***Лабораторная работа №2. Ручное построение нисходящих синтаксических анализаторов***

Шовкопляс Григорий M3339

28 апреля 2016

***Задание*:** Описание переменных в Паскале ― блок описания переменных в Паскале. Блок начинается ключевым словом “**var**”, далее идет несколько описаний переменных. Описание переменных содержит имена переменных через запятую, затем двоеточие, затем имя типа. Достаточно рассматривать только примитивные типы (массивы, записи и т. п. не требуется). Используйте один терминал для всех имен переменных и имен типов. Используйте один терминал для ключевого слова **var** (не три ‘**v**’, ‘**a**’, ‘**r**’). Пример: **var a, b: integer; c: integer;**

1. ***Грамматика***

Построим грамматику по описанию задания (n – имя переменной в Паскале)

S ⟶ V T

V ⟶ **var**

T ⟶ L:n;|L:n;T

L ⟶ n|n,L

|  |  |
| --- | --- |
| **Нетерминал** | **Описание** |
| S | Описание переменных в Паскале |
| V | Служебное слово **var** |
| T | Список объявлений переменных |
| L | Перечисление имен переменных |

В грамматике есть правое ветвление — устраним его

S ⟶ V T

V ⟶ **var**

T ⟶ L:n;T’

T’⟶ T|Ɛ

L ⟶ nL’

L’⟶ ,L|Ɛ

|  |  |
| --- | --- |
| **Нетерминал** | **Описание** |
| S | Описание переменных в Паскале |
| V | Служебное слово **var** |
| T | Непустой список объявлений переменных |
| L | Перечисление имен переменных |
| T’ | Список объявлений переменных |
| L’ | Продолжение перечислений имен переменных |

1. **Построение лексического анализатора**

В грамматике 6 терминалов — **var**, n(универсальный терминал для имени переменной или типа), , ‘***,***’, ‘***;***’, ‘*:*’.

Заведем класс **Token** для хранения терминалов, не забудем про конец строки.

public enum Token {  
 *VAR*, *CHARACTER*, *DIGIT*, *COMMA*, *SEMICOLON*, *VARTYPE*;  
}

|  |  |
| --- | --- |
| **Терминал** | **Токен** |
| **var** | VAR |
| n | VARTYPE |
| , | COMMA |
| ; | SEMICOLON |
| : | COLON |
| $ | END |

1. **Построение синтаксического анализатора**

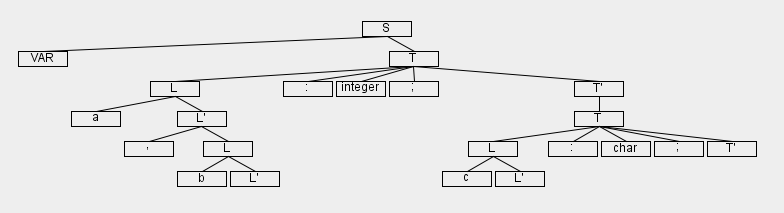
Построим множества **FIRST** и **FOLLOW** для нетерминалов грамматики

|  |  |  |
| --- | --- | --- |
| **Нетерминал** | **FIRST** | **FOLLOW** |
| S | Var | $ |
| V | Var | n |
| T | n | $ |
| L | n | ‘:’ |
| T’ | n, Ɛ | ‘;’ |
| L’ | ‘,’, Ɛ | ‘:’ |

1. **Визуализатор**

Мной были реализованы два варианта визуализатора:

А) Вывод в выходной файл дерева в порядке обхода поиска в глубину с отступами, зависящими от глубины.

Б) При помощи сторонней библиотеки **org.StructureGraphic.v1** Рассмотрим на примере “**var a, b: integer; c: char;**”

1. **Тестирование**

Программа была вручную протестирована на исчерпывающем наборе тестов.

|  |  |
| --- | --- |
| **Тест** | **Описание** |
| var va: vars; | Проверка обработки переменных имеющих вложенные символы со служебным словом **var** |
| VaR VeRy: sTrAnGe; | Проверка поддержки разных регистров букв |
| var a, b: integer; с: char; | Ручной тест для нескольких типов |
| var a1A2: Int64; | Проверка поддержки цифр |
| Abacaba | Рандомная ручная строка |
| var a,integer; | Неправильный разделительный символ в нетерминале **T** |
| var 1: integer; | Неправильный первый символ в названии переменной |
| var b | Отсутствие продолжения |
|  | Пустая строка |
| var b; | Преждевременный конец строки |
| var a,,b:int; | Переменная с нулевым именем |
| var a, b: integer | Отсутствие ; в конце |
| var a:b:c:d; | Игры с разделителем |
| var a, b: integer, vars: char; | Игры с разделителем |
| var a, b: integer: vars: char; | Игры с разделителем |

Также программа была проверена на нескольких больших специально сгенерированных тестах.