中山大学数据科学与计算机学院本科生实验报告

课程名称：编译器构造实验 任课教师：陈炬桦

|  |  |  |  |
| --- | --- | --- | --- |
| 学年度 | 2018-2019 | 学期 | 第二学期 |
| 年级 | 16级 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337341 | 姓名 | 朱志儒 |
| 电话 | 15989184223 | Email | 739741104@qq.com |
| 开始日期 | 2019/4/12 | 完成日期 | 2019/4/12 |

1. 实验题目

**LL(1)语法分析程序**

* 1. Description

输入开始符号，非终结符，终结符，产生式，LL(1)分析表  
输出LL(1)分析表

* 1. Input

 输入开始符号；  
非终结符个数，非终结符，空格符分隔；  
终结符个数，终结符，空格符分隔；  
产生式的个数,各产生式的序号,产生式的左边和右边符号,空格符分隔；  
LL(1)分析表中的产生式个数，序号，行符号，列符号，产生式编号，空格符分隔；  
输入一个算术式符号串，用#结束

* 1. Output

 输出推导过程,每一步一行,中间“ & ”前是已经识别的子串,后是栈中信息

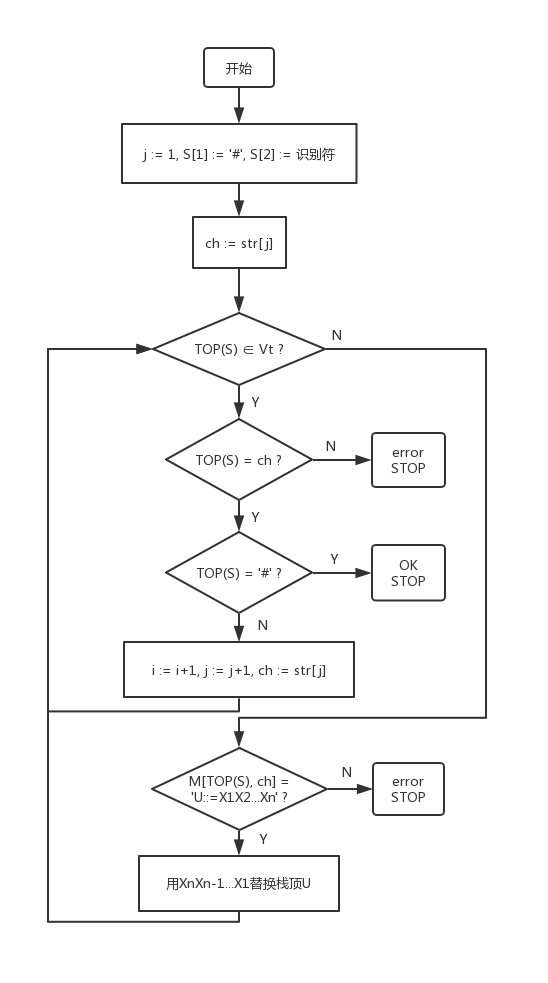
1. 算法描述（介绍程序模块功能，流程图）

算法：

待分析的符号串存放在数组str中，下标指针为j，在数组str中，输入符号串末尾符号为“#”。输入符号串的当前符号存放在字符变量ch中。在分析过程中，每次要判断TOP(S)（栈顶元素）是否为终结符（“#”也视为终结符）。

1. 如果，TOP(S)与ch相匹配，且均是“#”，则表明待分析符号串是一个句子，算法终止；
2. 如果ch与TOP(S)相匹配，但不是“#”，则表示输入符号串的当前符号是合法的，此时，再读输入符号串的下一个符号，同时S栈退栈；
3. 如果ch不是终结符而是非终结符，则需查分析表M。
4. 如果M[TOP(S)，ch]为出错标志或空白，则表示待分析符号串不是一个句子；
5. 如果M[TOP(S)，ch]是一条产生式，则倒置该产生式右部，并用它图替换S栈栈顶元素。

流程图：



1. 测试数据（2组）

|  |  |  |
| --- | --- | --- |
|  | 输入 | 输出 |
| 第一组 | E  6 E A T B F D  9 + - \* / ( ) x y z  13  1 E TA  2 A +TA  3 A -TA  4 A k  5 T FB  6 B \*FB  7 B /FB  8 B k  9 F (E)  10 F D  11 D x  12 D y  13 D z  25  1 E ( 1  2 E x 1  3 E y 1  4 E z 1  5 A + 2  6 A - 3  7 A ) 4  8 A # 4  9 T ( 5  10 T x 5  11 T y 5  12 T z 5  13 B + 8  14 B - 8  15 B \* 6  16 B / 7  17 B ) 8  18 B # 8  19 F ( 9  20 F x 10  21 F y 10  22 F z 10  23 D x 11  24 D y 12  25 D z 13  (x+(y-x\*z)\*(y+x\*z))+x/z# | # & E#  # & TA#  # & FBA#  # & (E)BA#  #( & E)BA#  #( & TA)BA#  #( & FBA)BA#  #( & DBA)BA#  #( & xBA)BA#  #(x & BA)BA#  #(x & A)BA#  #(x & +TA)BA#  #(x+ & TA)BA#  #(x+ & FBA)BA#  #(x+ & (E)BA)BA#  #(x+( & E)BA)BA#  #(x+( & TA)BA)BA#  #(x+( & FBA)BA)BA#  #(x+( & DBA)BA)BA#  #(x+( & yBA)BA)BA#  #(x+(y & BA)BA)BA#  #(x+(y & A)BA)BA#  #(x+(y & -TA)BA)BA#  #(x+(y- & TA)BA)BA#  #(x+(y- & FBA)BA)BA#  #(x+(y- & DBA)BA)BA#  #(x+(y- & xBA)BA)BA#  #(x+(y-x & BA)BA)BA#  #(x+(y-x & \*FBA)BA)BA#  #(x+(y-x\* & FBA)BA)BA#  #(x+(y-x\* & DBA)BA)BA#  #(x+(y-x\* & zBA)BA)BA#  #(x+(y-x\*z & BA)BA)BA#  #(x+(y-x\*z & A)BA)BA#  #(x+(y-x\*z & )BA)BA#  #(x+(y-x\*z) & BA)BA#  #(x+(y-x\*z) & \*FBA)BA#  #(x+(y-x\*z)\* & FBA)BA#  #(x+(y-x\*z)\* & (E)BA)BA#  #(x+(y-x\*z)\*( & E)BA)BA#  #(x+(y-x\*z)\*( & TA)BA)BA#  #(x+(y-x\*z)\*( & FBA)BA)BA#  #(x+(y-x\*z)\*( & DBA)BA)BA#  #(x+(y-x\*z)\*( & yBA)BA)BA#  #(x+(y-x\*z)\*(y & BA)BA)BA#  #(x+(y-x\*z)\*(y & A)BA)BA#  #(x+(y-x\*z)\*(y & +TA)BA)BA#  #(x+(y-x\*z)\*(y+ & TA)BA)BA#  #(x+(y-x\*z)\*(y+ & FBA)BA)BA#  #(x+(y-x\*z)\*(y+ & DBA)BA)BA#  #(x+(y-x\*z)\*(y+ & xBA)BA)BA#  #(x+(y-x\*z)\*(y+x & BA)BA)BA#  #(x+(y-x\*z)\*(y+x & \*FBA)BA)BA#  #(x+(y-x\*z)\*(y+x\* & FBA)BA)BA#  #(x+(y-x\*z)\*(y+x\* & DBA)BA)BA#  #(x+(y-x\*z)\*(y+x\* & zBA)BA)BA#  #(x+(y-x\*z)\*(y+x\*z & BA)BA)BA#  #(x+(y-x\*z)\*(y+x\*z & A)BA)BA#  #(x+(y-x\*z)\*(y+x\*z & )BA)BA#  #(x+(y-x\*z)\*(y+x\*z) & BA)BA#  #(x+(y-x\*z)\*(y+x\*z) & A)BA#  #(x+(y-x\*z)\*(y+x\*z) & )BA#  #(x+(y-x\*z)\*(y+x\*z)) & BA#  #(x+(y-x\*z)\*(y+x\*z)) & A#  #(x+(y-x\*z)\*(y+x\*z)) & +TA#  #(x+(y-x\*z)\*(y+x\*z))+ & TA#  #(x+(y-x\*z)\*(y+x\*z))+ & FBA#  #(x+(y-x\*z)\*(y+x\*z))+ & DBA#  #(x+(y-x\*z)\*(y+x\*z))+ & xBA#  #(x+(y-x\*z)\*(y+x\*z))+x & BA#  #(x+(y-x\*z)\*(y+x\*z))+x & /FBA#  #(x+(y-x\*z)\*(y+x\*z))+x/ & FBA#  #(x+(y-x\*z)\*(y+x\*z))+x/ & DBA#  #(x+(y-x\*z)\*(y+x\*z))+x/ & zBA#  #(x+(y-x\*z)\*(y+x\*z))+x/z & BA#  #(x+(y-x\*z)\*(y+x\*z))+x/z & A#  #(x+(y-x\*z)\*(y+x\*z))+x/z & # |
| 第二组 | E  6 E A T B F D  9 + - \* / ( ) x y z  13  1 E TA  2 A +TA  3 A -TA  4 A k  5 T FB  6 B \*FB  7 B /FB  8 B k  9 F (E)  10 F D  11 D x  12 D y  13 D z  25  1 E ( 1  2 E x 1  3 E y 1  4 E z 1  5 A + 2  6 A - 3  7 A ) 4  8 A # 4  9 T ( 5  10 T x 5  11 T y 5  12 T z 5  13 B + 8  14 B - 8  15 B \* 6  16 B / 7  17 B ) 8  18 B # 8  19 F ( 9  20 F x 10  21 F y 10  22 F z 10  23 D x 11  24 D y 12  25 D z 13  (x+y)\*z+x# | # & E#  # & TA#  # & FBA#  # & (E)BA#  #( & E)BA#  #( & TA)BA#  #( & FBA)BA#  #( & DBA)BA#  #( & xBA)BA#  #(x & BA)BA#  #(x & A)BA#  #(x & +TA)BA#  #(x+ & TA)BA#  #(x+ & FBA)BA#  #(x+ & DBA)BA#  #(x+ & yBA)BA#  #(x+y & BA)BA#  #(x+y & A)BA#  #(x+y & )BA#  #(x+y) & BA#  #(x+y) & \*FBA#  #(x+y)\* & FBA#  #(x+y)\* & DBA#  #(x+y)\* & zBA#  #(x+y)\*z & BA#  #(x+y)\*z & A#  #(x+y)\*z & +TA#  #(x+y)\*z+ & TA#  #(x+y)\*z+ & FBA#  #(x+y)\*z+ & DBA#  #(x+y)\*z+ & xBA#  #(x+y)\*z+x & BA#  #(x+y)\*z+x & A#  #(x+y)\*z+x & # |

1. 程序清单
2. #include <iostream>
3. #include <string>
4. **using** **namespace** std;
6. **struct** VN {
7. **int** nv;
8. **char** vn[100];
9. } vns;
11. **struct** VT {
12. **int** nt;
13. **char** vt[100];
14. } vts;
16. **struct** PS {
17. **int** np;
18. **char** pl[100];
19. string pr[100];
20. } ps;
22. **struct** LLtable {
23. **int** size;
24. **char** row[100];
25. **char** col[100];
26. **int** No[100];
27. } lltable;
29. string s;
31. **int** main() {
32. cin >> s >> vns.nv;
33. **for** (**int** i = 0; i < vns.nv; ++i)
34. cin >> vns.vn[i];
35. cin >> vts.nt;
36. **for** (**int** i = 0; i < vts.nt; ++i)
37. cin >> vts.vt[i];
38. vts.vt[vts.nt++] = '#';
39. cin >> ps.np;
40. **int** no;
41. **for** (**int** i = 0; i < ps.np; ++i)
42. cin >> no >> ps.pl[i] >> ps.pr[i];
43. cin >> lltable.size;
44. **for** (**int** i = 0; i < lltable.size; ++i)
45. cin >> no >> lltable.row[i] >> lltable.col[i] >> lltable.No[i];
46. string expression;
47. cin >> expression;
48. string states = s;
49. string syn;
50. **for** (**int** i = 0; i < expression.size(); ++i) {
51. **while** (expression[i] != states[0]) {
52. **if** (expression[i] == '#' && states.empty()) {
53. cout << '#' << syn << " & " << states << '#' << endl;
54. **return** 0;
55. }
56. **if** (states[0] == 'k') {
57. states.erase(0, 1);
58. **continue**;
59. }
60. cout << '#' << syn << " & " << states << '#' << endl;
61. **int** index = -1;
62. **for** (**int** x = 0; x < lltable.size; ++x)
63. **if** (lltable.row[x] == states[0] && lltable.col[x] == expression[i])
64. index = lltable.No[x] - 1;
65. states.erase(0, 1);
66. states = ps.pr[index] + states;
67. }
68. cout << '#' << syn << " & " << states << '#' << endl;
69. syn = expression.substr(0, i + 1);
70. states.erase(0, 1);
71. }
72. }