

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 3**

Лексический анализатор на основе регулярных выражений

по дисциплине «Конструирование компиляторов»

Вариант 9

Работу выполнил

студент группы ИУ9-62Б

Жук Дмитрий

Москва, 2022

**Цель работы**

Целью данной работы является приобретение навыка разработки простейших лексических анализаторов, работающих на основе поиска в тексте по образу, заданному регулярным выражением.

**Задание**

В лабораторной работе необходимо реализовать две первые фазы стадии анализа: чтение входного потока и лексический анализ. Чтение входного потока должно осуществляться из файла (в UTF-8), при этом лексический анализатор должен вычислять текущие координаты в обработанном тексте. Лексический анализатор должен иметь программный интерфейс для взаимодействия с парсером, реализованный через итератор.

**Индивидуальный вариант**

Идентификаторы: последовательность буквенных символов Unicode, цифр и дефисов, начинающихся с заглавной буквы. Директивы: любой знак валюты, после которого следует непустая последовательность заглавных букв.

**Реализация**

/\* eslint-disable no-console, import/prefer-default-export \*/

import fs from 'fs';

enum TokenType {

IDENT = 'ident',

DIR = 'dir',

}

export const parseToken = (s: string) => ({

\* [Symbol.iterator](): Generator<{type: TokenType, coord: [number, number], value: string}> {

const r = /(?<ident>^\p{L}[\p{L}0-9-]\*)|(?<dir>^\p{Sc}\p{Lu}+)|(?<newline>^\n)|(?<space>^\s)|(?<err>^.)/u;

for (let e = r.exec(s), line = 1, pos = 1; e?.groups; e = r.exec(s)) {

if (e.groups.ident) {

yield { type: TokenType.IDENT, coord: [line, pos], value: e.groups.ident };

} else if (e.groups.dir) {

yield { type: TokenType.DIR, coord: [line, pos], value: e.groups.dir };

} else if (e.groups.newline) {

line++;

pos = 0;

// yield {type: 'newline', coord: [line, pos], value: e.groups.newline}

} else if (e.groups.space) {

// yield {type: 'space', coord: [line, pos], value: e.groups.space}

} else if (e.groups.err) {

console.error('err: find trash symbol in', { token: e.groups.err });

// yield {type: 'err', coord: [line, pos], value: e.groups.err}

}

s = s.slice(e.index + e[0].length);

pos += e[0].length;

}

},

});

console.log([...parseToken(fs.readFileSync(process.argv[2], { encoding: 'utf8' }))]);

Листинг 1 — Код программы

**Вывод**

В ходе лабораторной работы было приобретен навык разработки простейших лексических анализаторов, работающих на основе поиска в тексте по образу, заданному регулярным выражением. Так же был усвоен навык написания итераторов на языке JavaScript.