**作者：20计算机科学与技术一班 刘其昕**

第一章

**1、编译器（编译程序）的组成部分及其任务：**

1. 词法分析器（扫描器）：输入**源程序**，进行**词法分析**，输出**单词符号**；
2. 语法分析器（分析器）：对**单词符号串**进行**语法分析**（根据语法规则进行推导或归约），识别出各类语法单位，最终判断输入串是否构成语法上正确的“程序”；
3. 语义分析与中间代码产生器：按照**语义规则对语法分析器归约（或推导）**出的语法单位进行语义分析并将其翻译成一定形式的**中间代码**；
4. 优化器：对中间代码进行优化处理；
5. 目标代码生成器：把中间代码翻译成目标程序；

**2、编译器前后端的划分：**

1. 编译前端：主要由与**源语言有关但与目标机**无关的部分组成，通常包括**词法分析、语法分析、语义分析与中间代码产生**，有的代码的优化工作也可包括在前端；
2. 编译后端：包括编译程序中与目标机有关的部分，如与**目标机有关的代码优化**和**目标代码生成**等。通常**不依赖于源语言而仅仅依赖于中间语言**。

3、计算机执行高级语言编写的程序的途径及区别：

1. 编译程序：**将源语言程序**（如FORTRAN、Pascal、C、Ada、Smalltalk、Java等“高级语言”）转换成**目标语言程序**（如汇编语言、机器语言等“低级语言”）的翻译程序被称为**编译程序**；
2. 解释程序：一个源语言的解释程序是以该语言写**的源程序作为输入**，**但不产生目标程序**，而是**边解释边执行源程序本身**。

第二章

**1、与文法有关的基本概念：**

1. ε、Φ（“{}”）和{ε}的区别；
2. 连接积和n次连接积、V0={ε}、V+=VV\*（V+是V的正则闭包）；
3. 文法：**描述语言的语法结构的形式规则**（即语法规则）；

**2、上下文无关文法的组成部分：**

1. **终结符号**：组成语言的不可再分的基本符号；
2. **非终结符**：代表语法范畴，表示一定符号串的集合（由终结符与非终结符组成的串）；
3. **开始符号**：特殊的**非终结符**，代表所定义语言的语法范畴（句子）；
4. **产生式**：定义语法范畴的书写规则；

**3、文法的二义性及二义性问题的不可判定：**

1. 二义文法：存在**某个句子**对应**两棵不同的语法树**（**最左或最右推导**）的文法；
2. 二义性问题的不可判定：不存在一个算法，能在**有限步骤**内，确切地判定一个文法是否为**二义的**；

**4、乔姆斯基四型文法：**

1. 0型文法（短语文法）：能力相当于图灵机，任何0型文法都是**递归可枚举**的；
2. 1型文法（上下文有关文法）：对**非终结符**进行**替换必须考虑上下文**，且一般**不允许替换为空串ε**；
3. 2型文法（上下文无关文法）：对**非终结符**进行替换可以**不必考虑上下文**，对应非确定的下推自动机；(NFA)
4. 3型文法（正规文法）：能力比**上下文无关文法**弱得多，无法产生部分上下文无关文法产生的语言；

第六章

**属性分类：**

1. 通常分为**综合属性**（“自下而上”传递信息）和**继承属性**（“自上而下”传递信息），终结符只有综合属性（由词法分析器提供），非终结符可以有两种属性，文法开始符号的所有继承属性作为属性计算前的初始值；
2. 语法树中一个结点**综合属性**的值由其**子节点的属性值**确定，仅使用综合属性的属性文法为S-属性文法；一个结点**继承属性**的值由此节点的**父节点和/或兄弟结点**的某些属性确定；

第七章

**不同中间代码之间的比较（中间语言？）**

第八章

**1、符号表**

符号表是一种用于**语言翻译器**（例如编译器和解释器）中的**数据结构**。在符号表中，程序源代码中的每个**标识符**都和**它的声明或使用信息**绑定在一起，比如**其数据类型、作用域以及内存地址**。符号表的每一项包含两个部分：一部分是名字（标识符）；另一部分是此名字的有关信息（名字栏[主栏]和信息栏），名字栏的内容称为**关键字**。

**2、符号表的不同实现方法及特点**

1. 线性表与线性查找：**最简单，但效率低**；
2. 二叉树与对折查找：**查找效率高一些**，实现上略困难；
3. 杂凑技术（哈希）：效率最高，实现上比较复杂且要消耗额外存储空间；

第十章

1. **优化的原则**

**等价、有效、合算=结果不变+效率提高+代价合理**

1. **等价原则**：经过优化后不应改变程序运行的结果；
2. **有效原则**：优化后所产生的的**目标代码的运行时间较短**，占用的存储空间较小；
3. **合算原则**：尽可能以较低的代价取得较好的优化效果；

**2、常见优化方法及其作用：**

1. **删除公共子表达式**：避免对公共子表达式的重复计算；
2. **复写传播**：使对某些变量的赋值变得无用；
3. **删除无用代码**：将某些对程序运算结果无用的赋值变量删除；
4. **代码外提**：使某些循环中产生结果不变的代码提出循环，减少运算次数；
5. **强度削弱**：将乘除法变换为加减法，提高代码运行速度；
6. **删除归纳变量**：减少代码量和部分代码的执行次数，提高运行速度；

# 局部优化、循环优化和数据流分析？