Лабораторная работа №2. Ручное построение нисходящих синтаксических анализаторов

Султанов Мирзомансурхон M33341, @Ultimatereo

Февраль 2024

Вариант 9. Описание заголовка функции в Kotlin Заголовок функции в Kotlin. Заголовок начинается ключевым словом “fun”, далее идет имя функции, скобка, несколько описаний аргументов через запятую, затем может идти двоеточие и имя возвращаемого типа. Используйте один терминал для всех имен переменных. Используйте один терминал для ключевых слов fun и т. п. (не несколько ‘f’, ‘u’, ‘n’). 9

Пример: fun printSum(a: Int, b: Int): Unit

1 Разработка грамматики

Разработайте контестно-свободную грамматику для языка, описанного в условии вашего варианта. Сначала разработайте грамматику, исходя из структуры языка, чтобы она максимально близко соответствовала интуитивным представлениям о построении слов из языка. Затем, при необходимости, устраните левую рекурсию и/или правое ветвление. В отчете приведите исходную и преобразованную грамматику, опишите смысл всех нетерминалов.

- Токены
  - fun
  - modifier(inline, infix, public, private, protected, internal, open)
  - name - name of functions, variables, classes etc
  - value
  - (
  -)
  - :
  - ,
  - <
  - >

  - ->
  - .
  - ?
  - *

- Нетерминалы
  - S - start
  - M - modifier
  - G - generic
  - GC - generic continuation
  - FN - function name
  - FNC - function name continuation
  - F(T)A - function arguments
  - F(T)AC - function arguments continuation
  - RT - return type
  - D - default
  - T - type
  - ST - simple type
  - FT - function type
  - N? - nullable?
  - GP - generic parameters
  - GP! - generic parameters (couldn't be empty)
  - GPC - generic parameters continuation
  - GT - generic type

- Правила (полная версия)
  - S → M S
  - S → fun G FN (FA) RT
  - M → inline | infix | public | private | protected | internal | open
  - G → ”
  - G → < name GC >
  - GC → , name GC
  - GC → ”
  - FN → name FNC
  - FNC → GP! N? . name
  - FNC → . name

  - FNC → ϵ
  - FA → ϵ
  - FA → name : T D FAC
  - FAC → , name : T D FAC
  - FAC → ϵ
  - RT → ϵ
  - RT → : T
  - D → = value
  - D → ϵ
  - T → ST
  - T → FT
  - ST → name GP N?
  - FT → (FTA) -> T
  - N? → ?
  - N? → ϵ
  - GP → ϵ
  - GP → GP!
  - GP! → < GT GPC >
  - GPC → ”
  - GPC → , GT
  - GT → *
  - GT → T
  - FTA → ϵ
  - FTA → name : T FTAC
  - FTAC → , name : T FTAC
  - FTAC → ϵ
- Правила (простая версия)
  - S → M S
  - S → fun FN (FA) RT
  - M → inline | infix | public | private | protected | internal | open
  - FN → name FNC
  - FA → ϵ
  - FA → name : T FAC
  - FAC → , name : T FAC

– FAC $\to \epsilon$
– RT $\to \epsilon$
– RT $\to$: T
– T $\to$ name

Заметим, что в построенной грамматике нет ни левой рекурсии, ни правого ветвления, так что можем переходить к следующему шагу.

2 Построение лексического анализатора

Лексический анализатор должен получать на вход строку и выдавать последовательность терминалов (токенов). Пробелы и переводы строк должны игнорироваться.

3 Построение синтаксического анализатора

Постройте множества FIRST и FOLLOW для нетерминалов вашей грамматики. Затем напишите синтаксический анализатор с использованием рекурсивного спуска

Нетерминал	FIRST	FOLLOW
S	fun, modifier	$
M	modifier	modifier, fun
G	ϵ, '<'	name
GC	',', ϵ	'>'
FN	name	'('
FNC	ϵ, '<', '.'	'('
FA	ϵ, name	')'
FAC	',', ϵ	')'
FTA	ϵ, name	')'
FTAC	',',ϵ	')'
RT	ϵ, ':'	$
D	ϵ, '='	',', ')'
T	name, '('	'=', ',', ')', '>', '?', '.' $
ST	name	'=', ',', ')', '>', '?', '.' $
FT	'('	'=', ',', ')', '>', '?', '.' $
N?	'?', ϵ	'=', ',', ')', '>', '?', '.' $
GP	ϵ, '<'	'=', ',', ')', '>', '?', '.' $
GP!	'<'	'=', ',', ')', '>', '?', '.' $
GPC	',', ϵ	'=', ',', ')', '>', '?', '.' $
GT	name, '(', '*'	'=', ',', ')', '>', '?', '.' $

Таблица 1: FIRST и FOLLOW множества для нетерминалов

Замечу так же, что в данной грамматике выполняется необходимое и достаточное условие LL(1) грамматики на FIRST и FOLLOW.

4 Визуализация дерева разбора

Для изучения результата разработайте систему визуализации дерева разбора. Рекомендуется использовать систему GraphViz (https://graphviz.org/)

5 Подготовка набора тестов для тестирования