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*»**

|  |  |
| --- | --- |
| Выполнили: | Проверил: |
| Студент гр. АВТ-709, АВТФ | *к.т.н., доцент* |
| *Рассомахин И. А.* | *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.

Вариант: 22231313

**Описание варианта:**

1. Идентификатор:

<бБ><пЦ><бБ>

<одна любая буква><непустая последовательность цифр><одна любая буква>

Примеры: *d23U, N1q, x15y.*

1. Константы:

целые по основаниям 2,8 и 10 (*2x01, 8x77, 123*);

вещественные (*0.123, 123.321*);

символьные (*'g', 'a'*).

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

int[\_u]

float

letter

Примеры:

*int i9t* – целый;

*float f104t* – вещественный;

*letter l3113r* – символьный.

1. Оператор присваивания:

<И> := <В>;

<Идентификатор> := <произвольное выражение>;

Примеры:

*f104t := 123.321;*

*i9t := 12 + 34;*

1. Условный оператор

at <ЛВ> do <ОБ> [ else do <ОБ> ]

at <логическое выражение> do <оператор или блок> [else do <оператор или блок>]

Примеры:

*at l0w < h1g do*

*{*

*int i9t := 0;*

*i9t := 23;*

*}*

*at i9t < 100 do i9t := 100;*

*else do i9t := 0;*

1. Оператор цикла:

cycle([<ОП>];[<ЛВ>];[<О>]) <ОБ>

cycle([<оператор присваивания>];[<логическое выражение>];[<одиночный оператор>]) <оператор или блок>

Примеры:

*cycle (;;) {}*

*cycle (i9t := 1; i9t < 100; i9t++) f104t := f104t + (i9t / 5);*

1. Опертаор переключателя:

select <В> case (<К> ) <ОБ> [break;] …[ case () <ОБ> ] end

select <произвольное выражение> case (<константа> ) <оператор или блок> [break;] …[ case () <оператор или блок> ] end

Пример:

*select i9t case (1) i9t :=0; break; case (2) i9t := 3; end*

*select i9t case (1) i9t :=0; break; case (2) i9t := 3; break; case () {*

*t3p := i9t;*

*i9t := i92t;*

*i92t := t3p;*

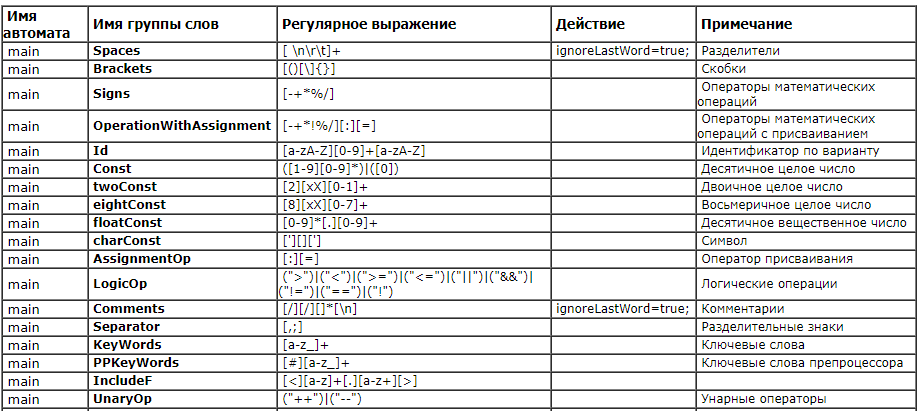
*}*

*end*

# Ход работы

**Доработки:**

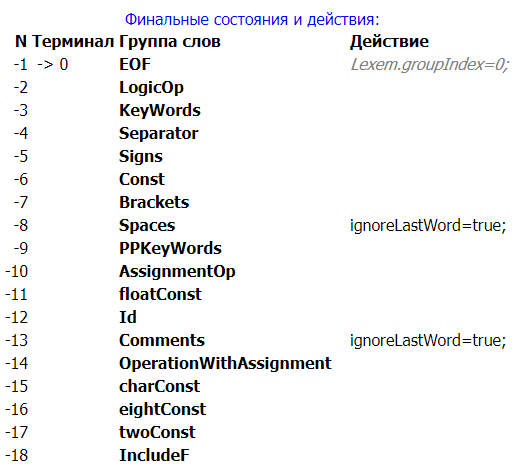
Были доработаны правила лексики. Обновлённые правила показаны на Рис. 1.



*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: q1S(a4r, 0, n1n - 1);

Конечный автомат, заданный таблицей:

*Id (q1S):*

*Brackets (“(“):*

*Id (*a4r*):*

*Separator ( , ):*

*Format (“ ”):*

*Const (0):*

*Separator ( , ):*

*Format (“ ”):*

*Id (*n1n*):*

*Format (“ ”):*

*Signs ( - ):*

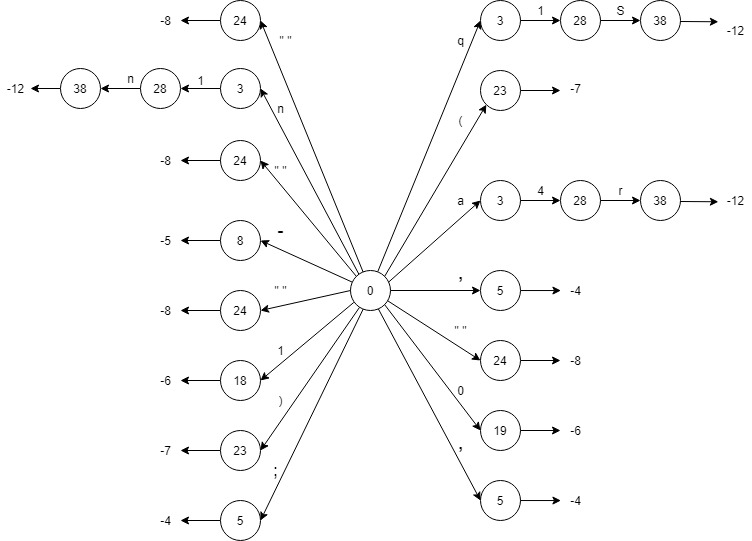
*Format (“ ”):*

*Const (1):*

*Brackets (“)“):*

*Separator ( ; ):*

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.

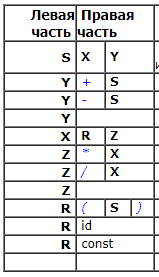
1. **Недостижимость**

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

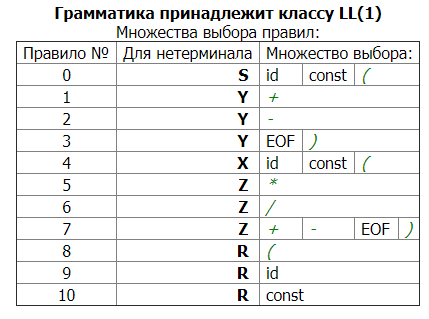
1. **Бесплодность**

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*



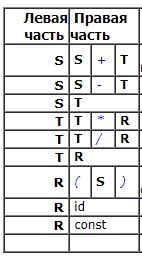
*Рис. 5 - Множество выбора правил языка*



*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*

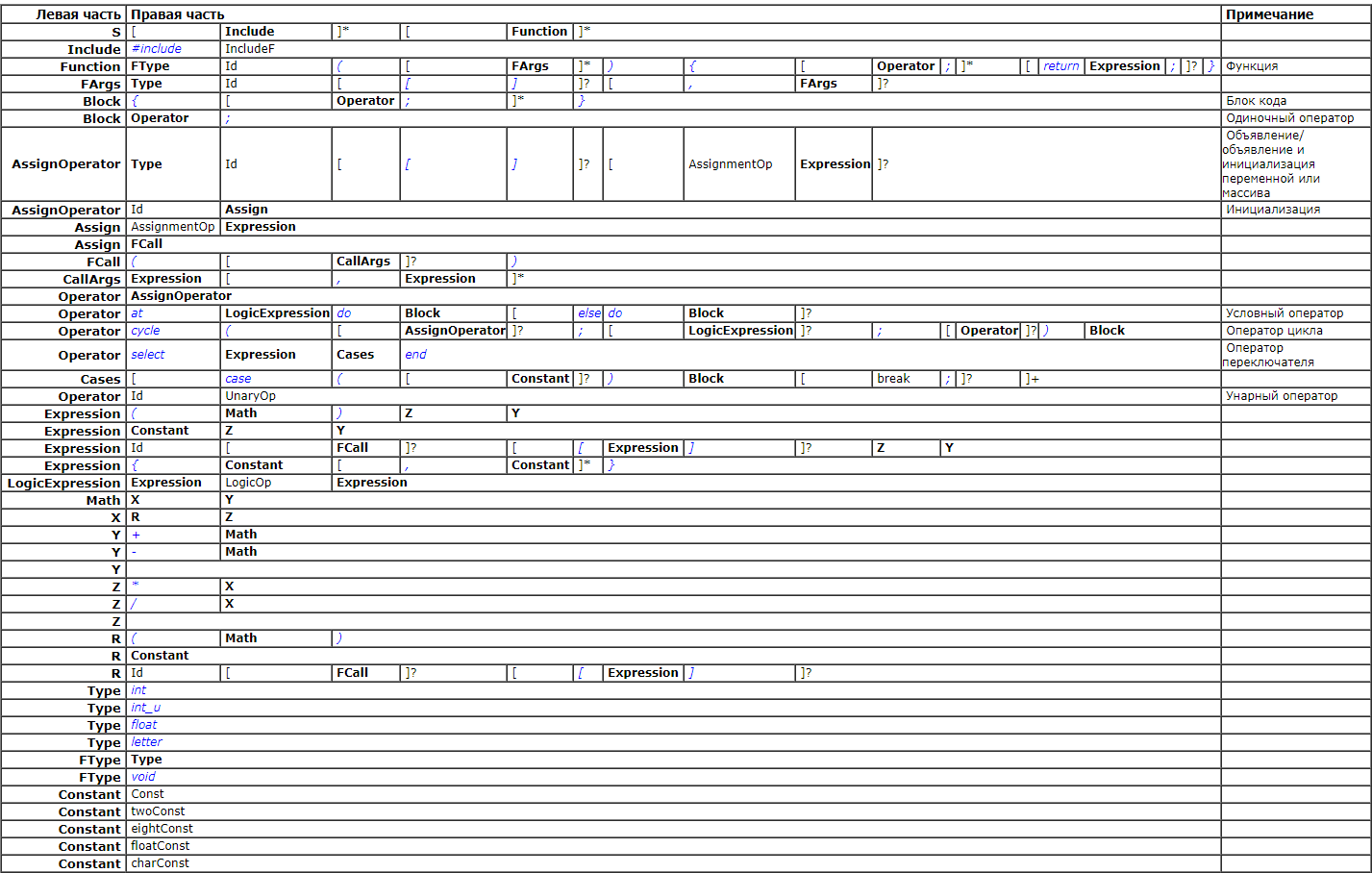


*Рис. 10 - Множество предшественников*

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

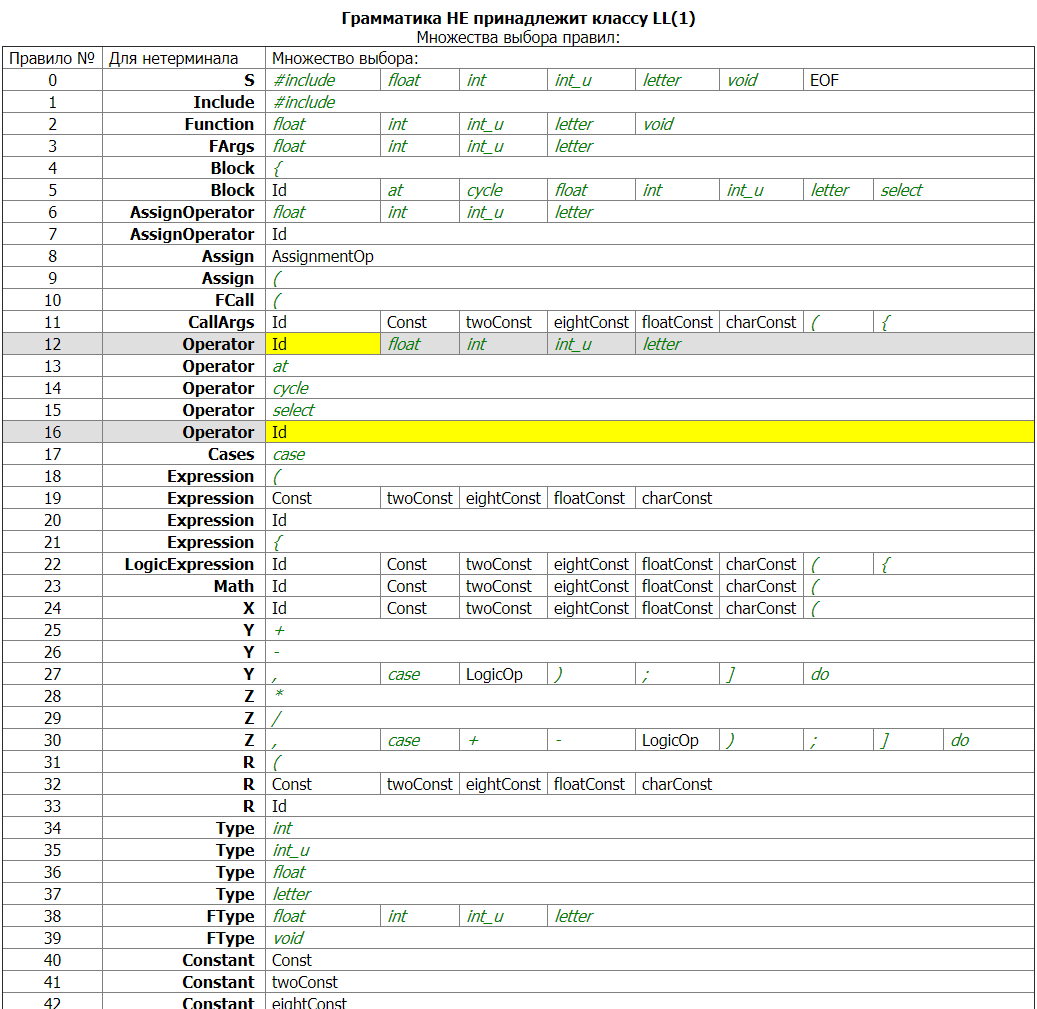
**Разработка грамматики программируемого языка**

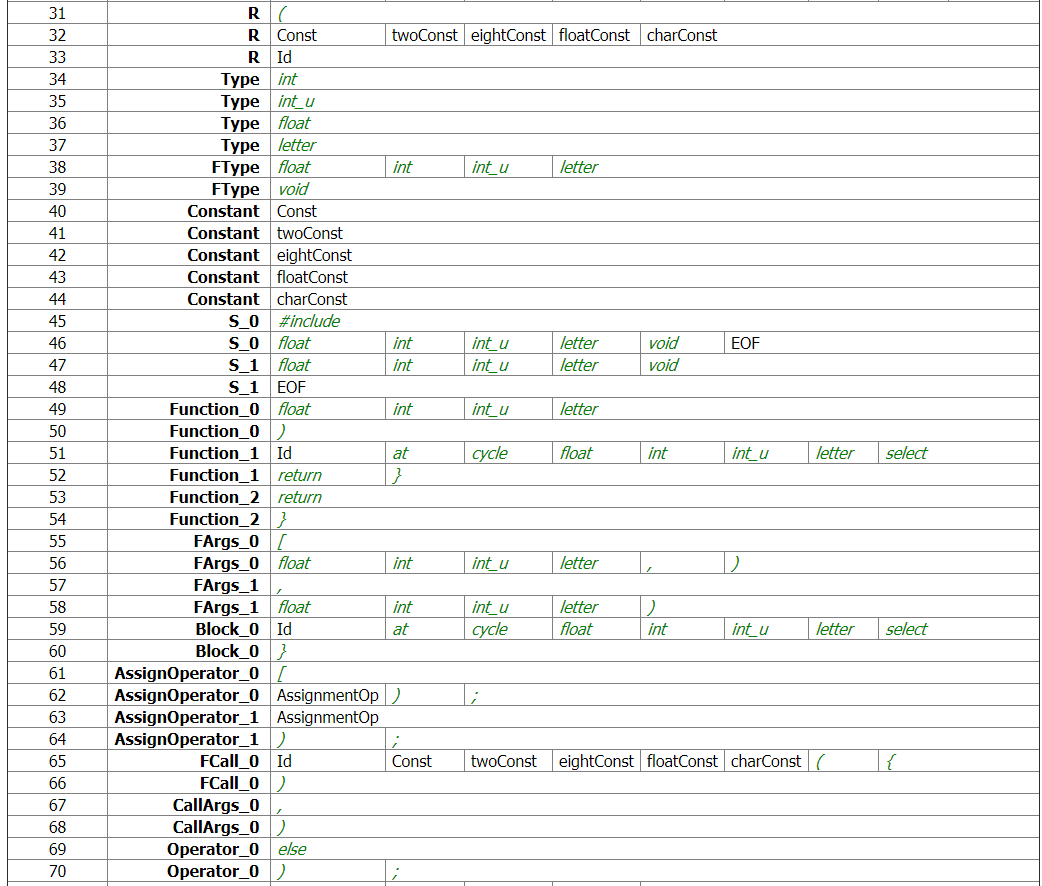
Была частично разработана грамматика для языка, в соответствии с заданным вариантом курсовой работы. Она показана на Рис. 11.



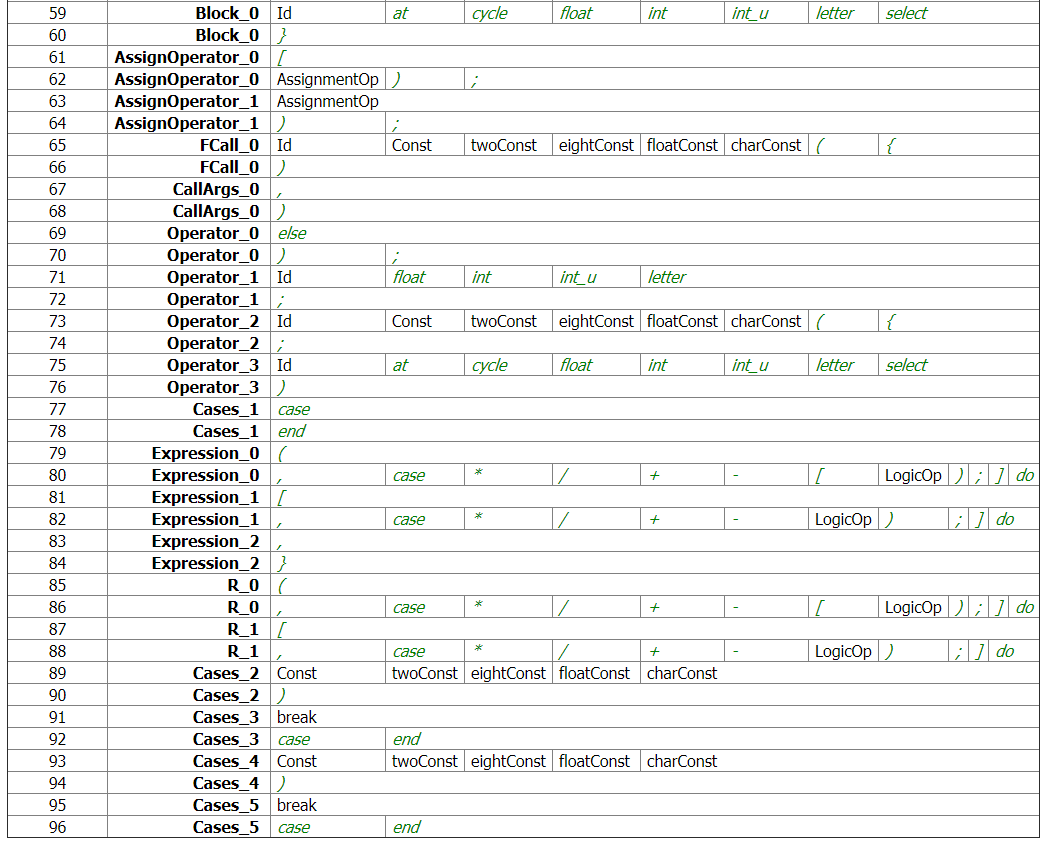
*Рис. 11 - Частично разработанная грамматика*

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*

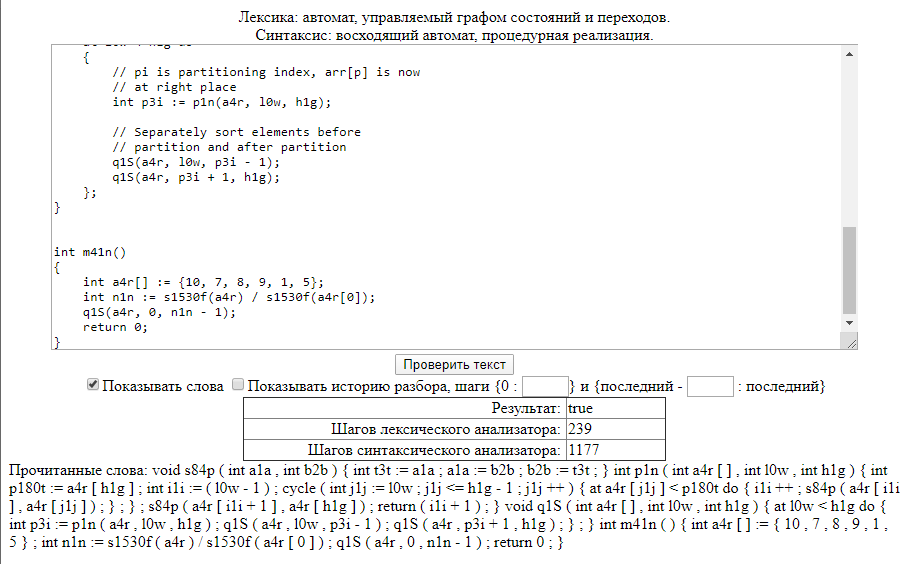


*Рис. 12.2 - Множества выбора правил*



*Рис. 12.3 - Множества выбора правил*

Проверка правильности, разработанной грамматики показана на Рис. 13.



*Рис. 13 - Проверка правильности грамматики*

Проверка проводилась на данной программе:

//quick sort programm

// A utility function to swap two elements

void s84p(int a1a, int b2b)

{

int t3t := a1a;

a1a := b2b;

b2b := t3t;

}

// This function takes last element as pivot, places

//the pivot element at its correct position in sorted

//array, and places all smaller (smaller than pivot)

//to left of pivot and all greater elements to right

//of pivot

int p1n (int a4r [], int l0w, int h1g)

{

int p180t := a4r[h1g]; // pivot

int i1i := (l0w - 1); // Index of smaller element

cycle (int j1j := l0w; j1j <= h1g - 1; j1j++)

{

// If current element is smaller than the pivot

at a4r[j1j] < p180t do

{

i1i++; // increment index of smaller element

s84p(a4r[i1i], a4r[j1j]);

};

};

s84p(a4r[i1i + 1], a4r[h1g]);

return (i1i + 1);

}

// The main function that implements QuickSort

//arr[] --> Array to be sorted,

//low --> Starting index,

//high --> Ending index \*/

void q1S(int a4r[], int l0w, int h1g)

{

at l0w < h1g do

{

// pi is partitioning index, arr[p] is now

// at right place

int p3i := p1n(a4r, l0w, h1g);

// Separately sort elements before

// partition and after partition

q1S(a4r, l0w, p3i - 1);

q1S(a4r, p3i + 1, h1g);

};

}

int m41n()

{

int a4r[] := {10, 7, 8, 9, 1, 5};

int n1n := s1530f(a4r) / s1530f(a4r[0]);

q1S(a4r, 0, n1n - 1);

return 0;

}

# 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*.*