HZ BOOKS
华章教育

本书可用于
ACM程序设计竞赛、各类大学生程序设计竞赛、
在线程序设计竞赛以及中学生信息学奥林匹克竞赛
的训练，全面提升程序设计能力

数据结构编程实验

大学程序设计课程与竞赛训练教材
第2版

Data Structure Practice

for Collegiate Programming Contests and Education
Second Edition

吴永辉 王建德 编著

机械工业出版社
China Machine Press

数据结构编程实验

大学程序设计课程与竞赛训练教材
第2版

Data Structure Practice

for Collegiate Programming Contests and Education
Second Edition

吴永辉 王建德 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

数据结构编程实验：大学程序设计课程与竞赛训练教材 / 吴永辉，王建德编著．—2 版．—北京：机械工业出版社，2016.9

ISBN 978-7-111-55055-6

I. 数… II. ① 吴… ② 王… III. 数据结构 - 程序设计 - 高等学校 - 教材 IV. TP311.12

中国版本图书馆 CIP 数据核字（2016）第 238492 号

数据结构编程实验

大学程序设计课程与竞赛训练教材　第 2 版

出版发行：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：佘　洁　　责任校对：董纪丽

印　　刷：三河市宏图印务有限公司　　版　　次：2016 年 10 月第 2 版第 1 次印刷

开　　本：185mm × 260mm　1/16　　印　　张：33

书　　号：ISBN 978-7-111-55055-6　　定　　价：79.00 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）88379426　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

前　言

我们长期从事数据结构、算法的教学和程序设计竞赛的训练，教学和训练的实践使我们产生了对程序设计、数据结构和算法的教学模式进行改革的想法。

1）在课程中增加思维方式和解题策略的引导，引导学生思考各类数据结构、算法的本质特征是什么，将“知识体系结构的掌握”与“思维方式的训练”结合起来。

我们认为，评价一个人的专业能力要看以下两个方面：①知识体系结构。他能用哪些知识解决问题，或者说，哪些是他所真正掌握并能应用而不仅仅是他学过的知识？②思维方式。在他面对问题，特别是不太标准化的问题的时候，他解决问题的策略是什么？比如，面对的问题为什么要采用这样的数据结构表示，而不宜采用那样的数据结构表示？当有多个数据结构可用时，怎样权衡时间复杂度、空间复杂度、编程复杂度和思维复杂度四个因素，寻找最合适的数据结构？等等。

2）在课程和训练中全部采用程序设计竞赛的试题作为案例来进行教学。

传统教学着重于理论教学和笔试，可能会使学生浑然不知程序设计语言、数据结构、算法在现实生活中究竟有什么用处，也就失去了学习的意义。

陆游诗云：“纸上得来终觉浅，绝知此事要躬行。”案例教学是通过模拟或者重现现实生活中的一些场景，让学生把自己置入问题情境之中，通过思考、讨论和上机编程来进行学习的方式：学生拿到试题后，先进行审题（What to do），然后考虑如何采用数据结构和算法来解决问题（How to do），这无形中激发了学生的求知欲望，加深了学生对知识的理解；在给出了通过编程解决问题的方法后，学生还要经过编程，将解决方法变为解决问题的程序，并进行调试，在允许的时间和空间范围内通过测试用例（Implementation）。这个“认识－实践－再认识－再实践”的过程，应视为知识理解上的一种提高，知识学习与应用能力间的一种转变和升华。

程序设计竞赛是通过编程解决问题的比赛。从 20 世纪 80 年代末期到现在，各类大学生程序设计竞赛、各种在线比赛以及中学生信息学奥林匹克竞赛构成了浩如烟海的试题库。这些来自全球各地且凝聚了无数命题者的心血和智慧的试题，为相关知识创设了丰富有趣的问题背景，而且针对这些试题有许多在线测试平台，学生可以通过这些平台测试自己编写的程序。因此，这些试题不仅可以用于程序设计竞赛选手的训练，而且可以用于程序设计语言、数据结构、算法的教学和实验，能够系统、全面地提高学生通过编程解决问题的能力。

3）从程序设计的本质上说，程序设计是技术。

正因为程序设计是技术，所以，要磨炼学生的程序设计能力。首先是要求学生练习、练习、再练习；其次是要安排学生系统地练习。程序设计的知识体系可以概括为“算法＋数据结构＝程序”这一公式，这也是计算机科学与技术的知识体系结构的核心。所谓系统地

练习，就是在试题及其测试数据、解答程序以及解题分析的引导之下，通过解题系统地磨炼学生应用算法和数据结构各个知识点解决实际问题的能力，从而有效地掌握程序设计的知识体系。

基于上述想法，我们规划了“大学程序设计课程与竞赛训练教材”系列，并于2012年出版了该系列的第一本著作《数据结构编程实验：大学程序设计课程与竞赛训练教材》。随后，在中国台湾，由碁峰出版了该书的繁体版《提升程式設計的資料結構力：國際程式設計競賽之資料結構原理、題型、解題技巧與重點解析》；在美国，由CRC Press出版了该书的英文版《Data Structure Practice : for Collegiate Programming Contests and Education》，并在全球发行。目前，该书在境内外广受读者的欢迎和好评。

在此基础上，我们根据境内外的同学和同行在使用该书的过程中提出的意见及建议，以及计算机科学技术和程序设计竞赛的发展，我们这几年在阿曼、中国台湾和香港、美国、马来西亚、孟加拉国等国家和地区的讲学和访学工作，对该书进行了“脱胎换骨”的改进，最终出版了《数据结构编程实验：大学程序设计课程与竞赛训练教材》第2版。

本书根据数据结构的知识体系结构，按照循序渐进的原则，分四大篇（历练基本编程能力、线性数据结构的编程实验、树的编程实验、图的编程实验）15章组织内容。每一章在介绍了相关的数据结构知识后，给出了相应的实验范例，并在最后一节给出相关题库。

相应于本书的第1版，我们除了修正原有的小错误、笔误以及改进了一些表述外，较大的改进如下：在第一篇的第3章中，由原来的“简单递归的编程实验”完善为“递归与回溯的编程实验”。在第二篇的第4章中，将“在数组中快速查找指定元素”和“通过数组分块技术优化算法”的实验范例融合到原有的内容中；在第5章的队列编程实验中，完整地给出了三种队列形式即顺序队列、优先队列、双端队列的实验范例。在第三篇的第8章中，加入了“用四叉树求解二维空间问题”的实验范例；在第10章，在已有的二叉排序树和二叉堆的编程实验基础上，给出兼具二叉排序树和二叉堆性质的树堆的实验范例。在第四篇中，加入第15章“应用状态空间搜索编程”。

我们对浩如烟海的ACM程序设计竞赛区预赛、总决赛、各大学程序设计竞赛、在线程序设计竞赛以及中学生信息学奥林匹克竞赛的试题进行了分析和整理，从中精选出227道试题作为书中内容。其中88道试题作为实验范例，每道试题不仅有详尽的解析，还给出带有详细注释的参考程序；139道作为题库试题，所有试题都有清晰的提示，同时对第1版的读者反映的题库中一些“看了提示也很难编程”的较难的试题给出了带详尽注解的解答程序（或者直接给出，或者作为电子版随试题原版给出）。

书中每道题都注明了试题来源和在线测试网址，学生可以通过在线测试平台测试自己编写的程序。

本书的网站（www.hzbook.com）提供了所有试题的英文原版以及大部分试题的官方测试数据和解答程序。

本书既可以用于大学程序设计课程的教学和实验，又可以用于程序设计竞赛的训练。对于本书，我们的使用建议是：书中每章的实验范例可以用于程序设计语言、数据结构课程的教学、实验和上机作业，以及程序设计竞赛选手掌握知识点的入门训练；而在每章最后给出的“相关题库”中的试题则可以作为程序设计竞赛选手的专项训练试题，以及学生进一步

提高编程能力的练习题。根据本书第 1 版的使用情况，本书也非常适合学生自学，在知识背景、试题、测试数据、解题分析或提示的支持之下，即使没有老师、没有同伴，同学们也能够系统、全面地提高自己的编程能力。

本书是在复旦大学程序设计集训队长期活动的基础上积累而成的。

感谢复旦大学计算机学院 2006 级、2007 级、2008 级同学，他们在使用本书第 1 版的讲义稿过程中提出了宝贵的意见和建议。

感谢阿拉法特·居来提、姚哲云、张昊同学，他们编写了本书第 1 版的所有程序。

感谢使用本书第 1 版的读者们，他们提出的宝贵意见和建议对第 2 版和英文版的出版贡献巨大。

感谢 Stony Brook University 的 Steven Skiena 教授、Rezaul Chowdhury 教授，Texas State University 的 C. Jinshong Hwang 教授、Ziliang Zong 教授和 Hongchi Shi 教授，German University of Technology in Oman 的 Rudolf Fleischer 教授，以及 North South University 的 Abul L. Haque 教授和 Shazzad Hosain 教授，他们为本书英文版的试用和改进提供了平台。

感谢中国台湾、中国香港、孟加拉国、马来西亚邀请我讲学的李哲荣、廖宜恩、杨昌彪、林盈达、李淑敏、傅楸善、曹建农等教授和参加我的讲学的老师和同学，这几年，我们并肩作战，风雨同舟，而他们在使用本书第 2 版书稿的过程中也提出了宝贵意见和建议，为本书第 2 版的完成做出了不可或缺的贡献。

由于时间和水平所限，书中肯定夹杂了不少缺点和错误，表述不当和笔误也在所难免，热忱欢迎学术界同仁和读者赐正。如果您在阅读中发现了问题，恳请通过书信或电子邮件告诉我们，以便我们及时整理成勘误表放在本书的网站上，供广大读者查询更正。我们更期望读者对本书提出建设性意见，以便再版时改进。我们的联系方式如下：

通信地址：上海市邯郸路 220 号复旦大学计算机科学技术学院 吴永辉（邮编：200433）

电子邮件：yhwu@fudan.edu.cn

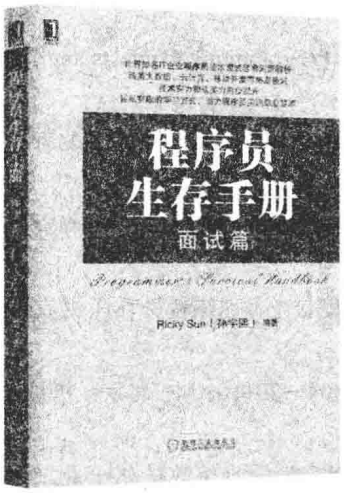
吴永辉　王建德

2016 年 6 月于上海

注：本书试题的在线测试地址如下：

在线评测系统	简称	网址
北京大学在线评测系统	POJ	http://poj.org/
浙江大学在线评测系统	ZOJ	http://acm.zju.edu.cn/onlinejudge/
UVA 在线评测系统	UVA	http://uva.onlinejudge.org/ http://livearchive.onlinejudge.org/
Ural 在线评测系统	Ural	http://acm.timus.ru/
SGU 在线评测系统	SGU	http://acm.sgu.ru/

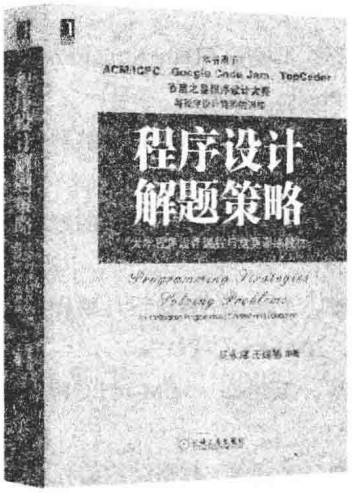
推荐阅读



程序员生存手册：面试篇

作者：Ricky Sun (孙宇熙)

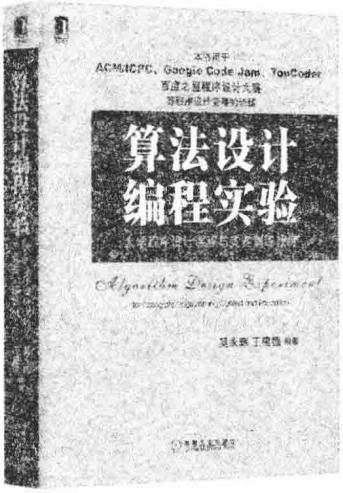
ISBN：978-7-111-52441-0 定价：69.00元



程序设计解题策略

作者：吴永辉 王建德

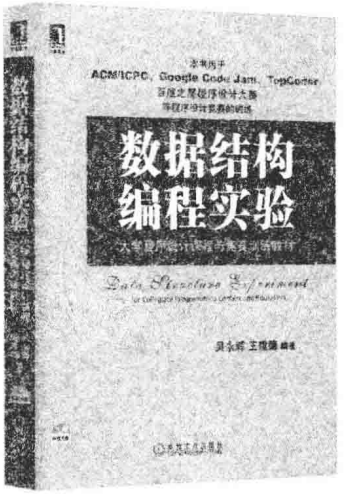
ISBN：978-7-111-48831-6 定价：79.00元



算法设计编程实验：大学程序设计课程与竞赛训练教材

作者：吴永辉 王建德

ISBN：978-7-111-42383-6 定价：69.00元



数据结构编程实验：大学程序设计课程与竞赛训练教材

作者：吴永辉 王建德

ISBN：978-7-111-37395-7 定价：59.00元

推荐阅读

作者：Kenneth H. Rosen
ISBN：978-7-111-38550-9
定价：99.00元

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

作者：Y. Daniel Liang
ISBN：978-7-111-42505-2
定价：79.00元

作者：Randal E. Bryant 等
ISBN：978-7-111-32631-1
定价：128.00元

作者：David A. Patterson 等
ISBN：978-7-111-45316-1
定价：139.00元

作者：Kevin R. Fall 等
ISBN：978-7-111-38228-7
定价：129.00元

作者：Abraham Silberschatz 等
ISBN：978-7-111-40086-8
定价：69.00元

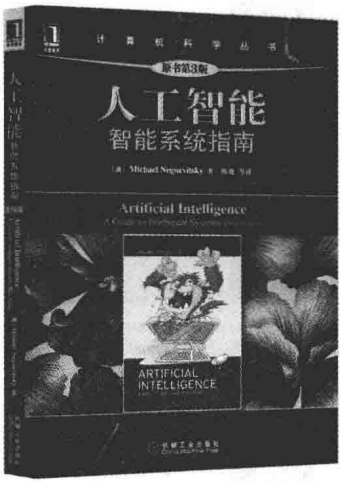
作者：Jiawei Han 等
ISBN：978-7-111-37431-2
定价：118.00元

作者：Roger S. Pressman 等
ISBN：978-7-111-48950-4
定价：119.00元

推荐阅读

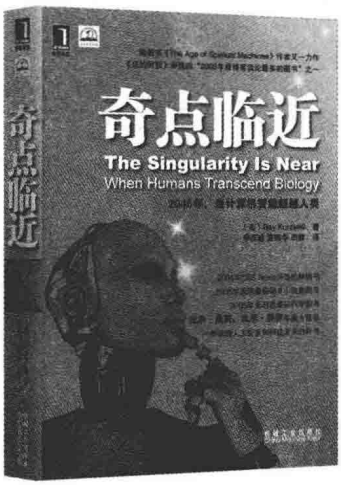
人工智能：计算Agent基础

作者：David L. Poole 等 ISBN：978-7-111-48457-8 定价：79.00元



人工智能：智能系统指南（原书第3版）

作者：Michael Negnevitsky ISBN：978-7-111-38455-7 定价：79.00元



奇点临近

作者：Ray Kurzweil ISBN：978-7-111-35889-3 定价：69.00元

机器学习

作者：Tom Mitchell ISBN：978-7-111-10993-7 定价：35.00元

目　录

第四篇 图的编程实验

第一篇

历练基本编程能力

程序设计是技术。正因为程序设计是技术，所以，程序设计、数据结构、算法这样的课程不是听会的，也不是看会的，而是练会的，是让学生在编程实践的过程中，逐步掌握通过编程解决实际问题的能力。数据结构的先导课程是程序设计语言，其教学目的是让学生学会用程序设计语言编写程序。因此，在学生进行数据结构编程实验之前，本篇先引领学生温故知新，进行如下三个方面的编程实验：

（1）简单计算

（2）简单模拟

（3）递归与回溯

这三类实验既是对程序设计语言课程的温习，也是数据结构编程实验课程的入门和基础。

第 1 章

简单计算的编程实验

所谓简单计算题，指的是在“输入－处理－输出”的模式中，“处理”这一环节所涉及的运算规则比较浅显，编程训练的重心放在如何正确地处理输入和输出，以及优化计算上。学生通过简单计算的编程实验，掌握 C/C++ 或 Java 程序设计语言的基本语法，熟悉在线测试系统和编程环境，初步学会怎样将一个自然语言描述的实际问题抽象成一个计算问题，给出计算过程，继而编程实现计算过程，并将计算结果还原成对原来问题的解答。

虽然简单计算题的运算相对简单，但还是应该持“举轻若重”的科学态度。因为试题的输入输出格式是多样的，而计算精度和时效一般有严格的定义。“细节决定成败”，一些编程细节若处理不好，则会导致整个程序“功亏一篑”。在本章，我们将在以下 4 个方面开展实验：

1）改进程序书写风格。

2）正确处理多个测试用例。

3）提高实数的计算精度。

4）用二分法提高计算效率。

一般来讲，较复杂的问题大都是由一些含简单计算的子问题组合而成的。“万丈高楼平地起”，要提高编程能力，就需要从训练简单计算题的编程能力做起。

1.1 改进程序书写风格的实验范例

如果一个程序具有良好的书写风格，不仅在视觉上给人以美感，而且也给程序的调试和检查带来方便。初看程序，往往可先从第一印象中判断出编程者思路是否清晰。但是，怎样的程序书写风格才算“好”呢？仁者见仁，智者见智。不过这并不意味着程序的书写风格“无章可循”。

【1.1.1 Financial Management】

Larry 今年毕业，找到了工作，并赚了很多钱。但不知为何，Larry 总感觉钱不够用。因此，Larry 要用财务报表来解决他的财务问题：他要计算他能用多少钱。现在可以通过 Larry 的银行账号看到他的财务状况。请你帮助 Larry 编写一个程序，根据过去 12 个月中他每个月的收入，计算若要达到收支平衡，每个月他平均能用多少钱。

输入

输入 12 行，每一行是一个月的收入，收入的数字是正数，精确到分，没有美元的符号。

输出

输出一个数字，是这 12 个月收入的平均值。精确到分，前面加美元的符号，后面加行结束符。在输出中没有空格或其他字符。

样例输入	样例输出
100.00 489.12 12454.12 1234.10 823.05 109.20 5.27 1542.25 839.18 83.99 1295.01 1.75	$1581.42

试题来源：ACM Mid-Atlantic 2001

在线测试地址：POJ 1004，ZOJ 1048，UVA 2362

试题解析

本题采用了非常简单的“输入－处理－输出”模式：

1）通过结构为 for(i=0; i<12; i++) 的循环输入 12 个月的收入 a[0..11]。

2）累计总收入 sum= $\sum_{i=0}^{11} a[i]$；计算月平均收入 avg= $\frac{\text{sum}}{12}$。

3）最后输出 avg。

参考程序

```
#include<iostream>                          //预编译命令
using namespace std;                        //使用 C++ 标准程序库中的所有标识符
int main()                                  //主函数
{                                           //主函数开始
    double avg, sum=0.0, a[12]={0};         //定义双精度实数变量 avg、sum 和实数数组 a 的初始值
    int i;                                  //声明整型变量 i
    for(i=0; i<12; i++){                    //依次读入每个月的收入，并累计年收入
            cin>>a[i];
            sum+=a[i];
        }
    avg=sum/12;                             //计算月平均收入
    printf("$%.2f", avg);                   //输出月平均收入
    return 0;
}
```

我们可从上述程序范例中得到 4 点启示：

1）严格按照题目要求的格式输入输出。本题要求输入的月收入是精确到分的正数，因此程序中用提取操作符“>>”，将键盘输入的月收入存储到双精度实数类型的数组元素 a[i] 中；同样，程序中采用了 printf(“$%.2f”, avg) 语句，使得输出的月平均收入精确到分，且前有美元符号，后有行结束符。您的程序运行于在线测试系统，关乎成败的首要因素是程序的输入输出格式是否符合题意。如果没有按照题目要求的格式输入输出，即便算法正确，结果也是“Wrong Answer”。

2）同一结构程序段内的所有语句（包括说明语句）与本结构程序段的首行左对齐。

3）程序行按逻辑深度呈锯齿状排列。例如循环体缩进几个字符，用缩进表示选择结构，等等。这种锯齿形的编排格式能够清晰地反映程序结构，改善易读性。

4）在程序段前或开始位置加上描述程序段功能的注释，对于变量及其变化也应该加上注释。这样做不仅是为了调试程序和日后阅读的方便，更重要的是培养一种团队合作的精神。将来你走上工作岗位后，可能参加一个研发团队，大家一起合作编程、互相协助，这就更需要将注释写得清清楚楚，让其他人看得明白。

1.2　正确处理多个测试用例的实验范例

【1.1.1　Financial Management】仅给出了一个测试用例，该测试用例中数据的个数是已知的（12个月的收入），且运算十分简单（累加月收入，计算月平均值）。但在通常情况下，为了全面检验程序的正确性，大多试题都要求测试多个测试用例，只有通过所有测试用例才算程序正确。如果测试用例的个数或每个测试用例中数据的个数是预先确定的，则处理多个测试用例的循环结构比较简单。但是，若测试用例的个数或每组测试用例中数据的个数未知，仅知道测试用例内数据的结束标志和整个输入的结束标志，此时怎么办？在数据量较大、所有测试用例都采用同一运算且数据范围已知的情况下，有无提高计算时效的办法呢？在本节中，我们给出以下两个实例。

【1.2.1　Doubles】

给出2～15个不同的正整数，计算在这些数里面有多少对数满足一个数是另一个数的两倍。比如给出

1　4　3　2　9　7　18　22

答案是3，因为2是1的两倍，4是2的两倍，18是9的两倍。

输入

输入包括多个测试用例。每个测试用例一行，给出2～15个两两不同且小于100的正整数。每一行最后一个数是0，表示这一行的结束，这个数不属于2～15个给定的正整数。输入的最后一行仅给出整数-1，该行表示测试用例的输入结束，不用进行处理。

输出

对每个测试用例，输出一行，给出有多少对数满足其中一个数是另一个数的两倍。

样例输入	样例输出
1 4 3 2 9 7 18 22 0 2 4 8 10 0 7 5 11 13 1 3 0 -1	3 2 0

试题来源：ACM Mid-Central USA 2003

在线测试地址：POJ 1552，ZOJ 1760，UVA 2787

试题解析

本题包含多个测试用例，因此需循环处理每个测试用例，整个输入的结束标志是当前测试用例的第一个数是-1。循环体内做两项工作：

1）通过一重循环读入当前测试用例的数组 a，并累计数据元素个数 n。当前测试用例的结束标志是读入数据 0。

2）通过两重循环结构枚举 $a[]$ 的所有数据对 $a[i]$ 和 $a[j]$（$0 \leqslant i < n-1$，$i+1 \leqslant j < n$），判断 $a[i]$ 和 $a[j]$ 是否呈两倍关系（$a[i]*2==a[j] \| a[j]*2==a[i]$）。

参考程序

```
#include <iostream>                          //预编译命令
using namespace std;                         //使用 C++标准程序库中的所有标识符
int main()                                   //主函数
{                                            //主函数开始
    int i, j, n, count, a[20];               //声明整型变量 i、j、n、count 和整型数组 a
    cin>>a[0];                               //输入第 1 个测试用例的首个数据
    while(a[0]!=-1)                          //若输入未结束，则输入下一个测试用例
    {  n=1;                                  //读入当前数据组
        for( ; ; n++)
        {
            cin>>a[n];
            if (a[n]==0) break;
        }
        count=0;  //处理：计算当前测试用例中有多少对数满足一个数是另一个数的两倍
        for (i=0; i<n-1; i++)                //枚举所有数据对
        {
            for (j=i+1; j<n; j++)
            {
                if (a[i]*2==a[j] || a[j]*2==a[i])   //若当前数据对满足两倍关系，则累计
                count++;
            }
        }
        cout<<count<<endl;                   //输出当前测试用例中满足两倍关系的数据对数
        cin>>a[0];                           //输入下一个测试用例的首个数据
    }
    return 0;
}
```

本题的测试用例数和测试用例长度都是未知的，本题的求解程序采用双重循环结构。

外循环：枚举各组测试用例，结束标志为输入结束符（本题的输入结束符为 -1）。

内循环：输入和处理当前测试用例中的数据，输入的结束标志为测试用例的结束符（本题的测试用例的结束符为 0）。

在处理多个测试用例的过程中，可能会遇到这样一种情况：数据量较大，所有测试用例都采用同一运算，并且数据范围已知。在这种情况下，为了提高计算时效，可以采用离线计算方法，即预先计算出指定范围内的所有解，存入某个常量数组；以后每测试一个测试用例，直接从常量数组中引用相关数据就可以了，这样做避免了重复运算。

【1.2.2　Sum of Consecutive Prime Numbers】

一些正整数能够表示为一个或多个连续素数的和。给出一个正整数，有多少个这样的表示？例如，整数 53 有两个表示，即 5+7+11+13+17 和 53；整数 41 有三个表示，即 2+3+5+7+11+13、11+13+17 和 41；整数 3 只有一个表示，即 3；整数 20 没有这样的表示。注意加法操作数必须是连续的素数，因此，对于整数 20，7+13 和 3+5+5+7 都不是有效的表示。

请写一个程序，对于一个给出的正整数，程序给出连续素数的和的表示数。

输入

输入一个正整数序列，每个数一行，在 2 ～ 10 000 之间取值。输入结束以 0 表示。

输出

输出的每一行对应输入的每一行，除了最后的 0。输出的每一行对于一个输入的正整数，给出连续素数的和的表示数。输出中没有其他字符。

样例输入	样例输出
2	1
3	1
17	2
41	3
20	0
666	0
12	1
53	2
0	

试题来源：ACM Japan 2005

在线测试地址：POJ 2739，UVA 3399

试题解析

由于每个测试用例都要计算素数，且素数上限为 10 000，因此：首先，我们离线计算出 [2..10001] 内的所有素数，按照递增顺序存入数组 prime[1.. total]。

然后，依次处理每个测试用例：

设当前测试用例的输入为 n；连续素数的和为 cnt；cnt==n 的表示数为 ans。

采用双重循环计算 n 的表示数 ans：

外循环 i：枚举所有可能的最小素数 prime[i]（for (int i=0; n>=prime[i]; i++)）；

内循环 j：枚举由 prime[i] 开始的连续素数的和 cnt，条件是所有素数在 prime[] 中且 cnt 不大于 n（for (int j=i; j < total && cnt<n; j++) cnt += prime[j]）。内循环结束后，若 cnt==n，则 ans++。

外循环结束后得出的 ans 即为问题解。

参考程序

```
#include <iostream>                          //预编译命令
using namespace std;                         //使用 C++标准程序库中的所有标识符
const int maxp = 2000, n = 10000;            //设定素数表长和输入值的上限
int prime[maxp], total = 0;                  //素数表和表长初始化为 0
bool isprime(int k)                          //判定 k 是否为素数
{
    for (int i = 0; i < total; i++)
        if (k % prime[i] == 0)
            return false;
    return true;
}
int main(void)                               //主函数
{
    for (int i = 2; i <= n; i++)             //预先建立素数表
        if (isprime(i))
```