信 息 工 程 学 院

《编译原理》实验报告

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学院**:信息工程学院 | **班级**:软工2001班 | **姓名**:张宇晨 | **学号**:2020012249 | **成绩**: |

# 实习目标

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。

# 实验过程

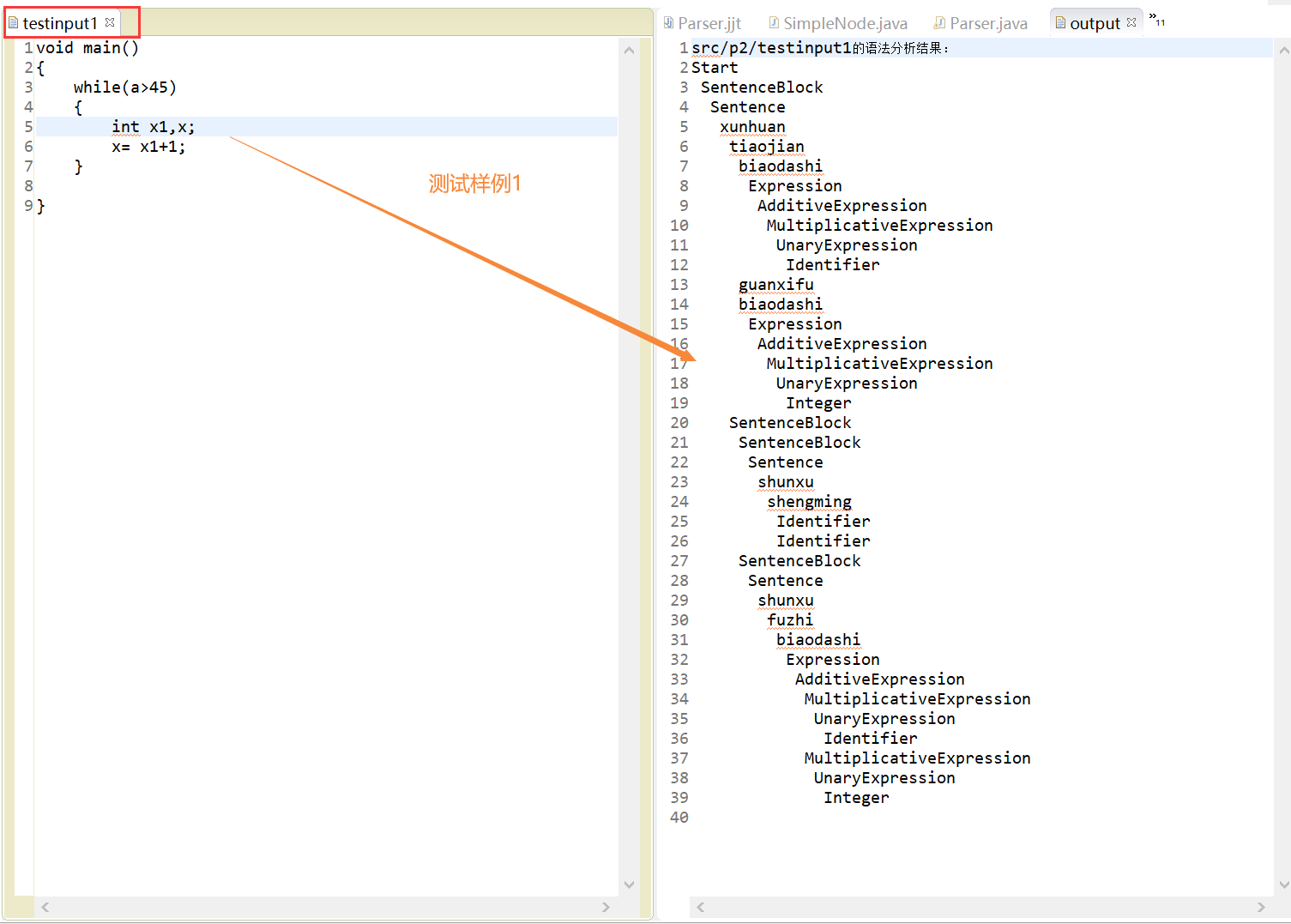
1. 词法定义
2. TOKEN : /\* LITERALS \*/
3. {
4. < INTEGER\_LITERAL :
5. < DECIMAL\_LITERAL > ([ "l", "L" ])?
6. | < HEX\_LITERAL > ([ "l", "L" ])?
7. | < OCTAL\_LITERAL > ([ "l", "L" ])?
8. >
9. | < #DECIMAL\_LITERAL : [ "1"-"9" ] ([ "0"-"9" ])\* >
10. | < #HEX\_LITERAL : "0" [ "x", "X" ] ([ "0"-"9", "a"-"f", "A"-"F" ])+ >
11. | < #OCTAL\_LITERAL : "0" ([ "0"-"7" ])\* >
12. }
14. TOKEN:/\*KERWORDS\*/
15. {
16. < IF : "if">
17. |   < ELSE : "else">
18. |   < INT : "int">
19. |   < RETURN : "return">
20. |   < VOID : "void">
21. |   < WHILE : "while">
22. | < CHAR:"char" >
23. | < FLOAT:"float" >
24. | < DOUBLE:"double" >
25. | < MAIN:"main" >
26. | < FOR:"for" >
27. }
29. TOKEN : /\* IDENTIFIERS \*/
30. {
31. < IDENTIFIER :
32. < LETTER >
33. (
34. < LETTER >
35. | < DIGIT >
36. )\* >
37. | < #LETTER : [ "\_", "a"-"z", "A"-"Z" ] >
38. | < #DIGIT : [ "0"-"9" ] >
39. }
41. TOKEN:/\*SPARATER\*/
42. {
43. < COMMA:"," >
44. | < SEM : ";">
45. |   < LFBRACKET:"{" >
46. |   < RFBRACKET:"}" >
47. |   < LBRACKET:"[" >
48. |   < RBRACKET:"]" >
49. |   < LRBRACKET:"(" >
50. |   < RRBRACKET:")" >
51. }
52. TOKEN :/\*OPERATOR\*/
53. {
54. < ADD:"+" >
55. | < MUL:"\*" >
56. | < MINUS:"-" >
57. | < DIVISOR:"/" >
58. | < LESSTHAN:"<" >
59. | < GREATERTHAN:">" >
60. | < EQUAL:"=" >
61. }
62. 语法定义
63. SimpleNode Start() :
64. {}
65. {
66. ("int"|"void")"main""("")""{"(SentenceBlock())\*"}"
67. {
68. **return** jjtThis;
69. }
70. //("int"|"void")"main""("")""{"(SentenceBlock())\*"}"
71. }
73. **void** SentenceBlock():
74. {}
75. {
76. Sentence()
77. | "{"(SentenceBlock())\*"}"
79. }
81. **void** Sentence():
82. {}
83. {
84. shunxu()|tiaojianyuju()|xunhuan()
85. }
87. **void** shunxu():
88. {}
89. {
90. (shengming()|fuzhi())";"
91. }
92. **void** shengming():
93. {}
94. {
95. ("int"|"double"|"float"|"char")Identifier()(","Identifier())\*
97. }
99. **void** fuzhi():
100. {}
101. {
102. < IDENTIFIER > "="biaodashi()
103. }
104. **void** tiaojianyuju():
105. {}
106. {
107. "if""("tiaojian()")"SentenceBlock()
108. }
109. **void** xunhuan():
110. {}
111. {
112. "while""("tiaojian()")"SentenceBlock()
113. }
114. **void** tiaojian():
115. {}
116. {
117. biaodashi()guanxifu()biaodashi()
119. }
120. **void** biaodashi():
121. { }
122. {
123. Expression()
124. }
125. **void** guanxifu():
126. {}
127. {
128. "<"|">"|"="|"<="|">="|"=="|"!="
129. }
130. **void** Expression() :
131. {}
132. {
133. AdditiveExpression()
134. }
136. **void** AdditiveExpression() :
137. {}
138. {
139. MultiplicativeExpression()
140. (
141. (
142. "+"
143. | "-"
144. )
145. MultiplicativeExpression()
146. )\*
147. }
149. **void** MultiplicativeExpression() :
150. {}
151. {
152. UnaryExpression()
153. (
154. (
155. "\*"
156. | "/"
157. | "%"
158. )
159. UnaryExpression()
160. )\*
161. }
163. **void** UnaryExpression() :
164. {}
165. {
166. "(" Expression() ")"
167. | Identifier()
168. | Integer()
169. }
171. **void** Identifier() :
172. {}
173. {
174. < IDENTIFIER >
175. }
177. **void** Integer() :
178. {}
179. {
180. < INTEGER\_LITERAL >
181. }
182. 修改main()函数实现文件读写
183. **protected** **static** JJTParserState jjtree = **new** JJTParserState();**public** **static** **void** main(String args [])
184. {
185. System.out.println("Reading from file input...");
186. String inputpath ="src/p2/testinput1";
187. String outputpath ="src/p2/output";
188. InputStream fin = **null**;
189. FileWriter fw =**null**;
190. **try** {
191. fin =**new** FileInputStream(inputpath);
192. } **catch** (FileNotFoundException e1) {
193. // TODO 自动生成的 catch 块
194. e1.printStackTrace();
195. }
196. **try** {
197. fw = **new** FileWriter(outputpath, **true**);
198. } **catch** (IOException e1) {
199. // TODO 自动生成的 catch 块
200. e1.printStackTrace();
201. }
202. PrintWriter pw = **new** PrintWriter(fw);
203. pw.print(inputpath+"\u7684\u8bed\u6cd5\u5206\u6790\u7ed3\u679c\uff1a\u005cn");
204. pw.flush();
205. Parser YF = **new** Parser (fin);
206. **try**
207. {
208. SimpleNode n = YF.Start();
209. String res = n.dump("");
210. //
211. System.out.println("\u8fd4\u56de\u7684\u7ed3\u679c:\u005cn"+res);
212. //
213. pw.print(res);
214. pw.flush();
215. pw.close();
216. fin.close();
217. }
218. **catch** (Exception e)
219. {
220. System.out.println("Oops.");
221. System.out.println(e.getMessage());
222. }
223. System.out.println("Analysis done! Please check it in file output.");
224. }
225. 修改dump()函数
226. **public** String dump(String prefix)
227. {
228. String res = toString(prefix)+"\n";
229. System.out.println(toString(prefix));
230. **if** (children != **null**)
231. {
232. **for** (**int** i = 0; i < children.length; ++i)
233. {
234. SimpleNode n = (SimpleNode)children[i];
235. **if** (n != **null**) {
236. res += n.dump(prefix + " ");
237. }
238. }
239. }
240. **return** res;
241. }
242. 错误提示

//修改ParserException.java文件中的initialise()方法

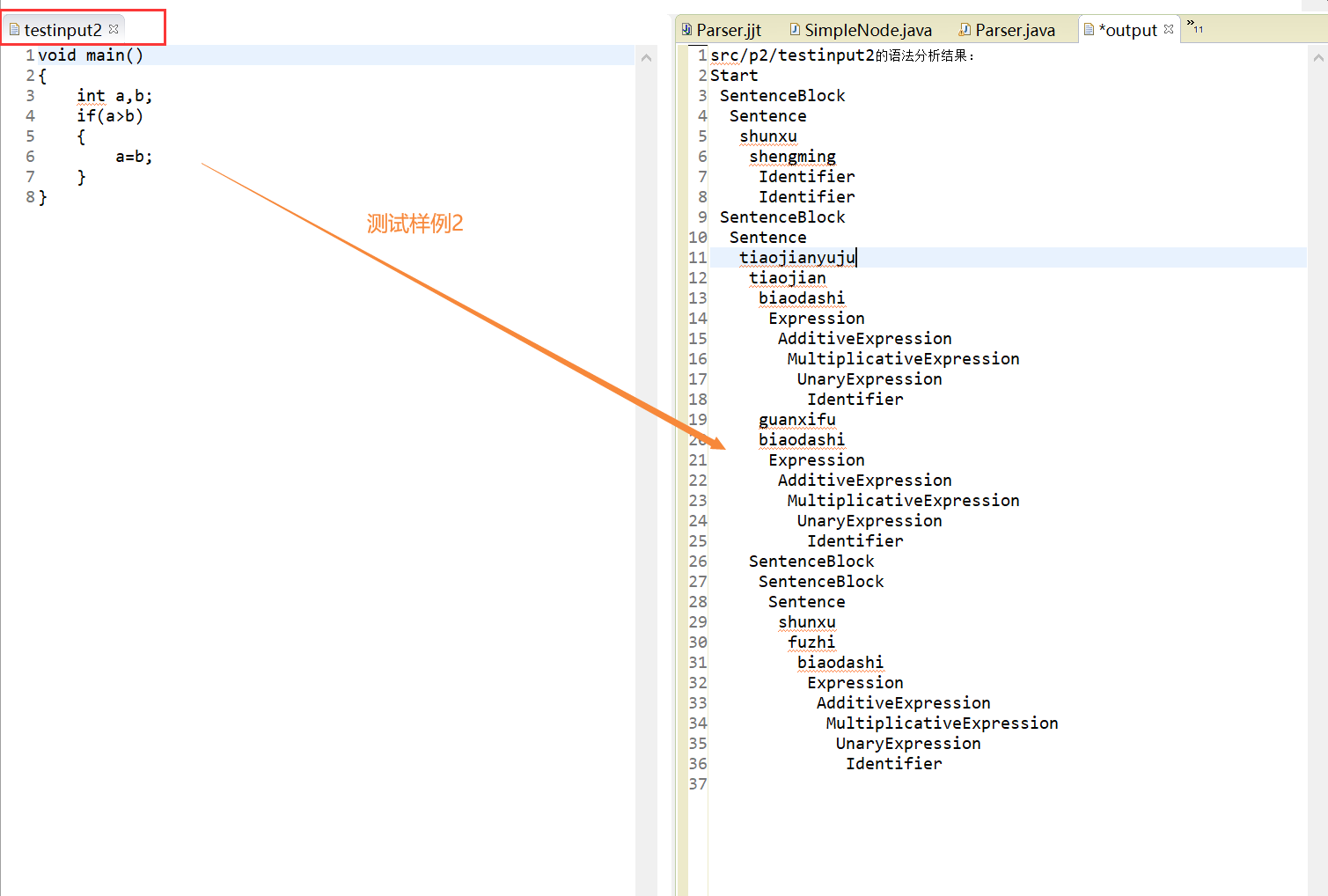
1. **private** **static** String initialise(Token currentToken,
2. **int**[][] expectedTokenSequences,
3. String[] tokenImage) {
4. String eol = System.getProperty("line.separator", "\n");
5. StringBuffer expected = **new** StringBuffer();
6. **int** maxSize = 0;
7. **for** (**int** i = 0; i < expectedTokenSequences.length; i++) {
8. **if** (maxSize < expectedTokenSequences[i].length) {
9. maxSize = expectedTokenSequences[i].length;
10. }
11. **for** (**int** j = 0; j < expectedTokenSequences[i].length; j++) {
12. expected.append(tokenImage[expectedTokenSequences[i][j]]).append(' ');
13. }
14. **if** (expectedTokenSequences[i][expectedTokenSequences[i].length - 1] != 0) {
15. expected.append("...");
16. }
17. expected.append(eol).append("    ");
18. }
19. String retval = "遇到了 \"";
20. Token tok = currentToken.next;
21. **for** (**int** i = 0; i < maxSize; i++) {
22. **if** (i != 0) retval += " ";
23. **if** (tok.kind == 0) {
24. retval += tokenImage[0];
25. **break**;
26. }
27. retval += " " + tokenImage[tok.kind];
28. retval += " \"";
29. retval += add\_escapes(tok.image);
30. retval += " \"";
31. tok = tok.next;
32. }
33. retval += "\" 在行数为 " + currentToken.next.beginLine + ", 列数为 " + currentToken.next.beginColumn;
34. retval += "处." + eol;
35. **if** (expectedTokenSequences.length == 1) {
36. retval += "这里应该是有:" + eol + "    ";
37. } **else** {
38. retval += "应该在这里有下列符号:" + eol + "    ";
39. }
40. retval += expected.toString();
41. **return** retval;
42. }

**结果测试：**

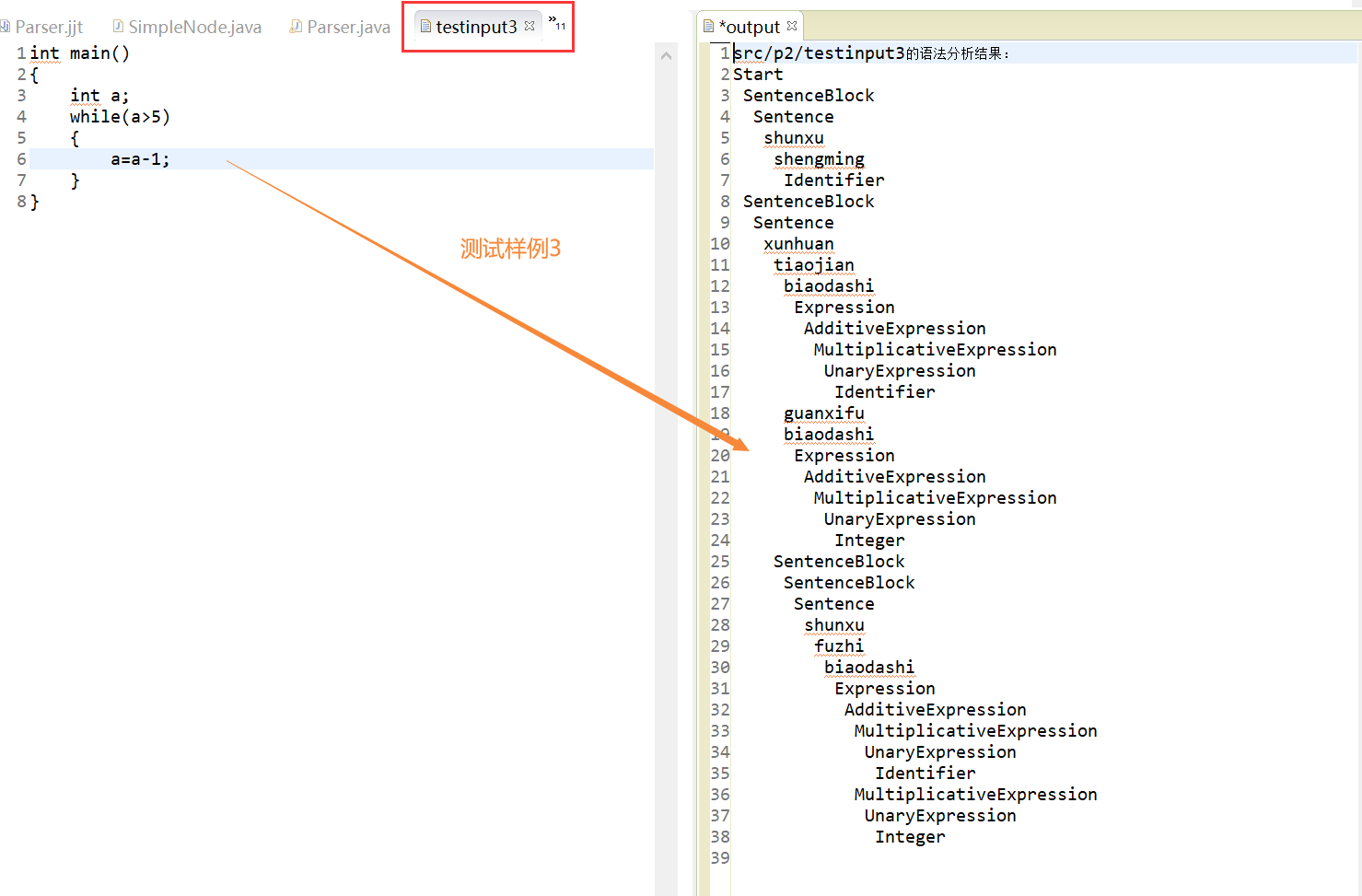
Testinput1:



Testinput2:



Testinput3:



TestErrorInput:



# 疑问和建议

## 疑问

1）Parser.java文件里面的jjtn00, jjtc00, jjte00都是什么含义？

2）jjtThis起什么作用？是用来处理源程序里面的return语句的吗？

1. SimpleNode Start() :
2. {}
3. {
4. ("int"|"void")"main""("")""{"(SentenceBlock())\*"}"
5. {
6. **return** jjtThis;
7. }
8. }

## 建议

实习讲解录屏讲的生动有趣，实习内容也贴近课堂知识，整个实习安排近乎完美，没有需要改进的建议了。

最后感谢老师在实习过程中的悉心指导与帮助！