HZ BOOKS
华章教育

高等院校计算机教材系列

编译原理

苏运霖 编著

本书将为教师提供电子教案

机械工业出版社
China Machine Press

高等院校计算机教材系列

编译原理

苏运霖 编著

机械工业出版社
China Machine Press

本书全面介绍编译原理的相关内容，包括词法分析、上下文无关文法和下推自动机、语法分析属性文法及其分析等，特别对面向对象的编译和网格计算的编译进行了介绍。本书内容翔实，融合作者多年来的教学心得，可作为高等院校相关专业本科生的教材，也可供从事相关工作的技术人员参考。

图书在版编目（CIP）数据

编译原理/苏运霖编著. -北京：机械工业出版社，2008.1
（高等院校计算机教材系列）
ISBN 978-7-111-22278-1

Ⅰ.编… Ⅱ.苏… 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137405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：朱 劼 王春华
北京京北制版厂印刷 · 新华书店北京发行所发行
2008年1月第1版第1次印刷
184mm × 260mm · 20.25印张
定价：33.00元

前　言

从20世纪80年代中期开始，我在暨南大学讲授编译原理这门课程。在几十年的教学中，我深切地感到国内高校缺乏一本实用的、高质量的编译原理教材，特别是在进入21世纪之后，这个问题愈加突出。主要表现在以下几个方面：

1）早年出版的被认为较好的教材内容已显陈旧，但并未及时更新内容，以适应新形势、新发展。

2）很多新推出的教材却沿用既有的框架来写，没有反映编译原理课程和技术的发展。

3）即使是一些国外教材也存在内容庞杂，不分主次、叙述冗长的问题。

面对国内急需一本能够反映最新知识发展且具有较高质量的编译原理教材的形势，我决定结合多年的教学经验和研究成果编写一本满足广大师生需求的优秀教材。

概括而言，本书与市场上的同类书相比，具有以下特色：

1）本书将人的智能中对于语言的理解同编译联系起来。

2）本书创立了源语言、目标语言和编译语言三者关系的符号，通过这个符号可以很形象地表达复杂的多层翻译过程。

3）本书第一次明确地阐明自动机与文法的所谓等价。其实它们并非真正地等价，而是要附上条件后才可以等价。但附上的条件实际上又违背了自动机或文法原来的定义。

4）对LL(1)和LR(1)文法进行了明晰、清楚的表达，使读者能知其然也知其所以然。

5）除了介绍传统方法之外，还介绍了形式化的方法。因为形式化方法在现代科学技术发展中的地位日益突出，所以介绍形式化方法对于读者有重大意义。

6）本书介绍了面向对象语言的编译，弥补了许多国内教材对此问题没有涉及的缺憾。

7）本书介绍了并行编译和网格计算语言的编译。毫无疑问，在本世纪里，并行编译将成为编译程序的主角。因为大量并行计算机的出现和网格计算的出现，必然会使传统的串行编译失去它独尊的地位。

正是上述特色使本书独树一帜，而其中绝大部分观点和结论都来自作者本人的创见和工作体会。

20多年来，我的学生们不断动员我写一本编译原理的教材，因为他们从我的讲授中也感受到我国编译原理教材的不足，他们的鼓励是我力量的源泉。他们许多人可能没有机会阅读这本教材了，但我仍愿意把这本教材献给他们，以表示我对他们的感谢。同时，我也要把这本书郑重献给我国计算机科学专业的师生，献给从事编译程序方面工作的研究人员、设计人员，以及这门课程的自学者。

在此，我要感谢广西梧州学院计算机科学与技术系和电子信息工程系的师生和领导，他们为我的写作提供了许多便利。我还要感谢机械工业出版社华章分社对我的大力支持，没有他们的支持，就不会有本书的问世。

虽然我自认为本书内容足以满足我的写作初衷，但它肯定不是完美的，书中难免会有缺点和错误，我衷心地希望读者予以批评指正。

编者

广西梧州市

2007年3月

打破多媒体光盘神话　铸造计算机图书精品

小时候看过一部电视连续剧《陈真》，其中的武状元黄埔一彪，出场时可谓“威猛无敌”。自从被日本浪人佯装醉酒“一探究竟”后，顿然间威风扫地。原来他不过是一个假冒的武状元而已，没有什么真才实学。

多媒体（multimedia）技术是一种具有集成性、实时性和交互性的计算机综合处理文本、图形、图像和声音信息的技术，如同“武状元”一样“威猛无敌”。自从与光盘“联姻”后，就被淹没在一片“铜臭”之中。真“武状元”也变成了假“武状元”。除了增加消费者的购书成本外，并没有发挥“特大作用”。我也写过附带光盘的IT图书，不过那是真“武状元”。当然有点贵。一直以来，我希望能够写一些具有“武状元”品质，而又让利广大消费者的IT图书精品。这种愿望终于实现了，电子工业出版社给我提供了一次展示真“武状元”品质的机会。当我精雕细琢铸造本书时，一直在为广大消费者展示一个真“武状元”的本来面目而努力。

2006年6月微软面向中国全体员工发出了试用Office 2007 Beta 2简体中文版的号召，同年12月人们期待已久的中文版Office 2007终于露出了庐山真面目。Office 2007是微软Office产品史上最具创新与革命性的一个版本。全新设计的用户界面、稳定安全的文件格式、无缝高效的沟通协作、面向21世纪的功能设计，为高效、愉快的商务办公提供可能。

Excel 2007是Office 2007的组件之一，是一款功能十分强大的数据处理软件，具有强大的计算和分析能力，以及出色的图表功能，能够胜任从简单的家庭理财到复杂的财务分析、计算等，是目前最流行的电子表格处理软件之一。本书以作者多年的实践经验为基础，以“上机操作实践”为案例，用13章的篇幅，图文并茂地阐述了Excel 2007在公式、函数和图表等相关领域中的广泛应用，并科学合理地编排了本书的知识结构：

1. 学习目标点击

通过两个小标题阐述“本章学什么”、“重点掌握什么”。

2. 基础知识精讲

这是本书的主体，对本章的知识点、重点、难点逐一进行精讲。使读者能够准确理解知识要点，把握知识的内在联系。

3. 上机操作实践

每小节后面的“上机操作实践”既是及时地对所学知识的巩固，又是读者实践能力提高的关键，使读者既学习了基础理论，又及时地验证了自己对所学知识的掌握程度。

4. 温馨提示

“温馨提示”既是章节的一朵小浪花，也是本章节内容的及时补充，主要用来介绍Excel发展的最新成果，并对Excel使用过程中的技巧进行及时点拨。

5. 典型案例指导

以Excel 2007在公式、函数和图表等相关领域中的广泛应用为基础，以本章主要知识点为依托，选取典型案例设计为范例，在教师的指导下，让读者参与到案例的制作中来，激发读者学习的热情和兴趣，培养读者的创新能力。

6. 基础知识检测

以巩固本章知识为目的进行设计，难度适中、题型新颖、答案简明。

7. 评价思悟提高

及时总结全章主要知识点及主要注意事项，让读者记录学习的随感，写下产生问题及如何解决的过程，使读者在总结、整理的基础上得到不断的提高。

在使用本书配套资料中的素材与源文件时，应该注意以下几点：

- 素材与源文件中文件夹的序号与书本中的序号一致。如果你需要使用什么样的素材，只需要按照书本上的相应序号在相应的文件夹中，就可以找到。
- 素材与源文件中案例名称与相对应的章节中制作的案例或者功能应用相一致。在许多“上机操作实践”中，为了节省篇幅，使读者获得更为超值的享受，对其中的基础部分的操作步骤大多采用直接打开某个文件的写作方式。对于那些学习稍感困难的读者，可以通过打开本书提供的相应源文件，从中不难获得相应的答案。这些源文件同时也为你进行图表设计、灵活运用函数与公式提供了重要的参考依据。
- 建议读者使用时将素材与源文件内容拷贝到电脑硬盘上，以加快文件运行速度。

最后，感谢和我们共同完成此书的合作者，他们是李立祥、俞园园、周其国、碗舒萍、周易华、李晓宇、周静聪、李水明、施捷利、石凯、周详水、严朱莉、王丽丽、李松桥、江水贵、卢跃进。感谢北京美迪亚电子信息有限公司的各位老师，感谢龙腾国技图书工作室的各位老师，谢谢你们的帮助和指导。由于本人水平有限，书中不可避免地存在着或多、或少的不足之处，欢迎大家批评指正！

为方便读者阅读，若需要本书配套资料，请登录“华信教育资源网”（http://www.hxedu.com.cn），在“资源下载”频道的“图书资源”栏目下载。

目　录

前　言
第1章　概论 ……1
1.1　语言和人类 ……1
1.2　语言和计算机 ……2
1.3　语言和编译 ……6
1.4　程序设计语言的编译 ……7
1.5　一个语句的编译举例 ……9
1.6　编译的遍数 ……11
1.7　本书的组成 ……11
习题 ……13
第2章　文法和语言 ……14
2.1　预备知识 ……14
2.2　文法 ……15
2.3　语言 ……18
2.4　文法所生成的语言 ……20
2.5　图灵机 ……22
2.6　有关文法和语言的问题 ……29
习题 ……30
第3章　有限自动机和正则表达式 ……34
3.1　确定的有限自动机 ……34
3.2　不确定的有限自动机 ……37
3.3　带有ε的有限自动机 ……42
3.4　正则表达式 ……44
3.5　两路有限自动机 ……49
3.6　正则文法 ……53
3.7　关于正则语言的判定 ……61
习题 ……62
第4章　词法分析 ……66
4.1　词法分析的作用 ……67
4.2　词法分析程序的输出 ……76
4.3　错误处理 ……78
习题 ……78
第5章　上下文无关文法和下推自动机 ……80
5.1　上下文无关文法 ……80
5.2　上下文无关文法的性质 ……83
5.3　下推自动机 ……90
5.4　下推自动机和上下文无关语言 ……93
习题 ……98
第6章　语法分析 ……104
6.1　LL(1)语法分析 ……104
6.2　LL(1)文法的确定 ……107
6.3　LL(1)语法分析方法 ……110
6.4　自底向上的语法分析 ……115
6.5　LR(1)语法分析方法 ……118
6.5.1　LR(0)语法分析 ……118
6.5.2　SLR(1)语法分析 ……120
6.5.3　LALR(1)语法分析 ……122
6.5.4　LR(1)语法分析 ……124
6.5.5　LL(1)语法分析方法和LR(1)语法分析方法的比较 ……130
习题 ……132
第7章　属性文法及其分析 ……136
7.1　属性文法 ……136
7.2　依赖图和属性计算 ……139
7.2.1　动态属性计算 ……143
7.2.2　循环处理 ……145
7.3　L属性文法和S属性文法 ……146
习题 ……148
第8章　编译程序设计的代数方法 ……149
8.1　源语言 ……149
8.2　代数基础和推理语言 ……154
8.2.1　代数基础 ……155
8.2.2　推理语言 ……160
8.3　一个简单的编译程序 ……178
8.3.1　规范形式 ……178
8.3.2　规范形式的归结 ……179
8.3.3　目标机器 ……182
8.3.4　表达式的化简 ……183
8.3.5　控制的消除 ……185
8.3.6　数据求精 ……187

8.3.7 编译过程 ……191
8.4 过程、递归和参数 ……192
8.4.1 记号 ……193
8.4.2 过程 ……193
8.4.3 递归 ……195
8.4.4 带参数的程序 ……197
8.4.5 带参数的过程 ……199
8.4.6 带参数的递归 ……200
8.4.7 讨论 ……202
8.5 小结 ……203
习题……204
第9章 中间代码的生成 ……205
9.1 为什么需要有中间代码生成阶段 ……205
9.2 中间代码语言 ……205
9.2.1 图形表示 ……206
9.2.2 后缀表示 ……208
9.2.3 四元组代码 ……209
习题……223
第10章 纠错与优化 ……226
10.1 错误检测和恢复 ……226
10.2 语法错误检查 ……227
10.2.1 LL(1)分析程序的错误处理 ……229
10.2.2 LR(1)分析中的错误处理 ……229
10.3 语义错误检查 ……230
10.4 程序的优化 ……230
10.5 程序优化的几个主要途径 ……234
10.5.1 公共子表达式的删除 ……234
10.5.2 副本传播 ……234
10.5.3 废代码的删除 ……235
10.5.4 循环优化 ……236
习题……237
第11章 存储管理 ……239
11.1 全局分配策略 ……239
11.2 动态分配 ……242
11.2.1 栈式分配 ……242
11.2.2 堆式分配 ……243
11.3 存储空间的回收 ……245
11.3.1 基本垃圾收集算法 ……245
11.3.2 编译程序对垃圾收集程序的支持…246
11.3.3 引用计数 ……247
11.3.4 标记和扫描 ……248
11.3.5 两空间复制 ……249
11.3.6 紧缩 ……250
11.4 参数传递 ……250
11.4.1 值调用 ……251
11.4.2 引用调用 ……251
11.4.3 复写-恢复调用 ……251
11.4.4 换名调用 ……251
习题……252
第12章 目标代码生成 ……254
12.1 代码生成程序设计的有关问题 ……254
12.1.1 代码生成程序的输入 ……254
12.1.2 目标程序 ……255
12.1.3 存储管理 ……255
12.1.4 指令选择 ……255
12.1.5 寄存器分配 ……256
12.1.6 计算顺序的选择 ……257
12.1.7 代码生成的方法 ……257
12.2 目标机器MMIX ……257
12.3 MMIX的汇编语言 ……274
12.4 MMIXAL目标代码的生成 ……279
12.4.1 表达式逆波兰表示的翻译 ……279
12.4.2 表达式三元式的翻译 ……280
12.4.3 表达式四元组形成的翻译 ……280
12.4.4 表达式的翻译 ……281
12.4.5 表达式的语法树形式的翻译 ……282
12.4.6 各种语句的翻译 ……282
习题……284
第13章 面向对象语言的编译 ……286
13.1 对象及其编译 ……286
13.2 对象的特征 ……287
习题……294
第14章 并行语言的编译……296
14.1 并行机和并行计算的提出 ……296
14.2 并行程序设计 ……298
14.2.1 共享变量和管程 ……299
14.2.2 消息传送模型 ……300
14.3 面向对象的语言 ……301
14.4 Linda元组空间……301
14.5 数据并行语言 ……303

14.6 隐式并行程序的代码生成 ……304
14.6.1 区域的类型 ……305
14.6.2 区域的形成 ……306
14.6.3 区域的几个调度算法 ……309
习题……309
第15章 网格计算的编译……310
15.1 网格计算的兴起与其内涵 ……310
15.2 网格计算模型 ……311
15.2.1 分组路由 ……312
15.2.2 线性阵列中的分组路由 ……313
15.3 网格计算的编译 ……315
习题……316
参考文献 ……318

第1章 概 论

1.1 语言和人类

什么是编译程序？简单地说，编译程序的功能是把一种用程序设计语言编写的程序翻译成为可以由计算机执行的程序代码。为了具体阐述编译程序，有必要对有关问题深入探讨一下。

语言是人类彼此沟通，表达彼此的意图和情感，描述事物和表达理解的一种工具，是人类智能的产物，也是智能的表现。在人类进化之初，还没有语言，在经历了一段漫长的时间之后，才产生出口头语言。这是人类在语言上的一个突破，也是人类文明的一次突破。从口头语言发展到书面语言，这中间又经历了更为漫长的时间。书面语言的出现是人类在语言上的又一个意义更为深远的重大的突破。语言是思维的一个工具或者载体。如果没有语言，人的思维过程就不可能是完整、连续和深入的。因为在没有语言时，人们根本无法向其他人表达他的思想，当他进行记忆时，也不可能描述涉及许多对象和复杂情节的过程。书面语言比口头语言更进一步，它不仅可以实现当代人之间的沟通，而且可以贯通古今，使现代人通过文字了解古代的事物。利用书面语言，不仅可以实现近距离的人们的沟通，而且可以使远距离的人们进行沟通。特别是在现代通信工具，如计算机网络以及电视、电话等的帮助下，人与人之间的沟通变得更为快捷、便利，更能保证信息的安全性、保密性，即书面语言改变了人们沟通的时空范围。

由于所处的地理位置以及进化的条件不同，人类形成了不同的种族和民族，各种族和民族所用的语言也不同。目前世界上语言的种类已达到数千种，但这比历史记载的语言种类少了很多，因为有的语言由于使用者的减少已经灭绝，还有些语言仅是口头语言而没有相应的书面语言，所以世界上真正有影响的语言不过几十种。在这些语言之间，为实现彼此间的相互理解，就需要进行翻译，即把一种语言所表达的信息翻译成为另一种语言。今天随着以知识为基础的全球化经济的发展和科学技术的发展，语言翻译，无论是书面翻译，还是口头翻译，都成为热门的行当。

以口头翻译而言，涉及三个人或三个对象，即使用语言A的人甲，要理解甲所表达的信息的人乙，他使用语言B。于是需要有第三个人丙，他既要懂得语言A，又要懂得语言B，因此我们可以把这种情形用图1-1表示出来。

反过来，乙理解了甲的意图之后，他也可能希望把他的答复或意图传送给甲，因此他可能又要丙去承担翻译的角色，或者依靠另一个人丁来完成这个任务。于是我们就会有图1-2。

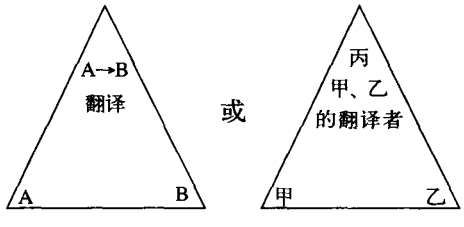

图1-1 口头翻译过程一

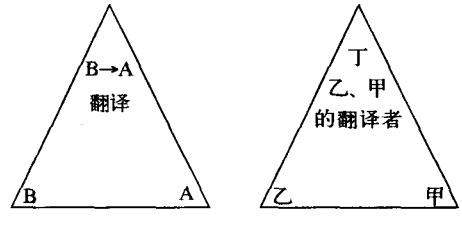

图1-2 口头翻译过程二

我们可以把这两者组合起来，成为图1-3。

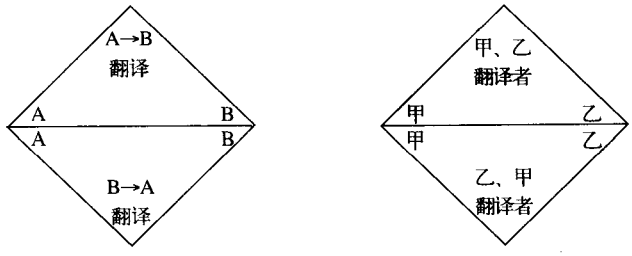

图1-3 双向翻译

这表示A语言翻译成为B语言，而B语言（具有不同内容）又被翻译成A语言。或者，甲的谈话经过翻译传达给乙，而乙经过翻译把自己的意图传递给甲。

如果甲使用的是语言A，乙使用的是语言B，丙使用的是语言Z，丁使用的是语言C：但丙同时懂A语言和B语言，丁同时懂B语言和A语言。于是，甲和乙的沟通必须同时依靠丙和丁两个人才能完成（因为丙是从事A语言到B语言的翻译，而丁是从事B语言到A语言的翻译）。这就出现了图1-4所示的模式。

图1-4 连续的翻译过程

在这里，我们假定丁恰好掌握B和C两种语言，因而他可以同时担任B和C两种语言的翻译者的角色，但假若丁只懂得语言B，那么为了将语言B翻译成语言C，那就还需要另一个专门的翻译。

对于书面语言及其翻译而言，情况是类似的。但不同的是，两种语言之间的翻译不再涉及三个人，而只涉及两种语言，即被翻译的语言以及翻译成的语言。因此三角形的顶部不再是另外一个人，而是翻译成语言的使用者。从更广义的角度考虑，则有下述情况：某本书由作者用A语言写成，为把该书提供给使用B语言的读者，就需要懂得这两种语言的翻译来把它翻译成B语言。这个翻译既可以是人类，也可以是翻译工具。如果由人来翻译，那么这个人可以就是作者本人，他把先前用某种语言写成的东西翻译成另外一种语言，所以三角形上仅涉及底部左边和右边的两种语言，以及顶部的翻译过程，而顶部是他从事的翻译过程或所使用的翻译工具。

1.2 语言和计算机

众所周知，现代计算机是人类最伟大的发明之一，反映了人类科技的最新发展。计算机依靠它的运行来解决人类设定的各种问题，而计算机的运行是依靠事先编制好的一组指令组成的程序来完成的。组成程序的指令便可以看作计算机所使用的语言。这种语言由0和1的序列组成。

因此，要让计算机运行，就需要掌握计算机的语言——它的指令系统。然而，对于一般的计算机使用者而言，要求他们掌握计算机繁琐、乏味而又不直观的语言，无疑是一个过分的要求。

当计算机刚刚问世时，人们确实也经历过直接使用计算机语言编写程序的时期，即手编程序时期。计算机的指令包含地址以及其他控制代码等。因此，要使用指令系统编写程序，不仅要确定每一个步骤应执行的操作，而且要把和每条指令相对应的地址信息等都确定下来，这无疑需要极大的工作量。为了摆脱直接使用机器语言的束缚，在计算机问世后不久，人们就开始寻求解决方案。最初，人们仅仅是把指令的操作码换成为便于记忆的符号，如利用ADD代表加、SUB代表减、MUL代表乘、DIV代表除，或者更简单地用+、－、×和/来表示加减乘除运算，以及使用文字地址来代替实际的数值地址，等等。进行上述改进后的语言自然就不是计算机语言，或者原来的计算机指令系统了，尽管它和原来的指令系统基本上是一一对应，完全等价的。这是人类在摆脱计算机的语言方面迈出的第一步。尽管步子不大，但却是十分关键的。这表明人类可以不再受到计算机指令系统的限制，可以使用更为方便的语言来为计算机编写程序。这样的语言称为汇编语言。在这里，前边给出的三角形模式仍然适用。如图1-5所示，三角形底部左边的是以汇编语言写成的任何程序，我们称之为源程序。三角形底部右边的是以计算机指令写成的和左边完全等价的程序。我们称之为目标程序。而顶部现在是把源程序翻译成目标程序的翻译程序。我们把它称之为汇编程序。因此汇编程序要在计算机上把源程序翻译成目标程序，那么它在计算机上必须是可执行的，也就是必须用机器指令写成。

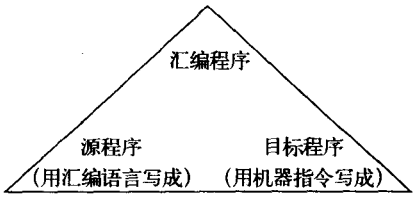

图1-5 计算机汇编语言的翻译

因此，汇编程序是最初形式的编译程序。但由于源程序所使用的语言，即汇编语言，只是机器指令系统的简单改写（如其操作码称为指令系统的记忆码），因此它又称为低级程序设计语言。这里所说的“低”，是指它是面向机器的（低层的），而不是面向人类本身（高层的）。汇编程序也是初级形式的编译程序，因为它还没有使用后来发展起来的高级语言的编译技术。

在汇编语言取得成功之后，人们开始设计面向人类的高级程序设计语言。这种高级程序设计语言的特点是摆脱了计算机指令系统的限制，采用自然语言（一般为英语）的一个子集并规定本身的语法，来描述人们在计算机上所要完成的某种任务或过程。因此，这样的语言也称为面向过程的语言，简称为过程化语言（Procedural Language）或者命令式语言（Impretive Language）。最早的程序设计语言有FORTRAN（Formula Translation，公式翻译），Algol（Algorithm Language，算法语言），COBOL（Common Business-Oriented Language，面向普通商业的语言）等。20世纪50年代末到20世纪60年代初是高级程序设计语言分枝繁衍的年代。当时，各种专用的（面向某种过程或应用目的的）以及通用的（面向一般过程的）程序设计语言如雨后春笋般地竞相出现，其中一些语言具有鲜明的风格，适用于不同的应用领域。例如，APL（A Programming Language，一种程序设计语言）是会话型语言；PL/1（Programming Language 1，程序设计语言1）适用于科学计算；Pascal（因纪念法国数学家Pascal而得名）是为适应结构化程序设计的思想而设计的；Lisp（List Processing，列表处理）；Prolog（Programming for logic，逻辑程序设计）；Snobol主要用于字符串处理等，稍后出现的语言有用于数组处理的SIMSCRIPT，主要用于模拟的SIMULA和MODULA，其后又出现了一个通用的大型语言Ada，这是为了纪念为计算机的先驱巴贝奇·查尔斯（Babbage Charles）（1792-1871）的差分机进行程序设计的Ada而命名的，还有一个最初用于编写UNIX操作系统的核心程序，而后广泛用于普通程序设计的C语言。

上述语言除个别特例外，基本上都属于面向过程的语言。在20世纪60年代末出现软件危机之后，和结构化程序设计的思想平行的另一种解决方法——面向对象的软件设计方法被提

出，由此产生了面向对象的程序设计语言。例如，在C语言的基础上发展起来的C++，以及随后又在C++基础上发展起来的Java语言，还有SMALLTALK语言等，它们都属于面向对象的程序设计语言。

硬件的不断发展也向软件提出了新的要求。分布式系统、并行计算机系统和计算机网络系统等新的计算机结构也对计算机的程序设计提出了新的要求和挑战，适应这些要求的新的程序设计语言也相继问世。

不论语言本身如何变化，有一点是不变的，即用这些语言编写的源程序，要变成可以在计算机上运行的目标程序，都必须由该语言的编译程序进行编译。也就是说，它具有图1-6所示的模式。

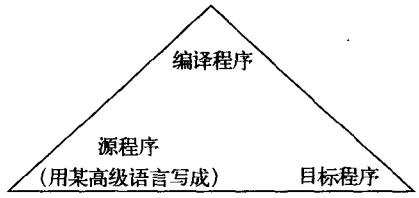

图1-6 编译过程

这里的编译程序要用机器语言编写，才能够把源程序翻译成目标程序，因为只有用机器语言编写，它才能运行。然而，直接用机器指令系统来编写编译程序并非易事，而且工作量很大。于是人们想到也用高级语言来写编译程序，再用稍微简单些的用机器指令写成的编译程序对于前一个编译程序进行编译，把它转化成为可以运行的真正的编译程序，再用它来对源程序进行编译，所以我们就有图1-7所示的模式。

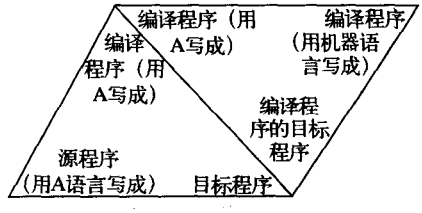

图1-7 连续的编译

在图1-7里的两个三角形中，首先运行的是第2个三角形。在该编译程序（因为它用机器语言写成的）运行后，得到了三角形底部右边的目标程序。它用来替代第1个三角形顶部的编译程序，通过运行它就产生出我们真正需要的源程序的目标程序了。

这个模式可以进一步扩展。比如，用A语言写成的源程序，用B语言来写A语言的编译程序，得到源程序的目标程序。但这显然是不能执行的，因此要用B语言编写的编译程序。对于B语言的一个源程序，可以用由C语言写成的编译程序来进行编译，得到其目标程序。对于由C语言写成的编译程序，再由用机器语言写成的编译程序在机器上运行，生成真正可以执行的编译程序。于是，如同在上述两个三角形模式中那样，逐次代入，在机器上运行，就得到A语言写成的程序的目标程序。这可以用图1-8表示。

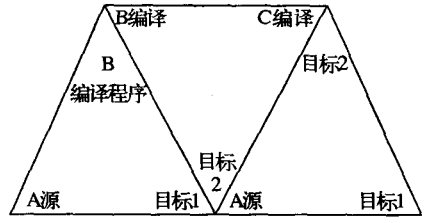

图1-8 三次连续的编译

这个过程可以扩展到任何需要的层次。只要最后一层是真正可以运行的，就可以通过逐层以目标程序替代源程序，则第一层的目标程序就可以用已经转化为可在机器上运行的它的编译程序产生出来。

在开发程序设计语言时，有许多必须考虑和处理的问题。如果这些问题处理和解决得不好，那么这样的语言就不会有生命力，也就不会为用户所接受。在开发程序设计语言时必须遵循的原则和应该达到的目标是：

1）清晰、简单和语言概念的统一。当程序员使用某种语言进行程序设计时，就如同使用该语言来说话或写字一样，因此这种语言应该为他提供思考算法的一个简单清晰的概念框架，也应该为他提供表达这些算法以便在机器上实现的工具。如果语言达不到清晰、简单和概念统一的目标，人们就绝不会乐意使用这样的程序设计语言。可以说，语言的清晰性、概念的清晰性和简单性是衡量一种语言的价值的最重要的因素。

就人们所使用的自然语言来说，包括中文、英文等广泛使用的语言在内，也并非完美无缺。它们都存在产生二义性或表述不够清晰的可能。但由于人们理解语言的能力远比计算机高，因此这很少给人类带来问题。但对于人们用来同计算机进行交互的程序设计语言来说，情况就不同了。在程序设计语言中，哪怕有一丁点的二义性，都可能使程序得不出正确结果。有时，这种语言的瑕疵又不能被轻易发现。如早期的语言Algol 60几经修改，并且使用了几乎所有人都公认为很好的方法描述工具BNF（Backus-Naur Form，即贝克斯－琀尔范式）来进行定义，而且广泛使用了许多年，才由D. E. Kunth于1967年指出它存在的问题。这距离Algol 60的正式文本的公开发表至少过去了四五年时间。可见，要真正实现我们提出的目标实属不易。这也是为什么当程序设计语言的设计进入高潮时出现了数以千计的语言，但经过时间的洗礼，如同大浪淘沙一般，大部分语言都不能避免被淘汰的命运的原因。

2）程序结构的明确性。同一种语言的语义清晰性密切相关的是用该语言编写的程序结构的明确性。这里所说的结构明确性，是指它有明确的层次，容易理解，使程序的编写、调试和修改都变得比较容易。语义的清晰性和结构的明确性是不同的，而两者又互相补充，可以共同增进程序的可读性。

3）应用的自然性。程序设计语言应该为要解决的问题提供适当的结构和操作，以及适当的控制结构。为此，它的方法也应该是自然的。程序设计语言之所以会出现迅速繁衍的时期，其主要原因之一就是人们对于自然性的探索和追求。一个在语法和语义上都特别适合于某种具体应用的语言，可以大大地简化这个领域的程序的编写。对于通用的语言来说，它是为满足不同的应用而设计的，它更应该具有自然性。

4）易于扩充。程序设计语言应允许通过简单、自然和优雅的机制来进行对它本身的扩充。几乎所有语言都为此提供了子程序定义机制。在构造大型程序时，程序员的很大一部分任务，都可看作是对语言的扩充。因为程序员必须决定，为了解决问题，他应该如何使用该语言所提供的原始特征来模拟问题的数据结构。因此从概念上说，这就相当于扩充原来的语言以包括被模拟的结构。

5）适应硬件环境的需求。迅速发展和变化的硬件环境，必然要求机器语言的程序也随之变化，特别是并行计算机系统、阵列计算机系统以及分布式计算机系统等，要求人们设计出适用于它们的程序设计语言。本书在后文中将讨论明显的并行指令计算（Explicitly Parallel Instruction Computing，EPIL）结构的计算机，它是非常长指令字（Very Long Instruction Word，VLIW）处理器的一个发展，这些硬件的发展同时要求适应于它们的程序设计语言以及对于它们的编译。

6）有效性。有效性无疑是评价某种程序设计语言优劣的一个重要准则。人们在使用程序设计语言编写程序的初期，或者说在创建程序设计语言的初期，关注的就是使用程序设计语言编写出的程序的效率。对于同一个问题，用不同的语言编写的求解程序，其长度和运行时间都可能有所不同，这就是我们所说的效率。不过，就效率而言，有几种不同的含义：

- 程序执行的效率。最初，提到效率几乎毫无例外地指程序的执行效率。这就涉及编译程序的质量问题，即编译程序能否产生执行效率高的目标程序。因此，优化的编译程序的设计，有效的寄存器分配以及有效的运行时间支持机制的设计是非常重要的。尽管程序执行的效率问题和语言设计有密切的关系，但编译程序的编译质量对于程序执行的效率有决定性影响。
- 程序编译的效率。大型生产性程序是指经常要运行的大型程序，因而它的执行效率当然是人们关注的问题。这个问题同编译的质量有关。还有另一类程序也同编译程序的效率密切相关，那就是学生程序或用于教学的编译程序的编译质量。因为典型地说，对于学生所写

的程序，我们并不用它作生产性的运行，而只关注它的正确性，所以我们只需要编译程序能有效地对程序进行编译，指出程序是否正确。所以在此情况下，重要的是要有一个快速高效的编译程序，而不是能产生优化的执行效率高的目标程序这样的编译程序。

- 程序的编写、调试和使用的效率。在程序设计语言的效率方面，其第三个含义是编写、调试和使用等方面的效率问题。比如，如果我们用某种语言来求解某类问题，结果设计、编码、调试、修改和运行这个程序所消耗的程序员的运行时间和精力最少，因此，从实际意义上说，这些才是衡量使用这种语言在计算机上解决问题的效率的最有效的标准。所以，在使用一种语言时，在这个意义下的效率要比传统意义上所关心的执行效率和编译效率更为重要。

有了上述明确的程序设计语言的设计原则之后，人们发现，设计上最好的一些语言却未必能满足这些原则。goto（转移）语句就是这样一个例子。在早期设计的语言中都设置了goto语句以实现程序执行的转移。从使得程序的执行除了顺序执行之外，也能够改变方向而言，设置goto语句的初衷是正当的。但在实际使用中，人们发现了goto语句存在的问题。1968年，埃德盖尔·狄伊克斯特拉（Edgar Dijkstra）提出了goto语句有害的论断，他力主把goto语句从程序设计语言中去除。于是科学家们围绕是否要去除goto语句进行了长时间的激烈争论，直到最后，D. E. Knuth于1974年对这个问题作了客观、科学、准确的论述，才使goto语句保留下来，但限制对它的使用。这样，在许多人的努力下，程序设计语言进入了新的阶段，这就是所谓的结构化程序设计语言的阶段，而使用这样的语言编写的程序称为结构化程序。相应地，编写这种程序的过程称为结构化程序设计。前面已经提到的Pascal程序设计语言就是结构化程序设计语言的代表。后来，在这种思想影响下出现的语言还有MODULA、C及Ada语言等。

随着人们进行的软件设计的实践的发展，以及对于程序设计过程认识的深化，产生了所谓的软件危机，即用程序设计语言所生成的程序，特别是大型程序的质量，总是难以保证，隐藏的错误总是难以发现，软件的研制周期常常滞后于计划的时间。因此人们又提出了新的思想，就是面向对象的程序设计方法，与之对应的语言就是面向对象的程序设计语言。这类语言包括SMALLTALK、C++、Java等。

网络环境和分布式环境以及并行机的出现，又向人们提出了并行程序设计和分布式程序设计的新思路，因而提出了相应的并行程序设计语言和分布式程序设计语言，如同我们前边已经提到过的那样。

1.3 语言和编译

从上述的讨论中，我们知道，用高级程序设计语言所编写的源程序都需要通过编译，才能在计算机上运行。因此，使用高级程序设计语言来编写程序就像两个人之间的沟通要经由第三个人来翻译一样。前面我们谈到，在这一过程中，人们关心的是程序执行的效率。实际上，在这一过程中，还有不同的关注目标或重点。目标不同，得到的结果也会不同。例如，考虑到“编译原理”这门课程的性质，我们自然要关注编译程序，则对于它来说，可能的目标是：

1）编写出尽可能小的编译程序。按照最省力的原则，我们当然要以编写出最小的编译程序作为自己的目标，也就是要求它能够处理基本的编译任务即可。不过，这样的编译程序可能是不完备的，它不可能处理一些复杂的情况。和这相应的可能是这个语言的子集，即仅考虑这个语言的基本成分而不是语言的全部。这种编译程序可以作为学生的实习课题，目的仅仅是为学生提供一个实践过程，让他们掌握设计编译程序的基本技巧，而对于实际应用，这种编译程序显然是不能胜任的。

2）编写出具有较好诊断能力和出错恢复能力的编译程序。编译程序应能发现用户所写的源程序的诸多错误，不仅能发现静态错误，还应能发现动态错误。在发现错误之后，它还应能确定错误来源，并且相应地进行纠错。这样的编译程序才称得上是对用户友好的，从而帮助用户保持程序的正确性。但它所得到的目标程序则未必是真正高效率的。这种编译程序适合在教学环境中使用，属于肮脏的编译程序。但它也可以作为预备性的编译程序，在通过了这种编译程序的编译之后，再运行专门进行优化的编译程序，以提高编译程序的质量。

3）产生可行和高效率的目标程序的编译程序。这种编译程序在保证从源程序产生出正确的目标程序的基础上，还要求目标程序有较高的运行效率。因此它的功能必须包括对于源程序的优化。

如果我们着眼于编译过程或者限于所得到的目标程序，则可能的目标有：

1）使编译程序对源程序进行编译所花的时间最少。如果着眼于此，则相当于我们要求编译程序以最快的速度确认源程序的正确性，然后，又以最快的速度把它翻译成目标程序，至于目标程序的运行效率如何，不在我们所考虑的范畴之内。因此，就其实质而言，这样所产生的是上述第二种编译程序。

2）使所产生的目标程序最有效。这和上一条形成对照，因为它的着眼点不在于编译过程的速度或效率，而在于所产生的目标程序的运行效率。就这一点来说，它和前边第3个编译程序目标相一致。

3）使所产生的目标程序比较小。应当注意，上一条考虑的是时间的有效性，而这里所考虑的是空间的有效性，因而两者并不等价。当然，在一般情况下，程序缩短了，运行时间也会相应缩短。但是，情况并非总是如此。比如，如果程序很短，但它却包含多次循环，那么运行时间就不一定很短。因此，这个目标更多地是从目标程序所占用的内存的数量来考虑的。如果计算机为用户提供的内存数量有限，则可以以此作为目标。

从上述分析中，我们可以看出，编写编译程序的目标可以不相同，而且不可能要求编写的编译程序同时满足所有要求。和编制其他系统一样，我们只能实现不同目标的一个折衷。

1.4 程序设计语言的编译

现在，我们专注于编译程序和编译过程本身。对于编译程序而言，由于它的目的是把一种特定的程序设计语言翻译成在某一类型的计算机上运行的目标程序，因此编译程序用于建立这两者之间的对应。换句话说，对于一种程序设计语言和一类计算机系统，就要有一种该语言的编译程序。如果有m种程序设计语言，有n类计算机系统，那我们就需要编写出m · n种编译程序。这当然不是我们所希望的情况，因为这意味着极其庞大的工作量。因此，我们要设法减少工作量。我们的办法是，找出在编译程序中，哪些是同具体机器有关的，还有哪些是同具体机器无关的。对于同具体机器无关的部分，我们使它们为所有的编译程序共用，只有同具体机器有关的部分，我们才针对不同的机器分别进行设计。正是出于这个原因，我们的编译程序都不直接用计算机的指令系统来编写，因为只有这样，才有可能不马上同具体机器发生关系。真正需要同具体机器相关联的工作应推迟到尽可能晚的时间（如使它发生在真正对源程序进行编译时，而不是在人们编写编译程序时）。通过这样的努力，可以使m种程序设计语言和n类计算机系统所需要的编译程序个数从m · n个减少到m+n个。这里仅仅简略地介绍这一思想，更详细的论述将在本书后面给出。

现在我们来谈一谈编译程序本身的工作原理。我们还是从一般的语言谈起。当我们学习一种语言时，无论是学习母语还是学习外语，首先都要学习词汇，即单个的词，这里包括它

的拼写、书写笔画、发音等。然后再学习文法，即把单个词连接成句子的规则，以组成文法上正确的有意义的句子。对于编译程序而言，它的工作过程也是这样。编译程序的工作主要包含两个部分，一是分析，二是综合。分析又包含词法分析和语法分析两个部分。词法分析从源程序的输入开始。源程序的输入被当作一个字符串流的输入，分析部分中的词法分析首先进行工作，把这个字符串分解为有意义的组成，同时检查它们是否有错误。如果发现错误，它还应当指出错误。词法分析要区分的程序中的词的种类有标识符、常数、关键词，包括程序设计语言所用的语句的名字以及运算符号、标点符号等。为了进行编译，这些词都将被转换为中间代码的形式，以使它们具有同一的格式，即程序的每个语句现在都变成用一个一个的记号（token）连接在一起的形式。

语法分析即以这些记号序列作为输入，它基于程序语言的语法来对各个句子进行分析，看看它们是否符合规定的语法。如果没有错误，则把整个源程序进一步转换成中间表示，以便在后续的综合阶段将中间表示转换为目标程序。因此，分析部分的工作流程如图1-9所示。

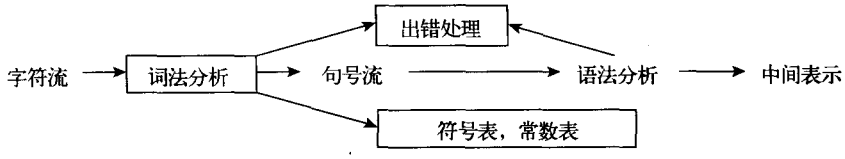

图1-9 编译中分析部分的工作

图1-9中的符号表表示对于每个标识符包含一个记录的数据结构，而常数表表示对于每个常数包含一个记录的数据结构。符号表中除标识符本身及为它分配的地址（但不是实际的内存地址，只是相对地址）外，还有表示它的各种属性的字段。这样的数据结构可以使我们快速找到每一个标识符的记录，并且还能快速地向该记录存储标识符或者从中检索该标识符。在词法分析程序工作时，当第一次遇到某个标识符时，如果词法分析程序断定它是标识符，则称为这个标识符的定义性出现，以后的出现就称为应用性出现。在定义性出现时，程序把它记入符号表中，并且按照它出现的先后顺序为它分配地址，同时依据源程序中对于这个标识符的定义或说明，把它的相关属性也存入符号表中。在应用性出现时，要按照它的定义性出现中得到的记录，把它改为中间代码的形式，还要检查这个应用性出现所蕴涵的属性是否和定义性出现相一致。如果不一致，词法分析程序就会认为这里出现一个错误。

常数表的数据结构和符号表类似。对于常数，词法分析程序首先要把每个作为字符的数字（如果是带符号的整数，还可能包含+、－号。如果是实数或浮点数，还可能包含+、－号、小数点以及指数符号）转换成与之相应的数值。在这个过程中，同样要检查它是否正确。在确认正确后，才把它放入常数表中，并给它分配一个地址，相应地记录它的属性。

有关符号表和常数表的具体细节，我们将在后面给出进一步的说明。

在词法分析阶段及随后的语法分析阶段都可以进行出错处理，甚至在后面的综合阶段（包括语义分析）仍然要进行出错处理。实际上，语法分析和语义分析阶段通常能够处理由编译程序检测出的大部分错误。词法分析阶段能够检测出的是这样的错误，即输入中字符不形成语言的任何记号；语法分析阶段通常能检测出记号流违犯语言的语法规则或结构规则的错误。在语义分析阶段，编译程序试图检测出下述构造：它具有正确的语法结构，但对于所涉及的操作却毫无意义。例如，我们要执行两个标识符相加的运算，但这两个标识符一个是数组的名字，另一个是过程的名字。出错处理应该能够在检测出一个错误之后，不立即停止编译程序的工作，而是继续进行编译工作，允许进一步检测出随后的错误。

前面我们对分析阶段进行了概述。如前所述，对于完整的编译过程而言，分析阶段完成