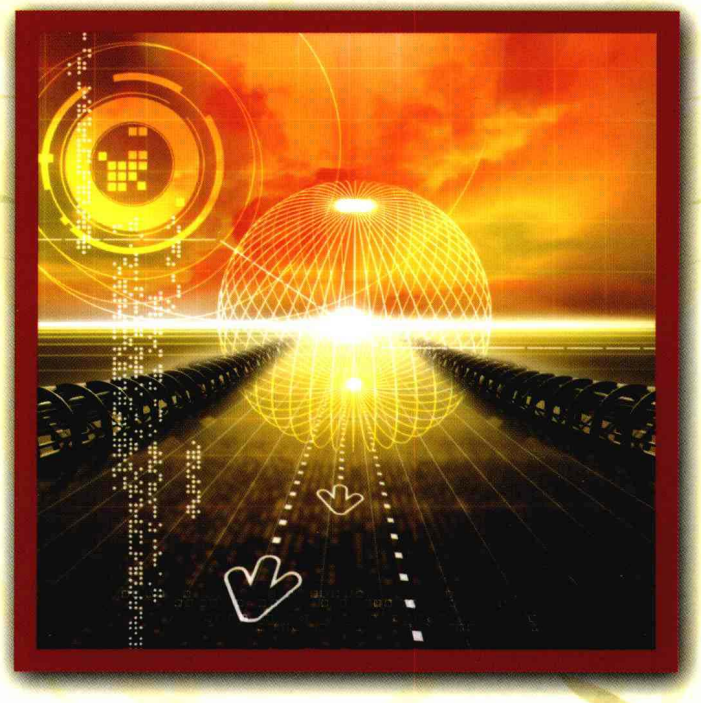
高等学校计算机教育系列规划教材

C语言程序设计

刘卫国　主编

贾宗福　沈根海　石玉晶　习胜丰　副主编



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

高等学校计算机教育系列规划教材

C 语言程序设计

刘卫国　主　编

贾宗福　沈根海　副主编

石玉晶　习胜丰

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书介绍C语言基础知识及其程序设计的基本方法，使读者掌握计算机程序设计的思想、方法和技术，具有利用C语言进行程序设计的能力和较强的计算机应用开发能力。全书内容包括C语言程序设计概述、C语言的基本数据类型与运算、顺序结构程序设计、选择结构程序设计、循环结构程序设计、函数与编译预处理、数组、指针、结构体、共用体与枚举以及文件操作等。

本书内容丰富，理论与实践相结合，强调程序设计方法与能力的培养。在编写过程中，力求做到概念清晰、取材合理，深入浅出、突出应用，为学生应用C语言进行程序设计和软件开发打下良好基础。

本书适合作为高等院校计算机程序设计课程的教材，也可供社会各类软件开发人员阅读参考。

图书在版编目（CIP）数据

C语言程序设计/刘卫国主编. —北京：中国铁道出版社，2008.1

（高等学校计算机教育系列规划教材）

ISBN 978-7-113-08549-0

Ⅰ.C… Ⅱ.刘… 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（2008）第014219号

书　　名：C语言程序设计

作　　者：刘卫国 等

出版发行：中国铁道出版社（100054，北京市宣武区右安门西街8号）

策划编辑：严晓舟　秦绪好

责任编辑：李　旸　姚文娟

封面设计：付　巍

封面制作：白　雪

印　　刷：北京市兴顺印刷厂

开　　本：787×1092　1/16　　**印张**：21.25　　**字数**：497千

版　　本：2008年2月第1次出版　　2008年2月第1次印刷

印　　数：1～5 000册

书　　号：ISBN 978-7-113-08549-0/TP·2683

定　　价：30.00元

FOREWORD

前 言

计算机是在程序的控制下进行自动工作的，它解决任何实际问题都依赖于解决问题的程序。程序设计是计算机程序开发人员的一项基本功，只有掌握程序设计的基本知识，才能具有一定的应用开发能力。

教育部高等学校非计算机专业计算机基础课程教学指导分委员会于2004年提出“1+X”课程设置模式，即一门“大学计算机基础”和若干门核心课程，“计算机程序设计基础”是其中一门重要的核心课程。通过本课程的学习，使学生了解程序设计语言的基本知识，掌握程序设计的基本方法和常用算法，掌握程序调试的基本技能，具有使用计算机解决实际问题的基本能力，也为学习后续课程打下良好基础。

“计算机程序设计基础”是从技术角度学习计算机的主要基础课程，已经作为大多数专业的必修课。由于不同学校、不同专业对学生程序设计能力的要求不尽相同，所以程序设计课程通常采用不同的教学语言。但程序语言只是一种载体，一种学习程序设计的工具，无论选用哪种语言，都应掌握程序设计的基础知识与基本编程技术，学习的重点应落在程序设计的方法与应用开发能力上。

C语言是目前流行的程序设计语言之一，具有程序简洁、数据类型丰富、表达能力强、使用灵活、实用高效等优点，在当今软件开发领域有着广泛的应用。作者结合多年从事程序设计教学与软件开发的经验，适应计算机基础教学工作的需要，组织编写了本书。

本书介绍C语言的基本知识及其程序设计的基本方法，使读者掌握计算机程序设计的思想、方法和技术，具有利用C语言进行程序设计的能力和较强的计算机应用开发能力。在编写过程中，力求体现以下特点：

（1）教材内容重点突出，取材适当。教材用通俗易懂的语言讲清C语言的重要概念，加强程序设计和程序调试能力的训练，但教材不追求内容的全面性，对于初学者不常用到的内容（如“位运算”）作了简化处理。教材也不过分死抠语言细节，引导读者在实践中去掌握语法规则。

（2）教材的组织结构遵循循序渐进原则。教材前6章体现了基本程序设计能力的训练。第1章介绍程序设计的基本知识；第2章介绍基本数据类型与运算，这些运算许多是数学中熟悉的运算，但也有许多不一致的地方，所以要注意区别，学会C语言中数据的正确表达方法；第3～5章分别介绍程序的3种基本结构，体现了最基本的程序设计方法；第6章介绍函数，体现了模块化程序设计的需要。前6章只涉及C语言的基本数据类型，重点放在程序的3种基本结构的实现方法和程序设计能力培养上。第7～10章是数组、指针和C语言的构造数据类型，涉及更复杂数据的表达方法。第11章是文件操作，这是程序设计语言的经典内容。这种内容编排有利于总体上把握全书内容，帮助读者逐步深入理解和掌握课程知识。

（3）教材注重应用。语言是程序设计的工具，学习C语言是为了能够设计解决问题的程序。书中穿插介绍了递推法、迭代法、穷举法、试探法、递归法等算法设计策略，书中还包含了大量的应用实例，这些有利于读者掌握有关程序设计方法。有些例子更多地是从教学的角度设计的，这是应用的基础和前提，有些例子则具有很强的实际应用背景，可以更好地培养读者的应用开发能力。在语言编译系统的选择上，本书使用Visual C++ 6.0作为上机环境，目的是让教材内容更加接近软件开发的实际需要，为读者进一步学习和应用C++打下基础。

（4）教材有配套的教学参考书、教学课件与相关教学资源。为了方便教学和读者上机操作练习，作者还组织编写了《C语言程序设计实践教程》一书，作为与本书配套的教学参考书。《C语言程序设计实践教程》既与本教材相互配套，又是本教材很好的补充。例如，实践教程中常用算法设计方法、程序测试与调试等内容与程序设计实践息息相关，可作为读者课外的阅读材料，达到巩固与提高的目的。另外，还有与本书配套的教学课件与相关教学资源，供教师教学参考。

本书适合作为高等院校计算机程序设计课程的教材，也可供社会各类软件开发人员阅读参考。

本书由刘卫国任主编，贾宗福、沈根海、石玉晶、习胜丰任副主编。第1、8、9章由刘卫国编写，第2章由贾宗福编写，第3章由沈根海编写，第4章由石玉晶编写，第5章由童键编写，第6、7章由蔡立燕编写，第10章及附录由舒卫真编写，第11章由习胜丰编写。参与程序调试与资料整理的还有杨斌、刘勇、张志良、李斌、康维、罗站城、邹美群等。此外，本书还得到了中南大学信息科学与工程学院施荣华教授的支持与帮助，在此表示感谢。

由于编者学识水平有限，书中的疏漏或错误之处在所难免，恳请广大读者批评指正。

编 者

2008年1月

CONTENTS

目 录

第 1 章 概 述

自从 1946 年第一台电子计算机诞生以来，计算机及计算机科学与技术得到了迅猛的发展。如今，计算机正深刻地影响着人们的生活和工作。计算机可以解决很多实际问题，但它本身并无解决问题的能力，而必须依赖于人们事先编写好的程序（Program），而要编写程序就要熟悉一种程序设计语言以及掌握编写程序的方法。在众多的程序设计语言中，C 语言有其独到之处，具有程序简洁、数据类型丰富、表达能力强、使用灵活、实用高效等优点，在当今软件开发中有着广泛的应用。学习 C 语言程序设计的目的，就是要学会利用 C 语言编写出适合自己实际需要的程序，让计算机完成自己指定的任务。

本章介绍程序设计的基本知识、C 语言的发展与特点、C 程序的基本结构以及 C 程序的执行步骤。通过本章的学习，将使读者对程序设计和 C 语言有一个概要认识，从而为以后各章的学习打下基础。

1.1 程序设计基本知识

计算机是在程序控制下进行自动工作的，它解决任何实际问题都依赖于解决问题的程序。程序设计是计算机应用人员的一项基本功，只有掌握程序设计的知识，才能具有一定的应用开发能力。在学习 C 语言程序设计之前，需要了解一些程序设计的基本知识。

1.1.1 程序与程序设计

从一般意义来说，程序是对解决某个实际问题的方法和步骤的描述，而从计算机角度来说，程序是用某种计算机能理解并执行的语言所描述的解决问题的方法和步骤。计算机执行程序所描述的方法和步骤，并完成指定的功能。所以，程序就是供计算机执行后能完成特定功能的指令序列。

一个计算机程序主要描述两部分内容：一是描述问题的每个对象和对象之间的关系，二是描述对这些对象作处理的处理规则。其中关于对象及对象之间的关系是数据结构（Data Structure）的内容，而处理规则是求解的算法（Algorithm）。针对问题所涉及的对象和要完成的处理，设计合理的数据结构可有效地简化算法，数据结构和算法是程序最主要的两个方面。

程序设计的任务就是设计解决问题的方法和步骤（即设计算法），并将解决问题的方法和步骤用程序设计语言来描述。什么叫程序设计？对于初学者来说，往往把程序设计简单地理解为只是编写

一个程序，这是不全面的。程序设计反映了利用计算机解决问题的全过程，包含多方面的内容，而编写程序只是其中的一个方面。使用计算机解决实际问题，通常是先要对问题进行分析并建立数学模型，然后考虑数据的组织方式和算法，并用某一种程序设计语言编写程序，最后调试程序，使之运行后能产生预期的结果。这个过程称为程序设计（Programming）。具体要经过以下 4 个基本步骤：

（1）分析问题，确定数学模型或方法。要用计算机解决实际问题，首先要对待解决的问题进行详细分析，弄清问题的需求，包括需要输入什么数据，要得到什么结果，最后应输出什么。即弄清要计算机“做什么”。然后把实际问题简化，用数学语言来描述它，这称为建立数学模型。建立数学模型后，需选择计算方法，即选择用计算机求解该数学模型的近似方法。不同的数学模型，往往要进行一定的近似处理。对于非数值计算则要考虑数据结构等问题。

（2）设计算法，画出流程图。弄清楚要计算机“做什么”后，就要设计算法，明确要计算机“怎么做”。解决一个问题，可能有多种算法。这时，应该通过分析、比较，挑选一种最优的算法。算法设计后，要用流程图把算法形象地表示出来。

（3）选择编程工具，按算法编写程序。当为解决一个问题确定了算法后，还必须将该算法用程序设计语言编写成程序，这个过程称为编码（Coding）。

（4）调试程序，分析输出结果。编写完成的程序，还必须在计算机上运行，排除程序中可能的错误，直到得到正确结果为止。这个过程称为程序调试（Debugging）。即使是经过调试的程序，在使用一段时间后，仍然会被发现尚有错误或不足之处。这就需要对程序做进一步的修改，使之更加完善。

解决实际问题时，应对问题的性质与要求进行深入分析，从而确定求解问题的数学模型或方法，接下来进行算法设计，并画出流程图。有了算法流程图，再来编写程序就容易多了。有些初学者，在没有把所要解决的问题分析清楚之前就急于编写程序，结果编程思路紊乱，很难得到预想的结果。

1.1.2 算法及其描述

计算机是通过执行人们所编制的程序来完成预定的任务。在广义上说，计算机按照程序所描述的算法对某种结构的数据进行加工处理。著名的瑞士计算机科学家 N. Wirth 教授曾提出：

算法+数据结构=程序

算法是对数据运算的描述，而数据结构是指数据的组织存储方式，包括数据的逻辑结构和存储结构。程序设计的实质是对实际问题选择一种好的数据结构，并设计一个好的算法，而好的算法在很大程度上取决于描述实际问题的数据结构。

1. 算法的概念

在日常生活中，人们做任何一件事情，都是按照一定规则、一步一步地进行的，这些解决问题的方法和步骤称为算法。比如工厂生产一部机器，先把零件按一道道工序进行加工，然后，把各种零件按一定法则组装起来，生产机器的工艺流程就是算法。总之，任何数值计算或非数值计算过程中的方法和步骤，都称之为算法。

计算机解决问题的方法和步骤，就是计算机解题的算法。计算机用于解决数值计算，如科学计算中的数值积分、解线性方程组等的计算方法，就是数值计算的算法；用于解决非数值计算，如用于管理、文字处理、图形图像处理等的排序、分类、查找的方法，就是非数值计算的算法。要编写解决问题的程序，首先应设计算法，任何一个程序都依赖于特定的算法，有了算法，再来编写程序是容易的事情。

下面举两个简单例子，以说明计算机解题的算法。

【例 1.1】求 $u=\dfrac{x-y}{x+y}$，其中 $x=\begin{cases}a^2+b^2 & a<b\\ a^2-b^2 & a\geqslant b\end{cases}$，$y=\begin{cases}\dfrac{a+b}{a-b} & a<b\\ \dfrac{4}{a+b} & a\geqslant b\end{cases}$。

这一题的算法并不难，可写成：

（1）从键盘输入 a，b 的值。

（2）如果 $a<b$，则 $x=a^2+b^2$，$y=\dfrac{a+b}{a-b}$，否则 $x=a^2-b^2$，$y=\dfrac{4}{a+b}$。

（3）计算 u 的值：$\dfrac{x-y}{x+y}$。

（4）输出 u 的值。

【例 1.2】输入 10 个数，要求找出其中最大的数。

设 max 单元用于存放最大数，先将输入的第 1 个数放在 max 中，再将输入的第 2 个数与 max 相比较，较大者放在 max 中，然后将第 3 个数与 max 相比，较大者放在 max 中，……，一直到比完 9 次为止。

算法要在计算机上实现，还需要把它描述为更适合程序设计的形式，对算法中的量要抽象化、符号化，对算法的实施过程要条理化。上述算法可写成如下形式：

（1）输入一个数，存放在 max 中。

（2）用 i 来统计比较的次数，其初值置 1。

（3）若 i≤9，执行第（4）步，否则执行第（8）步。

（4）输入一个数，放在 x 中。

（5）比较 max 和 x 中的数，若 x>max，则将 x 的值送给 max，否则，max 值不变。

（6）i 增加 1。

（7）返回到第（3）步。

（8）输出 max 中的数，此时 max 中的数就是 10 个数中最大的数。

从上述算法示例可以看出，算法是解决问题的方法和步骤的精确描述。算法并不给出问题的精确解，只是说明怎样才能得到解。每一个算法都是由一系列基本的操作组成的。这些操作包括加、减、乘、除、判断、置数等。所以研究算法的目的就是要研究怎样把问题的求解过程分解成一些基本的操作。

算法设计好之后，要检查其正确性和完整性，再根据它用某种高级语言编写出相应的程序。程序设计的关键就在于设计出一个好的算法。所以，算法是程序设计的核心。

2. 算法的特性

从上面的例子中，可以概括出算法的 5 个特性：

（1）有穷性。算法中执行的步骤总是有限次数的，不能无止境地执行下去。例如，计算圆周率 π 的值，可用如下公式：

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$$

这个多项式的项数是无穷的，因此，它是一个计算方法，而不是算法。要计算 π 的值，只能取有限项。例如，计算结果精确到第 5 位，那么，这个计算就是有限次的，因而才能称得上算法。

（2）确定性。算法中的每一步操作必须具有确切的含义，不能有二义性。

（3）有效性。算法中的每一步操作必须是可执行的。

（4）要有数据输入。算法中操作的对象是数据，因此应提供有关数据。但如果算法本身给出了运算对象的初值，也可以没有数据输入。

（5）要有结果输出。算法的目的是用来解决一个给定的问题，因此应提供输出结果，否则算法就没有实际意义。

3．算法评价标准

在算法设计中，只强调算法特性是不够的。一个算法除了满足 5 个特性之外，还有一个质量问题。一个问题可能有若干个不同的求解算法，一个算法又可能有若干个不同的程序实现。在不同算法中有好算法，也有差算法。设计高质量算法是设计高质量程序的基本前提。如何评价算法的质量呢？不同时期、不同环境其评价标准可能不同，但一些基本评价标准是相同的。目前，评价算法质量有 4 个基本标准：

（1）正确性。一个好算法必须保证运行结果正确。算法正确性，不能主观臆断，必须经过严格验证，一般不能说绝对正确，只能说正确性高低。目前程序正确性很难给出严格的数学证明，程序正确性证明尚处于研究阶段。要多选用现有的、经过时间考验的算法，或采用科学规范的算法设计方法，是保证算法正确性的有效途径。

（2）可读性。一个好算法应有良好的可读性，好的可读性有助于保证正确性。科学、规范的程序设计方法（如结构化方法和面向对象方法）可提高算法的可读性。

（3）通用性。一个好算法要尽可能通用，可适用一类问题的求解。例如，设计求解一元二次方程 $2x^2+3x+1=0$ 的算法，该算法应设计成求解一元二次方程 $ax^2+bx+c=0$ 的算法。

（4）高效率。效率包括时间和空间两个方面。一个好的算法应执行速度快、运行时间短、占用内存少。效率和可读性往往是矛盾的，可读性要优先于效率。目前，在计算机速度比较快，内存容量比较大的情况下，高效率已处于次要地位。

4．算法效率的度量

算法效率的度量分为时间度量和空间度量。

（1）时间度量

算法的执行时间需要依据该算法编制的程序在计算机上运行时所消耗的时间来度量。它大致等于计算机执行一种简单操作（如赋值、比较等）所需的平均时间与算法中进行简单操作的次数的乘积。因为执行一种简单操作所需的平均时间随计算机而异，它是由所使用计算机的软硬件环境决定的，与算法无关，所以只需讨论影响算法执行时间的另一因素，即算法中进行简单操作的次数。通常把算法中进行简单操作的次数的多少称为算法的时间复杂度，它是一个算法执行时间的相对度量。

一般用问题的规模来表示算法所处理数据的多少。若解决问题的规模为 n，那么算法的时间复杂度就是问题规模 n 的一个函数 $f(n)$，假定时间复杂度记作 $T(n)$，则：

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

例如，求 $s=1+2+3+\cdots+n$，这里问题的规模为 n，求 s 的值需要进行 $n-1$ 次加法，所以算法的时间复杂度 $T(n)=n-1$。

这里算法比较简单，时间复杂度容易计算，当算法比较复杂时，时间复杂度的计算就相对困难。实际上，一般也没必要计算出算法的精确复杂度，只要大致计算出相应的数量级即可，即随着问题规模 n 的增大，算法的执行时间的增长率如何，这个时间增长率称为阶。显然求 s 值算法的执行时间与 n 成正比，所以该算法的时间复杂度是 n 阶的，记为 $T(n)=O(n)$。

算法的复杂度采用数量级表示后，将给计算复杂度带来很大的方便，这时只需分析影响一个算法的主要部分即可，不必对每一步进行分析。同时对主要部分的分析也可简化，只需分析循环内简单操作的次数即可。例如，下面 C 语句的时间复杂度为 $O(n^2)$，即是 n 的平方阶的。

```
for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
      a[i][j]=i+j;
```

按数量级递增顺序，常见的几种时间复杂度有 $O(1)$、$O(\log_2 n)$、$O(n)$、$O(n\log_2 n)$、$O(n^2)$、$O(n^3)$、$O(2^n)$等。$T(n)=O(1)$表示算法执行时间与问题规模无关。图 1-1 给出了各种具有代表性的时间复杂度 $T(n)$与问题规模 n 的变化关系。从图中可以看到，当 $T(n)$为对数阶、线性阶、幂阶函数或它们的乘积时，算法的时间复杂度随问题规模的变化是可以接受的，当 $T(n)$为指数阶或它们的乘积时，算法的时间复杂度随问题规模的增大大幅增加，是不可以接受的，这种算法称为无效算法。

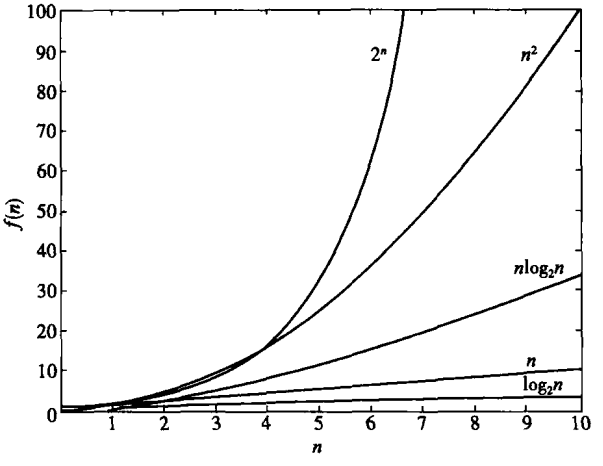


图 1-1 时间复杂度函数曲线比较

一个算法的时间复杂度除了与问题的规模有关外，还与输入的数据的次序有关，输入的次序不同，算法的复杂度也不同，所以当分析算法的复杂度时，还要考虑到最好、最坏和平均复杂度。

（2）空间度量

一个算法的实现所占用的存储空间，大致包括 3 个方面：一是存储算法本身所占用的存储空间；二是算法中的输入输出数据所占用的存储空间；三是算法在运行过程中临时占用的存储空间。存储算法本身所占用的存储空间与算法书写的长度有关，算法越长，占用的存储空间越多。算法中输入输出数据所占用的存储空间是由要解决的问题所决定的，它不随算法的改变而改变。算法在运行过程中临时占用的存储空间随算法的不同而改变，有的算法只需要占用少量的临时工作单元，与待解决问题的规模无关，有的算法需要占用的临时工作单元，与待解决问题的规模有关，即随问题的规

模的增大而增大。因此，通常把算法在执行过程中临时占用的存储空间定义为算法的空间复杂度。

算法的空间复杂度比较容易计算，它包括局部变量所占用的存储空间和系统为实现递归（如果采用递归算法）所占用的堆栈这两个部分。算法的空间复杂度也用数量级的形式给出。

5. 算法的描述

描述算法有很多不同的工具，前面两个例子的算法是用自然语言——汉语描述的，其优点是通俗易懂，但它不太直观，描述不够简洁，且容易产生二义性。在实际应用中，常用流程图、结构化流程图和伪代码等描述工具来描述算法。

（1）用流程图描述算法

流程图也称为框图，它是用一些几何框图、流程线和文字说明表示各种类型的操作。一般用矩形框表示进行某种处理，有一个入口，一个出口，在框内写上简明的文字或符号表示具体的操作。用菱形框表示判断，有一个入口，两个出口。菱形框中包含一个为真或为假的表达式，它表示一个条件，两个出口表示程序执行时的两个流向，一个是表达式为真（即条件满足）时程序的流向，另一个是表达式为假（即条件不满足）时程序的流向，条件满足时用 Y（即 Yes）表示，条件不满足时用 N（即 No）表示。流程图中用带箭头的流程线表示操作的先后顺序。

流程图是人们交流算法设计的一种工具，不是输入给计算机的。只要逻辑正确，且能被人们看懂就可以了，一般是由上而下按执行顺序画下来。

【例 1.3】用流程图来描述例 1.1 和例 1.2 的算法。

流程图分别如图 1-2 和图 1-3 所示。

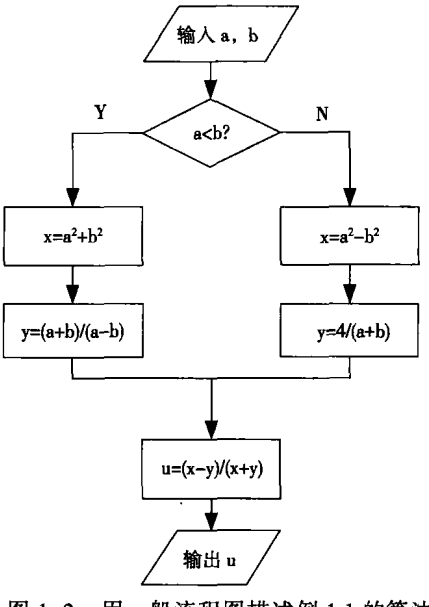


图 1-2　用一般流程图描述例 1.1 的算法

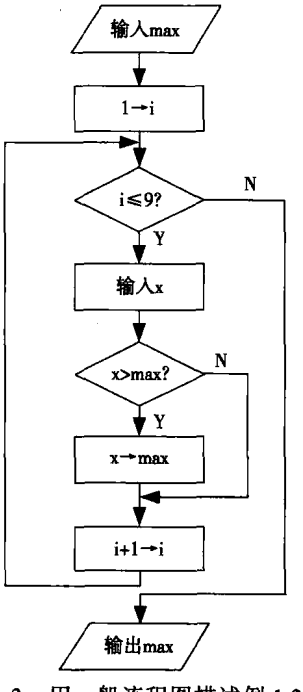


图 1-3　用一般流程图描述例 1.2 的算法

流程图的主要优点是直观性强，初学者容易掌握。缺点是对流程线的使用没有严格限制，如毫无限制地使流程任意转来转去，将使流程图变得毫无规律，难以阅读。为了提高算法的可读性