



新编高等院校计算机科学与技术规划教材

(第2版)

编译原理与技术

BIANYI YUANLI YU JISHU

主编 李劲华 陈宇 丁洁玉



北京邮电大学出版社
www.buptpress.com

014033466

TP314
70-2

新编高等院校计算机科学与技术规划教材

编译原理与技术

(第2版)

主 编 李劲华 陈 宇 丁洁玉



北京邮电大学出版社



北航

C1722003

TP314
70-2

332880310

内 容 简 介

本书介绍了计算机高级语言编译程序的基本原理和技术,主要包括词法分析、语法分析、语法制导翻译的语义分析与中间代码生成、符号表与运行时存储空间的组织、代码优化以及目标代码的生成。本书着重描述了编译构造的一些基础理论,如形式语言、有限自动机和属性文法。从构造编译程序的技术角度,描述了编译程序的各类算法,以及编译程序的自动构造工具,如词法分析生成器 Lex 和语法分析生成器 YACC。

本书系统性较强,基本概念阐述清晰,通俗易懂,便于阅读,可作为普通高等院校计算机学科及相关专业的本科教材,也可供教师、研究生及有关专业人员学习和参考。

图书在版编目(CIP)数据

编译原理与技术 / 李劲华,陈宇,丁洁玉主编. -- 2 版. -- 北京:北京邮电大学出版社,2014.2
ISBN 978-7-5635-3770-9

I. ①编… II. ①李…②陈…③丁… III. ①编译程序—程序设计 IV. ①TP314

中国版本图书馆 CIP 数据核字(2013)第 281271 号

书 名: 编译原理与技术(第 2 版)

著作责任者: 李劲华 陈 宇 丁洁玉 主编

责任编辑: 张珊珊

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(邮编: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: 北京源海印刷有限责任公司

开 本: 787 mm×1 092 mm 1/16

印 张: 20.5

字 数: 512 千字

印 数: 1—3 000 册

版 次: 2006 年 1 月第 1 版 2014 年 2 月第 2 版 2014 年 2 月第 1 次印刷

ISBN 978-7-5635-3770-9

定 价: 42.00 元

· 如有印装质量问题,请与北京邮电大学出版社发行部联系 ·

前 言

编译程序是计算机的核心系统软件之一,属于 ACM/IEEE Computing Curriculum 2004 的核心知识域,是掌握计算机理论和软件技术的关键知识。编译原理和技术为人们理解计算机程序语言、创造优秀的软件奠定了理论基础,拓宽了视野,开辟了捷径。不妨浏览一下计算机的历史,就会发现很多被誉为程序设计大师的人无一不是编译领域的泰斗:创建了 Pascal 语言及其系列的图灵奖获得者 Niklaus Wirth 教授;写出第一个微型机上运行的 Basic 语言的 Bill Gates;成为 Sun 公司副总裁的 Java 缔造者 James Gosling 博士;C++ 之父 Bjarne Stroustrup 教授;被誉为“世界上最优秀的程序员”的 Delphi 架构师和 C# 的缔造者 Anders Hejlsberg 等。编译领域堪为计算机程序设计英才的沃土。

编译的原理和技术可以应用在其他诸如软件建模语言、硬件描述语言、脚本语言等的翻译方面;在集成化软件开发环境以及软件安全、软件工程和软件逆向/再向工程中一直有着广泛的应用。而且,编译理论的研究有力地推动了计算机科学、计算机工程、软件开发以及人机工程等领域的研究和发展。

本书介绍了计算机程序语言编译程序的基本原理、设计方法和主要实现技术,可以作为普通高等学校计算机科学与技术及相关专业的教材和参考书。

本书系统性较强,基本概念阐述清晰,文字通俗易懂,便于读者学习和掌握。和其他同类教材相比,本书的特点如下:

(1) 在内容上增加了对面向对象语言一些特性的处理,讨论了 C++、C# 和 Java 语言的例子。

(2) 对多数编译教材中以集合为主要数据结构的抽象算法描述进行了改进,更加详尽、深入浅出地描述了编译中的主要算法,以便于读者的阅读理解和计算机的实现。例如在自底向上的 LR 分析器中,本书采用了图的深度优先策略,以增量的方式构造出识别活前缀的有限自动机。

本书在组织上也进行了新的尝试,力图保持知识的逻辑性和连贯性,同时减少读者的阅读和理解的难度。

在第 1 章中概述了编译,以后各章按照编译程序的构成和编译过程的顺序,



逐步介绍编译的基本原理、设计方法和构造技术,把读者的思路和精力保持在编译程序的构造上,强调对编译原理和技术的宏观理解和全局把握,按照需要和逻辑关系阐述和讲解抽象的基础概念和理论;第2章首先介绍词法分析的设计和词法分析程序的手工构造,然后讲述有限状态自动机的理论以及它在词法扫描器自动生成中的应用;第3章集中讲解描述计算机编程语言的形式化语言,包括上下文无关文法的基本概念和等价变换;第4章介绍自顶向下语法分析方法,包括递归下降分析和表驱动的LL(1)分析;第5章讨论自底向上的算符优先分析方法、各种类型的LR分析方法及其语法分析的自动生成。为了便于理解语义分析和代码生成,本书在第6章介绍了编译程序符号表的组织与管理;第7章讨论编译构造所需要的程序运行时的环境,包括运行时的内存分配和手工与自动化的管理;在第8章里对语义描述技术、属性文法以及语法制导的语义分析进行了详尽的阐述;第9章讨论了基于语法制导技术的中间代码翻译;第10章论述了目标代码生成的原理和技术;最后,在第11章集中介绍代码优化的基本技术,主要包括中间代码的局部优化和目标代码的优化方法。

本书每章都附有各种类型的练习题,便于读者理解基本概念和原理,掌握编译的基本算法和实现技术。

编译知识理论抽象,算法丰富,建议学习时配合上机实践,在加深对编译原理理解的同时,掌握编译程序的基本技术。编译中有很多算法,比如NFA到DFA的转换、求首符集、计算LR项集等,都可以作为上机实践题。

与本书配套的电子教案、教学计划以及部分课外阅读和学习材料可在北京邮电大学出版社网站上下载。本书另有配套的习题解答和上机指导教材。

使用本教材要求读者学习和掌握高级程序设计语言,如C、Pascal或Java,最好具备离散数学、数据结构、计算机组成和汇编语言的基本知识。

本书结合作者李劲华多年的科研工作和教学实践编写,主要参考了参考文献[1]和[2]以及国内外许多学者的研究成果,在此表示诚挚感谢。陈宇参与了第2版全书的编辑,并改写了第3章和第4章的内容。

由于编者时间紧迫、水平有限,书中难免存在一些疏误和缺点,恳请广大读者批评指正。

编者

目 录

第 1 章 概论	1
1.1 为什么学习编译	1
1.2 什么叫编译程序	2
1.3 编译过程概述	3
1.3.1 词法分析	3
1.3.2 语法分析	4
1.3.3 语义分析和中间代码生成	4
1.3.4 代码优化	5
1.3.5 目标代码生成	5
1.4 编译程序的构成	6
1.4.1 基本功能模块	7
1.4.2 符号表的组织与管理	7
1.4.3 错误诊断和报告	8
1.5 其他与编译有关的概念和技术	8
1.5.1 遍的概念	8
1.5.2 编译的前端和后端	9
1.5.3 编译程序的分类	9
1.5.4 编译技术和软件工具	10
1.6 如何开发编译程序	11
1.6.1 编译程序的自展技术	11
1.6.2 编译程序的移植技术	11
1.6.3 编译程序的自动生成技术	12
1.7 编译系统以及其他相关程序	12
练习 1	14
第 2 章 词法分析	15
2.1 词法分析器的设计	15
2.1.1 词法分析器的功能与输出	15
2.1.2 词法扫描器与符号表	17
2.1.3 词法分析器的两种实现模式	17
2.1.4 词法错误的处理	17





2.2 词法分析器的一种手工实现	18
2.2.1 输入的预处理	18
2.2.2 超前搜索和最长匹配	19
2.2.3 状态转换图	19
2.2.4 基于状态转换图的词法分析器的实现	22
2.3 正规表达式	25
2.3.1 符号、符号串与符号集合	26
2.3.2 正规式与正规集	27
2.3.3 扩展的正规式	28
2.4 有限自动机	29
2.4.1 确定的有限自动机	30
2.4.2 不确定的有限自动机 NFA	32
2.4.3 从 NFA 到 DFA 的等价变换	33
2.4.4 DFA 的最小化	36
2.4.5 从正规式到有限自动机	38
2.4.6 有限自动机在计算机中的表示	41
2.5 词法分析的自动生成器 Lex	42
2.5.1 Lex 概述	42
2.5.2 Lex 的语言与实现	43
练习 2	45
第 3 章 程序语言的语法描述	48
3.1 文法和语言	49
3.1.1 文法的形式定义	50
3.1.2 推导与归约	51
3.1.3 分析树与语法树	52
3.1.4 文法产生的语言	54
3.1.5 语言的验证	55
3.1.6 语言的文法表达	56
3.1.7 文法的二义性	58
3.1.8 BNF 与 EBNF	61
3.2 文法的分类	63
3.2.1 0 型文法	63
3.2.2 1 型文法	63
3.2.3 2 型文法——上下文无关文法	64
3.2.4 3 型文法	65
3.3 文法的等价变换	67
3.3.1 文法等价的概念	67
3.3.2 增广文法	67



3.3.3 提取左因子	68
3.3.4 消除左递归	68
3.3.5 对文法的使用限制	70
3.4 语法分析概述	70
3.4.1 自顶向下的语法分析	70
3.4.2 自底向上的语法分析	71
3.4.3 语法分析的基本问题	72
练习 3	73
第 4 章 自顶向下的语法分析	77
4.1 自顶向下语法分析的一般方法	77
4.2 LL(1)文法及其分析	78
4.2.1 首符集 FIRST	80
4.2.2 后继符集 FOLLOW	82
4.2.3 选择集 SELECT	84
4.2.4 LL(1)文法	85
4.2.5 LL(1)文法的分析	87
4.3 递归下降分析技术	88
4.3.1 递归下降分析器的设计	88
4.3.2 从 EBNF 构造递归下降分析器	92
4.3.3 递归下降分析的特点	93
4.4 预测分析技术	93
4.4.1 预测分析程序的工作过程	93
4.4.2 预测分析表的构造	95
4.5 LL(1)分析中的错误处理	98
练习 4	100
第 5 章 自底向上的语法分析	102
5.1 自底向上语法分析概述	102
5.1.1 自底向上语法分析器的体系结构	102
5.1.2 规范归约和算符优先归约	103
5.1.3 短语、句柄和最左素短语	105
5.2 算符优先分析方法	106
5.2.1 算符优先文法	107
5.2.2 算符优先关系的构造	109
5.2.3 算符优先分析算法	110
5.2.4 算符优先函数及其构造	112
5.3 LR 分析方法	114
5.3.1 LR 分析概述	114





5.3.2	LR(0)分析表的构造	118
5.3.3	SLR 分析表的构造	124
5.3.4	规范 LR 分析表的构造	129
5.3.5	LALR 分析表的构造	134
5.3.6	LR 分析方法小结	136
5.4	LALR 分析器的生成工具 YACC	141
5.4.1	YACC 概述	141
5.4.2	YACC 源程序	142
5.4.3	YACC 解决二义性和冲突的方法	144
5.4.4	YACC 对语法分析中的错误处理	145
练习 5		146
第 6 章	符号表的组织和管理	150
6.1	符号表的作用	150
6.2	符号表的主要属性及其作用	151
6.3	符号表的组织结构	154
6.3.1	符号表的整体组织结构	155
6.3.2	关键码域的组织	156
6.3.3	不等长域的组织	157
6.3.4	符号表的操作与符号表项的组织	158
6.4	名字的作用范围	161
6.4.1	名字的声明	161
6.4.2	块结构与符号表的分层次管理	162
6.4.3	静态作用域和动态作用域	164
练习 6		165
第 7 章	运行时环境	167
7.1	程序运行的基本概念	167
7.1.1	过程及其活动	167
7.1.2	活动记录	169
7.1.3	调用序列和返回序列	169
7.1.4	活动树	170
7.1.5	环境和名字的绑定	170
7.2	参数传递机制	171
7.2.1	按值调用	171
7.2.2	引用调用	172
7.2.3	值-结果调用	172
7.2.4	换名调用	173
7.3	运行时存储空间的组织和管理	174



7.3.1 局部数据的存放	174
7.3.2 运行时存储空间的划分	175
7.3.3 存储分配策略	176
7.4 静态运行时环境	176
7.5 栈式运行时环境	178
7.5.1 无过程嵌套的栈式运行时环境	178
7.5.2 有过程嵌套的栈式运行时环境	182
7.6 堆式运行时环境	186
7.6.1 堆式动态存储分配的实现	187
7.6.2 堆的自动管理	188
7.7 面向对象语言的运行时环境	191
7.7.1 面向对象语言的动态存储管理	191
7.7.2 Java 运行时环境	193
练习 7	194
第 8 章 属性文法和语义分析	199
8.1 语义分析概况	199
8.2 属性与属性文法	201
8.2.1 属性的引入	201
8.2.2 属性文法的定义	202
8.2.3 属性文法的扩展与简化	206
8.3 属性的计算	208
8.3.1 属性依赖图和计算顺序	208
8.3.2 综合属性和继承属性及其计算	212
8.3.3 语法分析的同时计算属性	216
8.4 数据类型与类型检查	224
8.4.1 类型表达式与类型构造器	225
8.4.2 类型等价	227
8.4.3 类型检查	230
8.4.4 类型转换	231
8.4.5 类型检查的其他问题	233
练习 8	234
第 9 章 语法制导的中间代码翻译	237
9.1 中间语言	237
9.1.1 后缀式	239
9.1.2 图形表示	241
9.1.3 字节代码	242
9.1.4 三地址代码及其四元式实现	243





9.2 声明语句的翻译	246
9.2.1 过程中的声明	246
9.2.2 保留声明的作用域信息	247
9.2.3 记录中的域名	251
9.3 赋值语句的翻译	251
9.3.1 简单算术表达式及赋值语句	252
9.3.2 数组元素的引用	254
9.3.3 记录和指针的引用	259
9.3.4 类型转换	260
9.4 基本控制结构的翻译	261
9.4.1 布尔表达式的翻译	261
9.4.2 控制流语句的多趟翻译模式	264
9.4.3 回填技术基础	266
9.4.4 控制流语句的单趟翻译模式	269
9.5 转向语句的翻译	272
9.5.1 标号语句与 goto 语句的翻译	272
9.5.2 出口语句的翻译	273
9.5.3 开关语句的翻译	274
9.5.4 过程调用的翻译	276
练习 9	277
第 10 章 目标代码生成	279
10.1 代码生成器设计的基本问题	279
10.1.1 目标程序	280
10.1.2 指令选择	280
10.1.3 寄存器分配	280
10.1.4 计算顺序的选择	281
10.2 虚拟计算机模型	281
10.3 语法制导的目标代码生成	282
10.4 基本块和待用信息	285
10.4.1 基本块及其构造	285
10.4.2 流图	287
10.4.3 待用信息	288
10.5 一个简单代码生成器	290
10.5.1 寄存器和地址的描述	291
10.5.2 寄存器的分配原则与选择算法	291
10.5.3 代码生成算法	292
10.5.4 其他三地址语句的目标代码	294
练习 10	295



第 11 章 代码优化	297
11.1 代码优化的概念	297
11.2 代码优化的基本技术	299
11.2.1 删除公共子表达式	299
11.2.2 复写传播	301
11.2.3 删除无用代码	301
11.2.4 代码外提	301
11.2.5 强度消弱和删除归纳变量	302
11.3 局部优化	303
11.3.1 基本块的变换	303
11.3.2 基本块的 DAG 实现	304
11.3.3 基于 DAG 的局部优化	307
11.4 机器代码优化——窥孔技术	309
11.4.1 冗余存取的删除	310
11.4.2 不可达代码的删除	310
11.4.3 控制流优化	310
11.4.4 代数化简与强度消弱	311
11.4.5 特殊指令的使用	312
11.5 代码优化的高级技术简介	312
练习 11	313
参考文献	315



第1章 概 论

1.1 为什么学习编译

编译程序构造的原理和技术一直属于最近公布的 ACM/IEEE Computing Curriculum 2004 的核心知识域,已成为计算机科学必备的专业基础知识。而且,编译程序的构造是计算机科学中一个非常成功的分支,也是最早获得成功的分支之一,它所建立的理论、技术和方法值得人们深入研究和學習。

第一,编译构造正确地建立了研究的问题领域和研究方式:分析输入内容、构造一个语义表示并合成输出。对于不同的源语言,可以用一个单一的语义表示产生出不同的目标语言,运行在不同的环境中。而且,编译构造可以划分成便于控制和管理阶段,每个阶段的工作结果正好对应编译程序的子系统或模块。编译程序的这种分析-合成模式以及解释程序的解释模式已经成为软件开发领域最成功的设计模式和软件架构,在软件开发中获得了广泛的应用。

第二,针对编译程序构造的某些部分已经开发了标准的形式化技术,依据这些形式化技术所研制的编译程序生成工具,极大地减轻了编译程序的构造工作,使得编译程序成为计算机系统最可靠的基础软件之一。这些形式化技术包括有限自动机理论、上下文无关文法、正规表达式、属性文法、机器代码描述、数据流分析方程式等。编译程序自动生成技术具有的高效性、正确性、灵活性、易更改性和易扩展性等优点,也扩大到普通的软件开发领域,例如,数据库程序的自动产生器以及从图形化的 UML 自动生成高级语言的程序。

第三,编译程序包含许多普遍使用的数据结构和算法,例如散列法(哈希算法)、栈机制、堆机制、垃圾收集、集合算法、表驱动算法、图算法等。尽管其中的每一种都可以独立地学习,但是,在诸如编译程序这样一个有意义的环境中学习将更有教育意义。

第四,编译程序的许多构造技术已经得到了广泛的应用。许多应用程序非常接近于编译程序,已经采用了编译构造的部分技术,例如读入格式化的数据、文件转换问题等。如果数据有清晰的结构,就可以为它设计文法,使用分析程序生成器自动产生一个分析程序。从编译程序构造技术受益的文件转换系统的范例包括著名的 Tex/Latex 文本格式化程序(Knuth 开发的、应用广泛的、共享的文字处理软件的文件格式),该程序把 Tex 文本转换成机器独立的 dvi 格式;还包括 PostScript(一种普遍应用的、与机器和软件无关的文件格式,主要用于文件的传输和打印)解释程序,它将 PostScript 文本转换为打印机的指令。这种技术还有利于迅速引进新的文件格式,例如,能够快速地创建 HTML、XML 和 UML 等文件的读入和分析程序。





第五,学习编译原理和技术还有助于理解程序设计语言,以编写出优秀的软件。浏览一下计算机的历史,就会发现很多被称为程序设计大师的人都是编译领域的泰斗:创建了 Pascal 语言及其系列的图灵奖获得者 Niklaus Wirth 教授;写出第一个微型机上运行的 Basic 语言的 Bill Gates;成为 Sun 公司副总裁的 Java 缔造者 James Gosling 博士;Delphi 的架构师与 C# 的缔造者 Anders Hejlsberg; C++ 之父 Bjarne Stroustrup 教授等。

1.2 什么叫编译程序

编译程序是计算机系统经典、核心的系统软件。自从 20 世纪 50 年代出现了以 Fortran 为代表的计算机高级编程语言以后,在抽象层次和使用方便等方面都超过机器代码和汇编程序的高级编程语言层出不穷,使用过的已经超过数千种。而且,随着计算机系统更加广泛的应用,研制和开发高级编程语言的工作远远没有停止的趋势。按照现在的计算机体系结构和组成原理以及软件开发的理论和实践,高级程序语言仍将是开发计算机应用系统的关键技术,与之不可分离的是高级程序语言的编译程序。

用高级编程语言,如 Fortran、Pascal、Ada、Smalltalk、C、C++、C# 和 Java 编写程序方便而且效率高,但是,计算机需要把高级编程语言的程序翻译成机器语言代码或汇编程序才能运行。翻译程序或翻译器是把一种语言(源语言)转换成等价的另外一种语言(目标语言)的程序。如果源语言是高级编程语言,目标语言是机器代码和汇编语言这样的低级语言,这类翻译程序就叫作编译程序或编译器。如果目标语言是汇编语言,则还需要汇编成机器代码之后才可以运行。由于汇编语言和机器代码十分接近,像其他编译教材一样,本书通常不区分目标程序是机器代码还是汇编程序。

源程序的运行实质上包含两个过程:首先是把源程序用编译程序翻译成机器可以执行的目标程序或目标代码,然后才能接受输入数据运行,图 1.1 显示的就是这种编译执行方式。

运行高级语言程序的另外一种方式是解释执行,它需要的翻译程序不是编译程序,而是解释程序。解释程序不产生源程序的目标代码,而是对源程序逐条语句进行分析,根据每个语句的含义执行产生结果,如图 1.2 所示。Basic 和多数脚本语言都是按照解释方式运行的。解释方式的主要优点是便于对源程序进行调试和修改,但是其加工过程降低了程序的运行效率。

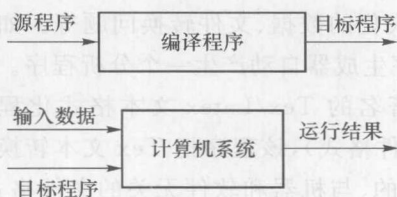


图 1.1 高级语言程序的编译执行方式

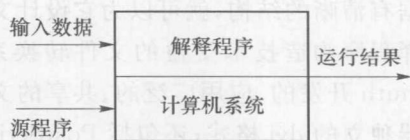


图 1.2 高级语言程序的解释执行方式

理论上,每一种语言都可以根据需要采用编译执行方式和解释执行方式,实践中每种语



言都主要采用一种执行方式。Java 语言同时具有编译执行方式和解释执行方式,分别应用于不同的计算机运行环境。由于编译程序和解释程序的原理基本一样,本书主要讲解编译程序的基本原理和技术,它们也可以适用于解释程序的构造与实现。

1.3 编译过程概述

把计算机高级编程语言翻译成计算机可以执行的代码的工作包括一系列的活动和任务,是一个复杂的完整过程。编译过程可以和把英语翻译成中文的过程相比较。在翻译一篇英文的时候,首先要确定英语的单词、标点符号等,把句子分解成单词以后去理解单词的基本含义;然后,分析英语句子的词组和语法结构,理解原句的含义,根据已知的中英文句型进行初步的翻译;接下来对译文进行修饰和润色;最后写出译文。

计算机程序的编译过程类似,一般划分为五个阶段:词法分析、语法分析、语义分析及中间代码生成、代码优化、目标代码生成。

1.3.1 词法分析

词法分析的任务是逐步地扫描和分解构成源程序的字符串,识别出一个一个的单词符号或符号。词法分析的工作主要包括:识别出程序中的单词符号,在编译程序符号表中查找并登记单词符号及其信息,譬如单词符号的类型、内部表示、数值等。计算机高级语言的单词符号通常包括:标识符、关键字或基本字、标点符号、常数、运算符、分隔符等。例如,对于 C 语言的 while 语句

while(i<100)sum += i++; (1.1)

词法分析的结果是识别出表 1.1 中所示的单词。

表 1.1 语句 while(i<100)sum += i++; 的单词及其类型

符号	类型	符号	类型
while	关键字	(	分隔符
i	标识符	<	运算符
100	整常数	)	分隔符
sum	标识符	+=	复合赋值符
i	标识符	++	运算符
;	分隔符		

这些单词是 C 语言的基本单词符号,是构成 C 程序的基本组成成分,是人们理解和编写 C 程序的基本要素。在词法分析的过程中通常都把分隔符类型中的空格符号删掉。

编译程序的词法分析也叫词法扫描或线性扫描。本书的第 2 章将重点介绍词法分析的基本原理、技术和工具,如有限状态机、正规表达式及其之间的等价转换以及词法分析生成器 Lex 等。





1.3.2 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元(语法范畴、语法短语),例如“短语”、“子句”、“语句”、“程序段”、“函数”和“程序”等。语法分析是把线性序列的单词符号根据语言的语法规则,按照层次分解,结果通常表示成语法分析树。

例如,C 语言的 while 语句格式是:

while (表达式) 循环体

其中表达式中包含赋值表达式、逗号表达式、复合赋值表达式等。语句 1.1 经过语法分析之后可以表示成如图 1.3 所示的分析树。

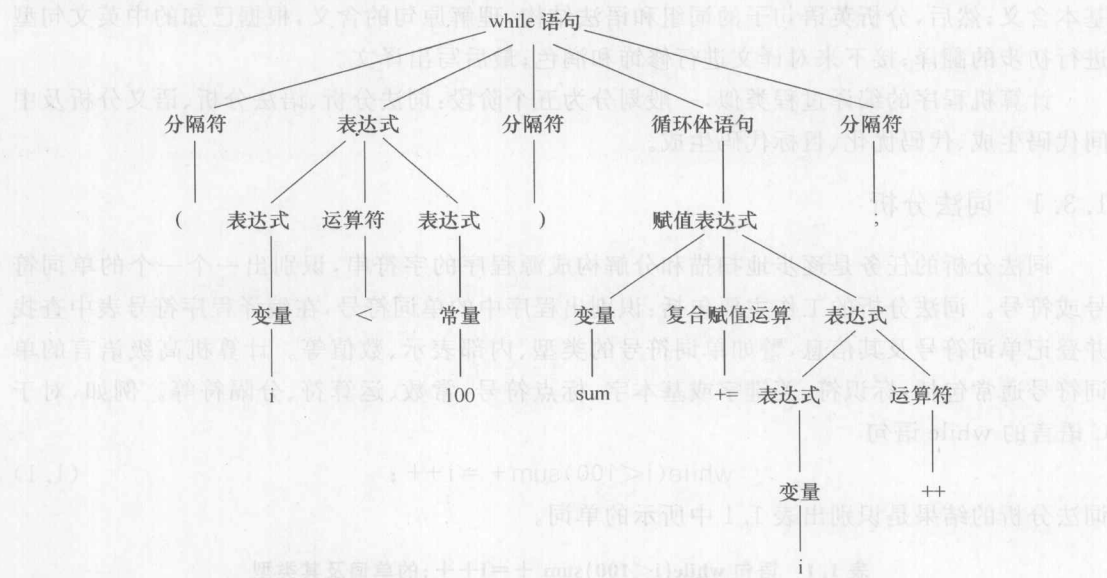


图 1.3 语句 while (i<100) sum +=i++;的语法分析树

第 3 章将讨论描述程序语言规则的形式语言基础,第 4 章和第 5 章将分别详细讨论典型的自顶向下语法分析的递归下降分析 LL(1)技术、自底向上的语法分析方法算符优先分析和 LR 分析技术以及语法分析工具 YACC。

1.3.3 语义分析和中间代码生成

语义分析的任务是检查程序语义的正确性,解释程序结构的含义。语义分析包括检查变量是否有定义、变量在使用前是否具有值、数值是否溢出等,其中的一个重要部分是进行类型的检查和转换。编译程序检查每个运算符的运算对象,看它们的类型是否合适,如果不合适,分析的结果应当是报告错误,同时给出类型转换的建议。例如,一个整数类型的变量和一个实数类型的变量相加,通常的编程语言要求编译发出警告信息,并且建议把整数类型转换成实数类型以后再进行运算,结果也是实数类型。

语义分析完成之后,编译程序通常就依据语言的语义规则,利用语法制导技术把源程序



翻译成某种中间代码。所谓“中间代码”是一种定义明确、便于处理、独立于计算机硬件的记号系统,可以认为是一种抽象机的程序。引入中间代码的目的主要是使编程语言及其程序不依赖于特殊的计算机类型,方便程序的移植和运行,同时可以利用词法分析、语法分析和语义分析对程序进行各种处理,因此,中间代码广泛应用在各类集成化软件开发环境中。

对中间代码的基本要求是:既要便于把源程序翻译成中间代码,又要易于把它翻译成各种计算机的指令或汇编语言。中间语言有多种形式,其中一类是三地址代码,很像机器的汇编语言,这种抽象机器的每个存储单元的作用类似于寄存器。三地址代码由三地址语句序列组成,每条三地址语句最多包含三个操作数。源程序的语句 1.1 在语义分析和中间代码翻译阶段可以变换成如下的三地址代码:

```

Lbegin :   if i < 100 goto Lbody
           goto Lend
Lbody :   t1 := sum + i
           sum := t1
           t2 := i + 1
           i := t2
           goto Lbegin
Lend :

```

(1.2)

对编程语言含义的形式化表示要依据语言的语法结构,在第 8 章将介绍一种为语法增添语义的方式——属性文法,结合语法制导技术执行语义分析。在第 9 章将介绍典型的中间代码语言,讨论程序语言基本结构的语法制导的翻译方法。

1.3.4 代码优化

代码优化的主要任务是对前一阶段产生的中间代码进行等价变换,以便产生速度快、空间小的目标代码。由于中间代码生成主要是按照语义规则把源程序自动翻译成中间代码,通常不考虑生成代码的执行效率,因此给代码优化留下了很多机会。例如,在语句(1.1)的翻译代码(1.2)中就可以去掉临时变量 t_1 和 t_2 ,变换成:

```

Lbegin :   if i < 100 goto Lbody
           goto Lend
Lbody :   sum := sum + i
           i := i + 1
           goto Lbegin
Lend :

```

(1.3)

同样,在下一阶段产生目标代码以后,也可以对目标代码进行优化,改善最终形成的目标程序的效率。无论是中间代码的优化,还是目标代码的优化都要占用编译程序的时间和空间资源,编译的优化需要更多的均衡。本书的第 11 章将集中讨论代码优化的各种策略和算法,其中重点介绍中间代码的优化技术以及目标代码的基本技术。

1.3.5 目标代码生成

编译的最后一个阶段是目标代码生成,其主要任务是把(经过优化处理的)中间代码翻

