编译原理语法分析器实验报告

学院： 计算机科学与工程

班级: 200402

姓名: 夏新宇

学号: 20203644

实验二：语法分析

1. 实验题目

**根据给定文法编写调试预测分析程序，对任意输入串用预测分 析法进行语法分析。**

1. 实验目的

**加深对预测分析法的理解。**

1. 实验内容

**对LL(1)文法**

**E→TEˊ**

**Eˊ→+TEˊ|ε**

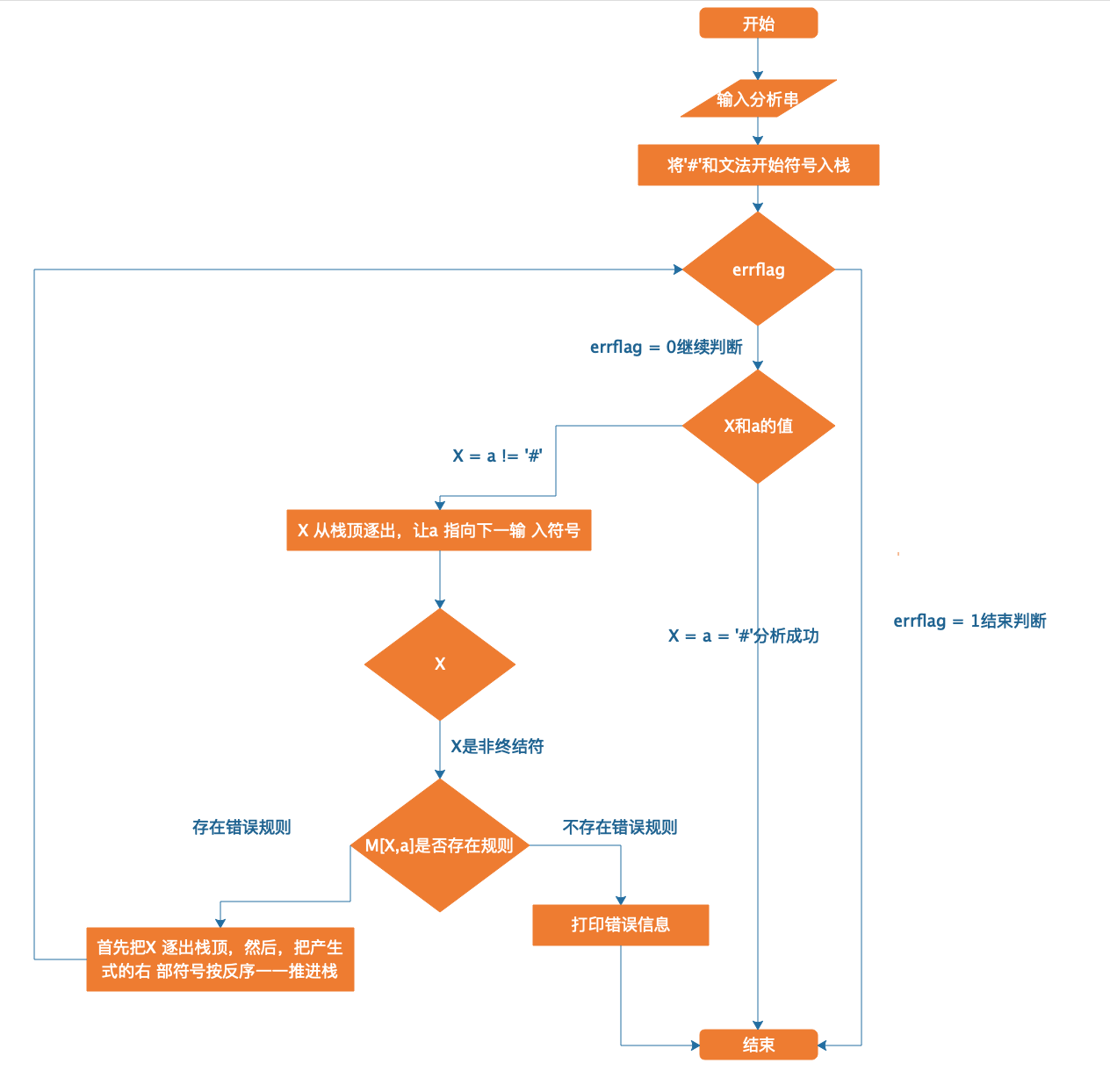
**T→FTˊ**

**Tˊ→\*FTˊ|ε**

**F→(E)|i**

**根据程序所给的预测分析表采用预测分析法实现对输入串进行语法分析。**

**算法分析流程图：**

****

**程序代码：**

#include <iostream>

#include <stdio.h>

#include <string>

#include <stack>

using namespace std;

char Vn[] = {'E', 'e', 'T', 't', 'F'};      //定义文法的非终结符，小写字母e表示E’

char Vt[] = {'i', '+', '\*', '(', ')', '#'}; //定义文法的终结符

int LENVt = sizeof(Vt);

void showstack(stack<char> st) //显示栈中信息

{ //从栈底开始显示栈中的内容

    int i, j;  //定义i，j作为数据指针

    char ch[100]; //定义长度为100的char数组

    j = st.size(); //j初始化为栈的大小

    for (i = 0; i < j; i++)

    {

        ch[i] = st.top(); //依次获取栈顶元素并赋值给ch数组

        st.pop();         //栈顶元素出栈

    }

    for (i = j - 1; i >= 0; i--)

    {

        cout << ch[i];  //逆序打印ch数组里的的元素

        st.push(ch[i]); //元素逆序入栈

    }

}

int find(char c, char array[], int n) //在array数组中查找字符c

{

    int i; //定义i作为数组指针

    int flag = 0; //定义flag标记，flag为1时代表找到字符c

    for (i = 0; i < n; i++) //遍历array数组字符

    {

        if (c == array[i]) //判断当前i指向字符是否等于c

            flag = 1; //flag赋值为1

    }

    return flag; //返回flag的值

}

int location(char c, char array[]) //在array数组中查找字符c,若查找成功返回字符c的下标

{

    int i; //定义i作为数组指针

    for (i = 0; c != array[i]; i++) //若当前i指向数组的字符等于c，循环终止

        ;

    return i; //返回i值，代表字符c在array数组中的下标

}

void error()

{

    cout << " error!" << endl; //打印程序出错信息

}

void analyse(char Vn[], char Vt[], string M[5][6], string str)

{

    int i, j, p, q, h, flag = 1; //初始化变量定义

    char a, X; //初始化字符定义

    stack<char> st; //初始化字型型栈的定义

    st.push('#');   //‘#’入栈。

    st.push(Vn[0]); // Vn[0]入栈

    j = 0;          // j指向输入串的指针

    h = 1;

    a = str[j]; //将输入字符串的第一个值赋给a

    cout << "步骤 "

         << "分析栈 "

         << "剩余输入串"

         << "所用产生式 " << endl;

    while (flag == 1) //flag等于1时循环终止

    {

        cout << h << " "; //显示步骤

        h++; //h 自增

        showstack(st); //显示分析栈中内容

        cout << " ";

        for (i = j; i < str.size(); i++)

            cout << str[i]; //打印剩余字符串

        X = st.top(); //将栈顶元素赋值给X

        if (find(X, Vt, LENVt) == 1) //判断是否在Vt数组中找到字符X

            if (X == a) //分析栈的栈顶元素和剩余输入串的第一个元素相比较

                if (X != '#') //判断X字符是否是 '#'

                {

                    cout << " " << X << "匹配" << endl; //输出提示信息

                    st.pop(); //将栈顶元素出栈

                    a = str[++j]; //读入输入串的下一个字符

                }

                else

                {

                    cout << " "

                         << "接受！" << endl //说明成功完成语法分析，打印提示信息

                         << endl;

                    flag = 0; //flag赋值为0;

                }

            else

            {

                error(); //调用error()函数，打印错误信息

                break; //结束while循环

            }

        else

        {

            p = location(X, Vn); //请填写语句实现下标的转化（非终结符转换为行下标）

            q = location(a, Vt); //请填写语句实现下标的转化（终结符转换为列下标）

            string S1("NULL"), S2("null"); //初始化两个NULL字符串

            if (M[p][q] == S1 || M[p][q] == S2)//查找二维数组中的产生式

            {

                error(); //调用error()函数，打印错误信息

                break; //对应项为空，则出错

            }

            else

            {

                string str0 = M[p][q]; //获取二维数组中的产生式

                cout << " " << X << "-->" << str0 << endl; //显示相应的产生式

                st.pop(); //栈顶元素出栈

                if (str0 != "$") //$代表“空”字符

                    for (i = str0.size() - 1; i >= 0; i--)  //请填写语句产生式右端逆序进栈

                        st.push(str0[i]); //入栈操作

            }

        }

    }

}

int main()

{

    string M[5][6] = {"Te", "NULL", "NULL", "Te", "NULL", "NULL",

                      "NULL", "+Te", "NULL", "NULL", "$", "$",

                      "Ft", "NULL", "NULL", "Ft", "NULL", "NULL",

                      "NULL", "$", "\*Ft", "NULL", "$", "$",

                      "i", "NULL", "NULL", "(E)", "NULL", "NULL"}; //预测分析表

    string str; //初始化输入字符串

    int errflag, i; //初始化循环控制变量

    cout << "文法：E->E+T|T T->T\*F|F F->(E)|i" << endl; //打印提示信息

    cout << "请输入分析串（以#结束）：" << endl;

    do

    {

        errflag = 0;  //循环控制变量赋值为0

        cin >> str; //读入分析字符串

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

            if (!find(str[i], Vt, LENVt))

            {

                cout << "输入串中包含非终结符" << str[i] << "（输入错误）！" << endl; //打印提示信息

                errflag = 1; //循环控制变量赋值为1

            }

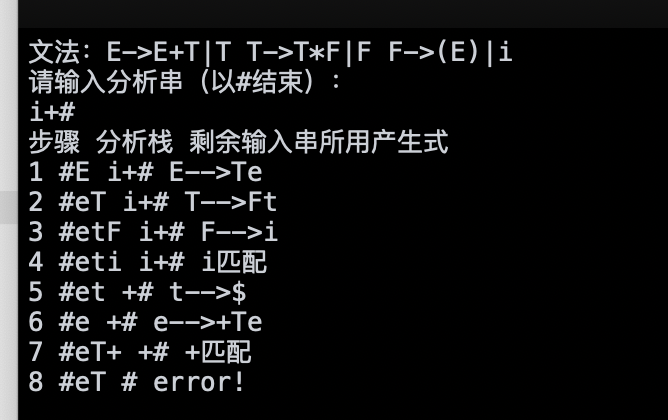
    } while (errflag == 1); //循环终止

    analyse(Vn, Vt, M, str); //调用语法分析函数

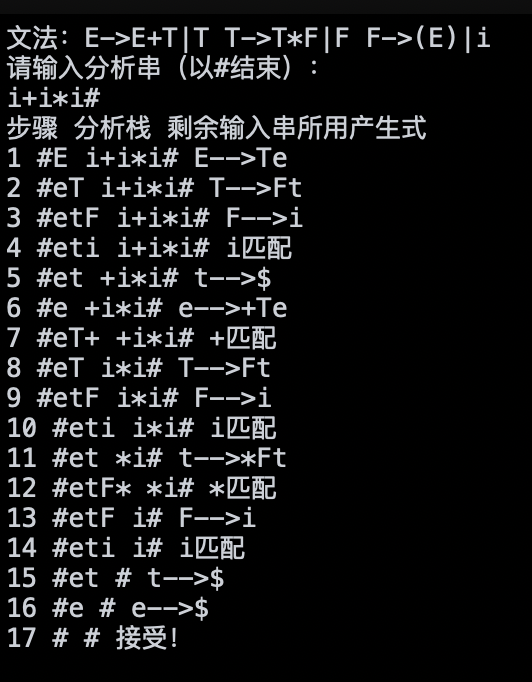
    return 0; //main函数执行完毕

}

1. 实验结果



结果1



结果2

五．实验心得

1.学习了如何根据给定文法编写调试预测分析程序，对任意输入串用预测分 析法进行语法分析。

2.对LL(1)文法有了更深刻的认识以及熟练掌握使用LL(1)文法的前提条件，以及分析过程

3.通过本次实验意识到，编程能力比较薄弱，应该继续加强