# 《2-构造一个简单的编译器》学案

1. **学习目的**

能力获取：纯手工打造一个简单的编译器

1. 对编译器六个阶段所采用的基本技术有一个初步认识，特别是语法分析阶段。
2. 理解以语法分析器为中心构造编译器，掌握递归下降构造语法分析器的方法和语法制导翻译方法。
3. **学习内容**
4. 上下文无关文法：
5. 理解CFG的用处：描述编程语言的语法。
6. 理解产生式的概念，描述了一条语法特性的规则，理解其结构，左部、右部、终结符、非终结符等概念。
7. 理解CFG的四元式定义，形式化定义，理解符号串、符号串集合及其运算的概念，为第3、4章系统讲解词法分析、语法分析打下基础。
8. 掌握产生式设计的基本方法。

**重点&难点：产生式的概念，结合第一章（P35、36），没有什么神秘的，就是我们对语法范畴的理解的一种符号化表示，与自然语言描述是等价的，左部是语法范畴，右部是其可具有的形式，涉及到的组成成分都符号化。**

**四元式表示，注意一些基本概念弄清楚，如空字等。**

**产生式设计，基本方法直接来源于产生式基本概念，对语法范畴有了准确理解后，进行符号化即可。**

1. 推导和语法树：
2. 理解推导的基本概念及——开始符号到单词串的语法等价替换链条，及其作用：静态的CFG🡪动态的语法分析——找到推导过程。
3. 理解推导过程与语法树的对应关系。
4. 理解二义性的概念，能对实例判断是否具有二义性，如何消去二义性。
5. 理解文法压缩的例子。

**重点&难点：推导和语法树的对应关系，推导过程——语法树自顶向下构造过程，每个推导步骤对应语法树的父子节点局部——被替换的非终结符（产生式左部）为父节点、替换的符号（产生式右部）创建为孩子节点。**

**二义性，同一个单词串有多棵语法树（而不是多个推导）。**

1. 语法制导翻译——语法制导定义
2. 理解为什么先学习语法制导翻译，然后再学习语法分析，学会设计语法制导翻译后，借助Yacc工具即可构造语法分析+翻译模块，而无需了解语法分析算法。
3. 理解“语法”“制导”翻译的核心思想，我们不可能穷举所有可能的单词串为它们逐个设计翻译方法——为每个语法范畴（CFG每个产生式）设计翻译方法即可搞定所有单词串。
4. 理解语法制导定义设计思路，理解基于语法制导定义进行翻译的方法，每个语法符号设计属性值表示翻译结果，对每个产生式设计语义规则——属性值计算表达翻译方法，翻译过程——语法树后序遍历计算属性值🡪注释语法树、根节点属性值为翻译结果。

**重点&难点：语法制导定义的设计，人理解的翻译方法符号化为每个产生式的语义规则——语法范畴的翻译结果（左部非终结符的属性值）用组成成分的翻译结果（右部终结符、非终结符的属性值）计算出来。通过讲义中的实例、产生式、语义规则的对应来理解这一设计思路。**

1. 翻译模式
2. 理解与语法制导定义的区别，前者描述了属性计算，翻译模式更贴近具体实现。
3. 理解翻译模式的设计：将实现属性计算（或直接输出翻译结果）的代码（“语义动作”）嵌入到产生式中；理解翻译过程：语义动作作为终结符、非终结符之外的特殊节点加入语法树中，后序遍历语法树，遍历到语义动作节点执行相应代码（完成属性计算或结果输出）。

**重点&难点：语义动作的嵌入非常重要——决定了语法树中所处位置——决定了翻译过程中执行顺序，通过中缀转后缀、中缀转前缀的例子理解这一点。**

**语法制导定义到翻译模式的转换，保持属性计算——考虑属性依赖关系确定嵌入位置；直接输出结果——子树中已输出的不重复输出——这一原则仍是根据翻译过程得来。**

1. Yacc编程
2. 理解Yacc工具的作用，我们无需掌握语法分析算法，即可借助它来构造语法分析+翻译的程序。
3. 掌握Yacc工具的使用，Yacc本身也是一个编译器，我们编写Yacc程序——翻译模式（嵌入语义动作的CFG），Yacc工具将其转换为语法分析+翻译的源程序，编译后即得到语法分析+翻译器。

**重点&难点：翻译模式到Yacc程序的转换，Yacc程序的三段式结构分别对应必要定义（未来希望置于.h和.c文件中的定义，及词法、语法的一些基本定义）、翻译模式和辅助子程序。语法符号属性对应的Yacc对象$$、$1、…。**

1. 自顶向下语法分析——预测分析法：
2. 理解自顶向下分析基本概念，第1章就见过，与推导过程完全对应，理解其中难点，如何确定用哪个产生式替换非终结符——平凡算法逐个尝试，回溯可能导致指数时间。
3. 理解预测分析法基本思想，如何避免回溯保证线性时间复杂度？结合输入单词和待替换非终结符的候选产生式的首单词来确定唯一可能的替换产生式。

**重点&难点：理解确定唯一产生式的规则。反过来思考，用产生式A🡪替换A及接下来的推导步骤是希望推导出输入单词串（以a开始），即A=>=>ax，这意味着必然能推导出以a开始的单词串，这就得出确定唯一可能的替换产生式的方法：产生式预先计算的首单词集合与输入单词a匹配。**

1. 设计实现预测分析器
2. 理解“首单词”集合的计算方法，本质上还是寻找推导结果，用其他产生式进行替换，直到首符号是单词为止，考虑所有可能替换。
3. 理解递归下降实现方法，预测分析法基本操作是对下一个NT A根据下一个输入单词a确定用哪个产生式进行替换，为每个NT设计一个函数完成此操作——文法的递归特性导致函数是递归的，确定了产生式后进行替换——即处理右部每个符号——具体的：T与输入单词匹配、NT调用对应函数。
4. 理解在递归下降实现中加入翻译模式，在处理右部符号时，遇到语义动作将代码插入到函数相应位置即可。
5. 理解预测分析法对文法结构的要求，不能有左递归、不能有左左公因子，掌握文法改写算法。

**难点：递归下降实现为何不适用于左递归文法？左递归文法意味着如果选择此产生式进行替换的话，接下来第一个操作是对自身的调用，这期间没有处理任何终结符，也就是说递归调用后仍会选择此产生式，继续递归调用，形成死循环。**

**左递归的消除，将递归（重复）部分用右递归描述，非递归的开头部分分离出来。**

1. 简单词法分析：
2. 理解词法分析的作用和在编译器中的位置，作为语法分析的功能函数，识别出下一个单词（以及滤掉空白符、注释等），理解朴素算法思想。
3. 理解符号表的基本概念，在词法分析阶段为每个标识符创建表项，在语法分析阶段填入更多信息，在综合阶段使用这些信息。

**重点&难点：理解朴素的词法分析算法。首先是描述单词，类似语法描述，描述清楚每个单词“长什么样”，还没有类似CFG的形式化描述方式，用自然语言表达即可，以无符号整数、标识符等为例思考此问题。然后，发现单词的“模样”中主要是特定符号的顺序结构和重复结构，“特定符号”——if语句搞定，重复结构——循环语句搞定。**

**理解词法分析和语法分析的交互，词法分析返回识别的单词给语法分析，如NUM、ID，这里只关心单词类别，不关系具体实例。但在综合阶段需要用到实例的属性进行代码生成，因此还需返回实例的属性（比如可通过全局变量）给语法分析。**