免费提供电子教案

高等院校规划教材
计算机科学与技术系列

数据结构简明教程

吴仁群　编著

机械工业出版社
CHINA MACHINE PRESS

高等院校规划教材·计算机科学与技术系列

数据结构简明教程

吴仁群　编著

机械工业出版社

本书是针对数据结构初学者编写的基础教程，书中不仅讲解了数据结构常用的基本理论知识，而且提供了大量的应用实例，以帮助初学者对知识进行充分的理解和掌握。全书共分8章，内容包括绪论，线性表，栈和队列，串和数组，树和二叉树，图，查找，排序等。

本书内容实用，结构清晰，实例丰富，可操作性强，可作为高等院校数据结构的教材，也可作为计算机相关专业的培训和自学教材。

图书在版编目（CIP）数据

数据结构简明教程/吴仁群编著. —北京：机械工业出版社，2010.12
（高等院校规划教材·计算机科学与技术系列）
ISBN 978-7-111-30178-3

Ⅰ.①数… Ⅱ.①吴… 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（2010）第050726号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
责任编辑：陈 皓
责任印制：乔 宇
三河市宏达印刷有限公司印刷
2011年1月第1版·第1次印刷
184mm×260mm·14.75印张·365千字
0001-3000册
标准书号：ISBN 978-7-111-30178-3
定价：27.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销 售 一 部：（010）68326294
销 售 二 部：（010）88379649
读者服务部：（010）68993821

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

封面无防伪标均为盗版

出 版 说 明

计算机技术的发展极大地促进了现代科学技术的发展，明显地加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容趋于更加科学和合理，计算机教材建设逐渐成熟。在“十五”期间，机械工业出版社组织出版了大量的计算机教材，包括“普通高等教育计算机规划教材”、“21 世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21 世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高等院校的计算机专家、教师及教务部门，针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成了“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高等院校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学而设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。另外，本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展而及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

数据结构是计算机相关专业中一门重要的专业基础课程。当用计算机来解决实际问题时，就要涉及数据及数据之间关系的表示及处理，而数据及数据之间关系的表示及处理正是数据结构的主要研究对象。通过数据结构的学习，可以为后续课程，尤其是软件方面的课程，打下厚实的知识基础。因此，数据结构在计算机相关专业中具有举足轻重的作用。

作为一本数据结构的基础教材，本书具有以下特点。

1）内容的讲述由浅入深，符合初学者的计算机语言学习习惯。

2）在讲述每个知识点时，都辅以图形或具体实例，读者能够从具体应用中掌握知识，能够很容易地将所学知识应用于实践。

3）每章后均附有习题，读者可通过做习题，巩固并掌握所学知识。

本书共有 8 章及 1 个附录。第 1 章讲述数据、数据结构、数据类型、抽象数据类型等基本概念以及算法和算法描述、算法的性能分析等有关知识；第 2 章介绍了线性表的含义及 ADT 描述、线性表的两种顺序存储和链式存储、不同存储方式下基本操作的实现及应用；第 3 章介绍了栈和队列的定义及 ADT 描述、栈和队列的存储结构、不同存储结构下基本操作的实现及应用；第 4 章介绍了串和数组的定义及 ADT 描述、串的存储结构及应用、数组的存储方式、压缩存储及应用；第 5 章讲述了树和二叉树的概念及 ADT 描述、树和二叉树的存储方式、树和二叉树及森林的遍历及应用、树和二叉树及森林的转换、哈夫曼树及应用；第 6 章讲述了图的概念及 ADT 描述、图的存储方式、图的遍历、最小生成树、有向无环图及应用等；第 7 章介绍了查找的基本概念、静态查找和动态查找的基本方法、哈希表的概念及查找方法等；第 8 章讲述了排序的基本概念，插入排序、交换排序、选择排序、归并排序和基数排序等排序的主要方法；附录提供了 3 个实验，目的在于帮助读者巩固所学知识。

本书由吴仁群（博士，北京印刷学院副教授，国家高级程序员）编写。在编写过程中，作者参考了大量的资料，在此向这些资料的作者表示深深的感谢！

由于作者水平有限，书中难免存在一些不足之处，敬请读者批评指正。

编　者

目　　录

出版说明

前言

第 1 章　绪论 …… 1

1.1　基本概念 …… 1

1.1.1　数据和数据结构 …… 1

1.1.2　数据类型 …… 3

1.1.3　抽象数据类型 …… 3

1.1.4　数据结构的符号描述举例 …… 4

1.2　算法和算法描述 …… 5

1.2.1　概念和特性 …… 5

1.2.2　算法设计要求 …… 6

1.2.3　算法描述 …… 6

1.3　算法的性能分析 …… 7

1.3.1　时间复杂度 …… 7

1.3.2　空间复杂度 …… 9

1.3.3　分析算法时间复杂度举例 …… 9

1.4　习题 …… 10

第 2 章　线性表 …… 12

2.1　线性表的含义及 ADT 描述 …… 12

2.2　顺序存储结构 …… 14

2.2.1　顺序表的存储表示 …… 14

2.2.2　顺序表基本操作的实现 …… 15

2.2.3　顺序表基本操作的时间复杂度分析 …… 19

2.2.4　顺序表的优缺点 …… 19

2.2.5　顺序存储结构的应用 …… 19

2.3　链式存储结构 …… 21

2.3.1　单链表的存储表示 …… 21

2.3.2　单链表基本操作的实现 …… 23

2.3.3　循环链表的表示和基本操作的实现 …… 28

2.3.4　双向链表的表示和基本操作的实现 …… 31

2.3.5　链式存储结构的应用 …… 32

2.4　习题 …… 36

第 3 章　栈和队列 …… 37

3.1　栈 …… 37

3.1.1　栈的定义及 ADT 描述 …… 37

3.1.2 栈的顺序存储结构 …… 38
3.1.3 栈的链式存储结构 …… 40
3.1.4 栈的应用 …… 42
3.2 队列 …… 45
3.2.1 队列的定义及 ADT 描述 …… 45
3.2.2 队列的顺序存储结构 …… 46
3.2.3 队列的链式存储结构 …… 49
3.2.4 队列的应用 …… 52
3.3 习题 …… 56
第 4 章 串和数组 …… 59
4.1 串 …… 59
4.1.1 串的定义及 ADT 描述 …… 59
4.1.2 串的顺序存储结构 …… 60
4.1.3 串的链式存储结构 …… 64
4.1.4 串的应用 …… 65
4.2 数组 …… 68
4.2.1 数组的定义及 ADT 描述 …… 68
4.2.2 数组的存储结构 …… 70
4.2.3 矩阵的压缩存储 …… 73
4.2.4 矩阵转置 …… 79
4.2.5 数组的应用 …… 82
4.3 习题 …… 85
第 5 章 树和二叉树 …… 88
5.1 树 …… 88
5.1.1 树的概念及 ADT 描述 …… 88
5.1.2 树的存储结构 …… 90
5.1.3 综合应用举例 …… 93
5.2 二叉树 …… 94
5.2.1 二叉树的概念及 ADT 描述 …… 94
5.2.2 二叉树的性质 …… 95
5.2.3 二叉树的存储结构 …… 98
5.2.4 遍历二叉树 …… 101
5.2.5 遍历算法的应用 …… 104
5.2.6 树、森林与二叉树的转换 …… 106
5.2.7 二叉树的综合应用 …… 110
5.3 树和森林的遍历 …… 113
5.3.1 树的遍历 …… 113
5.3.2 森林的遍历 …… 113
5.3.3 树和森林的遍历应用 …… 114

5.4 哈夫曼树及应用 …… 115
5.4.1 哈夫曼树 …… 115
5.4.2 判定树 …… 117
5.4.3 前缀编码 …… 118
5.5 习题 …… 120
第6章 图 …… 121
6.1 图的概述 …… 121
6.1.1 图的概念 …… 121
6.1.2 图的 ADT 描述 …… 124
6.2 图的存储结构 …… 125
6.2.1 邻接矩阵 …… 125
6.2.2 邻接表 …… 126
6.2.3 应用举例 …… 128
6.3 图的遍历 …… 129
6.3.1 深度优先遍历 …… 129
6.3.2 广度优先遍历 …… 130
6.3.3 应用举例 …… 131
6.4 最小生成树问题 …… 131
6.4.1 图的生成树和最小生成树 …… 131
6.4.2 最小生成树构造 …… 132
6.4.3 应用举例 …… 135
6.5 有向无环图及应用 …… 136
6.5.1 基本定义 …… 136
6.5.2 拓扑排序 …… 137
6.5.3 关键路径 …… 140
6.6 习题 …… 146
第7章 查找 …… 149
7.1 基本概念 …… 149
7.2 静态查找 …… 150
7.2.1 顺序查找 …… 150
7.2.2 折半查找 …… 152
7.2.3 折半查找的应用 …… 154
7.3 动态查找 …… 155
7.3.1 二叉排序树 …… 155
7.3.2 二叉排序树的查找 …… 156
7.3.3 二叉排序树的插入 …… 157
7.3.4 二叉排序树的删除 …… 159
7.3.5 二叉排序树的应用 …… 161
7.4 哈希表 …… 163

7.4.1 哈希表的概念 …… 163
7.4.2 哈希函数的构造 …… 164
7.4.3 冲突处理的方法 …… 166
7.4.4 哈希表查找及分析 …… 168
7.4.5 哈希表查找的应用 …… 169
7.5 习题 …… 171
第8章 排序 …… 172
8.1 基本概念 …… 172
8.2 插入排序 …… 173
8.2.1 直接插入排序 …… 173
8.2.2 希尔排序 …… 175
8.2.3 应用举例 …… 177
8.3 交换排序 …… 178
8.3.1 冒泡排序 …… 178
8.3.2 快速排序 …… 180
8.3.3 应用举例 …… 184
8.4 选择排序 …… 185
8.4.1 简单选择排序 …… 185
8.4.2 堆排序 …… 187
8.4.3 应用举例 …… 190
8.5 归并排序 …… 193
8.5.1 归并排序的基本思想 …… 193
8.5.2 2-路归并排序算法 …… 193
8.5.3 应用举例 …… 195
8.6 基数排序 …… 195
8.6.1 基数排序的基本思想 …… 195
8.6.2 链式基数排序算法 …… 199
8.6.3 应用举例 …… 201
8.6.4 排序方法的简单比较 …… 201
8.7 习题 …… 202
附录 实验指导 …… 204
实验一 通讯录管理信息系统模拟 …… 204
实验二 模拟停车场管理 …… 211
实验三 图的应用 …… 217
参考文献 …… 228

第1章 绪 论

本章学习目标：

- 了解数据结构的含义及有关概念。
- 了解数据结构的逻辑结构和物理结构。
- 了解算法的含义、描述方法及重要特性。
- 掌握估算算法的时间复杂度和空间复杂度的方法。

1.1 基本概念

数据结构是计算机科学与技术领域中广泛使用的术语。它用来反映一个数据的内部构成，即一个数据由哪些成分构成，这些成分的构成是什么样，呈现什么结构。数据结构作为一门学科主要研究数据的各种逻辑结构和存储结构，以及对数据的各种操作。因此，数据结构主要有3方面的内容：数据的逻辑结构、数据的物理存储结构和对数据的操作（或算法）。一般来说，算法的设计取决于数据的逻辑结构，算法的实现取决于数据的物理存储结构。

1.1.1 数据和数据结构

1. 数据

数据是对客观事物的符号表示，是所有能够输入到计算机中并被计算机程序处理的符号（如“0”、“1”、“a”、“b”等）的集合，包括字符、文字、表格、图像等，都可称为数据。例如，学生信息管理系统所要处理的数据可能是一张如表1-1所示的表格。

表1-1 学生信息表

姓 名	性 别	籍 贯	出生年月	政治面貌	联系方式
张三	男	湖北	1969-10-1	中共党员	29291230
李四	男	湖南	1969-5-1	群众	29293456
⋮	⋮	⋮	⋮	⋮	⋮

2. 数据元素

数据元素是数据集合中的一个实体，是计算机程序中加工处理的基本单位。数据元素按其组成可分为简单型数据元素和复杂型数据元素。简单型数据元素由一个数据项组成，所谓数据项就是数据中不可再分割的最小单位；复杂型数据元素由多个数据项组成，它通常包含着一个概念的多方面信息。

例如，一个在校大学生的基本信息组成如下。

姓名	性别	籍贯	出生年月	政治面貌	联系方式

在上述信息中，除“出生年月”外，其他都是简单型数据元素。“出生年月”可分为年、月、日3个部分，属复杂型数据元素。

出生年月：	年	月	日

3. 数据结构

简单地说，数据结构就是相互之间存在一种或多种特定关系的数据元素的集合。数据结构有逻辑上的数据结构（逻辑结构）和物理上的数据结构（物理结构）之分。

数据的逻辑结构是指数据元素之间的逻辑关系。常见的逻辑结构有集合结构、线性结构、树结构和图结构。

集合结构：数据元素之间的关系是“属于同一集合”，如图1-1a所示。

线性结构：数据元素之间存在一对一的关系，如图1-1b所示。

树结构：数据元素之间存在一对多的关系，如图1-1c所示。

图结构：数据元素之间存在多对多的关系，如图1-1d所示。

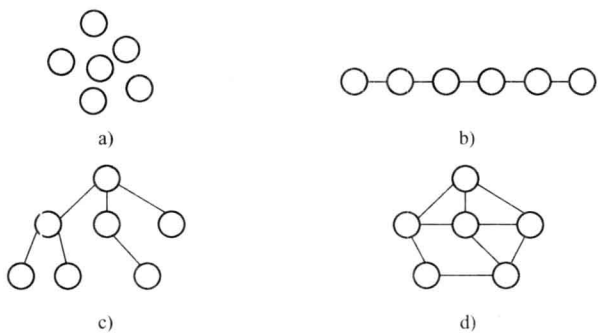

图1-1　基本逻辑结构

a）集合结构　b）线性结构　c）树结构　d）图结构

一般来说，一个数据结构（Data Struture，DS）可以表示为一个二元组：

$$DS = (D, S)$$

这里，D是数据元素的集合；S是定义在D（或其他集合）上的关系的集合。

例如，复数是一个数据结构，表示为：

$$Complex = (C, R) \quad C = \{c1, c2\} \quad R = \{ < c1, c2 > \}$$

数据的物理结构，也称存储结构，是指数据结构在计算机存储器中的具体实现，是逻辑结构的存储映像（Image）。常见的存储结构有顺序存储结构和链式存储结构。前者是借助于数据元素的相对存储位置来表示数据元素之间的逻辑结构，如图1-2a所示；后者是借助于指示数据元素地址的指针来表示数据元素之间的逻辑结构，如图1-2b所示。

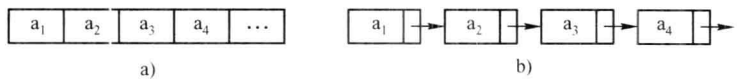

图1-2　基本存储结构

a）顺序结构　b）链式结构

为了叙述上的方便和避免产生混淆，通常把数据的逻辑结构统称为数据结构，把数据的物理结构统称为存储结构。

1.1.2 数据类型

在用高级程序语言编写的程序中，必须明确说明程序中出现的每个变量、常量或表达式所属的数据类型。数据类型是指数据的取值范围及其上可进行的操作的总称。例如，C 语言中提供的基本数据类型有整型、浮点型、双精度型、逻辑型和字符型等。

高级程序设计语言中的数据类型可分为简单类型和结构类型。简单类型中的值是不可分的，如整型、实型等。结构类型的值是由若干成分按某种结构组成的，是可分解的，如C 语言中的数组是一种结构类型，它是由固定个数的同一类型的数据顺序排列而成；结构体也是一种结构类型，它是由固定个数的不同类型的数据顺序排列而成。例如：

```
typedef    struct{
    int    age;
    char   name[20];
    float  score;
}STUDENT;
STUDENT     stu1, *p;
```

1.1.3 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论内部结构如何变化，只要它的数学特性不变，都不影响其外部使用。因此抽象数据类型可实现信息隐蔽和数据封装，以及使用与实现相分离。

抽象数据类型可以表示为一个三元组：

$$ADT = (D, S, P)$$

这里，D 是数据对象集合；S 是 D 上的关系集合；P 是 D 的基本操作。

实际中，可以按照如下结构来描述：

```
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
```

数据对象和数据关系的定义用伪码表示，基本操作定义格式为：

```
    基本操作名(参数表)
    初始条件:〈初始条件描述〉
    操作结构:〈操作结构描述〉
ADT Triplet{
    数据对象:D = {e1,e2,e3|e1,e2,e3 ∈ ElemSet}
```

数据关系:R1 = { <e1,e2>, <e2,e3> }

基本操作:

InitTriplet(&T,v1,v2,v3)

操作结果:构造了三元组 T,元素 e1,e2,e3 分别被赋以参数 v1,v2,v3 的值。

DestroyTriplet(&T)

操作结果:三元组被销毁。

Put(&T,i,e)

初始条件:三元组已存在,i ∈ [1,3]

操作结果:改变 T 中第 i 个元素的值为 e。

…

}//ADT Triplet

1.1.4 数据结构的符号描述举例

1. 集合结构

【例 1-1】 小组成员组成的数据结构。

Set = (D,R)

D = {张三,李四,王五,吴一,陈二}

R = { <张三,李四>, <张三,王五>, <张三,吴一>, <张三,陈二>, <李四,王五>, <李四,吴一>, <李四,陈二>, <王五,吴一>, <王五,陈二>, <吴一,陈二> }

这里,关系 <a, b> 表示 a 和 b 属于同一小组。

其数据结构如图 1-3a 所示。

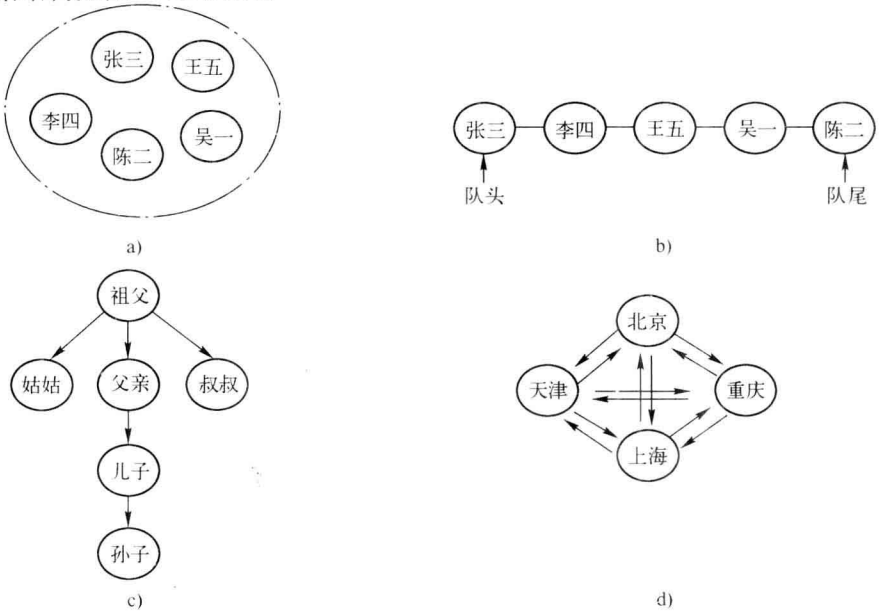

图 1-3 数据结构应用举例

a) 小组成员集合 b) 排队购买车票 c) 家庭成员组成 d) 航空网络

2. 线性结构

【例 1-2】 排队购买车票成员组成的数据结构。

List = (D,R)

D = {张三,李四,王五,吴一,陈二}

R = {<张三,李四>,<李四,王五>,<王五,吴一>,<吴一,陈二>}

这里，关系 <a，b> 表示在队列中 a 是 b 的直接前驱。

其数据结构如图 1-3b 所示。

3. 树结构

【例 1-3】 家庭成员组成的数据结构。

T = (D,R)

D = {祖父,姑姑,叔叔,父亲,儿子,孙子}

R = {<祖父,姑姑>,<祖父,叔叔>,<祖父,父亲>,<父亲,儿子>,<儿子,孙子>}

这里，关系 <a，b> 表示在队列中 a 是 b 的直接前驱。

其数据结构如图 1-3c 所示。

4. 图结构

【例 1-4】 4 个直辖城市航空网络的数据结构。

T = (D,R)

D = {北京,上海,天津,重庆}

R = {<北京,上海>,<北京,天津>,<北京,重庆>,<上海,北京>,<上海,天津>,<上海,重庆>,<天津,北京>,<天津,上海>,<天津,重庆>,<重庆,北京>,<重庆,天津>,<重庆,天津}

这里，关系 <a，b> 表示 a 有直达航班到 b。

其数据结构如图 1-3d 所示。

1.2 算法和算法描述

1.2.1 概念和特性

1. 算法概念

算法是在有限步骤内求解特定问题所使用的一组定义明确的规则。通俗点说，就是计算机求解特定问题的步骤的描述。特定的问题可以是数值的，也可以是非数值的。解决数值问题的算法叫做数值算法，科学和工程计算方面的算法都属于数值算法，如求解数值积分，求解线性方程组、求解代数方程、求解微分方程等；解决非数值问题的算法叫做非数值算法，数据处理方面的算法都属于非数值算法。例如，各种排序算法、查找算法、插入算法、删除算法、遍历算法等。数值算法和非数值算法并没有严格的区别。

2. 算法特征

一个算法应该具有有穷性、确定性、可行性、输入和输出 5 个重要特征。

有穷性是指一个算法必须保证执行有限步之后结束。

确定性是指算法的每一步骤必须有确切的定义，没有二义性。

可行性是指算法中描述的每一步操作都可以通过已有的基本操作执行有限次来实现。

输入是指一个算法有零个或多个输入，以描述运算对象的初始情况，所谓零个输入是指算法本身定出了初始条件。

输出是指一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

1.2.2 算法设计要求

评价一个好的算法有以下标准。

1）正确性：算法对于一切合法的输入数据都能产生满足规格说明的结果。

2）可读性：算法应该好读，易于理解，一般在满足正确性的前提下，算法越简单越好。

3）健壮性：算法应具有容错处理。当输入非法数据时，算法应对其做出反应，而不是产生莫名其妙的输出结果。

4）效率与存储量需求：效率是指算法执行的时间；存储量需求指算法执行过程中所需要的最大存储空间。

在保证满足前3个标准的情况下，当然还希望算法执行所需时间比较短，所占用存储空间比较小。实际中，要满足这两点往往是很困难的，因为时间和空间是彼此冲突的。因此，应该根据具体情况来权衡时间和空间。

1.2.3 算法描述

算法可以采取多种方式来描述。常见的描述方式有：采用自然语言描述，采用程序流程图的形式描述，采用某种具体程序语言描述，等等。

1. 采用自然语言描述

这种方式是使用自然语言描述问题的求解过程。下面举例说明。

问题P：判断正整数N是否为素数。

使用自然语言描述的算法如下。

Step1：令 i = 2；

Step2：判断 i 是否小于或等于 N/2，若是，转到 Step3；否则，转到 Step4。

Step3：判断 N 除以 i 的余数 R 是否等于零。

　　若 R 等于零，转到 Step5；

　　否则，i 加 1，转到 Step2。

Step4：输出 N 为素数。

Step5：算法结束。

2. 采用程序流程图的形式描述

这种方式是使用流程图符号描述问题的求解过程。求解问题P所对应的流程如图1-4所示。

3. 采用某种具体程序语言描述

这种方式是使用某种具体语言（如C语言）描述问题的求解过程。求解问题P所对应的C程序如下。

```
Void PrimeNumber( N)
int   N;
{
    int   i,j;
    for( i =2;i <= N/2;i ++ ) if ( N% i ==0) break;
    if( i > N/2) printf( "N 是素数")
}
```

除了上述3种方式外，还可以利用类语言（如类C语言、类Pascal等）描述问题的求解过程，感兴趣的读者可以参看有关文献。不管采用哪种方式描述算法，都必须能够正确描述求解过程。本书中所有算法均采用C语言描述。

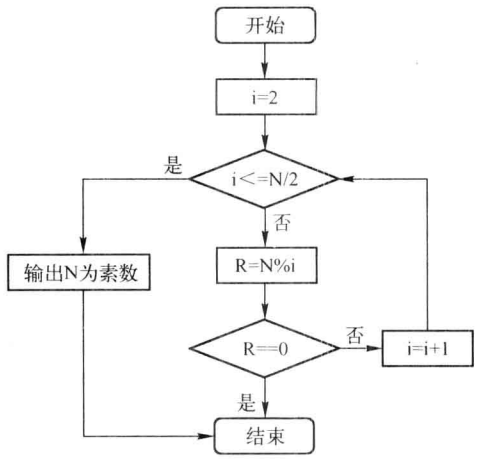

图1-4　程序流程图

4. 设计算法的基本过程

一般来说，针对某个具体设计求解算法的基本过程包括以下4个阶段。

1）通过对问题进行详细的分析，抽象出相应的数学模型。

2）确定使用的数据结构，并在此基础上设计对此数据结构实施各种操作的算法。

3）选用某种语言将算法转换成程序。

4）调试并运行这些程序。

1.3　算法的性能分析

算法的性能分析是指对算法质量优劣的评价。除了正确性、可读性和健壮性等特性外，重点要分析算法的时间复杂度、空间复杂度。

1.3.1　时间复杂度

算法的时间复杂度是依据该算法编制的程序在计算机上执行所消耗的时间来度量的。这种度量可采用事后统计和事前估计两种方式。

事后统计就是利用计算机内的计时功能，不同算法的程序可以用一组或多组相同的统计

数据区分。这种方式的缺点在于：必须先运行依据算法编制的程序，所得时间统计量依赖于硬件、软件等环境因素，掩盖算法本身的优劣。

一个高级语言程序在计算机上运行所消耗的时间取决于：

1）依据的算法选用何种策略。

2）问题的规模。

3）程序语言。

4）编译程序产生机器代码质量。

5）机器执行指令速度。

同一个算法用不同的语言、不同的编译程序，在不同的计算机上运行，其效率均不同，所以使用绝对时间单位衡量算法效率不合适。

实际中，可以撇开那些与计算机硬件和软件有关的因素，可以认为一个特定算法的“运行工作量”的大小，只依赖于问题的规模（通常用整数量表示），或者说，它是问题规模的函数。

任何一个算法都是由控制结构和若干基本操作组成的。一般情况下，算法中基本操作重复执行的次数是问题规模 n 的函数，记为 T(n)。以下以一个矩阵相乘算法来说明如何计算一个算法中语句执行的次数。

语　句	执行次数
maxtrixMultiply(A,B,C)	
int A[n][n],B[n][n],C[n][n]	
{	
for(i=0;i<n;i++)	n+1
{	
for(j=0;j<n;j++)	n(n+1)
{	
C[i][j]=0;	n^2
for(k=0;k<n;k++)	$n^2*(n+1)$
C[i][j]=C[i][j]+A[i][k]*B[k][j];	n^3
}	
}	
}	

总执行次数 T(n)为：

$$T(n)=n+1+n(n+1)+n^2+n^2*(n+1)+n^3=2n^3+3n^2+2n+1$$

定义：如果存在一个 g(n)，当 $n\to\infty$ 时，有

$$T(n)/g(n)=\text{常数}\neq 0$$

则称函数 T(n)与 g(n)同阶，或者说，T(n)与 g(n)同一个数量级，记为

$$T(n)=O(g(n))$$

其中，n 为问题的规模的量度。

上式称为算法的时间复杂度，或称该算法的时间复杂度为 O(g(n))。

基于高等数学中的极限知识可知，当一个算法的执行次数可以表达为如下形式：

$$T(n)=\alpha_m n^{\beta_m}+\alpha_{m-1}n^{\beta_{m-1}}+\cdots+\alpha_0,\ \beta_i>\beta_{i-1},\ i=1,2,\cdots,m$$

则该算法的时间复杂度为 $O(n^{\beta_m})$。