**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 教学助理（TA）：

|  |  |  |  |
| --- | --- | --- | --- |
| 学年度 | 2018-2019 | 学期 | 第二学期 |
| 年级 | 2016级 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337327 | 姓名 | 郑映雪 |
| 电话 | 18956073450 | Email | 646352420@qq.com |
| 开始日期 | 2019.5.17 | 完成日期 | 2019.5.17 |

**实验题目**

## 1. Description

输入开始符号，非终结符，终结符，产生式

输出LR(k)优先分析过程

以拓广算术表达式G[A]: 为例

A→E

E→E+T | T

T→T\*F | F

F→(E) | a

## 2. Input

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；

状态数，ACTION列数，GOTO列数，空格符分隔；

状态，ACTION矩阵(k 0 表示空 A 0 表示接收)，GOTO矩阵(0表示 空)，空格符分隔；

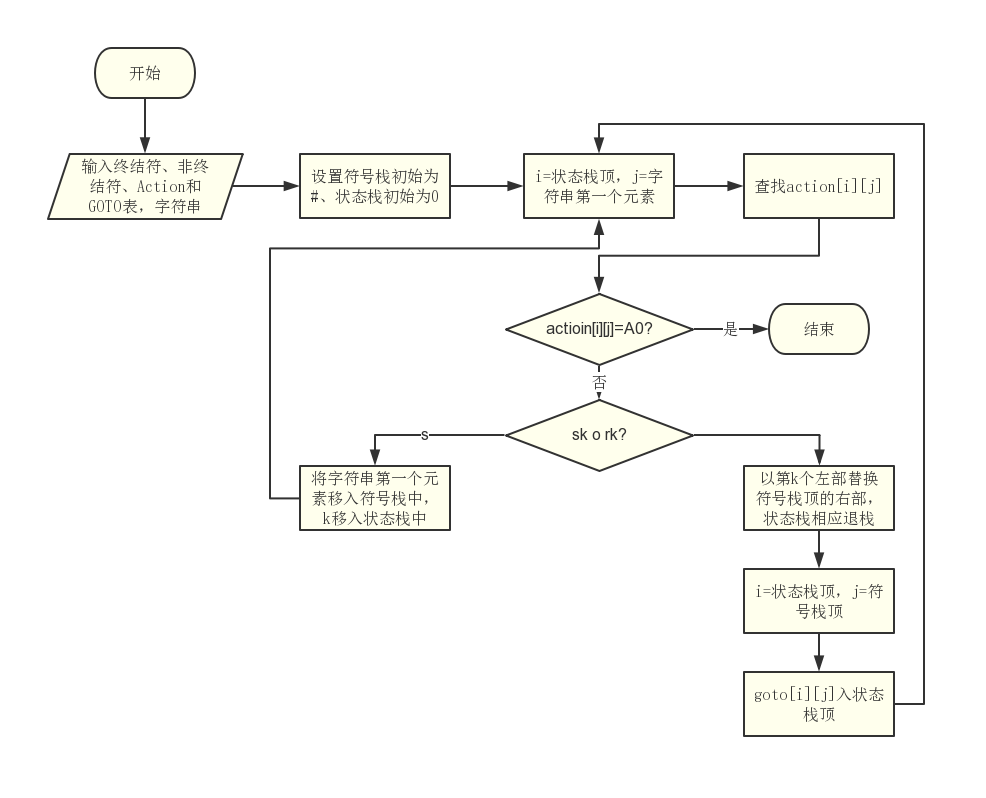
输入分析字符串，#结束

## 3. Output

用“ & ”分隔，左边表示栈底到栈顶符号；右边表示尚未分析的字符串。

**算法描述**

## 流程总览



## 2.各模块功能

**a.查表模块**

分析程序需要查找两个表——Action表和Goto表。Action是在判断字符串最左部状态转移和归约时使用的，Goto表是在归约操作之后进行下一步状态查询时使用的。查表的操作比较简单，只要添加两个findindex函数（根据终结符或非终结符匹配纵坐标）即可。

**b.查到action为sk时**

如果action为sk，说明需要转到下一个状态，这时候就将当前的符号压入符号栈，将状态压入状态栈，循环进行查表操作。

**c.查到action为rk时**

如果action为rk，说明应该进行归约操作。利用第k个左部代替其右部（此时应该在符号栈顶），并将相应的状态退栈。然后根据状态栈顶元素和刚刚替换的左部元素，查找goto表，将新的状态入状态栈。

**d.查到A0时**

如果查到A0，则代表accepted，此时分析结束，退出循环。

**e.查到k0时**

如果查到k0，则说明有语法错误。

**f．输出模块。**

为了使程序的逻辑更加清晰，我使用了两个栈分别存储符号和状态，但在输出时是不方便的。注意到题中要求的输出是每次完全处理完后输出，则说明此时两个栈长度应该是相等的。所以直接循环一遍，两个栈交替输出即可达到题目要求效果。

**测试数据(2组)**

|  |  |  |
| --- | --- | --- |
| 测试数据 | 测试结果 | 结果判定 |
| A  4 E T F A  6 a + \* ( ) #  7  0 A E  1 E E+T  2 E T  3 T T\*F  4 T F  5 F (E)  6 F a  12 6 3  0 s 5 k 0 k 0 s 4 k 0 k 0 1 2 3  1 k 0 s 6 k 0 k 0 k 0 A 0 0 0 0  2 k 0 r 2 s 7 k 0 r 2 r 2 0 0 0  3 k 0 r 4 r 4 k 0 r 4 r 4 0 0 0  4 s 5 k 0 k 0 s 4 k 0 k 0 8 2 3  5 k 0 r 6 r 6 k 0 r 6 r 6 0 0 0  6 s 5 k 0 k 0 s 4 k 0 k 0 0 9 3  7 s 5 k 0 k 0 s 4 k 0 k 0 0 0 10  8 k 0 s 6 k 0 k 0 s 11 k 0 0 0 0  9 k 0 r 1 s 7 k 0 r 1 r 1 0 0 0  10 k 0 r 3 r 3 k 0 r 3 r 3 0 0 0  11 k 0 r 5 r 5 k 0 r 5 r 5 0 0 0  (a+a)\*a# | (a+a)\*a#  #0 & (a+a)\*a#  #0(4 & a+a)\*a#  #0(4a5 & +a)\*a#  #0(4F3 & +a)\*a#  #0(4T2 & +a)\*a#  #0(4E8 & +a)\*a#  #0(4E8+6 & a)\*a#  #0(4E8+6a5 & )\*a#  #0(4E8+6F3 & )\*a#  #0(4E8+6T9 & )\*a#  #0(4E8 & )\*a#  #0(4E8)11 & \*a#  #0F3 & \*a#  #0T2 & \*a#  #0T2\*7 & a#  #0T2\*7a5 & #  #0T2\*7F10 & #  #0T2 & #  #0E1 & # | 正确 |
| S  2 S B  3 a b #  3  0 S BB  1 B aB  2 B b  7 3 2  0 s 3 s 4 k 0 1 2  1 k 0 k 0 A 0 0 0  2 s 3 s 4 k 0 0 5  3 s 3 s 4 k 0 0 6  4 r 2 r 2 r 2 0 0  5 r 0 r 0 r 0 0 0  6 r 1 r 1 r 1 0 0  bab# （数据来源：哈工大编译原理课件） | bab#  #0 & bab#  #0b4 & ab#  #0B2 & ab#  #0B2a3 & b#  #0B2a3b4 & #  #0B2a3B6 & #  #0B2B5 & #  #0S1 & # | 正确 |

**程序清单**

#include <iostream>

#include <stack>

#include <string>

#include <vector>

using *namespace* std;

*struct* { int Nv; string VN[10]; } Vns = { 0 };

*struct* { int Nt; string VT[10]; } Vts = { 0 };

*struct* { int Np; string PL[20], PR[20]; } ps = { 0 };

string action[100][100];

*int* gotom[100][100];

string S,input;

vector <*char*> str;

vector <*int*> num;

*int* findindexaction(*char* *a*) {

    for (*int* i = 0; i < Vts.Nt; ++i)

        if (a == Vts.VT[i][0]) return i;

    return -1;

}

*int* findindexgoto(*char* *a*) {

    for (*int* i = 0; i < Vns.Nv; ++i)

        if (a == Vns.VN[i][0]) return i;

    return -1;

}

*int* main() {

    cin >> S;

    cin >> Vns.Nv;

    for (*int* i = 0; i < Vns.Nv; ++i)

        cin >> Vns.VN[i];

    cin >> Vts.Nt;

    for (*int* i = 0; i < Vts.Nt; ++i)

        cin >> Vts.VT[i];

    cin >> ps.Np;

*int* tmp;

    for (*int* i = 0; i < ps.Np; ++i)

        cin >> tmp >> ps.PL[i] >> ps.PR[i];

*int* n, na, ng;

    cin >> n >> na >> ng;

    string tmp1, tmp2;

    for (*int* i = 0; i < n; i++) {

        cin >> tmp;

        for (*int* j = 0; j < na; ++j) {

            cin >> tmp1 >> tmp2;

            action[i][j] = tmp1 + tmp2;

        }

        for (*int* k = 0; k < ng; k++)

            cin >> gotom[i][k];

    }

    cin >> input;

    str.push\_back('#');

    num.push\_back(0);

*int* flag = 0;

    while (true) {

        for (*int* i = 0; i < str.size(); ++i) {

            cout << str[i] << num[i];

        }

        cout << " & ";

        for (*int* i = 0; i < input.size(); ++i)

            cout << input[i];

        cout << endl;

*int* index = findindexaction(input[0]);

        string ele = action[num.back()][index];

        if (ele == "A0") break;

        if (ele[0] == 's') {

            str.push\_back(input[0]);

            input = input.substr(1);

            if (ele.size()==2)

                num.push\_back(ele[1]-'0');

            else {

                num.push\_back((ele[1] - '0') \* 10 + (ele[2] - '0'));

            }

        }

        else {

*int* tmpindex;

            if (ele.size() == 2)

                tmpindex = ele[1] - '0';

            else {

                tmpindex = (ele[1] - '0') \* 10 + (ele[2] - '0');

            }

*int* length = ps.PR[tmpindex].size();

            for (*int* i = 0; i < length; ++i) {

                num.pop\_back();

                str.pop\_back();

            }

            str.push\_back(ps.PL[tmpindex][0]);

*int* tmpgoto = gotom[num.back()][findindexgoto(str.back())];

            num.push\_back(tmpgoto);

        }

    }

}