МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙРАБОТЕ №2**

**ПО ДИСЦИПЛИНЕ «ТЕОРИЯ ФОРМАЛЬНЫХ ЯЗЫКОВИ КОМПИЛЯТОРОВ»**

**НА ТЕМУ**

**«ЛЕКСИКА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ. КОНЕЧНЫЕ АВТОМАТЫ БЕЗ ПАМЯТИ ОБНАРУЖЕНИЯ СЛОВ В ТЕКСТЕ ПРОГРАММЫ»**

Факультет: АВТФ Преподаватель: Малявко А.А.

Группа: АВТ-709

Студент: Кузнецов А.В.

Вариант: 33312212

Новосибирск 2020 г.

# Цели работы

Изучение конечных автоматов (КА) без памяти, способов определения КА – канонического, графового и табличного, методов построения недетерминированного КА по системе регулярных выражений, методов эквивалентных преобразований недетерминированных КА в оптимальные полностью определенные КА – лексические акцепторы.

# Задание

1.   Используя пакет ВебТрансЛаб, освоить:

-  создание лексических правил на языке регулярных выражений (РВ);

-  использование операций «+, \*, ?, конкатенации и выбора» языка РВ для построения сложных регулярных выражений;

-  преобразование простой системы РВ в одноавтоматный лексический акцептор;

-  добавление правил и действий в систему РВ для построения мультиавтоматного лексического акцептора;

2. Разработать (доработать разработанный при выполнении работы №1) фрагмент системы регулярных выражений для всех (или выбранной самостоятельно части) групп слов языка, определенного заданием на курсовую работу. Построить по этому фрагменту:

-  программный модуль, управляемый графом состояний и переходов;

-  программный модуль, управляемый таблично;

3. Изучить структуру программных модулей, построенных ВебТрансЛабом, изучить алгоритмы работы лексического акцептора для графового и табличного способов реализации КА, сравнить реализации конечных автоматов, управляемых различными способами, между собой, оценить их затраты времени в шагах на распознавание слов;

4. Изучить по тексту программного модуля способ реализации вызова действий, определенных в лексических правилах и алгоритм работы формирователя лексем.

5. Проверить функционирование конечных автоматов, построенных ВебТрансЛабом (подготовить тестовый пример программы на языке, заданном на курсовую работу, запустить каждый автомат на выполнение, протрассировать вручную работу лексического акцептора в графовой и табличной реализации, убедиться в работоспособности автоматов, в противном случае – доработать систему РВ и добиться правильного функционирования лексического акцептора).

6. Подготовить, сдать и защитить отчет к лабораторной работе.

# Ход работы

В данной лабораторной работе были описаны некоторые элементы лексики, согласно варианту курсовой работы.

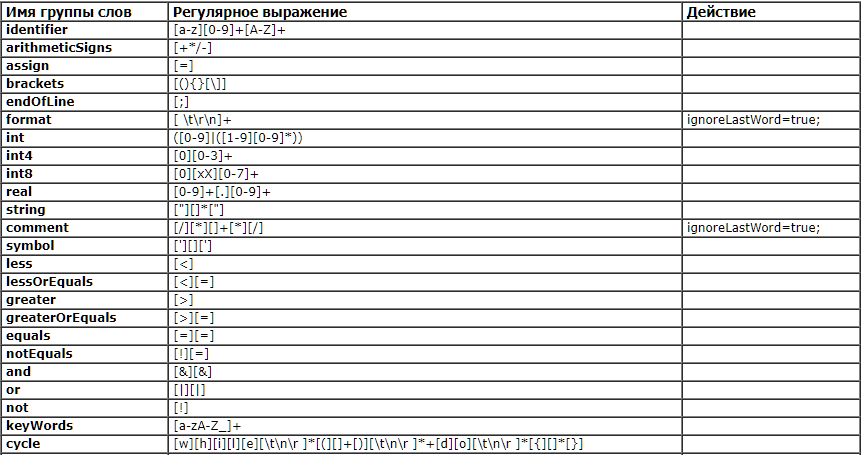


Рисунок 1. Правила лексики языка.

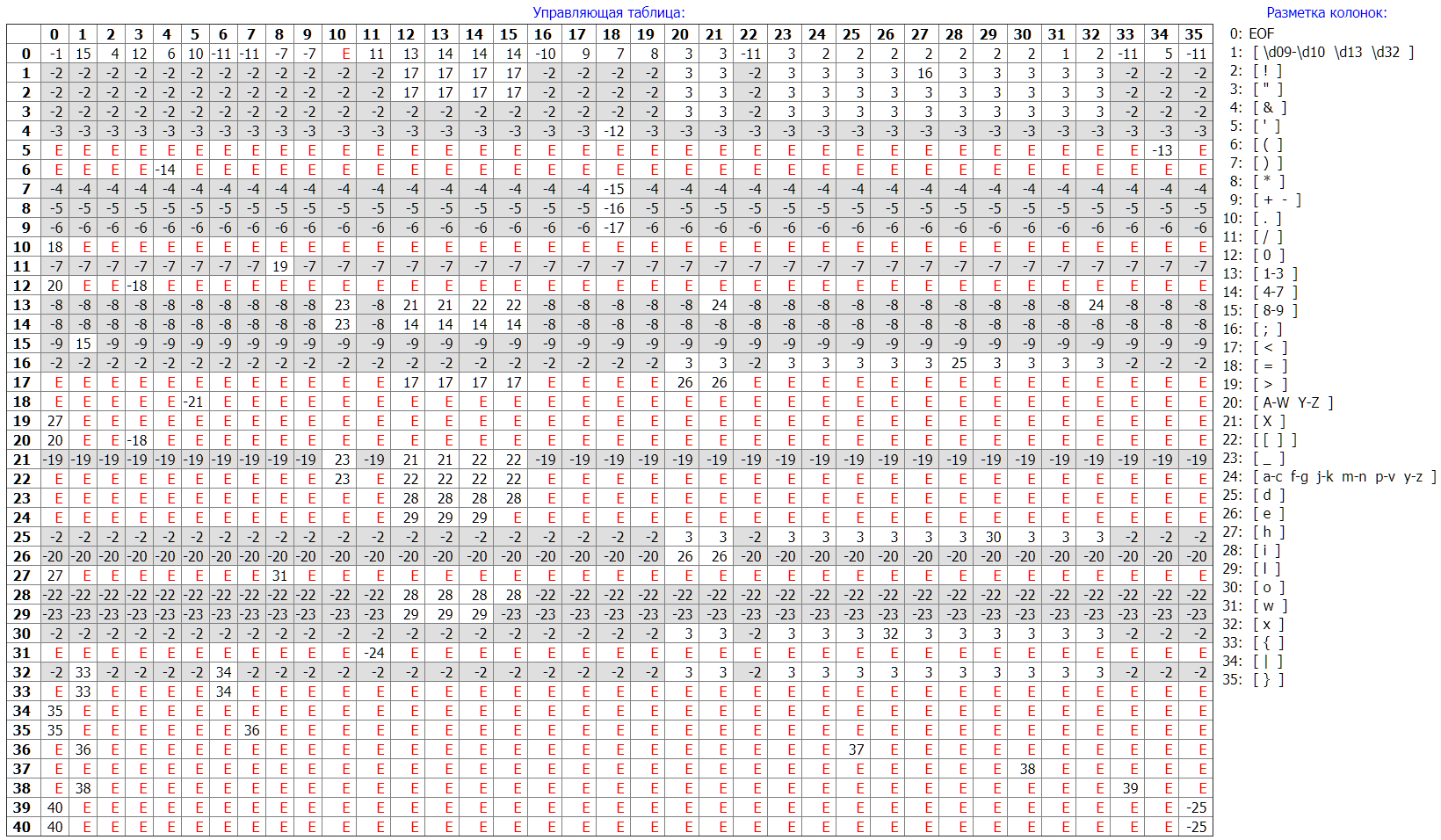
****

Рис. 2 – Управляющая таблица автомата main

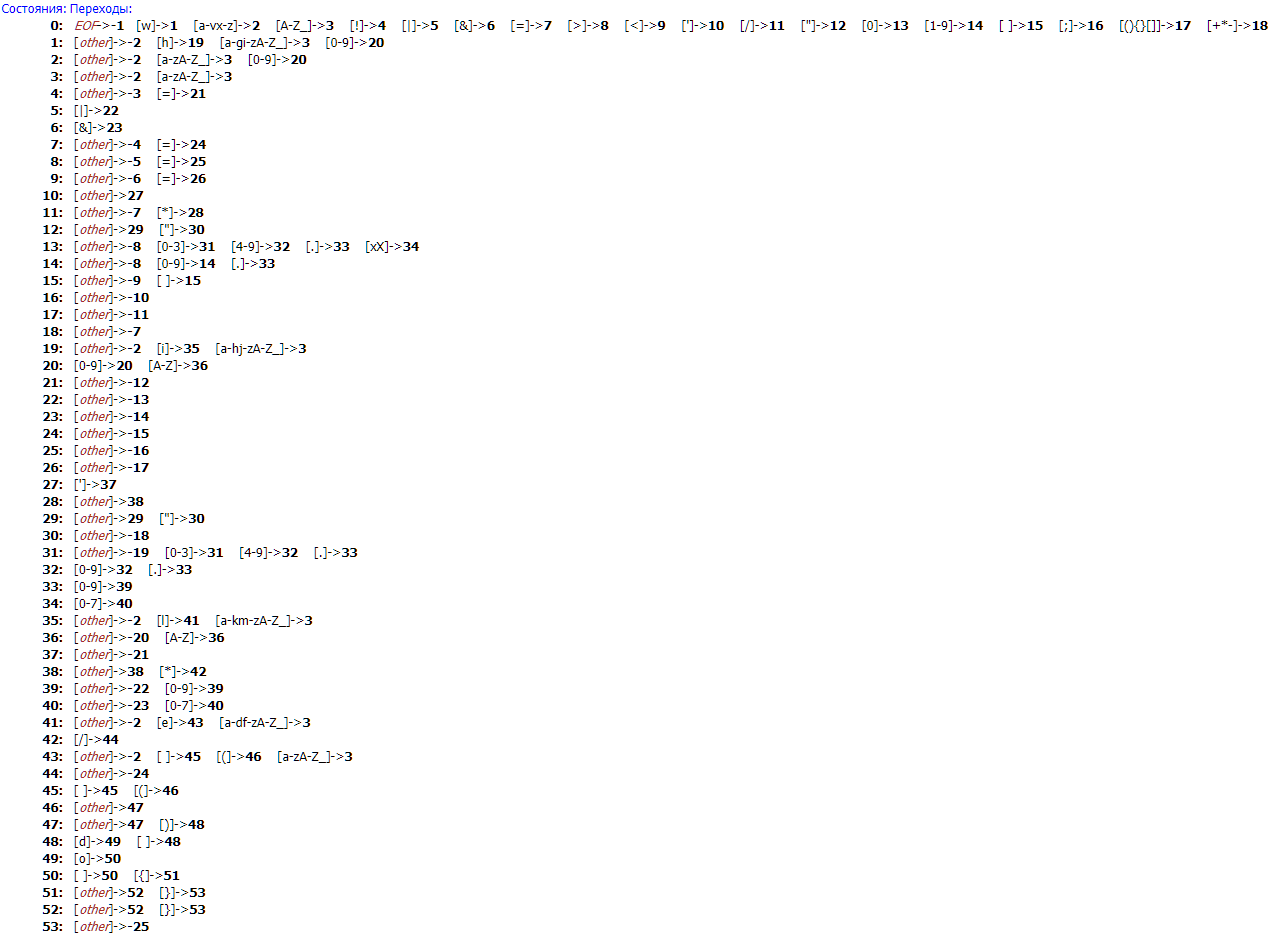


Рисунок 3. Граф состояний и переходов автомата main.

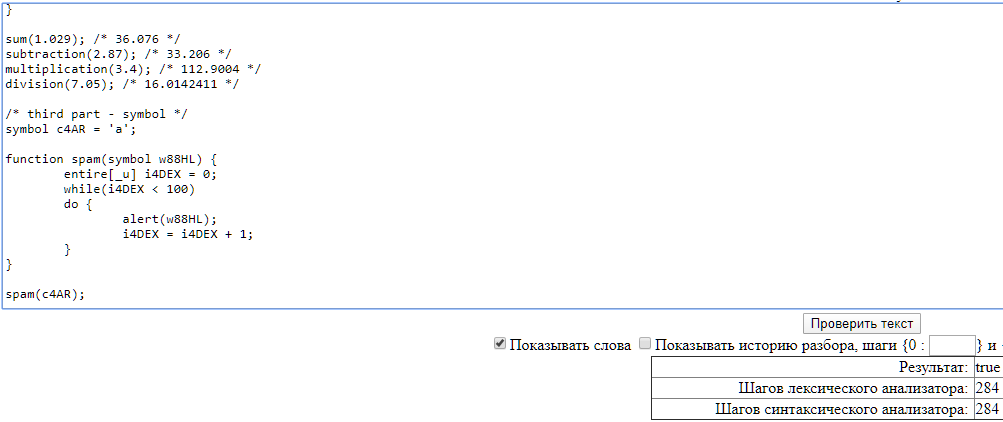


Рисунок 4. Пример распознавания слов лексическим анализатором.

**Описание языка**

Программа на языке программирования 33312212 должна быть написана в одном файле. Строки в данном языке разделяются символом “ ; ”. В языке присутствует возможность однострочных комментариев, написать которые можно с использованием символов “ /\* … \*/ “. Кроме того, в языке поддерживается написание скобок (), {}, а также использование \. В языке существует конструкция while(<логическое выражение>) do {<необязательная часть>}.

Типы данных:

1. int – целочисленный тип, основание 10. Пример формата записи данных: 109.
2. int4 – целочисленный тип, основанием 4. Пример формата записи данных: 02013.
3. int8 – целочисленный тип, основание 8. Пример формата записи данных: 0x064107 или 0X302.
4. real – вещественный тип данных, значение не может быть отрицательным. Пример формата записи данных: 0.0379.
5. string – строковый тип данных. Может содержать любую последовательность символов или не содержать ее вовсе. Пример формата записи данных: “some string123-/+\_”.
6. symbol – символьный тип данных. Обязан содержать ровно один любой символ. Пример формата записи данных: ‘c’.

Арифметические операторы:

1. + - сложение.
2. – - вычитание.
3. \* - умножение.
4. / - деление.

Логические операторы:

1. < - меньше.
2. > - больше.
3. <= - меньше либо равно.
4. >= - больше либо равно.
5. == - эквивалентно.
6. != - неэквивалентно.
7. ! – логическое НЕ.
8. && - логическое И.
9. || - логическое ИЛИ.

Другие операторы:

1. = - присваивание.
2. while(<логическое выражение>) do {<необязательная часть>}. Между ключевыми словами while, do и скобками может быть написано сколько угодно символов форматирования (возврат каретки, перенос строки, табуляция, пробел). После ключевого слова while в скобках обязательно должно присутствовать логическое выражение. В скобках после ключевого слова do присутствие каких-либо выражений необязательно.

# Выводы

В результате выполнения данной лабораторной работы в среде ВебТрансЛаб были доработаны и созданы новые лексические правила языка 33312212.

**Приложение**

Тексты автоматов, построенных по фрагменту системы лексических правил

/\* first part - entire[\_u] \*/

entire[\_u] i1NT = 356;

function print(string str) {

if (str != null) alert(str) fi

}

function isGreater(entire[\_u] a6G) {

if (a6G >= i1NT) return 1 fi

}

function isLess(entire[\_u] a6G) {

if (a6G <= i1NT) return 2 fi

}

function checkResult(entire[\_u] a6G) {

if (a6G == 2 || a6G == 1) alert("less or greater") fi

if (a6G != 2 && a6G != 1) alert("undefined") fi

}

print("Hello world!");

entire[\_u] r31S = isGreater(199); /\* r31S == 0 \*/

entire[\_u] r32S = isLess(21); /\* r32S == 2 \*/

check(r31S); /\* prints "undefined" \*/

check(r32S); /\* prints "less or greater" \*/

/\* second part - real \*/

real r3AL = 35.047;

function sum(real o2YP) {

r3AL = r3AL + o2YP;

}

function subtraction(real o2YP) {

r3AL = r3AL - o2YP;

}

function multiplication(real o2YP) {

r3AL = r3AL \* o2YP;

}

function division(real o2YP) {

r3AL = r3AL / o2YP;

}

sum(1.029); /\* 36.076 \*/

subtraction(2.87); /\* 33.206 \*/

multiplication(3.4); /\* 112.9004 \*/

division(7.05); /\* 16.0142411 \*/

/\* third part - symbol \*/

symbol c4AR = 'a';

function spam(symbol w88HL) {

entire[\_u] i4DEX = 0;

while(i4DEX < 100)

do {

alert(w88HL);

i4DEX = i4DEX + 1;

}

}

spam(c4AR);

/\* fourth part - string \*/

string s7R = "any string";

print(s7R);

/\* int4 \*/

entire[\_u] f0UR = 0103;

/\* int8 \*/

entire[\_u] e1GHT = 0x0173;

e1GHT = 0X564;