|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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

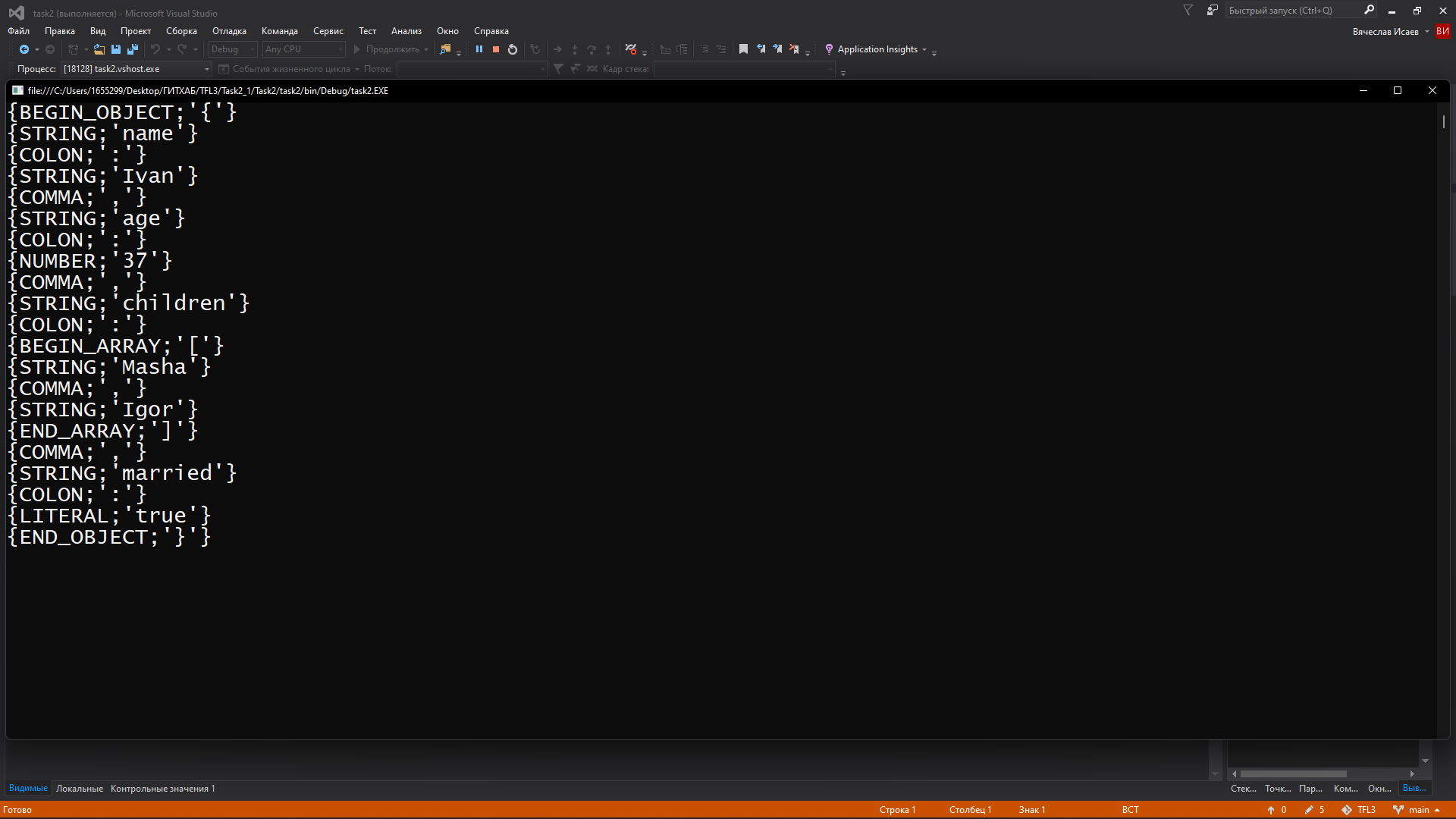


Рисунок 1. Результат программы

# Вывод

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