

编译原理

**递归下降分析实验报告**

**Recursiv Descent Experimental Report**

学 院： 计算机与信息技术

专 业： 计算机科学

学生姓名： 刘宜进

学 号： 14282008

指导教师： 徐金安

**北京交通大学**

2017年5月

目 录

[目 录 ii](#_Toc483421677)

[1 实验目的 3](#_Toc483421678)

[2 实验内容 3](#_Toc483421679)

[2.1程序功能描述 3](#_Toc483421680)

[2.2程序结构 3](#_Toc483421681)

[2.2.1 读取用户输入 4](#_Toc483421682)

[2.2.2 递归下降分析 4](#_Toc483421683)

[2.2.3 过程展示 4](#_Toc483421684)

[2.3数据结构 5](#_Toc483421685)

[2.4主要函数 5](#_Toc483421686)

[2.5 程序流图 6](#_Toc483421687)

[3 程序测试 7](#_Toc483421688)

[3.1测试用例 7](#_Toc483421689)

[3.2测试结果 7](#_Toc483421690)

[3.3 结果分析 8](#_Toc483421691)

[附 录 9](#_Toc483421692)

1. 实验目的

完成以下描述算术表达式的 LL(1)文法的递归下降分析程序 G[E]:

E→TE

E′→ATE′|ε

T→FT′

T′→MFT′|ε

F→ (E)|i

A→+|-

M→\*|/

1. 输入串应是词法分析的输出二元式序列，即某算术表达式“专题 1”的输出结果，输出为输入串是否为该文法定义的算术表达式的判断结果；
2. 递归下降分析 程序应能发现输入串出错；
3. 设计两个测试用例（尽可能完备，正确和出错），并给出测试结果。
4. 实验内容

该实验运用递归下降分析法的基本原理，针对以上文法描述语言，利用Python语言实现了的递归下降分析程序。下面，通过程序功能描述、程序结构、数据结构、主要函数、程序执行图等五方面展开详细介绍。

2.1程序功能描述

该程序能够持续读取用户输入，并递归下降分析，并展示函数调用过程。同时，对为了方便中间过程的展示，特意写了一个输入表格的库函数Drawtable，后面将会具体介绍。

2.2程序结构

该程序主要有三大部分组成：

1. 读取用户输入
2. 递归下降分析
3. 中间过程展示、

2.2.1 读取用户输入

利用一个While(1)循环，持续读取用户输入，直至用户输入“exit”时退出程序。首先对用户输入源串进行基本的处理，如取出空格的影响。

inputString = input("请输入语句(递归下降)：")

再将用户输入转化为列表(List)形式以方便后期分析使用，同时在列表的最后添加上一个“#”表示源串的结束。

inputString = list(inputString)

inputString.append('#')

2.2.2 递归下降分析

这是程序的核心部分，程序的入口通过调用E()启动。函数E是最顶层的分析，分别调用T()和T\_()两个函数，如果均成功，表示分析正确，否则分析错误。

同理T()再调用F()和T\_()函数进行判断，等等。如果中间任意一步的调用发生错误，则返回“分析错误”。

2.2.3 过程展示

为了方便看出在什么时候那个函数调用了那个函数，我特意写了一个方便展示结果的命令行端绘制表格工具DrawTable()。

该函数接受五个参数：

Header是字符串变量，表示表格的题目；

SubHeader也是字符串变量，是表格的副标题；

Component是二维链表，分别对应着表格的内容；

Length 是一个整数，它表示绘制表格的长度，缺省值为80；

Center 是一个布尔值，center = 1是表示居中显示，0表示左对齐显示。

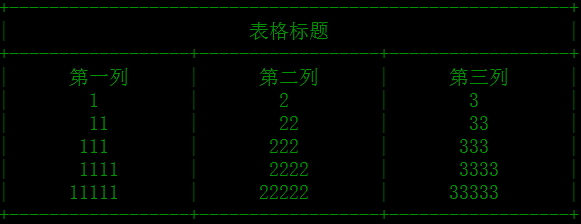
该库函数的示例如下：

图2-1 表格示例图

2.3数据结构

该函数主要是函数间调用，不涉及过多数据结构。

四个全局变量current、inputString、component、step分别表示当前字符、输入串、表格内容和分析步骤。

2.4主要函数

表2-1 主要函数及功能介绍

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数 | 返回值 | 用途 |
| Advance() | no | no | 推进一个字符 |
| A() | no | no | 判断A |
| M() | no | no | 判断M |
| F() | no | no | 判断F |
| T() | no | no | 判断T |
| T\_() | no | no | 判断T\_ |
| E() | no | no | 判断E |
| E\_() | no | no | 判断E\_ |

2.5 程序流图

该程序的执行流图如下所示，可以看出调用关系比较繁琐。

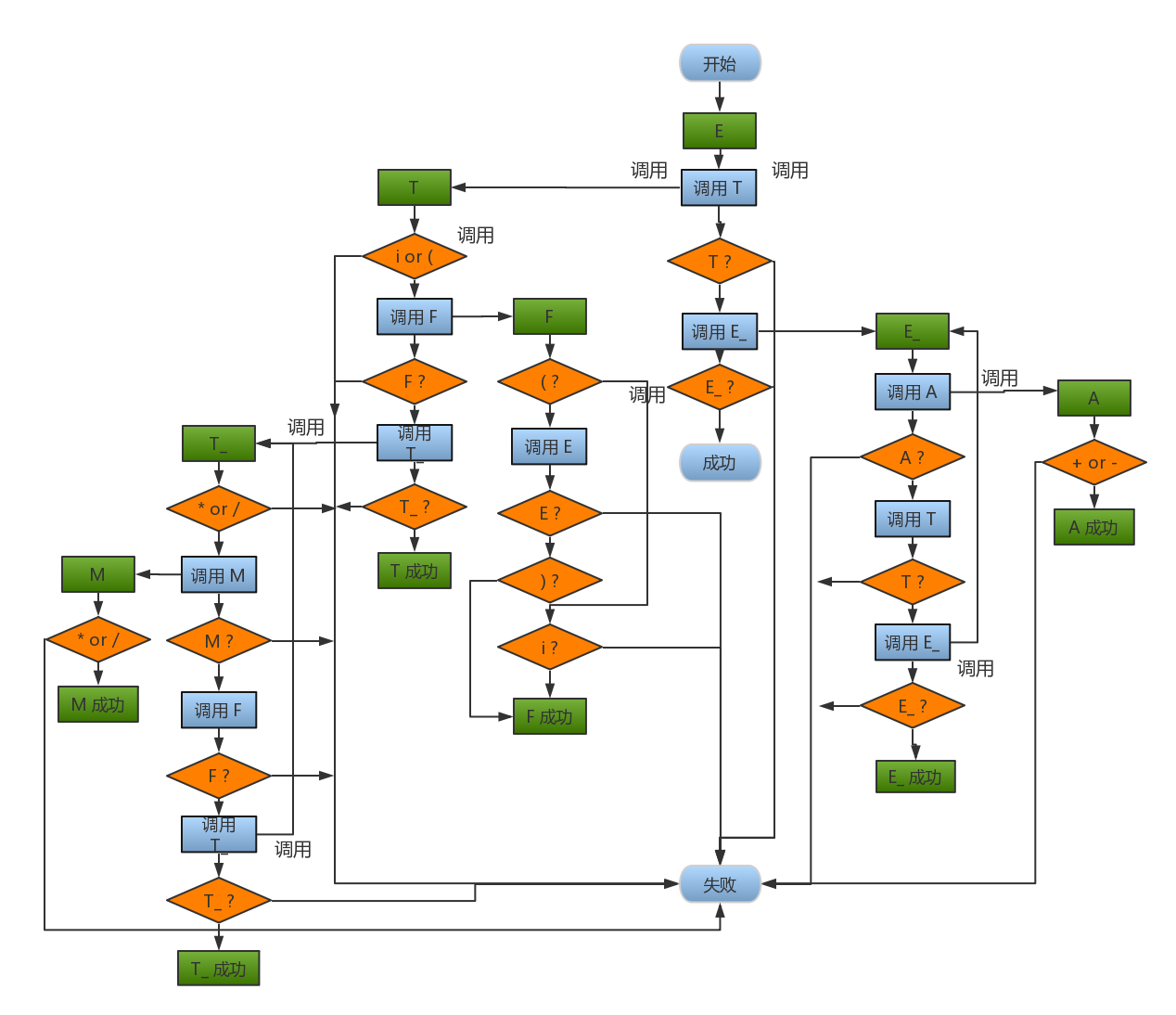


图2-2 程序执行图

1. 程序测试

3.1测试用例

测试语句采取两个比较复杂的表达式，如下：

i-i+i\*i/(i\*(i\*i)-i)

i\*(i+i/(i-i)\*i/i+(i\*i))

分别进行测试，并对测试结果进行测评。

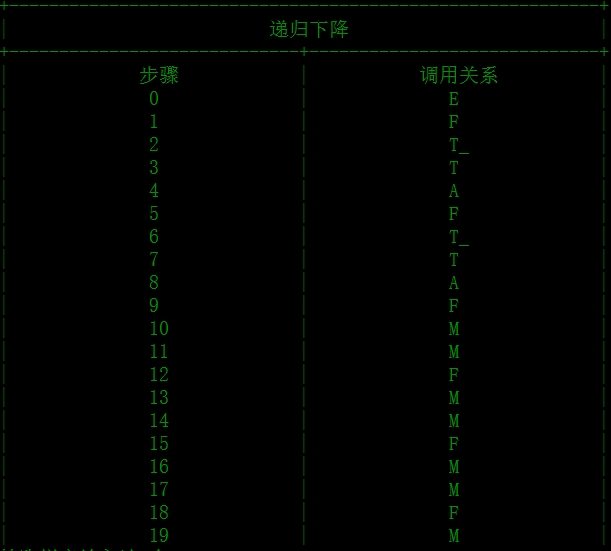
3.2测试结果

图3-1 递归下降分析结果1（部分图）

3.3 结果分析

采用以上两个测试用例分别进行测试，测试结果均显示正确，列表中第一列表示分析步骤，第二列表示调用函数关系。

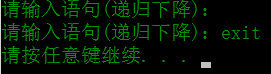
后经过多次测试验证了程序的正确性与健壮性，对于边沿性测试数据也有良好的表现，比如用户输入空串，则提醒用户继续输入。如果用户输入“exit”则退出程序。如下图所示：

图3-2 边沿数据处理图

附 录

附录A 程序代码

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

**from** drawTable **import** drawTable

**global** current

**global** inputString

**global** component

**global** step

**def** advance**():**

**global** current

current **=** inputString**.**pop**(**0**)** #pop默认退最后一个

**def** A**():**

**global** component

**global** step

**if** **(**current **==** '+' **or** current **==** '-'**):**

advance**()**

**return** **True**

**else:**

**return** **False**

**def** M**():**

**global** component

**global** step

step **=** step **+** 1

component**.**append**([**step **,** 'M'**])**

**if** **(**current **==** '\*' **or** current **==** '/'**):**

advance**()**

**return** **True**

**else:**

**return** **False**

**def** F**():** # ok

**global** component

**global** step

**if** **(**current **==**'('**):**

advance**()**

**if(**E**()):** # E

step **=** step **+** 1

component**.**append**([**step **,** 'E'**])**

**if(**current **==** ')'**):**

advance**()**

**return** **True**

**elif(**current **==**'i'**):**

advance**()**

**return** **True**

**return** **False**

**def** T\_**():**

**global** component

**global** step

**if(**current **==** '\*' **or** current **==**'/'**):**

**if(**M**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'M'**])**

**if(**F**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'F'**])**

**if(**T\_**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'T\_'**])**

# advance()

**return** **True**

**elif(**current **==**')' **or** current **==**'#' **or** current **==**'+' **or** current **==**'-'**):**

# advance() 不确定

**return** **True**

**return** **False**

**def** T**():**

**global** component

**global** step

**if(**current **==**'i' **or** current **==**'('**):**

**if** **(**F**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'F'**])**

**if(**T\_**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'T\_'**])**

**return** **True**

**return** **False**

**def** E\_**():**

**global** component

**global** step

**if(**current **==**'+' **or** current **==**'-'**):**

**if** **(**A**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'A'**])**

**if(**T**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'T'**])**

**if** **(**E\_**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'E\_'**])**

**return** **True**

**elif(**current **==**')' **or** current **==**'#'**):**

**return** **True**

**return** **False**

**def** E**():**

**global** component

**global** step

**if(**current **==**'i' **or** current **==**'('**):**

**if(**T**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'T'**])**

**if(**E\_**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'E\_'**])**

**return** **True**

step **=** step **+** 1

component**.**append**([**step **,** 'fail'**])**

**return** **False**

**def** main**():**

**global** current

**global** inputString

**global** component

**global** step

**while(**1**):**

component **=[]**

step **=** **-**1

inputString **=** input**(**"请输入语句(递归下降)："**)**

**if(**len**(**inputString**)** **==**0**):**

**continue**

**if(**inputString **==** 'exit'**):**

**break**

inputString **=** inputString**.**replace**(**' '**,**''**)**

inputString **=** list**(**inputString**)**

inputString**.**append**(**'#'**)**

current**=**inputString**.**pop**(**0**)**

step **=** step **+** 1

component**.**append**([**step **,** 'E'**])**

**if(**E**()):**

step **=** step **+** 1

component**.**append**([**step **,** 'succes'**])**

header **=** '递归下降'

subHeader **=** **[** '步骤'**,** '调用关系'**]**#

drawTable**(**header**,**subHeader**,**component**,**60**,**1**)** # 最后一个参数为总长度

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

main**()**

附录B 测试用例

i\*(i+i/(i-i)\*i/i+(i\*i))

i-i+i\*i/(i\*(i\*i)-i)

i-i+(i\*i/i\*(i\*i)-i)

i-(i\*(i\*i)-i)\*i+i

i-(i+i-i/i\*(i\*i-i))

i-i+i-(i/i\*(i\*i)-i)

i(-i+i)\*i/(i\*(i\*i)-i)