Приложение В

Листинг программы

Parcer.cs

```
using System.Collections.Generic;
namespace CompilCourseWork.model;
public class Parser
{
    private int _pos;
    private readonly List<string> _outputLog = new();

    public List<string> Parse(string input)
    {
        _pos = 0;
        _outputLog.Clear();

        if (ParseKeyword(input, "const"))
        {
            if (_pos < input.Length && input[_pos] != ' ')
            {
                _outputLog.Add($"Ошибка: ожидался пробел после
'const'");
            }
        }
        SkipWhiteSpaces(input);

        if (ParseKeyword(input, "val"))
        {
            if (_pos < input.Length && input[_pos] != ' ')
            {
                _outputLog.Add($"Ошибка: ожидался пробел после
'val'");
            }
        }
```

```
        SkipWhiteSpaces(input);

        ParseId(input);
        SkipWhiteSpaces(input);
        ParseColon(input);
        SkipWhiteSpaces(input);
        ParseKeyword(input, "Double");
        SkipWhiteSpaces(input);
        ParseEqual(input);
        SkipWhiteSpaces(input);
        ParseNum(input);
        SkipWhiteSpaces(input);
        ParseEnd(input);

        return _outputLog;
    }

    private bool ParseKeyword(string input, string expectedKeyword)
    {
        bool success = true;
        List<char> trashChars = new List<char>();
        int initialPos = _pos;

        while (_pos < input.Length && input[_pos] !=
expectedKeyword[0])
        {
            if (input[_pos] == 'v' || input[_pos] == 'D' ||
input[_pos] == '=')
            {
                _outputLog.Add($"Ожидалось ключевое слово
{expectedKeyword}");
                return false;
            }
```

```
            if (input[_pos] == ':')
            {
                _outputLog.Add($"Ожидалась переменная");
                return false;
            }
            if (input[_pos] == 'D')
            {
                _outputLog.Add($"Ожидалась ':'");
                return false;
            }
            trashChars.Add(input[_pos]);
            _pos++;
        }

        if (trashChars.Count != 0)
        {
            _outputLog.Add($"Ошибка: Неизвестная конструкция в
позиции {initialPos}: отброшенный фрагмент: {new
string(trashChars.ToArray())}");
        }

        int wordStartPos = _pos;
        int expectedIndex = 0;
        bool alignmentDone = false;

        while (_pos < input.Length && expectedIndex <
expectedKeyword.Length)
        {
            char inputChar = input[_pos];
            char expectedChar = expectedKeyword[expectedIndex];

            if (inputChar == expectedChar)
            {
                expectedIndex++;
```

```
                _pos++;
            }
            else
            {
                int lookaheadIndex = -1;
                if (expectedIndex + 1 < expectedKeyword.Length)
                {
                    lookaheadIndex =
expectedKeyword.IndexOf(inputChar, expectedIndex + 1);
                }
                if (lookaheadIndex != -1)
                {
                    _outputLog.Add($"Ожидалось ключевое
слово:'{expectedKeyword}'");
                    expectedIndex = lookaheadIndex;
                    alignmentDone = true;
                    if (inputChar == expectedKeyword[expectedIndex])
                    {
                        expectedIndex++;
                        _pos++;
                    }
                    else
                    {
                        success = false;
                        _pos++;
                    }
                }
                else
                {
                    _outputLog.Add($"Ошибка: Неожиданный символ
'{inputChar}' на позиции {_pos}");
                    success = false;
                    _pos++;
                }
```

```
            }
        }
        if (expectedIndex < expectedKeyword.Length)
        {
            if (success && !alignmentDone)
            {
                _outputLog.Add($"Ошибка: Ожидалось
'{expectedKeyword}', но строка закончилась (проверка началась с
{wordStartPos})");
            }
            else if (!success)
            {
                _outputLog.Add($"Ошибка: Ключевое слово
'{expectedKeyword}' не распознано полностью из-за ошибок (проверка
началась с {wordStartPos})");
            }
            else if (alignmentDone)
            {
                _outputLog.Add($"Ошибка: Ключевое слово
'{expectedKeyword}' не завершено после выравнивания (проверка
началась с {wordStartPos})");
            }
            success = false;
        }
        return success;
    }
    private bool ParseId(string input)
    {
        bool success = true;
        List<char> trashChars = new List<char>();
        while (_pos < input.Length && !(char.IsLetter(input[_pos]) &&
input[_pos] != '_'))
        {
            if (input[_pos] == ':')
```

```
            {
                _outputLog.Add($"Ожидалась переменная");
                return false;
            }
            trashChars.Add(input[_pos]);
            _pos++;
        }
        if (trashChars.Count != 0)
        {
            _outputLog.Add($"Ошибка: Неизвестная конструкция в
позиции {_pos - trashChars.Count}: отброшенный фрагмент: {new
string(trashChars.ToArray())}");
        }
        if (_pos < input.Length && (char.IsLetter(input[_pos]) ||
input[_pos] == '_'))
        {
            while (_pos < input.Length &&
(char.IsLetterOrDigit(input[_pos]) || input[_pos] == '_' ||
char.IsDigit(input[_pos])))
            {
                _pos++;
            }
        }
        else
        {
            _outputLog.Add($"Ошибка: ожидался идентификатор
(начинается с буквы или '_') на позиции {_pos}");
            success = false;
        }
        return success;
    }
    private bool ParseColon(string input)
    {
        bool success = true;
```

```
        if (_pos < input.Length && input[_pos] == ':')
        {
            _pos++;
        }
        else
        {
            _outputLog.Add($"Ошибка: ожидалось двоеточие ':' на
позиции {_pos}");
            success = false;
        }
        return success;
    }
    private bool ParseEqual(string _input)
    {
        bool success = true;
        if (_pos < _input.Length && _input[_pos] == '=')
        {
            _pos++;
        }
        else
        {
            _outputLog.Add($"Ошибка: ожидалось знак равенства '=' на
позиции {_pos}");
            success = false;
        }

        return success;
    }
    private bool ParseNum(string _input)
    {
        bool success = true;
        List<char> trashChars = new List<char>();
        while (_pos < _input.Length && !char.IsDigit(_input[_pos]))
        {
```

```
            trashChars.Add(_input[_pos]);
            _pos++;
        }
        if (trashChars.Count != 0)
        {
            _outputLog.Add($"Ошибка: Неизвестная конструкция в
позиции {_pos - trashChars.Count}: отброшенный фрагмент: {new
string(trashChars.ToArray())}");
        }
        int dotCount = 0;
        if (_pos < _input.Length && char.IsDigit(_input[_pos]))
        {
            while (_pos < _input.Length)
            {
                if (char.IsDigit(_input[_pos]))
                {
                    _pos++;
                }
                else if (_input[_pos] == '.')
                {
                    dotCount++;
                    if (dotCount > 1)
                    {
                        _outputLog.Add($"Ошибка: лишняя точка в
позиции {_pos}");
                        success = false;
                    }
                    _pos++;
                }
                else if (_input[_pos] == ';')
                {
                    break;
                }
                else
```

```
                {
                    _outputLog.Add($"Ошибка: неожиданный символ
'{_input[_pos]}' на позиции {_pos}");
                    _pos++;
                    success = false;
                }
            }
        }
        else
        {
            _outputLog.Add($"Ошибка: ожидалось число на позиции
{_pos}");
            success = false;
        }

        return success;
    }
    private bool ParseEnd(string input)
    {
        bool success = true;

        if (_pos < input.Length && input[_pos] == ';')
        {
            _pos++;
        }
        else
        {
            _outputLog.Add($"Ошибка: ожидался завершающий символ ';'
на позиции {_pos}");
            success = false;
        }
        return success;
    }
    private void SkipWhiteSpaces(string input)
```