МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО 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*

Выполнил:Проверил: *Доцент*

Студент гр. АВТ-709, АВТФ *Малявко А. А.*

*Гунгер А. К.*

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

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

(подпись) (подпись)

Новосибирск 2020

**Цель 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.

**Краткое описание метаязыка формальных грамматик**

Формально грамматика G определяется как четвёрка G (VT, VN, P, S), где:

* VT – множество терминальных символов, или алфавит терминальных символов;
* VN – множество нетерминальных символов, или алфавит нетерминальных символов;
* Р – множество правил (продукций) грамматики вида α → β, где α ∈ (VN ∪ VT) +, β ∈ (VN ∪ VT)\*;
* S – целевой (начальный) символ грамматики, S ∈ VN.

Алфавиты терминальных и нетерминальных символов грамматики не пересекаются: VN ∩ VT = ∅. Это значит, что каждый символ в грамматике может быть либо терминальным, 4 либо нетерминальным, но не может быть терминальным и нетерминальным одновременно. Целевой символ грамматики – это всегда нетерминальный символ. Множество V = VN ∪ VT называют полным алфавитом грамматики G. Множество терминальных символов VT содержит символы, которые входят в алфавит языка, порождаемого грамматикой. Как правило, символы из множества VT встречаются только в цепочках правых частей правил. Множество нетерминальных символов VN содержит символы, которые определяют слова, понятия, конструкции языка. Каждый символ этого множества может встречаться в цепочках как левой, так и правой частей правил грамматики. Во множестве правил грамматики может быть несколько правил, имеющих одинаковые левые части вида: α → β1, α → β2, …, α → βn. Эти правила можно объединить вместе и записать в виде: α → β1 | β2 | … | βn. Одной строке в такой записи соответствует сразу n правил. Такую форму записи правил грамматики называют формой Бэкуса-Наура 1 . Форма Бэкуса-Наура (англ. Backus-Naur Form (BNF)), как правило, предусматривает также, что нетерминальные символы берутся в угловые скобки: < >. Пример грамматики, которая определяет язык целых десятичных чисел со знаком в форме Бэкуса-Наура: G ({0, 1, 2, 3, 4, 5, 6, 7, 8, 9, –, +}, {, , }, Р, ) Р: → | + | – → | → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 Рассмотрим составляющие элементы грамматики G:

* множество терминальных символов VT содержит двенадцать элементов: десять десятичных цифр и два знака;
* множество нетерминальных символов VN содержит три элемента: символы , и ;
* множество правил содержит 15 правил, которые записаны в три строки (то есть имеется только три различные левые части правил);

целевым символом грамматики является символ . Та же самая грамматика для языка целых десятичных чисел со знаком, в которой нетерминальные символы обозначены большими латинскими буквами (далее это будет часто применяться в примерах): G' ({0, 1, 2, 3, 4, 5, 6, 7, 8, 9, –, +}, {S, T, F}, Р, S) Р: S → Т | +Т | –Т Т → F | TF F → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 Здесь изменилось только множество нетерминальных символов. Теперь VN = {S, T, F}. Язык, заданный грамматикой, не изменился – грамматики G и G' эквивалентны. Особенность рассмотренных выше формальных грамматик в том, что они позволяют определить бесконечное множество цепочек языка с помощью конечного набора правил.

**Свойства грамматик из примеров и символов этих грамматик**

**Свойства грамматик**

1. **Рекурсивность**

Нетерминальный символ Х называется рекурсивным, если из него могут быть выведены цепочки, содержащие сам этот символ, т. е.

Х ⇒ μ Х η,

где μ и η – произвольные цепочки символов.

Грамматика называется рекурсивной, если рекурсивен хотя бы один нетерминальный символ, и нерекурсивной в противном случае.

**Примеры рекурсивности из грамматик Ga1 и Ga2:**

S: X Y

Y: +S

S: S + T

S: S – T

1. **Однозначность**

Грамматика называется однозначной, если любое правильное предложение порождаемого ею языка имеет единственное дерево грамматического разбора, и неоднозначной в противном случае.

**Свойства символов грамматик**

1. **Аннулируемость**

Нетерминальный символ называется аннулируемым, если из него может быть выведена пустая цепочка символов. В противном случае нетерминальный символ называется неаннулируемым.

1. **Недостижимость**

Символ называется недостижимым, если он не появляется ни в одной цепочке символов, выводимой из начального нетерминала грамматики.

1. **Бесплодность**

Нетерминальный символ называется бесплодным, если из него не может быть выведена ни одна цепочка, состоящая только из терминальных символов.

**Матричное представление отношений предшествования и последования для символов изученных грамматик**



Рис. 1. Отношение предшествования для грамматики из примера 4

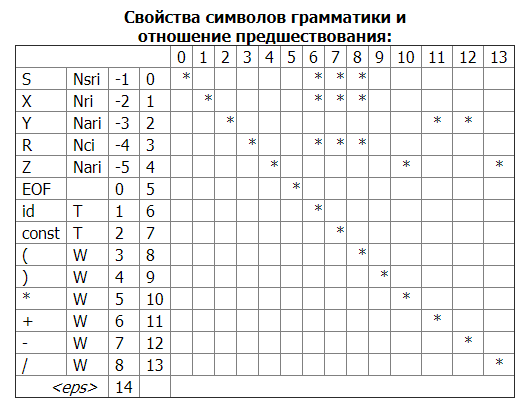


Рис. 2. Отношение предшествования для грамматики из примера 3

Предшественником некоторого символа Х называется символ, с которого начинается цепочка, выводимая из Х. Считается, что любой символ является предшественником самого себя, т.е. учитываются выводы длины 0.



Рис. 3. Отношение последования для грамматики из примера 4



Рис. 4. Отношение последования для грамматики из примера 3

Символ Y называется последователем символа Х, если хотя бы в одной цепочке η, выводимой из начального нетерминала грамматики, символ Y непосредственно следует за Х.

**Разработанная грамматика**

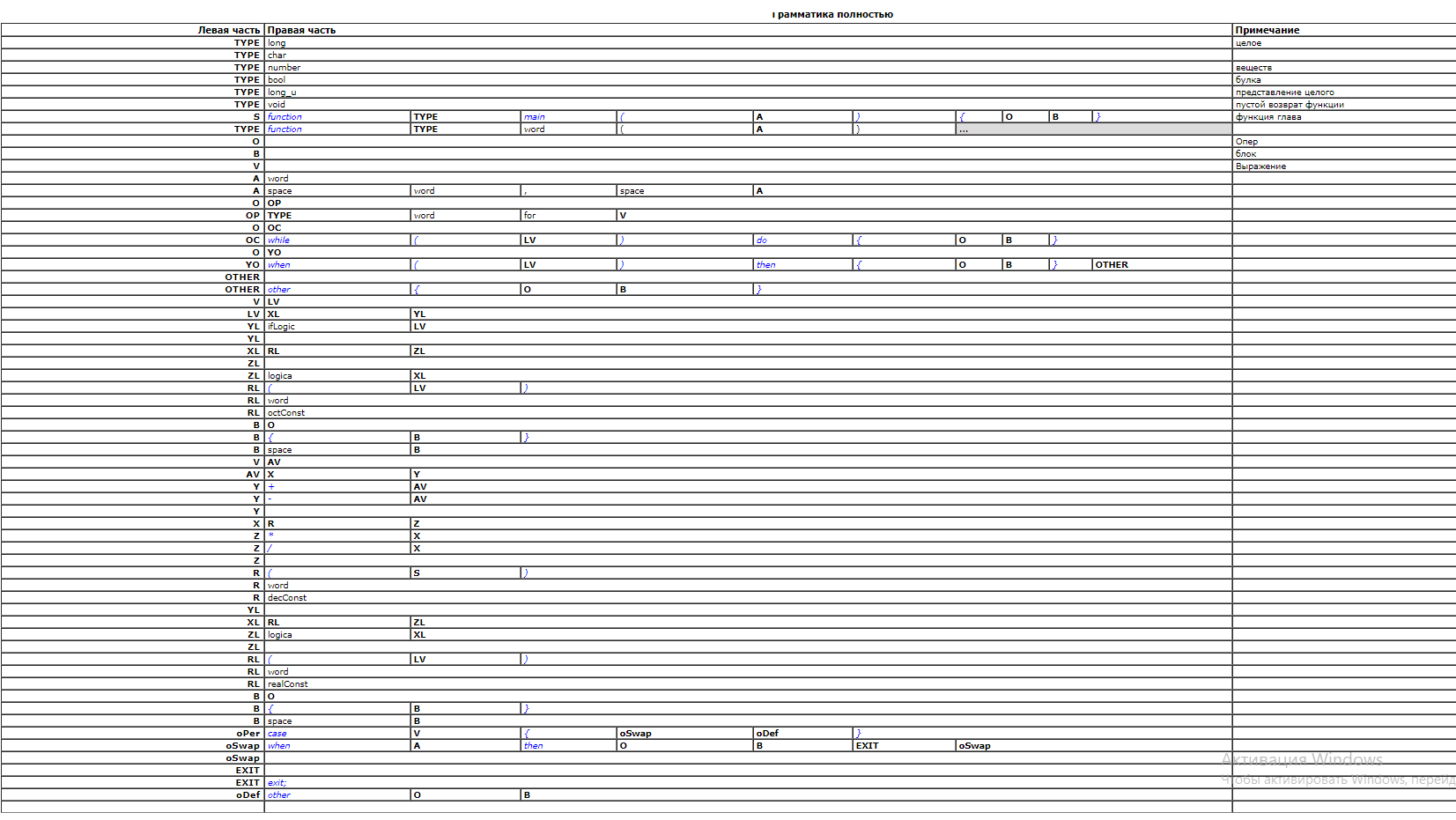


Рис. 5. Грамматика заданного языка

**Описание разработанной грамматики:**

Начальный нетерминальный символ – S. В качестве S могут быть использована либо функция main, либо функция с произвольным названием.

Объявление функций начинается с ключевого слова function.

Далее описана грамматика для оператора присваивания, цикла, условного оператора и оператора переключателя. Во всех операторах могут присутствовать выражения (V). Выражения представляют из себя константы, арифметические выражения или логические выражения.

Все основные блоки могут выполнятся, только находясь внутри структуры function.

Общая структура рабочей программы должна выглядеть примерно так:

function void main(){

$1sum(12.4)

}

function int $1sum(number $1arg){

return $1arg\*3;

}

Пока корректно работает только один оператор в блоке операторов.

В грамматике описаны типы данных, которые можно использовать в программе. В лексику еще не внесены слова для логического типа данных.

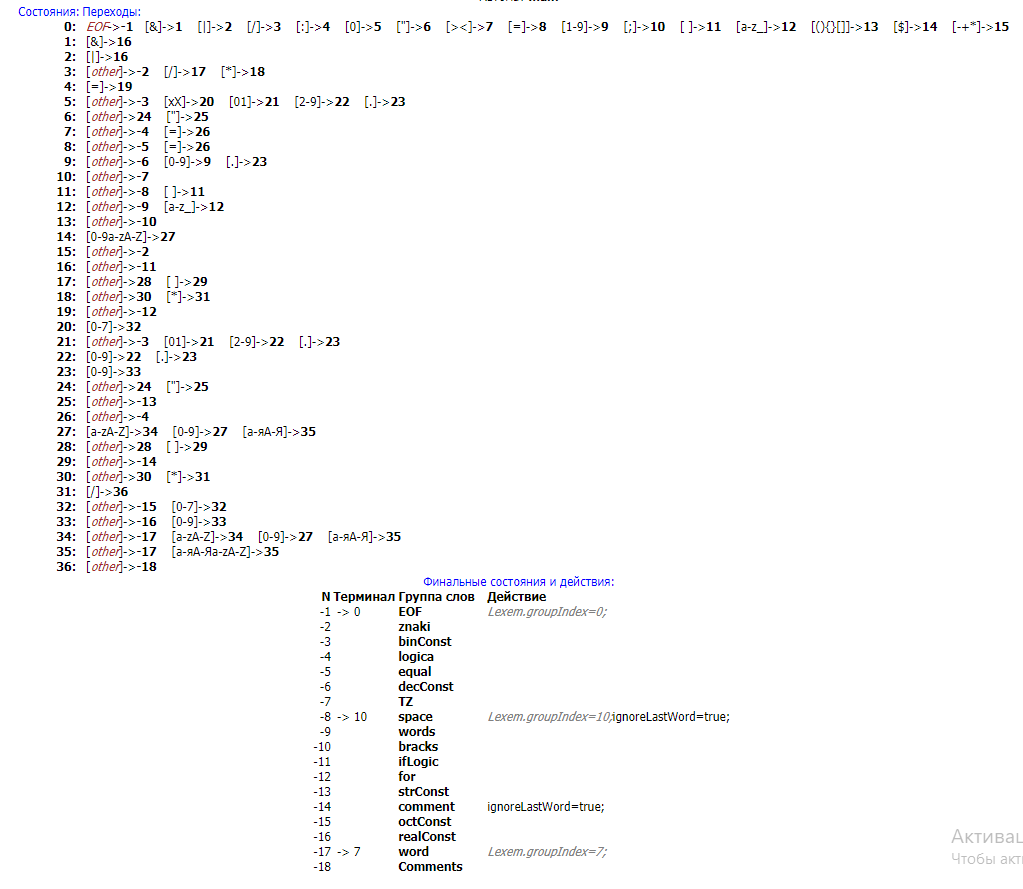
**Выводы**

Изучены основные понятия метаязыка формальных грамматик, свойства грамматик и нетерминальных символов, рекурсивности и однозначности грамматик, недостижимости, бесплодности, аннулируемости и рекурсивности нетерминальных символов, отношений предшествования и последования между символами, освоены технологии разработки формальных грамматик.

**Защита лаб. работы №2**

Истории работы графового и табличного лексических анализаторов по обработке такой последовательности символов: **$12c[$0k+1] := $1b / 10;**

**Графовый**



0: [$] → 14

14: [1] → 27

27: [2] → 27

13: [c] → 34

34: [[] → -17 word

0: [[] → 13

13: [$] → -10 – bracks

0: [$] → 14

14: [0] → 27

27: [k] → 34

34: [+] → -17 word

0: [+] → 15

15: [1] → -2 znaki

0: [1] → 9

9: []] → -6 decConst

0: []] → 13

13: [ ] → -10 – bracks

0: [ ] → 11

11: [:] → -8 space

0: [:] → 4

4: [=] → 19

19: [ ] → -12 for

0: [ ] → 11

11: [$] → -8 space

0: [$] → 27

27: [1] → 27

27: [b] → 34

34: [ ] → -17 word

0: [ ] → 11

11: [/] → -8 space

0: [/] → 3

3: [ ] → -2 znaki

0: [ ] → 11

11: [1] → -8 space

0: [1] → 9

9: [0] → 9

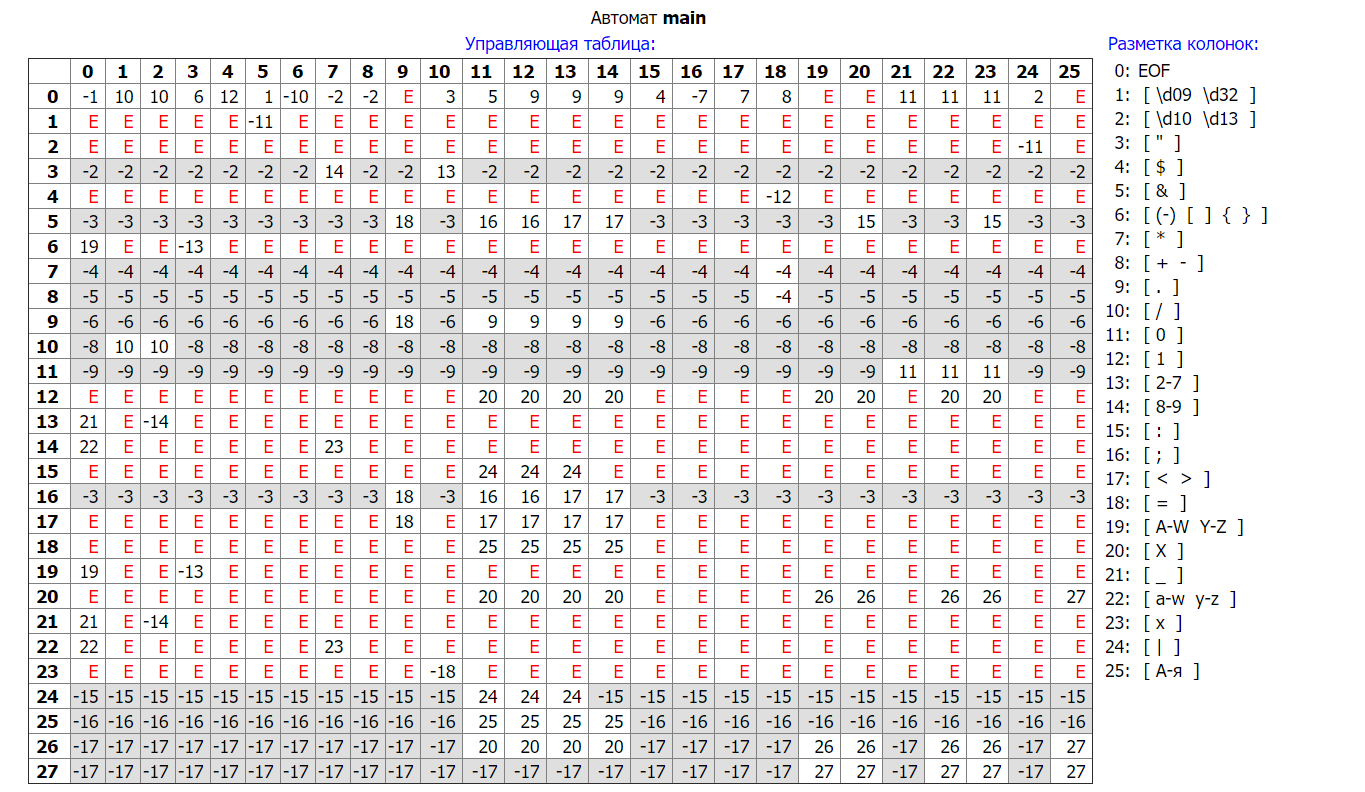
9: [;] → -6 decConst

0: [;] → 10

10: [\n] → -7 – TZ

**Табличный**

$12c[$0k+1] := $1b / 10;



0: [$] (4) → 12

12: [1] (12) → 20

20: [2] (13) → 20

20: [c] (22) → 26

26: [[] (6) → -17 word

0: [[] (6) → -10 bracks

0: [$] (4) → 12

12: [0] (11) → 20

20: [k] (22) → 26

26: [+] (6) → -17 word

0: [+] (8) → -2 znaki

0: [1] (12) → 9

9: []] (6) → -6 decConst

0: []] (6) → -10 bracks

0: [ ] (1) → 10

10: [:] (15) → -8 space

0: [:] (15) → 4

4: [ =] (18) → -12 - for

0: [ ] (1) → 10

10: [$] (4) → -8 space

0: [$] (4) → 12

12: [1] (12) → 20

20: [b ] (22) → 26

26: [ ] (1) → -17 word

0: [ ] (1) → 10

10: [/] (4) → -8 space

0: [/] (10) → 3

3: [ ] (1) → -2 – znaki

0: [ ] (1) → 10

10: [1] (12) → -8 space

0: [1] (12) → 9

9: [0] (11) → 9

9: [;] (16) → -6 decConst

0: [;] (16) → -7 TZ