普通高等教育农业部“十二五”规划教材

全国高等农林院校“十二五”规划教材

C语言程序设计基础

石玉强 主编

中国农业出版社

普通高等教育农业部“十二五”规划教材
全国高等农林院校“十二五”规划教材

C语言程序设计基础

石玉强 主编

中国农业出版社

图书在版编目（CIP）数据

C语言程序设计基础/石玉强主编．—北京：中国农业出版社，2012.12

普通高等教育农业部“十二五”规划教材　全国高等农林院校“十二五”规划教材

ISBN 978-7-109-17340-8

Ⅰ.①C…　Ⅱ.①石…　Ⅲ.①C语言－程序设计－高等学校－教材　Ⅳ.①TP312

中国版本图书馆CIP数据核字（2012）第262792号

内　容　简　介

本书通过案例的模式引出C语言所涉及的知识点，着重引导读者正确的逻辑思维与解题编程的思路，突出培养读者的分析问题、解决问题的能力，提高读者编程及上机操作的能力，并在通俗易懂的内容、规范化源程序的引导下使读者养成良好的编程习惯。

本书共分12章，分别是程序设计基础知识、C语言概述、顺序结构程序设计、分支结构程序设计、循环结构程序设计、数组、函数、预处理命令、指针、结构体与共用体、位运算和文件。

本书思路清晰、重点突出、易学易用，可作为高等院校应用型本科计算机科学与技术及相关专业C语言程序设计课程的教材，也可以作为全国计算机等级考试的培训教材，还可以作为从事应用软件开发的专业人员的参考书。

中国农业出版社出版

（北京市朝阳区农展馆北路2号）

（邮政编码100125）

策划编辑　朱　雷

文字编辑　赵　渴

北京中新伟业印刷有限公司印刷　　新华书店北京发行所发行

2013年1月第1版　　2013年1月北京第1次印刷

开本：787mm×1092mm　1/16　　印张：20.5

字数：508千字

定价：34.00元

前　言

C语言是国内外广泛使用的计算机程序设计语言，是许多计算机专业人员和计算机爱好者学习程序设计的首选语言，也是软件开发人员必须掌握的一种计算机语言。各类高等学校普遍开设了C语言程序设计课程，全国计算机考试二级和三级考试中也包括了C语言程序设计的内容。

全书共分12章。第1章主要介绍程序与程序语言、算法和算法描述、结构化程序设计方法，程序调试方法；第2、3章介绍C语言的基本概念，包括C语言的发展、特点、程序的基本组成和结构、程序上机执行的过程、C语言基本数据类型、常量和变量、常用函数及运算符的使用、顺序结构程序设计；第4、5章分别介绍选择结构程序设计和循环结构程序设计；第6、7章分别介绍数组和函数的概念及使用方法；第8章介绍预处理命令；第9章介绍指针；第10章介绍结构体和共用体数据类型及使用方法；第11章介绍C语言的位运算；第12章介绍文件的概念及操作，并通过综合实例介绍文件在编程中的应用。每章后面均配有习题，以加强对C语言程序设计知识和编程方法的理解和掌握。

C语言程序设计基础是计算机类专业学生进入大学以后的第一门程序设计类课程，这门课程的学习效果对以后的专业课，如数据结构、面向对象程序设计、操作系统、嵌入式系统原理及应用、单片机原理及接口等专业课的学习都有举足轻重的作用。通过C语言程序设计基础这门课程的学习，不仅要让学生掌握程序设计语言的语法，熟悉一个集成开发环境的使用，更重要的是让学生掌握程序设计的基本方法与步骤，形成一个良好的编程风格。本教材具有以下几个特色：

1. 采用案例、问题的方式引入知识点

传统的教材一般是先讲知识点，然后通过例题来加深学生对知识点的掌握。虽然中规中矩，但亦有它固有的缺点，在讲授知识点时，有些内容由于知识点多，学生注意力容易分散，讲授例题时没有给学生充分的思考时间，从而对程序设计知识的掌握不够。在本教材中，通过精心设计的案例引入知识点，通过问题分析让学生思考如何合理运用知识点更好地解决问题，从而达到对知识点的理解和掌握。通过案例提高学生对C语言程序设计基础的学习兴趣。

2. 突出实践能力的培养

C语言程序设计基础是一门实践性很强的课程，要突出实践能力的培养，不仅要求学生掌握C语言程序设计的基础知识，更重要的是让学生学会思考，知其然更知其所以然。对于给定的问题，要让学生学会如何进行有效的算法描述，学会如何正确表示和存储数据，学会如何进行设计、编码、测试和调试程序。本教材将对精心选择的案例进行详细的分析、描述，让学生从案例中学会分析问题、解决问题，以提高实践能力。

3. 突出分析问题、解决问题的能力

现在有很多大学生在编程时往往会有这样一个弊病：一看到程序设计的题目，就会想到

用什么样的语句、什么样的结构来实现，没有突出分析问题的过程，没有分析算法设计的过程。这样做的后果就是写出来的程序容易出现错误，甚至漏洞百出。为了摆脱这种窘况，在本教材中对每一个案例的处理先进行分析，然后进行算法思路的描述，之后才是进行代码设计，最后进行测试。通过大量案例，让学生掌握这种模式，并且应用到实践中去，从而提高学生的分析问题能力，提高学生的编程质量。

4. 采用规范化的编程风格

程序的可读性是衡量程序质量高低的一个重要标准，一个可读性很强的程序，在其维护的过程中将会变得容易，一个晦涩难懂的源程序对于维护来说将是一场灾难。在软件产业化的今天，规范化的编程风格就尤显重要。在本教材中，将采用规范化的编程风格来进行编码，这既有利于学生的学习，更有利于学生养成一种良好的编程习惯。

本教材的主编、参编人员由多年从事C语言程序设计基础教学工作第一线的骨干教师组成，有着丰富的教学经验，深知学生在学习程序设计基础课程中遇到的一些重点、难点及处理方法，为新编一本适用于应用型本科教学的C语言程序设计基础教材奠定了良好的基础。

本书可作为高等院校应用型本科计算机科学与技术及相关专业C程序设计课程的教材，也可以作为全国计算机等级考试的培训教材，同时也可作为从事应用软件开发的专业人员的参考书。

本书由石玉强任主编，负责全书内容的取材和组织，闫大顺、李晟、符志强、汪文彬任副主编，石玉强编写了第1、2章，汪文彬编写了第3、7章，符志强编写了第4、5、6章，李晟编写了第8、9章，闫大顺编写了第10、11、12章。另外，我们要感谢吴家培、沈玉利、成筠、曾宪贵、刘磊安、黄洪波、陈勇、郑建华、蒋明亮、杜淑琴、张垒、王成、杨灵、王冬、李应勇、黄应红、陈德祥、刘雍、吴蒋等老师，为本教材的编写提出了许多宝贵意见。

由于时间仓促，编者水平有限，书中难免存在错误和不足之处，欢迎广大读者和同行批评指正。作者电子邮箱：yuqiangshi@163.com。

编　者

2012年8月

目　录

第1章 程序设计基础知识

[本章学习目标]

- 了解程序、计算机语言等基本概念
- 了解算法的概念及其特性
- 掌握常用的算法描述的方法
- 掌握结构化程序设计方法

计算机都是由程序控制的，程序是由事先编写的一条条语句组成的。只有掌握了程序设计的基本原理、方法和技术，才能真正地深入了解计算机，编写出质量较高的程序，使计算机的功能更加强大。

1.1 程序与程序语言

1.1.1 计算机程序

现在的计算机都是基于冯·诺依曼结构设计的，它们是由运算器、控制器、存储器、输入设备、输出设备等5部分构成，如图1-1所示。计算机硬件是由存储在存储器中的程序控制下自动运行的，将数据输入到计算机，按照程序的指令序列处理数据，并把结果输出到输出设备。在计算机中，硬件是躯体，程序是灵魂，计算机的一切操作都是由程序控制的，离开了程序，计算机硬件什么功能都完成不了。

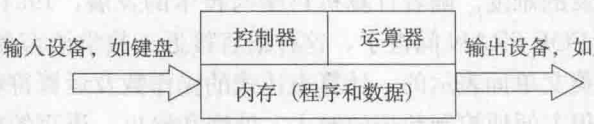

图1-1 计算机组成

狭义上讲，程序(program)就是为了完成某个任务而存储在存储器中的语句序列，每条语句使计算机执行特定的操作。组成程序的指令集对于每个计算机系统都是确定的，人们按照程序的目标，编写出一个特定的语句序列，装载到计算机存储器中，让计算机硬件按照其语句序列来自动执行，完成程序的功能。

广义上讲，程序是为了实现特定目标或解决特定问题，由一组计算机能识别和执行的命令序列组成的。程序不仅仅是存储器中的指令序列，保存在外围设备中的可执行文件也可以称为程序或软件。

1.1.2 计算机语言

按照解决问题的需要，选择恰当的指令构成一组指令序列的过程就是编写程序。从语言的角度来看，使用一种语言，告诉计算机如何输入数据？如何处理数据？如何输出？这种功能受限制的语言就称为计算机语言。程序设计语言的发展总体上经历了从低级语言到高级语言的过程。其中，低级语言包括机器语言、汇编语言等，高级语言包括Fortran、Basic、Pascal、Java、C和C++等。

1. 机器语言

机器语言是指一台计算机硬件的全部指令集合。指令是设计计算机系统时确定的1和0的序列，不同的1和0的序列就构成了特定计算机的指令集，计算机只能识别和执行这些二进制指令，所以称为机器指令。在计算机应用的早期，只能使用由0和1构成的指令来编写程序，但这些指令与人们习惯的语言差别太大，难学、难记、难写，只能由专家才能编写程序。程序难检查、难修改，大大限制了程序的开发和应用。

另外，不同类型计算机系统的指令集往往各不相同，所以，在一台计算机上编写的机器语言程序要想在另一台计算机上执行，必须修改程序，造成重复工作，这称为机器语言的可移植性比较差。

2. 汇编语言

汇编语言也称为符号语言，它是用助记符来代替指令和地址，即用英文单词的简写和数字来简化二进制机器指令，比如" MOV" 代表数据传递操作、" MUL" 代表乘法运算等等。这样比较容易记住指令和编写程序，也容易读懂并理解程序在干什么，便于纠错及维护，提高了编程的效率。

由助记符或者符号表示的指令集称为汇编语言。用汇编语言编写的程序是由符号构成，计算机不能识别更不能执行这些指令，必须通过汇编程序把由符号组成的程序转换成机器语言的指令序列，这个转化的过程就是汇编。

助记符表示的指令和机器指令基本上是一一对应的，汇编语言也是面向特定的计算机系统，所以称机器语言和汇编语言是低级语言，它们的可移植性都比较差，但用它们编写出来的程序效率高，很多操作系统、驱动程序和一些系统程序都是用汇编语言开发的。

3. 高级语言

(1) 高级语言的出现　人们习惯使用自然语言，使用面向机器的低级语言增加了程序设计和开发的难度。随着计算机科学与技术的发展，1954年，第一个完全脱离机器硬件的高级语言FORTRAN问世了，这种语言接近于数学语言和自然语言，程序中所用的语句和指令是用英文单词表示的，计算表达式的操作数及运算符和数学公式近似，非常容易理解，从而可以很方便地控制数据的输入、处理和输出，语言的功能很强。由于这种语言不依赖于具体的计算机系统，编写的程序具有很强的可移植性，故称为高级语言。

由高级语言编写的程序称为源程序，它不能被计算机识别，大多数源程序是通过编译程序（或者称为编译器软件）转换为机器语言的程序（或者汇编语言的程序），这个过程称为编译。少量高级语言的源程序是经过解释程序处理的，一条高级语言语句翻译成一组机器指令后立即执行，再翻译下一条语句再执行，直到程序结束，该过程称为解释。

(2) 程序设计语言的发展史　高级语言的发展经历了从早期非结构化到结构化程序设计语言，从面向过程到面向对象程序语言的过程。相应地，程序或软件的开发也由最初的充满设计技巧的个人设计到小团队设计，最后发展为全球化协同程序开发的开放式软件工业生产的过程。

20世纪60年代中后期，软件功能越来越多，规模越来越大，但软件的生产基本上仍是各自为战，缺乏科学规范的系统规划与测试、评估标准，程序的正确性难以保证，其后果是大批耗费巨资开发的软件系统，由于含有错误而无法使用，带来巨大损失。软件带给人的感觉越来越不可靠，以致几乎没有不出错的软件。这一切，极大地震动了计算机工业界和研究界，这就是“软件危机”。人们认识到：大型程序的编制不同于写小程序，它应该是一项新的技术，应该像处理工程一样管理软件研制的全过程。程序的设计应易于保证正确性，便于验证正确性。1969年，提出了结构化程序设计方法，1970年，第一个结构化程序设计语言

Pascal 语言出现，标志着结构化程序设计时期的开始。

20 世纪 80 年代初，在软件设计思想上，又产生了一次革命，其标志性成果就是面向对象的程序设计。在此之前的高级语言，几乎都是面向过程的，程序的执行如流水线，在一个模块被执行完成前，操作人员不能控制，也无法动态地改变程序的执行方向。这和人们日常处理事物的方式是不一致的，对人而言希望发生一件事就处理一件事，也就是说，不能面向过程，应面向具体的应用功能，也就是对象(Object)。其方法就是软件的集成化，如同硬件的集成电路一样，生产一些通用的、封装紧密的功能模块，按照功能需求组合在一起，称之为软件集成块，它与具体应用无关，但能相互组合，完成具体的应用功能，同时又能重复使用。对使用者来说，只关心它的接口(输入量、输出量)及能实现的功能，至于如何实现的，是属于它内部的事，使用者完全不用关心，这就是面向对象的语言，如 C++、Visual Basic、Java 就是典型代表。

从 20 世纪 50 年代中期到现在，全世界涌现了 2500 多种高级程序设计语言，每种语言都有特定的用处，应用比较广泛的有 100 多种，影响较大、使用较普遍的有 FORTRAN（适合数值计算的面向过程语言）、COBOL（适合商业管理的面向过程语言）、BASIC（适合初学者的小型解释性语言）、Pascal（适合教学的结构化程序设计语言）、C（规模小、功能强、面向过程的系统描述高级程序设计语言）、C++（面向对象语言）、Visual C++（可视化面向对象语言）、Visual Basic（支持面向对象的 Basic 语言）、Java（适合于网络的面向对象语言）、Lisp 和 Prolog（人工智能语言）等。

高级语言的下一个发展目标是面向应用，也就是说，只需要告诉计算机你要干什么，计算机就能自动生成算法，自动进行处理，就是自动化程序设计语言。

C 语言自 20 世纪 70 年代诞生以来，经过几十年的发展，目前仍然是一种长盛不衰、深受人们喜爱的程序设计语言。C 语言系统规模短小精悍，既适合于编写应用程序，又能编写系统程序，具有丰富的数据类型和表达式，语法限制比较少，书写比较自由，编写出的程序效率比较高，在 C 语言的语法和基本结构的基础上诞生了现在主流的 Java、C++、C# 等语言，所以 C 语言成为全球程序设计的公共语言。

1.2　算法和算法描述

用计算机求解实际问题一般分为三个基本步骤：首先从实际问题中抽象出数学模型，然后设计解决该数学模型的算法，最后用计算机语言开发出对应的程序，运行该程序解决问题。从实际问题中抽象出数据模型的实质，就是用数学的方法抽取其主要的、本质的内容，实现对该问题的认识。这是计算机求解问题中最难的部分，在学习高等数学、数学建模等相关课程后，读者会在实践中逐步掌握数学建模的理论和方法。

算法描述是程序设计的一个重要组成部分。从另外一个方面来讲，对于一个需要解决的实际问题，怎样才能够设计出计算机可以运行的程序？首先要选定合理的数据结构，然后要设计解决问题的算法。有了一个好的算法，才可以选用一种程序设计语言把算法转变为程序，这也是 1984 年的图灵奖获得者瑞士学者和计算机科学家尼克劳斯·沃思(Niklaus Wirth) 提出的著名公式“数据结构＋算法＝程序”。

为此，必须掌握四方面的知识：①数据结构，程序中要处理和保存数据的描述，即指定数据的类型和数据的组织形式。②算法，对有限操作步骤的确定性描述。③程序设计方法，比如尼克劳斯·沃思提出的结构化程序设计方法。④程序设计语言和程序开发平台，用集成

的程序语言开发工具编写、编辑、编译、调试程序，程序中的操作语句就是对算法的实现。

在这四部分中，算法是灵魂，它是解决“做什么”和“怎么做”的问题，不了解算法就谈不上程序设计。因为有专门的课程讲授《数据结构》、《算法设计与分析》、《程序设计方法》等等，本教材主要讲授C程序设计语言，通过程序设计的练习，把算法与高级语言的学习紧密结合起来，使读者掌握基本程序设计的方法，逐步培养其基本算法设计能力。

1.2.1 算法的概念

算法（algorithm）是指解题方案的准确而完整的描述，是一系列解决问题的指令。从另外一个层面来看，算法是对问题求解过程的一种描述，是为解决一个或一类问题给出的一个确定的、有限长的操作序列。广义上讲，为解决一个问题而采取的方法和步骤，都称为算法。

现实生活中，我们做任何事情都有一定的步骤。比如计算机科学与技术专业的毕业典礼的日程安排，班级的团日活动安排等等，只有事先很好地计划，安排好步骤，按部就班地进行，才能完成任务，以避免出错。计算机算法是为了完成任务而规划的有限的计算机操作序列或计算步骤。

计算机算法分为数值算法和非数值算法两种。数值算法是对问题进行数值求解，这类算法已经研究得比较深入，往往有比较成熟的算法可供直接使用。非数值算法包括非常广泛的领域，如信息检索、事务管理等等，该类算法的种类繁多，要求不一，往往没有现成的算法可用，需要按照用户的不同要求进行设计。

一个计算机算法应当具有以下5个基本特性：

（1）有穷性　一个算法应包含有限个操作步骤，也就是说在执行若干个操作步骤之后，算法将结束，而不是无限地执行下去。

（2）确定性　算法中的每一个步骤都应当是明确的、无二义性，而不应当是含糊的、模棱两可的。所谓确定性是指算法满足具体问题的要求，对于相同的数据输入必然有相同的输出结果。

（3）可行性　算法的每一步都是能够实现的、可操作的，并得到确定的结果。也常称为算法的有效性。

（4）输入　一个算法一般有零个或多个数据输入。在计算机上实现的算法，一般是用来处理数据的，在大多数情况下这些数据需要通过不同的输入设备输入到存储器，也可以没有数据输入。

（5）输出　一个算法一般有一个或多个输出。算法的目的就是为了求问题的“解”，这些解只有通过输出才能得到。为此一个算法必须有数据输出，没有输出的算法是没有任何意义的。

1.2.2 算法设计的要求

一个算法的设计需要注意以下四个方面：

（1）正确性　算法的正确性包含四个层次，即算法不含语法错误、算法对于几组输入数据能够得出满足要求的结果、算法对于精心选择的典型及苛刻而带有刁难性的几组输入数据能够得出满足要求的结果、算法对于一切合法的输入数据都能产生满足要求的结果。

（2）可读性　算法主要是用于人的阅读与交流，可读性好有助于人们对算法的理解与掌握。

（3）健壮性　算法具有抵御“恶劣”输入信息的功能，当输入数据非法时，算法应适当地做出反应或进行处理，而不会产生错误的输出结果。

（4）高效率和低存储量　效率指的是算法的执行时间，对于解决同一问题的多个算法，

执行时间短的算法效率高。存储量是指算法执行过程中所需要的最大存储空间。两者都与问题的规模有关。

1.2.3　算法的描述

一个算法可以用自然语言、计算机语言、数学语言、流程图、PAD 图等来描述，或者用介于自然语言和计算机语言之间的伪代码来描述。对这些算法描述方法的唯一要求是必须精确地描述计算过程或步骤。好的描述方法使算法表达更加清晰和简洁。本书仅仅介绍几种常用的描述方法。

1. 用自然语言描述算法

自然语言就是人们日常生活、工作中使用的语言，可以是汉语、英语或其他语言。

例 1.1　求 n 的阶乘。

如果 $n=6$，即求 $1\times2\times3\times4\times5\times6$ 连乘的积。设 s 代表连乘之积，t 代表乘数，以自然语言描述求 n 的阶乘的算法：

步骤 1：使 $s=1$，$t=1$；

步骤 2：让 $s\times t$，得到的乘积仍放在 s 中；

步骤 3：使 t 的值增加 1；

步骤 4：如果 $t\leqslant n$ 那么转到步骤 2 继续执行；如果 $t>n$ 那么转到步骤 5；

步骤 5：连乘结束，s 中的值就是 n 的阶乘，输出 s。

用自然语言描述算法通俗易懂，但自然语言描述风格自由，比较适合描述简单问题的求解算法。如果问题复杂，描述会比较繁杂、冗长，并且也很难清楚地表达复杂逻辑的算法，比如判断和循环，容易出现“歧义性”，往往需要根据上下文判断语句的含义，这样算法很难进行准确地设计、交流和使用。

2. 用计算机语言描述算法

计算机高级程序设计语言具有很强的描述能力，接近于自然语言和数学语言。但是用计算机语言描述的算法实际上就是对算法的实现，通过编译或解释后直接可以执行，这是算法设计的终极目的，这样违反了算法描述方法所需要的简单、直观、抽象级别高的要求，所以在算法设计过程中用计算机语言描述算法是不合适的。

3. 用流程图描述算法

流程图是用一些几何图形、带方向的线条以及问题说明来描述算法操作步骤的方法。该方法形象、简明直观、易于理解、便于交流，是描述算法的主要方法之一。流程图分为传统的流程图、结构化流程图、N-S 流程图等多种。

(1) 传统流程图　传统流程图又称程序框图，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，如图 1-2 所示，这种符号目前已经被大多数国家接受和使用。

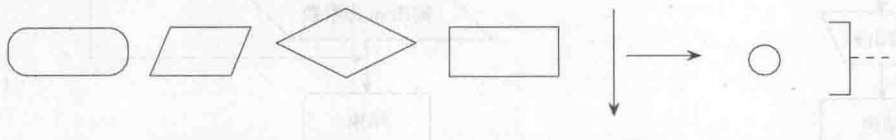

图 1-2　流程图常用符号

说明：

“起止框”用在流程图的开始和结束位置，标志算法的开始和结束；

“输入/输出框”的作用是给出执行中需要输入的数据信息和输出算法的结果；

“判断框”的作用是对一个给定的条件进行判断，根据给定的条件是否成立来决定如何执行其后的操作，它有一个入口，两个出口；

“处理框”表示算法中的各种具体操作；

“流程线”表示各个操作步骤的执行顺序，用箭头表示操作的先后顺序；

“连接符”用在当流程图较大，一页画不下时流向本流程图外某个地方的出口点，或从流程图外的某个地方进入的入口点；

“注释框”用于对框内信息的进一步说明，使问题或处理更加清晰。

图 1-3 为求 n 的阶乘的流程图，与自然语言描述方法比较，它的好处是更易懂，便于初学者掌握。传统流程图是历史最悠久使用最广泛的描述算法的工具。

例 1.2 判断某数 m（大于等于 3 的正整数）是否为素数的算法。

素数是一个自然数，它除了 1 和本身之外，不能被任何其他整数整除。判断一个数 m 是否为素数的方法很简单，如果 m 不能为 2 至 $m-1$ 的所有整数 i 整除则为素数，如果被其中任何一个数整除则不是素数。判断 m 为素数的流程图如图 1-4 所示。

（2）结构化流程图　传统流程图用流程线指出各框操作的先后执行顺序，它对流程线没

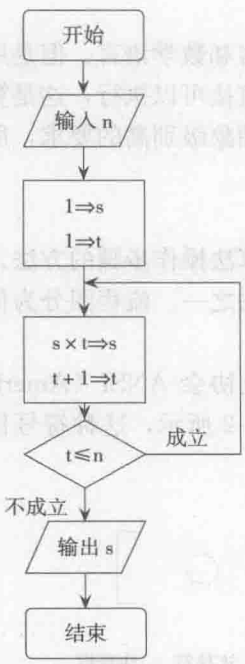

图 1-3　求 n 阶乘的流程图

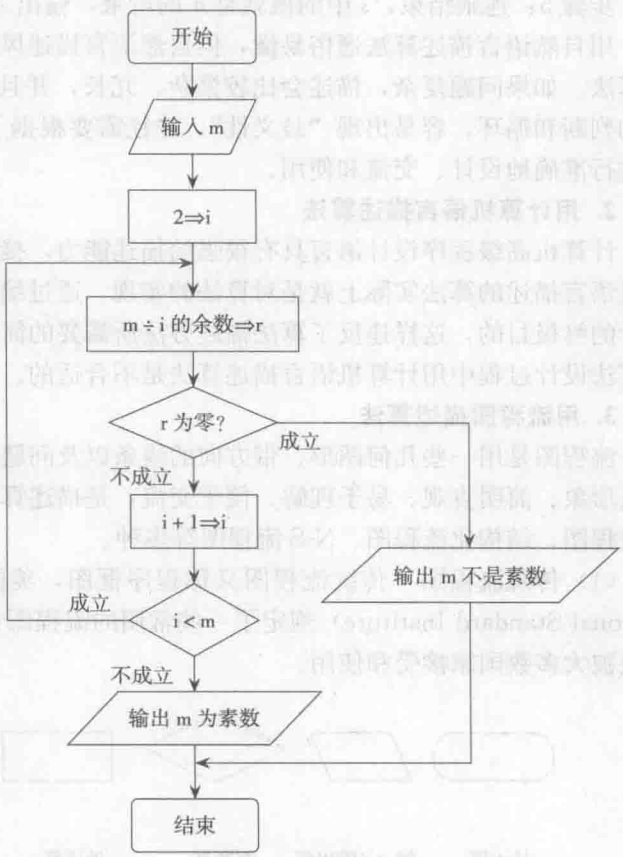

图 1-4　判断 m 是否为素数的流程图

有严格的限制，如图 1-4 所示。如果问题复杂，流程随意转向，流程线可能会非常密集、交叉较多，从而使人们在阅读算法时需要花费许多精力去追踪操作流程，使人们难以阅读、修改，算法的可靠性和可维护性难以保证。1966 年，C. Bohm 和 G. Jacopini 提出了顺序结构、分支结构、循环结构这 3 种基本结构，如图 1-5 所示，限制了箭头的使用，解决了传统流程图的弊端。现在各种程序设计语言都支持这三种结构，它们可以解决任何可计算的复杂问题。

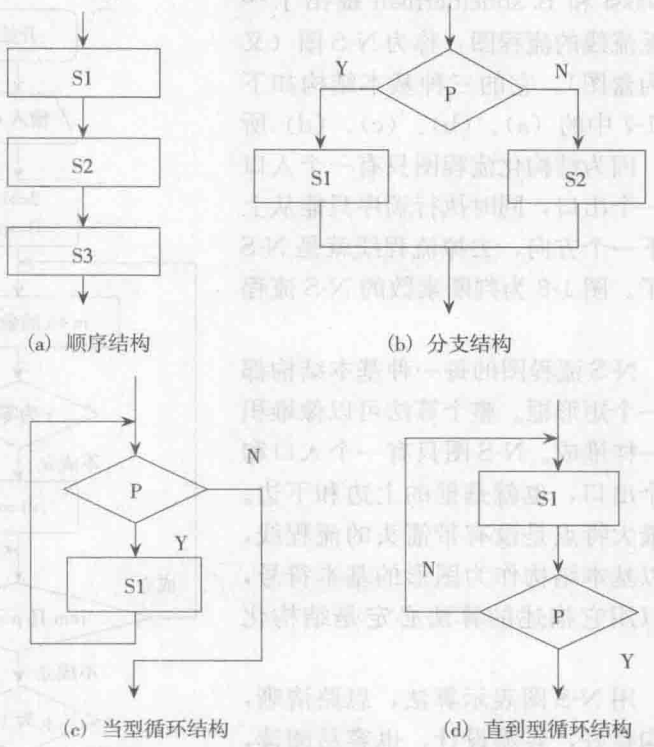

图 1-5　三种基本结构的流程图

说明：

“顺序结构”严格按照 S1（可以是一条语句，也可以是多条语句，即复合语句，后面相同）、S2、S3 的先后顺序执行，也就是 S1 执行完才能执行 S2，S2 执行完才能执行 S3。顺序结构是最简单的程序结构，也是程序中最常用的结构；

“分支结构”也称为选择结构，根据判断框中条件 P 的真假值来决定执行的顺序。如果条件 P 成立（真，用 Y 表示），执行 S1；如果条件 P 不成立（假，用 N 表示），执行 S2；

“循环结构”也称为重复结构，通常用来重复执行某些语句，分为当型循环结构和直到型循环结构两种。“当型循环结构”是先判断循环条件 P，如果条件成立则执行循环体 S1，再返回 P 的判断，条件成立循环执行，当条件 P 不成立时，跳出循环执行下面的程序。“直到型循环结构”是先执行循环体 S1，再判断条件 P，如果条件不成立继续执行循环体，直到条件成立时跳出循环。

三种基本结构即顺序结构、分支结构、循环结构，它们的共同特点是：只有一个入口，一个出口，限制了无规律的流程随意转向，只能顺序地向下执行。每个基本结构中的每一部分都有机会被执行到，结构内部不存在死循环。使用三种基本结构描述的算法是结构化的算法，按照结构化算法编写出来的程序具有良好的可读性和可维护性。

修改图 1-4 为结构化流程图需要增设一个条件 p（prime 的简写），初始时 p 为 0，如果在循环中 m 能够被代表 2 至 m－1 之间的数 i 整除，则设 p 为 1，表示 m 不是素数；如图 1-6 所示。从图中可以看出程序结构是嵌套的结构，判断整除的余数是否为零的选择结构成为直到型循环结构的一个语句，在循环结构之后是一个判断结构，根据循环结构设置的条件 p 来判断是否为素数并进行输出。

（3）N-S 流程图　为了简化控制流、适应“结构化”算法的要求，1973 年美国学者