



МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной техники

ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ №3

«Анализатор JSON файла, обработка лексем»

по дисциплине

«Теория формальных языков»

Выполнил студент группы ИКБО-04-21

Исаев В. В.

Принял ассистент

Боронников А. С.

Практическая работа
выполнена

«__»_____2022 г.

«Зачтено»

«__»_____2022 г.

Задание

На любом языке программирования (или с помощью FLEX-лексического анализатора) реализовать простой анализатор JSON файла. Обработка несуществующей лексемы: либо завершить выполнение программы с соответствующим сообщением, либо вывести лексему без соотношения к определенному классу токенов.

Лексемы:

- СИМВОЛЫ:
 - BEGIN_OBJECT ({);
 - END_OBJECT (});
 - BEGIN_ARRAY ([);
 - END_ARRAY (]);
 - COMMA (,);
 - COLON (:);
- литералы
 - LITERAL (true, false, null);
- строки
 - STRING (“string”);
- числа
 - NUMBER (1, -1, +1, 1e1000).

Пример работы программы:

ввод (через файл или консоль):

```
{
"name": "George",
"age": 19e5,
"EducationalEstablishments": [ "School2200", "UniversityMIREA" ],
"Student": true
}
```

Реализация решения задания на языке C#

```
public static string word_read;
```

```

public static char[] word_char;

public static lexems[] lexems_tokens=new lexems[100];
public static string[] tokens = new string[100];
public enum lexems
{
    BEGIN_OBJECT,
    END_OBJECT,
    BEGIN_ARRAY,
    END_ARRAY,
    COMMA,
    COLON,
    LITERAL,
    STRING,
    NUMBER
}
enum states
{
    H,
    OPEN_BRACKET,
    CLOSE_BRACKET,
    OPEN_SQUARE,
    CLOSE_SQUARE,
    STRING,
    ENUMERABLE,
    REZ,
    BOOLEAN,
    NUMBER
}
public static bool getWord(string a)
{
    if(a=="true"||a=="false"||a=="null")
    {
        return true;
    }
}

```

```

        }
    else
    {
        return false;
    }
}

static void Main(string[] args)
{
    string buf="";
    word_read = File.ReadAllText("word.txt");
    word_char = word_read.ToCharArray();
    int cur_token = 0;
    var a = states.H;
    int cur_pos = 0;
    while(cur_pos<word_char.Length)
    {
        switch(a)
        {
            case states.H:
                if(word_char[cur_pos] == ' '||
word_char[cur_pos] == '\\t' || word_char[cur_pos] == '\\n' ||
word_char[cur_pos] == '\\r')
                {
                    cur_pos++;
                }
                else if(word_char[cur_pos]=='{')
                {
                    a = states.OPEN_BRACKET;
                }
                else if(word_char[cur_pos]=='[')
                {
                    a = states.OPEN_SQUARE;
                }
                if (word_char[cur_pos] == '}')

```

```

        {
            a = states.CLOSE_BRACKET;
        }
        else if (word_char[cur_pos] == ']')
        {
            a = states.CLOSE_SQUARE;
        }
        else if (word_char[cur_pos] == '"')
        {
            a = states.STRING;
        }
        else if (word_char[cur_pos] == ',')
        {
            a = states.ENUMERABLE;
        }
        else if (word_char[cur_pos] == ':')
        {
            a = states.REZ;
        }
        else if (word_char[cur_pos] == 't' ||
word_char[cur_pos] == 'f' || word_char[cur_pos] == 'n')
        {
            buf = "";
            a = states.BOOLEAN;
        }
        else if
(Char.IsDigit(word_char[cur_pos]))
        {
            a = states.NUMBER;
        }
        break;
    case states.OPEN_BRACKET:
        lexems_tokens[cur_token] =
lexems.BEGIN_OBJECT;

```

```

        tokens[cur_token] = "{";
        cur_token++;
        cur_pos++;
        a = states.H;
        break;
    case states.CLOSE_BRACKET:
        lexems_tokens[cur_token] =
lexems.END_OBJECT;

        tokens[cur_token] = "}";
        cur_token++;
        cur_pos++;
        a = states.H;
        break;
    case states.OPEN_SQUARE:
        lexems_tokens[cur_token] =
lexems.BEGIN_ARRAY;

        tokens[cur_token] = "[";
        cur_token++;
        cur_pos++;
        a = states.H;
        break;
    case states.CLOSE_SQUARE:
        lexems_tokens[cur_token] =
lexems.END_ARRAY;

        tokens[cur_token] = "]";
        cur_token++;
        cur_pos++;
        a = states.H;
        break;
    case states.STRING:
        cur_pos++;
        if (word_char[cur_pos] != '"')
        {
            lexems_tokens[cur_token] =

```

```

lexems.STRING;

                                tokens[cur_token] =
tokens[cur_token] += word_char[cur_pos];
                                }
                                else
                                {
                                    cur_token++;
                                    cur_pos++;
                                    a = states.H;
                                }
                                break;
case states.ENUMERABLE:
    lexems_tokens[cur_token] =

lexems.COMMA;

    tokens[cur_token] = ",";
    cur_token++;
    cur_pos++;
    a = states.H;
    break;
case states.REZ:
    lexems_tokens[cur_token] =

lexems.COLON;

    tokens[cur_token] = ":";
    cur_token++;
    cur_pos++;
    a = states.H;
    break;
case states.BOOLEAN:
    buf = buf + word_char[cur_pos];
    if (getWord(buf))
    {
        lexems_tokens[cur_token] =

lexems.LITERAL;

        tokens[cur_token] = buf;

```

```

        cur_token++;
        cur_pos++;
        a = states.H;
    }
    cur_pos++;
    break;
case states.NUMBER:
    if (word_char[cur_pos] != ',' &&
word_char[cur_pos] != ']' && word_char[cur_pos] != '}')
    {
        lexems_tokens[cur_token] =
lexems.NUMBER;

        tokens[cur_token] =
tokens[cur_token] += word_char[cur_pos];
        cur_pos++;
    }
    else
    {
        cur_token++;
        a = states.H;
    }
    break;
default:
    Console.WriteLine("Необработанная
лексема");

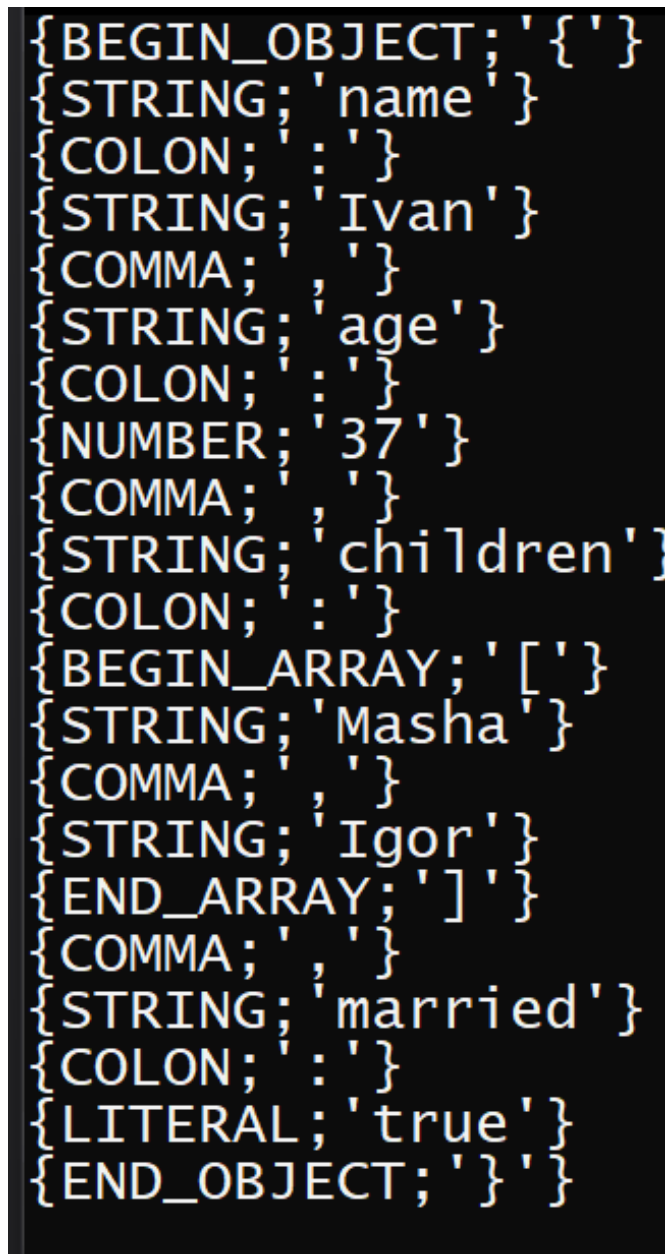
    Console.ReadLine();
    Environment.Exit(0);
    break;
}
}
for(int i=0;i<lexems_tokens.Length;i++)
{
    if (tokens[i] != null)
    {
        Console.WriteLine("{ " + lexems_tokens[i] +

```



```
    ";" + tokens[i] + "'}");  
    }  
}  
Console.ReadLine();  
}
```

Тестирование

A screenshot of a program's output, likely a debugger or console window, showing the step-by-step parsing of a JSON object. The text is displayed in a monospaced font with a yellow/gold color on a black background. The output consists of 28 lines, each representing a token and its corresponding JSON syntax element. The tokens are: BEGIN_OBJECT, STRING, COLON, STRING, COMMA, STRING, COLON, NUMBER, COMMA, STRING, COLON, BEGIN_ARRAY, STRING, COMMA, STRING, END_ARRAY, COMMA, STRING, COLON, LITERAL, and END_OBJECT. The JSON object being parsed is: {"name": "Ivan", "age": 37, "children": ["Masha", "Igor"], "married": true}.

```
{BEGIN_OBJECT; '{'}  
{STRING; 'name'}  
{COLON; ':'}  
{STRING; 'Ivan'}  
{COMMA; ','}  
{STRING; 'age'}  
{COLON; ':'}  
{NUMBER; '37'}  
{COMMA; ','}  
{STRING; 'children'}  
{COLON; ':'}  
{BEGIN_ARRAY; '['}  
{STRING; 'Masha'}  
{COMMA; ','}  
{STRING; 'Igor'}  
{END_ARRAY; ']'}  
{COMMA; ','}  
{STRING; 'married'}  
{COLON; ':'}  
{LITERAL; 'true'}  
{END_OBJECT; '}'}
```

Рисунок 1. Результат программы

Вывод

В данной практической работе были получены навыки в работе с JSON файле, получены навыки в обработке лексем.