高等学校电子信息类“十二五”规划教材

C语言程序设计

主编 邵军 王忠

西安电子科技大学出版社
http://www.xduph.com

高等学校电子信息类“十二五”规划教材

C语言程序设计

主　编　邵　军　王　忠

参　编　焦　康　陈　菁

吴　旻　蔡艳宁

西安电子科技大学出版社

内 容 简 介

本书是根据教育部《关于进一步加强高等学校计算机基础教学的意见》的基本要求编写的。

本教材从C语言程序设计的基本原理及程序设计的基本思想出发，以“实践应用”为目标，紧扣基础，循序渐进，面向应用。

全书主要包括两部分，即基础部分和面向应用部分。基础部分的主要内容包括程序设计中的基本概念与应用，如变量、表达式、输入/输出函数、流程控制结构等。面向应用部分包括函数、数组、指针的概念及其应用，结构型数据的应用及文件的操作等知识。

本书可作为高等院校相关专业的教材及国家计算机水平考试、各类成人教育的培训教材，也可供计算机爱好者自学。

图书在版编目(CIP)数据

C语言程序设计/邵军，王忠主编. —西安：西安电子科技大学出版社，2013.4
高等学校电子信息类“十二五”规划教材
ISBN 978-7-5606-3050-2

Ⅰ. ① C… Ⅱ. ① 邵… ② 王… Ⅲ. ① C语言—程序设计—高等学校—教材 Ⅳ. ① TP312

中国版本图书馆CIP数据核字(2013)第066000号

策　　划　戚文艳
责任编辑　阎　彬　戚文艳
出版发行　西安电子科技大学出版社(西安市太白南路2号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　西安文化彩印厂
版　　次　2013年4月第1版　　2013年4月第1次印刷
开　　本　787毫米×1092毫米　1/16　　印　张　17.5
字　　数　414千字
印　　数　1～3000册
定　　价　30.00元

ISBN 978-7-5606-3050-2/TP

XDUP 3342001-1

前　言

本书是根据教育部《关于进一步加强高等学校计算机基础教学的意见》的基本要求，为普通高等学校非计算机专业学生编写的教材。

程序设计是高等学校重要的计算机基础课程，它以编程语言为平台，介绍程序设计的思想和方法。通过该课程的学习，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思想和方法，提高问题求解和程序设计语言的应用能力。

程序设计是每个科技工作者使用计算机的基本功。C 语言是目前使用最广泛的一种程序设计语言。它既具备高级语言的特性，又具有直接操纵计算机硬件的能力，并因其丰富灵活的控制和数据结构、简洁而高效的语句表达、清晰的程序结构和良好的可移植性而拥有大量的使用者，也成为高校计算机语言类课程的首选。

本书旨在讲授程序设计基础和 C 语言基础，突出 C 语言课程本身实践性强的特点，以解决实践中的问题为目标，通过应用案例讲解程序设计的基本思想和方法以及相关的语言知识。本书以启发式教学和研究性学习为核心，激发学习者的兴趣和潜能，注重学习者思考能力和创新能力的培养，即从重视知识目标转向重视智能目标。本书在内容组织上循序渐进，在结构上做了精心安排。

全书共分 11 章，主要包括两部分，即基础部分和面向应用部分。基础部分的主要内容包括程序设计中的基本概念与应用，如变量、表达式、输入/输出函数、流程控制结构等。面向应用部分包括函数、数组、指针的概念及其应用，结构型数据的应用及文件的操作等等知识，然后在上述知识点的基础上初步涉及数据结构的一些简单专题，如排序和链表等。

C 语言程序设计是一门实践性很强的课程，学习者必须通过大量的编程训练，在实践中掌握语言知识，培养程序设计的基本能力，并逐步理解和掌握程序设计的思想和方法。因此，为了配合该课程的实验教学，本书配有相应的实验指导和学习辅导配套教材。

本书第 1、2 章由邵军、王忠编写，第 3、4、5 章由焦康编写，第 6、7 章由陈菁编写，第 8、9 章由吴旻编写，第 10、11 章由蔡艳宁编写。全书由邵军和王忠主编并统稿。

由于作者水平有限，书中难免存在疏漏之处，敬请读者指正。

编　者

2013 年 3 月

目　录

第 1 章　概　　述

1.1　程序设计概述

作为 20 世纪最重要的发明之一，计算机自问世以来，几乎在人类活动的各个领域都得到了应用，它深刻而持久地改变着我们的社会和生活。

计算机系统由计算机硬件系统和计算机软件系统组成。计算机硬件系统提供了一个具有广泛通用性的计算平台，没有软件支持的硬件系统只能是一堆“废铁”，俗称“裸机”。计算机系统功能的多样化和复杂化，主要取决于计算机软件系统的功能。软件功能的发展主要取决于人们的需求以及软件工作者的创造力和程序设计能力。

1.1.1　程序、程序设计与程序设计语言

1. 程序

计算机的每一个操作都是根据人们事先指定的指令进行的。每一条指令执行特定的操作。所谓程序，就是一组计算机能够识别和执行的指令，用以完成一定的功能。计算机执行程序的过程就是“自动地”执行各条指令的过程。计算机软件系统包括系统软件和应用软件两部分。系统软件一般由计算机生产厂家提供，是为方便用户使用、管理和维修计算机而编制的程序的总称。应用软件一般是指用户在各自的应用领域中，为解决各类实际问题而编制的程序。

2. 程序设计

计算机的一切操作都是由程序来控制的。计算机本质上是执行程序的机器。只有懂得程序设计，才能真正使用好计算机这一工具。

计算机作为一个工具，主要用来解决人类所面临的各种问题。只有最终在计算机上能够运行良好的程序才能为人们解决特定的实际问题。因此，程序设计的过程就是利用计算机求解问题的过程。

程序设计是设计、编制和调试程序的过程。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。程序设计过程应当包括分析、设计、编码、测试、排错等不同阶段。由于软件的质量主要是通过程序的质量来体现的，因此程序设计在软件研究中的地位就显得非常重要，其内容涉及有关的基本概念、规范、工具、方法以及方法学。

3. 程序设计语言

程序设计是一个过程，需要借助程序设计语言来描述解决问题的方法和步骤。人和人之间的交流需要通过语言。比如，中国人之间用汉语交流，英国人之间用英语交流。人和

计算机交流信息，也要解决语言问题。程序设计语言是为了方便描述计算过程而人为设计的符号语言，是人与计算机进行信息交流的语言工具。

1.1.2 程序设计的一般过程

程序设计过程不能简单地理解为只编制一个程序。实际上程序设计包括多方面的内容，具体编制程序只是其中的一个方面。有人将程序设计描述成如下的一个公式：

程序设计 = 算法 + 数据结构 + 方法 + 工具

由此看出，在整个程序设计的过程中，要涉及算法的设计、方法的设计和设计工具的选择等诸多方面。虽然人们用计算机求解某一问题时可能编制出各种不同的程序，但是人们编制程序时一般都会按照共同的基本步骤进行，对于大型或复杂程序更是如此。从这个概念出发，一般来说，可以将程序设计的过程分为以下几个步骤：

(1) 分析问题；

(2) 确定解题思路(建立数学模型)；

(3) 绘制流程图或结构图(选择或设计算法)；

(4) 编写源程序；

(5) 上机调试；

(6) 修改源程序，最后确定源程序。

无论是什么类型的实际问题，要用计算机来求解，首先必须分析问题，从具体问题抽象出一个适当的数学模型，用这个数学模型应能得出该问题的精确或近似解。然后确定数学模型的计算方法(即算法)，根据问题的具体要求，可在已知的各种算法中选择一种合适的算法或另设计一种新的算法。接下来就是用某种程序设计语言为确定的算法编制计算机程序，同时准备好作为程序处理对象的各种数据。再接下来就开始程序的调试运行，用一些典型的数据和描述边界条件的数据对程序进行测试，以便发现和纠正程序中的错误。这一过程可能导致前面步骤的多次反复。最后，在调试达到所要求的质量标准之后，程序就可正式投入运行，最终在计算机上得出问题的解。

1.1.3 程序设计语言发展

计算机不能理解和执行人类的自然语言，计算机与人类交流必须使用计算机能够识别的语言。因此，需要一种能够准确表达问题的求解步骤，同时还能够被计算机接受的表达方法，即程序设计语言。自从有了计算机，也就有了计算机编程语言。计算机语言的发展经历了以下几个阶段。

1. 第一代程序设计语言——机器语言

最初的计算机编程语言是所谓的机器语言。一组机器指令就是程序，称为机器语言程序。计算机可以理解并执行的命令即为指令。每种计算机都有自己的指令集合。计算机能够执行的全部指令集合构成计算机的指令系统。每条指令都是由二进制代码 0、1 组成的。因此机器语言程序是二进制代码 0、1 的集合。每种计算机的指令系统都是不同的，因此同一个题目在不同的计算机上计算时，必须编写不同的机器语言程序。

机器语言是低级语言，是面向机器的语言。用机器语言编写程序相当繁琐，程序产生

率很低，质量难以保证，并且程序不能通用。另外，用机器语言编写程序易出错，程序难以检查和调试。

2．第二代程序设计语言——汇编语言

20 世纪 50 年代出现了汇编语言，它使用助记符表示每条机器指令。用指令助记符及地址符号书写的指令称为汇编指令，而用汇编指令编写的程序称为汇编语言程序。例如在 8086 CPU 的指令系统中，用 MOV 表示数据传送，ADD 表示加，DEC 表示将数据减 1，可以使用十进制数和十六进制数。

需要指出的是，计算机不能直接识别用汇编语言编写的程序，必须先由一种专门的翻译程序将汇编语言程序翻译成机器语言程序之后，计算机才能识别和执行。这种翻译的过程称为“汇编”，负责翻译的程序称为汇编程序。汇编语言程序与硬件密切相关，因此这种程序也不能通用。

例如，为了计算表达式“5 + 3”的值，用汇编语言编写的程序与用机器语言(8086 CPU 的指令系统)编写的程序如下：

```
PUSH    BP              01010101
MOV     BP，SP          10001011   11101100
DEC     SP              01001100
DEC     SP              01001100
PUSH    SI              01010110
PUSH    DI              01010111
MOV     DI，0005        10111111 00000101   00000000
MOV     SI，0003        10111111 00000011   00000000
MOV     AX，DI          10001011 11000110
MOV     AX，SI          00000011 11000110
MOV     [BP-02]，AX     10001001 01000110 11111110
POP     DI              01011111
POP     SI              01011110
MOV     SP，BP          10001011   11100101
POP     BP              01011110
RET                     11000011
```

其中每一行的前半部分为汇编语言指令，后半部分(二进制形式的指令代码)为对应的机器语言指令。

3．第三代程序设计语言——高级语言

虽然汇编语言相对于机器语言有很大改进，但依然对机器依赖性大，用其开发的程序通用性差。因此，汇编语言也是低级语言。在保证程序正确的前提下，程序设计的主要目标是程序的可读性、易维护性和可移植性。机器语言程序和汇编语言程序很难达到这样的目标。

随着计算机技术的发展以及计算机应用领域的不断扩大，计算机用户队伍也不断壮大。为了使广大的计算机用户也能胜任程序的开发工作，从 20 世纪 50 年代中期开始逐步发展

了面向问题的程序设计语言，称为高级语言，如 Fortran、Basic、Pascal、Java、C 和 C++ 等，其中 C 和 C++ 是当今流行的高级语言。高级语言与具体的计算机硬件无关，其表达方式接近于被描述的问题，易为人们接受和掌握。用高级语言编写程序比用低级语言容易得多，并大大简化了程序的编制和调试，使编程效率大大提高。高级语言的显著特点是独立于具体的计算机硬件，通用性和可移植性较好。用高级语言编写的程序同自然英语语言非常接近，易于学习。同时，一条高级语言程序的语句相当于几条机器语言的指令，用高级语言编写程序也不需要熟悉计算机硬件。

要计算表达式“5 + 3”的值，如果使用高级语言来编程就简单得多。

用 Basic 语言编写的程序如下：

```
10  I=5
20  J=3
30  K=I+J
```

用 C 语言编写的程序如下：

```
main()
{   int i，j，k;
    i=5;
    j=3;
    k=i+j;
}
```

必须指出，用高级语言编写的程序(称为源程序)都要翻译成机器语言程序(称为目标程序)后才能被计算机执行。

从程序设计语言的发展过程可以看出，程序设计语言越来越接近人们的自然语言。目前高级语言已经形成了一个庞大的家族，包括结构化程序设计语言、面向对象程序设计语言、可视化程序设计语言、网络程序设计语言等。

随着计算机硬件性能的不断提高，人们使用计算机解决问题的能力不断提高，用高级语言编写的计算机程序也越来越复杂，但同时也出现了一些问题。1968 年，荷兰数学家迪杰斯特拉发表了论文《GOTO 语句的害处》，指出调试和修改程序的难度与程序中包含 GOTO 语句的数量成正比。从此，结构化程序设计理念逐渐确立起来。结构化程序设计思想包括：整个程序由若干模块搭接而成，每个模块采用顺序、选择和循环三种基本结构作为程序设计的基本单元。这样的程序有以下四个特征：只有一个入口；只有一个出口；无死语句；无死循环。

C 语言是这种程序设计语言的典型代表。

面向对象的程序设计最早是在 20 世纪 70 年代提出的，其出发点和基本原则是尽可能地模拟现实世界中人类的思维进程，使程序设计的方法和过程尽可能接近人类解决现实问题的方法和过程。随着面向对象程序设计方法和工具的成熟，从 20 世纪 90 年代开始，面向对象程序设计逐渐成为最流行的程序设计技术。Java、C++、C# 等都是面向对象程序设计语言。

可视化程序设计是在面向对象程序设计的基础上发展起来的。可视化程序设计语言把图形用户界面设计的复杂性封装起来，编程人员只需用系统提供的工具在屏幕上画出各种

图形对象，并设置这些图像的属性，就可以让语言工具自动生成代码，这大大提高了编程效率。Visual Basic、Visual C++ 等都是可视化程序设计语言。

网络程序设计是在网络环境下进行的程序设计，包括服务器端程序设计和客户端程序设计。常用的服务器端程序设计语言有 JSP、ASP 和 PHP，常用的客户端程序设计语言有 JavaScript 和 VBScript。

4．第四代程序设计语言——非过程式语言

20 世纪 80 年代初，随着数据库技术和微型计算机的发展，出现了面向问题的非过程式程序设计语言。利用第四代语言工具开发软件只需考虑“做什么”，而不必考虑“如何做”，不涉及太多的算法细节，编程效率大大提高。迄今为止，使用最广泛的第四代程序设计语言是数据库查询语言，如 Oracle、Sybase 等都包含有第四代语言成分。

5．第五代程序设计语言——智能型语言

第五代程序设计语言是智能型的计算机语言。这代程序设计语言力求摆脱传统语言的状态转换语义模式，以适应现代计算机系统知识化、智能化的发展趋势，主要用于人工智能的研究。其代表语言是 LISP 语言和 PROLOGE 语言。LISP 语言属于函数型语言，以 λ 演算为基础。PROLOGE 语言属于逻辑型语言，以形式逻辑和谓词演算为基础。

未来，第四代和第五代程序设计语言会有很大发展，但目前它们还很不成熟，还有很多问题。目前常用的程序设计语言仍然是第三代高级语言。同时，由于汇编语言运行效率较高，因此在实时控制、实时检测等领域的应用软件中仍然较多地使用汇编语言程序。

1.2 算法及其描述

在程序设计过程中，建立数据模型和确定算法是两个至关重要的步骤。算法是程序设计的灵魂，而语言是形式和工具。只要有正确的算法，就可以使用任何一种程序设计语言编写程序。

1.2.1 算法及其特性

所谓“算法”，是指为了解决一个问题而采取的方法和步骤。

对于解决同一问题，可能有不同的解题方法和步骤。例如求 $1+2+3+\cdots+100$，即有不同的算法。一种方法是进行 1 + 2，然后其和加 3，再加 4，一直加到 100，等于 5050。另一种方法可以采取 $\sum_{n=1}^{100} n=(100+1)\times 50=5050$。对于计算机而言，解决同一问题的不同算法的执行效率有所不同。一般而言，应当选择简单、运算步骤少、运算快、内存开销小的算法(考虑算法的时间复杂度和空间复杂度)。此外，算法还必须满足下列 5 个特征：

(1) 输入：一个算法可以有零个(即算法可以没有输入)或多个输入，这些输入通常取自于某个特定的对象集合。

(2) 输出：一个算法有一个或多个输出(即算法必须有输出)，通常输出与输入之间有某种特定的关系。

(3) 有穷性：一个算法必须是在执行有穷步之后结束(对任何合法的输入)，且每一步都在有穷时间内完成。

(4) 确定性：算法中每一条指令必须有确切的含义，不存在二义性，并且在任何条件下，对于相同的输入只能得到相同的输出。

(5) 可行性：算法描述的操作可以通过已经实现的基本操作执行有限次来实现。

1.2.2 算法的表示

算法的表示可以有多种方法，常用的有自然语言、伪代码、传统流程图、结构化流程图(NS 流程图)和 PAD 图等。下面以求两个自然数的最大公约数的问题为例，分别用几种不同的方法描述算法。

我们采用欧几里得算法求两个自然数的最大公约数，该算法的基本思想是：

设两个自然数是 m 和 n，且 m≥n。欧几里得算法的基本思想是将 m 和 n 辗转相除直到余数为 0。例如，m=36，n=16，m 除以 n 的余数用 r 表示，计算过程如下：

被除数 m	除数 n	余数 r
36	16	4
16	4	0

当余数 r 为 0 时，除数 n 就是 m 和 n 的最大公约数。

1. 自然语言

用自然语言描述算法，其优点是容易理解，缺点是算法冗长、繁琐，容易出现二义性。欧几里得算法的自然语言描述如下：

步骤 1：将 m 除以 n 得到余数 r；

步骤 2：若 r 等于 0，则 n 为最大公约数，算法结束，否则执行步骤 3；

步骤 3：将 n 的值放在 m 中，将 r 的值放在 n 中，重新执行步骤 1。

2. 伪代码

伪代码是介于自然语言和程序设计语言之间的描述方法，它保留了程序设计语言的结构、语句的形式和控制成分等，处理和条件部分允许使用自然语言来表达。至于算法中的自然语言的成分有多少，取决于算法的抽象级别。

欧几里得算法的伪代码描述如下：

```
step1: r←m%n;
step2: 当 r≠0 时，重复执行下述操作；
       step2.1:  m←n;
       step2.2:  n←r;
       step2.3:  r←m%n;
step3: 输出 n。
```

3. 流程图

人们在程序设计实践的过程中，总结出了一套用图形来描述问题的处理方法，从而使流程更直观，更易被一般人所接受。用图形描述处理流程的工具称为流程图。目前用得比

较普遍的是传统流程图和结构化流程图(NS 图)。

(1) 传统流程图。传统流程图是使用一些约定的几何图形来描述算法的组合图，比如用矩形框表示某种操作，用箭头表示算法的执行方向。传统流程图的主要优点是对控制流程的描述很直观，便于初学者掌握。表 1-1 所示的图例就是美国标准化协会(ANSI)规定的一些常用的流程图符号，它们已被世界各国程序工作者普遍采用。图 1-1 为欧几里得算法的流程图。

表 1-1　流程图符号表

符号名称	符　　号	功　　能
起止框		表示算法的开始或结束，每个算法流程图中必须有且仅有一个开始框和一个结束框
输入/输出框		表示算法的输入/输出操作，框内填写需输入或输出的各项
处理框		表示算法中的各种处理操作，框内填写处理说明或算式
判断框		表示算法中的条件判断操作，框内填写判断条件
注释框		表示算法中某种操作的说明信息，框内填写文字说明
流程线		表示算法的执行方向
连接点		表示流程图的延续

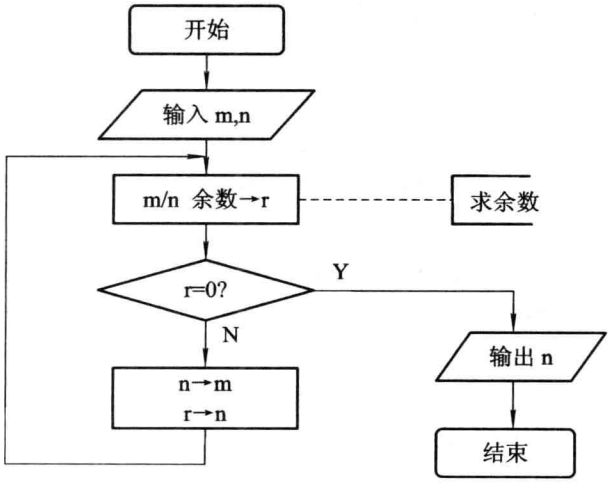

图 1-1　欧几里得算法的流程图

(2) 结构化流程图(NS 图)。在结构化程序设计中，经常使用的工具是结构化流程图，即 NS 图。

NS 图是一种不允许破坏结构化原则的图形算法描述工具，又称盒图。在 NS 图中，去掉了传统流程图中容易引起麻烦的流程线，全部算法都写在一个框内，每一种基本结构也被表示为一个框。用基本结构的顺序组合可以表示任何复杂的算法结构。

NS 图比文字描述直观、形象，便于理解，比传统流程图紧凑易画，尤其是它去除了流程线，结构更清晰。图 1-2 是欧几里得算法的 NS 图。

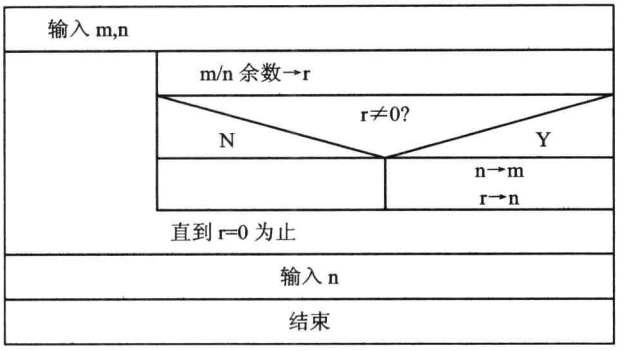

图 1-2　欧几里得算法的 NS 图

1.3　C 语言概述

1.3.1　C 语言的发展

C 语言是目前国际上广泛流行的一种基础的结构化程序设计语言，它不仅是很好的开发系统软件的工具，而且也是很好的开发应用软件的程序设计语言。因此，它深受广大程序设计者的欢迎。

C 语言是在 20 世纪 70 年代初由美国贝尔实验室设计的，并首先在安装 UNIX 操作系统的 DEC PDP-11 计算机上实现。1973 年，K.Thompson 和 Dennis M.Ritchie 把 UNIX 90%以上的内容用 C 语言进行了改写，即大家熟知的 UNIX 第五版。因此，当初的 C 语言是为开发 UNIX 操作系统而研制的，它随着 UNIX 而闻名于世。随着微型计算机的普及，出现了较多的 C 语言系统，其中多数系统接受的源程序都能高度兼容，但是因为没有统一的标准，再怎么兼容，也存在一定的差异。

随着 C 语言被广泛应用，新的 C 语言版本不断推出，其性能也越来越强。到了 1975 年，随着 UNIX 第六版的推出和面向对象程序设计技术的出现，C 语言的突出优点引起了人们的普遍关注。为了克服多个 C 语言系统没有统一标准的不利局面，ANSI(美国国家标准协会)于 1983 年成立了专门定义 C 语言标准的委员会，花了 6 年时间使 C 语言迈向了标准化。ANSI C 标准于 1989 年被采用，该标准一般称为 ANSI/ISO Standard C，简称为 C89。其中详细说明了使用 C 语言书写程序的形式，规范了对这些程序的解释。其主要内容包括：

(1) C 语言程序的表示法；

(2) C 语言的语法和约束；

(3) 解释 C 语言程序的语义规则；

(4) C 语言程序输入和输出的表示；

(5) 一份标准的实现的限定和约束。

1995 年，出现了 C 的修订版，其中增加了一些库函数，出现了初步的 C++，在此基础上，C89 成为 C++ 的子集。此后，C 语言不断发展，在 1999 年又推出了 C99。C99 在基本保留 C 语言的特性的基础上增加了一系列新的特性，随后又几经修改和完善，从面向过程的编程语言发展到面向对象的程序设计语言。

1.3.2 C 语言的特点

C 语言之所以能被推广并被广泛使用，概括地说主要因其具有如下特点：

(1) C 语言是一种表达能力非常强的程序设计语言。它既有高级语言面向用户、容易记忆、便于阅读和书写的优点，又有面向硬件和系统、像汇编语言那样可以直接访问内存物理地址，允许对位、字节和地址这些计算机功能中的基本成分进行操作的功能。C 语言集低级语言和高级语言的开发于一体，因此常被称为中级语言。

(2) C 语言代码紧凑、高效，使用方便、灵活。其程序的目标码执行效率仅比汇编语言低 10%～20%。C 语言仅有 32 个关键字和 9 种控制语句，语言规则也不复杂，核心小巧，所以学习 C 语言相对简单。

(3) C 语言数据类型丰富，具有现代程序设计语言的各种数据类型。C 语言的数据类型主要有整型、实型、字符型、指针类型、数组类型、结构类型等。用它们可以实现各种复杂的数据结构(如链表、树、图、栈等)。因此，C 语言具有较强的数据处理能力。

(4) C 语言运算符丰富。C 语言的运算符共有 34 种，括号、赋值、强制类型转换等都被作为运算符处理，从而使 C 语言的运算类型极其丰富，表达式类型多样化，可以灵活实现其他高级语言难以实现的运算。

(5) C 语言是一种结构化程序设计语言。它具有诸如 if-else、for、do-while、while、switch 等结构化语句，便于采用自顶向下、逐步求精的结构化程序设计技术。

(6) C 语言是便于模块化软件设计的程序语言。C 语言程序可以分割成几个源文件分别进行编译。C 语言的函数结构利于功能模块的分解，并且为程序模块间的相互调用以及数据传递提供了便利。这一特点也为大型软件模块化、由多人同时进行集体开发的软件工程技术方法提供了强有力的支持。

(7) C 语言具有很好的可移植性。大部分代码不用改动就可以从一种机器移植到另一种机器上运行。

(8) C 语言是 C++ 的基础，而 C++ 是目前最流行的面向对象的程序设计语言。

从以上特点不难看出，用 C 语言编写出的程序，既可达到汇编级的效率，又有良好的程序结构。因此，C 语言不仅在各类程序设计中得到广泛运用，而且对于初学者来说，易学易用。

同时，C 语言也存在一些不足。C 语言类型转换比较随便，对数值越界及变量类型的一致性不作语法上的严格检查，这虽然给编程提供了较大的灵活性，但是也限制了编译程序查错的能力。因此，不能过分依赖 C 编译程序而要强调由程序员自己去保证程序的正确性。

1.4 C 程序的基本结构

1.4.1 简单的 C 语言程序例子

为了使读者对 C 语言程序有个初步印象，下面介绍几个例子，以便读者了解 C 程序的基本结构和组成。

【例 1.1】 编写显示字符串“This is my first C program”的 C 语言程序。

程序如下：

```
#include   <stdio.h>
main()
{
    printf("This is my first C program\n");
}
```

这是一个最简单的 C 程序，它把字符串“This is my first C program”显示在屏幕上。该程序由一个函数 main(叫主函数)构成。任何一个程序都必须有此函数。花括号所括的内容是 main 的函数体。每个 C 语言程序的函数都至少有一对{}。

“printf( )”是 C 提供的标准库函数，它完成输出功能。C 语言的输出不像有的高级语言那样由语句完成，而是由函数来完成的，这是它的特点之一。为了顺利调用函数库中的函数 printf，还需要嵌入 stdio.h 这个头文件。C 的系统中预定义了很多不同的头文件，供用户调用，以满足基本要求。头文件中定义了许多函数原型。在头文件 stdio.h 中就定义了函数 printf 的原型，因此，这里要调用系统预定义函数 printf，就必须在程序的最前面增加 #include<stdio.h> 这个预处理命令；否则，printf 函数将无法工作。printf()后的分号是语句结束符。C 语言的每一个语句都以分号终止。

【例 1.2】 编写程序，计算三个实型数的平均值。

程序如下：

```
#include<stdio.h>                          /*嵌入头文件*/
main()                                     /*主函数入口*/
{
    float a，b，c，aver;                    /*定义变量的数据类型为实型*/
    printf("请输入三个实型数\n：");          /*输出语句，作为提示*/
    scanf("%f，%f，%f", &a，&b，&c);         /*从键盘输入 a、b、c 三个实型数*/
    aver=(a+b+c)/3;                        /*求平均值*/
    print("\n average=%f\n", aver);        /*输出计算结果*/
}
```

运行该程序时，首先提示用户输入三个实型数 a、b 和 c，然后用户输入三个数，例如 1、2、3，计算机根据用户输入的三个数求出其平均值，然后把结果以如下形式显示在屏幕上：

```
average=2.0000
```

在此程序中，“/* */”是注释标识符，其间的内容是注释内容，它在程序的编译过程中