专题3\_LL(1)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) 分析表的构造；LL(1)分析过程；LL(1)分析器的构造。

# 目标任务

## 实验项目

实现LL(1)分析中控制程序（表驱动程序）；完成以下描述算术表达式的 LL(1)文法的LL(1)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. LL(1)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)表的构造和LL(1)分析器的实现。

# 实现过程

## 实现LL(1)分析器

1. 将#号放在输入串S的尾部

|  |
| --- |
|  |

1. S中字符顺序入栈
2. 反复执行c)，任何时候按栈顶Xm和输入ai依据分析表，执行下述三个动作之一。

|  |
| --- |
|  |
|  |
|  |

## 构造LL(1)分析表

**构造LL(1)分析表需要得到文法G[E]的FIRST集和FOLLOW集。**

|  |
| --- |
|  |

**构造FIRST(α)**

|  |
| --- |
|  |

**构造FOLLOW(A)**

|  |
| --- |
|  |

**构造LL(1)分析表算法**

|  |
| --- |
|  |

**根据上述算法可得G[E]的LL(1)分析表，如表3-1所示：**

|  |
| --- |
| **表3-1** |
|  |

## 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。

**map<string, int>:**

存储离散化后的终结符和非终结符。

**vector<string>[][]:**

存储LL(1)分析表

## 函数定义

|  |
| --- |
| **init:** |
| void init(); |
| **功能：** |
| 初始化LL(1)分析表，关键字及识别码对照表，离散化（非）终结符 |
| **传入参数：** |
| （无） |
| **传出参数：** |
| （无） |
| **返回值：** |
| （无） |
|  |
| **Parse:** |
| bool Parse( const vector<PIS> &vec, int &ncol )； |
| **功能：** |
| 进行该行的语法分析 |
| **传入参数：** |
| vec:该行二元式序列 |
| **传出参数：** |
| emsg:出错信息 |
| epos:出错标识符首字符所在位置 |
| **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 ); |
| **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 |
| **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