Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра "Компьютерные технологии"

Е.А. Воробьева, Ю.В. Прощенко, А.А. Шалыто

Текстовый редактор для цветной “подсветки синтаксиса” программного кода

Проектная документация

Проект создан в рамках
"Движения за открытую проектную документацию"
http://is.ifmo.ru

Санкт-Петербург
2004

Оглавление

Введение

С появлением конкуренции на рынке языков программирования возникла необходимость обеспечить программисту максимально удобную среду разработки – *Interface Development Environment (IDE)*. Такая среда повышает производительность труда программиста.

В настоящей работе предложена среда разработки *Fairy Editor (FEd),* которая поддерживает **цветную "подсветку синтаксиса" программного кода**. Текст программы с цветной подсветкой повышает производительность труда программиста, благодаря тому, что такой код удобнее и быстрее читать и понимать.

Эту задачу целесообразно решать с использованием автоматов. Для ее алгоритмизации и программирования оказалось удобным применить SWITCH-технологию [1]. Подробно ознакомиться с этой технологией и с конкретными примерами ее применения можно на сайте http://is.ifmo.ru.

При реализации проекта была разработана указанная среда, позволяющая динамически подсвечивать синтаксис различных языков.

Существуют аналоги данной среды разработки. Перечислим некоторые из них:

- *UltraEdit 7.0* - текстовый редактор, позволяющий редактировать большие файлы (до 2 Гб) и имеющий встроенную подсветку синтаксиса при написании программ на языках *C*, *Basic*, *HTML* и т.д.;
- *R-WIN Editor 6.0* - многооконный текстовый редактор, который распознает кодировку *TXT* и *HTML* файлов (кириллица 855, 866, KOI8, 1251). В нем реализованы: поиск, замена текста, печать, просмотр, история открытия файлов.

Более подробно об этих программах можно прочитать на сайте http://softsearch.ru/programs. Указанные редакторы не только не содержат проектную документацию, но и открытых исходных кодов.

Настоящая работа призвана восполнить указанный пробел – разработать проектную документацию и создать по ней программу, содержащую такие коды.

1. Постановка задачи

Целью настоящей работы является создание текстового редактора с подсветкой синтаксиса программного кода.

Редактор должен обеспечивать представление текстов программ, как показано на рис. 1.

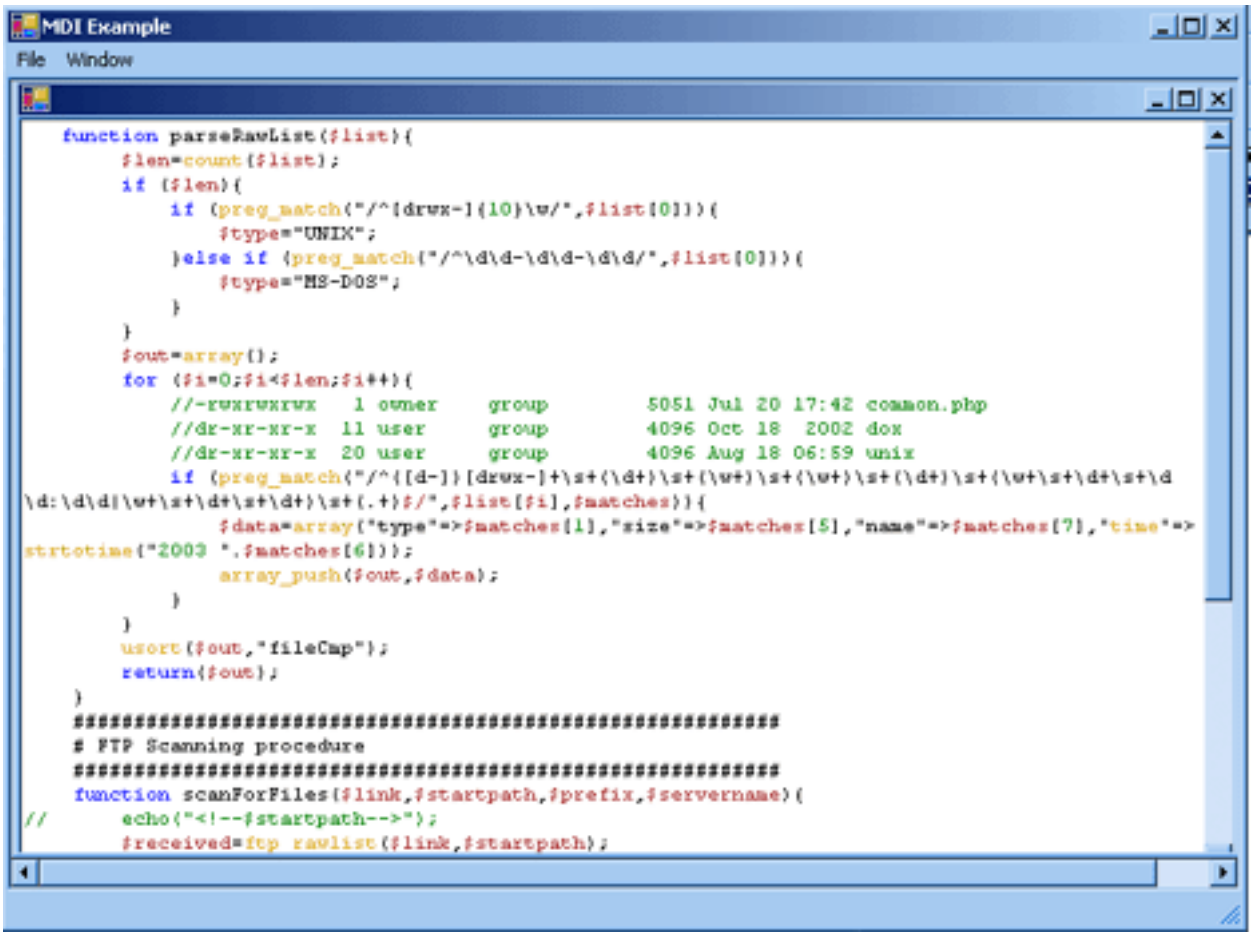

Рис. 1. Внешний вид среды

2. Структура программы

Программа *FEd* написана на языке *C#* в среде *Microsoft Visual C# .NET*. Для работы программы требуется *Microsoft .NET Framework v1.0.3705* или более поздняя версия.

В программе можно выделить два основных элемента:

- текст программы;
- словарь.

Первый элемент – это текст, который может модифицировать пользователь.

Второй элемент – это набор ключевых (зарезервированных) слов. В каждом языке программирования есть такие слова, которые несут в себе определенную информацию (указания) для компилятора. Поэтому их нельзя использовать для других целей.

Программа *FEd* имеет оконный интерфейс (рис.1). Интерфейс содержит:

- меню пользователя;
- поле для редактирования теста программы (текстовое поле).

В меню реализованы стандартные функции, такие как загрузка, сохранение и создание нового файла, отмена последнего действия, вставку текста и т.д.

При загрузке и создании файла его текст отображается в текстовое поле, где программист может работать с текстом.

Как только произошло какое-либо изменение в текстовом поле, инициализируется событие *TextChanged.* Далее производится анализ изменений, в ходе которого определяется, цвет какой части текста мог измениться, а затем происходит передача управления автомату *Parser* (разд. 3), который обеспечивает окончательную раскраску текста.

Код программы *FEd* разделен на несколько файлов. Это связано с тем, что код программы можно разделить на тематически разные части, которые удобно хранить в различных файлах.

Программа состоит из файлов, приведенных в Приложении 1:

- *AdvTextBox.cs*;
- *AssemblyInfo.cs*;
- *Block.cs*;
- *BlockList.cs*;
- *Document.cs*;
- *Group.cs*;
- *GroupList.cs*;
- *Log.cs*;
- *MainForm.cs*;
- *Parser.cs*.

В файле *AdvTextBox.cs* описан класс *AdvTextBox*, объектом которого является поле для редактирования текста программ. В этом классе перегружена процедура обработки сообщений.

Основная информация о программе хранится в файле *AssemblyInfo.cs*.

В файле *Block.cs* описан класс *Block*, который используется для хранения блоков текста. Файл *BlockList.cs* содержит описание массива объектов класса *Block* (разд. 2.1).

В файле *Document.cs* реализован поиск блоков текста, в которых были произведены изменения. Затем эти блоки передаются автомату *Parser* для получения окончательной раскраски.

В файле *Group.cs* описан класс *Group*, который служит для хранения ключевых слов, относящихся к данной группе. Поскольку используется не одна группа ключевых слов, то в

программе описан класс *GroupList* в файле *GroupList.cs*, который является массивом объектов класса *Group* (разд. 2.2).

Класс *Log*, который описан в файле *Log.cs*, используется для протоколирования действий автомата.

В файле *MainForm.cs* описан основной класс *MainForm* создания графического интерфейса. В нем реализовано создание основного окна приложения, создание дочерних окон и связь с объектом класса *Document*.

Автомат *Parser* описан в файле *Parser.cs*. В нем реализована обработка строк, в результате которой текст меняет свой цвет.

2.1. Представление текста программы

При отображении кода программы на экран важен не только его текст, но и используемый цвет. Поэтому при выводе фрагмента программы на экран необходимо знать два его параметра: сам текст и его цвет. Поэтому будем использовать две структуры: для хранения текста и для хранения его цвета.

Структура для хранения текста – строка.

Структура для хранения цвета текста - более сложная структура, которую рассмотрим на примере:

```
echo"hello world";
```

Здесь:

- `echo` – команда вывода на экран текста на языке *PHP;*
- `"hello world"` – строка;
- ; - конец строки.

Рассмотренные компоненты разнотипны, поэтому они должны быть разного цвета:

```
echo"hello world";
```

Этот фрагмент программы представим тремя отрезками, которые имеют начало, конец и свойство (цвет отрезка):

первый отрезок: начало - 0, конец - 4, свойство - зеленый;
второй отрезок: начало - 4, конец - 17, свойство - синий;
третий отрезок: начало - 17, конец - 18, свойство - черный.

Каждый отрезок сопоставлен с определенной частью текста:

первый: `echo`
второй: `"hello world"`
третий: `;`

Таким образом, структура хранения цвета текста является последовательностью непрерывных взаимонепересекающихся отрезков. При этом предполагается, что конец одного отрезка совпадает с началом следующего.

В программе *FEd* каждый отрезок представлен объектом класса *Block,* а список отрезков (весь текст) – объектом класса *BlockList*.

Структура представления данных, описанная в настоящем разделе, обеспечивает не только простоту, скорость и удобство подсветки текста, но и позволяет оптимизировать данные, подаваемые на вход автомата *Parser*.

2.2. Представление словаря

Для удобства список зарезервированных слов разделим на группы:

1. *numbers* – цифры;
2. *delimiters* – разделители;
3. *keywords* – ключевые слова;
4. *words* – слова;
5. *operators* – операторы;
6. *variables* – переменные;
7. *constants* – константы;
8. *functions* – функции.

Каждой группе сопоставляется свой цвет отображения на экран.

При открытии или создании нового файла загружается словарь зарезервированных слов из файла `wordfile.txt`, который имеет следующий формат:

```
#group_name word1[ word2[ word2]]
```

Здесь `group_name` – имя одной из вышеперечисленных групп. После имени группы может располагаться любое количество “слов”. Квадратные скобки обозначают включение их содержимого ноль и более раз.

Пример:

```
#numbers 1 2 3 4 5 6 7 8 9 0
#delimiters ~ ! @ % ^ & * ( )
#keywords $argc $argv $_COOKIE
#words __sleep __wakeup as bre
#operators ! != % & && ( ) * *=
#variables ** $
#constants __FILE__ __LINE__
#functions abs accept_connect
```

При загрузке словаря зарезервированные слова сохраняются в массиве по группам. После этого, при необходимости, можно легко проверить является ли данное слово зарезервированным и к какой группе оно относится.

2.3. Принцип работы

Как уже было описано ранее, при каком-либо изменении в текстовой области инициализируется событие *TextChanged*. После этого управление передается процедуре обработки текста. На основе анализа произведенных изменений в текстовой области, процедура определяет какие блоки текста могли изменить свой цвет, а затем передает эти блоки автомату.

3. Автомат разбора текста

3.1. Словесное описание

Автомат выполняет следующие действия:

- инициализируется при входе;
- разбирает текст;
- при необходимости меняет блочную структуру текста;
- протоколирует свое поведение, описывая каждый шаг.

3.2. Схема связей

Схема связей автомата *Parse* представлена на рис. 2.

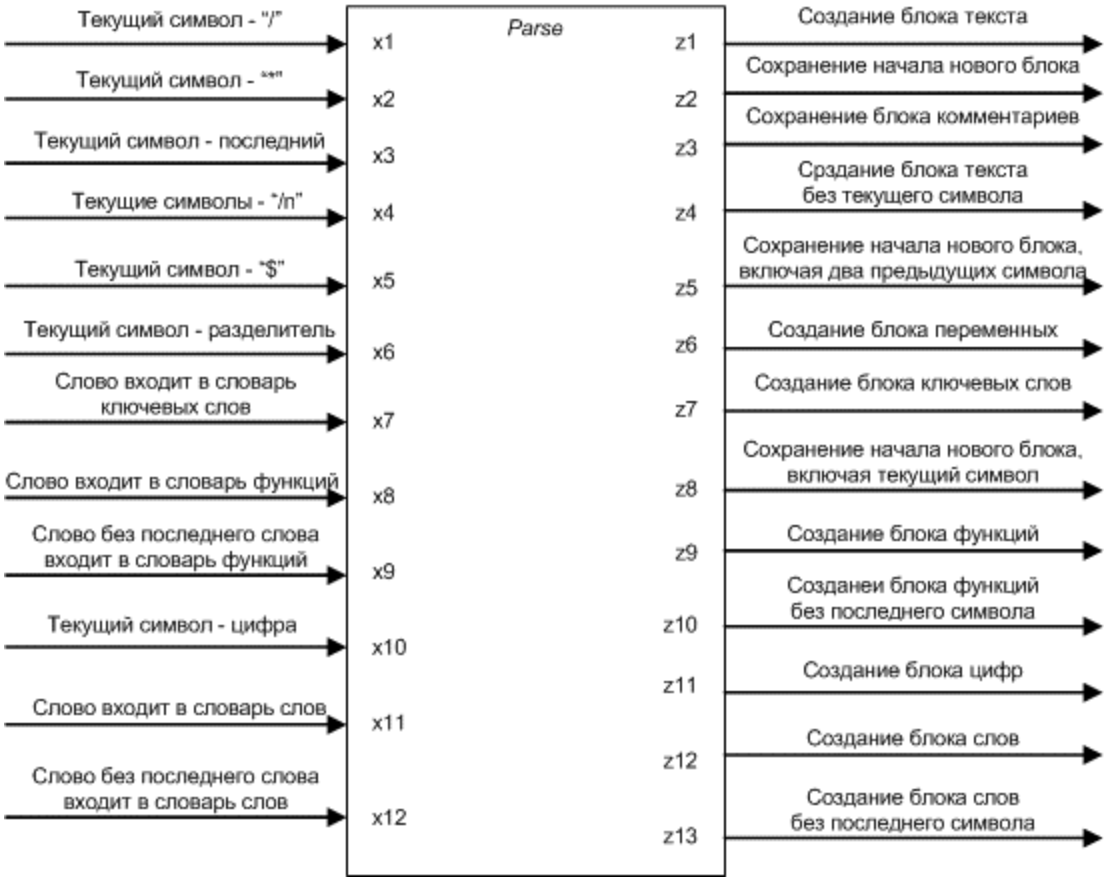

Рис. 2. Схема связей автомата *Parse*

3.3. Граф переходов

Граф переходов автомата *Parse* представлен на рис. 3.

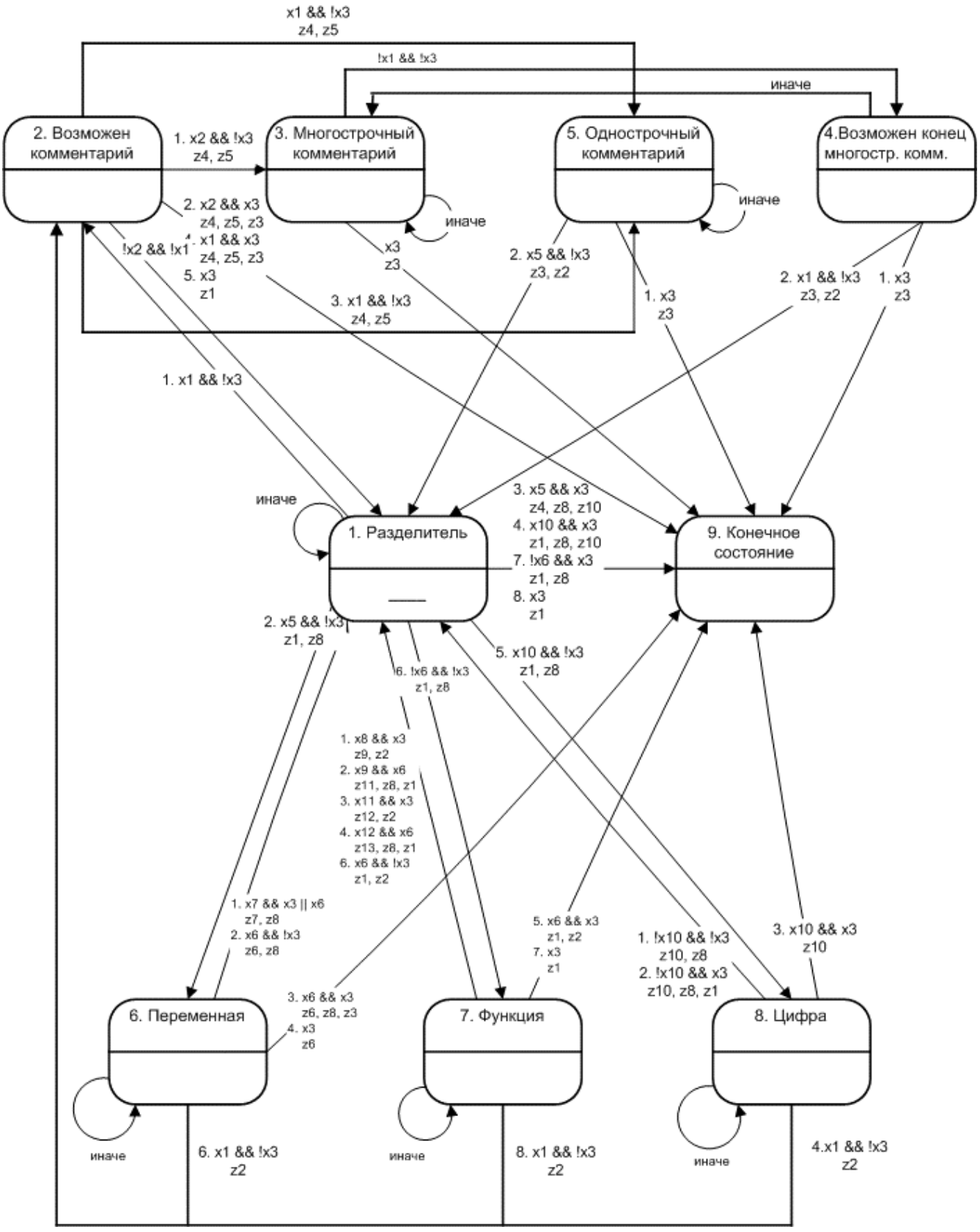

Рис. 3. Граф переходов

4. Протоколирование действий автомата

4.1. Протоколирование

Во время работы программы ведется протокол поведения автомата. При этом на каждом шаге автомат записывает в файл-протокол изменения или пишет, что остался в состоянии. Такой подход удобен при отладке программы, поскольку с помощью файла-протокола можно намного быстрее найти ошибку.

Протокол работы автомата выводится в файл *log.html*.

4.2. Пример отладочного протокола

Пусть задан код программы:

```
/* Check file input */
global $dbml;
if ( !isset($dbml) ) {
    echo "No data sent!";
    exit;
}
global $all;

/* Counters */
$er = 0;
$ad = 0;
$up = 0;
```

В результате работы среды должна быть получена раскрашенная программа следующего вида:

```
/* Check file input */
global $dbml;
if ( !isset($dbml) ) {
    echo "No data sent!";
    exit;
}
global $all;

/* Counters */
$er = 0;
$ad = 0;
$up = 0;
```

Правильность работы программы демонстрируется протоколом, приведенным в Приложении 2.

Заключение

Применение автоматного подхода в этой задаче показало, что он достаточно эффективен при ее спецификации, реализации и протоколировании.

Литература

1. *Хопкрофт Дж., Мотвани Р., Ульман Дж.* Введение в теорию автоматов, языков и вычислений. СПб.: Вильямс, 2002.
2. *Корман Т., Лейзерсон Ч., Ривест Р.* Алгоритмы построение и анализ. М.: МЦНМО, 2001.

Приложение 1. Исходные коды программы

П1.1. AdvTextBox.cs

В файле *AdvTextBox.cs* описан класс *AdvTextBox*, объектом которого является поле для редактирования текста программ. В этом классе перегружена процедура обработки сообщений.

```
using System;
using System.Windows.Forms;
namespace MDIApp.GUI
{
    /// <summary>
    /// AdvTextBox - класс с возможностью отмены перерисовки
    /// </summary>
    public class AdvTextBox : RichTextBox
    {

        // #define WM_PAINT                        0x000F
        const short WM_PAINT = 0x00f;

        public AdvTextBox()
        {
            //
            // TODO: Add constructor logic here
            //
            //          SetStyle(ControlStyles.AllPaintingInWmPaint, true);
            //          SetStyle(ControlStyles.DoubleBuffer , true);
            //          SetStyle(ControlStyles.UserPaint , true);
        }

        public bool _Paint = true;


        protected override void WndProc(ref System.Windows.Forms.Message m)
        {

            // Перекрываем процедуру обработки сообщений и отменяем перерисовку
            // (если нам это надо)

            if (m.Msg == WM_PAINT)
            {
                if (_Paint)
                    base.WndProc(ref m);

                else
                    m.Result = IntPtr.Zero;
            }

            else
                base.WndProc (ref m);
        }
    }

}
```

П1.2. AssemblyInfo.cs

Основная информация о программе храниться в файле *AssemblyInfo.cs*.

```
using System.Reflection;
using System.Runtime.CompilerServices;

//
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
//
[assembly: AssemblyTitle("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

//
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:

[assembly: AssemblyVersion("1.0.*")]

//
// In order to sign your assembly you must specify a key to use. Refer to the
// Microsoft .NET Framework documentation for more information on assembly signing.
//
// Use the attributes below to control which key is used for signing.
//
// Notes:
//   (*) If no key is specified, the assembly is not signed.
//   (*) KeyName refers to a key that has been installed in the Crypto Service
//       Provider (CSP) on your machine. KeyFile refers to a file which contains
//       a key.
//   (*) If the KeyFile and the KeyName values are both specified, the
//       following processing occurs:
//       (1) If the KeyName can be found in the CSP, that key is used.
//       (2) If the KeyName does not exist and the KeyFile does exist, the key
//           in the KeyFile is installed into the CSP and used.
//   (*) In order to create a KeyFile, you can use the sn.exe (Strong Name) utility.
//       When specifying the KeyFile, the location of the KeyFile should be
//       relative to the project output directory which is
//       %Project Directory%\obj\<configuration>. For example, if your KeyFile is
//       located in the project directory, you would specify the AssemblyKeyFile
//       attribute as [assembly: AssemblyKeyFile("..\\..\\mykey.snk")]
//   (*) Delay Signing is an advanced option - see the Microsoft .NET Framework
//       documentation for more information on this.
//
```

```
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile("")]
[assembly: AssemblyKeyName("")]
```

П1.3. Block.cs

В файле *Block.cs* описан класс *Block*, который используется для хранения блоков текста.

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

namespace MDIApp.Syntax
{
    /// <summary>
    ///
    /// </summary>
    public class Block : IComparable
    {
        private BlockList parent = null;

        private int begin;
        private int end;
        private int type;

        public int Index
        {
            get
            {
                return parent.IndexOf(this);
            }
        }

        public int Begin
        {
            get
            {
                return this.begin;
            }
            set
            {
                begin = value;
            }
        }

        public int End
        {
            get
            {
                return this.end;
            }
            set
            {
                end = value;
            }
        }
```

```
    public int Type
    {
        get
        {
            return this.type;
        }
        set
        {
            type = value;
        }
    }


    private bool repaint;

    #region конструкторы
    public Block()
        : this(0, 0, 0, false) {}

    public Block(int begin, int end)
        : this(begin, end, 0, true) {}

    public Block(int begin, int end, int type)
        : this(begin, end, type, true) {}

    public Block(ref BlockList parent, int begin, int end)
        : this(ref parent, begin, end, 0, true) {}

    public Block(ref BlockList parent, int begin, int end, int type)
        : this(ref parent, begin, end, type, true) {}

    public Block(ref BlockList parent)
        : this(ref parent, 0, 0, 0, false) {}

    public Block(ref BlockList parent, int begin, int end,int type,bool repaint)
        : this(begin, end, type, repaint) { this.parent = parent; }

    public Block(int begin, int end, int type, bool repaint)
    {
        this.begin = begin;
        this.end   = end;
        this.type  = type;
        this.repaint = repaint;
}

    public Block(Block s)
        : this(ref s.parent, s.begin, s.end, s.type, s.repaint) {
    parent = parent;
    }
    #endregion

    public void SetParent(ref BlockList parent)
    {
        this.parent = parent;
    }

    public void mergeWithLeft()
    {
```

```
    mergeWithLeft(false);
}

public void mergeWithRight()
{
    mergeWithRight(false);
}

public void mergeWithLeft(bool repaint)
{
    if (Index != 0)
        begin = parent[Index -1].begin;
    this.repaint = repaint;
}

public void mergeWithRight(bool repaint)
{
    if (Index != parent.Count-1)
        end = parent[Index+1].end;
    this.repaint = repaint;
}

public void MoveRight(int n)
{
    this.begin += n;
    this.end += n;
}


public void Collapse(bool start)
{
    if (start)
        begin = end;
    else
        end = begin;
}


public override string ToString()
{
    if (parent != null)
        return
            "begin: " + "\"" + this.begin.ToString() + "\"\n" +
            "end: " + "\"" + this.type.ToString() + "\"\n" +
            "type: " + "\"" + this.type.ToString() + "\"\n" +
            "index: " + "\"" + this.Index.ToString() + "\"\n";
    else
        return
            "begin: " + "\"" + this.begin.ToString() + "\"\n" +
            "end: " + "\"" + this.type.ToString() + "\"\n" +
            "type: " + "\"" + this.type.ToString() + "\"\n" +
            "parent: is null";
}

#region IComparable Members
public int CompareTo(Object obj)
{
    if (obj is Block)
    {
```

```
                int a = this.begin.CompareTo(((Block)obj).begin);
                int b = this.end.CompareTo(((Block)obj).end);

                if (a == b)
                    return a;

                a = b;
            }
            obj.ToString();
            return 0;
        }
        #endregion
    }
}
```

П1.4. BlockList.cs

Файл *BlockList.cs* содержит описание массива объектов класса *Block*.

```
using System;
using System.Collections;

namespace MDIApp.Syntax
{

    /// <summary>
    ///
    /// </summary>
    public class BlockList : ArrayList, ICloneable
    {

        public BlockList() : base() {}

        public new Block this[int index]
        {
            get
            {
                return (Block)base[index];
            }
            set
            {
                base[index] = value;
            }
        }

        public override string ToString()
        {

            string result = "";
            foreach(Block block in this)
                result += block.Begin.ToString() + "-" + block.End.ToString() + ";";
            return result;

        }
    }
}
```

П1.5. Document.cs

В файле *Document.cs* зеализован поиск блоков текста, в которых были произведены изменения. Затем эти блоки передаются автомату *Parser* для получения окончательной расскраски.

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;
using System.Collections;

using MDIApp.Syntax;

namespace MDIApp.GUI
{
    public class Document : Form
    {
        private string oldText;
        private int selStart;
        private int selLength;
        private MDIApp.GUI.AdvTextBox editor;

        private string path;

        private BlockList blocks;
        private GroupList language;
       private Parser parser;


        /// <summary>
        ///    Required designer variable.
        /// </summary>
        //        private System.ComponentModel.Container components;

        public Document(string docName) : base()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            this.path = Application.ExecutablePath.Remove(
               Application.ExecutablePath.LastIndexOf("\\"),
                              Application.ExecutablePath.Length -
                              Application.ExecutablePath.LastIndexOf("\\")) + "\\";

            this.editor.TextChanged += new System.EventHandler(this.editor_TextChanged);

            this.editor.MouseUp += new MouseEventHandler(this.editor_MouseUp);
            this.editor.KeyUp += new KeyEventHandler(this.editor_KeyUp);

            oldText = "";

            language = new GroupList();
            language.SetZis(ref language);
            parser = new Parser(ref language);
               parser.log.BeginLog();
```

```
        blocks = new BlockList();
    blocks.Add(new Block(ref blocks));
  }

#region Windows Form Designer generated code

#region Dispose
/// <summary>
///    Clean up any resources being used.
/// </summary>
protected override void Dispose(bool disposing)
{
     parser.log.EndLog();
    base.Dispose(disposing);
}
#endregion

/// <summary>
///    Required method for Designer support - do not modify
///    the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        this.editor = new MDIApp.GUI.AdvTextBox();
        this.SuspendLayout();
        //
        // editor
        //
        this.editor.Anchor = (((System.Windows.Forms.AnchorStyles.Top |
              System.Windows.Forms.AnchorStyles.Bottom)
              | System.Windows.Forms.AnchorStyles.Left)
              | System.Windows.Forms.AnchorStyles.Right);
        this.editor.Font = new System.Drawing.Font("Courier New", 8.25F,
            System.Drawing.FontStyle.Regular,
            System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));
        this.editor.ForeColor = System.Drawing.SystemColors.WindowText;
        this.editor.Name = "editor";
        this.editor.Size = new System.Drawing.Size(560, 314);
        this.editor.TabIndex = 2;
        this.editor.Text = "";
        //
        // Document
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
        this.ClientSize = new System.Drawing.Size(560, 301);
        this.Controls.AddRange(new System.Windows.Forms.Control[] {

        this.editor});
        this.Name = "Document";
        this.ResumeLayout(false);

  }

#endregion

private void buttonFormat_Click(object sender, System.EventArgs e)
{
    language.Clear();
    language.LoadFromFile("wordfile.txt");
```

```
    string values = "";

    foreach (MDIApp.Syntax.Group group in language)
    {
        values += group.ToString();
    }

    MessageBox.Show(values);
}

private void editor_CursorMoved()
{
    selStart = editor.SelectionStart;
    selLength = editor.SelectionLength;
}

private void editor_KeyUp(object sender, KeyEventArgs e)
{
    editor_CursorMoved();
}

private void editor_MouseUp(object sender, MouseEventArgs e)
{
    editor_CursorMoved();
}
private void editor_TextChanged(object sender, System.EventArgs e)
{
        editor._Paint = false;

        int ss = editor.SelectionStart;
        int sl = editor.SelectionLength;

        if (editor.Text.Length < oldText.Length)
        {
              Block head = null, tail = null;
              foreach(Block block in blocks)
              {
                // Начало...
                if ((head == null) && (block.Begin <= 0) && (block.End >= 0))
                head = block;
                // ...и конец
                if ((tail == null) && (block.Begin <= (editor.Text.Length))
                          && (block.End >= (editor.Text.Length)))
                          tail = block;
                // Выходим из поиска если всё нашли
                if ((tail != null) && (head != null))
                break;
              }

              int headIndex = head.Index;
              int offset = head.Begin;

              #region удаление блоков, находящихся между head и tail
              if ((tail.Index-head.Index-2) > 0 )
                    blocks.RemoveRange(head.Index+1,tail.Index-head.Index-1);
              #endregion

              // Запуск парсера
```

```
            BlockList list = parser.Parse(editor.Text);

            if (head.Equals(tail))
                  blocks.RemoveAt(head.Index);
            else
            {
                  blocks.RemoveAt(tail.Index);
                  blocks.RemoveAt(head.Index);
            }

            foreach(Block block in list)
            {
                  block.SetParent(ref blocks);
                  block.MoveRight(offset);
            }
            blocks.InsertRange(headIndex, list);

            foreach (Block block in blocks.GetRange(headIndex, list.Count))
            {
                  editor.SelectionStart = block.Begin;
                  editor.SelectionLength = block.End;
                  editor.SelectionColor =parser.language[block.Type].Color;
            }

            // Установка курсора в исходную позицию
            editor.SelectionStart = ss;
            editor.SelectionLength = sl;
            editor_CursorMoved();
            oldText = editor.Text;
            editor._Paint = true;

      }
      else
      {
            Block head = null, tail = null;

            #region поиск блоков вокруг области изменения
            foreach(Block block in blocks)
            {
               // Начало...
               if ((head == null) && (block.Begin <= selStart) &&
                        (block.End >= selStart))
                        head = block;
               // ...и конец
               if ((tail == null) && (block.Begin <= (selStart+selLength))
                        && (block.End >= (selStart+selLength)))
                        tail = block;
               // Выходим из поиска если всё нашли
               if ((tail != null) && (head != null))
               break;
            }
#endregion

      #region проверка совпадения блоков начала и конца
            if (head.Equals(tail))
            {
                  // Курсор находился на границе блока head
                  if (head.End == selStart + selLength)
                  {
```

```
                    if (head.Index < (blocks.Count-1))
                    {
                          tail = blocks[head.Index+1];
                    }
              }
      }
       else
    #region удаление блоков, находящихся между head и tail
              if ((tail.Index-head.Index-2) > 0 )
              blocks.RemoveRange(head.Index+1,tail.Index-head.Index-1);
    #endregion
#endregion

         int delta = editor.Text.Length - oldText.Length;
         int offset = head.Begin;

#region обновление конца списка блоков
         if (!head.Equals(tail))
               tail.MoveRight(delta);
         else
               head.End += delta;

         for (int i = tail.Index+1; i < blocks.Count; i++)
               blocks[i].MoveRight(delta);
#endregion

         // Запуск парсера
         BlockList list = parser.Parse(editor.Text.Substring(offset,
                    tail.End - offset));

         int headIndex = head.Index;

         if (head.Equals(tail))
               blocks.RemoveAt(head.Index);
         else
         {
               blocks.RemoveAt(tail.Index);
               blocks.RemoveAt(head.Index);
         }

         foreach(Block block in list)
         {
               block.SetParent(ref blocks);
               block.MoveRight(offset);
         }
         blocks.InsertRange(headIndex, list);

         foreach (Block block in blocks.GetRange(headIndex, list.Count))
         {
               editor.SelectionStart = block.Begin;
               editor.SelectionLength = block.End;
               editor.SelectionColor =parser.language[block.Type].Color;
   }
         // Установка курсора в исходную позицию
         editor.SelectionStart = ss;
         editor.SelectionLength = sl;
         editor_CursorMoved();
         oldText = editor.Text;
         editor._Paint = true;
```

```
                }
            }
        }
    }
```

П1.6. Group.cs

В файле *Group.cs* описан класс *Group*, который служит для хранения ключевых слов, относящихся к данной группе.

```
using System;
using System.Drawing;
using System.Collections;
using System.Collections.Specialized;

namespace MDIApp.Syntax
{
    /// <summary>
    ///
    /// </summary>
    public class Group
    {
        private GroupList parent = null;

        private string name;
        private System.Drawing.Color color;

        private Font font;
        private StringCollection strings;

        public int Index
        {
            get
            {
                return parent.IndexOf(this);
            }
        }

        public string Name
        {
            get
            {
                return name;
            }
        }

        public System.Drawing.Color Color
        {
            get
            {
                return color;
            }
            set
            {
                color = value;
            }
        }
```

```
        public Group()
            : this("default", null, System.Drawing.SystemColors.WindowText) {}

        public Group(ref GroupList parent)
            : this(ref parent, "default", null,
               System.Drawing.SystemColors.WindowText) {}

        public Group(ref GroupList parent,string name,StringCollection strings,Color color)
            : this(name, strings, color) {this.parent = parent;}

        public Group(string name, StringCollection strings, Color color)
        {
            this.name = name;

            if (strings != null)
            {
                if (this.strings == null)
                    this.strings = new StringCollection();
                else
                    this.strings.Clear();

                foreach (string word in strings)
                {
                    this.strings.Add(word);
                }
            }
            this.color = color;
        }
        public override string ToString()
        {
            string result = name + ":";

            if (strings != null)
                foreach (string word in strings)
                {
                    result += " \"" + word + "\";";
                }
            return result;
        }

        public bool Contains(string text)
        {
            if (strings == null) return false;
               return strings.Contains(text);
        }
    }
}
```

П1.7. GroupList.cs

Поскольку используется не одна группа ключевых слов, в программе описан класс *GroupList* в файле *GroupList.cs*, который является массивом объектов класса *Group.*

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

```
using System.Collections.Specialized;
using System.Drawing;

namespace MDIApp.Syntax
{
    /// <summary>
    ///
    /// </summary>
    public class GroupList : ArrayList
    {
        GroupList zis = null;

        public void SetZis(ref GroupList zis)
        {
            this.zis = zis;
        }

        public GroupList() : base()
        {
            Add(new Group(ref zis));
        }

        public Color ColorOf(string text)
        {
            return GroupTypeOf(text).Color;
        }
        public Group GroupTypeOf(string text)
        {
            foreach(Group Group in this)
            {
                if (Group.Contains(text)) return Group;
            }
            return (Group)this[0];
       }
        public int intTypeOf(string text)
        {
            foreach(Group Group in this)
            {
                if (Group.Contains(text)) return Group.Index;
            }
            return 0;
        }

        public new Group this[int index]
        {
            get
            {
                return (Group)base[index];
            }
            set
            {
                base[index] = value;
            }
        }

        public void LoadFromFile(string filename)
        {
            System.IO.StreamReader words = System.IO.File.OpenText(filename);
               int n = 0;
```

```
            string buf;
            string[] group = new string [8];
            StringCollection strings = new StringCollection();

            this.Add(new Group(ref zis, "strings", strings, Color.Black));
            this.Add(new Group(ref zis, "comments", strings, Color.Green));

            buf = words.ReadToEnd();
            n = 8;//buf.Split('#').Length;
            group = buf.Split('#');

            for (int i = 1; i <= n; i++)
            {
                  int k = group[i].Split(' ').Length;
                  string[] word = new string[k];
                  word = group[i].Split(' ');
                  int j;
                  for (j = 1; j < k; j++)
                  {
                        strings.Add(word[j]);
                  }
                  switch (word[0])
                  {
                        case "numbers":
                           this.Add(new Group(ref zis, word[0], strings, Color.Green));
                           break;
                        case "delimeters":
                           strings.Add(" ");
                           strings.Add("\n");
                           this.Add(new Group(ref zis, word[0], strings, Color.Black));
                           break;
                        case "keywords":
                           this.Add(new Group(ref zis, word[0], strings, Color.Blue));
                           break;
                        case "words":
                           this.Add(new Group(ref zis, word[0], strings, Color.Blue));
                           break;
                        case "operators":
                           this.Add(new Group(ref zis, word[0], strings, Color.Blue));
                           break;
                        case "variables":
                           this.Add(new Group(ref zis, word[0], strings, Color.Brown));
                           break;
                        case "constants":
                           this.Add(new Group(ref zis, word[0], strings, Color.DarkRed));
                           break;
                        case "functions":
                           this.Add(new Group(ref zis, word[0],strings, Color.Goldenrod));
                           break;
                  }
             strings.Clear();

            }
      }

   }
}
```

П1.8. Log.cs

Класс *Log*, который описан в файле *Log.cs*, используется для протоеолирования действий автомата.

```
using System;

namespace MDIApp.Syntax
{
    /// <summary>
    /// create log file
    /// </summary>
    public class Log
    {
          private System.IO.StreamWriter log;

          public Log()
          {
          }

          public void BeginLog()
          {
                this.log = System.IO.File.AppendText("log.html");
                this.log.WriteLine("<html>");
                this.log.WriteLine("<body>");
                this.log.Flush();
                this.log.Close();
          }

          public void EndLog()
          {
                this.log = System.IO.File.AppendText("log.html");
                this.log.WriteLine("</body>");
                 this.log.WriteLine("</html>");
                this.log.Flush();
                this.log.Close();
          }

          private string GetNameState(int num)
          {
                string str_cut = "";
                switch (num)
                {
                      case 0: str_cut = "\"Разделитель\"";                  break;
                      case 1: str_cut = "\"Возможен комментарий\"";         break;
                      case 2: str_cut = "\"Многострочный комментарий\"";    break;
                      case 3: str_cut = "\"Возможен конец многострочного комментария\"";
                              break;
                      case 4: str_cut = "\"Однострочный комментарий\"";     break;
                      case 5: str_cut = "\"Переменная\"";                   break;
                      case 6: str_cut = "\"Функция\"";                      break;
                      case 7: str_cut = "\"Цифра\"";                        break;
                      default:                                              break;
                }
                str_cut +=  " (y0 = " + (num + 1) + ")";
                return str_cut;
          }
          public void StartAuto()
          {
```

```
      this.log = System.IO.File.AppendText("log.html");
      this.log.WriteLine("<p><font face=Courier New Size=3
      color=f0429CA>Автомат начал работу в начальном состоянии.</font><br>");
      this.log.Flush();
      this.log.Close();
}

public void FinishAuto()
{
      this.log = System.IO.File.AppendText("log.html");
      this.log.WriteLine("<font face=Courier New Size=3
      color=f0429CA>Автомат закончил работу в конечном состоянии.</font></p>");
      this.log.Flush();
      this.log.Close();
}

public void PrintTime()
{
      this.log.WriteLine("[" + System.DateTime.Now.Hour + ":"  +
      System.DateTime.Now.Minute + ":" + System.DateTime.Now.Second + "]");
}

public void PrintLog(int cur, int old)
{
      this.log = System.IO.File.AppendText("log.html");
      string str_cur, str_old;
      str_cur = this.GetNameState(cur);
      str_old = this.GetNameState(old);
      this.log.WriteLine("<font face=Courier New size=2>");
      this.PrintTime();
      if (cur != old)
            this.log.WriteLine(" Автомат перешел из состояния<font color=0429CA>"
            + str_old + "</font> в состояние <font color=0429CA>" + str_cur +
            "</font>.<br><br>");
      else this.log.WriteLine(" Автомат остался в состоянии<font color=0429CA>" +
      str_cur + "</font>.<br><br>");
      this.log.WriteLine("</font>");
      this.log.Flush();
      this.log.Close();
}

public void PrintXLog(int x)
{
      string comment = "";
      switch (x)
      {
            case 1:  comment = "x1() - Текущий символ \"/\".<br>"; break;
            case 2:  comment = "x2() - Текущий символ \"*\".<br>"; break;
            case 3:  comment = "x3() - Текущий символ последний.<br>";
                     break;
            case 4:  comment = "x4() - Текущий символ \"\\n\".<br>"; break;
            case 5:  comment = "x5() - Текущий символ \"$\".<br>"; break;
            case 6:  comment = "x6() - Текущий символ разделитель.<br>";
                     break;
            case 7:  comment = "x7() - Слово входит в словарь ключевых слов.<br>";
                     break;
            case 8:  comment = "x8() - Слово входит в словарь функций.<br>";
                     break;
            case 9:  comment = "x9() - Слово без последнего символа входит в словарь
```

```
                        функций.<br>"; break;
                case 10: comment = "x10() - Текущий символ цифра.<br>"; break;
                case 11: comment = "x11() - Слово входит в словарь слов.<br>"; break;
                case 12: comment = "x12() - Слово без последнего символа входит в
                      словарь слов.<br>"; break;
                default: break;
            }
            this.log = System.IO.File.AppendText("log.html");
            this.log.WriteLine("<font face=Courier New size=2>");
            this.log.WriteLine("         " + comment);
            this.log.WriteLine("</font>");
            this.log.Flush();
            this.log.Close();
        }

        public void PrintZLog(int z)
        {
            string comment = "";
            switch (z)
            {
                case 1: comment = "z1() - Создание блока текста.<br>"; break;
                case 2: comment = "z2() - Сохранение начала нового блока.<br>";
                        break;
                case 3: comment = "z3() - Создание блока комментраиев.<br>";
                        break;
                case 4: comment = "z4() - Создание блока без текущего
                        символа.<br>"; break;
                case 5: comment = "z5() - Сохранение начала нового блока,
                        включая текущий символ и предыдущий.<br>"; break;
                case 6: comment = "z6() - Создание блока переменных.<br>";
                         break;
                case 7: comment = "z7() - Создание блока ключевых слов.<br>";
                        break;
                case 8: comment = "z8() - Сохранение начала нового блока,
                        включая текущий символ.<br>"; break;
                case 9: comment = "z9() - Создание блока функций.<br>"; break;
                case 10: comment = "z10() - Создание блока цифр.<br>"; break;
                case 11: comment = "z11() - Создание блока функций без
                          последнего символа.<br>"; break;
                case 12: comment = "z12() - Создание блока слов.<br>"; break;
                case 13: comment = "z13() - Создание блока слов без последнего
                символа.<br>"; break;
                default: break;
            }
            this.log = System.IO.File.AppendText("log.html");
            this.log.WriteLine("<font face=Courier New size=2>");
            this.log.WriteLine("         " +
            comment);
            this.log.WriteLine("</font>");
            this.log.Flush();
            this.log.Close();
        }
    }
}
```

П1.9. MainForm.cs

В файле *MainForm.cs* описан основной класс *MainForm* создания графического интерфейса. В нем реализовано создание основного окна приложения, создание дочерних окон и связь с объектом класса *Document*.

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;

namespace MDIApp.GUI
{
    public class MainForm : System.Windows.Forms.Form
    {
        /// <summary>
        ///    Required designer variable.
        /// </summary>
        private System.Windows.Forms.MainMenu mainMenu;
        protected internal System.Windows.Forms.StatusBar statusBar1;

        private int windowCount = 0 ;

        public MainForm()
        {

            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //Setup MDI stuff
            this.IsMdiContainer = true;
            this.MdiChildActivate += new EventHandler(this.MDIChildActivated);

            //Add File Menu
            MenuItem miFile = mainMenu.MenuItems.Add("&File");
            miFile.MergeOrder=0;
            miFile.MergeType = MenuMerge.MergeItems;

            MenuItem miAddDoc = new MenuItem("&Add Document", new
               EventHandler(this.FileAdd_Clicked), Shortcut.CtrlA);
            miAddDoc.MergeOrder=100;

            MenuItem miExit = new MenuItem("E&xit", new
               EventHandler(this.FileExit_Clicked), Shortcut.CtrlX);
            miExit.MergeOrder=110;

            miFile.MenuItems.Add(miAddDoc);
            miFile.MenuItems.Add("-");     // Gives us a seperator
            miFile.MenuItems.Add(miExit);

            //Add Window Menu
            MenuItem miWindow = mainMenu.MenuItems.Add("&Window");
            miWindow.MergeOrder = 10;
            miWindow.MenuItems.Add("&Cascade", new
               EventHandler(this.WindowCascade_Clicked));
```

```
    miWindow.MenuItems.Add("Tile &Horizontal", new
       EventHandler(this.WindowTileH_Clicked));
    miWindow.MenuItems.Add("Tile &Vertical", new
       EventHandler(this.WindowTileV_Clicked));
    miWindow.MdiList=true;//Adds the MDI Window List to the bottom of the menu

    AddDocument(); //Add an initial doc
}


//Add a document
private void AddDocument()
{
    windowCount++ ;
    Document doc = new Document("Document " + windowCount);
    doc.MdiParent = this;
    doc.Show();
}


//File->Add Menu item handler
protected void FileAdd_Clicked(object sender, System.EventArgs e)
{
    AddDocument() ;
}


//File->Exit Menu item handler
protected void FileExit_Clicked(object sender, System.EventArgs e)
{
    this.Close();
}


//One of the MDI Child windows has been activated
protected void MDIChildActivated(object sender, System.EventArgs e)
{
    if (null == this.ActiveMdiChild)
    {
        statusBar1.Text = "";
    }
    else
    {
        statusBar1.Text = this.ActiveMdiChild.Text;
    }
}

//Window->Cascade Menu item handler
protected void WindowCascade_Clicked(object sender, System.EventArgs e)
{
    this.LayoutMdi(MdiLayout.Cascade);
}


//Window->Tile Horizontally Menu item handler
protected void WindowTileH_Clicked(object sender, System.EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileHorizontal);
}
```

```
        //Window->Tile Vertically Menu item handler
        protected void WindowTileV_Clicked(object sender, System.EventArgs e)
        {
            this.LayoutMdi(MdiLayout.TileVertical);
        }

        /// <summary>
        ///    Clean up any resources being used.
        /// </summary>
        protected override void Dispose(bool disposing)
        {
            base.Dispose(disposing);
        }

        /// <summary>
        ///    Required method for Designer support - do not modify
        ///    the contents of this method with the code editor.
        /// </summary>
        protected void InitializeComponent()
        {
                this.mainMenu = new System.Windows.Forms.MainMenu();
                this.statusBar1 = new System.Windows.Forms.StatusBar();
                this.SuspendLayout();
                //
                // statusBar1
                //
                this.statusBar1.Location = new System.Drawing.Point(0, 329);
                this.statusBar1.Name = "statusBar1";
                this.statusBar1.Size = new System.Drawing.Size(616, 20);
                this.statusBar1.TabIndex = 1;
                //
                // MainForm
                //
                this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                this.BackColor = System.Drawing.SystemColors.AppWorkspace;
                this.ClientSize = new System.Drawing.Size(616, 349);
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                        this.statusBar1});
                this.Menu = this.mainMenu;
                this.Name = "MainForm";
                this.Text = "MDI Example";
                this.ResumeLayout(false);

          }

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        public static void Main(string[] args)
        {
            Application.Run(new MainForm());
        }

    }
}
```

П1.10. Parser.cs

Автомат *Parser* описан в файле *Parser.cs*. В нем реализована обработка строк, в результате которой текст меняет свой цвет.

```
using System;

namespace MDIApp.Syntax
{
    /// <summary>
    ///
    /// </summary>
    public class Parser
    {
        readonly public GroupList language;
         private int y0; // Состояние автомата
          private int i;  // Текущий обрабатываемый символ
           private string text;

           private int begin;
           private int end;

           System.IO.StreamWriter log_;
           BlockList result;
           public Log log;


        public Parser(ref GroupList language)
        {
            this.language = language;
               this.y0 = 0;
               this.i = 0;
               this.text = "";
               this.begin = -1;
               this.end = -1;
               this.result = new BlockList();
               this.log_ = System.IO.File.CreateText("mylog.txt");
               this.log = new Log();
               this.language.LoadFromFile("wordfile.txt");
        }


        public BlockList Parse(string txt)
        {
               int st_old = 0;

               #region
               this.result = new BlockList();

            if (txt == "")
            {
                result.Add(new Block());
                return result;
            }
            #endregion
```

```
log.StartAuto();
this.y0 = 1;
this.text = txt;
this.begin = 0;
this.end = 0;

for (int j = 0; j < this.text.Length; j++)
{

      this.i = j;
      log_.WriteLine("symbol " + (i + 1) + " : " + text.Substring(i,
                      1) + " state " + y0 + "");
      log_.Flush();
      switch(y0)
      {
        // Начальное состояние
        case 1:
         if (x1() && !x3())     {                              y0 = 2;}
         else if (x5() && !x3()) { z1(); z8();                 y0 = 6;}
         else if (x5() && x3())  { z8(); z6();                 y0 = 9;}
         else if (x10() && x3()) { z1(); z8(); z10();          y0 = 9;}
         else if (x10() && !x3()){ z1(); z8();                 y0 = 8;}
         else if (!x6() && !x3()){ z1(); z8();                 y0 = 7;}
         else if (!x6() && x3()) { z1(); z8();                 y0 = 9;}
         else if (x3())          { z1();                       y0 = 9;}
         break;

        // Подозрение на комментарий
        case 2:
         if (x2() && !x3())      { z4(); z5();                 y0 = 3;}
         else if (x2() && x3())  { z4(); z5(); z3();           y0 = 9;}
         else if (x1() && !x3()) { z4(); z5();                 y0 = 5;}
         else if (x1() && x3())  { z4(); z5(); z3();           y0 = 9;}
         else if (x3())          { z1();                       y0 = 9;}
         else                    {                              y0 = 1;}
         break;

        // Комментарий
        case 3:
         if (x2() && !x3())      {                              y0 = 4;}
         if (x3())               { z3();                       y0 = 9;}
         break;

        // Подозрение на конец комментария
        case 4:
          if (x3())              { z3();                       y0 = 9;}
          if (x1() && !x3())     { z3(); z2();                  y0 = 1;}
          else                   {                             y0 = 3;}
          break;



           // Однострочные комментарии
        case 5:
         if (x4() && !x3())       { z3(); z2();              y0 = 1;}
         if (x3())                { z3();                    y0 = 9;}
         break;
```

```
                // Переменная
                case 6:
                 if (x7() && (x3() || x6())) { z7(); z8();          y0 = 1;}
                 else if (x6() && !x3())     { z6(); z8();          y0 = 1;}
                 else if (x6() && x3())      { z6(); z8(); z3();    y0 = 9;}
                 else if (x3())              { z6();                y0 = 9;}
                 if (x1() && !x3())          {          z2();       y0 = 2;}
                 break;

                // Функция
                case 7:
                 if (x8() && x3())           { z9();  z2();         y0 = 1;}
                 else if (x9() && x6())      { z11(); z8(); z1();   y0 = 1;}
                 else if (x11() && x3())     { z12(); z2();         y0 = 1;}
                 else if (x12() && x6())     { z13(); z8(); z1();   y0 = 1;}
                 else if (x6() && x3())      { z1();  z2();         y0 = 9;}
                 else if (x6() && !x3())     { z1();  z2();         y0 = 1;}
                 else if (x3())              { z1();                y0 = 9;}
                 if (x1() && !x3())          {        z2();         y0 = 2;}
                 break;

                // Цифра
                case 8:
                 if (!x10() && !x3())        { z10(); z8();         y0 = 1;}
                 if (!x10() && x3())         { z10(); z8(); z1();   y0 = 1;}
                 else if (x10() && x3())     { z10();               y0 = 9;}
                 if (x1() && !x3())          {        z2();         y0 = 2;}
                 break;

                // Конечное состояние
                case 9:
                    break;

                default:
                    break;
            }
            log.PrintLog(y0, st_old);
            st_old = y0;
        }
        log.FinishAuto(y0);
        #region Возвращаем значение
    return result;
    #endregion
}


 #region x1: является ли текущий символ "/"?
 public bool x1()
 {
       if (this.text.Substring(this.i, 1) == "/") return true;
       else return false;
 }
 #endregion

 #region x2: является ли текущий символ "*"?
 public bool x2()
 {
       if (this.text.Substring(this.i, 1) == "*") return true;
       else return false;
```

```
}
#endregion

#region x3: является ли текущий символ последним?
public bool x3()
{
      if (this.i == this.text.Length - 1)  return true;
      else return false;
}
#endregion

#region x4: является ли текущий символ "\n"?
public bool x4()
{
      if (this.text.Substring(this.i, 1) == "\n") return true;
      else return false;
}
#endregion

#region x5: является ли текущий символ "$"?
public bool x5()
{
      if (this.text.Substring(this.i, 1) == "$") return true;
      else return false;
}
#endregion

#region x6: является ли текущий символ разделителем?
public bool x6()
{
      log_.WriteLine("whether it symbol is delimeters");
      log_.Flush();
      if (language[4].Contains(this.text.Substring(this.i, 1)))
      {
            log_.WriteLine("delim");
      }
      if (language[4].Contains(this.text.Substring(this.i, 1)))
            return true;
      else return false;
}
#endregion

#region x7: является ли текущее слово словом из словаря ключевых слов
public bool x7()
{
      if (this.begin >= this.i) return false;
      string txt = this.text.Substring(this.begin, this.i - this.begin+ 1);
      if (language[5].Contains(txt)) return true;
      else return false;
}
#endregion


#region x8: является ли текущее слово функцией?
public bool x8()
{
      log_.WriteLine("whether it symbol is function");
      log_.Flush();
      if (this.begin >= this.i) return false;
```

```
      string txt = this.text.Substring(this.begin, this.i -this.begin + 1);
      log_.WriteLine(txt);
      if (language[10].Contains(txt)) return true;
      else return false;
}
#endregion

#region x9: является ли текущее слово без последнего символа функцией?
public bool x9()
{
      if (this.begin >= this.i) return false;
      string txt = this.text.Substring(this.begin, this.i - this.begin);
      if (language[10].Contains(txt)) return true;
      else return false;
}
#endregion

#region x10: является ли текущий символ цифрой?
public bool x10()
{
      log_.WriteLine(this.text.Substring(this.i, 1));
      if (language[3].Contains(this.text.Substring(this.i, 1)))
      return true;
      else return false;
}
#endregion

#region x11: является ли текущее слово словом из словаря?
public bool x11()
{
      log_.WriteLine("whether it symbol is function");
      log_.Flush();
      if (this.begin >= this.i) return false;
      string txt = this.text.Substring(this.begin, this.i- this.begin + 1);
      log_.WriteLine(txt);
      if (language[6].Contains(txt)) return true;
      else return false;
}
#endregion

#region x12: является ли текущее слово без последнего символа словом из
словаря?
public bool x12()
{
      if (this.begin >= this.i) return false;
      string txt = this.text.Substring(this.begin, this.i - this.begin);
      if (language[6].Contains(txt)) return true;
      else return false;
}
#endregion


#region z1: создание блока текста
public void z1()
{
      log_.WriteLine("Create simple text block");
      log_.Flush();

      this.end = this.i + 1;
```

```
      if (this.begin != this.end)
            this.result.Add(new Block(this.begin, this.end, 1));
}
#endregion


#region z2: сохранение начала нового блока
public void z2()
{
      this.begin = this.i + 1;
}
#endregion


#region z3: создание блока комментариев
public void z3()
{
      this.end = this.i + 1;
      log_.WriteLine("Create comment block " + begin + " " + end + "");
      log_.Flush();
      result.Add(new Block(this.begin, this.end, 2));
}
#endregion


#region z4: создание блока без текущего символа
public void z4()
{
      this.end = this.i - 1;
      log_.WriteLine("Create text block " + begin + " " + end + "");
      log_.Flush();
      if (this.begin != this.end)
            result.Add(new Block(this.begin, this.end, 1));
}
#endregion



#region z5: сохранение начала нового блока, включая текущий символ  и
предыдущий
public void z5()
{
      this.begin = this.i - 1;
}
#endregion


#region z6: создание блока переменных
public void z6()
{
      this.end = this.i + 1;
      log_.WriteLine("Create variable block" + begin + " " + end + "");
      log_.Flush();
      result.Add(new Block(this.begin, this.end, 8));
}
#endregion
```

```
#region z7: создание блока ключевых слов
public void z7()
{
      this.end = this.i + 1;
      result.Add(new Block(this.begin, this.end, 5));
}
#endregion


#region z8: сохранение начала нового блока, включая текущий символ
public void z8()
{
      this.begin = this.i;
}
#endregion


#region z9: создание блока функций
public void z9()
{
      this.end = this.i + 1;
      result.Add(new Block(this.begin, this.end, 10));
}
#endregion


#region z10: создание блока цифр
public void z10()
{
      log_.WriteLine("Number");
      this.end = this.i + 1;
      result.Add(new Block(this.begin, this.end, 3));
}
#endregion


#region z11: создание блока функций без последнего символа
public void z11()
{
      this.end = this.i;
      result.Add(new Block(this.begin, this.end, 10));
}
#endregion


#region z12: создание блока слов
public void z12()
{
      this.end = this.i + 1;
      result.Add(new Block(this.begin, this.end, 6));
}
#endregion


#region z13: создание блока сов без последнего символа
public void z13()
{
      this.end = this.i;
      result.Add(new Block(this.begin, this.end, 6));
```

```
            }
            #endregion

    }
}
```

Приложение 2. Протокол работы программы

Автомат начал работу в начальном состоянии.

x1() - Текущий символ "/".
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Возможен комментарий" (y0 = 2).

x2() - Текущий символ "*".
z4() - Создание блока без текущего символа.
z5() - Сохранение начала нового блока, включая текущий символ и предыдущий.
[3:8:26] Автомат перешел из состояния "Возможен комментарий" (y0 = 2) в состояние "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

x2() - Текущий символ "*".
[3:8:26] Автомат перешел из состояния "Многострочный комментарий" (y0 = 3) в состояние "Возможен конец многострочного комментария" (y0 = 4).

x1() - Текущий символ "/".
z3() - Создание блока комментариев.
z2() - Сохранение начала нового блока.
[3:8:26] Автомат перешел из состояния "Возможен конец многострочного комментария" (y0 = 4)

в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x11() - Слово входит в словарь слов.
[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x12() - Слово без последнего символа входит в словарь слов.
x6() - Текущий символ разделитель.
z13() - Создание блока слов без последнго символа.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель" (y0 = 1).

x5() - Текущий символ "$".
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

x6() - Текущий символ разделитель.
z6() - Создание блока переменных.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Переменная" (y0 = 6) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция" (y0 = 7).

x11() - Слово входит в словарь слов.
[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x12() - Слово без последнего символа входит в словарь слов.
x6() - Текущий символ разделитель.
z13() - Создание блока слов без последнего символа.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x8() - Слово входит в словарь функций.
[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x9() - Слово без последнего символа входит в словарь функций.
x6() - Текущий символ разделитель.
z11() - Создание блока функций без последнего символа.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель" (y0 = 1).

x5() - Текущий символ "$".
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

x6() - Текущий символ разделитель.
z6() - Создание блока переменных.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Переменная" (y0 = 6) в состояние "Разделитель" (y0 = 1).

```
         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         z1() - Создание блока текста.
         z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция"
(y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

         x11() - Слово входит в словарь слов.
[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

         x12() - Слово без последнего символа входит в словарь слов.
         x6() - Текущий символ разделитель.
         z13() - Создание блока слов без последнего символа.
         z8() - Сохранение начала нового блока, включая текущий символ.
         z1() - Создание блока текста.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель"
(y0 = 1).

         x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

         z1() - Создание блока текста.
         z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция"
(y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

         x6() - Текущий символ разделитель.
```

```
          x6() - Текущий символ разделитель.
          z1() - Создание блока текста.
          z2() - Сохранение начала нового блока.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель"
(y0 = 1).

          z1() - Создание блока текста.
          z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция"
(y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

          x6() - Текущий символ разделитель.
          x6() - Текущий символ разделитель.
          z1() - Создание блока текста.
          z2() - Сохранение начала нового блока.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель"
(y0 = 1).

          z1() - Создание блока текста.
          z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция"
(y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в  состоянии "Функция" (y0 = 7).

          x6() - Текущий символ разделитель.
          x6() - Текущий символ разделитель.
          z1() - Создание блока текста.
          z2() - Сохранение начала нового блока.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель"
(y0 = 1).

          x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

          x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

          x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

          x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

          x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).

          x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в  состоянии "Разделитель" (y0 = 1).
```

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x11() - Слово входит в словарь слов.
[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x12() - Слово без последнего символа входит в словарь слов.
x6() - Текущий символ разделитель.
z13() - Создание блока слов без последнего символа.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x11() - Слово входит в словарь слов.
[3:8:26] Автомат остался в состоянии "Функция" (y0 = 7).

x12() - Слово без последнего символа входит в словарь слов.
x6() - Текущий символ разделитель.
z13() - Создание блока слов без последнего символа.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:26] Автомат перешел из состояния "Функция" (y0 = 7) в состояние "Разделитель" (y0 = 1).

x5() - Текущий символ "$".
z1() - Создание блока текста.

z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:26] Автомат остался в состоянии "Переменная" (y0 = 6).

x6() - Текущий символ разделитель.
z6() - Создание блока переменных.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:26] Автомат перешел из состояния "Переменная" (y0 = 6) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:26] Автомат остался в состоянии "Разделитель" (y0 = 1).

x1() - Текущий символ "/".
[3:8:26] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Возможен комментарий" (y0 = 2).

x2() - Текущий символ "*".
z4() - Создание блока без текущего символа.
z5() - Сохранение начала нового блока, включая текущий символ и предыдущий.
[3:8:26] Автомат перешел из состояния "Возможен комментарий" (y0 = 2) в состояние "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:26] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

[3:8:27] Автомат остался в состоянии "Многострочный комментарий" (y0 = 3).

x2() - Текущий символ "*".
[3:8:27] Автомат перешел из состояния "Многострочный комментарий" (y0 = 3) в состояние "Возможен конец многострочного комментария" (y0 = 4).

x1() - Текущий символ "/".
z3() - Создание блока комментраиев.
z2() - Сохранение начала нового блока.

[3:8:27] Автомат перешел из состояния "Возможен конец многострочного комментария" (y0 = 4) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x5() - Текущий символ "$".
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Переменная" (y0 = 6).

[3:8:27] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:27] Автомат остался в состоянии "Переменная" (y0 = 6).

x6() - Текущий символ разделитель.
z6() - Создание блока переменных.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Переменная" (y0 = 6) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x10() - Текущий символ цифра.
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Цифра" (y0 = 8).

z10() - Создание блока цифр.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:27] Автомат перешел из состояния "Цифра" (y0 = 8) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x5() - Текущий символ "$".
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Переменная" (y0 = 6).

[3:8:27] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:27] Автомат остался в состоянии "Переменная" (y0 = 6).

x6() - Текущий символ разделитель.
z6() - Создание блока переменных.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Переменная" (y0 = 6) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x10() - Текущий символ цифра.
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Цифра" (y0 = 8).

z10() - Создание блока цифр.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:27] Автомат перешел из состояния "Цифра" (y0 = 8) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x5() - Текущий символ "$".
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Переменная" (y0 = 6).

[3:8:27] Автомат остался в состоянии "Переменная" (y0 = 6).

[3:8:27] Автомат остался в состоянии "Переменная" (y0 = 6).

x6() - Текущий символ разделитель.
z6() - Создание блока переменных.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Переменная" (y0 = 6) в состояние "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x6() - Текущий символ разделитель.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

x10() - Текущий символ цифра.
z1() - Создание блока текста.
z8() - Сохранение начала нового блока, включая текущий символ.
[3:8:27] Автомат перешел из состояния "Разделитель" (y0 = 1) в состояние "Цифра" (y0 = 8).

z10() - Создание блока цифр.
z8() - Сохранение начала нового блока, включая текущий символ.
z1() - Создание блока текста.
[3:8:27] Автомат перешел из состояния "Цифра" (y0 = 8) в состояние "Разделитель" (y0 = 1).

x3() - Текущий символ последний.
x6() - Текущий символ разделитель.
z1() - Создание блока текста.
[3:8:27] Автомат остался в состоянии "Разделитель" (y0 = 1).

Автомат закончил работу в конечном состоянии.