Лабораторная работа №2. Ручное построение нисходящих синтаксических анализаторов. Вариант 11

Ян Должанский, M33381

5 ноября 2020 г.

0 Задание

Массив в Kotlin. Описание начинается ключевым словом “`var`”, далее идет имя массива, двоеточие, имя типа “`Array`”, далее в угловых скобках имя типа элементов массива.
Используйте один терминал для всех имен переменных и имен типов. Используйте один терминал для ключевого слова “`var`” (не несколько ‘v’, ‘a’, ‘r’).
Пример: `var x: Array<Int>;`

1 Разработка грамматики

Построим грамматику:

$$S \longrightarrow V\,I\,\texttt{:}\,A\texttt{<}G\texttt{>;}$$

$$V \longrightarrow \texttt{var}$$

$$I \longrightarrow \texttt{[\_a-zA-Z][\_0-9a-zA-Z]*}$$

$$A \longrightarrow \texttt{Array}$$

$$G \longrightarrow I$$

$$G \longrightarrow I\texttt{<}GG''\texttt{>}$$

$$G'' \longrightarrow \epsilon$$

$$G'' \longrightarrow \texttt{,}GG''$$

Нетерминал	Описание
S	Инструкция объявления массива
V	Ключевое слово `var`
I	Действительный идентификатор
A	Ключевое слово `Array`
G	Тип, возможно generic
G''	Аргументы generic, начиная со второго

Данная грамматика не принадлежит классу LL(1). Чтобы добиться этого, достаточно устранить правое ветвление.
Новая грамматика:

$$S \longrightarrow VS'$$

$$V \longrightarrow \texttt{var}$$

$$S' \longrightarrow I\,\texttt{:}\,A\texttt{<}G\texttt{>;}$$

$$I \longrightarrow \texttt{[\_a-zA-Z][\_0-9a-zA-Z]*}$$

$$A \longrightarrow \texttt{Array}$$

$$G \longrightarrow IG'$$

$$G' \longrightarrow \epsilon$$

$$G' \longrightarrow \texttt{<}GG''\texttt{>}$$

$$G'' \longrightarrow \epsilon$$

$$G'' \longrightarrow \texttt{,}GG''$$

Нетерминал	Описание
S	Инструкция объявления массива
V	Ключевое слово `var`
S'	Продолжение инструкции объявления массива после ключевого слова `var`
I	Действительный идентификатор
A	Ключевое слово `Array`
G	Тип, возможно generic
G'	Возможный аргумент generic
G''	Аргументы generic, начиная со второго

2 Построение лексического анализатора

Лексический анализатор реализован как класс `LexicalAnalyzer`. Возможные терминалы:

Терминал	Токен
`[_a-zA-Z]`	`ALPHA_OR_UNDERSCORE`
`[0-9]`	`DIGIT`
`:`	`COLON`
`;`	`SEMICOLON`
`,`	`COMMA`
`<`	`LEFT_ANGLE_BRACKET`
`>`	`RIGHT_ANGLE_BRACKET`
`$`	`END`

3 Построение синтаксического анализатора

Множества FIRST и FOLLOW для нетерминалов грамматики:

Нетерминал	FIRST	FOLLOW
S	'v'	$
V	'v'	'_', 'a', ..., 'z', 'A', ..., 'Z'
S'	'_', 'a', ..., 'z', 'A', ..., 'Z'	$
I	'_', 'a', ..., 'z', 'A', ..., 'Z'	'<', '>', ':', ','
A	'A'	'<'
G	'_', 'a', ..., 'z', 'A', ..., 'Z'	'>', ','
G'	ϵ, '<'	'>', ','
G''	ϵ, ','	'>'

Синтаксический анализатор реализован как класс `Parser`.

4 Визуализация дерева разбора

Ряд визуализаций деревьев разбора некоторых выражений:

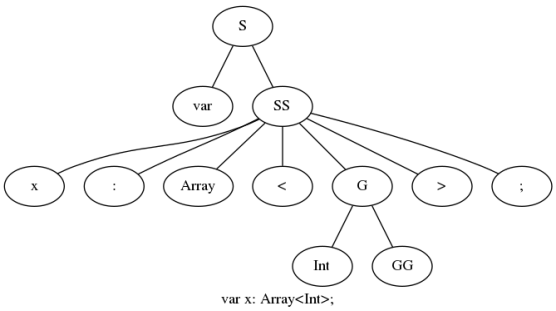

Рис. 1: Простое выражение

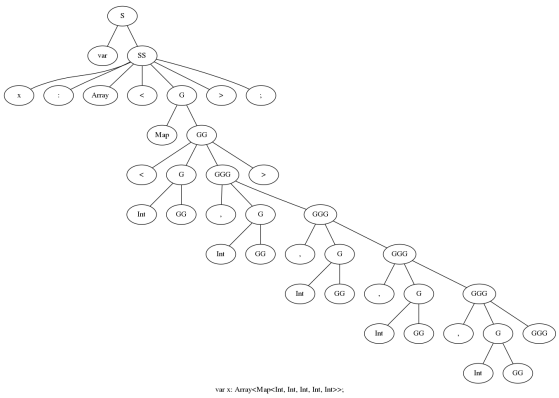

Рис. 2: Выражение с несколькими generic параметрами

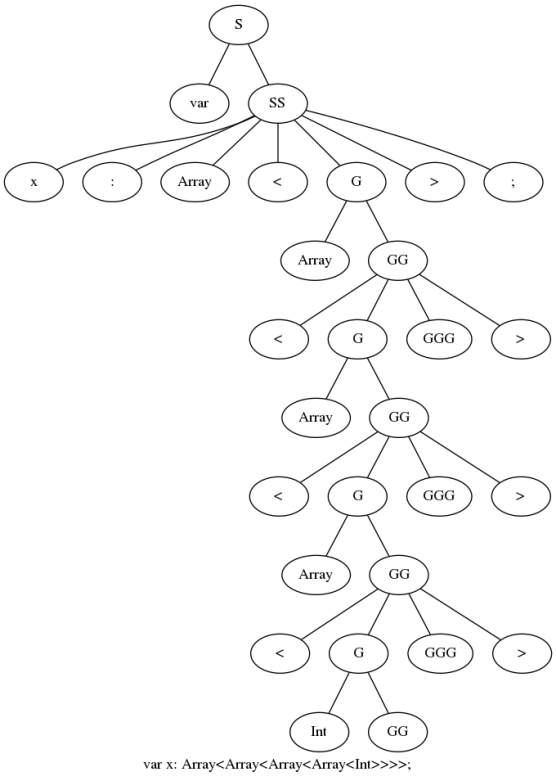

Рис. 3: Выражение со вложенностью generic

5 Подготовка наборов тестов

Тесты реализованы посредством фреймворка JUnit в проекте. Краткое описание каждого теста заложено в отображаемое имя каждого тестового метода.