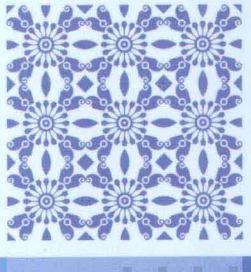
HZ BOOKS
华章教育

高等院校计算机教材系列

数据结构
C++语言描述

苏仕华 刘燕君 刘振安 编著
中国科学技术大学

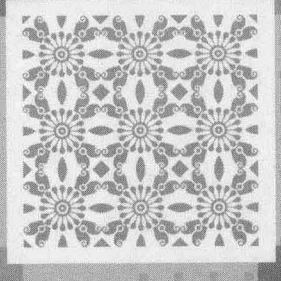


机械工业出版社
China Machine Press

高等院校计算机教材系列

数据结构
C++语言描述

苏仕华 刘燕君 刘振安 编著
中国科学技术大学



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

数据结构：C++语言描述 /苏仕华，刘燕君，刘振安编著. —北京：机械工业出版社，2014.1
（高等院校计算机教材系列）

ISBN 978-7-111-44926-3

Ⅰ. 数… Ⅱ. ①苏… ②刘… ③刘… 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

中国版本图书馆CIP数据核字（2013）第284094号

本书使用模板描述算法，实现参数化类型，使得对算法的描述更接近自然语言和更容易理解。另外，书中还精选了典型例题、实验和习题，并有配套的课程设计，帮助学生进一步加深对算法的理解。同时，为了方便读者考研，本书还在附录部分给出了考研指导，并提供了一些复习方法、考试技巧以及真题练习和参考答案，指导读者复习并深入掌握相关知识。

本书取材新颖、结构合理、概念清楚、语言简洁、通俗易懂、实用性强，重在培养学生对各种基本算法的理解和应用技能，特别适合作为高等院校相关专业的教材，也可以作为培训班教材、自学教材及工程技术人员的参考书。

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：刘立卿
北京诚信伟业印刷有限公司印刷
2014年1月第1版第1次印刷
185mm×260mm·17.5印张
标准书号：ISBN 978-7-111-44926-3
定 价：35.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

数据结构不仅是计算机软件和计算机应用专业的核心课程，也是很多电子信息类专业举足轻重的基础课程，它已经成为很多专业的考研内容，所以学好本课程至关重要。

数据结构课程早期使用 Basic 语言描述，后来使用 Pascal 语言描述，Pascal 语言曾经占据很长时间。如今由于 C 语言和 C ++ 语言的迅猛发展，又开拓了新的描述方法。

本书作者一直从事数据结构的教学，编写过使用不同语言的数据结构教材，而且在教学中不断探索，一直想找到更好的描述方法——希望不受数据类型的限制，能更方便地描述数据结构。但 C 语言限定不同数据类型的数据不能混用，所以在使用 C 语言描述算法时，必须针对具体的数据类型（如整数、实数和字符等），这就给算法描述带来很多不便。C ++ 语言虽然有类，但类也与数据类型有关。而 C ++ 语言提供的模板则很好地解决了这一难题。

使用模板描述算法，既接近自然语言，又不需要考虑具体的数据类型，从而可以把精力集中在对算法的描述上。因此本教材尝试使用模板描述算法，并希望能起到抛砖引玉的作用。本教材有如下几个特点：

- 使用模板描述算法，实现参数化类型，使得对算法的描述更接近自然语言，更容易理解。
- 使用类类型来定义和处理数据，实现数据共享，简化算法设计。
- 例题典型，且配有实验和习题，有助于对课文的理解。
- 全书的例题均配有验证程序，并提供全部源程序及 PPT，以方便学习和教学。
- 为帮助学生学习，编写了与教材配套的课程设计。课程设计的章与教材对应，并给出该章学习的重点和难点，配有典型例题、实验解答和课程设计题目。
- 为帮助学生考研，专门在附录部分提供了考研指导帮助复习。

本书编写采取分工负责、集体讨论的方式。苏仕华执笔第 1、6、8、10 章及附录，刘燕君执笔第 3、4、5、7、9 章，刘振安执笔第 2、11 章并负责统稿。编写本书时，正值刘燕君老师去亚洲大学做博士后研究工作，得到导师逢甲大学张真诚教授及亚洲大学资讯学院黄明祥院长的支持，他才得以完成所承担的写作任务，在此表示衷心感谢。中国科学技术大学软件学院院长陈国良院士及南京大学计算机系陈本林教授在百忙之中审阅了书稿并提出一些宝贵意见，特此感谢。写作中还参考了大量资料，这些资料有的列入参考文献中，还有些没有列入，在此对这些作者表示感谢。

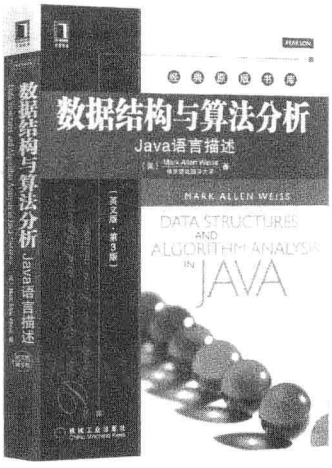
由于我们才疏学浅，本书中的不妥之处在所难免，敬请读者不吝赐教，给予指正为盼。

联系方式：zaliu@ ustc. edu. cn。

作者

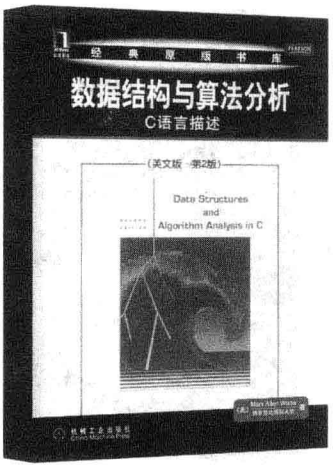
2013 年 8 月于中国科学技术大学

推荐阅读



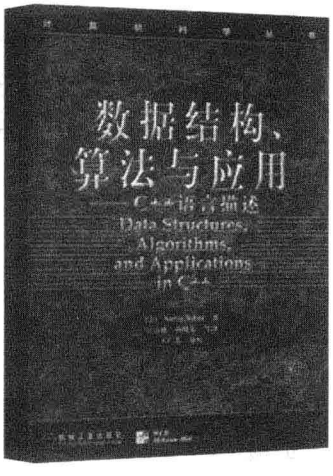
数据结构与算法分析：Java语言描述（英文版·第3版）

作者：Mark Allen Weiss ISBN：978-7-111-41236-6 定价：79.00元



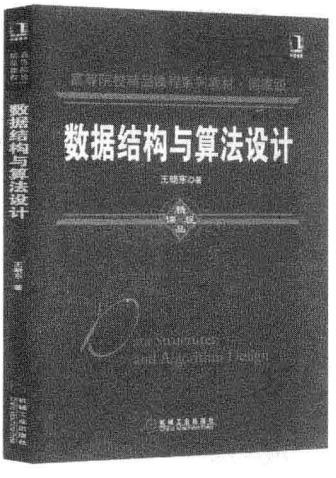
数据结构与算法分析：C语言描述（英文版·第2版）

作者：Mark Allen Weiss ISBN：978-7-111-31280-2 定价：45.00元



数据结构、算法与应用：C++语言描述

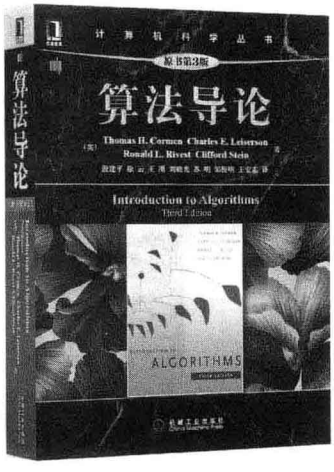
作者：Sartej Sahni ISBN：7-111-07645-1 定价：49.00元



数据结构与算法设计

作者：王晓东 ISBN：978-7-111-37924-9 定价：29.00元

推荐阅读

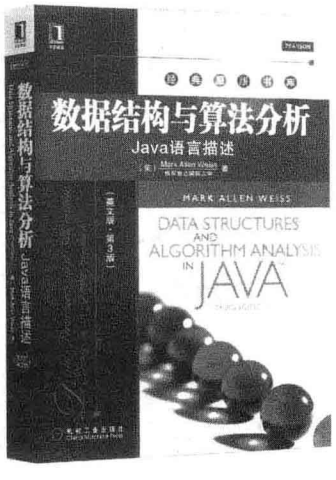


算法导论（原书第3版）

作者：Thomas H.Cormen 等 ISBN：978-7-111-40701-0 定价：128.00元

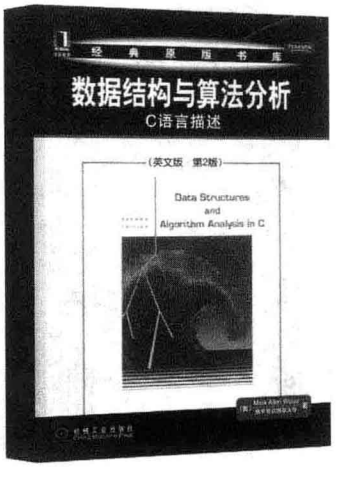
C程序设计导引

作者：尹宝林 ISBN：978-7-111-41891-7 定价：35.00元



数据结构与算法分析
——Java语言描述（英文版·第3版）

作者：Mark Allen Weiss ISBN：978-7-111-41236-6 定价：79.00元



数据结构与算法分析
——C语言描述(英文版·第2版)

作者：Mark Allen Weiss ISBN：978-7-111-31280-2 定价：45.00元

目　录

第 1 章　数据结构概论

数据结构是计算机软件和计算机应用专业的核心课程。为方便数据结构课程的学习，本章首先介绍数据结构的基本概念和常用术语，以及算法描述和分析的基础知识。

1.1　引言

自 1946 年世界上第一台计算机问世以来，计算机科学和技术都得到了飞速的发展，与此同时，计算机的应用也从最初的科学计算逐步发展渗透到人类社会的各个领域。计算机的处理对象不仅仅是简单的数字，现已发展到包括字符、表格、图形、图像、声音等各种非数值数据。因此，要开发出一种结构合理、性能良好的软件，编写一个“好”的程序，不仅需要至少掌握一种合适的计算机高级语言及其相应的软件开发工具，还必须会分析待处理对象的特性以及对象之间存在的关系，这就是“数据结构”这门学科形成和发展的背景。

例如，从一个实际问题的提出到使用计算机解出答案，一般需要经过以下典型步骤：

①给出一个实际问题。

②根据问题抽象出数学模型。

③给出求解数学模型的算法。

④根据算法进行编程和调试。

⑤可能还要根据实际情况调整算法。

⑥给出答案。

因此，要设计出一个“好”的程序，必须有一个“好”的算法，而好的算法必须建立在研究数据的特性以及数据之间存在的关系基础上。

数据结构是计算机软件和计算机应用专业的核心课程，在众多的计算机系统软件和应用软件中要用到各种数据结构。因此，仅掌握几种计算机语言是难以应付众多复杂的课题的，要想有效地使用计算机，还必须学习数据结构的有关知识。

到底什么是数据结构呢？先看一个例子。以计算机管理图书目录的问题为例，当读者想借阅一本书，书库中是否有这本书，或不知道图书馆中有哪些书时，就需要到图书馆去查阅如表 1-1所示的目录卡片。

表 1-1　图书目录卡片

登录号	书　名	作　者	出版社	出版时间	分类号
01771778	数据结构	刘晓阳	高等教育	2000. 08	73. 961162
01509429	操作系统	许海平	中国科学	1999. 12	73. 752196
00592056	数据库原理	孙华英	人民邮电	2001. 05	73. 323265
01267435	软件工程	陈大鹏	清华大学	1998. 11	73. 561238

卡片有的是按书名编排的，有的则是按作者或分类号等编排的。若计算机处理上述检索问题，列在每一张卡片上的一本书的书目信息一般包括登录号、书名、作者、分类号、出版社和出版时间等项，其中登录号是唯一的。可按登录号、书名、作者等建立相应的索引表。这些表

构成的文件就是图书目录检索的数学模型。计算机的主要操作就是按某个特定要求（如书名、作者）对书目文档进行查询。诸如此类的还有查号系统、仓库系统管理、账务处理等。这些都是最简单的线性关系，所对应的数学模型称为线性数据结构。

再例如家族的血统关系、博弈树问题（人－机下棋）、计算机的文件系统等都是树形结构；城市之间的交通网络、多岔路口交通灯的管理问题等都是图结构。它们都是非线性结构。

从上面的例子可见，描述这类非数值计算问题的数学模型不再是数学方程，而是诸如表、树和图之类的数据结构。因此，简单地说，**数据结构**是研究非数值计算问题，涉及程序设计中的计算机的操作对象和操作方法以及操作对象之间的关系。

具体地说，数据结构指的是数据元素之间的逻辑结构、存储结构及其数据的抽象运算，即按某种逻辑关系组织起来的一批数据，按一定的存储表示方式把它们存储在计算机的存储器中，并在这些数据上定义一个运算的集合。

1.2 基本概念和常用术语

1. 数据结构研究的内容

（1）数据

数据是描述客观事物的数值、字符以及能输入计算机中并被计算机处理的符号的集合。对计算机科学而言，数据是抽象化了的数据，包括数字、字符、图形、图像、声音等。

（2）数据元素

数据元素是数据的基本单位。如前例目录卡片中的一张卡片（表格中的一行）、树中的一个结点、图的顶点等都是数据元素。有时一个数据元素可由若干个数据项（也称为字段、域、属性）组成，数据项是具有独立含义的最小标识单位，如目录卡片中的登录号、书名、作者等。

（3）数据对象

数据对象是具有相同特性的数据元素的集合，是数据的一个子集。

（4）结构

结构指的是数据元素之间的相互关系，即数据的组织形式。

（5）结点

结构中的数据元素称为结点。

（6）数据结构

对数据结构的概念，至今还没有一个标准的定义，所以只能从研究的领域来理解数据结构。数据结构是指带有结构的数据元素的集合，描述的是数据之间的相互关系，即数据的组织形式。数据结构一般包括以下三个方面的内容：

①数据元素之间的逻辑关系，也称为数据的逻辑结构。

②数据元素及其关系在计算机存储器内的表示，称为数据的存储结构。

③数据的运算，即对数据施加的操作。

2. 数据的逻辑结构

数据的逻辑结构是从逻辑关系上描述数据的，它与数据元素的存储结构无关，是独立于计算机的。因此，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。例如，表1-1中所表示的图书目录卡片，表中数据元素之间的逻辑关系就是一种相邻关系：对表中任一个结点，与它相邻且在它前面的结点称为**直接前驱**，这种直接前驱最多只有一个；与表中任一结点相邻且在其后面的结点称为**直接后继**，直接后继也最多只有一个。表中只有第一个结点没有直

接前驱，称为**开始结点**；也只有最后一个结点没有直接后继，称为**终端结点**。例如，表 1-1 中的“操作系统”所在结点的直接前驱结点和直接后继结点分别是“数据结构”和“数据库原理”所在的结点，这种结点之间的关系就构成了图书目录卡片表的逻辑结构。

数据的逻辑结构有如下两大类。

（1）线性结构

线性结构的特征是：有且仅有一个开始结点和一个终端结点，并且所有结点都最多只有一个直接前驱和一个直接后继。线性表是一个典型的线性结构。本书第 3 ~ 5 章都是介绍线性结构的。

（2）非线性结构

非线性结构的特征是：一个结点可能有多个直接前驱和直接后继，如广义表、树和图。本书第 6 ~ 8 章讨论的数据结构都是非线性结构。

3. 数据的存储结构

数据的存储结构是逻辑结构用计算机语言的实现（亦称为映像），它是依赖于计算机语言的。

数据的存储结构可以用以下 4 种基本的存储方法实现。

（1）顺序存储方法

顺序存储方法是把逻辑上相邻的结点存储在物理位置相邻的存储单元里。由此得到的存储结构称为顺序存储结构。这通常是借助于程序设计语言的数据类型描述的。该方法主要应用于线性数据结构，非线性数据结构也可通过某种线性化的方法来实现顺序存储。

（2）链式存储方法

链式存储方法是用一组不一定连续的存储单元存储逻辑上相邻的结点，结点间的逻辑关系是由附加的指针域来表示的。由此得到的存储表示称为链式存储结构。

（3）索引存储方法

索引存储方法通常是在存储结点信息的同时，还建立附加的索引表。表中索引项的一般形式是含有关键字和地址，关键字是能唯一标识一个结点的数据项。

（4）散列存储方法

散列存储方法的基本思想是根据结点的关键字直接计算出该结点的存储地址。

同一种逻辑结构采用不同的存储方法，可以得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构，要视具体的应用系统要求而定，而主要考虑的还是运算方便及算法的时间和空间上的要求。

无论怎样定义数据结构，都应该将数据的逻辑结构、数据的存储结构及数据的运算这三方面看成一个整体。因此，存储结构是数据结构不可缺少的一个方面。

4. 数据的运算

数据的运算是定义在数据的逻辑结构上的，每种逻辑结构都有一个运算的集合，最常用的运算有检索、插入、删除、更新和排序等。

数据运算是数据结构不可分割的一个方面，在给定了数据的逻辑结构和存储结构之后，按定义的运算集合及其运算性质的不同，可能导致完全不同的数据结构。例如，若对线性表的插入、删除运算限制在表的一端进行，则该线性表称为栈；若对线性表的插入运算限制在表的一端，而删除运算限制在表的另一端，则该线性表称为队列。

5. 数据类型

数据类型（data type）是与数据结构密切相关的一个概念。所谓数据类型是一个值的集合

和定义在这个值集上的一组操作的总称。在使用高级程序设计语言编写的程序中，每个变量、常量或表达式都有一个它所属的数据类型。类型规定了在程序执行期间变量或表达式可能的取值范围以及在这些值上所允许的操作运算。例如，C++语言中的整数类型，就给出了一个整型量的取值范围（依赖于不同的机器或编译系统），定义了对整型量可施加的加、减、乘、除和取模等算术运算。

在高级程序设计语言中，按“值”的不同特性，可将数据类型分为两类：一类是其值不可分解，称为**原子类型**（或非结构类型），例如C++语言中的基本类型（整型、实型、字符型）以及指针类型和空类型等简单类型；另一类则是**结构类型**，其值可由若干个分量（或成分）按某种结构组成，它的分量可以是非结构型的，也可以是结构型的，例如C++语言中的数组、结构等类型。通常数据类型可以看做程序设计语言中已实现的数据结构。

6. 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是20世纪70年代提出的一种概念，它是抽象数据的组织和与之相关的操作。一个ADT可以看做定义了相关操作运算的一个数学模型。例如，集合与集合的并、交、差运算就可以定义为一个抽象数据类型。

抽象数据类型可以看做描述问题的模型，它独立于具体实现，其特点是将数据定义和数据操作封装在一起，使得用户程序只能通过在ADT中定义的某种操作来访问其中的数据，从而实现了信息的隐藏。这种抽象数据类型类似于C++中的类定义。

作为一个例子，看一个“圆”数据类型的描述。我们知道，要表示一个圆，一般应包括圆心的位置和半径的大小。如果只关心圆的面积，那么这个抽象数据类型中就只需要有表示半径的数据。假设要设计一个圆（Circle）的抽象数据类型，它包括计算面积（Area）、周长（Circumference）的操作，Circle的抽象数据类型描述如下。

```
ADT Circle {
    Data
    非负实数表示圆的半径
    Operations
        Constructor
            输入的初值：非负实数
            处理：给半径赋初值
        Area
            输入：无
            处理：计算圆面积
            输出：圆的面积
        Circumference
            输入：无
            处理：计算圆周长
            输出：圆周长
} //ADT Circle
```

由于本书是以C++语言为基础来描述算法的，而C++语言中提供“类”这一数据类型，所以可以实现抽象数据类型，因此也将采用ADT的形式来描述数据结构。

其实，读者只需要记住，ADT实际上等价于定义的数据的逻辑结构以及在逻辑结构上定义

的抽象操作。

1.3 算法的描述和分析

研究数据结构的目的在于更好地进行程序设计，而程序设计离不开数据的运算，通常将这种运算的过程（或解题的方法）称为算法。

假设有3个坐标点 $a(x1, y1)$、$b(x2, y2)$、$c(x3, y3)$，现在要用计算机求解它们所构成的三角形的面积。这时需要使用求解三角形面积的相关计算公式（抽象出数学模型），然后再根据公式逐步求解计算。假设选取如下公式：

$$\text{Area} = \sqrt{s(s-ab)(s-ac)(s-bc)}$$

来计算三角形的面积，则对应边长和半周长 s 的计算公式为：

$$ab = \sqrt{(x1-x2)^2 + (y1-y2)^2}$$

$$ac = \sqrt{(x1-x3)^2 + (y1-y3)^2}$$

$$bc = \sqrt{(x2-x3)^2 + (y2-y3)^2}$$

$$s = \frac{ab + ac + bc}{2}$$

有了这些公式（模型）之后，就可以给出求解问题的过程（又叫解题的方法或步骤），这就是所谓的算法。该问题的算法描述如下。

①输入三角形的3个坐标点 a、b 和 c；

②计算三条边长及半周长；

③计算三角形的面积 Area；

④输出三角形的面积。

然后再根据算法的描述，编写相应的程序代码并上机调试运行，直至得出正确结果。

从这个例子可以看出，算法是对问题求解步骤的一种描述。

1.3.1 算法描述

数据的运算是通过算法描述的。通俗地说：一个算法就是一种解题的方法。更严格地说，算法是由若干条指令组成的有穷序列，其中每条指令表示一个或多个操作。它必须满足以下五个准则。

①**输入**。算法开始前必须对算法中用到的变量初始化。一个算法的输入可以包含零个或多个数据。

②**输出**。算法至少有一个或多个输出。

③**有穷性**。算法中每一条指令的执行次数是有限的，即算法必须在执行有限步之后结束。

④**确定性**。算法中每一条指令的含义都必须明确，无二义性。

⑤**可行性**。算法是可行的，即算法中描述的操作都可以通过有限次的基本运算来实现。

因此，一个程序如果对任何输入都不会陷入无限循环，则它就是一个算法。

算法的含义与程序十分相似，但二者是有区别的。例如，一个程序就不一定满足有穷性。

一个算法可用自然语言、数学语言或约定的符号语言来描述，如类 Pascal 语言、类 C 语言和 C++ 语言等描述方法。目前流行的是使用类 C 语言和 C++ 语言描述算法。类 C 语言类似于 C 语言，但不完全是 C 语言。类 C 语言借助于 C 语言的语法结构，附之以自然语言的叙述，使得用它编写的算法既具有良好的结构，又不拘泥于具体程序语言的某些细节。虽然类 C 语言使得算法易读易写，但 C 语言没有类，所以还必须自己用 C 语言去实现，这也给学习带来不便。

C++语言不仅支持类，而且具有类模板，这就使描述更加接近自然语言。本书采用C++语言描述算法，第2章将介绍本书使用的类和类模板的基础知识，这里不再赘述。

1.3.2 算法分析

求解一个问题可能有多种不同的算法，而算法的好坏直接影响程序的执行效率，而且不同的算法之间的运行效率相差巨大。

【例1.1】 鸡兔同笼问题。

大约在1500年前，《孙子算经》中记载了一个有趣的问题。书中是这样叙述的："今有鸡兔同笼，上有三十五头，下有九十四足，问鸡兔各几何?"

解答思路是这样的：假如砍去每只鸡、每只兔一半的脚，则每只鸡就变成了"独脚鸡"，每只兔就变成了"双脚兔"。

由此可知：

①鸡和兔的脚的总数就由94只变成了47只（一只兔子2只脚，一只鸡1只脚）。

②假设笼子里只有一只兔子，这个兔子有2只脚和1个头，即脚的数目比头的数目多1。现在脚的总只数是47，总头数是35，47－35＝12。即脚的总数与总头数之差就是兔子的只数。

③知道兔子的只数，则鸡的只数为35－12＝23（只）。

这一思路新颖而奇特，其"砍足法"也令古今中外数学家赞叹不已。这种思维方法叫化归法。化归法就是在解决问题时，先不对问题采取直接的分析，而是将题中的条件或问题进行变形，使之转化，直到最终把它归成某个已经解决的问题。

下面使用计算机来求解鸡兔同笼问题。

设鸡为i只，兔为j只。则有：

$$\begin{cases} i + j = 35 \\ 2i + 4j = 94 \end{cases}$$

使用i和j分别表示两层循环，逐次枚举试验，当满足上述条件时，就可求出鸡有i只，兔有j只。下面是按此思想编写的C++程序，sum表示执行循环的总次数。

```
// 鸡兔同笼—J11.cpp
void main()
{
    int sum=0;
    for( int i=1;i<35; i++)
    {
            sum++;
            for(int j=1;j<35;j++)
            {
                sum++;
                if((i+j==35)&&(2*i+4*j==94))
                        cout<<"鸡有"<<i<<"只,兔有"<<j<<"只"<<endl;
            }
    }
    cout<<"一共循环"<<sum<<"次。"<<endl;
}
```

程序运行结果如下：

```
鸡有23只,兔有12只。
一共循环1190次。
```

其实，第二个循环执行1156次。由此可见，这个循环次数很大，应该减少第二个循环的次数。如果将它改为“$j=35-i$”，则会降为595次。

通过分析鸡兔的如下关系，可以改进程序的效率：

①两只鸡和一只兔子的脚数相等，所以鸡头的数量不会超过三分之二，即$i<25$，$j<13$。

②给定一个i，j的初始应该是$35-i$。

```
// 改进的算法
void main()
{
    int sum=0;
    for( int i=1;i<24; i++)
    {
      sum++;
      for( int j=35-i;j<13;j++)
      {
            sum++;
            if((i+j==35)&&(2*i+4*j==94))
                        cout<<"鸡有"<<i<<"只,兔有"<<j<<"只"<<endl;
      }
    }
    cout<<"一共循环"<<sum<<"次。"<<endl;
}
```

程序运行结果如下：

```
鸡有23只,兔有12只。
一共循环24次。
```

其实，要等到$j=35-i<13$时，才进入第二个循环，而且仅执行1次。

求解一个问题可能有多种不同的算法，那么如何来评价这些算法的优劣好坏并从中选择好的算法呢？显然，算法的“正确性”是首先要考虑的。所谓一个算法的正确性，是指对于一切合法的输入数据，该算法经过有限时间的执行都能得到正确的结果。但是，不仅要考虑算法的正确性，还要考虑其他因素。一般主要考虑如下几点：

①执行算法所耗费的时间，即时间复杂性。

②执行算法所耗费的存储空间，主要是辅助空间，即空间复杂性。

③算法应易于理解、易于编程、易于调试等，即可读性和可操作性。

在这几点当中，最主要的还是时间复杂性。一个算法所耗费的时间，应是该算法中每条语句的执行时间之和。每条语句的执行次数又称为**频度**，所以每条语句的执行时间就是该语句的执行次数与该语句执行一次所需时间的乘积。

由于不同的计算机一条语句执行一次所需要的时间千差万别，无法用一个统一的标准来计算，因此，一个算法的时间耗费往往就用该算法中所有语句的频度之和来表示。

【例1.2】 求两个n阶矩阵的乘积$\boldsymbol{C}=\boldsymbol{A}\times\boldsymbol{B}$。

下面是该算法的基本操作部分，为了便于分析，用序号标注语句（后面不再说明）。

```
for(i=1;i<=n;i++)                                          // ①
      for(j=1;j<=n;j++){                                   // ②
         c[i][j]=0;                                        // ③
         for(k=1;k<=n;k++)                                 // ④
           c[i][j]=c[i][j]+a[i][k]* b[k][j];}              // ⑤
```

语句①的循环控制变量 i 要增加到 $n+1$，测试 $i=n+1$ 成立时，循环才会终止，因此它的频度为 $n+1$，但它的循环体却只能执行 n 次。语句②作为①的循环内语句应执行 n 次，但语句②本身要执行 $n+1$ 次，所以②的频度为 $n(n+1)$，同理可得③、④、⑤的频度分别为 n^2、$n^2(n+1)$、n^3。

该算法中所有语句的频度之和，即运行时间为

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

当 n 足够大时，$T(n)$ 与 n^3 之比是一个不等于零的常数，则称 $T(n)$ 和 n^3 是同阶的，记为 $T(n)=O(n^3)$。一般情况下，算法中基本操作重复执行的次数是问题规模的某个函数 $f(n)$，因此，算法的时间度量记作

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

也称为该算法的渐近时间复杂度，简称时间复杂度。$f(n)$ 一般为算法中频度最大的语句频度。

对于较复杂的算法，可以将它分成几个容易估算的部分，然后利用“O”的求和原则和乘法原则计算整个算法的时间复杂度。

①大“O”下的求和准则。若算法的两部分的时间复杂度为

$$T_1(n) = O(f(n))$$

和

$$T_2(n) = O(g(n))$$

则 $T_1 + T_2 = O(max(f(n), g(n)))$。

又若

$$T_1(m) = O(f(m))$$

和

$$T_2(n) = O(g(n))$$

则 $T_1 + T_2 = O(f(m) + g(n))$。

②大“O”下的乘法准则。若算法的两部分的时间复杂度为

$$T_1(n) = O(f(n))$$

和

$$T_2(n) = O(g(n))$$

则 $T_1 \cdot T_2 = O(f(n) \cdot g(n))$。

【例1.3】 分析计算下面程序段的时间复杂度。

```
s=0;                                    // ①
  for (i=1;i<=n;i++)                    // ②
     for (j=1;j<=n;j++)                 // ③
        s=s+1;                          // ④
```

【分析】 按常规方法求算法时间复杂度，该算法的①行频度为1；②、③、④行的频度分别为 $n+1$、$n(n+1)$、n^2，因此算法中所有频度之和为

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

即 $f(n)=n^2$，所以，该算法的时间复杂度为 $T(n)=O(f(n))=O(n^2)$。

因为执行一条赋值语句与 n 无关，时间复杂度为 $O(1)$，因此 $T_4(n)=O(1)$，对于第③条语句，$T_3(n)=O(n)$，如果利用上面的求和准则，有 $T_4(n)+T_3(n)=O(n)$，第②条语句 $T_2(n)=O(n)$，而②与③、④是循环嵌套，故有 $T_2(n)*(T_3(n)+T_4(n))=O(n^2)$，对第①条语句，$T_1(n)=O(1)$，它与其他语句之间的关系是顺序执行，所以该程序段的算法时间复杂