# 语法分析程序指导手册

## 实例（Tiny语言文法）

*program -> stmt-sequence*

*stmt-sequence ->  stmt-sequence;statement|statement*

*statement -> if-stmt|repeat-stmt;assign-stmt;read-stmt;write-stmt*

*if-stmt -> if exp then stmt- sequence end*

*| if exp then stmt- sequence else stmt-sequence end*

*repeat-stmt -> repeat stmt-sequence until exp*

*assign-stmt -> identifier := exp*

*read-stmt -> read identifier*

*write-stmt -> write exp*

*exp -> simple-exp comparison-op simple-exp | simple-exp*

*comparison-op -> <| =*

*simple-exp -> simple-exp addop term |term*

*addop -> +|-*

*term -> term mulop factor |factor*

*mulop -> \*|/*

*factor -> (exp)|number | identifier*

## 二、语法分析程序实现步骤

### 2.1将文法用EBNF表示（本实验中的语法结构已经用EBNF表示）

*program*→*stmt-sequence*

*stmt-sequence*→*statement*{ **；***statement* }

*statement*→*if-stmt | repeat-stmt | assign-stmt | read-stmt | write-stmt*

*if-stmt*→**if** *exp* **then** *stmt-sequence* **[ else** *stmt-squence* **] end**

*repeat-stmt*→**repeat** *stmt-sequence* **until** *exp*

*assign-stmt*→**identifier :=** *exp*

*read-stmt*→**read** **identifier**

*write-stmt*→**write** *exp*

*exp*→*simple-exp* **[** *comparison-op simple-exp* **]**

*comparison-op*→**<** | **=**

*simple-exp*→*term* { *addop term* }

*addop*→ **+**| **-**

*term*→*factor* { *mulop factor* }

*mulop*→ **\*** | **/**

*factor*→**(** *exp* **)** | **number** |**identifier**

### 2.2设计抽象语法树

*program*→*stmt-sequence*

*stmt-sequence→statement{ ；statement }*

*if-stmt→if exp then stmt-sequence [ else stmt-squence ] end*

*repeat-stmt*→**repeat** *stmt-sequence* **until** *exp*

*assign-stmt*→**identifier :=** *exp*

*read-stmt*→**read** **identifier**

*write-stmt*→**write** *exp*

*exp→simple-exp [ comparison-op simple-exp ]*

*comparison-op→****<*** *|* ***=***

*simple-exp→term { addop term }*

*addop→* ***+****|* ***-***

*term→factor { mulop factor }*

*mulop→* ***\**** *|* ***/***

*factor*→**(** *exp* **)** | **number** |**identifier**

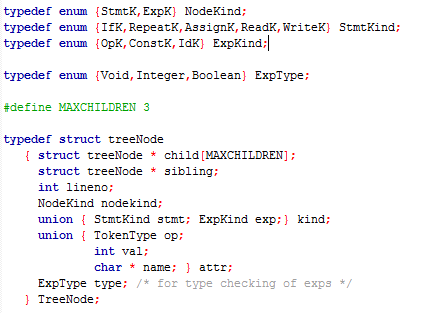
### 2.3定义数据结构

TINY有两种基本的结构类型：语句和表达式。

语句共有5类：（if语句、repeat语句、assign语句、read语句和read语句），

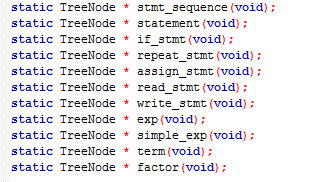
表达式共有3类（算符表达式、常量表达式和标识符表达式）。

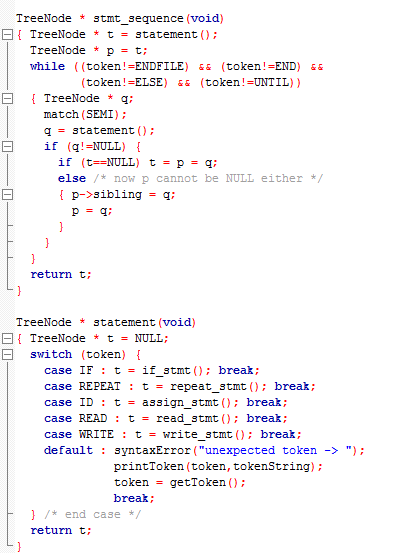
以C代码为例：

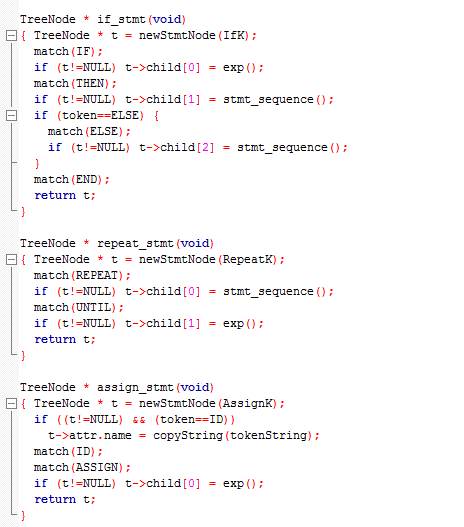


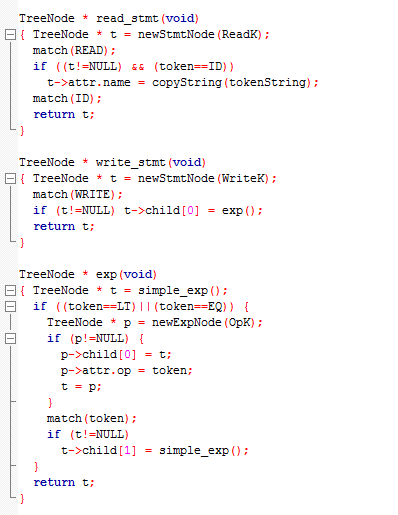
### 2.4实现递归下降分析程序

1、语法分析递归下降核心代码

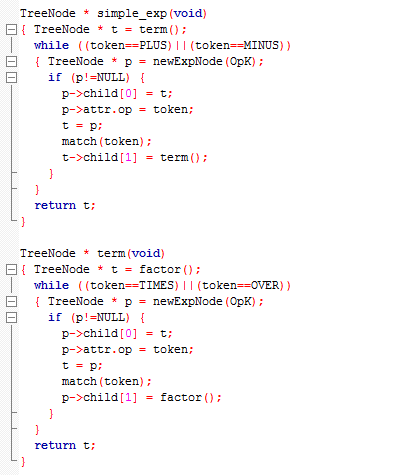


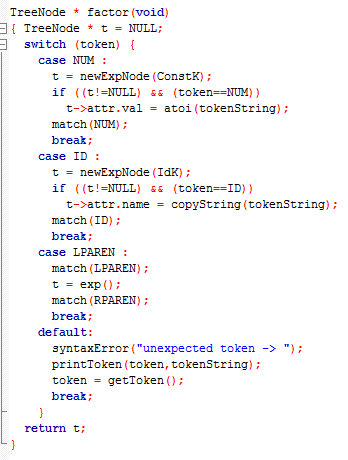






*simple-exp*→*term* { *addop term* }*3+4-5*





2、完整源码

见压缩包“源码.zip”；

### 2.5 程序输出结果参考

可以参考Pyc parser C语言语法树生成输出结果。

见ftp中文件：

1. C语言抽象语法树的生成-14011406班的薛浩
2. Pyc parserC语言语法树生成-贾钰杨

### 其他

如果用LR算法写语法分析程序，实验中语法规则可以改写为（注：曹超提供）：

BEGIN -> Thread $

identifiers -> identifier :: identifiers

identifiers -> identifier ::

associations -> association associations

associations -> association

package\_name -> identifiers identifier

reference -> package\_name :: identifier

reference -> identifier

splitter -> +=>

splitter -> =>

association -> none

association -> identifier :: identifier splitter constant access decimal

association -> identifier :: identifier splitter access decimal

association -> identifier splitter constant access decimal

association -> identifier splitter access decimal

flow\_path\_spec -> identifier : flow path identifier -> identifier ;

flow\_sink\_spec -> identifier : flow sink identifier { associations } ;

flow\_sink\_spec -> identifier : flow sink identifier ;

flow\_source\_spec -> identifier : flow source identifier { associations } ;

flow\_source\_spec -> identifier : flow source identifier ;

flow\_spec -> none ;

flow\_spec -> flow\_path\_spec ;

flow\_spec -> flow\_sink\_spec ;

flow\_spec -> flow\_source\_spec ;

IOtype -> in out ]

IOtype -> out ]

IOtype -> in ]

Parameter -> identifier : IOtype parameter reference { associations } ;

Parameter -> identifier : IOtype parameter reference ;

Parameter -> identifier : IOtype parameter ;

port\_type -> event port ]

port\_type -> event data port reference

port\_type -> event data port

port\_type -> data port reference

port\_type -> data port

port\_spec -> identifier : IOtype port\_type { associations } ;

port\_spec -> identifier : IOtype port\_type ;

feature -> none

feature -> Parameter

feature -> port\_spec

Thread -> thread identifier features feature flows flow\_spec properties association ; end identifier ;

Thread -> thread identifier features feature flows flow\_spec end identifier ;

Thread -> thread identifier features feature end identifier ;

Thread -> thread identifier end identifier ;