普通高等教育“十一五”规划教材

数据结构

SHUJU JIEGOU

周屹　任文　主编

机械工业出版社
CHINA MACHINE PRESS

普通高等教育“十一五”规划教材

数据结构

主　编　周　屹　任　文
副主编　邢传军　杨泽雪
参　编　李　萍　于雪梅　李向宏
主　审　雷国华

机械工业出版社

本书是作者结合多年教学实践经验，并根据数据结构课程知识丰富、内容抽象等特点，编写而成的一本具有较强实际应用价值的高职示范专业规划教材。全书共分9章，分别介绍数据结构相关基本概念、线性表、栈和队列、串、数组和广义表、树、图等基本数据结构以及典型的查找、排序方法的应用。

本书知识叙述简明扼要、通俗易懂，内容安排由浅入深、循序渐进，同时注意突出重点、分散难点。每章都附有小结、习题，便于教师教学和学生课后复习。

本书可作为计算机类专业或信息类相关专业的本科或专科教材，也可供从事计算机工程与应用工作的科技工作者参考。

为方便教学，本书配备电子课件等教学资源。凡选用本书作为教材的教师均可登录机械工业出版社教材服务网 www. cmpedu. com 免费下载。如有问题请致信 cmpgaozhi@ sina. com，或致电 010-88379375 联系营销人员。

图书在版编目（CIP）数据

数据结构/周屹，任文主编. —北京：机械工业出版社，2009.9
普通高等教育“十一五”规划教材
ISBN 978-7-111-28162-7

Ⅰ. 数… Ⅱ. ①周…②任… Ⅲ. 数据结构—高等学校—教材
Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字(2009)第 151547 号

机械工业出版社(北京市百万庄大街22号　邮政编码 100037)
策划编辑：王玉鑫　刘子峰　责任编辑：刘子峰
版式设计：霍永明　责任校对：张晓蓉
封面设计：鞠　杨　责任印制：李　妍
中国农业出版社印刷厂印刷
2009 年 9 月第 1 版第 1 次印刷
184mm×260mm · 12.5 印张 · 303 千字
0001—4000 册
标准书号：ISBN 978-7-111-28162-7
定价：22.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：(010)68326294
购书热线电话：(010)88379639　88379641　88379643
编辑热线电话：(010)88379543
封面无防伪标均为盗版

前　言

随着计算机软件和硬件的发展，计算机技术已经深入到社会的各个领域，各行各业都需要对大量的非数值数据进行存储、加工和管理。如何根据实际应用的要求，对数据进行有效的组织、存储和处理，进而编制出相应的高效率算法，是数据结构这门课程所要研究并加以解决的问题。

数据结构是计算机及其相关专业的一门重要的专业基础课，也是计算机及其相关专业的水平考试等必考科目。通过对数据结构课程的学习，学生应能够应用数据结构的知识和技巧，更好地进行算法和程序的设计，完成软件工程开发任务，并为学习操作系统、编译技术和数据库等后续课程打下良好的基础。

本书的编写充分考虑到数据结构课程知识丰富、内容抽象的特点。书中基础理论知识的阐述由浅入深、通俗易懂，突出各基本数据类型结构之间的区别和联系，引导学生体会将数据问题结构化、将应用问题计算机化的过程和方法，为后继课程学习打下良好基础。全书共分9章：第1章绪论，主要介绍数据、数据结构和算法等基本概念；第2章至第7章分别讨论线性表、栈和队列、串、数组和广义表、树和二叉树以及图等基本类型的数据结构，内容包括它们的逻辑结构、存储结构以及在各种存储结构下相应运算的算法；第8章和第9章介绍查找和排序，以及几种常用的查找和排序方法。各章节列举了很多实用的例子，有助于学生加深对基础理论知识的理解。书中所用的程序和算法均以C语言的形式给出，读者可以直接应用。

本书由长期承担计算机专业基础课教学、具有丰富教学经验的一线教师编写，概念清楚、内容丰富、针对性强，着重培养学生实际应用的能力，突出实践性和实用性。

本书由周屹、任文任主编，邢传军、杨泽雪任副主编，李萍、于雪梅、李向宏任参编。编写分工如下：周屹编写第1、2、6章，任文编写第3、8章，邢传军编写第7章，杨泽雪编写第5章，李萍、于雪梅、李向宏编写第4、9章。雷国华教授任本书的主审，周屹进行了全书的统稿工作。

本书的编写得到各方面的大力支持，在此一并表示衷心的感谢。

由于编者水平有限，书中难免出现错漏之处，希望广大读者批评指正。

编　者

目　录

第1章 绪 论

学习目标：

1）理解数据结构的基本概念。
2）了解算法的五个基本特性。
3）掌握类 C 语言的基本语法结构。
4）掌握计算算法时间复杂度的方法。

计算机科学是一门研究数据表示和数据处理的科学。数据是计算机化的信息，是计算机可以直接处理的最基本、最重要的对象。无论是进行科学计算、数据处理、过程控制，还是对文件进行存储和检索等操作，实际上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出相应的算法和程序。

1.1 数据结构概述

众所周知，计算机程序的作用是对信息（数据）进行加工处理。在大多数情况下，这些信息之间往往具有重要的结构关系，这就是数据结构所要研究的内容。

在计算机发展的初期，人们使用计算机的目的主要是处理数值计算问题。当使用计算机来解决一个具体问题时，一般需要经过下列几个步骤：首先要从该具体问题中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，最后编出程序进行测试，直至得到最终的解答。由于当时所涉及的运算对象是简单的整型、实型或布尔型数据，所以程序设计者的主要精力集中在程序设计的技巧上，而无须考虑数据结构。

随着计算机应用领域的扩大和软、硬件的发展，非数值计算问题越来越显得重要。据统计，当今处理非数值计算性问题占用了 90% 以上的机器时间。这类问题涉及的数据结构更为复杂，数据元素之间的相互关系一般无法用简单的数学方程式加以描述。因此，有效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。下面所列举的就是属于这一类的具体问题。

例 1-1 学生信息检索系统。当需要查找某个学生有关情况，或者想查询某个专业或年级的学生整体情况的时候，只要建立了相关的数据结构，按照某种算法编写了相应程序，就可以实现计算机自动检索。由此，可以在学生信息检索系统中建立一张按学号顺序排列的学生信息表和分别按姓名、专业、年级顺序排列的索引表，如图 1-1 所示。

学 号	姓 名	性 别	专 业	年 级
010001	孙雷	男	计算机科学与技术	2001 级
010002	李文伟	女	信息管理	2001 级
020301	姚路	女	数学与应用数学	2002 级
020302	李世民	男	信息管理	2002 级
020303	石宝	男	计算机科学与技术	2002 级
030801	文颖	女	计算机科学与技术	2003 级
030802	赵利	男	数学与应用数学	2003 级
030803	刘文靖	男	信息管理	2003 级
040601	姚路	女	计算机科学与技术	2004 级
040602	王永鸣	男	数学与应用数学	2004 级

a)

刘文靖	8
文颖	6
李文伟	2
姚路	3，9
石宝	5
王永鸣	10
孙雷	1
赵利	7
李世民	4

b)

计算机科学与技术	1，5，6，9
信息管理	2，4，8
数学与应用数学	3，7，10

c)

2004 级	9，10
2003 级	6，7，8
2002 级	3，4，5
2001 级	1，2

d)

图 1-1 学生信息查询系统中的数据结构

a）学生信息表 b）姓名索引表 c）专业索引表 d）年级索引表

由这四张表构成的文件就是学生信息检索的数学模型，计算机的主要操作便是按照某个特定要求（如给定姓名）对学生信息文件进行查询。

诸如此类的信息检索还有电话自动查号系统、考试查分系统、仓库库存管理系统等。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着一种简单的线性关系，这类数学模型可称为线性的数据结构。

例 1-2 八皇后问题。在八皇后问题（在 8×8 的网格中摆放 8 枚棋子，使任意两枚棋子都不在同一行、同一列或同一斜线上）中，处理过程不是根据某种确定的计算法则，而是利用试探和回溯的探索技术求解。为了求得合理布局，在计算机中要存储布局的当前状态。从最初的布局状态开始，一步步地进行试探，每试探一步形成一个新的状态，整个试探过程形成了一棵隐含的状态树，如图 1-2 所示（为了方便示意，这里将八皇后问题简化为四皇后问题）。

回溯法求解过程实质上就是一个遍历状态树的过程。在这个问题中所出现的树也是一种数据结构，它可以应用在许多非数值计算问题中。

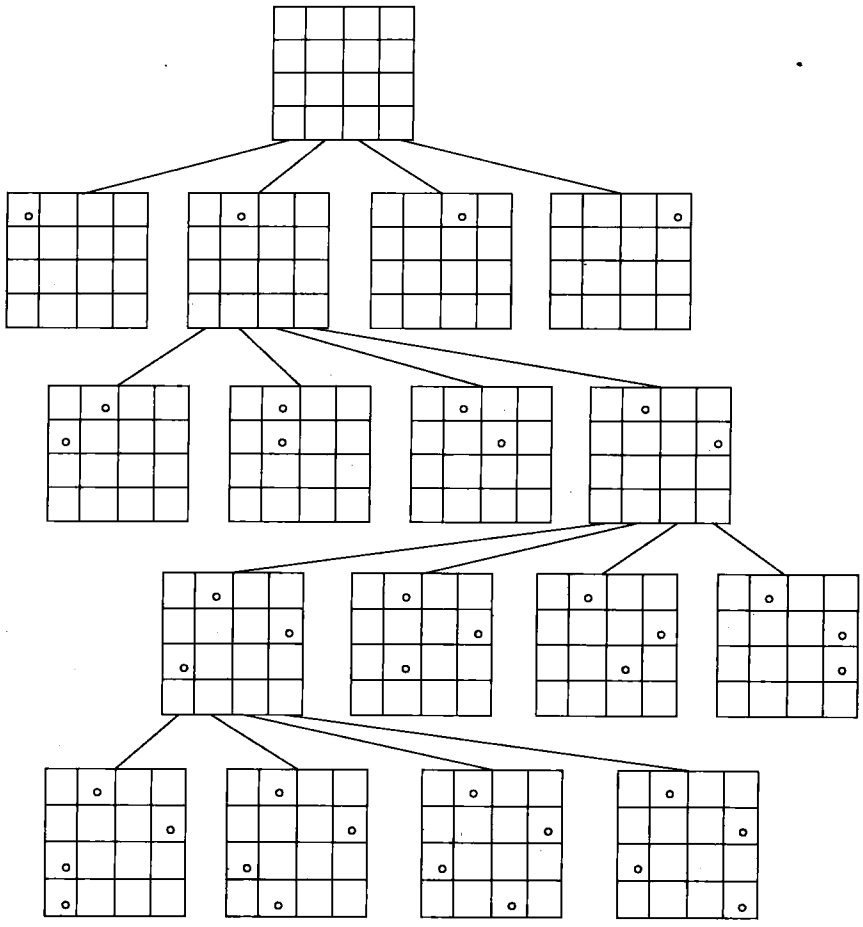
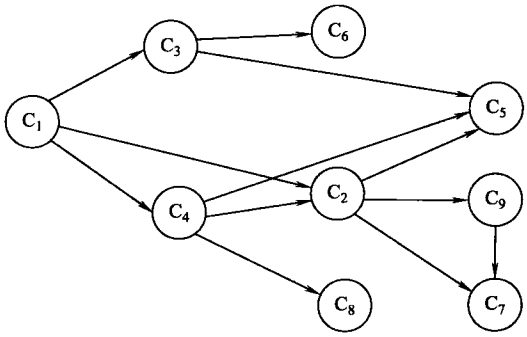


图 1-2　四皇后问题中隐含的状态树

例 1-3　教学计划编排问题。一个教学计划包含许多课程，在这些课程之间，有些必须按规定的先后次序进行，有些则没有次序要求。这种各个课程之间的次序关系可用一个称作“有向图”的数据结构来表示，如图 1-3 所示。

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)



b)

图 1-3　教学计划编排问题的数据结构

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

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。

1.2 基本概念和术语

数据（Data）是对信息的一种符号表示。在计算机科学中，数据是指所有能输入到计算机中并被计算机程序处理的符号的总称。

数据元素（Data Element）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处理。

有时，一个数据元素可由若干个**数据项（Data Item）**组成，例如，学籍管理系统中学生信息表的每一个学生记录就是一个数据元素，它包括学生的学号、姓名、性别、籍贯、出生年月、成绩等数据项。数据项是数据的不可分割的最小单位。

数据对象（Data Object）是性质相同的数据元素的集合，是数据的一个子集。在某个具体问题中，数据元素都具有相同的性质，属于同一数据对象。

数据结构（Data Structure）是相互之间存在一种或多种特定关系的数据元素的集合。在任何问题中，数据元素之间都不会是孤立的，在它们之间都存在着这样或那样的关系，这种数据元素之间的关系称为结构。根据数据元素间关系的不同特性，通常有下列四类基本的结构（如图 1-4 所示）：

1）集合结构。在集合结构中，数据元素除了同属于一种类型外，别无其他关系。集合是元素关系极为松散的一种结构。

2）线性结构。该结构的数据元素之间存在着一对一的关系。

3）树形结构。该结构的数据元素之间存在着一对多的关系。

4）图形结构。该结构的数据元素之间存在着多对多的关系，因此也称作网状结构。

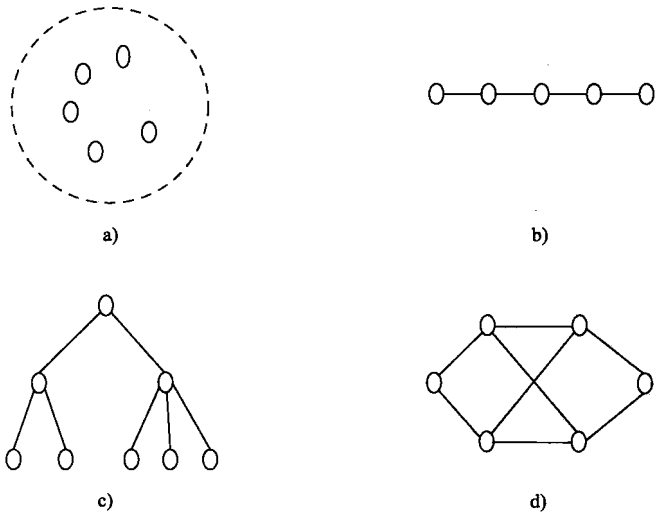


图 1-4 四类基本结构的示意图

a）集合结构 b）线性结构 c）树形结构 d）图形结构

一个数据结构有两个要素：一个是数据元素的集合，另一个是关系的集合。因此，在形式上，数据结构通常可采用一个二元组来表示。其形式定义如下：

$$Data\_Structure = (D, R)$$

其中，D 是数据元素的有限集；R 是 D 上关系的有限集。

数据结构不同于数据类型，也不同于数据对象。它不仅要描述数据类型的数据对象，而且要描述数据对象各元素之间的相互关系。

数据结构包括数据的**逻辑结构（Logical Structure）**和数据的**物理结构（Physical Structure）**。数据的逻辑结构可以看做是从具体问题抽象出来的数学模型，它与数据的存储无关。

研究数据结构的目的是为了在计算机中实现对它的操作，为此还需要研究如何在计算机中表示一个数据结构。数据结构在计算机中的标识（又称映像）称为数据的物理结构或存储结构。它所研究的是数据结构在计算机中的实现方法，包括数据结构中元素的表示及元素间关系的表示。

数据的存储结构可采用**顺序存储（Sequential Storage）**或**链式存储（Linked Storage）**的方法。

1）顺序存储方法是把逻辑上相邻的元素存储在物理位置相邻的存储单元中，由此得到的存储表示称为顺序存储结构。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。

2）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来实现。

除了通常采用的顺序存储方法和链式存储方法外，有时为了查找的方便还采用**索引存储（Indexed Access）**方法和**散列存储（Hashed Access）**方法。

数据类型（Data Type）是和数据结构密切相关的一个概念。它最早出现在高级程序设计语言中，用以描述程序中操作对象的特性。在用高级语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。类型显式地或隐含地规定了在程序执行期间变量或表达式所有可能的取值范围，以及在这些值上允许进行的操作。因此，数据类型是一个值的集合和定义在这个集合上的一组操作的总称。

在高级程序设计语言中，数据类型可分为两类：一类是原子类型，另一类是结构类型。原子类型的值是不可分解的，如 C 语言中整型、字符型、浮点型、双精度型等基本类型，分别用保留字 int、char、float、double 等标识。而结构类型的值是由若干成分按某种结构组成的，因此是可分解的，并且它的成分可以是非结构的，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，也可以是数组等。在某种意义上，数据结构可以看成是“一组具有相同结构的值”，而数据类型则可被看成是由一种数据结构和定义在其上的一组操作所组成的。

抽象数据类型（Abstract Data Type，ADT）是指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。即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各种计算机都拥有的整数类型就是

一个抽象数据类型，尽管它们在不同处理器上的实现方法不尽相同，但由于其定义的数学特性相同，因此在用户看来都是相同的。可见，“抽象”的意义在于数据类型的数学抽象特性。但在另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型，还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的重用性，在近代程序设计方法学中，要求在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作，并在模块的内部给出这些数据的表示及其操作的细节，而在模块的外部使用的只是抽象的数据及抽象的操作。这也就是面向对象的程序设计方法。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作组成，而数据结构又包括数据元素及元素间的关系，因此抽象数据类型一般可以由元素、关系及操作三种要素来定义。

1.3 算法和算法分析

算法与数据结构的关系紧密，在算法设计时先要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。下面就从算法特性、算法描述和算法性能分析与度量三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。一个算法必须在执行有限步之后结束，即必须在有限时间内完成。

2）确定性。算法的每一步必须有确切的定义，无二义性。算法的执行对应着的相同的输入仅有唯一的一条路径。

3）可行性。一个算法必须是可行的，即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）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）输出。一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。

算法的含义与程序十分相似，但又有区别。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，它仍处于动态等待中，因此操作系统不是一个算法。另一方面，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的解，而程序则是算法在计算机上的特定实现。一个算法若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承的。解决某一特定类型问题的算法可以选定不同的数据结构，而且选择恰当与否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来体现。

1.3.2 算法的描述方法

算法可以使用各种不同的方法来描述。最简单的方法是使用自然语言，其优点是简单且便于人们对算法的阅读，缺点是不够严谨；也可以使用程序流程图、N-S 图等算法描述工

具，其特点是描述过程简洁、明了。用以上两种方法描述的算法都不能够直接在计算机上执行，若要将它转换成可执行的程序还有一个编程的问题。可以直接使用某种程序设计语言来描述算法，不过直接使用程序设计语言并不容易，而且不太直观，常需要借助于注释才能使人看明白。

为了解决理解与执行这两者之间的矛盾，人们常使用一种称为伪码语言的描述方法来进行算法描述。伪码语言介于高级程序设计语言和自然语言之间，它忽略了高级程序设计语言中一些严格的语法规则与描述细节，因此它比程序设计语言更容易描述和被人理解，而比自然语言更接近程序设计语言。它虽然不能直接执行但很容易被转换成高级语言。

综上所述，一个算法可以用自然语言、程序设计语言或伪码语言来描述。本书选用类 C 语言作为描述算法的工具，下面简单介绍类 C 语言的一些语法结构。

(1) 预定义常量和类型 例如以下语句格式：

```
# define    TRUE    1;
# define    FALSE   -1;
# define    ERROR   NULL;
```

(2) 函数的定义 定义函数的语法格式如下：

```
[数据类型]  函数名([形式参数])
  [形式参数说明;]
      {  内部数据说明;
         执行语句组;
      }/* 函数名 */
```

函数的定义主要由函数名和函数体组成，函数体用花括号括起来。函数中用方括号括起来的部分为可选项，函数名之后的圆括号不可省略。函数的结果可由指针或别的方式传递到函数之外。执行语句可由各种类型的语句所组成，两个语句之间用分号分隔。可将函数中表达式的值通过 return 语句返回给调用它的函数。最后的“/* 函数名 */”为注释部分。

(3) 赋值语句

1) 简单赋值语句。语法格式如下：

〈变量名〉=〈表达式〉; /* 表示将表达式的值赋给左边的变量 */

〈变量〉++; /* 表示变量加 1 后赋值给原变量 */

〈变量〉--; /* 表示变量减 1 后赋值给原变量 */

2) 条件赋值语句。语法格式如下：

〈变量名〉 = 〈条件表达式〉?〈表达式 1〉:〈表达式 2〉; /* 判断条件表达式，如果非零，则将表达式 1 的值赋给左边的变量；否则将表达式 2 的值赋给该变量 */

3) 交换赋值语句。语法格式如下：

〈变量 1〉 ⟷ 〈变量 2〉; /* 表示变量 1 和变量 2 互换 */

(4) 条件选择语句

1) if 语句。语法格式如下：

```
if(〈表达式〉)  语句;
```

2) if-else 语句。语法格式如下：

```
if(〈表达式〉)
```

语句1;
else
语句2;

3）switch 语句。语法格式如下：

```
switch(〈表达式〉)
{  case  判断值1:   语句组1;   break;
   case  判断值2:   语句组2;   break;
         ……
   case  判断值n:   语句组n;   break;
  [default: 语句组;   ]
}
```

注意：switch 语句是先计算表达式的值，然后用其值与判断值相比较，若它们相一致时，就执行相应的 case 后的语句组；若都不一致，则执行 default 后的语句组。其中的方括号代表可选项。

（5）循环语句

1）for 语句。语法格式如下：

```
for(〈表达式1〉;〈表达式2〉;〈表达式3〉)
    {循环体语句;}
```

首先计算表达式 1 的值；然后求表达式 2 的值，若结果非零则执行循环体语句；最后对表达式 3 运算。如此循环，直到表达式 2 的值为零时止。

2）while 语句。语法格式如下：

```
while(〈条件表达式〉)
      {循环体语句;}
```

while 循环首先计算条件表达式的值，若条件表达式的值非零，则执行循环体语句；然后再次计算条件表达式，重复执行，直到条件表达式的值为零时退出循环，执行该循环之后的语句。

3）do-while 语句。语法格式如下：

```
do {循环体语句;
   } while(〈条件表达式〉)
```

该循环语句首先执行循环体语句，然后再计算条件表达式的值，若条件表达式成立，则再次执行循环体，再计算条件表达式的值，直到条件表达式的值为零，即条件不成立时结束循环。

（6）输入、输出语句

1）输入语句：用 scanf 函数实现。当数据为字符时，也可用 getchar 函数实现。

2）输出语句：用 printf 函数实现。当输出字符数据时，也可用 putchar 函数实现。

（7）其他一些语句

1）return 表达式或 return：用于函数结束。

2）break 语句：可用在循环语句或 case 语句中结束循环过程或跳出情况语句。

3）exit 语句：表示出现异常情况时，控制退出语句。

（8）注释形式 可用“/* 字符串 */”或“// 文字序列”或者单行注释对语句进行注释。

（9）一些基本的函数

1）max 函数：用于求一个或几个表达式中的最大值。

2）min 函数：用于求一个或几个表达式中的最小值。

3）abs 函数：用于求表达式的绝对值。

4）eof 函数：用于判定文件是否结束。

5）eoln 函数：用于判断文本行是否结束。

1.3.3 算法性能分析与度量

评价一个好的算法有以下几个标准：

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。

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。

3）健壮性。算法应具有容错处理，当输入非法数据时，应能作适当处理，不至引起严重后果。

4）效率与存储量需求。效率指的是算法执行的时间；存储量需求是指算法执行过程中所需要的最大存储空间。一般，这两者与问题的规模有关。一个好的算法应当具有较高的时间效率，并且可以有效使用存储空间。

一个算法可以从时间复杂度与空间复杂度来评价其优劣。

当将一个算法转换成程序并在计算机上执行时，其运行所需要的时间取决于下列因素：

1）硬件的速度。

2）书写程序的语言。

3）编译程序所生成目标代码的质量。

4）问题的规模。

显然，在各种因素都不能确定的情况下，很难比较出算法的执行时间。也就是说，使用执行算法的绝对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，这样一个特定算法的运行工作量的大小就只依赖于问题的规模（通常用正整数 n 表示），或者说它是问题规模的函数。

1. 时间复杂度

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

为了便于比较同一问题的不同算法，通常的做法是从算法中选取一种对于所研究的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规模 n 的某个函数 T(n)。

许多时候，要精确地计算 T(n)是困难的，因此引入渐进时间复杂度，在数量上估计一个算法的执行时间，也能够达到分析算法的目的。

定义（大 O 记号）：如果存在两个正常数 c 和 n_0，使得对所有的 $n \geqslant n_0$，有

$$f(n) \leqslant cg(n_0)$$

则记

$$f(n) = O(g(n_0))$$