21世纪高等院校教材

计算机编译原理

——编译程序构造实践

张幸儿 编著

本书是《计算机编译原理》（第二版）的配套教材

科学出版社
www.sciencep.com

21世纪高等院校教材

计算机编译原理——编译程序构造实践

张幸儿　编著

科学出版社
北京

内 容 简 介

本书是编译原理课程的配套教材，第一篇概论包括编译程序概述与实践指南，第二篇实践篇包括文法及相关概念、词法分析、语法分析、语义分析和目标代码生成、目标代码优化。本书尝试以实习题的形式探讨编译程序构造全过程的实现，使读者对于从源程序字符串到等价的目标代码的翻译全过程有深刻的理解。书中还介绍了程序（软件）的一般研制过程，特别是C型语言程序界面的设计与实现。

本书可作为计算机及相关专业编译原理课程的富有启发性的配套实践教材，同时也可作为计算机软件工作者及广大计算机爱好者学习的参考用书。

图书在版编目(CIP)数据

计算机编译原理：编译程序构造实践/张幸儿编著. —北京：科学出版社，2005

（21世纪高等院校教材）

ISBN 7-03-015322-7

Ⅰ. 计… Ⅱ. 张… 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(2005)第028479号

责任编辑：资丽芳 姚庆爽/责任校对：包志虹

责任印制：钱玉芬/封面设计：陈 敬

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

双青印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2005年7月第 一 版 开本：B5(720×1000)

2005年7月第一次印刷 印张：20 3/4

印数：1—3 000 字数：401 000

定价：30.00元

（如有印装质量问题，我社负责调换〈环伟〉）

前　言

编译原理讨论编译程序构造的基本原理和技术，它以形式语言理论的基本概念为基础，讨论编译全过程，剖析编译程序的构造，既有深沉的理论性，又有浓厚的实践性。编译原理课程与计算机科学中的众多学科相关联，其基本原理与技术对计算机应用领域的许多方面有着相当大的影响，对程序研制也有一定的指导意义。

为什么编译原理课程是理论性很强、实践性也很强的课程？这是因为应把编译原理的基本概念、技术与方法应用于编译程序构造的实践。然而目前不足的是，还没有一本编译原理教材明确地阐明如何把编译程序构造的各阶段有机地联系起来，实现从输入源程序到输出等价的目标代码的全过程。现有的编译原理教材基本上可以说还是仅仅局限于原理性、概念性的理论介绍的。

事实上，编译原理课程中讨论的实现技术与实际实现有着一定的距离，仅考察下列几点便已清楚：

- 文法的书写形式与实际的机内存放形式；
- 消去左递归文法等价变换的算法与实际实现；
- 非确定有穷状态自动机的确定化与实际实现；
- 预测识别算法与语法分析树的生成；
- LR(1)识别算法与实际实现。

尤其是语义分析时的翻译方案(语法制导定义)中语义动作(语义规则)的书写形式与实际实现。

鉴于语义动作等在翻译方案中仅是一个字符序列，要执行它们，实现语义分析，就必须有一个符号处理程序来分析处理这些语义动作(字符序列)，产生相应的目标代码从而实现这些语义动作。这要与语法规则相结合，其复杂程度相当于构造一个高级程序设计语言编译程序。

本书尝试以实习题的形式探讨编译程序构造全过程的实现，使读者对于从源程序字符串到等价的目标代码的翻译全过程有深刻的理解。

概括起来，本书旨在：

- 简略回顾与编译程序构造相关的概念，使读者加深对编译原理课程要点的理解与掌握；
- 简略介绍程序(软件)开发方法，尤其是应用 C、C++与 VC++系统平台来设计与实现应用程序界面，使读者了解与掌握这些系统平台的初步使用；
- 结合编译程序的构造，把各个阶段有机地联系起来，使读者深切地体会从源

程序到目标代码的实际实现全过程。为编译原理课程提供上机实验题的富有启发性的参考资料。

笔者殷切期望，通过对本书的阅读，读者将能：

• 了解如何分析一个问题，明确自己的思路，从而有助于解决问题；

• 掌握各种实际实现方法和技巧，从而有助于读者对其他程序（软件）的研制。

若能如此，对笔者将是莫大的鼓舞。

本书的内容主要源自笔者历年来的教学和科研，其中包括了笔者部分本科生和研究生的实践成果。感谢曾奇、严萍与李进为本书积累了部分资料。感谢张福炎教授对本书成稿的大力支持。感谢李意勉女士，在她的支持和大力帮助下，本书才能顺利完成。最后，感谢资丽芳编辑，是她的关心、大力支持和辛勤工作，才使笔者得以实现奉献此书给广大读者的愿望。

本书是在编译原理课程领域的一个初步尝试，限于笔者水平限制，错误与不妥之处在所难免，综观全书，也不乏需要改进与完善之处。欢迎读者批评指正。

联系地址：南京大学计算机科学与技术系（210093）

E-mail 地址：zhangxr0@sina.com 或 zhang_xinger@hotmail.com

作 者

2005 年 2 月于南京大学计算机科学与技术系

目　录

第一篇　概　　论

第 1 章　编译程序概述

1.1　编译程序及其构造

1. 编译程序的引进

一个高级程序设计语言归根结底是一个基本符号序列，或进一步说是一个字符序列。例如下列 C 语言程序：

```
main( )
{  float x,y;
   printf("Input values of x and y.\n");
   scanf("%f, %f", &x, &y);
   if(x>y)
     printf("max(x,y)=%f",x);
   else
     printf("max(x,y)=%f",y);
}
```

当我们上机运行时，计算机并不能直接接受、理解与运行该程序。计算机所能直接接受、理解与运行的是机器语言的二进位位串，而现在却是字母 m、a、i、n 与括号(、)等字符。为此需要一个软件处理这个字符序列，识别出这是 main，那是 float 等，进一步把它们翻译成可由计算机执行的等价的目标代码，达到运行且获得所期望效果的目的。这种把高级程序设计语言程序翻译成等价的低级语言程序的软件就是编译程序。高级程序设计语言程序称源程序，所翻译成的等价的低级程序设计语言程序称为目标程序或目标代码。

编译程序是一种符号处理工具，它把高级程序设计语言源程序翻译成等价的低级语言目标代码。此处低级语言可以是汇编语言，也可以是机器语言。等价指的是源程序和目标程序两者效果必须完全一样，也就是说，当源程序正确时运行得到的效果，运行目标代码时同样能得到，若源程序有错误，目标代码也将有相应的反映。

以翻译方式运行的任何一种高级程序设计语言程序都必须有编译程序。这种编译程序在当前都被开发成了集编辑、编译、调试与运行于一体的程序设计语言支

持环境，如 Turbo PASCAL、Turbo C、Borland C++Builder 与 Visual C++等。可视化版本编译程序的引进更为用户设计应用程序界面带来了极大的方便。由于编译程序一般较为庞大，结构复杂，往往称为编译系统。这些编译系统软件往往由著名的公司开发，具有人机接口良好、使用方便、目标代码质量高等优点。

这些编译系统软件垄断了绝大部分的软件市场，国内自行开发编译系统已近乎没有，一些学生可能会说，我们毕业后不会去开发编译程序，没有必要学习编译原理课程。殊不知，编译程序作为符号处理的工具，其基本原理和技术有着典型性和广泛性，在软件工程、逆向工程、软件再工程、语言转换及其他领域都有着广泛的应用。例如，语法制导的程序编辑软件、程序结构分析软件、程序调试与测试软件、反汇编软件、符号执行软件与超文本语言的实现等都需要应用编译程序构造的基本原理与技术；编译原理对软件的开发也有一定的启发与指导作用。编译程序作为符号处理的工具，只要涉及符号处理，就需要采用编译程序实现的基本原理与技术。

2. 编译程序的功能

基于对编译程序作用的认识，总的来说，编译程序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代码。

鉴于目标代码一般总是机器相关的，也就是说为某种型号计算机产生机器语言目标代码，必须对此种型号计算机的指令系统（包括指令格式、不同操作码和寻址方式等）有详细的了解。这样需要花费大量的时间，而且仅限于一种型号的计算机，有相当的局限性。因此在学校教学中往往采用臆设的虚拟机，即按照教学的要求设计指令格式、操作码和寻址方式。这样就无需把大量时间花在对具体计算机细节的了解，而且可以根据需要随时扩充。

编译程序的输入是某种高级程序设计语言源程序字符序列（字符串），以 C 语言为背景时输入就是 C 程序字符串，输出是等价的虚拟机目标代码。

一个编译程序输入高级程序设计语言源程序字符串，要识别出一个个具有独立含义的最小语法单位——符号（或称单词），如识别出 main、float 与 if 等，并把它们加工处理成内部中间表示——属性字，这就是词法分析。词法分析时还进行一些力所能及的工作，如删除注解、空格与不可见的控制字符等。对识别出的连续若干个符号，编译程序要识别出各个语法成分，如 if 语句、输入输出语句与 main 函数定义，直到识别出程序，这时属性字序列被加工处理成了一种内部中间表示，即推导或语法分析树等，这就是语法分析。

对于语法分析树或推导这些内部中间表示，识别所相应的各个语法成分的含义，为此进行确定类型、类型检查，以及识别语义并进行相应的处理，例如，生成一些表格和生成目标代码或者内部中间表示代码，这就是语义分析。语义分析时还进行若干静态语义检查。

引进内部中间表示代码的目的是为了改进目标代码的质量。例如，在编译时

刻计算好 2 * 3.1416 的值，运行时刻直接使用该乘积值 6.2832，或者把循环时每次重复都不改变值的表达式外提以大幅度减少重复计算量，等等，可以提高目标代码运行效率，这种在编译时刻为改进目标代码的质量而进行的工作就是目标代码优化。

概括起来，一个编译程序一般要经历下列五个阶段，即词法分析、语法分析、语义分析、目标代码优化和目标代码生成。

词法分析：扫描（读入）源程序字符序列，识别出单词（符号），把它们加工处理成内部中间表示形式（属性字），并进行力所能及的一些工作（如删除注解与空格等无效字符，以及预处理工作）。

语法分析：对符号序列识别出各个语法成分，把它们加工处理成等价的内部中间表示（语法分析树或推导）。

语义分析：确定类型、类型检查、识别含义并做相应处理（生成目标代码或内部中间表示代码），还做一些静态语义检查。当生成内部中间表示时，这些中间表示可以是抽象语法树、逆波兰表示、四元式序列或三元式序列。

目标代码优化：在内部中间表示（主要是四元式序列或三元式序列）的基础上进行基本块的优化和与循环有关的优化，直至全局优化。代码优化后生成优化了的内部中间表示代码。

目标代码生成：当语义分析所生成的是内部中间表示代码时，就需要对这些内部中间表示代码来生成目标代码。为了教学的目的，此目标代码是虚拟机目标代码。不言而喻，此虚拟机目标代码是不能直接运行的。为了能运行，需要设计并实现一个解释程序来模拟执行虚拟机目标代码。编译原理通常不讨论此问题。

上述这五个阶段有机地结合起来就可实现相应的编译程序原型。

在实际的编译系统中，往往有不同的结构。例如，对源程序字符序列从头到尾扫描一次就完成词法分析、语法分析、语义分析与目标代码生成；也可以词法分析时从头到尾扫描一次源程序字符序列，生成相应的属性字序列，再以属性字序列作为语法分析与语义分析的输入，同时完成语法分析与语义分析，生成相应的内部中间表示代码，最终从头到尾地扫描内部中间表示代码而生成目标代码。一个编译系统，从头到尾扫描源程序或内部中间表示一次，就称为一趟（一遍），总共几趟（遍）就称为几趟（遍）的编译系统。在我们的讨论中，每个阶段看作一趟（遍）。

这里我们可以看到，在进行编译程序实践时，除了确定语言背景，即确定一个要进行编译程序实践的程序设计语言或其子集外，重要的是设计各个中间表示，即属性字的机内表示、语法分析树或推导的机内表示、内部中间表示代码（四元式等）的机内表示等。最后要设计虚拟机指令系统。

要说明的一点是：在编译原理课程中，编译期间的每一个阶段，除了词法分析以源程序字符序列作为输入外，其余各阶段都是以符号序列作为输入，并不是实际

的内部中间表示。例如，语法分析以符号序列作为输入，并不以属性字序列作为输入，又如，语义分析阶段也不是以语法分析树作为输入，而是仍然以符号串作为输入。这并不失一般性，事实上，我们可以在取符号函数 GetSymbol 中做相应的变动即可。下面以语法分析为例说明之。

假定属性字之数据结构定义如下：

```
#define MaxNumber 1000
struct AttributeWordType    /* 属性字之结构 */
{   char SymbolName[10];    /* 符号名，实现中可以用符号在终结符号集中
                               的序号代表 */
    int kind; /* 符号种类 */
    int type; /* 符号类型 */
    ⋮
} AttributeWord[MaxNumber];
```

为简单起见，用数组存放属性字序列，且假定符号数最多为 MaxNumber 个。假定 i 是当前符号计数器，只需在 GetSymbol 函数中回送 AttributeWord[i]. SymbolName，使用函数调用语句：

```
strcpy(symbol, GetSymbol( ));
```

在变量 symbol 中便置好当前符号。又如，对于语法分析树的数据结构，可用如下定义：

```
#define MaxNumber 1000
struct NodeType
{ char SymbolName[10];          /* 符号名，实现中可以用符号在相应符号
                                   集合中的序号代表 */
  int NumberOfNode;             /* 结点序号 */
  int NumberOfFatherNode;       /* 父结点序号 */
  int NumberOfBrotherNode;      /* 左兄结点序号 */
  int NumberOfSonNode;          /* 右子结点序号 */
} 语法分析树[MaxNumber];
```

同样地，为简单起见，用数组存放语法分析树结点，结点数最多为 MaxNumber，只需同样地在 GetSymbol 函数中回送语法分析树[i]. SymbolName，使用函数调用语句 strcpy(symbol,GetSymbol( ))同样地可以在变量 symbol 中置好当前符号。

中间表示(数据结构)随实现而不同，省略了对具体表示法的描述，使得大大简化了讨论，但又不失一般性。因此本书讨论的编译程序各阶段之输入在语法分析与语义分析阶段都以符号序列作为输入，而不是具体的属性字序列或语法分析树。

3. 编译程序的构造

一般地，编译程序由两大部分组成，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，即目标代码优化与目标代码生成。

前端往往涉及一个程序设计语言的定义，而与具体目标机无关。后端在生成目标代码时则一般与目标机有关，必须考虑各个具体细节，如指令系统(包括指令格式、不同的操作码和寻址方式)、寄存器个数与使用约定以及特殊指令的使用等。当目标机是虚拟机时，情况要简单得多。

本书讨论的编译程序可以看作五趟的编译程序，即词法分析、语法分析、语义分析、目标代码优化和目标代码生成各看成一趟。这样的好处是结构清晰、重点突出，有利于理解与掌握编译程序构造原理及其实现。

1.2 编译程序构造实践之必要性

编译原理以形式语言理论相关概念为基础，尤其是实现语义分析的语法制导的翻译，以基于形式定义的属性文法为基础，其本身决定了有较强的理论性，但所讨论的编译程序作为高级程序设计语言支持软件，作为符号处理的工具，它又有很强的实践性。要学好编译原理，必须兼顾理论性与实践性两方面。

• 通过实践，加深对概念的理解。

这里以一个简单的例子说明之。

例如，设有文法 G[Z]:Z::=Zbc|AE　A::=da，要求写出其非终结符号集合 V_N。少数学生可能会写出 V_N={ Z,A,E}与 V_T={ a, b, c,d}，因为在他(她)们心目中，大写字母代表非终结符号，小写字母代表终结符号。但若正确地实现了自动构造 V_N 与 V_T 的程序，就会发现对于上述文法应是 V_N={ Z,A}，V_T={ a, b, c,d, E}。从而形成一个深刻的概念：只有出现在重写规则左部的才是非终结符号，与大小写无关。

另外一个例子是有穷状态自动机。随着有穷状态自动机构造程序的实现，将深刻体会到 DFA 与 NFA 之间的两个根本区别，即开始状态的个数的差异与状态映象之间的差异。

其他例子是在采用 SLR(1)方法构造 SLR(1)分析表时后继项集的计算，容易对不同项集中相同的终结符号 T 之 T-后继取相同的后继项集名。通过 SLR(1)分析表自动生成程序的设计与实现，就会体会到必须不仅项的个数相同，而且所包含的项全部相同时，两个项集才能有相同的项集名。

事实上，对编译原理中某个方面编写实现程序的过程本身就是不断认真学习概念和消化理解概念的过程。

通过编译程序构造实践，对相关部分的概念之理解和掌握将提升到一个新的

高度。

• 通过实践，深刻理解概念与具体实现之间存在的差异。

在编译原理课程上讲解编译程序的构造原理，应该说仅是概念上的，譬如说，应用 LR(1)分析技术进行语法分析时，分析表是关键，其中 ACTION 部分的元素确定分析动作。移入是 Si，归约是 rj，接受是 acc，而出错是空白元素。但具体实现时原封不动便难以实现。为方便处理，用一个整数值表示：大于 0 为移入，小于 0 为归约，出错为 0，而接受则可用某个特殊值，如－1111 或 9999 等。诸如此类，通过实践，摸索出实际实现时可以采用的合适方法与技巧，为今后程序研制积累经验。

• 培养与提高上机动手与开发软件能力。

学习的一个重要方面是理论与实践相结合。学生做习题是实践的一个手段，通过做习题加深对概念的理解。仅仅做习题的一个缺陷是并不能提高学生上机动手与开发软件的能力。

在编译原理课程中安排学生一定的上机时间量，完成若干题目的上机实习，无疑对学生提高软件开发能力有显著的效果。学生将了解开发这类软件的全过程；经历这一全过程，从中积累知识和经验，了解在开发一个软件时所应经历的步骤，在每一步应考虑哪些问题，提高编程能力，尤其是提高上机动态调试的能力。这对于计算机专业学生来说，对其进一步的发展有莫大的帮助。

本书期望在如何开发软件上对读者有启发提示与指导作用。通过阅读本书，结合上机实践，能大幅度提高软件开发与上机动手能力。

1.3 编译程序实现要点

对于一个完整而独立的编译程序的实现来说，其基本步骤大致如下：

步骤 1 确定背景程序设计语言。这可以是某个新设计的高级程序设计语言，也可以是现有的某个程序设计语言的扩充或子集。

步骤 2 确定编译程序设计指标与设计要求，包括确定目标机目标语言。

步骤 3 确定总体结构，包括分多少趟，各趟的功能与各趟之间的联系，并设计各个中间表示语言。

步骤 4 确定各趟的设计方案，并实现之。

1. 确定背景程序设计语言

编译程序的工作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代码，因此必须确定是哪一种高级程序设计语言(源语言 SL)。依据该程序设计语言的定义(词法、语法与语义)来实现编译程序。

一般地说，此语言应是结构化程度高的、定义确切且结构严谨的，便于用形式

语言来定义。当然这不能脱离开发此编译程序的目的。对于教学目的，鉴于大多数高校学生学习的是C型语言，可以选择C语言。由于C语言成分之定义过于灵活，要讨论整个C语言的实现将过于庞杂，难以达到预期的教学效果。合适的是，设计C语言的一个小子集，把涉及编译原理基本内容的语法成分保留下来，以严谨又简洁的形式定义该子集，为其实现编译程序可达到事半功倍的效果。

2. 确定编译程序设计目标与设计要求

设计目标与设计要求涉及待开发编译程序应达到的性能指标与要求，它们可以包括：

- 目标机是哪一型号计算机？或者目标机是虚拟机？目标代码是相对地址还是绝对地址？
- 单处理器还是多处理器？（我们可不考虑多处理器的情况。）
- 寄存器最多可多少个？（我们可假定任意多个。）如何管理寄存器？
- 编译速度要求快还是不做要求？
- 目标代码优化程度要求多高？实现哪些种类的优化？
- 运行时刻存储管理采用静态方式还是兼有静态与动态两种方式？无用存储区域如何回收？
- 表达式的运算次序又如何确定？

如此等等，不一而足。

总之，在实现编译程序之前需要对各个方面做好仔细的考虑。

设计目标和设计要求的确定决定了编译程序的进一步设计与实现。

3. 进行总体设计，确定总体结构

首先明确编译程序分成几趟，每一趟的功能是什么，以及各趟之间的联系。这时重要的是各趟的输入输出，应该确定各个中间表示 L_1、L_2、…、L_{n-1}，如词法分析阶段的输出属性字，语法分析阶段的输出语法分析树，语义分析阶段的输出四元式序列，等等。

在这时除了明确各趟的功能及相互的联系外，必须明确各趟的输入与输出。如明确词法分析输出的属性字序列具有怎样的结构，明确语法分析采用语法分析树还是其他作为输出，明确中间代码是利用四元式序列还是三元式序列，对于目标代码生成则明确确定语法成分与目标代码的对应关系，等等。每一趟的输出作为下一趟的输入，必须有利于下一趟的工作。

编译期间需使用一些表格，如字符表、标识符表或符号表、常量表、数组信息表、函数信息表等。对于存在于编译全过程，甚至在运行期间还存在的表（如符号表）必须全局地考虑，使得能一致而方便、安全地使用。

对于仅在某一趟中局部地使用的表格可在具体实现时再细加考虑。例如，词法分析阶段的符号机内表示对照表之具体实现可在词法分析时考虑。当然，涉及

与语法分析之衔接的符号值编码应在总体设计期间确定。

总之，涉及两个或多个阶段的全局性数据之结构必须在总体设计时全局地考虑。仅在一个阶段使用的局部数据之结构可在相应阶段之实现时考虑。

4. 确定各阶段之实现技术

众所周知，从形式语言角度出发，词法分析、语法分析与语义分析各自在不同的 Chomsky 文法类基础上进行，因而可以采用最适合于本阶段的分析技术。词法分析采用正则文法分析技术，与有穷状态自动机相联系。语法分析采用上下文无关文法分析技术，与语法分析树或推导相联系。而语义分析则以属性文法为基础，实施语法制导的翻译。这仅是概括的情况，事实上，各阶段可以对实现技术做出进一步的选择。例如，语法分析可采用自顶向下或自底向上分析技术，这两大类中又可有若干种选择。

当然，只要各阶段的相互接口已经确定，明确输入是什么、输出又是什么，那么作为黑箱，每一阶段可自行选择实现技术与方法。只是考虑到总的设计指标，各阶段应采用合适的技术。

5. 实现各趟

依据各趟的功能及其输入与输出进行设计，包括数据结构与控制部分，画出控制流程示意图(或其他表示)，编写程序，设计调试实例，进行调试。最终完成各趟而实现编译程序的构造。

1.4 本书阅读指南

1. 关于全书

本教材是计算机编译原理课程的配套教材，共分两部分，第一部分主要讨论了计算机程序的研制与软件的开发，讨论了如何利用 C/C++语言开发平台进行程序研制，并给出了上机实习报告提纲。第二部分各章在概括了编译程序构造相关的基本概念之后给出了实习题，除了明确各题的功能要求，给出了主要数据结构，还进行问题分析，指出各实习题的实现要点及其解决，供读者参考。

本教材期望有助于下列几类读者：

- 选修编译原理课程的学生，作为学习的辅助教材，加深对编译原理课程概念的理解；
- 自学编译原理课程的读者，加深对编译原理课程的理解和要点的领会；
- 打算使用 C/C++语言开发平台研制程序的读者，本教材可作为这些平台的操作入门；
- 编译程序构造实践者，作为实现的指南。

2. 关于C/C++语言开发平台

本教材从实用操作角度介绍了C/C++语言开发平台，不熟悉其操作的读者上机时可以对照进行界面的设计，建立应用程序，反复练习，达到熟练的程度，在具体实现实习题时便可把重点集中到实习题本身的内容。

3. 关于编译程序构造的相关概念

本教材是编译程序构造的实践指南，为了使得程序正确地反映每个实习题的功能要求，在实践篇每一章第一部分中对相关概念进行了简单的回顾与概括。原先对概念理解不够清楚的读者可以参考；原先掌握较好的读者则可以跳过不读。

4. 关于实习题

1）阅读要点

本教材中每个实习题给出下列内容：

一、题目

二、功能

三、要求与说明

四、问题分析

五、数据结构设计

六、界面设计

七、程序控制流程示意图

八、调试实例

九、补充说明

其中问题分析部分概括了解决本实习题的要点问题及其解决；数据结构部分列出了所涉及的主要数据结构。通常数据结构的详细设计在每章第二部分中给出，此处仅给出与本实习题相关的数据结构之设计。界面设计部分无具体内容，仅提醒读者需要考虑界面设计问题。程序控制流程示意图则给出主要部分的控制流程示意图。有些实习题中给出了伪代码程序，特别在语义分析部分，在附文中给出了相应的语义子程序，读者可从中得到启发。建议读者开始时仅仅细读一个实习题的前面三部分，明确问题及其要求，然后自行考虑解决问题的思路，找出应解决的要点问题及其解决办法等，之后与教材相应部分相对照，做出自己的取舍。在完成实习题后再结合具体实现，回顾问题的要点及其解决，进行概括，这样将能取得更好的效果。

2）阅读方法

在前面的1）中我们看到，一个阅读方法是独立思考，先自行独立考虑实现思路等。在这里要强调的阅读方法是“静态”执行。

“静态”执行程序，甚至执行伪代码程序、流程图、四元式序列或虚拟机目标代码，也就是把自己的大脑看成是一台计算机，模拟执行这些程序、伪代码程序、流程