|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №4** «Лексический анализатор на базе конечного автомата» |
|  |
| **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

Задание: написать на любом языке программирования (или доработать листинг 1) лексический анализатор на базе конечного автомата входного языка, описанного диаграммой состояний рис. 1.

# Реализация решения задания на языке C#

|  |
| --- |
| public static bool IsLetter(char a)  {  char[] letters = new char[52] { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };  foreach(var s in letters)  {  if(a==s)  {  return true;  }  }  return false;  }  public static bool IsNumeric(char a)  {  if(a=='0'||a=='1'||a=='2'||a=='3'||a=='4'||a=='5'||a=='6'||a=='7'||a=='8'||a=='9')  {  return true;  }  else  {  return false;  }  }  public static bool IsKeyWord(string word, string[] words)  {  foreach (var s in words)  {  if(word==s)  {  return true;  }  }  return false;  }  public static bool IsNumber(string word)  {  char[] symb = word.ToCharArray();  Regex ful\_sign = new Regex(@"^[/+/-]{1}[0-9]\*$");  Regex broke\_sign = new Regex(@"^[\+\-]{1}[0-9]\*\.[0-9]\*$");  Regex broke\_without\_sign = new Regex(@"^[0-9]\*\.[0-9]\*$");  Regex ful\_without\_sign = new Regex(@"^[0-9]\*$");  MatchCollection matches1 = ful\_sign.Matches(word);  MatchCollection matches2 = broke\_sign.Matches(word);  MatchCollection matches3 = broke\_without\_sign.Matches(word);  MatchCollection matches4 = ful\_without\_sign.Matches(word);  if(matches1.Count>0||matches2.Count>0||matches3.Count>0||matches4.Count>0)  {  return true;  }  else  {  return false;  }  }  static void Main(string[] args)  {  Console.Clear();  string word="";  int cur\_pos = 0;  string buf\_word = "";  string buf\_num = "";  string[,] lexems = new string[100,3];  string[] key\_word = new string[100];  string[] world = new string[100];  int cur\_lexem = 0;  string CurCond = "H";  Console.WriteLine("Введите номер команды\n1.Ввод строки вручную\n2.Ввод строки из файла\n3.Изменить файл с кодом\n4.Изменить файл с кодовыми словами\n5.Выход");  int g = Convert.ToInt32(Console.ReadLine());  switch(g)  {  case 1:  Console.WriteLine("Введите строку");  word = Convert.ToString(Console.ReadLine());  break;  case 2:  word = File.ReadAllText("code.txt");  Console.WriteLine("Считывание строки из файла.\nДля продолжения нажмите Enter");  Console.ReadLine();  break;  case 3:  Process.Start("notepad.exe","code.txt");  Main(world);  break;  case 4:  Process.Start("notepad.exe","KeyWords.txt");  Main(world);  break;  case 5:  Console.WriteLine("Вы уверены, что хотите выйти? Введите y/n");  string k = Console.ReadLine().ToLower();  char j = Convert.ToChar(k);  if(j=='y')  {  Environment.Exit(0);  }  else if(j=='n')  {  Main(world);  }  else  {  Console.WriteLine("Ошибка во вводе");  Console.ReadLine();  Main(world);  }    break;  default:  Console.WriteLine("Введен неверный номер команды.\nДля продолжение нажмите Enter");  Console.ReadLine();  Environment.Exit(2);  break;  }  char[] word\_char = new char[word.Length];  word\_char = word.ToCharArray();  key\_word = File.ReadAllLines("KeyWords.txt");  Console.WriteLine("Текущий символ\tТекущее состояние");  while (cur\_pos<word\_char.Length)  {  switch (CurCond)  {  case "H":  Console.Write(word\_char[cur\_pos]+"\t\t");  if (word\_char[cur\_pos] == '\t' || word\_char[cur\_pos] == '\n' || word\_char[cur\_pos] == ' '||word\_char[cur\_pos]=='\r')  {  cur\_pos++;  }  else if (word\_char[cur\_pos] == '+' && word\_char[cur\_pos + 1] == '+' || word\_char[cur\_pos] == '-' && word\_char[cur\_pos + 1] == '-' || word\_char[cur\_pos] == '=' && word\_char[cur\_pos + 1] == '=' || word\_char[cur\_pos] == '!' && word\_char[cur\_pos + 1] == '=' || word\_char[cur\_pos] == '>' && word\_char[cur\_pos + 1] == '=' || word\_char[cur\_pos] == '<' && word\_char[cur\_pos + 1] == '=')  {  CurCond = "DLM";  }  else if(word\_char[cur\_pos]==':')  {  CurCond = "ASGN";  cur\_pos++;  }  else if(word\_char[cur\_pos] == '\_' ||IsLetter(word\_char[cur\_pos]))  {  buf\_word = "";  CurCond = "ID";  }  else if (word\_char[cur\_pos] == '-' || word\_char[cur\_pos] == '+' || word\_char[cur\_pos] == '.' || IsNumeric(word\_char[cur\_pos]))  {  buf\_num = "";  CurCond = "NM";  }  else  {  CurCond = "DLM";  }  Console.WriteLine(CurCond);  break;  case "ASGN":  Console.Write(word\_char[cur\_pos] + "\t\t");  if (word\_char[cur\_pos]=='=')  {  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem+1);  lexems[cur\_lexem, 1] = "=";  lexems[cur\_lexem, 2] = "оператор присванивания";  cur\_lexem++;  cur\_pos++;  CurCond = "H";  }  else  {  cur\_pos++;  CurCond = "ERR";  }  Console.WriteLine(CurCond);  break;  case "ID":  Console.Write(word\_char[cur\_pos] + "\t\t");  if(word\_char[cur\_pos]=='\_'||IsNumeric(word\_char[cur\_pos])||IsLetter(word\_char[cur\_pos]))  {  buf\_word += word\_char[cur\_pos];  if (IsKeyWord(buf\_word, key\_word))  {  cur\_pos++;  CurCond = "ID";  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem + 1);  lexems[cur\_lexem, 1] = buf\_word;  lexems[cur\_lexem, 2] = "Ключевое слово";  }  else  {  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem + 1);  lexems[cur\_lexem, 1] = buf\_word;  lexems[cur\_lexem, 2] = "Идентификатор";  cur\_pos++;  CurCond = "ID";  }  }  else  {  cur\_lexem++;  CurCond = "H";  }  Console.WriteLine(CurCond);  break;  case "NM":  Console.Write(word\_char[cur\_pos] + "\t\t");  if (word\_char[cur\_pos] == '.' || IsNumeric(word\_char[cur\_pos]) || word\_char[cur\_pos] == '-' || word\_char[cur\_pos] == '+')  {  buf\_num += word\_char[cur\_pos];  if (IsNumber(buf\_num))  {  cur\_pos++;  CurCond = "NM";  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem + 1);  lexems[cur\_lexem, 1] = buf\_num;  lexems[cur\_lexem, 2] = "Число";  }  else  {  cur\_pos++;  CurCond = "ERR";  }  }  else  {  cur\_lexem++;  CurCond = "H";  }  Console.WriteLine(CurCond);  break;  case "DLM":  Console.Write(word\_char[cur\_pos] + "\t\t");  if(word\_char[cur\_pos]=='('|| word\_char[cur\_pos] == ')' || word\_char[cur\_pos] == ';'||word\_char[cur\_pos]=='{'||word\_char[cur\_pos]=='}')  {  CurCond = "H";  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem + 1);  lexems[cur\_lexem, 1] = Convert.ToString(word\_char[cur\_pos]);  lexems[cur\_lexem, 2] = "Разделитель";  cur\_pos++;  cur\_lexem++;  }  else if (word\_char[cur\_pos] == '+' && word\_char[cur\_pos + 1] == '+' || word\_char[cur\_pos] == '-' && word\_char[cur\_pos + 1] == '-' || word\_char[cur\_pos] == '=' && word\_char[cur\_pos + 1] == '=' || word\_char[cur\_pos] == '!' && word\_char[cur\_pos + 1] == '=' || word\_char[cur\_pos] == '>' && word\_char[cur\_pos + 1] == '=' || word\_char[cur\_pos] == '<' && word\_char[cur\_pos + 1] == '=')  {  CurCond = "H";  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem + 1);  lexems[cur\_lexem, 1] = Convert.ToString(word\_char[cur\_pos]) + Convert.ToString(word\_char[cur\_pos + 1]);  lexems[cur\_lexem, 2] = "Операция";  cur\_pos++;  cur\_pos++;  cur\_lexem++;  }  else if ( word\_char[cur\_pos] == '<' || word\_char[cur\_pos] == '>')  {  CurCond = "H";  lexems[cur\_lexem, 0] = Convert.ToString(cur\_lexem + 1);  lexems[cur\_lexem, 1] = Convert.ToString(word\_char[cur\_pos]);  lexems[cur\_lexem, 2] = "Операция";  cur\_pos++;  cur\_lexem++;  }  else  {  CurCond = "ERR";  }  Console.WriteLine(CurCond);  break;  case "ERR":  Console.WriteLine("Неизвестная операция");  CurCond = "H";  cur\_pos++;  break;  }  }  Console.WriteLine("Таблица лексем");  for(int i=0;i<cur\_lexem;i++)  {  Console.WriteLine(lexems[i, 0] + "\t" + lexems[i, 1] + "\t" + lexems[i, 2]);  }  Console.ReadLine();  Main(world);  } |

# Тестирование

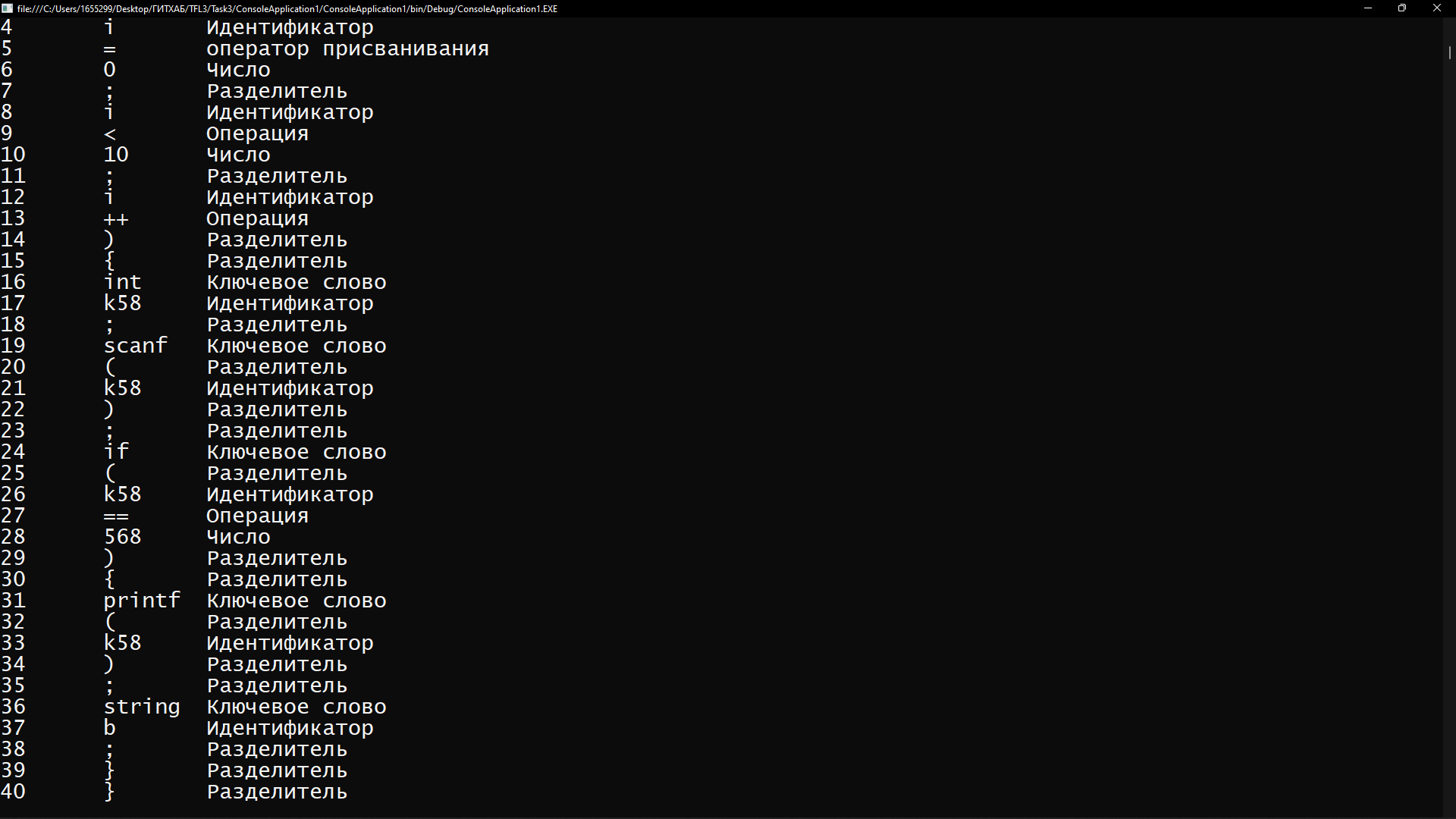


Рисунок 1. Результат программы

# Вывод

В данной практической работе были получены навыки в работе с конечными автоматами и навыки в распознавании лексем.