电子与通信工程学科精品教程

C语言程序设计教程

主　编　鞠剑平　主　审　孙俊逸

C YUYAN CHENGXU SHEJI JIAOCHENG

華中科技大學出版社
http://www.hustp.com

电子与通信工程学科精品教程

C语言程序设计教程

C YUYAN CHENGXU SHEJI JIAOCHENG

主　编　鞠剑平
副主编　方　洁　韩桂华　丁　崧　陈海洲　陈次球
参　编　徐　鹏　黎　曦　吴　慰　吴思华
主　审　孙俊逸

華中科技大學出版社
http://www.hustp.com
中国·武汉

图书在版编目(CIP)数据

C语言程序设计教程/鞠剑平　主编．—武汉：华中科技大学出版社，2009年11月
ISBN 978-7-5609-5660-2

Ⅰ．C…　Ⅱ．鞠…　Ⅲ．C语言-程序设计-高等学校-教材　Ⅳ．TP312

中国版本图书馆CIP数据核字(2009)第152391号

C语言程序设计教程　　鞠剑平　主编

策划编辑：张　毅
责任编辑：孙基寿　　封面设计：潘　群
责任校对：刘　竣　　责任监印：周治超

出版发行：华中科技大学出版社(中国·武汉)
武昌喻家山　邮编：430074　电话：(027)87557437

录　　排：华中科技大学惠友文印中心
印　　刷：华中科技大学印刷厂

开本：787 mm×960 mm 1/16　印张：26　插页：1　字数：553 000
版次：2009年11月第1版　印次：2009年11月第1次印刷　定价：40.00元
ISBN 978-7-5609-5660-2/TP·703

内 容 简 介

本书共分 11 章。前 10 章介绍了 C 语言的概念、数据类型、控制结构、数组、函数、编译预处理、指针、结构体和共用体、位运算和文件等相关知识。第 11 章是综合程序设计：设计了一个学生成绩管理系统（给出了完整的源代码，并做了必要的注释），并介绍了软件开发中的基本思路和方法。

本书适用于计算机和非计算机专业的程序设计初学者，同时也可作为编程人员和 C 语言自学者的参考用书。

前　言

计算机诞生至今不过几十年的时间，但人们的学习、工作、生活、娱乐的方方面面都已经离不开计算机和计算机技术了。在计算机技术的各个组成部分中，软件设计占有重要的地位。

程序设计是学习软件设计的入门课程，是进一步学习“面向对象程序设计”、“数据结构”、“算法设计与分析”等课程的基础。程序设计是以编程语言为平台，介绍程序设计的思想和方法。通过“C 语言程序设计”课程的学习，学生不仅要掌握高级程序设计语言的知识，更重要的是，要在实践中逐步掌握程序设计的基本思想和方法。

C 语言是一种历史悠久的程序设计语言。今天的 C++、Java、PHP，以及.NET 中的 C#和 Visual Basic.NET 等，都是以 C 语言为基础的。C 语言具有表达能力强、功能丰富、目标程序质量高、可移植性好、使用灵活等特点。C 语言既具有高级语言的优点，又具有低级语言的某些特性，特别适合于编写系统软件和嵌入式软件。C 语言的上述特点使得我国绝大部分高等院校都把 C 语言作为计算机和非计算机专业的第一门程序设计语言课程。全国计算机等级考试、全国计算机应用技术证书考试也都将 C 语言列入考试范围。

目前国内外的 C 语言教材很多，但很多教材重理论轻实践，大讲 C 语言的相关语法和一些编程技巧，而对如何编写高质量的应用程序涉及甚少。本书编写的初衷就是要给 C 语言的学习者提供更多的动手机会，加强学习者对程序设计和 C 语言概念、原理和规则的理解，培养学习者形成良好的编程风格和工程纪律，为学习者进一步学习其他程序设计语言打下基础。

本书共分 11 章。其中第 1 章、第 9 章、第 10 章由鞠剑平编写，第 2 章由吴思华编写，第 3 章由黎曦编写，第 4 章由吴慰编写，第 5 章由徐鹏编写，第 6 章由方洁编写，第 7 章、第 8 章由韩桂华编写，第 11 章由丁崧编写，陈海洲、陈次球负责部分章节的编写。全书由鞠剑平负责统稿工作。

湖北省高等教育学会高校计算机教学专业委员会主任孙俊逸教授审阅了全书，并提出了宝贵的修改意见。在本书编写过程中，湖北工业大学商贸学院电子信息工程系主任张胜利教授对编者给予了极大的支持和鼓励，程贵卿副教授、陈小常副教授和金国芳副教授给予了无私的帮助，武汉大学东湖分校的领导和老师也给予了很大的支持并提出了许多有益的意见，使本书增色不少。在此，对他们一并表示衷心感谢！

由于时间紧迫，编者水平有限，书中错误及疏漏之处在所难免，敬请读者批评指正。

编　者

2009 年 7 月

目　录

第 1 章　C 语言程序设计基础 ..... (1)
1.1　程序与程序设计语言 ..... (1)
1.2　算法与结构化程序设计 ..... (5)
1.3　C 语言概述 ..... (13)
1.4　C 语言程序的结构 ..... (17)
1.5　C 语言程序的实现 ..... (24)
1.6　程序的编辑、编译、调试和运行实训 ..... (30)
习题 1.1 ..... (33)
习题 1.2 ..... (33)
第 2 章　数据类型、运算符和表达式 ..... (36)
2.1　C 语言的数据类型 ..... (36)
2.2　常量与变量 ..... (37)
2.3　变量赋初值 ..... (45)
2.4　各类数值型数据间的混合运算 ..... (46)
2.5　C 语言的运算符和表达式 ..... (47)
2.6　数据类型与运算符实训 ..... (59)
习题 2.1 ..... (61)
习题 2.2 ..... (62)
第 3 章　C 语言控制结构 ..... (64)
3.1　顺序程序设计 ..... (64)
3.2　分支结构程序 ..... (79)
3.3　循环控制 ..... (92)
3.4　控制结构实训 ..... (103)
习题 3.1 ..... (104)
习题 3.2 ..... (105)
第 4 章　数组 ..... (106)
4.1　概述 ..... (106)
4.2　一维数组 ..... (107)
4.3　二维数组 ..... (117)
4.4　字符数组与字符串 ..... (124)

4.5 数组综合实例 (134)
4.6 数组与数据的排序实训 (136)
习题 4.1 (139)
习题 4.2 (140)
第 5 章 函数 (143)
5.1 函数概述 (143)
5.2 函数的定义 (145)
5.3 函数的调用 (146)
5.4 函数的嵌套与递归调用 (152)
5.5 变量的作用域和存储域 (157)
5.6 函数间的数据传递 (167)
5.7 main () 函数的参数和返回值 (173)
5.8 文件包含与条件编译 (176)
5.9 函数应用综合实例 (179)
5.10 函数应用实训 (182)
习题 5.1 (183)
习题 5.2 (183)
第 6 章 指针 (189)
6.1 地址和指针的概念 (189)
6.2 变量的指针和指向变量的指针变量 (191)
6.3 数组的指针和指向数组的指针变量 (200)
6.4 字符串与指针 (221)
6.5 指向函数的指针 (228)
6.6 返回指针值的函数 (231)
6.7 指针数组和指向指针的指针 (233)
6.8 指针与数组实训 (240)
6.9 指针与字符串实训 (242)
习题 6.1 (243)
习题 6.2 (244)
第 7 章 预处理命令 (251)
7.1 宏定义 (251)
7.2 文件包含 (260)
7.3 条件编译 (261)
7.4 编译预处理实训 (263)
习题 7.1 (265)

习题 7.2 ..... (265)
第 8 章 结构体与共用体 ..... (268)
8.1 结构体类型定义和结构体变量说明 ..... (268)
8.2 结构体变量的引用和初始化 ..... (272)
8.3 结构体指针变量 ..... (277)
8.4 常用的内存管理函数 ..... (282)
8.5 链表 ..... (284)
8.6 共用体 ..... (292)
8.7 枚举类型数据 ..... (296)
8.8 用 Typedef 定义类型 ..... (298)
8.9 综合实例 ..... (299)
8.10 结构体和共用体实训 ..... (302)
习题 8.1 ..... (303)
习题 8.2 ..... (303)
第 9 章 位运算 ..... (308)
9.1 位运算的概述 ..... (308)
9.2 位运算 ..... (310)
9.3 位段(位域) ..... (317)
9.4 综合案例分析 ..... (321)
9.5 位运算实训 ..... (324)
习题 9.1 ..... (327)
习题 9.2 ..... (327)
第 10 章 文件 ..... (329)
10.1 概述 ..... (329)
10.2 文件的输入/输出 ..... (333)
10.3 文件的定位操作 ..... (347)
10.4 文件的检测 ..... (350)
10.5 综合实例 ..... (351)
10.6 文件实训 ..... (354)
习题 10.1 ..... (355)
习题 10.2 ..... (355)
第 11 章 综合程序设计 ..... (360)
11.1 程序的功能 ..... (360)
11.2 程序模块构思 ..... (361)
11.3 程序界面设想 ..... (365)

11.4 程序代码清单和注释 ..... (366)
附录 A 基本控制字符/字符与 ASCII 代码对照表 ..... (387)
附录 B C 语言操作符的优先级 ..... (388)
附录 C Turbo C 2.0 集成环境简介 ..... (388)
习题参考答案 ..... (397)
参考文献 ..... (406)

第1章 C语言程序设计基础

本章介绍了计算机程序设计语言的功能、算法的概念及其描述、C语言的发展历史、C语言的特点、C程序的结构和C程序的上机步骤。学习本章，要求重点掌握算法的描述、C程序的结构和上机运行C程序的方法。学完本章之后，读者将对程序设计以及C语言有一个初步的完整印象。

1.1 程序与程序设计语言

程序是一个非常普遍的概念：为解决某些问题用计算机可以识别的代码编排的按照一定的顺序安排的工作步骤。计算机严格按照这些步骤去做，包括计算机对数据的处理。程序的执行过程实际上是对程序所表达的数据进行处理的过程。一方面，程序设计语言提供了一种表达数据与处理数据的功能；另一方面，编程人员必须按照语言所要求的规范(即语法规则)进行编程。

1.1.1 程序与指令

计算机处理数据的基本单元是计算机指令。单独的一条指令本身只能完成计算机的一个最基本的功能，计算机所能实现的指令的集合称为计算机的指令系统。虽然一条计算机指令只能实现一个简单的功能，而且指令系统的指令数目也是有限的，但是一系列有序的指令组合却能完成很复杂的功能。一系列计算机指令的有序组合就构成了程序。

一般情况下程序的执行是按照指令的排列顺序一条一条地执行的，但是有的程序往往需要通过判断不同的情况执行不同的指令分支，还有些指令需要被反复执行。

程序在计算机中是以0、1组成的指令码(即机器语言)来表示的，即程序是0、1组成

的序列，这个序列能被计算机所识别。一般情况下，程序和数据均存储在存储器中(这种结构称为冯·诺伊曼结构，而程序和数据分开存储的结构称为哈佛结构)。当程序要运行的时候，当前准备运行的指令从内存中被调入CPU，由CPU处理该指令。

1.1.2 程序设计语言

语言是人们交换思想的工具，我们日常生活中使用的汉语、英语等称为自然语言。计算机诞生以后，人们要指挥计算机工作就产生了计算机语言。用于程序设计的计算机语言基本上可分为三种：机器语言、汇编语言和高级语言。

1. 机器语言

计算机诞生的初期，人们使用的计算机语言仅由计算机能够识别的0和1代码组成，被称为机器语言。下面是某CPU指令系统中的两条指令：

```
1 0 0 0 0 0 0 0  (进行一次加法运算)
1 0 0 1 0 0 0 0  (进行一次减法运算)
```

用机器语言编程序，就是从所使用的CPU的指令系统中挑选合适的指令，组成一个指令序列。这种程序虽然可以被机器直接理解和执行，却由于它们不直观、难记、难认、难理解、不易查错，只能被少数专业人员掌握，并且编写程序的效率很低，质量难以保证，这使计算机的推广使用受到了极大的限制。

2. 汇编语言

为减轻人们在编程中的劳动强度，20世纪50年代中期人们开始用一些英文助记符号来代替0、1代码编程，于是便产生了符号语言(或称汇编语言)。如前面的两条机器指令可以写为

```
ADD A,B
SUB A,B
```

用汇编语言编程，程序的编写效率及质量都有所提高。但是，汇编语言指令是机器不能直接识别和执行的，而要先翻译成机器语言程序才能被机器识别和执行。将汇编语言程序转换成为二进制代码表示的机器语言的程序称为汇编程序，经汇编程序“汇编(翻译)”得到的机器语言程序称为目标程序，原来的汇编语言程序称为源程序。由于汇编语言指令与机器语言指令基本上具有一一对应的关系，所以汇编语言源程序的代换可以由汇编系统以查表的方式进行。用汇编语言编写的程序效率高，占用存储空间小，运行速度快。而且用汇编语言能编写出非常优化的程序。

汇编语言和机器语言都不能脱离具体机器即硬件，均是面向机器的语言。不同类型的

计算机所用的汇编语言和机器语言是不同的，缺乏通用性，因此，汇编语言被称为低级语言。用面向机器的语言编程，可以编出效率极高的程序，但是程序员用它们编程时，不仅要考虑解题思路，还要熟悉机器的内部结构，并且要“手工”地进行存储器分配，因而其劳动强度仍然很大，给计算机的普及推广造成了很大的障碍。

3. 高级语言

1954 年出现的 FORTRAN 语言以及随后相继出现的其他高级语言，开始使用接近人类自然语言的、但又消除了自然语言中的二义性的语言来描述程序。高级语言不受具体机器的限制，使用了许多数学公式和数学计算上的习惯用语，非常擅长于科学计算。用高级语言编写的程序通用性强，直观、易懂、易学，可读性好。到目前为止，世界上有数百种高级语言，常用的有几十种，如 FORTRAN、PASCAL、C、LISP、COBOL 等。这些高级语言使人们开始摆脱进行程序设计必须先熟悉机器的桎梏，把精力集中于解题思路和方法上，使计算机的使用得到了迅速普及。

1.1.3 高级语言程序的开发过程

高级语言程序的开发过程主要包括如下五大步骤。

1. 分析和建立模型

一般来说，一个具体的问题会涉及许多方面，这是问题的复杂性所在。为了便于求解，往往要忽略一些次要方面。这种通过忽略次要方面从而找出解题规律的过程，称为建立模型。

2. 表现模型

表现模型就是用一种符号语言系统来描述模型。一般来说，模型的表现会随着人们对问题抽象程度的加深和细化，不断由领域特色向计算机可解释、可执行的方向靠近(中间也可能采用一些其他的符号系统，如流程图等)，直到最后用一种计算机程序设计语言将其描述出来。

3. 源程序的编辑

源程序的编辑就是在某种字处理环境下，用具体的程序设计语言编写源程序的过程。这不仅要掌握一种计算机程序设计语言，还要应用一种专用程序编辑器或通用的文字编辑器。

4. 程序的编译(或解释)与链接

写出一个高级语言程序后，并不是可以立即拿来执行的。要让机器执行，还要将它翻

译成由机器可以直接辨认并可以执行的机器语言程序。为区别它们，把用高级语言编写的程序(文件)称为源程序(文件)，把机器可以直接辨认并执行的程序(文件)称为可执行程序(文件)。这一过程一般分为两步。

第 1 步。在程序编辑过程中输入到源文件中的是一些字符码，但是机器可以直接处理的是 0、1 信息。为此，首先要将源程序文件翻译成 0、1 码表示的信息，并用相应的文件保存。这种保存 0、1 码信息的文件称为目标程序文件。由源文件翻译成目标文件的过程称为编译。在编译过程中，还要对源程序中的语法和逻辑结构进行检查。编译任务是由称为编译器(compiler)的软件完成的。目标程序文件还不能被执行，它们只是一些不连续的目标程序模块。

第 2 步。将目标程序模块以及程序所需的系统中固有的目标程序模块(如执行输入、输出操作的模块)链接成一个完整的程序。经正确链接所生成的文件才是可执行文件。完成链接过程的软件称为链接器(linker)。

图 1.1.1 为编译和链接过程示意图。

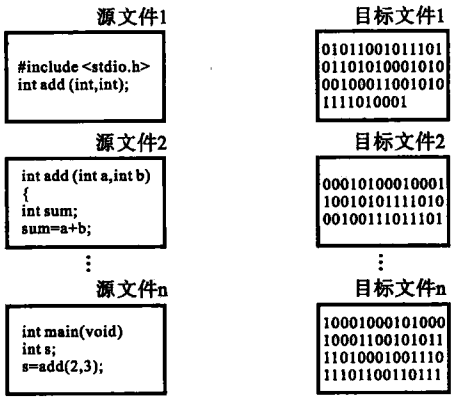

图 1.1.1　编译和链接过程示意图

5. 程序的测试与调试

程序文件经编译、链接之后，生成可执行文件，这就可以让计算机执行了。但是，并不是一定就可以得到预期的结果，因为程序仍然可能会存在某些错误。因此，每一个人编写出一个程序后，在正式交付使用前，总要试运行该程序，也就是对程序进行测试。

测试是以程序通过编译后没有语法和链接上的错误为前提的，目的是找出程序中可能存在的错误并加以改正。因此，应该测试程序在不同情况下运行的情况，输入不同的数据可以检测出程序在不同情况下运行的情况。测试的数据应是以“程序是会有错误的”为前提精心设计出来的，而不是随心所欲地乱凑而成的。它们不仅应含有被测程序的输入数据，而且还应包括程序执行它们后所得预期的结果。每次测试都要把实际的结果与预期的结果相比较，以观察程序是否出错。

1.2 算法与结构化程序设计

1.2.1 算法的概念

计算机尽管可以完成许多极其复杂的工作，但实质上这些工作都是按照事先编好的程序进行的。所以人们常把程序称为计算机的灵魂。1976 年瑞士计算机科学家 Niklaus Wirth 提出了一个著名的公式：

算法 + 数据结构 = 程序

他认为，“程序就是在数据的某些特定表示方式和结构的基础上对抽象算法的具体表述”。从今天的观点来看，它只能是对过程化程序的一个抽象定义，对面向对象的程序而言则不尽然。不过对学习 C 语言这样的面向过程的程序设计语言而言，是完全适用的。也就是说，面向过程的程序有两大要素：数据结构和算法。数据结构是程序所处理的对象——数据的表示方法和组织形式，数据类型是其重要内容。算法(algorithm)是对操作的描述，即操作步骤。

做任何事情都有一定的步骤。为解决一个问题而采取的方法和步骤，就称为算法。例如，菜谱实际上是做菜肴的算法，乐谱实际上是演奏的算法。对同一个问题，可有不同的解题方法和步骤，产生的结果或者效果也可能有所差异。计算机算法是计算机能够执行的算法，计算机程序是用某种程序设计语言描述的解题算法。

算法含有两大组成要素：操作和控制结构。

算法是由一系列操作组成的。每个操作的确定不仅取决于问题的需求，还取决于它们取自哪个操作集，与使用的工具系统有关。如算盘上的操作集由进、退、上、下、去等组成；做菜的操作集包括架锅、加油、炒、煮、炸、蒸、焖、加水、加调料等；驾驶汽车的操作集包括踩离合器、踩油门、开电门、换挡、左转、右转、开灯、关灯等。计算机算法要由计算机实现，组成它的操作集是计算机所能进行的操作。而且这些操作的描述与程序设计语言的级别有关。在高级语言中所描述的操作主要包括：算术运算(+、-、*、/)、逻辑运算(“与”、“或”、“非”等)、关系运算(==、>=、<=、>、<、!=等)、函数运算、位运算、I/O 操作等。计算机算法是由这些操作所组成的。

算法的另一要素是其控制结构。每一个算法都要由一系列的操作组成。同一操作序列，按不同的顺序执行，就会得出不同的结果。控制结构即如何控制组成算法的各操作的执行顺序。结构化程序设计方法要求：一个程序只能由三种基本控制结构(顺序结构、选择结构

和循环结构)或由它们派生出来的结构组成。1966年Bohm和Jacopini证明，由这三种基本结构可以组成任何结构的算法，以解决任何问题。

通常认为算法有如下性质。

有效性 有效性指算法所规定的操作都应当是能够有效执行的，并能得到确定的结果。例如，对于操作汽车的算法，有效的操作就是加速、刹车、换挡、转动方向盘、鸣笛等，要让汽车执行“跳起”就是无效的操作。同样，一个计算机算法必须是计算机能够执行的。

确定性 确定性具有两重意义，其一是所描述的操作应当具有明确的意义，不应当有歧义性。例如，不能发出这样的操作指令：“执行一个算术操作”。因为它既没有指出算术操作的类型，也没有指出操作数。确定性的另一重意义是，操作序列只有一个初始动作，序列中每一动作仅有一个后继动作。序列终止表示问题得到解答或问题没有解答，不能没有任何结论。

有穷性 有穷性指算法所规定的操作序列必须在允许的时间内结束。例如，一个计算机算法若要执行100年以上，就失去有穷性。

确定的输入和输出性 输入是指在执行算法时需要从外界取得必要的信息。一个算法可以有零个或多个输入。算法执行完毕，必须有一个或若干个输出结果，没有输出的算法是没有意义的。

1.2.2 算法的描述

为了描述算法，人们创建了许多算法描述工具。常用的几种工具是：自然语言描述法、流程图表示法、N-S流程图表示法、伪代码表示法和计算机语言表示法等。

1. 自然语言描述

自然语言是人们日常使用的语言，可以是汉语、英语或其他语言。用自然语言表示的优点是通俗易懂，但文字冗长，容易出现“歧义性”。而自然语言表示的含义往往不太严格，要根据上下文才能判断其正确含义，且描述包含分支和循环的算法时也不很方便。因此，除了对那些很简单的问题外，一般不用自然语言描述算法。

【例1.2.1】 对一个大于或等于3的正整数，判断它是不是一个素数。

算法可表示如下。

S1：输入n的值

S2：i=2

S3：n被i除，得余数r

S4：如果r=0，表示n能被i整除，则打印n“不是素数”，算法结束；否则执行S5

S5: i+1→i

S6: 如果 i≤n-1，返回 S3；否则打印 n“是素数”，算法结束

改进的算法可表示如下。

S6:如果 $i \leqslant \sqrt{n}$，返回 S3；否则打印 n“是素数”，算法结束

2. 流程图

流程图是一种使用很广泛的算法描述工具。这种工具的特点是用一些图框表示各种类型的操作，用线表示这些操作的执行顺序。美国国家标准化协会 ANSI(American National Standard Institute)规定了一些常用的流程图符号，具体见图 1.2.1。

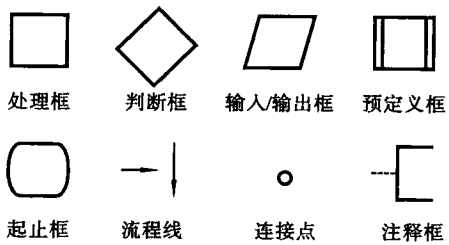

图 1.2.1　常用的流程图标准化符号

处理框　表示各种处理功能。例如，执行一个或一组特定的操作，从而使信息的值、信息的形式或所在位置发生变化。矩形内可注明处理名称或其简要功能。

判断框　表示判断。菱形内可注明判断的条件。它只有一个入口，但可以有若干个可供选择的出口，在对定义的判断条件求值后，有一个且仅有一个出口被选择。求值结果可在表示出口路径的流程线附近写出。

输入/输出框　表示数据，其中可注明数据名称、来源、用途或其他的文字说明。

预定义框　表示已命名的处理。该处理为在另外地方已得到详细说明的一个操作或一组操作。例如库函数或其他已定义的函数等。矩形内可注明特定的处理名称或其简要功能。

起止框　表示算法的开始和结束。

流程线　表示执行的流程。当流程自上向下或由左向右时，流程线可不带箭头，其他情况下应加箭头表示流程。

连接点　用于将画在不同位置的流程线连接起来。使用连接点，可以避免流程线的交叉或过长，使流程图清晰。

注释框　注释是程序的编写者向阅读者提供的说明。注释框由粗边线构成，它用虚线连接到被注解的符号或符号组上。

一般情况下，一个流程图包括以下几部分：表示相应操作的框；带箭头的流程线；框

内外必要的文字说明。

【例 1.2.2】将例 1.2.1 判断素数的算法用流程图表示(见图 1.2.2)。

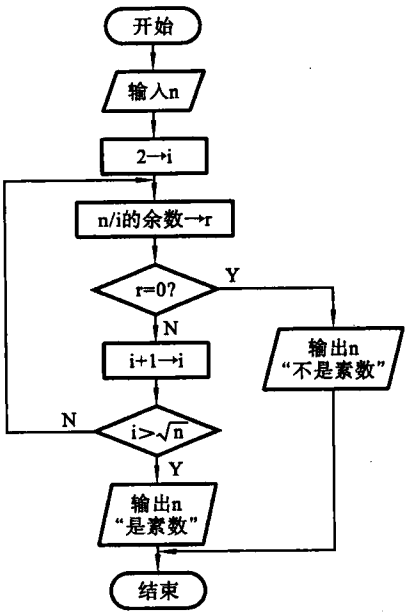

图 1.2.2 判断素数的算法流程图

【例 1.2.3】判定 2000—2500 年中的每一年是否为闰年，将结果输出。分别用自然语言和流程图表示该算法。

闰年的条件：

能被 4 整除，但不能被 100 整除的年份；

能被 100 整除，又能被 400 整除的年份。

设 y 为被检测的年份，则算法可表示如下。

S1:2000→y

S2:若 y 不能被 4 整除，则输出 y“不是闰年”，然后转到 S6

S3:若 y 能被 4 整除，不能被 100 整除，则输出 y“是闰年”，然后转到 S6

S4:若 y 能被 100 整除，又能被 400 整除，则输出 y“是闰年”，否则输出 y“不是闰年”，然后转到 S6

S5:输出 y“不是闰年”

S6:y+1→y

S7:当 y≤2500 时，返回 S2 继续执行，否则结束

该算法的流程图如图 1.2.3 所示。