**《编译原理》实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | |  | | **姓名** |  |
| **实验题目** | **构造simple语言语法分析器** | | | | |
| **实验时间** |  | | **实验地点** |  | |
| **实验成绩** |  | | **实验性质** | **□验证性 □设计性 ■综合性** | |
| 教师评价：  □算法/实验过程正确；□源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | |
| 一、实验目的  🡪了解编译器中语法分析环节的作用和意义  🡪掌握语法分析的方法和步骤  🡪熟悉自底向上、自顶向下等各类语法分析方法；掌握语法树的构造方法 | | | | | |
| 二、实验项目内容  构造simple语言语法分析器  1、自选语法分析方法  2、完成simple语言语法分析器，要求输出语法分析结果，语法分析树，符号表  3、请提交实验报告，包括所识别的simple语言文法、所用分析方法的相关分析过程说明，输入输出文档说明，必要的测试用例及代码  4、自选实现语言及工具，可以使用语法分析器生成器 | | | | | |
| 三、实验过程或算法（源程序） | | | | | |
| 本实验采用LR(1)分析技术，使用C++作为编程语言实现。  语法分析程序处理的最小单位是由词法分析程序生成的的token串序列；按照定义可知，词法分析输出的token串中的每一个token实际上都是终结符；语法分析程序以这些终结符作为输入，根据LR(1)分析表不断地归约句柄，直到归约出文法的开始符号或者发现句子无法归约报错为止。   1. symbol类的定义   首先需要定义语法分析器处理的最小单元——符号（symbol）：    其中type字段指定symbol是终结符或者是非终结符；id是symbol的编号，人为定义。  对于symbol类，由于在程序中经常需要查找某一符号，届时需要判断两个符号是否是同一个符号，所以为symbol类重载“==”操作符和“!=”操作符：    程序需要用到一种数据结构——由多个symbol组成的集合；程序使用了C++的STL对象set来实现这样的集合；但是set需要对加入其中的元素进行排序，而如果symbol类不具有比较的方法则程序无法运行，所以还需要为symbol类重载一个比较运算符“<”：    为了程序调试和结果输出的便利性，本程序重载输出操作，定义了输出symbol类的对象的方法：     1. item类的定义   在基本元素symbol之上的元素是“项目”（item）。这里的项目即为LR(1)分析方法中的项目，包括产生式的左部、右部、当前的识别态和一个前看符号：  虽然程序还需要用到“产生式”这一元素，而“产生式”和“项目”这两个概念之间存在一定的差异，但是在程序的处理过程中，可以将一个“产生式”当做一个“项目”来进行处理（忽略“项目”的当前识别态和前看符号这两个信息即可）。  对“项目”（item）类进行如下的定义：    其中leftSide代表项目的左部，由于处理的文法是上下文无关文法所以左部一定是单个symbol；rightSide是项目的右部，用symbol的向量来进行表示，以vector类来实现；lookahead是项目的前看符；position表示了项目当前的识别态，加入一个项目的右部的长度为，则position合法的取值包括中的所有整数，如果，对应的项目形式为；如果，对应的项目形式为；id为项目在某一个项目集中的编号，由程序随机构造并使用。  为了方便起见，本实验中，把一个项目中表示当前识别态的“”记号称为“小圆点”（DOT）。  注意到，由于程序经常需要考察某一个项目的“小圆点”后面的那个symbol，以及这个symbol之后的symbol序列，于是相应地定义以下方法，作为一个项目向外提供这两个信息的接口：      不过，需要注意的是，如果一个项目属于“归约项目”，则该项目不存在“小圆点”之后的symbol，或者“小圆点”之后的symbol是空串；于是相应地，项目需要提供方法来反映当前项目是否是“归约项目”（方法很简单，判断“小圆点”当前的位置即可）：    在程序中，一个项目总是在某一项目集中被考虑，程序经常会搜索某一项目集中是否存在某一个项目，为了完成这样的搜索，必须为item类定义“==”比较运算符：    将两个项目相等定义为：左部、右部、当前处理位置和前看符号全部相等。其中符号之间的“==”操作符已经在symbol类中被重载。  同symbol类类似，本程序为item类重载输出符号：      为了将“小圆点”表现得更明显，使用字符串“!DOT!”来代替“小圆点”的“”表示法。   1. setOfItems类的定义   在“项目”这一元素之上的概念，是“项目”的集合——“项目集”（setOfItems）。在LR(1)的分析中，经常处理“项目集”，所以需要定义“项目集”（setOfItems）类：    使用一个item的向量vitem作为一个项目集中所有项目的容器。虽然在程序中使用的是vector类的对象作为一个项目集中所有item对象的容器，但是这并不意味着一个项目集中的项目之间存在先后顺序关系——“项目集”顾名思义，仅仅是“项目”的集合。  虽然一个项目集中的项目没有先后之分，但是程序为了处理的方便，将一个项目集中的项目都进行了编号（比如在构建ACTION表的时候，如果当前状态需要归约，可以很快地找出使用什么编号的产生式进行归约）；在使用append方法向一个项目集中加入项目的时候，程序就自动为一个项目进行编号：    其中cntItems表示一个项目集中当前有多少项目。  同symbol和item类一样，程序也经常会搜索setOfItems类的对象，比如寻找某一个项目集族中是否存在一个项目集；所以同样需要为setOfItems类重载“==”运算符：    注意到，在比较两个项目集是否相同的时候，先比较它们中所包含的项目数量是否一样多，如果不一样多则它们一定不相同；然后比较是否某一项目集中的所有项目都能够在另一项目集中找到。   1. clusterOfSets类的实现   在LR(1)分析中，有“项目集族”（clusterOfSets）的概念；在一个“项目集族”中，某一个“项目集”，可以因为“移进”一个终结符而转移到另外一个“项目集”，可以在“归约”出一个非终结符后而转移到另外一个“项目集”；所有需要对“项目集族”（clusterOfSets）进行定义：    为了配合GOTO函数的计算，在clusterOfSets类中，定义了表gotoTable和表visGoto。程序将计算过的GOTO函数值记录在表gotoTable中，避免重复计算。visGoto表记录了程序是否已经计算过某一个GOTO函数的值。  事实上，记录GOTO函数值的方法有很多。比如，可以简单地定义一个静态数组：  然后，在计算GOTO函数时，如果  则记录  但是，这样的方法存在很多缺点：首先，程序能够分配的静态数组的空间是非常有限的，当文法的规模扩大时，生成的项目集族中的项目可能有成百上千个，这时，可能设置的值就很大，很可能导致程序无法满足二维表gotoTable的空间需求；其次，即使程序能够为二维表gotoTable分配足够的空间，当文法的规模很大的时候，整个二维矩阵会变得相当“稀疏”——二维表中只有很小比例的空间存储了GOTO函数的信息——这样一来空间的浪费会很严重。  为了避免这样的问题，本程序将gotoTable和visGoto定义为：    如果  则记录  这样一来，二维表中每一行的列数是可变的，由gotoTable[i].size()可进行计算，空间不会被浪费。  当然，这样做的坏处在于对二维表的访问会慢一些，因为每一次访问都需要遍历整行的元素：     1. analyzeManager类的实现   为了配合语法分析的进行，程序定义了一个analyzeManager类。其中实现了一些LR(1)分析所需要使用的计算过程。   1. closure方法   这个方法以一个项目集作为输入，计算出其中的项目在某一文法下所有等价项目组成的项目集（闭包）：    其中，参数中的productions是文法的所有产生式组成的项目集。  closure方法的算法参照LR(1)分析方法中闭包的算法进行即可，首先找出项目集中格式符号要求的项目：    根据“小圆点”后的symbol类型进行讨论：    根据要求构造出可能加入的产生式，并判断产生式是否已存在与项目集，如果不存在则将其加入项目集，其中用到了非终结符的first集的计算（first集的计算在后面会提到）：        然后根据项目集是否改变来决定是否继续做循环。   1. calcGoto函数   该函数即是算法中的GOTO(I, X)函数；它将按照当前项目集中的项目中小圆点后移一个指定的符号，得到一些新的项目，然后产生闭包即可：     1. calcCluster函数   该函数构造出LR(1)分析需要的项目集族。首先在项目集族C中加入初始的项目集：closure({[S’->S, $]))：    然后对于C中的每个项目集和每个symbol都计算GOTO，如果得到的项目集不属于C则加入C：       1. calcFirstSet函数     这个函数计算出的First集，会在计算闭包的时候被使用。  其中使用如下的集合来表示每一个非终结符的first集：    first[id][j]则表示编号为id的frist集中的第j个symbol。用种类为INVALID\_TYPE的symbol来表示first集的结束：     1. LR1Table类的实现   这个类的功能是生成LR1分析表和生成语法分析的语法树结构：    按照LR(1)的算法，分析表分成ACTION和GOTO两部分。  关于语法树的实现和构造，将在后文提到。  LR1Table类中定义了calcLR1Table函数来生成ACTION表和GOTO表。这两个表的生成方法很简单，由于GOTO函数已经被计算，所以只需要根据项目集族中的项目集中的项目的情况和前看符号的情况进行讨论即可：     1. LR(1)分析的主控程序   主控程序在LR1Table类中的LR1Analyze函数中被定义：    其中symbolStack和stateStack分别为符号栈和状态栈。  程序比较简单。在循环中，每次根据状态栈栈顶的元素和当前输入的元素来判断查询ACTION表，判断下一步的动作：    如果是“移入”，则为符号栈和状态栈压进相应的内容：    如果是“归约”，则从两个栈中弹出用于归约的产生式右部长度的元素，并压入新的非终结符和新的状态：      最后，再加入“接收”状态和“错误”状态：     1. 语法树的构建和展示   为了构建和展示语法树，需要在LR1Table类中定义如下的字段：    其中father字段用来表示树。利用father数组，一棵树可以这样被定义：根节点root的father映射值是自己的id值：  其余节点的father映射值是其父亲节点的id值：  假如一棵树如下所示：    按照这样的表示方法，则有：  由于在树中，每一个节点至多有一个父亲，所以这样的表示方法是成立的。  nodeNum字段表示树中的节点总数；nodeInfo字段用来存储每一个节点代表的是什么symbol。  构造语法树的过程在主控程序中，构造过程和语法分析过程是一同进行的。  定义一个树节点栈，用于辅助语法树的构造过程：    在语法分析的时候，每一次遇到“移入”操作，则增加一个新的叶子节点，并将这一节点压入节点栈中：    在遇到“归约”操作的时候，创建一个新的节点：    并从nodeStack中弹出归约式右部这么多的节点，将这些节点设置为新节点的儿子节点：    然后将新节点压入nodeStack：    在进入“接收”状态时，将栈内节点设置为根节点的子节点：    这样一来，语法树即可构建成功。  对于语法树的显示，可以使用MATLAB的treeplot函数进行：    它可以直接以本实验提到的father数组作为参数，绘制出一棵树。   1. 符号表的实现   本实验使用哈希方法来实现符号表。虽然哈希方法的空间代价很大，但是不可否认的是，用其实现的符号表的插入和查找的效率都会很高。  定义一个符号表类symbolTable：    其中，类的成员变量table即用来存放哈希函数的映射值。为了实现哈希方法，需要在类中定义一个哈希映射函数，该函数的原理是字符串哈希，以一个字符串为输入，输出的是字符串对应的哈希值：    定义表中的项的类tableItem：    其中variableName是变量名、type是变量的种类（constant、integer、char、bool、real）、scope代表的是变量的范围域。  然后，为符号表实现添加、删除和访问的方法：    由于符号表在词法分析阶段就已经建立，语法分析阶段可以利用词法分析阶段生成的最原始的符号表，由此构造一个方法，以读取词法分析阶段生成的符号表信息：    在进行语法分析的时候，需要将定义的变量（常量）加入到符号表，此时的符号表，同词法分析阶段产生的符号表相比，此阶段产生的符号表需要加入更多的信息：变量的类型、变量的范围域。符号表的构建工作在symbolTable类中的setUpTable方法中进行描述：    在此方法中， 当程序处于状态0的时候，不对符号表进行任何操作；  通过扫描输入的符号串，一旦发现词语“var”对于的终结符，则代表程序需要定义新的变量，程序由静默状态0，改变状态到预备状态1：    当state等于1的时候，程序需要将输入的符号串中，代表标识符的所有终结符进行记录，因为程序即将对这些标识符进行变量定义（例如分析语句“var a, b, c: integer”的时候，需要将标识符a、b和c先存储起来，在读取到终结符“integer”的时候，再将a、b和c对应的项加入到符号表）：    程序在处于状态1的时候不断地将看到的标识符存入vector变量“entriesReady”中；而程序读入了一个分号对应的symbol的时候，状态会转移到2,：    此时程序会将之前存储在entriesReady中的标识符加入到符号表：    其中，关于变量所处于的范围域scope的确定比较容易；由于一个范围域总是会被一对（begin，end）符号所包围，所以，在程序中，一旦读取到了begin的终结符，则增加范围域变量scope，而读取到end终结符则减少范围域变量：     1. 对于产生式中右部为空串的处理   之前的讨论没有提到如何处理文法中存在产生式右部为空串的情况。但是这样的情况是不可忽视的。例如文法规则为：  其中a、b、c为终结符，为空串。  如果不处理产生式右部为的情况，那么句子将会被程序判定为非法。但实际上句子是合法的，因为：  本程序关于空串的处理方法是：假设所有的非终结符都无法推出空串，并增加新的产生式来进行弥补。  例如在上面的例子中，本程序将假设A、B和C都无法推导出，然后在文法的定义中新增加如下的一些产生式：  这样的文法定义实际上同之前的文法定义是等价的。   1. 实验测试方法   本实验不光对simple语言文法进行了定义和测试，还对一系列较为简单的文法情况进行了定义和测试，具体的定义和测试过程将在“实验测试”部分详细讲解。  为了方便起见，本文对simple语言文法定义中的终结符和非终结符进行取了新的别名：  非终结符别名表：   |  |  | | --- | --- | | **别名** | **含义** | |  | 扩展的文法符号 | |  | 开始符号 | |  | 分程序 | |  | 常数说明 | |  | 变量说明 | |  | 过程说明 | |  | 复合句 | |  | 常数定义 | |  | 常数 | |  | 变量定义 | |  | 标识符表 | |  | 类型 | |  | 简单类型 | |  | 复合类型 | |  | 索引 | |  | 形参表 | |  | 语句表 | |  | 执行句 | |  | 简单句 | |  | 结构句 | |  | 赋值句 | |  | 调用语句 | |  | 读语句 | |  | 写语句 | |  | 表达式 | |  | 算术表达式 | |  | 布尔表达式 | |  | 字符表达式 | |  | 项 | |  | 因子 | |  | 关系符 | |  | 实参表 | |  | 输入变量表 | |  | 表达式表 | |  | if句 | |  | while句 | |  | for句 | |  | repeat句 |   终结符别名表：   |  |  |  |  | | --- | --- | --- | --- | | **别名** | **含义** | **别名** | **含义** | |  | program |  | end | |  | 标识符 |  | := | |  | constant |  | + | |  | = |  | - | |  | ; |  | \* | |  | 整数 |  | / | |  | 布尔常数 |  | < | |  | 字符常数 |  | <> | |  | 实数 |  | <= | |  | var |  | >= | |  | : |  | > | |  | , |  | = | |  | integer |  | call | |  | bool |  | read | |  | char |  | write | |  | real |  | if | |  | array |  | then | |  | [ |  | else | |  | ] |  | while | |  | of |  | do | |  | procedure |  | repeat | |  | ( |  | until | |  | ) |  | for | |  | begin |  | to |   对于文法产生式的构造见头文件“analyzeManager.h”中的函数“analyzeSimpleLanguage”： | | | | | |
| 四、实验测试 | | | | | |
| 1. 小样例测试   在测试simple语言的文法的语法分析之前，本实验首先对一组定义简单的文法规则进行语法分析测试，初步地验证程序的执行结果。  该样例文法（扩展后的）的定义如下：  首先，在主程序中定义相关的终结符和非终结符：    其中“dollar”是结束符。  然后，在主程序中建立对文法进行定义的产生式：    然后在分析程序中，对结束符的编号的定义进行修改（非常重要）：    然后程序调用各种方法，构建初始项目集、计算项目集族和LR(1)分析表：    假如要分析的句子为：cccdcd  这是一个符合语法规则的句子，因为：    在程序中输入要分析的句子，调用语法分析过程：    程序打印出的句子分析的过程如下：  程序输出的信息首先是在分析符号串时，每一次进行“移进”或者“归约”操作时，符号栈与状态栈中的内容：    然后程序输出分析结果信息：    可以看到，分析的结果是“accept”，已接受。  然后程序输出了由father数组表达的语法分析树：    调用MATLAB的treeplot函数：    语法树绘制的结果为：    程序对正确的句子进行了判断。  下面来看，如果输入的是错误的句子，程序是否能够发现。  根据文法的定义，可以知道正确的句子中有且仅有两个‘d’，第一个‘d’之前和之后都可以有无限多个‘c’，第二个‘d’必须在输入串的末尾。  那么，如果输入了含有三个‘d’的符号串ccddcd：    运行程序后，程序会发现句子的语法错误：    如果输入了虽然含有两个‘d’但是末尾不为‘d’的符号串ccdcdc：    程序同样发现了语法的错误：     1. 符号表的测试   构建好simple语言文法，先来测试一下符号表的情况，现在只测试一个程序片段，片段如下：    翻译后的symbol串如下：    然后调用符号表的构建过程：    然后，程序设计循环，接收用户的输入：    用户输入一个变量名，程序则查询符号表，如果存在则返回变量的类型和scope：    很显然，由于程序确实定义了real（type=16）变量va、vb、vc和bool变量（type=14）vd，而vd是在一个嵌套过程中定义的，所以scope为1（嵌套一次）。  而当输入了符号表中不存在的变量的名字的时候，打印出的type即为-1：     1. simple文法语法分析   语法分析器从文件“in.txt”中读取输入的符号串信息。在该文件中，每一行都是符号串中的一个符号，以一个二元组的形式来表达：(id, pos)；其中id为symbol的编号，pos在当前symbol表达的是标识符的时候起作用，表示标识符在符号表中的入口编号。  测试用的simple语言程序如下：    翻译成对语法分析器的输入后，编程如下的序列：    程序对要实现的simple语言的文法的定义说明如下：            然后，程序需要对上述的符号和生成式进行构造：      文法构建完毕后，程序可以对符号串进行分析  程序对样例程序生成的符号串的分析过程如下：  --------------------  0:  0  --------------------  1:  (TERMINAL, id = 0)  0 1  --------------------  2:  (TERMINAL, id = 0)(TERMINAL, id = 1)  0 1 3  --------------------  3:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)  0 1 3 4  --------------------  4:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)(TERMINAL, id = 1)  0 1 3 4 7  --------------------  5:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)(TERMINAL, id = 1)(TERMINAL, id = 3)  0 1 3 4 7 11  --------------------  6:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)(TERMINAL, id = 1)(TERMINAL, id = 3)(TERMINAL, id = 6)  0 1 3 4 7 11 18  --------------------  7:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)(TERMINAL, id = 1)(TERMINAL, id = 3)(NON-TERMINAL, id = 8)  0 1 3 4 7 11 21  --------------------  8:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)(TERMINAL, id = 1)(TERMINAL, id = 3)(NON-TERMINAL, id = 8)(TERMINAL, id = 4)  0 1 3 4 7 11 21 38  --------------------  9:  (TERMINAL, id = 0)(TERMINAL, id = 1)(TERMINAL, id = 2)(NON-TERMINAL, id = 7)  0 1 3 4 8  --------------------  10:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)  0 1 3 6  --------------------  11:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)  0 1 3 6 9  --------------------  12:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(TERMINAL, id = 1)  0 1 3 6 9 12  --------------------  13:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(TERMINAL, id = 1)(TERMINAL, id = 11)  0 1 3 6 9 12 22  --------------------  14:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(TERMINAL, id = 1)(TERMINAL, id = 11)(TERMINAL, id = 1)  0 1 3 6 9 12 22 12  --------------------  15:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(TERMINAL, id = 1)(TERMINAL, id = 11)(NON-TERMINAL, id = 10)  0 1 3 6 9 12 22 39  --------------------  16:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 10)  0 1 3 6 9 14  --------------------  17:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 10)(TERMINAL, id = 10)  0 1 3 6 9 14 23  --------------------  18:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 10)(TERMINAL, id = 10)(TERMINAL, id = 12)  0 1 3 6 9 14 23 40  --------------------  19:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 10)(TERMINAL, id = 10)(NON-TERMINAL, id = 12)  0 1 3 6 9 14 23 46  --------------------  20:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 10)(TERMINAL, id = 10)(NON-TERMINAL, id = 11)  0 1 3 6 9 14 23 45  --------------------  21:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 10)(TERMINAL, id = 10)(NON-TERMINAL, id = 11)(TERMINAL, id = 4)  0 1 3 6 9 14 23 45 71  --------------------  22:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(TERMINAL, id = 9)(NON-TERMINAL, id = 9)  0 1 3 6 9 13  --------------------  23:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)  0 1 3 6 10  --------------------  24:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)  0 1 3 6 10 15  --------------------  25:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)  0 1 3 6 10 15 24  --------------------  26:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)  0 1 3 6 10 15 24 48  --------------------  27:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(TERMINAL, id = 1)  0 1 3 6 10 15 24 48 72  --------------------  28:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 24 48 76  --------------------  29:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)  0 1 3 6 10 15 24 48 76 93  --------------------  30:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(TERMINAL, id = 5)  0 1 3 6 10 15 24 48 76 93 124  --------------------  31:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 24 48 76 93 125  --------------------  32:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 25)  0 1 3 6 10 15 24 48 75  --------------------  33:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 24)  0 1 3 6 10 15 24 48 74  --------------------  34:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 20)  0 1 3 6 10 15 33  --------------------  35:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 18)  0 1 3 6 10 15 31  --------------------  36:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)  0 1 3 6 10 15 30  --------------------  37:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)  0 1 3 6 10 15 30 68  --------------------  38:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)  0 1 3 6 10 15 30 68 25  --------------------  39:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)  0 1 3 6 10 15 30 68 25 49  --------------------  40:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 49 72  --------------------  41:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 30 68 25 49 78  --------------------  42:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)  0 1 3 6 10 15 30 68 25 49 78 95  --------------------  43:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 49 78 95 128  --------------------  44:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 30 68 25 49 78 95 130  --------------------  45:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)  0 1 3 6 10 15 30 68 25 49 77  --------------------  46:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)  0 1 3 6 10 15 30 68 25 49 77 94  --------------------  47:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(TERMINAL, id = 30)  0 1 3 6 10 15 30 68 25 49 77 94 126  --------------------  48:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)  0 1 3 6 10 15 30 68 25 49 77 94 127  --------------------  49:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)  0 1 3 6 10 15 30 68 25 49 77 94 127 162  --------------------  50:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 72  --------------------  51:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 78  --------------------  52:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 78 95  --------------------  53:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 78 95 128  --------------------  54:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 78 95 130  --------------------  55:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 187  --------------------  56:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)(NON-TERMINAL, id = 30)(TERMINAL, id = 21)(NON-TERMINAL, id = 25)(TERMINAL, id = 22)  0 1 3 6 10 15 30 68 25 49 77 94 127 162 187 212  --------------------  57:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)  0 1 3 6 10 15 30 68 25 50  --------------------  58:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)  0 1 3 6 10 15 30 68 25 50 79  --------------------  59:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 50 79 96  --------------------  60:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)  0 1 3 6 10 15 30 68 25 50 79 96 131  --------------------  61:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 50 79 96 131 72  --------------------  62:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 30 68 25 50 79 96 131 165  --------------------  63:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)  0 1 3 6 10 15 30 68 25 50 79 96 131 165 188  --------------------  64:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(TERMINAL, id = 1)  0 1 3 6 10 15 30 68 25 50 79 96 131 165 188 213  --------------------  65:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 28)(TERMINAL, id = 26)(NON-TERMINAL, id = 28)  0 1 3 6 10 15 30 68 25 50 79 96 131 165 188 215  --------------------  66:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 25)  0 1 3 6 10 15 30 68 25 50 79 96 131 164  --------------------  67:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(TERMINAL, id = 1)(TERMINAL, id = 25)(NON-TERMINAL, id = 24)  0 1 3 6 10 15 30 68 25 50 79 96 131 163  --------------------  68:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(NON-TERMINAL, id = 20)  0 1 3 6 10 15 30 68 25 50 79 104  --------------------  69:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(NON-TERMINAL, id = 18)  0 1 3 6 10 15 30 68 25 50 79 102  --------------------  70:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(TERMINAL, id = 39)(NON-TERMINAL, id = 26)(TERMINAL, id = 40)(NON-TERMINAL, id = 17)  0 1 3 6 10 15 30 68 25 50 79 101  --------------------  71:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(NON-TERMINAL, id = 34)  0 1 3 6 10 15 30 68 34  --------------------  72:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(NON-TERMINAL, id = 19)  0 1 3 6 10 15 30 68 32  --------------------  73:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(NON-TERMINAL, id = 17)  0 1 3 6 10 15 30 68 30  --------------------  74:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 17)(TERMINAL, id = 4)(NON-TERMINAL, id = 16)  0 1 3 6 10 15 30 68 89  --------------------  75:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 16)  0 1 3 6 10 15 29  --------------------  76:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(TERMINAL, id = 23)(NON-TERMINAL, id = 16)(TERMINAL, id = 24)  0 1 3 6 10 15 29 67  --------------------  77:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 3)(NON-TERMINAL, id = 4)(NON-TERMINAL, id = 6)  0 1 3 6 10 16  --------------------  78:  (TERMINAL, id = 0)(TERMINAL, id = 1)(NON-TERMINAL, id = 2)  0 1 3 5  --------------------  79:  (NON-TERMINAL, id = 1)  0 2  --------------------  accept  可以看出，最终程序认为输入的符号串符合文法的语法规则：    当然，假如删除输入符号串中的某一个，使其变成不正确的句子输入，则程序会发现这样的错误：    上面的正确样例中，最终生成的由father数组表达的语法树的信息为：  79 79 10 9 9 7 9 9 10 78 23 16 16 15 16 22 22 19 20 22 22 23 78 77 34 34 28 32 32 31 32 33 34 35 36 75 75 71 57 41 45 45 44 45 57 57 48 57 57 51 55 55 54 55 57 57 71 71 68 68 62 66 66 65 66 67 68 69 70 71 72 73 74 75 77 77 78 79 0  下面对符号表进行测试，调用符号表的构建函数：    然后，接收用户的字符串输入。用户可以输入变量名，然后程序就会查找符号表，并返回变量的类型和scope：    程序定义过整形变量va和vb，所以符号表查找到了对应于va和vb两个变量的项，并返回其类型“13”（integer）和范围域0（没有嵌套）。  MATLAB绘图：    绘图结果：  如下图所示： | | | | | |

