

# Лабораторная работа № 7.2

## «Самоприменимый генератор компиляторов на основе предсказывающего анализа»

Скоробогатов С. Ю., Коновалов А. В.

11 апреля 2016

### 1 Цель работы

Целью данной работы является изучение алгоритма построения таблиц предсказывающего анализатора.

### 2 Исходные данные

В данной лабораторной работе требуется разработать простейший генератор компиляторов, который по описанию грамматики рабочего языка, записанному на входном языке, строит таблицы предсказывающего анализа в виде составного литерала на целевом языке.

Здесь *рабочий язык* — это некоторый формальный язык с LL(1)-грамматикой, синтаксический анализатор которого должен быть построен генератором компиляторов.

В качестве *входного языка* генератора компиляторов должен выступать язык представления правил грамматики, варианты лексики и синтаксиса которого можно восстановить по примерам из таблицы 1.

И, наконец, *целевым языком* мы будем называть язык реализации компилятора рабочего языка. Так как разрабатываемый генератор компиляторов должен быть самоприменимым, целевой язык должен совпадать с языком реализации генератора компиляторов.

### 3 Задание

Лабораторная работа делается на основе выполненной лабораторной работы 7.1. Выполнение данной лабораторной работы состоит из следующих этапов:

1. Переписывание грамматики входного языка на самом входном языке.
2. Добавление в программу, написанную на лабораторной работе 7.1, генератора таблицы разбора на основе дерева разбора. Таблица разбора должна представлять собой инициализированный двумерный массив на рабочем языке (он совпадает с языком реализации генератора компиляторов).
3. Тестирование генератора компиляторов путём написания простейшего калькулятора арифметических выражений на основе грамматики, описанной в таблице 1.

Таблица 1: Варианты входного языка в примерах описаний грамматик

1	<pre># объявления non-terminal E, E1, T, T1, F; terminal '+', '*', '(', ')', n;  # правила грамматики E ::= T E1; E1 ::= '+' T E1   epsilon; T ::= F T1; T1 ::= '*' F T1   epsilon; F ::= n   '(' E ')';  axiom E;</pre>	2	<pre>\$AXIOM E \$NTERM E' T T' F \$TERM "+" "*" "(" ")" "n"  * правила грамматики \$RULE E = T E' \$RULE E' = "+" T E' \$EPS \$RULE T = F T' \$RULE T' = "*" F T' \$EPS \$RULE F = "n" "(" E ")"</pre>
3	<pre>; только правила грамматики  (F) = n   \( (E) \). (T) = (F) (T1). (T1) = * (F) (T1)   . (axiom E) = (T) (E1). (E1) = + (T) (E1)   .</pre>	4	<pre>/* аксиома помечена звёздочкой */ F ("n") ("(" E ")") T (F T') T' ("*" F T') () * E (T E') E' ("+" T E') ()</pre>
5	<pre>tokens &lt;plus sign&gt;, &lt;star&gt;, &lt;n&gt;. &lt;E&gt; is &lt;T&gt; &lt;E 1&gt;. &lt;E 1&gt; is &lt;plus sign&gt; &lt;T&gt; &lt;E 1&gt;. &lt;E 1&gt; is . &lt;T&gt; is &lt;F&gt; &lt;T 1&gt;. &lt;T 1&gt; is &lt;star&gt; &lt;F&gt; &lt;T 1&gt;. &lt;T 1&gt; is . &lt;F&gt; is &lt;n&gt;. tokens &lt;left paren&gt;,       &lt;right paren&gt;. &lt;F&gt; is &lt;left paren&gt; &lt;E&gt;       &lt;right paren&gt;. &lt;!-- аксиома --&gt; start &lt;E&gt;.</pre>	6	<pre>-- аксиома заключена -- в фигурные скобки T, T', { E }, E', F [ E : T E' ] [ E' : "+" T E' : @ ] [ T : F T' ] [ T' : "*" F T' : @ ] [ F : "n"   : "(" E ")" ] ]</pre>
7	<pre>' аксиома &lt;axiom &lt;E&gt;&gt; ' правила грамматики &lt;E &lt;T E'&gt;&gt; &lt;E' &lt;+ T E'&gt; &lt;&gt;&gt; &lt;T &lt;F T'&gt;&gt; &lt;T' &lt;* F T'&gt; &lt;&gt;&gt; &lt;F &lt;n&gt; &lt;( E )&gt;&gt;</pre>	8	<pre># ключевые слова # начинаются с кавычки  F -&gt; "n" 'or       "(" E ")" 'end T -&gt; F T1 'end T1 -&gt; "*" F T1 'or       'epsilon 'end 'axiom E -&gt; T E1 'end E1 -&gt; "+" T E1 'or       'epsilon 'end</pre>

4. Раскрутка генератора компиляторов путём подачи на вход грамматики входного языка, написанной на самом входном языке (пункт 1) и замены таблицы разбора, написанной вручную, на сгенерированную таблицу разбора.

Отметим, что парсер входного языка должен выдавать сообщения об обнаруженных ошибках, включающие координаты ошибки. Восстановление при ошибках, а также выдачу специфичных текстовых описаний ошибок реализовывать не нужно.

Кроме того, генератор компиляторов должен уметь обнаруживать следующие ошибки в грамматике:

- наличие нетерминального символа, не присутствующего в левой части ни одного правила;
- грамматика не относится к классу  $LL(1)$ -грамматик;
- не указана аксиома грамматики;
- указано более одной аксиомы грамматики;
- используется необъявленный символ или символ объявлен дважды (для вариантов входных языков, подразумевающих обязательное объявление терминальных и/или нетерминальных символов);

В качестве языка реализации используется тот же язык, что и в лабораторной работе 7.1.