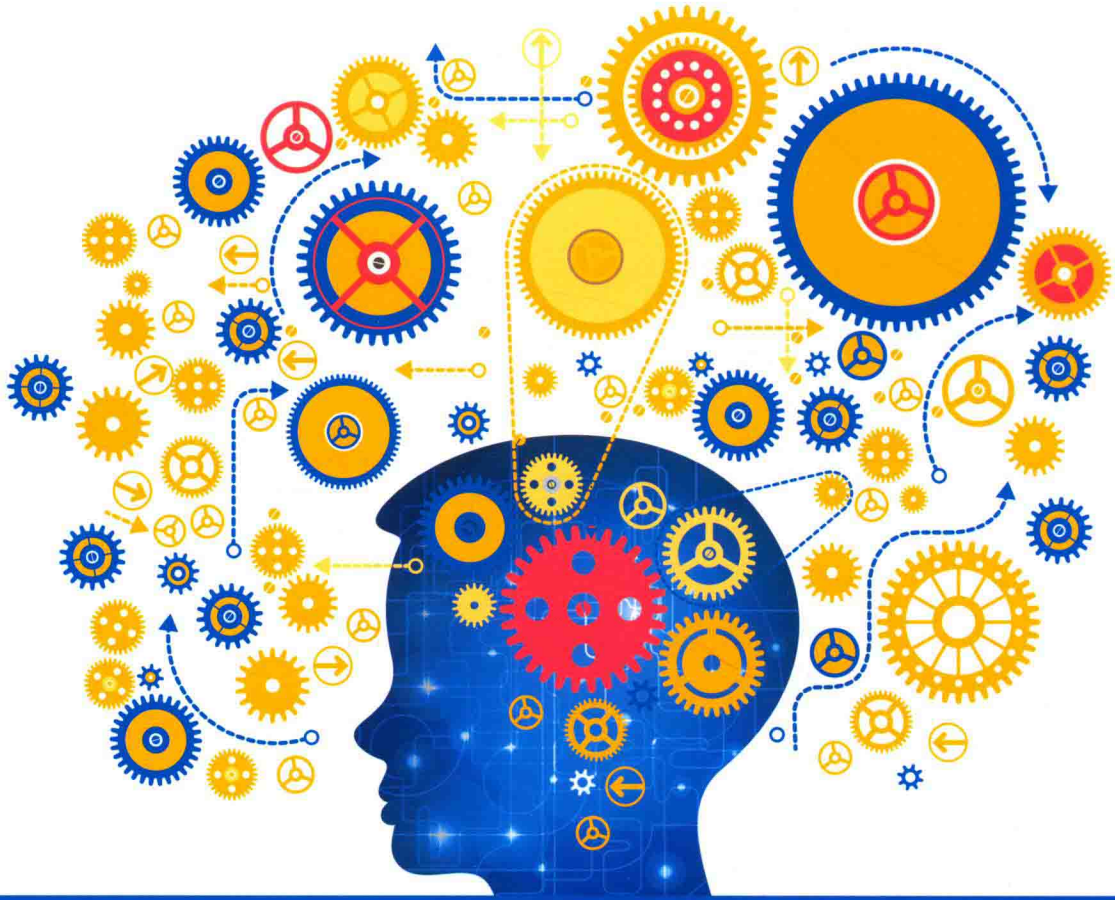
普通高等教育“十三五”规划教材

新工科建设之路·计算机类规划教材



数据结构教程

主编　胡元义　黑新宏

副主编　罗作民　雷西玲　孙旭霞　费蓉　何文娟

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育“十三五”规划教材
新工科建设之路·计算机类规划教材

数据结构教程

胡元义　黑新宏　主编

罗作民　雷西玲　孙旭霞　费　蓉　何文娟　副主编

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书系统地介绍了数据结构的有关内容，主要包括：线性表、栈、队列、串、数组、广义表、树、图等常用的数据逻辑结构和存储结构，各种数据结构的基本操作，以及查找、排序算法等。

本书采用的算法全部用 C 语言描述，各章均附有大量习题。与本书配套的《数据结构教程习题解析与上机指导》（胡元义，黑新宏主编，ISBN 978-7-121-35132-7）详细给出了本书习题的解题思路和参考答案，对书中的所有算法和涉及算法的示例都给出了完整的 C 语言实现程序，并且在 VC++ 6.0 环境下通过上机验证。

本书结构清晰、算法突出。在内容的组织上，本书强调知识的实用性，既注重理论的完整性，化繁为简，又将理论融入具体实例中，化难为易，以达到准确、清楚地阐述相关概念和原理的目的。本书注重对数据结构各章节知识阐述的条理性，书中给出的例子也具有较强的实用性与连贯性，以便使读者对数据结构有全面、透彻的认识。

本书可作为高等院校相关专业本科生及硕士研究生的专业教材或参考书，也可作为相关技术人员的自学用书。

图书在版编目（CIP）数据

数据结构教程 / 胡元义，黑新宏主编. — 北京：电子工业出版社，2018.12
ISBN 978-7-121-35131-0

I. ①数… II. ①胡… ②黑… III. ①数据结构－高等学校－教材 IV. ①TP311.12

中国版本图书馆 CIP 数据核字（2018）第 223129 号

策划编辑：孟 宇
责任编辑：章海涛
印 刷：三河市鑫金马印装有限公司
装 订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×1092 1/16 印张：19 字数：474 千字
版 次：2018 年 12 月第 1 版
印 次：2018 年 12 月第 1 次印刷
定 价：48.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：mengyu@phei.com.cn。

前言

Perface

如果把程序设计比作一棵大树，那么数据结构无疑是大树的躯干，可以说程序设计的精髓就是数据结构。计算机各个领域的应用都要用到各种数据结构，只有较好地掌握数据结构知识，才能在程序设计中做到游刃有余，从而在计算机应用领域的研究和开发中做到胸有成竹。

数据结构课程是计算机专业的一门核心课程，它是在长期的程序设计实践中提炼、升华而成的，反过来又应用于程序设计。数据结构课程同时又是操作系统、编译原理等计算机核心课程的基础，在计算机专业课程的学习中起着承上启下的作用。数据结构是一门应用广泛且最有实用价值的课程，不掌握数据结构知识就难以成为一名合格的软件工程师或计算机工作者。

由于数据结构的原理和算法都比较抽象，且数据结构又具有概念性强、内容灵活、不易掌握和难度大等特点，因此学习起来比较困难。本书在内容的组织和编排上注意掌握内容的难易程度、把握理论的深度、追求应用的广度；在内容的讲解上注重条理性和连贯性，做到化繁为简、化难为易，引导学生由基本概念出发思考数据结构的实质内容及相应算法求解的思路和实现方法，并由此深化对数据结构基本概念的理解，达到举一反三、培养分析问题和解决问题能力的目的。

本书共分 9 章，采用的算法均用 C 语言描述。第 1 章简要介绍了数据结构的基本概念与算法基础；第 2 章介绍了线性表及其两种存储结构——顺序表和单链表的有关概念；第 3 章介绍了栈和队列的概念、特点及应用；第 4 章涉及串的概念，特别是串的模式匹配的有关内容；第 5 章介绍了数组和广义表的有关概念；第 6 章介绍了树与二叉树的有关概念及其应用；第 7 章介绍了图的有关知识和应用；第 8 章给出了各种静态和动态查找表的方法；第 9 章详细讨论了各种排序的方法和特点。

为了便于正确理解数据结构的概念和相关内容，本书除使用大量有针对性的实例外，还采用了图解的方式来解析数据结构中的算法和示例的实现过程。本书的另一个特点是各章习题的概念性强、覆盖面广、内容全面，具有典型性和代表性，这对加深理解各章的知识是必不可少的。习题中算法设计题的选材得当、针对性强，可提高学生的动手能力并培养其良好的程序设计习惯。

本书与配套的辅助教材《数据结构教程习题解析与上机指导》配合使用，将会得到更好的学习效果。此外，书中带“*”的章节为选学内容。

在本书中，作者设计了平衡二叉树算法，真正实现了无须改动二叉树存储结构（不使用平衡因子）就可以对不平衡的子树进行调整。此外，作者还设计了基于二叉树节点存储

结构的哈夫曼树生成算法和哈夫曼编码生成算法，对许多常用的数据结构算法也进行了完善和修改。此外，书中出现的所有算法和涉及算法的示例都在 VC++ 6.0 环境下通过上机验证。

由于编者水平有限，书中难免存在不足，敬请广大读者批评指正。

本书的相关教学资源，可以从华信教育资源网站（http://www.hxedu.com.cn）下载。

编　者

2018 年 5 月

目录

CONTENT

第 1 章

绪论

计算机发展的初期其应用主要是数值计算问题，即所处理的数据都是整型、实型及布尔型等简单数据。但随着计算机应用领域的不断扩大，非数值计算问题占据了目前计算机应用的绝大部分，计算机所处理的数据也不再是简单的数值，而是扩展到字符串、图形、图像、语音、视频等复杂的数据，这些复杂的数据不仅量大，而且具有一定的结构。例如，一幅图像是由简单的数值组成的矩阵；一个图形中的几何坐标可以组成表；语言编译程序中使用的栈、符号表和语法树，操作系统中所用到的队列、树形目录等，这些都是有结构的数据。为了有效地组织和管理好这些数据，设计出高质量的程序及高效率地使用计算机，就必须深入研究这些数据自身的特性及它们之间的相互关系，这正是数据结构这门课程形成与发展的原因。数据结构的研究涉及构筑计算机求解问题过程的两大基石：刻画实际问题中信息及其关系的数据结构和描述问题解决方案的逻辑抽象算法。

数据结构从 1968 年开始成为一门独立的课程，但在此之前与其有关的内容已出现在编译原理和操作系统的课程中。20 世纪 60 年代中期，美国的一些大学已经开设了有关数据结构的课程，但当时的课程名称并不叫数据结构。1968 年 D. E. Knuth 教授开创了数据结构的最初体系，他所著的《计算机程序设计技巧》（*Art of Computer Programming*）第一卷《基本算法》是第一本较系统阐述数据的逻辑结构、存储结构及相应操作的著作。20 世纪 60 年代末到 70 年代初，出现了大型程序并且软件也相对独立，结构程序设计成为程序设计方法学的主要内容，数据结构越来越受到人们的重视。20 世纪 70 年代中期到 80 年代，各种版本的数据结构著作相继问世，这对此后的计算机应用产生了深远影响。至今数据结构的发展并未终结，一方面，面向各种专业领域中特殊问题的数据结构得到研究和发展，如多维图形数据结构等；另一方面，从抽象数据类型和面向对象的观点来讨论数据结构已成为一种新的趋势。面对当今的信息化时代，计算机处理的数据越来越多、越来越复杂，数据结构和算法也越来越受到重视。数据结构和算法的研究也成为面向专业方向的研究，如遗传算法的研究、云计算的并行算法等。

现在，数据结构已是计算机及相关专业必不可少的专业基础课程，主要学习使用计算机实现数据组织和数据处理的方法。对数据结构知识的了解和掌握，将会为学习计算机相关课程（操作系统、编译原理、数据库原理、人工智能和软件工程等）打下坚实的基础。

1.1 数据结构的概念

人们利用计算机的目的是解决实际问题。在明确所要解决问题的基础上，经过对问题的深入分析和抽象，为其在计算机中建立一个模型；然后确定恰当的数据结构表示该模型；在

此基础上设计合适的算法；最后根据设计的数据结构和算法进行相应的程序设计来模拟和解决实际问题。这就是计算机求解问题的一般过程。因此，用计算机解决实际问题的软件开发一般分为下面 4 个步骤。

（1）分析阶段：分析实际问题，从中抽象出一个数学模型。

（2）设计阶段：设计出解决数学模型的算法。

（3）编程阶段：用适当的编程语言编写出实现该算法的可执行程序。

（4）测试阶段：对程序进行测试、修改，直到得到该问题的解答。

数据结构课程集中讨论软件开发过程中的设计阶段，同时涉及分析阶段和编程阶段的若干基本问题。此外，为了构造出好的数据结构及其实现，还需考虑数据结构及其实现的评价与选择。因此，数据结构课程的内容包括如表 1.1 所示的数据表示和数据处理方面所对应的 3 个层次。

表 1.1　数据结构课程内容体系

层　次	方　面	
	数 据 表 示	数 据 处 理
抽象	逻辑结构	基本运算
实现	存储结构	算法
评价	不同数据结构的比较与算法分析	

数据结构的核心内容是分解与抽象。通过对问题的抽象，舍弃数据元素（含义见后面内容）的具体内容，得到逻辑结构。类似地，通过分解将处理要求划分成各种功能，再通过抽象舍弃实现的细节，就得到基本运算的定义。上述两个方面的结合使人们将问题转换为数据结构，这是一个从具体（即具体问题）到抽象（即数据结构）的过程。然后，通过增加对实现细节的考虑，进一步得到存储结构和实现算法，从而完成设计任务，这是一个从抽象（即数据结构）到具体（即具体实现）的过程。熟练掌握这两个过程是数据结构课程在专业技能培养方面的基本目标。

在系统地学习数据结构知识之前，我们先对一些基本概念和术语予以说明。

1.1.1　数据与数据元素

数据是人们利用文字符号、数学符号及其他规定的符号对现实世界的事物及其活动所做的抽象描述。简而言之，数据是信息的载体，是对客观事物的符号化表示。从计算机角度看，数据是计算机程序对所描述客观事物进行加工处理的一种表示，凡是能够被计算机识别、存取和加工处理的符号、字符、图形、图像、声音、视频、信号等都可以称为数据。

我们日常涉及的数据主要分为两类：一类是数值数据，包括整数、实数和复数等，它们主要用于工程和科学计算及商业事务处理；另一类是非数值数据，主要包括字符和字符串及文字、图形、语音等，它们多用于控制、管理和数据处理等领域。

数据元素是数据集合中的一个“个体”，是数据的基本单位。在计算机中，数据元素通常被作为一个整体来进行考虑和处理。在有些情况下，数据元素也称元素、节点（也称结点）、顶点和记录等。一个数据元素可以由一个或多个数据项组成，数据项是具有独立含义的数据最小单位，有时也称字段或域。

例 1.1 一个学生信息（数据）表如表 1.2 所示，请指出表中的数据、数据元素及数据项，并由此得出三者之间的关系。

表 1.2 学生信息数据表

姓　名	性　别	年　龄	专　业	其　他
刘小平	男	21	计算机	…
王　红	女	20	数　学	…
吕　军	男	20	经　济	…
⋮	⋮	⋮	⋮	⋮
马文华	女	19	管　理	…

【解】 表 1.2 构成了全部学生信息的数据。表中的每一行是记录一个学生信息的数据元素，而该行中的每一项则为一个数据项。数据、数据元素和数据项实际上反映了数据组织的三个层次，数据可以由若干数据元素构成，而数据元素又可以由若干数据项构成。

1.1.2 数据结构

数据结构是指数据元素及数据元素之间的相互关系，即数据的组织形式，可以看成是相互之间存在着某种特定关系的数据元素集合。也即，可以把数据结构看成是带结构的数据元素集合。进一步说，数据结构描述按照一定逻辑关系组织起来的待处理数据元素的表示及相关操作，涉及数据的逻辑结构、存储结构和运算。因此，数据结构包含以下三个方面的内容。

（1）数据元素之间的逻辑关系，即数据的逻辑结构。

（2）数据元素及其关系在计算机存储器中的存储方式，即数据的存储结构（物理结构）。

（3）施加在数据上的操作，即数据的运算。

数据的逻辑结构是从逻辑关系上（主要指相邻关系）来描述数据的，它与数据如何存储无关，是独立于计算机的。因此，数据的逻辑结构可以看成是从具体问题中抽象出来的数学模型。

数据的存储结构是指数据的逻辑结构在计算机存储器中的映像表示，即在能够反映数据逻辑关系的前提下数据在存储器中的存储方式。

数据的运算是在数据上所施加的一系列操作，这个过程称为抽象运算，它只考虑这些操作的功能，而暂不考虑如何完成，只有在确定了存储结构后，才会具体实现这些操作。即抽象运算是定义在逻辑结构上的，而实现则是建立在存储结构上的。最常用的运算包括检索、插入、删除、更新及排序等。

1.2 逻辑结构与存储结构

1.2.1 逻辑结构

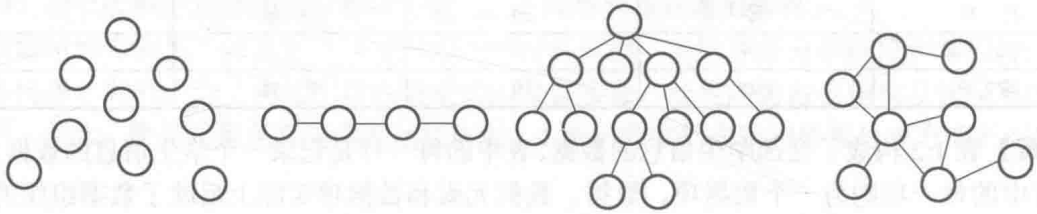
数据的逻辑结构是对数据元素之间逻辑关系的描述，它与数据在计算机中的存储方式无关。根据数据元素之间关系的不同特性，可以划分出以下 4 种基本逻辑结构，如图 1-1 所示。

（1）集合结构：数据元素之间除“属于同一个集合”的联系外，没有其他关系。

（2）线性结构：数据元素之间存在着“一对一”的关系。数据之间存在前后顺序关系，除第一个元素和最后一个元素外，其余元素都有唯一一个前驱元素和唯一一个后继元素。

（3）树形结构：数据元素之间存在着“一对多”的关系。数据之间存在层次关系，除一个根节点（即元素）外，其余元素都有唯一一个前驱元素，并且可以有多个后继元素。

（4）图结构（或称网状结构）：数据元素之间存在着“多对多”的关系，即每个元素都可以有多个前驱元素和多个后继元素。



(a) 集合结构　(b) 线性结构　(c) 树形结构　(d) 图结构

图 1-1　4 种基本逻辑结构

由于集合结构的简单性和松散性，因此通常只讨论其他三种逻辑结构。数据的逻辑结构可以分为线性结构和非线性结构两类，若数据元素之间的逻辑关系可以用一个线性序列简单地表示出来，则称为线性结构；否则称为非线性结构，树形结构和图结构就属于非线性结构。现实生活中的楼层编号属于线性结构，而省、市、地区的划分属于树形结构，城市交通图则属于图结构。

关于逻辑结构需要注意以下三点。

（1）逻辑结构与数据元素本身的形式和内容无关。例如，给表 1.2 中的每个学生增加一个数据项“学号”，就得到另一个数据，但由于所有的数据元素仍是“一个接一个排列的”，因此新数据的逻辑结构与原来数据的逻辑结构相同，仍是一个线性结构。

（2）逻辑结构与数据元素的相对位置无关。例如，将表 1.2 中的学生按年龄由大到小的顺序重新排列就得到另一个表格，但这个新表格中的所有数据元素“一个接一个排列”的性质并没有改变，其逻辑结构与原表格相同，还是线性结构。

（3）逻辑结构与所含数据元素的个数无关。例如，在表 1.2 中增加或删除若干学生信息（数据元素），所得到的表格仍为线性结构。

1.2.2　存储结构

数据的存储结构是数据结构在计算机中的表示方法，即数据的逻辑结构到计算机存储器的映像，包括数据结构中数据元素的表示及数据元素之间关系的表示。数据元素及数据元素之间的关系在计算机中有以下 4 种基本存储结构。

（1）顺序存储结构：借助于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关系。通常顺序存储结构是利用程序语言中的数组来描述的。

（2）链式存储结构：在数据元素上附加指针域，并借助指针来指示数据元素之间的逻辑关系。通常链式存储结构是利用程序语言中的指针类型来描述的。

（3）索引存储结构：在存储所有数据元素信息的同时，建立附加索引表。索引表中表项的一般形式是（关键字，地址）。关键字是数据元素中某个数据项的值，它唯一标识该数据元

素；地址则是指向该数据元素的指针。由关键字可以立即通过地址找到该数据元素。

（4）哈希（或散列）存储结构：此方法的基本思想是根据数据元素的关键字通过哈希（或散列）函数直接计算出该数据元素的存储地址。

顺序存储结构的主要优点是节省存储空间，即分配给数据的存储单元全部用于存放数据元素的数据信息，数据元素之间的逻辑关系没有占用额外的存储空间。采用这种存储结构可以实现对数据元素的随机存取，即每个数据元素对应一个序号，并由该序号可以直接计算出数据元素的存储地址（如对于数组 A 其序号为数组元素的下标，数组元素 A[i]可以通过*(A+i)进行存取）。但顺序存储结构的主要缺点是不便于修改，对数据元素进行插入、删除运算时，可能要移动一系列的数据元素。

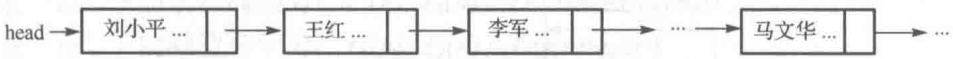
链式存储结构的主要优点是便于修改，在进行插入、删除运算时仅需修改相应数据元素的指针值，而不必移动数据元素。与顺序存储结构相比，链式存储结构的主要缺点是存储空间的利用率较低，因为除用于数据元素的存储空间外，还需要额外的存储空间来存储数据元素之间的逻辑关系。此外，由于逻辑上相邻的数据元素在存储空间中不一定相邻，因此不能对数据元素进行随机存取。

线性结构在采用索引存储方法后就可以对数据元素进行随机访问。在进行插入、删除运算时，只需改动存储在索引表中数据元素的存储地址，而不必移动数据元素，所以仍保持较高的数据修改和运算效率。索引存储结构的缺点是增加了索引表，这样就降低了存储空间的利用率。

哈希（或散列）存储结构的优点是查找速度快，只要给出待查数据元素的关键字，就可以立即计算出该数据元素的存储地址。与前面三种存储方法不同的是，哈希存储结构只存储数据元素的数据而不存储数据元素之间的逻辑关系。哈希存储结构一般只适用于快速查找和插入数据元素的场合。

图 1-2 分别给出了表 1.2 在顺序存储结构下和链式存储结构下的示意。

(a) 表1.2的顺序存储结构示意



(b) 表1.2的链式存储结构示意

图 1-2　表 1.2 在不同的存储结构下的存储示意

1.3 算法与算法分析

概要地说，算法是程序的逻辑抽象，是解决某类客观问题的过程。计算机求解问题的核心是算法设计，而算法设计又高度依赖于数据结构，这是因为在算法设计时必须先确定相应的数据结构，并且在讨论某种数据结构时也必然会涉及相应的算法。

1.3.1 算法的定义与描述

算法是建立在数据结构基础上的对特定问题求解步骤的一种描述，是若干条指令组成的解决问题的有限序列，其中每条指令表示一个或多个操作。算法必须满足以下性质。

（1）有穷性：一个算法必须在有穷步之后结束，即必须在有限时间内完成。

（2）确定性：算法的每一步都必须有确切的含义而没有二义性。对于相同的输入，算法执行的路径是唯一的。

（3）可行性：算法所描述的操作都可以通过可实现的基本运算在有限次执行后得以完成。

（4）输入：一个算法可以有零个或多个输入。

（5）输出：一个算法具有一个或多个输出，且输出与输入之间存在某种特定的关系。

算法的含义与程序十分相似但又有区别。一个程序不一定满足有穷性，例如，对操作系统来说，操作系统程序执行后只要计算机不关机就一直运行下去，永不终止，因此操作系统不是一个算法。此外，程序中的语句最终都要转化（编译）成计算机可执行的指令；而算法中的指令则无此限制，即一个算法可采用自然语言（如英语、汉语）描述，也可以采用图形方式（如流程图、拓扑图）描述。算法给出了对一个问题的求解，而程序则仅是算法在计算机上的实现。一个算法若用程序设计语言来描述，则此时该程序也就是算法。

对某个特定问题的求解究竟采用何种数据结构及选择什么算法，需要考虑问题的具体要求和现实环境的各种条件；数据结构的选择是否恰当将直接影响到算法的效率，只有把数据结构与算法有机地结合起来才能设计出高质量的程序来。

例 1.2 对两个正整数 m 和 n，给出求它们最大公因数的算法。

【解】 此题也称欧几里得算法或辗转相除法，算法设计如下。

（1）求余数：用 n 除 m，余数为 r 且 $0 \leqslant r < n$。

（2）判断余数 r 是否等于零：若 r 为零，则输出 n 的当前值（即为最大公因数），算法结束；否则执行（3）。

（3）将 n 值传给 m，将 r 值传给 n，转到（1）。

也可以用流程图描述该算法，如图 1-3 所示。

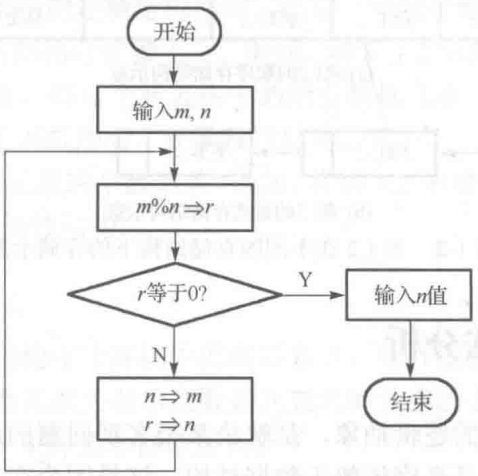


图 1-3 求最大公约数的算法流程图

上述算法给出了三个计算步骤，而且每个步骤意义明确并切实可行。虽然出现了循环，但 m 和 n 都是已给定的有限整数，并且每次 m 除以 n 后得到的余数 r 即使不为零也总有 $r < \min(m,n)$，这就保证循环执行有限次后必然终止，即满足算法的所有特征，所以该算法是一个正确的算法。

1.3.2 算法分析与复杂度计算

算法设计主要考虑可解算法的设计，而算法分析则研究和比较各种算法的性能与优劣。算法的时间复杂度和空间复杂度是算法分析的两个主要方面，其目的主要是考察算法的时间效率和空间效率，以求改进算法或对不同的算法进行比较。

（1）时间复杂度：一个程序的时间复杂度是指程序运行从开始到结束所需要的时间。

（2）空间复杂度：一个程序的空间复杂度是指程序运行从开始到结束所需的存储量。

在复杂度计算中，实际上是把求解问题的关键操作，如加法、减法和比较运算指定为基本操作，然后把算法执行基本操作的次数作为算法的时间复杂度，而算法执行期间占用存储单元的数量作为算法的空间复杂度。

在此，涉及频度的概念，即语句（指令）的频度是指它在算法中被重复执行的次数。一个算法的时间耗费就是该算法中所有语句（指令）的频度之和（记作 $T(n)$），它是该算法所求解问题规模 n 的某个函数 $f(n)$。当问题规模 n 趋向无穷大时，$T(n)$的数量级称为时间复杂度，记作

$$T(n) = O(f(n))$$

上式中“O”的文字含义是 $T(n)$的数量级，其严格的数学定义是：若 $T(n)$和 $f(n)$是定义在正整数集合上的两个函数，则存在正常数 C 和 n_0，使得当 $n \geq n_0$ 时满足

$$0 \leqslant T(n) \leqslant C \cdot f(n)$$

例如，若一个程序的实际执行时间为 $T(n)=2.7n^3+8.3n^2+5.6$，则 $T(n) = O(n^3)$。当 n 趋于无穷大时，n^3 前的 2.7 可以忽略，即该程序的时间复杂度的数量级是 n^3。

算法的时间复杂度在采用这种数量级的形式表示后，将给分析算法的时间复杂度带来很大的方便；即对一个算法，只需分析影响该算法时间复杂度的主要部分即可，而无须对该算法的每一个语句都进行详细的分析。

若一个算法中的两个部分其时间复杂度分别为 $T_1(n) = O(f(n))$和 $T_2(n) = O(g(n))$，则：

（1）在“O”下的求和准则为：$T_1(n)+T_2(n) = O(\max(f(n),g(n)))$

（2）在“O”下的乘法准则为：$T_1(n)\times T_2(n) = O(f(n)\times g(n))$

当算法转换为程序后，每条语句执行一次所需的时间取决于机器的指令性能、速度以及编译所生成的代码质量，这是难以确定的。因此，我们假设每条语句执行一次所需的时间均是单位时间，则程序计算的时间复杂度法如下：

（1）执行一条读写语句或赋值语句所用的时间为 $O(1)$；

（2）依次执行一系列语句所用的时间采用求和准则；

（3）条件语句 if 的耗时主要是当条件为真时执行语句体所用的时间，而检测条件是否为真还需耗费 $O(1)$；

（4）对 while、do-while 和 for 这样的循环语句，其运行时间为每次执行循环体及检测是否继续循环的时间，故常用乘法准则。

例 1.3 试求下面程序段的时间复杂度。

```
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
```