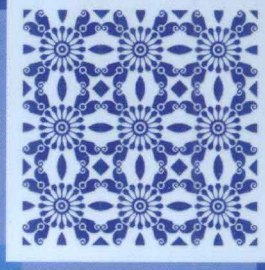
高等院校计算机教材系列

HZ BOOKS
华章教育

PROGRAMMING IN C

C语言程序设计
工程化方法

杨颂华 熊海灵 主编
杨明 毛顺兵 黄春伦 王芳 等编著



机械工业出版社
China Machine Press

高等院校计算机教材系列

PROGRAMMING IN C
C语言程序设计
工程化方法

杨颂华 熊海灵 主编
杨明 毛顺兵 黄春伦 王芳 等编著

机械工业出版社
China Machine Press

本书吸收当前一些C语言教材的优点，加入多年教学和科研积累的丰富经验，由浅入深、地讲授C语言程序设计的基础知识和应用技巧。首先从几个基本案例让读者快速入门，通过小而简单的示例使读者先建立清晰的概念，然后再逐步建立完整的程序；其次介绍C语言的基本语法，各个语法结构的引入都通过实际问题来展开，这样既能引导读者积极思考问题的解决方法，建立程序设计的思想，还能提高读者解决实际问题的能力；最后的高级话题是将一些高级内容从基本主题中分离出来，引导读者不断深入思考程序设计的精髓。

本书是程序设计的入门教材，适合计算机及相关专业学生或工程开发人员使用。

图书在版编目（CIP）数据

C语言程序设计：工程化方法／杨颂华，熊海灵主编；杨明等编著. —北京：机械工业出版社，2012.2

（高等院校计算机教材系列）

ISBN 978-7-111-37282-0

Ⅰ.C…　Ⅱ.①杨…　②熊…　③杨…　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（2012）第014305号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：刘立卿　　佘　洁

北京瑞德印刷有限公司印刷

2012年2月第1版第1次印刷

185mm×260mm·14.75印张

标准书号：ISBN 978-7-111-37282-0

定价：29.8元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

前　言

C 语言是目前在国内外得到最广泛使用的程序设计语言之一。国内外高校的计算机及相关专业几乎都开设了 C 语言程序设计课程作为基础入门课程。由于 C 语言既是一种经典的编程语言，又是大多计算机及相关专业的入门专业课程，对培养学生的计算机专业素养、专业兴趣意义重大。而 C 语言的学习却不是一件简单的事，尤其是对于零基础的初学者来说。目前，市面上有关 C 语言的教材种类繁多，很多教材都是对语法面面俱到，纠缠不休，使得学生学习找不到重点。另外开放平台众说纷纭，更是造成学生不能精通，会看不会用。本书编写的目的是期望帮助初学编程的学生快速入门，并通过积极主动的实践，使编程能力得到较快的提高。

为了真正达到我们的教学目的，学院组织了讲授 C 语言程序设计多年的课程组的老师进行了大量的考证和探讨，同时为支持研究性教学的需要编写了本教材。

全书采用由浅入深、循序渐进的学习方式，将“基本、必需”的教学内容在每章的知识点中循序渐进地引入，根据人类的认知规律——知识点从引入到掌握需要多次重复，我们从第 1 章开始就对基本知识点步步渗透，通过精选的丰富实例，从不同角度进行知识的讲解，帮助读者理解和掌握 C 语言的本质。

本书大致包括三个部分：

- C 语言入门。在介绍基本的 C 语法之前，先是通过回答大家常见的问题来帮助没有任何编程经验的读者理解什么是源程序，怎么学习 C 语言，如何培养程序员的思维习惯，使用什么样的平台等。接下来的快速入门，让读者通过依葫芦画瓢的方式学习编写简单程序，并对其中的知识点进行逐步渗透。
- C 语言的基本内容。本部分主要包括第 2 ~ 4 章，介绍 C 语言的基本语法，包括常量变量定义、数据类型、运算符与表达式、数组、结构体、控制结构及语句、函数、指针等。这一部分篇幅最多，针对各个知识环节，通过启发读者思考的问题、例子讲解语法知识。
- C 语言底层知识与高级话题。第 5 章对 C 语言的输入输出给出了较为彻底的剖析，有一定难度，所以初学者可根据自己学习的程度来理解这一章。另外，前面基本内容中有些高级话题我们剥离出来作为第 6 章。这一部分适合有一定经验后希望深入理解 C 语言的读者学习。

本教材有以下几点特色：

- 首先在导读部分就 C 语言的特点、优势、为什么学习、如何学习、如何教学等展开了探讨，同时要求初学者从开始就要遵守编程规范。
- 在强调基本概念和基本原理方面，遵照人类的认知规律，前面已经讲过的概念，在后面又进一步解释完整，这种循序渐进、重复的方式是比较适合初学者的。
- 使用任务驱动教学方式，用实例讲解语法，同一个问题用不同知识点分别实现。
- 本书定位在入门级，它不是一本百科全书，除了涵盖 C 语言基本内容之外，本书不追求完整性，因此要求授课老师对 C 语言的讲授要重点突出，只讲解编程开发中常用的，其余的内容可要求学生自学。学习编程，最有效的方法是实践，本教材注重实用性。

另外，本书第 7 章给出了一套循序渐进的课程设计题目用于实验/实践环节，并在附录中给出了实际编程开发中的开发规范供读者参考。完成课程设计需要的背景知识读者可从华章网站

（www. hzbook. com）下载。

本书由熊海灵、杨颂华、杨明负责策划、组织，并对全书进行了统稿和审校。其中杨明老师编写了第0、1章，黄春伦老师编写了第2章，王芳老师编写了第3章，毛顺兵老师编写了第4章，杨颂华老师编写了第5章、第7章和附录，第6章由黄春伦、毛顺兵和杨颂华老师合写。

感谢课程组其他老师和许多研究生、本科生对本教材的编写提出的宝贵意见，感谢陈强、闫艳、郭士会、康南南等提供的大量习题和资料。

特别要感谢中国科学院光电技术研究所的马佳光、陈洪斌、任戈、包启亮、毛耀、田竞、陈兴龙等在国家科研一线奋战的资深专家为本书实践部分提供了支持和指导，他们的科研开发和团队管理经验是本书编程规范要求的主要来源，这些工程经验来自无数成功和失败的实践。另外，本书还获得了“中央高校基本科研业务费专项资金资助”（编号2120131001）的支持。

感谢机械工业出版社对教材编写工作的建议及全程支持。

感谢本书所引用的图书资料和网站资料的全体作者和出版社。

由于水平有限，经验不足，加之编写时间仓促，书中难免有错漏不当之处，望广大读者朋友不吝赐教。我们的联系方式：swujsj_c@163. com。

教学安排建议

教学章节	教学要求	课时
第1章 快速入门 （共3学时）	掌握注释，了解变量、输入输出	1
	了解 if 和 for、while	1
	了解函数的概念	1
第2章 C语言基本概念 （共9学时）	准确掌握数据类型、标识符、常量、变量	3
	准确掌握运算符与表达式	3
	掌握数组、结构体的定义和引用	3
第3章 程序结构 （共12学时）	准确掌握 if-else、switch-case 语句	3
	准确掌握 while、do-while、for、转移语句	3
	掌握函数的定义、调用、嵌套调用，理解递归	4
	掌握变量的作用域和存储方式，理解内部函数和外部函数，掌握 typedef 和 const 关键字	2
第4章 指针 （共12学时）	理解指针，能通过指针访问数据，能将指针作为函数参数	2
	准确理解数组元素的地址，理解一维数组与指针的关系，理解二维数组与指向一维数组的指针的关系	4
	理解字符串的两种存储方式，理解指针数组，理解指向指针的指针，理解 main 的参数	3
	了解结构体、malloc、free、链表操作	3
第5章 I/O操作 （共9学时）	了解标准库函数，了解常用库函数的原型，准确理解 printf 和 scanf 函数	3
	了解流和文件的概念，能用常用的库函数来操作文件	4
	了解设备的操作	2
第6章 高级话题 （共9学时）	了解 gcc 编译器、地址映射、预编译	3
	了解联合体、函数指针	4
	了解网络通信	2
总课时	第1~6章建议课时	54
	建议实验课时（第7章）	36

说明：

1）建议课堂教学全部在多媒体机房内完成，实现“讲－练”结合。

2）建议教学分为核心知识技能模块（前4章的内容）和技能提高模块（第5~6章的内容），其中核心知识技能模块建议教学学时为36，技能提高模块建议教学学时为18，不同学校可以根据各自的教学要求和计划学时数对教学内容进行取舍。

目　录

第0章 导 读

0.1 来源与背景

0.1.1 为什么学习C语言

C语言是目前在国内外最广泛使用的程序设计语言之一，不仅计算机专业的工作者在使用，而且广大工程技术人员也喜爱使用。国内外高校的计算机及相关专业几乎都开设了C语言程序设计课程作为基础入门课程，非计算机专业也普遍开设了该课程作为参加全国计算机等级考试的程序语言之一。

为什么C语言如此普及以及被喜爱？这跟它的一系列优势是分不开的：

首先，C语言作为一种结构化语言，描述能力很强，支持当前程序设计中广泛采用的自顶向下的结构化程序设计技术，适合作为教学语言，加上C语言强调灵活性，使程序设计人员能有较大的自由度以适应宽广的应用面，有着广泛的应用领域，不仅可以用来编写系统软件，也常用来编写应用软件。

其次，C语言是一种编译型程序设计语言，它兼顾了多种高级语言的特点，并具备汇编低级语言的功能，这可以帮助人们更好地了解计算机。C语言有功能丰富的库函数、运算速度快、编译效率高、有良好的可移植性，而且可以直接实现对系统硬件的控制。用C语言来编写目标系统软件，不仅会大大缩短开发周期，而且明显地增加了软件的可读性，且程序便于改进和扩充，从而利于研制出规模更大、性能更完备的系统软件。

此外，C语言程序具有完善的模块化程序结构，从而为软件开发中采用模块化程序设计方法提供了有力的保障。如果学习过C语言，就能很容易地学习现有的其他高级编程语言，因为目前大多高级语言的基本语法都是以C语言为基础（像Java，C++，C#等）。因此，使用C语言进行程序设计已成为软件开发的一个主流。

当然很多关于C语言的教材中都会讲C语言有丰富的运算符、可移植性好、生成的目标代码质量高等特点，这些优点初学者也许暂时还不能深刻理解，待学完C语言以后，在应用中对C语言和其他传统的高级语言作一简单比较，就会有较深的体会。

也有人一直以为C++是C的升级，比C更先进，所以建议直接学习C++。从某种意义上来说，C++与C几乎是完全不同的东西，C是面向过程的，C++是面向对象的，两者根本不具有可比性。

强调学习C语言，并不是排斥其他编程语言的学习，本书的例子用其他编程语言也容易学习并实现，然而，我们的观点是，充分了解最接近硬件的编程语言，有助于更好地理解抽象的编程语言及其用法，也有助于提高最实用的编程技巧。

0.1.2 C语言的来源

C语言是1972年由美国的丹尼斯·里奇（Dennis Ritchie）设计发明的，并首次在Unix操作系统的DECPDP-11计算机上使用。它由早期的编程语言BCPL（Basic Combined Programming Language）发展演变而来。1970年，AT&T贝尔实验室的肯·汤普森（Ken Tompson）根据BCPL语言设计出较

先进的，但很接近硬件的 B 语言（取 BCPL 第一个字母），由于 B 语言过于简单，功能有限，在 1972 至 1973 年间，丹尼斯·里奇在 B 语言的基础上设计出了 C 语言（取 BCPL 第二个字母）。

后来，C 语言做了多次改进，但主要还是在贝尔实验室内部使用，直到 1975 年 Unix 第 6 版公布后，C 语言的突出优点才引起人们的普遍注意。1977 年出现了不依赖于具体机器的 C 语言编译文本——《可移植 C 语言编译程序》，使 C 移植到其他机器时所做的工作大大简化，这也推动了 Unix 操作系统迅速在各种机器上的实现。随着 Unix 的日益广泛使用，C 语言也迅速得到推广，它们像一对孪生兄弟，在发展过程中相辅相成。1978 年以后，C 语言已先后移植到大、中、小、微型机上，已独立于 Unix 和 PDP 了。现在，C 语言已风靡全世界，成为世界上最广泛应用的几种计算机语言之一了。

0.1.3 C 语言的标准

随着微型计算机的日益普及，出现了许多 C 语言版本。由于没有统一的标准，使得这些 C 语言之间出现了一些不一致的地方。

以 1978 年发表的 Unix 第 7 版中的 C 编译程序为基础，Brian W. Kernighan 和 Dennis M. Ritchie（合称 K&R）合著了影响深远的名著《The C Programming Language》，这本书中介绍的 C 语言成为后来广泛使用的 C 语言版本的基础，被称为标准 C。

1983 年，美国国家标准协会（American National Standards Institute，ANSI）委任一个委员会 X3J11 对 C 语言进行标准化。经过长期艰苦的过程，该委员会的工作报告于 1989 年 12 月 14 日被正式批准并于 1990 年春天颁布，这就是 ANSI C89 标准。ANSI C89 比原来的标准 C 有了很大的发展，譬如加入了一些新的特性，规定了一套标准库函数。随后 K&R 按照 ANSI C 的标准重新修改了他们的经典著作《The C Programming Language》。接下来的几年内，ANSI C89 都是以发行技术勘误和标准附录的形式不断更新。1999 年推出了 C99 标准（ISO/IEC 9899：1999），该标准又加入了许多新的特性，但目前仍没有得到广泛支持，因此 C89 仍然是目前最广泛采用的 C 语言标准[⊖]，大多数编译器都完全支持 C89。

0.2 如何学习 C 语言

很多人对学习 C 语言感到无从下手，作为老师，经常被问到同一个问题：怎样学习 C 语言？还有一些学生在学完一学期的 C 语言后，感觉语法都记得差不多了，书中例题也懂、也会，就是碰到实际问题时不知如何编程，也不敢编程。那么，如何才能学好 C 语言呢？

首先学习 C 语言不是一朝一夕的事情，但也不是说需要花费十年时间才能精通。如何在较短的时间内获得最多的收获，需要考虑几个方面：作为学生，首先要选择比较好的教材，市面上关于 C 语言的教材非常多，选择优秀教材是事半功倍的关键因素。然而即使是最经典、最权威的书，也没有办法面面俱到，所以手边常备一本《C 语言参考手册》是十分必要的，当然随着网络的普及，相关资料都可在网上查到。

接下来就是编程环境的选择：选择 Unix/Linux 还是 Windows，这是个很大的问题，因为不同的编程环境会造就出不同思维的程序员。Windows 的程序员大多依赖集成开发环境，因为集成开发环境很容易上手，在 Windows 上学习 C 语言，只需要会按几个基本的 Visual C++ 工具栏按钮就可以开始写“Hello，World!”了，而在 Unix 下，则需要一些控制台操作的基本知识。Visual C++6.0 使用很方便，调试也很直观，其默认的编译器也包括了 C 编译器，可以使用 C 语言进行编程，并在命令行模式下运行调试，但 Visual C++6.0 毕竟主要是为 C++ 特别是 MFC（微软基础类库）所设计的集成开发环境所使用的，因此使用中多有不便之处。而且 Visual C++6.0 界面较为复杂，很多功能

⊖ C 标准的目的是为了精确定义 C 语言，在表达上追求准确和无歧义，而不是教大家怎么编程。

在C编程开发中用不到，因此作者不推荐它作为首选。此外，还有一些大学的C语言课程使用Turbo C 2.0作为实验环境，这也是不可取的，在多年的教学和培训中，我们发现由于DOS操作系统的淡出，Turbo C 2.0在除了计算机等级考试之外的领域中基本没有了用武之地。而Unix与C可以说是共生的，Unix的思考方式和习惯更加符合C语言的思考方式和习惯。在Unix下，你可以找到无数优秀的源代码尽情阅读，你可以方便地查看某个库函数的联机手册，还可以看到最优秀的代码风格。虽然使用Unix平台需要面对的是各种纷繁复杂的命令，完全不同于Windows平台的思考方式，但是几乎所有C语言的高级教程都是基于Unix平台的（比如《C专家编程》等），所以一开始学习有点痛苦是正常的，但也是值得的。GNU/Linux平台继承了Unix平台的优秀之处，其应用范围非常广泛。从服务器系统到嵌入式系统，从民用到军用，都有Linux系统的应用，特别是Internet网络中，Linux服务器更是扮演了非常重要的角色。当你连接到Internet时，其实有很多Linux服务器在默默为你服务。因此，在GNU/Linux平台上编程，有很多直接应用的可能。而且，GNU/Linux平台下的编程概念、方法和操作，20多年来基本没有什么变化，因为它们都是经过时间沉淀后的精华，具有很高的稳定性，这对于程序员的学习投资是一种良好的保护。

在GNU/Linux平台上编程，可较好地体会Unix哲学思想。Unix哲学思想可归纳为KISS（Keep It Simple，Stupid）原则，中文可以翻译为“大道至简”。Linux平台源自Unix这一古老的系统，至今已经在很多嵌入式系统和服务器上默默为整个世界提供服务，这一巨大的应用规模就是依赖这一基本的哲学思想实现的。然而，前辈高手们并没有先著书立说阐述完整的Unix哲学思想，而是用无数简洁优美的范例来行不言之教。在GNU/Linux平台上编程，时时处处去尝试体悟其中的Unix哲学，不仅仅是一次次的恍然明悟，更可以直接为程序设计带来先进的指导思想，让编写出来的程序具有较好的稳定性和扩展性。虽然目前已经出版了或者在网络上流传着很多种版本的Unix哲学诠释，但我们建议初学编程的人们不必花费很多时间来细读，因为这些前辈的体悟只有与自己的编程经验结合起来，才能大放异彩，点燃编程中的智慧。

再接下来就是对教材的使用。一般建议读者从导读开始，在导读中，作者针对如何学、如何教、怎样学跟大家分享了一些经验。看完导读，还要浏览一下目录，了解书的整体结构，顺便给自己安排一个学习计划。学习C语言，必须注意每一个细节，书上的例子代码一定要自己亲自输入一遍，编译、执行、输出都跟书上说的一致才能算是学完了一个例子，如果不一致，就要仔细找原因。另外在已有的例子上进行改造，并将改造后的例子归类保存。接下来就是独立完成大量的习题，在不断实践中寻找学习的方法，编程的经验。

总之，希望读者在不断的练习过程中积累经验和自信，要敢于编程，积极编程，不怕错误，认真分析错误，不断调试，在实践中积累自信心与成就感，最终成为优秀的编程人员！

0.3 如何教C语言

C语言作为大多数高校的第一门计算机基础程序设计的入门课程，其自身的重要性及其在课程体系中的基础性决定了大家对其教学效果和教学质量的重视，同时对学生后续的学习及应用都有着广泛的影响。然而目前很多高校的C语言教学，由于教材内容面面俱到，部分老师抓不住重点，教学过程过分偏重于C语言知识的详细讲解，使得学生对**程序设计的思想**、**程序设计的方法**、**程序设计的风格**不太重视，动手能力较差。最终结果是课堂能听懂，笔试题会做，但就是不会、也不敢编写和调试程序，即便写出也不能测试程序的正确性，同时程序的可读性很差。作者希望教师一开始严格按照规范教学生，强迫学生从最开始的学习就必须模仿最优秀的代码风格去编写程序。同时，也希望教师在整个教学过程要不断重复前面章节的内容，通过大量的**实验与设计**让学生找到学习C语言的乐趣。

本教材的编写旨在帮助读者提高编程技能，超越入门水准。因此教师在教授过程中，要把模块

化的思想清晰地传授给学生，使学生在经过对编程知识的基本了解后，能从较高的角度知道如何把握以及如何开发程序，而不仅仅是对语法的死记硬背。书中也提供了关于编程的实践步骤，实际操作的实例，以及实例之后需要改进的思考等等。通过这一系列的学习，希望读者能够掌握系统编程的技巧。

0.4 编译过程

在真正进入学习编程之前，我们简单介绍几个概念。初学编程，学生会问什么叫源代码？什么叫可执行程序？怎样从源代码编译成可执行程序？这些概念可能在不同的教材里有不同的说法，为了简化，我们只讲跟本教材相关的 C 语言的编译过程。

源代码也称源程序，是指人们编写的计算机语言指令。因为计算机能够理解的语言是机器语言（二进制代码），然而用二进制代码书写的程序人们不容易理解，因此有了接近自然语言的高级语言，用高级语言编写的程序便于理解及修改，我们一般把这种用高级语言编写的程序叫源程序。然而编写好的源程序（如 C 语言编写的源程序）并不能被计算机直接理解并运行，必须把它“翻译”成机器语言，才能运行程序，得到相应的程序功能。这个可以执行的程序就是可执行程序，也叫做目标代码，是指源代码经过编译程序产生的能被计算机直接识别的二进制代码。

初学 C 语言者都想当然地以为自己编写了源程序，就应该直接看到程序运行后的结果，这是不可能的。因为我们编写好的 C 语言源程序只是一些可读的语句指令，只有“翻译”成计算机可以理解的并能执行的二进制指令，才有可能得到期望的结果，这个“翻译”的过程叫做编译，是通过编译器来完成的。人们编写计算机源代码的最终目的也是将这些可读的文本翻译成计算机可以执行的二进制指令。

对于编译语言来说，例如 C、C ++ 、Java，源代码的修改不能改变已经生成的目标代码。如果需要目标代码做出相应的修改，必须重新编译。

用 C 语言编写的源程序，保存后的文件名形式一般为 *.c；经过编译生成目标程序；由于 C 语言已经设计好很多通用子程序，称为函数库，编写的程序或多或少地用到函数库中的一些函数，因此除了编译以外，还需要从函数库中把需要的函数“连接”到目标程序中，形成操作系统能执行的可执行文件，这个连接过程由连接程序来完成，最后运行可执行文件才能得到期望的结果。

所以对初学者来说，一个 C 语言程序的开发过程需要编辑源程序、编译生成目标程序、连接生成可执行文件、运行可执行文件得到结果。源程序是可以直接阅读的、可修改的、可编辑的程序文本，目标程序和可执行文件都是由开发平台自动生成的。

0.5 gcc 简介

gcc 是 Linux 下基于命令行的 C 语言编译器，也是 GNU（一个技术组织）最具有代表性的作品。在 GNU/Linux 平台上编程最常用的命令就是 gcc。gcc 设计之初仅仅作为一个 C 语言的编译器，可是经过十多年的发展，gcc 已经不仅仅能支持 C 语言，它现在还支持 Ada 语言、C ++ 语言、Java 语言、Objective C 语言、Pascal 语言、COBOL 语言，以及支持函数式编程和逻辑编程的 Mercury 语言，等等。因此 gcc 已不再简单地看做 GNU C Compiler 的意思，而是 GNU Compiler Collection，也就是 GNU 编译器家族的意思了，它目前已经成为 Linux 下最重要的编译工具之一。

用 gcc 编译程序生成可执行文件的过程不单单是一个编译的过程，而且要经过预处理（Pre-Processing）、编译（Compiling）、汇编（Assembling）、连接（Linking）几个过程。就 gcc 来说，其本身是一个十分复杂的命令，合理地使用其命令选项可以有效地提高程序的编译效率及优化代码。gcc 拥有众多的命令选项，有 100 多个编译选项可用。初学者只需要掌握基本的几个选项就可以学习编程了。

第1章 快速入门

与其他教材不同，我们在第1章就会把C语言的主要概念展示给读者。读者将会看到，C语言程序无非是由函数组成的，每个函数用一对 {} 表示其范围。读者还会初步接触数据类型、变量、常量、数组等概念，接触到if语句、for循环语句等语句格式。总之，读者会通过我们的每一个实例，了解到C语言程序是什么样子，当然，这只是第一次接触。进一步的学习将会在后续章节展开，这也是一个不断重复、加深的过程。好吧，我们还是从“Hello，world”开始，进入C语言的编程大门。

1.1 Hello，world

【例1-1】 屏幕输出“Hello，world”信息。

```
1    #include <stdio.h>                    // 编译预处理命令
2
3    int main(void)                        // 主函数
4    {
5            printf("Hello, world\n");     // 输出
6            return 0;                     // 返回
7    }
```

运行结果如下：

```
Hello, world
```

上面这几行代码就是一个简单的C语言程序，它实现的功能是在屏幕上显示一行信息“Hello，world”。

一个C语言程序，通常是由带有#号的编译预处理命令开始（关于编译预处理详见第2章）。程序中的第一行#include <stdio.h>就是编译预处理命令，因为花括号内调用了printf()函数，该函数是C语言提供的标准输出函数，已经在系统文件stdio.h中声明，使用文件包含命令include后，可以在程序中直接调用。

程序第三行int main(void)是主函数，其中int是整型数据类型（关于数据类型详见第2章），这里是指函数返回的值类型为整型。main是函数名，函数名后紧跟小括号，小括号内可以有参数，也可以无参数。有参数时一定要声明参数的数据类型，本例无参数，可以使用空类型void。花括号内是程序体（也叫函数体），其内包含一条或多条语句，语句用分号结束。每一个C源程序都必须有且只能有一个主函数，即main()函数。

这个程序之所以简单，是因为花括号内即程序体只有一条输出语句和一条函数返回语句。输出语句（行号5）完成的功能是在屏幕上输出“Hello，world”信息，即输出printf()函数中双引号内的内容，但是细心的同学会发现双引号内的“\n”没有输出。这里\是转义符号，它告诉编译器\后面的字符需要用特殊的方式进行处理，不再照原样输出，\n代表换行。关于printf()函数，我们会通过几个例子不断应用以加深理解。关于转义字符和其对应的意义，我们会在第2章进行详细介绍。

程序一般都带有注释，注释在C语言中有两种使用方法，一是/*…*/，二是//，它们既可以出

现在语句行里（注释可以占一行的一部分），也可以单独占一行或多行。用/* 和 */括起来的内容或者在//之后的部分为注释内容。注释内容都不参加编译、运行。但好的程序是应当加上必要的注释的，它可以用来提高程序的可读性和可维护性。因为一个较大的程序，经过一段时间，有些地方可能连编程者都忘记了，增加注释可以帮助恢复记忆，调试程序时也容易找出错误。但注释也不是要求面面俱到、每行都有，错误的注释反而影响对程序的正确理解。

注意：注释不可嵌套。

读者可以仿照例子试着编写显示下列两行信息的 C 语言程序：

```
Hello, C program!
I will be a good programmer.
```

一开始，我们希望初学者仿照例子去编写，在编写过程中慢慢理解各种语法，譬如 printf() 函数是格式化输出函数，一般用于向标准输出设备（屏幕）按规定格式输出信息。在编写程序时经常会用到此函数。随着学习的不断深入，我们将慢慢展开对输出函数的介绍，但在本章我们不对语法做很详细的阐述。

1.2 变量与表达式

编写程序是为了让计算机根据我们的要求来解决问题；而要解决问题，计算机就必须做各种运算（计算机只会做运算）；做运算就必然涉及常量、变量、运算符及表达式；参与某个运算的数据（常量和变量）应该满足某种关系，这种关系就体现为参与运算的数据的类型。

【例 1-2】 计算两个整数的和。

```
1  #include  <stdio.h>
2
3  int sum = 0;                                  // 声明了 1 个整型变量,其名字为 sum,并初始化为 0
4  int main( void )
5  {
6          int i, j;                             // 声明了 2 个整型变量: i 和 j
7          i = 3;                                // 使变量 i 的值为 3
8          j = 4;                                // 使变量 j 的值为 4
9          sum = i + j;                          // 计算 i 和 j 的和,并保存到 sum 中
10         printf("%d + %d = %d\n", i, j, sum);  // 调用 printf()函数输出结果
11         return 0;
12 }
```

运行结果如下：

```
3 + 4 = 7
```

上例中声明了三个变量：sum、i 和 j，它们的类型都是整型（int）。**一个变量名，对应一块内存区域，程序中使用变量名来存取该内存区域中的数据。**C 语言规定，源程序中所有用到的变量都必须先声明后使用。声明变量的语法形式为：

```
类型区分符 变量1,变量2[…];
```

“类型区分符”规定了变量 1、变量 2……的数据类型，数据类型决定了变量在内存中的存储形式、变量的取值范围、可以对该变量进行的操作（运算）等。最常用的基本数据类型有：int、float 和 char。

- int 表示整型，其值可以是正整数、负整数和 0，例如 - 1000、 - 23、 - 1、0、1、23、10000 等。

- float 表示浮点型，一般用来存放带有小数点的数，当然也可以存放整数（默认小数点在最低权值位的右边）。例如 3.14、2.713、0.04、4.0 等，常用 float 型的变量来存放。
- char 表示字符型，主要用来存放字符，如'a'、'b'、'!'等。

对某个变量存入一个新数据，需要用到赋值运算符（=）。赋值运算符的简单使用形式是：

```
变量1 = 变量2(或常量);
```

其作用是把变量2（或常量）的值存储到变量1中，变量2（或常量）的内容不变，即把变量2（或常量）所对应的内存区域的数据拷贝到变量1所对应的内存区域。赋值运算要求变量1和变量2（或常量）的类型相同，如果不同将发生类型转换（详见第2章）。

除了赋值运算符外，C语言中还提供了其他丰富的运算符，最常用的是算术运算符、关系运算符和逻辑运算符。

- 算术运算符包括加法（+）、减法（-）、乘法（*）和除法（/）。这些运算符的作用跟数学中对应的运算符一样。例如上例中的“sum = i + j;”，其作用是先计算 i 和 j 的和，然后赋值给 sum。之所以这样做，是因为 + 的优先级比 = 高。运算符的优先级详见第2章。
- 关系运算符是对两个操作数据进行比较的运算符，包括大于（>）、小于（<）、等于（==）、不等于（!=）、大于等于（>=）、小于等于（<=）。关系运算的结果为0（代表假）或1（代表真）。结果为真是指关系运算符所描述的关系成立，否则为假。
- 逻辑运算符也叫布尔运算符，是对逻辑值进行的运算，包括与（&&）、或（||）、非（!）。C语言中的逻辑运算与数字逻辑中的逻辑运算规则一样，只是**C语言中参与逻辑运算的操作数可以是任意数值，0代表假，非0代表真**，运算结果是0（代表假）或者1（代表真）。

上例中还使用了0、3、4，C语言中，称这种直接输入的数字为常量。常量不仅可以是整数，也可以是浮点数，如3.14、2.713等，也可以是字符，如'A'、'b'等（字符常量必须用单引号括起来），还可以是字符串常量，如"Hello World!"、"How do you do?"等（字符串常量必须用双引号括起来）。关于常量的更多细节参见第2章。

在C源程序中，可以根据变量的定义位置将它分成两类。一类是在一对大括号 {} 内定义的变量，这类变量称为**局部变量**，该变量只在该对大括号内有效。例如，上例中的 i 和 j 就只在 main 函数体中有效。另一类是不在任何大括号内定义的变量，这类变量称为**全局变量**，全局变量不受任何大括号的限制，在其后的任何大括号内都可以使用（在其前面的需要先声明一下）。例如，上例中的 sum，在 main 函数体外定义，在 main 函数体内也可以使用。有关变量的作用域问题，详见第3章。

printf() 是格式化输出库函数，其意义是按指定的格式在屏幕上显示信息。其格式如下：

```
printf("格式控制串",参数表);
```

“格式控制串”是一个字符串，它表示了输出量的数据类型。这里%d为格式字符，表示按十进制整型处理。其他各种类型的格式表示法可参阅第5章。

在 printf() 函数中还可以在格式控制串内出现非格式控制字符，它们在屏幕上将原文照印，如例1-1中的“printf("Hello, world");”。

在例1-2的 printf() 函数中，“参数表”给出了多个输出的量。当有多个量时，用逗号隔开。如“printf("%d + %d = %d\n", i, j, sum);”这条语句的执行过程是：把 i 变量的值输出在第一个格式符%d的位置，+ 号不动，j 变量的值输出在第二个格式符%d的位置，= 号不动，原样照印，i + j计算后的值（即变量 sum 的值）输出在第三个格式符%d的位置，接下来是换行符\n，屏幕输

出3+4=7后回车换行。

通过两个简单的程序，我们简要介绍一下编程风格问题，后面在不同阶段的学习中也会不断提到，要成为优秀的程序员，在最初的编程学习中就要养成良好的编程风格。

从书写清晰，便于阅读、理解以及维护的角度出发，编写程序时应遵循以下规则：

1）一个说明或一个语句占一行。

2）用｛｝括起来的部分，通常表示了程序的某一层次结构。｛｝一般与该结构语句的第一个字母对齐，并单独占一行。

3）低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写，以便看起来更加清晰，增加程序的可读性。

4）不得在变量声明中赋值，赋值等号应对齐，等号后应有空格，语句段之间应空行并在段前给出注释，输出的内容应完备，譬如例1-2，如果只输出一个求和结果7就不恰当，而应输出i+j=7或者3+4=7。

5）英文书写习惯：逗号后应有空格，一句话的首字母应大写。

在编程时应力求遵循这些规则，以养成良好的编程风格的习惯。

1.3 输入输出

前面的例题中要求计算两个整数的和，是通过给两个变量赋值，然后计算其和来完成的。如果换成其他两个整数求和，这个程序就要修改两条赋值语句，很不方便。C语言提供了一个标准输入函数，对于同样的问题使用该函数就可以输入任意两个整数，对其求和，而不需要修改源程序。见例1-3。

【例1-3】 输入两个整数，输出其和。

```
1   #include <stdio.h>
2
3   int main(void)
4   {
5           int i, j;                                       // 定义两个变量
6           scanf("%d %d", &i, &j);                         // 从键盘输入两个整数
7           printf("i=%d, j=%d, i+j=%d \n", i, j, i+j);// 输出
8           return 0;
9   }
```

运行结果如下：

运行时等待输入两个整数，从键盘输入，譬如5、空格、8、回车，屏幕显示：

```
i=5, j=8, i+j=13
```

例1-2是定义两个变量后分别给两个变量赋值，然后输出时求和。现在例1-3比较灵活了，它从键盘输入任意两个整数，然后输出其和。这里使用了格式化输入函数scanf()。

与格式化输出函数printf()一样，它们可以在标准输入输出设备（键盘、屏幕）上以各种不同的格式读写数据。scanf()函数用来从标准输入设备（键盘）上读数据，printf()函数用来向标准输出设备（屏幕）写数据。

scanf()函数的调用格式为：

```
scanf(<格式化字符串>, <地址表>);
```

其中的“格式化字符串”仍以%开始，后跟一个或几个规定字符，用来确定输入内容的数据格式。“地址表”是需要读入的所有变量的地址，而不是变量本身，取地址符为'&'。各个变量的地址之间用','分开。关于取地址符详见第4章。