21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

编译原理——编译程序构造与实践教程

Principles of Compiling—A Course in Constructing and Practising of Compiler

张幸儿 戴新宇 编著

- 牢牢掌握编译程序构造的基本原理
- 深入理解C程序设计语言的特性
- 实践中领悟、实践中提升

精品系列

人民邮电出版社
POSTS & TELECOM PRESS

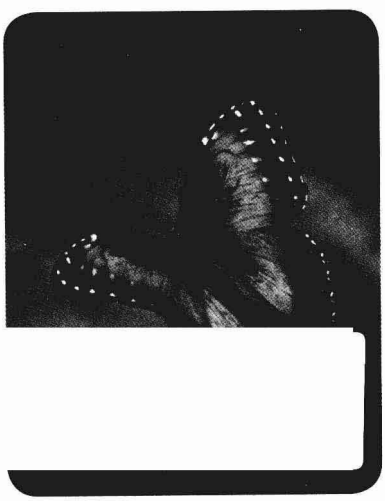
21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

编译原理——编译程序构造与实践教程

Principles of Compiling—A Course in Constructing and Practising of Compiler

张幸儿 戴新宇 编著



精品系列

人民邮电出版社

北京

图书在版编目（CIP）数据

编译原理 : 编译程序构造与实践教程 / 张幸儿, 戴新宇编著. -- 北京 : 人民邮电出版社, 2010.4
21世纪高等学校计算机规划教材
ISBN 978-7-115-21512-3

Ⅰ. ①编… Ⅱ. ①张… ②戴… 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(2009)第191479号

内 容 提 要

本书系统而简洁地介绍编译程序的构造原理，内容主要包括：概论、编译程序构造的基础知识、词法分析、语法分析、语义分析与目标代码生成、中间表示代码与代码优化、程序错误的检查与校正、目标代码的运行，以及虚拟机目标程序的解释程序的编制。各章开始于本章导读，各章末有本章小结、复习思考题以及习题。本书突出实践性，在编译程序构造的各个环节中，提供了具体可行的实现方法和技巧，供读者参考。

本书可作为计算机及相关专业的编译原理课程教材，也可作为计算机软件技术人员、研究生及广大计算机爱好者的参考用书。

21 世纪高等学校计算机规划教材

编译原理——编译程序构造与实践教程

◆ 编　　著　张幸儿　戴新宇
责任编辑　武恩玉

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：18.25
字数：477 千字　　2010 年 4 月第 1 版
印数：1－3 000 册　　2010 年 4 月北京第 1 次印刷

ISBN 978-7-115-21512-3

定价：31.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

出版者的话

计算机应用能力已经成为社会各行业最重要的工作要求之一，而计算机教材质量的好坏会直接影响人才素质的培养。目前，计算机教材出版市场百花争艳，品种急剧增多，要从林林总总的教材中挑选一本适合课程设置要求、满足教学实际需要的教材，难度越来越大。

人民邮电出版社作为一家以计算机、通信、电子信息类图书与教材出版为主的科技教育类出版社，在计算机教材领域已经出版了多套计算机系列教材。在各套系列教材中涌现出了一批被广大一线授课教师选用、深受广大师生好评的优秀教材。老师们希望我社能有更多的优秀教材集中地呈现在老师和读者面前，为此我社组织了这套“21世纪高等学校计算机规划教材-精品系列”。

“21世纪高等学校计算机规划教材-精品系列”具有下列特点。

（1）前期调研充分，适合实际教学需要。本套教材主要面向普通本科院校的学生编写，在内容深度、系统结构、案例选择、编写方法等方面进行了深入细致的调研，目的是在教材编写之前充分了解实际教学的需要。

（2）编写目标明确，读者对象针对性强。每一本教材在编写之前都明确了该教材的读者对象和适用范围，即明确面向的读者是计算机专业、非计算机理工类专业还是文科类专业的学生，尽量符合目前普通高等教学计算机课程的教学计划、教学大纲以及发展趋势。

（3）精选作者，保证质量。本套教材的作者，既有来自院校的一线授课老师，也有来自IT企业、科研机构等单位的资深技术人员。通过他们的合作使老师丰富的实际教学经验与技术人员丰富的实践工程经验相融合，为广大师生编写出适合目前教学实际需求、满足学校新时期人才培养模式的高质量教材。

（4）一纲多本，适应面宽。在本套教材中，我们根据目前教学的实际情况，做到“一纲多本”，即根据院校已学课程和后续课程的不同开设情况，为同一科目提供不同类型的教材。

（5）突出能力培养，适应人才市场要求。本套教材贴近市场对于计算机人才的能力要求，注重理论技术与实际应用的结合，注重实际操作和实践动手能力的培养，为学生快速适应企业实际需求做好准备。

（6）配套服务完善，共促提高。对于每一本教材，我们在教材出版的同时，都将提供完备的PPT课件，并根据需要提供书中的源程序代码、习题答案、教学大纲等内容，部分教材还将在作者的配合下，提供疑难解答、教学交流等服务。

在本套教材的策划组织过程中，我们获得了来自清华大学、北京大学、人民大学、浙江大学、南京大学、吉林大学、武汉大学、哈尔滨工业大学、东南大学、四川大学、上海交通大学、西安交通大学、电子科技大学、西安电子科技大学、北京邮电大学、北京林业大学等院校老师的大力支持和帮助，同时获得了来自信息产业部电信研究院、联想、华为、中兴、同方、爱立信、摩托罗拉等企业和科研单位的领导和技术人员的积极配合。在此，人民邮电出版社向他们表示衷心的感谢。

我们相信，“21世纪高等学校计算机规划教材-精品系列”一定能够为我国高等院校计算机课程教学做出应有的贡献。同时，对于工作欠缺和不妥之处，欢迎老师和读者提出宝贵的意见和建议。

前 言

编译原理是计算机专业的一门重要课程，本课程开设的目的是讲解编译程序的构造原理。由于编译程序是把高级程序设计语言源程序翻译成等价的低级语言目标程序，其处理对象是符号序列，因此可以说编译程序是符号处理的工具，只要是与符号处理相关的领域，都将可能需要应用编译原理中的基本原理。例如，编译原理中所讨论的正则表达式与状态转换图就应用在很多领域。编译原理中的代码优化，尽管是在源程序的内部中间表示一级上进行的，但优化的思想可以为软件开发人员，特别是计算机程序设计爱好者所借鉴。

编译原理课程的特点是理论性强，因为它与形式语言理论紧密相关，可以说是在形式语言理论基础上讨论编译原理，这使得编译原理成为计算机专业最为困难的课程之一。其实，实践性应是编译原理课程更为关注的问题，也就是说，掌握编译程序构造原理的目的是实践编译程序构造。本书着力降低理论难度，围绕编译程序的编制开展讨论，以加强读者的实践应用能力。

本书有如下一些特点。

（1）以 C 语言为背景。紧密结合 C 语言，尽可能处处以 C 语言相关内容为例子进行讨论，使得讲解更有针对性，读者也更易于理解和接受。

（2）突出实践性。对编译过程的每个方面进行论述时，都有计算机实现的讨论，例如文法的存储表示、推导和语法分析树的生成、词法分析、识别程序句型分析，以及翻译方案的实现，都有相当的篇幅。最后还讨论了符号模拟执行虚拟机目标代码的解释程序的编制，给出可运行的解释程序。

（3）明确 C 语言语法成分目标代码结构与其他语言的区别。例如函数调用中，实际参数的计算次序与赋值语句左部变量地址的计算次序等，全书有多个 C 语言程序作为例子，包括应用指针与结构，以及建立结点链的 C 程序编写等，使读者对 C 语言有更深入的认识。

（4）加强讲解编程知识。本书中，除了作为例子的一些经典算法，如排序程序与计算多项式的霍纳方案外，不乏编程的方法和技巧。例如，文法的存储表示、LR 分析表的计算机存储表示与特征变量的引进以及置初值等，以启发读者应用 C 语言编程的能力。

学习是吸收新知识的过程，更重要的是培养和提高分析问题和解决问题能力的过程。一个常见的思维方法是：以简单的实例，通过对它的观察分析，得到解决问题的思路。另一种惯用的思维方法是：从已给的输入（Input）和要得到的输出（Output）出发，设法进行处理（Process），以得到解决方案，这就是 IPO。读者可以体会本书中各种方法和技巧的应用。

总的来说，本书力图以简洁易懂的文字阐述主要的基本概念，用朴实的实例展示实用的方法，按直观的思维方法启发寻找问题的解答。

本书在教学内容安排上，不追求面面俱到，突出从词法分析、语法分析、语义

分析和目标代码生成这条主线，因此词法分析部分突出词法分析程序的实现，语法分析部分突出LR分析技术，而语义分析部分突出与LR分析技术结合的翻译方案。正则表达式、状态转换图与有穷状态自动机、算符优先分析技术等内容可根据读者的情况安排选用。作者建议，把更多的注意力放到计算机实践中去，以培养和提高程序编写能力和计算机实践能力。建议上机实践安排下列几个题目：文法的计算机存储表示、词法分析程序的实现、LR识别程序句型分析与赋值语句目标代码生成，还可以实现符号模拟执行虚拟机目标代码的解释程序。

由于时间仓促，书中难免有错误与不足之处，欢迎读者批评指正。作者邮箱：

zhangxr0@sina.com 或 zhang_xinger@hotmail.com

最后要感谢武恩玉编辑，是她的大力支持和为本书所做的大量工作，使作者能有本书以飨广大读者，也感谢我的家人，是他（她）们的大力支持，使本书能够顺利完成。

作者

于南京大学计算机技术与科学系

2009年6月25日

目 录

第1章 概论

本章导读

编译原理课程讨论什么？编译程序又是什么？本章将讨论编译程序概况与编译程序的构造，并简单说明编译程序的实现，使读者对本课程有一个初步而较为全面的认识，充分认识到编译程序与高级程序设计语言的紧密联系，从而领会编译程序的作用、功能和组成，并理解为什么有这样的功能和组成。

1.1 编译程序概况

编译原理课程讨论的是编译程序构造原理。如编译程序是什么？为什么需要编译程序？编译程序又有什么样的结构，应如何去构造？这些都是编译原理课程讨论的内容。

编译程序是一种系统软件，它的功能是把高级程序设计语言源程序翻译成等价的低级语言目标程序。编译程序通常由前端与后端组成，前端进行分析，即词法分析、语法分析与语义分析；后端进行综合，即对经分析生成的内部中间表示生成目标程序，或者对内部中间表示进行优化后再生成目标程序。为了理解编译程序为什么有这样的构造，需要先了解高级程序设计语言及程序的概念。

1.1.1 编译程序的引进

首先理解程序的概念是什么？怎样书写和执行程序？

日常生活中人们有着对程序的一般理解：一系列依次执行的命令。例如开大会，主持人宣布“大会开始、主席发言、×××代表发言、……”，这是大会的一般程序，它由人来执行。现在假定要从键盘上键入3个整数，求其平均值，分别用汉语和英语来表达如下。

汉语	英语
提示：“请键入a、b与c的值”。	Prompt: “Type the values of a, b and c, please”.
键入a、b与c的值。	Type the values of a, b and c.
求a、b与c的平均值ave。	Calculate the average ave of a, b and c.
打印输出平均值ave。	Print the value of ave.

若用C程序设计语言来表达这个过程，则可写出如下代码：

```
printf("Type the values of a, b and c, please:");
```

```
scanf("%d,%d,%d", &a, &b, &c);
ave=( a+b+c) / 3;
printf("ave=%f\n", ave);
```

显然上面前两种表达方式都是容易理解的，对于第 3 种表达方式，熟悉 C 语言的读者也能理解。这是因为所有这些表达方式在书写上都是正确的，即单词和符号都是正确的，顺序安排符合语法，含义也表达清晰。但这些仅是片段，不是完整的表述，尤其是上述第 3 种不是一个完整的程序。

一个 C 语言程序必须遵循 C 语言程序书写规定，按照特定的格式来书写。例如，下述程序才是可运行的，假定该程序保存在文件 ave3.c 中。

```
/* ave3.c */
main( )
{ int a,b,c; float ave;
  printf("Type values of a, b and c, please:");
  scanf("%d,%d,%d", &a, &b, &c);
  ave=(a+b+c)/3.0;
  printf("ave=%f\n", ave);
}
```

现在打开文件 ave3.c，能得到平均值 ave 吗？显然不行。为了得到预期的结果，通常做法如下：运行 Turbo C 2.0，在相应界面主菜单的 File 选项下打开文件 ave3.c，然后按 Ctrl+F9 组合键来运行该文件上的 C 语言程序 ave3.c。

这里的 Turbo C 2.0 就是编译程序，更确切地可以说是编译系统，它把编辑、编译、运行与调试集成于一体。当打开相应文件夹，查看与 ave3.c 相关的文件时，可以发现除了文件 ave3.c 外，还有文件 ave3.obj 与 ave3.exe 等。如果直接运行 ave3.exe，可达到与按 Ctrl+F9 组合键执行 ave3.c 相同的效果。这表明生成了与 ave3.c 等价的可执行程序 ave3.exe。每次执行 ave3.exe，为 a、b 与 c 键入不同的值，就可以得到不同的平均值，如图 1-1 所示。

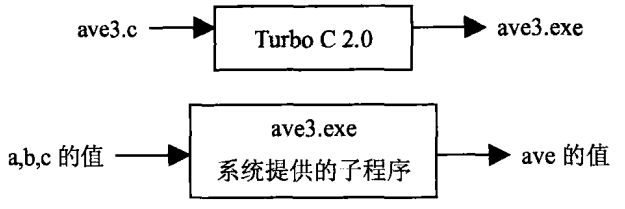


图 1-1

其中，系统提供的子程序在本书以后的部分将称为运行子程序。

通过执行 ave3.c 这样一个特例，可以考察一般情形。

任何语言都有一组记号，并由这些记号组成语法成分的规定，C 语言也不例外。一个 C 语言程序，必须按照 C 语言的拼写约定正确地写出一切符号，并由这些符号正确地组成各个语法成分，最终写出一个完整的程序。然而即使它书写正确，还是不能直接运行，因为它仅是一个符号序列，或者归根结底是一个由字母、数字、括号和标点符号等组成的字符序列。这是不能为计算机所直接接受、理解和执行的，计算机能直接接受、理解和执行的仅是二进位串形式的机器语言指令。

一般情况下，高级程序设计语言程序的执行有两种方式，即解释方式与翻译方式。

当计算机按解释方式执行高级程序设计语言程序时，有一个解释程序，由它对组成程序的各个语句，逐个语句地进行分析，模拟产生相应的运行效果，最终获得所期望的结果。解释方式的

优点是可以边运行边修改，一旦发现程序中有错误，可以立即改正，不足之处是功效较低。例如，当执行如下的循环语句：

```
s=0;
for( k=1; k<=n; k++)
  s=s+k;
```

每次重复执行循环体，便需要重新分析 for 语句，因此 for 语句总共要重复分析 n 次。

当计算机按翻译方式执行高级程序设计语言程序时，并不是直接执行高级程序设计语言程序，而是有一个翻译程序，它把高级程序设计语言程序翻译成等价的低级语言程序，然后执行所生成的低级语言程序，获得所期望的效果，此翻译程序称为**编译程序**。高级程序设计语言程序称为**源程序**，而等价的低级语言程序称为**目标程序**。相应的语言分别称为源语言和目标语言。把源程序翻译成为等价的目标程序的过程也称编译。编译方式执行高级程序设计语言程序的过程如图 1-2 所示。

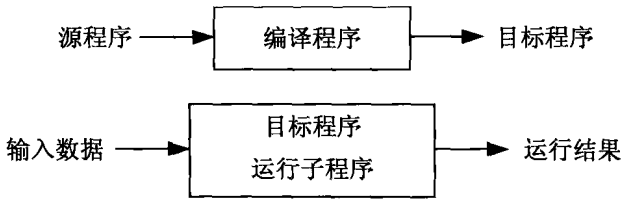


图 1-2

编译方式执行高级程序设计语言程序的优点很明显：功效高。源程序只需编译一次，目标程序可运行任意多次。当编译程序进行目标程序优化工作，改进目标程序质量时，更可大大提高功效；不足之处是修改源程序较为困难，必须对照目标程序中出现的错误，在源程序中找出相应的位置，从而才能进行修改。值得庆幸的是，当前的众多编译程序扩充为集成式的开发环境，其中提供符号调试功能，可以在源程序一级进行调试。当然，作为编译程序的开发人员，就不仅要实现编译程序本身，还得考虑符号调试程序功能的实现。

编译原理课程讨论的是编译程序的构造原理，重点讨论构造编译程序本身的实现技术和方法。希望读者在学习完本课程后，了解编译程序有怎样的结构，理解为什么有这样的结构，掌握编译实现的技术，了解在应用各种编译技术时的应用条件与可能存在的问题。

1.1.2　编译程序与高级程序设计语言的联系

由上述可见，编译程序是执行高级程序设计语言程序的一种手段。作为编译程序的输入，源程序是用某种高级程序设计语言编写的，必须遵循这种语言的书写规定。以 C 语言为例，一个 C 程序由一组函数定义组成，函数定义之外，还可能有一些全局量定义。为简化起见，仅讨论函数定义的情况。

假定有如下的 C 程序：

```
(1)        void main( )
(2)        {  int x, y, max;
(3)           printf("Input values of x and y:");
(4)           scanf("%d,%d", &x, &y);
(5)           if(x>y) max=x; else max=y;
(6)           printf("max=%d\n", max);
           }
```

读者一定熟悉，标有(1)的行是函数首部，其中指明函数名、函数值的类型与参数及其类型。函数体用一对大括号括住，一般包含说明部分与控制部分。显然(2)是说明部分，(3)到(6)是控制部

分。为了描述函数定义如何书写，可以用口语陈述，例如：

“函数定义由函数首部后跟以函数体组成；函数首部则首先是函数值类型（可能省略）后跟以函数名，再跟以用小括号对括住的参数表列（允许没有参数表列）；函数体由用大括号对括住的说明部分和控制部分组成，且说明部分在前，控制部分在后。说明部分与控制部分可能缺省其中一个，甚至两者都缺省。”

显然这样陈述十分累赘，可以改用下列图示的形式描述，如图 1-3 所示。

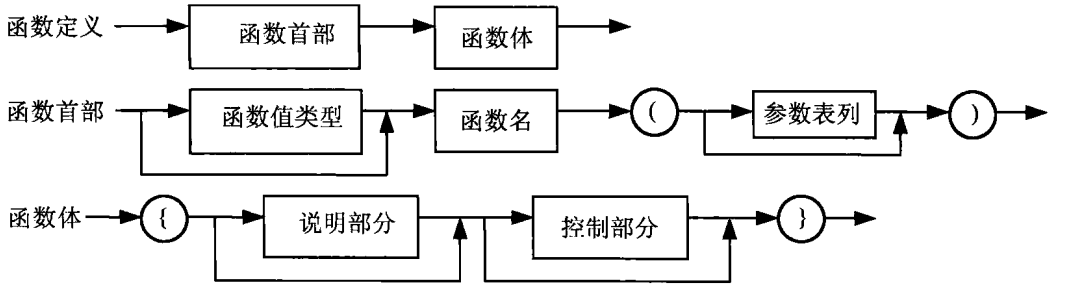


图 1-3

图 1-3 这样的图称为语法图，用图的形式规定了一个 C 语言函数定义的结构，也即书写规则。其特点是直观、形象，一眼便明了一个函数定义应该怎样书写。但其不足是篇幅太大，且是平面图形，难以由计算机处理，特别是不利于编译程序自动构造。

本书将采用称为 **BNF 表示法**的形式来描述书写规则。例如，函数定义可描述如下：

<函数定义>::=<函数首部> <函数体>

<函数首部>::=<函数值类型><函数名>(<参数表列>)

| <函数值类型><函数名>( )

| <函数名>(<参数表列>)

| <函数名>( )

<函数体>::={ <说明部分> <控制部分> } | { <说明部分> } | {<控制部分> }| { }

其中，记号“::=”解释为“定义为”，记号“|”解释为“或”。记号“<”与“>”括住的内容称为语法实体（为简化，可以不用“<”与“>”括住），它们实际上是对程序中某部分所取的语法名称，以便于进行讨论与分析，并不会出现在程序中。上述的描述可解释为：

“函数定义定义为：函数首部，后跟以函数体；函数首部定义为：函数值类型后跟以函数名，再跟以用小括号对括住的参数表列，或者定义为：函数值类型后跟以函数名，再跟以小括号对，或者定义为：…，或者定义为：…。”

显然这与前面的口语描述及语法图描述都是一致的，但它们能以一种确切的、无歧义的简洁方式描述完整的意思。这种写在一行上的记号更便于计算机处理。

概括起来，程序设计语言的书写规则可以用 3 种方式描述：

- 口语
- 语法图
- BNF 表示法

本书将主要采用类似 BNF 表示法的形式来描述语法。

当进一步分析程序的组成时，或者说，分析函数体的组成时，可以进一步引进语句、表达式和量等概念。一般地，一个 C 语言程序可以有如下层次。

C 程序—函数定义—说明部分、控制部分—语句—表达式—量—符号—基本符号

基本符号的例子：void　m　a　i　n　(　)　{　int　x　;　if　else　=　}

符号的例子：void　main　(　)　{　int　x　max　if　else　=　}

量的例子：x　y　max

表达式的例子：x　max=x

语句的例子：max=x;　scanf("%d, %d", &x, &y);

说明部分的例子：int x, y, max;

控制部分的例子：if(x>y) max=x; else max=y; printf("max=%d\n", max);

其中，基本符号是组成程序的最基本成分。例如，括号(与)是基本符号，关键字 if 与 else 也是基本符号，这些符号都作为一个整体，具有特定的含义，是不可分割的。但对于 C 语言或其他语言，字母与数字也是基本符号，这类符号并不一定具有含义，仅作为符号或符号的组成部分时，才有含义，即表示整个符号的含义。例如，字母 x，作为基本符号，没有含义，若作为标识符，就有了含义。组成 max 的 m、a 与 x 情况一样，单独作为基本符号时并无含义，整个符号 max 才有含义，即是标识符。

概括起来，基本符号可以有含义，也可以没有含义。C 语言的基本符号包括：字母、数字与界限符，界限符则包括关键字、括号、运算符与其他一些专用符号。由这些基本符号可以组成 C 语言标识符、常量、标号与界限符等。

一个程序可以看成是由基本符号组成的，但归根结底是由字符组成的。例如，关键字 if 由字母 i 与字母 f 组成。这就决定了一个编译程序首先必须识别各个基本符号或符号，然后识别出由各个符号组成的语法成分，最后识别出整个程序，进一步依据各个语法成分的含义，生成目标程序。

一个高级程序设计语言通常涉及 4 个方面，即语法、语义、语用与语境。

① 语法指明程序设计语言程序的书写规则。由基本符号组成符号的拼写规则称为词法规则，由符号组成语法成分的规则称为语法规则。词法规则与语法规则全体统称为语法。如前所述，语法可用口语、语法图和 BNF 表示法 3 种形式描述。

② 语义指明按语法规则构成的各个语法成分的含义，尤其是程序的含义。程序设计语言的语义决定了语法成分的含义，即程序执行的效果。语义通常用口语方式来描述。例如，用 BNF 表示法描述的 C 语言赋值语句：

<赋值语句>::=<左部变量>=<表达式>;

其语义用口语方式描述如下。

“赋值语句的语义是：首先计算右部表达式的值，然后把该值赋给左部变量。必要时先对右部表达式的值进行类型转换，转换到左部变量的类型。”

口语定义方式的不足是明显的：不严格、不精确，并可能存在歧义，不利于保证或验证程序的正确性，特别是自动生成程序。计算机科学中的一个分支：形式语义学，研究的是如何形式地定义一个程序设计语言的语义。但此已超出本书的讨论范围，且这远未实用化，难以高效实现编译。本书不讨论这一范畴的内容，只是采用折衷的语法制导翻译技术，即引进所谓的属性文法来描述语义。

③ 语用一般用来表示语言中的符号及其使用者之间的关系。例如，程序中的注解指明某变量的物理意义与用途等，这就是语用的体现。

④ 语境指明理解和实现程序设计语言的环境，包括编译环境与实现环境。不同的语境明显地影响着语言的实现，例如，C 语言中，整型量通常占 2 个存储字节，这意味着一个整型量最大值

为 $2^{16}-1=32\,767$，当值 32 767 再加 1 时将得到结果－32 768，这显然是错误的。但如果让一个整型量占 4 个字节，情况就完全不一样，32 767+1，这时不会发生错误。在应用某程序设计语言编写程序时，注意语境问题也是必要的。

1.1.3 编译原理课程的教学内容、教学目标和要求

从前面的讨论可知，编译原理课程讨论的是编译程序的构造原理，而编译程序是为执行高级程序设计语言程序而开发的系统软件，只有深刻理解编译程序与高级程序设计语言之间的紧密联系，才能真正理解编译程序的作用与结构。编译原理课程将讨论下列 3 方面内容：

- 高级程序设计语言的相关知识；
- 编译程序实现的基础知识；
- 编译程序构造原理。

希望读者在学习完本课程后，了解编译程序的作用是什么，功能是什么；理解编译程序的结构如何，为什么有这样的结构；掌握构造编译程序的原理、技术和方法，了解应用时的应用条件，可能会发生什么问题和如何解决这些问题等。

本书结合 C 语言进行讨论，强调了 C 语言有别于其他语言的特殊之处，期望读者通过本书的学习，加深对 C 语言的理解。

本书不仅要求读者掌握相关的概念，更希望读者注意实践性，期望读者应用书中讨论的相关技术与方法，在计算机上实践从实践中加深对概念的理解，尤其是通过实践，培养和提高自己研制程序的能力与计算机实践能力。

1.2 编译程序的构造

1.2.1 编译程序的功能

编译程序是一款系统软件，它实现把高级程序设计语言源程序翻译成等价的低级语言目标程序。等价的含义就是：当源程序正确时，运行目标程序，可达到相同的效果；如果源程序中存在错误，则目标程序有相应的反映。

由于源程序归根结底是一个符号序列，往往把编译程序看作是符号处理的工具。具体说，读者需要了解编译程序需要完成哪些工作？

编译程序与高级程序设计语言紧密相关。从前面的讨论不难了解编译程序必须依次进行的工作如下。

① **词法分析**。编译程序首先对源程序从左到右逐个字符地进行扫描（读入），识别开各个具有独立意义的最小语法单位（即符号或单词，如标识符、常量和关键字等），并把它们转换成称为属性字的内部中间表示，同时进行一些力所能及的其他工作，例如删除注释与非 C 语言成分的字符（如换行字符与空格字符等）。请注意，对于 C 语言来说，由于包含预处理功能，即文件包含（include）命令、宏定义（define）命令，以及条件编译命令，在进行了预处理之后才正式进行词法分析。预处理工作无实质性的困难，并且不是编译程序的“份内”工作，在此不加讨论。

完成词法分析的部分称为**词法分析程序**，也称为**扫描程序**。概括起来，词法分析程序的功能是：扫描（读入）源程序，识别开各个符号，并把它们转换为相应的内部中间表示（属性字），以

及进行删除注释与非语言成分的字符等力所能及的工作。必要时可能进行类似于C语言预处理的工作。

② **语法分析**。词法分析时识别开各个符号之后，由语法分析部分根据程序设计语言的语法规则，识别出各个语法成分，最终识别出完整的程序。在识别各类语法成分的同时，也就检查了语法的正确性。当识别出是语法上正确的程序时，生成相应的内部中间表示（通常是语法分析树或其他内部中间表示），如果存在错误，则给出相应的报错信息。

完成语法分析的部分称为**语法分析程序**，或称为**识别程序**。概括起来，语法分析程序的功能是识别出各个语法成分，生成相应的内部中间表示，同时进行语法正确性的检查。

③ **语义分析**。编译程序继语法分析之后进行语义分析，即基于语法分析时输出的内部中间表示，依据各个语法成分的含义进行语义分析。由于一个程序通常由数据结构和控制结构两部分组成，必然对这两部分进行语义分析。对于数据结构，语义分析部分进行的语义分析工作是确定类型和类型检查，确切地说，检查标识符是否有定义，确定标识符所对应数据对象的数据类型等属性，检查运算的合法性及运算分量数据类型的一致性；对于控制结构，根据程序设计语言所规定的语义，对它们进行相应的语义处理，这时可以生成相应的目标代码。例如，对于一个加法运算，当检查了两个运算分量都有定义，它们都能进行加法运算（运算是合法的），且两个运算分量的类型一致（相容）时，可以生成进行加法的目标代码。不言而喻，执行语义分析的同时，还进行一些语义检查，当然这只是静态语义检查，即在编译时刻所能进行的语义检查，例如，检查是否从循环外通过控制转移语句把控制转入循环体。在运行时刻才能进行的语义检查称为动态语义检查，如检查数组元素下标是否越界，以及指针变量是否有初值等，自然不在语义分析时刻进行。为了改进目标程序质量，语义分析时可能不生成目标代码，而是生成另外一种内部中间表示，或称中间表示代码。代码优化阶段就是基于这种中间表示代码进行优化，然后再从优化了的中间表示代码生成目标代码。语义分析工作通常由语义子程序完成。

完成语义分析的部分称为**语义分析程序**。概括起来，语义分析程序的功能是确定类型、类型检查、识别含义与相应语义处理，以及其他一些静态语义检查等。

④ **代码优化**。代码优化指的是为改进目标程序质量而在编译时刻进行的各项优化工作。代码优化通常基于语义分析部分生成的中间表示代码进行，把它变换成功能相同、但功效更高的优化了的中间表示代码。本书的中间表示代码主要是四元式序列。代码优化有与机器无关的优化和与机器有关的优化之分。本书重点讨论与机器无关的优化。除了对中间表示代码进行优化外，也可以对目标代码进行优化，这种优化通常称为窥孔优化。窥孔优化是机器语言级上仅在一个很小的范围内进行的、不太复杂因而代价不是很高的一类优化。

当前的很多高级程序设计语言的编译程序都进行代码优化，C语言编译程序是突出的优化典型。

完成代码优化的部分称为**代码优化程序**。概括起来，代码优化程序的功能是：在编译时刻基于中间表示代码进行目标程序质量的改进。

⑤ **目标程序生成**。如前所述，语义分析时可以直接生成目标程序，这里的目标程序生成指的是基于优化了的中间表示代码（也即四元式序列）生成目标程序。目标程序的生成与运行目标程序的计算机密切相关。为了学习的目的，本书避免把大量的时间花在对具体计算机细节的了解上，将基于一种所谓的虚拟机来讨论目标程序的生成。

1.2.2 编译程序的组成

由上述可见，编译程序在对一个源程序进行翻译时，将经历词法分析、语法分析、语义

分析、代码优化和目标程序生成等阶段，因此通常把编译程序看作由两个部分组成，即前端与后端。前端进行分析，即词法分析、语法分析与语义分析；后端进行综合，即代码优化和目标程序生成。前端基本上与计算机无关，而后端，特别是目标程序生成，一般与运行目标程序的计算机密切相关。

对应于编译程序要完成的各项功能，分别有词法分析程序（扫描程序）、语法分析程序（识别程序）、语义分析程序、代码优化程序与目标程序生成程序，把这些程序有机地结合起来，从而实现对高级程序设计语言源程序的编译。显然，编译程序实际上是一个结构庞杂的程序系统，因此往往称为编译系统。但是在实际实现时，并非一定按上述 5 个方面依次实现编译。编译系统的开发人员往往根据实际情况，例如编译系统性能指标设计、参与人员的配备与交付日期的缓急等，把整个编译程序分成若干阶段来开发。每一阶段都以上一阶段的输出作为本阶段的输入进行相应的处理（例外是，第一阶段时是源程序）。每一阶段生成等价的内部中间表示作为输出，这输出要有利于下一阶段的处理。

每个阶段从头到尾读入整个输入并进行处理的过程称为**遍**（或趟）。一个编译程序由几遍完成编译便称为几遍编译程序。如果对源程序从头到尾扫描一次便完成词法分析、语法分析、语义分析、代码优化与目标程序生成等全部工作，就称这样的编译程序为一遍编译程序。一遍编译程序对于功能不太强的小语言还可以，但一般的情况是对各个基本工作进行适当的组合，分成若干遍。例如，词法分析作为第一遍，语法分析与语义分析作为第二遍，代码优化与目标程序生成作为第三遍，这样的编译程序便是三遍编译程序。三遍编译程序的工作示意图，如图 1-4 所示。

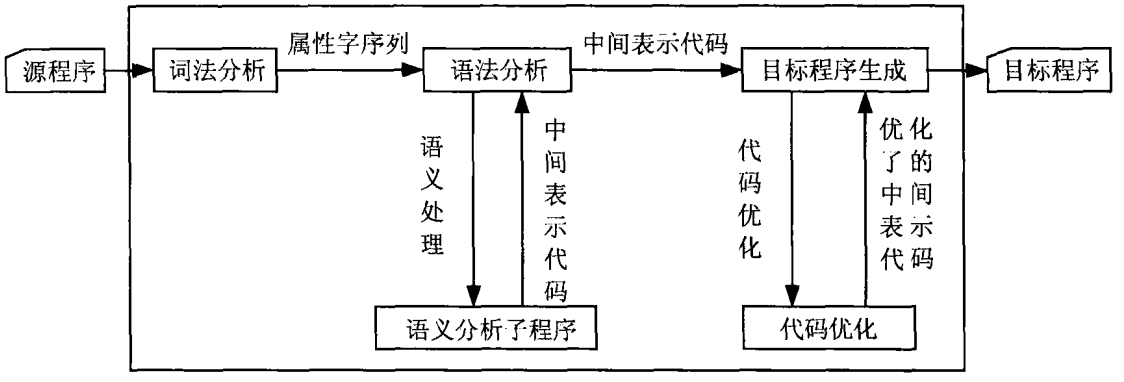


图 1-4

这里要提醒的是：为了目标程序能运行，编译系统开发人员必须研制相应的一些系统配套程序，即运行子程序。只有在运行子程序的支持下，目标程序才能运行，并获得期望的效果。关于运行子程序将在第 9 章 9.4 节中讨论。

1.2.3 编译程序的种类

不同的程序设计语言有各自的编译程序，即使是同一个程序设计语言，也可以按照不同的用途、设计的性能指标，以及开发人员配备情况，研制不同的编译程序。一般，按照用途与侧重面，大致有如下几类编译程序。

① 诊断型编译程序。这类编译程序专门设计来帮助开发与调试程序。当进行编译时，诊断型编译程序不仅可以查出、甚至可以自动校正一些小错误（例如，左右括号不配对、缺少分号等小