编译原理

前言

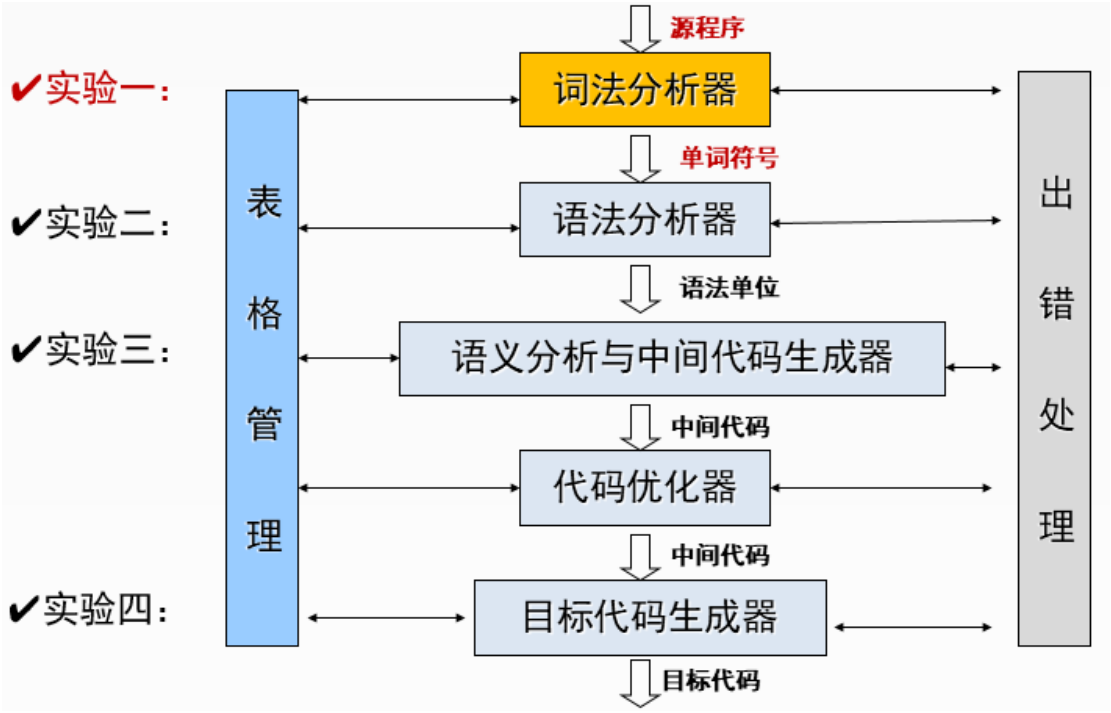
本学期的编译原理实验总共有 4 次实验，全部为设计型实验。

实验一：词法分析器的实现；

实验二：自底向上的语法分析—LR（1）；

实验三：典型语句的语义分析及中间代码生成；

实验四：目标代码生成。



实验一　词法分析器的实现

1.1 实验题目

手工设计类 C 语言的词法分析器（可以是 c 语言的子集）

1.2实验目的

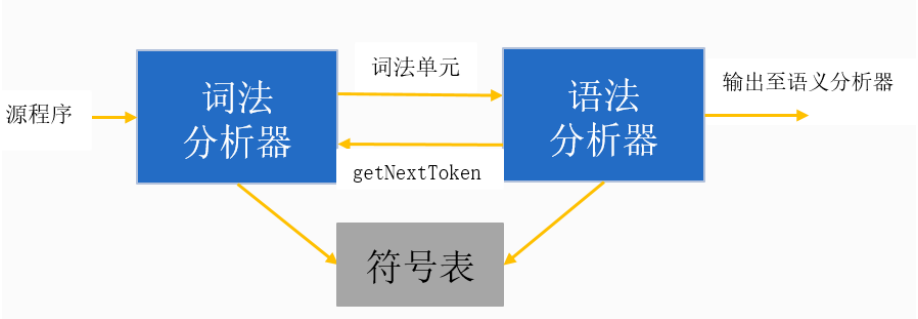
1．加深对词法分析程序的功能及实现方法的理解；
2．对类 C 语言的文法描述有更深的认识，理解有穷自动机、编码表和符号表在编译的整个过程中的应用；
3．设计并编程实现一个词法分析程序，对类 C 语言源程序段进行词法分析，加深对高级语言的认识。
4. 实验学时数：2 学时。

1.3实验内容

编写一个词法分析程序，读取代码文件，对文件内的类 C 语言程序段进行词法分析。 处理 C 语言源程序，过滤掉无用符号，分解出正确的单词，以二元组形式存输出放在文件中。
1．词法分析程序输入：以文件形式存放的类 C 语言程序段;
2．词法分析程序输出：以文件形式存放的 Token 串和简单符号表；

1.4 分析与设计

要手工设计词法分析器，实现 C 语言子集的识别，就要明白什么是词法分析器，它的功能是什么。



词法分析是编译程序进行编译时第一个要进行的任务，主要是对源程序进行

编译预处理（去除注释、无用的回车换行找到包含的文件等）之后，对整个源程序进行分解，分解成一个个单词。这些单词只有五类，分别是标识符、保留字、常数、运算符、界符。可以说词法分析面向的对象是单个字符，目的是把它们组成有效的单词（字符串），从而作为语法分析的输入来分析是否符合语法规则，并且进行语法制导的语义分析产生中间代码，进而优化并生成目标代码。

综上：词法分析器的输入是源程序字符流，输出是单词序列。

1．什么是关键字（也是保留字）？

关键字是又程序语言定义的具有固定意义的标识符。例如 C 语言中的 if，else，for，while 都是保留字，这些字通常不用作一般标识符。

2．什么是标识符？

标识符用来表示各种名字，如变量名，数组名，过程名等。

3．什么是常数？

常数的类型一般是有整型，实型、布尔型、文字型等。

4．什么是运算符？

运算符如+、-、*、/等等。

5. 什么是界符？

界符如逗号、分号、括号等等。

1.4词法分析程序测试

输入程序代码：

input_code.txt

输出示例：

Token 串：

token.txt

符号表：

symbol_table.txt

1.5 实验总体步骤

1.5.1 创建编码表

词法分析器的输出是单词序列，在 5 种单词种类中，关键字、运算符、分界符都是程序设计语言预先定义的，其数量是固定的。而标识符、常数则是由程序设计人员根据具体的需要按照程序设计语言的规定自行定义的，其数量可以是无穷多个。编译程序为了处理方便，通常需要按照一定的方式对单词进行分类和编码，在此基础上，将单词表示成二元组的形式（类别编码，单词值）。

单词名称	类别编码	单词值
int	1	–
return	2	–
=	3	–
,	4	–
Semicolon	5	–
+	6	–
–	7	–
*	8	–
/	9	–
(	10	–
)	11	–
id	51	内部字符串
IntConst	52	整数值
……	……	……
布尔常数	80	0 或 1
字符串常数	81	内部字符串

1.5.2 创建类 C 语言文法

多数程序语言单词的词法都能用正则文法来描述，基于单词的这种形式化描述会给词法分析器的设计与实现带来很大的方便，支持词法分析器的自动构造，比如 Flex、ANTLR 词法分析器生成工具。

1．正则文法表示

G=(V,T,P,S)，其中 V={S,A,B,Cdigit,no_0_digit,,char},T={任意符号}，P 定义如下

约定：用 digit 表示数字：0,1,2,…,9; no_0_digit 表示数字：1,2,…,9;

用 letter 表示字母：A,B,…,Z,a,b,…,z,_

标识符：S→letter A　　A→letter A|digit A|ε

运算符、分隔符：S->B　B->=|*|+|-|/|(|)|;

整常数：S ->no_0_digitB　　B -> digitB | ε

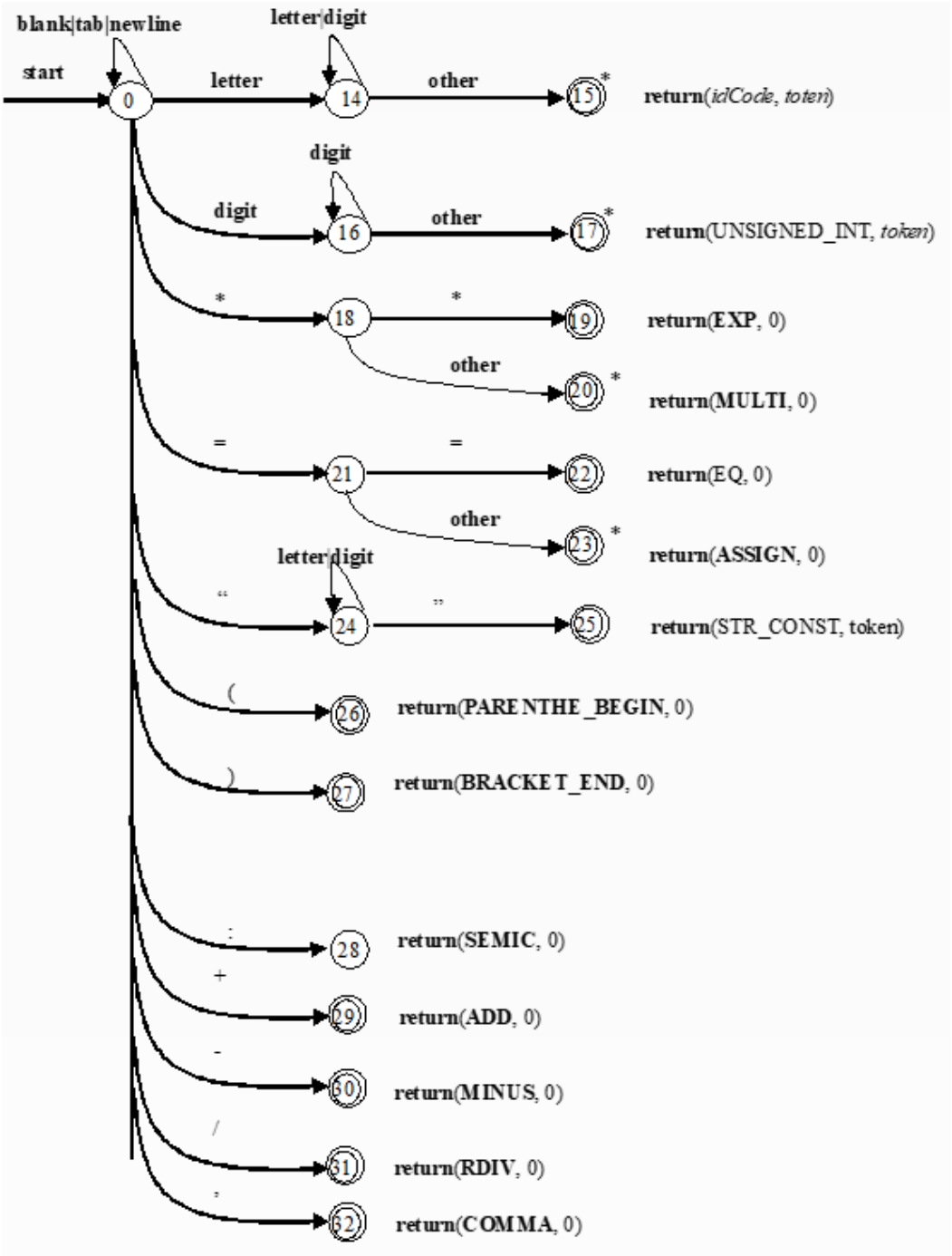
字符串常量：S -> “C”

字符常量：S->’D’

1.5.3 有限自动机

有限自动机识别的语言称为正则语言，有限自动机分为确定的有限自动机（DFA）和非确定的有限自动机（NFA）两种。DFA 和 NFA 都可以描述正则语言，DFA 规定只能有一个开始符号，且转移标记不能为空，代码实现较为方便，所以词法分析器使用 DFA 描述词法记号。

识别各类单词的状态转化图合并图：



1.5.4 编程实现

1．符号表

编译器用符号表来记录、收集和查找出现在源程序中的各种名字及其语义信息。每当源程序中出现一个新的名字时，则向符号表中填入一个新的表项来记录该名字；当在源程序中发现某个名字的新属性时，要找到该名字所对应的符号表表项，在表中记录其新发现的属性信息。因此，从数据结构的角度来看，符号表

是一个以名字为关键字来记录其信息的数据结构，支持插入和查找两个基本操作。在词法分析阶段主要是往符号表里插入名字信息。

符号表的组织结构可以是线性表或者散列表。首先设计者可能会想到选择数组作为符号表的具体物理实现结构，在符号表的规模较小的情况下，线性结构可以满足。当设计的编译器规模较大时，可以考虑引入散列表来提高查找插入的效率。

2．定义数据结构

数据结构在框架代码已定义，请熟读框架代码；

3．创建关键字列表（关键字可以写入文件中，读文件读入到存储的数据结构）；

4．把源程序代码读入到输入缓冲区，利用指针移动读入下一个字符，根据有限自动机的状态转移识别一个 Token：

1）判断 Token 是标识符，查询关键字列表，判断该 Token 是否是关键字，若是：则返回关键字的种别码和关键字本身；若不是：把该 Token 插入到符号表中，返回 Token 的种别码和 Token 本身；

2）判断 Token 是常量，返回 Token 对应常量的种别码和 Token 本身；

3）判断 Token 是运算符，返回 Token 对应运算符类型的种别码和 Token 本身；

4）判断 Token 是分界符，返回 Token 对应分界符类型的种别码和 Token 本身；