C语言程序设计
——基于计算思维培养

杨俊生　谭志芳　王兆华　编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

工业和信息产业科技与教育专著出版资金项目

C 语言程序设计

——基于计算思维培养

杨俊生　谭志芳　王兆华　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书是以培养计算思维为导向的“C 语言程序设计课程改革”项目规划教材，由多年从事 C 语言程序设计教学的一线教师编写。本书分为 11 章，主要内容包括程序设计基础、C 语言概述、数据类型与表达式、顺序结构程序设计、选择结构程序设计、循环结构程序设计、数组、函数、指针、结构体与共用体、文件。

本书内容全面，由浅入深，循序渐进，在注重语言知识讲解的同时，更注重逻辑思维能力、程序设计能力、初步的算法设计能力、自主学习能力的培养。本书免费提供电子课件，可登录华信教育资源网（www.hxedu.com.cn）注册后下载。

本书可作为高等学校程序设计课程的入门教学用书，也可作为专科及成人教育的培训教材和教学参考书。

图书在版编目（CIP）数据

C 语言程序设计：基于计算思维培养 / 杨俊生，谭志芳，王兆华编著. —北京：电子工业出版社，2015.3

ISBN 978-7-121-25092-7

Ⅰ. ①C… Ⅱ. ①杨… ②谭… ③王… Ⅲ. ①C 语言—程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 290652 号

策划编辑：袁　玺
责任编辑：郝黎明
印　　刷：北京京师印务有限公司
装　　订：北京京师印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：19.75　字数：505.6 千字
版　　次：2015 年 3 月第 1 版
印　　次：2015 年 3 月第 1 次印刷
定　　价：42.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010) 88258888。

反侵权盗版声明

前　言

本书符合教育部高等学校计算机基础课程教学指导委员会2011版《高等学校计算机基础核心课程教学实施方案》的基本要求，符合学生学习的认知规律，是工业和信息产业科技与教育专著出版资金项目的规划教材。

本书突出“厚基础、重思维、提倡自主学习、注重能力培养”教学理念和指导思想，主要表现在以下几个方面。

（1）突出科学思维意识和能力的培养。教材加入了算法设计方法、常见经典算法、程序设计方法等与科学思维相关的内容；每章后的小结除了对本章语法要点、常见错误总结外，还着重对本章所涉及的能力点、典型算法、思维或算法设计方法进行了总结，并以思维导图的形式给出。

（2）重视拓展和探究性教学，培养学生自主学习能力。教材每章后都有一个探究性题目，用来引导学生通过查阅相关资料，综合运用所学知识完成一个难度稍大的题目，从而培养学生自主学习的能力；教师可根据不同的教学对象和教学要求对探究性题目进行取舍，便于开展因材施教；书中提供了大量的思考或自主学习题目，鼓励学生独立动手动脑，通过自己的努力拓展教材中所学知识。

（3）提升学生综合运用所学知识编写程序的能力。通过引入一个贯穿整本书的综合案例，使学生对使用计算机来解决实际问题的过程有一个切实的、整体的认识。

（4）注重编程逻辑的培养。通过引入Microsoft Visio 2010、RAPTOR等可视化算法设计工具，突出学生思维逻辑的培养，使学生的注意力集中在算法的设计上。

（5）从程序设计者的角度而不是从阅读者的角度来设计本书的例子，采用“提出问题→分析问题→设计算法→程序实现→测试→总结、优化或扩展深化（以讨论或思考题的方式）”形式来描述例子，从而达到启发读者编程思路，培养逻辑思维能力的目的。

（6）为了学生能更好、更快地适应市场的需求，本书在“函数”一章中增加了工程化开发程序的方法，从工程组织的角度介绍了规模稍大的多文件程序的科学合理的组织形式。

（7）为了拓宽并启发学生设计算法和程序时从多角度考虑问题，对同一任务采用了多种设计方式。如第1章中的猴子吃桃采用了递推算法，而在“函数”一章中采用了递归算法实现。

（8）将学生容易犯错的地方，以特殊格式突出显示了注意事项，避免学生在细节上浪费时间。

（9）为了满足课堂教学和教师备课的需要，教材配有电子课件，登录华信教育资源网（www.hxedu.com.cn）注册后免费下载。

本书内容全面，由浅入深，循序渐进，在打好“基础知识、基本技能”的基础上，注重培养学生的逻辑思维能力、程序设计能力、初步的算法设计能力、自主学习能力。

本书由杨俊生、谭志芳和王兆华编著。第1、7、9、11章由杨俊生编写，第2、4、5、8章由谭志芳编写，第3、6、10章由王兆华编写。

由于编者水平有限，书中难免有不妥之处，敬请读者批评指正。

编著者

目　录

第1章　程序设计基础

1.1　引例

从普通的办公自动化、上网冲浪到基因测序、嫦娥三号发射，都能看到计算机的影子；从智能手机中的单片机到“天河2号”超级计算机，各种计算机层出不穷。可以这么说，当今世界的各行各业都离不开计算机。那么人类是怎样“命令”计算机完成特定任务的呢？

通常情况下，人类是通过程序去“命令”计算机按照自己的意图工作的，这里大致存在两种情况：一是程序已经存在，如 Microsoft Word，专业人员使用该程序操作计算机完成相应的工作；二是程序不存在，需要组织一个团队去编写具有所需功能的程序。本书讨论的是第二种情况。

1.1.1　软硬件基础

“工欲善其事，必先利其器”。无论是编写程序还是运行程序，都需要一定的软硬件基础。

1．开发环境

开发环境就是编写程序所需的工具以及该工具运行的环境，主要包括：

（1）满足性能要求的计算机系统，包括计算机硬件和操作系统；

（2）开发工具，包括编译器、函数库、数据库等；

（3）数据库管理系统；

（4）必要的网络支持。

2．运行环境

运行环境是编写完成的应用程序运行的环境，主要包括：

（1）满足性能要求的计算机系统，包括计算机硬件和操作系统；

（2）程序运行必需的函数库、数据库系统等；

（3）必要的网络支持。

1.1.2　编写程序

具备必要的开发环境后，就要编写相应的程序来“命令”计算机完成特定的任务。下面就以模拟控制“玉兔号”月球车移动为例，来简要说明编写程序的大致过程。

【例1.1】编写程序来控制“玉兔号”月球车的移动。

通过仔细分析这个问题，将月球表面抽象为一个平面直角坐标系（图1.1.1），月球车处于某点（x，y）处，每一点的横、纵坐标均为整数。月球车可以实现4种运动方式，即向前、向后、向左、向右方每次走一个单位。

设定x、y、direction三个变量，其中（x，y）来表示“玉兔号”月球车当前坐标，direction来表示月球车接收到的移动命令。假设向前移动纵坐标值增大，向后

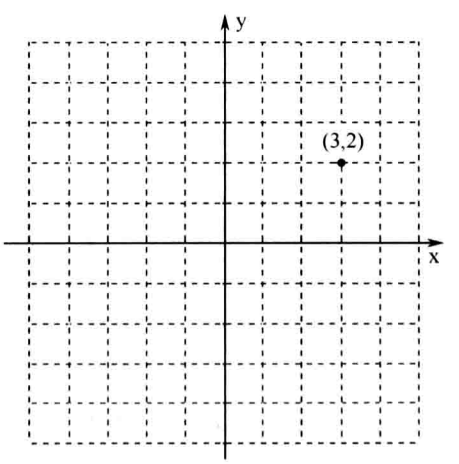


图1.1.1　抽象的平面直角坐标系

移动纵坐标值减小，向左移动横坐标值减小，向右移动横坐标值增大，那么 x、y、direction 三个变量之间的关系如表 1.1.1 所示。

表 1.1.1　控制“玉兔号”月球车移动命令说明

direction	direction 含义	目标点横（纵）坐标值变化
1	向前	y = y + 1
2	向后	y = y - 1
3	向左	x = x - 1
4	向右	x = x + 1

接下来，列出解决该问题的主要步骤，即所谓的算法。

（1）输入月球车初始坐标值（x，y）。

（2）输入移动命令给 direction。

（3）如果 direction<1 或 direction>4，转步骤（5）；否则，按表 1.1.1 所示规则，分情况求得移动后目标点的坐标值。

（4）输出求得的目标点的坐标值（x，y），转步骤（2）

（5）算法结束。

在设计好算法后，应该使用流程图作为工具来描述算法，如图 1.1.2 所示。

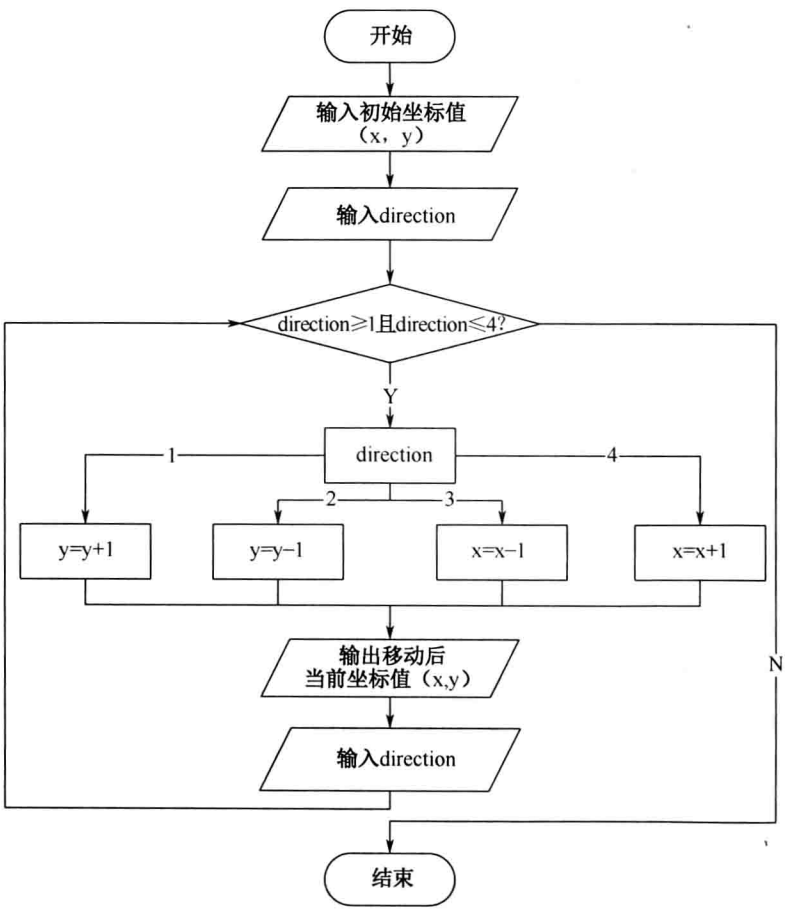


图 1.1.2　模拟控制“玉兔号”月球车移动流程图

为了实现以上的解决方案，使用 C 语言编写程序来实现该算法：

```
//该程序用来模拟控制"玉兔号"月球车的移动
#include<stdio.h>
int main()
{
    int x,y,direction;
    printf("请输入月球车的初始坐标（x, y）\n");
    scanf("%d,%d",&x,&y);
    printf("请输入移动方向（1-前  2-后  3-左  4-右）：");
    scanf("%d",&direction);
    while(direction>=1&&direction<=4)
    {
        switch(direction)
        {
        case  1:y=y+1;break;        //前
        case  2:y=y-1;break;        //后
        case  3:x=x-1;break;        //左
        case  4:x=x+1;break;        //右
        }
        printf("月球车当前坐标为（%d, %d）\n",x,y);
        printf("请输入移动方向（1-前  2-后  3-左  4-右）：");
        scanf("%d",&direction);
    }
    return 0;
}
```

在编写好程序后，应该仔细检查程序是否能够解决问题。

1.2 算法

由上面的介绍可知，设计算法是人们编写程序去“命令”计算机解决问题过程中非常重要的一环，算法是计算机科学最基本的概念，是计算机科学研究的核心之一。因此，了解算法及其表示和设计方法是程序设计的基础和精髓，也是读者学习程序设计过程中最重要的一环。

1.2.1 算法及其特性

1．什么是算法

算法（Algorithm）就是一组有穷的规则，它规定了解决某一特定问题的一系列运算。通俗地说，为解决问题而采用的方法和步骤就是算法。本书中讨论的算法主要是指计算机算法。

2．算法的特性

（1）确定性（Definiteness）。算法的每个步骤必须要有确切的含义，每个操作都应当是清晰的、无二义性的。例如，算法中不允许出现诸如“将 3 或 5 与 y 相加”等含混不清、具有歧义的描述。

（2）有穷性（Finiteness）。一个算法应包含有限的操作步骤且在有限的时间内能够执行完毕。例如，在计算下列近似圆周率的公式时：

$$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots \tag{1.2.1}$$

当某项的绝对值小于 10^{-6} 时算法执行完毕。

> **注意：如何正确理解算法的有穷性？**
>
> 一个实用的算法，不仅要求步骤有限，同时要求运行这些步骤所花费的时间是人们可以接受的。例如，使用暴力破解密码的算法可能要耗费成百上千年。显而易见，这个算法是可以在有限的时间内完成，但是对于人类来说是无法接受的。

（3）有效性（Effectiveness）。算法中的每个步骤都应当能有效地执行，并得到确定的结果。例如，算法中包含一个 m 除以 n 的操作，若除数 n 为 0，则操作无法有效地执行。因此，算法中应该增加判断 n 是否为 0 的步骤。

（4）有零个或多个输入（Input）。在算法执行的过程中需要从外界取得必要的信息，并以此为基础解决某个特定问题。例如，在求两个整数 m 和 n 的最大公约数的算法中，需要输入 m 和 n

的值。另外，一个算法也可以没有输入，例如，在计算式（1.2.1）时，不需要输入任何信息，就能够计算出近似的π值。

（5）有一个或多个输出（Output）。设计算法的目的就是要解决问题，算法的计算结果就是输出。没有输出的算法是没有意义的。输出与输入有着特定的关系，通常，输入不同，会产生不同的输出结果。

☞注意：如何正确理解算法的输出形式？

算法的输出就是算法的计算结果，其输出形式多种多样：打印数值、字符、字符串，显示一幅图片，播放一首歌曲或音乐，播放一部电影……

3．算法的分类

根据待解决问题的形式模型和求解要求，算法分为数值和非数值两大类。

（1）数值运算算法。数值运算算法是以数学方式表示的问题求数值解的方法。例如，代数方程计算、线性方程组求解、矩阵计算、数值积分、微分方程求解等。通常，数值运算有现成的模型，这方面的现有算法比较成熟。

（2）非数值运算算法。非数值运算算法通常为求非数值解的方法。例如，排序、查找、表格处理、文字处理、人事管理、车辆调度等。非数值运算算法种类繁多，要求各自不同，难以规范化。本节主要讲述的是一些典型的非数值运算算法。

1.2.2 算法的表示方法

设计出一个算法后，为了存档，以便将来算法的维护或优化，或者为了与他人交流，让他人能够看懂、理解算法，需要使用一定的方法来描述、表示算法。算法的表示方法很多，常用的有自然语言、流程图、伪代码和程序设计语言等。

1．自然语言（Natural Language）

用人们日常生活中使用的语言，如中文、英文、法文等来描述算法。

【例 1.2】使用中文来描述计算 5！的算法，其中假设变量 t 为被乘数，变量 i 为乘数。

（1）初始化 t=1。

（2）初始化 i=2。

（3）计算 t×i，乘积仍放在 t 中。

（4）将 i 的值加 1 再放回到 i 中。

（5）如果 i 不大于 5，则返回步骤（3）执行；否则，进入步骤（6）。

（6）输出 t 中存放的 5!值。

使用自然语言描述算法的优点是通俗易懂，没有学过算法相关知识的人也能够看懂算法的执行过程。但是，自然语言本身所固有的不严密性使得这种描述方法存在以下缺陷：

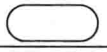
① 文字冗长，容易产生歧义性，往往需要根据上下文才能判别其含义；

② 难以描述算法中的分支和循环等结构，不够方便、直观。

2．流程图（Flow Chart）

流程图是最常见的算法图形化表达，它使用美国国家标准化学会（American National Standards Institute，ANSI）规定的一些图框、线条来形象、直观地描述算法处理过程。常见的流程图符号如表 1.2.1 所示。

表 1.2.1　常见流程图符号

符号名称	图　形	功　能
起止框		表示算法的开始或结束
处理框		表示一般的处理操作，如计算、赋值等

续表

符号名称	图　形	功　能
判断框		表示对一个给定的条件进行判断
流程线		用流程线连接各种符号，表示算法的执行顺序
输入/输出框		表示算法的输入/输出操作
连接点		成对出现，同一对连接点内标注相同的数字或文字，用于将不同位置的流程线连接起来，避免流程线的交叉或过长

【例 1.3】使用流程图来描述计算 5!的算法，其中假设变量 t 为被乘数，变量 i 为乘数。流程图如图 1.2.1 所示。

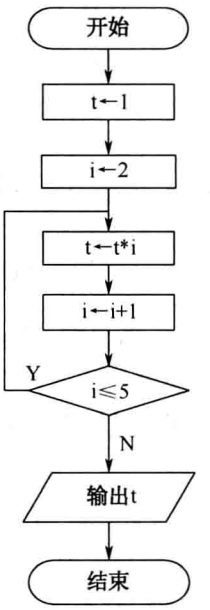


图 1.2.1　计算 5!的流程图

从本例可以看出，使用流程图描述算法简单、直观，能够比较清楚地显示出各个符号之间的逻辑关系，因此流程图是一种表示算法的好工具。流程图使用流程线指出各个符号的执行顺序，对流程线的使用没有严格限制，使用者可以毫无限制地使流程随意地转来转去。但是，当算法规模较大，操作比较复杂时，人们难以理解算法的逻辑。

为了提高算法的质量，便于阅读理解，应限制流程的随意转向。为了达到这个目的，人们规定了 3 种基本结构，由这些基本结构按一定规律组成一个算法结构。

（1）顺序结构。顺序结构是最简单、最常用的一种结构，如图 1.2.2 所示。图中操作 A 和操作 B 按照出现的先后顺序依次执行。

（2）选择结构。选择结构又称为分支结构。这种结构在处理问题时根据条件进行判断和选择。图 1.2.3（a）是一个“双分支”选择结构，如果条件 p 成立则执行处理框 A，否则执行处理框 B。图 1.2.3（b）是一个“单分支”选择结构，如果条件 p 成立则执行处理框 A。

（3）循环结构。循环结构又称为重复结构，在处理问题时根据给定条件重复执行某一部分的操作。循环结构有当型和直到型两种类型。

当型循环结构如图 1.2.4 所示。功能是：当条件 p 成立时，执行处理框 A，执行完处理框 A 后，再判断条件 p 是否成立，若条件 p 仍然成立，则再次执行处理框 A，如此反复，直至条件 p 不成立才结束循环。

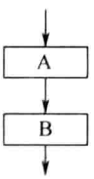


图 1.2.2　顺序结构

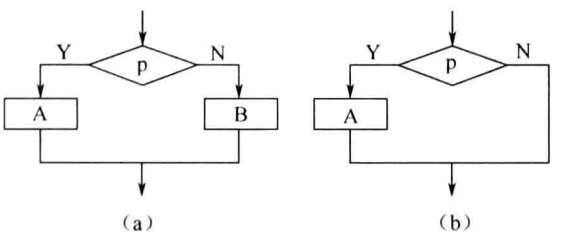


图 1.2.3　选择结构

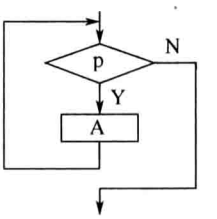


图 1.2.4　当型循环结构

直到型循环结构如图 1.2.5 所示。功能是：先执行处理框 A，再判断条件 p 是否成立，如果条件不成立，则再次执行处理框 A，如此反复，直至条件 p 成立才结束循环。

当型循环结构与直到型循环结构的区别如表 1.2.2 所示。

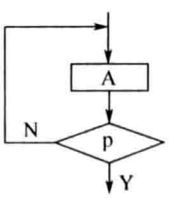


图 1.2.5　直到型循环结构

表 1.2.2　当型循环结构与直到型循环结构的比较

比 较 项 目	当型循环结构	直到型循环结构
何时判断条件是否成立	先判断，后执行	先执行，后判断
何时执行循环	条件成立	条件不成立
循环至少执行次数	0 次	1 次

以上 3 种基本结构具有以下的特点：

（1）只有一个入口，只有一个出口；

（2）结构中的每一部分都有机会被执行到；

（3）结构内不存在“死循环”。

计算机科学家已经证明，使用以上 3 种基本结构顺序组合而成的算法结构，可以解决任意复杂的问题。由基本结构所构成的算法就是所谓的“结构化”的算法。

自主学习：N-S 流程图

N-S 流程图是由美国学者 I.Nassi 和 B.Shneiderman 在 1973 年提出的一种新的流程图形式。读者可以上网查阅相关资料深入学习这种算法表示方法。

3．伪代码（Pseudocode）

虽然使用流程图来描述算法简单、直观，易于理解，但是画起来费事，修改起来麻烦。因此流程图比较适合于算法最终定稿后存档时使用，而在设计算法的过程中常用一种称为“伪代码”的工具。

伪代码是一种介于自然语言和程序设计语言之间描述算法的工具。程序设计语言中与算法关联度小的部分往往被伪代码省略，如变量的定义等。

【例 1.4】使用伪代码来描述计算 5!的算法，其中假设变量 t 为被乘数，变量 i 为乘数。

```
begin
    t←1
    i←2
    while(i≤5)
    {
        t←t*i
        i←i+1
    }
    print t
end
```

4．程序设计语言（Programming Language）

计算机无法识别自然语言、流程图、伪代码，因此算法最终要用程序设计语言实现，再被翻译成可执行程序后在计算机中执行。用程序设计语言描述算法必须严格遵守所选择语言的语法规则。

【例 1.5】使用 C 语言来描述计算 5!的算法。

```
#include<stdio.h>
int main()
{
    int i,t;
    t=1;
    i=2;
    while(i<=5)
    {
        t=t*i;
        i=i+1;
    }
    printf("5!=%d\n",t);
    return 0;
}
```

本节介绍了 4 种算法描述方法，读者可根据自己的喜好和习惯，选择其中一种。建议在设计算法过程中使用伪代码，交流算法思想或存档算法时使用流程图。

1.2.3 算法设计的基本方法

算法设计的任务是对各类问题设计良好的算法及研究设计算法的规律和方法。常用的数值算法设计方法包括迭代法、插值法、差分法等；非数值算法设计方法包括分治法、贪心法、回溯法等；还有些方法既适用于数值算法的设计也适用于非数值算法的设计。本书主要介绍后两种情况下算法设计的基本方法，数值算法的设计方法请读者参考《计算方法》、《数值分析》等课程的教材。

针对一个给定的实际问题，要找出确实行之有效的算法，就需要掌握算法设计的策略和基本方法。算法设计是一个难度较大的工作，初学者在短时间内很难掌握。但所幸的是，前人通过长期的实践和研究，已经总结出了一些算法设计的基本策略和方法，如穷举法、递推法、递归法、分治法、回溯法、贪心法、模拟法和动态规划法等。

1．穷举法（Exhaustive Algorithm）

穷举法（Exhaustive Algorithm）也称为枚举法、蛮力法，是一种简单、直接解决问题的方法。

使用穷举法解决问题的基本思路：依次穷举问题所有可能的解，按照问题给定的约束条件进行筛选，如果满足约束条件，则得到一组解，否则不是问题的解。将这个过程不断地进行下去，最终得到问题的所有解。

要使用穷举法解决实际问题，应当满足以下两个条件：

（1）能够预先确定解的范围并能以合适的方法列举；

（2）能够对问题的约束条件进行精确描述。

穷举法的优点是：比较直观，易于理解，算法的正确性比较容易证明；缺点是：需要列举许多种状态，效率比较低。

【例 1.6】使用流程图描述判断一个正整数 m（m≥2）是否为素数的算法。

素数也称为质数，其特点是，除了 1 和它自身之外没有其他的约数。例如，19 除了 1 和 19 之外没有其他约数，因此它就是一个素数。

通过观察和分析，该问题的求解符合穷举法解决问题的条件。

① 给定正整数 m 所有可能的约数(除了 1 和它自身)在[2,m−1]区间内，而且每两个可能的约数之间差 1。

② 约束条件非常容易描述：m mod n = 0，其中 n 表示正整数 m 的所有可能的约数，mod 的含义是求 m 除以 n 的余数，即求余运算，高级程序设计语言中基本都有该运算符。

判别素数算法的基本思路是，从区间[2,m−1]中取出一个数，看它是否满足约束条件 m mod n=0，如果满足，那么根据素数的定义可知，m 不是素数，算法结束；否则继续判别下一个可能的约数 n+1。重复上述过程，直到区间[2,m−1]中每个数都被判断过，并且都不是 m 的约数为止，这说明 m 是素数。流程图如图 1.2.6 所示。

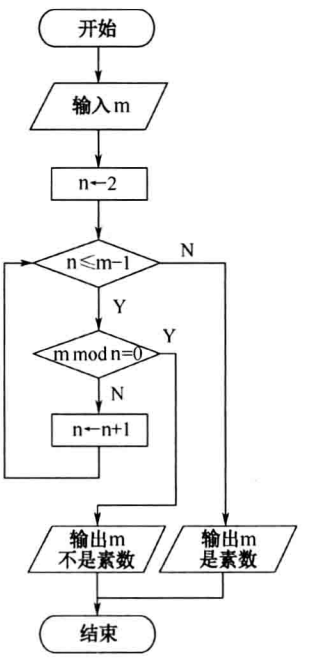


图 1.2.6　判别素数流程图

☞ 思考：如何优化判断素数算法，使其比较次数大幅减少？

2．递推法（Recurrence）

递推法是一种重要的算法设计思想。一般是从已知的初始条件出发，依据某种递推关系，逐次推出所要求的各中间结果及最后结果。其中初始条件可能由问题本身给定，也可能是通过对问

题的分析与化简后确定。在实际应用中，题目很少会直接给出递推关系式，而是需要通过分析各种状态，找出递推关系式，这也是应用递推法解决问题的难点所在。

在日常应用中，递推算法可分为顺推法和逆推法两种。

（1）顺推法。从已知条件出发，逐步推算出要解决问题的方法。例如，斐波那契数列就可以通过顺推法不断推算出新的数据。

（2）逆推法。逆推法也称为倒推法，是顺推法的逆过程。该方法从已知的结果出发，用迭代表达式逐步推算出问题开始的条件。

【例 1.7】使用顺推法解决斐波那契数列问题。

1202 年，意大利数学家斐波那契在他的著作《算盘书》中提出了一个有趣的问题：有一对兔子，从出生后第 3 个月起每个月都生 1 对小兔子。小兔子长到第 3 个月后每个月又生 1 对小兔子。假定在不发生死亡的情况下，由 1 对刚出生的兔子开始，1 年能繁殖出多少对兔子？

解题思路：

（1）首先，将兔子按照出生的月数分为 3 种：满 2 个月以上的兔子为老兔子；满 1 个月但是不满 2 个月的兔子为中兔子；不满 1 个月的兔子为小兔子。其中，只有老兔子具有繁殖能力。

（2）第 1 个月时，只有 1 对小兔子，总数为 1 对；

（3）第 2 个月时，1 对小兔子长成了 1 对中兔子，总数仍为 1 对；

（4）第 3 个月时，1 对中兔子长成了老兔子，并繁殖了 1 对小兔子，总数是 2 对；

（5）第 4 个月时，1 对小兔子长成了中兔子，原来的 1 对老兔子又繁殖了 1 对小兔子，总数是 3 对；

（6）以此类推，兔子的繁殖过程如表 1.2.3 所示。

表 1.2.3　兔子繁殖过程

月份	小兔子对数	中兔子对数	老兔子对数	兔子总数
1	1	0	0	1
2	0	1	0	1
3	1	0	1	2
4	1	1	1	3
5	2	1	2	5
6	3	2	3	8
7	5	3	5	13
8	8	5	8	21
9	13	8	13	34
10	21	13	21	55
11	34	21	34	89
12	55	34	55	144

解答：从表 1.2.3 可以看到每个月的兔子总数依次是 1,1,2,3,5,8,13,21,34,55,89,144,…，这就是著名的斐波那契数列。这个数列有着非常明显的特点，即从第 3 项开始，每一项都是其前面相邻两项之和。使用数学公式来表示：

$$\begin{cases} F_1 = 1 & (n = 1) \\ F_2 = 1 & (n = 2) \\ F_n = F_{n-1} + F_{n-2} & (n \geqslant 3) \end{cases} \tag{1.2.2}$$

从以上的分析可知，斐波那契数列可使用递推算法来计算求得，式（1.2.2）就是递推关系式。图 1.2.7 就是使用顺推法解决斐波那契数列问题的流程图（流程图中略去了输出斐波那契数列的部分）。