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 №2**

*Лексика языков программирования. Конечные автоматы без памяти для обнаружения слов в тексте программы*

**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

**Оглавление**

[**Введение** 3](#_Toc35780796)

[**Вариант** 4](#_Toc35780797)

[**Ход работы** 5](#_Toc35780798)

[*1.* *Лексика* 5](#_Toc35780799)

[*2.* *Типы данных* 6](#_Toc35780800)

[*3.* *Операции* 7](#_Toc35780801)

[*4.* *Операторы* 8](#_Toc35780802)

[*5.* *Функции* 10](#_Toc35780803)

[*6.* *Элементы языка* 10](#_Toc35780804)

[*7.* *Структура программы* 11](#_Toc35780805)

[*8.* *Пример кода* 11](#_Toc35780806)

[**Вывод** 15](#_Toc35780807)

## **Введение**

*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.

*Порядок выполнения работы:*

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;

1. 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;

1. 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;
2. Изучить по тексту программного модуля способ реализации вызова действий, определенных в лексических правилах и алгоритм работы формирователя лексем.
3. 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).
4. Подготовить, сдать и защитить отчет к лабораторной работе.

## **Вариант**

|  |  |
| --- | --- |
| Идентификаторы | $<пЦ><пБ>  ($1ad, $74c, $0B…) |
| Константы | целые по основаниям 2,8 и 10;  вещественные;  символьные |
| Объявление примитивных типов | Целое – long[\_u]  Вещественное – number  Символьное – char |
| Оператор присваивания | <И> :**=** <В> |
| Условный оператор | **when**<ЛВ> **then** <ОБ>**[ other**<ОБ> **]** |
| Оператор цикла | **while(**<ЛВ>**) do**<ОБ> |
| Оператор переключателя | **case**<В> **{ when**<К> **then** <ОБ>    **[ exit; ]** …**[** **other** <ОБ> **]** **}** |
| Формат псевдокода | Триады <Код><Оп><Р> |

Обозначения:

**[...]** – необязательная часть конструкции;

**…** – предшествующая часть конструкции повторяется произвольное количество раз;

< > – описание сокращений:

<пБ>|<пЦ> – непустая последовательность букв | непустая последовательность цифр

<И> – Идентификатор (имя переменной / объекта); <К> – Константа;

<В> – произвольное Выражение;

<ЛВ> – Логическое Выражение;

<ОБ> – Оператор или Блок;

<О> – одиночный оператор;

<ОП> – оператор присваивания;

<Код> – поле кода операции;

<М> – поле метки;

<Оп> – поле наименования операнда;

<Р> – поле наименования результата

## **Ход работы**

В данной лабораторной работе были доработаны правила лексики проектируемого языка программирования, на основе лабораторной работы №1.

Приведем подробное описание проектируемого языка программирования. Для удобства описания вместо названия проектируемого языка используем кодовое название **Clap**.

## *Лексика*

В языке **Clap** имеется шесть классов лексем: идентификаторы, ключевые слова, константы, текстовые константы, знаки операций и прочие разделители (комментарии, символы табуляции, символы перевода строки и т.д.). Важно отметить, что в проектируемом языке допустимы только буквы латинского алфавита.

* Комментарии

В языке **Clap** имеют место не только однострочные комментарии, но и комментарии заключенные в контейнер.

Однострочные обозначаются совокупностью знаков **//**и следующий после них текст игнорируется вплоть до символа перевода строки. Пример закомментированной строки: *// Комментарий*.

Многострочные обозначаются совокупностью знаков **/\***и следующий после них текст игнорируется вплоть до закрывающего символа **/\***. Пример таких коментарий: */\*Комментарий*\*/.

* Идентификаторы (имена)

Идентификатор в **Clap** - это последовательность букв и цифр. Первым символом должен быть символ **$**. Далее должна идти непустая последовательность цифр (включает в себя последовательность или одиночный символ). После должна идти непустая последовательность букв (прописные и строчные буквы различаются, может присутствовать может как одиночный, так и последовательность символов). Ограничение на длину имени отсутствует. Пример идентификатора: *$4M.*

* Ключевые слова

Следующие имена в языке **Clap** могут использоваться исключительно, как ключевые слова и не могут быть использованы в другом смысле:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| -- | **Types** | **Сonditional op** | **Cycle op** | **Switch op** | **System** |
| 1 | long | when | while | case | function |
| 2 | char | then | do | when |  |
| 3 | number | other |  | then |  |
|  |  |  |  | exit |  |
| -- | **Another+** | **Another+** |  | other |  |
| 6 | const | not |  |  |  |
| 7 |  | or |  |  |  |
| 8 |  | and |  |  |  |
| 9 |  |  |  |  |  |

* Константы

В языке **Clap** имеется несколько видов констант.

Целые константы по основанию 10 состоят из одной цифры или последовательности цифр, причем данная константа не может начинаться с 0. Примеры константы по основанию 10: *10235, 1, 23.*

Целые константы по основанию 8 состоят из последовательности цифр от 0 до 7, которой предшествует **8x**. Иначе, число не будет трактоваться, как восьмеричное. Пример восьмеричной константы: *8х0157, 8х77.*

Целые константы по основанию 2 состоят из последовательности цифр 0 и 1, которой предшествует **2х**. Пример двоичной константы: *2х0101.*

Символьные константы представляют собой символ или последовательность символов, заключенных в одинарные кавычки. Пример символьной константы: *‘String value’.*

Вещественные константы состоят из целой части, десятичной точки, дробной части. Целая и дробная части являются последовательностями цифр. Пример вещественной константы: *3.14.*

## *Типы данных*

Язык **Clap** поддерживает несколько базовых типов объектов.

Целочисленный тип данных имеет обозначениt: *long[\_u]*. С помощью нее описываются все целочисленные переменные, а также константы по основанию 8 и 2.

Вещественный тип определяется ключевым словом *number.* Под этот тип попадают вещественные константы и могут быть описаны с помощью цифр и точки, разделяющей дробную и целую часть.

Символьный тип обозначается ключевым словом *char*, с помощью которого можно описать переменную, которой можно присвоить символ или последовательность символов, заключенных в одинарные кавычки.

## *Операции*

Выражение состоит из одного или нескольких операндов (α, β) и символов операций. В данном языке операнд может быть целым, вещественным или символьным.

* Арифметические операции

Таблица 1. Арифметические операции

|  |  |
| --- | --- |
| *Оператор* | *Значение* |
| α + β | **Сложение**, результат – сумма двух значений; |
| α \* β | **Произведение** |
| α - β | **Вычитание**, результат – разность двух значений; |
| α / β | **Частное от деления**, если операнды целого типа, то деление с отбрасыванием остатка; |
| α+ + | **Инкремент**, приращение значения на 1 |
| α - - | **Декремент**, обратное приращение на 1 |
| -α | **Изменение знака** |

* Операции сравнения

Операндами операций сравнения могут быть любые арифметические выражения. Результатом операций сравнения всегда является целое выражение со значением 0, обозначающим *ложь*, и – 1, обозначающим *истину*.

Таблица 2. Операции сравнения

|  |  |
| --- | --- |
| *Оператор* | *Значение* |
| α = = β | **Равно**; результат равен 1, если значения операндов равны, в противном случае результат равен 0; |
| α != β | **Не равно**; результат равен 1, если значения операндов не равны, в противном случае результат равен 0; |
| α < β | **Меньше**; результат равен 1, если значение первого операнда меньше значения второго, в противном случае результат равен 0; |
| α > β | **Больше**; результат равен 1, если значение первого операнда больше значения второго, в противном случае результат равен 0; |
| α <= β | **Меньше или равно**; результат равен 1, если значение первого операнда меньше или равно значению второго, в противном случае результат равен 0; |
| α >= β | **Больше или равно**; результат равен 1, если значение первого операнда больше или равно значению второго, в противном случае результат равен 0. |

* Логические операции

Операндами логических операций могут быть любые арифметические выражения; операнд считается *ложью*, если он содержит нулевое значение, и *истиной* в противном случае. Результатом логических операций всегда является целое выражение со значением 0, обозначающим *ложь*, и – 1, обозначающим *истину*.

Таблица 3. Логические операции

|  |  |
| --- | --- |
| *Оператор* | *Значение* |
| not | **Инверсия**, не нулевое значение даст в результате 0, а нулевое – 1; например: если *α =1*, то *not α* будет равно *0*; |
| and | **Конъюнкция** или логическое умножение; например: если α =5, β=0, θ=2, то результат *α and β* равен *0*, а результат α *and* θ равно *1*; |
| or | **Дизъюнкция** или логическое сложение; например: если *α* =5, β=0, θ=2, то результат *α or β* равен *1*, а результат *α or θ* равно *1*; |

## *Операторы*

* **Оператор присваивания**

α := β – в языке **Clap** обозначает операцию присваивания. Это значит, что выражение α := β имеет значение, и это значение равно значению правой части операции присваивания, т.е. в данном случае значению β.

* **Условный оператор**

Условный оператор выполняет действия в том случае, когда заданное условие – истина. Возможен как неполный, так и полный условный оператор.

Структура неполного оператора:

*when (<ЛВ>) then <ОБ>;*

Структура полного оператора:

*when (<ЛВ>) then <ОБ 1>;*

*other <ОБ 2>;*

Описание уловного оператора начинается с ключевого слова *when*, которое предшествует логическому выражению. Далее следует ключевое слово *then*,после которого пишется оператор или действия. На этом заканчивается описание неполного условного оператора. В случае использования полного оператора, после первого блока действий и оператора следует ключевое слово other и другой блок действий.

* Оператор цикла

Оператор цикла выполняет заданные действия с определенным шагом.

Структура неполного оператора:

*while <ЛВ> do {*

*<ОБ>*

*}*

Как видно из структуры описание оператора цикла начинается с ключевого слова, после которого описывается условие поведения цикла. После ключевого слова *do* идет блок обработки кода заданного для данного условия. Окончанием работы цикла будет срабатывание триггера блока while, когда условие внутри будет выполнено полностью.

* Оператор переключателя(**Switcher**)

Оператор перехода или оператор переключения выполняет разные заданные действия независимо друг от друга, при определенном результате заданного выражения.

Структура оператора:

*case <В>{*

*when (<К>)*

*then <ОБ>*

*exit;*

*when (<К1>)*

*then <ОБ1>*

*exit;*

*other<ОБ2>*

*}*

Оператор начинается с ключевого слова *case* со следующим за ним выражением. После ключевого слова *when* в круглых скобках пишется константа, которая может соответствовать значению выражения. После do, описывается действие, которое должно быть выполнено при входе в блок *when*. Любой блок *when* после его выполнения можно закончить с помощью ключевого слова *exit*.

## *Функции*

* Язык **Clap** использует функции, которые могут возвращать значения и принимать их в качестве аргументов при вызове. Пример объявления и вызова функции:

*function sum(long $1a,long $1b){*

*$1a := $1a +$1b;*

*return($1a);*

*}*

*function main(){*

*print(sum(15,35));*

*}*

* name(ident *$1a*) - таким образом происходит передача аргументов в функцию.
* return - возвращает значение, которое может быть полученным внутренними расчетами функции.

## *Элементы языка*

Язык Clap использует классическое стандартное множество символов, которые используются для создания идентификаторов и ключевых слов, а также для разделения текста программы, написания выражений и др.

Таблица 4. Символы

|  |  |
| --- | --- |
| Латинские прописные буквы | A B C D E F G H I J K L M N O P Q R S T U V W X Y Z |
| Латинские строчные буквы | a b c d e f g h I g k l m n o p q r s t u v w x y z |

Таблица 5. Цифры

|  |  |
| --- | --- |
| Арабские цифры | 0 1 2 3 4 5 6 7 8 9 |

Таблица 6. Символы

|  |  |  |  |
| --- | --- | --- | --- |
| $ | \* | { | , |
| + | / | } | < |
| - | ( | ‘ | > |
| = | ) | . | ! |
| [ | ] | ; |  |

Таблица 7. Управляющие

|  |  |
| --- | --- |
| Пробел |  |
| Табуляция | \t |
| Возврат каретки | \r |
| Перенос строки | \n |

## *Структура программы*

Исходный файл программы на языке **Clap** содержит следующие разделы:

* Раздел описаний

В данном разделе происходит объявление и описание переменных и констант.

* Раздел функций

Данный раздел содержит полное определение и описание функции, определение используемых ею переменных. Может быть пропущен.

* Основной раздел

В данном разделе содержится сам код, где и будут использованы все объявленные в первом разделе переменные, константы и функции.

Объявление переменных начинается с ключевого слова, определяющего тип переменной, далее следует идентификатор переменной.

Определение константы происходит с использованием ключевого слова *const*, после которого пишется тип, идентификатор и значение константы.

Объявление функции начинается с ключевого слова *function*, после чего идет тип возвращаемого значения, название функции. В скобках после названия объявляются используемые внутри функции переменные и их тип. В начале и конце блока действий ставятся фигурные скобки.

## *Пример кода*

function() {

number $1b;

number $1k;

number $2k;

char $12c;

char $0k;

char $0i;

char $0j;

while($0n < 10000) do{

$12c := 0;

$2k := n0n;

}

while(test > 0) do{

$12c[$0k+1] := $1b / 10; //справить

$1b := 10;

}

$1k := 0;

while($0i < $0k) do{ //comment

$2k := 1;

/\* возводим в степень \*/

while($0j < $0k) do {$2k := $2k \* $12c[$0i];

//складываем с предыдущим

$1k := $1k + $2k;

}

if($1b == $1k) $0i << $0k;

}

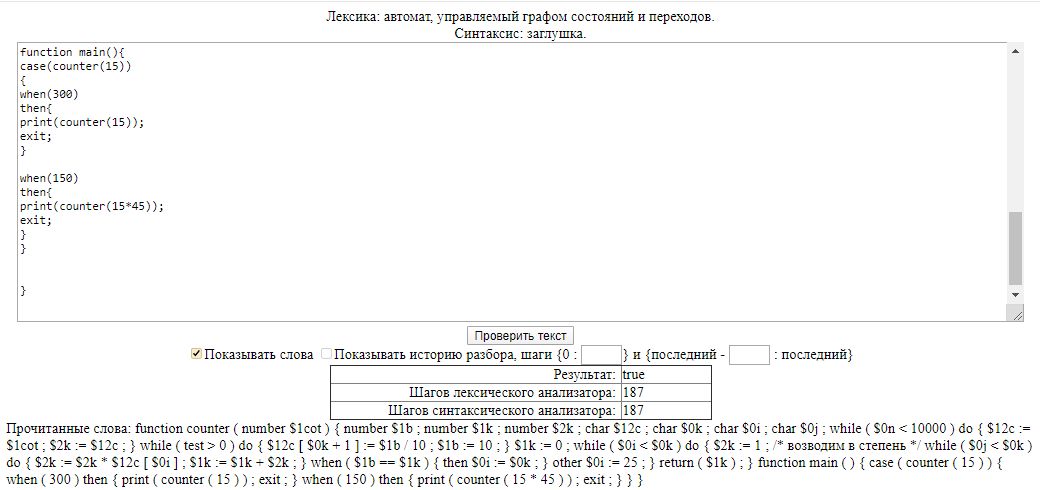
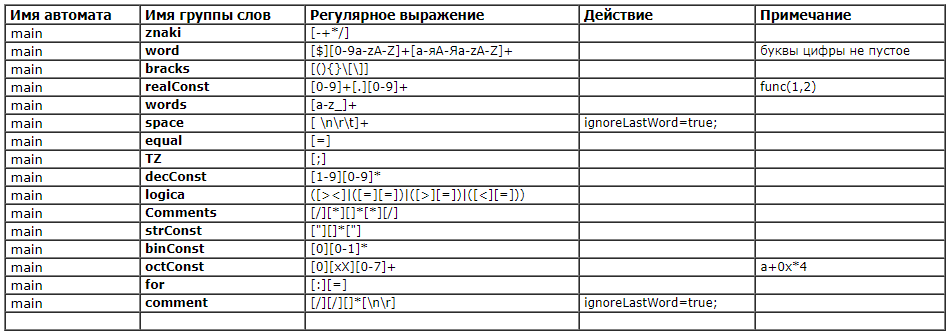
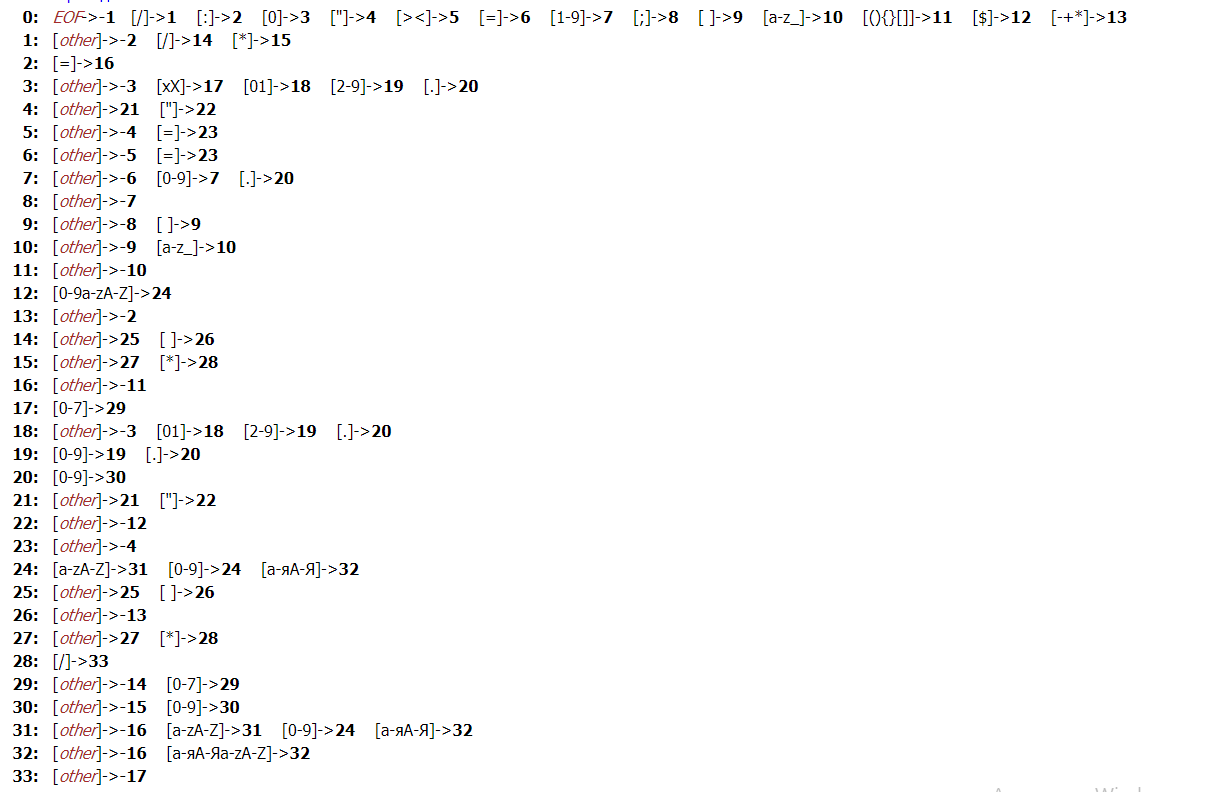
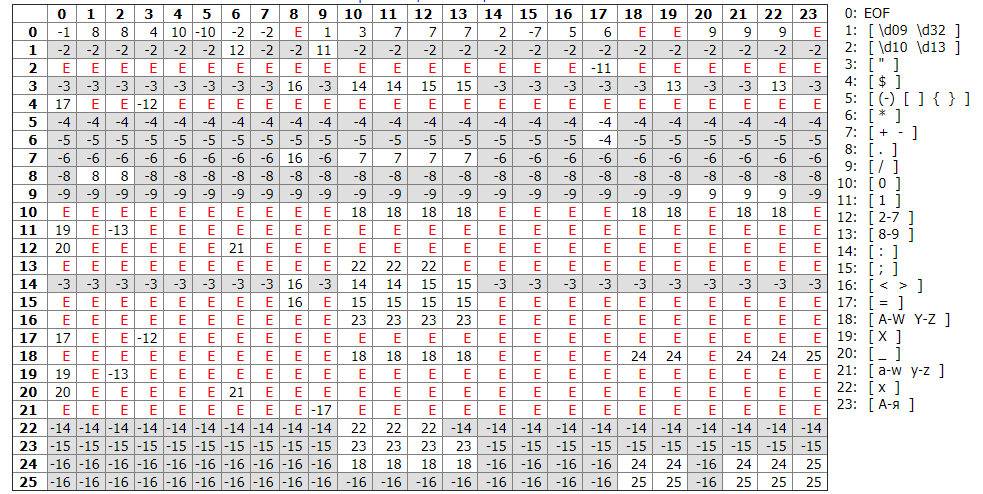
} 

Рисунок 1. Проверка кода

Рисунок 2. Доработанный набор правил

 Рисунок 3. Графовый КА

Рисунок 4. Табличный КА

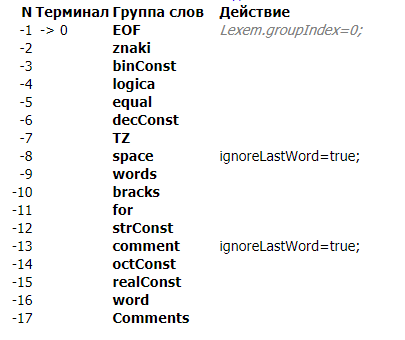


Рисунок 5. Финальные состояния

## **Вывод**

В результате выполнения лабораторной работы были доработаны правила лексики языка программирования. Были описаны правила разрабатываемого языка программирования. Были изучены графовый и табличный способ определения КА.