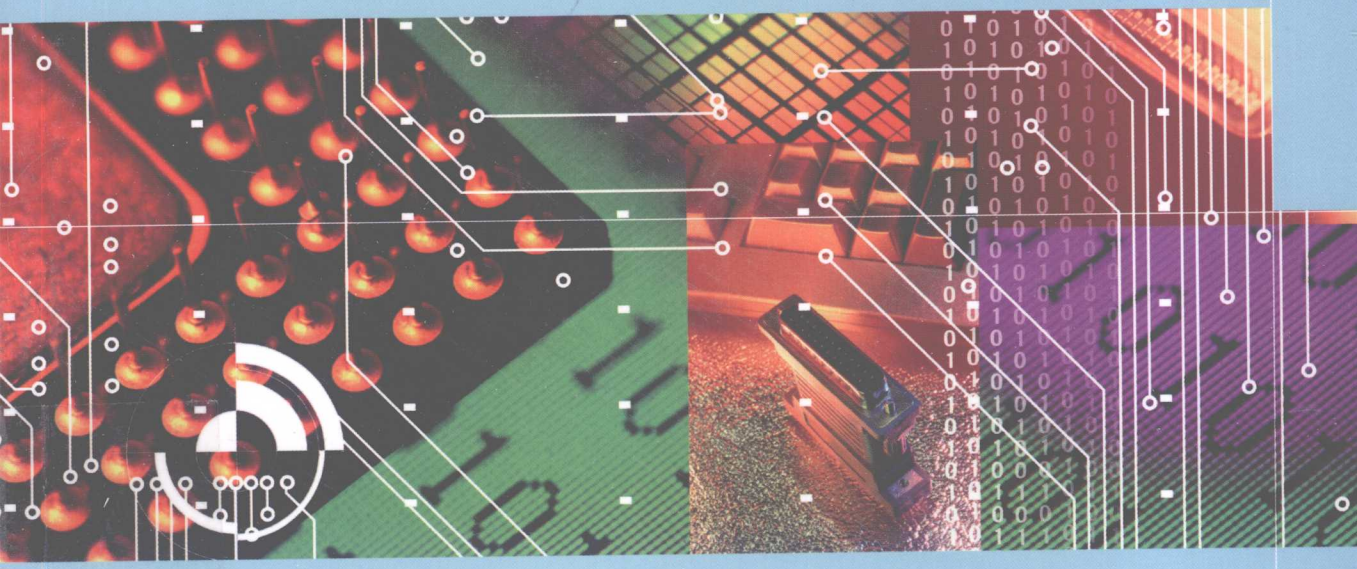
普通高等教育“十一五”国家级规划教材

# 程序设计基础（第2版）

张杰敏 主编



高等教育出版社

普通高等教育“十一五”国家级规划教材

# 程序设计基础

（第2版）

张杰敏　主编

高等教育出版社

## 内容提要

本书是普通高等教育“十一五”国家级规划教材，是为程序设计的初学者编写的教材，凡具备初级计算机知识的读者都能读懂。本书将对C语言进行系统化的讲解，并适时、适当地介绍相关的程序设计理论，将理论和实践有机结合，形成相得益彰的知识体系，以灌输朴素的软件工程思想，培养可持续发展的程序设计能力。

本书描述的是基于ANSI标准的C语言。主要内容包括程序设计语言的介绍和C语言的基本概念，程序设计中数据的类型、地址、值、运算、存储和传输等多方面属性，结构化程序设计和C控制流，模块化设计和C函数，自定义数据类型指针、数组、结构、联合及其在程序设计中的应用，程序输入输出操作和界面，应用标准库编写应用程序的方法。有关程序设计理论方面的内容，无论理解程度的深浅，都会在指导实践中有所收获。

本书适合作为各大专院校C语言程序设计相关课程的教材，也可供对程序设计有兴趣的读者参考阅读。

**图书在版编目（CIP）数据**

程序设计基础 / 张杰敏主编. —2 版.—北京：高等教育出版社，2009.3

ISBN 978-7-04-026264-3

Ⅰ.程… Ⅱ.张… Ⅲ.程序设计-高等学校-教材

Ⅳ.TP311.1

中国版本图书馆CIP数据核字（2009）第008057号

**策划编辑** 冯 英 **责任编辑** 萧 潇 **封面设计** 张志奇 **责任绘图** 杜晓丹

**版式设计** 陆瑞红 **责任校对** 俞声佳 **责任印制** 宋克学

| | | | |
|---|---|---|---|
| **出版发行** | 高等教育出版社 | **购书热线** | 010-58581118 |
| **社 址** | 北京市西城区德外大街4号 | **免费咨询** | 800-810-0598 |
| **邮政编码** | 100120 | **网 址** | http://www.hep.edu.cn |
| **总 机** | 010-58581000 | | http://www.hep.com.cn |
| | | **网上订购** | http://www.landraco.com |
| **经 销** | 蓝色畅想图书发行有限公司 | | http://www.landraco.com.cn |
| **印 刷** | 北京地质印刷厂 | **畅想教育** | http://www.widedu.com |
| | | **版 次** | 2003年9月第1版 |
| **开 本** | 787×1092 1/16 | | 2009年3月第2版 |
| **印 张** | 18.75 | **印 次** | 2009年3月第1次印刷 |
| **字 数** | 460 000 | **定 价** | 22.30元 |

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。



**物料号 26264-00**

# 前　言

程序设计是计算机技术领域的核心工作，任何对于目标的创意或规划，都需要通过程序设计最终实现。程序设计语言因计算机的产生而产生，伴随计算机的发展而发展。迄今为止，被广泛使用的程序设计语言多达几十种，且仍在不断地推陈出新。

在程序设计的浩瀚书海中，注重实践、关注某一种具体的程序设计语言的内容和使用方法的书籍很多。而注重理论、讨论语言的特性和结构、介绍程序设计语言中的基本概念和基本成分、分析不同语言的设计风格和优缺点的书籍较少，这样的书籍提供程序设计语言方面广泛而深入的知识，有助于程序设计思想的形成和程序设计素养的培养，对选择合适的语言进行软件开发、学习新的程序设计语言和进一步研究语言都有好处，同时也是学习编译器设计的基础。但是，阅读这样的书籍需要至少实践过一门程序设计语言，需要具有一定的编程能力。

其实，程序设计理论与程序设计实践是无法分割的。单纯的程序设计理论显得空泛，而单纯的程序设计实践显得没有根源。正确的设计理念和良好的设计习惯在程序设计伊始就应注重培养并逐渐形成，不要在学习过、进行过编程之后才来规范编程行为。理论指导下的实践将有益于程序设计能力沿着正确、有效的途径发展和提高。为初涉编程的学习者提供理论指导是《程序设计基础》第2版希望尝试的，为此，我们在第1版的基础上付出了更多的努力。就这一点而言，第2版较第1版有非常大的改进，它更像是一本有关程序设计基础的书籍，而不仅仅是有关C语言的教材。

C语言是非常具有代表性的高级程序设计语言。通过C语言建立程序设计基础，至今仍是计算机科学教育中最重要的途径。借助于对C语言系统化的讲解，适时、适当地提升到一定的理论高度，有助于读者将理论和实践有效地结合起来，形成相得益彰的知识体系，建立朴素的软件工程思想，形成可持续发展的程序设计能力，正确处理程序设计中存在的问题。本书第1章在介绍程序设计语言的基本功能、实现方法、评价标准等基础上，介绍C语言的基本概念。第2章在介绍程序设计中数据的类型、值、运算等多方面属性的基础上，介绍C语言的各种运算符和表达式。第3章在介绍结构化程序设计概念和结构化设计优点的基础上，介绍C语言的控制流，包括选择结构和循环结构。第4章在介绍程序的模块化设计思想和设计原则的基础上介绍C语言的函数等。第5章在介绍程序设计中的数据结构（类型）的基础上，讨论C语言的自定义数据类型指针和数组。第6章进一步讨论C语言的数据类型，对各种数据类型特别是自定义数据类型进行有意义的比较和总结，介绍自定义数据类型结构和联合。第7章在介绍程序输入输出的作用、输入输出操作、输入输出和界面的关系以及界面的设计原则等基础上，讨论标准库输入和输出函数，关注应用标准库编写应用程序的方法。各章中有关程序设计理论方面的内容，在具体学习C语言之前或之后阅读都可以。之前读有助于学习，之后读有助于领会。无论之前还是之后，理解的程度可能有深有浅，但都会有收获。

本书保留了第1版的一些特点，以问题引导概念，由概念孕育解决方法，内容力求简明并深入

浅出，注重程序设计方法的讲解，注意避免繁杂的语法细节。示例程序完整、实用、有效、兼顾连续性，尽量避免为示范语法效应而引用缺少设计价值和应用价值的例子程序，保留了对标准库函数的简化和引用。我们相信示范的效应非常重要，对程序设计学习者起到潜移默化的作用，是对“言不尽意、意不尽理”的补充。优秀的库函数无疑是最好的程序设计典范。

本书保留并丰富了第 1 版中的图形和图表，特别为所有的例子附上了运行结果或运行样本。运行时输入的数据具有一定的代表性，带有一定的测试信息，在潜意识中培养设计调试和测试数据的正确的、趋向全面的思维方式。同时，分析运行结果将有助于对程序的理解。

遵循良好的编程习惯才能落实良好的程序设计风格，但相关的规则庞杂琐碎、巨细有别，难于自成体系。本书依托章节内容介绍，不脱离主体语言，使良好习惯言之有物，易于接受和实践。

本书设置常见编程错误的目的并不是简单罗列错误范例，所关注的是揭示错误的本质原因以期规避，并减少挫折感。我们认为向正确的方向引导应该是教学的主要途径，不断指出错误的教学方式显得曲折，收效甚微。所以，尽管我们在多年的教学中积累了许多常见于学生的错误，但多数并未列于本书中。

以上的努力出于我们对计算机科学教育中程序设计类课程教学目标的一种理解，即程序设计课程的设置并非为编程语言本身，而是以编程语言为教育工具，以培养软件开发素养和能力为目的。本书力求不脱离语言，但又不局限于语言，着手于编程，着眼于现代软件的开发。

本书由张杰敏主编。张杰敏编写了第 1～4 章、6.1 节和第 7 章。丁跃潮编写了第 5 章。王巍编写了第 6 章的其余部分，并编写了各章习题及附录。鄂大伟教授审阅了全书。感谢陈启安教授一直以来的关心和提出的宝贵建议。在此对付出心血的各位同仁表示诚挚敬意和感谢。

毕竟本书只是一种探索和尝试，书中定有疏漏之处，恳请读者雅正和指教。编者的联系方式为 zhang_jiemin@sina.com。

作 者

2009 年 1 月 于厦门

# 目　录

# 第1章

# 基本概念

## 学习目标

本章介绍程序设计的基本概念，讲解组成C语言的基本元素，包括数据类型、基本运算、变量和常量等，这些元素是构建C程序的基础。通过本章的学习，读者将能够：

- 了解程序设计的基本概念
- 认识C系统和C程序
- 定义和使用基本数据类型的变量
- 定义和使用符号常量
- 定义枚举类型，使用枚举常量
- 根据需要使用typedef语句进行类型再定义
- 实现编辑、编译和运行一个C程序的过程

## 1.1 关于程序设计语言

自然语言是人与人之间进行信息交流的最主要的方法，而程序设计语言是人与计算机之间进行信息交流的最主要的方法，是使用计算机的一种工具。程序设计语言规定了书写程序时可以使用的符号集合及其语法规则。人类通过程序指示计算机完成规定的任务。实际上，人机之间的程序设计语言远比人类之间的自然语言要简单并规范得多。

### 1.1.1 程序设计语言的基本功能

程序设计语言用于书写计算机源程序，通过程序控制计算机运行，达到使用计算机解决实

际问题的目的。

在计算机发展史上，程序设计语言出现早，种类繁多，但依然可概括出若干共同的特征。一般来说，语言包含5类成分：数据、计算、控制、传输和程序结构，分别用于描述程序中所涉及的数据、对数据的操作、程序的串行或者并行执行方式以及数据传输方法，程序结构将前面4类成分组织成源程序。

千差万别的各类程序设计语言是人们为了满足各种不同的目的而开发的，适合不同的计算机应用领域，如科学计算、商务应用、人工智能、系统程序设计、脚本开发、专用领域等。C语言最初用于系统程序设计，后来被广泛应用于多种领域。

按照程序设计语言对机器的依赖程度，可将其划分为机器语言、汇编语言和高级语言三大类。

机器语言是绝大多数计算机硬件可以理解的唯一语言，因而机器语言程序的执行速度最快。但由于机器语言每一条指令的功能都很微弱，所有的指令又均以0、1代码表示，因此用机器语言编写程序非常费力且枯燥、烦琐，程序不具有可读性，调试、修改都很困难。由于机器语言程序紧密依赖于具体的机器，程序不可移植。

汇编语言是机器语言的简记形式。它采用容易记忆的符号和标记表示机器语言的指令，使程序具备了一定的可读性，也变得较为容易调试和修改。但计算机硬件无法理解汇编语言，汇编语言程序需要经过“汇编”转换为等价的机器语言程序，才能在机器上执行。“汇编”通过调用称为“汇编器”的专用软件来完成。与机器语言相同，汇编语言也是面向机器的程序设计语言，对机器有很强的依赖性，程序不可移植。汇编语言程序的执行速度低于机器语言程序的执行速度。

高级语言使用人们容易理解的符号和单词组成语句，语句间依照规定的语法规则构建程序。由于更接近于自然语言，高级语言程序具有良好的可读性，易于维护。高级语言的每条语句都具有很强的功能，所构建的程序独立于机器，具有可移植性。勿庸置疑，使用高级语言编程，使软件的生产率、可靠性、简洁性和可维护性获得了重大突破，使开发大型系统成为可能。各种高级语言是各个应用领域最主要的编程工具，C语言是其中的一种。

当然，计算机硬件不理解高级语言，高级语言程序必须先经过转换才能在机器上实现。采用不同的转换方法，程序的实现方法就不同。

## 1.1.2 程序设计语言的实现方法

高级程序设计语言有3种基本的实现方法，分别是编译、解释和混合方法。高级程序设计语言可采用3种基本方法之中的任何一种实现在机器上的执行。

编译通过专用软件“编译器”完成。编译器将高级语言源程序翻译成在计算机上直接运行的机器语言目标程序。编译方法的优越性是，一旦完成翻译过程，所获得的目标程序可以脱离原高级语言环境独立存在和运行，程序执行速度非常快。但无论源程序发生任何变化，都必须重新编译源程序以获得新的目标程序，这使修改程序的过程显得有些烦琐。大多数程序设计语言都是用编译方法实际实现的，如C、COBOL、Ada语言等。

与编译不同的解释方法不需要任何翻译过程。由专用软件承当的“解释器”，一边将高

级语言源程序的语句解释为机器语言表示的目标代码，一边执行。这种边解释、边执行的方式类似于日常生活中的“同声翻译”。解释方法的优点是，容易定位源程序出错的位置，进而容易实现对源程序的动态调试和修改操作。但相对于编译实现的程序，解释实现的程序运行速度会慢10～100倍。程序的每次运行都需要解释，无法脱离解释器形成独立的可执行文件。结构比较简单的程序设计语言适合用解释方法实现，如APL和LISP语言常实现为单纯的解释系统。大多数操作系统命令是用解释器实现的，如DOS的.bat文件。JavaScript尽管不是特别简单的语言，却也是单纯地解释执行的。

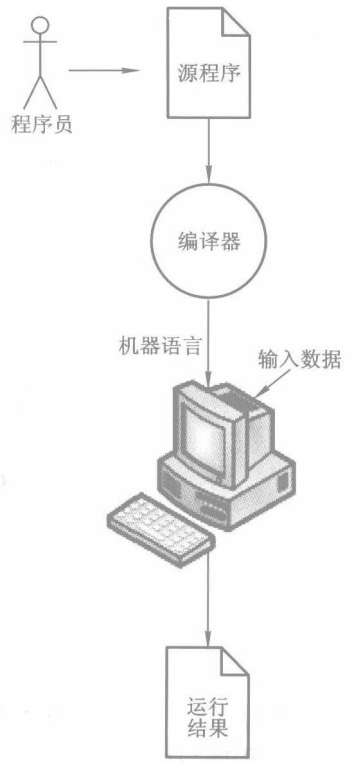


图1-1 编译方法

有一些程序设计语言的实现介于编译和解释之间，称其为混合实现方法。混合实现方法是将高级语言源程序翻译成一种易于解释的中间语言，再由解释器解释执行中间语言。如Java语言采用混合实现方法。Java源程序被编译成可独立存在的字节码文件，字节码文件是各平台通用的，不依赖于任何特定的机器或者操作系统，这一点非常重要。对于每个运行Java程序的平台，都有一个与之相应的字节码解释器，通称Java虚拟机（Java virtual machine，JVM），用于解释执行字节码文件。经过翻译获得的字节码文件，解释执行的速度相当快。同时，由于字节码文件的通用性，给装有Java运行时系统和JVM的机器提供了可移植性，使Java程序可以体现其最重要的特点“一次编写，到处运行”。

程序的各种实现方法如图1-1、图1-2和图1-3所示。

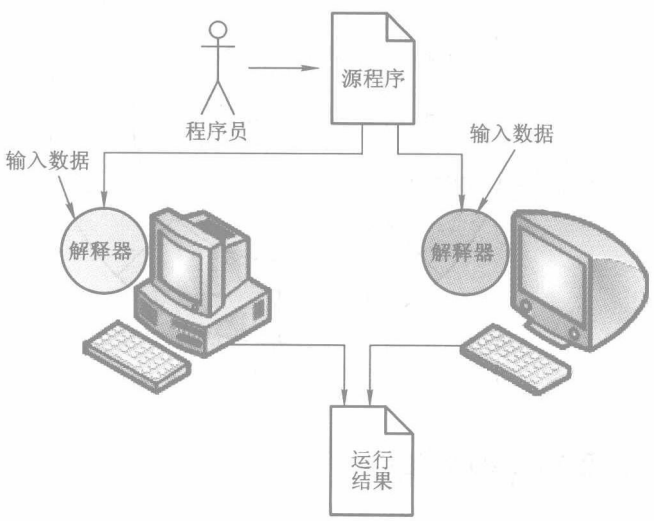


图1-2 解释方法

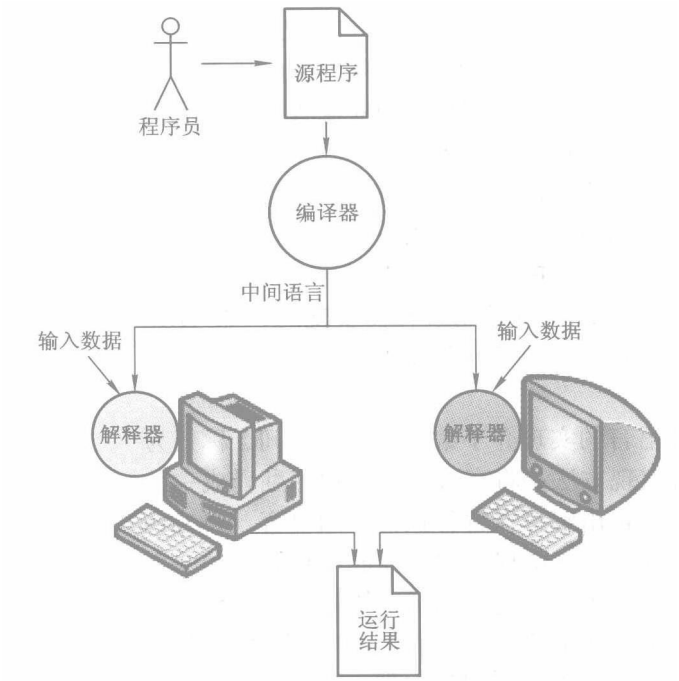



图 1-3 混合方法

事实上，可将计算机系统简单地看成是由硬件、操作系统和应用程序组成的分层结构。高级语言程序与编译器（解释器）、操作系统、计算机硬件之间的层次关系如图 1-4 所示。

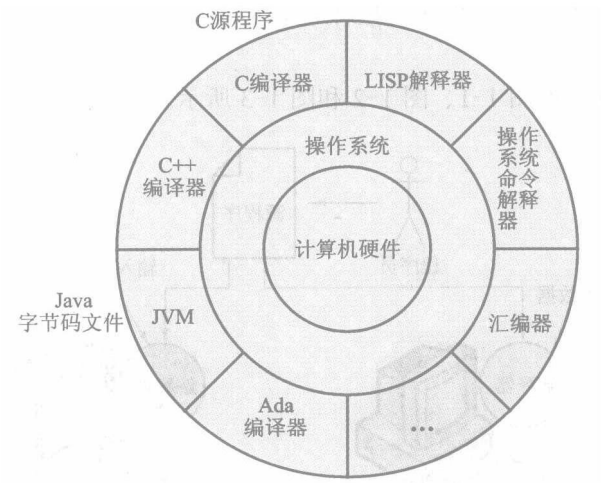



图 1-4 典型计算机系统的分层结构

## 1.1.3 程序设计语言的评价标准

对于程序设计语言，并不存在一整套无争议的评价标准。但判断程序设计语言优劣的一个最重要的标准是使用它所编写的程序的易读易懂的程度。

自从 20 世纪 70 年代软件生命周期概念（Booch，1987）被提出之后，编程就成为生命周期中的次要任务，而软件维护则是生命周期中时间最长、代价最高的一个阶段。由于软件维护的难易在很大程度上取决于程序的可读性，所以可读性就成了衡量程序以及程序设计语言质量的重要标志。

影响可读性的语言特征包括程序设计语言的语法设计、数据类型和控制结构等，它们是程序的基本组成部分。这些基本组成部分所体现出的整体简单程度极大地影响着所编写程序的可读程度。有少量基本组成的语言比有大量基本组成的语言更简单，也就更容易读。相对而言，C 语言的基本组成非常精简，具有良好的可读性。

除可读性，程序设计语言的可写性和可靠性也是影响其优劣的重要标准。

程序设计语言的可写性是指在给定应用领域用该程序设计语言开发程序的难易程度。大多数影响可读性的语言特征也影响着可写性，这是因为在编写程序的过程中，编程人员要不断阅读已编写了的部分程序。

如果一个程序在任何条件下的运行都能达到它所说明的标准，称这个程序是可靠的。可读性和可写性都会影响程序的可靠性。原因是特定的程序设计语言有特定的限制，影响着程序员编程时对问题的描述和处理。如果编写程序的语言不支持问题域所需表达的数据结构和算法，程序就会不自然且复杂，因而就会特别难读、难写，程序可靠性也会受到影响。越是容易编写的程序，其运行正确的可能性就越大，也就越可靠。

换句话说，不同的语言适用于不同的应用领域，例如，工程计算可选 FORTRAN、Pascal 等；商务应用可选 COBOL、SQL 等；人工智能可选 LISP、Prolog 等；脚本开发可选 JavaScript 等。对于程序设计而言，只有最适合的语言，没有最好的语言。选择适宜的程序设计语言非常重要，直接影响着程序设计的质量。

可读性、可写性和可靠性等评价标准是很难精确定义的，也无法精确测量，然而却是很有用的概念，提供了有关程序设计的非常有价值的观念。在选定程序设计语言后，尽力提高程序的可读性就成为编程中一项重要工作，它是一切程序质量标准的基础。

### 1.1.4 程序的效率、风格和可读性

程序的效率通过程序需要的处理时间和占用的存储空间衡量。对程序效率的追求，建立在不损害程序可读性和可靠性的基础上。当效率与可读性发生矛盾时，应选择以时间和空间为代价来换取程序的可读性。在现代程序设计中，可读性至关重要。

具备良好的程序设计风格最主要的作用是使代码更容易阅读，无论是对程序员本人还是其他人。好的风格对于好的程序设计起着至关重要的作用。有关程序设计风格的诸多原则均根源于由实际经验中得到的常识，它并不是随意的规则或者处方。如代码应当简单、清楚，具有直截了当的逻辑、自然的表达方式、通行的语言使用方式、有意义的名字和有帮助作用的注释，不使用非正规的程序结构，避免别出心裁，等等。如果大家都坚持同样的风格，就会发现无论是谁编写的代码，都能很容易实现相互间的交流，程序具有良好的可读性。

好风格应该成为一种习惯。基于原则的程序设计风格的规则或细节与具体使用的程序设计语言有一定的关系，在设计程序时遵循这些规则，就会形成良好的编程习惯。良好的编程习惯

有助于提高程序的可读性。有鉴于此，本书中的每一章都将针对所讲述的主要内容介绍一些良好的编程习惯。

良好的编程习惯是在程序设计中始终如一地注重细节的过程里慢慢修养成的。细节虽小，但程序设计初学者却不应忽略，细节决定成败。具备良好习惯的程序员即使在工作压力下所写出的代码也会很好。注重良好的编程习惯是一名优秀程序员应该具备的基本素质。

## 1.2 关于C语言

C语言是国际上广泛流行的、很有发展前途的计算机程序设计语言，它精巧灵活、功能齐全，既适合编写系统程序，又适合编写应用程序。历经多年，人们借助C语言所开发的各种软件广泛应用于图形、通信、工程、网络、数据库、CAD/CAM以及字处理等许多方面。随着计算机的迅速发展和深入应用，C语言在计算机软件开发中的作用日益重要。在具体学习C语言之前，有必要简单了解C语言的发展历程及其特点，并初步了解C系统的组成。

### 1.2.1 C语言的历史

C语言是在BCPL和B两种语言的基础上发展起来的。BCPL是由Martin Richards在1967年开发用来编写操作系统软件和编译器的语言。Ken Thompson借鉴了BCPL的许多特点开发了B语言，并于1970年在贝尔实验室的一台DEC PDP-7上实现了最早的UNIX操作系统版本。无论是BCPL还是B语言都是无类型的语言，即每个数据占用存储单元的一个“字”，如读取数据，需由程序员决定将这个数据当做整数还是实数使用。

1972年贝尔实验室的Dennis Ritchie在B语言的基础上开发了C语言，并在DEC PDP-11计算机上实现。C语言吸收了BCPL和B语言的许多概念，保持了精炼、接近硬件的优点，同时增加了数据类型和其他一些有用的语言特征。1973年Ken Thompson和Dennis Ritchie合作用C重写了UNIX，使C语言作为UNIX操作系统的开发语言而被广泛了解。如今，几乎所有新的大型操作系统都是由C或C++语言（C的扩展）编写的。

C不捆绑于任何特定的机器和系统，大多数计算机都可以运行C程序。在不同的计算机硬件平台上广泛使用着C语言及其各种改进版本，这些版本虽然相似，却并不兼容。编写适合在不同平台上运行的可移植C程序是件很麻烦的事情。各种版本需要有一个共同的参照标准。因此，1983年，美国国家标准协会（American National Standards Institute，ANSI）计算机和信息处理标准化委员会成立了X3J11技术委员会，目标是根据C语言问世以来各种版本对C的发展和扩充，产生“一个无二义性的、独立于机器的C语言定义”。其结果是1989年获得通过的ANSI C标准。1990年，国际标准化组织为推进世界范围内C的标准化，接受ANSI C为ISO C的标准，与ANSI合作出版了最终的C标准ANSI/ISO 9899:1990，成为目前被广泛使用的各种C语言版本的标准。本书描述的C语言基于ANSI C标准。

虽然基于同一标准，C语言的各种版本间依然略有差异，如用于微型机上的Microsoft C、Turbo C、Quick C和Borland C等的特点和规定并不完全相同，在使用时还是应该首先了解所

用计算机系统配置的具体 C 语言版本。

## 1.2.2 C 语言的特点

对 C 语言产生过深远影响的经典之作是 Brian Kernighan 和 Dennis Ritchie 的合著 *The C Programming Language*，1978 年由 Prentice Hall 出版，1988 年再版了他们按照 ANSI C 标准修订的 *The C Programming Language*。

Brian Kernighan 和 Dennis Ritchie 在 *The C Programming Language* 的第 1 版序中谈到，C 语言是一种通用的程序设计语言，它包含了紧凑的表达式、现代控制流和现代数据结构以及一个丰富的运算符集合。C 不是一种“很高级”的语言，也不“大”，不特定于某一个应用领域。但是 C 的限制少、通用性强，这使得它比一些被认为功能强大的语言更方便、效率更高。

C 语言是一种相对“低级的”语言，这种说法并无贬义，仅仅意味着 C 语言可以处理大多数计算机可以处理的对象，如字符、数字和地址。C 语言具有低级语言的功能，允许直接访问物理地址，能对位进行运算。同时，C 语言也具有高级语言的功能，有结构化控制流语句，便于写结构化程序。正是 C 语言的这种双重性，使 C 语言既可用于开发系统软件，也可用于开发应用软件。

C 语言不“大”，它不提供直接处理字符串、集合、列表和数组等复合对象的操作。C 语言没有内存分配和释放语句。C 语言本身不提供输入、输出工具，没有 READ 和 WRITE 语句，也没有内部文件访问方法，所有这些高级机制都需要通过显式调用的标准库函数实现，这使得 C 语言很灵活、简洁。

以上特点使得 C 语言是一种令人愉快的、高效的、多样化的语言，适合编写各种程序。它容易学习，并且随着开发经验的增加，它会变得更为好用。

## 1.2.3 C 系统

C 系统由 C 语言、预处理器、编译器、标准库和程序员使用的诸如编辑器和调试器等工具组成。

在不同的 C 系统环境中，编写和运行一个 C 程序的具体操作不同，但各种情况下的大体过程都是相同的。

- 用编辑器创建 C 程序文本。
- 成功地对程序文件进行编译。
- 装入并运行程序。
- 查看所产生的输出。

在初学 C 语言时，熟悉编写、编译和运行程序的过程是一个障碍，越过这个障碍，其他内容就比较容易学了。

编译程序时，系统依次调用预处理器、编译器和加载器。

在 C 程序中，以符号#开始的行称为预处理命令。这些预处理命令由 ANSI C 统一规定，但它不是 C 语言的组成部分，在编译之前需要先由预处理器对其进行预处理。编译器将对预处理之后的源程序进行编译，得到目标程序。加载器负责连接目标程序和标准库函数源代码等，创建可供执行的目标程序。

标准库包含很多有用的函数，如标准输入函数 scanf( )、输出函数 printf( )。库函数丰富了 C 语言的功能，增加了 C 语言的灵活性，但不是 C 语言的组成部分。在使用库函数前，要用预处理命令“# include”将含有所用函数相关信息的头文件包括到用户的源文件中。

有了这些知识，接下来就开始学习编写第一个简单而完整的 C 程序。

## 1.3 第一个程序

学习新的程序设计语言的最佳途径是编写程序，本书所编写的第一个程序的功能是在屏幕上打印字符串“hello, world”。

【例 1-1】在屏幕上输出字符串“hello, world”。

```
#include <stdio.h>

/* 第一个 C 程序 */
void main()
{
    printf("hello, world\n");
}
```

“#include <stdio.h>”是预处理命令，用于引入头文件 stdio.h，这个头文件是 C 系统提供的，含有关于标准库函数 printf( )的信息。

“/*第一个 C 程序 */”是注释行。C 程序中的注释用一对符号“/*”和“*/”括起，供阅读程序的人参考，目的是增加程序的可读性。注释对编译和运行不起作用，可以出现在程序的任何位置。

main( )表示主函数，程序要从这里开始执行。main( )函数中的语句用一对花括号{ }括起来，是 main( )函数的函数体。main( )前的修饰符 void 表示函数类型为空类型。

“{”表示 main( )函数体开始。

“printf("hello, world\n");”是 main( )函数的语句，main( )函数调用标准输出库函数 printf( )打印字符序列“hello, world”。“\n”的作用是在打印“hello, world”之后将光标移到下一行起始处。语句以分号结尾。

“}”表示 main( )函数体结束。

运行程序，将在屏幕上打印输出：

```
hello, world
```

光标停在下一行起始处。

通过这个简单的程序，可以初步了解到 C 语言的程序结构和书写特点。

C 程序由函数组成，组成 C 程序的函数以文件的形式存储。函数由函数名紧跟一对圆括号“( )”括起来的参数表组成，函数的语句用一对花括号“{ }”括起来，称为函数体。例 1-1 的 C 程序由 main( )函数组成，可将其存储的文件命名为 first.c。main( )函数不要求任何参数，故为空参数表。在例 1-1 中，main( )的函数体只包含一条语句：“printf("hello, world\n");”。main( )

是一个特殊的函数名，称为主函数。每一个程序都从主函数的起点开始执行。这意味着每一个C程序都必须有且仅有一个main( )函数。换句话说，C程序可简单地只由单一主函数main( )组成，也可以由main( )和若干其他函数共同组成。

C语言的函数间可以相互调用。被调用的函数可能是程序员自己编写的，也可能是由系统函数库提供的，但不能是main( )函数。当调用一个函数时，要给出被调用函数的名字和由圆括号括起的一组参数值，这些参数值由调用函数提供，对应被调用函数的参数表。在例1-1中，语句

```
printf("hello,  world\n");
```

表示main( )用参数值"hello,  world\n"来调用库函数printf( )。C语句以分号结尾。

在调用printf( )函数前，要用预处理命令

```
#include <stdio.h>
```

告诉编译器将文件<stdio.h>包含在本程序中。<stdio.h>文件含有与标准输入输出库函数相关的定义。预处理命令不属于C语句，由预处理器处理。

在C语言中，'\n'只表示一个字符，称为换行单字符。其中，\称为转义字符，起到转义后续字符的作用，如\将n转义为换新的一行。'\n'是非打印字符，用于控制打印格式，作用是在打印后将光标移到下一行的起始位置。

如果在字符串“hello,  world\n”中省略'\n'，那么输出打印后光标停在本行。printf( )函数不会自动换行，将上面的例子写成如下的形式，将产生相同的输出。

```
#include <stdio.h>

 void main()
{
   printf("hello,  ");
   printf("world");
   printf("\n");
}
```

类似'\n'的还有'\t'（表示横向制表符）、'\"'（表示双引号）等，均用来控制打印格式。

## 1.4 基本数据类型

程序描述的是数据和基于数据的操作。操作处理初始数据使数据发生变化，产生结果数据。因此，数据是程序的基础。数据不仅有值，而且有类型。数据的类型是指数据的内在表现形式。通俗地说，数据在加工计算中所表现出的特征称为数据类型。数据类型决定着数据的取值范围及可以对此类数据施行的运算。

例如，年龄一般用整数表示，称为整型。工资常用实数表示，称为实型。人的姓名具有文字的特征，常用字符串表示。两个人的年龄可以进行减法运算，以求年龄差。两个人的工资可以进行加法运算，以求工资总和。而对两个人的姓名进行加法或减法运算就显得没有意