Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра вычислительной техники



Отчет по лабораторной работе № 3

по дисциплине «Теория формальных языков и компиляторов»

на тему «Синтаксис языков программирования. Формальные грамматики»

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| Студент гр. *АВТ-709*, *АВТФ* |  |
| *Андреева А.О.* | *Малявко Александр Антонович* |
| «\_\_\_» \_\_\_\_\_\_ 20\_\_г. | «\_\_\_» \_\_\_\_\_\_ 20\_\_г. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (подпись) | (подпись) |

Новосибирск, 2020

# Цель работы

Изучение основных понятий метаязыка формальных грамматик, свойств грамматик и нетерминальных символов, рекурсивности и однозначности грамматик, недостижимости, бесплодности, аннулируемости и рекурсивности нетерминальных символов, отношений предшествования и последования между символами, приобретение навыков эквивалентных преобразований формальных грамматик, освоение технологий разработки формальных грамматик.

# Постановка задачи

1. Изучить интерфейс пакета ВебТрансЛаб, предназначенный для работы с синтаксическими правилами.

2. На примерах 3LL1 и 4SLR1GrammarArithmExpr (грамматики Ga1 и Ga2 из учебника) изучить и освоить:

* ввод и редактирование правил грамматики;
* построение синтаксических акцепторов с целью вычисления свойств грамматики и ее символов – пункт меню «Построить» с выбором нужного шаблона;
* понятия множеств предшественников и множеств последователей для нетерминальных символов;
* понятие множеств выбора правил грамматики и их пересечений для правил с одинаковыми нетерминалами в левой части;
* просмотр свойств символов грамматики (следует достичь понимания того, почему те или иные символы грамматики имеют свой конкретный набор свойств – пункт меню «Показать» и подпункты «Правила грамматики», «Отношение предшествования», «Отношение последования» и «Множества выбора правил».

3. Изучить способы эквивалентных преобразований грамматики из примера 4SLR1GrammarArithmExpr для устранения пересечений множеств выбора правил с одинаковыми нетерминалами в левой части, т.е. превращения этой грамматики в LL(1)-грамматику.

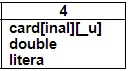
4. Ориентируясь на свой вариант задания на курсовую работу, модифицировать (расширить) грамматики Ga1 и Ga2 до грамматик операторов присваивания, условного оператора и оператора цикла, добавляя новые нетерминалы и правила по технологии разработки «снизу вверх». Изучить свойства полученных грамматик и их символов.

5. Используя полученные навыки работы с грамматиками и программным обеспечением, начать поэтапную разработку грамматики по технологии «сверху вниз» для заданного варианта курсовой работы.

6. Оценить свои субъективные проблемы и трудности разработки формальных грамматик при использовании разных технологий («снизу вверх» и «сверху вниз»). Выбрать технологию для выполнения задания на курсовую работу (рекомендуется технология «сверху вниз»).

7. Подготовить, сдать и защитить отчет к лабораторной работе.

Вариант:

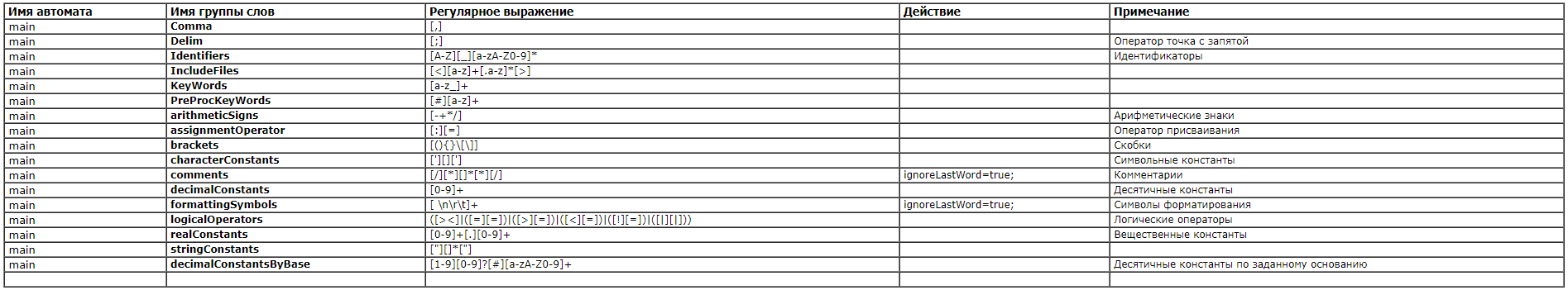
**

*Рисунок 1 – Типы*

# Ход работы

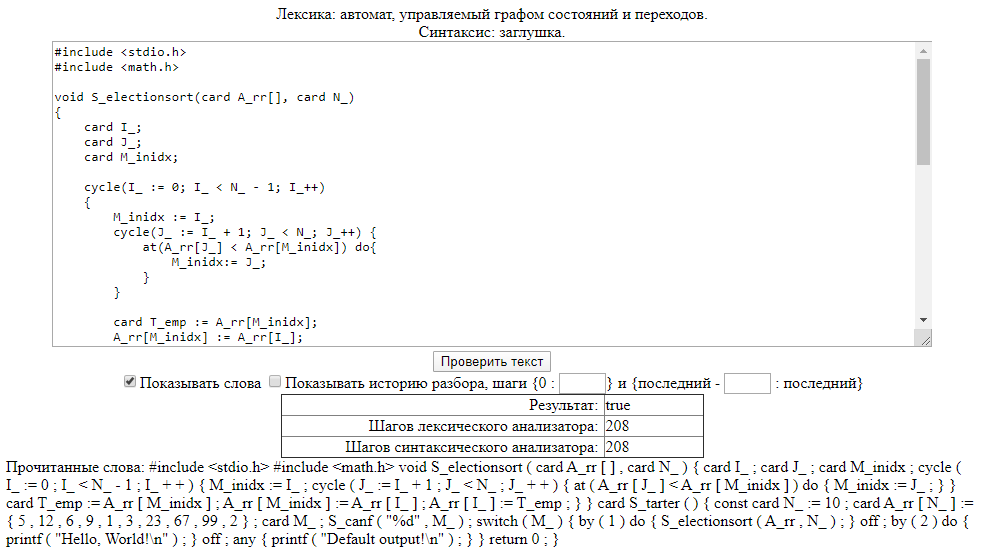
**Исправления:**

В соответствие с замечаниями по лабораторной работе №2 были внесены изменения в правила лексики.



*Рис. 2 - Правила лексики с правками*

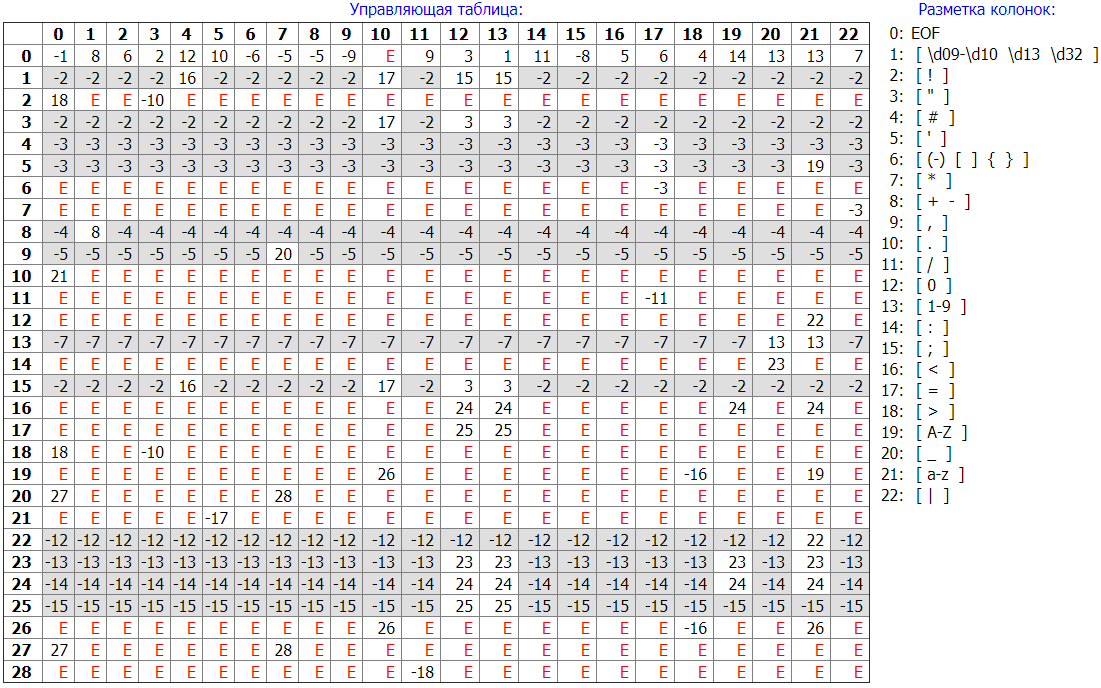
Проверим обновленные правила лексики на примере программы из лабораторной работы №2.



*Рис. 3 - Проверка обновленных правил лексики.*

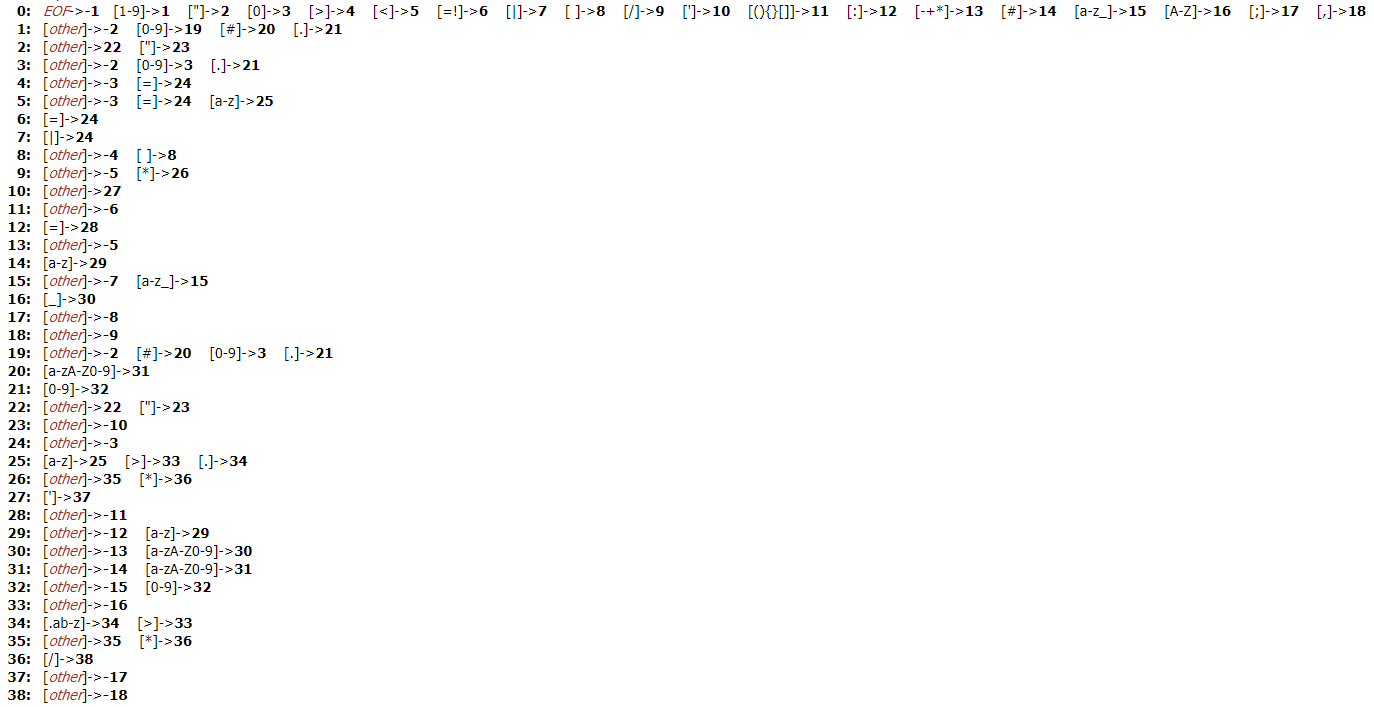
**Работа лексических автоматов по разбору заданного фрагмента:**

Управляющая таблица лексического анализатора выглядит следующим образом. Рис. 4.



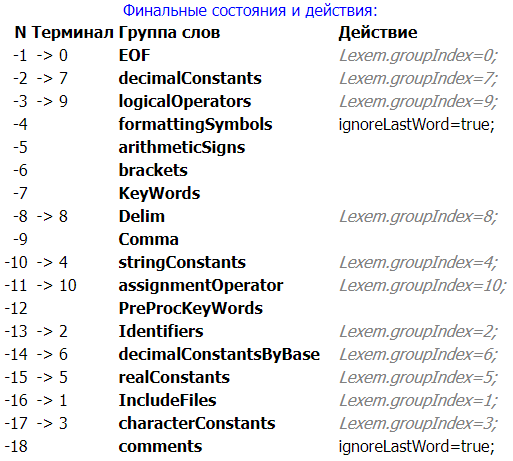
*Рис. 4 – Управляющая таблица лексического анализатора.*

Список состояний и переходов лексического анализатора выглядит следующим образом. Рис. 5.



*Рис. 5 – Список состояний и переходов лексического анализатора.*

Финальные состояния и действия представлены на рисунке 12.



*Рис. 6 - Финальные состояния и действия.*

**Разбор заданного фрагмента:**

Заданный фрагмент программы: const card N\_:=10;

* Конечный автомат, заданный таблицей:

Составим таблицу, задающую конечный автомат.

Вид таблицы: (*текущая строка, текущий столбец, текущий элемент*)

*KeyWords (const):*

*Format (“ ”):*

*KeyWords (card):*

*Format (“ ”):*

*Identifiers (N\_):*

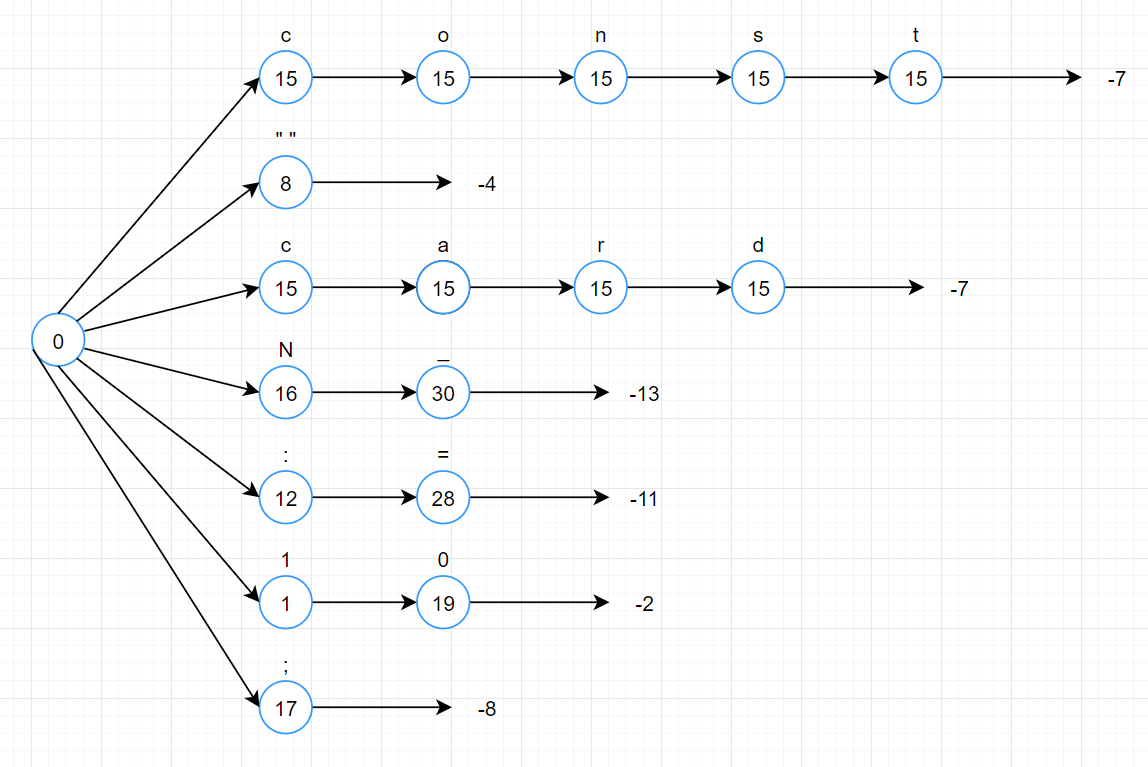
*AssignmentOperator (:=):*

*DeсimalConst (10):*

*Delim (;):*

* Конечный автомат, заданный графом:

Составим граф, демонстрирующий работу конечного автомата. Рис. 13.



*Рис. 7 - Конечный автомат, заданный графом*

**Описание метаязыка формальных грамматик**

*Формальной грамматикой называется совокупность , где:*

*алфавит терминальных символов, это есть конечное множество всех слов языка, порождаемого данной грамматикой. Понятие «терминальный» здесь обозначает неразложимость таких символов с точки зрения синтаксических правил;*

*алфавит нетерминальных символов, это есть конечное множество названий синтаксических конструкций, например: <предложение>, <выражение>, <список аргументов>, <условный оператор>, <тело функции>. Нетерминальные символы используются только в метаязыке, на котором описывает язык программирования;*

*начальный терминальный символ, это есть один из нетерминальных символов. Этим символом обычно обозначается наиболее общая синтаксическая конструкция, например: <правильная программа>;*

*система правил подстановки, это есть конечное множество пар цепочек вида α : β, причем цепочка α (левая часть правила) должна содержать хотя бы один нетерминальный символ;*

**Свойства грамматик:**

* **Рекурсивность**

*Нетерминальный символ Х называется рекурсивным, если из него могут быть выведены цепочки, содержащие сам этот символ, т. е.*

*Х ⇒ μ Х η,*

*где μ и η – произвольные цепочки символов.*

*Грамматика называется рекурсивной, если рекурсивен хотя бы один нетерминальный символ, и не рекурсивной в противном случае.*

* **Однозначность**

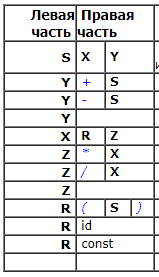
*Грамматика называется однозначной, если любое правильное предложение порождаемого ею языка имеет единственное дерево грамматического разбора, и неоднозначной в противном случае.*

**Свойства символов грамматик:**

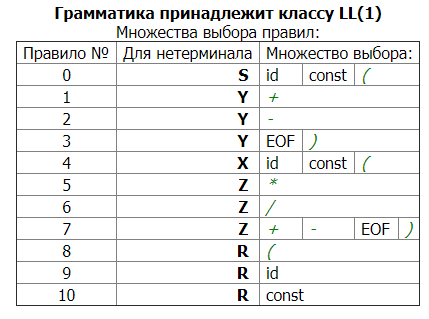
* **Аннулируемость -** *Нетерминальный символ называется аннулируемым, если из него может быть выведена пустая цепочка символов. В противном случае нетерминальный символ называется неаннулируемым.*
* **Недостижимость -** *Символ называется недостижимым, если он не появляется ни в одной цепочке символов, выводимой из начального нетерминала грамматики.*
* **Бесплодность -** *Нетерминальный символ называется бесплодным, если из него не может быть выведена ни одна цепочка, состоящая только из терминальных символов.*

**Разбор заданных примеров:**

***Файл «3LL1.xml»***



*Рис. 8 – Правила Грамматики*



*Рис. 9 - Множество выбора правил языка*



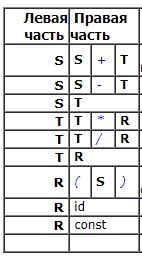
*Рис. 10 - Множество предшественников.*



*Рис. 11 - Множество последователей.*

Как можно заметить, в системе правил, представленной в примере, не содержится недостижимых и бесплодных грамматик, но, содержится два аннулируемых правила (Z и Y). Множества выбора для всех нетерминалов не пересекаются. Язык принадлежит классу LL(1) и пригоден для корректного построения и работы синтаксического акцептора.

***Файл «4SLR1grammarArithmExpr.xml»***



*Рис. 12 - Правила грамматики.*



*Рис.13 - Множество выбора правил.*



*Рис. 14 - Множество предшественников.*



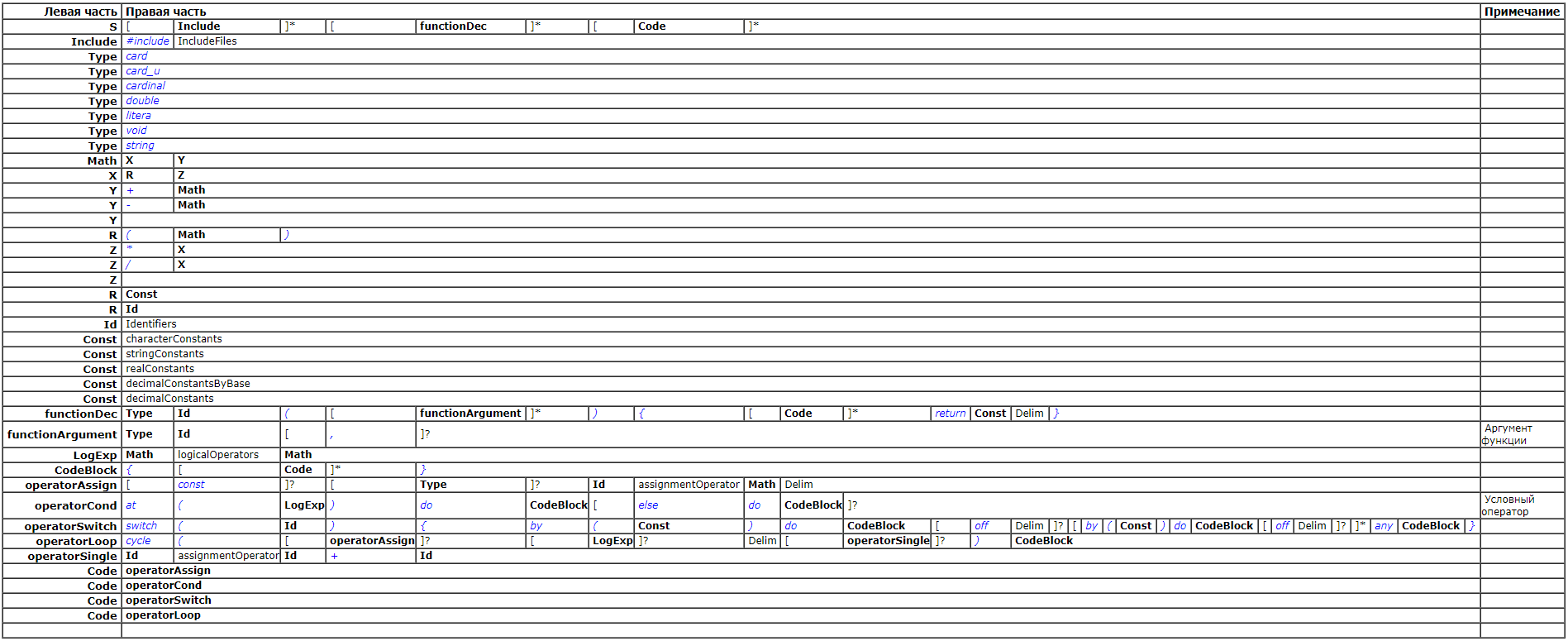
*Рис. 15 - Множество предшественников.*

Как можно заметить, во множестве выбора данной грамматики содержатся пересечения, что говорит о том, что грамматика не является LL(1) и ее работа будет некорректна.

**Разработка грамматики программируемого языка**

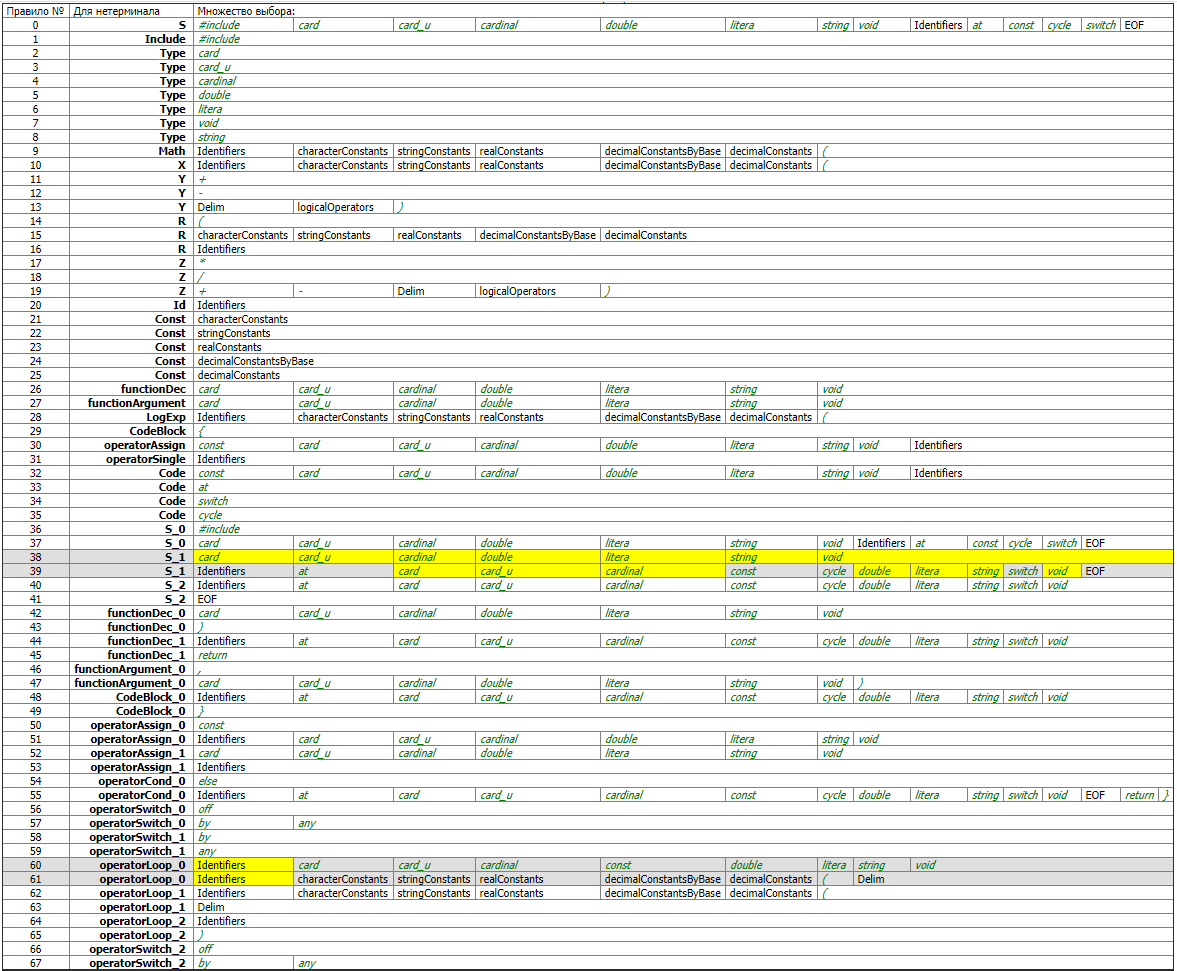
Приступим к разработке грамматики для нашего языка, в соответствии с заданным вариантом.

Первичные правила грамматики представлены на рисунке 16.



*Рис. 16 - Первичная грамматика.*

Были описаны грамматики операторов присваивания, условного оператора, оператора переключателя и оператора цикла.

  
*Рис. 17 - Множества выбора правил*

Как можно заметить, грамматика не принадлежит к классу LL(1). В следующей лабораторной работе постараемся это исправить.

Программа для проверки

#include <math.h>

#include <iostream>

card S\_tarter(){

card N\_ := 10;

cycle(card I\_ := 0; I\_ < N\_; ){

**switch**(I\_){

by(2) **do** {

I\_ := I\_ + 1;

}

off;

any {

I\_ := I\_ + 2;

}

}

}

at(N\_ < 10) **do** {

N\_:=15;

}

**else** **do** {

N\_:=5;

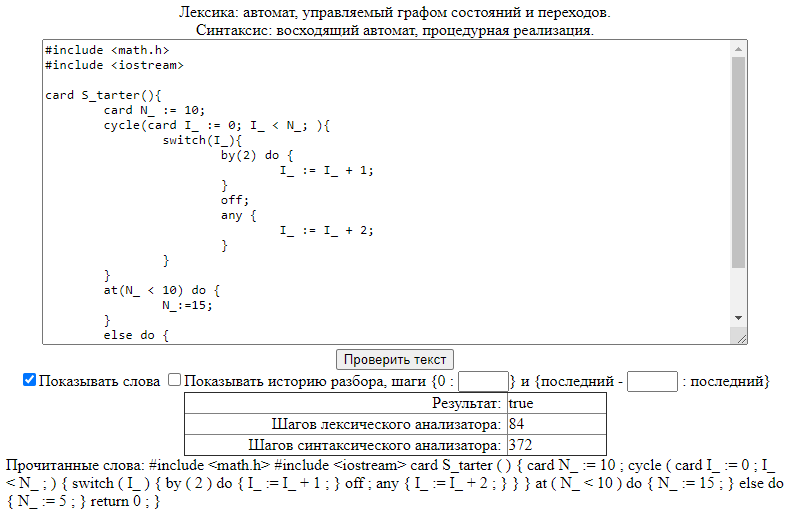
}

**return** 0;

}

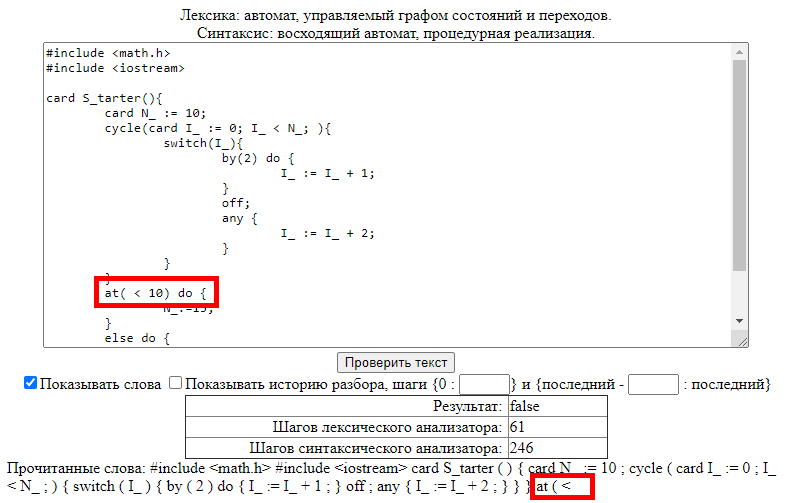
В начале программы имеется команда препроцессора, после идет функция S\_taret(). В теле функции можно заметить: оператор цикла, оператор переключателя и условный оператор.

Проверка правильности, разработанной грамматики показана на Рис. 18.



*Рис. 18 - Проверка правильности грамматики (случай без ошибки).*

Допустим ошибку и не напишем один из операндов в условном операторе.



*Рис. 19 - Проверка правильности грамматики (случай с ошибкой)*

# Вывод

В ходе данной лабораторной работы были изучены основные понятия метаязыка формальных грамматик, свойств грамматик и нетерминальных символов. Была частично разработана грамматика для языка, в соответствии с заданным вариантом*.*

Были написаны тестовые программы, которые показали успешный разбор грамматики и не успешный.

В соответствие с заданным фрагментом кода программы были составлены: конечный автомат, заданный таблицей и конечный автомат, заданный графом.

**Приложение:**

1. **<?xml** version="1.0" encoding="windows-1251"**?>**
2. **<transLab>**
3. **<lexic>**
4. **<automat** name='main'**>**
5. **<rule** groupWordsName='Comma'**>**
6. **<expression>**[,]**</expression>**
7. **</rule>**
8. **<rule** groupWordsName='Delim'**>**
9. **<expression>**[;]**</expression>**
10. **<comment>**Оператор точка с запятой**</comment>**
11. **</rule>**
12. **<rule** groupWordsName='Identifiers'**>**
13. **<expression>**[A-Z][\_][a-zA-Z0-9]\***</expression>**
14. **<comment>**Идентификаторы**</comment>**
15. **</rule>**
16. **<rule** groupWordsName='IncludeFiles'**>**
17. **<expression>**[**<**][a-z]+[.a-z]\*[**>**]**</expression>**
18. **</rule>**
19. **<rule** groupWordsName='KeyWords'**>**
20. **<expression>**[a-z\_]+**</expression>**
21. **</rule>**
22. **<rule** groupWordsName='PreProcKeyWords'**>**
23. **<expression>**[#][a-z]+**</expression>**
24. **</rule>**
25. **<rule** groupWordsName='arithmeticSigns'**>**
26. **<expression>**[-+\*/]**</expression>**
27. **<comment>**Арифметические знаки**</comment>**
28. **</rule>**
29. **<rule** groupWordsName='assignmentOperator'**>**
30. **<expression>**[:][=]**</expression>**
31. **<comment>**Оператор присваивания**</comment>**
32. **</rule>**
33. **<rule** groupWordsName='brackets'**>**
34. **<expression>**[(){}\[\]]**</expression>**
35. **<comment>**Скобки**</comment>**
36. **</rule>**
37. **<rule** groupWordsName='characterConstants'**>**
38. **<expression>**['][][']**</expression>**
39. **<comment>**Символьные константы**</comment>**
40. **</rule>**
41. **<rule** groupWordsName='comments'**>**
42. **<expression>**[/][\*][]\*[\*][/]**</expression>**
43. **<action>**ignoreLastWord=true;**</action>**
44. **<comment>**Комментарии**</comment>**
45. **</rule>**
46. **<rule** groupWordsName='decimalСonstants'**>**
47. **<expression>**[0-9]+**</expression>**
48. **<comment>**Десятичные константы**</comment>**
49. **</rule>**
50. **<rule** groupWordsName='formattingSymbols'**>**
51. **<expression>**[ \n\r\t]+**</expression>**
52. **<action>**ignoreLastWord=true;**</action>**
53. **<comment>**Символы форматирования**</comment>**
54. **</rule>**
55. **<rule** groupWordsName='logicalOperators'**>**
56. **<expression>**([**><**]|([=][=])|([**>**][=])|([**<**][=])|([!][=])|([|][|]))**</expression>**
57. **<comment>**Логические операторы**</comment>**
58. **</rule>**
59. **<rule** groupWordsName='Const'**>**
60. **<expression>**([1-9][0-9]\*)|([0])**</expression>**
61. **<comment>**Десятичное целое число**</comment>**
62. **</rule>**
63. **<rule** groupWordsName='twoConst'**>**
64. **<expression>**[2][xX][0-1]\***</expression>**
65. **<comment>**Двоичное целое число**</comment>**
66. **</rule>**
67. **<rule** groupWordsName='eightConst'**>**
68. **<expression>**[8][xX][0-7]\***</expression>**
69. **<comment>**Восьмеричное целое число**</comment>**
70. **</rule>**
71. **<rule** groupWordsName='floatConst'**>**
72. **<expression>**[0-9]\*[.][0-9]+**</expression>**
73. **<comment>**Десятичное вещественное число**</comment>**
74. **</rule>**
75. **<rule** groupWordsName='charConst'**>**
76. **<expression>**['][]?[']**</expression>**
77. **<comment>**Символ**</comment>**
78. **</rule>**
79. **</rule>**
80. **</automat>**
81. **</lexic>**
82. **<syntax>**
83. **<rule** for='S' leftPart='S'**>**
84. **<rightPart>**
85. **<symbol>**[**</symbol>**
86. **<symbol>**Include**</symbol>**
87. **<symbol>**]\***</symbol>**
88. **<symbol>**[**</symbol>**
89. **<symbol>**functionDec**</symbol>**
90. **<symbol>**]\***</symbol>**
91. **<symbol>**[**</symbol>**
92. **<symbol>**Code**</symbol>**
93. **<symbol>**]\***</symbol>**
94. **</rightPart>**
95. **</rule>**
96. **<rule** for='S' leftPart='Include'**>**
97. **<rightPart>**
98. **<string>**#include**</string>**
99. **<symbol>**IncludeFiles**</symbol>**
100. **</rightPart>**
101. **</rule>**
102. **<rule** for='S' leftPart='Type'**>**
103. **<rightPart>**
104. **<string>**card**</string>**
105. **</rightPart>**
106. **</rule>**
107. **<rule** for='S' leftPart='Type'**>**
108. **<rightPart>**
109. **<string>**card\_u**</string>**
110. **</rightPart>**
111. **</rule>**
112. **<rule** for='S' leftPart='Type'**>**
113. **<rightPart>**
114. **<string>**cardinal**</string>**
115. **</rightPart>**
116. **</rule>**
117. **<rule** for='S' leftPart='Type'**>**
118. **<rightPart>**
119. **<string>**double**</string>**
120. **</rightPart>**
121. **</rule>**
122. **<rule** for='S' leftPart='Type'**>**
123. **<rightPart>**
124. **<string>**litera**</string>**
125. **</rightPart>**
126. **</rule>**
127. **<rule** for='S' leftPart='Type'**>**
128. **<rightPart>**
129. **<string>**void**</string>**
130. **</rightPart>**
131. **</rule>**
132. **<rule** for='S' leftPart='Type'**>**
133. **<rightPart>**
134. **<string>**string**</string>**
135. **</rightPart>**
136. **</rule>**
137. **<rule** for='S' leftPart='Math'**>**
138. **<rightPart>**
139. **<symbol>**X**</symbol>**
140. **<symbol>**Y**</symbol>**
141. **</rightPart>**
142. **</rule>**
143. **<rule** for='S' leftPart='X'**>**
144. **<rightPart>**
145. **<symbol>**R**</symbol>**
146. **<symbol>**Z**</symbol>**
147. **</rightPart>**
148. **</rule>**
149. **<rule** for='S' leftPart='Y'**>**
150. **<rightPart>**
151. **<string>**+**</string>**
152. **<symbol>**Math**</symbol>**
153. **</rightPart>**
154. **</rule>**
155. **<rule** for='S' leftPart='Y'**>**
156. **<rightPart>**
157. **<string>**-**</string>**
158. **<symbol>**Math**</symbol>**
159. **</rightPart>**
160. **</rule>**
161. **<rule** for='S' leftPart='Y'**>**
162. **<rightPart>**
163. **</rightPart>**
164. **</rule>**
165. **<rule** for='S' leftPart='R'**>**
166. **<rightPart>**
167. **<string>**(**</string>**
168. **<symbol>**Math**</symbol>**
169. **<string>**)**</string>**
170. **</rightPart>**
171. **</rule>**
172. **<rule** for='S' leftPart='Z'**>**
173. **<rightPart>**
174. **<string>**\***</string>**
175. **<symbol>**X**</symbol>**
176. **</rightPart>**
177. **</rule>**
178. **<rule** for='S' leftPart='Z'**>**
179. **<rightPart>**
180. **<string>**/**</string>**
181. **<symbol>**X**</symbol>**
182. **</rightPart>**
183. **</rule>**
184. **<rule** for='S' leftPart='Z'**>**
185. **<rightPart>**
186. **</rightPart>**
187. **</rule>**
188. **<rule** for='S' leftPart='R'**>**
189. **<rightPart>**
190. **<symbol>**Const**</symbol>**
191. **</rightPart>**
192. **</rule>**
193. **<rule** for='S' leftPart='R'**>**
194. **<rightPart>**
195. **<symbol>**Id**</symbol>**
196. **</rightPart>**
197. **</rule>**
198. **<rule** for='S' leftPart='Id'**>**
199. **<rightPart>**
200. **<symbol>**Identifiers**</symbol>**
201. **</rightPart>**
202. **</rule>**
203. **<rule** for='S' leftPart='Const'**>**
204. **<rightPart>**
205. **<symbol>**characterConstants**</symbol>**
206. **</rightPart>**
207. **</rule>**
208. **<rule** for='S' leftPart='Const'**>**
209. **<rightPart>**
210. **<symbol>**stringConstants**</symbol>**
211. **</rightPart>**
212. **</rule>**
213. **<rule** for='S' leftPart='Const'**>**
214. **<rightPart>**
215. **<symbol>**realConstants**</symbol>**
216. **</rightPart>**
217. **</rule>**
218. **<rule** for='S' leftPart='Const'**>**
219. **<rightPart>**
220. **<symbol>**decimalСonstantsByBase**</symbol>**
221. **</rightPart>**
222. **</rule>**
223. **<rule** for='S' leftPart='Const'**>**
224. **<rightPart>**
225. **<symbol>**decimalСonstants**</symbol>**
226. **</rightPart>**
227. **</rule>**
228. **<rule** for='S' leftPart='functionDec'**>**
229. **<rightPart>**
230. **<symbol>**Type**</symbol>**
231. **<symbol>**Id**</symbol>**
232. **<string>**(**</string>**
233. **<symbol>**[**</symbol>**
234. **<symbol>**functionArgument**</symbol>**
235. **<symbol>**]\***</symbol>**
236. **<string>**)**</string>**
237. **<string>**{**</string>**
238. **<symbol>**[**</symbol>**
239. **<symbol>**Code**</symbol>**
240. **<symbol>**]\***</symbol>**
241. **<string>**return**</string>**
242. **<symbol>**Const**</symbol>**
243. **<symbol>**Delim**</symbol>**
244. **<string>**}**</string>**
245. **</rightPart>**
246. **</rule>**
247. **<rule** for='S' leftPart='functionArgument'**>**
248. **<rightPart>**
249. **<symbol>**Type**</symbol>**
250. **<symbol>**Id**</symbol>**
251. **<symbol>**[**</symbol>**
252. **<string>**,**</string>**
253. **<symbol>**]?**</symbol>**
254. **</rightPart>**
255. **<comment>**Аргумент функции**</comment>**
256. **</rule>**
257. **<rule** for='S' leftPart='LogExp'**>**
258. **<rightPart>**
259. **<symbol>**Math**</symbol>**
260. **<symbol>**logicalOperators**</symbol>**
261. **<symbol>**Math**</symbol>**
262. **</rightPart>**
263. **</rule>**
264. **<rule** for='S' leftPart='CodeBlock'**>**
265. **<rightPart>**
266. **<string>**{**</string>**
267. **<symbol>**[**</symbol>**
268. **<symbol>**Code**</symbol>**
269. **<symbol>**]\***</symbol>**
270. **<string>**}**</string>**
271. **</rightPart>**
272. **</rule>**
273. **<rule** for='S' leftPart='operatorAssign'**>**
274. **<rightPart>**
275. **<symbol>**[**</symbol>**
276. **<string>**const**</string>**
277. **<symbol>**]?**</symbol>**
278. **<symbol>**[**</symbol>**
279. **<symbol>**Type**</symbol>**
280. **<symbol>**]?**</symbol>**
281. **<symbol>**Id**</symbol>**
282. **<symbol>**assignmentOperator**</symbol>**
283. **<symbol>**Math**</symbol>**
284. **<symbol>**Delim**</symbol>**
285. **</rightPart>**
286. **</rule>**
287. **<rule** for='S' leftPart='operatorCond'**>**
288. **<rightPart>**
289. **<string>**at**</string>**
290. **<string>**(**</string>**
291. **<symbol>**LogExp**</symbol>**
292. **<string>**)**</string>**
293. **<string>**do**</string>**
294. **<symbol>**CodeBlock**</symbol>**
295. **<symbol>**[**</symbol>**
296. **<string>**else**</string>**
297. **<string>**do**</string>**
298. **<symbol>**CodeBlock**</symbol>**
299. **<symbol>**]?**</symbol>**
300. **</rightPart>**
301. **<comment>**Условный оператор**</comment>**
302. **</rule>**
303. **<rule** for='S' leftPart='operatorSwitch'**>**
304. **<rightPart>**
305. **<string>**switch**</string>**
306. **<string>**(**</string>**
307. **<symbol>**Id**</symbol>**
308. **<string>**)**</string>**
309. **<string>**{**</string>**
310. **<string>**by**</string>**
311. **<string>**(**</string>**
312. **<symbol>**Const**</symbol>**
313. **<string>**)**</string>**
314. **<string>**do**</string>**
315. **<symbol>**CodeBlock**</symbol>**
316. **<symbol>**[**</symbol>**
317. **<string>**off**</string>**
318. **<symbol>**Delim**</symbol>**
319. **<symbol>**]?**</symbol>**
320. **<symbol>**[**</symbol>**
321. **<string>**by**</string>**
322. **<string>**(**</string>**
323. **<symbol>**Const**</symbol>**
324. **<string>**)**</string>**
325. **<string>**do**</string>**
326. **<symbol>**CodeBlock**</symbol>**
327. **<symbol>**[**</symbol>**
328. **<string>**off**</string>**
329. **<symbol>**Delim**</symbol>**
330. **<symbol>**]?**</symbol>**
331. **<symbol>**]\***</symbol>**
332. **<string>**any**</string>**
333. **<symbol>**CodeBlock**</symbol>**
334. **<string>**}**</string>**
335. **</rightPart>**
336. **</rule>**
337. **<rule** for='S' leftPart='operatorLoop'**>**
338. **<rightPart>**
339. **<string>**cycle**</string>**
340. **<string>**(**</string>**
341. **<symbol>**[**</symbol>**
342. **<symbol>**operatorAssign**</symbol>**
343. **<symbol>**]?**</symbol>**
344. **<symbol>**[**</symbol>**
345. **<symbol>**LogExp**</symbol>**
346. **<symbol>**]?**</symbol>**
347. **<symbol>**Delim**</symbol>**
348. **<symbol>**[**</symbol>**
349. **<symbol>**operatorSingle**</symbol>**
350. **<symbol>**]?**</symbol>**
351. **<string>**)**</string>**
352. **<symbol>**CodeBlock**</symbol>**
353. **</rightPart>**
354. **</rule>**
355. **<rule** for='S' leftPart='operatorSingle'**>**
356. **<rightPart>**
357. **<symbol>**Id**</symbol>**
358. **<symbol>**assignmentOperator**</symbol>**
359. **<symbol>**Id**</symbol>**
360. **<string>**+**</string>**
361. **<symbol>**Id**</symbol>**
362. **</rightPart>**
363. **</rule>**
364. **<rule** for='S' leftPart='Code'**>**
365. **<rightPart>**
366. **<symbol>**operatorAssign**</symbol>**
367. **</rightPart>**
368. **</rule>**
369. **<rule** for='S' leftPart='Code'**>**
370. **<rightPart>**
371. **<symbol>**operatorCond**</symbol>**
372. **</rightPart>**
373. **</rule>**
374. **<rule** for='S' leftPart='Code'**>**
375. **<rightPart>**
376. **<symbol>**operatorSwitch**</symbol>**
377. **</rightPart>**
378. **</rule>**
379. **<rule** for='S' leftPart='Code'**>**
380. **<rightPart>**
381. **<symbol>**operatorLoop**</symbol>**
382. **</rightPart>**
383. **</rule>**
384. **</syntax>**
385. **</transLab>**