Lab 4:

Code:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CCLab04

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

// Button click event for lexical analysis

private void btnAnalyze\_Click(object sender, EventArgs e)

{

string input = txtInput.Text;

List<Token> tokens = LexicalAnalyze(input);

dataGridView1.Rows.Clear();

// Add tokens to the DataGridView

foreach (var token in tokens)

{

dataGridView1.Rows.Add(token.Type, token.Value);

}

}

// Lexical analyzer function using two buffers

private List<Token> LexicalAnalyze(string input)

{

List<Token> tokens = new List<Token>();

int currentPos = 0;

int bufferPos = 0;

StringBuilder buffer = new StringBuilder();

while (currentPos < input.Length)

{

char currentChar = input[currentPos];

buffer.Append(currentChar);

// Skip whitespace

if (char.IsWhiteSpace(currentChar))

{

currentPos++;

buffer.Clear();

continue;

}

// Handle keywords and identifiers

if (char.IsLetter(currentChar))

{

while (currentPos + 1 < input.Length && char.IsLetterOrDigit(input[currentPos + 1]))

{

currentPos++;

buffer.Append(input[currentPos]);

}

if (IsKeyword(buffer.ToString()))

{

tokens.Add(new Token("Keyword", buffer.ToString()));

}

else

{

tokens.Add(new Token("Identifier", buffer.ToString()));

}

}

// Handle numbers

else if (char.IsDigit(currentChar))

{

while (currentPos + 1 < input.Length && char.IsDigit(input[currentPos + 1]))

{

currentPos++;

buffer.Append(input[currentPos]);

}

tokens.Add(new Token("Number", buffer.ToString()));

}

// Handle operators or punctuation

else if (IsOperator(currentChar))

{

tokens.Add(new Token("Operator", currentChar.ToString()));

}

else

{

tokens.Add(new Token("Unknown", currentChar.ToString()));

}

buffer.Clear();

currentPos++;

}

return tokens;

}

// Check if a word is a keyword

private bool IsKeyword(string word)

{

string[] keywords = { "if", "else", "for", "while", "int", "float", "return" };

return keywords.Contains(word);

}

// Check if a character is an operator

private bool IsOperator(char c)

{

return "+-\*/=<>!".Contains(c);

}

}

// Token class to represent each token type

public class Token

{

public string Type { get; set; }

public string Value { get; set; }

public Token(string type, string value)

{

Type = type;

Value = value;

}

}

}

Output:



