重点大学计算机教材

HZ BOOKS
华章教育

编译原理

编译程序构造与实践

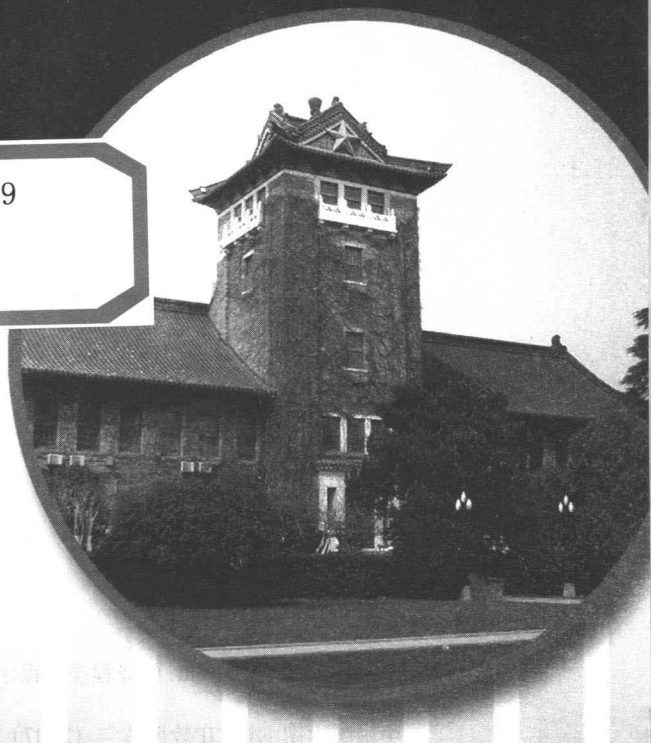
张幸儿 编著
南京大学

为教师配有电子教案

机械工业出版社
China Machine Press

重点大学计算机教材

编译原理
编译程序构造与实践



张幸儿 编著
南京大学

机械工业出版社
China Machine Press

本书系统地介绍了高级程序设计语言编译程序的构造原理，重点讨论了词法分析、语法分析、语义分析以及目标代码的生成与优化。各章末有本章小结，许多章还附有习题与上机实习题。本书实践性强，在编译程序构造的主要环节给出了实现之考虑，提供了具体而实际可行的实现方法和技巧供读者参考。

本书可作为计算机及相关专业编译原理课程的教材，同时也是计算机软件技术人员、研究生以及广大计算机爱好者的极佳参考。

图书在版编目（CIP）数据

编译原理：编译程序构造与实践 / 张幸儿编著. －北京：机械工业出版社，2007.10
（重点大学计算机教材）
ISBN 978-7-111-22251-4

Ⅰ.编… Ⅱ.张… Ⅲ.编译程序－程序设计－高等学校－教材 Ⅳ.TP314

中国版本图书馆CIP数据核字（2007）第135304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：王春华
北京瑞德印刷有限公司印刷·新华书店北京发行所发行
2008年1月第1版第1次印刷
184 mm × 260 mm · 19印张
定价：32.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线：（010）68326294

前　言

当前，仅有极少数的计算机工作者从事编译程序的开发工作，那么为什么还要开设编译原理课程呢？这是因为编译原理课程在计算机科学中有着重要的地位和作用。该课程与计算机其他学科紧密相关，在软件工程、逆向工程、软件再工程、语言转换及其他领域中有着广泛的应用，对软件开发也有一定的启发与指导作用。编译程序是一种符号处理工具，可以说，只要涉及符号语言的处理，就可以（也应该）采用编译的基本原理与技术。

设置编译原理课程的目的在于让学生系统地了解并掌握程序设计语言编译程序的构造原理。编译原理课程以形式语言理论中的有关概念为基础，讨论程序设计语言及编译程序实现技术。该课程的内容由三部分组成，即程序设计语言（相关概念）、形式语言理论（与编译实现相关的基本概念）和编译程序构造原理。从中也可以看出该课程的特点：理论性强、实践性强。因此必须考虑的一个问题是：如何将两者完美地结合起来，或者说，对于一般的本科院校，如何降低理论难度，突出实践性，加强实践应用能力。

本书尝试把理论与实践有机地结合起来。通常的编译原理教材虽然讨论了概念，也讨论了实现算法，但这些实现算法往往仍是理论上和概念上的。举几个例子：

- 语言的文法如何在计算机内表示？
- 语言的句子由推导生成，怎样由计算机自动进行推导来生成句子？
- 词法分析程序如何实现？特别是怎样由计算机自动识别标识符之作用域？
- SLR识别程序是典型的识别程序，但分析表如何存入计算机，又如何与驱动程序相配合来实现句子识别，并生成语法分析树？特别是，翻译方案中的语义动作实质上也是字符串，怎样实现这些语义动作？

……

一般的教材中并没有为读者给出这些问题的解答，但读者在本书中将得到满意的答案。

本书编写的指导思想是：

1) 内容覆盖编译程序全过程，但突出主要部分，简洁紧凑地陈述要点，文字简练。特别是系统而又紧凑地讨论了语法制导的翻译，有利于读者学习。

2) 结合作者的理解与体会阐述概念，观点明确，特别是内容中融合了作者的多年科研工作实践与教学经验，易于读者理解与接受。

3) 考虑到C语言是当前大多数读者熟悉的程序设计语言，因此尽管C语言不是最合适的，但还是以C语言为背景语言进行编译原理的讨论，使读者在掌握编译程序构造原理的同时，对C程序设计语言的相关概念也有更深入的理解与认识。

4) 强调实践性，对编译程序构造的各个主要方面都给出实现之考虑，给出具体而实际可行的实现方法和技巧。读者可以体会从源程序到目标程序的编译全过程，可以从本书中得到软件开发的方法和经验的启发。

本书各章末除了全章小结外，还附有习题和上机实习题。希望读者在解题时注意解题规范，培养有条理地思考问题、严谨的学风与工作作风；同时注意提高上机动手能力，有条件的读者可以根据自己的实际情况适当选题，进行上机实践。

学习，不仅仅是知识的积累，更是能力的培养。对于一个学生，重要的是打好基础、练

好基本功、 提高自身素质，才能适应多方面的需要。期望通过编译原理课程的学习，读者能朝着这些目标努力，不仅仅较为深刻地认识和理解程序设计语言及编译原理与技术的有关问题，也能为进一步开展计算机软件工作打下良好的基础。本书希望成为理论和实践紧密结合的编译原理课程教材，通过对编译程序全过程中各主要部分的实践，加深读者对编译程序构造原理的理解，并通过实践来培养和提高分析问题与解决问题的能力。

本书各章末的习题没有给出习题解答，这样可以更充分地发挥读者的独立思考能力。当然，如果有读者愿意和作者一起切磋习题或其他一些问题，作者将热忱欢迎。

作者E-mail地址：zhangxr0@sina.com 或 zhang_xinger@hotmail.com

在这里要感谢温莉芳总编对本书的大力支持，感谢王春华编辑为本书的出版所做的大量工作，也要感谢刘振安教授对本书的关注和支持。最后要感谢李意勉女士，在她的大力支持和帮助下，本书才得以完成，最终与读者见面。

作 者
2007年5月
于南京大学计算机科学与技术系

目　录

第1章 概 述

1.1 编译程序的引进

1.1.1 高级程序设计语言与程序

当今计算机技术发展异常迅速，其应用范围越来越广泛，几乎遍及人类社会的各个领域，使用计算机已是人们日常生活中一件不可或缺的事情。计算机之所以能发挥如此巨大的作用，除了高速发展的硬件外，便是因为有软件的支持，进一步说是因为软件的核心——程序的支持。

计算机运行程序，使得人们能获得所期望的效果。众所周知，计算机能直接接受、理解和运行的程序是机器语言程序。这类程序用特定的计算机机器语言编写，是一个二进制位串，其书写、阅读理解、查错改错均十分困难，更不用说人们之间的交流。不言而喻，机器语言程序的生产率是十分低下的。一种改进的方法是引进汇编语言，人们可以用一些符号表示来编写程序，尤其是引进了宏设施，使得与计算机性能细节密切相关的输入输出之类功能可以很简单地用汇编语言的宏功能实现，给程序书写者带来很大的方便。但汇编语言毕竟仅是符号化的机器语言，仍属低级语言，在程序生产率上并没有根本性突破。

高级程序设计语言是一种人为设计的符号语言。由于它采用了接近于日常用语和通常的数学表示法，在易写、易读、易理解、易查错改错等方面有了实质性的改进，尤其是用高级语言编写的程序几乎与具体计算机无关，便于相互交流以及在不同型号计算机之间移植。从而，提高了程序生产率，满足了人们对计算机软件日益增长的需求。可以说，高级程序设计语言的引进为计算机的推广应用打开了新的局面。

例如，下列C语言程序片段：

```
printf("请输入长方形的相邻两边之长:");
scanf("%f,%f",&a,&b);
s=a*b;
printf("长方形面积s=%5.2f\n", s);
```

其功能一目了然。高级程序设计语言的优点尽显无遗。

1.1.2 高级程序设计语言程序的执行

高级程序设计语言程序有着无可非议的优点，但一个明显的不足是：不能被计算机所直接接受、理解与运行。

高级程序设计语言作为一种语言，也是人们对话的工具，人们用某种高级程序设计语言写出程序来表达自己想让计算机做的事或想达到的效果。但任何高级语言程序归根结底是一个字符串，其中包括英文字母、数字、运算符与括号等。而计算机所能直接接受和理解的只能是二进制机器指令。为能让计算机确切地理解高级程序设计语言程序，其间应有一个“翻译”。如果这个“翻译”对高级程序设计语言程序逐个语句地进行分析，根据每个语句的含义模拟地执行，则这个“翻译”称为解释程序。典型的以解释方式执行的高级程序设计语言是BASIC和PROLOG，它们的优点是易于查错，执行时一发现错误，可立即进行改正。

以解释方式执行程序时功效较低，常常把高级程序设计语言程序翻译成计算机可直接接受、理解和运行的等价的二进制机器语言程序或汇编语言程序，这个“翻译”就是编译程序。编译程序是高级程序设计语言的支持环境或支持工具。它把高级程序设计语言程序翻译成等价的低级语言程序。前者称为源程序，后者称为目标程序，因此相应地有源语言和目标语言。

高级程序设计语言程序执行的示意图如图1-1所示。

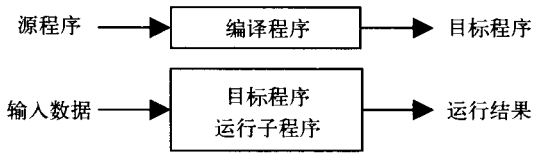


图 1-1

其中运行子程序是编译程序开发人员在开发时实现的一些系统子程序，有一些不为用户所了解，有一些则提供给用户直接使用，例如，计算三角正弦和余弦等初等函数值的子程序。这些运行子程序一般是精心研制的，功效好，质量高。它们的存在大大地方便了用户。没有运行子程序的支持，目标程序是无法独立运行的。

从源程序到目标程序的翻译仅进行一次，然而目标程序的运行可能进行任意多次，因此，倾向于设计和实现能产生功效较高的目标程序的编译程序，即使编译阶段花费更多的时间也无妨。编译时刻改进目标程序质量的工作称为优化。

从一个源程序到目标程序的翻译过程要经历哪些阶段，进行哪些加工处理，才能得到优化的目标程序？在对高级程序设计语言进一步讨论之后，我们就能理解编译程序与高级程序设计语言之间的联系，从而理解编译程序的构造是怎样的，编译过程要经历哪些阶段。

编译原理讨论的是编译程序构造的原理、技术和方法。当给定一个高级程序设计语言而要为它构造编译程序时，此编译程序将有怎样的结构，可采用什么技术，每种技术的要点是什么，在实现时又可能遇到什么问题，应如何解决？

如前所见，编译程序与高级程序设计语言紧密相关，当对高级程序设计语言有进一步的认识时，将对编译程序有更好的理解。另一方面，在对源程序进行编译的过程中，应对各个不同阶段分别采用最合适的技术，这与形式语言的分类有关。事实上，在讨论编译程序实现技术时往往以形式语言理论中的一些相关基本概念为基础进行讨论。因此，编译原理课程将包含三个方面，即

- 高级程序设计语言的定义和相关概念
- 与编译实现相关的形式语言理论基本概念
- 编译程序构造的原理、技术和方法

本教材将突出编译程序的构造原理及实现，使读者清楚地认识理论与实践之间的差距，掌握具体实现时的要点、方法与技巧，使读者不仅掌握编译程序的构造原理，更进一步地培养分析问题和解决问题的能力，以及上机实践的能力。

1.2 编译程序和程序设计语言的联系

1.2.1 程序设计语言的定义

程序设计语言作为人机通信和人们相互交流的工具而给出一组符号，以及由这些符号组成语法成分的规定，使得用户能表达要让计算机进行的工作，以达到预期的效果，这就是程序。程序是计算机系统中计算任务的处理对象和处理规则的描述。程序规定了计算机依次进行的操作。例如，下列C语言程序：

```
main( )
{ int x=13, y=5, temp;
```

```
    printf("x=%d, y=%d\n", x, y);
    temp=x;  x=y;  y=temp;
    printf("x=%d, y=%d\n", x, y);
}
```

该C语言程序按C程序设计语言的一定的框架组成，且其结构成分按概念层次由大到小可表示为：

程序—函数定义—说明和语句—表达式—量—符号（单词）—基本符号

其中基本符号是字母、数字、界限符（包括关键字、运算符、括号，及其他专用符号）。它们有的有意义，有的无独立意义。如单个字母无独立的意义，而+与-等有意义，即是运算符。由基本符号组成符号。C语言的符号，即单词或词法单位，可以是标识符、无符号数、字符串、标号和界限符，再由这些符号逐次构成更大的语法成分：量、表达式、说明和语句、函数定义，直到构成整个程序。但归根结底，一个程序是由基本符号组成的，或进一步说是由字符组成的。例如，main是标识符，它由m、a、i和n 4个字母组成，x和y也是标识符，却各由一个字母组成；关键字int由字母i、n和t组成；复合赋值运算符+=则由加号+和赋值号=组成，等等。这表明，组成符号有一定的规则，由字符组成符号的规则称为词法规则。进一步看到上述函数定义由函数首部和函数体两部分组成。这里的函数首部main( )由标识符main及一对括号组成。一般情况下，函数首部描述如下：

函数值类型 函数名（形参表列）

而函数体一般定义如下：

{ 说明部分 语句部分 }

一般地，程序设计语言的定义涉及四个方面，即语法、语义、语用和语境。

语法：指如何由程序设计语言基本符号组成程序中各个语法成分（包括程序）的一组规则。如上所述，由基本符号构成符号（单词）的书写规则称为词法规则，而由符号构成语法成分的规则称为语法规则。词法规则和语法规则统称为语法，它们都描述程序的书写规则。语法规则可用非形式的方式描述（如语法图），甚至用口语方式描述，但更可取的，也是本书采用的，是用形式的方式描述（如BNF表示法）。

语义：程序设计语言中按语法规则构成的各个语法成分的含义。一个程序执行的效果表达了该程序的含义，也就是程序的语义。程序的语义取决于组成程序的各个语法成分的语义定义，即取决于程序设计语言的语义。语义分静态语义和动态语义，编译时刻能确定的语法成分的含义是静态语义，运行时刻才能理解和确定的含义是动态语义。

期望像语法那样定义语义，以便精确而一致地定义程序设计语言的语义，有利于保证或验证程序的正确性，特别是自动生成程序。对此，已形成了计算机科学的又一个重要分支，即形式语义学。但此已超出本书的范畴，我们不作讨论。更因形式地定义语义问题还远非实用化，难以高效地实现编译，通常以非形式的口语描述方式来定义语义。例如，对于C语言if语句，其语法规则如下：

if (表达式) 语句 else 语句

其口语描述方式定义的语义如下。

首先计算表达式的值，然后判别此表达式的值是真还是假。当表达式的值是真时，执行紧跟其后的语句，然后跳过else后的语句执行if语句的后继语句，否则，跳过表达式后的语句去执行紧跟else之后的语句，然后执行if语句的后继语句。

语用：一般表示语言符号及其使用者之间的关系，涉及符号串的来源、使用和影响。对于程序设计语言，语用表示程序和使用者之间的关系。在程序中，往往用注释的形式注解某些变量的物理意义与用途等，这可看成是语用的体现。

语境：指理解和实现程序设计语言的环境，包括编译环境和运行环境。语境的不同将明显地影响到语言的实现。例如，C语言整型量通常占用2个字节，若占用4个字节，则C程序的书写与运行将有很大的不同。显然，同一种程序设计语言的易移植性很大程度上受语境的影响。

1.2.2 编译程序构造

1. 编译程序的功能

编译程序的功能是把高级程序设计语言源程序翻译成等价的低级语言目标程序。编译程序与程序设计语言紧密相关。由于程序设计语言的定义包括语法和语义，为了要识别出一个程序就得根据语法和语义定义来进行分析。根据程序设计语言的词法规则识别出各个具有独立含义的最小语法单位——符号，把它们处理成一种内部中间表示（这就是词法分析），然后根据一般的语法规则识别出一些连续的符号组成了什么语法成分，进一步把它们处理成一种内部中间表示（这就是语法分析），之后的工作是进行语义分析，即根据各个语法成分的语义定义进行相应的语义处理。在完成分析工作之后，便可以进行综合，进行目标程序的生成。为了生成高质量的目标程序，便在内部中间表示的基础上进行优化。

鉴于对于一个高级程序设计语言程序，为了从它生成等价的低级语言目标程序，需进行分析和综合，于是一个编译程序应由两部分组成：前端和后端。前端完成分析，包括词法分析、语法分析和语义分析。后端完成综合，综合出等价的目标语言程序。这时包括目标程序生成和代码优化。

编译程序各部分的功能可概括如下。

词法分析：编译程序首先对源程序进行词法分析。完成词法分析的部分称为词法分析程序，也称扫描程序。词法分析程序从左到右逐个字符地对源程序字符序列进行扫描（读入），根据所实现程序设计语言的词法规则识别出具有独立意义的各个最小语法单位（即符号或单词，如标识符、常量和关键字等），并把它们转换成属性字形式的内部中间表示，以便下一阶段语法分析程序使用。词法分析程序往往还完成其他工作，包括删除注释之类非必需信息、处理编译程序控制指示、把标识符登录到符号表及加工宏功能等各项预处理工作。

语法分析：编译程序中完成语法分析的部分称为语法分析程序，也称为识别程序。语法分析程序读入由词法分析程序生成的内部中间表示—属性字序列，根据所实现程序设计语言的语法规则，识别出各个语法成分，最终识别出程序。在进行语法分析时，识别出语法成分的同时也检查了语法的正确性。如果发现存在语法错误，则给出相应的出错信息，否则，从属性字序列生成另一种内部中间表示，如语法分析树与推导等。

语义分析：编译程序继语法分析之后的工作是语义分析。语义分析以语法分析程序的输出（语法分析树或其他内部中间表示）为基础进行。进行语义分析的同时进行静态语义检查，并可能生成目标程序。因为一个程序由数据结构和控制结构两部分组成，所以要分别对这两部分进行语义分析。对于数据结构，要检查所涉及的变量是否有定义、进行运算时其类型是否正确等，因此需确定类型，即确定标识符所代表数据对象的数据类型，进而检查运算分量类型的一致性和运算的合法性。对于控制结构，则根据程序设计语言所规定的语义，对它们进行相应的语义处理。这时可以生成相应的目标代码，例如对于加法运算，在检查了两个运算分量都有定义，且它们的类型一致（相容）能进行加法运算后，生成执行加法的目标代码。

为了改进目标程序的质量，这时也可能不生成目标程序，而是生成另一种内部中间表示，在代码优化阶段对这种内部中间表示进行优化，然后生成目标程序。语义分析阶段所生成的内部中间表示可以是抽象语法树，一般生成的是逆波兰表示、四元式序列或三元式序列。

概括起来，语义分析完成确定类型、类型检查、识别含义与相应的语义处理，以及其他一些静态语义检查等功能。

语义分析工作通常由一些语义子程序完成。

为了更系统地实现语义分析，能像语法分析那样更形式地表达语义分析，当前一般采用语法制导的翻译技术。

代码优化：为改进目标程序质量而在编译期间进行的各项工作称为代码优化。优化通常基于语义分析阶段生成的内部中间表示进行，把它变换成功能相同但功效更高的优化了的中间表示代码。

优化分为与机器无关的优化及与机器有关的优化两类，今后重点讨论的将是与机器无关的优化。除了对内部中间表示进行优化，还可以对目标语言代码进行优化，这通常是不太复杂因而代价不是很高的一类优化，一般称为窥孔优化。

由于源程序仅编译一次，而相应的目标程序可能运行无数次，因此进行优化十分必要，尤其是对程序量仅占整个程序很小比例，但运行时间却占整个程序运行时间之极大部分的循环结构，进行优化将使目标程序质量大幅度提高。目前常用的一些程序设计语言编译程序都具有很强的优化功能，C语言编译程序是典型的代表。

目标程序生成：目标程序可以在语义分析时生成，但为了改进目标代码质量，往往先生成中间表示代码，待优化之后再从优化了的中间代码生成目标程序。这就是编译程序之目标程序生成部分的工作。目标程序的生成与运行它的计算机密切相关。为了教学的目的，避免把大量的时间用在具体计算机细节的学习上，我们将基于一种虚拟机来讨论目标程序的生成。

2. 编译程序的构造

概括起来，一个编译程序由前端和后端组成，前端进行分析，完成词法分析、语法分析和语义分析。前端基本上与机器无关。后端进行综合，完成目标程序的生成和代码优化。后端一般与运行目标程序的计算机密切相关。相应于各个要完成的基本工作分别有词法分析程序（扫描程序）、语法分析程序（识别程序）、语义子程序、目标程序生成程序和代码优化程序。将这些程序有机地结合起来，从而完成对源程序的编译。

3. 遍的概念

一个编译程序看似简单，好像只需完成上述的五项基本工作。但对于一个内涵丰富、表达能力较强的高级程序设计语言，编译程序事实上是一个结构庞杂的程序系统，因此往往也称为编译系统。为了合理安排参加编译程序研制的人员，在计划完成期限内达到编译速度、目标程序运行速度、查错改错能力及调试功能等设计指标，通常把一个编译程序的工作分成若干阶段来完成，每一阶段都以源程序（第一阶段）或内部中间表示（上一阶段产生的输出结果）作为输入进行相应的处理，生成等价的内部中间表示作为输出。上一阶段的输出作为下一阶段的输入，它总应是有利于下一阶段的处理。每个阶段从头到尾读入整个输入并进行处理的过程称为遍（或趟）。

一般来说，不把所有的编译工作都放在唯一的一遍中，也不必把每个基本工作作为独立的一遍来实现，往往是进行适当的组合。例如，把词法分析作为第一遍，语法分析与语义分析作为第二遍，而目标程序生成与代码优化作为第三遍。

一个编译程序由几遍完成编译便称为几遍编译程序。如果对源程序从头到尾扫描一次便

完成词法分析、语法分析、语义分析、目标程序生成与代码优化等全部工作，这样的编译程序称为一遍编译程序。上面的例子为三遍编译程序。三遍编译程序的工作示意图如图1-2所示。第一遍的输入是源程序，相应的语言记为SL（Source Language），最后一遍的输出是目标程序，相应的语言记为TL（Target Language）。对于n遍编译程序，中间各遍之输出的内部中间表示相应的语言可记为L_1、L_2、…、L_{n-1}。

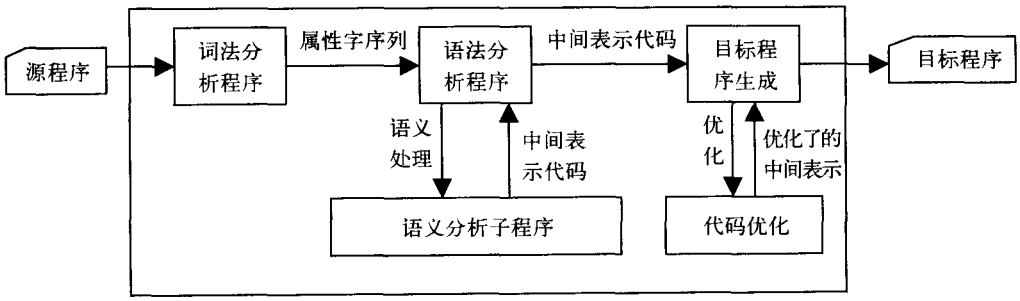


图 1-2

1.3 编译程序的分类

如同存在形形色色的程序设计语言一样，也存在着各种各样的编译程序。按照用途和侧重面，编译程序大致可分成如下几类。

诊断型编译程序：此类编译程序专门设计来帮助开发与调试程序。它们对源程序进行检查，当发现存在错误时往往能自动校正一些小错误。例如遗漏分号或括号等。对于只有在运行时刻才可能查出的错误，如不合法的下标、指针误用以及不合法的文件管理等，诊断型编译程序能在目标程序中生成可查出运行时刻错误并因运行时刻错误而放弃程序执行的代码。由于在目标程序中过多地包含了检查错误的代码，运行效率较低，因此，诊断型编译程序只在程序开发初期阶段使用。当程序接近完成或已完成时，其相应的目标程序中不应再包含诊断型代码，应使用“产品型”编译程序以提高运行速度。

优化型编译程序：此类编译程序用来产生高功效目标程序，功效指时空功效，即运行速度和存储空间。C语言编译程序是优化型编译程序的典型代表。由于在达到高功效时有很多因素是相互抵触的，一般是某种折衷或在某侧重面上的优化。例如，在一般情况下，往往要速度快就得多占用存储空间，反之，要想存储空间占用少，就得以时间为代价。又如对使用频率高的变量，让它们的值存放在寄存器中可减少存取时间，但在调用函数时需保护和恢复这些寄存器，又使函数的调用更费时，因此没有最优目标程序可言。优化型编译程序给出若干级优化供用户选择，使用户以合理的代价获得所期望的优化效果。

可重定目标型编译程序：通常为一个特定的程序设计语言而设计的编译程序，生成在一个特定目标机上运行的目标程序。换言之，此目标程序只能在该特定计算机上运行。当为另一型号计算机设计编译程序时，即使是对于同一个程序设计语言，原有编译程序也将不再适用而需重新开发。显然，只需重写与计算机相关的部分。与计算机无关部分不必重写。可重定目标型编译程序是不必重写此编译程序中与计算机无关部分，只需重写与一种目标计算机相关部分的程序，就可把目标计算机改变成那种计算机的编译程序。因此，可重定目标型编译程序是易移植的编译程序。

交叉型编译程序：通常运行由编译程序所生成的目标程序的计算机应与运行编译程序的

计算机具有相同的型号。当一个编译程序在一种型号计算机上运行，却生成另一种型号计算机上运行的目标程序时，这类编译程序称为交叉型编译程序。

增量型编译程序：当编译程序对一个源程序进行编译，发现源程序中存在错误时，便停止编译，由用户对源程序进行修改，然后从头开始重新编译。然而，如果修改处很少，那么全部重新编译就会造成计算机资源的浪费。增量型编译程序在发现源程序错误时，仅从源程序中修改处附近的正文开始重新编译，这样就节省了大量时间。显然，在集编辑、编译、运行与调试于一体的程序设计语言支持环境中采用增量型编译方式是十分合适的。

应用并行技术的编译程序：通常关于单处理器计算机系统，编译程序处理的是顺序程序设计语言程序，生成的是顺序地执行的目标程序。随着计算机技术的发展，出现越来越多的多处理器计算机，甚至多计算机系统，例如并行系统和分布式系统。相应地出现支持并行和通信的高级程序设计语言，典型的有并行PASCAL和ADA等。编译程序技术应能处理这样的语言，包括处理并行和通信成分，实现共享变量、消息传递和同步等。关于“应用并行技术”还有另一层含义，即在顺序程序中，自动寻找并行性，换句话说，编译程序将在原有的顺序程序中寻找并行执行的可能性。例如，关于数组，当对其一切元素赋以相同的初值时可并行进行。一般地，当数组元素可相互无关地独立计算时，就可能并行地处理。这样将大大提高程序运行效率。应用并行技术的编译程序通常作为研究生课程内容讨论。

1.4 实际应用中的编译程序

程序设计语言如C语言等，通常有标准版本，然而为这些程序设计语言开发的实际应用中的编译程序，往往并非“标准”的。这表现在以下两方面：预处理与非标准版本。

1. 预处理

在一个具体的源程序中，除了高级程序设计语言标准版本中所规定的语法成分外，往往还包含一些与具体实现有关的部分。例如，C语言程序中往往包含把其他文件上的内容安插到所指明位置处的包含（include）子句。另外还可能包含关于宏定义与条件编译的一些预处理命令，它们均以符号#开头，都不是C语言本身的成分，必须在预处理阶段进行处理，然后再进入编译的第一阶段进行词法分析。

2. 程序设计语言的非标准版本

一个编译程序理所当然地根据所实现程序设计语言的定义来实现，尤其是应按照标准版本来实现。然而现实并非总是如此。实现的往往是某种“方言”。究其原因，有如下几个方面。

第一种情况是某些符号的表示有所不同，例如乘幂运算，有的用**表示，有的用^表示。

第二种情况是编译程序仅实现程序设计语言的某个子集。尽管不同的编译程序实现的是同一个程序设计语言，但却是不同的子集，也就成为不同的版本。例如对于PASCAL语言，有的编译程序能处理上下界可变的（形式）数组，有的则不能。

第三种情况是程序结构有根本的变化。典型的例子还是PASCAL语言。PASCAL语言是嵌套分程序结构的程序设计语言。这样，一个完整的PASCAL程序必须存放在一个文件中。当程序较长时，这将带来很大的不便。为适应模块化，便于分块编译，允许把一个PASCAL程序分散在几个文件上。PASCAL方言版本Turbo PASCAL实现了多文件的模块化PASCAL语言，使得程序不再由单一的嵌套分程序结构组成，而是由多个编译单位组成，Turbo PASCAL提供了编译单位间相互联系的设施。

C语言编译程序还因实现细节而有不同的版本，其中一个突出的例子就是关于表达式的计算次序。例如，下列表达式：

(++k)+(++k)+(++k)

假定k的值是3，此表达式的值是15。但也有版本计算此表达式的结果是18。

因此，在使用编译程序时，应该首先了解它所实现的语言版本，了解它与标准版本的区别。有一些与标准版本的差异不能在相应程序设计语言说明书中反映出来，便只能通过实际应用编译程序来了解。

鉴于我们要讨论的是编译程序的基本构造原理和方法，对于程序设计语言尽可能针对标准版本（的子集）进行讨论。

本章小结

本章讨论程序设计语言与编译程序之间的联系，主要内容包括高级程序设计语言的定义、源程序的执行及编译程序的构造。

高级程序设计语言程序的执行通常有解释与翻译两种方式。把高级程序设计语言程序翻译成等价的低级语言目标程序的翻译程序称为编译程序。

由于高级程序设计语言程序归根结底是字符序列或基本符号序列，故编译程序实质上是一种符号处理工具。基于程序设计语言的特征，一个编译程序由前端和后端两部分组成，前端进行分析：词法分析、语法分析与语义分析，后端进行综合：目标程序生成与代码优化。

按照用途与侧重面，存在若干不同种类的编译程序。当前的编译程序往往是集编辑、编译、运行与调试于一体的高级程序设计语言集成支持环境。

第2章　编译程序构造基础知识

2.1　文法和语言

程序设计语言是什么？可以说，它是全部程序组成的集合，换言之，某种程序设计语言所能写出的全部程序组成了此程序设计语言。C语言就是全部C语言程序组成的集合。作为一个人工设计的符号语言，每种程序设计语言都有其自己的一组特定符号（或基本符号），以及由这些符号组成语法成分的规则（语法规则）。因此，一种程序设计语言的程序是该程序设计语言之符号集合上的、按一定规则组成的符号串。如何系统地构造程序？一个程序可以看成是一个形式地定义的语言的句子。本节从形式语言的角度来讨论句子（即程序）的生成，以及识别一个输入符号串是否是某文法的句子。

2.1.1　符号串和符号串集合

1. 符号串

程序是在程序设计语言之基本符号集上按照语法规则而构造的基本符号串，程序设计语言是程序的集合。因此，首先讨论符号串与符号串集合的概念。这里把程序设计语言的基本符号集称为字母表。

定义2.1　*字母表是非空的有穷符号集合。*

字母表包含一种语言中所允许出现的一切符号，当然其中至少要包含一个符号。不言而喻，不同的语言有不同的字母表。例如，对于C语言，字母表包含大小写英文字母和数字，以及+、－、%与{、}等字符。PASCAL语言的字母表与C语言的不同，例如它至少包含字符^。可以把允许出现在程序中的一切字符组成的集合作为一种语言的字母表，也可以把字母、数字、关键字等组成的基本符号集合看作一种语言的字母表。注意，一般情况下，字母表中的元素总是称为符号。

由于字母表是符号的有穷集合，通常在字母表的集合表示中明确写出一切符号，且用大写英文字母A、B、…，以及希腊字母Σ等来表示字母表。例如，字母表A={0，1}与字母表Σ={a，b，c}等。

字母表中的元素（即符号）组成符号串。

定义2.2　*符号串是由字母表中的符号组成的有穷序列。*

例2.1　0、1、01、10、100、…，即一切二进制数（可能包含前导0，例如01），是字母表B={0，1}上的符号串。字母表A={a，b，c}上符号串的例子是a、b、c、ab、ac、bc、aaa、…。

符号串总是建立在某个特定字母表上，且只由字母表上的有穷多个符号组成。注意，符号串中符号出现的顺序是重要的。例如，01与10及ab与ba都是不同的符号串。符号串通常用t、u、v、w与x等小写英文字母来表示，也可用希腊字母α与β等来表示。

允许有不包含任何符号的符号串，这种符号串称为空符号串，简称空串，用ε表示。

符号串x中所包含的符号个数称为符号串x的长度，用|x|表示。例如，符号串01001的长度是|01001|=5，而符号串 abc的长度是|abc|=3。显然，空串ε的长度是|ε|=0。

在以后的讨论中，经常要考虑一个符号串中若干个相继符号组成的部分，即子符号串，

其一般定义如下。

定义2.3 设有非空符号串u=xvy，其中符号串v≠ε，则称v为符号串u的子符号串。

例2.2 设字母表Σ={a，b，c，+，−，*，/，(，)}上的符号串t=a*(b + c)，则a、a*(、*(b+等都是t的子符号串，且长度分别是|a|=1、|a*(|=3与|*(b+|=4。

显然，对于定义2.3中的u与v，有|u|≥|v|>0。

当仅对一个符号串的某个子符号串感兴趣时，可以用省略形式来表示。例如，对于符号串z，z=x…表示只对z开始的子符号串x感兴趣；z=…x表示只对z尾部的子符号串x感兴趣；而z=…x…表示只对符号串z中某处出现的子符号串x感兴趣。当x中仅包含一个符号T时，可分别写作：

$$z=T\cdots \quad z=\cdots T\cdots \quad z=\cdots T$$

为了能生成一个字母表上的一切符号串，可以对符号串进行运算。下面引进联结（或并置）与方幂运算。

定义2.4 设x与y是同一个字母表上的两个符号串，把y的各个符号相继写在x的符号之后所得到的符号串称为x和y的联结（或并置），记为xy。

例2.3 设在字母表{a，b，c}上有符号串x=abc与y=ca，则x与y的联结z=xy=abcca，这里|x|=3、|y|=2，|z|=5。易见|xy|=|x|+|y|。显然，对于字母表上的任何符号串x，有εx=xε=x。

当把某符号串相继地重复写若干次时便得到该符号串的方幂。方幂一般定义如下。

定义2.5 设x是某字母表上的符号串，把x自身联结n次，即z=xx…x (n个x)，称z为符号串x的n次方幂，记为$z=x^n$。

例如，对应于n=1、2、3，分别有$x^1=x$、$x^2=xx$ 与$x^3=xxx$。注意，当n=0时，$x^0=\varepsilon$，这时$|x^0|=0$。

例2.4 设字母表{a，b，c}上有x=abc，则$x^0=\varepsilon$，$x^3=abcabcabc$。

显然，$x^n=xx^{n-1}=x^{n-1}x$，且$|x^n|=n|x|$。于是，当|x|=1时，$|x^n|=n|x|=n$。

2. 符号串集合

定义2.6 若集合A中的一切元素都是某字母表Σ上的符号串，则称A为该字母表Σ上的符号串集合。

字母表上的符号串集合通常用大写英文字母A、B、C、…表示。例如，用B表示字母表Σ={0，1}上的符号串集合，即二进制数集合。

符号串集合作为一种集合，可以用一般集合的表示法来表示，即可用枚举法、省略法和描述法来表示。用枚举法表示时是在集合中明确写出（枚举出）符号串集合中的全部元素，例如{1，11，111，1111}。当不可能或不便于穷尽一切元素时采用省略法，例如{1，11，111，1111，…}。采用描述法来刻画一个符号串集合时，必须指明该集合中一切元素所应满足的条件，即{x|x满足条件C}。例如，{x|x全由1组成，且1≤|x|≤10}与$\{1^i \mid i\geq 1\}$。注意，后者中的1^i是一种形式表示，表示符号串1的i次方幂，它不涉及含义。

当把字母表本身看成该字母表上的符号串集合时，由于一切符号串的长度都是1，因此可以写出：

字母表={x|x是该字母表上的符号串，且|x|=1}

当把C语言基本符号集看作字母表时，C语言是该字母表上的某个符号串集合。

如果某个符号串集合中不包含任何元素，即是一个空集，那么沿用通常的空集表示法，用∅表示。