**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 | 2019.4.14 | 完成日期 | 2019.4.14 |

**实验题目**

## 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

## 2. 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

## 3. 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。

**算法描述**

## 1. 流程总览

## 2.各模块功能

本次实验较为简单，是理解在有预测分析表的情况下进行非递归预测的过程，主要是理解辅助栈的过程。主要的模块如下：

**a.初始模块**

初始设置终结符集、非终结符集、表达式集和预测分析表。先根据预测分析表中的表达式集进行填表（也是题1000的内容）。

**b.预测分析模块**

这是算法的核心模块。主要的算法流程如下:

**①将结束符#、开始符号入栈，指针指向算术式符号串第一个元素。**

**②对于栈顶符号，在预测分析表里搜索到以该栈顶符号为行，指针指向的算术式符号为列的表达式，用该表达式右部替换栈顶元素。**

**③判断栈顶元素是否与指针指向的元素相同，如相同，则指针指向表达式下一个元素，栈顶元素出栈，加入已识别子串中，重复第2步；如不相同，重复第②步。**

**④如果栈顶元素为#且指针指向字符串最后一个元素#，则预测结束。**

**c.输出模块**

本次实验要求的输出部分比较简单。只要输出每一次循环的已识别串和栈中元素即可。已识别串可以在每一次循环时输出，如果有栈顶元素出栈且在终结符表中，则已识别串添加该元素。栈中元素在每一次循环时输出即可。

**d.算法总结**

本实验题目是单独进行非递归预测的运算，所以预测表直接给了我们。但是在正常的分析中，预测表推出是比较麻烦的。FIRST、SELECT、ALLOW集合的推理到预测分析表的填写再到非递归预测的过程是步步连接的，可以说是一步错步步错，如果在平时做题的时候或者是考试的时候，要注意这是一个完整的过程，要足够细心进行各步骤的操作。

**测试数据(2组)**

|  |  |  |
| --- | --- | --- |
| 测试数据 | 测试结果 | 结果判定 |
| 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 & # | 正确 |
| 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# | # & E#  # & TA#  # & FBA#  # & DBA#  # & xBA#  #x & BA#  #x & A#  #x & +TA#  #x+ & TA#  #x+ & FBA#  #x+ & DBA#  #x+ & yBA#  #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 & # | 正确 |

**程序清单**

#include<iostream>

#include <string>

#include<stack>

using namespace std;

string S;

struct {

    int num;

    string vnsign[100];

    string LLresult[100][100];

}VN = { 0 };

struct {

    int num;

    string vtsign[100];

}VT = { 0 };

struct {

    int num;

    int order[100];

    string left[100], right[100];

}pro = { 0 };

struct {

    int num;

    int order[100];

    int pronum[100];

    string row[100], col[100];

}LL = { 0 };

void printresult(string left, stack<string> stackresult) {

    string str = "";

    cout << "#" << left << " & ";

    while (!stackresult.empty()) {

        cout << stackresult.top();

        str += stackresult.top();

        stackresult.pop();

    }

    cout << "#" << endl;

    while (str.length() != 0) {

        string str2 = str.substr(str.length() - 1, 1);

        stackresult.push(str2);

        str = str.substr(0, str.length() - 1);

    }

    return;

}

int main() {

    cin >> S >> VN.num;

    for (int i = 0; i < VN.num; i++)

        cin >> VN.vnsign[i];

    cin >> VT.num;

    for (int i = 0; i < VT.num; i++)

        cin >> VT.vtsign[i];

    cin >> pro.num;

    for (int i = 0; i < pro.num; i++)

        cin >> pro.order[i] >> pro.left[i] >> pro.right[i];

    cin >> LL.num;

    for (int i = 0; i < LL.num; ++i)

        cin >> LL.order[i] >> LL.row[i] >> LL.col[i] >> LL.pronum[i];

    string ts;

    cin >> ts;

    stack<string> stackresult;

    stackresult.push(S);

    string left = "";

    printresult(left, stackresult);

    while(!stackresult.empty()) {

        string strtop = stackresult.top(),fl=ts.substr(0,1);

        if (strtop == fl) {

            left += strtop;

            ts = ts.substr(1, ts.length() - 1);

            stackresult.pop();

            printresult(left, stackresult);

            continue;

        }

        else {

            for (int i = 0; i < LL.num; ++i) {

                if (LL.row[i] == strtop && LL.col[i] == fl) {

                    stackresult.pop();

                    string str = pro.right[LL.pronum[i] - 1];

                    if (str == "k") continue;

                    while (str.length() != 0) {

                        string str2 = str.substr(str.length() - 1, 1);

                        stackresult.push(str2);

                        str = str.substr(0, str.length() - 1);

                    }

                }

            }

            printresult(left, stackresult);

        }

    }

}