编译原理学习笔记

老师：东南大学计算机院廖力老师主讲　　编辑：刘文东　　时间：**2014** 年 **5** 月 **1** 日 **23:11:58**

1. 编译原理课程引论

1.1 课程概述

1.1.1 课程地位

1、计算机专业的专业基础课；

2、软件技术基础；

3、计算机专业学生必修的主干课；

4、研究生入学考试的课程之一。

1.1.2 课程作用

1、编译原理是介绍如何将高级程序设计语言变换成计算机硬件所能识别的机器语言，以便计算机进行处理。

2、理论基础坚实，其形式化系统不仅应用于编译技术，还大量应用于人工智能，多媒体技术及数据库等领域。

1.1.3 学习任务

1、掌握编译的理论基础和形式化（数学形式）系统；

2、了解编译的全过程及其具体实现方法，大作业就是做一个简单的编译程序。

1.1.4 学习方法

1、认真听课，认真理解书中的基本概念、基本原理和基本算法；

2、弄懂书中的例题与习题；

3、在看书和理解例题时，一定要画出相应的细节变换过程，通过画图加深理解；

4、在理解的基础上记忆；

5、理论结合实践。

1.1.5 学习要求

1、成绩考核方式：平时 30%+期末考试 70% = 100%

2、平时成绩包括点名、作业和大作业。

1.2 课程引论

1.2.1 程序设计语言与编译

1、程序设计语言：机器语言——>汇编语言——>高级语言；

1) 机器语言：01 串，计算机硬件识别容易，但人理解起来比较困难；

2) 汇编语言：对机器语言的抽象，加入助记符，帮助人们理解；

3) 高级语言：引入编译程序，将高级语言变成计算机可以识别的程序。

2、在计算机上执行一个高级语言程序的步骤：

1) 将高级语言程序翻译成机器语言程序；

2) 运行所得到的机器语言程序求得计算结果。

3、程序设计语言之间的转换

1) 翻译：指能把某种语言的源程序，在不改变语义的条件下，转换成另外一种语言程序——目标语言程序；

2) 编译：专指由高级语言转换为低级语言，对整个源程序全部翻译好后，再执行；

3) 编译的转换过程：

- 两个阶段转换：编译——>运行，编译后直接产生机器语言目标代码；
  - ✓ 编译时：源程序——>编译程序——>目标代码，此时直接产生机器语言；
  - ✓ 运行时：初始数据——>运行子程序目标代码——>计算结果；
- 三个阶段转换：编译——>汇编——>运行，编译后现产生汇编语言，然后汇编后，转换成机器语言目标代码。
  - ✓ 编译时：源程序——>编译程序——>汇编程序；
  - ✓ 汇编时：汇编语言——>汇编程序——>目标代码；
  - ✓ 运行时：初始数据——>运行子程序目标代码——>计算结果；

4) 解释：接收某种高级语言的一个语句输入，进行解释并控制计算机执行，马上得到这句的执行结果，然后再接收下一句。如 Basic 语言、Java 语言等。

- 含义：以源程序作为输入，不产生目标程序，一边解释一边执行；
- 优点：直观易懂，结构简单，易于实现人机对话；
- 缺点：效率低，不产生目标程序，每次都有重复的动作。

1.2.2 编译程序概述

1、编译程序的工作

编译程序的工作如图 1.1 所示：

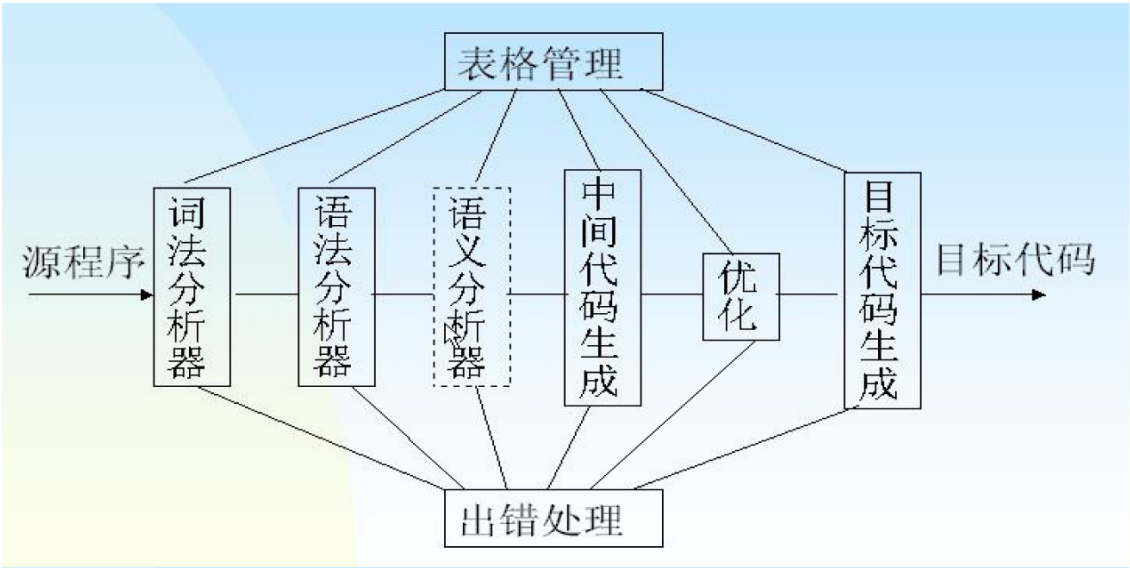

图 1.1 编译程序的工作流程

2、自然语言的翻译步骤：

1) 识别出句子的一个个单词；

2) 分析句子的语法结构；

3) 根据句子的含义进行初步翻译；

4) 对译文进行修饰；

5) 写出最后译文。

3、本节中要用到的 **C** 代码示例

本节用到的 C 代码示例如图 1.2 所示：

```
Void  jisuan()
{int y,c,d;
  float x,a,b;
  x=a+b*50;
  y=c+)d*(x+b;
  }
```

图 1.2 C 代码示例

4、程序语言的翻译步骤：

1) 词法分析；
   - 任务：输入源程序，对构成源程序的字符串进行扫描和分解，识别出一个个单词；
   - 单词：高级语言中有实在意义的最小语法单位，它由字符组成，单词可以分为：
     - ✓ 基本字（保留字）：void、int、float、
     - ✓ 标示符：a、b、c、d、x、y、jisuan、
     - ✓ 整常数：50、
     - ✓ 运算符：+、-、*、=、
     - ✓ 界定符：{、}、;、,、(、)、
   - 功能：词法分析依照语法规则，识别出正确的单词。转换成统一规格；
   - 转换：
     - ✓ 对基本字、运算符、界限符的转换；
     - ✓ 对标识符的转换，标识符由用户定义，转换比较麻烦，转换后的格式为：（类号，内码）；
     - ✓ 对常数的转换；
     - ✓ 转换完成后的格式：类号、内码；
   - 描述词法规则的有效工具：正规式和有限自动机；

2) 语法分析；
   - 任务：在词法分析的基础上，根据语言的语法规则，把单词符号组成各类的语法单位：短语、子句、语句、过程、程序；
   - 语法规则：又称文法，规定单词如何构成短语、语句、过程和程序。
   - 语法规则的表示：
     - ✓ BNF： $A ::= B \mid C$
       - ✧ 赋值语句的语法规则：

$A ::= V = E$

$E ::= T \mid E + T$

$T ::= F \mid T * F$

$F ::= V \mid (E) \mid C$

$V ::=$ 标识符

$C ::=$ 常数

- 方法：推导（derive）和规约（reduce）
  - ✓ 推导（derive）：分为最左推导和最右推导；
  - ✓ 规约（reduce）：分为最右规约和最左规约；
    - ✧ 最右推导和最左规约的图示如图 1.3 所示：

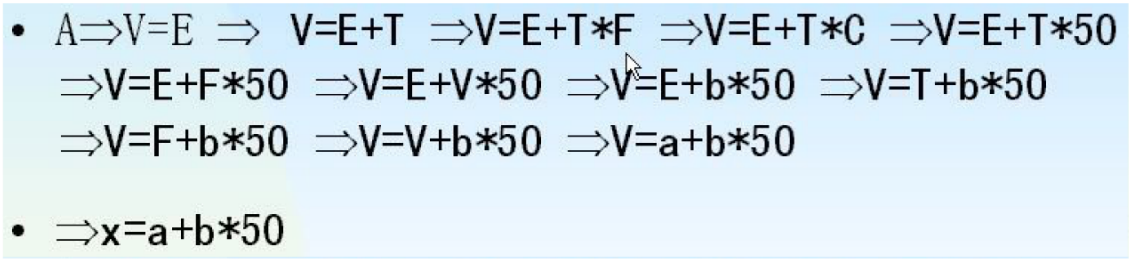

图 1.3 最右推导和最左规约的图示

- ✧ 最左推导和最右规约的图示如图 1.4 所示：

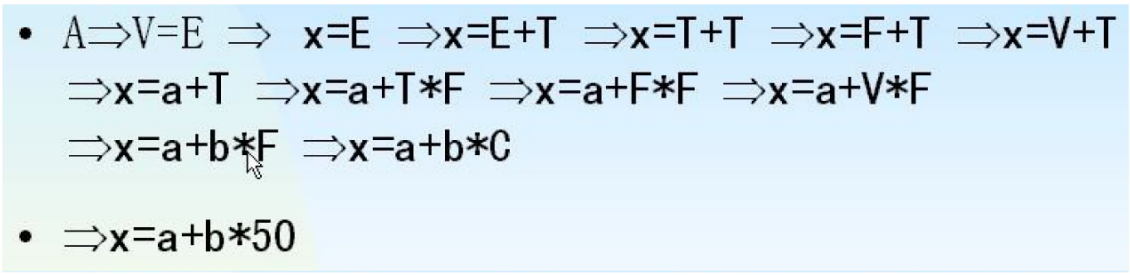

图 1.4 最左推导和最右规约的图示

- ✧ 错误语句推导图示如图 1.5 所示：

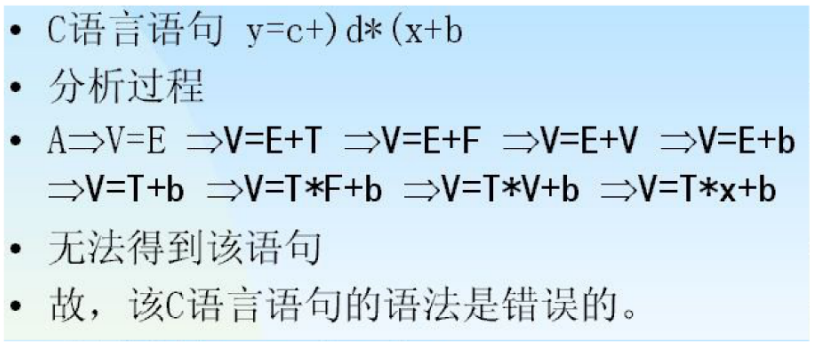

图 1.5 错误语句推导图示

- 语法树的表示方法：

  如图 1.6 所示：

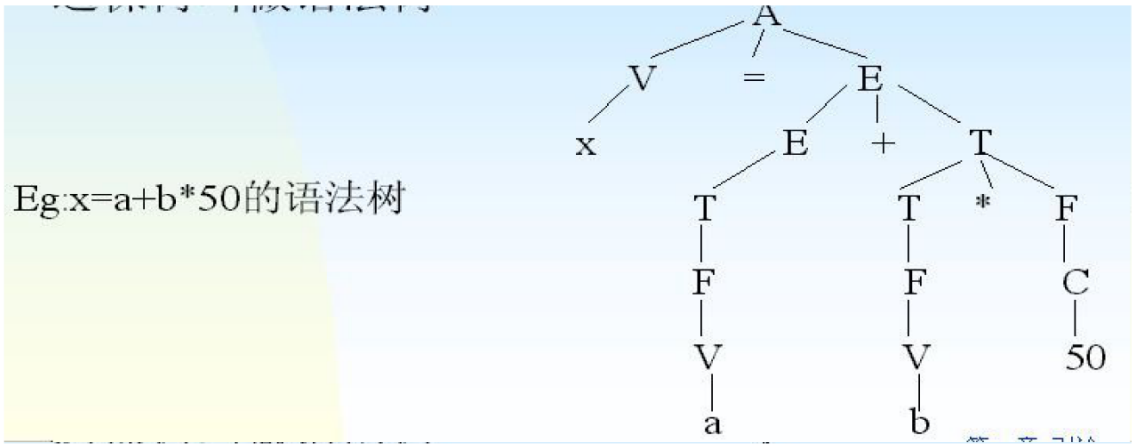

图 1.6 语法树的分析图示

3) 语义分析和中间代码生产；

- 任务：对语法分析识别出的各类语法范畴，分析其含义，进行和初步翻译，产生介于源代码和目标代码之间的一种中间代码。
- 分为两个阶段：
  - ✓ 对每种语法范畴进行静态语义检查；
  - ✓ 若语义正确，就进行中间代码的翻译；
- 中间代码形式：四元式、三元式、逆波兰式；
- 例如，将 $x=a+b*50$ 变成中间代码，如图 1.7 所示：

序号	算符	左操作数	右操作数	结果
(1)	将整常数50转换为实常数			T_1
(2)	*	b	T_1	T_2
(3)	+	a	T_2	T_3
(4)	=	T_3		x

图 1.7 中间代码示例

4) 中间代码优化；

- 任务：对前面产生的中间代码进行加工变换，以期在最后阶段能产生更为高效的目标代码；
- 原则：等价变换；
- 主要方面：公共子表达式的提取、合并已知量、删除无用语句（注释等）、循环优化（使循环中的内容是必须的）等；

5) 目标代码生成。

- 任务：将经过优化的中间代码转换成特定机器上的低级语言代码；
- 目标代码的形式：
  - ✓ 绝对指令代码：可立即执行的目标代码，完全有 01 串组成；
  - ✓ 汇编指令代码：汇编语言程序，需要通过汇编程序汇编后才能执行；
  - ✓ 可重定位指令代码：现将各目标模块连接起来，确定变量、常量在主存中的位

置，装入主存后才能成为可以运行的绝对指令代码。

5、表格与表格管理

1) 表格作用：用来记录源程序的各种信息以及编译过程中的各种状况；
2) 与编译前三个阶段有关的表格有：符号表、常数表、标号表、分程序入口表、中间代码表等；
   - 符号表：用来登记源程序中的常量名、变量名、数组名、过程名等，记录其性质、定义和引用情况；如图 1.8 所示：

NAME	INFORMATION
m	整型、变量地址
n	整型、变量地址
k	整型、变量地址

图 1.8 符号表示例

   - 常数表和标号表：如图 1.9 所示：

常数表

值
1
4

（登记各类常量值）

标号表

NAME	INFORMATION
……	…….
10	四元式序号4

（登记标号的定义与应用）

图 1.9 常数表和标号表

   - 入口名表：登记过程的层号，分程序的符号表入口等，如图 1.10 所示：

NAME	INFORMATION
……	…….
INCWAP	二目子程序、四元式序号1

图 1.10 入口名表示例

   - 中间代码表：如图 1.11 所示：

序号	OP	ARG1	ARG2	RESULT
（1）	=	I		m
（2）	=	j		n
（3）	=	1		k
（4）	J<	100	k	(9)
（5）	+	m	10	m
（6）	+	n	10	n
（7）	+	k	1	k
（8）	j			(4)
（9）	return			

图 1.11 中间代码表示例

6、出错处理

1) 任务：如果源程序中有错误，编译程序应设法发现错误，并报告给用户；
2) 完成：由专门的出错处理程序来完成；
3) 错误类型：
   - 语法错误：在词法分析和语法分析阶段检测出来；
   - 语义错误：一般在语义分析阶段检测。
   - *逻辑错误：在编译阶段不处理。*

7、遍

1) 遍：指对源程序或源程序的中间结果从头到尾扫描一遍，并做有关的加工处理，生成新的中间结果或目标代码；注：遍与阶段的含义毫无关系；
2) 一遍扫描：如图 1.12 所示：

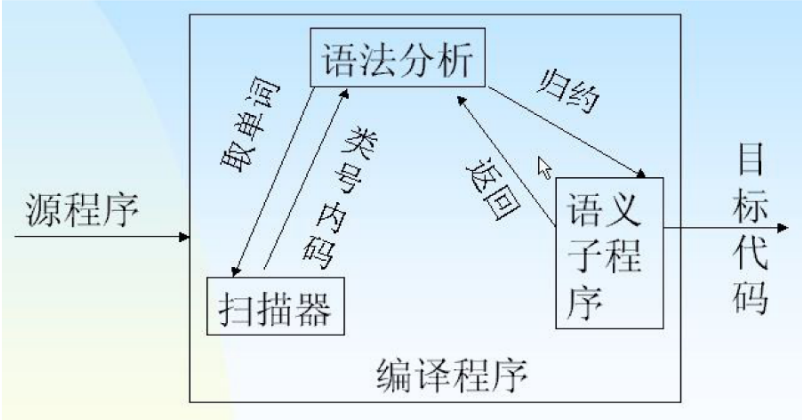

图 1.12 一遍扫描示意图（以语法分析为中心）

3) 多遍扫描：
   - 优点：节省内存空间，提高目标代码质量，使编译的逻辑结构清晰；
   - 缺点：编译时间长；
   - 注：在内存许可的情况下，还是遍数尽可能少些为好。

1.2.3 编译程序生成

1、直接用机器语言编写编译程序；
2、用汇编语言编写编译程序：编译程序的核心部分常用汇编语言编写；
3、用高级语言编写编译程序，这是普遍采用的方法。
4、自编译：先编写编译程序的核心部分，然后再考虑其他部分。
5、编译工具：LEX（词法分析）与 YACC（用于语法分析中自动产生 LALR 分析表）；
6、移植：同种语言的编译程序在不同类型的机器之间移植

1.2.4 编译程序构造

1、在某机器上为某种语言构造编译程序要掌握以下三方面：

1) 源语言：
2) 目标语言：
3) 编译方法：本门课要讲解的重点！

1.3 本章小结：

1、掌握编译程序与高级程序设计语言之间的关系；

2、掌握编译分为哪几个阶段；

3、了解各个阶段完成的主要功能和采用的主要方法。

2. 编译基础知识

2.1 编译基础知识

2.1.1 高级语言与词法分析、语法分析概述：

1、程序语言是一个记号系统；

1) 语法：
2) 语义：

2、语法：

1) 任何语言程序都可以看成是一定字符集（字母表）上的字符串；
2) 语法使得这串字符形成一个形式上正确的程序；
3) 语法 = 词法规则 + 语法规则；
4) 例如：$0.5*x1+c$，其中：
   - 语言的单词符号：0.5、x1、c、*、+
   - 语言的语法单位：$0.5*x1+c$
5) 单词符号：语言中具有独立意义的最基本的结构；
6) 词法规则：
   - 词法规则规定了字母表中哪些字符串是单词符号；
   - 单词符号一般包括：常数、标识符、基本字、算符、界限符等；
   - 我们用正规式和有限自动机理论来描述词法结构和进行词法分析。
7) 语法单位：表达式、子句、语句、函数、过程、程序等；
8) 语法规则：
   - 规定了如何从单词符号来形成语法单位；
   - 现在多数程序语言使用上下文无关文法来描述语法规则；
   - 语言的词法规则和语法规则定义了程序的形式结构，是判断输入字符串是否形成一个形式上正确的程序的依据。

3、语义：

1) 对于一个语言来说，不仅要给出它的词法、语法规则，而且要定义它的单词符号和语法单位的意义；
2) 离开语义，语言只是一堆符号的集合；
3) 各种语言中有形式上完全相同的语法单位，含义却不尽相同；
4) 对某种语言，可以定义一个程序的意义的一组规则称为语义规则；
5) 目前，大多数编译程序使用基本属性文法的语法制导翻译方法来分析语义。

6) 对于高级程序设计语言及其编译程序来说，语言的语义分析是很重要的。本章主要介绍语法结构的形式描述问题，编译原理的主要内容也可以归结为应用形式语言理论，并将它贯穿于词法分析和语法分析两个阶段；
7) 本章重点讨论正规文法、上下文无关文法及其对应的有限自动机和下推自动机，以及在构造编译程序时如何应用。

2.1.2 字母表与符号串

1、相关概念：

1) 字母表：
   - 是符号的非空有穷集合；
   - 用Σ、V 表示；
2) 符号：语言中最基本的不可再分的单位；
3) 符号串：
   - 符号串是字母表中符号组成的有穷序列；
   - 空串：不含有任何符号的串，记作：ε
4) 句子：字母表上符合某种规则构成的串；
5) 语言：字母表上句子的集合；
6) 注：约定用 a,b,c...表示符号；用α,β,γ...表示符号串；用 A,B,C...表示其集合。

2、符号串结合的运算：

1) 连接（乘积）运算：
   - 定义：若串集$A=\{\alpha_1,\alpha_2,...\}$，串集$B=\{\beta_1,\beta_2,...\}$，则乘积$AB=\{\alpha\beta|\alpha\in A \text{ and } \beta\in B\}$
   - 注：
     - ✓ 串集的自身乘积称作串集的方幂；
     - ✓ $A^0=\{\varepsilon\}$
     - ✓ 字母表 A 的 n 次方幂是字母表 A 上所有长度为 n 的串集。
   - 举例说明：如图 1.13 所示：

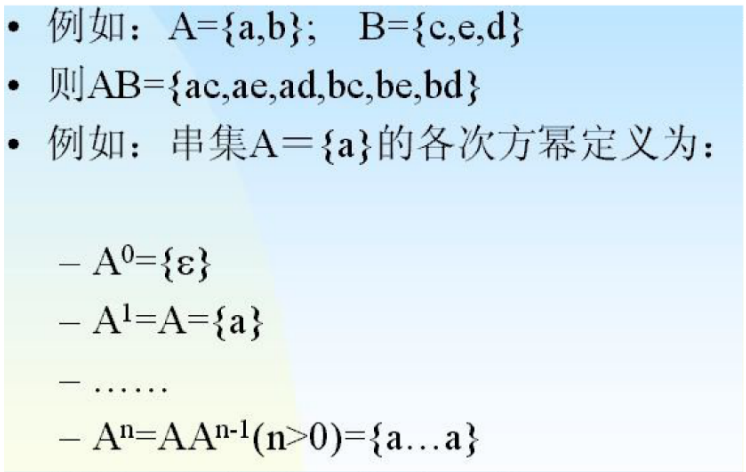

图 1.13 连接运算举例

3、字母表的闭包与正闭包

1) 字母表A的闭包（A^*）：$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots$
即：由A上符号组成的所有串的集合（包含空串ε）

2)