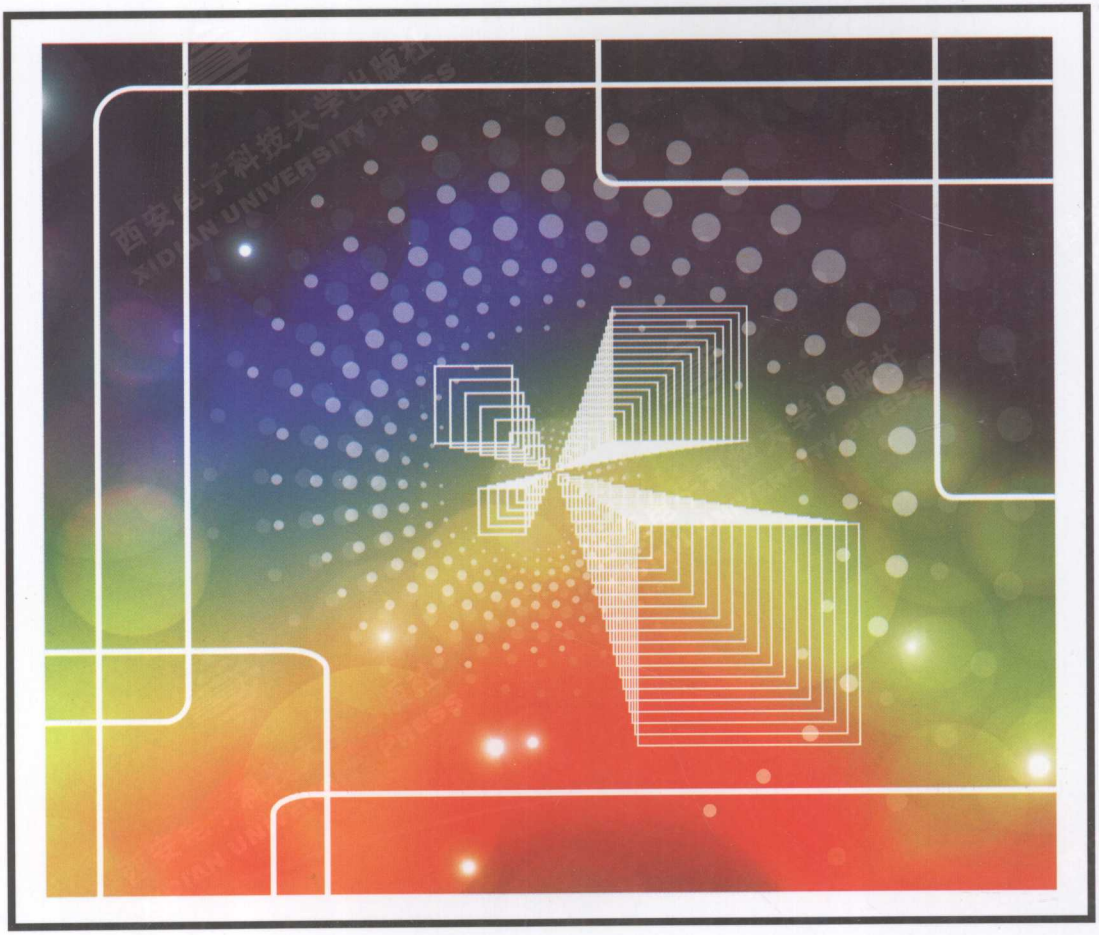
新世纪计算机类本科规划教材
COMPUTER

编译原理

主编　鱼滨　王小兵　张琛



西安电子科技大学出版社
http://www.xduph.com

新世纪计算机类本科规划教材

编译原理

主编 鱼 滨 王小兵 张 琛

西安电子科技大学出版社

内 容 简 介

本书是按照国家教育部制定的计算机专业编译原理课程教学大纲并兼顾目前授课时数压缩的现实情况编写而成的。

本书系统地介绍了高级程序设计语言编译程序构造的一般原理和实现方法，主要内容包括编译程序的构成、词法分析、语法分析、语法制导翻译与中间代码生成、自动机的应用、符号表与运行时环境、代码优化与代码生成等。通过本书的学习，读者可以对编译的基本概念、原理和构造方法有完整的认识和理解，并能正确地运用。

本书可作为高等学校计算机类专业的本科生教材，也可作为相关技术人员的参考书。

图书在版编目(CIP)数据

编译原理/鱼滨，王小兵，张琛主编. —西安：西安电子科技大学出版社，2014.3
新世纪计算机类本科规划教材
ISBN 978-7-5606-3332-9

Ⅰ. ①编… Ⅱ. ①鱼… ②王… ③张… Ⅲ. ①编译程序—程序设计—高等学校—教材
Ⅳ. ①TP314

中国版本图书馆 CIP 数据核字(2014)第 021925 号

策　　划　王　飞
责任编辑　王　瑛
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　陕西天意印务有限责任公司
版　　次　2014 年 3 月第 1 版　2014 年 3 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　　印　　张 12.5
字　　数　292 千字
印　　数　1～3000 册
定　　价　21.00 元

ISBN 978-7-5606-3332-9/TP

XDUP 3624001-1

前　言

“编译原理”是高等学校计算机科学与技术专业的一门专业必修课，在计算机本科教学中占有十分重要的地位。该课程从理论和实践两个方面讲授编译程序涉及的计算机技术基础理论及程序设计技术。编译程序是计算机系统软件的重要组成部分，是计算机科学中发展迅速、技术成熟的一个分支，其基本原理和技术也适用于一般软件的设计和实现，而且在软件工程、软件自动化、逆向软件工程、再造软件工程等领域有着广泛的应用。学习“编译原理”课程有助于学生掌握编译程序本身的实现技术，加深对程序设计语言的理解，也有助于学生抽象思维能力和软件开发能力的培养。

“编译原理”课程涉及程序设计语言的相关基础理论，形式化成分较多，学生感觉不易学，加之目前国内大多数“编译原理”教材内容偏多且深，而授课时数又不断压缩，从而造成教材内容不能完全使用、学生负担较重的现象。本书旨在以简练的语言、较少的篇幅讲述编译的原理和技术，并涵盖课程所要求学生掌握的内容。作为原理性教科书，本书着重介绍编译的基本理论和构造方法，对于一些具体的实现细节以及实际中不再使用的构造技术未予描述。为使学生对编译原理涉及的基础理论有更好的理解，本书还特别编排了“自动机的应用”一章。

全书共分7章。第1章绪论，介绍程序设计语言和语言翻译的基本概念，内容包括语言翻译与编译程序、编译器与解释器、编译程序的工作原理与基本结构、编译器的编写等。第2章词法分析，是整个编译内容的基础，除了介绍词法分析器的构造外，主要讲授正规式和有限状态自动机的基本知识，内容包括词法分析器的工作方式、模式的形式化描述、有限自动机、正规式到词法分析器、词法分析器的自动生成等。第3章语法分析，讲授文法和基本语法分析器的构造，内容包括上下文无关文法、自上而下的语法分析、自下而上的语法分析、二义文法的应用、语法分析器的自动生成工具YACC简介等。第4章语法制导翻译与中间代码生成，讲授语法制导翻译生成中间代码的方法，内容包括语法制导翻译、中间代码、说明性语句的翻译、执行性语句的翻译等。第5章自动机的应用，讲授自动机在不同领域的应用，内容包括有限自动机在自动控制软件设计中的应用、移动通信营业系统中的自动机模型、图形识别的有限自动机方法、基于广义有限自动机的图像压缩方法等。第6章符号表与运行时环境，主要讲授符号表、目标程序运行时环境、目标程序运行时存储器的划分及存储分配策略等内容。

第7章代码优化与代码生成，讲授代码生成所需考虑的问题，内容包括代码优化、代码生成的实现过程、简单的代码生成器、DAG的代码生成等。每章末均配有丰富的习题，以帮助学生更好地掌握编译原理的相关知识。本书配有《〈编译原理〉习题解答及上机指导》一书，对学生有指导和帮助作用。

编者力图反映编译的基本理论和程序构造技术，用简洁的语言讲述抽象的原理，但由于水平所限，书中难免存在不当之处，敬请读者批评指正。

编　者

2013年12月

目　录

第1章 绪 论

1.1 语言翻译与编译程序

使用过现代计算机的人都知道，多数用户都应用高级语言来实现他们所需要的计算。而在计算机上执行一个高级语言程序一般分为两步：首先，使用编译程序把高级语言翻译成机器语言程序；其次，运行机器语言程序求得计算结果。通常所说的编译程序(编译器)是指一种能够把某一种语言程序(源程序)转换成另一种语言程序(目标程序)的程序。后者与前者在逻辑上是等价的。编译程序根据不同的用途和侧重点可进一步分为诊断编译程序(Diagnostic Compiler)和优化编译程序(Optimizing Compiler)。诊断编译程序是专门用于帮助程序开发和调试的编译程序；优化编译程序是着重于提高目标代码效率的编译程序。现在很多编译程序同时提供了调试和优化等多种功能，用户可以通过“开关”进行选择。

通常将运行编译程序的计算机称为宿主机，运行编译程序产生目标代码的计算机称为目标机。如果一个编译程序产生不同于其宿主机的机器代码，则称它为交叉编译程序(Cross Compiler)。如果不需要重新编译程序中与机器无关的部分就能改变目标机，则称该编译程序为可变目标编译程序(Retargetable Compiler)。

现在计算机系统一般都含有不止一个的高级语言编译程序，对有些高级语言甚至配置了几个不同性能的编译程序，供用户按不同需要进行选择。高级语言编译程序是计算机系统软件最重要的组成部分之一，也是用户最关心的工具之一。

1.2 编译器与解释器

编译器是将一种语言翻译为另一种语言的计算机程序。编译器将源语言(Source Language)编写的程序(即源程序)作为输入，产生用目标语言(Target Language)编写的等价程序(即目标程序)。通常，源程序用高级语言(如 Java、C++、FORTRAN 等)编写；目标程序则是目标机的目标代码(Object Code)，有时也称做机器代码(Machine Code)，也就是写在计算机机器指令中的用于运行的代码。这一过程表示如下：

源程序→编译器→目标程序

编译器是一种相当复杂的程序，其代码长度从 10 000 行到 1 000 000 行不等。编写甚至读懂这样的一个程序并非易事，大多数计算机科学家和专业人员从来没有编写过一个完

整的编译器。但是，几乎所有形式的计算均要用到编译器，而且任何一个与计算机打交道的专业人员都应掌握编译器的基本结构和操作。除此之外，计算机应用程序中经常遇到的一个任务就是开发命令解释程序和界面程序，这些程序比编译器要小，但使用的技术却是相同的。因此，掌握这一技术具有非常重要的实际意义。

高级语言程序除了像上面所说的先编译后执行外，有时候也可以用“解释”来执行。解释器是这样一种程序，它以高级语言编写的源程序作为输入，但不产生目标程序，而是一边解释一边执行源程序本身。

从原理上讲，任何程序设计语言都可被解释或被编译，但是根据所使用的语言和翻译情况看，很可能会选用解释器而不用编译器。例如，我们经常解释 BASIC 语言而不是去编译它，但是，当执行速度是最重要的因素时就要使用编译器。这是因为被编译的目标代码比被解释的源代码要快得多，有时要快 10 倍或更多。

☞ 1.3 编译程序的工作原理与基本结构

1.3.1 高级语言的主要成分

构造编译程序前，定义和理解高级语言是必不可少的。因为对于高级语言的定义和理解是人们进行编译的基础。人们使用高级语言来实现自己所需要的计算，而这些高级语言就构成了编译器的输入——源语言。

高级语言是用来描述算法和计算机实现的。目前，世界上的高级语言有上千种，在较大范围内得到使用的语言也有几十种甚至上百种。从应用角度看，它们各有不同的侧重点。如 FORTRAN 适用于数值计算，COBOL 适用于事务处理，PROLOG 适用于人工智能，Ada 适用于大型嵌入式实时处理，SNOBOL 则适用于符号处理。从语言范畴来分，高级语言可分为强制式语言、作用式语言、基于规则的语言和面向对象语言等。

一个程序语言是一个记号系统。如同自然语言一样，程序主要由语法和语义两个方面定义。有时候语言定义中也包含语用信息。语用主要是有关程序设计技术和语言成分的使用方法，它将语言的基本概念与语言的外界(如数学或计算机的对象和操作)联系起来。下面对语法和语义做进一步的解释。

1. 语法

任何语言程序都可以看成是一定字符集(称为字母表)上的字符串(有限序列)，并且一个程序语言只使用一个有限字符集作为字母表。一个语言的语法是指这样的一组规则，用它可以形成和产生一个合式的程序。这些规则的一部分称为词法规则，另一部分称为语法规则(或产生规则)。

词法规则是指单词符号的形成规则。单词符号是语言中具有独立意义的最基本的结构。在现今的多数程序语言中，单词符号一般包括各类型的常数、标识符、基本字、运算符和分界符等。由于单词符号本身很简单，因此形成规则也不复杂。正规式和有限自动机理论就是描述词法结构和进行词法分析的有效工具。

语言的语法规则规定了如何从单词符号形成更大的结构(即语法单位)。换言之，语法

规则是语法单位的形成规则。一般程序语言的语法单位有表达式、语句、分程序、函数、过程和程序等。要真正地描述语法规则一般是很不容易的，但就现今的多数程序语言来说，上下文无关文法仍是一种可取的有效工具。有限自动机和上下文无关文法是我们讨论词法分析和语法分析的主要理论基础。语法单位比单词符号具有更丰富的意义。例如，单词符号串“2＋3＊3.14”代表一个算术式，具有通常的算术意义。如何定义各种语法单位的含义属于语言的语义问题。

语言的词法规则和语法规则定义了程序的形式结构，是判断输入字符串是否构成一个形式上正确(即合式)程序的依据。

一般而言，程序语言的词法、语法规则并不限定程序的书写格式。但是，某些程序语言要求程序的书写服从一定的格式，而这样的要求就增加了词法分析的复杂性。现在多数语言倾向于使用自由格式书写法，容许程序员随自己的意愿编写程序。这既便于阅读，又可以回避因书写格式不正确而造成的错误。

有些语言规定，空白字符除了在文字常数中出现之外，在其他任何地方出现都是没有意义的。在这种情况下，空白字符可用于编排程序格式，但增加了词法分析的复杂度。在某些语言中，空白字符用做间隔符，它们的出现决定了单词符号的划分。

2. 语义

对于一个语言来说，不仅要给出它的词法、语法规则，而且要定义它的单词符号和语法单位的意义，这就是语义问题。离开语义，语言只不过是一堆符号的集合。在许多语言中，形式上完全相同的语法单位，其含义却不尽相同。同样的符号串代表的“算术表达式”，但是含义却有可能不同。对编译而言，只有了解程序的语义，才知道应把它翻译成什么样的目标指令代码。

所谓一个语言的语义是指这样的一组规则，使用它可以规定一个程序的意义，这些规则称为语义规则。阐明语义要比阐明语法难得多。现在还没有一种公认的形式系统，借助于它可以自动地构造出实用的编译程序。目前大多数编译程序普遍采用的一种方法是基于属性文法的语法制导翻译方法，虽然这还不是一种形式系统，但它还是比较接近形式化的。

一个程序的基本功能是描述数据和数据的运算。所谓一个程序，从本质上来说是描述一定数据的处理过程。在现今的程序语言中，一般程序的层次结构如图 1.1 所示。

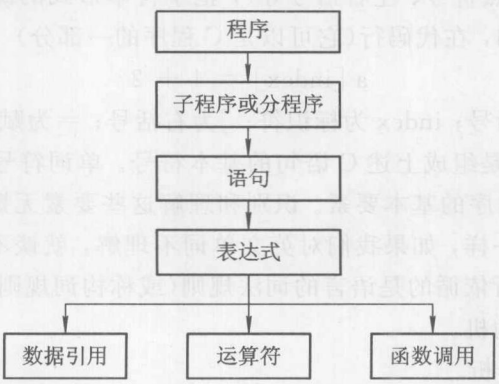


图 1.1　一般程序的层次结构

自上而下看图 1.1 的层次结构，顶端是程序本身，它是一个完整的执行单位，通常由若干个子程序或分程序组成，子程序与分程序常常含有自己的数据；子程序或分程序是由语句组成的；组成语句的成分则是各种类型的表达式，表达式是表述数据运算的基本结构，它通常含有数据引用、运算符和函数调用。

1.3.2 编译的基本过程

编译过程是指从输入源程序开始到输出目标程序为止的整个过程。编译过程与人们进行自然语言之间的翻译有许多相近之处。例如，将英文翻译成中文所需的五个步骤与编译程序的五个阶段如图 1.2 所示。

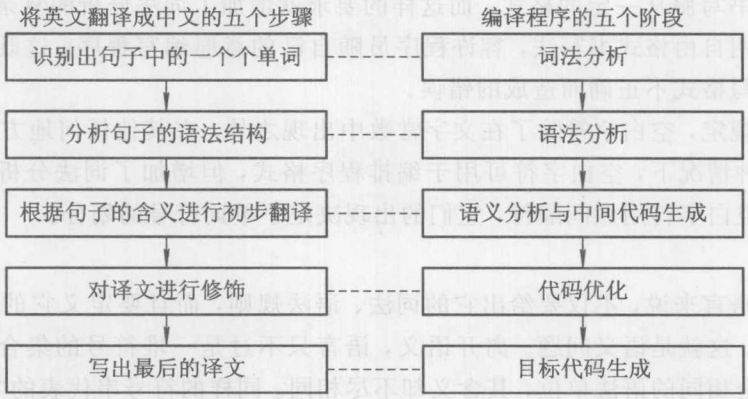


图 1.2　编译程序的主要工作过程

1.3.3 编译程序各阶段的工作

编译程序分五个阶段：词法分析、语法分析、语义分析与中间代码生成、代码优化、目标代码生成。

第一个阶段：词法分析。

词法分析的任务是：输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个的单词(也称单词符号或符号)，如基本字(begin、end、if、for、while 等)、标识符、常数、运算符和分界符(标点符号、左右括号等)，把字符串形式的源程序改造成为单词符号串形式的中间程序。例如，在代码行(它可以是 C 程序的一部分)

a [index] = 4 + 2

中：a 为标识符；[为左括号；index 为标识符；]为右括号；=为赋值号；4 为常数；+为加号；2 为常数。这些单词是组成上述 C 语句的基本符号。单词符号是源程序的基本组成成分，是人们理解和编写程序的基本要素。识别和理解这些要素无疑也是翻译的基础。与把英文翻译成中文的情形一样，如果我们对英文单词不理解，就谈不上进行正确的翻译。在词法分析阶段的工作中所依循的是语言的词法规则(或称构词规则)。描述词法规则的有效工具是正规式和有限自动机。

第二个阶段：语法分析。

语法分析的任务是：在词法分析的基础上，根据语言的语法规则，把单词符号串分解

成各类语法单位(语法范畴)，如短语、字句、句子(语句)、程序段和程序等。通过语法分析，确定整个输入串是否构成语法上正确的程序。语法分析所依循的是语法规则。语法规则通常用上下文无关文法描述。词法分析是一种线性分析；语法分析是一种层次结构分析。例如，符号串

$$c=a+3*b$$

代表一个赋值语句，而其中的“a+3 * b”代表一个算术表达式。语法分析的任务是识别“a+3 * b”为算术表达式，同时还识别上述整个符号串属于赋值语句的范畴。

第三个阶段：语义分析与中间代码生成。

语义分析与中间代码生成的任务是：对语法分析所识别出的各类语法范畴分析其含义，并进行初步翻译(产生中间代码)。这一阶段通常包括两个方面的工作：首先，对每种语法范畴进行静态语义检查；其次，如果语义正确，则进行下一步的工作，即进行中间代码的翻译。这一阶段所依循的是语言的语义规则。通常使用属性文法描述语义规则。

中间代码是一种含义明确、便于处理的记号系统，它通常独立于具体的硬件。这种记号系统或者与现代计算机的指令形式有某种程度的接近，或者能够比较容易地把它变换成现代计算机的机器指令。如采用四元式作为中间代码，中间代码生成的任务就是按语言的语义规则把各类语法范畴译成四元式序列。如：

运算符	左操作数	右操作数	结果

其意义是对左、右操作数进行运算符规定的运算，把运算所得的值作为结果保留下来。例如，表达式

$$z=(a+b)*c/d$$

的四元式可为

$$(+\quad a\quad b\quad T_1)$$
$$(*\quad T_1\quad c\quad T_2)$$
$$(/\quad T_2\quad d\quad z)$$

其中，T_1、T_2 都是编译期间引进的临时工作变量。

一般而言，中间代码是一种独立于具体硬件的记号系统。常用的中间代码除了四元式外，还有三元式、逆波兰记号和树形表示等。

第四个阶段：代码优化。

代码优化的任务是：对前段产生的中间代码进行加工变换，以求在最后阶段能产生出更为高效(省时间和空间)的目标代码。它包括以下几个方面：公共子表达式的提取、循环优化、删除无用代码等。有时，为了便于并行运算，还可以对代码进行并行化处理。优化所依循的原则是程序的等价变换规则。

第五个阶段：目标代码生成。

目标代码生成的任务是：把中间代码(或经优化处理之后的代码)变换成特定机器上的低级语言代码。这个阶段实现了最后的翻译，它的工作依赖于硬件系统结构和机器指令含义。这一阶段的工作非常复杂，涉及硬件系统功能部件的运用、机器指令的选择、各种数据类型变量的存储空间分配以及寄存器和后援寄存器的调度等。如何产生足以充分发挥硬件效率的目标代码是一件非常不容易的事情。

目标代码可以是绝对指令代码、可重定位的指令代码或汇编指令代码。如目标代码是绝对指令代码，则这种目标代码可立即执行。如果目标代码是汇编指令代码，则需要汇编器汇编之后才能运行。必须指出，现代多数使用编译程序所产生的目标代码都是一种可重定位的指令代码。这种目标代码在运行前必须借助于连接装配程序把各个目标模块连接在一起，确定程序变量在主存中的位置，装入内存中指定的真实地址，使之成为一个可以运行的绝对指令代码程序。

上述编译程序的五个阶段是一种典型的逻辑模型。事实上，并非所有编译程序都分成这五个阶段。有些编译程序对优化没有要求，优化阶段就可以略去；在某些情况下，为了加快编译速度，中间代码生成阶段也可以略去；有些最简单的编译程序是在语法分析的同时产生目标代码。但是，大多数使用编译程序的工作过程均由上述五个阶段构成。

1.3.4　编译程序的基本结构

编译过程被分为了五个阶段，编译程序的结构也可以按照这五个阶段的任务分模块进行设计，如图 1.3 所示。

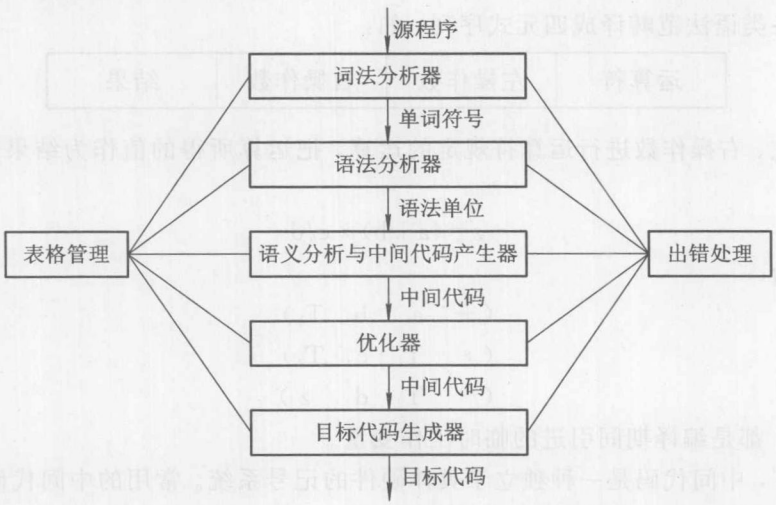


图 1.3　编译程序的总框架

词法分析器：输入源程序，进行词法分析，输出单词符号。

语法分析器：简称分析器，对单词符号串进行语法分析，识别出各类语法单位，最终判断输入串是否构成语法上正确的程序。

语义分析与中间代码产生器：按照语义规则对语法分析器归约出(或推导出)的语法单位进行语义分析并把它们翻译成一定形式的中间代码。

优化器：对中间代码进行优化处理。

目标代码生成器：把中间代码翻译成目标程序。

在上述框架中还有两个重要的概念就是表格管理和出错处理。

表格管理：编译程序在工作中需要保持一系列的表格，以登记源程序中各类信息的编译中各阶段的进展状况。合理地设计和使用表格是编译程序构造的一个重要问题。在编译

程序使用的表格中，最重要的是符号表。它用来登记源程序中出现的每个名字以及名字的各种属性，例如一个名字是常量名、变量名，还是过程名，以及它的类型、内存地址等信息。

编译的各个阶段都涉及了对这些表格的修改、查找或更新等工作，所以对于这些表格的管理也是编译程序的主要工作之一。

出错处理：一个编译程序不但要能对书写正确的程序进行翻译，而且应该能对出现在源程序中的错误进行处理。如果源程序中发现错误，编译器就应将信息发送给用户。这部分的工作是由专门的一组程序(出错处理程序)完成的。如果这部分程序不但能够发现错误，而且还能自动校正错误，就更好了，但是自动校正的代价是非常高的。

源程序中的错误通常分为语法错误和语义错误两大类。语法错误是指源程序中不符合词法或语法规则的错误，它们可在词法分析或语法分析时检测出来。语义错误是指源程序中不符合语义规则的错误，通常包括说明错误、作用域错误、类型不一致等。

1.3.5　编译的前端和后端

编译程序一般分为编译前端和编译后端。编译前端主要由与源语言有关但与目标机无关的部分组成，通常包括词法分析、语法分析、语义分析与中间代码生成，有的代码优化工作也可包括在编译前端。编译后端包括编译程序中与目标机有关的部分，如与目标机有关的代码优化和目标代码生成等。

可以取编译程序的前端，改写其后端，以生成针对不同目标机的编译程序。如果编译后端的设计是经过精心考虑的，那么编译后端的改写将不会有太大的工作量，这样就可实现编译程序的目标机改变。也可以设想将几种源语言编译成相同的中间语言，虽然这些语言的编译程序有不同的前端，但它们都可以配上相同的后端，这样就可以高效地为同一台机器生成不同语言的编译程序。然而，由于不同语言存在某些微妙的区别，因此在这方面所取得的成果非常有限。

1.3.6　编译的遍数

前面所讲的编译过程的五个阶段仅仅是逻辑功能上的一种划分，具体实现时，受不同源语言、设计要求、使用对象和计算机条件(如主存容量)等限制，往往将编译程序组织为若干遍(pass)。所谓遍，就是对源程序或源程序的中间结果从头到尾扫描一次，并做有关的加工处理，生成新的中间结果或目标程序。通常，每遍的工作由从外存上获得的前一遍的中间结果开始(对于第一遍而言，从外存上获得源程序)，完成它所含的有关工作之后，再把结果记录于外存。这既可以将几个不同阶段合为一遍，也可以把一个阶段的工作分为若干遍。为了便于处理，语法分析和语义分析与中间代码生成又常常合为一遍。若优化要求很高，还可以把优化阶段分为若干遍来实现。当一遍中包含若干阶段时，各阶段的工作是穿插进行的。

一个编译程序究竟应分成几遍，如何划分，是与源语言、设计要求、硬件设备等诸因素有关的，因此难于统一划定。遍数多一些是有好处的，即整个编译程序的逻辑结构会更清晰。但是遍数多肯定要增加输入/输出所消耗的时间，因此，一般的编译程序遍数都比较少。应当注意，并不是每种语言都可以用单遍编译程序实现的。

1.4 编译器的编写

人们早期使用汇编语言来编写编译器，但是由于编译器本身是一个十分复杂的系统，用汇编语言编写编译器的效率很低，往往给实现带来很大的困难，因此，除了特别需求，人们已经不再使用汇编语言编写编译器了。现在常用高级语言编写编译器，它的效率比汇编语言要高得多，不过用单纯的程序语言来编写编译器显然是不够的，因此需要一些专门的编译器编写工具来支持编译器某些部分的自动生成。比较成熟和通用的工具有词法分析器和语法分析器，如被广泛应用的 Lex 和 YACC；另外，还有一些工具，如语法制导翻译工具(用于语义分析)、自动代码生成器(用于中间代码与目标代码生成)、数据流工具(用于优化)等。这些工具有一些共同的特点，就是仅需要对语言相应部分的特征进行描述，而把生成算法的过程隐蔽起来，同时所生成的部分可以统一并入到编译器的其他部分中。因此，这些工具往往与某程序设计语言联系在一起，如与 YACC 联系的程序设计语言就是 C 语言。

1.5 本 章 小 结

本章介绍了有关编译原理的一些基本概念和知识，例如编译程序、高级程序语言、编译器和解释器、编译的前端和后端、编译的遍数等。读者应掌握的重点以及难点内容如下：

首先，了解什么是程序设计语言以及高级语言的翻译。

程序设计语言是用来编写程序的工具，分为低级和高级语言。低级语言包括机器语言及其面向机器的程序设计语言。而高级语言有上千种之多，常用的有 Java、C++、C、Pascal 等。高级语言比起低级语言在算法描述能力、编程和调试效率上都有较高的优越性。

如果在计算机上使用高级语言，需要该语言能够被计算机所理解。使用的方法就是对程序进行翻译或解释。编译程序就是将高级语言编写的程序(源程序)翻译为汇编语言或者机器语言形式(目标程序)的一种翻译程序。

其次，清楚一个编译程序需要做的工作和它的逻辑结构。

编译程序的主要工作有五个：词法分析、语法分析、语义分析与中间代码生成、代码优化、目标代码生成。

一般编译程序的逻辑结构包含以下几个部分：

(1) 词法分析程序；
(2) 语法分析程序；
(3) 语义分析程序；
(4) 中间代码生成程序以及优化程序；
(5) 目标代码生成程序；
(6) 错误检查和处理程序；
(7) 各种信息表的管理程序。