郑国梁 徐永森编著

计算机的编译方法

计算机的编译方法

郑国樑　徐永森　编著

人民邮电出版社

内 容 简 介

编译程序是计算机软件的基本组成部分之一，是计算机软件人员必须掌握的知识。本书较为全面地介绍了编译程序的基本概念、主要内容和实现方法。全书共分十二章。首先叙述了基本概念和有关的数据结构。接着介绍了编译程序的主要内容：词法分析，语法语义分析，优化，存贮分配，运行子程序。最后四章介绍解释程序的结构和实现方法，源程序中的静态语法错误和动态逻辑错误的检查和修改方法，编译自动化，设计和实现编译程序时应该注意的问题。

本书可作为大专院校计算机专业的编译方程课程的教材，也可供计算机软件人员学习参考。

计 算 机 的 编 译 方 法

郑国樑 徐永森 编著

*

人 民 邮 电 出 版 社 出 版

北京东长安街 27 号

北 京 印 刷 一 厂 印 刷

新华书店北京发行所发行

各 地 新 华 书 店 经 售

*

开本：787×1092 1/32　　1982 年 3 月第一版

印张：10 28/32 页数：174　1982 年 3 月北京第一次印刷

字数：249 千字　　印数：1—13,000 册

统一书号：15045·总 2565—有 5238

定价：1.10 元

序　言

本书对编译程序中有关方面的内容和方法作了介绍。可以作为一个学期的编译方法课程的教材，也可以供计算机科学领域中的新手自学或专业人员参考。我们力求使本书深入浅出平易好懂，便于初学者学习。我们还力求使初学者不但能对编译程序有一个一般的了解，而且能对编译程序有较为具体的认识。希望初学者在通过对本书的学习后，能够具有编制编译程序的能力。当然，这种能力单纯依靠学习一本书是不能真正得到的，还必须要一边学习一边在机器上实际动手才能真正达到。这是因为编制编译程序的工作是一种软件工程，实践性强，不通过实践就不能真正掌握。

学习本书之前应该具备有 ALGOL 60 等程序设计语言的知识以及使用程序设计语言编写程序和算题的经验。本书中很多算法都是用类似于 ALGOL 的语言来描述的。这样描述比较简捷清楚，有利于读者的学习。如果有的读者认为不直观的话，也可以自己把它改写成流程图的形式。这种改写工作对初学者是有益的。

本书没有涉及形式语言理论方面的内容。这不是因为形式语言理论不重要。恰好相反，正是考虑到形式语言理论的重要性，我们认为把它与编译方法分开来独立地进行介绍更为合适。这样做的好处是既可使形式语言理论及其在编译程序上的应用讲得更深更透一点，又可使初学者开始时不必马上接触到形式语言中那些深奥难懂的理论知识，从而更有利于学习。虽然这

样做也会带来一些缺点，会使不少语法分析的方法不易讲深讲透。对此，我们采取了一些补救措施，但还不能解决根本问题。这只有通过学习形式语言理论才能得到解决。我们教学实践证明，先学编译方法再学习形式语言理论，既可使学生学习编译程序时不产生太大困难，又可使学生对形式语言理论有更深的理解。

本书是在南京大学计算机科学系软件教研室、研究室编写的编译方法教材基础之上改编而成的。在改编过程中自始至终都得到该教研室、研究室的同志的关心和帮助。特别是徐家福教授对本书的编写作了许多具体的指导，他和张幸儿、钱士钧、金志权、陈佩佩四位同志一起对书稿作了审阅。另外，张小兰、胡又林、高海燕等同志为本书做了大量抄写工作。在此，我们对他们以及教研室、研究室的全体同志表示衷心感谢。

限于我们的水平，书中一定存在不少错误和不足之处，欢迎读者批评指正。

编者 1981 年 1 月于南京

目　　录

第一章 总 论

§ 1.1 术 语

在计算机领域里，通常人们把计算机可以直接执行的代码形式指令系统称为**机器语言**；把计算机的符号形式的指令系统称为**汇编语言**(或称为**符号汇编语言**)；把诸如 ALGOL，FORTRAN，COBOL 等语言称为**高级程序设计语言**（简称为**高级语言**）。机器语言和汇编语言都是与特定的计算机有关的语言，高级程序设计语言一般都是与具体计算机无关的语言。

大家知道，用高级程序设计语言编写程序要比用汇编语言或机器语言编写程序方便容易得多。

但是计算机不懂高级语言，也不懂本台机器的汇编语言，而只懂本台机器的机器语言，所以计算机无法直接执行用高级语言（或汇编语言）编写的程序。如果要在计算机上执行用高级语言（或汇编语言）编写的程序，就必须通过特定的途径来进行。通常有两种途径，一种是所谓编译的途径，另一种是所谓解释的途径。

1.1.1 编译

编译这条途径是一种分阶段进行的途径。一般说来，首先进行“翻译”，把用高级语言（或汇编语言）编写的程序翻译成与之等价的用机器语言书写的程序，然后对翻译出来的程序进行运行计算。前一阶段的翻译工作由翻译程序完成，后一阶段的运行计算需要有运行程序来配合完成。

所谓**翻译程序**是指这样一种程序，它能够把用甲语言编写的程序翻译成与之等价的用乙语言书写的程序。甲语言称为该翻译程序的**源语言**，乙语言称为该翻译程序的**目标语言**（或称**结果语言**），用源语言写的程序称为**源程序**，与源程序等价的用目标语言书写的程序称为**目标程序**（或称为**结果程序**）。

如果源语言是某一高级语言，目标语言是某台计算机的汇编语言或机器语言，这种翻译程序特称为**编译程序**。如果源语言是一汇编语言，目标语言是某台计算机的机器语言，这种翻译程序特称为**汇编程序**。用机器语言构成的目标程序又称为**目标代码程序**或简称为**代码程序**，有时称为**目标代码**或**结果代码**。

所谓**运行程序**是指运行目标代码程序时必须配置的各种子程序的全体。这些子程序将在第八章介绍。

综上所述，我们可以把编译途径归纳成三点：

1．编译程序是与源语言以及计算机有关的概念。任何一个具体的编译程序都是某一特定机器上的关于某一特定语言的编译程序。我们用 $T_{M,L}$ 表示 M 机器上的关于 L 语言的编译程序。例如 $T_{200,\ \mathrm{ALGOL}}$ 表示 DJS-200 系列机上的 ALGOL 语言的编译程序。显然，$T_{M1,L}$ 与 $T_{M2,L}$ 是两个不同的编译程序，是两台不同计算机上同一种语言的编译程序，$T_{M,L1}$ 与 $T_{M,L2}$ 也是两个不同的编译程序，是同一台计算机上两个不同语言的编译程序。当既不强调机器，也不强调语言时，我们就用 T 表示编译程序。

2．对编译程序而言，源程序是被加工的对象，目标程序是加工后的结果，这就是说，源程序是编译程序的输入数据，目标程序是输出结果。从数学上看，一个编译程序 T 是一个映射函数，它把源程序 P 映射成目标程序 Q：

$$Q=T(P)$$

3．源程序的执行需要分成两大阶段：编译阶段和运行阶段(见图 1-1)。

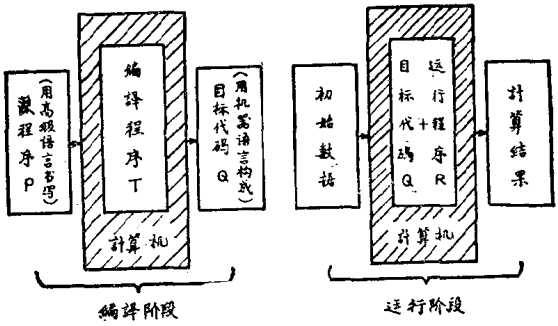


图 1-1　分成两个阶段的源程序的执行历程图

如果编译阶段生成的目标程序不是机器代码程序，而是符号汇编指令程序，那么，源程序的执行历程就必须分成三大阶段：编译阶段，汇编阶段和运行阶段（见图 1-2）。

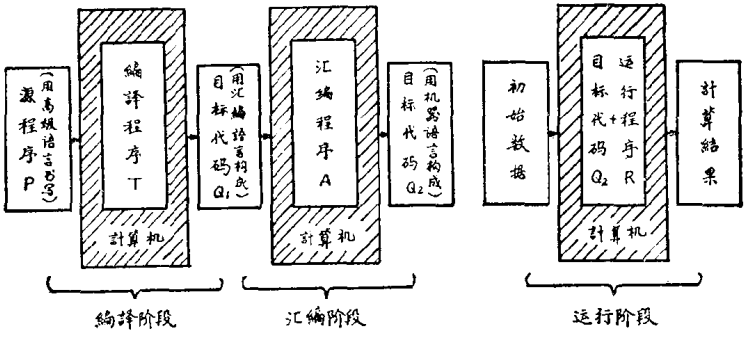


图 1-2　分成三个阶段的源程序的执行历程图

1.1.2　解释

源程序的另一种执行途径是解释途径。解释与编译的根本

区别在于：编译把源程序的执行进程划分成两大阶段，即编译阶段和运行阶段。解释却不然，它把这两个阶段合并成为一个阶段，称为解释执行阶段。具体地说，解释是这样一种途径，它将按源程序中语句的动态顺序，逐句地进行分析解释，并立即予以执行。这个解释执行的工作是由解释程序来完成的。这就是说，**解释程序**能够按源程序的动态顺序逐句地进行分析解释并予以执行，直至结束(参见图 1-3)。

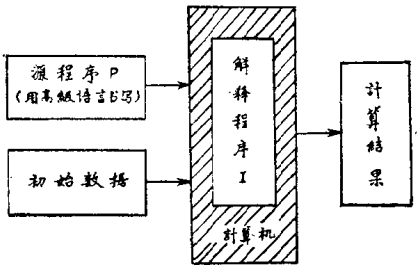


图 1-3　解释执行的历程图

本书重点讨论编译程序的构造，但是对解释程序也作了详细的介绍。在我们的讨论中源语言是 ALGOL 60 或类似于 ALGOL 的语言，目标语言则是一种假想的汇编语言。书中很多算法都是用类似 ALGOL 的语言来描述的，但都不是完整的 ALGOL 程序，其中用了很多 ALGOL 的基本符号之外的符号（如 λ、π 等），甚至还用了汉语语句，许多说明也都省略未写。

§ 1.2　编译程序的结构

为了把源程序翻译成与之等价的目标程序，编译程序需要做五方面工作。

第一、词法分析。这个部分的主要任务是分析由字符组成的源程序，把它们识别为一个一个的具有独立意义的“单词”，并识别出与其有关的属性(如是标识符、是界限符还是数等等属

性），再转换成长度上统一的标准形式，这种统一的标准形式既刻划了单词本身，又刻划了它所具有的属性，称为**属性字**、或称为**编排单位**，以供其它部分分析使用。

第二、语法分析。这个部分的主要任务是根据语言的语法规则逐一地分析词法分析时得到的属性字，以确定它们是怎样组成说明和语句的。说明和语句是怎样组成程序的。分析时如发现有不合语法规则的地方，便把这些出错的地方及错误性质打印输出报告算题人员；如没有语法错误，则给出正确的语法结构，以供以后部分分析使用。

第三、语义分析。这部分的主要任务是根据语法结构分析其含义，并用一种内部形式表示出来，或者直接用目标语言表示出来。

第四、代码生成。如果语义分析时把源程序表示成内部形式而不是表示成目标指令，则由本部分完成从内部形式到目标指令的生成工作。如果语义分析时，已直接生成目标指令，则无需做代码生成工作。

第五、优化。这是为了提高目标程序的质量而进行的工作。

下面，我们用一个例子来对这些方面的工作作些具体的说明。

例 1.1 计算长为 l、外径为 D、内径为 d、比重为 e 的钢管的重量 M

我们知道 M 的计算公式是：

$$M=\left(\pi\frac{D^2}{4}-\pi\frac{d^2}{4}\right)\cdot l\cdot e$$

程序是：

begin

comment 因为D, d, l, e, 未赋初值，所以这个程序是无法执行的；

real D, d, l, e, M;

$M:=3.14159/4*(D*D-d*d)*l*e$

end

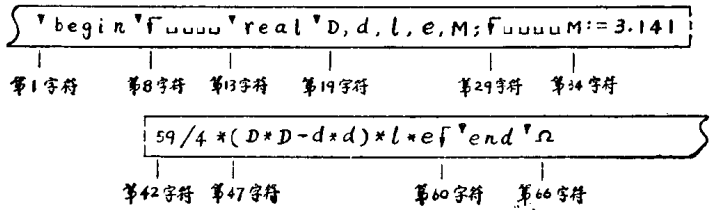
为了讨论方便起见，我们约定

1．将一个程序穿孔时，注解一律不穿孔。这个限制是合理的，因为注解部分对编译程序无任何作用。即使对注解部分穿孔，编译程序也应掠过不管。

2．穿孔时，粗体字一律用引号括起来。例如 **begin** 将穿孔成'begin'，这个约定应看成是对 ALGOL 语言给出一种具体的硬件表示。ALGOL 中的粗体字的硬件表示可以有各种不同的形式。大体上有三种，一是用引号括起来，二是在下面或后面划一横线表示，(例如 **begin** 将穿孔成b-e-g-i-n-)，三是用一般的标识符形式来表示（例如 **begin** 穿孔成 begin)，并规定这些标识符都具有特定的含义，不得作为变量标识符或过程标识符等使用。通常称这类标识符为保留字。粗体字的表示方式直接影响到编译程序的处理方式，但是不是本质上的影响。

3．纸带上的空格符、回车换行符和停码书写时分别用␣，↵和 Ω 表示。

这样，按此约定穿孔，纸带将呈下面的形式：



从机器角度看，这是一个由 66 个字符组成的 字符串。编译程序处理这个程序时要做下面四方面的工作。

第一、词法分析

词法分析将把这个由 66 个字符组成的字符 串分析 成下面的结果：

界限符	**begin**	的属性字
界限符	**real**	的属性字
标识符	*D*	的属性字
界限符	,	的属性字
标识符	*d*	的属性字
界限符	,	的属性字
标识符	*l*	的属性字
界限符	,	的属性字
标识符	*e*	的属性字
界限符	,	的属性字
标识符	*M*	的属性字
界限符	;	的属性字
标识符	*M*	的属性字
界限符	: =	的属性字
实数	3.14159	的属性字
运算符	/	的属性字
整数	4	的属性字
运算符	*	的属性字
界限符	(	的属性字
标识符	*D*	的属性字
运算符	*	的属性字
标识符	*D*	的属性字
运算符	—	的属性字
标识符	*d*	的属性字
运算符	*	的属性字
标识符	*d*	的属性字
界限符	)	的属性字

运算符	*	的属性字
标识符	l	的属性字
运算符	*	的属性字
标识符	e	的属性字
界限符	**end**	的属性字

第二、语法分析

根据 ALGOL 60 的语法规则，对词法分析后的程序进行语法分析，其结果可得图 1-4 所示的语法结构：

这里，我们对图 1-4 作两点解说

解说 1．图中的单线箭头“→”可读作“直接归约为”，双线箭头“⇒”可读作“归约为”。它们表示“根据语法规则，箭头左方的语法成分可归约为箭头右方的语法成分”。例如时刻 t_1 处的“M⇒〈简单变量〉”，其意为“因为〈简单变量〉::=〈变量标识符〉和〈变量标识符〉::=〈标识符〉，故根据这些语法规则，标识符 M 可归约为〈简单变量〉”。又例如，时刻 t_{12} 处的“〈项〉〈乘法运算符〉〈因式〉→〈项〉”，其意为“根据语法规则〈项〉::=〈项〉〈乘法运算符〉〈因式〉，故〈项〉〈乘法运算符〉〈因式〉可直接归约为〈项〉”。

解说 2．分析时刻 $t_1,t_2,\cdots,t_{28}$ 表示生成诸归约式的先后次序。例如首先生成归约式 M⇒〈简单变量〉，最后生成归约式〈分程序〉→〈程序〉。

显然，经过语法分析便可得知，例 1-1 这个源程序是一个语法上正确的程序，而且得到了这个程序的语法结构。问题是怎样进行图 1-4 所示的语法分析。实现上述语法分析的方法很多，我们将在第四，第五两章中叙述。

第三、语义分析和代码生成