**The Formatted Text Editor**

Software Used:

Visual Community 2017

Visual Code

Tools for Diagnose etc:

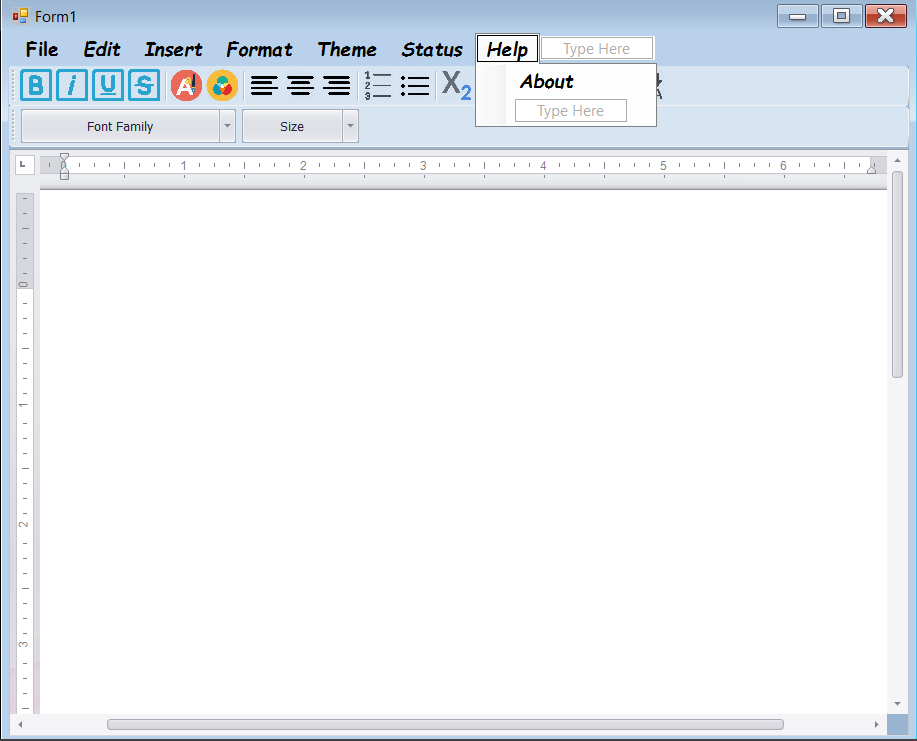
Code Maid : for code cleaning and spading

Build tool : MSBuild

Inbuilt Diagnostic tool (ctrl + alt + F2)

Plugins and External Libraries are contained in Folder namedlibrary texeditor>texteditor>library

Form1.cs main form

****

using AutocompleteMenuNS;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Net.Mail;

using System.Text;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Texditor

{

public partial class Form1 : Form

{

private Bitmap bmp;

private int column;

private int columnline = 1;

//string signature = "Yash Jain \n ";

private string currentdirec = System.IO.Directory.GetCurrentDirectory();

private int currentline = 1;

// Initializing the filename varible

private string fileName = "";

private string[] Fontarr = new string[4];

private float fontSizecon = 0;

private string ftoin = "";

private int line;

private int totalline = 1;

public Form1()

{

InitializeComponent();

toolStripLabel5.Enabled = false;

toolStripLabel6.Enabled = false;

toolStripLabel7.Enabled = false;

toolStripLabel1.Enabled = false;

toolStripLabel3.Enabled = false;

foreach (FontFamily oneFontFamily in FontFamily.Families)

{

fontname.Items.Add(oneFontFamily.Name);

}

fontname.Text = this.richTextBox1.Font.Name.ToString();

fontsize.Text = this.richTextBox1.Font.Size.ToString();

autocompleteMenu1.SetAutocompleteItems(new DynamicCollection(richTextBox1));

// richTextBox1.Focus();

}

/\* private RichTextBox GetCurrentDocument

{

get { return (RichTextBox)Controls["Body"]; }

}\*/

public void mailSSend(string from, string to, string password, string subject, StringBuilder abc, string simpletext)

{

try

{

MailMessage mail = new MailMessage();

SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");

mail.From = new MailAddress(from);

mail.To.Add(to);

mail.Subject = subject;

mail.Body = simpletext;

SmtpServer.Port = 587;

SmtpServer.Credentials = new System.Net.NetworkCredential(from, password);

SmtpServer.EnableSsl = true;

SmtpServer.Send(mail);

MessageBox.Show("Mail successfully sended to " + to);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

//richTextBox1.Rtf = "" + rtbb;

richTextBox1.Rtf = "" + abc;

}

public void Whenreplaced(StringBuilder replacing)

{

richTextBox1.Rtf = "" + replacing;

}

private void closeFileToolStripMenuItem\_Click(object sender, EventArgs e)// Saving File

{//Closing File

try

{

string rtext = richTextBox1.Text;

if (rtext.Length > 0)

{

DialogResult res = MessageBox.Show("You want to save the document?", "Confirmation", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);

if (res == DialogResult.Yes)

{// SAveDialog box

saveFileDialog1.Filter =

"Files (\*.txt)|\*.txt| rtf (\*.rtf)|\*.rtf| Docx (\*.Doc)|\*.Doc ";

if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

}

}

else if (res == DialogResult.No)

{

Form1 frm = new Form1();

frm.Show();

this.Hide();

}

else

{

}

}

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

}

private void columnsToolStripMenuItem\_Click(object sender, EventArgs e)

{//Initializing column number should be displayed or not

if (columnline == 1)

{

ColumnNumber.Text = "";

columnline = 0;

}

else if (columnline == 0)

{

ColumnNumber.Text = "";

columnline = 1;

}

}

private void copyToolStripMenuItem\_Click(object sender, EventArgs e)

{

Clipboard.SetText(richTextBox1.SelectedText);

}

private void dateAndTimeToolStripMenuItem\_Click(object sender, EventArgs e)

{//Inserting Date and Time

DateTime dt = DateTime.Now;

string wDateTime = " ";

wDateTime = dt.ToString();

int CaretPosition = richTextBox1.SelectionStart;

string TextBefore = richTextBox1.Text.Substring(0, CaretPosition);

string textAfter = richTextBox1.Text.Substring(CaretPosition);

richTextBox1.Text = TextBefore + wDateTime + textAfter;

/\*StringBuilder abc = new StringBuilder(dt.ToString());

StringBuilder appendingtext = new StringBuilder(richTextBox1.Rtf + "\n" +abc+ "\n");

richTextBox1.Rtf = ""+appendingtext;

\*/

/\*

DateTime myTime = new DateTime();

myTime = DateTime.Now;

string wDateTime = " ";

wDateTime = myTime.ToString("g");

And then using what you gave me, I finished it with:

int CaretPosition = rtbMain.SelectionStart;

string TextBefore = rtbMain.Text.Substring(0, CaretPosition);

string textAfter = rtbMain.Text.Substring(CaretPosition);

rtbMain.Text = TextBefore + wDateTime + textAfter;

\*/

}

private void decodeAnImageToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Decode An image in base64 string

openFileDialog1.FileName = "";

openFileDialog1.Filter = "Images (\*.BMP;\*.JPG;\*.GIF)|\*.BMP;\*.JPG;\*.GIF|" +

"All files (\*.\*)|\*.\*";

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

richTextBox1.Text += Convert.ToBase64String(File.ReadAllBytes(openFileDialog1.FileName));

File.WriteAllText("Decoded Image.txt", Convert.ToBase64String(File.ReadAllBytes(openFileDialog1.FileName)));

}

}

private void decodeImageToolStripMenuItem\_Click(object sender, EventArgs e)

{

var img = System.Drawing.Image.FromStream(new MemoryStream(Convert.FromBase64String(richTextBox1.Text)));

richTextBox1.Rtf = "" + img;

}

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)// Exit Application

{

System.Windows.Forms.Application.Exit();

}

private void fileToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void findAndReplaceToolStripMenuItem\_Click(object sender, EventArgs e)

{

StringBuilder sb = new StringBuilder(richTextBox1.Rtf);

FindReplace form2 = new FindReplace(sb);

form2.Show();

this.Hide();

}

private void fontname\_SelectedIndexChanged(object sender, EventArgs e)

{

//Font Family

try

{

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

String fontName = fontname.SelectedItem.ToString();

FontFamily family = new FontFamily(fontName);

richTextBox1.SelectionFont = new System.Drawing.Font(family, richTextBox1.SelectionFont.Size, richTextBox1.SelectionFont.Style);

}

catch (Exception eee) { MessageBox.Show(eee.Message); }

//richTextBox1.SelectionFont = new Font(richTextBox1.Font.FontFamily.(fontfam));

// richTextBox1.SelectionFont = new Font();

}

private void fontsize\_DropDownClosed(object sender, EventArgs e)

{

//Font Size change

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

/\*

try

{

int fontheight = (int)fontsize22.SelectedItem;

richTextBox1.SelectionFont = new Font(currentFont.FontFamily, fontheight, richTextBox1.SelectionFont.Style);

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

\*/

}

private void fontsize\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void fontSize2\_SelectedIndexChanged(object sender, EventArgs e)

{

//Font Size change

try

{

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

ftoin = fontsize.SelectedItem.ToString();

fontSizecon = float.Parse(ftoin);

richTextBox1.SelectionFont = new System.Drawing.Font(currentFont.FontFamily, fontSizecon, richTextBox1.SelectionFont.Style);

}

catch (Exception eee)

{

MessageBox.Show(eee.Message);

}

}

private void fontToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Font Dialog Box

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

richTextBox1.SelectionFont = new System.Drawing.Font(fontDialog1.Font.FontFamily, fontDialog1.Font.Size, fontDialog1.Font.Style);

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void formatToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void imageSendToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void imageToolStripMenuItem\_Click(object sender, EventArgs e)

{//Inserting an image

openFileDialog1.Filter = "Images (\*.BMP;\*.JPG;\*.GIF)|\*.BMP;\*.JPG;\*.GIF|" +

"All files (\*.\*)|\*.\*";

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

//StringBuilder abc = new StringBuilder();

//Getting File Name

string myfile = openFileDialog1.FileName;

//Creating object of Bitmap

Bitmap mybitmap = new Bitmap(myfile);

Clipboard.SetDataObject(mybitmap);

DataFormats.Format myformat = DataFormats.GetFormat(DataFormats.Bitmap);

if (this.richTextBox1.CanPaste(myformat))

{

richTextBox1.Paste(myformat);

}

}

/\*

try

{

openFileDialog1.Multiselect = true;

dynamic orgdata = Clipboard.GetDataObject();

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

foreach (string fname in openFileDialog1.FileNames)

{

System.Drawing.Image img = System.Drawing.Image.FromFile(fname);

richTextBox1.Paste();

}

}

Clipboard.SetDataObject(orgdata);

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

OpenFileDialog dlg = new OpenFileDialog();

dlg.Filter =

"Images (\*.BMP;\*.JPG;\*.GIF)|\*.BMP;\*.JPG;\*.GIF|" +

"All files (\*.\*)|\*.\*";

dlg.ShowDialog();

dlg.Title = "My Image Browser";

string file = openFileDialog1.FileName;

PictureBox pb = new PictureBox();

pb.Image = Image.FromFile(file);

pb.SizeMode = PictureBoxSizeMode.StretchImage;

richTextBox1.Controls.Add(pictureBox1);

\*/

}

private void italic\_Click(object sender, EventArgs e) // For Italic

{

//Storing current font style family and size

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

System.Drawing.FontStyle newFontStyle;

int arr = 0;

//Initializing Array of arr variable

//Checking condition whether selected text have which font style

if (richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 0;

}

if (richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 1;

}

if (richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 2;

}

if (richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 3;

}

if (richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 4;

}

if (richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 5;

}

if (richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 6;

}

if (richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 7;

}

if (richTextBox1.SelectionFont.Italic == true)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

else if (richTextBox1.SelectionFont.Italic == false)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold | FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Italic | FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

}

private void newToolStripMenuItem\_Click(object sender, EventArgs e) // New Document

{

try

{

//Getting all content of richtextbox in string rtext

string rtext = richTextBox1.Text;

if (rtext.Length > 0)

{

//Getting result of dialog in the res dialog result

DialogResult res = MessageBox.Show("You want to save the document?", "Confirmation", MessageBoxButtons.YesNoCancel);

//Checking condition of res result

if (res == DialogResult.Yes)

{// SAveDialog box

saveFileDialog1.Filter =

"Text (\*.txt)|\*.txt| rtf (\*.rtf)|\*.rtf| Docx (\*.Doc)|\*.Doc";

if (saveFileDialog1.FileName.Contains(".txt"))

{

//Writing the test to file

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

}

else

{

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Rtf);

}

/\*if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

}\*/

}

else if (res == DialogResult.No)

{

Form1 frm = new Form1();

//Showing the form1 by calling an object

frm.Show();

//This will hide the form

this.Hide();

}

else

{

}

}

}

catch (Exception ee)

{

//Showing the Message Dialog box

MessageBox.Show(ee.Message);

}

}

private void oHYEAHToolStripMenuItem\_Click(object sender, EventArgs e)

{

StringBuilder sending = new StringBuilder(richTextBox1.Rtf);

mailSend mlsd = new mailSend(sending, richTextBox1.Text);

mlsd.Show();

//mailItem.PropertyAccessor.SetProperty("http://schemas.microsoft.com/mapi/id/{00062008-0000-0000-C000-000000000046}/8582000B", true);

}

private void openToolStripMenuItem\_Click(object sender, EventArgs e) // Open File Dialog

{

try

{

//Getting all text in rtext

string rtext = richTextBox1.Text;

//Checking the Length of text written in textbox

if (rtext.Length > 0)

{

DialogResult res = MessageBox.Show("You want to save the document?", "Confirmation", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);

if (res == DialogResult.Yes)

{// SAveDialog box

//Setting the filter of dialog box

saveFileDialog1.Filter =

"Files (\*.txt)|\*.txt| rtf (\*.rtf)|\*.rtf| Docx (\*.Doc)|\*.Doc ";

if (saveFileDialog1.FileName.Contains(".txt"))

{

//Writing the text to FIle

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

}

else

{

//Writing the RTF text into file

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Rtf);

}

}

else if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.Cancel)

{

//It will do nothing no new process executed

}

else

{

//It initialize new dialog box

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.ShowDialog();

//Shows the dialog box

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

fileName = openFileDialog1.FileName;

//getting text in line string

string lines = System.IO.File.ReadAllText(fileName);

richTextBox1.Text = "" + lines;

}

}

}

else

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

fileName = openFileDialog1.FileName;

string lines = System.IO.File.ReadAllText(fileName);

richTextBox1.Text = lines;

}

}

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

}

private void pasteToolStripMenuItem\_Click(object sender, EventArgs e)

{

richTextBox1.SelectedText = Clipboard.GetText(TextDataFormat.Text);

}

private void printDocument1\_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)//Function For Printpreview

{

e.Graphics.DrawString(richTextBox1.Text, richTextBox1.Font, Brushes.Black, 100, 200);

}

private void redoToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (richTextBox1.CanRedo == true)

{

//Redo

richTextBox1.Redo();

}

}

private void richTextBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Control && e.KeyCode == Keys.Space)

autocompleteMenu1.Show(richTextBox1, true);

int tempNum;

if (e.KeyCode == Keys.Enter)

try

{

if (char.IsDigit(richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine()]))

{

if (char.IsDigit(richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine() + 1]) && richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine() + 2] == '.')

tempNum = int.Parse(richTextBox1.Text.Substring(richTextBox1.GetFirstCharIndexOfCurrentLine(), 2));

else tempNum = int.Parse(richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine()].ToString());

if (richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine() + 1] == '.' || (char.IsDigit(richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine() + 1]) && richTextBox1.Text[richTextBox1.GetFirstCharIndexOfCurrentLine() + 2] == '.'))

{

tempNum++;

richTextBox1.SelectedText = "\r\n" + tempNum.ToString() + ". ";

e.SuppressKeyPress = true;

}

}

}

catch { }

}

private void richTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

}

private void richTextBox1\_KeyUp(object sender, KeyEventArgs e)

{//Key Up event

int numbr = 0;

numbr = richTextBox1.Text.Length;

if (numbr > 0)

{

if (richTextBox1.Text[numbr - 1].ToString() == "{")

richTextBox1.Text += "}";

if (richTextBox1.Text[numbr - 1].ToString() == "(")

richTextBox1.Text += ")";

if (richTextBox1.Text[numbr - 1].ToString() == "[")

richTextBox1.Text += "]";

if (richTextBox1.Text[numbr - 1].ToString() == "<")

richTextBox1.Text += ">";

}

int countline = 0;

if (totalline == 1)

{

for (int i = 0; i < richTextBox1.Text.Length; i++)

{

if (richTextBox1.Text[i] == '\n')

{

countline++;

}

}

TotalLines.Text = "Total Lines : " + (countline + 1);

}

if (currentline == 1)

{

LineNumber.Text = "Current Line : " + line;

}

if (columnline == 1)

{

ColumnNumber.Text = "Column Number : " + column;

}

}

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

{

//calculation of line number

//calculation of column number

richTextBox1.ScrollBars = System.Windows.Forms.RichTextBoxScrollBars.Both;

line = 1 + richTextBox1.GetLineFromCharIndex(richTextBox1.GetFirstCharIndexOfCurrentLine());

column = 1 + richTextBox1.SelectionStart - richTextBox1.GetFirstCharIndexOfCurrentLine();

}

private void saveAsToolStripMenuItem\_Click(object sender, EventArgs e) // Save As Dialog

{

//This is initializing savedialog box

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

//Setting filter of dialog box

saveFileDialog1.Filter =

"Text (\*.txt)|\*.txt|Rtf (\*.rtf)|\*.rtf|Doc (\*.doc)|\*.doc ";

if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

if (saveFileDialog1.FileName.Contains(".txt"))

{

//Writing the text to FIle

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

}

else

{

//Writing the RTF text into file

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Rtf);

}

}

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{//Save Dialog

try

{

if (fileName.Length == 0)

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter =

"Files (\*.txt)|\*.txt|rtf (\*.rtf)|\*.rtf|Doc (\*.Doc)|\*.Doc ";

if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

fileName = saveFileDialog1.FileName.ToString();

if (saveFileDialog1.FileName.Contains(".txt"))

{

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

fileName = saveFileDialog1.FileName.ToString();

}

else

{

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Rtf);

fileName = saveFileDialog1.FileName.ToString();

}

}

}

else

{

///error

// if (saveFileDialog1.FileName.Contains(".txt"))

File.WriteAllText(saveFileDialog1.FileName, richTextBox1.Text);

}

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

}

private void setToolStripMenuItem\_Click(object sender, EventArgs e)

{

signature sgn = new signature();

sgn.ShowDialog();

}

private void sETToolStripMenuItem1\_Click(object sender, EventArgs e)

{

try

{

if (File.Exists("signature.txt"))

{

var lines = File.ReadAllLines("signature.txt");

foreach (var line in lines)

richTextBox1.Text += line + "\n";

}

else { MessageBox.Show("Signature doesn't exist"); }

}

catch (Exception ee)

{ MessageBox.Show(ee.Message); }

}

private void showLinesToolStripMenuItem\_Click(object sender, EventArgs e) // Current Line

{

//TextPointer caretLineStart =

if (currentline == 1)

{

LineNumber.Text = "";

currentline = 0;

}

else if (currentline == 0)

{

LineNumber.Text = "";

currentline = 1;

}

}

private void strikethrough\_Click(object sender, EventArgs e) // For strikethrough

{

//Storing current font style family and size

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

System.Drawing.FontStyle newFontStyle;

int arr = 0;

//Initializing Array of arr variable

//Checking condition whether selected text have which font style

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == true)

{

arr = 0;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == false)

{

arr = 1;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == true)

{

arr = 2;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == false)

{

arr = 3;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == true)

{

arr = 4;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Underline == false)

{

arr = 5;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == true)

{

arr = 6;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Underline == false)

{

arr = 7;

}

if (richTextBox1.SelectionFont.Strikeout == true)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

else if (richTextBox1.SelectionFont.Strikeout == false)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Italic | FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Italic | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Italic | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Italic;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Strikeout | FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

}

private void ststuaToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void tableToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Jumping to new form of table creation

StringBuilder sb = new StringBuilder(richTextBox1.Rtf);

InsertTable form2 = new InsertTable(sb);

form2.Show();

this.Hide();

}

private void testingBetsToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (testingBetsToolStripMenuItem.Checked == true)

{

menuStrip1.BackColor = Color.FromArgb(179, 236, 255);

toolStrip1.BackColor = Color.FromArgb(179, 236, 255);

toolStrip2.BackColor = Color.FromArgb(179, 236, 255);

richTextBox1.BackColor = Color.White;

Form1 frm = new Form1();

frm.BackColor = Color.FromArgb(179, 236, 255);

richTextBox1.ForeColor = Color.Black;

}

if (testingBetsToolStripMenuItem.Checked == false)

{

menuStrip1.BackColor = Color.Gray;

toolStrip1.BackColor = Color.Gray;

toolStrip2.BackColor = Color.Gray;

richTextBox1.BackColor = Color.Black;

Form1 frm = new Form1();

frm.BackColor = Color.Gray;

richTextBox1.ForeColor = Color.White;

}

}

private void testingToolStripMenuItem\_Click(object sender, EventArgs e)

{

//System.Diagnostics.Process.Start("

}

private void toolStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void toolStrip2\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void toolStripButton1\_Click(object sender, EventArgs e) // UpperCasing

{

richTextBox1.SelectedText = richTextBox1.SelectedText.ToUpper();

//richTextBox1.SelectedText.ToUpper();

}

private void toolStripButton10\_Click(object sender, EventArgs e)

{//Numbering the Selected Text

string temptext = richTextBox1.SelectedText;

//This is just storing the text starting index

int SelectionStart = richTextBox1.SelectionStart;

//This variable is just getting the length of selected text

int SelectionLength = richTextBox1.SelectionLength;

richTextBox1.SelectionStart = richTextBox1.GetFirstCharIndexOfCurrentLine();

richTextBox1.SelectionLength = 0;

richTextBox1.SelectedText = "1. ";

int j = 2;

for (int i = SelectionStart; i < SelectionStart + SelectionLength; i++)

if (richTextBox1.Text[i] == '\n')

{

richTextBox1.SelectionStart = i + 1;

richTextBox1.SelectionLength = 0;

richTextBox1.SelectedText = j.ToString() + ". ";

j++;

SelectionLength += 3;

}

/\*

private void btnNumbers\_Click(object sender, EventArgs e)

{

string temptext = rtbMain.SelectedText;

int SelectionStart = rtbMain.SelectionStart;

int SelectionLength = rtbMain.SelectionLength;

rtbMain.SelectionStart = rtbMain.GetFirstCharIndexOfCurrentLine();

rtbMain.SelectionLength = 0;

rtbMain.SelectedText = "1. ";

int j = 2;

for( int i = SelectionStart; i < SelectionStart + SelectionLength; i++)

if (rtbMain.Text[i] == '\n')

{

rtbMain.SelectionStart = i + 1;

rtbMain.SelectionLength = 0;

rtbMain.SelectedText = j.ToString() + ". ";

j++;

SelectionLength += 3;

}

}

private void rtbMain\_KeyDown(object sender, KeyEventArgs e)

{//this piece of code automatically increments the bulleted list when user //presses Enter key

int tempNum;

if (e.KeyCode == Keys.Enter)

try

{

if (char.IsDigit(rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine()]))

{

if (char.IsDigit(rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine() + 1]) && rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine() + 2] == '.')

tempNum = int.Parse(rtbMain.Text.Substring(rtbMain.GetFirstCharIndexOfCurrentLine(),2));

else tempNum = int.Parse(rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine()].ToString());

if (rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine() + 1] == '.' || (char.IsDigit(rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine() + 1]) && rtbMain.Text[rtbMain.GetFirstCharIndexOfCurrentLine() + 2] == '.'))

{

tempNum++;

rtbMain.SelectedText = "\r\n" + tempNum.ToString() + ". ";

e.SuppressKeyPress = true;

}

}

}

catch{}

}

\*/

/\*

try

{

int count = 0;

int inc = 1;

int firstIndex = richTextBox1.SelectionStart;

int lastIndex = richTextBox1.SelectionStart + richTextBox1.SelectionLength;

string texttobechanged = richTextBox1.Text;

for (int i = firstIndex; i < lastIndex; i++)

{

if (richTextBox1.Text[i] == '\n')

{

count++;

if ((i + 1) < lastIndex)

{

richTextBox2.Text += "\n" + richTextBox1.Text[i + 1] + (i + 1);

richTextBox3.Text += richTextBox1.Text.Insert(i + 1, inc + ". ") + "\n";

/\*string temp = inc + ". " + richTextBox1.Text[i];

string toberp = "aaa";

toberp = toberp.Replace(richTextBox1.Text[i], temp);

// Convert.ToString(richTextBox1.Text[i]).Replace(richTextBox1.Text[i],temp);

inc++;

}

}

}

richTextBox1.Text = texttobechanged;

check.Text += "This is the number : " + count;

}

catch (Exception ee)

{

MessageBox.Show(ee.Message);

}

if (richTextBox1.selection == true)

{ richTextBox1.SelectionBullet = false; }

else

{ richTextBox1.SelectionBullet = true; }

\*/

}

private void toolStripButton11\_Click(object sender, EventArgs e) // Bulleting

{

//It is basically toggling between selection text bulleting

if (richTextBox1.SelectionBullet == true)

{ richTextBox1.SelectionBullet = false; }

else

{ richTextBox1.SelectionBullet = true; }

}

private void toolStripButton12\_Click(object sender, EventArgs e) // Subscript

{

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

//Getting current selected font from richtextbox

if (richTextBox1.SelectionCharOffset == -3)

{

richTextBox1.SelectionCharOffset = 0;

}

else

{

richTextBox1.SelectionCharOffset = -3;

//richTextBox1.SelectionFont = new System.Drawing.Font(currentFont.FontFamily, 7, richTextBox1.Font.Style);

}

}

private void toolStripButton13\_Click(object sender, EventArgs e) // SuperScript

{

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

//Char Offset is basically a base line defined for textbox or richtextbox

if (richTextBox1.SelectionCharOffset == 3)

{

//Has its own unit of setting character putting 3/4 or any unit above the base line

richTextBox1.SelectionCharOffset = 0;

}

else

{

richTextBox1.SelectionCharOffset = 3;

richTextBox1.SelectionFont = new System.Drawing.Font(currentFont.FontFamily, 7, richTextBox1.Font.Style);

}

}

private void toolStripButton14\_Click(object sender, EventArgs e) //Print Dialog

{

printDialog1.Document = printDocument1;

//Instanciating document as object of printdialog

if (printDialog1.ShowDialog() == DialogResult.OK)

{

printDocument1.Print();

}

/\* PrintDialog dlg = new PrintDialog();

dlg.ShowDialog();\*/

}

private void toolStripButton15\_Click(object sender, EventArgs e)//PrintPreview

{

/\*int XX = richTextBox1.Location.X;

int YY = richTextBox1.Location.Y;

Point locationOnForm = richTextBox1.FindForm().PointToScreen(richTextBox1.Location);

XX = locationOnForm.X;

YY = locationOnForm.Y;

Graphics g = this.CreateGraphics();

bmp = new Bitmap(this.Size.Width, this.Size.Height);

Graphics mg = Graphics.FromImage(bmp);

mg.CopyFromScreen(45 + XX, 65 + YY, 100, 100, this.Size);

\*/

printPreviewDialog1.ShowDialog();

}

private void toolStripButton16\_Click(object sender, EventArgs e) // For Bold

{//Storing current font style family and size

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

System.Drawing.FontStyle newFontStyle;

int arr = 0;

//Initializing Array of arr variable

//Checking condition whether selected text have which font style

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 0;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 1;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 2;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 3;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 4;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Underline == true && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 5;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 6;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Underline == false && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 7;

}

if (richTextBox1.SelectionFont.Bold == true)

{

//Switch case

//checking varible getting value

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

else if (richTextBox1.SelectionFont.Bold == false)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Italic | FontStyle.Underline | FontStyle.Strikeout;

//Setting Font

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Italic | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Italic | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Italic;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Underline;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Bold | FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

}

private void toolStripButton2\_Click(object sender, EventArgs e) // LowerCasing

{

richTextBox1.SelectedText = richTextBox1.SelectedText.ToLower();

}

private void toolStripButton5\_Click(object sender, EventArgs e) // Font Color

{

//Instanciating an object of colordialog

ColorDialog colorDialog1 = new ColorDialog();

//Checking condition whether the dialog box is successfully opened or not

if (colorDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

richTextBox1.SelectionColor = colorDialog1.Color;

}

else

MessageBox.Show("Color dialog error");

}

private void toolStripButton6\_Click(object sender, EventArgs e)//Background Color

{//this is for Backgrounf color for the richtextbox

ColorDialog colorDialog1 = new ColorDialog();

//Setting the background color of rich

//colorDialog1.Color = richTextBox1.BackColor;

if (colorDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK &&

colorDialog1.Color != richTextBox1.BackColor)

{

richTextBox1.BackColor = colorDialog1.Color;

}

}

private void toolStripButton7\_Click(object sender, EventArgs e) // Left Alignment

{

richTextBox1.SelectionAlignment = HorizontalAlignment.Left;

}

private void toolStripButton8\_Click(object sender, EventArgs e) // Center Alignment

{

richTextBox1.SelectionAlignment = HorizontalAlignment.Center;

}

private void toolStripButton9\_Click(object sender, EventArgs e) // Right Alignment

{

richTextBox1.SelectionAlignment = HorizontalAlignment.Right;

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

/\* FileStream outstream = File.Create(richTextBox1.Rtf);

Document docText = new Document();

PdfWriter writer = PdfWriter.getInstance(docText, outstream);

docText.Open();

\*/

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.DefaultExt = ".pdf";

saveFileDialog1.Filter =

"Files (\*.pdf)|\*.pdf";

if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

/\*

\*

\*

\*

\* Aspose.Pdf.Generator.Pdf pdf1 = new Aspose.Pdf.Generator.Pdf();

Aspose.Pdf.Generator.Section sec1 = pdf1.Sections.Add();

sec1.Paragraphs.Add(new Aspose.Pdf.Generator.Text(richTextBox1.Rtf));

pdf1.Save(saveFileDialog1.FileName);

Working Condition\*\*\*

\*

\*

\*

\*/

SautinSoft.PdfMetamorphosis p = new SautinSoft.PdfMetamorphosis();

string fromtext = this.richTextBox1.Rtf;

byte[] pdf = p.RtfToPdfConvertByte(fromtext);

File.WriteAllBytes(saveFileDialog1.FileName, pdf);

System.Diagnostics.Process.Start(saveFileDialog1.FileName);

}

/\*

try

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.DefaultExt = ".pdf";

saveFileDialog1.Filter =

"Files (\*.pdf)|\*.pdf";

if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

Document doc = new Document(iTextSharp.text.PageSize.LETTER, 10, 10, 42, 35);

PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream(saveFileDialog1.FileName.ToString(), FileMode.Create));

doc.Open();

iTextSharp.text.Paragraph para = new iTextSharp.text.Paragraph(richTextBox1.Rtf);

doc.Add(para);

doc.Close();

string str = this.richTextBox1.Rtf;

TextReader tr = new StringReader(str);

Document doc = new Document();

FlowDocument floDoc = new FlowDocument(new Paragraph(new Run("Simple Document")));

System.Windows.Controls.RichTextBox rtb = new System.Windows.Controls.RichTextBox();

rtb.Document = floDoc;

PdfDocument pdf = new PdfDocument();

string richtt = "PDF";

pdf.Info.Title = richtt;

PdfPage pdfPage = pdf.AddPage();

XGraphics graph = XGraphics.FromPdfPage(pdfPage);

string tobetransform = richTextBox1.Font.ToString();

Font tobeconvert = richTextBox1.Font;

double transform = double.Parse(tobetransform.Substring(tobetransform.IndexOf("Size") + 5, (tobetransform.IndexOf(",", tobetransform.IndexOf("Size"))) - (tobetransform.IndexOf("Size") + 5)));

XFont font = new XFont(currentFont.FontFamily.ToString(), transform, XFontStyle.Regular);

graph.DrawString(richTextBox1.Rtf, font, XBrushes.Black, new XRect(15, 40, pdfPage.Width.Point + 50, pdfPage.Height.Point + 50), XStringFormats.TopLeft);

string pdfFilename = saveFileDialog1.FileName;

pdf.Save(pdfFilename);

Process.Start(pdfFilename);

}

}

catch (Exception mess)

{

MessageBox.Show(mess.Message);

}

\*/

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

if (richTextBox1.SelectionLength > 0)

{

Clipboard.SetText(richTextBox1.SelectedText);

richTextBox1.SelectedText = "";

}

}

private void toolStripMenuItem3\_Click(object sender, EventArgs e)

{ //Printing Dialog

printDialog1.Document = printDocument1;

if (printDialog1.ShowDialog() == DialogResult.OK)

{

printDocument1.Print();

}// PrintDialog dlg = new PrintDialog();

// dlg.ShowDialog();

}

private void toolStripMenuItem4\_Click(object sender, EventArgs e)

{

//Print Preview Dialog

printPreviewDialog1.ShowDialog();

/\*int XX = richTextBox1.Location.X;

int YY = richTextBox1.Location.Y;

Point locationOnForm = richTextBox1.FindForm().PointToScreen(richTextBox1.Location);

XX = locationOnForm.X;

YY = locationOnForm.Y;

Graphics g = this.CreateGraphics();

bmp = new Bitmap(this.Size.Width, this.Size.Height);

Graphics mg = Graphics.FromImage(bmp);

mg.CopyFromScreen(45 + XX, 65 + YY, 100, 100, this.Size);

printPreviewDialog1.ShowDialog();

\*/

}

//int linecount = 0;

private void totalLinesToolStripMenuItem\_Click(object sender, EventArgs e)

{//Initializing line number should be displayed or not

if (totalline == 1)

{

totalLinesToolStripMenuItem.Text = "Total Lines";

TotalLines.Text = "";

totalline = 0;

}

else if (totalline == 0)

{

TotalLines.Text = "";

totalline = 1;

totalLinesToolStripMenuItem.Text = "Total Lines";

}

}

private void underline\_Click(object sender, EventArgs e) // For Underline

{

//Storing current font style family and size

System.Drawing.Font currentFont = richTextBox1.SelectionFont;

System.Drawing.FontStyle newFontStyle;

int arr = 0;

//Initializing Array of arr variable

//Checking condition whether selected text have which font style

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 0;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 1;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 2;

}

if (richTextBox1.SelectionFont.Italic == true && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 3;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 4;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == true && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 5;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Strikeout == true)

{

arr = 6;

}

if (richTextBox1.SelectionFont.Italic == false && richTextBox1.SelectionFont.Bold == false && richTextBox1.SelectionFont.Strikeout == false)

{

arr = 7;

}

if (richTextBox1.SelectionFont.Underline == true)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Italic;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

else if (richTextBox1.SelectionFont.Underline == false)

{

switch (arr)

{

case 0:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Italic | FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 1:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Italic | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 2:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Italic | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 3:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Italic;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 4:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Bold | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 5:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Bold;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 6:

newFontStyle = FontStyle.Regular;

newFontStyle = FontStyle.Underline | FontStyle.Strikeout;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

case 7:

newFontStyle = FontStyle.Underline | FontStyle.Regular;

richTextBox1.SelectionFont = new System.Drawing.Font(

currentFont.FontFamily,

currentFont.Size,

newFontStyle

);

break;

}

}

}

private void undoToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (richTextBox1.CanUndo == true)

{//Undo

richTextBox1.Undo();

}

}

private void webBrowser1\_DocumentCompleted(object sender, WebBrowserDocumentCompletedEventArgs e)

{

}

private void wrapTextToolStripMenuItem\_Click(object sender, EventArgs e)

{

//Wrapping Text

if (wrapTextToolStripMenuItem.Checked == true)

{

richTextBox1.WordWrap = true;

}

else

{

richTextBox1.WordWrap = false;

}

}

}

internal class DynamicCollection : IEnumerable<AutocompleteItem>

{

private TextBoxBase richTextBox1;

public DynamicCollection(TextBoxBase richTextBox1)

{

this.richTextBox1 = richTextBox1;

}

public IEnumerator<AutocompleteItem> GetEnumerator()

{

return BuildList().GetEnumerator();

}

System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

private IEnumerable<AutocompleteItem> BuildList()

{

//find all words of the text

var words = new Dictionary<string, string>();

foreach (Match m in Regex.Matches(richTextBox1.Text, @"\b\w+\b"))

words[m.Value] = m.Value;

//return autocomplete items

foreach (var word in words.Keys)

yield return new AutocompleteItem(word);

}

}

}