```
using System;
using System.Collections.Generic;

class LexicalAnalyzer
{
   static readonly HashSet<string> Keywords = new HashSet<string> { "if", "else", "while", "return",
"int", "float" };
   static readonly HashSet<char> Operators = new HashSet<char> { '+', '-', '*', '/', '=', '<', '>', '!' };
   static readonly HashSet<char> Separators = new HashSet<char> { ';', ',', '(', ')', '{', '}' };

   private const int BufferSize = 16;
   private char[] buffer1 = new char[BufferSize];
   private char[] buffer2 = new char[BufferSize];
   private bool useFirstBuffer = true;
   private int bufferIndex = 0;
   private int bufferLength = 0;
   private string inputCode;

   public LexicalAnalyzer(string code)
   {
      inputCode = code;
      FillBuffer();
   }

   private void FillBuffer()
   {
      int length = Math.Min(BufferSize, inputCode.Length);
      for (int i = 0; i < length; i++)
         buffer1[i] = inputCode[i];

      bufferLength = length;
      bufferIndex = 0;
   }

   private char? GetNextChar()
   {
      if (bufferIndex >= bufferLength)
         return null;

      char ch = useFirstBuffer ? buffer1[bufferIndex] : buffer2[bufferIndex];
      bufferIndex++;

      return ch;
   }

   public List<(string, string)> Tokenize()
   {
      List<(string, string)> tokens = new List<(string, string)>();
      string currentToken = "";
      char? ch = GetNextChar();

      while (ch != null)
      {
         if (char.IsWhiteSpace(ch.Value))
         {
            if (currentToken.Length > 0)
            {
```

```
                tokens.Add(ClassifyToken(currentToken));
                currentToken = "";
            }
        }
        else if (Operators.Contains(ch.Value) || Separators.Contains(ch.Value))
        {
            if (currentToken.Length > 0)
            {
                tokens.Add(ClassifyToken(currentToken));
                currentToken = "";
            }
            tokens.Add((ch.Value.ToString(), "SYMBOL"));
        }
        else
        {
            currentToken += ch.Value;
        }
        ch = GetNextChar();
    }

    if (currentToken.Length > 0)
        tokens.Add(ClassifyToken(currentToken));

    return tokens;
  }

  private (string, string) ClassifyToken(string token)
  {
    if (Keywords.Contains(token)) return (token, "KEYWORD");
    if (char.IsDigit(token[0])) return (token, "NUMBER");
    return (token, "IDENTIFIER");
  }
}

class Program
{
  static void Main()
  {
    Console.WriteLine("Enter your code:");
    string inputCode = Console.ReadLine();

    LexicalAnalyzer lexer = new LexicalAnalyzer(inputCode);
    var tokens = lexer.Tokenize();

    Console.WriteLine("\nTokenized Output:");
    foreach (var (token, type) in tokens)
        Console.WriteLine($"{token}: {type}");
  }
```

Output:

```
Enter your code:
int x = 10;if (x > 5) {    return x;}

Tokenized Output:
int: KEYWORD
x: IDENTIFIER
=: SYMBOL
10: NUMBER
;: SYMBOL
if: KEYWORD
(: SYMBOL
x: IDENTIFIER
```