和空间性能等方面的分析，来确定算法或存储结构的性能。

有关数据结构几个方面的联系如图 1－2 所示。

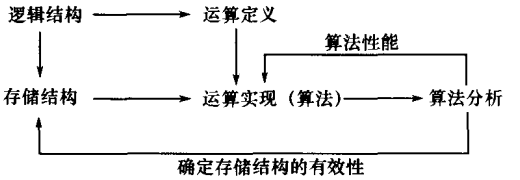


图 1－2 数据结构各方面的联系

算法的时间性能以时间复杂度来描述，是对算法运行时间的近似估计，通常以基本语句的执行次数的数量级来描述，其形式是数据规模（最常见的是 n）的函数的数量级。

习 题 一

1. 选择一个单位的工资表，指出其中的元素、元素的字段以及元素之间的关系，并给出一些最基本的运算。

2. 描述数据结构、逻辑结构、存储结构和运算的有关概念及其相互之间的关系。

3. 已知一个群体中有 n 个人，这些人之间可能存在同学关系，请用一个数据模型来描述这一关系，并给出可能的基本运算。

4. 描述算法所具备的基本特征，并指出算法与程序之间的差异。

5. 计算下列各程序段的时间复杂度。

```
(1) for   (i = 0; i<n; i++ )
          for   (j = i; j<n;j++ ) x++ ;
(2) i = n;
    While   (i>1) i/ = i/2;
(3) for (i = 1; i< = n; i++ )
       for (j = 1; j< = n; j++ )
          for (k = 1; k< = n; k++ )
            x++ ;
(4) for (i = 1; i<n; i++ )
        for (j = 1; j<n; j++ ) x++ ;
    for (k = 1; k<n;k++ ) x++ ;
(5) for (i = 1; i<n; i++ )
      { j = i;
        while (j<n) j* = 2;
      }
```

第2章 线性表

本章导读

线性表是数据结构课程中所介绍的第一种数据结构，是软件设计中最常用也是最基本的一种数据结构。本章是整个课程的基础，首先给出线性表的定义、相关概念和基本运算，然后分别讨论采用顺序存储方式和链式存储方式存储线性表、实现前面所提出的基本运算的算法性能的比较等，并通过实例说明了线性表的应用。最后介绍了串这种特殊的线性表结构。

2.1 线性表的定义和运算

软件设计中经常会涉及到按顺序方式排列的表。例如，英文26个字母表；在编写输出年历的程序时，需要用到一年中每个月的天数表；8086系列中的中断向量表；某班级的学生成绩表等。所有这些表面上不完全相同的表可抽象为本章所要介绍的线性表，其中各元素是依次排列的，这是最为常用的一类数据结构。

2.1.1 线性表的定义

定义：线性表 L 是 n 个元素 $a_1, a_2, \cdots, a_n$ 组成的有限序列，记做 $L=(a_1, a_2, \cdots, a_n)$，其中 $n \geqslant 0$，为**表长度**；当 $n=0$ 时为**空表**，记做 $L=()$。

对线性表中某个元素 a_i 来说，称其前面的元素 a_{i-1} 为 a_i 的直接前驱，称其后面的元素 a_{i+1} 为 a_i 的直接后继。显然，线性表中每个元素最多有一个直接前驱和一个直接后继，这是线性表的特点。

由前面的描述可知，线性表是许多实际应用领域中表结构的抽象形式，因此，线性表中元素在不同的场合可以有不同的**含义**。例如，在字母表(A,B,C,…,Z)中，每个元素是一个字母；在一个学生成绩表中，每个元素是一个学生的成绩信息，其中可能包含学号、姓名、成绩等。但要注意，在同一个表中的各元素的类型是一致的。

2.1.2 线性表的运算

如前所述，线性表结构是许多实际应用中所用到的表结构的抽象，因此对线性表的实际运算可以有很多，例如对我们所熟知的工资表和成绩表就有很多运算的要求。为了便于研究，一般不会也不可能讨论所有的运算，取而代之的是只讨论其基本运算。在此基础上，可以较方便地实现更多的运算。

线性表常用的基本运算有以下 6 个：

(1) 初始化线性表 initial_List(L)：建立线性表的初始结构，即建空表。这也是各种结构都可能要用的运算。

(2) 求表长度 List_length(L)：即求表中元素的个数。

(3) 按序号取元素 get_element(L,i)：取出表中序号为 i 的元素。

(4) 按值查询 List_locate(L,x)：取出指定值为 x 的元素。若存在该元素，则返回其地址；否则，返回一个能指示其不存在的地址值或标记。

(5) 插入元素 List_insert(L,i,x)：在表 L 的第 i 个位置上插入值为 x 的元素。显然，若表中的元素个数为 n，则插入序号 i 应满足 $1 \leqslant i \leqslant n+1$。

(6) 删除元素 List_delete(L,i)：删除表 L 中序号为 i 的元素，显然，待删除元素的序号应满足 $1 \leqslant i \leqslant n$。

虽然只给出了这 6 个基本运算，但借助于这些基本运算，可以构造出其他更为复杂的运算。例如，如果要求删除线性表中值为 x 的元素，则可用上述运算中的两个运算来实现：先引用 List_locate 求出元素的位置，再引用 List_delete 来实现删除。尽管这一实现的时间性能不太好，但在讨论基本运算时，主要还是侧重于其逻辑上的实现，而不是具体程序上的实现。

另外，在特定的存储结构上实现基本运算时，可能会有一定的差异。

2.2 线性表的顺序表存储结构

2.2.1 顺序存储结构

为了在计算机上实现数据结构，首先需要将数据结构存储到计算机中。对此可有许多不同的方法，其中最为简单的方法是**顺序存储方式**：假设有一个足够大的连续的存储空间，则可将表中元素按照其逻辑次序依次存储到这一存储区中，把由此得到的线性表称为**顺序表**。

在具体实现时，一般用高级语言中的数组来对应连续的存储空间。假设最多可存放 maxlen 个元素，则可用数组 data[maxlen]来存储表中元素。另外，由于线性表有插入和删除元素这类改变表中元素个数的运算，因此，为随时了解线性表当前的元素个数(即表长度)，需要另外设置一个变量，以记录其元素个数，此处不妨用 listlen 来记录元素个数。这样，一个线性表的顺序存储结构就有两个分量，将这两个分量合在一起构成了一个整体。如图 2－1 所示，对所设计的存储形式，需要让计算机知道，这就是数据结构的描述。在用 C 语言描述这个由两个分量组成的结构