C语言程序设计理论与实践

霍利岭　周云成　傅伟玉◎著

新华出版社

C语言程序设计理论与实践

霍利岭　周云成　傅伟玉◎著

新 华 出 版 社

图书在版编目（CIP）数据

C 语言程序设计理论与实践/霍利岭，周云成，傅伟玉著．—北京：新华出版社，2016.3
ISBN 978-7-5166-2415-9
Ⅰ．①C… Ⅱ．①霍… ②周… ③傅… Ⅲ．①C 语言－程序设计 Ⅳ．①TP312

中国版本图书馆 CIP 数据核字（2016）第 060068 号

C 语言程序设计理论与实践

作　　者：霍利岭　周云成　傅伟玉

出 版 人：张百新
责任编辑：郑建玲
封面设计：周香菊

出版发行：新华出版社
地　　址：北京石景山区京原路 8 号　　**邮　　编**：100040
网　　址：http：//www.xinhuapub.com　　http：//press.xinhuanet.com
经　　销：新华书店
购书热线：010-63077122　　**中国新闻书店购书热线**：010-63072012

照　　排：北京海胜印刷排版工作室
印　　刷：北京厚诚则铭印刷科技有限公司

成品尺寸：185mm×260mm
印　　张：18.375　　**字　　数**：380 千字
版　　次：2016 年 3 月第一版　　**印　　次**：2016 年 3 月第一次印刷

书　　号：ISBN 978-7-5166-2415-9
定　　价：38.00 元

图书如有印装问题请与印刷厂联系调换：010-60277330

前　言

C 语言是结构化的程序设计语言，它概念简洁、语句紧凑，具有很强的表达能力，既可用来编写系统软件，也可以用来编写应用软件，是目前最为流行的程序设计语言之一。学习 C 语言是学习其他高级编程语言的基础，因此，“C 语言程序设计”也是许多高校计算机及相关专业的必修课程。

本书较全面地介绍了 C 语言程序设计的基础理论知识和基本编程技能，内容新颖，体系合理，内容翔实，通俗易懂，配备了大量经过筛选的例题，能够使初学者快速掌握 C 语言的基础理论，并将其运用到实践之中。

本书共分 13 章，第 1 章为程序设计基础，概述了程序设计语言与算法的基本知识；第 2 章为 Visual C++ 6.0，对目前较主流的 C 语言编程环境 Visual C++ 6.0 作了简单介绍；第 3 章为数据类型、运算符与表达式，介绍了数据的类型、运算与输入输出；第 4 章至第 6 章分别介绍了顺序结构程序设计、选择结构程序设计与循环结构程序设计的语句及应用；第 7 章介绍函数，意在使读者较全面的了解 C 语言程序的组织结构；第 8 章介绍数组；第 9 章介绍指针，重点说明指针及指针数据的使用方法；第 10 章介绍文件的打开、读写与关闭；第 11 章介绍预处理命令；第 12 章为其他数据类型，列举了枚举、结构体类型、共用体、链表等几种数据类型的常见形式。第 13 章为项目实战，通过两个具体实例说明项目设计的几个重要阶段。书中的每一章都配备了大量经过精心筛选的例题，既能帮助理解知识，又极具启发性。

本书的撰写是笔者多年教学经验与研究的总结，在撰写过程中受到了来自相关领导和同事们的支持与帮助，也参考了大量专家学者和业界同仁的文献资料，在此一并致以深深的谢意。

全书由河北师范大学汇华学院霍利岭老师，沈阳农业大学周云成老师，江苏财经职业技术学院傅伟玉老师合计完成，共计 38 万字。其中霍利岭老师负责全书统稿以及第 1、2、3、4、5、10 章的编写工作，合计 15 万字；周云成老师负责第 6、7、8、9 章的编写工作，合计 12 万字；傅伟玉老师负责第 11、12、13 章的编写工作，合计 10 万字。

由于本书篇幅较长，加之笔者自身水平有限，难免存在错讹脱漏之处，敬请读者批评指正。

目　录

第1章　程序设计基础

1.1　计算机程序与程序设计语言

一个完整的计算机系统由硬件系统和软件系统组成。主机、显示器、键盘和鼠标等都是组成计算机系统的硬件，但这些只是计算机系统的物理基础，计算机的运行还要有众多软件支持才能让计算机做人们想做的事情。软件系统由计算机运行时所需的各种程序、数据和文档组成，软件来源于程序设计与开发，而程序设计与开发的平台是各种计算机程序设计语言。

1.1.1　计算机系统结构

“计算机之父”冯·诺依曼提出的计算机系统结构如下：

（1）计算机由控制器、运算器、存储器、输入设备和输出设备5个部分构成。

（2）计算机采用二进制，指令和数据均以二进制数形式表示和存放。

（3）计算机按照程序规定的顺序将指令从存储器中取出，并逐条执行。

控制器集中控制其他设备，信息分为数据信息和控制信息两种，在控制指令的控制下，数据按照由输入设备输入数据，存储在存储器中，控制器和运算器直接从存储器中取出数据（包括程序代码和运算对象）进行处理，结果存储在存储器内，并由输出设备输出的方式进行“流动”。

1.1.2　程序与程序设计

人们操作计算机完成各项工作，实际上是执行计算机中的各种程序，如操作系统、文字处理程序、手机内置的各类应用程序等。

程序是用来完成特定功能的一系列指令。通过向计算机发布指令，程序设计人员可以控制其执行某个操作或进行某种运算。一组指令构成一个程序，可以用来解决一个具体问题。简单的程序可能仅仅向屏幕输出一段符号，而复杂的程序可以完成更多功能。

日常生活中，“程序”是事情进行的先后次序，例如“工作程序”“会议程序”等，完成不同事情所采用的程序也不同。计算机程序是为完成特定功能按照一定的顺序而设计的。而编写程序就称为程序设计，俗称“编程”。

程序设计的本质就是利用计算机的控制器、运算器、存储器、输入设备和输出设备这

5 个部分完成特定任务的指令序列。在编写程序解决实际问题时，一般按照如下六个步骤进行：

1. 分析和定义实际问题

通过对实际问题的深入分析，准确地提炼、描述要解决的问题，明确要求。

2. 建立处理模型

实际问题都是有一定规律的数学、物理等过程，用特定方法描述问题的规律和其中的数值关系，是为确定计算机实现算法而做的理论准备。如求解图形面积一类的问题，可以归结为数值积分，积分公式就是为解决这类问题而建立的数学模型。

3. 设计算法

将要处理的问题分解成计算机能够执行的若干特定操作，也就是确定解决问题的算法。例如，由于计算机不能识别积分公式，需要将公式转换为计算机能够接受的运算，如选择梯形公式或辛普森（Simpson）公式等。

4. 设计流程图

在编写程序前给出处理步骤的流程图，能直观地反映出所处理问题中较复杂的关系，从而使编程时思路清晰，避免出错。流程图是程序设计的良好辅助工具，它作为程序设计资料也便于交流。

5. 编写程序

编程是指用某种高级语言按照流程图描述的步骤写出程序，也叫作编码。使用某种语言编写的程序叫源程序。

6. 调试程序和运行程序

将写好的程序上机检查、编译、调试和运行，并纠正程序中的错误。

1.1.3 程序设计语言

程序设计语言是计算机能够理解和识别的一种语言体系，它按照特定的规则组织计算机指令，使计算机能够自动进行各种操作处理。不同的程序设计语言具有不同的使用规则，因而编写的计算机程序也不同。

自计算机诞生以来，产生了上千种程序设计语言，有些已被淘汰，有些得到了推广和发展。程序设计语言经历了由低级到高级的发展过程，可以分为机器语言、汇编语言、高级语言和面向对象的语言。低级语言包括机器语言和汇编语言，高级语言有很多种，如c、Basic、Fortran 等，面向对象的语言如C ++ 、Visual Basic、Java 等。越低级的语言越接近计算机的二进制指令，越高级的语言越接近人类的思维方式。

1. 机器语言

机器语言是计算机能够直接识别并执行的二进制指令。机器语言指令由计算机的指令系统提供，阅读与编写比较费事，浪费时间，也容易出错。不同计算机的指令系统也不同，因此使用机器指令编写的程序通用性较差。

例如，某计算机中的指令 1011011000000000 的作用是让计算机进行一次加法运算，指令 1011010100000000 是让计算机进行一次减法运算等。要处理一个问题，需要编写很多条类似的指令所组成的程序。这种程序称为机器语言程序，它能被计算机直接执行，而且速度快。

2. 汇编语言

汇编语言用助记符来代替机器语言的指令码，使机器语言符号化。如加法运算表示为“ADDAX，DX”。汇编语言比机器语言更进一步，但是编程人员仍然需要对机器硬件有深入了解，没有摆脱对具体机器的依赖，编程仍然具有较大难度：

3. 高级语言

为了解决计算机硬件的高速度和程序编制的低效率之间的矛盾，20 世纪 50 年代末期产生了“程序设计语言”，也称为高级语言。高级语言比较接近自然语言，直观、精确、通用、易学、易懂，编程效率高，便于移植。例如，语句“c = a + b”表示“求 a + b 的值并赋给 c”。高级语言有上千种，但实际应用的仅有十几种，如 BASIC、PASCAL、C、FORTRAN、ADA、COBOL、PL/I 等。

4. 面向对象的程序设计语言

面向对象的程序设计语言更接近人们的思维习惯。它将事物或某个操作抽象成类，将事物的属性抽象为类的属性，事物所能执行的操作抽象为方法。常用的面向对象语言有 Visual C++、Visual Basic、Java 等。

因为计算机不能直接识别高级语言，因此用高级语言编写的程序必须被翻译成计算机能识别的目标程序。程序执行有编译执行和解释执行两种方式。

（1）编译执行是将源程序翻译生成一个可执行的目标程序，该目标程序可以脱离编译环境和源程序独立存在和执行。

（2）解释执行是将源程序逐句解释成二进制指令，解释一句执行一句，不生成可执行文件，它的执行速度比编译方式慢。

1.2　算法

瑞士科学家、Pascal 语言发明者 Niklaus Wirth 对计算机程序给出了一个著名的定义，即：程序 = 数据结构 + 算法。该定义归结了计算机程序的两个核心问题，强调了算法在程序中的重要性。

1.2.1　算法的概念

算法是逐步求解问题的方法，是在有限步骤内求解某一问题所使用的一组定义明确的规则，是计算机处理问题所需要的具体步骤。算法的最终实现是计算机程序，程序设计人员只有将算法转变为计算机程序，才能利用计算机解决问题。

算法的建立通常会经过由粗略到细化的过程，先找出解决问题的基本思路，把解决问题的基本过程表达出来，确立粗略的算法框架，然后对框架中的内容进行逐步细化，添加必要的细节，形成解决问题的有效算法。

1.2.2　算法的特性

算法具有以下特性：

1. 有穷性

算法经过有限次的运算就能得到结果，而不能无限执行或超出实际可以接受的时间。如果一个程序需要执行 1000 年才能得到结果，对于程序执行者而言，基本就没有什么意义了。

2. 确定性

算法中的每一个步骤都是确定的，不能含糊、模棱两可。算法中的每一个步骤不应当被解释为多种含义，而应当十分明确。比如描述“小王递给小李一件他的衣服”，这里，衣服究竟是小王的，还是小李的呢？

3. 有效性

一个算法中不能出现无效的步骤，每一个步骤必须能够有效地执行，并且得到确定的结果。例如，如果一个算法中含有 x 作除数的步骤，当 x 为 0 时执行该步骤就是一个无效的操作，若不能排除 0 作除数的情况，这个算法就是无效的。

4. 输入和输出特性

每个有意义的算法有 0 个或多个输入，并且提供一个或多个输出。所谓 0 个输入是指算法本身定义了初始条件，能够提供处理数据；而多个输入是指算法能够从外部获得处理数据。输出是指算法执行后能够产生输出信息，以反映数据处理结果，没有输出的算法是毫无意义的。

另外，在进行算法设计时，还需要考虑算法的多样性和通用性特点。

算法的多样性是算法的一种自然属性。由于一个具体问题可以有不同的解决方法，自然就可以设计解决问题的不同算法。因此，即便使用同一种计算机语言，解决同一个问题时也可能有多个不同的计算机程序，认识这一点对学习程序设计是非常重要的。

算法的通用性是对算法的一种技术要求。一个算法应是适用于某类问题，而不只是适用于某一个问题。例如，求解一个一元二次方程时，应设计一个适用于所有一元二次方程的算法，而不是设计只能求解某个方程的算法。具有广泛适应性的算法自然具有更好的推广应用价值。

1.2.3 算法的表示

算法的表示方法有很多种，常用的有自然语言、伪代码、传统流程图、N－S 流程图、PAD 图等。

1. 自然语言

使用自然语言表示，就是采用人们日常生活中的语言。如求两个数的最大值，可以表示为：如果 A 大于 B，那么最大值为 A，否则最大值为 B。但在描述“陶陶告诉贝贝她的小猫丢了”时，表示的是陶陶的小猫丢了还是贝贝的小猫丢了呢？就出现了歧义。可见使用自然语言表示算法时拖沓冗长，容易出现歧义，因此不常使用。

2. 伪代码

伪代码用介于自然语言和计算机语言之间的文字和符号来描述算法。例如，求两个数的最大值可以表示为：

if　A 大于 B，then 最大值为 A，else 最大值为 B。

伪代码的描述方法比较灵活，修改方便，易于转变为程序，但是当情况比较复杂时，

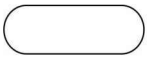
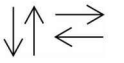
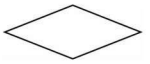
不够直观，而且容易出现逻辑错误。软件专业人员一般习惯使用伪代码，而初学者最好使用流程图。

3. 传统流程图

流程图表示算法比较直观，它使用一些图框来表示各种操作，用箭头表示语句的执行顺序。

传统流程图的常用符号如表 1－1 所示。

表 1－1　常用的流程图符号及其功能

流程图符号	符号功能	流程图符号	符号功能
	开始、结束		输入、输出
	处理		流程方向
	判断		

例 1.1　画出计算 1 +2 +3 + … +100 之和的流程图。

解：流程图如图 1－1 所示：

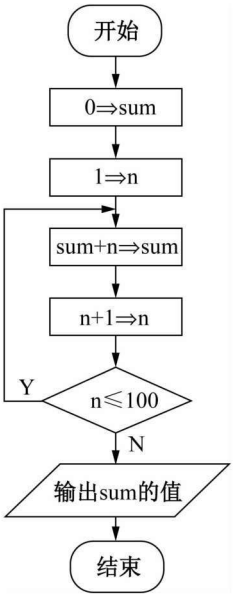


图 1－1　例 1.1 的流程图

例 1.2　画出判断给定年份是否是闰年的流程图。

解：流程图如图 1－2 所示。

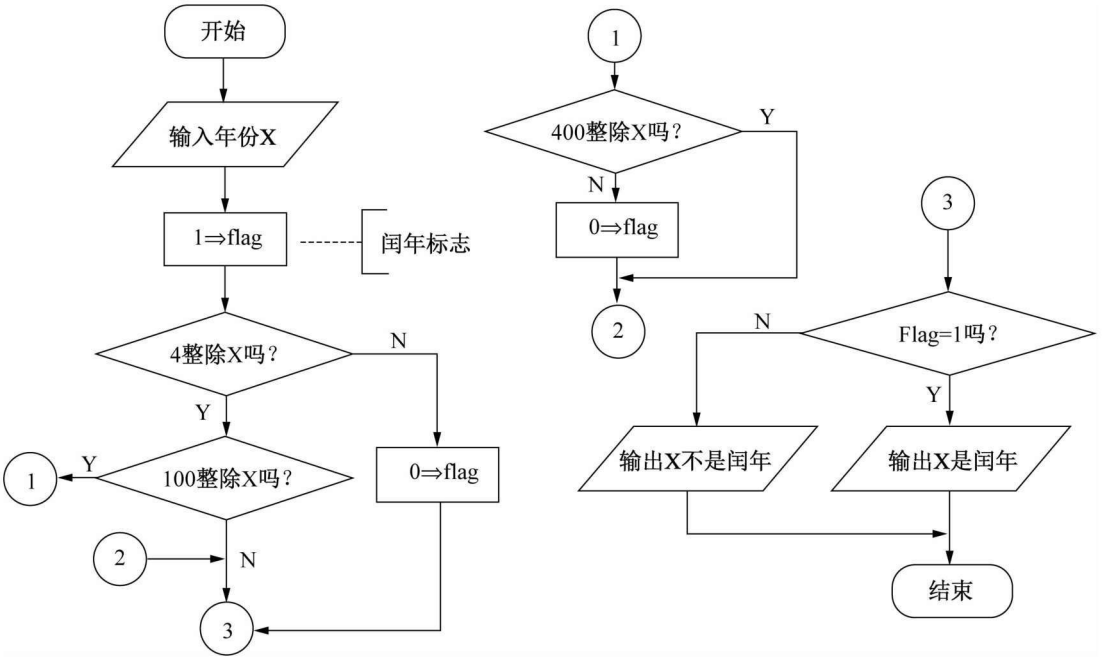


图 1－2　例 1.2 的流程图

3. N－S 流程图

N－S 流程图又称盒图，其特点是所有的程序结构均用方框表示。N－S 流程图的绘制比较节省空间，它避免了使用箭头任意跳转程序所造成的混乱，更加符合结构化程序设计的原则。它按照从上往下的顺序执行语句。

N－S 流程图用以下的流程图符号。

①顺序结构。如图 1－3 所示，A 和 B 两个框组成一个顺序结构。

②选择结构。如图 1－4 所示，当条件 P 成立时执行 A，当 P 不成立时执行 B。

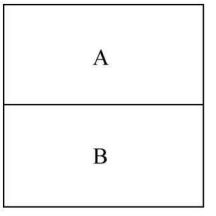


图 1－3　顺序结构

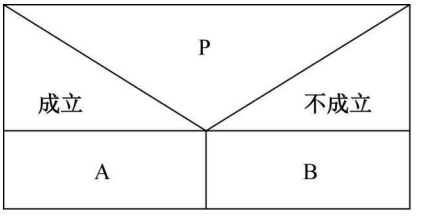


图 1－4　选择结构

③循环结构。当型循环结构如图 1－5 所示，当条件 P 成立时反复执行 A，直到 P 不成立时结束循环；直到型循环结构如图 1－6 所示，反复执行 A 直到条件 P 成立时结束循环。

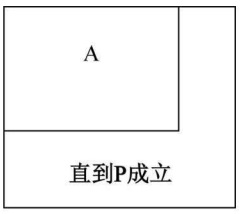


图 1 – 5　当型循环结构

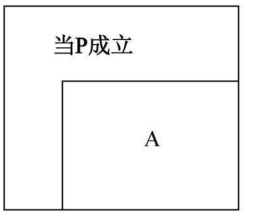


图 1 – 6　直到型循环结构

例 1.1 和例 1.2 的算法用 N – S 流程图表示分别如图 1 – 7、图 1 – 8 所示。

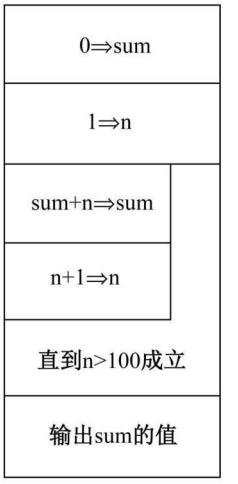


图 1 – 7　例 1.1 的 N – S 图

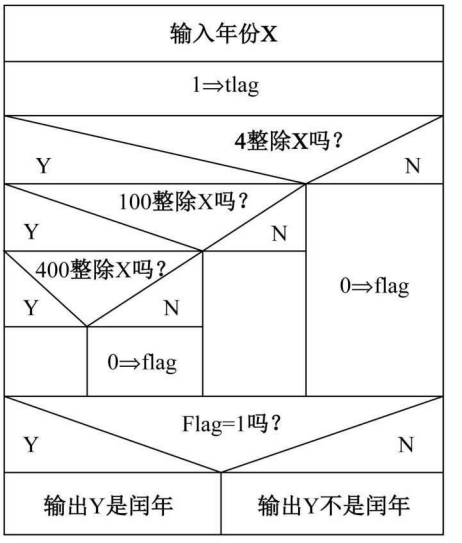


图 1 – 8　例 1.2 的 N – S 图

1.2.4 算法的评价

由于算法的多样性特点，自然就产生了算法优劣的评价标准。评价标准涉及很多方面，如算法实现后的执行速度（严格而言，称为算法的时间复杂性，表示问题规模与算法执行时间的关系）、算法对系统资源的需求程度（通常称为算法的空间复杂性）、算法本身的复杂程度（算法的可读性）、算法的通用性等，但不管采用什么标准进行评价，正确和清晰易读永远是一个好算法的基本条件。

1.3 C 语言程序设计

1.3.1 C 语言的发展历史

C 语言是国际上广泛流行的计算机高级语言。它既可用来写系统软件，也可用来写应用软件。

C 语言的祖先是 BCPL 语言：1967 年英国剑桥大学的 Mattin Richards 推出了 BCPL（Basic Combined Programming Language）语言。BCPL 语言是一种无类型的系统程序语言，它的基本数据类型是机器字，使用较多的指针和地址运算，这与其他常见的各种高级语言有着明显的差异。1970 年，美国贝尔（Bell）实验室的肯·汤姆逊（Ken Thompson，UNIX 操作系统的主要研制者）以 BCPL 语言为基础，又发表了一种新的语言——B 语言，并用汇编语言和 B 语言写成了 UNIX 的初版。B 语言接近机器硬件，但较为简单，而且无常用的数据类型。

为了克服 B 语言的局限性，美国贝尔实验室的 D. M. Ritchie 设计出了 C 语言。C 语言既保持了 BCPL 和 B 语言精练且接近硬件的优点，又克服了它们过于简单，无数据类型等的缺点，C 语言的新特点主要表现在具有多种数据类型。开发 C 语言的目的在于尽可能降低用它开发的软件对硬件平台的依赖程度，使之具有可移植性。

随着 C 语言的发展，1977 年出现了不依赖于具体机器的 C 语言编译文本，而 C 语言的真正定义是在 1978 年，由布莱思·科尼汉（Brian Kemighan）和丹尼斯·里奇（Dennis Ritchie）所著的《The Programming Language》中被阐述清楚。这是一本影响深远的名著，实际上成了后来的 C 语言标准。

1983 年，美国国家标准化协会（ANSI）整理和扩充了当时的各种 C 语言版本，并制定了一个 C 语言标准（即 83 ANSI C）。随后，在 1987 年对其做了重新修订，制订了 87 ANSI 标准 C。1989 年，ANSI 又公布了一个新的 C 语言标准 ANSI X3. 159—1989，简称为 C89。国际标准化组织 ISO 在 1990 年接受了该标准，使其正式成为国际标准 ISO/IEC 9899：1990，简称为 C90。随后，ISO 在 1995 年和 1999 年分别又对该标准进行过 2 次修订，为 C 语言增加了面向对象的特征，并命名为 ISO/IEC 9899：1999，简称为 C99。

近年来的 C 语言发展十分迅速，以面向对象的技术和 C 语言语法相结合的C++ 语言在软件设计领域占有举足轻重的地位，其应用的广泛性也已逐渐超过了传统的 C 语言，

但它更适合开发大型的应用系统。在一些涉及硬件开发和嵌入式的应用中，C 语言仍处于重要地位并被广泛使用，且传统的以汇编为开发语言的应用也基本由 C 语言取代。同时，作为了解程序设计技术的一种基础语言，C 语言也具有独特的优势，因为大量的新兴语言如 Java、C#等都由 C 语言发展而来，与 C 语言有着大量一致的语法现象。因此，不管这些较新的语言如何流行，C 在软件开发产业中仍然是一种重要的编程语言，已被广泛用于嵌入式系统编程，在汽车、照相机、DVD 播放器等现代化设备的微处理编程中起着举足轻重的作用。此外，C 语言还适用于操作系统的开发，伴随 UNIX 和 Linux 的成长，它将扮演更重要的角色。因此，在未来很长的一段时间内，C 语言仍将保持强劲的势头。

1.3.2　C 语言的主要特点

C 语言能够生存和发展，并具有较强的生命力，主要是 C 语言具有以下特点。

1. C 语言是比较“低级”的语言

C 语言允许直接访问物理内存，能够进行位（bit）操作，这使 C 语言在运行系统程序时，显得非常有效，而原来通常用汇编语言来编写，现在用 C 语言代替汇编语言，使程序员可以减轻负担、提高效率，且程序具有更好的移植性。

2. 语言简洁、书写自由

C 语言是一种很小的语言，具有精心选择的控制结构和数据类型，摒弃了一切不必要的成分，从而使其规模缩减到最小，代码简洁而高效。C 程序的书写也几乎不受什么限制，可以在一行上书写多个语句，也可以把一个语句写在多个行上。当然，这种灵活性可能使程序缺少一种可以遵循的标准，因此，学习时应尽量注意程序书写的“规范性”，这一点可以根据本书所提供的示例程序去体会。

3. 运算符丰富

运算符的多少体现了语言对数据的加工能力的强弱。C 语言提供了极为丰富的运算符，共 40 多种，这使 C 语言可以直接实现其他语言中很多难以实现的运算，同时也提高了语句的能力。

4. 语法限制不太严格，程序自由度大

这是 C 语言较有争议的一个特点，这是因为，语法限制不严给程序设计带来了灵活性，但也给程序设计造成了一定的困难。例如，最为典型的数组超界问题就是语法限制不严的产物，不论程序员使用了数组的多少元素，是否与定义吻合，C 语言的编译器都不会给出错误通知，因为 C 编译器不做数组边界的检查。如果在设计时没确保引用的正确性，超界的数组元素有可能导致灾难性的后果。

5. C 语言是结构化设计语言

C 语言在结构上类似于 ALGOL 60、Pascal 等结构化语言，C 语言的主要成分就是函数，可以对一个程序中的各任务分别定义和编码，使程序模块化，而在函数的外部只需要了解函数的功能，而将实现的细节隐藏起来，对于设计得好的函数能够正确地工作而对程序的其他部分不产生副作用。另外 C 语言还提供了多种结构化的控制语句，如循环 for、

do…while、while 语句，用于分支控制的 if、if…else、switch 语句等，可以满足结构化程序设计的要求。

6. C 语言是程序员的语言

很多程序语言是专为某一领域设计的，如 FORTRAN 是为工程师设计的，COBOL 是为商业人员设计的，Pascal 是为教学设计的，BASIC 是为非程序员设计的。但 C 语言是为专业程序员设计的，最初是为编写 UNIX 操作系统而设计的，这是因为 C 语言具有很少的限制、很小的要求、程序设计自由度大、方便地控制结构、独立的函数、紧凑关键字集合和较高的程序执行效率，用 C 语言编写的程序可以获得高效的机器代码，只比汇编语言编写的程序效率低 10% ~20%，但其结构又具有 Pascal 的特点。

C 语言能得到广泛的使用主要是因为程序员喜欢它，但现在它不再是程序员的专利了，非专业人员经过学习和实践也能熟练掌握，现在很多不同专业的非计算机专业人员都在使用 C 语言。

1.3.3 C 语言的结构

C 程序结构由头文件、主函数、系统的库函数和自定义函数组成，因程序功能要求不同，C 程序的组成也有所不同。其中 main 主函数是每个 C 语言程序都必须包含的部分。

我们先从两个简单的例子来认识 C 语言程序设计的结构。

例 1.3　输出一行信息，内容为：This is my first C program.

程序如下：

```
/*exal  - 3. c  */
#include  <stdio. h >
void main( )
{
     printf( " This isrny first C program.  \\ n" ) ;
}
```

经过编译、连接后，程序正确运行结果为：

```
This is my first C program.
```

说明：

① C 语言程序的基本形式如下：

程序首部

```
Main( )
    {
      句序列
    }
```

— 函体