

## 实习二 语法分析

### 一、实验目的：

1. 掌握 JavaCC 语法分析器工作原理；
2. 设计 MiniC 的上下文无关文法，在“Parser.jjt”文件中表示该文法，生成调试递归下降分析程序，以便对任意输入的符号串进行分析；
3. 输出语法树。

### 二、MiniC 语法

```
⟨程序⟩    → int main() { ⟨语句块⟩ * }
⟨语句块⟩  → ⟨语句⟩ | { ⟨语句块⟩ * }
⟨语句⟩    → ⟨顺序语句⟩ | ⟨条件语句⟩ | ⟨循环语句⟩
⟨顺序语句⟩ → ( ⟨声明语句⟩ | ⟨赋值语句⟩ ) ";"
⟨声明语句⟩ → int ID, ID, ..., ID //思考如何表示
⟨赋值语句⟩ → ID = ⟨表达式⟩
⟨条件语句⟩ → if ( ⟨条件⟩ ) ⟨语句块⟩
⟨循环语句⟩ → while ( ⟨条件⟩ ) ⟨语句块⟩
⟨条件⟩     → ⟨表达式⟩ ⟨关系符⟩ ⟨表达式⟩
⟨表达式⟩   → // 可以使用javacc自动生成的 表达式(Expression)的文法
⟨关系符⟩   → < | <= | > | >= | == | !=
```

注意：该文法为参考文法，可以对其适当修改。首先，文件中的文法必须不含左递归和回溯；其次，同学们可以补充其它语法规则。

### 三、实习要求

1. 以文件流的形式读入要分析的 C 语言程序；
2. 如果输入的源程序符合 MiniC 的语法规则，输出语法树。

扩展要求：具有错误检查的能力，如果有能力可以输出错误所在的行号，并简单提示。

### 四、实验过程和指导：

1. 新建一个用于 javacc 编辑的 jjt 模板文件：
  - (1) 新建 java 项目
  - (2) 建立一个语法分析包（例如：package parser）

- (3) 在 parser 包内，“新建” - “其它” - “JavaCC Template File”
- (4) 创建一个“.jtt”文件，如图 1 所示，建议选用“Non static”模式：

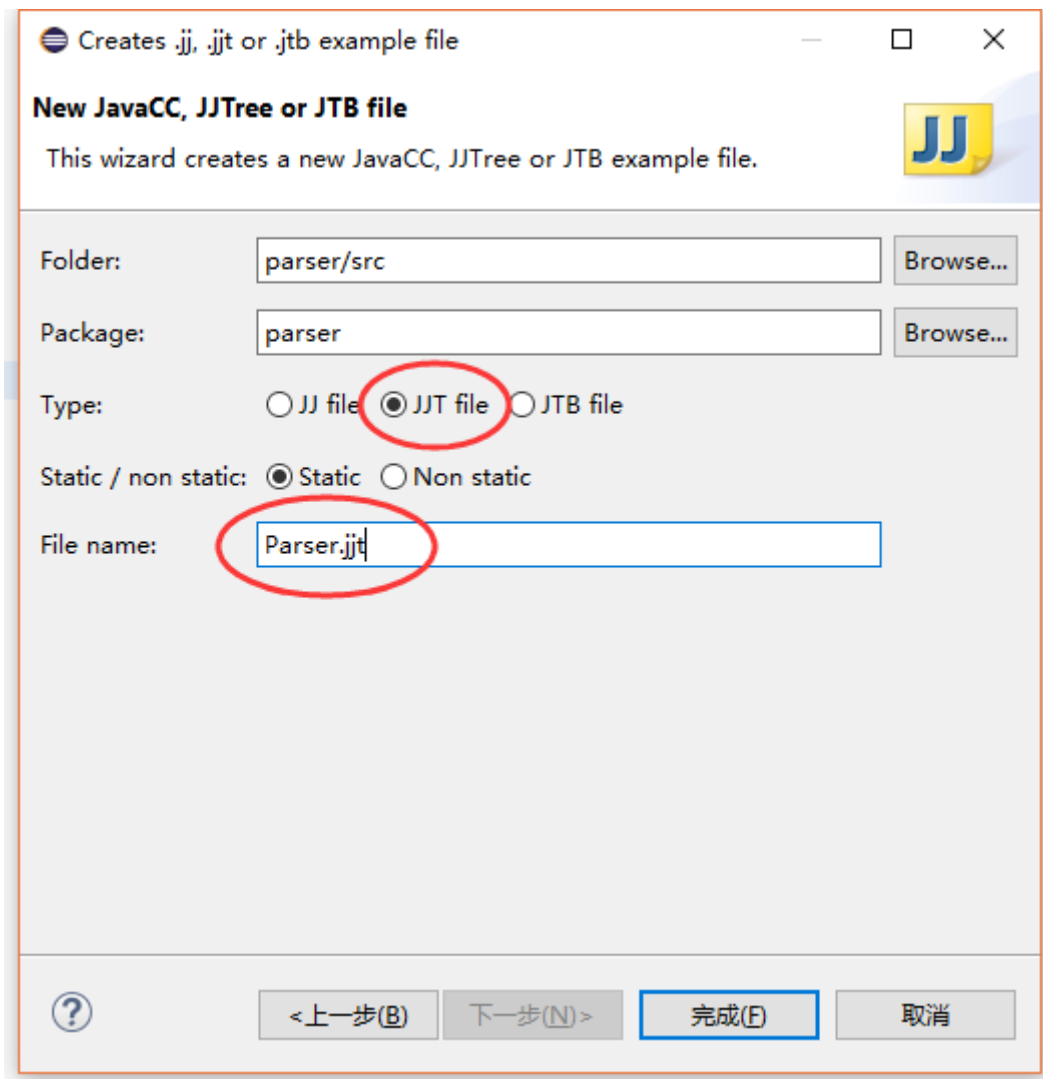


图 1 新建 jtt 模板文件

2. 在模板文件中**修改词法部分和语法部分**（如图 2 所示），将 MiniC 的文法依照要求写入 jtt 文件。
3. **修改 jtt 文件的 main 方法**（如图 2 所示），使其执行任务是：指定要进行语法分析的源程序，执行语法分析，输出语法树。

## 1.1 jjt文件的结构

/\*第一部分\*/

语法分析器的属性设置

/\*第二部分\*/

语法分析器的类声明

/\*第三部分\*/

词法规则声明

/\*第四部分\*/

语法规则实现

```
4 options
5 {
6     JDK_VERSION = "1.5"; //JDK版本
7     LOOKAHEAD = 1;       //默认, LL(K)
8     static = true;       //语法分析器是否静态
9 }
10
11 PARSER_BEGIN(AAA)
12 package jjtfile;        //可以添加一个包名
13
14 public class AAA
15 {
16     public static void main(String args []){}
17 }
18 PARSER_END(AAA)
19
20 SKIP : /*jjt自动生成, 空格和换行等*/{}
21 TOKEN : /* LITERALS */{}
22 TOKEN : /* KEY, 先于标识符声明 */{}
23 TOKEN : /* IDENTIFIERS */{}
24
25
26 SimpleNode Start() :{}
27 void Expression() :{}
28 void AdditiveExpression() :{}
29 void MultiplicativeExpression() :{}
30 void UnaryExpression() :{}
31 void Identifier() :{}
32 void Integer() :{}
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

图2 jjt 文件结构

## 五、程序输入/输出示例：

**要求：**测试多组源程序查看结果。

图3和图4为一组作为参考的测试源代码及其对应的语法树。

**注意：**文法的定义不同，语法树的层级关系则不同。

```
1 void main()
2 {
3     while(a>45){
4         int x, x1;
5         x=x1+1;
6     }
7 }
8
```

图3 测试源代码

```
<已终止> Parser ( 1 ) [Java 应用程序] C:\Program Files (x86)\Java\jre7\bin\java.exe
Start
Program
Function
type
StatementBlock
statement
loopStatement
whileStatement
BoolExpression
Expression
AdditiveExpression
MultiplicativeExpression
UnaryExpression
Identifier
Expression
AdditiveExpression
MultiplicativeExpression
UnaryExpression
Integer
StatementBlock
StatementBlock
statement
seqStatement|
decStatement
type
Identifier
Identifier
StatementBlock
statement
seqStatement
assStatement
Identifier
Expression
AdditiveExpression
MultiplicativeExpression
UnaryExpression
Identifier
MultiplicativeExpression
UnaryExpression
Integer
```

图 4 语法分析输出：语法树（参考）

实验报告提交：将实验报告和整个 project 目录打包后提交。