МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

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

Кафедра вычислительной техники



**ОТЧЕТ**

**По лабораторной работе №3**

**Синтаксис языков программирования. Формальные грамматики**

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

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| Студент гр. АВТ-709, АВТФ | *к.т.н., доцент* |
| *Убушеев Т. О.* | *Малявко А.А.* |
| «\_\_\_» \_\_\_\_\_\_ 2020 г. | «\_\_\_» \_\_\_\_\_\_ 2020 г. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

г. 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.

# Постановка задачи

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.

**Описание варианта:**

1. Идентификатор:

<бБ><пЦ><бБ>

Примеры: *d23U, N0Nq, x15y.*

1. Константы:

целые по основаниям 4,8 и 10 (*4x1, 8x5, 456*);

вещественные (*123.456*);

строковые (*'abc', "ABC").*

1. Объявления примитивных типов (целый, вещественный, символьный):

long[\_u]

number

char

Примеры:

*long l07g* – целый;

*number n0n23r* – вещественный;

*char c4r* – символьный.

1. Оператор присваивания:

<И> = <В>;

Примеры:

*n0n = 123.321;*

*l07g = 12 + 34;*

1. Условный оператор

if (<ЛВ>) <ОБ> [ ifnot <ОБ> ] fi

Пример:

*if (l0w < h1g)*

*{*

*long i9t = 0;*

*i9t = 23;*

*}*

*fi*

*if (i9t <= 100) i9t = 100;*

*ifnot i9t = 0; fi*

1. Оператор цикла:

loop <ОБ> until <ЛВ>

Пример:

*loop d123s = d123s + 100; until d123s <= 1000;*

1. Оператор переключателя:

switch <В> { case <К> : <ОБ> [gout;] …[ case () <ОБ> ] }

Пример:

*switch i9t { case 1 : i9t = 0; gout; case 2 : i9t = 3; }*

*switch i9t { case 1 : i9t =0; gout; case 2 : i9t = 3; gout; case 3 : {*

*t3p = i9t;*

*i9t = i92t;*

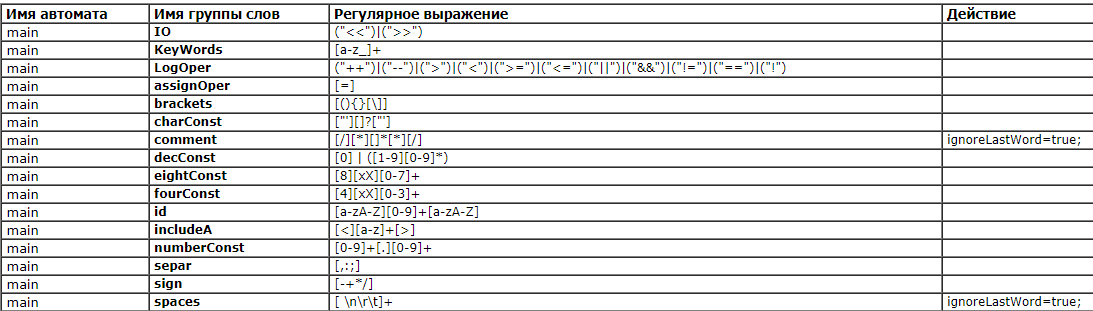
*i92t = t3p;*

*}}*

# Ход работы

**Доработки:**

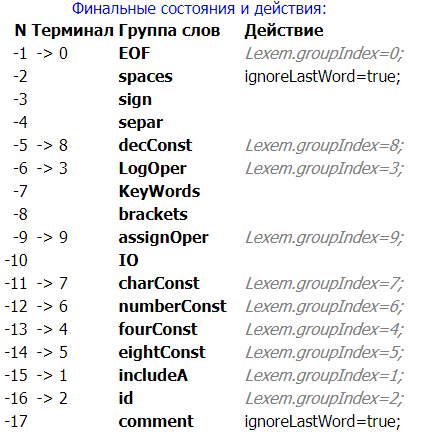
Были доработаны правила лексики. Обновлённые правила показаны на Рис. 1.



*Рис. 1 - Обновленные правила лексики*

**Работа лексических автоматов по разбору заданного фрагмента:**

Финальные состояния и действия представлены на Рисунке 2.



*Рис. 2 - Финальные состояния и действия*

Заданный фрагмент: long n0n = 10;

Конечный автомат, заданный таблицей:

*KeyWords (long):*

*Spaces (“ ”):*

*id (n0n):*

*Spaces (“ ”):*

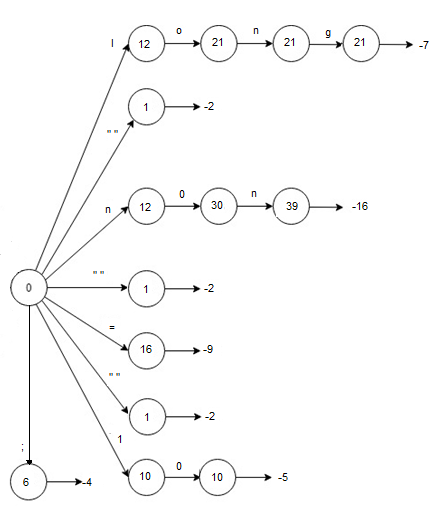
*assignOper (*=*):*

*Spaces (“ ”):*

*decConst (10):*

*separ ( ; ):*

Конечный автомат, заданный графом:



*Рис.3 - Конечный автомат, заданный графом*

**Описание метаязыка формальных грамматик**

Формальной грамматикой называется совокупность , где:

алфавит терминальных символов, это есть конечное множество всех слов языка, порождаемого данной грамматикой. Понятие «терминальный» здесь обозначает неразложимость таких символов с точки зрения синтаксических правил;

алфавит нетерминальных символов, это 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 терминальный символ, это есть один из нетерминальных символов. Этим символом обычно обозначается наиболее общая синтаксическая конструкция, например: <правильная программа>;

система правил подстановки, это есть конечное множество пар цепочек вида α : β, причем цепочка α (левая часть правила) должна содержать хотя бы один нетерминальный символ;

**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 не рекурсивной в противном случае.

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.

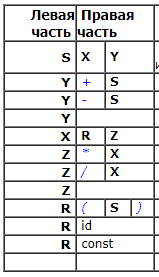
1. **Недостижимость**

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

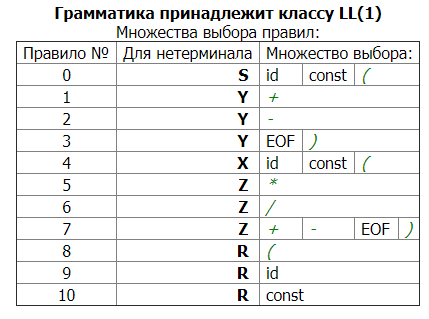
1. **Бесплодность**

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

**Пример из файла 3LL1.xml.**



*Рис. 4 - Правила грамматики*



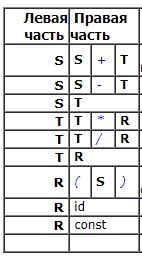
*Рис. 5 - Множество выбора правил языка*



*Рис. 6 - Множество предшественников и множество последователей*

В системе правил представленной в примере не содержится недостижимых и бесплодных грамматик, но, содержится два аннулируемых правила (Z и Y). Множества выбора для всех нетерминалов не пересекаются. Язык принадлежит классу LL(1) и пригоден для корректного построения и работы синтаксического акцептора.

**Пример из файла 4SLR1grammarArithmExpr.xml.**



*Рис. 7 - Правила грамматики*



*Рис. 8 - Множество выбора правил*



*Рис. 9 - Множество предшественников*

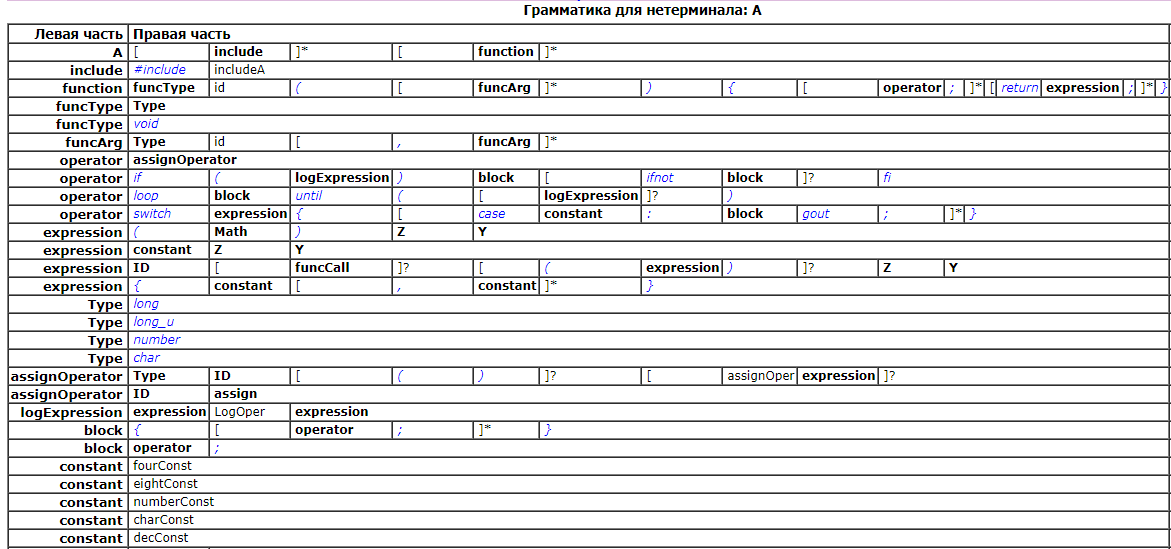


*Рис. 10 - Множество предшественников*

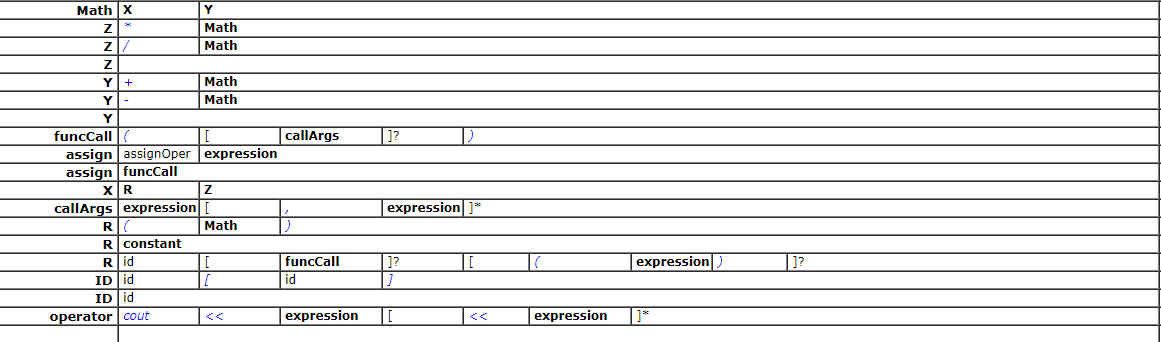
В данном примере во множестве выбора данной грамматике содержатся пересечения, поэтому грамматика не является LL(1), ее работа будет некорректна.

**Разработка грамматики программируемого языка**

Была частично разработана грамматика для языка, в соответствии с заданным вариантом курсовой работы. Она показана на Рис. 11.

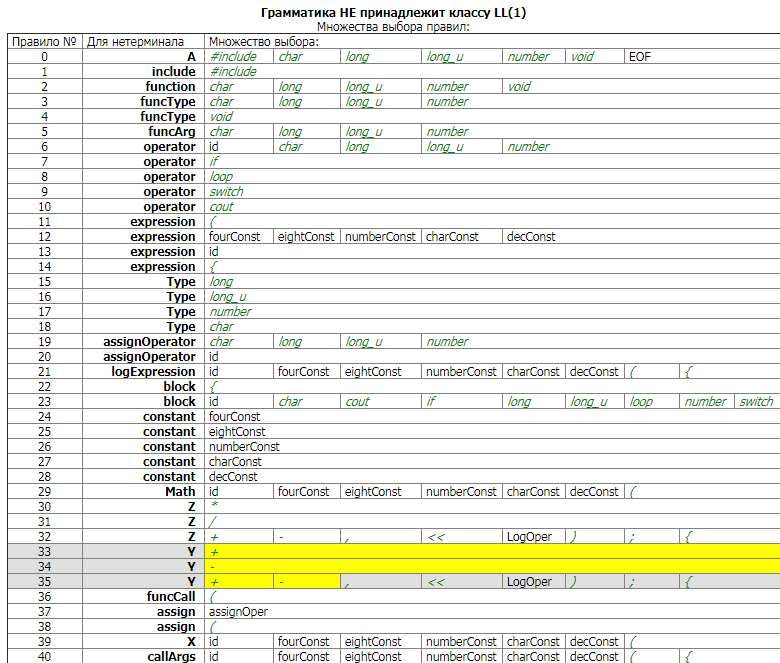


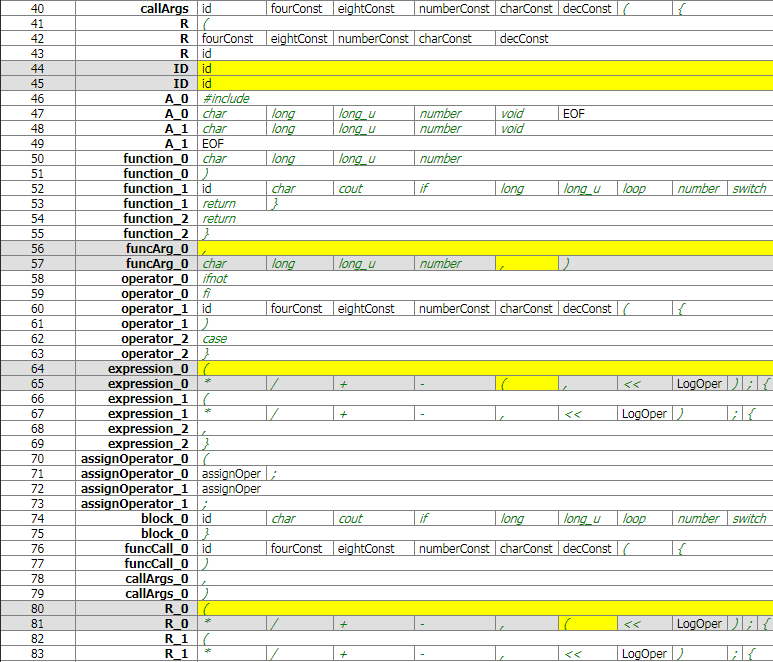
*Рис. 11.1 - Разработанная грамматика*



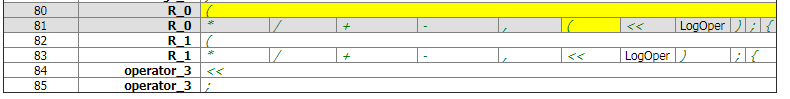
*Рис. 11.2 - Разработанная грамматика*

Были описаны грамматики операторов присваивания, условного оператора и оператора цикла. Имеются пересечения множества выбора. (Рис. 12) Описанная грамматика не принадлежит классу LL(1).

  
*Рис. 12.1 - Множества выбора правил*

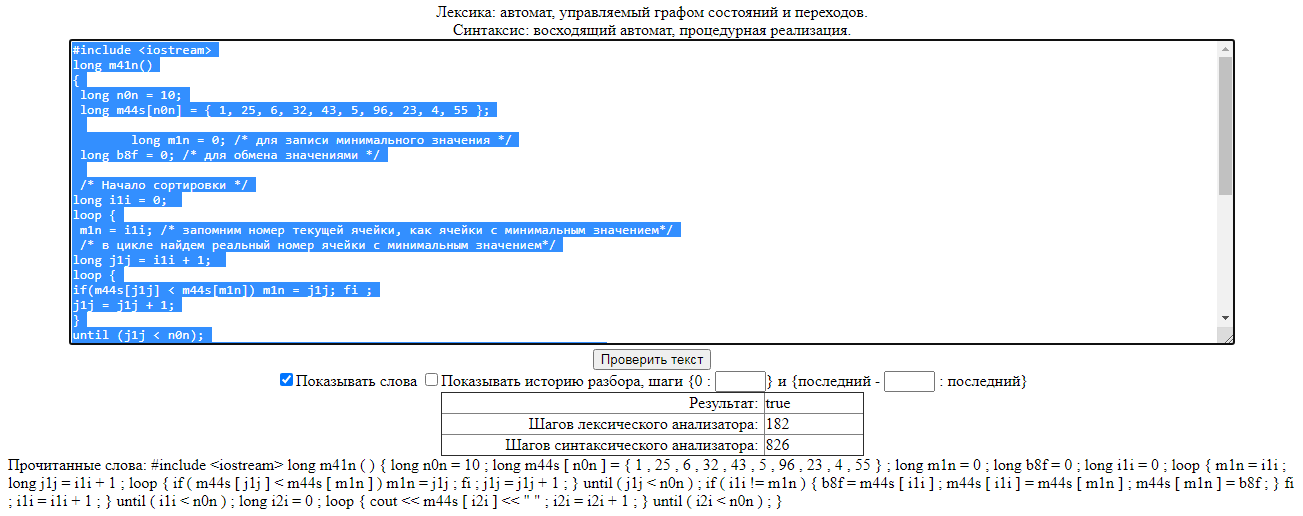


*Рис. 12.2 - Множества выбора правил*



*Рис. 12.3 - Множества выбора правил*

Проверка правильности, разработанной грамматики показана на Рис. 13.



*Рис. 13 - Проверка правильности грамматики*

Проверка проводилась на данной программе:

*#include <iostream>*

*long m41n()*

*{*

*long n0n = 10;*

*long m44s[n0n] = { 1, 25, 6, 32, 43, 5, 96, 23, 4, 55 };*

*long m1n = 0; /\* для записи минимального значения \*/*

*long b8f = 0; /\* для обмена значениями \*/*

*/\* Начало сортировки \*/*

*long i1i = 0;*

*loop {*

*m1n = i1i; /\* запомним номер текущей ячейки, как ячейки с минимальным значением\*/*

*/\* в цикле найдем реальный номер ячейки с минимальным значением\*/*

*long j1j = i1i + 1;*

*loop {*

*if(m44s[j1j] < m44s[m1n]) m1n = j1j; fi ;*

*j1j = j1j + 1;*

*}*

*until (j1j < n0n);*

*/\* cделаем перестановку этого элемента, поменяв его местами с текущим\*/*

*if (i1i != m1n)*

*{*

*b8f = m44s[i1i];*

*m44s[i1i] = m44s[m1n];*

*m44s[m1n] = b8f;*

*}fi;*

*i1i = i1i + 1;*

*}until (i1i < n0n);*

*long i2i = 0;*

*loop{*

*cout << m44s[i2i] << “ “;*

*i2i = i2i + 1;*

*}until(i2i < n0n);*

*}*

*/\* Конец сортировки\*/*

# Вывод

В ходе лабораторной работы были изучены основные понятия метаязыка формальных грамматик, свойств грамматик и нетерминальных символов. Была частично разработана грамматика для языка, в соответствии с заданным вариантом курсовой работы*.*