21世纪高等学校计算机
基础实用规划教材

数据结构(C语言版)
(第5版)

◎ 邓文华 主编　谢胜利 副主编

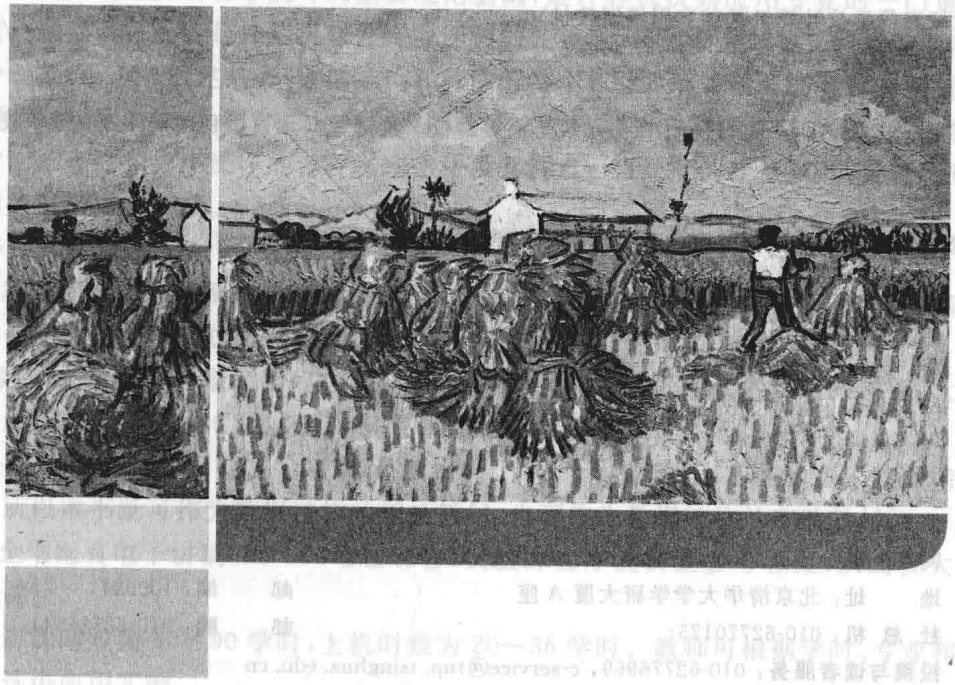
- 教材＋练习册＋上机指导
- **53**个实例，**10**道上机实验题，**306**道复习题

清华大学出版社

21世纪高等学校计算机
基础实用规划教材

数据结构(C语言版)
(第5版)

◎ 邓文华 主编　谢胜利 副主编



清华大学出版社
北京

内容简介

本书对常用的数据结构作了系统的介绍，概念论述清晰，注重实际应用。全书共分9章，依次介绍了数据结构的基本概念、线性表、栈和队列、串和数组、树和二叉树、图结构以及查找和排序等基本运算，第9章设计了10个实验，基本涵盖了主要的数据结构与算法。全书用C语言作为算法描述语言，每一章均附有典型例题与小结，便于总结与提高。

本书叙述简洁，深入浅出，注重实践和应用，一书两用（理论与实验），主要面向应用型大学计算机类专业的学生，也可以作为大学非计算机专业学生的选修课教材和计算机应用技术人员的自学参考书。

图书在版编目(CIP)数据

数据结构：C语言版/邓文华主编. —5版. —北京：清华大学出版社，2018（2018.9重印）
（21世纪高等学校计算机基础实用规划教材）
ISBN 978-7-302-49109-5

Ⅰ. ①数… Ⅱ. ①邓… 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. ①TP311.12 ②TP312.8

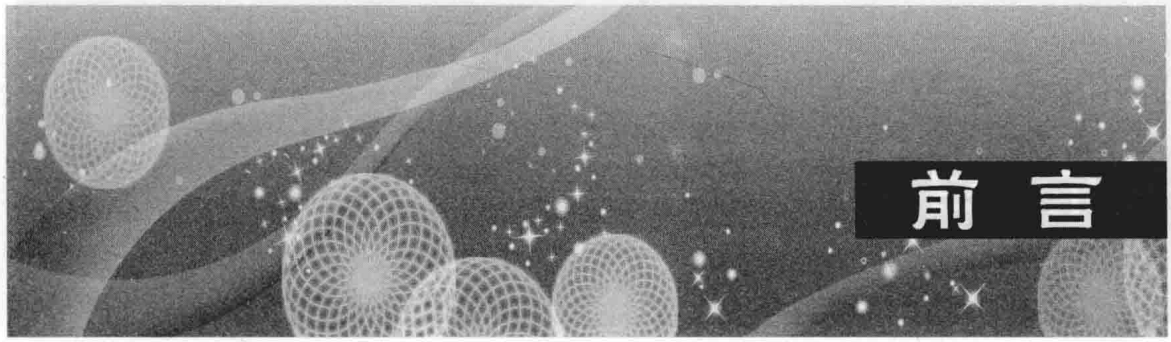
中国版本图书馆CIP数据核字(2017)第301369号

责任编辑：魏江江　王冰飞
封面设计：刘　键
责任校对：白　蕾
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954

印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16.25　　**字　　数**：416千字
版　　次：2004年8月第1版　2018年2月第5版　　**印　　次**：2018年9月第3次印刷
印　　数：3001～6000
定　　价：39.50元

产品编号：077761-01



前言

本书以第4版为基础，在广泛听取广大读者的意见与建议的情况下修改而成。本版主要对第4版中的一些描述、笔误、解释等进行了订正，取消了一些过时的内容，简化了使用不多的内容，删除了第9章“综合应用实例”，增加了新的第9章“实验”。

“数据结构”是计算机程序设计的重要理论基础，是计算机及其应用专业的一门重要基础课程和核心课程。它不仅是计算机软件专业课程的先导，而且逐渐为其他工科类专业所重视。全书共分9章，第1章叙述数据、数据结构和算法等基本概念；第2章至第6章分别讨论了线性表、栈和队列、串和数组、树和二叉树以及图等基本数据结构的实现及其应用；第7章和第8章分别讨论了查找和排序操作的各种实现方法及其特点；第9章设计了10个实验，基本涵盖了主要的数据结构与算法。

本书有以下特点：

(1) 基础理论知识的阐述由浅入深、通俗易懂。内容组织和编排以应用为主线，略去了一些理论推导和数学证明的过程，淡化算法的设计分析和复杂的时空分析。

(2) 除第1章和第9章外都配有相应的典型例题，列举分析了众多实用的例子，大多数算法都直接给出了其相应的C语言设计程序，以便学生上机练习、实践。

(3) 一书两用，本书配有实验一章，该章设计了10个实验项目，基本涵盖了数据结构的主要内容。所以本书既可作为数据结构的理论教材，也可作为数据结构的实验教材。

(4) 全书配有电子讲稿(PPT)、习题解答、实验源程序及实验参考答案，从而极大地方便了教师备课。

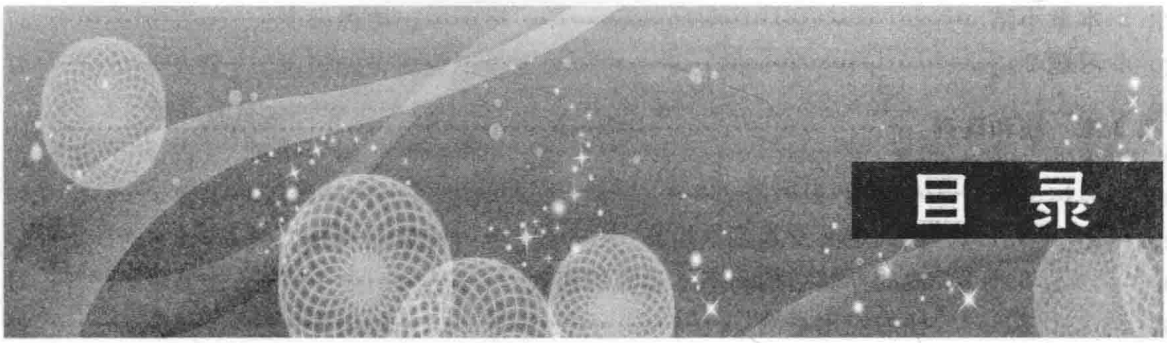
本书讲课时数为60～90学时，上机时数为20～36学时。教师可根据学时、专业和学生的实际情况选讲应用实例。

本书由邓文华任主编，谢胜利任副主编。其中，第5～9章由邓文华修改、编写；第1～4章由谢胜利修改、编写。其他执笔者还有谢翠华、邹华胜、毕保祥、戴大蒙、赵丽央、刘文斌、邓泽川、李元华。全书由邓文华统稿。

由于编者水平有限，书中难免存在不妥之处，敬请读者赐教指正。

编　者

2018年1月



目 录

第1章 绪 论

数据作为计算机加工处理的对象，如何在计算机中表示、存储数据是计算机科学研究的主要内容之一，更是计算机技术需要解决的关键问题之一。数据是计算机化的信息，是计算机处理的主要对象。科学计算、数据处理、过程控制、文件存储、数据库技术等都是对数据进行加工处理的过程。因此，要设计出一个结构好、效率高的程序，必须研究数据的特性、数据间的相互关系及其对应的存储表示，并利用这些特性和关系设计相应的算法和程序。

1.1 从问题到程序

“数据结构”作为计算机科学与技术专业的专业基础课，是十分重要的核心课程，其主要研究内容是数据之间的逻辑关系和物理实现，即探索有利的数据组织形式及存取方式。计算机系统软件和应用软件的设计、开发要用到各种类型的数据结构。因此，要想更好地运用计算机来解决实际问题，仅仅依赖几种计算机程序设计语言是不够的，还必须学习和掌握数据结构的有关知识。

在计算机发展初期，人们使用计算机的目的主要是处理数值计算问题。当我们使用计算机来解决一个具体问题时，一般需要经过几个步骤：首先要从该具体问题抽象出一个适当的数学模型，然后设计或选择一个解此数学模型的算法，再编写程序并进行调试、测试，最后运行程序得到答案(如图 1.1 所示)。例如，求解梁架结构中应力数学模型的线性方程组，该方程组可以使用迭代算法来求解。

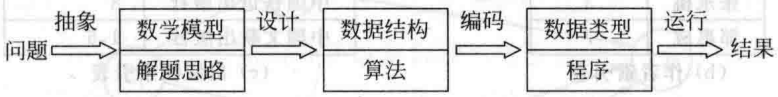


图 1.1 计算机解决问题的一般过程

由于当时涉及的运算对象是简单的整型、实型或布尔类型数据，所以，程序设计者主要将精力集中于程序设计的技巧上，而无须重视数据结构。随着计算机应用领域的扩大和软、硬件的发展，非数值计算问题显得越来越重要。据统计，当今处理非数值计算问题占用了90%以上的机器时间。这类问题涉及的处理对象不再是简单的数据类型，其形式更加多样，结构更为复杂，数据元素之间的相互关系一般无法直接用数学方程式加以描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是要设计出合适的数据结构，以便有效地解决问题。

例 1.1 图书信息检索系统。在现代图书馆中，往往借助计算机图书检索系统来查找需要的图书信息，或者直接通过图书馆信息系统进行图书的借阅。为此，需要将图书信息按不同

分类进行编排,建立合适的数据结构进行存储和管理,并按照某种算法编写相关程序实现计算机的自动检索。由此,一个简单的图书信息检索系统包括一张按图书分类号和登录号顺序排列的图书信息表,以及分别按作者、出版社等顺序排列的各类索引表,如图 1.2 所示。由这 3 张表构成的文件便是图书信息检索的数学模型,计算机的主要操作是按照用户的要求(例如给定作者)通过不同的索引表对图书信息进行检索、查询。

序号	图书分类号	登录号	书 名	作 者	出版社
1	B259.1	3240	梁启超家书	张品兴	中国文联出版社
2	C52	5231	探寻语碎	李泽厚	上海文艺出版社
3	D035.5	6712	市政学	张永桃	高等教育出版社
4	G206	1422	传播学	邵培仁	高等教育出版社
5	H319.4	1008	英语阅读策略	李宗宏	兰州大学出版社
6	K825.4.00	5819	围棋人生	聂卫平	中国文联出版社
7	P1.00	8810	通向太空之路	邹惠成	科学出版社
8	TN915	7911	通信与网络技术概论	刘云	中国铁道出版社
9	TP312	7623	计算机软件技术基础	王宇川	科学出版社
10	TP393.07	8001	网络管理与应用	张琳	人民邮电出版社
11	Q3.00	2501	普通遗传学	杨业华	高等教育出版社

(a) 图书信息表

作者	序号
邵培仁	4
李泽厚	2
李宗宏	5
刘云	8
聂卫平	6
王宇川	9
杨业华	11
张琳	10
张品兴	1
张永桃	3
邹惠成	7

(b) 作者索引表

出版社	序 号
高等教育出版社	3,4,11
科学出版社	7,9
兰州大学出版社	5
人民邮电出版社	10
上海文艺出版社	2
中国铁道出版社	8
中国文联出版社	1,6

(c) 出版社索引表

图 1.2 图书信息检索系统中的数据结构

诸如此类的还有电话自动查号系统、学生信息查询系统、仓库库存管理系统等。在这类数学模型中,计算机处理的对象之间通常存在着一种简单的线性关系,这类数学模型可以称为线性数据结构。

例 1.2 人机对弈问题。人机对弈是一个古老的人工智能问题,其解题思想是将对弈的策略事先存入计算机,策略包括对弈过程中所有可能的情况以及相应的对策。在决定对策时,根据当前状态考虑局势发展的趋势做出最有利的选择。因此,计算机操作的对象(数据元素)是对弈过程中的每一步棋盘状态(格局),数据元素之间的关系由比赛规则决定。通常,这个关系不是线性的,因为从一个格局可以派生出多个格局,所以通常用树形结构来表示。图 1.3 所示是井字棋的对弈树。

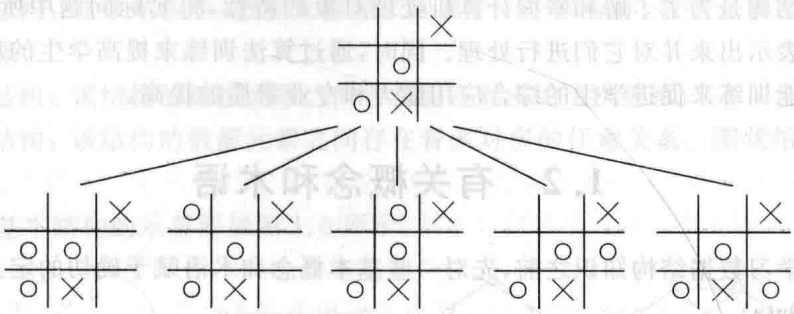
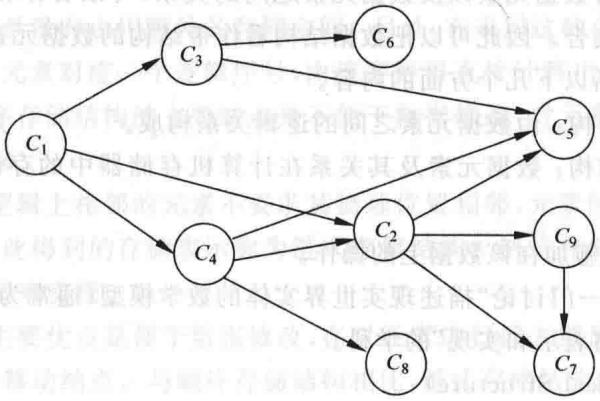


图 1.3　井字棋的对弈树

例 1.3　教学计划编排问题。一个教学计划包含许多课程，在教学计划包含的许多课程之间，有些课程必须按规定的先后次序进行学习，即有些课程之间有先修和后修的关系，有些则没有次序要求。课程之间先修和后修的次序关系可以用一个被称为图的数据结构来表示，如图 1.4 所示。有向图中的每个顶点表示一门课程，如果从顶点 v_i 到 v_j 之间存在有向边 $<v_i, v_j>$，则表示课程 i 必须先于课程 j 进行。

课程编号	课 程 名 称	先修课程
C_1	计算机导论	无
C_2	数据结构	C_1、C_4
C_3	汇编语言	C_1
C_4	C程序设计语言	C_1
C_5	计算机图形学	C_2、C_3、C_4
C_6	接口技术	C_3
C_7	数据库原理	C_2、C_9
C_8	编译原理	C_4
C_9	操作系统	C_2

(a) 计算机专业的课程设置



(b) 表示课程之间优先关系的有向图

图 1.4　教学计划编排问题的数据结构

由以上例子可知，描述非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。因此，数据结构课程是研究非数值计算的程序设计问题中计算机处理对象以及它们之间关系和操作的学科。

学习数据结构是为了了解和掌握计算机处理对象的特性,将实际问题中所涉及的处理对象在计算机中表示出来并对它们进行处理。同时,通过算法训练来提高学生的思维能力,通过程序设计的技能训练来促进学生的综合应用能力和专业素质的提高。

1.2 有关概念和术语

在系统地学习数据结构知识之前,先对一些基本概念和术语赋予确切的定义。

1. 数据(Data)

数据是信息的载体,能够被计算机识别、存储和处理。数据是计算机程序加工的原料,应用程序能处理各种各样的数据,包括数值数据和非数值数据。数值数据是一些整数、实数或复数;非数值数据包括字符、文字、图形、图像、语音等。

2. 数据元素(Data Element)

数据元素是数据的基本单位,在计算机程序中通常作为一个整体进行考虑和处理。一个数据元素可由若干个数据项(Data Item)组成。在不同的条件下,数据元素又可称为元素、结点、顶点、记录等。例如,学生信息检索系统中学生信息表中的一个记录、教学计划编排问题中的一个顶点等,都被称为一个数据元素。

3. 数据项(Data Item)

数据项指不可分割的、具有独立意义的最小数据单位,有时也被称为字段(field)或域。例如,学籍管理系统中学生信息表的每一个数据元素就是一个学生记录。它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。这些数据项可以分为两种:一种叫作初等项,如学生的性别、籍贯等,这些数据项是在数据处理时不能再分割的最小单位;另一种叫作组合项,如学生的成绩,它可以再划分为数学、物理、化学等更小的项。通常,在解决实际应用问题时把每个学生记录当作一个基本单位进行访问和处理。

4. 数据结构(Data Structure)

数据结构是指所有数据元素以及数据元素之间的关系,可以看作相互之间存在着某种特定关系的数据元素的集合。因此可以把数据结构看作带结构的数据元素的集合。

数据结构通常包括以下几个方面的内容。

(1) 数据的逻辑结构:由数据元素之间的逻辑关系构成。

(2) 数据的存储结构:数据元素及其关系在计算机存储器中的存储表示,通常也叫作数据的物理结构。

(3) 数据的运算:施加在该数据上的操作。

因此,数据结构是一门讨论"描述现实世界实体的数学模型(通常为非数值计算)及其之上的运算在计算机中如何表示和实现"的学科。

1) 逻辑结构(Logical Structure)

在任何问题中,数据元素之间都不会是孤立的,都存在着这样或那样的关系,这种数据元素之间存在的关系称为数据的**逻辑结构**。数据的逻辑结构与数据的存储无关,是独立于计算机的,因此数据的逻辑结构可以看作从具体问题抽象出来的数学模型。

根据数据元素之间关系的不同特性,通常有以下4类基本的逻辑结构。

(1) 集合结构:在集合结构中,数据元素之间的关系是"属于同一个集合"。数据元素之间除了同属一个集合外,不存在其他关系。

(2) 线性结构：在该结构中，数据元素除了同属于一个集合外，数据元素之间还存在着一对一的顺序关系。

(3) 树形结构：该结构的数据元素之间存在着一对多的层次关系。

(4) 图状结构：该结构的数据元素之间存在着多对多的任意关系。图状结构也称作网状结构。

上述 4 类基本结构的示意图如图 1.5 所示。

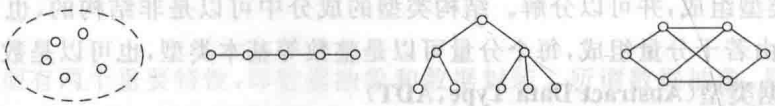


图 1.5　4 种基本逻辑结构的示意图

由于集合是数据元素之间极为松散的一种结构，本书不作专门讨论。因此，本书主要讨论线性结构(表、栈、队、串等)和非线性结构(树、图或网)。

2) 存储结构(Storage Structure)

数据的逻辑结构在计算机中的存储表示(又称映像)称为数据的**存储结构**，或称**物理结构**。它所研究的是数据结构在计算机中的实现方法，包括数据结构中数据元素的存储表示以及数据元素之间关系的表示。

显然数据的存储结构是依赖于计算机的，通常设计数据的存储结构是借助某种计算机语言来实现的，一般只在高级语言的层次上讨论存储结构，如 C、C++语言等。

在计算机中，数据的存储方法主要有**顺序存储**和**链式存储**。

顺序存储方法是通过数据元素在计算机中的存储位置关系来表示元素间的逻辑关系，通常把逻辑上相邻的元素存储在物理位置相邻的存储单元中，该存储方式称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。

顺序存储结构的主要优点是存储效率高，因为分配给数据的存储单元全用于存储数据元素，元素之间的逻辑关系没有占用额外的存储空间；另外，在采用这种存储方法时可实现对元素的随机存取，即每个元素对应一个逻辑序号，由该序号可直接计算出对应元素的存储地址，从而获取元素值。顺序存储结构的主要缺点是不便于数据修改，对元素的插入和删除操作可能需要移动一部分数据。

链式存储方法对逻辑上相邻的元素不要求其物理位置相邻，元素间的逻辑关系通过附设的指针字段来表示，由此得到的存储表示称为链式存储结构。链式存储结构通常借助于程序设计语言中的指针类型来实现。

链式存储结构的主要优点是便于数据修改，在对元素进行插入或删除操作时仅需修改相应结点的指针域，不必移动结点。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利用率较低，因为分配给元素的存储单元有一部分被用来存储结点之间的逻辑关系；另外，由于逻辑上相邻的元素在存储空间上不一定相邻，所以不能对元素进行随机存取。

除了顺序存储方法和链式存储方法外，有时为了查找方便，还采用索引存储方法和散列表(Hash)存储方法，这里不再赘述。

5. 数据类型(Data Type)

数据类型是和数据结构密切相关的一个概念，是指在高级程序设计语言中用以限制变量

取值范围和可能进行的操作的总和。因此所谓数据类型,一是限定了数据的取值范围(实际上与存储形式有关),二是规定了数据能够进行的一组操作(运算)。

在用高级程序语言编写的程序中,必须对程序中出现的每个变量、常量或表达式,**明确说明**它们所属的**数据类型**。

数据类型可分为两类:一类是非结构的原子类型,原子类型的值是不可再分解的,如C语言中的基本类型,如整型、实型、字符型及指针类型和空类型;另一类是结构类型,它的成分可以由多个结构类型组成,并可以分解。结构类型的成分中可以是非结构的,也可以是结构的。例如,数组的值由若干分量组成,每个分量可以是整数等基本类型,也可以是数组等结构类型。

6. 抽象数据类型(Abstract Data Type,ADT)

抽象数据类型指的是用户进行软件系统设计时从问题的数学模型中抽象出来的逻辑结构和逻辑结构上的运算,而不考虑计算机的具体存储结构和运算的具体实现算法。抽象数据类型中的数据对象和数据运算的声明与数据对象的表示和数据运算的实现相互分离。

一个具体问题的抽象数据类型的定义通常采用简洁、严谨的文字描述,一般包括数据对象、数据关系和数据运算三方面内容,所以抽象数据类型可用(D,S,P)三元组表示。

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。

我们采用以下格式来定义ADT:

```
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,基本操作的定义格式为:

```
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〉
    赋值参数:只为操作提供输入值。
    引用参数:除可提供输入值外,还将返回操作结果,一般前面加&以示区别。
    初始条件:描述了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,并返回相
              应出错信息。
    操作结果:说明了操作正常完成之后,数据结构的变化状况和应返回的结果。若初始条件为空,则
              省略之。
```

例如,抽象数据类型**复数**的定义:

```
ADT Complex {
数据对象:
    D={e1,e2|e1,e2∈RealSet  }
数据关系:
    R1={<e1,e2>| e1是复数的实数部分,e2是复数的虚数部分}
基本操作:
  AssignComplex( &Z, v1, v2 )
    操作结果:构造复数Z,其实部和虚部分别被赋以参数v1和v2的值。
  DestroyComplex( &Z)
    操作结果:复数Z被销毁。
  GetReal( Z, &realPart )
    初始条件:复数已存在。
```

操作结果：用 realPart 返回复数 Z 的实部值。

GetImag(Z, &ImagPart)

初始条件：复数已存在。

操作结果：用 ImagPart 返回复数 Z 的虚部值。

Add(z1,z2, &sum)

初始条件：z1、z2 是复数。

操作结果：用 sum 返回两个复数 z1、z2 的和值。

} ADT Complex

抽象数据类型有两个重要特性，即数据抽象和数据封装。所谓数据抽象，是指用 ADT 描述程序处理的实体时强调的是其本质的特征、其所能完成的功能以及它和外部用户的接口（即外界使用它的方法）。所谓数据封装，是指将实体的外部特性和其内部实现细节分离，并且对外部用户隐藏其内部实现细节。

1.3　算法及算法分析

算法与数据结构的关系非常紧密，在设计算法时总是要先确定相应的数据结构，而在讨论某一种数据结构时也必然会涉及相应的算法。下面就从算法的特性、算法的描述、算法的性能分析与衡量 3 个方面对算法进行介绍。

1.3.1　算法的特性

算法（Algorithm）是对特定问题求解步骤的一种描述，是指令的有限序列。其中，每一条指令表示一个或多个操作。一个算法应该具有下列特性：

（1）有穷性。一个算法必须在有穷步之后结束，即必须在有限的时间内完成。

（2）确定性。算法的每一步必须有确切的定义，无二义性，且在任何条件下算法只有唯一一条执行路径，即对于相同的输入只能得到相同的输出。

（3）可行性。算法中的每一步都可以通过已经实现的基本运算的有限次执行得以实现。

（4）有输入。一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。

（5）有输出。一个算法具有一个或多个输出，这些输出和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，例如操作系统，只要整个系统不遭破坏，它将永远不会停止，即使没有作业需要处理，它也处于动态等待中。因此，操作系统不是一个算法。另外，程序中的指令必须是计算机可执行的，而算法中的指令则无此限制。算法代表了对问题的求解方法，而程序则是算法在计算机上的特定实现。若一个算法用程序设计语言来描述，则它就是一个程序。

算法与数据结构是相辅相成的。解决某一类特定问题的算法可以选择不同的数据结构，而且选择的恰当与否将直接影响算法的效率。反之，一种数据结构的优劣由各种算法的执行效果来体现。

在算法设计时通常需要考虑以下几个方面的要求：

（1）正确性。算法的执行结果应当满足预先规定的功能和性能要求。正确性要求表明算法必须满足实际需求，达到解决实际问题的目标。

（2）可读性。一个算法应当思路清晰、层次分明、简单明了、易读易懂。可读性要求表明

算法主要是人与人之间交流解题思路和进行软件设计的工具,因此可读性必须强。同时,一个可读性强的算法,其程序的可维护性、可扩展性都要好许多,因此,许多时候人们往往在一定程度上牺牲效率来提高可读性。

(3) 健壮性。当输入不合法的数据时,应能做适当处理,不至于引起严重的后果。健壮性要求表明算法全面细致地考虑所有可能的边界情况,并对这些边界条件做出完备的处理,尽可能使算法没有意外的情况。

(4) 高效性。一个算法应当有效地使用存储空间和有较好的时间效率。高效性主要指时间效率,即解决相同规模的问题的时间尽可能短。

一般来说,数据结构上的**基本操作**主要有以下几种:

(1) 查找。查找指寻找满足特定条件的数据元素所在的位置。

(2) 读取。读取指读出指定位置上数据元素的内容。

(3) 插入。插入指在指定位置上添加新的数据元素。

(4) 删除。删除指删去指定位置上对应的数据元素。

(5) 更新。更新指修改某个数据元素的值。

1.3.2 算法的描述

算法的描述方法有很多,根据描述方法的不同,大致可以将算法描述分为以下4种:

(1) 自然语言算法描述。用人类自然语言(例如中文、英文等)来描述算法,还可插入一些程序设计语言中的语句来描述,这种方法也称为非形式算法描述。其优点是不需要专门学习,任何人都可以直接阅读和理解,但其直观性很差,复杂的算法难写、难读。

(2) 框图算法描述。这是一种图示法,可以采用框图、流程图、N-S图等来描述算法,这种描述方法在算法研究的早期曾流行过。它的优点是直观、易懂,但用来描述比较复杂的算法显得不够方便,也不够清晰、简捷。

(3) 伪代码算法描述。例如类C语言算法描述,这种算法描述很像程序,但它不能直接在计算机上编译、运行。这种方法很容易编写、阅读算法,而且格式统一,结构清晰,专业设计人员经常使用类C语言来描述算法。

(4) 高级程序设计语言编写的程序或函数。这是直接用高级语言来描述算法,可以在计算机上运行并获得结果,使给定问题能在有限时间内被求解,通常这种算法描述也称为程序。

例 1.4 求两个整数 m、$n(m \geqslant n)$的最大公因子,该算法的不同描述方法如下:

(1) 非形式算法描述(自然语言算法描述)。

① [求余数]以 n 除 m,并令 r 为余数($0 \leqslant r < n$);

② [判断余数是否为零]若 $r=0$,则结束算法,n 就是最大公因子;

③ [替换并返回步骤①]若 $r \neq 0$,则 $m \leftarrow n$、$n \leftarrow r$,返回①。

(2) 框图描述(程序流程图)如图1.6所示。

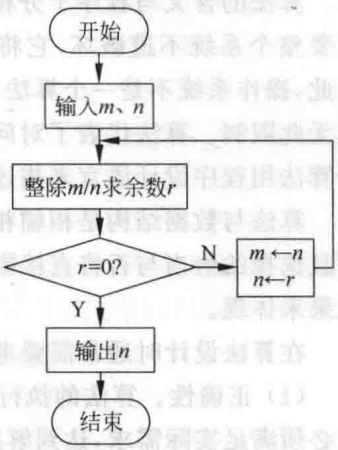


图 1.6 算法的框图描述

(3) C语言函数描述。

```
int max_common_factor(int m, int n)
{
    int r;
```

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

本书主要介绍算法的思路和实现过程，且尽可能给出算法对应的 C 语言函数或程序（或类 C 语言算法描述），以方便读者阅读或上机运行，从而更好地理解算法。

1.3.3　算法的性能分析与衡量

所谓好的算法，除了要满足上文提到的几个基本要求外，还必须以较少的时间与空间代价来解决相同规模的问题。因此，一个算法的好坏，可以从该算法在计算机上运行的时间和所占的存储空间来衡量和评判。算法分析就是预先分析算法在实际执行时的时空代价指标。

当一个算法被转换成程序并在计算机上执行时，其运行所需要的时间一般取决于下列几个因素：

（1）硬件的速度。硬件的速度即主机本身运行的速度，主要与 CPU 的主频和字长有关，还与主机系统采用的技术有关。例如，多机系统的运算速度一般比单机系统快。

（2）实现算法的程序设计语言。实现算法的语言的级别越高，其执行效率相对越低。

（3）编译程序所生成目标代码的质量。代码优化较好的编译程序所生成的程序质量较高。

（4）算法所采用的策略。采用不同的设计思路与解题方法，其时间和空间代价是不同的，一般情况下，时间指标与空间指标常常是矛盾的两个方面。

（5）问题的规模。例如，求 100 以内的素数和求 1000 以内的素数，其执行时间必然是不同的。

显然，在各种因素都不能确定的情况下，很难比较算法的执行时间。也就是说，用算法的绝对执行时间来衡量算法的效率是不合适的。为此，可以将上述各种与计算机相关的软、硬件因素都确定下来，仅对采用不同策略的算法分析其运行代价随问题规模大小变化的对应关系，即运行代价仅依赖于问题的规模（通常用正整数 n 表示），或者说它是问题规模的函数，这种函数被称为算法的**时间复杂度**和**空间复杂度**。

1. 时间复杂度

一个程序的**时间复杂度**（Time Complexity）是指该程序的运行时间与问题规模的对应关系。一个算法是由控制结构和原操作（所谓原操作，是指从算法中选取一种对所研究问题是基本运算的操作）构成的，其执行时间取决于两者的综合效果。为了便于比较同一问题的不同算法，通常的方法是从算法中选取一种对于所研究问题来说是基本运算的原操作，以该原操作重复执行的次数作为算法的时间度量。一般情况下，算法中原操作重复执行的次数是该算法所处理问题的规模 n 的某个函数 $T(n)$。

例 1.5　两个 $n \times n$ 阶的矩阵相乘的程序中的主要语句及其重复次数如下：

```
                                          //原操作语句的执行频度
for ( i = 0; i < n; i++)
  for ( j = 0; j < n; j++)
    {  s[i][j] = 0                        //n²
      for ( k = 0; k < n; k++)
```