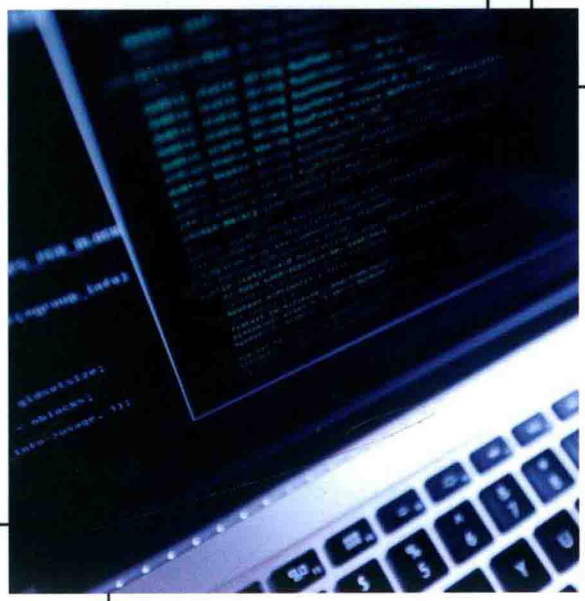
国家新闻出版改革发展项目库入库项目
高等院校计算机类规划教材
河北省高等教育教学改革研究与实践项目成果
全国高等院校计算机基础教育研究会立项项目成果



计算机软件基础

秦金磊 李 整 编著

北京邮电大学出版社
www.buptpress.com

国家新闻出版改革发展项目库入库项目
高等院校计算机类规划教材
河北省高等教育教学改革研究与实践项目成果
全国高等院校计算机基础教育研究会立项项目成果

计算机软件基础

秦金磊　李　整　编著

北京邮电大学出版社
www.buptpress.com

内 容 简 介

本书是计算机软件技术基础的综合教材，共分为7章，包括软件与程序、算法与数据结构、线性与非线性结构、内排序、软件开发与维护、软件测试、自动化测试及应用等软件基础核心技术。各章节内容清晰，图文并茂，所展现的程序代码及算法实现均经过运行验证。作者对书中特定内容录制了相应视频，便于学生全方位掌握知识。各章精心选取了阅读材料，这些阅读材料知识性与趣味性并重，可为开展课程思政提供参考。同时，各章配备了适量习题，以便于学生巩固章节知识。

本书可作为高等院校理工科计算机、自动化、电气与电子类等相关专业的本科、成人高等教育或大专层次的教材，同时本书对研究生和从事软件开发及测试相关工程技术的人员也是一本很好的参考书。

图书在版编目（CIP）数据

计算机软件基础 / 秦金磊，李整编著. -- 北京：北京邮电大学出版社，2022.8

ISBN 978-7-5635-6742-3

Ⅰ. ①计… Ⅱ. ①秦… ②李… Ⅲ. ①软件－教材 Ⅳ. ①TP31

中国版本图书馆 CIP 数据核字(2022)第 151943 号

策划编辑：马晓仟　**责任编辑**：马晓仟　**责任校对**：张会良　**封面设计**：七星博纳

出版发行：北京邮电大学出版社

社　　址：北京市海淀区西土城路 10 号

邮政编码：100876

发 行 部：电话：010-62282185　传真：010-62283578

E-mail：publish@bupt.edu.cn

经　　销：各地新华书店

印　　刷：唐山玺诚印务有限公司

开　　本：787 mm×1 092 mm　1/16

印　　张：16.25

字　　数：425 千字

版　　次：2022 年 8 月第 1 版

印　　次：2022 年 8 月第 1 次印刷

ISBN 978-7-5635-6742-3　　定价：42.00 元

前　言

为适应计算机软件技术更新快、内容丰富的特点，本书以软件开发过程中的核心内容为抓手，凝练了学生必须掌握的基本理论及开发方法。“计算机软件技术基础”课程的教学目标是为学生奠定良好的软件开发基础，从而使其具备进行各类应用程序设计与开发的能力。作者通过总结多年的教学科研及实践经验，结合计算机技术的最新发展并对课程相关资料进行综合分析提炼，编写了本书。

本书在选取与组织内容方面有所突破，以软件开发中的必备知识为主线，由浅入深，从软件与程序的基本概念出发，涉及算法分析与设计、线性与非线性结构、内排序等核心基础知识，并由此过渡到软件开发与维护的一般方法。特别是针对软件测试与质量保证，本书在把内容讲述清楚的前提下，结合自动化测试工具的使用，使得相关概念更加具体形象。

本书经过精心规划，具有以下特色。

(1) 注重基础讲解。以软件开发过程中所必须掌握的基本原理、技术、方法和工具为核心，循序渐进，图文并茂，把基础讲透，把重难点讲清。

(2) 知识性与趣味性并重。精心选取的阅读材料不仅能够增加各章知识的广度，还可激发学生的学习兴趣。同时，材料内容还可为开展课程思政提供参考。

(3) 案例资源丰富。本书提供了翔实的案例，这些案例的代码均经过运行调试，以保证正确无误。需要用到本书源代码的读者可从北京邮电大学出版社官方网站上直接下载使用。编者对书中特定内容录制了视频，以便学生全方位地掌握所学知识。

本书由秦金磊、李整共同编著，电子资源的整理及视频录制等工作由秦金磊完成。全书由秦金磊统稿并最终定稿。本书定稿后，由鲁斌教授主审。

本书的编写得到了河北省高等教育教学改革研究与实践项目(编号：2021GJJG411)、华北电力大学2021年校级教学改革与研究项目的支持；得到了华北电力大学负责计算机

专业平台建设的领导的支持；得到了华北电力大学计算机软件教学团队全体老师的支持；得到了全国高等院校计算机基础教育研究会和北京邮电大学出版社的支持。在此，全体编著人员向所有对本书的编写、出版等工作给予支持的单位和领导表示真诚的感谢！

由于作者水平有限，书中难免有错误和不妥之处，敬请广大同人和读者提出宝贵意见。

作　者

2022 年 3 月于华北电力大学

目　录

第1章　软件与程序

1.1　软件的分类及特性

1.1.1　软件的分类

软件内容丰富、种类繁多，传统上根据软件用途可将其分为系统软件和应用软件两大类。系统软件用于管理、控制和维护计算机的系统资源，而应用软件侧重于解决某一应用领域的具体问题。目前，随着整个社会信息化进程的不断加快，系统软件和应用软件的区分界线正在逐步模糊。例如，数据库管理软件及其服务程序，在数据库系统早期仅用于数据处理领域，应看成应用软件；而随着科学计算、工程控制、专业管理等新兴领域中应用软件的出现，现在则被视为系统软件。

应用软件的使用者通常是最终用户，一般不需要编制程序即可利用应用软件解决自己的问题。最终用户只需要进行使用培训，就可以正常使用软件，而不需要做软件技术知识的相关培训。有时针对特殊应用场合，在已有的应用软件的基础上进一步编写新的程序，称为二次开发。例如，利用 Visio 的绘图功能，编制基于故障树的可靠性评估软件；再如，VBA（Visual Basic for Applications）作为一种内置在 Excel 中的编程语言，可以通过二次开发增加新的功能。

随着计算机技术的不断发展，软件应用领域也不断扩充。根据其应用领域的不同，软件可分为以下几类。

1. 操作系统

操作系统（OS，Operating System）是直接运行在裸机上的最基本的系统软件，它管理着计算机系统的软/硬件资源（如 CPU、内存储器、硬盘等设备和各种服务程序）并向上层软件提供服务，任何非系统软件必须在操作系统的支持下才能运行。操作系统与计算机硬件系统密切相关，通常情况下某一种操作系统只能运行在某一类硬件架构上。当然，同类硬件架构上也可以运行不同的操作系统。如运行在 Intel 平台上的操作系统有 Windows、OS/2、NetWare、Linux、SCO Unix 等，运行于苹果计算机上的 Mac OS。也有可运行于多种硬件平台上的各种 Unix 操作系统，如 SUN 公司的 Solaris、IBM 公司的 AIX、我国独立开发的 COSIX 等。随着智能手机的广泛使用，运行于智能手机的 Android、iOS、Symbian、Windows Phone 和 BlackBerry OS 等手机操作系统，可显示与计算机相同的网页，具有良好的用户界面和很强的扩展性。从事计算机开发的人员应当掌握操作系统的基本理论和基础知识。

2. 办公软件套件

办公软件套件是指日常办公所用到的一系列软件的总称，通常包括文字处理软件、电子表

格处理软件、演示制作软件、个人数据库等。常见的软件套件有 Microsoft Office、金山公司 WPS、Adobe Reader 等。

3. 多媒体处理软件

随着多媒体技术的广泛应用，多媒体处理软件也成为应用软件中的重要分类。多媒体处理软件主要包括图形/图像处理软件、动画制作软件、音频/视频处理软件、格式转换软件等。常用的多媒体处理软件包括 Photoshop、Flash、3d Max、Premier、格式工厂等。

4. 科学计算软件

科学计算软件主要用于特定科学研究领域的计算，如经典分子动力学、数学计算、计算天文学、量子化学、计算材料物理等领域。常用于数学计算的软件有 Mathematica、MATLAB 和 Maple，可实现矩阵计算、微分方程求解、图形化展示等功能。

5. 嵌入式软件

嵌入式软件与嵌入式系统是密不可分的。嵌入式系统一般由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户的应用程序 4 个部分组成，用于实现对其他设备的控制、监视或管理等功能。嵌入式软件就是基于嵌入式系统设计的软件，它也是计算机软件的一种，同样由程序及其文档组成，是嵌入式系统的重要组成部分。常用的嵌入式系统有 μClinux、μC/OS-II、eCos 等。

6. 实时软件

实时软件是必须满足严格时间约束条件的软件，用来监控、分析、控制实时事务。它包括从外部环境收集信息，分析后按要求转移信息，处理后做出响应，监控部件在指定时间内（通常在 1 μs～1 s 之间）完成相应动作，多用于对实时性要求高的工业控制系统，如电力系统保护、电力生产调度，常用的实时系统有 ros2 等。

7. 程序开发工具环境

程序开发语言种类繁多，目前所使用的编程语言多以集成开发环境（IDE，Integrated Development Environment）的形式出现。即在此集成开发环境中，包含了语言的编辑、调试、编译、运行、图标图像制作等工具。在 Windows 环境下，常用的 IDE 有 Visual Studio 开发套件，包括 Visual C++、Visual C#等。

8. 网络工具软件

随着计算机网络的发展和普及，出现了许多基于网络的软件，主要有 Web 服务器软件、Web 浏览器、邮件软件、网络聊天软件、网络会议软件等。常用的有 IIS、TomCat、FoxMail、QQ、微信（WeChat）等。

除上述的几类软件外，还有各种学习软件、翻译软件、电子词典软件、视频播放软件等。

1.1.2 软件的特性

尽管软件种类繁多，但具有以下一些共同特性。

1. 软件是功能和性能相对完备的程序系统

软件不仅包括程序及所使用的数据，还包括说明其功能、性能的说明性信息，如使用维护说明、指南、培训材料等。在 1983 年由 ANSI/IEEE（美国国家标准学会/电气与电子工程师协会）制定的 IEEE Standard Glossary of Software Engineering Terminology（IEEE 软件工程标准术语）中，给出关于 Software（软件）的定义："Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system."（计算

机程序、过程、可能相关的文档以及与计算机系统操作有关的数据)。由此可见,软件涵盖的范围更广,有程序、有关数据、文件以及规程、规则等,软件与程序的包含关系如图 1-1 所示。

图 1-1　软件与程序的包含关系

2. 软件是具有使用性能的软设备

人们编写软件,往往是用于解决某个具体问题。同时,该软件具有良好的使用性能并可转让给其他人使用。

3. 软件是一种信息商品

信息商品的生产不同于其他传统产业对商品的生产,软件产生初期的研制开发作为其主要的生产方式,往往占用了很大的成本。一旦完成,通过复制再进行大批量生产则变得十分容易。

4. 软件只有"过时"而无"磨损"

软件和硬件不同,硬件产品都有使用寿命,而软件只有"过时"而无"磨损"一说。所谓的过时往往是因硬件环境及配套软件升级而导致软件原来的版本不再适用,所以必须对软件进行升级来避免过时。

1.2　程序及其特性

1.2.1　程序的概念

软件中最核心的部分是程序,程序是为解决某个问题而编写的计算机指令序列。图灵奖获得者、Pascal 语言之父瑞士计算机科学家尼古拉斯·沃斯(Niklaus Wirth)曾提出"程序=算法+数据结构",该公式揭示了程序的本质,同时指明了如何设计程序。将设计好的程序装入计算机内存,按控制结构依次逐条执行,最终解决指定的问题。

1.2.2　程序的特性

程序具有如下特性。

1. 静态表示与动态运行

程序需要采用某一种编程语言进行描述,其表示是静态的。但是,编写程序的目的是要用它来解决问题,所以程序必须能够运行,否则毫无用处。因此,从程序的表示来看,若干代码体现了其静态属性。同时,程序通过运行才能发挥作用,体现了其动态特性。

2. 程序是抽象的符号表达

为了在计算机屏幕上画出一个"笑脸"的图形,需要编写相应的代码。

【例 1-1】 采用某种编程语言实现"笑脸"图形,对应的主要代码如下。

```
X = 100;      //设定横坐标
Y = 100;      //设定纵坐标
...           //此处省略了其他可能参数代码
Circle();     //调用画圆函数
Line();       //调用画线函数
```

运行上述代码,可画出如下的"笑脸"图形,如图 1-2 所示。

代码运行后,展现出具体生动的"笑脸"图形,而在画出对应图形之前,仅看代码是难以想象其执行结果的。相对于具体的结果来说,代码则是一种利用某种编程语言的抽象表达和

图 1-2 “笑脸”代码运行效果图

组织。

3. 程序是对数据施加算法的过程

在程序的编写过程中，不仅要考虑编程语言是否符合语法要求，还要考虑如何对数据进行操作以达到要求。其中的数据处理方法即算法。数据一般用于描述事物的属性和状态，而算法则是要设计一定的数据处理方法和过程，以满足实际问题的需要。从这个角度来看，程序的编写就是对数据设计相应算法的过程。

4. 程序是分层嵌套的

程序在执行过程中，利用底层的中断指令，可以在某个中间位置暂停执行(挂起)转而去执行另一个子程序，待子程序执行完成后再返回源程序继续执行，直到完毕后 CPU 不再执行任何指令。在程序调用过程中，能够看出其结构也是层层嵌套的。

【例 1-2】 编写 C 语言程序代码如下。

```
#include<stdio.h>
void main()
{
...
printf("%s","Hello, World");
...
}
```

上述代码的执行过程可以采用如图 1-3 所示的流程表示。在执行过程中，当执行到 printf 函数时，会跳转到该函数的内部过程，当完成输出后再返回源程序。若代码中包含个人编写的函数，其执行过程类似。

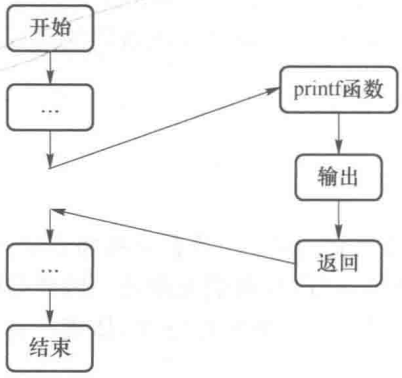


图 1-3 程序的嵌套结构

1.3 程序的运行过程

1.3.1 程序的执行

1. 高级语言的产生

计算机可以直接执行的语言称为机器语言。而机器语言利用二进制编码表示，使用过程枯燥且极易出错。例如约定操作码 00000100 为“加”运算，则计算机执行上述二进制代码，可

实现加法运算。但若将二进制代码写成 00010100，则其代表操作为其他类型，从而导致计算错误。在不断实践的过程中，人们发现可以利用容易记忆的英文单词代替约定的指令，容易地完成程序的读/写，汇编语言由此产生。例如用 ADD 代表 00000100，就可以使用 ADD 指令完成加法操作。汇编程序同样需要转换成二进制代码后执行，该工作由汇编程序自动完成。

汇编语言面向机器，使用汇编语言编程需要直接安排存储，规定寄存器、运算器的动作次序，还必须知道计算机对数据约定的表示(定点、浮点、双精度)方法等，这对大多数人来说不是一件简单的事情。此外，虽然汇编语言对操作码、寄存器做了一些抽象说明，但是它与计算机紧密相关，不同的计算机在指令长度、寻址方式、寄存器数目、指令表示等方面都不一样，这使得汇编程序不仅不可移植，而且读起来也很困难，从而促成了高级语言的出现。

高级语言采用类似人类语言的方式编写代码，特别是在数学计算方面，更加接近数学公式。同时，编程人员无须关心机器的运算器、寄存器和内存地址等内容。因此，编程人员的工作重心就转换为仅考虑如何使用高级程序设计语言表示相关数据及问题的求解步骤。

2. 高级语言的翻译

高级语言必须经过翻译变成机器语言后才可以被计算机执行，这个工作一般由翻译程序自动完成。把一种语言翻译成另一种语言的翻译程序叫作翻译器，翻译器可以起到将高级语言翻译成低级语言(包括汇编语言和机器语言)的作用，其翻译过程如图 1-4 所示。

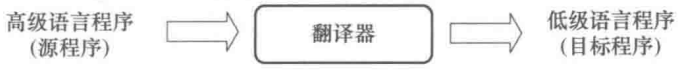


图 1-4 翻译器的转换作用

一个翻译器的重要性体现在，编程人员借助翻译器可以不用考虑与计算机有关的烦琐细节，编程人员的主要精力可以放在程序的设计与实现上，从而独立于具体使用的计算机，进一步提高编程效率。

翻译器对高级语言进行翻译的方式有编译和解释两种，对应的翻译程序分别叫作编译器和解释器。以编译器的执行过程为例，高级语言的编译过程如图 1-5 所示。

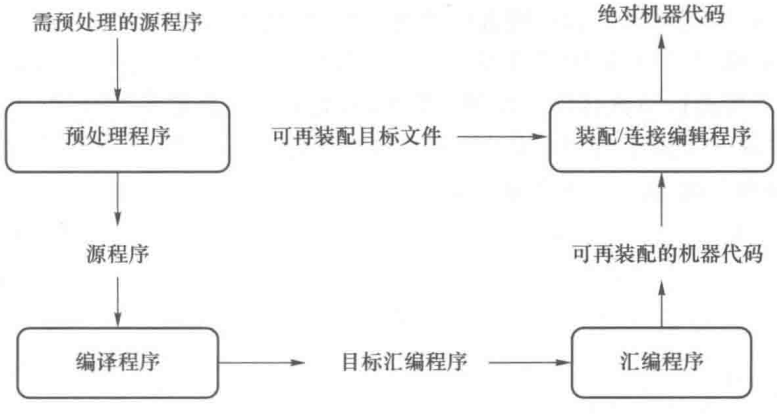


图 1-5 高级语言的编译过程

1.3.2 编译器的工作原理

高级语言程序利用编译器可以转换为一个可执行的机器语言程序，编译器的基本工作过程包括 6 个步骤。

第 1 步是词法分析(Lexical Analysis)。编译器逐行扫描源程序并识别符号串，主要包括关键字、字面量、标识符(变量名、数据名)、运算符、注释行和特殊符号(如续行、语句结束、数组等)。

将上述六类符号分别归类并等待处理，如语句 position= initial + rate * 60 会被编译器从左到右逐个字符读入并识别形成记号流 position (id1)，=，initial (id2)，+，rate (id3)，*，60。其中，分别使用 id1、id2、id3 表示实型变量 position、initial、rate，单词间的空格被忽略。

第 2 步是语法分析(Syntax Analysis)。根据词法分析得到的记号流，将该语句作为一串记号流由语法分析器进行处理。将记号流(即形成的单词序列)分解成各类语法短语，如“程序”“语句”“表达式”等。一般这种语法短语也叫语法单位，可表示成语法树。如上述单词序列 id1＝id2＋id3 * 60 经语法分析知其是 C 语言的“赋值语句”，可表示成如图 1-6 所示的语法树。

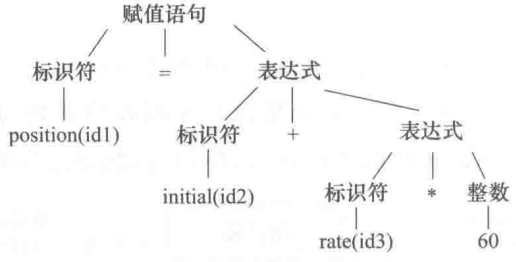


图 1-6 语句对应的语法树

在实际分析的过程中，也可以将上述语法树用一种简化的形式表示，如图 1-7 所示。

根据语言的文法检查每个语法分析树，判定其是否为符合语法的句子。如果是合法的句子，就以内部格式把此语法树保存起来，否则报错，像这样直到检查完所有程序为止。

第 3 步是语义分析(Semantic Analysis)。语义分析器对各个句子的语法树进行检查，确保源程序各部分之间的语义一致性，以保证程序各部分能有意义地结合在一起。检查的内容包括：运算符两边的类型是否兼容；该做哪些类型转换；是否控制转移到不该去的地方；是否有重名或使语义模糊的记号等。如有则进行处理，否则生成中间代码。

例如，在计算机内部，整数的二进制表示和实数的二进制表示是有区别的。因此，在进行二者的运算时，需要进行隐式转换。在图 1-7 中，所有的变量都是实型变量，而 60 是整数。通过类型检查会发现 * 作用于实型变量 id3 和 60，需要建立一个额外的算符结点 inttofloat，显式地将整数转换为实数，如图 1-8 所示。

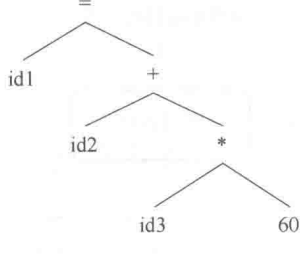


图 1-7 语法树的简化表示

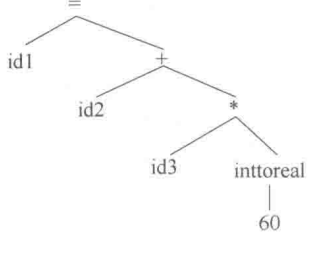


图 1-8 语义分析插入类型转换 inttofloat

第 4 步是生成中间代码。经过前面的步骤，可将源程序转换为一种内部表示形式即中间代码。中间代码是向目标码即机器语言的代码过渡的一种编码，其形式尽可能和机器的汇编语言相似，以便下一步的代码生成。中间代码不涉及具体机器的操作码和地址码。采用中间代码的好处是可以在中间代码上做优化。

很多编译程序生成的中间代码具有“四元式”的形式，即运算符、运算对象 1、运算对象 2、结果。图 1-8 所示的语法树可以通过中间代码生成算法转换成 4 条中间代码，其中 t1、t2、t3 为编译程序生成的临时名字，具体如下所述。

```
(inttoreal     60    -    t1)
(*             id3   t1   t2)
(+             id2   t2   t3)
(=             t3    -    id1)
```

第 5 步是代码优化。独立于机器的代码优化阶段试图改进中间代码，以便产生具有运行更快、占用空间更小等优点的目标码。代码的优化可以从局部优化和全局优化两个方面进行，如局部优化包括合并冗余操作、简化计算等。例如，“x=0;”可用一条“清零”指令替换。而全局优化则包括改进循环、减少调用次数和快速地址算法等。

具体在进行优化时，通常是在中间代码生成后再利用代码优化器进行优化。例如，代码优化器用 60.0 代替 60 就可以把 inttofloat 运算删除。此外，t3 只被引用一次，就是取它的值传给 id1。因此，可以用 id1 代替 t3。这样直接产生的 4 条中间代码，经过优化后可以得到如下结果：

```
(*             id3   60.0   t1)
(+             id2   t1     id1)
```

不同的编译器所实现的优化程度是不同的，能完成大部分优化的编译器称为优化编译器，但这时编译时间中相当长的一部分都消耗在这种优化上。简单的优化也可以使目标程序的运行时间大大缩短，而编译速度并没有降低太多。

第 6 步是代码生成。由代码生成器生成目标机器的目标码(或汇编)程序，要做数据分段、选定寄存器等工作，然后生成机器可执行的代码。此过程的一个关键问题是寄存器的分配，例如使用寄存器 R1 和 R2，则优化后的中间代码可以翻译为

```
MOV R2,id3
MUL R2,#60.0
MOV R1, id2
ADD R1,R2
MOV id1,R1
```

上述汇编指令的第 1 个和第 2 个操作数分别代表目的操作数和源操作数。上述指令的过程是第 1 条指令先将 id3 放入寄存器 R2，然后第 2 条指令将 R2 和 60.0 相乘，结果仍存放在 R2 中，其中的#代表 60.0 作为立即数处理。第 3 条指令将 id2 放入 R1 中，第 4 条指令实现 R2 和 R1 的内容相加，并把结果放在 R1 中。第 5 条指令是将 R1 中的值传递给 id1。这样该代码段实现了赋值语句。

高级语言源程序经编译后得到目标码程序，但它还不能立即装入机器执行，一般情况下它是不够完整的。如程序中用到 abs()、sin()这些函数，可直接调用，不需求绝对值、求正弦的程序，它们已作为目标码存放在机器中。

编译后得到的目标模块还需进行连接。连接程序(即 Linker)找出需要连接的外部模块

并到模块库中找出被调用的模块，调入内存并连接到目标模块上，形成可执行程序。把执行程序加载(Loading)到内存中合适的位置，即可执行。例如，使用C语言编写的源程序得到结果的过程如图1-9所示。

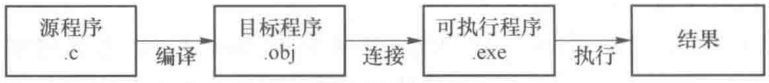


图1-9　C语言源程序的执行过程

1.3.3　解释器的工作原理

编译型语言由于可进行优化(有的编译器可做多次优化)，目标码效率很高，所以它是目前软件实现的主要方式。例如使用C语言编写的源程序，都需要进行编译、连接，才能生成可执行程序。编译时虽然花费时间，但程序的执行效率将会提高。如果不把整个程序全部编译完成，则是不能运行该程序的。编译和运行是两个独立分开的阶段。

但是在一个交互的环境中，不需要将上述两个阶段分开，此时编译就不如解释方便。当采取解释执行的方式时，需要有一个解释器(Interpreter)，它将源代码逐句读入。先做词法分析，建立内部符号表；再做语法和语义分析，即以中间代码建立语法树，并做类型检查。完成检查后把每一语句压入执行堆栈，压入后立即解释执行。

在图1-10所示的堆栈中，首先弹出栈顶元素“*”，从符号表中得知它是“乘法”操作，翻译为机器的乘法指令，要求有两个操作数。接着弹出“id3”，查表知道这是变量，可作为赋值号左端操作数。再往下弹出“inttoreal()”，inttoreal()不是数值而是函数调用(其功能是把整数转换为实数，其执行代码此前已压入执行堆栈)，于是寻找inttoreal函数，并弹出参数“60”，执行完的结果作为原表达式的第2个操作数。

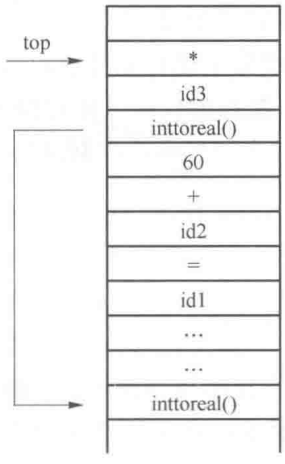


图1-10　执行堆栈中存放的元素

接着再弹出“+”，表明这是加法运算，第1个操作数已在加法器中，再弹出“id2”，知道是一个变量，可做第2个操作数。执行加法操作后再弹出“=”，再弹出“id1”作为赋值对象，完成赋值。

所有的记号idi按符号表对应地址码，所有运算符对应操作码，换成机器码后立即执行，接着下一句又开始压入栈。