**编译原理课程实验报告**

**实验1：词法分析**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 张恒 | | 院系 | | 计算机 | | | 学号 | | | 1160300620 | | |
| 任课教师 | | 辛明影 | | | | 指导教师 | 辛明影 | | | | | | |
| 实验地点 | | 格物208 | | | | 实验时间 | 2019.4.14 | | | | | | |
| 实验课表现 | | 出勤、表现得分 | |  | | 实验报告  得分 |  | | 实验总分 | | |  | |
| 操作结果得分 | |  | |
| **一、需求分析** | | | | | | | | | | 得分 | | |  |
| 设计并实现一个词法分析器，能够输入的源程序，输出单词符号。即把构成源程序的字符串转换成等价的单词（记号）序列。具体来说，包括以下方面:   1. 能够以文件方式输入源程序； 2. 能够展示分词分析的结果，并且能够保存这个结果，用于语法分析阶段；词法分析器的输入和输出的格式如图1-1所示；     图1-1 词法分析器的输入与输出实例   1. 界面需要美观、人性化，具有良好的演示效果； 2. 能够根据词法规则识别以及组合单词。这里的单词包括五类，分别为关键字，也称为基本字；标识符，由用户定义，表示各种名字；常数，整常数、实常数、布尔常数等；运算符，即算数运算符、逻辑运算符、和关系运算符；分界符，逗号、句号、括号等； 3. 对数字常数完成数字字符串到二进制数值的转换； 4. 查填符号表。在词法分析阶段，主要是构建符号表，这里不要求输出完整的符号表，能够生成符号表供下一阶段使用即可； 5. 删去空格字符和注释； 6. 错误检查。词法分析阶段的错误检查主要是检查词法错误，即非法的字符。本词法分析器要求做简单的不封闭错误检查； 7. 关于词法分析器输出的token序列，对于标识符对应的属性值应当是符号表中该标识符的入口地址，在本词法分析器中暂不要求分配具体地址； 8. 词法分析器应当基于DFA技术； | | | | | | | | | | | | | |
| **二、文法设计** | | | | | | | | | | 得分 | | |  |
| 1. 各类单词的词法规则     词法规则主要包括六个部分，即关键字、标识符、常数、运算符、界符和注释。词法规则可以用正则文法描述如下：   |  |  | | --- | --- | | 规则 | 说明 | | Skeyword|indentifier|arithmetic|logistic|const|note| delimiter|compare | 词法的总规则，由8类词组成 | | keywordint|float|bool|if|else|do|while | 关键词组成规则，包含在标识符组成规则中，实现时，采用优先识别关键字的方式来区分这两个情况 | | lettera|b|c|…|z|A|B|C|…|Z  digit0|1|2|3|4|5|6|7|8|9  identifier(letter|\_)(letter|digit|\_)\* | 标识符组成规则， | | arithmetic+|-|\*|/|= | 算数运算符和赋值号规则 | | logisticand|or|not | 逻辑运算符组成，包含在标识符组成规则中，实现时，采用优先识别关键字的方式来区分这两个情况 | | compare>|>=|<|<=|== | 比较运算符的词法规则 | | constconsti|constf  constidigit(digit)\*  constf=digit(digit)\*.(digit)\* | 常数的词法规则，包括正整数和浮点数 | | delimiter,|;|(|)|{|}|: | 界符的词法规则 | | note/\*…\*/ | 这个表达式中，表示由两个“/\*”个中间的任何字符组成的字串都是注释 |   可以看到，在上表中，关键字和逻辑运算符都是标识符中的特殊部分，这个部分被称为保留字。在具体实现的时候，采用优先判断是否为保留字，来区分保留字和普通的标识符。   1. 各类单词的编码表  |  |  |  |  | | --- | --- | --- | --- | | 单词 | 种别码 | 标记符 | 属性值 | | 错误字符 | 1 | ERROR | 错误的字符 | | int | 2 | INT | - | | float | 3 | FLOAT | - | | bool | 4 | BOOL | - | | record | 5 | RECORD | - | | if | 6 | IF | - | | else | 7 | ELSE | - | | do | 8 | DO | - | | while | 9 | WHILE | - | | + | 10 | ADD | - | | - | 11 | SUB | - | | \* | 12 | MUL | - | | / | 13 | DIV | - | | <> | 14 | NE | - | | > | 15 | G | - | | < | 16 | L | - | | >= | 17 | GE | - | | <= | 18 | LE | - | | == | 19 | E | - | | and | 20 | AND | - | | or | 21 | OR | - | | not | 22 | NOT | - | | = | 23 | ASSIGN | - | | ; | 24 | SEMI | - | | , | 25 | COMMA | - | | (letter|\_)(letter|digit|letter|\_)\* | 26 | ID | 符号表相应地址 | | ( | 27 | LB | - | | ) | 28 | RB | - | | { | 29 | LCB | - | | } | 30 | RCB | - | | : | 31 | COLON | - | | /\*…\*/ | 32 | NOTE | 注释字符串 | | digit(digit)\* | 33 | CONSTI | 相应整数 | | digit(digit)\*.(digit)\* | 34 | CONSTF | 相应浮点数 | | true | 35 | TRUE | - | | false | 36 | FALSE | - |  1. 各类单词的转换图   关键字的识别比较麻烦，因为关键并没有规律可言，只能一个一个识别。  识别关键字和逻辑运算符（即保留字）的NFA如图2-1所示：    图2-1 识别保留字的NFA  识别标识符的NFA如图2-2所示：    图2-2 识别标识符的NFA  识别正整数和浮点数常数的DFA如图2-3所示：    图2-3 识别常数的NFA  识别注释的NFA如图2-4所示：    图2-4 识别注释的NFA  识别其余有效字符的NFA如图2-5所示：    图2-5 识别其余有效字符的NFA  有了NFA后，就可以很简单的得到每个转换图的DFA，比如将每个非法输入引入到一个错误状态，识别为错误字符即可。得到NFA，就可以很容易进行词法分析了。 | | | | | | | | | | | | | |
| **三、系统设计** | | | | | | | | | | 得分 | | |  |
| 1. 系统概要设计   系统每次运行可以进行多次词法分析，系统的每次词法分析的流程如图3-1所示    图3-1 系统每次词法分析的流程图  系统的数据流如图3-2所示    图3-2 程序的数据流图  系统的功能模块如图3-3所示    图3-3 系统的功能模块   1. 系统详细设计   **核心的数据结构**主要有两个，一个是Token，另一个就是缓存区Buffer。还有一个数据结构就是符号表，但是符号表在词法分析中只用到建立表，因此表项只是简单表示。具体来说，有：  LexicalToken。用以表示词法分析结果中一个次的token，有三个属性：源字符串、种别码（这里用种别码的助记符表示）以及属性值。  LexicalBuffer。词法分析器的缓存区，内部用一个char数组缓存源程序，提供两个重要的功能：读取下一个字符和回滚到上一个字符。此外，还记录符号在源文件中的位置信息。  CategoryCode。用以记录种别码和助记符的对应关系。  SignTableItem。用以表示符号表的一个表项，主要为后面语法分析等准备，暂时为空。  **主要功能函数。**  next。LexicalBuffer中的函数，用以读取下一个字符；  rollback。LexicalBuffer中的函数，用以回滚一个字符；  removeSpace。LexicalScanner的函数，用以出去连续的空格；  recognizeV。LexicalScanner的函数，用以识别保留字和标识符；  recognizeN。LexicalScanner的函数，用以识别常数；  recognizeO。LexicalScanner的函数，用以识别其他的有效字符；  **核心部分数据流程图**如图3-2所示。 | | | | | | | | | | | | | |
| **四、系统实现及结果分析** | | | | | | | | | | 得分 | | |  |
| 要求：对如下内容展开描述。   1. 系统实现过程中遇到的问题； 2. 针对某测试程序输出其词法分析结果； 3. 输出针对此测试程序对应的词法错误报告； 4. 对实验结果进行分析。   注：其中的测试样例自行产生。   1. 遇到的问题   系统的整体结构如图4-1所示，LexicalBuffer首先获取源文件并缓存，之后LexicalScanner从缓存区中获取字符并判断是否是合法的字符，并且将合法的字符串处理成token，并记录token出现的行，以便语法分析的时候报错。  在系统实现的过程中首先遇到的问题就是种别码的定义，作为一个初学者，我希望能够尽量完整的实现一个编译系统，这就要求要编译的语言不能太过复杂，否则在语义分析阶段难以完成。所以我必须选择一个尽量精简的词法，但是这个词法又要足够完整。  第二个就是系统结构的设计。我希望能够有一个清晰高效的系统结构，如果有必要，在继续写语法语义分析时可以快速修改词法实现。  第三个就是在识别注释时后，需要判断注释的开始和结束，而注释的第一个字符是“/”，与算数运算符中乘号冲突了，在注释结束时也是用连个连续的字符来标识，因此，无论是在注释的开始还是在注释的结束，都需要预取一个字符来判断，稍微有点小麻烦。  第四个就是token的表示，以及token中种别码和助记符的映射。Token的定义是由种别码和属性值对组成，但是仅有这些信息在后续分析中无法给出错误的定位，因此需要一个方式来记录位置信息。我想到两种方式，一是在token序列中插入一种表示位置信息的特殊的token序列，在每一行开始的token之前，用以标识这是新的一行；二是为每个token加上位置信息。我这里选择的是第二种方式。   1. 词法分析结果   如图4-1所示是词法分析器的界面。    图4-1 词法分析器的输入输出界面  可以选择从文件导入源程序，或者是直接粘贴到输入框。我们这里从文件中导入，并进行分析，可以得到如图4-2所示的结果。    图4-2 词法分析的结果  左边的框中会显示待分析的源程序，右边的框中就是分析结果。在token中，我新增了三种token：助记符为“ERROR”的token，用以记录非法字符；助记符为“NOTE”的token，用以记录注释；助记符为“END”的token，用以标识源程序的结束。   1. 错误报告   在词法分析阶段，能识别的错误包括注释的未封闭，以及非法字符，这些字符均可以直接定位，我在词法阶段用一个特殊的token标识出来，但是并没有直接向用户报错。我计划在语法分析阶段统一处理这些错误。这样做有一个好处，就是在一次分析中能够尽可能多的发现程序中存在的错误。  所以，token中的助记符为“ERROR”的token可以视为词法分析阶段的错误报告。   1. 结果分析   总的来说，词法分析器的表现达到了我的预期，能够正常的无词法错误的源程序，也能发现源程序中的非法字符。  当然，在初次尝试中，词法分析器也表现出了一些不正常的举动，比如说不能自动删去源代码中的tab符。这部分是由于在ppt以及指导中完全没有提到tab字符的处理。这主要还是暴露了当前词法分析器的一个问题，就是对于源程序中可能存在非法字符考虑不够完整。 | | | | | | | | | | | | | |
| 指导教师评语：  日期： | | | | | | | | | | | | | |