专题2\_递归下降语法分析设计原理与实现

李若森 13281132 计科1301

# 理论传授

语法分析的设计方法和实现原理；LL(1)文法及其判定；无回溯的递归下降分析的设计与实现。

# 目标任务

## 实验项目

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→\* | /

## 设计说明

终结符号i为用户定义的简单变量，即标识符的定义。加减乘除即运算符。

## 设计要求

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。

## 任务分析

重点解决LL(1)文法到递归子程序框图的设计。

# 实现过程

## 消除左递归

该文法无左递归。

## 消除回溯

该文法无回溯。

## FIRST集和FOLLOW集

根据FIRST集和FOLLOW集的定义，可得到文法G[E]的FIRST集和FOLLOW集。

如表3-1所示：

|  |  |  |
| --- | --- | --- |
| 表3-1 | | |
| 产生式 | FIRST(α) | FOLLOW(A) |
| E→TE’ | { (, i } | { ), # } |
| E’→ATE’ | { +, - } | { ), # } |
| E’→ε | { ε } |
| T→FT’ | { (, i } | { +, -, ), # } |
| T’→MFT’ | { \*, / } | { +, -, ), # } |
| T’→ε | { ε } |
| F→(E) | { ( } | { +, -, \*, /, ), # } |
| F→i | { i } |
| A→+ | { + } | { (, i } |
| A→- | { - } |
| M→\* | { \* } | { (, i } |
| M→/ | { / } |

## 函数递归框图

递归框图算法如下：

设：

* + Current 中放置当前正扫描的输入符号
  + NextSym 表示输入符号指针后移一位

假定：

* 当进入某子程序时,要分析的输入符号已经在Current中
* 在从某一子程序退出时,下一个要分析的输入符号放入Current中

根据算法可推得图3-1至图3-7：

|  |
| --- |
|  |
| 图3-1 |
|  |
| 图3-2 |
|  |
| 图3-3 |
|  |
| 图3-4 |
|  |
| 图3-5 |
|  |
| 图3-6 |
|  |
| 图3-7 |

## 主要数据结构

**pair<int, string>:**

用pair<int, string>来存储单个二元组。该对照表由专题1定义。

**vector<string>:**

将二元式序列通过ntable转化后的的字符串序列。

## 函数定义

|  |
| --- |
| **init:** |
| void init(); |
| **功能：** |
| 初始化关键字及识别码对照表 |
| **传入参数：** |
| （无） |
| **传出参数：** |
| （无） |
| **返回值：** |
| （无） |
|  |
| **Error:** |
| void Error( int &pos, string msg ); |
| **功能：** |
| 存储错误信息 |
| **传入参数：** |
| msg:错误信息 |
| **传出参数：** |
| pos:出错标识符为该行第pos个标识符 |
| **返回值：** |
| （无） |
|  |
| **ProcessorF:** |
| bool ProcessorF( int &pos ); |
| **功能：** |
| 非终结符F的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **ProcessorA:** |
| bool ProcessorA( int &pos ); |
| **功能：** |
| 非终结符A的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **ProcessorM:** |
| bool ProcessorM( int &pos ); |
| **功能：** |
| 非终结符M的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **ProcessorTQ:** |
| bool ProcessorTQ( int &pos ); |
| **功能：** |
| 非终结符T’的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **ProcessorT:** |
| bool ProcessorT( int &pos ); |
| **功能：** |
| 非终结符T的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **ProcessorEQ:** |
| bool ProcessorEQ( int &pos ); |
| **功能：** |
| 非终结符E’的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **ProcessorE:** |
| bool ProcessorE( int &pos ); |
| **功能：** |
| 非终结符E的处理过程 |
| **传入参数：** |
| pos:当前第pos个标识符 |
| **传出参数：** |
| pos:当前第pos个标识符 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
|  |
| **Parse:** |
| bool Parse( vector<PIS> &veco, int &epos, string &emsg ); |
| **功能：** |
| 进行该行的语法分析 |
| **传入参数：** |
| vec:该行字符串的二元式序列 |
| **传出参数：** |
| epos:出错标识符首字符所在位置 |
| emsg:出错信息 |
| **返回值：** |
| 是否成功解析。是则返回true，否则返回false。 |
| **errMsg:** |
| void errMsg( string filename, int rowNo, int colNo, string errmsg ); |
| **功能：** |
| 向屏幕输出错误信息 |
| **传入参数：** |
| filename:正在处理的文件的文件名称 |
| rowNo:出错行 |
| colNo:出错列 |
| errmsg:错误信息 |
| **传出参数：** |
| （无） |
| **返回值：** |
| （无） |

# 程序测试

测试用例详见文件夹中test1.lexer和test2.lexer。

其中，

test1.lexer、test2.lexer为测试输入文件。

在命令行中运行parse [name]即可运行测试用例。

test1为正确文法二元序列，test2为非法文法输入二元序列。

|  |
| --- |
|  |

# 心得体会