Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы трансляции

ОТЧЁТ

к лабораторной работе №3

на тему

СИНТАКСИЧЕСКИЙ АНАЛИЗАТОР

Выполнил студент гр.153502 Толстой Д.В.

Проверил ассистент кафедры информатики Гриценко Н.Ю.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc158758016)

[2 Описание функций программы 4](#_Toc158758017)

[Приложение А (обязательное) Листинг исходного кода 6](#_Toc158758018)

1 ФОРМУЛИРОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является разработка синтаксического анализатора подмножества языка программирования, определенного в лабораторной работе 1. В ходе синтаксического анализа исходный текст программы проверяется на соответствие синтаксическим нормам языка с построением дерево разбора (синтаксическое дерево), которое отражает синтаксическую структуру входной последовательности и удобно для дальнейшего использования, а также в случае несоответствия – позволяет вывести сообщения об ошибках.

Как правило, результатом синтаксического анализа является синтаксическое строение предложения, представленное либо в виде дерева зависимостей, либо в виде дерева составляющих, либо в виде некоторого сочетания первого и второго способов представления.

2 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Программа реализует перевод потока лексем в абстрактное синтаксическое дерево методом рекурсивного спуска. При успешном выполнении программа выводит в консоль абстрактное синтаксическое дерево. Вывод успешно выполненной программы предоставлен на рисунке 1.

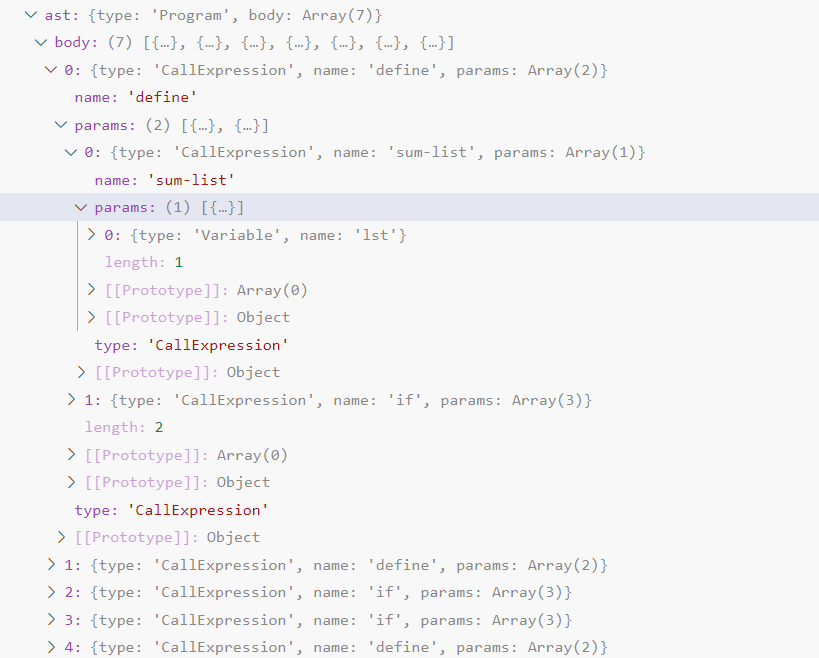


Рисунок 1 – Вывод программы

На рисунке 2 предоставлена реакция программы на незакрытую скобку.

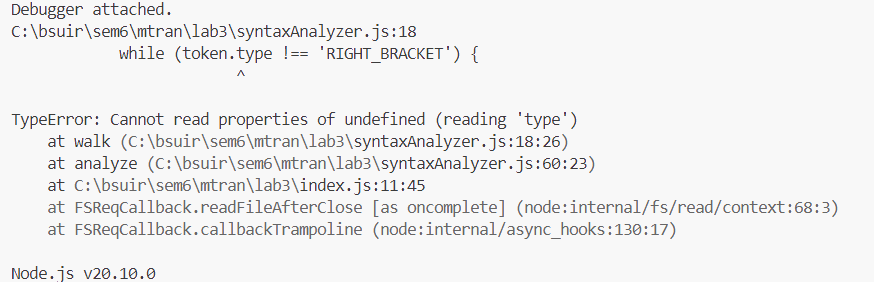


Рисунок 2 – Реакция программы на незакрытую скобку

На рисунке 3 предоставлена реакция программы на случайный символ в коде.

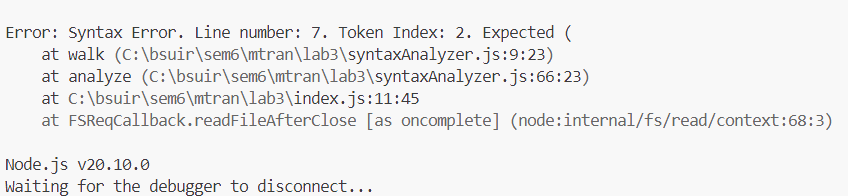


Рисунок 3 – Реакция программы на случайный символ в коде

ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг исходного кода**

Файл index.js

const tokenize = require('../lab2/analyzer');

const analyze = require('./syntaxAnalyzer');

const fs = require('fs');

fs.readFile('code.txt', 'utf8', (err, data) => {

    if (err) {

      console.error(err);

      return;

    }

    let [answer, normalAnswer] = tokenize(data);

    fs.writeFile('tree.txt', JSON.stringify(analyze(normalAnswer), null, 2), function(error){

        if(error){

            return console.log(error);

        }

        console.log("Файл успешно записан");

    });

});

Файл syntaxAnalyzer.js

function analyze(tokens) {

    let current = 0;

    function walk(topLevelExpression = false) {

        let token = tokens[current];

        if (topLevelExpression) {

            if (token.type !== 'LEFT\_BRACKET') {

                throw new Error(`Syntax Error. Line number: ${token.lineNumber}. Token Index: ${token.tokenIndex}. Expected (`);

            }

        }

        if (token.type === 'LEFT\_BRACKET') {

            token = tokens[++current];

            const node = {

                type: 'CallExpression',

                name: token.element,

                params: []

            };

            token = tokens[++current];

            while (token.type !== 'RIGHT\_BRACKET') {

                node.params.push(walk());

                token = tokens[current];

            }

            current++;

            return node;

        }

        if (token.type === 'SYS\_FUNC' || token.type === 'OPERATOR' || token.type === 'KEYWORD') {

            current++;

            return {

                type: 'Identifier',

                name: token.element

            };

        }

        if (token.type === 'IDENTIFICATOR') {

            current++;

            return {

                type: 'Variable',

                name: token.element

            };

        }

        if (token.type === 'LITERAL\_NUMBER' || token.type === 'LITERAL\_STRING' || token.type === 'CONSTANT') {

            current++;

            return {

                type: 'Literal',

                value: token.element

            };

        }

        throw new Error(`Syntax Error. Line number: ${token.lineNumber}. Token Index: ${token.tokenIndex}. Unknown token type: ${token.type}`);

    }

    const ast = {

        type: 'Program',

        body: []

    };

    while (current < tokens.length) {

        ast.body.push(walk(true));

    }

    return ast;

  }

  module.exports = analyze;