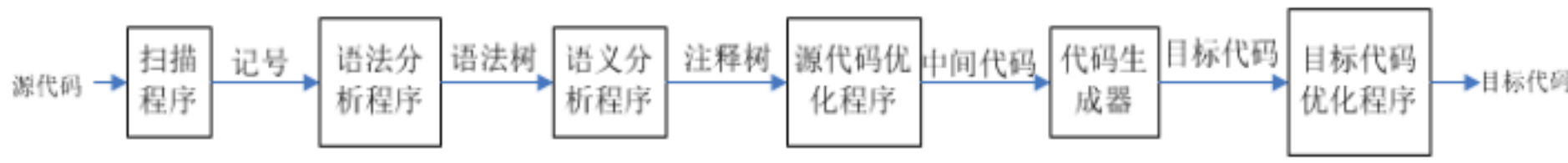
1. 编译器翻译步骤



2. ①词法分析： 将源程序读作字符文件并将其分为若干记号。

②语法分析： 从扫描程序中获得记号形式的源代码, 并确定程序的语法结构, 以及构造出表达该结构的语法树或分析树。

3. 代码优化的目的： 产生更加高效的代码，提高程序运行速度。

4. 编译器的几种数据结构：

①符号表 ： 是存放源程序中定义的所有符号信息的数据结构, 这个数据结构中的信息与标识符有关： 函数、 变量、 常量以及数据类型。 符号表几乎与编译器的所有阶段交互： 扫描程序、分析程序或将标识符输入带表格中的语义分析程序； 语义分析程序将增加数据类型和其他信息；优化阶段和代码生成阶段也将利用由符号表提供的信息选出恰当的代码。

②文字表 ：存放程序中用到的数字常量和字符串常量，且无需删除。

③错误处理器： 对源程序中错误的反应, 包括不同的操作, 每个操作给出指定的阶段和结构。

5. 编译器的前端和后端 分别包括哪几个阶段？前后端分开有什么好处？

前端： 只依赖于源语言的操作，包括扫描程序、分析程序和语义分析程序

后端 ：只依赖于目标语言的操作，包括代码生成器和一些优化分析

优点： 这一结构对于编译器的可移植性十分重要, 此时设计的编译器既能改变源代码, 又能改变目标代码。

6. 解释器和编译器 的本质区别：

①编译器把每一条语句都翻译成机器语言, 生成了可多次使用的目标代码； 而解释器没有生成目标代码，每一次执行程序都需要重新翻译。

②解释程序立即执行源程序，而不是生成在翻译完成之后才执行的目标代码。

7. NFA 和 DFA

①NFA（非确定性有穷自动机） 的基本特性在于从一个状态遇到同一个输入符号时, 可能有多个转换。

②相对于 NFA， DFA（确定性有穷自动机）中的状态转换都是确定的，在同一个状态下遇到同一个输入符号时只会存在一个转换。

8. 分析树和抽象语法树 的区别是什么？

①分析树是一个作了标记的树, 其内部的节点由其表示的结构名标出, 树叶节点则表示输入中的记号序列。②抽象语法树是源代码记号序列的抽象表示。

③分析树不利于表示程序结构，包括了比生成可执行代码所需要的更多信息。

④抽象语法树不能重新得到记号序列, 但是包含了转换所需的所有信息, 而且比分析树效率更高。

⑤分析程序可通过一个分析树表示所有步骤，但是通常只能构造出一个抽象语法树。

9. 最左推导 ：每一步中最左的非终结符都要被替换的推导。

最右推导 ：每一步中最右的非终结符都要被替换的推导。

10. 上下文无关文法 的定义：

上下文无关文法由以下各项组成：

①终结符集合 T

②非终结符集合 N

③产生式或文法规则 $A \to \partial$ 的集合 P，A 是 N 的一个元素，∂ 是 $(T \cup N)^*$ 的一个元素

④来自集合 N 的开始符号

11. 句子和句型 ：

①设有文法 G[Z]，如果有 $Z \Rightarrow x$，则符号串 x 为文法 G[Z] 的句型（即符号串 x 是由识别符号 Z 推导出来的）

②仅有终结符号组成的句型称为句子。

12. ①短语 ：语法树字数的末端结点从左到右排列所组成的符号串即是语法树所描述句型（相对于子树的根）的短语。

②直接短语 ：能直接推出叶子节点的根所在的短语

③句柄： 最左简单子树的末端结点符号串即是语法树所描述句型的句柄。

13. 正规集 ：设有字母 $\sum$ ，$\sum$ 上的正规表达式及其描述的符号串的集合即为正规集。

7.可生成带有两个不同分析树的串的文法称作 二义性文法 。

14. 建立 LL（1）语法分析器， 提左因子和消除左递归的目的 ：

①消除左递归是为消除死循环， LL（1）文法不能处理含有左递归的文法；

②提左因子只是推后产生式的选择决定，等到获取足够多的输入再做选择，避免程序回溯。

15. LL （1）文法需满足的条件 ：

① 在 每 个 产 生 式 $A \to \alpha_1 | \alpha_2 | ... | \alpha_n$ 中 ， 对 于 所 有 的 i 和 j ：$1 \le i, j \le n, i \ne j, First(\alpha_i) \cap First(\alpha_j)$ 为空。

②若对于每个非终结符 A 都有 First （A）包含了 ε，那么 $First(A) \cap Follow(A)$ 为空。

16. 自顶向下分析和自底向上分析

自下而上的分析与自上而下的过程相反。 不是根据产生式规则取代扩展到后继的非终结符，而是每次自底向上归约当前字符串或者句型来预测下一个合法的符号， 直至到达骑士的非终结符，该过程称为一系列的归约。