什么是编译？

把源程序转换成等价的目标程序的过程即是编译

低级语言:机器语言 符号语言 汇编语言

高级语言: 过程性语言—面向用户的语言如：C、Pascal

专用语言—面向问题的语言 如：SQL

面向对象的语言 如：Java、C++

高级语言优点：

高级语言独立于机器，所编程序移植性比较好。

不必考虑存储单元的分配问题，数据的外部形式转换成机器的内部形式等细节。

具有丰富的数据结构和控制结构。

更接近于自然语言。

可读性好，便于维护。

编程效率高。

编译程序的设计涉及到的知识：

程序设计语言

形式语言与自动机理论

计算机体系结构

数据结构

算法分析与设计

操作系统

软件工程

编译阶段

一、分析阶段 根据源语言的定义，分析源程序的结构（根据源语言的定义，对源程序进行结构分析和语义分析，从而把源程序正文转换为某种内部表示）

1.词法分析

2.语法分析

3.语义分析

二、综合阶段 根据分析结果构造出所要求的目标程序（根据所制定的源语言到目标语言的对应关系，对分析阶段所产生的中间形式进行综合加工，从而得到与源程序等价的目标程序。）

4.中间代码生成

5.代码优化

6.目标代码生成

三、符号表的管理

四、错误诊断和处理

词法分析器：

依次读入源程序中的每个字符，对构成源程序的字符串进行分解，识别出每个具有独立意义的字符串作为记号（token）并组织成记号流。

把需要存放的单词放到符号表中，如变量名，标号，常量等。

形成记号的字符串叫做该记号的单词(lexeme)。

工作依据：源语言的构词规则(即词法)，也称为模式(pattern)

语法分析：

层次结构的分析

把记号流按语言的语法结构层次地分组，以形成语法短语。

源程序的语法短语常用分析树表示

工作依据：源语言的语法规则

语义分析：

对语句的意义进行检查分析

收集类型等必要信息

用语法分析确定的层次结构表示各语法成份

工作依据：源语言的语义规则

一个重要任务：类型检查

根据规则检查每个运算符及其运算对象是否符合要求；

数组的下标是否合法；

过程调用时，形参与实参个数、类型是否匹配等

中间代码生成：

中间代码：一种抽象的机器程序

中间代码应具有两个重要的特点：

易于产生

易于翻译成目标代码

中间代码有多种形式

三地址代码具有的特点：

每条指令除了赋值号之外，最多还有一个运算符。

编译程序必须生成临时变量名，以便保留每条指令的计算结果。

有些“三地址”指令少于三个操作数

目标代码生成：

生成的目标代码一般是可以重定位的机器代码或汇编语言代码。

涉及到的两个重要问题：

对程序中使用的每个变量要指定存储单元

对变量进行寄存器分配

错误处理

词法分析程序可以检测出非法字符错误。

语法分析程序能够发现记号流不符合语法规则的错误。

语义分析程序试图检测出具有正确的语法结构，但对所涉及的操作无意义的结构。

代码生成程序可能发现目标程序区超出了允许范围的错误。

由于计算机容量的限制，编译程序的处理能力受到限制而引起的错误。

前端后端

前端主要由与源语言有关而与目标机器无关的那些部分组成

词法分析、语法分析、符号表的建立、语义分析和中间代码生成

与机器无关的代码优化工作

相应的错误处理工作和符号表操作

后端由编译程序中与目标机器有关的部分组成

与机器有关的代码优化、目标代码的生成

相应的错误处理和符号表操作

把编译程序划分成前端和后端的优点：

便于移植、便于编译程序的构造。

一“遍”是指对源程序或其中间形式从头到尾扫描一遍，并作相关的加工处理，生成新的中间形式或目标程序。

分遍的主要好处：

可以减少对主存容量的要求

可使各遍编译程序功能独立、单纯，相互联系简单，编译程序结构清晰。

能够实现更充分的优化工作，获得高质量目标程序。

通过分遍将编译程序的前端和后端分开，可以为编译程序的移植创造条件。

分遍的缺点：

增加了不少重复性的工作。

预处理器的主要功能：

1. 宏处理

宏定义通常用统一的字符或关键字表示，如define或macro，宏定义由宏名字及宏体组成，通常宏处理器允许在宏定义中使用形参。

宏调用由调用宏的命令名（宏名）和所提供的实参组成。宏处理器用实参代替宏体中的形参，再用变换后的宏体替换宏调用本身。

2.文件包含：预处理器把文件的包含声明扩展为程序正文。

3.语言扩充：有些预处理器用更先进的控制流和数据结构来增强原来的语言。

编译原理的应用

语法制导的结构化编辑器

程序格式化工具

软件测试工具

程序理解工具

高级语言的翻译工具

词法分析程序的作用：

扫描源程序字符流

按照源语言的词法规则识别出各类单词符号

产生用于语法分析的记号序列

词法分析程序与语法分析程序之间的三种关系

词法分析程序作为独立的一遍

词法分析程序作为语法分析程序的子程序

词法分析程序与语法分析程序作为协同程序

分离词法分析程序的好处

以简化设计

可以改进编译程序的效率

可以加强编译程序的可移植性

词法分析程序的实现方法

利用词法分析程序自动生成器

利用传统的系统程序设计语言来编写

利用汇编语言来编写

缓存区的必要性

为了得到某一个单词符号的确切性质，需要超前扫描若干个字符

记号、模式和单词

记号：是指某一类单词符号的种别编码，如标识符的记号为id，数的记号为num等。

模式：是指某一类单词符号的构词规则，如标识符的模式是“由字母开头的字母数字串”。

单词：是指某一类单词符号的一个特例，如position是标识符。

词法分析输出形式：二元式的形式：<记号，属性>

LEX源程序

1. 说明部分：变量说明 标识符常量说明 正规定义

2. 翻译规则

3. 辅助过程：对翻译规则的补充

LEX的工作原理

扫描每一条翻译规则Pi，为之构造一个非确定的有限自动机NFA Mi

将各条翻译规则对应的NFA Mi合并为一个新的NFA M

将NFA M确定化为DFA D，并生成该DFA D的状态转换矩阵和控制执行程序

二义性处理:最长匹配原则，优先匹配原则:字符串长度相同匹配最上面一项

语法分析程序

自顶向下

递归下降分析

从文法的开始符号出发，进行推导，试图推出要分析的输入串的过程。

对给定的输入串，从对应于文法开始符号的根结点出发，自顶向下地为输入串建立一棵分析树。

试探过程，是反复使用不同产生式谋求匹配输入串的过程。

为什么最左推导？对输入串的扫描是自左至右进行的，只有使用最左推导，才能保证按扫描的顺序匹配输入串。

递归下降分析方法的实现

文法的每一个非终结符号对应一个递归过程，即可实现这种带回溯的递归下降分析方法。

每个过程作为一个布尔过程，一旦发现它的某个产生式与输入串匹配，则用该产生式展开分析树，并返回true，否则分析树不变，返回false。

递归下降分析的困难和缺点

左递归的文法，可能导致分析过程陷入死循环。

回溯

工作的重复

效率低、代价高：穷尽一切可能的试探法。

递归调用分析

一种确定的、不带回溯的递归下降分析方法（消除左递归，提取左公因子

）

构造步骤： 描述结构的上下文无关文法

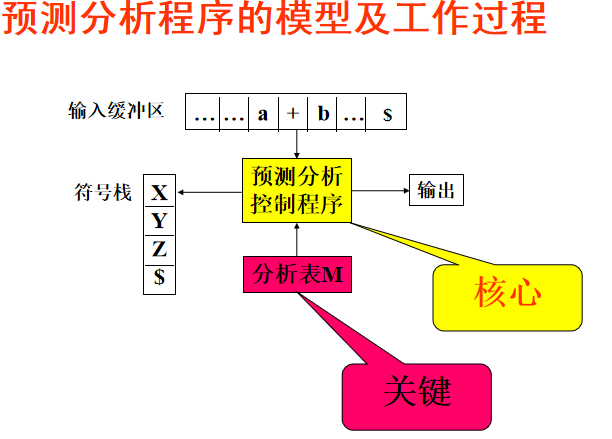
根据文法构造预测分析程序的状态转换图

状态转换图化简

根据状态转换图构造递归过程

非递归预测分析

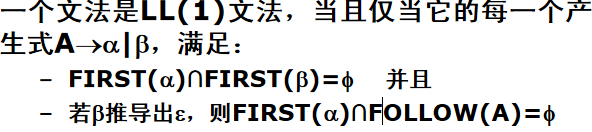
使用一张分析表和一个栈联合控制，实现对输入符号串的自顶向下分析。



FIRST(A)是开头终结符号集合

FOLLOW(A)是所有句型中，紧跟A之后出现的终结符号或$组成的集合

LL(1):如果一个文法的预测分析表M不含多重定义的表项，则称该文法为LL(1)文法。



LL(1)的含义：

第一个L表示从左至右扫描输入符号串

第二个L表示生成输入串的一个最左推导

1表示在决定分析程序的每步动作时，向前看一个符号

错误处理:栈顶终结符与当先输入符号不一致（弹出栈顶的终结符号）

栈顶非终结符A，当前输入a但分析表M[A,a]为空（跳过剩余输入符号串中的若干个符号，直到可以继续进行分析为止）。

自底向上

分析过程：

从输入符号串开始分析

查找当前句型的“可归约串”（关键）

使用规则，把它归约成相应的非终结符号

重复

规范归约是一个最右推导的逆过程，因此规范归约也称为最左归约（右句型序列）。

“移进-归约”方法的实现：使用一个寄存文法符号的栈和一个存放输入符号串的缓冲区。

移进-归约步骤：

移进：把下一个输入符号移进到栈顶。

归约：用适当的归约符号去替换这个串。

接受：宣布分析成功，停止分析。

错误处理：调用错误处理程序进行诊断和恢复

LR分析

LR(k)的含义：

L表示自左至右扫描输入符号串

R表示为输入符号串构造一个最右推导的逆过程

k表示为作出分析决定而向前看的输入符号的个数。

LR分析方法的基本思想

“历史信息”：记住已经移进和归约出的整个符号串；

“预测信息”：根据所用的产生式推测未来可能遇到的输入符号；

根据“历史信息”和“预测信息”，以及“现实”的输入符号，确定栈顶的符号串是否构成相对于某一产生式的句柄。

LR分析方法的优点：

可以分析所有能用上下文无关文法书写的程序设计语言的结构；

最一般的无回溯的“移进-归约“方法；

能分析的文法类是预测分析方法能分析的文法类的真超集

分析过程中，能及时发现错误，快到自左至右扫描输入的最大可能

LR分析方法的不足之处

手工编写LR分析程序的工作量太大

需要专门的工具，即LR分析程序生成器（如YACC）

活前缀：一个规范句型的一个前缀，如果不含句柄之后的任何符号，则称它为该句型的一个活前缀

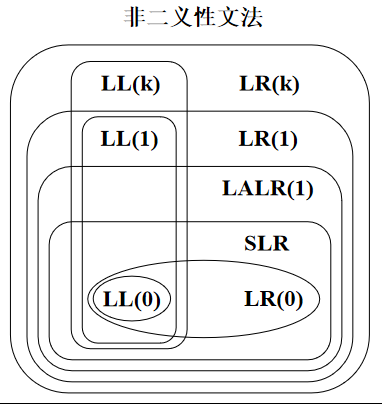
SLR(1)分析表的构造：

为给定的文法构造一个识别它所有活前缀的DFA

根据该DFA构造文法的分析表

每一个SLR(1)文法都是无二义的文法

如果在执行上述算法的过程中，始终没有向前看任何输入符号，则构造的SLR分析表称为LR(0)分析表



YACC说明文件

说明部分：C语言的声明、文法记号的声明

翻译规则部分：每条规则由一个产生式和有关的语义动作组成

辅助过程：用C语言书写一些语义动作中用到的辅助程序