



“十二五”普通高等教育本科国家级规划教材

编译原理

(第3版)

陈意云 张 昱 编著

高等教育出版社

“十二五”普通高等教育本科国家级规划教材

编译原理

Bianyi Yuanli

(第3版)

陈意云 张 昱 编著

高等教育出版社·北京

内容提要

本书介绍编译器构造的一般原理和基本实现方法,内容包括词法分析、语法分析、语义分析、中间代码生成、目标代码生成、独立于机器的优化和依赖于机器的优化等。除了介绍命令式编程语言的编译技术外,本书还介绍面向对象编程语言和函数式编程语言的实现技术。另外,本书还强调一些相关的理论知识,如形式语言和自动机理论、语法制导的定义和属性文法、类型论和类型系统等。

本书内容丰富、讲解深入,注意理论联系实际,可作为高等学校计算机科学及相关专业的教材,也可供计算机软工技术人员参考。

图书在版编目(CIP)数据

编译原理/陈意云,张昱编著.--3版.--北京:
高等教育出版社,2014.9
ISBN 978-7-04-040491-3

I. ①编… II. ①陈… ②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 160816 号

策划编辑 刘 艳 责任编辑 张海波 封面设计 于文燕 版式设计 童 丹
插图绘制 郝 林 责任校对 陈旭颖 责任印制 朱学忠

出版发行	高等教育出版社	网 址	http://www.hep.edu.cn
社 址	北京市西城区德外大街 4 号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.landaco.com
印 刷	北京鑫海金澳胶印有限公司		http://www.landaco.com.cn
开 本	787mm×1092mm 1/16	版 次	2003 年 8 月第 1 版
印 张	28.5		2014 年 9 月第 3 版
字 数	650 千字	印 次	2014 年 9 月第 1 次印刷
购书热线	010-58581118	定 价	39.00 元
咨询电话	400-810-0598		

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换

版权所有 侵权必究

物料号 40491-00

第3版前言

本书连续三次(“十五”、“十一五”和“十二五”)入选普通高等教育本科国家级规划教材,并于2009年被评为普通高等教育精品教材,感谢广大读者、教师和评审专家对本教材的肯定和期待。本次改版受2012年底第三次入选规划教材的推动。

编译器技术近年来的主要进展在面向新型计算机体系结构的优化方面,例如并行化和低功耗。本教材第2版已增加一章专门介绍依赖于机器的优化,再增加这方面内容会使得教材各部分的比例失调。另一方面,对高可信软件的需求也在推动编译器技术的研究,例如出具证明的编译器(certifying compiler)和经过验证的编译器(verified compiler),但相关成果尚未成熟到值得写入教材。

基于上述考虑,本次改版没有增删内容,但添加了30多道习题。这些习题大部分都从学习编程语言和编译技术时碰到的实际问题中抽象出来,对把握编程语言和理解编译技术很有帮助。此外,本次改版还对各章内容进行了局部修订。配合本次改版,作者也改版了与本教材配套的《编译原理习题精选与解析》(见参考文献[8])。

对于40学时的课程,作者建议讲授第1~9章(不包括加星号的节),也可以直接采用作者专为普通高等学校学生编写的《编译原理与技术》(见参考文献[9]),它就是按这个讲授建议形成的精简教材。对于60学时的课程,作者建议讲授第1~9章、第11章(不含11.3节)和第12章,第10章和第13章留给感兴趣的读者。作者讲授编译原理课程的教学课件和历年试题在作者教学网页(<http://staff.ustc.edu.cn/~yiyun>, <http://staff.ustc.edu.cn/~yuzhang>)上。

对于课程的综合实践,若不打算涉及代码生成和代码优化,作者建议选用《编译原理与技术》附录给出的基于PL/0语言的课程实践项目,该项目的参考资料、项目要求、测试程序,以及用C语言写的PL/0语言编译器和解释器的源代码都可以从作者教学网页下载。若打算实施覆盖编译所有阶段的大型综合课程实践,可以采用作者编写的《编译原理实验教程》(见参考文献[10])上的项目,相关资料也可以从作者教学网页下载。

本次改版得到中国科学技术大学的支持和一些专家的建议,得到了高等教育出版社刘艳编辑的匡正与审订。作者谨向所有为本次改版提供支持与帮助的人致以最诚挚的谢意。

虽经多次改版,书中仍难免存在不妥和错误之处。若发现本书有错误或对本书提出建议,欢迎发送电子邮件(yiyun@ustc.edu.cn, yuzhang@ustc.edu.cn)批评指正,作者将与出版社联系,在再次印刷或再版时改正错误和采纳合理的建议。

作 者

于中国科学技术大学

2013年8月

第2版前言

本书改版的动力主要来自下列几个方面:

- (1) 编译器技术的发展,特别是计算机体系结构的发展对编译器技术的推动;
- (2) 第1版教材在使用中得到的积极反馈和建议;
- (3) 作者在教学实践中的心得和体会,在相关科研活动中的进展与收获。

第2版增加了有关新理论和新技术的介绍,重写了因理论和技术发展而需要调整的部分,改写了原先有疏漏而需要改善的地方,并且继承和发扬了前一版的特色。改动主要体现在:

(1) 增加了第10章“依赖于机器的优化”。简要介绍现代计算机体系结构、指令调度、基本块调度、全局调度、软件流水、并行性和数据局部性优化。

(2) 重写了第9章“独立于机器的优化”。突出数据流分析的理论基础,强调在数据流分析的一般框架下解决各个具体数据流问题,使本章立足于更高层面讨论代码优化。

(3) 改写了第5章“类型检查”的前两节。减少一些不必要的概念,把保留的概念区分得更清楚一些。

(4) 删掉一些过时的或较少使用的内容。此外,考虑到Pascal语言目前已较少使用,因此,把第1版中采用的大部分Pascal示例代码改为C语言示例代码。

本次改版主要参考了文献[9],在此向作者表示衷心的感谢。

特别感谢中国科学院计算技术研究所张兆庆研究员和冯晓兵研究员,他们在百忙之中仔细审阅了全稿,并对书稿提出了宝贵的意见。中国科学技术大学为本书的编写提供了充分的资金支持。付雄博士为新增的第10章准备了初稿。在此谨向所有为本次改版提供支持帮助的人致以最诚挚的谢意。

未加星号的各章已构成编译原理知识的一个基本架构,本科阶段的教学可在此基础上适当地增删,加星号的各章可供更深入学习时使用。与本书配套的习题解答是由作者编写、高等教育出版社出版的《编译原理习题精选与解析》。由作者编写的课程实践教材《编译原理实验教程》将在2008年年底由高等教育出版社出版。作者讲授编译原理课程的教学课件和历年试题可在作者的教学网页(<http://staff.ustc.edu.cn/~yiyun>,<http://staff.ustc.edu.cn/~yuzhang>)上下载。

限于时间和学识水平,书中难免存在不妥和错误之处。如果发现本书有任何错误或有任何建议,欢迎给作者发送电子邮件(yiyun@ustc.edu.cn,yuzhang@ustc.edu.cn)批评指正,作者将及时改正错误。

作者

于中国科学技术大学

2008年3月

第 1 版前言

本书介绍编译器构造的一般原理、基本设计方法和主要实现技术,可作为高等院校计算机科学及相关专业的教材。

虽然只有少数人从事构造或维护程序设计语言编译器的工作,但是编译原理和技术对高校学生和计算机软件工程技术人员来说仍是重要的基础知识之一。本书能使读者对程序设计语言的设计和实现有深刻的理解,对和程序设计语言有关的理论有所了解,对宏观上把握程序设计语言来说,能起一个奠基的作用。本书的学习还有助于读者快速理解、定位和解决在程序调试与运行中出现的问题。

对软件工程来说,编译器是一个很好的实例(基本设计、模块划分、基于事件驱动的编程等),本书所介绍的概念和技术能应用到一般的软件设计之中。

大多数程序员同时也是语言的设计者,虽然是一些简单语言(如输入输出、脚本语言)的设计者,但学习本书有助于提高他们设计这些语言的水平。

编译技术在软件安全、程序理解和软件逆向工程等方面有着广泛的应用。

作为一本教材,本书有如下一些特点:

(1) 在介绍语言实现技术的同时,强调一些相关的理论知识,如形式语言和自动机理论、语法制导的定义和属性文法、类型论和类型系统等。它们是计算机专业理论知识的一个重要部分,在本书中结合应用来介绍这些知识,有助于学生较快领会和掌握。

(2) 在介绍编译器各逻辑阶段的实现时,强调形式化描述技术,并以语法制导定义作为翻译的主要描述工具。

(3) 强调对编译原理和技术的宏观理解及全局把握,而不把读者的注意力分散到一些枝节的算法上,如计算开始符号集合和后继符号集合的算法、回填技术等。出于同样的目的,本书较详细地介绍了编译系统和运行系统。

(4) 本书还介绍面向对象语言和函数式语言的实现技术,有助于加深读者对语言实现技术的理解。书中带星号的章节,作为教学的可选部分。

(5) 作为原理性教材,本书介绍基本的理论和方法,而不偏向于某种源语言或目标机器。

(6) 我们鼓励读者用所学的知识去分析和解决实际问题,因此本书中有很多习题是从实际碰到的问题中抽象出来的。这些习题也能激发读者学习编译原理和技术的积极性。

(7) 为了便于读者学习,本书配有习题解答(见参考文献 8)。

本书的多数章节是参考了参考文献 1 和参考文献 7 编写的,部分习题取自参考文献 8。在此向有关作者表示感谢。

本书第1章到第6章以及第12章主要由陈意云编写,第7章到第11章主要由张昱编写。作者的学生陈晖准备了10.2节的初稿,李筱青准备了10.3节的初稿,吴萍和项森也为第10章的编写做了很多技术工作。

中国科学院软件研究所研究员程虎先生审阅了全书,并提出了许多宝贵的意见,在此表示衷心的感谢。

由于作者水平有限,书中难免存在一些缺点和错误,恳请广大读者批评指正。

作者

于中国科学技术大学

2003年5月

郑重声明

高等教育出版社依法对本书享有专有出版权。任何未经许可的复制、销售行为均违反《中华人民共和国著作权法》，其行为人将承担相应的民事责任和行政责任；构成犯罪的，将被依法追究刑事责任。为了维护市场秩序，保护读者的合法权益，避免读者误用盗版书造成不良后果，我社将配合行政执法部门和司法机关对违法犯罪的单位和个人进行严厉打击。社会各界人士如发现上述侵权行为，希望及时举报，本社将奖励举报有功人员。

反盗版举报电话 (010)58581897 58582371 58581879

反盗版举报传真 (010)82086060

反盗版举报邮箱 dd@hep.com.cn

通信地址 北京市西城区德外大街4号 高等教育出版社法务部

邮政编码 100120

目 录

第 1 章 引论	1	2.3.1 不确定的有限自动机	24
1.1 编译器概述	1	2.3.2 确定的有限自动机	25
1.1.1 词法分析	1	2.3.3 NFA 到 DFA 的变换	26
1.1.2 语法分析	3	2.3.4 DFA 的化简	29
1.1.3 语义分析	3	2.4 从正规式到有限自动机	31
1.1.4 中间代码生成	4	2.5 词法分析器的生成器	34
1.1.5 代码优化	4	习题 2	37
1.1.6 代码生成	5	第 3 章 语法分析	40
1.1.7 符号表管理	5	3.1 上下文无关文法	40
1.1.8 阶段的分组	6	3.1.1 上下文无关文法的定义	41
1.1.9 解释器	7	3.1.2 推导	42
1.2 编译器技术的应用	7	3.1.3 分析树	43
1.2.1 高级语言的实现	8	3.1.4 二义性	44
1.2.2 针对计算机体系结构的优化	9	3.2 语言和文法	45
1.2.3 新计算机体系结构的设计	10	3.2.1 正规式和上下文无关文法的比较	45
1.2.4 程序翻译	10	3.2.2 分离词法分析器的理由	46
1.2.5 提高软件开发效率的工具	11	3.2.3 验证文法产生的语言	46
习题 1	12	3.2.4 适当的表达式文法	47
第 2 章 词法分析	13	3.2.5 消除二义性	48
2.1 词法记号及属性	13	3.2.6 消除左递归	49
2.1.1 词法记号、模式、词法单元	14	3.2.7 提左因子	50
2.1.2 词法记号的属性	15	*3.2.8 非上下文无关的语言构造	51
2.1.3 词法错误	16	*3.2.9 形式语言鸟瞰	52
2.2 词法记号的描述与识别	16	3.3 自上而下分析	54
2.2.1 串和语言	16	3.3.1 自上而下分析的一般方法	54
2.2.2 正规式	18	3.3.2 LL(1)文法	55
2.2.3 正规定义	19	3.3.3 递归下降的预测分析	56
2.2.4 状态转换图	20	3.3.4 非递归的预测分析	57
2.3 有限自动机	23		

3.3.5	构造预测分析表	60	4.2.2	构造语法树的语法制导定义	115
3.3.6	预测分析的错误恢复	61	4.2.3	S 属性的自下而上计算	117
3.4	自下而上分析	65	4.3	L 属性定义的自上而下计算	119
3.4.1	归约	65	4.3.1	L 属性定义	120
3.4.2	句柄	65	4.3.2	翻译方案	120
3.4.3	用栈实现移进-归约分析	66	4.3.3	预测翻译器的设计	124
3.4.4	移进-归约分析的冲突	68	4.3.4	用综合属性代替继承属性	126
3.5	LR 分析器	69	4.4	L 属性的自下而上计算	126
3.5.1	LR 分析算法	69	4.4.1	删除翻译方案中嵌入的动作	127
3.5.2	LR 文法和 LR 分析方法的 特点	73	4.4.2	分析栈上的继承属性	127
3.5.3	构造 SLR 分析表	74	4.4.3	模拟继承属性的计算	130
3.5.4	构造规范的 LR 分析表	81	习题 4		132
3.5.5	构造 LALR 分析表	85	第 5 章	类型检查	135
3.5.6	非二义且非 LR 的上下文 无关文法	88	5.1	类型在编程语言中的作用	135
3.6	二义文法的应用	89	5.1.1	执行错误和安全语言	136
3.6.1	使用算符的优先级和 结合性来解决冲突	90	5.1.2	类型化语言和类型系统	136
3.6.2	使用其他约定来解决冲突	92	5.1.3	类型化语言的优点	139
3.6.3	LR 分析的错误恢复	93	5.2	类型系统的描述语言	139
3.7	语法分析器的生成器	95	5.2.1	定型断言	140
3.7.1	分析器的生成器 Yacc	95	5.2.2	定型规则	141
3.7.2	用 Yacc 处理二义文法	98	5.2.3	类型检查和类型推断	142
3.7.3	Yacc 的错误恢复	101	5.3	一个简单类型检查器的规范	143
习题 3		103	5.3.1	一个简单的语言	143
第 4 章	语法制导的翻译	109	5.3.2	类型系统	143
4.1	语法制导的定义	109	5.3.3	类型检查	145
4.1.1	语法制导定义的形式	110	5.3.4	类型转换	147
4.1.2	综合属性	111	*5.4	多态函数	148
4.1.3	继承属性	111	5.4.1	为什么要使用多态函数	148
4.1.4	属性依赖图	112	5.4.2	类型变量	149
4.1.5	属性计算次序	113	5.4.3	一个含多态函数的语言	151
4.2	S 属性定义的自下而上计算	114	5.4.4	代换、实例与合一	153
4.2.1	语法树	115	5.4.5	多态函数的类型检查	154
			5.5	类型表达式的等价	158
			5.5.1	类型表达式的结构等价	158
			5.5.2	类型表达式的名字等价	159

5.5.3	记录类型	160	7.1.1	后缀表示	213
5.5.4	类型表示中的环	161	7.1.2	图形表示	213
5.6	函数和算符的重载	162	7.1.3	三地址代码	214
5.6.1	子表达式的可能类型集合	162	7.1.4	静态单赋值形式	216
5.6.2	缩小可能类型的集合	163	7.2	声明语句	217
习题 5		164	7.2.1	过程中的声明	217
第 6 章	运行时存储空间的组织和管理	171	7.2.2	作用域信息的保存	218
6.1	局部存储分配	172	7.2.3	记录的域名	220
6.1.1	过程	172	7.3	赋值语句	221
6.1.2	名字的作用域和绑定	172	7.3.1	符号表中的名字	221
6.1.3	活动记录	174	7.3.2	数组元素的地址计算	222
6.1.4	局部数据的布局	175	7.3.3	数组元素地址计算的翻译 方案	223
6.1.5	程序块	175	7.3.4	类型转换	225
6.2	全局栈式存储分配	177	7.4	布尔表达式和控制流语句	226
6.2.1	运行时内存的划分	177	7.4.1	布尔表达式	227
6.2.2	活动树和运行栈	178	7.4.2	控制流语句的翻译	227
6.2.3	调用序列	180	7.4.3	布尔表达式的控制流翻译	229
6.2.4	栈上可变长度数据	183	7.4.4	开关语句的翻译	231
6.2.5	悬空引用	183	7.4.5	过程调用的翻译	233
6.3	非局部名字的访问	184	习题 7		234
6.3.1	无过程嵌套的静态作用域	185	第 8 章	代码生成	241
*6.3.2	有过程嵌套的静态作用域	185	8.1	代码生成器设计中的问题	241
*6.3.3	动态作用域	189	8.1.1	目标程序	241
6.4	参数传递	190	8.1.2	指令选择	242
6.4.1	值调用	190	8.1.3	寄存器分配	243
6.4.2	引用调用	191	8.1.4	计算次序选择	243
6.4.3	换名调用	191	8.2	目标语言	244
*6.5	堆管理	192	8.2.1	目标机器的指令集	244
6.5.1	内存管理器	192	8.2.2	指令代价	245
6.5.2	计算机内存分层	193	8.3	基本块和流图	247
6.5.3	程序局部性	195	8.3.1	基本块	248
6.5.4	手工回收请求	195	8.3.2	基本块的优化	249
习题 6		196	8.3.3	流图	250
第 7 章	中间代码生成	212	8.3.4	下次引用信息	251
7.1	中间语言	213			

8.4 一个简单的代码生成器	252	9.5.1 冗余的根源	301
8.4.1 寄存器描述和地址描述	252	9.5.2 能否删除所有的冗余	303
8.4.2 代码生成算法	253	9.5.3 惰性代码移动问题	304
8.4.3 寄存器选择函数	254	9.5.4 预期表达式	305
8.4.4 为变址和指针语句产生代码 ..	255	9.5.5 惰性代码移动算法	305
8.4.5 条件语句	256	9.6 流图中的循环	310
习题8	257	9.6.1 支配结点	311
*第9章 独立于机器的优化	267	9.6.2 回边和可归约性	312
9.1 优化的主要种类	267	9.6.3 流图的深度	314
9.1.1 优化的主要源头	268	9.6.4 自然循环	314
9.1.2 一个实例	268	9.6.5 迭代流图算法的收敛速度	315
9.1.3 公共子表达式删除	270	习题9	317
9.1.4 复写传播	272	*第10章 依赖于机器的优化	329
9.1.5 死代码删除	273	10.1 处理器体系结构	330
9.1.6 代码外提	274	10.1.1 指令流水线和分支延迟 ..	330
9.1.7 强度削弱和归纳变量删除	274	10.1.2 流水化的执行	331
9.2 数据流分析介绍	276	10.1.3 多指令发射	331
9.2.1 数据流抽象	277	10.2 代码调度的约束	332
9.2.2 数据流分析模式	278	10.2.1 数据相关	332
9.2.3 到达一定值	279	10.2.2 发现内存访问中的相关性 ..	333
9.2.4 活跃变量	284	10.2.3 寄存器使用和并行执行	
9.2.5 可用表达式	285	之间的折中	334
9.2.6 小结	287	10.2.4 寄存器分配和代码调度的	
9.3 数据流分析的基础	288	次序安排	335
9.3.1 半格	289	10.2.5 控制相关	336
9.3.2 迁移函数	291	10.2.6 投机执行的支持	336
9.3.3 一般框架的迭代算法	293	10.2.7 一个基本的机器模型	337
9.3.4 数据流解的含义	295	10.3 基本块调度	338
9.4 常量传播	297	10.3.1 数据依赖图	338
9.4.1 常量传播框架的数据流值	297	10.3.2 基本块的表调度	340
9.4.2 常量传播框架的迁移函数	298	10.3.3 区分优先级的拓扑次序 ..	341
9.4.3 常量传播框架的单调性	298	10.4 全局代码调度	341
9.4.4 常量传播框架的非分配性	299	10.4.1 简单的代码移动	342
9.4.5 结果的解释	300	10.4.2 向上的代码移动	343
9.5 部分冗余删除	301	10.4.3 向下的代码移动	344

10.4.4	更新数据相关	345	11.2.3	即时编译器	384
10.4.5	全局调度的其他问题	345	* 11.3	无用单元收集	386
10.4.6	静态调度器和动态调度器 的相互影响	346	11.3.1	标记和清扫	386
10.5	软件流水	346	11.3.2	引用计数	388
10.5.1	引言	347	11.3.3	复制收集	388
10.5.2	循环的软件流水	348	11.3.4	分代收集	390
10.5.3	寄存器分配和代码生成	350	11.3.5	渐增式收集	391
10.5.4	Do-Across 循环	351	11.3.6	编译器与收集器之间的 相互影响	391
10.5.5	软件流水的目标和约束	352	习题 11		394
10.5.6	软件流水算法	354	* 第 12 章	面向对象语言的编译	399
10.5.7	无环数据依赖图的调度	355	12.1	面向对象语言的概念	399
10.6	并行性和数据局部性优化		12.1.1	对象和对象类	399
	概述	356	12.1.2	继承	400
10.6.1	多处理器	357	12.1.3	信息封装	402
10.6.2	应用中的并行性	358	12.2	方法的编译	402
10.6.3	循环级并行	359	12.3	继承的编译方案	405
10.6.4	数据局部性	360	12.3.1	单一继承的编译方案	406
10.6.5	矩阵乘法算法	362	12.3.2	多重继承的编译方案	408
10.6.6	矩阵乘法算法的优化	364	习题 12		412
习题 10		365	* 第 13 章	函数式语言的编译	415
第 11 章	编译系统和运行时系统	369	13.1	函数式编程语言简介	415
11.1	C 语言的编译系统	369	13.1.1	语言构造	415
11.1.1	预处理器	370	13.1.2	参数传递机制	417
11.1.2	汇编器	371	13.1.3	变量的自由出现和约束 出现	418
11.1.3	连接器	371	13.2	函数式语言的编译简介	419
11.1.4	目标文件的格式	373	13.2.1	几个受启发的例子	420
11.1.5	符号解析	375	13.2.2	编译函数	422
11.1.6	静态库	376	13.2.3	环境与约束	422
11.1.7	可执行目标文件及装入	378	13.3	抽象机的体系结构	423
11.1.8	动态连接	379	13.3.1	抽象机的栈	424
11.1.9	处理目标文件的一些工具	381	13.3.2	抽象机的堆	425
11.2	Java 语言的运行时系统	381	13.3.3	名字的寻址	426
11.2.1	Java 虚拟机语言简介	382	13.3.4	约束的建立	427
11.2.2	Java 虚拟机	382			

13.4 指令集和编译	428	13.4.5 构造和计算闭包	436
13.4.1 表达式	428	13.4.6 letrec 表达式和局部变量	437
13.4.2 变量的引用性出现	430	习题 13	439
13.4.3 函数定义	431	参考文献	441
13.4.4 函数应用	432		

第 1 章

引 论

从理论上说,构造专用计算机来直接执行用某种高级语言编写的程序是可能的。但实际上,目前的计算机能执行的都是非常低级的**机器语言**。那么,一个基本问题是:用高级语言编写的程序是怎样变成能在计算机上执行的机器语言程序的?

能够完成从一种语言到另一种语言的保语义变换的软件称为**翻译器**,这两种语言分别称为该翻译器的**源语言**和**目标语言**。**编译器**是一种翻译器,它的特点是目标语言比源语言低级。

本章通过简要描述编译器的各个组成部分以及编译器技术的各种应用来介绍编译这个课题。该课题涉及编程语言、计算机体系结构、形式语言理论、类型论、算法和软件工程等方面的知识。

1.1 编译器概述

编译器的工作可以分成若干阶段,每个阶段把源程序从一种表示变换成另一种表示。编译过程的一种典型分解见图 1.1,图中左边一列的每个方框表示它的一个阶段。

本节以赋值语句

$$\text{position} = \text{initial} + \text{rate} * 60 \quad (1.1)$$

的翻译(假定变量都是实型)为例,简要介绍编译的各个阶段。

1.1.1 词法分析

词法分析阅读构成源程序的字符流,按编程语言的词法规则把它们组成**词法记号**(token)流。对于一个词法单元,词法分析产生的记号是

〈记号名,属性值〉

二元组。记号名是同类词法单元共用的名称,而属性值是一个词法单元有别于同类中其他词法单元的特征值。赋值语句(1.1)的字符流在词法分析时被依次组成下面这些记号。

(1) 标识符 `position` 形成的记号是〈`id`, 1〉,其中 `id` 是标识符的总称,1 代表 `position` 在符号

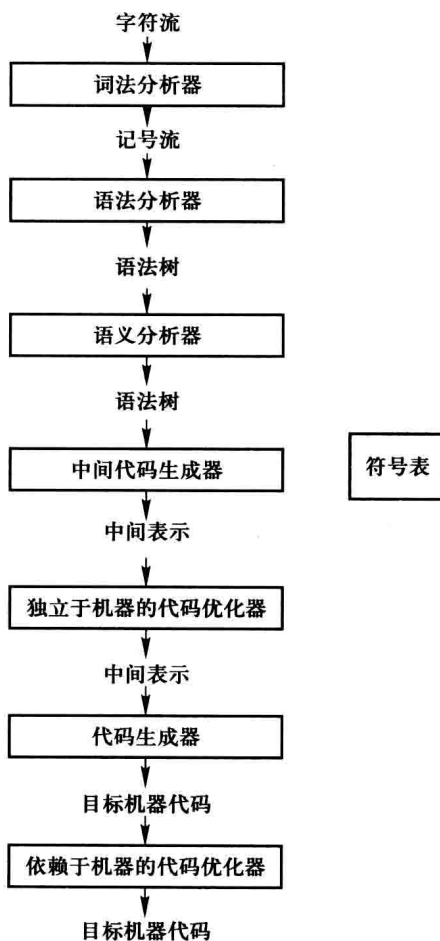


图 1.1 编译的各个阶段

表中的条目,符号表的条目用来存放标识符的各种属性,如它的名字和类型。

(2) 赋值号 = 形成的记号是 $\langle \text{assign} \rangle$, 因为该记号只有一个实例, 因此不需要以属性值来区分实例。为了直观起见, 下面直接用赋值号作为记号名, 写成 $\langle = \rangle$ 。

(3) 标识符 initial 形成的记号是 $\langle \text{id}, 2 \rangle$ 。

(4) 加号 + 形成的记号是 $\langle + \rangle$ 。

(5) 标识符 rate 形成的记号是 $\langle \text{id}, 3 \rangle$ 。

(6) 乘号 * 形成的记号是 $\langle * \rangle$ 。

(7) 数 60 形成的记号是 $\langle 60 \rangle$ 。从技术上讲, 程序中出现的常数也要放到符号表或单独的常数表中, 形成记号 $\langle \text{number}, 60 \text{ 在表中的条目} \rangle$ 。这个问题的讨论留到词法分析中具体介绍。为直观起见, 这里直接使用 60 在源程序中的字符序列作为记号名。

分隔单词的空格通常在词法分析时被删去。

于是,赋值语句(1.1)在词法分析后形成的记号流是

$$\langle \text{id}, 1 \rangle \langle = \rangle \langle \text{id}, 2 \rangle \langle + \rangle \langle \text{id}, 3 \rangle \langle * \rangle \langle 60 \rangle \quad (1.2)$$

第2章讨论词法分析,词法分析也称为线性分析或扫描。

1.1.2 语法分析

语法分析(syntax analysis)简称分析(parsing),它检查词法分析输出的记号流是否符合编程语言的语法规则,并依据这些规则所体现出的语言构造(construct,如函数、语句、表达式等)的层次性,用各记号的第一元建成一种树形的中间表示,这个中间表示用抽象语法的方式描绘了该记号流的语法情况。一种典型的中间表示是语法树,其中内部结点表示运算,它们的子结点代表该运算的运算对象。为记号流(1.2)建立的语法树见图1.2(a)。

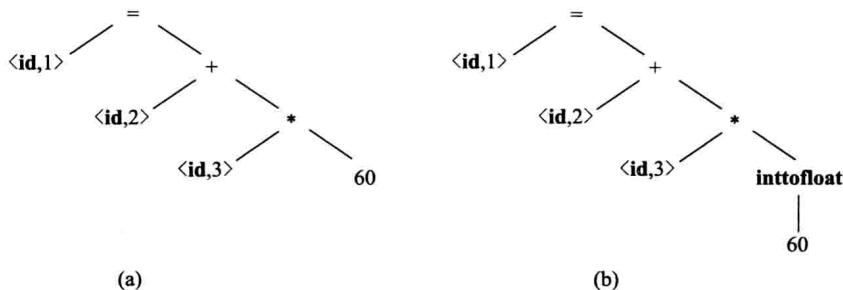


图 1.2 语义分析插入了类型转换

编译器后面各个阶段使用这种语法树进一步分析源程序和生成目标代码。第3章专门讨论语法分析算法,图1.2这种形式的语法树将在4.2节讨论。在第4章中,还将详细讨论编译器如何利用输入所含的层次结构来产生语法树。

1.1.3 语义分析

语义分析阶段使用语法树和符号表中的信息,依据语言定义来检查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。它还收集类型信息,把它们保存在符号表或语法树中。

语义分析的一个重要部分是类型检查,编译器检查每个算符的运算对象,看它们的类型是否适当。例如,当实数作为数组的下标时,许多语言的定义都要求编译器报告错误。语言定义也可能允许运算对象的类型作隐式转换,例如当二元算术算符作用于一个整数和一个实数时,编译器会把其中的整数转换为实数。

例 1.1 在机器内部,整数的二进制表示和实数的二进制表示是有区别的,不论它们是否有