**Assignment - 3**

**COMP 4040**

**Spring 2020**

**Instructor: Fatih Şen**

1. [100 pts] Using C#, implement a console application of the source code of a lexical analyzer (front.c) on page 166 in the textbook. Your console application should receive an input (for example sum + 56/total) and give the similar output like on page 171.

Note: The entire project is included in this ZIP file

Output:

A screenshot of a cell phone

Description automatically generated

Input File (front.in):

(sum + 47) / total

Source Code (program.cs):

using System;

using System.IO;

namespace Assignment\_3\_COMP\_4040

{

class Program

{

public const int LETTER = 0;

public const int DIGIT = 1;

public const int UNKNOWN = 99;

public const int INT\_LIT = 10;

public const int IDENT = 11;

public const int ASSIGN\_OP = 20;

public const int ADD\_OP = 21;

public const int SUB\_OP = 22;

public const int MULT\_OP = 23;

public const int DIV\_OP = 24;

public const int LEFT\_PAREN = 25;

public const int RIGHT\_PAREN = 26;

public const int EOF = -1;

static int charClass;

static char[] lexeme = new char[100];

static char nextChar;

static int lexLen;

static int nextToken;

static private StreamReader fileStream;

static string workingDirectory = Environment.CurrentDirectory;

static string projectDirectory = Directory.GetParent(workingDirectory).Parent.Parent.FullName;

public static void Main(string[] args)

{

fileStream = new StreamReader(projectDirectory + "/front.in", System.Text.Encoding.UTF8);

getChar(); do

{

lex();

} while (nextToken != EOF);

}

static int lookup(char ch)

{

switch (ch)

{

case '(':

addChar();

nextToken = LEFT\_PAREN; break;

case ')':

addChar();

nextToken = RIGHT\_PAREN; break;

case '+':

addChar(); nextToken = ADD\_OP; break;

case '-':

addChar(); nextToken = SUB\_OP; break;

case '\*':

addChar();

nextToken = MULT\_OP; break;

case '/':

addChar(); nextToken = DIV\_OP; break;

default: addChar(); nextToken = EOF; break;

}

return nextToken;

}

static void addChar()

{

if (lexLen <= 98)

{

lexeme[lexLen++] = nextChar;

}

else

Console.WriteLine("Error - lexeme is too long \n");

}

static void getChar()

{

nextChar = (char)fileStream.Read();

if (nextChar != EOF)

{

if (Char.IsLetter(nextChar))

charClass = LETTER;

else if (Char.IsDigit(nextChar))

charClass = DIGIT;

else charClass = UNKNOWN;

}

else charClass = EOF;

}

static void getNonBlank()

{

while (Char.IsWhiteSpace(nextChar)) getChar();

}

static int lex()

{

lexLen = 0; getNonBlank();

switch (charClass)

{

case LETTER:

addChar();

getChar();

while (charClass == LETTER || charClass == DIGIT)

{

addChar();

getChar();

}

nextToken = IDENT; break;

case DIGIT:

addChar();

getChar();

while (charClass == DIGIT)

{

addChar();

getChar();

}

nextToken = INT\_LIT; break;

case UNKNOWN:

lookup(nextChar); getChar();

break;

case EOF:

nextToken = EOF; lexeme[0] = 'E'; lexeme[1] = 'O'; lexeme[2] = 'F';

break;

}

Console.WriteLine("Next token is: {0}, Next lexeme is {1}", nextToken, new string(lexeme).Replace("", "EOF"));

lexeme = new char[100];

return nextToken;

}

}

}