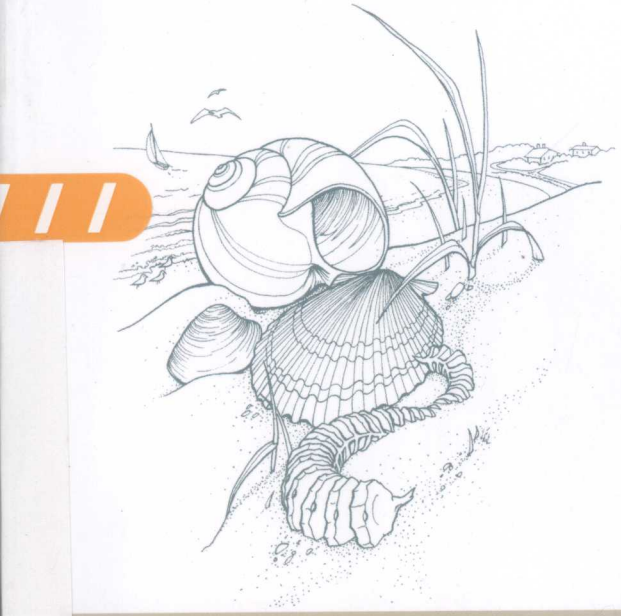
工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计

C How to Program

白忠建 杨剑 丁晓峰 编著

- 案例驱动展开知识点
- 遵循C语言最新标准
- 理论与实践紧密结合



高校系列

人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化普通高等教育“十二五”规划教材
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计

C How to Program

白忠建 杨剑 丁晓峰 编著

高校系列

人民邮电出版社
北京

图书在版编目（CIP）数据

C语言程序设计 / 白忠建，杨剑，丁晓峰编著. --
北京 : 人民邮电出版社，2013.9
21世纪高等学校计算机规划教材
ISBN 978-7-115-32571-6

Ⅰ. ①C… Ⅱ. ①白… ②杨… ③丁… Ⅲ. ①
C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第168190号

内 容 提 要

C 语言是世界上使用频度最高的计算机程序设计语言，是许多计算机专业人员编写应用程序和计算机的爱好者学习程序设计的首选。

本书通过一个贯穿全书的案例，逐步引出 C 语言的基础知识和应用方式，其中包括：C 语言的数据类型和运算、顺序结构、选择结构、循环结构、函数、数组、指针、结构体和文件。文中穿插了较多的示例程序，这些程序代码都经过 VC9 和 gcc 双重编译调试通过。

书中涉及的 C 语法完全符合 C99 标准。

本书可作为普通高校本、专科学生的教学用书，也可供一般工程技术人员参考使用。

◆ 编　　著　白忠建　杨　剑　丁晓峰
责任编辑　刘　博
责任印制　彭志环　焦志炜
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷
◆ 开本：787×1092　　1/16
印张：14.75　　2013 年 9 月第 1 版
字数：406 千字　　2013 年 9 月北京第 1 次印刷

定价：32.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第 0021 号

前　言

C 语言可谓是拥有显赫的身世：它诞生于拥有 7 位诺贝尔奖获得者、享有极高声誉的贝尔实验室。然而，炫目的光环并未使 C 语言仅仅局限于被少数精英程序员使用，相反，它以极强的亲和力迅速普及，受到无数程序员的喜爱。那些极具天赋的程序员们用 C 语言写出了海量的程序，为后辈学习者留下了宝贵的财富。

C 语言一诞生便显示出超群的能力，从大型复杂系统的编写，到直接与硬件打交道都能胜任。这样的气派，将它的前辈 FORTRAN 语言拍在了沙滩上，同时也使 PASCAL 语言边缘化（很长一段时间内 PASCAL 曾是众多程序设计初学者的首选学习语言），更将 BASIC 语言逼到了绝境。这些活生生的案例，无不昭示着 C 语言的强大。

然而，C 语言并非是那个一心想要一统江湖的魔教首领，它只是代表了程序设计艺术中一支广为接受的流派。如果说 PASCAL 是学院派的典型，那么 C 语言就是自由派的佼佼者。不过，所有的流派都各有千秋，作为程序设计艺术家的程序员们，只是根据不同的应用选择适合的语言，以便创造出杰出的作品。

从技术的角度来讲，C 语言属于面向过程的程序设计语言。这类语言在 20 世纪 60～80 年代可谓盛极一时，尤其是 C 语言，在整个 80 年代几乎一枝独秀。不过，当 C++、Java 这类更贴近自然思维过程、更适合大型程序设计、面向对象的程序设计语言如雨后春笋般涌现出来后，C 语言这种面向过程的语言受到了严峻的挑战。很多大型的应用都用面向对象语言来编写或者改写，C 语言的地位看上去岌岌可危。

然而，C 语言的应变能力使其一直保持着旺盛的生命力。虽然 C 语言已经较少涉足大型复杂应用程序的编写，但经过改造后，它依然在特种应用，比如在单片机、嵌入式等有苛刻条件要求的应用开发中，发挥着不可取代的作用。

时至今日，C 语言仍是世界上使用频度最高的程序设计语言，也仍然是很多程序设计初学者的首选学习语言。当然，学习的重点不再是强调 C 语言如何强大、如何灵活，而是把它作为一种与计算机沟通的媒介，学习如何用计算机技术来帮助人们解决实际问题；同时尝试接触和传播 IT 文化。其实，这正好体现了语言（包括人类使用的自然语言和计算机程序设计语言）的本质功能：交流和传承文化。

很多初学者曾问过编者：如何才能学好 C 语言？其实，任何语言的学习都没有捷径可走，唯一的“窍门”就是实践，大量的实践。所以，编者建议各位读者在学习过程中，要非常注重编程练习，除了做给出的习题外，更能在实际工作和生活中发现问题，并运用所学的知识加以解决。这才是我们编写应用程序的目的。

学好 C 语言不等于能够用好它。那么如何才能用好它呢？编者认为，在读者进行程序设计之前，一定要回答这样几个问题：

- 用户为什么需要这个程序？
- 用户要求程序做什么？

- 用户要求程序以何种方式呈现在他们眼前？
- 用户将会以什么样的方式使用程序？
- …

也许问题还不止这些。实际上，上述问题的解决是有方法论支持的。在《软件工程》这门课中可以找到问题的答案。

另一个常被问到的问题是：在学习 C++/Java/C#之前是否必须学习 C？诚然，它们都与 C 有着千丝万缕的联系，但这并不意味 C 是它们的前导课程。学习者最好把它们视为与 C 完全不同的语言去学习，因为它们承载着与 C 完全不同的编程思想。实际上，目前流行的很多程序设计语言，如 PHP、Objective-C 等都有着与 C 相似的语法。这里笔者强烈建议在学习时，要更多关注它们与 C 的不同之处。

为了使初学者能方便地阅读本教材，编者在正文中插入了一些额外的内容。

1. 解释性文字或扩展内容，它们类似于：

> ⓘ 计算机是一种电子设备……

框中的内容是对最近提及概念的解释，或者是对概念的扩展。

2. 概念引用记号，它们类似于：

语句（3.2）

其含义是“语句”这个概念出现在第 3 章第 2 节（记为 3.2）。

3. 超前引用概念总结，它们类似于：

> 前向引用概念：C 语句（3.2），顺序结构（3.3），选择结构（4.2），循环结构（5.2）

框中的内容是本小节正文中所有超前引用概念的集合。

4. 习题，它们类似于：

> 习题 1-1　请撰写一篇小论文介绍 C 语言的发展史。

习题嵌入在正文中，是希望学习者能够在完成这些习题后再进行后续内容的学习。

本书的作者白忠建、杨剑和丁晓峰同在电子科技大学成都学院从事教学工作。其中，杨剑编写了第 3 章、第 7 章，丁晓峰负责编写了第 4 章、第 10 章，其余章节均由白忠建负责编写。

感谢罗佳和杨菊英老师授权使用她们的工作成果。

C 语言博大精深，书中内容难免挂一漏万，可能还有错误，请读者们不吝赐教。如有高见，请发邮件至：baizj@uestc.edu.cn。万分感谢！

编　者

2013 年夏

目 录

第1章 引论

学习要求

- 掌握程序设计的概念
- 掌握C语言程序的框架
- 了解程序设计语言的种类
- 了解数据结构和算法的概念

C语言是一门应用非常广泛的语言，拥有简洁、灵活、表达能力超强的特点，并且现有的很多种软硬平台都支持C语言程序的编写与运行，所以C语言往往是初学者学习的第一门程序设计语言。

本章首先介绍程序语言和程序设计的概念，以便使初学者建立起最基本的概念，同时简单介绍了与程序设计密切相关的软硬件知识，旨在为今后的学习打下基础。

本章还引入了一些简单的C语言程序来展示C语言的特点，最后引入一个贯穿于整篇教材的待解决的案例。从这个实例出发，在后续的章节中将会逐步引入C语言的各项知识点，最后用案例的完整实现来对学习过程做一个总结。

1.1 程序设计概述

现代人的生活已经离不开计算机了。每一天人们都会使用计算机完成各种各样的工作，例如使用浏览器浏览网页，用电子邮件联系朋友或处理业务，用字处理软件来编辑文档等。可以说，计算机已经不仅仅是以前那种只用于科学计算的大型计算器了，而是能处理多种复杂信息的综合设备。总之，与几十年前相比，计算一词的概念已经发生了相当大的变化。

> 目前，一般意义上的计算（computing）可以定义为任何从计算机获益的面向目标，或者制造计算机的行为。因此，计算包括：设计和制造应用范围广泛的软硬件系统；处理、组织和管理各种各样的信息；利用计算机作科学研究；让计算机更加智能化；创建和使用通信和娱乐媒体；查找与各种特殊目的相关的各种信息，等等。这个列表可以很长，有很多种可能性。

以上种种实例似乎都在证明：计算机已经变得越来越强大，智能化程度也越来越高，使用者只需简单地输入网址，或者用鼠标单击一个按钮就可以将工作轻松搞定。然而，事情并非如此简单。这些方便都是各种各样的软件应用带来的。而在这些软件背后，蕴藏着大量程序员的辛苦工作。

程序员是这样一群人：他们使用各种各样的程序设计语言来编写简单或复杂的应用程序，而这些应用程序能完成某些特定的功能，为特定的人群提供服务。那么，什么又是程序设计？程序设计的步骤是怎样的？成为一个程序员需要具备哪些知识？

程序设计当然离不开程序设计语言，读者正在学习的 C 语言就是其中一种。不过，在学习 C 语言的语法之前，还是从程序设计的概念开始吧。

1.1.1 为什么需要程序设计

人类在面临一个待解决的问题的时候，会根据实际情况制定出一套可行的解决方案，而这个方案往往基于一个或多个数学模型。读者都知道，数学是一门精确的学科，它解决问题的方式就是将问题分解为一系列的子问题，再一步一步地将子问题分解为更细的、可行的解题步骤。当每一步的关键问题都被正确解决后，那么最终的问题也就迎刃而解了。

解题过程往往含有多个步骤，其中每一步都会用一条或多条确定的指令来实现。这里用指令这个术语来描述一项最基本的操作，比如 $y=2x$。这些指令之间是有逻辑关系的。严密的逻辑关系在时间和空间上限定了指令执行的顺序，从而避免了执行过程中的顺序错误。

在求解数学问题的时候，都是严格按照上面描述的过程进行的。现在通过一个实例来重温一下这个过程。

【例】试求出一元二次方程 $x^2-3x+2=0$ 的实数根。

【解决方案一】

一般一元二次方程的基本形式是 $ax^2+bx+c=0$，它的通解为：

$$x=\frac{b\pm\sqrt{b^2-4ac}}{2a}$$

根据以上规则，可以确立以下的解题步骤：

（1）确定未知数 x 不同次数的系数。这里 $a=1,b=-3,c=2$；

（2）计算判别式 $\Delta=b^2-4ac$ 的值。

（3）验证 Δ 的符号。带入值后得到 $\Delta=1>0$，可知方程有两个不同的实根；

（4）求第一个根：$x_1=(3+1)/2=2$；

（5）求第二个根：$x_2=(3-1)/2=1$；

（6）测试。将两个根带入原方程，等式成立；

（7）求解完毕。

【解决方案二】

（1）首先将方程分解为两个一次运算式的积：$(x-2)(x-1)=0$；

（2）根据公理，两个数的积等于零，那么这两个数里一定有一个为 0，或者两者都为 0。因此，我们可以推断，下面两个等式都可能成立（也许不同时）：

$x-2=0$

$x-1=0$

（3）求解上面的两个线性方程，得到 $x_1=2$，$x_2=1$；

（4）测试，将两个根带入原方程，等式成立；

（5）求解完毕。

可以看到，无论以上哪种解决方案，都包含了相应的数学运算，此外，非常重要地，还包含了用以承上启下的逻辑，它们用一种严密的方式连接了运算的每一步。这里的逻辑指的是一种秩序，指示了运算的先后顺序。读者都知道秩序对于一个文明社会来说多么重要，对于一个再简单不过的问题也都要按顺序解决。而且，一些关键顺序是不可以缺省或者交换的。

上面讲述的是用手工计算的过程。而当要把解决方案移植到计算机上面时，一个问题就横在了面前：如何让计算机理解人们的意图呢？虽然计算机看起来已经很“聪明”，但遗憾的是，目前的计算机技术还不能使计算机直接理解人们用自然语言描述的问题。也就是说，如果计算机内部

没有人们预先准备好的软件支持系统，它是无法理解人们提出的问题的，更谈不上提出问题解决方案了。因此，必须将解决方案转换成计算机能识别的方式。这个转换过程就是**程序设计**（programming），也就是将方案的每一步用某种程序设计语言的指令来描述。一般地，称一条计算机能够识别和完成的运算为**指令**（instruction）。那么，一段按一定的逻辑顺序组成的并且能够完成一定功能的指令序列就是一段**程序**（program）。

计算机是一种功能强大的工具，它可以用比人类思维更快、更精密、更有效的方式解决问题。通过程序设计，程序员可以编写复杂的**应用程序**（application），让计算机代替人类完成手工作业，甚至完成人类无法完成的工作。

程序设计不仅仅只是一个简单的转换过程，它由 3 个重要的设计阶段组成。

1. 问题求解阶段

（1）问题分析。正确理解问题，明白问题要求做什么。

（2）确立求解方案。在计算机术语中，细化的求解方案称为算法（algorithm）。简单地说，就是解决问题的逻辑步骤。

> 算法较精确的定义是：对待定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令都表示一个或多个操作。

（3）验证。严格按照解题步骤，看看算法是否能真正地解决问题。

2. 实现阶段

（1）程序设计。将算法转换成计算机能够识别的指令，即转换成某种计算机**程序设计语言**（programming language）编写的程序。这个过程称为算法**编码**（coding）。

（2）测试（testing）。验证程序运行的结果是否达到要求。如果没有，仔细分析算法，追踪算法的每一步，看看问题出在哪个环节，然后修改程序，直到结果达到预期要求。用计算机运行程序并得到结果的过程称之为执行（running/executing）。发现并纠正算法或程序中错误的过程称之为纠错（debugging）。

3. 维护阶段

（1）使用程序。将程序交给用户使用，等待用户的反馈。

（2）维护。搜集用户的反馈意见，修改程序以使它能够符合变化的需求，或者纠正在测试阶段没有发现的错误。

以上三个步骤合称为程序的**生命周期**（life cycle）。

在以上的步骤中，确立算法是最耗时也是最为关键的一步。但需要提到的是，虽然算法如此重要，但解决问题的算法并非总是唯一的。就像人们在真实世界中一样，存在一题多解。而在这些解决方案中，有“笨”的、低效的，也有“聪明”的、高效的。但无论如何，只要没有特别的条件限制，比如时效限制，那么就可以说，在用户能够容忍的情况下，只要是能正确解决问题的算法就是好算法，没有必要为了追求效率而用到太多的技巧。要知道，技巧虽好，却使算法难以维护，程序也难以阅读。

> **专家语录：**程序首先是给人阅读的，其次才是让计算机执行的。

另外需要提到的是，算法是可以借鉴的。在一个问题的解决中用到的算法在其他问题的解决中也可能被用到（可能会有一些修改）。所以，程序员在设计算法的时候往往会考虑它的可重用性。

一个程序在运行时会使用很多的**数据**（data），而这些数据组织的好坏也将直接影响程序的设计。计算机存储、组织数据的方式称为**数据结构**（data structure），并且其设计要与算法设计同

时进行，二者相互依赖、共生共荣。设计良好的数据结构是算法设计的基础，能有效提高算法的效率。但如何有效地设计数据结构也是一件令人头疼的事情。“数据结构”这门课将会覆盖上述内容。

1.1.2 什么是程序设计语言

前面提到过，计算机程序设计需要使用某种程序设计语言。那么，什么是程序设计语言呢？既然提到语言，那就先从人类的语言谈起。

人类使用的语言一般统称为**自然语言**（natural language），是人们进行沟通、互相传递信息的重要载体。纵观自然语言，虽然数量众多，但它们都至少有以下两点语法规则。

- 词法（morphology）。利用词法规则可以构造出承载信息片段的最小单位：词汇。除了在特定的上下文中，单个词汇或词汇的组合表达的信息是不完整的。每一种语言都有一张词汇表，而表中的条目可能会很多。例如英语，它的词汇量在百万以上。除了这些固定的词汇外，语言的词法规则允许人们创造新词汇，但只有那些被广泛认可的新词汇才会被融入到语言体系中。因此，一种语言的新增词汇总是非常少的。
- 句法（syntax）。利用句法规则，使用合理的词汇可以构成能表达较完整信息的单位：语句。当更多的语句按照规则组成一个序列的时候，信息的表达将会更加完整。在书面表达中，词法和句法规则的执行都是相当严格的，因此承载的信息表达也相对精确。相对而言，口头表达的信息的结构要松散得多。

无论语言的语法规则多么复杂，它们都是为了一个主要目的而存在的，这就是人与人之间的沟通。

相比之下，人在使用计算机的时候，实际上就是人与计算机之间在进行沟通。显然，这需要某种信息载体作为沟通媒介。从某种角度来说，**程序设计语言**就是担当沟通任务的媒介之一。人们利用程序设计语言编写一段程序，“告诉”计算机要做什么、应该怎么做、要得到什么结果，那么计算机就会按照人的意图去完成相应的工作。而众所周知地，计算机完成这些工作也许比人类更有效率。

与自然语言一样，一种程序设计语言也有词法和句法等语法规则。不过不同的是，所有程序设计语言都比较**形式化**（formalized），也就是说，人们必须严格遵循语法规则来编写程序，否则将会带来错误。另一个显而易见的不同是，程序设计语言保留的固有词汇表一般都很小，而更多的词汇是程序员创造的。当然，这些新造的词汇必须遵循词法规则，并且是容易被理解和使用的。

1.1.3 程序设计语言的多样性

自然语言的种类非常多，比如汉语、英语、德语等，每种不同的语言都有不尽相同的词法、句法规则。这使得自然语言体系显得非常复杂，从而使沟通也变得艰难。

同样地，在计算机世界中，也存在着语言多样性，有的简单直接，有的则晦涩难懂。由于计算机构造的特殊性，它唯一能“懂”并能直接执行的语言是最原始的、用建立在计算机内部的指令体系构成的语言：**机器语言**（machine language），也常被称为**低级语言**（low-level language）。

机器语言是一种由二进制（binary）代码构成的指令体系，该体系使用一组二进制数来表示一条指令。例如：加法指令的机器语言形式是 100101，而减法的是 010011。

> 计算机是一种电子设备。简单地说，它进行的运算都基于电平的高低这两种状态。因此，二进制体系是计算机的唯一选择。自从冯·诺依曼先生和艾伦·图灵先生提出现代计算机的结构蓝图后，这种结构就一直应用在主流计算机系统结构上，并且至今没有改变过。

不难想象，直接用机器语言编写程序是一件多么痛苦的事情。所以，为了减轻程序员的负担，计算机科学家们发明了**汇编语言**（assembly language）。汇编语言实际上是一种**助忆**（mnemonic）**体系**，使用一些简单的（缩写）文字符号来代替二进制机器指令，例如：ADD 表示加法，MOV 表示数据传送等，这些符号与二进制指令一一对应。只要经过简单的转换（这个过程称为**汇编** assembling），汇编语言程序就能变成一系列机器指令。可以说，汇编语言编写的程序在执行效率上是非常高的，仅次于机器语言。

然而，这并没有从根本上减轻程序员的负担。汇编语言虽然高效，但其简单的语法使其在编写复杂程序的时候显得非常笨拙，代码维护的成本也相当高。因此，为适应复杂应用的编写，**高级**（high-level）程序设计语言应运而生了。高级程序设计语言的特点是比较接近于自然语言，从而容易记忆和书写，特别是编写复杂应用程序。

计算机的发展史上出现了很多的高级程序设计语言，以下是一些典型的例子（以首字母排序）。

- Ada
- BASIC
- C
- C++
- C#
- FORTRAN
- Java
- Pascal

每一种语言都有各自鲜明的特点。也正是由于这些特点，让这些语言可以适应不同的需求。例如，拥有强劲计算能力的 FORTRAN 语言适用于需要大量科学计算的场合，严谨、良好的结构使 Pascal 语言应用在早期的教学环境中，灵活的表达能力和很强的适应能力使 C 语言成为编写大型复杂系统的首选语言，而现代需要充分可重用、可移植、可快速生成应用的场合使程序员聚焦到 C++、Java、C#等面向对象的程序设计语言上。

随着时代的发展，新的高级程序设计语言被不断地研发出来，例如 Objective-C、PHP、Python 等。它们都拥有一些卓越的特性，使得它们在编写特定应用的时候显得更加得心应手。

习题 1-1　请撰写一篇小论文介绍 C 语言的发展史。

习题 1-2　请撰写一篇小论文介绍高级程序设计语言的发展史。

习题 1-3　请撰写一篇小论文介绍高级程序设计语言的分类情况。

1.1.4　高级程序设计语言的基本结构

从本质上来说，每一种高级语言都有高度相似的语法结构，特别是**控制结构**（control structure），它们用于控制算法的流程。以下是三种典型的控制结构。

- 顺序（sequence）结构（3.3）：由一组顺序执行的语句序列构成。
- 选择（selection）结构（4.2）：由一个测试部分和两个或多个分支构成。根据测试的结果，程序的流程会向不同的分支转移。
- 循环（iteration）结构（5.2）：由一组可以重复执行的语句序列构成。通过一个测试决定是否终止循环。

图 1-1 形象地说明了以上 3 种结构的执行流程。

这三种结构在一个程序中可以重复嵌套出现，构成了复杂的程序流程，在以后的章节里会详细地学习。

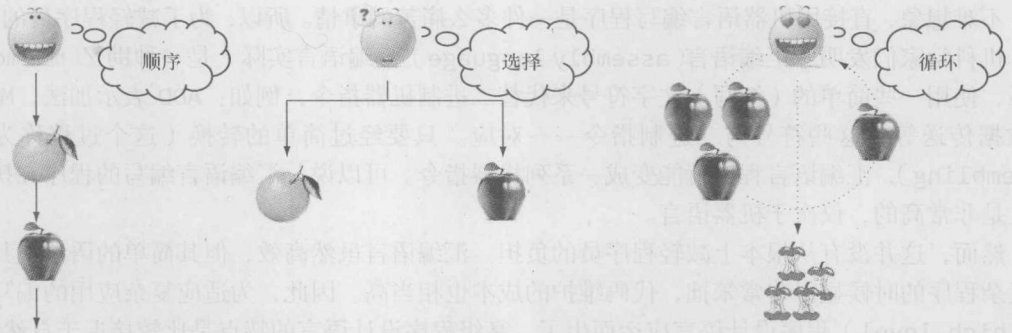


图 1-1　3 种程序控制结构

> **前向引用概念：** C 语句（3.2），顺序结构（3.3），选择结构（4.2），循环结构（5.2）前向引用（Forwards）概念是指那些在后面的章节中会出现而在这里超前引用的概念。

1.1.5　高级程序设计语言的开发过程

虽然高级语言拥有比低级语言更为优秀的表达能力，但一个致命的问题是：既然计算机只懂机器语言，那么怎么才能让计算机也懂得高级程序设计语言呢？很明显，必须引入一种翻译系统作为二者之间的桥梁。这种翻译功能往往由一个称为**编译器**（compiler）的系统程序完成。编译器能将高级程序设计语言写成的程序翻译成机器语言指令。图 1-2 示意了这种过程。

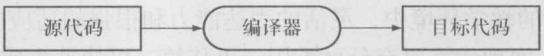


图 1-2　编译器将源代码翻译成目标代码

一个高级语言程序被称为**源代码**（source code），是编译器的输入数据。编译器的输出数据是机器语言代码，称为**目标代码**（object code）。目标代码中还可能包含了编译器加入的其他有用信息，比如调试信息。如果源代码中包含错误（称为编译时错误 compiling error，一般都为语法错误），那么编译器不会输出目标代码，取而代之的是一些出错信息。

> 编译器只能检测出源代码中包含的语法错误，而对隐含的逻辑错误和其他的错误就无能为力了。这类错误一般在程序运行时才能发现，因此称为运行时错误（run-time error）。

经过编译的目标代码还不能直接运行，它还必须经过**链接器**（linker）处理。链接器就像生产线上最终的装配器，将一个应用程序需要的所有目标代码以及运行时所必要的资源装配在一起，以生成**可执行**（executable）**代码**。图 1-3 示意了链接过程。

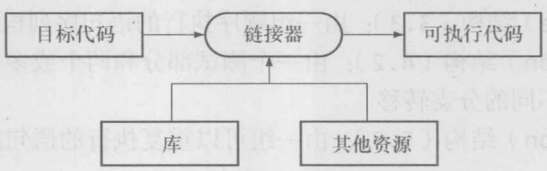


图 1-3　链接器将目标代码装配成可执行代码

可执行代码是以二进制文件的形式存放在计算机的**存储设备**（storage，多数情况下是硬盘）中的。在执行的时候，计算机首先将可执行代码**装入**（loading）内存，然后期待用户输入数据

（如果有的话），并根据程序员设定的逻辑得到结果。图 1-4 示意了这种过程。

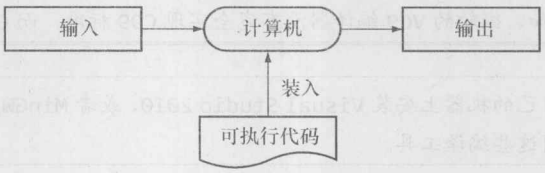


图 1-4　计算机执行代码的过程

链接器可以进一步检测到在编译环节中没有发现的错误，但还是无法检测出隐含的**逻辑错误**（logical error）以及**运行时错误**（run-time error）。如果出现了这种情况，就需要程序员在程序运行时通过对结果进行分析来发现那些逻辑上的疏漏。这是一项很细致且费神的工作，从而催生了**软件测试**（software testing）这一行业。

综上所述，使用一个高级程序语言开发程序需要经过多个阶段，每个阶段的详细步骤如图 1-5 所示。

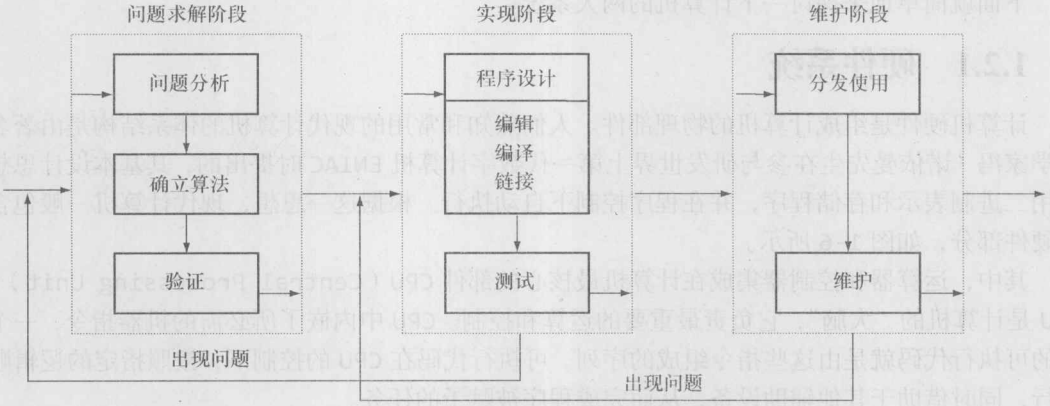


图 1-5　高级语言程序的开发流程

1.1.6　高级程序设计语言的标准化

在不同的应用环境中，人们使用了多种不同架构的计算机，例如处理日常小规模事务的微型计算机、处理大规模事务的 IBM 的大型主机，这些计算机拥有不同的机器指令系统。此外，不同结构的操作环境，例如常用的 Windows 操作系统和 Unix 操作系统（包括类 Unix 的 Linux），又拥有不同的信息组织的方式。因此，针对上述不同，同一种语言的编译器在这些机器上是不同的。而这些不同有时是非常令人头痛的，因为它们造成了用同一种语言写成的、完成相同任务的程序在不同的软硬件环境中可能会有少许的不同。这无疑增加了维护程序的成本，而程序员的期望是程序能够不加修改地在不同的环境中运行，也就是可移植的（portable）。

语言的标准化（standardization）可以在很大程度上解决可移植性问题。国际标准化组织 ISO 会适时颁布或者修订一些国际标准，程序设计语言也不例外。几乎每一种高级程序设计语言都有属于自己的 ISO 标准，例如，读者正在学习的 C 语言，它最近的 ISO 标准是 ISO/IEC 9899:1999，俗称 C99。因此，遵循标准的高级语言程序可以在任何一台计算机上被同样遵循标准的编译器编译。虽然在不同的软硬环境中，相同的源代码极可能会生成不同的目标代码和可执行代码，但可以肯定的是，如果给定相同的输入数据，那么一定会得到相同的运行结果。

遗憾的是，由于 C99 的新增特性与它的后继语言 C++不兼容，因此不是所有的主流编译器都支持这个标准。例如，微软的 VC9 编译器没有完全实现 C99 标准，而 GNU gcc 则几乎完全实现。

习题 1-4　请在你自己的机器上安装 Visual Studio 2010，或者 MinGWgcc，或者 gcc for Linux，然后试着使用这些编译工具。

1.2 计算机系统

程序员编写的程序都运行在计算机之上。这里所说的计算机，其实是计算机系统（computer system）的简称。一个计算机系统主要包括以下两大部分。

- 硬件系统（hardware system）
- 软件系统（software system）

下面就简单地来探讨一下计算机的两大系统。

1.2.1 硬件系统

计算机硬件是组成计算机的物理部件。人们熟知和常用的现代计算机的体系结构是由著名的数学家冯·诺依曼先生在参与研发世界上第一代数字计算机 ENIAC 时提出的，其基本设计思想是采用二进制表示和存储程序，并在程序控制下自动执行。根据这一思想，现代计算机一般包含五大硬件部分，如图 1-6 所示。

其中，运算器和控制器集成在计算机最核心的部件 CPU（Central Processing Unit）中。CPU 是计算机的“大脑”，它负责最重要的运算和控制。CPU 中内嵌了所必需的机器指令，一个程序的可执行代码就是由这些指令组成的序列。可执行代码在 CPU 的控制下，按照指定的逻辑顺序执行，同时借助于其他辅助设备，从而完成程序被赋予的任务。

存储器（memory）一般特指内存储器（简称内存）。用简单的语言来描述，内存就是一个大量的以线性方式安排的存储单元的集合。内存的最小存储单元由 8 个二进制位（bit/比特，缩写为 b）组成，称为一个**字节**（byte，缩写为 B）。每一个字节单元用来存储二进制形式的指令和数据，同时被赋予了一个唯一的物理**地址**（address）以便访问。简单来说，实际上这些物理地址就是字节单元的顺序编号。图 1-7 是物理内存的简单示意，其中每一个小格代表一个字节，右面的数是该字节的编号（即物理地址）。

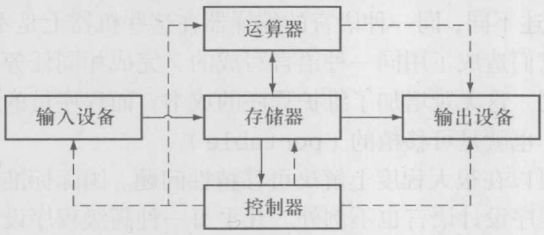


图 1-6　计算机的五大部件

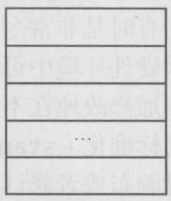


图 1-7　内存的简单示意

最后一个字节的地址值要视计算机而定。在 32 位计算机中，n 的值是 $2^{32}-1$，那么该计算机的内存容量是 4GB（Giga Bytes）。