МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное образовательное бюджетное учреждение высшего образования
«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»
Кафедра САПР ВС

К защите
Руководитель проекта

_____
дата, подпись

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине
«Лингвистическое и программное обеспечение САПР»
по теме
«Синтаксический анализ программ»

Выполнил студент группы 846 _____
Шмаков А.Ю. дата сдачи на проверку, подпись

Руководитель проекта _____ _____
Профессор Скворцов С.В. оценка дата защиты, подпись

Рязань 2021

Содержание

Задание на курсовой проект

Введение

Особое место в САПР занимает программное обеспечение (ПО), поскольку именно в машинных программах реализуются методы автоматизированного проектирования. На разработку ПО затрачивается до 90% средств, выделяемых на создание САПР. Каждая вычислительная машина имеет свой язык программирования – язык команд (или машинный язык) – и может исполнять программы, записанные только на этом языке. С помощью машинного языка можно описать любой алгоритм, но при этом затраты на программирование будут чрезвычайно велики. Это обусловлено тем, что машинный язык позволяет описывать и обрабатывать только примитивные структуры данных – бит, байт, слово, а программирование в машинных кодах требует чрезмерной детализации программы и доступно лишь программистам, хорошо знающим устройство функционирования ЭВМ. Поэтому машинные языки при разработке САПР обычно не применяются.

В соответствии с ГОСТ 23501.0-79, совокупность языков, терминов и определений, необходимых для выполнения автоматизированного проектирования, называется лингвистическим обеспечением (ЛО) САПР. Все языки, входящие в состав ЛО САПР, делятся на языки программирования и языки проектирования. При разработке ПО САПР наиболее часто используются языки программирования высокого уровня (алгоритмические языки) и машинно-ориентированные языки программирования (языки ассемблера). Языки проектирования являются средством взаимодействия пользователя САПР с ЭВМ и служат для описания информации об объектах и задачах проектирования.

Исполнение на ЭВМ заданий, представленных на каком-либо языке САПР, отличном от машинного, требует интерпретации или предварительной трансляции исходной программы с помощью системной программы на машинном языке, называемой языковым процессором. Различают два основных типа языковых процессоров, используемых в САПР: интерпретаторы и трансляторы.

Интерпретатор – это системная программа, которая считывает исходную программу пользователя или разработчика САПР, анализирует ее и по мере распознавания конструкций исходного языка реализует (исполняет) их, выдавая получаемые результаты. При решении задачи в оперативной памяти ЭВМ присутствуют прикладная программа на исходном языке и интерпретатор.

Транслятор – это системная программа, которая преобразует исходную программу пользователя или разработчика САПР в некоторую функционально эквивалентную форму, называемую объектной программой. Объектная программа формируется на объектном языке. Если объектный язык - машинный или машинно-ориентированный, то транслятор называется компилятором, а процесс трансляции - компиляцией. Если исходный язык - язык ассемблера, то транслятор на машинный язык (компилятор) называется ассемблером. При использовании компиляторов решение задачи осуществляется в два этапа. Сначала в оперативной памяти ЭВМ размещаются прикладная программа на исходном языке и компилятор, результатом работы которого будет рабочая программа. Затем скомпилированная рабочая программа исполняется аппаратными средствами ЭВМ.

Таким образом, функции ЛО САПР выполняет совокупность языков и языковых процессоров, поскольку любой формальный язык не может использоваться в автоматизированном проектировании без соответствующего транслятора или интерпретатора. Включению любого языка в состав ЛО САПР должна предшествовать разработка языкового процессора. При этом трудоемкость его создания, особенно при компиляции, существенно превосходит трудоемкость разработки языка САПР.

1 Постановка задачи

Дан исходный текст программы, написанный на языке программирования Паскаль, который расположен в текстовом файле.

Необходимо написать программный продукт, который проанализирует правильность использования парных элементов программы: begin-end; case-end; repeat-until; record-end. А также реализовать диагностику возможных ошибок с указанием места их расположения, результаты необходимо поместить в текстовый файл.

2 Алгоритм решения задачи

В курсовом проекте использует анализ скобочных структур нескольких типов, поэтому для согласования типов парных элементов программы (скобок) используется стек, который в исходном состоянии является пустым.

Если в потоке лексем встречается левая скобка, то она записывается в стек. Когда считывается правая скобка, из стека выбирается левая скобка (верхний элемент стека) и проверяется совпадение их типов.

Скобочная структура будет синтаксически корректной при выполнении трех условий:

1. При чтении правой скобки стек не должен быть пустым
2. При чтении правой скобки ее тип должен совпадать с типом левой скобки, расположенной на вершине стека.
3. После просмотра всей скобочной структуры стек должен быть пустым.

Очевидно, что нарушение одного из трех пунктов может сопровождаться диагностическим сообщением. Для первого условия – «Отсутствует левая скобка определенного типа», который должен совпадать с типом считанной правой скобки, для второго условия – «Не совпадают типы скобок», для третьего – «Отсутствует правая скобка определенного типа».

Схема алгоритма представлена на рисунке 1.

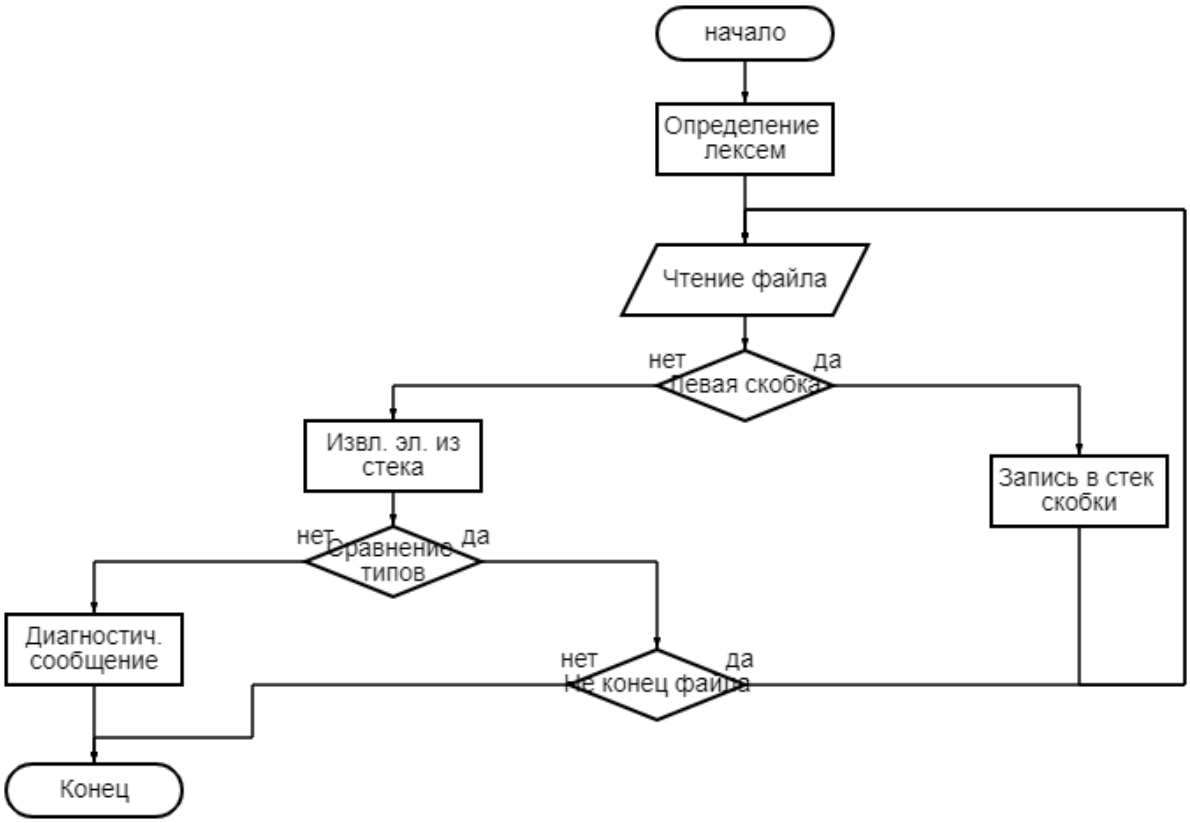

Рисунок 1 – Схема алгоритма

3 Программная реализация алгоритма

Для программной реализации алгоритма был выбран язык высокого уровня C#. А также использована библиотека Windows Forms для создания оконного приложения Windows.

Функция OpenFile() реализует открытие и загрузку исходного текста программы на языке паскаль для дальнейшего анализа.

Функция Algorithm() выполняет основную задачу анализа скобочных структур.

Функция SaveWarnings() сохраняет в текстовый файл результат анализа скобочных структур анализируемого текста программы.

Функция MenuAboutClick() вызывает всплывающее окно, в котором содержится информация о курсовом проекте.

Функция MenuClose_Click() вызывает системную функцию закрытия приложения.

Описание всех переменных алгоритма находиться в приложении 1 и реализовано с помощью комментариев к программе.

4 Тестирование программы

Для тестирования программы, используется простой пример программы, которая подсчитывает количество слов в исходном тексте.

Тест 1.

Исходный текст программы для анализа представлен на рисунке 2:

```
1  begin
2  const Alpha : set of char=['A'..'Z','a'..'z'];
3  var s:string;
4      i:integer;
5      wc:integer;
6  begin
7       writeln('vvedite tekst'); readln(s);
8       i:=1; wc:=0;
9       Repeat
10         while NOT(s[i] in Alpha) and (i<=length(s)) do inc(i);
11         if (i<=length(s)) then inc(wc);
12         while (s[i] in Alpha) and (i<=length(s)) do inc(i);
13      Until (i>length(s));
14      writeln('kol-vo bykv v tekste = ',wc);
15 end.
16
```

Рисунок 2 – Программа для анализа.

Ожидаемый результат программы: Ошибок не найдено

Фактический результат представлен на рисунке 3.

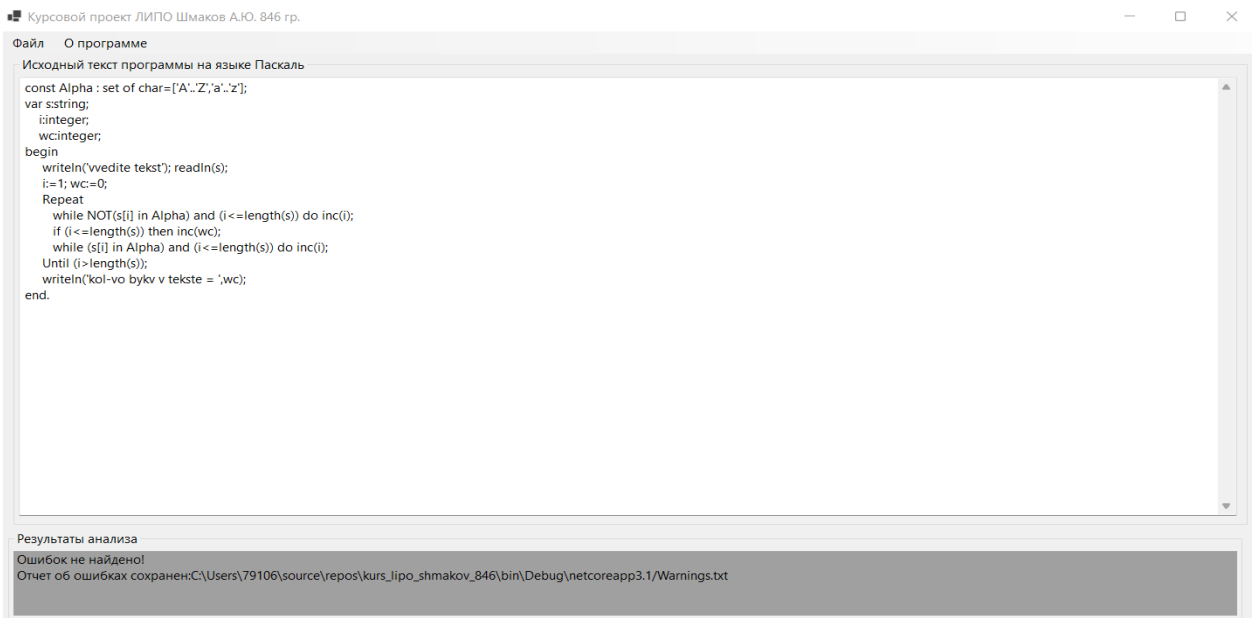

Рисунок 3 – фактический результат тестирования 1.

Тест 2.

Изменим текст программы, допустив в нем ошибку, не закроем цикл repeat.

Ожидаемый результат: Диагностическое сообщение “Не совпадают типы скобок”.

Фактический результат представлен на рисунке 4.

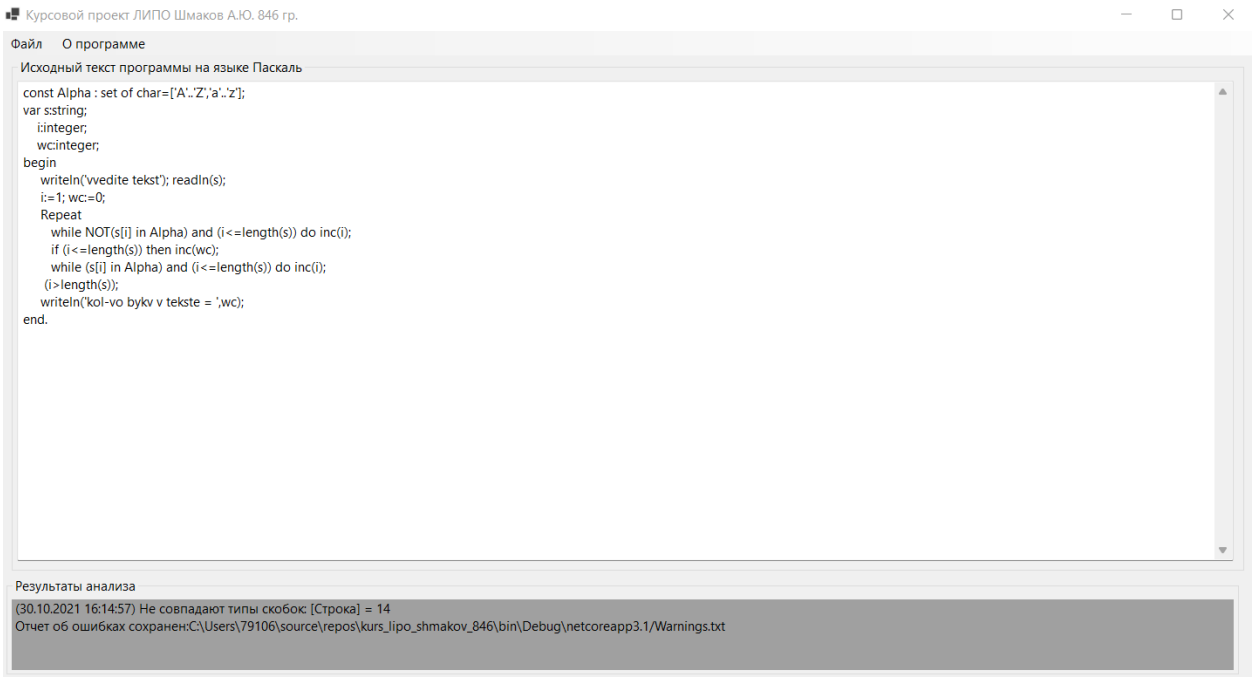

Рисунок 4 – фактический результат теста 2.

Тест 3.

Изменим текст программы, допустив в нем ошибку, поставим в конце лишнюю правую скобку end.

Ожидаемый результат: Диагностическое сообщение “Отсутствует левая скобка”.

Фактический результат представлен на рисунке 5.

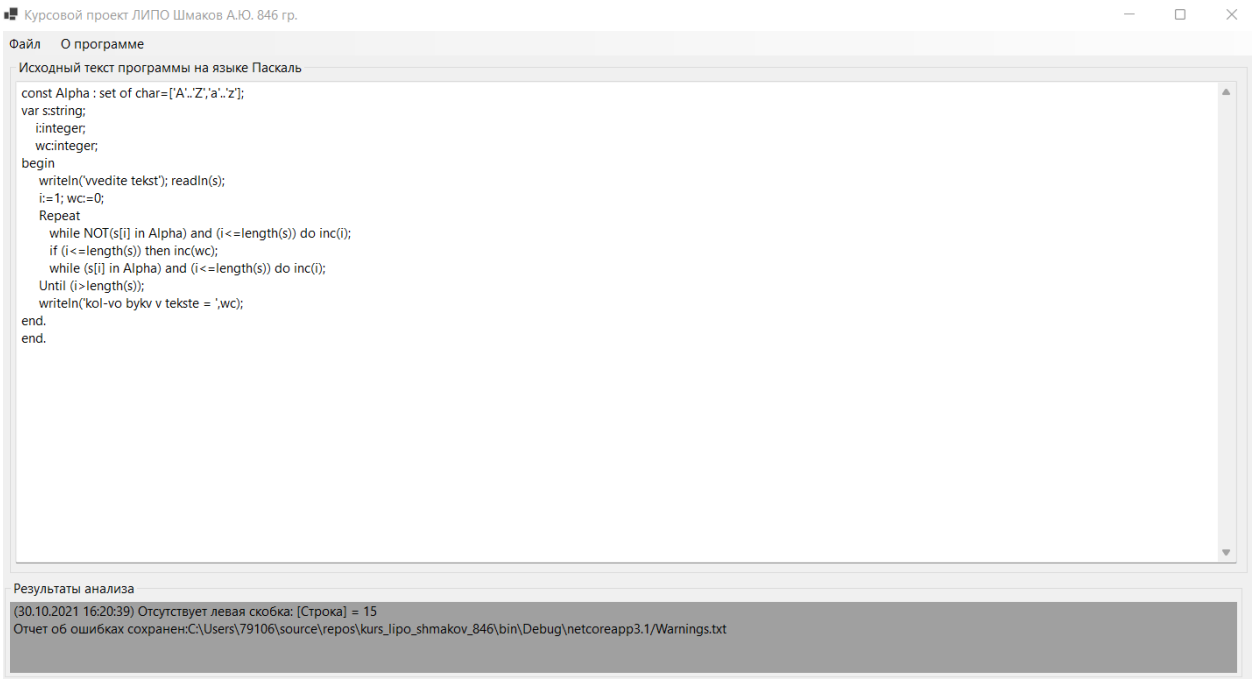

Рисунок 5 – фактический результат теста 3.

Тест 4.

Изменим текст программы, допустив в нем ошибку, поставим в начале лишнюю левую скобку begin.

Ожидаемый результат: Диагностическое сообщение “Отсутствует правая скобка”.

Фактический результат представлен на рисунке 6.

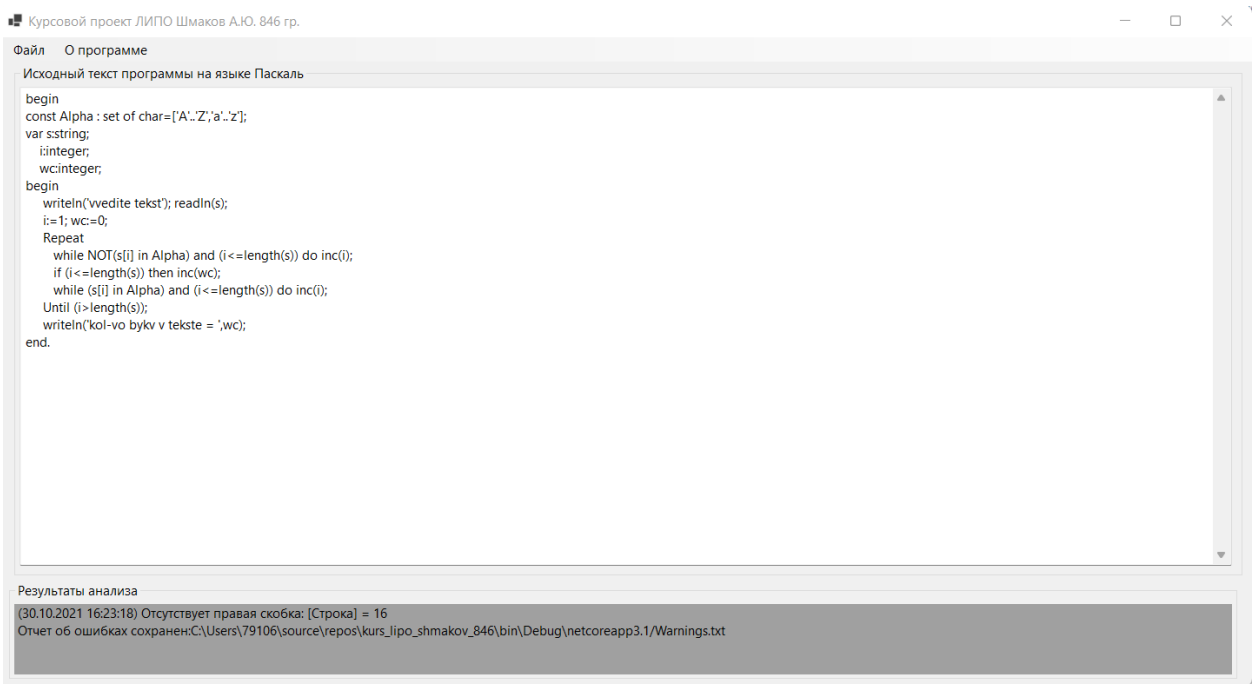

Рисунок 6 – фактический результат теста 4.

5 Программная документация

5.1 Назначение и условия применения программы

Программа предназначена для анализа парных элементов исходных текстов программ на языке программирования паскаль. Для запуска программы необходима операционная система Windows от 7 версии и выше.

5.2 Характеристики программы

Программа написана на языке высокого уровня C#, что обеспечивает высокую скорость работы программы, а также ее отказоустойчивость.

5.3 Обращение к программе

Для запуска программы необходимо открыть .exe файл и в меню открыть выбрать требуемый файл исходного теста программы с расширением .pas.

5.4 Входные и выходные данные

Входными данными является исходный текст программы, находящийся в файле с расширением .pas. Выходные данные – текстовый файл, в котором содержится информация о результатах анализа.

5.4 Сообщения

Все сообщения находятся в текстовом файле, получаемом при анализе исходного теста программы, в случае если обнаружена ошибка в тексте программы, ее необходимо устранить и провести анализ повторно.

Заключение.

В ходе выполнения курсового проекта была разработана программная реализация алгоритма, который позволяет анализировать правильность использования парных элементов в тексте программы, а также была реализована возможность получения диагностического сообщения и места нахождения ошибки.

Библиографический список

1. Лингвистическое и программное обеспечение САПР: Методические указания к курсовой работе / Рязан. гос. радиотехн. акад.; Сост.: С.В. Скворцов, В.И. Хрюкин.

2. Гудман С., Хидетниеми С. Введение в анализ и разработку алгоритмов. М.: Мир 1980./

3. Методы построения языковых процессоров САПР: Учеб. пособие / С.В. Скворцов; Рязан. гос. радиотехн. акад. Рязань 21001. 56 с.

Приложение 1

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;

namespace kurs_lipo_shmakov_846
{
  public partial class Form1 : Form
  {
    // Директория в которой находится исполняемая программа
    readonly string CurrentDirectory = Directory.GetCurrentDirectory();
    // Расположение считываемого файла
    string DirectoryReadFile;
    public Form1() => InitializeComponent();

    // Чтение исходных текстов программ на языке Паскаль
    public void OpenFile()
    {
      using OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.InitialDirectory = CurrentDirectory;
        openFileDialog.Filter = "Исходные тексты программ (*.pas)|*.pas";
        openFileDialog.FilterIndex = 2;
        openFileDialog.RestoreDirectory = true;

      if (openFileDialog.ShowDialog() == DialogResult.OK)
      {
        // Расположение считываемого файла
        DirectoryReadFile = openFileDialog.FileName;

        // Считывание содержимого файла в поток
        var fileStream = openFileDialog.OpenFile();

        using StreamReader reader = new StreamReader(fileStream);
        string fileContent = reader.ReadToEnd();
        textBox.Text = fileContent;
      }

    }

    // Переопределение стандартной функции поиска слова в строке, для
расширения возможности поиска без учета регистра
    public static bool Contains( string source, string toCheck, StringComparison comp)
    {
      return source != null && toCheck != null && source.IndexOf(toCheck, comp) >= 0;
    }




    // Алгоритм
```

```
        public void Algorithm()
        {
            // Счетчик строки
            int counter = 1;
            // Сообщение об ошибке
            string Warning = "Ошибок не найдено!";
            // Список всех анализируемых лексем
            List<string> lexemes = new List<string>() { "repeat", "case", "begin", "record" };
            // Стек лексем, найденных в тексте программы
            Stack<string> tokens = new Stack<string>();

            // Считываем текст программы по строкам
            foreach (string line in System.IO.File.ReadLines(DirectoryReadFile))
            {
                // В каждой строке ищем все анализируемые лексемы
                foreach (string lexeme in lexemes)
                {
                    // Если лексема найдена, добавляем ее в стек
                    if (line.Contains(lexeme, StringComparison.OrdinalIgnoreCase))
                        tokens.Push(lexeme);

                }

                // Если встретили правую скобку типа until
                if (line.Contains("until", StringComparison.OrdinalIgnoreCase)))
                {
                    // Если при найденной правой скобке пустой стек
                    if (tokens.Count == 0)
                    {
                        Warning = "(" + DateTime.Now + ") " + "Отсутствует левая скобка: [Строка]
= " + counter;
                        tokens.Clear();
                        break;
                    }

                    // Если левая скобка из стека не совпадает с найденной правой по типу
                    if (tokens.Pop() != lexemes[0])
                    {
                        Warning = "(" + DateTime.Now + ") " +  "Не совпадают типы скобок:
[Строка] = " + counter;
                        tokens.Clear();
                        break;
                    }
                }

                // Если встретили правую скобку типа end
                if (line.Contains("end", StringComparison.OrdinalIgnoreCase)))
                {
                    // Если при найденной правой скобке пустой стек
                    if (tokens.Count == 0)
                    {
```

```
                Warning = "(" + DateTime.Now + ") " + "Отсутствует левая скобка: " +
"[Строка] = " + counter;
                tokens.Clear();
                break;
            }

            // Проверяем совпадают ли по типу найденная правая скобка с левыми
            if ((tokens.Peek() != lexemes[1]) && (tokens.Peek() != lexemes[2]) &&
(tokens.Peek() != lexemes[3]))
            {
                Warning = "(" + DateTime.Now + ") " + "Не совпадают типы скобок: [Строка]
= " + counter;
                tokens.Clear();
                break;
            }
            tokens.Pop();
          }
          counter++;
        }

        // Если после прохода алготима стек оказался не пустой
        if (tokens.Count != 0)
        {
          Warning = "(" + DateTime.Now + ") " +  "Отсутствует правая скобка: [Строка] = "
+ counter;

        }

        textBoxWarnings.Text = Warning + "\r\n" + "Отчет об ошибках сохранен:" +
Directory.GetCurrentDirectory() + "/Warnings.txt";
        SaveWarningFile(Warning);
    }

    // Сохранение результатов анализа в файл
    public void SaveWarningFile(string Warning)
    {
        using FileStream fstream = new FileStream(Directory.GetCurrentDirectory() +
"/Warnings.txt", FileMode.OpenOrCreate);
          // Преобразуем строку в байты
          byte[] array = System.Text.Encoding.Default.GetBytes(Warning);
          // Запись массива байтов в файл
          fstream.Write(array, 0, array.Length);
    }

    // Клик меню открыть
    private void MenuOpen_Click(object sender, EventArgs e)
    {
      OpenFile();
      Algorithm();
    }
```

```
        // Клик меню о программе
        private void MenuAbout_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Дисциплина: Лингвистическое и программное обеспечение
САПР." +
                "Тема курсового проекта: Синтаксический анализ программ.\n" +
                "Вариант 7: СИНТАКСИЧЕСКИЙ АНАЛИЗ ПАРНЫХ ЭЛЕМЕНТОВ
ПРОГРАММ\n" +
                "Выполнил: Студент 4 курса, Кафедра САПР ВС, Группа 846, Шмаков А.Ю.,
Проверил Скворцов С.В.");
        }

        // Клик меню выйти
        private void MenuClose_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```