|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК5 «Системы обработки информации»***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЛАБОРАТОРНАЯ РАБОТА**

**«Разработка лексического анализатора»**

**ДИСЦИПЛИНА: «Конструирование компиляторов»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК5-11М | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ефремов И.В.)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Потапов А.Е.)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

Цели и задачи

**Цели работы:**

* Получить практические навыки синтеза лексического анализаторов.
* Разработать программу лексического анализа входного файла программы с разбиением его на лексемы и записи их в выходной файл.

**Порядок выполнения работы:**

* Ознакомиться с разделом «Построение лексического анализатора».
* По варианту задания определить, какие классы лексем будут в данном языке.
* Составить контрольные примеры на реализуемом языке. Хотя бы один пример должен проверять поведение вашей программы при наличии недопустимых символов в транслируемом файле.
* Запрограммировать и отладить модуль сканирования. Выполнить тестирование на контрольных примерах. Результатом работы должна быть таблица, содержащая лексемы и признаки их классов. Необходимо включить в результирующий файл информацию о номерах строк исходного текста транслируемой программы.
* Оформить отчет.

Практическая часть

**Вариант 4**

Ниже приведена БНФ реализуемого языка согласно варианту.

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= <Список присваиваний>

<Объявление переменных> ::= Var <Список переменных> ;

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список присваиваний> ::= <Присваивание>|

<Присваивание> <Список присваиваний>

<Присваивание> ::=<Идент> := <Выражение>;

<Выражение> ::= <Ун.оп><Подвыражение>|<Подвыражение>

<Подвыражение> :: = (<Выражение> )|<Операнд> |

<Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп .>::= "-"

<Бин.оп.> ::= "-" | " + " | "\*" | "/"

<Операнд> ::= <Идент>|<Константа>

<Идент> ::= <Буква><Идент>|<Буква>

<Константа> ::= <Цифра><Константа>|<Цифра>

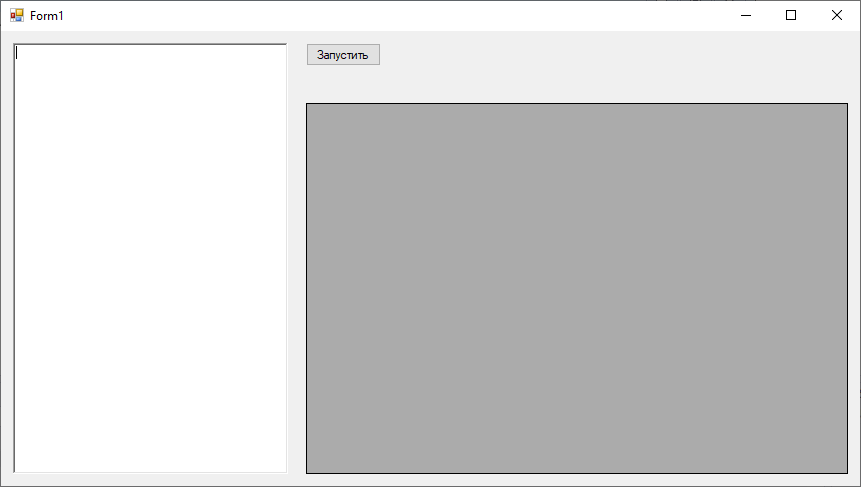
1. Классы лексем

Список классов лексем реализуемого языка приведен в таблице 1.

1. Список классов лексем реализуемого языка

|  |  |  |
| --- | --- | --- |
| **Класс лексемы** | **Флаг** | **Описание** |
| Зарезервированное слово | Variable | Var |
| Число | Constant | Числа |
| Присвоение типа переменной | Assign | := |
| Терминал строки | Semilicon | ; |
| Запятая | Semilicon | , |
| Скобки | Right bracket / Left bracket | (  ) |
| Унарный оператор | Minus | - |
| Бинарный оператор | Addition, Subtraction,  Multiplication, Division | -  +  \*  / |
| Идентификатор | Ident | Переменные |

Интерфейс программы представлен на рисунке 1.



1. Интерфейс программы

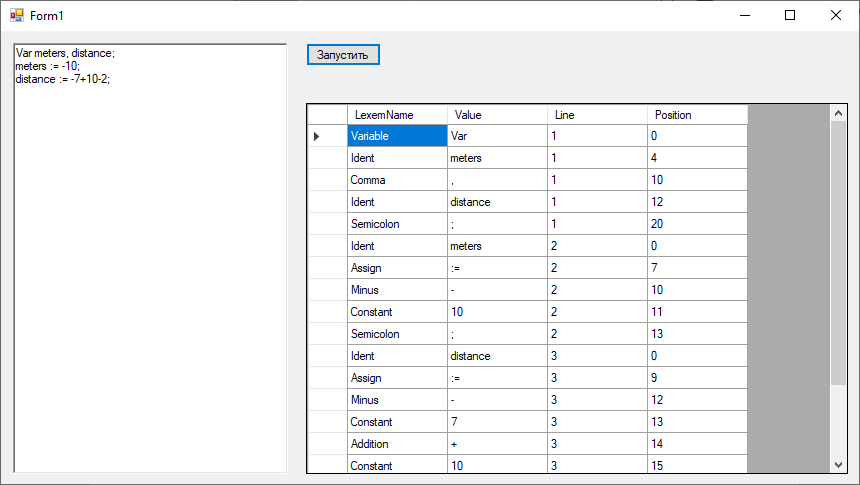
Исходный код:

Var meters, distance;

meters := -10;

distance := -7+10-2;

Результат работы программы лексического анализа приведен на рисунке 2.



1. Результат работы программы

Выводы

В ходе выполнения лабораторной работы были получены практические навыки синтеза лексического анализатора. Разработана программа лексического анализа входного файла программы с разбиением его на лексемы и записи их в выходной файл. Проведено ознакомление с разделом «Построение лексического анализатора». Определены классы лексем реализуемого языка. Составлены контрольные примеры на реализуемом языке. Запрограммирован и отлажен модуль сканирования. Выполнено тестирование на контрольных примерах.

ПРИЛОЖЕНИЕ А

Листинг 1 –Файл класса Form1

**public** **partial** class Form1 **:** Form

**{**

**private** string text**;**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

**private** void richTextBox1\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

**throw** **new** NotImplementedException**();**

**}**

**private** void button1\_Click**(object** sender**,** EventArgs e**)**

**{**

**try**

**{**

var lexer **=** **new** Lexer**(**TextConsole**.**Text**);**

var tokens **=** lexer**.**LexAnalisis**();**

var bindingList **=** **new** BindingList**<**Token**>(**tokens**);**

dataGridView1**.**DataSource **=** **new** BindingSource**(**bindingList**,** **null);**

**}**

**catch** **(**CodeException ex**)**

**{**

MessageBox**.**Show**(**$"{ex.Message}"**);**

**}**

**}**

**}**

Листинг 2 – Файл класса Lexer

**public** class Lexer

**{**

**private** string \_code**;**

**private** int \_line **=** 1**;**

**private** int \_position **=** 0**;**

**private** List**<**Token**>** \_tokens **=** **new** List**<**Token**>();**

**public** Lexer**(**string code**)**

**{**

\_code **=** code**;**

**}**

/// <summary>

/// Возвращаем все токены кода

/// </summary>

/// <returns></returns>

**public** List**<**Token**>** LexAnalisis**()**

**{**

**while** **(**GetTokensFromCode**())**

**{**

**}**

**return** \_tokens**;**

**}**

//todo надо красиво переделать

/// <summary>

/// Ищем в коде все токены.

/// </summary>

/// <returns></returns>

**private** bool GetTokensFromCode**()**

**{**

**if** **(**\_code **!=** **null** **&&** \_code**.**Length **==** 0**)**

**{**

**return** **false;**

**}**

**foreach** **(**var tokenType **in** \_tokenTypes**)**

**{**

var regex **=** **new** Regex**(**"^" **+** tokenType**.**Regex**);**

**if** **(**\_code **==** **null)** **return** **true;**

var result **=** regex**.**Match**(**\_code**).**ToString**();**

**if** **(**string**.**IsNullOrEmpty**(**result**))** **continue;**

var \_token **=** **new** Token **{** Type **=** tokenType**,** LexemName **=** tokenType**.**Name**,** Value **=** result**,** Line **=** \_line**,** Position **=** \_position **};**

**if** **(**tokenType **==** \_tokenTypes**[**1**]** **&&** \_tokens**.**Count **!=** 0

**&&** **(**\_tokens**[**\_tokens**.**Count **-**1**].**Type **==** \_tokenTypes**[**3**]** **||** \_tokens**[**\_tokens**.**Count **-** 1**].**Type **==** \_tokenTypes**[**14**]))**

**{**

\_token**.**Type **=** \_tokenTypes**[**8**];**

\_token**.**LexemName **=** \_tokenTypes**[**8**].**Name**;**

**}**

**if** **(**result **!=** " " **&&** result **!=** "\n"**)**

**{**

\_tokens**.**Add**(**\_token**);**

**}**

**if** **(**result **==** "\n"**)**

**{**

\_line**++;**

\_position **=** 0**-**1**;**

**}**

\_position **+=** result**.**Length**;**

\_code **=** \_code**.**Substring**(**result**.**Length**);**

**return** **true;**

**}**

**throw** **new** CodeException**(**$"Ошибка в коде. Строка {\_line}, позиция {\_position}"**);**

**}**

/// <summary>

/// Список всех токенов.

/// </summary>

**private** List**<**TokenType**>** \_tokenTypes **=** **new** List**<**TokenType**>**

**{**

**new** TokenType **{** Name **=** "Variable"**,** Regex **=** "Var" **},**

**new** TokenType **{** Name **=** "Minus"**,** Regex **=** "-" **},**

**new** TokenType **{** Name **=** "Space"**,** Regex **=** " " **},**

**new** TokenType **{** Name **=** "Ident"**,** Regex **=** "[a-z]\*" **},**

**new** TokenType **{** Name **=** "Comma"**,** Regex **=** "," **},**

**new** TokenType **{** Name **=** "Semicolon"**,** Regex **=** ";" **},**

**new** TokenType **{** Name **=** "Assign"**,** Regex **=** ":=" **},**

**new** TokenType **{** Name **=** "Addition"**,** Regex **=** "\\+" **},**

**new** TokenType **{** Name **=** "Subtraction"**,** Regex **=** "-" **},**

**new** TokenType **{** Name **=** "Multiplication"**,** Regex **=** "\*" **},**

**new** TokenType **{** Name **=** "Division"**,** Regex **=** "/" **},**

**new** TokenType **{** Name **=** "Left bracket"**,** Regex **=** "\\(" **},**

**new** TokenType **{** Name **=** "Right bracket"**,** Regex **=** "\\)" **},**

**new** TokenType **{** Name **=** "Line break"**,** Regex **=** "\\n" **},**

**new** TokenType **{** Name **=** "Constant"**,** Regex **=** "[0-9]\*" **}**

**};**

**}**

Листинг 3 – Файл класса Token

**public** class Token

**{**

/// <summary>

/// Тип токена.

/// </summary>

**[**Browsable**(false)]**

**public** TokenType Type **{** **get;** **set;** **}**

/// <summary>

/// Название лексемы.

/// </summary>

**public** string LexemName **{** **get;** **set;** **}**

/// <summary>

/// Значение лексемы.

/// </summary>

**public** string Value **{** **get;** **set;** **}**

/// <summary>

/// Номер линии, на которой находиться токен.

/// </summary>

**public** int Line **{** **get;** **set;** **}**

/// <summary>

/// Позиция токена на строчке.

/// </summary>

**public** int Position **{** **get;** **set;** **}**

**}**