МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра вычислительной техники



**ОТЧЁТ**

**по лабораторной работе №3**

*Синтаксис языков программирования. Формальные грамматики*

**по дисциплине:** *Теория формальных языков и компиляторов*

**Вариант: 42411412**

Выполнила:Проверил:

Студентка гр. АВТ-709, АВТФ *Доцент*

*Андерсон Д. В. Малявко А. А.*

«\_\_\_» \_\_\_\_\_\_ 2020 г.«\_\_\_» \_\_\_\_\_\_ 2020 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2020

**Оглавление**

[**Введение** 3](#_Toc37171134)

[**Вариант** 4](#_Toc37171135)

[**Ход работы** 5](#_Toc37171136)

[*1.* *Доработки* 5](#_Toc37171137)

[*2.* *Работа лексических автоматов по разбору заданного фрагмента* 9](#_Toc37171138)

[*3.* *Описание метаязыка формальных грамматик* 9](#_Toc37171139)

[*4.* *Свойства грамматик из примера* 10](#_Toc37171140)

[*5.* *Разработка грамматики программируемого языка* 12](#_Toc37171141)

[**Вывод** 14](#_Toc37171142)

## **Введение**

*Цели 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.

*Порядок выполнения работы:*

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**

|  |  |
| --- | --- |
| Идентификаторы | $<пЦ><пБ>  ($1ad, $74c, $0B…) |
| Константы | целые по основаниям 2,8 и 10;  вещественные;  символьные |
| Объявление примитивных типов | Целое – card[inal][\_u]  Вещественное – double  Символьное – litera |
| Оператор присваивания | <И> **=** <В>; |
| Условный оператор | **at** <ЛВ> **do** <ОБ> [**else do** <ОБ>] |
| Оператор цикла | **exec** <ОБ> **with** <И> **from** <К> **to** <К>[**step** <К>] |
| Оператор переключателя | **select** <B> **case (**<К>**) <**ОБ**> [break;]**…**[case()** <ОБ>**] end** |
| Формат псевдокода | Пентады <М><Код><Оп><Оп><Р> |

Обозначения:

**[...]** – необязательная часть конструкции;

**…** – предшествующая часть конструкции повторяется произвольное количество раз;

< > – описание сокращений:

<пБ>|<пЦ> – непустая последовательность букв | непустая последовательность цифр

<И> – Идентификатор (имя переменной / объекта); <К> – Константа;

<В> – произвольное Выражение;

<ЛВ> – Логическое Выражение;

<ОБ> – Оператор или Блок;

<О> – одиночный оператор;

<ОП> – оператор присваивания;

<Код> – поле кода операции;

<М> – поле метки;

<Оп> – поле наименования операнда;

<Р> – поле наименования результата

## **Ход работы**

## *Доработки*

В данной лабораторной работе были доработаны правила лексики.

Рисунок 1. Обновленные правила лексики

Так как в прошлой лабораторной работе приведенный пример кода был слишком простой, содержал ошибки, и не содержал всех элементов языка, было принято решение дополнить пример кода.

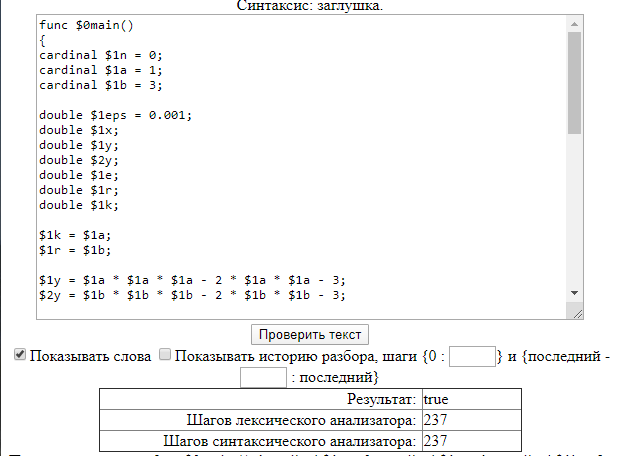
Старый код:

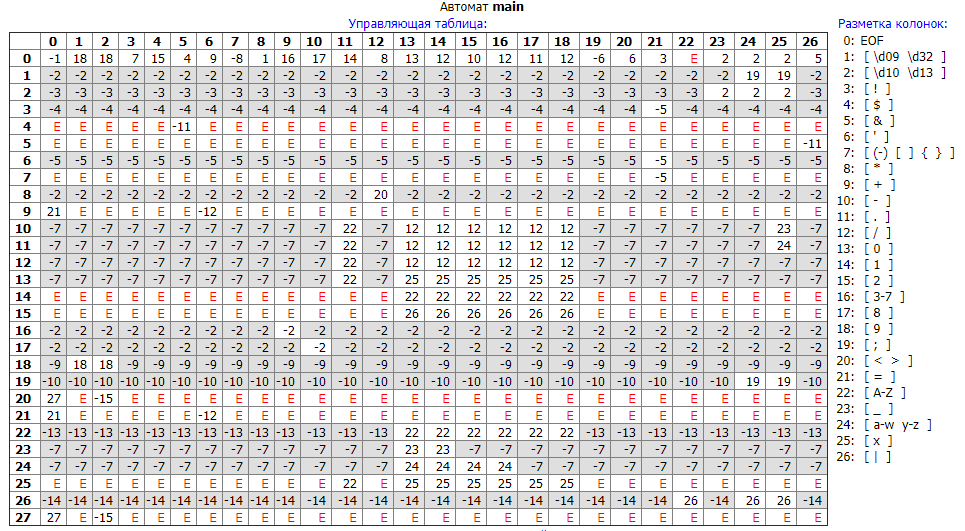
|  |
| --- |
| func power (double $1x)  {  card $0i;  double $3v;  const $4q = 5;  $3v = 1;  exec  $3v = $3v \* $1x  from 1 to 10  }  func main()  {  double $9P;  $9P = power(2.125);  select ($9P)  case(1) speak('Good');  end;  } |

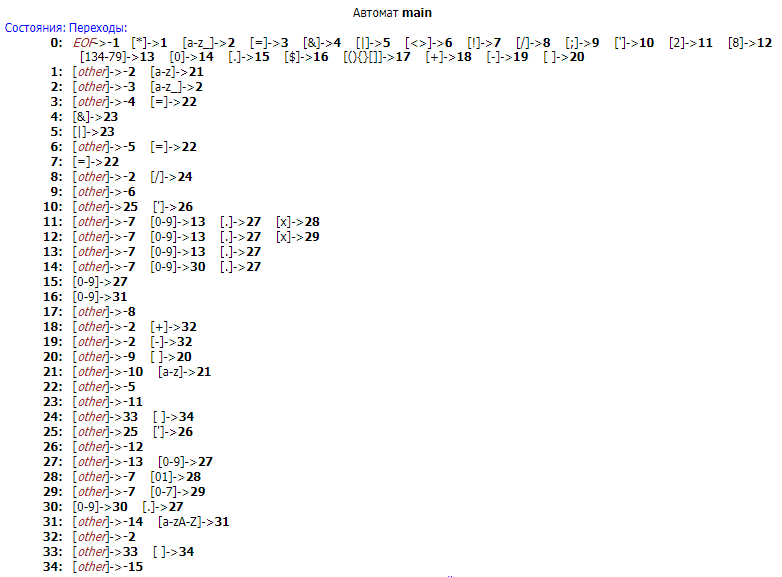
В качестве дополнительного примера была взята задача поиска корня уравнения f(x) = x3 – 2x2 – 3 = 0 на отрезке [1; 3] с заданной точностью eps = 0.001 методом деления отрезка пополам.

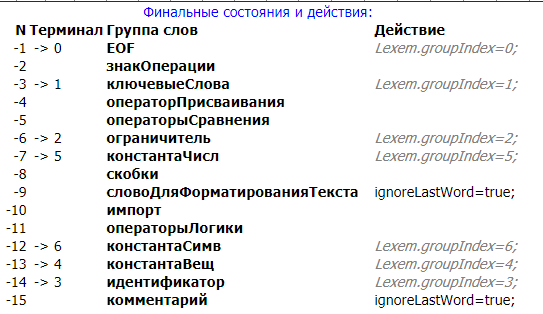
Новый код:

|  |
| --- |
| func $0main()  {  cardinal $1n = 0;  cardinal $1a = 1;  cardinal $1b = 3;  double $1eps = 0.001;  double $1x;  double $1y;  double $2y;  double $1e;  double $1r;  double $1k;  $1k = $1a;  $1r = $1b;  $1y = $1a \* $1a \* $1a – 2 \* $1a \* $1a – 3;  $2y = $1b \* $1b \* $1b – 2 \* $1b \* $1b – 3;  at ($1y \* $2y > 0) do  speak(‘No roots’);  exec  {  ++$1n;  $1x = ($1a + $1b)/2;  $1y = $1a \* $1a \* $1a – 2 \* $1a \* $1a – 3;  $2y = $1x \* $1x \* $1x – 2 \* $1x \* $1x – 3;  at ($1y \* $2y > 0) do  $1a = $1x;  else do  $1b = $1x;  } with ($1b - $1a) from 2 to eps;  $1x = ($1a + $1b)/2;  speak(‘1 - Answer’);  speak(‘2 – Steps’);  card $1z;  listen($1z);  select ($1z)  case 1:  speak(‘The root is: ‘);  speak($1k);  case 2:  speak(‘Steps: ’);  speak($1n);  } |

Рисунок 2. Проверка обновленного кода

Рисунок 3. Обновленный табличный КА

 Рисунок 4. Обновленный графовый КА

Рисунок 5. Обновленные финальные состояния

## *Работа лексических автоматов по разбору заданного фрагмента*

Заданный фрагмент: *$9P = power(2.125);*

Конечный автомат, заданный таблицей:

*идентификатор ($9P);*

*словоДляФорматированияТекста (“ ”);*

*операцияПрисваивания (“=”);*

*словоДляФорматированияТекста (“ ”);*

*ключевыеСлова (power);*

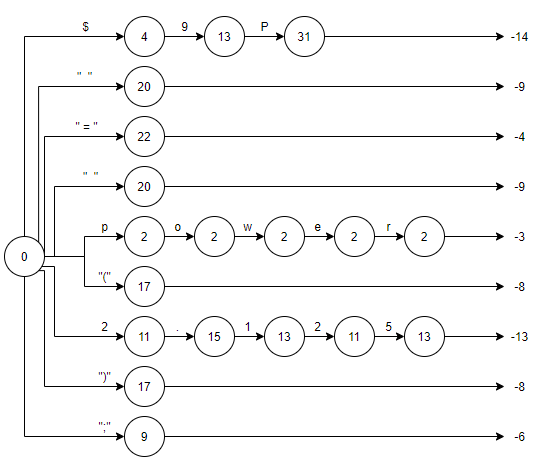
*скобки (“(”);*

*константаВещ (2.125);*

*скобки (“)”);*

*ограничитель (“;”).*

Конечный автомат, заданный графом:

Рисунок 6. КА, заданный графом

## *Описание метаязыка формальных грамматик*

Формальной грамматикой *G* называется совокупность , где:

алфавит терминальных символов;

алфавит нетерминальных символов;

начальный терминальный символ;

система правил подстановки.

Алфавит терминальных символов , есть конечное множество всех слов языка, порождаемого данной грамматикой. Понятие «терминальный» в данном случае обозначает неразложимость таких символов с точки зрения синтаксических правил.

Алфавит нетерминальных символов 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 нетерминальный символ S есть один из нетерминальных символов. Этим символом обычно обозначается наиболее общая синтаксическая конструкция, например: <правильная программа>.

Система правил подстановки P (система порождающих правил или продукций) есть конечное множество пар цепочек вида α : β, причем цепочка α (левая часть правила) должна содержать хотя бы один нетерминальный символ.

Для языка ADV используются контекстно-свободные грамматики, то есть грамматики, у которых левая часть каждого порождающего правила состоит из одного нетерминального символа. Также существуют общие, контекстно-зависимые и регулярные грамматики. Использование контекстно-свободных грамматик позволяет заменять нетерминал из левой части правила на правую часть любого из правил в любой цепочке, содержащей этот нетерминал.

## *Свойства грамматик из примера*

Пример из файла 3LL1.xml.

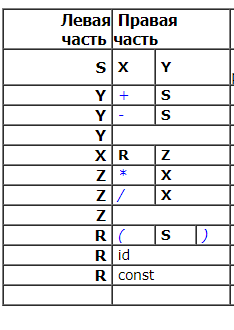
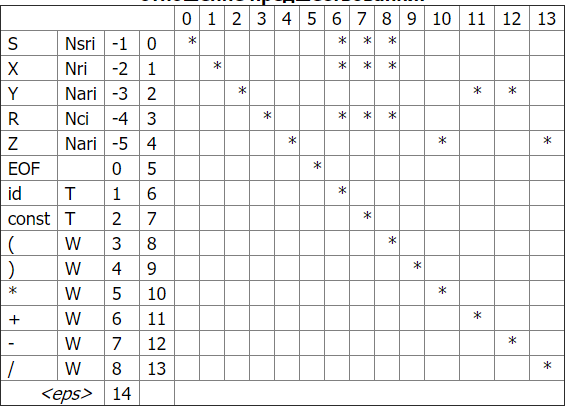
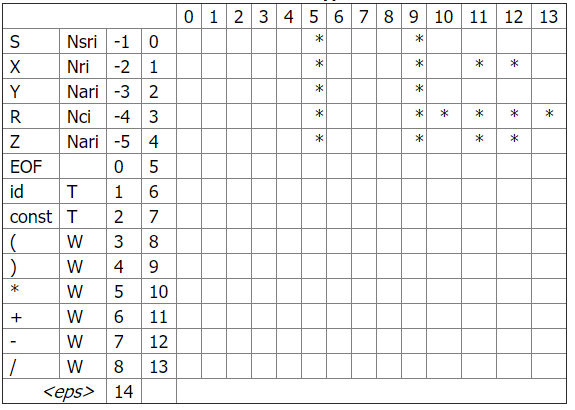
**

Рисунок 6. Правила грамматики языка

**

Рисунок 7. Множество выбора правил языка

Рисунок 8. Множество предшественников языка

Рисунок 9. Множество последователей языка

В данной системе правил не содержится недостижимых и бесплодных грамматик, однако, содержится два аннулируемых правила (Z и Y). Множества выбора в данной грамматике для всех нетерминалов не пересекаются. Язык принадлежит классу LL1 и пригоден для корректного построения и работы синтаксического акцептора.

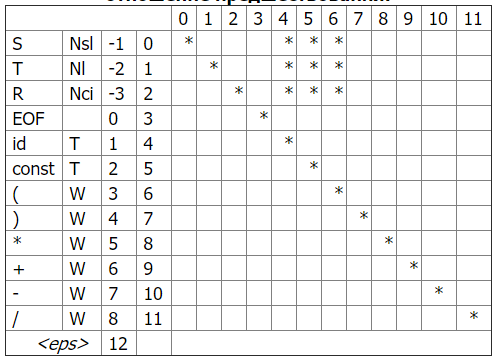
Пример из файла 4SLR1grammarArithmExpr.xml.

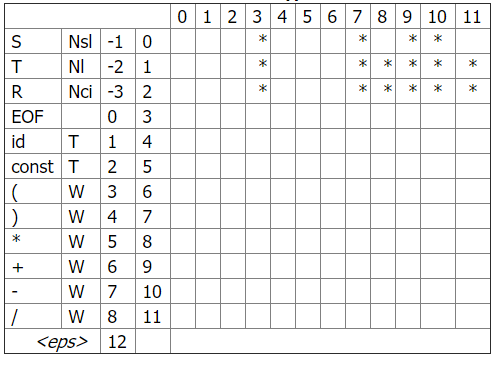


Рисунок10. Правила грамматики языка



Рисунок 11. Множество выбора правил языка

Рисунок 12. Множество предшественников языка

**Рисунок 13. Множество последователей языка

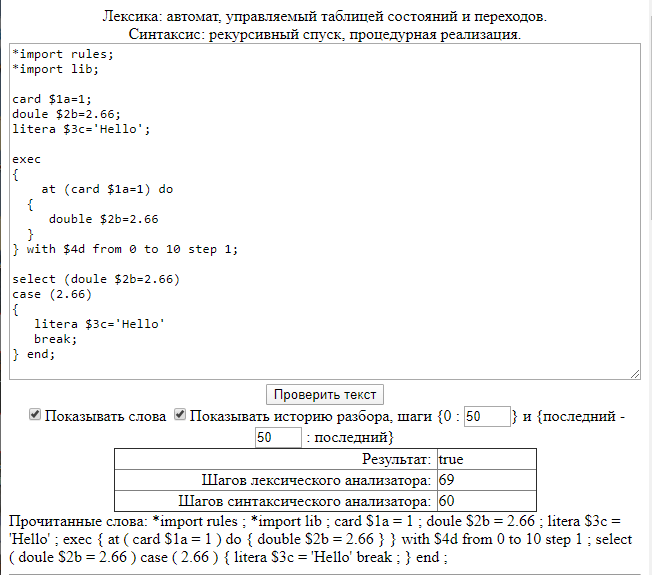
Во множестве выбора данной грамматике содержатся пересечения, поэтому грамматика не является LL1, ее работа будет некорректна.

## *Разработка грамматики программируемого языка*

В данной лабораторной работе была частично разработана грамматика для языка ADV, в соответствии с заданным вариантом.

Рисунок 14. Часть разработанной грамматики

Для проверки правильности разработанной грамматики была написана небольшая программа.

Рисунок 15. Результат работы анализатора

## **Вывод**

В результате выполнения лабораторной работы были доработаны правила лексики языка программирования. Были изучены графовый и табличный способ определения КА, а также построена работа КА в соответствии с заданным фрагментом. Были изучены основные понятия метаязыка формальных грамматик. Была разработана часть грамматики программируемого языка в соответствии с вариантом.