

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО

Доцент факультета компьютерных
наук, заместитель декана по
учебно-методической работе,
канд. социол. наук

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Программная инженерия» профессор
департамента программной
инженерии, канд. техн. наук

« ____ » _____ 2020 г.

« ____ » _____ 2020 г.

**ПРИЛОЖЕНИЕ ДЛЯ ВИЗУАЛИЗАЦИИ МЕТОДА РЕКУРСИВНОГО
СПУСКА**

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.13-01 ТП 01-1-ЛУ

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.13-01 ТП 01-1				

Исполнитель:
студент группы БПИ 199

« ____ » _____ 2020 г.

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.13-01 ТП 01-1				

**ПРИЛОЖЕНИЕ ДЛЯ ВИЗУАЛИЗАЦИИ МЕТОДА РЕКУРСИВНОГО
СПУСКА**

Текст программы

RU.17701729.04.13-01 ТП 01-1

Листов 27

Содержание

1	Текст программы	3
1.1	ParserApp	3
1.1.1	ParserApp.csproj	3
1.1.2	App.xaml	3
1.1.3	App.xaml.cs	4
1.1.4	AssemblyInfo.cs	4
1.1.5	MainWindow.xaml	4
1.1.6	MainWindow.xaml.cs	8
1.2	ParserLib	19
1.2.1	ParserLib.csproj	19
1.2.2	HistoryEntry.cs	20
1.2.3	HistoryToken.cs	21
1.2.4	Parser.cs	22
1.2.5	ParserHistory.cs	23
1.2.6	ParserTreeToken.cs	25
1.2.7	RtfBuilder.cs	26

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1 Текст программы

1.1 ParserApp

1.1.1 ParserApp.csproj

```
<Project Sdk="Microsoft.NET.Sdk.WindowsDesktop">

  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net45</TargetFramework>
    <UseWPF>true</UseWPF>
  </PropertyGroup>

  <ItemGroup>
    <Content Include="tutorials\*.*">
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </Content>
  </ItemGroup>

  <ItemGroup>
    <PackageReference Include="Newtonsoft.Json" Version="12.0.3" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\ParserLib\ParserLib.csproj" />
  </ItemGroup>

  <Target Name="PreBuild" BeforeTargets="PreBuildEvent">
    <Exec Command="copy /Y ..\pre-commit ..\.git\hooks" />
  </Target>

</Project>
```

1.1.2 App.xaml

```
<Application x:Class="ParserApp.App"

  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="clr-namespace:ParserApp"
  StartupUri="MainWindow.xaml">
  <Application.Resources>

  </Application.Resources>
</Application>
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1.1.3 App.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;

namespace ParserApp {
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application {
    }
}
```

1.1.4 AssemblyInfo.cs

```
using System.Windows;

[assembly: ThemeInfo(
    ResourceDictionaryLocation.None, //where theme specific resource
    dictionaries are located
                                     //(used if a resource is not found in
    the page,
    // or application resource
    dictionaries)
    ResourceDictionaryLocation.SourceAssembly //where the generic resource
    dictionary is located
                                     //(used if a resource is not
    found in the page,
    // app, or any theme specific
    resource dictionaries)
)]
```

1.1.5 MainWindow.xaml

```
<Window x:Class="ParserApp.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:ParserApp"
    mc:Ignorable="d"
    Title="Визуализация парсеров" Height="450" Width="800"
    AllowDrop="true">
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

<Window.Resources>
    <RoutedUICommand x:Key="TogglePauseCommand"/>
    <RoutedUICommand x:Key="NextFrameCommand"/>
    <RoutedUICommand x:Key="PrevFrameCommand"/>
    <RoutedUICommand x:Key="FirstFrameCommand"/>
    <RoutedUICommand x:Key="LastFrameCommand"/>
    <RoutedUICommand x:Key="ReverseCommand"/>
    <RoutedUICommand x:Key="SaveCommand"/>
    <RoutedUICommand x:Key="LoadCommand"/>
    <RoutedUICommand x:Key="NextTutorialCommand"/>
    <RoutedUICommand x:Key="PrevTutorialCommand"/>

    <Style x:Key="commonButtonStyle" TargetType="{x:Type Control}">
        <Setter Property="Margin" Value="7,0,0,0"/>
        <Setter Property="FontFamily" Value="Segoe UI Symbol"/>
        <Setter Property="Height" Value="20"/>
        <Setter Property="Width" Value="20"/>
    </Style>
</Window.Resources>

<Window.CommandBindings>
    <CommandBinding Command="{StaticResource TogglePauseCommand}"
        Executed="TogglePauseEvent"/>
    <CommandBinding Command="{StaticResource NextFrameCommand}"
        Executed="NextFrameEvent"/>
    <CommandBinding Command="{StaticResource PrevFrameCommand}"
        Executed="PrevFrameEvent"/>
    <CommandBinding Command="{StaticResource FirstFrameCommand}"
        Executed="FirstFrameEvent"/>
    <CommandBinding Command="{StaticResource LastFrameCommand}"
        Executed="LastFrameEvent"/>
    <CommandBinding Command="{StaticResource ReverseCommand}"
        Executed="ReverseEvent"/>
    <CommandBinding Command="{StaticResource SaveCommand}"
        Executed="SaveEvent"/>
    <CommandBinding Command="{StaticResource LoadCommand}"
        Executed="LoadEvent"/>
    <CommandBinding Command="{StaticResource NextTutorialCommand}"
        Executed="NextTutorialEvent"/>
    <CommandBinding Command="{StaticResource PrevTutorialCommand}"
        Executed="PrevTutorialEvent"/>
</Window.CommandBindings>

<Window.InputBindings>
    <KeyBinding Key="Space" Command="{StaticResource
        TogglePauseCommand}"/>
    <KeyBinding Key="N" Command="{StaticResource NextFrameCommand}"/>
    <KeyBinding Key="P" Command="{StaticResource PrevFrameCommand}"/>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТИ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

<KeyBinding Key="Home" Command="{StaticResource
FirstFrameCommand}"/>
<KeyBinding Key="End" Command="{StaticResource LastFrameCommand}"/>
<KeyBinding Key="R" Command="{StaticResource ReverseCommand}"/>
<KeyBinding Key="S" Command="{StaticResource SaveCommand}"/>
<KeyBinding Key="S" Modifiers="Ctrl" Command="{StaticResource
SaveCommand}"/>
<KeyBinding Key="O" Command="{StaticResource LoadCommand}"/>
<KeyBinding Key="O" Modifiers="Ctrl" Command="{StaticResource
LoadCommand}"/>

<KeyBinding Key="L" Command="{StaticResource NextFrameCommand}"/>
<KeyBinding Key="K" Command="{StaticResource TogglePauseCommand}"/>
<KeyBinding Key="J" Command="{StaticResource PrevFrameCommand}"/>

<!-- незя тк у нас textbox -->
<!-- <KeyBinding Key="OemPeriod" Command="{StaticResource
NextFrameCommand}"/> -->
<!-- <KeyBinding Key="OemComma" Command="{StaticResource
PrevFrameCommand}"/> -->
</Window.InputBindings>

<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="200"/>
    <RowDefinition />
    <RowDefinition Height="30"/>
    <RowDefinition Height="20"/>
    <RowDefinition Height="30"/> <!-- 20+30 = 50 -->
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition/>
    <ColumnDefinition Width="400"/>
  </Grid.ColumnDefinitions>

  <RichTextBox x:Name="richTextBox" Grid.Column="1" IsReadOnly="True"
Panel.ZIndex="1"/>
  <RichTextBox x:Name="tutorialBox" Grid.Column="1" IsReadOnly="True"
Panel.ZIndex="1" Grid.Row="1" Grid.RowSpan="3"
VerticalScrollBarVisibility="Auto" BorderBrush="#FFABADB3"/>

  <Border Grid.Column="1" Grid.Row="4" Panel.ZIndex="2"
BorderBrush="#ABADB3" BorderThickness="1,0,0,0">
    <DockPanel HorizontalAlignment="Left">
      <DockPanel.Resources>
        <Style TargetType="{x:Type Button}"
BasedOn="{StaticResource commonButtonStyle}"/>
        <Style TargetType="{x:Type ToggleButton}"
BasedOn="{StaticResource commonButtonStyle}"/>

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТИ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

</DockPanel.Resources>

<Button x:Name="prevTutorialButton" ToolTip="Предыдущий
туториал" Content="&#x25c0;" Command="{StaticResource
PrevTutorialCommand}"/>
<Button x:Name="nextTutorialButton" ToolTip="Следующий
туториал" Content="&#x25b6;" Command="{StaticResource
NextTutorialCommand}"/>
<ToggleButton x:Name="trimTreeButton" ToolTip="Подровнять
дерево" Content="&#x0001f334;"/>
<ToggleButton x:Name="oriTreeButton" ToolTip="Поменять
ориентацию дерева" Content="&#x0001f332;"
RenderTransformOrigin="0.5,0.5">
    <ToggleButton.RenderTransform>
        <RotateTransform Angle="-180"/>
    </ToggleButton.RenderTransform>
</ToggleButton>
<ToggleButton x:Name="gravTreeButton" ToolTip="Поменять
гравитацию дерева" Content="&#x0001f30c;"/>
</DockPanel>
</Border>

<Canvas x:Name="canvas" Grid.RowSpan="2">
    <TextBox x:Name="inputBox" Opacity="0" Panel.ZIndex="1"/>
</Canvas>

<Slider x:Name="mainSlider" Margin="42,0,10,0"
VerticalAlignment="Center" Grid.Row="3" IsSnapToTickEnabled="True"
TickFrequency="1" Minimum="0" Maximum="1" Grid.RowSpan="2"/>
<Button
    x:Name="playButton" ToolTip="Воспроизведение" Content="&#x25b6;"
    Command="{StaticResource TogglePauseCommand}" Margin="7,0,0,0"
    HorizontalAlignment="Left" Grid.Row="3" VerticalAlignment="Center"
    Height="30" Width="30" FontFamily="Segoe UI Symbol"
    Grid.RowSpan="2"/>

<DockPanel Grid.Row="2" VerticalAlignment="Center">
    <DockPanel.Resources>
        <Style TargetType="{x:Type ToggleButton}"
            BasedOn="{StaticResource commonButtonStyle}"/>
        <Style TargetType="{x:Type Button}" BasedOn="{StaticResource
            commonButtonStyle}"/>
    </DockPanel.Resources>

    <!-- &#x0001f4be;&#x0001f5ab;
    &#x0001f331;&#x0001f332;&#x0001f333;&#x0001f334;
    &#x0001f335;&#x0001f33e;&#x0001f33f;&#x2618;&#x0001f340;
    &#x0001f4c2; -->

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```

<Button x:Name="loadButton" ToolTip="Загрузить сохранение"
Content="#x0001f4c2;" Command="{StaticResource LoadCommand}"/>
<Button x:Name="saveButton" ToolTip="Сохранить"
Content="#x0001f4be;" Command="{StaticResource SaveCommand}"/>
<ToggleButton x:Name="reverseButton" ToolTip="Прокрутка назад"
Content="#x25c0;" Command="{StaticResource ReverseCommand}"/>
<Button x:Name="firstButton" ToolTip="Первый кадр"
Content="#x23ee;" Command="{StaticResource
FirstFrameCommand}"/>
<Button x:Name="prevButton" ToolTip="Предыдущий кадр"
Content="#x23ea;" Command="{StaticResource PrevFrameCommand}"/>
<Button x:Name="nextButton" ToolTip="Следующий кадр"
Content="#x23e9;" Command="{StaticResource NextFrameCommand}"/>
<Button x:Name="lastButton" ToolTip="Последний кадр"
Content="#x23ed;" Command="{StaticResource LastFrameCommand}"/>

<TextBlock Text="Скорость:" ToolTip="(кадров в секунду)"
Margin="7,0,0,0" VerticalAlignment="Center"/>
<TextBox x:Name="speedBox" Width="40" Margin="7,0,0,0"/>
<Slider x:Name="speedSlider" Maximum="60" Margin="7,0,10,0"
VerticalAlignment="Center"/>
</DockPanel>
</Grid>
</Window>

```

1.1.6 MainWindow.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Markup;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;
using Microsoft.Win32;
using Newtonsoft.Json;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

using Newtonsoft.Json.Linq;
using ParserLib;

namespace ParserApp {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    [JsonObject(MemberSerialization.OptIn)]
    public partial class MainWindow : Window {
        #region json properties
        [JsonProperty]
        private int historyIndex = 0;
        [JsonProperty]
        private bool isPaused = true;
        [JsonProperty]
        private bool isReversed = false;

        [JsonProperty]
        private double speed { get => speedSlider.Value; set =>
            SetSpeed(value); }
        [JsonProperty]
        private string inputString { get => theHistory.InputString; set =>
            RunParser(value); }
        #endregion

        private int tutorialIndex;

        private DispatcherTimer mainTimer = new DispatcherTimer();
        private DispatcherTimer speedBoxTimer = new DispatcherTimer();

        private Parser parser = new Parser("simple");
        private ParserHistory theHistory;
        private Dictionary<string, Brush> colorDict;

        /// <summary>
        /// настройки интерфейсов всяких
        /// </summary>
        private readonly Brush[] colors = {
            Brushes.Green,
            Brushes.Red,
            Brushes.LightBlue,
            Brushes.Gray,
        };

        private readonly FontFamily font = new FontFamily("Consolas");
        const int charWidth = 20;
        const int textStart = 15;
        const string autosavePath = "./autosave.json"; // todo

        public MainWindow() {

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
InitializeComponent();
InitializeEvents();

speed = 4;
SelectTutorialPage(0);
}

private void InitializeEvents() {
    this.Drop += MainWindow_Drop;
    mainSlider.ValueChanged += MainSlider_ValueChanged;
    speedSlider.ValueChanged += SpeedSlider_ValueChanged;

    speedBoxTimer.Interval = TimeSpan.FromSeconds(2);
    speedBoxTimer.Tick += SpeedBox_ValueChanged;

    speedBox.LostFocus += SpeedBox_ValueChanged;
    speedBox.TextChanged += (o, e) => {
        speedBoxTimer.Stop();
        speedBoxTimer.Start();

        double r = speed;
        double.TryParse(
            speedBox.Text.Replace(',', '.', '.'),
            NumberStyles.Any,
            CultureInfo.InvariantCulture,
            out r
        );
        if (r < 0) r = 0;
        if (r > 60) r = 60;
        if (r.ToString("0.###", CultureInfo.InvariantCulture) ==
            speedBox.Text) {
            SpeedBox_ValueChanged(o, e);
        }
    };
    speedBox.KeyDown += (o, e) => {
        if (e.Key == Key.Return || e.Key == Key.Escape) {
            SpeedBox_ValueChanged(o, e);
        }
    };

    inputBox.LostKeyboardFocus += InputBox_ValueChanged;
    inputBox.KeyDown += (o, e) => {
        if (e.Key == Key.Return || e.Key == Key.Escape) {
            InputBox_ValueChanged(o, e);
            Keyboard.ClearFocus();
        }
    };
    inputBox.GotKeyboardFocus += (o, e) => { inputBox.Opacity = 1;
};
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТИ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
mainTimer.Tick += (ob, ea) => { if (!isPaused) NextFrame(); };
mainTimer.Start();

RoutedEventHandler t = (ob, ea) => DisplayHistoryEntry();
trimTreeButton.Click += t;
oriTreeButton.Click += t;
gravTreeButton.Click += t;
}

private void SelectTutorialPage(int i) {
    try {
        if (!File.Exists($"./tutorials/{i}.rtf")) return;
        using (var fs = File.OpenRead($"./tutorials/{i}.rtf")) {
            tutorialBox.SelectAll();
            tutorialBox.Selection.Load(fs, DataFormats.Rtf);
        }
        if (File.Exists($"./tutorials/{i}.json"))
            Load($"./tutorials/{i}.json");

        tutorialIndex = i;
        prevTutorialButton.IsEnabled = tutorialIndex != 0;
        nextTutorialButton.IsEnabled = File.Exists($"./tutorials/{i
            + 1}.rtf");
    } catch (Exception) {
        MessageBox.Show(
            "Что-то пошло не так, и у нас не получилось загрузить
            tutorial",
            "Тьюториал",
            MessageBoxButton.OK,
            MessageBoxImage.Error
        );
    }
}

// костыль потомучто richTextBox.Rtf = str; не работает на wpf
// то есть "он меняет содержимое richTextBox-а на то что написано в
// строке document"
private void SetRtf(string document) {
    var documentBytes = Encoding.UTF8.GetBytes(document);
    using (var reader = new MemoryStream(documentBytes)) {
        reader.Position = 0;
        richTextBox.SelectAll();
        richTextBox.Selection.Load(reader, DataFormats.Rtf);
    }
}

private void CanvasWrite(string text) {
    // типа такая конфигурация
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
const int fontSize = 26;
const int top = 10;
int pos = textStart;

// удаляем весь старый текст (если он есть)
foreach (var tb in canvas.Children.OfType<TextBlock>().ToList())
{
    canvas.Children.Remove(tb);
}

foreach (var chr in text) {
    var txt = new TextBlock();
    txt.FontSize = fontSize;
    txt.Text = chr.ToString();
    txt.FontFamily = font;
    Canvas.SetTop(txt, top);
    Canvas.SetLeft(txt, pos);
    canvas.Children.Add(txt);
    pos += charWidth;
}

Canvas.SetTop(inputBox, top);
Canvas.SetLeft(inputBox, textStart);
inputBox.Width = pos - textStart;
inputBox.Height = fontSize;
inputBox.FontSize = fontSize - 8;
}

private void CanvasLegend(bool drawText = false) {
    // удаляем все старые кружочки
    foreach (var el in canvas.Children.OfType<Ellipse>().ToList()) {
        canvas.Children.Remove(el);
    }

    // удаляем все старые подписи
    foreach (var tb in canvas.Children.OfType<TextBlock>().ToList())
    {
        if (tb.FontSize == 12) canvas.Children.Remove(tb);
    }

    var pos = 5;
    foreach (var pair in colorDict) {
        var el = new Ellipse();
        el.Width = el.Height = 12;
        el.Fill = pair.Value;
        el.ToolTip = pair.Key;
        Canvas.SetTop(el, pos);
        Canvas.SetRight(el, 5);
        canvas.Children.Add(el);
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        if (drawText) {
            var txt = new TextBlock();
            txt.FontSize = 12;
            txt.Text = pair.Key;
            txt.FontFamily = font;
            Canvas.SetTop(txt, pos);
            Canvas.SetRight(txt, 20);
            canvas.Children.Add(txt);
        }

        pos += 19;
    }

}

private void CanvasDrawRect(HistoryToken tok, int pos) {
    var rect = new Border();
    rect.CornerRadius = new CornerRadius(5, 5, 5, 5);
    var end = tok.EndPos;
    if (end == -1) {
        end = pos;
        rect.CornerRadius = new CornerRadius(5, 0, 0, 5);
    }

    rect.Background = colorDict[tok.Name];
    rect.Height = 10;
    rect.Width = (end - tok.StartPos) * charWidth;
    rect.ToolTip = tok.Name;

    if (tok.Trimmable) rect.Opacity = .5;

    Canvas.SetTop(rect, 50 + tok.DisplayLevel * 15);
    Canvas.SetLeft(rect, textStart + tok.StartPos * charWidth);
    canvas.Children.Add(rect);
}

private void DisplayHistoryEntry(HistoryEntry entry = null) {
    if (entry == null) {
        entry = theHistory[historyIndex];
        mainSlider.ToolTip = historyIndex.ToString();
        mainSlider.Value = historyIndex;
    }
    entry.SetSettings(
        (bool)trimTreeButton.IsChecked,
        (bool)oriTreeButton.IsChecked,
        (bool)gravTreeButton.IsChecked
    );
    SetRtf(entry.RtfGrammar);
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
// удаляем строное дерево
foreach (var tb in canvas.Children.OfType<Border>().ToList()) {
    canvas.Children.Remove(tb);
}

foreach (var tok in entry) {
    CanvasDrawRect(tok, entry.CursorPos);
}
}

private void RunParser(string input) {
    double historyProgress = 0;
    if (theHistory != null) {
        historyProgress = historyIndex / (double)(theHistory.Count()
        - 1);
    }

    inputBox.Text = input;
    theHistory = parser.Run(input);
    var i = 0;
    colorDict = theHistory.RuleNames.ToDictionary(e => e, e =>
    colors[i++]);

    historyIndex = (int)(historyProgress * (theHistory.Count() -
    1));

    mainSlider.Maximum = theHistory.Count() - 1;
    CanvasWrite(input);
    CanvasLegend();
    DisplayHistoryEntry();
}

private void SetSpeed(double newValue) {
    if (newValue < 0) newValue = 0;
    if (newValue > 60) newValue = 60;

    speedBox.Text = newValue.ToString("0.###",
    CultureInfo.InvariantCulture);
    speedSlider.Value = newValue;

    var t = 1 / newValue;
    var max = int.MaxValue / 1e7;
    if (t > max || t <= 0) t = max;
    mainTimer.Interval = TimeSpan.FromSeconds(t);
}

private void Load(string path = autosavePath) {
    try {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

var str = File.ReadAllText(path);
JsonConvert.PopulateObject(str, this);
reverseButton.IsChecked = isReversed;
playButton.Content = isPaused ? "\u25b6" : "\u23f8";
playButton.ToolTip = isPaused ? "Воспроизведение" : "Пауза";
} catch (Exception) {
    if (path == autosavePath) return;
    MessageBox.Show(
        "Что-то пошло не так, и у нас не получилось загрузить
        файл",
        "Загрузка",
        MessageBoxButton.OK,
        MessageBoxImage.Error
    );
}
}

private void Save(string path = autosavePath) {
    try {
        if (path.EndsWith(".png")) {
            CanvasLegend(true);
            ExportToPng(path, canvas);
            CanvasLegend(false);
        } else if (path.EndsWith(".svg")) {
            throw new NotImplementedException();
        } else if (path.EndsWith(".xaml")) {
            File.WriteAllText(path, XamlWriter.Save(canvas));
        } else {
            var str = JsonConvert.SerializeObject(this,
                Formatting.Indented);
            File.WriteAllText(path, str);
        }
    } catch (Exception) {
        if (path == autosavePath) return;
        MessageBox.Show(
            "Что-то пошло не так, и у нас не получилось сохранить
            файл",
            "Сохранение",
            MessageBoxButton.OK,
            MessageBoxImage.Error
        );
    }
}

#region events
private void MainSlider_ValueChanged(object o, EventArgs e) {
    historyIndex = (int)mainSlider.Value;
    DisplayHistoryEntry();
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
private void SpeedSlider_ValueChanged(object o, EventArgs e) {
    SetSpeed(speedSlider.Value);
}

private void SpeedBox_ValueChanged(object o, EventArgs e) {
    speedBoxTimer.Stop();
    // есть ли какойто менее костыльный способ парсить оба стиля
    дабла?
    double r = speed;
    double.TryParse(
        speedBox.Text.Replace(',', '.', ' '),
        NumberStyles.Any,
        CultureInfo.InvariantCulture,
        out r
    );
    SetSpeed(r);
}

private void InputBox_ValueChanged(object o, EventArgs e) {
    inputBox.Opacity = 0;
    inputString = inputBox.Text;
}

private void MainWindow_Drop(object o, DragEventArgs e) {
    if (!e.Data.GetDataPresent(DataFormats.FileDrop)) return;
    string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);
    Load(files[0]);
}

private void NextFrame() {
    historyIndex += isReversed ? -1 : 1;

    if (historyIndex >= theHistory.Count() || historyIndex < 0) {
        historyIndex -= isReversed ? -1 : 1;
        if (!isPaused) TogglePause();
        return;
    }

    DisplayHistoryEntry();
}

private void PrevFrame() {
    isReversed = !isReversed;
    NextFrame();
    isReversed = !isReversed;
}

private void FirstFrame() {
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        historyIndex = !isReversed ? 0 : theHistory.Count() - 1;
        DisplayHistoryEntry();
    }

    private void LastFrame() {
        historyIndex = isReversed ? 0 : theHistory.Count() - 1;
        DisplayHistoryEntry();
    }

    private void TogglePause() {
        isPaused = !isPaused;
        playButton.Content = isPaused ? "\u25b6" : "\u23f8";
        playButton.ToolTip = isPaused ? "Воспроизведение" : "Пауза";
    }

    private void Reverse() {
        isReversed = !isReversed;
        reverseButton.IsChecked = isReversed;
        if (isPaused && isReversed) TogglePause();
    }

    private void NextFrameEvent(object o, EventArgs e) {
        NextFrame();
    }

    private void PrevFrameEvent(object o, EventArgs e) {
        PrevFrame();
    }

    private void FirstFrameEvent(object o, EventArgs e) {
        FirstFrame();
    }

    private void LastFrameEvent(object o, EventArgs e) {
        LastFrame();
    }

    private void TogglePauseEvent(object o, EventArgs e) {
        TogglePause();
    }

    private void ReverseEvent(object o, EventArgs e) {
        Reverse();
    }

    private void SaveEvent(object o, EventArgs e) {
        var dia = new SaveFileDialog();
    }
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

dia.Filter = "Сохранить текущее состояние
(*.json)|*.json|Векторный рисунок дерева
(*.xaml)|*.xaml|Растровый рисунок дерева (*.png)|*.png";
dia.DefaultExt = "json";
dia.FileName = "tree.png";
dia.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

var t = isPaused;
isPaused = true;
var rt = dia.ShowDialog();
isPaused = t;
if (rt != true) return; // так надо: !tr не работает

Save(dia.FileName);
}

private void LoadEvent(object o, EventArgs e) {
    var dia = new OpenFileDialog();
    dia.Filter = "Загрузить текущее состояние (*.json)|*.json";
    dia.DefaultExt = "json";
    dia.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

    var t = isPaused;
    isPaused = true;
    var rt = dia.ShowDialog();
    isPaused = t;
    if (rt != true) return; // так надо: !tr не работает

    Load(dia.FileName);
}

private void NextTutorialEvent(object o, EventArgs e) {
    SelectTutorialPage(tutorialIndex + 1);
}

private void PrevTutorialEvent(object o, EventArgs e) {
    SelectTutorialPage(tutorialIndex - 1);
}

#endregion

static private void ExportToPng(string path, FrameworkElement
element) {
    if (path == null) return;

    // Save current canvas transform
    Transform transform = element.LayoutTransform;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
// reset current transform (in case it is scaled or rotated)
element.LayoutTransform = null;

// Get the size of canvas
Size size = new Size(element.ActualWidth, element.ActualHeight);
// Measure and arrange the surface
// VERY IMPORTANT
element.Measure(size);
element.Arrange(new Rect(size));

// Create a render bitmap and push the surface to it
RenderTargetBitmap renderBitmap = new
RenderTargetBitmap((int)size.Width, (int)size.Height, 96d, 96d,
PixelFormat.Pbgra32);
renderBitmap.Render(element);

// Create a file stream for saving image
using (FileStream outputStream = new FileStream(path,
FileStream.Create)) {
    // Use png encoder for our data
    PngBitmapEncoder encoder = new PngBitmapEncoder();
    // push the rendered bitmap to it
    encoder.Frames.Add(BitmapFrame.Create(renderBitmap));
    // save the data to the stream
    encoder.Save(outputStream);
}

// Restore previously saved layout
element.LayoutTransform = transform;
}
}
```

1.2 ParserLib

1.2.1 ParserLib.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net45</TargetFramework>
    <OutputType>Library</OutputType>
    <GenerateAssemblyInfo>>false</GenerateAssemblyInfo>
  </PropertyGroup>

  <ItemGroup>
    <Content Include="parsers\*.*">
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </Content>
  </ItemGroup>
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТИ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

</Project>

1.2.2 HistoryEntry.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ParserLib {
    public class HistoryEntry : IEnumerable<HistoryToken> {
        public HistoryToken[] TreeRanges { get; } // can be private
        public string RtfGrammar { get; }
        public int CursorPos { get; }

        private bool isTrimmed = false;

        internal HistoryEntry(HistoryToken[] ranges, string rtf) {
            RtfGrammar = rtf;
            TreeRanges = ranges;
            CursorPos = TreeRanges.Max(e => Math.Max(e.StartPos, e.EndPos - 1)) + 1;
        }

        public void SetSettings(bool trim, bool orientation, bool gravity) {
            isTrimmed = trim;
            CalculateDisplayLevels(orientation ^ gravity);
            if (gravity) InvertDisplayLevels();
        }

        private void InvertDisplayLevels() {
            var maxDisplayLevel = this.Max(e => e.DisplayLevel);
            foreach (var tok in this) {
                tok.DisplayLevel = maxDisplayLevel - tok.DisplayLevel;
            }
        }

        private void CalculateDisplayLevels(bool orientation = false) {
            int[] recLvs = new int[CursorPos];

            IEnumerable<HistoryToken> t = this.OrderBy(e => -e.RecLevel);
            if (orientation) t = t.Reverse();
            foreach (var tok in t) {
                var end = tok.EndPos;
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

    if (end == -1) end = CursorPos;

    var slice = new ArraySegment<int>(recLvs, tok.StartPos, end
    - tok.StartPos);
    tok.DisplayLevel = slice.Max();

    for (int i = tok.StartPos; i < end; i++) {
        recLvs[i] = tok.DisplayLevel + 1;
    }
}

// todo: public SortLevels
public override string ToString() {
    return string.Join(" ", (object[])TreeRanges);
}

public IEnumerator<HistoryToken> GetEnumerator() {
    var t = TreeRanges.Where(e => !e.Name.StartsWith("\\"));
    if (isTrimmed) t = t.Where(e => !e.Trimmable);
    return t.GetEnumerator();
}

IEnumerator IEnumerable.GetEnumerator() {
    return GetEnumerator();
}
}
}

```

1.2.3 HistoryToken.cs

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ParserLib {
    public class HistoryToken {
        public string Name { get; }
        public int StartPos { get; }
        public int EndPos { get; }

        internal int RecLevel { get; }
        public bool Trimmable { get; }
        public int DisplayLevel { get; internal set; }
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТИ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

internal HistoryToken(ParserTreeToken tok) {
    Name = tok.Name;
    StartPos = tok.StartPos;
    EndPos = tok.EndPos;
    RecLevel = tok.RecLevel;
    DisplayLevel = RecLevel;
    Trimmable = tok.ChildCount == 1 && tok.EndPos >= 0;
}

public override string ToString() {
    if (EndPos == -1) return $"{StartPos}:-({Name}, {RecLevel})";
    return $"{StartPos}:{EndPos}({Name}, {RecLevel})";
}
}
}

```

1.2.4 Parser.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;

namespace ParserLib {
    public class Parser {
        private string name;
        public Parser(string name) {
            this.name = name;
        }

        public ParserHistory Run(string input) {
            var rtf = File.ReadAllText("parsers/" + this.name + ".rtf");
            var tree = new ParserHistory(rtf, input);

            var process = new Process();
            process.StartInfo.CreateNoWindow = true;
            process.StartInfo.UseShellExecute = false;
            process.StartInfo.RedirectStandardInput = true;
            process.StartInfo.RedirectStandardOutput = true;
            process.OutputDataReceived += (sender, args) => {
                tree.Add(args.Data);
            };
            process.StartInfo.FileName = "parsers/" + this.name + ".exe";
            process.Start();
            process.BeginOutputReadLine();
        }
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТИ 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        process.StandardInput.Write(input);
        process.StandardInput.Close();

        process.WaitForExit();
        return tree;
    }
}
}

```

1.2.5 ParserHistory.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ParserLib {
    public class ParserHistory : IEnumerable<HistoryEntry> {
        private Stack<ParserTreeToken> stack = new Stack<ParserTreeToken>();
        private List<ParserTreeToken> state = new List<ParserTreeToken>();
        private int prevPos = -1;

        /// <summary>
        /// Inherited parser parameters.
        /// </summary>
        public string OriginalRtf { get; }
        public string InputString { get; }

        public IEnumerable<string> RuleNames =>
            RtfBuilder.GetNames(OriginalRtf);

        private List<HistoryEntry> history = new List<HistoryEntry>();

        private HistoryToken[] CopyState() {
            return state.Select(e => e.Clone()).ToArray();
        }

        internal ParserHistory(string rtf, string input) {
            OriginalRtf = rtf;
            InputString = input;
        }

        private void SaveState() {
            var tokens = CopyState();
            var rtf = RtfBuilder.Build(OriginalRtf, stack);

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата


```
var r = new HistoryEntry(tokens, rtf);
history.Add(r);
}

public void Add(string line) {
    if (line == null) return;
    line = line.Trim();
    if (line == "") return;

    if (line.StartsWith("eval failed: SyntaxError:")) {
        // todo
        return;
    }

    var words = line.Split(new string[] { " " },
        StringSplitOptions.RemoveEmptyEntries);
    var pos = int.Parse(words[0].Split(':').Last()) - 1;

    var hasFailed = prevPos > pos;
    while (prevPos > pos) {
        var t = state.Last();
        if (t.EndPos == -1) break;
        t.Parent.ChildCount--;
        state.RemoveAt(state.Count - 1);
        prevPos = t.EndPos;
    }
    prevPos = pos;

    if (hasFailed) SaveState();

    if (words[1] == "rule.enter") {
        var val = 0;
        ParserTreeToken parent = null;
        if (stack.Count != 0) {
            parent = stack.Peek();
            parent.ChildCount++;
            var dict = stack.Peek().Dict;
            dict.TryGetValue(words[2], out val);
            dict[words[2]] = val + 1;
        }
        var t = new ParserTreeToken(parent, words[2], val, pos,
            stack.Count);
        state.Add(t);
        stack.Push(t);
    } else {
        var t = stack.Pop();
        t.EndPos = pos;
        if (words[1] != "rule.match") {
            t.EndPos = -2;
        }
    }
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        t.Parent.ChildCount--;
        var i = state.IndexOf(t);
        state.RemoveRange(i, state.Count - i);
    }
}

SaveState();
}

public IEnumerator<HistoryEntry> GetEnumerator() =>
    history.GetEnumerator();
IEnumerator IEnumerable.GetEnumerator() => history.GetEnumerator();
public HistoryEntry this[int i] => history[i];
}
}

```

1.2.6 ParserTreeToken.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;

namespace ParserLib {
    internal class ParserTreeToken {
        public ParserTreeToken Parent { get; }
        public string Name { get; }
        public int Index { get; }
        public int StartPos { get; }
        public int RecLevel { get; }
        public int EndPos { get; set; }
        public int ChildCount { get; set; }
        public Dictionary<string, int> Dict { get; }

        public ParserTreeToken(ParserTreeToken parent, string name, int
            index, int startPos, int recLevel) {
            if (name.StartsWith("_") && name.EndsWith("_") && name.Length >
                1) {
                var chars = Enumerable.Range(0, name.Length / 2 - 1)
                    .Select(i => (char)Convert.ToUInt16(name.Substring(i * 2
                        + 1, 2), 16));
                name = '"' + string.Join("", chars) + '"';
            }
            this.Parent = parent;
            this.Name = name;
            this.Index = index;

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```

        this.StartPos = startPos;
        this.RecLevel = recLevel;
        this.EndPos = -1;
        this.Dict = new Dictionary<string, int>();
    }

    public override string ToString() {
        return $"{StartPos}:{EndPos}({Name}, {Index})";
    }

    public HistoryToken Clone() {
        return new HistoryToken(this);
    }
}

```

1.2.7 RtfBuilder.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.IO;
using System.Diagnostics;

namespace ParserLib {
    internal class RtfBuilder {
        private string[] lines;

        private RtfBuilder(string rtf) {
            lines = rtf.Split('\n');
            //.Where(e=>e.StartsWith("\\cf")).ToArray();
        }

        private void HighlightIdentifier(ParserTreeToken t) {
            if (t.Parent == null) return;
            var lineIndex = Array.FindIndex(lines, e => e.StartsWith("\\cf2"
            " + t.Parent.Name));
            var line = lines[lineIndex];

            var i = 0;
            var regex = @"\\b\\{\\}(\\cf[0-9] )" + Regex.Escape(t.Name);
            line = Regex.Replace(line, regex, m => {
                if (i++ == t.Index) return @"\\b " + m.Groups[1].Value +
                t.Name;
                return m.Groups[1].Value + t.Name;
            });
        }
    }
}

```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

```
        line = line.Replace("\\b{", "");

        lines[lineIndex] = line;
    }

    private string End() {
        return string.Join("\n", lines).Replace("{", "0");
    }

    public static string Build(string rtf, IEnumerable<ParserTreeToken>
tokens) {
        var builder = new RtfBuilder(rtf);
        foreach (var tok in tokens) {
            builder.HighlightIdentifier(tok);
        }
        return builder.End();
    }

    public static IEnumerable<string> GetNames(string rtf) {
        return rtf.Split('\n')
            .Where(e => e.StartsWith("\\cf"))
            .Select(e => e.Split(new string[] { "\\cf4" },
StringSplitOptions.None)[0].Split(' ')[1]);
    }
}
}
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.13-01 ТП 01-1				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Лист регистрации изменений

[illegible]