МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

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

Кафедра вычислительной техники



**ОТЧЕТ**

**По лабораторной работе №4**

**Синтаксис языков программирования. Нисходящий синтаксический анализ.**

**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 идей и понятий нисходящих методов синтаксического анализа, выявление свойств формальных грамматик, необходимых для реализации нисходящего восстановления дерева грамматического разбора, приобретение навыков построения процедурной и различных автоматных реализаций нисходящего анализа, исследование поведения нисходящих синтаксических акцепторов.

# Постановка задачи

* 1. Используя пакет ВебТрансЛаб:
* изучить и освоить проверку принадлежности грамматики к классу **LL(1)**, используя в качестве проверяемых грамматики, полученные при выполнении работы №4;
* построить конечный автомат со стековой памятью и несколькими состояниями (шаблон …SyntAsMultiFSM…), разобраться в структуре управляющей таблицы автомата, уяснить способы формирования и использования всех полей;
* построить конечный автомат со стековой памятью и одним состоянием, управляемый входным символом и символом, снятым с верхушки стека (шаблон …SyntAsSingleFSM…), разобраться в структуре управляющей таблицы автомата, уяснить способы формирования и использования клеток таблицы;
* построить процедурную реализацию рекурсивного спуска (шаблон …SyntAsRD…), уяснить способы формирования функций этого акцептора;
  1. Выполнить трассировку процессов нисходящего синтаксического акцепта, изучить поведение всех построенных синтаксических акцепторов при разборе как правильных предложений, так и предложений с намеренно внесенными синтаксическими ошибками.
  2. Проанализировать и сравнить между собой все полученные тексты программ и результаты выполнения пункта 3.2. Оценить степень пригодности изученных вариантов реализации нисходящих синтаксических акцепторов для выполнения курсовой работы.
  3. Подготовить, сдать и защитить отчет к лабораторной работе.

Вариант: 23112143

**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:

?? <В> { ? <К> : <ОБ> [gout;] …[ ?~ : <ОБ> ] }

Пример:

*?? i9t { ? 1 : i9t = 0; gout; ?~ 2 : i9t = 3; }*

*?? i9t { ? 1 : i9t =0; gout;* ?~ *2 : i9t = 3; gout;* ?~ *3 : {*

*t3p = i9t;*

*i9t = i92t;*

*i92t = t3p;*

*}}*

# Теоретическая часть

*LL(1)-грамматикой* называется такая контекстно-свободная грамматика, у которой множества выбора правил с одинаковым нетерминалом в левой части попарно не пересекаются.

Принято считать, что символы в названии класса *LL(1)-грамматик* обозначают следующее.

*Первая буква L (сокращение слова left - левый)* - чтение слов анализируемого предложения производится слева направо.

*Вторая буква L (сокращение слова leftmost - самый левый)* - на каждом шаге принимается решение для замены самого левого нетерминала из текущего уровня восстанавливаемого дерева.

*Цифра 1 в скобках* обозначает количество символов из начала остатка предложения, необходимых для принятия решения о выборе правила на каждом шаге детерминированного нисходящего восстановления дерева грамматического разбора.

Функционирование *конечного автомата со стековой памятью и несколькими состояниями* определяется управляющей таблицей. Предполагается, что автомат при запуске оказывается в особом начальном состоянии, на каждом такте по входному символу и текущему состоянию определяет и выполняет операции над входным потоком символов, стековой памятью и собственным состоянием.

Каждому символу каждого правила грамматики должно быть поставлено в соответствии в точности одно состояние автомата. С каждым состоянием должно быть связано множество выбора и два адреса перехода. Под адресом перехода понимается номер состояния.

При соблюдении определенных правил нумерации состояний и введении операции управления остановом по ошибке можно обойтись только одним адресом перехода.

С каждым состоянием должны быть также связаны операции управления стековой памятью и чтением следующего входного символа. Все операции управления могут задаваться булевскими значениями *true / false*, которые называются флажками. Обозначения для флажков управления операциями:

* + *флаг a* управляет чтением следующего входного символа;
  + *флаг s* управляет занесением адреса точки возврата в стек;
  + *флаг r* обеспечивает переключение автомата в состояние, номер которого снимается с верхушки стека возвратов;
  + *флаг e* запрещает останов по ошибке, когда состояние соответствует нетерминалу из левой части и есть еще хотя бы одно правило для такого нетерминала.

*LL(1)-грамматику* легко можно преобразовать в *конечный автомат с единственным состоянием* и стековой памятью, управляемый:

* + текущим входным символом;
  + символом, находящимся на верхушке стека.

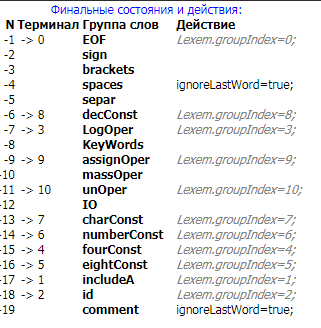
Поведение такого автомата определяется управляющей таблицей, столбцы которой соответствуют входным символам, строки - символам, которые могут находиться в стеке, а в клетках указана некоторая последовательность операций над стеком, входным потоком и состоянием автомата.

# Ход работы

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

Название файла: lab4.xml

Финальные состояния и действия представлены на Рисунке 1.



*Рис. 1 - Финальные состояния и действия*

Заданный фрагмент: long n0n = 10;

Конечный автомат, заданный таблицей:

*KeyWords (long):*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *№ шага* | *0* | *10* | *18* | *18* | *18* | *-8* |
| *№ состояния* | *24* | *24* | *24* | *24* | *24* |  |
| *Вход. символ* | *l* | *o* | *n* | *g* | *“ “* |  |

*Spaces (“ ”):*

|  |  |  |  |
| --- | --- | --- | --- |
| *№ шага* | *0* | *3* | *-4* |
| *№ состояния* | *1* | *24* |  |
| *Вход. символ* | *“ “* | *n* |  |

*id (n0n):*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№ шага* | *0* | *10* | *25* | *-18* |
| *№ состояния* | *24* | *13* | *24* |  |
| *Вход. символ* | *n* | *0* | *n* |  |

*Spaces (“ ”):*

|  |  |  |  |
| --- | --- | --- | --- |
| *№ шага* | *0* | *3* | *-4* |
| *№ состояния* | *1* | *20* |  |
| *Вход. символ* | *“ “* | *=* |  |

*assignOper (*=*):*

|  |  |  |  |
| --- | --- | --- | --- |
| *№ шага* | *0* | *13* | *-9* |
| *№ состояния* | *20* | *1* |  |
| *Вход. символ* | *=* | *“ “* |  |

*Spaces (“ ”):*

|  |  |  |  |
| --- | --- | --- | --- |
| *№ шага* | *0* | *3* | *-4* |
| *№ состояния* | *1* | *14* |  |
| *Вход. символ* | *“ “* | *1* |  |

*decConst (10):*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№ шага* | *0* | *8* | *8* | *-6* |
| *№ состояния* | *14* | *13* | *9* |  |
| *Вход. символ* | *1* | *0* | *;* |  |

*separ ( ; ):*

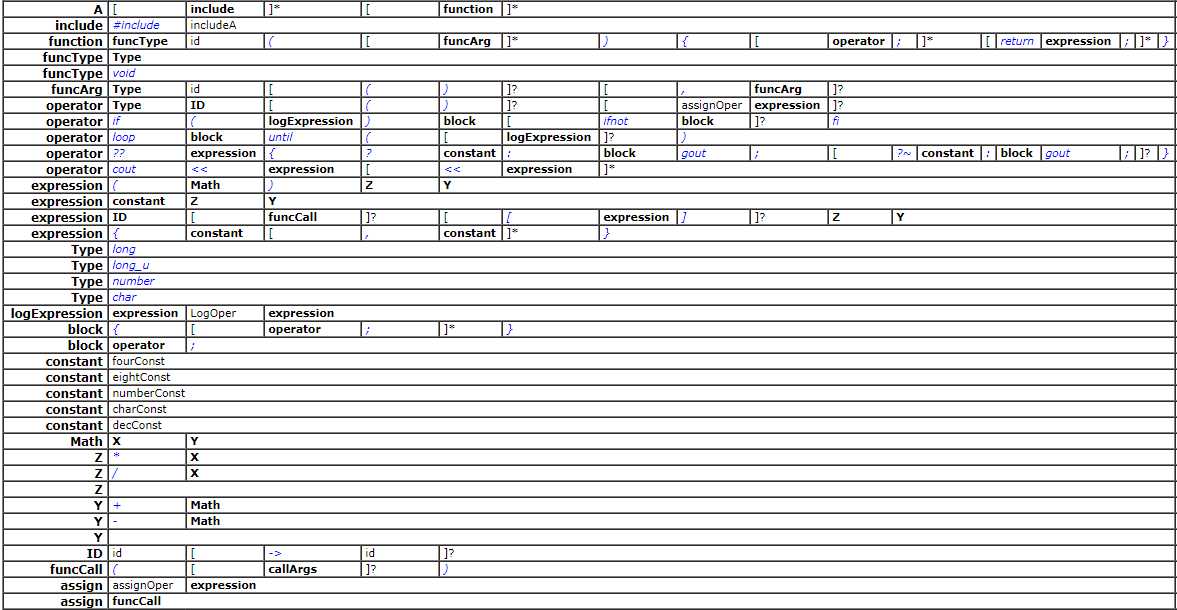
|  |  |  |
| --- | --- | --- |
| *№ шага* | *0* | *-5* |
| *№ состояния* | *9* |  |
| *Вход. символ* | *;* |  |

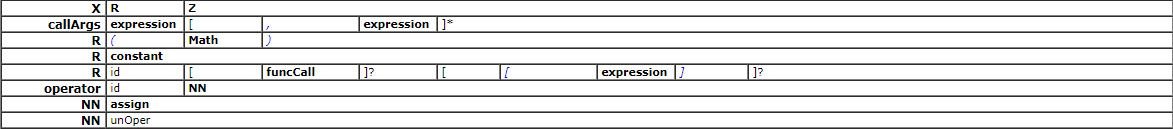
**Доработки:**

Исправлены привила лексики и синтаксиса. Убрано пересечение правил присваивания и унарных операций. Добавлены дополнительные символы для ключевых слов. Добавлено еще одно правило лексики для массивов, теперь елемент массива обозначается так ”->”. Множества выбора правил с одинаковым нетерминалом в левой части попарно не пересекаются. Значит грамматика принадлежит классу *LL(1)-грамматик*.

Таблица множества выбора правил предоставлена в Приложении.

Обновлённые правила показаны на Рис. 2.





*Рис. 2 - Обновленные правила синтаксиса*

**Тестирование программы и шаблоны автоматов**

Правильный вариант программы:

*#include <iostream>*

*void m41n(){*

*long a4r->n0n;*

*}*

Неправильный вариант программы:

*#include <iostream>*

*void main(){*

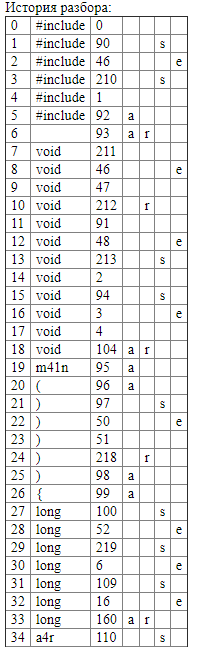
*long a4r->n0n;*

*}*

Построим конечный автомат по шаблону ..SyntAsMultiFSM.. и запустим оба варианта тестовой программы:

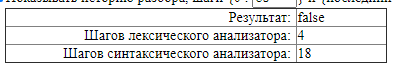


*Рис. 3 - Результат теста верной программы по шаблону SyntAsMultiFSM*

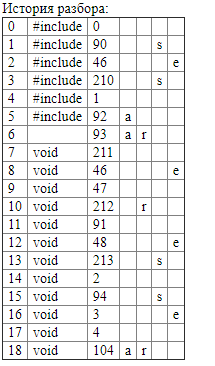


*Рис. 4 - Результат теста верной программы по шаблону SyntAsMultiFSM*

*(часть истории разбора)*

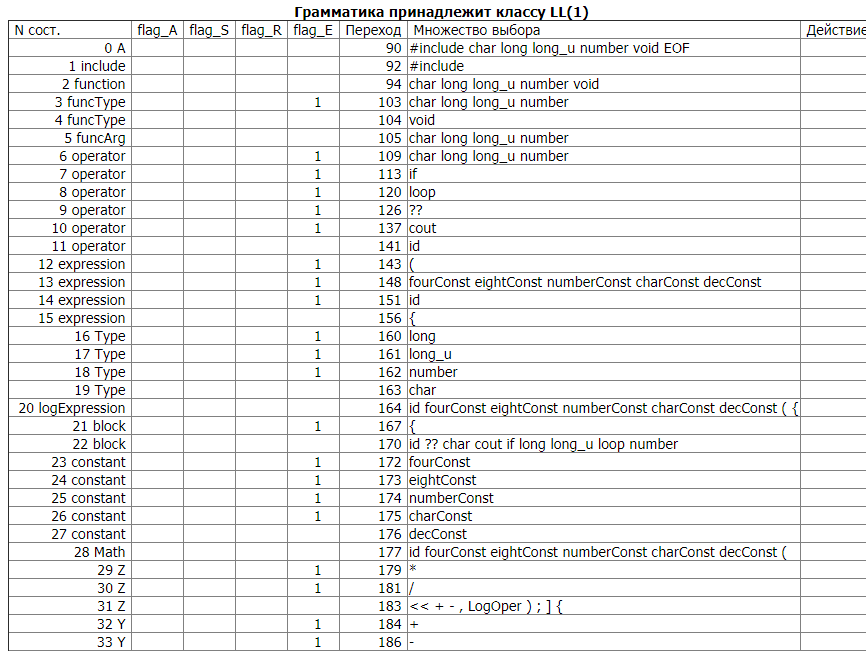


*Рис. 5 - Результат теста ошибочной программы по шаблону SyntAsMultiFSM*



*Рис. 6 - Результат теста ошибочной программы по шаблону SyntAsMultiFSM*

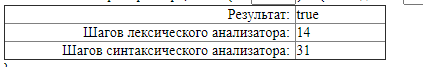
*(история разбора)*



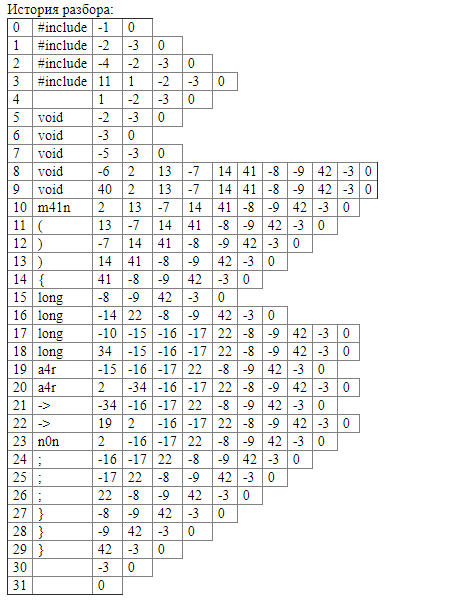
*Рис. 7 - Упр. Таблица с неск. Сост. По шаблону SyntAsMultiFSM*

Каждому номеру состояния данной таблицы соответствует множество выбора, которое ожидается в каждом из состояний. Если, например, мы находимся в состоянии 9 и получаем не “??“, то получится ошибка. Также в данной таблице имеется информация о флагах программы, описанных выше, и переходах.

Построим конечный автомат по шаблону SyntAsSingleFSM и изучим управляющую таблицу автомата.

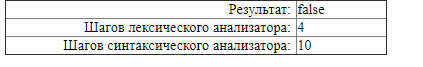


*Рис. 8 - Результат теста верной программы по шаблону SyntAsSingleFSM*

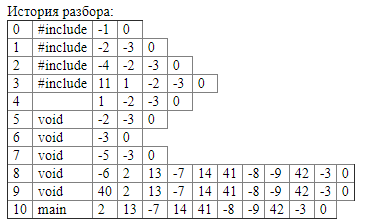


*Рис.9 - Результат теста верной программы по шаблону SyntAsSingleFSM*

*(история разбора)*

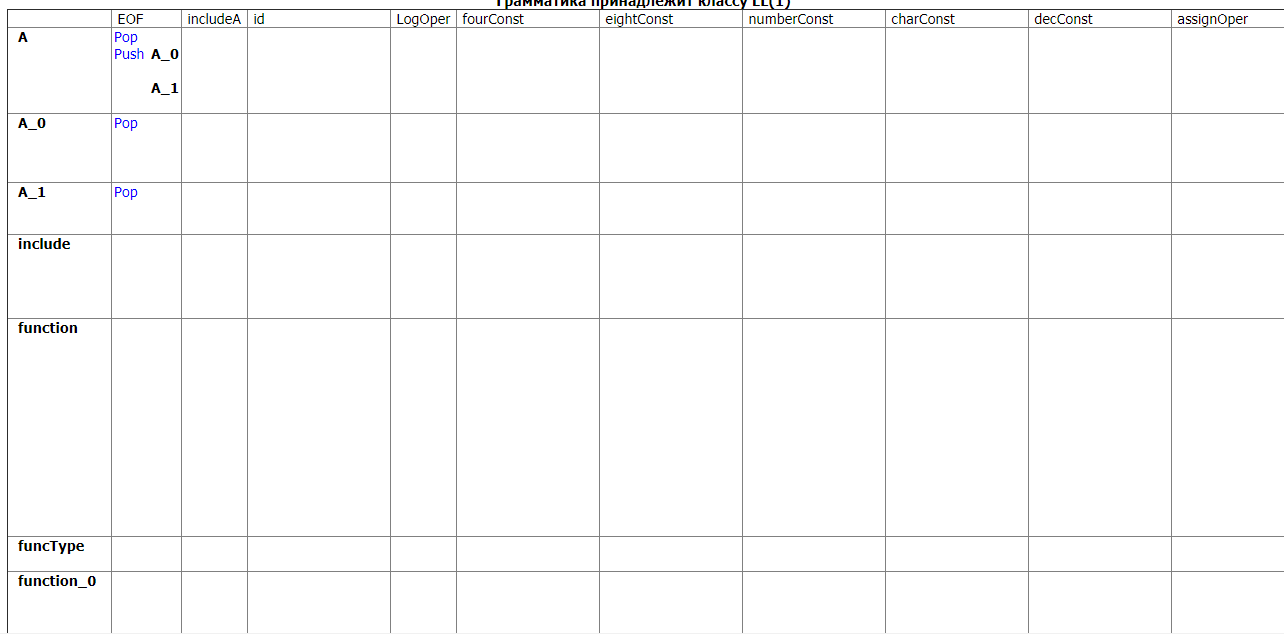


*Рис.10 - Результат теста ошибочной программы по шаблону SyntAsSingleFSM*



*Рис.11 - Результат теста ошибочной программы по шаблону SyntAsSingleFSM*

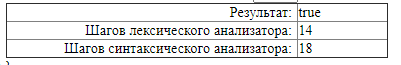
*(история разбора)*

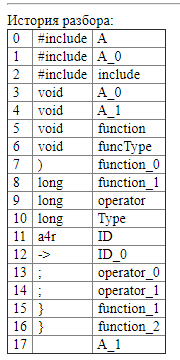


*Рис. 12 - Часть управляющей таблицы по шаблону SyntAsSingleFSM*

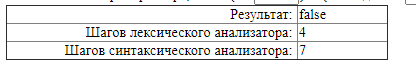
Данная таблица также сообщает о множествах выбора в каждом состоянии программы. Строки – состояния, в которых может находиться программа, столбцы – общее множество выбора. Если пересечение пустое, значит для данного состояния не описан этот выбор т.е. попадание в пустое пересечение выдаст ошибку.

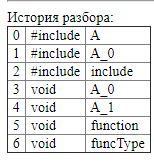
Построим конечный автомат по шаблону SyntAsRD, проверим работоспособность тестовой программы:





*Рис. 13 - Результат теста с историей разбора верной программы по шаблону SyntAsRD*





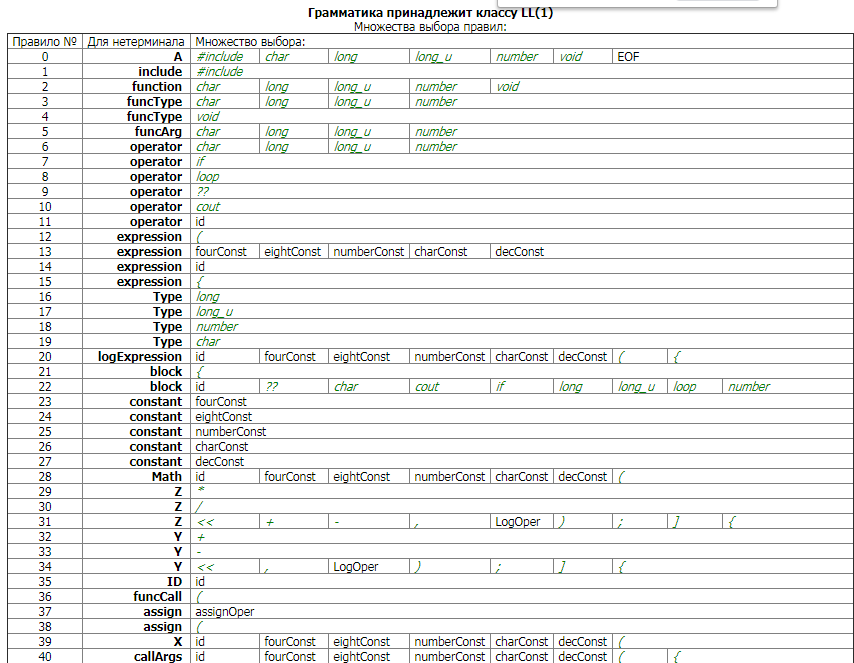
*Рис. 14 – Результат теста с историей разбора ошибочной программы по шаблону SyntAsRD*

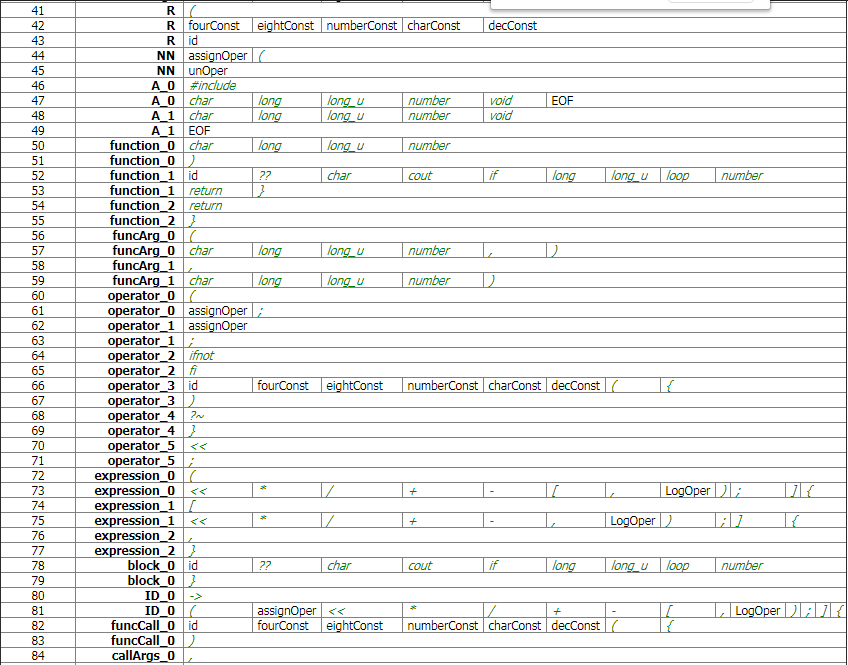
Во втором столбце истории разбора показано использованное слово из множества выбора для данного состояния, а в третьем столбце состояние на данном шаге разбора.

# Вывод

В ходе лабораторной работы были изучены основных идеи и понятия нисходящих методов синтаксического анализа. Была доработана грамматика для языка в соответствии с заданным вариантом курсовой работы*.*

**Приложение**



**Приложение**

**Приложение**

