江苏省精品课程主讲教材

数据结构

（C++语言描述）

Data Structures in C++

吉根林　陈　波　主　编
王　琼　周俊生　于　泠　编　著

高等教育出版社

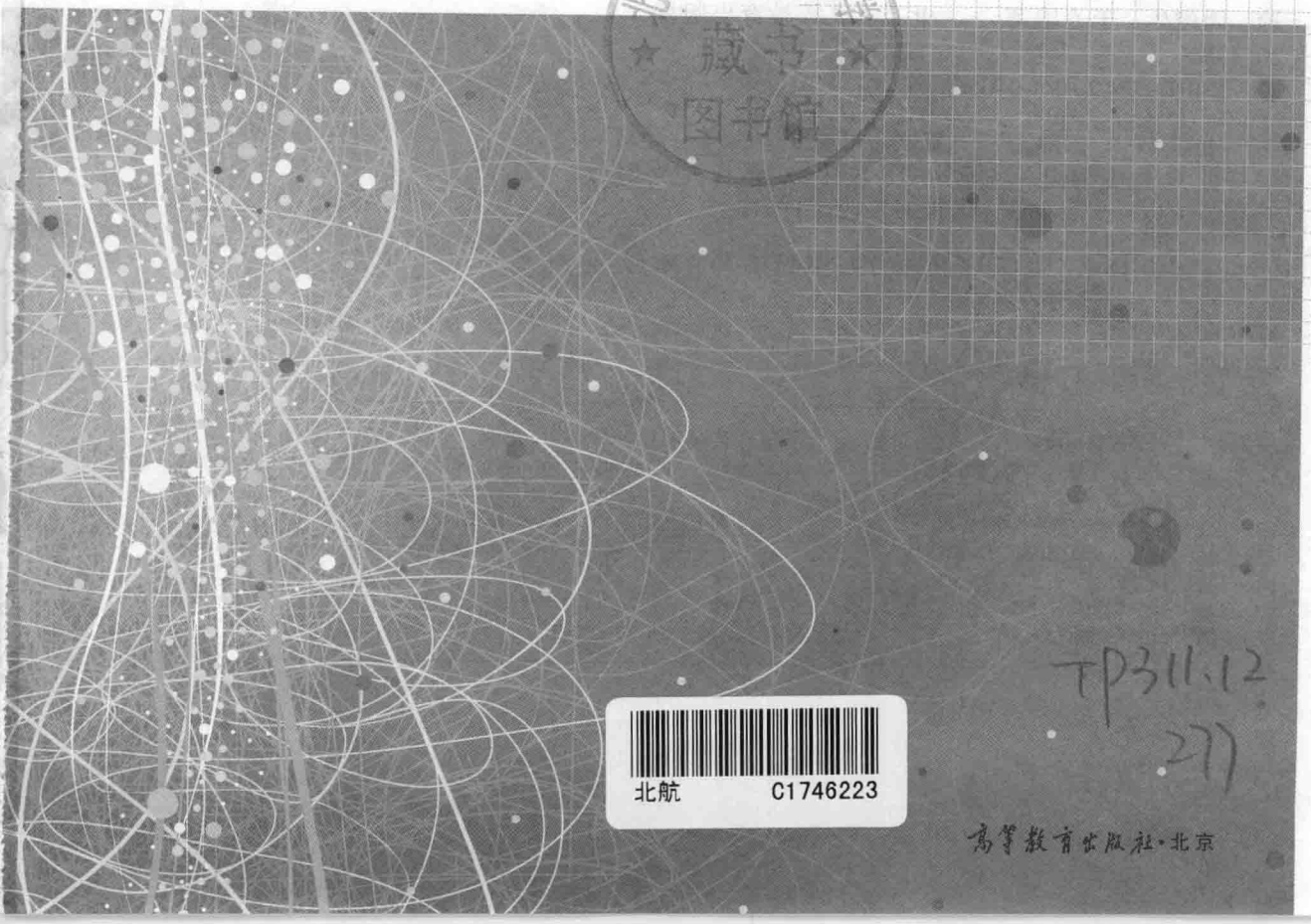
江苏省精品课程主讲教材

数据结构

（C++语言描述）

Data Structures in C++

吉根林　陈　波　主　编
王　琼　周俊生　于　泠　编　著



高等教育出版社·北京

内容提要

本书是江苏省精品课程“数据结构”的建设成果。全书共10章，介绍了各种常用的数据结构（线性表、栈和队列、串、数组和特殊矩阵、广义表、树和二叉树、图）的基本概念、逻辑关系、存储结构、操作运算及其实现算法；阐述了各种常用的查找算法和排序算法，并对各种算法的性能进行分析。书中使用C++类定义各种数据结构，利用C++伪代码描述算法，并给出了许多经典算法和典型题例。每章均附有小结、习题和上机实验题，附录给出了6套课程考试样卷和5道课程设计题。

本书既注重基本原理，又重视算法实现；既体现先进性，又强调实用性；内容丰富，重点突出，条理清晰，由浅入深，语言流畅，具有特色。本书的PPT课件和相关教学资源可从江苏省精品课程和南京师范大学精品课程“数据结构”网站 http://computer.njnu.edu.cn/datastructure/index.asp 下载。

本书可作为高等学校计算机类专业及相关专业“数据结构”课程的教材，也可供从事计算机软件开发人员参考。

图书在版编目（CIP）数据

数据结构：C++语言描述／吉根林，陈波主编；王琼，周俊生，于泠编著．--北京：高等教育出版社，2014.8

ISBN 978-7-04-040560-6

Ⅰ.①数… Ⅱ.①吉… ②陈… ③王… ④周… ⑤于… Ⅲ.①数据结构－高等学校－教材②C语言－程序设计－高等学校－教材 Ⅳ.①TP311.12②TP312

中国版本图书馆CIP数据核字(2014)第161846号

策划编辑	倪文慧	责任编辑	倪文慧	封面设计	王 洋	版式设计	杜微言
插图绘制	杜晓丹	责任校对	胡美萍	责任印制	尤 静		

出版发行	高等教育出版社	咨询电话	400－810－0598
社 址	北京市西城区德外大街4号	网 址	http://www.hep.edu.cn
邮政编码	100120		http://www.hep.com.cn
印 刷	三河市华润印刷有限公司	网上订购	http://www.landraco.com
开 本	850mm×1168mm 1/16		http://www.landraco.com.cn
印 张	18.25	版 次	2014年8月第1版
字 数	390千字	印 次	2014年8月第1次印刷
购书热线	010－58581118	定 价	27.00元

物 料 号 40560-00

前　言

“数据结构”是计算机类专业及相关专业的核心课程，主要研究和分析计算机存储、组织数据的方式和相关操作运算算法。通过本课程学习，学生不仅应当掌握数据结构和算法的基本概念和技术，掌握线性表、栈、队列、串、数组和特殊矩阵、广义表、树和二叉树、图等常用数据结构及相关算法，以及排序、查找等重要技术，而且能够针对应用问题选择合适的数据结构，并设计相应的操作运算算法。

南京师范大学的“数据结构”课程经过多年建设与探索，对教学内容与教学方法进行了改革与实践，有效地提高了教学质量，并被评为江苏省精品课程。课程组在教学中贯彻下列指导思想：

（1）基础性。数据结构、算法和程序设计是计算机科学的核心知识，本课程应为学生的软件开发能力培养打下扎实的基础。

（2）系统性。本课程以系统的观点研究数据组织和操作算法，重点在抽象思维、算法设计等方面加强学生的能力培养。

（3）先进性。由于本课程的新思想和新方法不断产生，因此需要不断更新教学内容以拓宽学生的知识面，适应计算机应用和发展的需要。

（4）实践性。本课程是一门实践性很强的课程，在课程实验中不仅要进行验证性实验，训练程序设计技能和操作能力，更应包括设计算法的创新性实验能力。

本书集作者多年讲授“数据结构”课程的教学经验，体现了科学性、先进性和实用性的原则，既注重基本原理又重视算法实现，力求内容丰富、重点突出、条理清晰、由浅入深、语言流畅、具有特色。全书使用C++类定义各种数据结构，利用C++伪代码描述算法，并给出了许多经典算法和典型题例。每章均附有小结、习题和上机实验题；附录给出课程考试样卷和课程设计题，以供教学参考。

本书共分10章。第1章简要介绍数据结构和算法的基本概念；第2~5章介绍线性结构及其算法，包括线性表、栈、队列、串、数组和特殊矩阵；第6~8章介绍非线性结构及其算法，包括广义表、树、二叉树和图；第9章介绍各种常用的查找算法；第10章介

绍各种常用的排序算法。本书的PPT课件和相关教学资源可从江苏省精品课程和南京师范大学精品课程“数据结构”网站 http://computer.njnu.edu.cn/datastructure/index.asp 下载。

本书由南京师范大学“数据结构”课程组编写，其中，第1章和第10章由吉根林编写，第2章和第3章由陈波编写，第6章和第7章由王琼编写，第5章和第8章由周俊生编写，第4章和第9章由于泠编写。全书由吉根林和陈波担任主编，并负责统稿、修改和定稿。

由于作者水平有限，书中难免存在不妥之处，敬请读者批评指正。

编者 E-mail：glji@njnu.edu.cn

编者

2014年6月

目　录

第1章 绪 论

“数据结构”是计算机科学与技术、软件工程、信息安全、信息管理等专业的重要核心课程，主要分析计算机中数据的组织方式、存储结构和处理方法。数据结构课程的学习将为计算机及相关专业的后读课程（如操作系统、编译原理、数据库原理、软件工程等）的学习打下基础。实际上，要编写出“好”的程序，需要选择合理的数据结构和好的算法，而“好”算法的选择在很大程度上取决于描述实际问题所采用的数据结构。因此，要编写出“好”的程序，仅学习程序设计语言是不够的，还必须很好地掌握数据结构的基本知识和基本技能。本章将概要地介绍数据结构课程的研究内容、基本概念和基本思想。

1.1 数据结构课程的研究内容

数据结构起源于程序设计。随着计算机科学技术的发展，计算机应用领域不再局限于科学计算，而是更多地应用于信息处理、智能控制、办公自动化等领域。不仅计算机处理的对象由数值发展到字符串、表格、图形、图像、声音等数据，而且处理的数据量也越来越大。在程序设计中，应如何来组织和处理这样的数据呢？这正是“数据结构”课程需要研究的问题。

在使用计算机解决问题时，一般要经过下面几个步骤。首先，要将实际问题抽象出数学模型；然后，要针对数学模型设计出求解算法；最后，要编写程序并上机调试，直到求出最终结果。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。然而，对于非数值计算问题，如图书资料的检索、人－机博弈、课程表编排、最短路径求解等问题，它们的数学模型无法用数学方程或数学公式来描述，而是要使用线性表、树、图等数据结构来描述，并且要对这些模型设计相应算法来求解。数据结构就是研究计算机在解决非数值计算问题中使用的数据对象以及它们之间的关系和操作算法的学科，具体主要包含三个方面的内容：数据的逻辑结构、数据的存储结构和数据的操作算法。

数据结构作为一门独立的课程是1968年首先在美国开设的。在这之前，它的某些内容在其他课程中已有涉及。1968年，Standford大学的D. E. Knuth教授建立了数据结构的最初体系，他所著的《计算机程序设计艺术：第1卷 基本算法》是一本较系统地阐述数据的逻辑结构、存储结构及其操作算法的著作。后来不同语言描述的数据结构图书相继出版，如有PASCAL语言、C语言、C++语言、Java语言等。我国于20世纪80年代初开始开设“数据结构”课程，该课程不仅是计算机专业核心课程，而且是其他信息类专业的必修课。

“数据结构”课程的内容随着程序设计技术的发展而发展，经历了结构化阶段和面向对象阶段。20世纪60—80年代，计算机开始广泛应用于非数值计算领域，数据组织成为程序设计的重要问题。人们认识到程序设计规范的重要性，提出了结构化程序设计的思想。数据结构概念的引入对程序设计的规范化起到了重要作用。图灵奖获得者瑞士计算机科学家N. Wirth教授曾提出“程序=数据结构+算法”。由此可以看出，数据结构和算法是构成程序的两个重要组成部分。

20世纪90年代以来，面向对象技术成为最流行的程序设计技术。在面向对象技术中，现实世界的实体被看作是一个对象。对象由属性和方法构成，属性描述实体的状态和特征，方法用以改变实体的状态或行为。一组具有相同属性和方法的对象集合称为类，每个具体的对象称为类的一个实例。例如，“学生”是一个类，“张丽”、“李明”等对象都是“学生”类的实例。

数据结构主要强调两个方面的内容：数据之间的关系，即数据之间的逻辑结构和存储结构；针对这些关系的基本操作。这两个方面实际上蕴含着面向对象的思想：类描述实体的属性和行为，而数据结构描述数据之间的关系及其基本操作。类与数据结构之间的对应关系如图1-1所示。

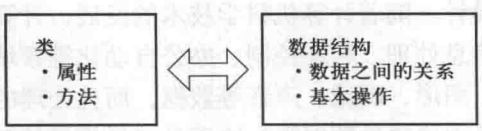


图1-1 类与数据结构之间的对应关系

值得一提的是，数据结构的发展并未终结。一方面，数据结构将继续随着程序设计技术的发展而发展；另一方面，面向专门领域的数据结构得到研究和重视，如研究人员提出了一些空间数据结构。

1.2 基本概念及术语

本节将介绍数据结构相关的基本概念及术语。

1. 数据

数据（Data）是信息的载体，是指所有能输入到计算机中并能被计算机程序识别和处理的符号集合。数据可以分为两大类：一类是整数、实数等数值数据，另一类是图形、图像、声音、文字等非数值数据。

2. 数据元素

数据元素（Data Element）是组成数据的基本单位，在计算机程序中通常作为一个整体进

行考虑和处理。构成数据元素的不可分割的最小单位称为数据项。例如，对于学生档案登记表，每个学生的档案是一个数据元素，而档案中的学号、姓名、出生日期等是数据项。数据元素是讨论数据时涉及的最小数据单位，其中的数据项一般不予考虑。

数据元素具有广泛的含义。一般来说，能独立、完整地描述问题世界的一切实体都是数据元素。例如，对弈中的棋盘格局、教学计划中的某门课程、一年中的4个季节，甚至一次学术报告、一场足球比赛都是数据元素。数据元素又称为元素、结点、顶点或记录。

3. 数据对象

数据对象（Data Object）是具有相同性质的数据元素的集合，是数据的子集。在实际应用中处理的数据元素通常具有相同性质。例如，学生档案登记表中每个数据元素具有相同数目和类型的数据项，所有数据元素（学生的档案）的集合就构成了一个数据对象。又如，字母数据集合 $M=\{'A','B',\cdots,'Z'\}$ 也是一个数据对象。

4. 数据结构

数据结构（Data Structure）是指数据元素及其相互关系的集合。这种相互关系即数据的组织形式，可分为数据的逻辑结构和数据的存储结构。

数据的逻辑结构（Logical Structure）是指数据元素之间的逻辑关系，即数据元素之间的关联方式或邻接关系。数据的逻辑结构分为以下4类：

（1）集合。数据元素之间的关系是属于同一个集合，除此之外，没有任何关系。

（2）线性结构。数据元素之间存在着一对一的线性关系。

（3）树结构。数据元素之间存在着一对多的层次关系。

（4）图结构。数据元素之间存在着多对多的任意关系。

树结构和图结构也称为非线性结构。

数据的逻辑结构常用逻辑结构图来描述，其描述方法是：将每一个数据元素看做一个结点，用圆圈表示；元素之间的逻辑关系用结点之间的连线表示。如果强调关系的方向性，则用带箭头的连线表示关系。图1-2描述了4种基本的数据逻辑结构。

为了更确切地描述一种数据结构，通常采用二元组形式化定义数据结构：

$$\text{Data\_Structure}=(D,R)$$

其中，D 是数据元素的有限集合，R 是 D 上关系的有限集合。

例如，有一种数据结构 $T=(D,R)$，其中：

$D=\{a,b,c,d,e,f,g,h,i,j\}$

$R=\{(a,b),(a,c),(a,d),(b,e),(c,f),(c,g),(d,h),(d,i),(d,j)\}$

显然，数据结构 T 是一棵树形结构。

数据的存储结构（Storage Structure）又称物理结构，是数据及其逻辑结构在计算机中的表示。换言之，存储结构除了存储数据元素之外，还隐式或显示地存储数据元素之间的逻辑关

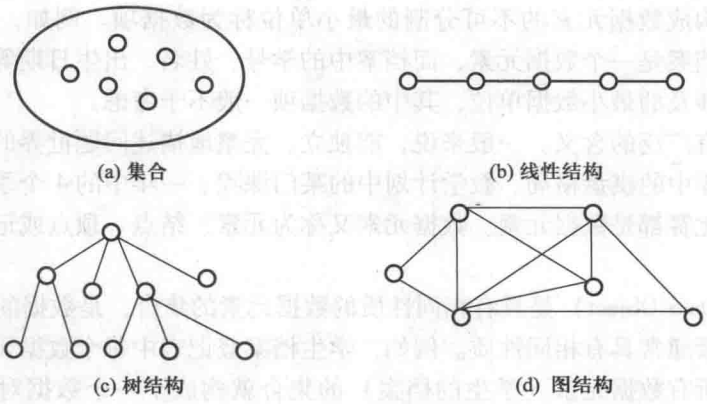


图 1-2　4 种基本的数据逻辑结构

系。通常有两种存储结构：顺序存储结构和链接存储结构。

顺序存储结构的基本思想是用一组连续的存储单元存储数据元素，数据元素之间的逻辑关系是由元素的存储位置来表示的。例如，线性表(a, b, c)的顺序存储示意图如图 1-3 所示。

链接存储结构的基本思想是用一组任意的存储单元存储数据元素，数据元素之间的逻辑关系是用指针来表示的。例如，线性表(a, b, c)的链接存储示意图如图 1-4 所示。

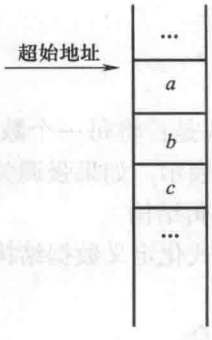


图 1-3　线性表的顺序存储示意图

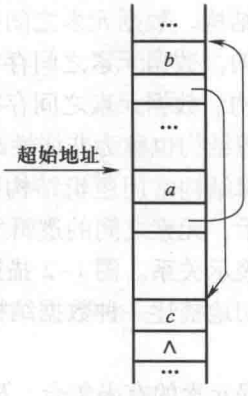


图 1-4　线性表的链接存储示意图

数据的逻辑结构和存储结构是密切相关的两个方面。一般来说，一种数据的逻辑结构可以用多种存储结构来存储；而采用不同的存储结构，其数据处理的效率往往是不同的。

5. 数据类型

数据类型（Data Type）是一组值的集合以及定义于这个值集上的一组操作的总称。每种程序设计语言都定义了自己的数据类型，如整型、实型、字符型、指针、数组、结构体、类

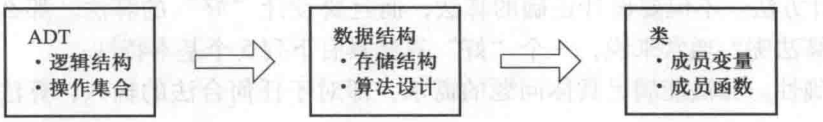
等。数据类型规定了该类型数据的取值范围和对这些数据所能采取的操作。例如，C++中的整型变量可以取的值是机器所能表示的最小负整数和最大正整数之间的任何一个整数，允许的操作有+、-、*、/、%、<、<=、>、>=、==、!=等。

6. 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是一种数据结构以及定义在该结构上的一组操作的总称，可被理解为对数据类型的进一步抽象。数据类型和ADT的区别在于：数据类型是指高级程序设计语言支持的基本数据类型，而ADT是指自定义的数据类型。例如，本课程将要学习的表、栈、队列、树、图等结构就是一些不同的ADT。

ADT包括定义和实现两个方面。其中，定义是独立于实现的，仅给出ADT的逻辑特征，而不必考虑如何在计算机中实现。ADT的特征是将使用与实现分离，从而实现封装和信息隐藏。例如，整数的数学概念和施加到整数的运算构成一个ADT，C++的变量类型int是对这个ADT的物理实现。各种程序设计语言都有整数类型。尽管它们在不同处理器上实现的方法不同，但由于其ADT相同，因此在用户看来都是相同的。

在设计ADT时，需要把ADT的定义和实现分开来，定义部分只包含数据逻辑结构的定义和所允许的操作集合。一方面，使用者依据这些定义来使用ADT，即通过操作集合对该ADT进行操作；另一方面，ADT的实现者依据这些定义来完成该ADT的各种操作的具体实现。图1-5所示的是ADT的不同视图。



(a) 使用视图：ADT的定义　(b) 设计视图：ADT的设计　(c) 实现视图：ADT的实现

图1-5　ADT的不同视图

C++中的类体现了抽象数据类型的思想。在ADT中定义的每个操作由类中的成员函数来实现，数据以及数据之间的逻辑关系由类中的成员变量来实现。

1.3　算法与算法分析

1.3.1　算法

1. 什么是算法

算法（Algorithm）是计算机求解特定问题的方法和步骤，是指令的有限序列。通常一个问题可以有多种算法，一个给定算法解决一个特定的问题。

算法具有下列5个重要特性：

（1）输入。一个算法有零个或多个输入（即算法可以无输入），这些输入通常取自于某个特定的对象集合。

（2）输出。一个算法有一个或多个输出（即算法必须要有输出），通常输出与输入之间有着某种特定的关系。

（3）有穷性。对任何合法的输入，一个算法必须在执行有穷步之后结束，且每一步都在有穷时间内完成。

（4）确定性。算法中的每一条指令都必须有确切的含义，即不存在二义性。而且，在任何条件下，给定的算法对于相同的输入只能得到相同的输出。

（5）可行性。算法描述的操作可以通过已经实现的基本操作执行有限次来实现。

程序（Program）是对一个算法使用某种程序设计语言的具体实现，原则上任一算法可以用任何一种程序设计语言实现。算法的有穷性意味着不是所有的计算机程序都是算法。例如，操作系统是一个在无限循环中执行的程序，而不是一个算法。但是可以把操作系统的各个任务看成是一个单独的问题，每个问题由操作系统中的一个子程序通过特定的算法来实现，得到输出结果后便终止。

2. 算法的评价

数据结构与算法之间存在着本质联系。本课程学习的目的就是要在某一种数据结构基础上学习算法设计方法。不但要设计正确的算法，而且要设计“好”的算法。那么什么样的算法是“好”的算法呢？通常来说，一个“好”算法具有下列5个基本特性：

（1）正确性。算法能满足具体问题的需求，即对于任何合法的输入，算法都会得出正确的结果。

（2）健壮性（鲁棒性）。算法对非法输入的抵抗能力，即对于错误的输入，算法应能识别并做出处理，而不是产生错误动作或陷入瘫痪。

（3）可读性。好的算法应该便于人们理解和相互交流。可读性好的算法有助于人们对算法的理解；反之，难懂的算法易于隐藏错误且难于调试和修改。

（4）高效率。算法的效率通常是指算法的执行时间。对于同一个问题，如果有多个算法可以使用，那么执行时间短的算法效率高。

（5）低存储空间需求。算法需要的存储空间是指算法在执行过程中所需要的最大存储空间，它与问题规模有关。一个“好”算法应该占用较少的辅助空间。

3. 算法的描述方法

设计好了一个算法之后，必须清楚、准确地将所设计的求解步骤表达出来，即描述算法。算法描述方法通常有自然语言、流程图、伪代码和程序设计语言等。下面以欧几里得算法（用辗转相除法求两个自然数 m 和 n 的最大公约数，并假设 $m \geqslant n$）为例，介绍算法的描述

方法。

(1) 自然语言

用自然语言描述算法的最大优点是容易理解，缺点是容易出现二义性，且算法描述通常都很冗长。欧几里得算法用自然语言描述如下：

① 输入 m 和 n。

② 求 m 除以 n 的余数 r。

③ 若 r 等于0，则 n 为最大公约数，算法结束；否则执行第④步。

④ 将 n 的值放在 m 中，将 r 的值放在 n 中。

⑤ 重新执行第②步。

(2) 流程图

用流程图描述算法的优点是直观易懂，缺点是严密性不如程序设计语言，且灵活性不如自然语言。欧几里得算法的流程图如图 1-6 所示。

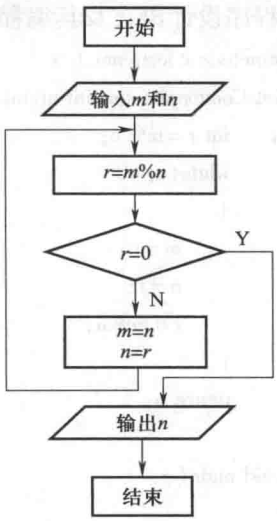


图 1-6 欧几里得算法的流程图

在计算机应用的早期，使用流程图描述算法占有统治地位。但是实践证明，除了一些非常简单的算法以外，这种描述方法使用起来非常不方便。

(3) 伪代码

伪代码是介于自然语言和程序设计语言之间的算法描述方法。计算机科学家对伪代码的书写形式没有做出严格的规定，它采用某一种程序设计语言的基本语法，操作指令可以结合自然语言来设计。伪代码的算法描述中自然语言的成分有多少，取决于算法的抽象级别。抽象级别高的伪代码中自然语言多一些，抽象级别低的伪代码中程序设计语言的语句多一些。只要具备程序设计语言基础的人都能阅读伪代码。用 C ++ 伪代码描述的欧几里得算法如下：

```
int CommonFactor(int m,int n)
{   r = m% n;
    while(r!=0)
    {
        m = n;
        n = r;
        r = m% n;
    }
    return n;
}
```

虽然伪代码不是一种实际的编程语言，但是由于它在表达能力上类似于编程语言，同时极小化了描述算法的不必要的技术细节，因此是比较合适的描述算法的方法，被称为算法语言。

本教材采用基于C++语言的伪代码来描述算法，以使得算法的描述简明清晰，既不拘泥于C++语言的实现细节，又容易转换为C++程序。

(4) 程序设计语言

用程序设计语言描述的算法能由计算机直接执行，但缺点是抽象性差，使算法设计者拘泥于描述算法的具体细节，忽略了“好”算法和正确逻辑的重要性；此外，还要求算法设计者掌握程序设计语言及其编程技巧。欧几里得算法用C++语言书写的程序如下：

```
#include <iostream.h>
int CommonFactor(int m,int n)
{   int r=m%n;
    while(r!=0)
    {
        m=n;
        n=r;
        r=m%n;
    }
    return n;
}
void main()
{
    cout<<CommonFactor(63,54)<<endl;
}
```

1.3.2 算法分析

一种数据结构的优劣是由实现其各种操作的算法决定的，对数据结构的分析实质上就是对实现各种操作的算法进行分析。分析时除了要验证算法是否能够正确解决问题外，还需要对算法的效率进行评价。对于一个实际问题的解决可以提出若干算法，那么如何从这些可行的算法中找出最有效的算法呢？或者有了一个解决实际问题的算法，如何来分析它的性能呢？这些问题都需要通过算法分析来确定。通常来说，算法分析主要分析算法的时间代价和空间代价这两个主要指标。

可能有人会认为，随着计算机功能的日益强大，程序的运行效率变得越来越不那么重要了。然而，计算机功能越强大，人们就越想去尝试解决更复杂的问题，而更复杂的问题就需要更大的计算量。实际上，我们不仅需要算法，而且需要“好”的算法。以破解密码的算法为