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 работе была доработана лексика языка, предлагаемого вариантом задания на курсовую работу.

Тестовая программа предлагается к рассмотрению в Приложении 1, правила лексики разрабатываемого языка в Приложении 2.

**Описание языка**

На данный момент язык 33312212 способен воспринимать программы, состоящие из одного файла. В таких файлах может быть любое количество функций, последней из которых должна быть функция main, с нее начнется выполнение программы. Выражения разделяются символом ‘;’.

Типы данных:

1. entire – целочисленный
2. symbol – символьный
3. real – вещественный
4. entire unsigned – целочисленный беззнаковый

Математические операции:

1. + - сложение
2. – - вычитание
3. \* - умножение
4. / - деление
5. % - остаток от деления

Оператор присваивания: =

Возможность вставки комментариев, которые будут игнорироваться при трансляции программы: /\* комментарий \*/

Оператор цикла: while(<логическое выражение>) do <оператор или блок>

Условный оператор: if (<логическое выражение>) <оператор или блок> [ ifnot<оператор или блок> ] fi

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

Блоки: {<блок>}

Константы:

1. По основанию 4
2. По основанию 8
3. По основанию 10
4. Вещественные
5. Символьные
6. Строковые

Структура идентификаторов: <одна маленькая буква><непустая последовательность цифр><непустая последовательность цифр>. Пример записи: a8FN

Структура функций: <возвращаемый тип данных> <имя> (набор аргументов) {Тело}

Для того, чтобы вернуть значение из функции, используется ключевое слово return. Пример записи:real myfunc (real arg){return 0;}

Для вывода в стандартный поток вывода используется конструкция cout <<

Для чтения из стандартного потока ввода используется конструкция cin >>

В языке реализованы наиболее распространенные логические операторы

В языке реализованы глобальная и локальная области видимости. Объявление глобальных переменных происходит перед описанием всех функций. Такие переменные являются видимыми в любой части программы. Локальные переменные объявляются внутри функции. Такие переменные видны в том блоке, в котором они объявлены. Их инициализация происходит во время выполнения функции. После завершения выполнения функции такие переменные удаляются из памяти.

Для вызова функции используется конструкция ИмяФункции(Параметры), где в круглых скобках передаются аргументы с указазанными типами данных.

# Выводы

В данной работе были изучены конечные автоматы (КА) без памяти, способы определения КА – канонического, графового и табличного, методы построения недетерминированного КА по системе регулярных выражений, методы эквивалентных преобразований недетерминированных КА в оптимальные полностью определенные КА – лексические акцепторы.

Был доработан собственный конечный автомат и написана тестовая программа на разработанном языке.

**Приложение**

**Приложение 1.**

Программа находит максимальное число, меньшее заданного, которое равно сумме его десятичных цифр, возведенных в степень, большую 1

entire sumOfNumbers(entire\_u i1NT) {

entire s7M = 0;

while (i1NT > 0) {

s7M = s7M + i1NT % 10;

i1NT = i1NT / 10;

}

return sum;

}

real usingCPU(entire\_u n8W) {

entire m4X = 0;

entire m4XEXP = 0;

cout << "Using CPU.";

real s7ART;

real e2D;

s7ART = omp\_get\_wtime();

entire i1NT = 0;

while (i1NT < n8W)

do {

entire s9M = sumOfNumbers(i1NT);

entire unsigned r3SULT = 0;

entire e8P = 2;

while (r3SULT < e8P)

do {

r3SULT = s9M \* e8P;

if (i1NT == r3SULT)

m4X = i1NT;

m4EXP = e8P;

fi

e8P = e8P + 1;

}

i1NT = i1NT + 1;

}

e2D = omp\_get\_wtime();

real t1ME = e2D - s7ART;

if (n4X != 0)

cout << "Max number using CPU: " << m4X << " Max exponent: " << m4XEXP;

ifnot cout << "Max number not found.";

fi

return t1ME;

}

enitre main()

{

entire unsigned n5S;

cout << "Enter N: ";

cin >> n5S;

c6UTIME = usingCPU(n5S);

cout << "CPU time: " << c9UTIME;

return 0;

}**Приложение 2**

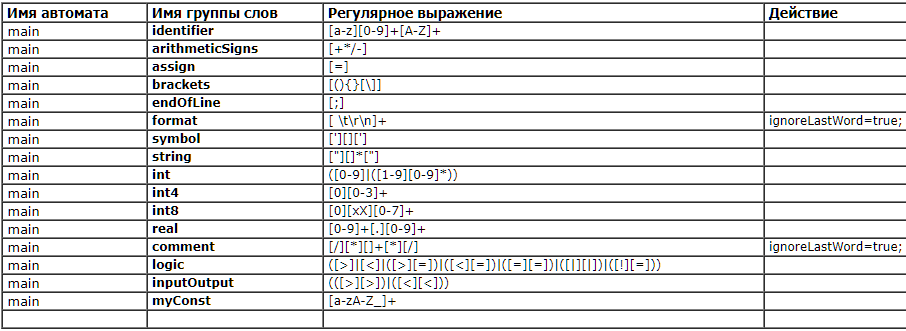


Рис. 1 – Правила лексики.

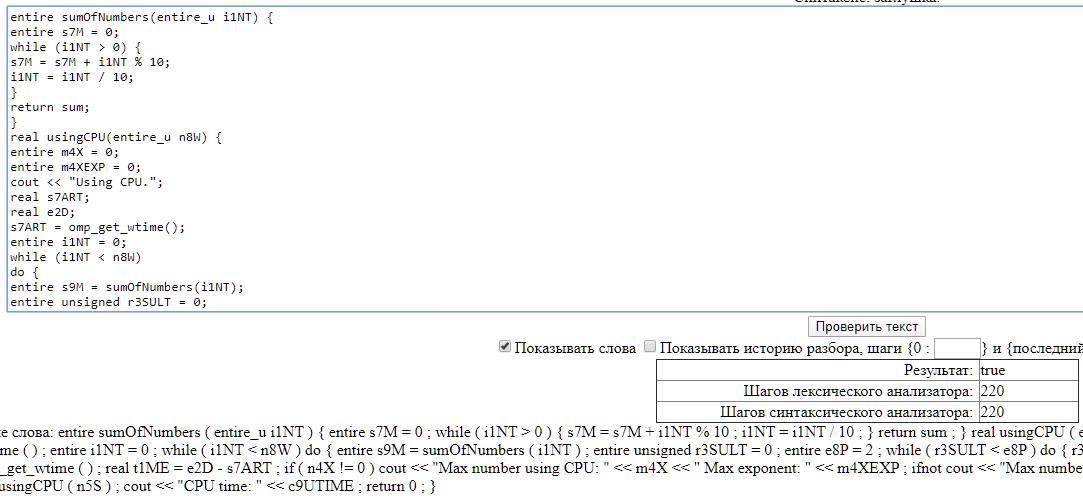


Рис. 2 – Проверка лексики программы, написанной на языке 33312212.

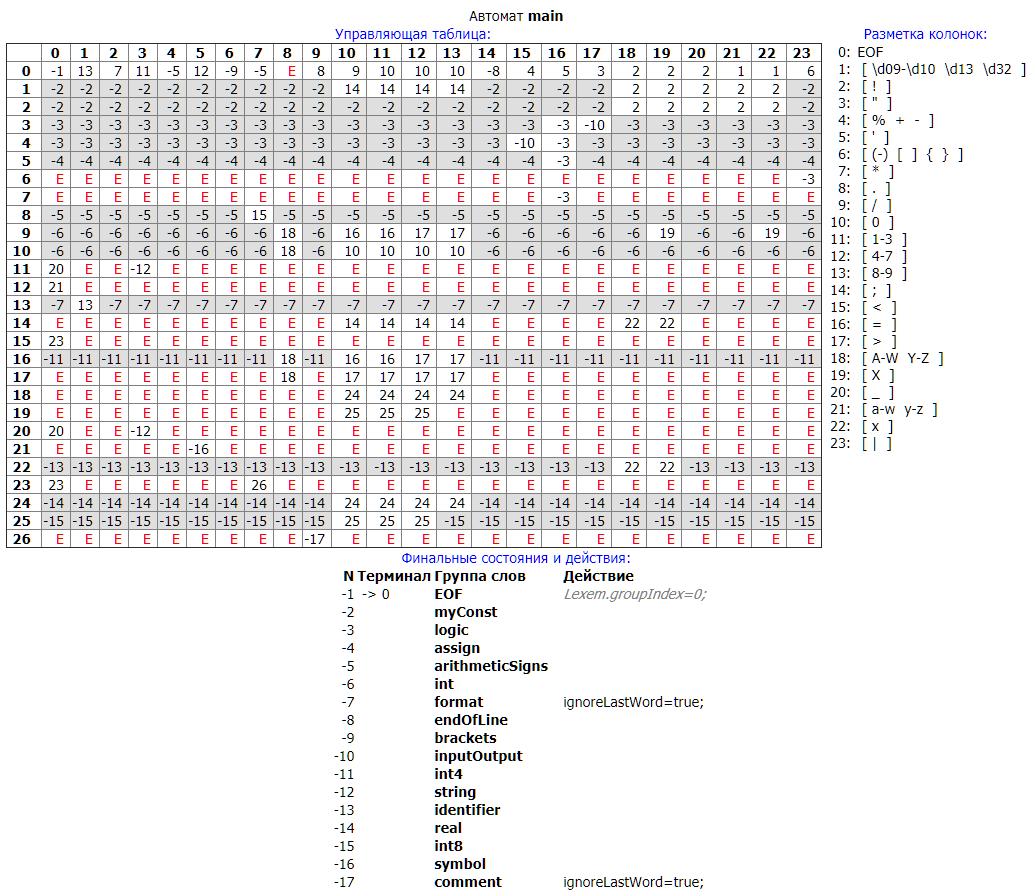


Рис. 3 – Управляющая таблица лексического анализатора.

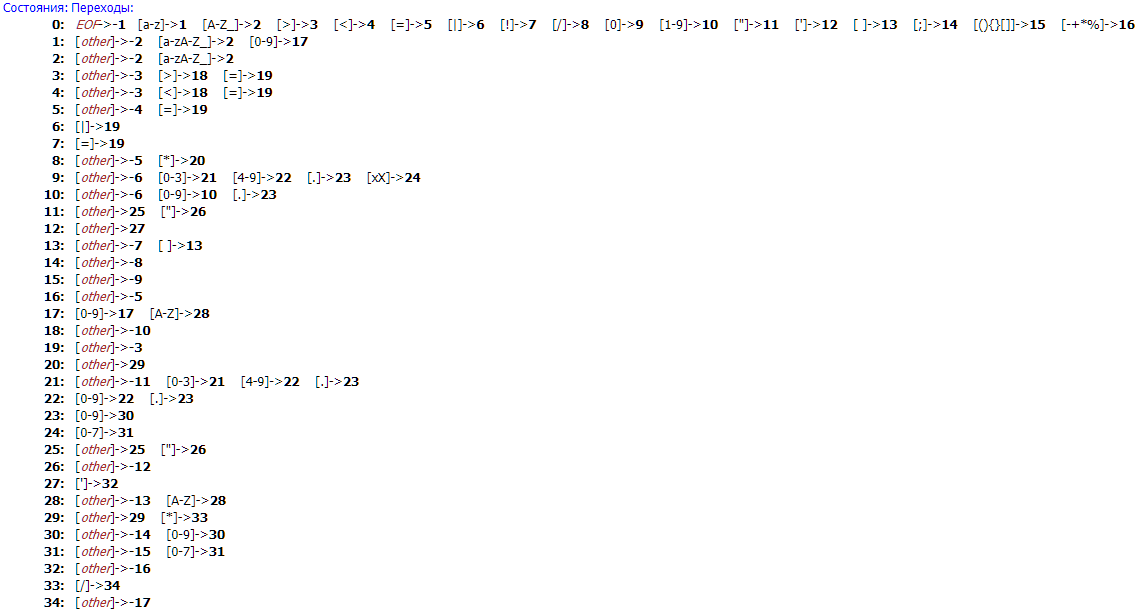


Рис. 4 – Список состояний и переходов лексического анализатора.

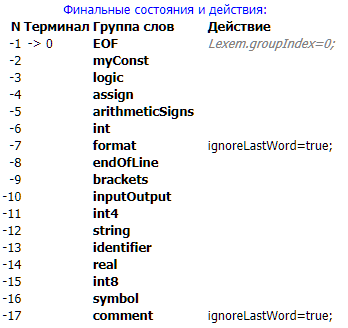


Рис. 5 – Финальные состояния и действия лексического анализатора.