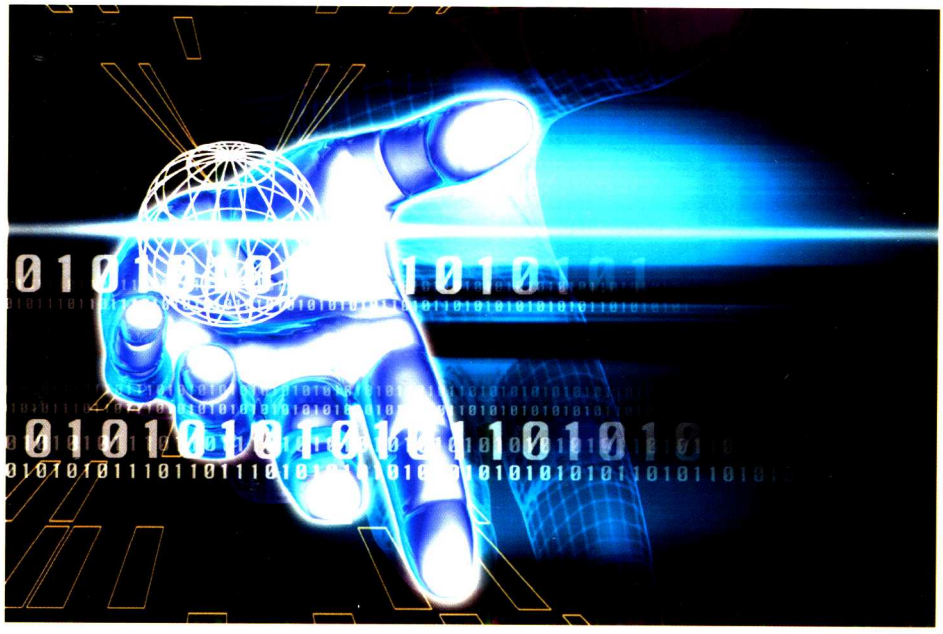
HZ BOOKS

高等院校计算机课程设计指导丛书

编译原理课程设计

王 雷 刘志成 周 晶 编著

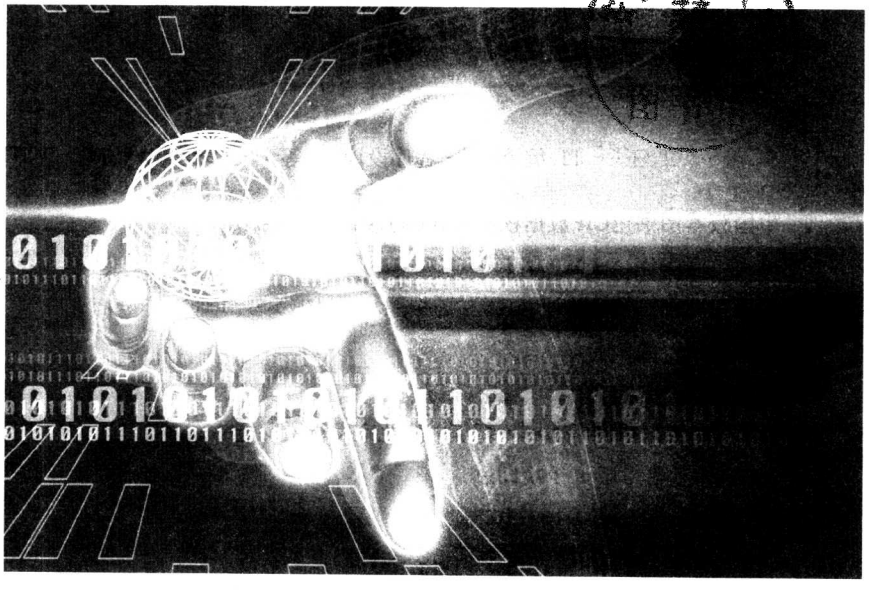


机械工业出版社
China Machine Press

高等院校计算机课程设计指导丛书

编译原理课程设计

王雷 刘志成 周晶 编著



机械工业出版社
China Machine Press

编译原理是大学计算机专业的必修课程。本书使用优秀的开源Java编译器GJC作为编译教学的基础平台，通过分析一个真正实用的现代编译系统，把编译理论应用到实际的工程实践中。全书不仅包括对编译器源代码的分析、对实例的讲解，还在最后给出3个具体的课程设计实验，介绍如何用书本上的编译理论实现一个真正的编译器。

本书适合作为大专院校编译原理课程设计的指导用书，相关的从业人员和研究人员也可以从中获得有益的参考。

版权所有，侵权必究。
本书法律顾问　北京市展达律师事务所

图书在版编目（CIP）数据

编译原理课程设计/王雷等编著. －北京：机械工业出版社，2005.3
（高等院校计算机课程设计指导丛书）
ISBN 7-111-15877-6

Ⅰ.编… Ⅱ.王… Ⅲ.编译程序-程序设计-高等学校-教学参考资料 Ⅳ.TP314

中国版本图书馆CIP数据核字（2004）第140798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）
策划编辑：温莉芳
责任编辑：朱起飞
北京牛山世兴印刷厂印刷·新华书店北京发行所发行
2005年3月第1版第1次印刷
787mm×1092mm 1/16·14.5印张
印数：0 001- 5 000册
定价：23.00元

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

丛书序言

近年来，我国在计算机应用、计算机软件和电子类相关专业的人才培养方面，取得了长足的进展，每年的毕业生都有数十万人。但是这些毕业生走进企业、公司、政府机构或研究单位之后，往往深刻地感觉到缺乏实际开发设计项目的经验，不善于综合运用所学理论，对知识的把握缺乏融会贯通的能力。

综合考察目前高等院校教学大纲、课程设置以及内容安排等方面的情况，多数学校还是比较重视训练学生的实际设计能力。但是，从安排设计实践的内容上看，基本上是围绕相关课程教学内容而展开的，不能够构成对实际问题的解决方案；从配套程序的规模上看，一般只是几十行到几百行的源代码，或者是一个单独电路的设计，远远小于一个小型项目的规模；从设计的结构上看，由于设计实践是围绕着课程教学内容而进行的，问题已经高度抽象，学生很难得到有关综合运用所学知识的整体训练机会。而且，这些内容相对简单、问题域已经高度抽象、规模较小的设计实践一人基本上就能完成，学生几乎无法通过这些设计实践，去真正获得有关项目管理和团队协作等方面的基本训练和工作经验。

由此可以看出，大多数学校对学生实际设计能力的训练与国外知名大学和国内精品课程相比较，还是存在一些差距的。为此，机械工业出版社华章分社和一批高等院校的教师，针对当前高等院校计算机硬件、软件和电子类相关课程教学中存在的问题，参考国内外知名大学相关课程成功的教学经验，设计编写了这套“高等院校计算机课程设计指导丛书”，其目的就是通过课程设计的一系列训练，把知识获取和项目实践两个方面有机地结合起来。

在这套“高等院校计算机课程设计指导丛书”中的每一门课程设计里，都安排了由多个子项目组成的一个课程设计项目。学生们可以在教师的指导下，逐步设计实现这些子项目，并最终完成一个功能相对完整，可以运行的系统，其代码可以是数千行，甚至上万行。通过这种设计课程，学生一方面可以结合课程的教学内容循序渐进地进行设计方面的实践训练，另一方面，在参与一系列子项目的实践过程中，还能提高如何综合运用所学知识解决实际问题的能力，以及获得有关项目管理和团队合作等众多方面的具体经验，增强对相关课程具体内容的理解和掌握能力，培养对整体课程知识综合运用和融会贯通能力。

参加丛书编写的各高等院校的教师都有着丰富的教学、科研，以及与企业合作开发项目等多方面的经验。每个课程设计中的子项目和整体项目，都来自教师们具体的科研和设计开发实践，所选设计项目与教学内容配合紧密，项目的难度与规模适宜。

最后，感谢机械工业出版社华章分社编辑们的大力支持，使出版有关这套丛书的计划，从单纯的构想演化成带有油墨芳香的真实。

丛书写作组

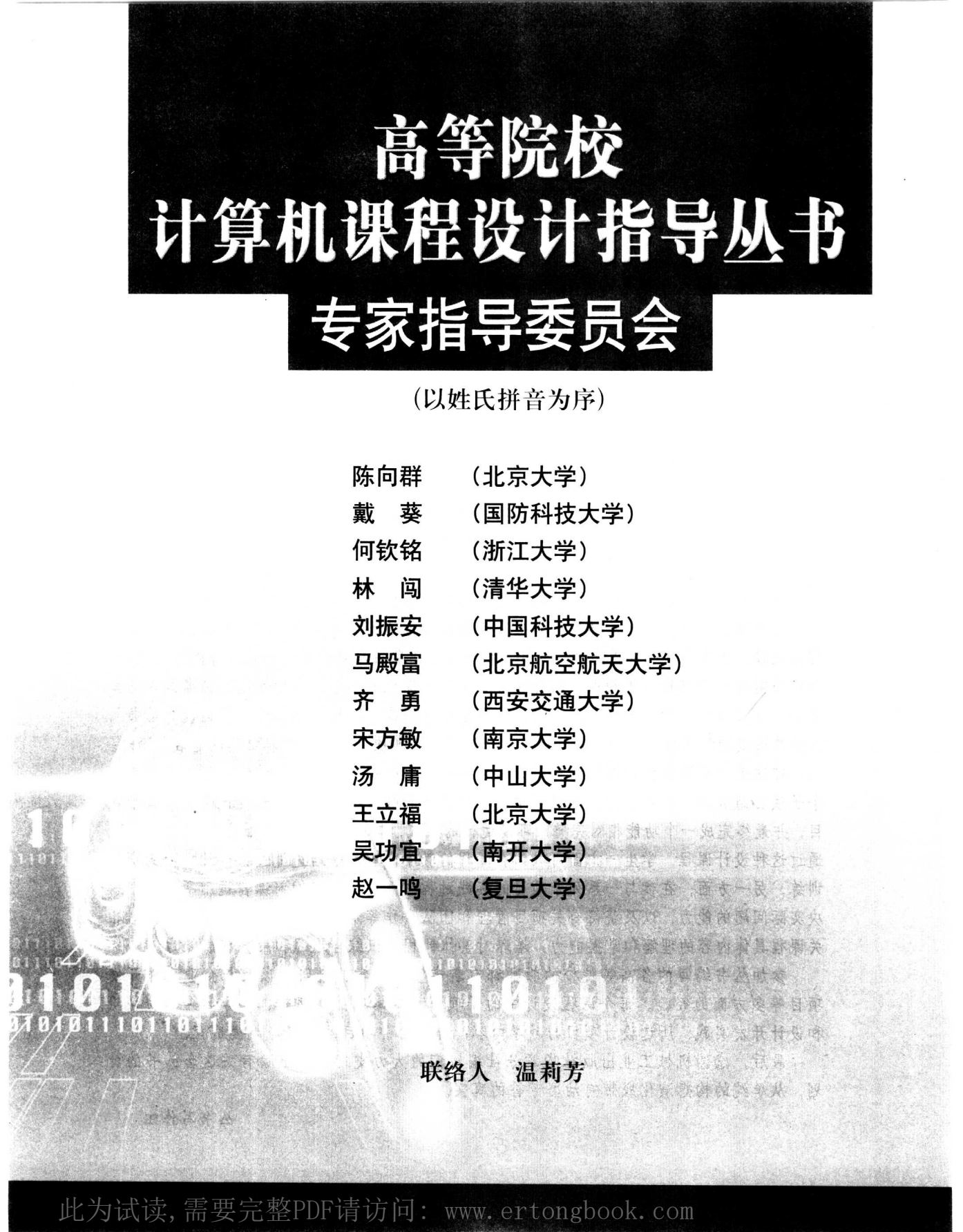
高等院校
计算机课程设计指导丛书

专家指导委员会

（以姓氏拼音为序）

陈向群　（北京大学）
戴　葵　（国防科技大学）
何钦铭　（浙江大学）
林　闯　（清华大学）
刘振安　（中国科技大学）
马殿富　（北京航空航天大学）
齐　勇　（西安交通大学）
宋方敏　（南京大学）
汤　庸　（中山大学）
王立福　（北京大学）
吴功宜　（南开大学）
赵一鸣　（复旦大学）

联络人　温莉芳



此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

编译理论和技术作为计算机科学研究和工程应用的基础，受到了广泛的重视。编译原理是一门实践性很强的课程，但是在教学过程中容易偏重于理论的介绍，而忽视了实验环节，因此学生很难真正掌握这门学科的精髓。本书正是为弥补这一缺陷而编写的编译实验教程，因此可以和讲授编译理论的教材配合使用。

本书希望通过分析一个真正实用的、开放源代码的现代编译系统，使读者了解编译器的构造。目前影响比较广泛的Java编译器有：GNU GCC中的GCJ编译器、IBM的Jikes编译器、Sun Hotspot J2SE使用的javac编译器（内部称作gjc-generic java compiler，GJC）以及实验Java新特性的pizza编译器等。最终本书选择了GJC作为研究的平台。

为了使读者能更清晰地理解编译器各个部分的代码，本书采用了静态分析和动态跟踪相结合的方法。针对编译器的每个部分，首先对源代码进行静态的分析，比较复杂的部分给出了相应的类图；然后，在每个部分的最后对一个程序进行动态跟踪，各个动态跟踪部分合起来就是GJC的整个编译过程。通过对编译器的分析，本书不仅介绍了一个真正的编译器是如何用编译理论来指导实现，而且还对编译器所采用的一些设计模式（例如工厂模式、访问者模式等）进行了详细分析。本书的主要内容如下：

第1章，介绍了开放源代码编译器的选取、GJC整体的代码分布、GJC总体结构以及编译执行的流程。

第2章，简单介绍了词法分析的原理，详细分析了GJC采用Java语言实现的一个针对Java语言的词法分析程序。

第3章，简单介绍了语法分析的原理，详细分析GJC语法分析中的基本数据结构和各种语法成分，介绍了GJC采用的递归下降分析法和算符优先方法。

第4章，介绍GJC中与符号管理相关的数据结构和设计方法。

第5章，简单介绍了编译系统常用的中间语言，详细分析GJC中抽象语法树的定义以及相关数据结构。

第6章，详细介绍了GJC中的上下文环境、对符号表的操作、类型检查等功能，分析了GJC语义分析的主要功能。

第7章，讨论了编译器中错误的分类，详细介绍GJC中的错误处理功能。

第8章，介绍了Java虚拟机的指令集编码、各条指令的意义以及如何使用这些指令。这些知识是代码生成部分的基础。

第9章，介绍了代码生成涉及到的指令编码、生成代码的管理、指令发射、存储管理等。详细分析了GJC为Java语言的各种控制结构生成代码的方法。

此外，本书以GJC为平台设计了三个实验。为了使读者能够顺利地完成实验，本书

不仅给出了一些参考文献和代码，而且还为每个实验给出了实验指南和参考难度。读者可以根据自己的兴趣对实验进行扩充，也可以根据实际的教学情况对实验进行选择。

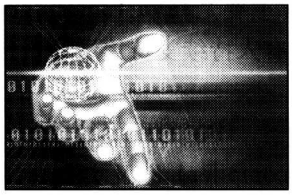
限于时间和水平，书中对GJC源代码的理解不一定完全准确，给出的实验指南也未必是最好的方案，欢迎广大读者多提宝贵意见。

最后，感谢机械工业出版社华章分社的编辑们对本书的大力支持。感谢所有亲人和朋友们的理解和支持！

作　者

2004.12

目　录



第1章 引 言

1.1 本书的目的

编译理论和技术作为计算机科学研究和工程应用的基础，受到了广泛的重视。编译原理也是大学计算机专业的必修课程。但是在这门课程的教学过程中，容易偏重于理论的介绍，而忽视了实践环节。学生对于编译的理解只停留在书本的概念上，而不知道怎样才能把编译理论应用到实际的工程实践中。本书正是为弥补这一缺陷而编写的编译实验教程，因此需要和一本讲授编译理论的教材配合使用。

我们希望通过分析一个真正实用的现代编译系统，使学生了解编译器的构造，而不是介绍一个玩具式的编译器。以往，编译器作为技术含量很高的产品，往往只有为数不多的几个公司可以开发，一旦开发成功就作为专有软件加以保护，不公开它的源代码。这样就使我们不知道一个真正的、符合工业标准的编译器是如何实现的。近年来随着开放源代码运动的发展，我们可以获得许多优秀编译器的源代码。撰写本书的初衷就是利用这个机遇，使用优秀的开源编译器作为编译教学的一个基础平台。通过对编译器源代码的分析，介绍一个真正的编译器是如何用我们所学到的编译理论实现的。

1.2 平台的选择

我们选取实验平台的基本原则是：该平台一定要开放源代码，这样读者可以从Internet上自由下载；另外要保证该平台一定是完整的，也就是说可以获得全部源代码，而不是源代码的片断，这样我们才能在平台上进行实验，而不仅仅是进行代码分析。

由于开放源代码运动的广泛开展，许多优秀的编译器都开放了源代码，供人们研究和改进。比较著名的有GCC，最初是由Richard Stallman开发的C编译器，现在已经成为一个支持多种语言和平台的编译器集合。GCC的含义也由最初的GNU C Compiler转变为GNU Compiler Collection。另外，还有很多其他的开源编译器项目，如SGI开发的针对IA-64体系结构的pro64/open64编译器，以及中科院和Intel合作开发的基于open64的orc编译器等。因此，我们有许多可选的系统。

Java语言是一种新兴的语言，继承了许多语言的优秀特性，而且由于使用了虚拟机技术，实现了平台无关性。如何实现一个Java语言的编译器，是一个许多人都感兴趣的问题。而Java语言也是一种在不断演进的语言，慢慢吸收了一些C++的特性，如泛型的机制等，如何支持这些新的特性也是一项非常有意义的工作。所以本书决定选择一个

Java的编译器作为实验平台，该编译器应该符合Java的标准（主要是Java Language Specification 和 Java Virtual Machine Specification）。另外，应该有人持续支持该编译器，不要让它变成一个无人维护的项目。

目前影响比较广泛的Java编译器有：GNU GCC中的GCJ编译器、IBM的Jikes编译器、Sun Hotspot J2SE使用的javac编译器（内部称作gjc-generic java compiler）以及实验Java新特性的Pizza编译器等。它们各有特色，其中GCJ是C语言实现的Java编译器，由于它融于GCC大的框架之下，显得过于庞大，因而不适合本书使用。Jikes是用C++语言开发的Java编译器，规模不大，有着很好的社区支持，并且用于IBM另一个开源虚拟机项目Jikes RVM之中，但是一个不足之处是缺少相关文档，源码中的注释不详细。Pizza是Java的超集，是用pizza语言来实现的，不是严格意义上的Java语言。

虽然这些系统没有被选为本书的实验平台，但是都是一些非常优秀的编译系统，可以供对编译技术有兴趣的同学研究使用，列在这里也是为了给感兴趣的读者提供相关信息。

最终本书选择了GJC作为研究的平台，它是由Java语言实现的Java编译器，与Java语言的标准完全兼容。整个系统规模不大，源程序有着很好的注释。作为Sun Hotspot J2SE使用的javac编译器，GJC是一个优秀的工业级编译器，并且它将会被有效而持续地支持，并随着Java语言的演变而发展。虽然GJC是Sun公司的专有软件，但是Sun公司提供了它的源代码供研究使用，可以在Sun的网站上通过授权获得hotspot虚拟机环境的全部源代码，GJC作为J2SE的一个工具正是其中的部分。为了方便分析，我们将GJC从整个代码中单独分离了出来（详见附录二）。

1.3 GJC的总体结构

为了使同学们在学习之前对GJC有一个整体印象，也为了避免在以后章节的学习过程中陷入实现细节，而无法把握系统的整体结构，我们在这里对GJC的源代码规模、源代码的分布和总体结构进行了简单介绍。GJC的整体规模不大，共74个文件，34 000行源程序，具体的代码量分布如下：

文件	大小	行数
v8\util\Abort.java	398	18
v8\comp\Attr.java	54383	1392
v8\comp\AttrContext.java	1772	77
v8\comp\AttrContextEnv.java	670	27
v8\util\Bits.java	5828	228
v8\util\ByteBuffer.java	4272	164
v8\code\ByteCodes.java	18103	1167
v8\comp\Check.java	38001	1025
v8\code\ClassFile.java	4610	144
v8\code\ClassReader.java	43841	1393
v8\code\ClassWriter.java	27509	846

此为试读，需要完整PDF请访问：www.ertongbook.com

文件	大小	行数
v8\code\Code.java	37074	1317
v8\CommandLine.java	2284	72
v8\code\CompleteClassReader.java	3205	99
v8\comp\ConstFold.java	15453	438
v8\util\Context.java	1908	76
v8\util\Convert.java	9801	298
v8\code\CRTable.java	18336	575
v8\code\CRTFlags.java	1063	34
v8\comp\Enter.java	37182	962
v8\util\Enumeration.java	362	17
v8\comp\Env.java	2816	111
v8\util\FatalError.java	943	32
v8\util\FileEntry.java	3527	163
v8\code\Flags.java	4142	167
v8\comp\Flow.java	42127	1177
v8\comp\Gen.java	74740	2069
v8\util\Hashtable.java	3740	138
Index.html	4758	111
v8\Index.html	1031	36
v8\code\Index.html	1127	37
v8\comp\Index.html	875	33
v8\parser\Index.html	1091	38
v8\tree\Index.html	1110	37
v8\util\Index.html	931	35
v8\comp\Infer.java	1678	58
v8\comp\Items.java	23419	866
v8\JavaCompiler.java	15451	469
v8\parser\Keywords.java	6641	221
v8\code\Kinds.java	1063	55
v8\util\LayoutCharacters.java	938	46
v8\util\List.java	6995	286
v8\util\ListBuffer.java	4102	200
v8\util\Log.java	16137	522
Main.java	1328	49
v8\Main.java	25301	680
v8\util\Name.java	17434	563
v8\util\Options.java	1273	47
v8\util\Pair.java	980	39
v8\parser\Parser.java	65983	2338
v8\code\Pool.java	4044	156
v8\util\Position.java	1379	51
v8\tree\Pretty.java	22228	885
v8\comp\Resolve.java	49031	1281
v8\Retro.java	655	31

```
文件                                                    大小     行数
------------------------------------------------------------------
v8\parser\Scanner.java                                29583   1248
v8\code\Scope.java                                     7725    279
v8\util\Set.java                                       2660    103
v8\code\Source.java                                    1713     63
v8\code\Symbol.java                                   30644    986
v8\code\Symtab.java                                   19809    464
v8\comp\Symtab.java                                   15647    321
v8\code\Target.java                                    4714    132
v8\comp\Todo.java                                      1570     66
v8\parser\Tokens.java                                  5593    122
v8\comp\TransInner.java                               73592   2011
v8\comp\TransTypes.java                               12518    396
v8\tree\Tree.java                                     35786   1457
v8\tree\TreeInfo.java                                 16667    644
v8\tree\TreeMaker.java                                15496    568
v8\tree\TreeScanner.java                               5871    264
v8\tree\TreeTranslator.java                            8641    322
v8\code\Type.java                                     45461   1495
v8\code\TypeTags.java                                  1996    119

总文件数:       74
总字节数:       1070759
总行数:         34456
```

为了更清晰地介绍GJC的总体结构，我们使用工具生成了GJC源代码树的结构示意图，如图1-1所示。

```
javac
|-- Main.java
|-- index.html
`-- v8
    |-- CommandLine.java
    |-- JavaCompiler.java
    |-- Main.java
    |-- Retro.java
    |-- code
    |   |-- ByteCodes.java
    |   |-- CRTFlags.java
    |   |-- CRTable.java
    |   |-- ClassFile.java
    |   |-- ClassReader.java
    |   |-- ClassWriter.java
    |   |-- Code.java
    |   |-- CompleteClassReader.java
    |   |-- Flags.java
    |   |-- Kinds.java
```

图1-1 GJC的源代码树

```
|    |-- Pool.java
|    |-- Scope.java
|    |-- Source.java
|    |-- Symbol.java
|    |-- Symtab.java
|    |-- Target.java
|    |-- Type.java
|    |-- TypeTags.java
|    `-- index.html
|-- comp
|    |-- Attr.java
|    |-- AttrContext.java
|    |-- AttrContextEnv.java
|    |-- Check.java
|    |-- ConstFold.java
|    |-- Enter.java
|    |-- Env.java
|    |-- Flow.java
|    |-- Gen.java
|    |-- Infer.java
|    |-- Items.java
|    |-- Resolve.java
|    |-- Todo.java
|    |-- Todo.ksm
|    |-- TransInner.java
|    |-- TransTypes.java
|    `-- index.html
|-- index.html
|-- parser
|    |-- Keywords.java
|    |-- Parser.java
|    |-- Scanner.java
|    |-- Tokens.java
|    `-- index.html
|-- resources
|    |-- compiler.properties
|    |-- compiler_ja.properties
|    |-- javac.properties
|    `-- javac_ja.properties
|-- tree
|    |-- Pretty.java
|    |-- Tree.java
|    |-- TreeInfo.java
```

图1-1 （续）

```
|   |-- TreeMaker.java
|   |-- TreeScanner.java
|   |-- TreeTranslator.java
|   `-- index.html
`-- util
    |-- Abort.java
    |-- Bits.java
    |-- ByteBuffer.java
    |-- Context.java
    |-- Convert.java
    |-- Enumeration.java
    |-- FatalError.java
    |-- FileEntry.java
    |-- Hashtable.java
    |-- LayoutCharacters.java
    |-- List.java
    |-- ListBuffer.java
    |-- Log.java
    |-- Name.java
    |-- Options.java
    |-- Pair.java
    |-- Position.java
    |-- Set.java
    `-- index.html
```

图1-1 （续）

javac目录下的Main.java是整个GJC程序的入口，它调用v8目录下的程序来完成具体功能。V8代表的是版本8，该目录下的4个源代码目录是GJC编译器的主体部分，它们分别是/code、/comp、/parser和/util。

/parser子目录下的源程序是GJC编译器的前端，它进行了词法分析和语法分析，把java源程序转换为语法树形式的中间形式，这两部分在一遍中完成。其中Scanner.java是词法分析程序，Parser.java是语法分析程序，Keywords.java定义了关键字，Tokens.java定义了java语言的单词符号（token）。注意，GJC中的词法分析程序和语法分析程序都是手工编写的，没有借助工具自动生成。

GJC编译器的中间形式采用抽象语法树的形式，抽象语法树使用的数据结构和接口都定义在/tree目录下的源文件中。Tree.java定义了Tree作为抽象语法树结点的基类，还定义了表示各种语法成分结点的派生类。TreeMaker.java定义了一个抽象语法树的工厂类。TreeTranslator.java和TreeScanner.java分别从抽象语法树的访问者类（class Visitor）派生了两个类，进一步定义了对抽象语法树进行翻译和扫描操作的接口。

/util目录下主要是编译器要用到的各种辅助类的源代码，例如定义了GJC使用的列

表、集合、哈希表、对、枚举和位置（position）等。符号表管理中使用的名字表也在这里定义。

/code目录下主要集中了符号表相关和Java虚拟机类文件格式（Class File Format）读写相关的源文件。其中Symbol.java定义了GJC中使用的符号。一个符号包括符号的种类（kind）和类型（type）等信息（在第4章中会详细说明），种类和类型的区别以后在介绍符号表的实现时会详细说明，它们分别在code/目录下的Kind.java和Type.java中定义。Symtab.java定义了GJC使用的符号表，Scope.java完成了上下文管理的功能，ByteCodes.java定义了Java字节码的编码，Code.java实现了代码缓冲区的管理。ClassFile.java、ClassReader.java、ClassWriter.java实现了Java虚拟机类文件格式和对其读写的功能。

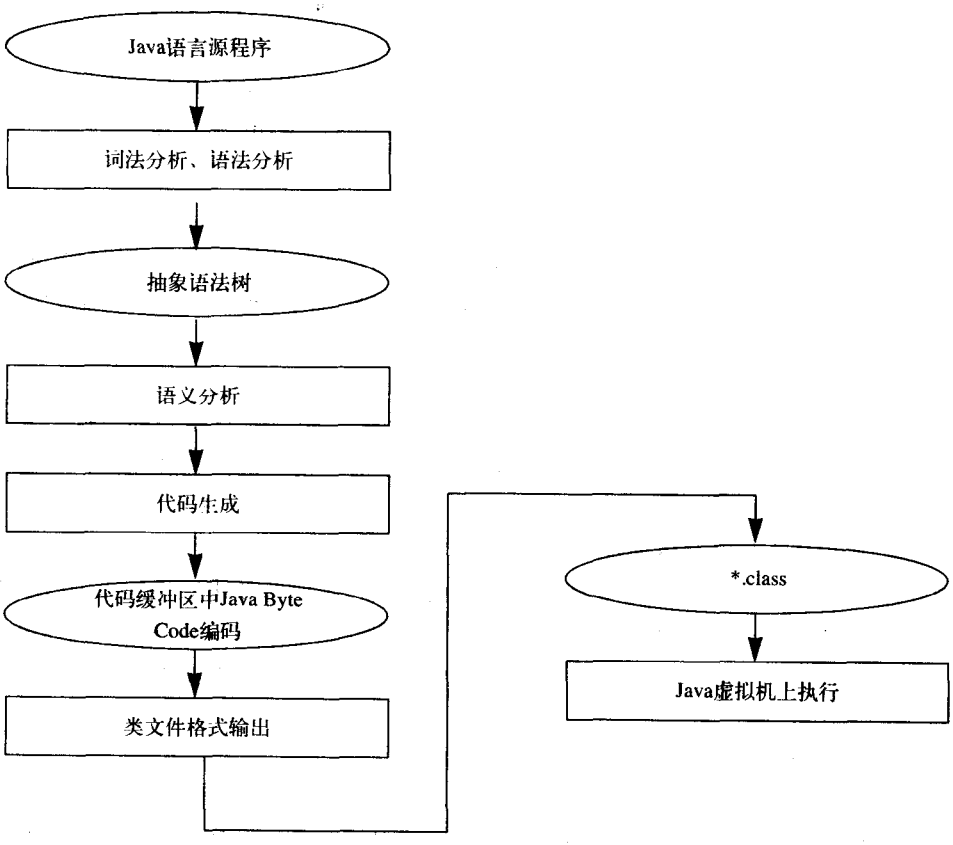


图1-2　GJC的编译流程

语义分析、代码生成、内嵌类和泛型相关的源代码都集中在/comp子目录下。GJC中的语义分析程序遍历抽象语法树，进行各种语义检查，是作为编译器中单独的一遍来完成的。它的源代码主要包括Attr.java和Check.java。GJC的代码生成部分也是作为编译器中独立的一遍来完成。为Java语言的各种结构生成代码的程序在Gen.java中实现。/comp

子目录下，TransInner.java和TransType.java分别实现了GJC中内嵌类和泛型相关的内容，两者在GJC中都是作为独立的一遍分别实现的。

GJC编译器执行的大致流程如图1-2所示。图的左边每个方框代表编译器的一遍，椭圆代表编译器的每一部分的输入或者输出数据。图的右边表示编译结果可以在Java虚拟机上执行。图中的各个部分将在本书后续章节中详细介绍，这里先给出一个简单说明。词法分析的功能在GJC的Scanner类中实现，语法分析的功能在Parser类中实现，这两个功能在编译器的一遍中完成。语法分析程序调用词法分析程序中的nextToken()方法取得每一个单词符号，语法分析程序负责通过分析的结果来建立抽象语法树。抽象语法树的具体表示在GJC中由Tree类及其派生类定义。语义分析的功能在GJC中由Attr类来实现，作为编译器中独立的一遍，它遍历抽象语法树检查其正确性。代码生成的功能在GJC中由Gen类来实现，作为编译器中独立的一遍，它遍历抽象语法树，为每一种语言结构生成其对应的代码，在代码缓冲区中输出Java字节码（Byte Code）编码。代码缓冲区在GJC中由Code类来管理。类文件格式的输出在GJC中由ClassWriter来实现。

1.4 实验设计

编译原理是一门实践性很强的学科。如果只学习原理而不进行具体的设计实践，很难真正掌握这门学科的精髓。为了帮助读者在理解和掌握编译原理和基本概念的同时掌握现代编译系统的概貌，本书设计了三个实验：

1. 为Java语言增加默认参数的特性

实验一就是通过改写Java语言的编译器来为Java语言增加默认参数的特性。通过该实验，读者可以进一步了解语言的演进，熟悉编译器各个部分的结构和功能，了解同一特性在不同语言中不同的实现技术。

2. Java虚拟机上的PL/0编译器

PL/0语言被很多大学开设的编译原理课程所采用。本实验要求采用Java语言实现一个可以在Java虚拟机上运行的PL/0编译器。通过该实验，读者可以掌握使用Java语言实现一个简单编译器的方法，熟悉编译器各个部分的实现方法，掌握将生成的代码写入到规定文件格式中的技术。

3. 使用工具自动生成词法分析器和语法分析器

本实验要学习使用工具自动生成词法分析器和语法分析器，用来替换GJC编译器中的词法分析程序和语法分析程序。通过该实验，读者可以学习工具的使用，培养解决实际工程问题的能力，了解设计与实现一个实际的高级语言编译器所面临的各种问题。

为了使读者能够顺利地完成实验，本书不仅给出了一些实验的参考文献和代码，而且还为每个实验给出了实验指南和参考难度。读者可以根据自己的兴趣对实验进行扩充，也可以根据实际的教学情况对实验进行选择。

此为试读，需要完整PDF请访问：www.ertongbook.com