Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра «Информационных технологий и систем»

**Проектирование лексического анализатора**

Лабораторная работа

по дисциплине:

«Теория языков программирования и методы трансляции»

Разработал:

Студент группы 8091

Григорьев Д. И. \_\_\_\_\_\_

«\_\_» \_\_\_\_\_2020г

Проверил:

Макаров В. А. \_\_\_\_\_\_

«\_\_» \_\_\_\_\_2020г

**Великий Новгород**

**2020**

**Постановка задачи**

**Формулировка**

Разработать программу, решающую задачу лексического разбора исходного текста программы. Распознаваемый язык выбирается самостоятельно. В качестве прототипов может быть выбран синтаксис следующих языков программирования: Паскаль, СИ, Бейсик.

**Представление входных данных**

В виде входных данных программе подается текст. Для успешного распознавания он должен удовлетворять правилам грамматики из файла *grammar.txt* в папке *attachments*. Описанная в данном файле грамматика выбрана на основе варианта №4(арифметические выражения в языке C) с добавлением своих особенностей.

Алфавит состоит из: [[a-z], [A-Z], [0-9]], [int, float, double, char, short, long, string, if, else], ['**(**', '**)**', '**[**', '**]**', '**{**', '**}**', '**.**', '**,**', '**:**', '**;**'], [‘**+**’, ‘**-**‘, ‘**\***’, ‘**/**’, ‘**%**’, ‘**|**’, ‘**&**’, ‘!’], [‘**<**’, ‘>’, ‘=’], [‘\_’]

Классы лексем:

1. Идентификаторы;
2. Ключевые слова;
3. Числовые константы;
4. Знаки операций;
5. Разделители.

**Описание выбранного метода решения задачи ЛА**

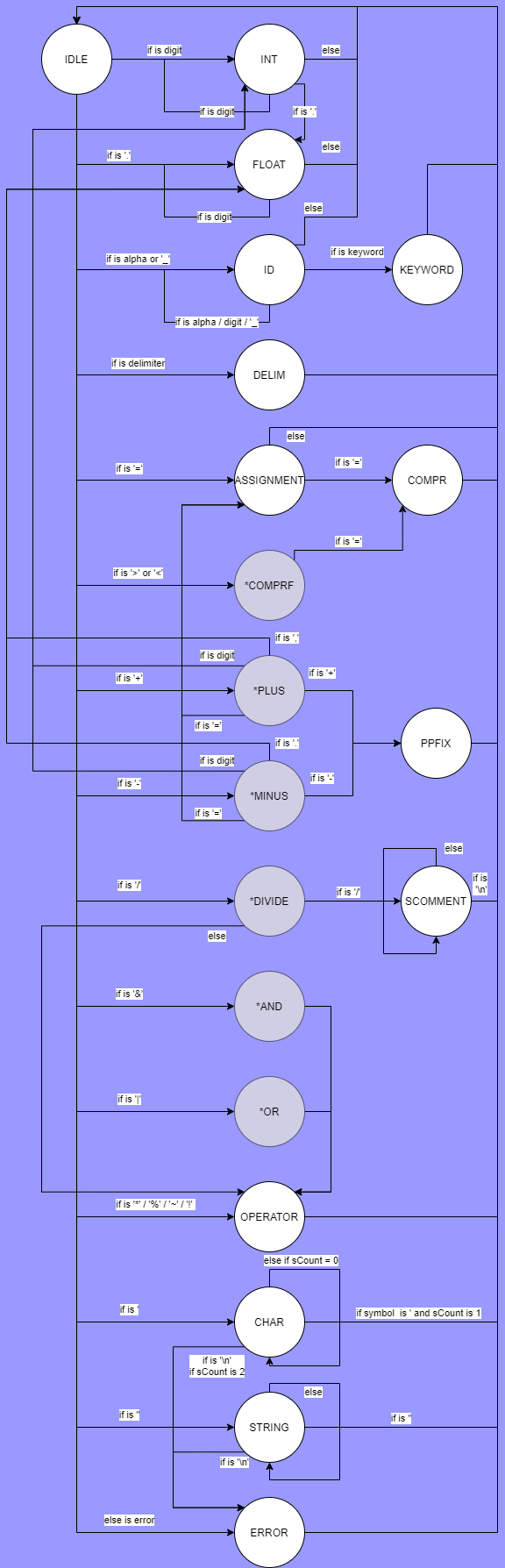
Для решения задачи лексического анализатора был выбран конечный автомат, поскольку его использование позволяет обрабатывать сложные константы (например, числа с плавающей точкой со знаком, порядком и мантиссой).

Для хранения состояний были использованы enum, для хранения идентификаторов были использованы словари (dict), для знаков операций, сравнения, разделителей – списки.

Для удобства были реализованы классы StateMachine (КА), Lexem.

**Описание реализации**

Для решения задачи лексического разбора был построен граф с 6-ю вспомогательными и 14-ю «основными» состояниями. С помощью вспомогательных состояний удалось определить: “<=”, “>=”, “++”, “--", “||”, “&&”, а также комментарии. Итоговый алгоритм был разработан в соответствии с схемой, представленной ниже (файл *FSMgraph.png* в папке *attachment*). Таблица с лексемами выводится в отдельный файл (*result.csv)*.



**Обработка ошибок**

Лексема помечается «ошибочной», если следующий рассматриваемый символ будет недопустимым, а также если в начальном состоянии придет символ, необрабатываемый программой.

**Методика тестирования**

При тестировании было использовано два файла (*example.c* и *digits.c*) со всеми предусматриваемыми правилами.

**Вывод**

В итоге проделанной работы был разработан лексический анализатор на основе рассматриваемой грамматики, позволяющий распознать простейшие лексемы, чтобы в будущем использовать их для компиляции программы.