



Benemérita Universidad Autónoma de Puebla

Materia: *Sistemas Operativos II*

Profesor: *Gilberto López Poblano*

Alumno: *Christian Amauri Amador Ortega - 201927821*

- Proyecto Word -

Para esta serie de prácticas con código utilizamos el lenguaje C# en el IDE visual studio (la versión más reciente en septiembre de 2022). Puede haber variaciones en el funcionamiento de exactamente los mismos códigos pero con otra versión de visual studio u otro IDE. De todas formas estas variaciones deben de ser mínimas e insignificantes.

!!! Documentaremos el funcionamiento de las aplicaciones (en su ejecución) y haremos algunos comentarios / notas / consideraciones / advertencias al respecto. Para ahorrar tiempo y esfuerzo, trataremos de no ahondar demasiado en la estructura del código, sino simplemente explicar brevemente cómo funciona (se asume que el lector tiene ya conocimientos suficientes sobre conceptos varios de programación estructurada, ambiente gráfico, variables, datos, y demás...)

Finalmente hay que tomar en cuenta que C# y Visual studio son herramientas pesadas y caprichosas. Y que el equipo que vayamos a utilizar debe tener cierta capacidad de rendimiento superior a sólo básica.

Primero, el objetivo del proyecto es crear un programa en C# con el que podamos simular un pequeño programa tipo Microsoft Word, para crear documentos de texto con variaciones de fuente en color, tamaño, tipo y alineado, además de la opción de insertar imágenes. Guardar, y abrir archivos con un formato definido por nosotros.

La ventaja que tenemos en este proyecto es que aparentemente va a estar construido enteramente con funciones predefinidas por el lenguaje, así que mucho trabajo de ingeniería no debemos hacer realmente, simplemente usar y conocer las herramientas que ya fueron hechas por alguien más en algún momento (lo cual es muy bueno para la práctica, pero considerablemente desfavorable para la documentación).

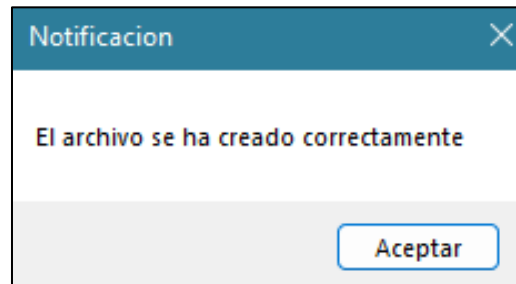
En cualquier caso, el repositorio en el que este código se encuentra (y otros códigos en C# del curso de Sistemas operativos 2, BUAP FCC, Otoño 2022) está disponible haciendo click en el siguiente enlace:

<https://drive.google.com/drive/folders/1yZujMI51XAnEZBMj-4ykUWMfHXBC9uYp?usp=sharing>

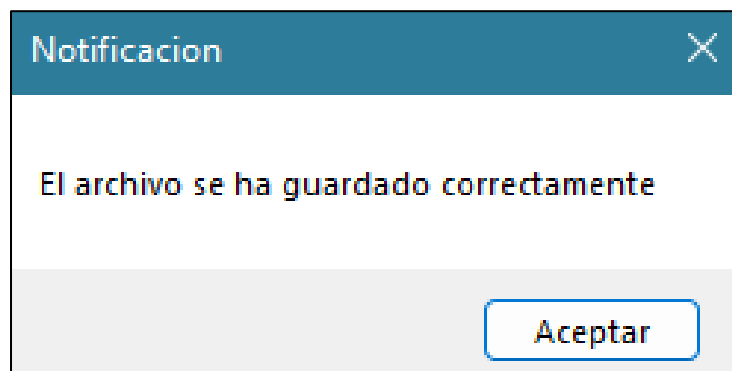
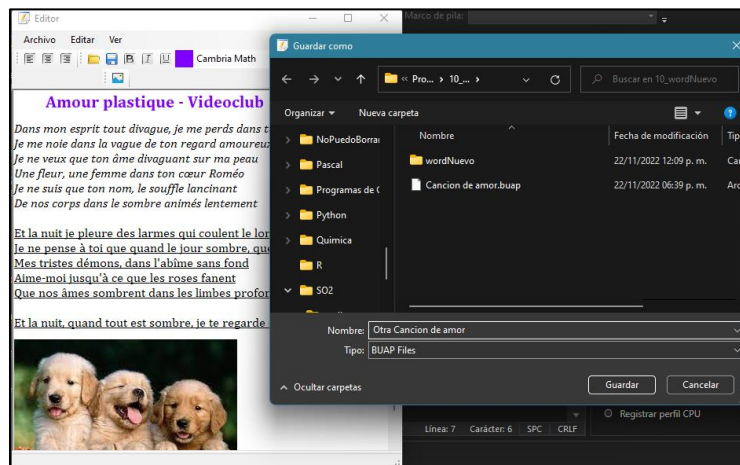
Y estará disponible al menos durante el resto de 2022. Gracias. 😊

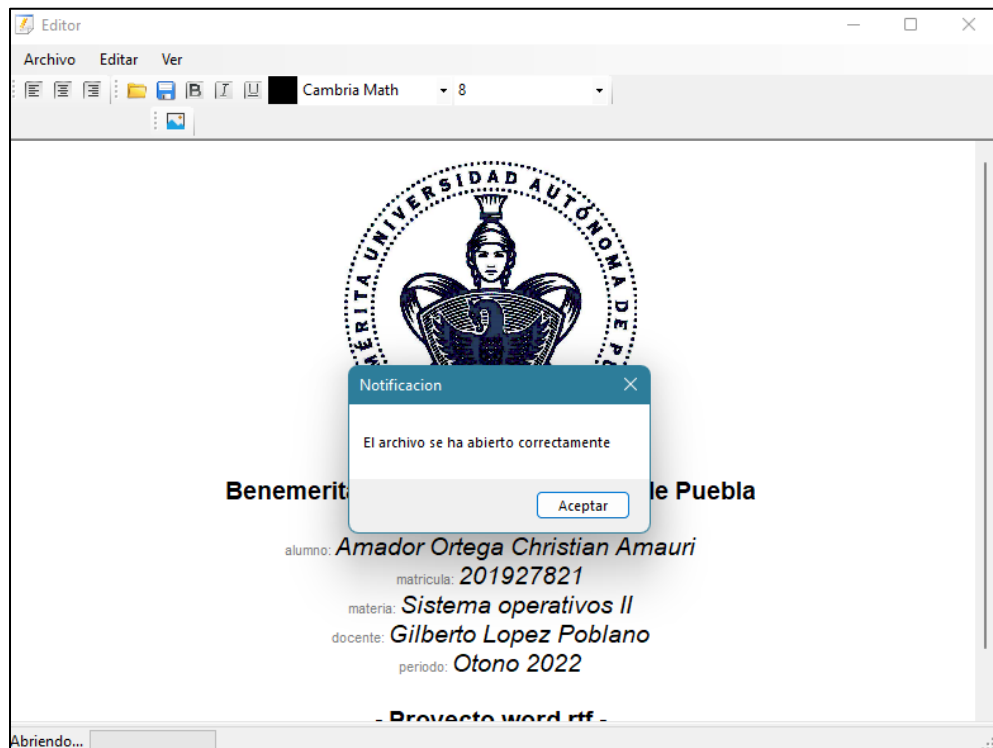
Ejemplo rápido del funcionamiento general del proyecto, para resumir:

- Podemos crear documentos con nuestro propio tipo de formato (BUAP files)
- Estos documentos consisten en textos modificables en color, fuente, estilo, y la opción de insertar imágenes.



- Tenemos la opción de guardar documentos de forma rápida, o específica.

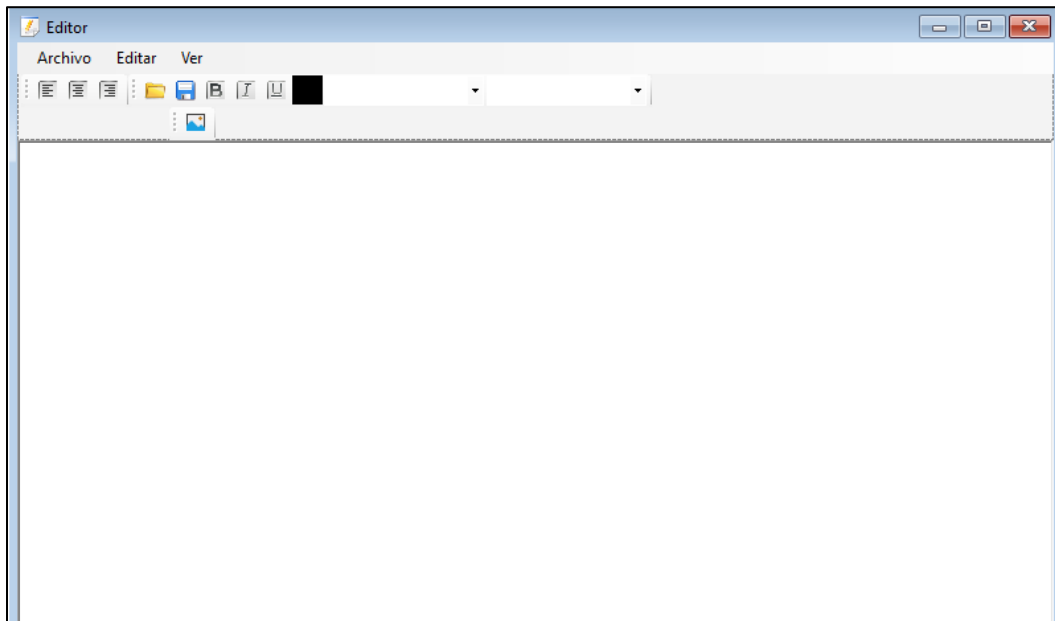




(Podemos guardar cambios sobre un mismo archivo predefinido. Como todos los editores de documentos de diversos formatos)

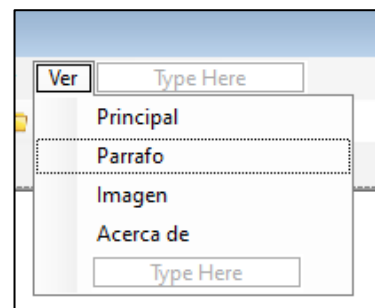
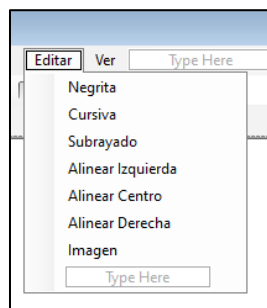
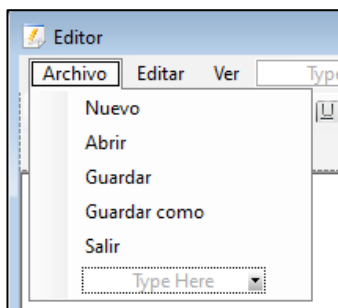
Ahora, los archivos .cs del proyecto (que nosotros creamos) corresponden a 2 diseños y 2 scripts. Comencemos por los diseños...

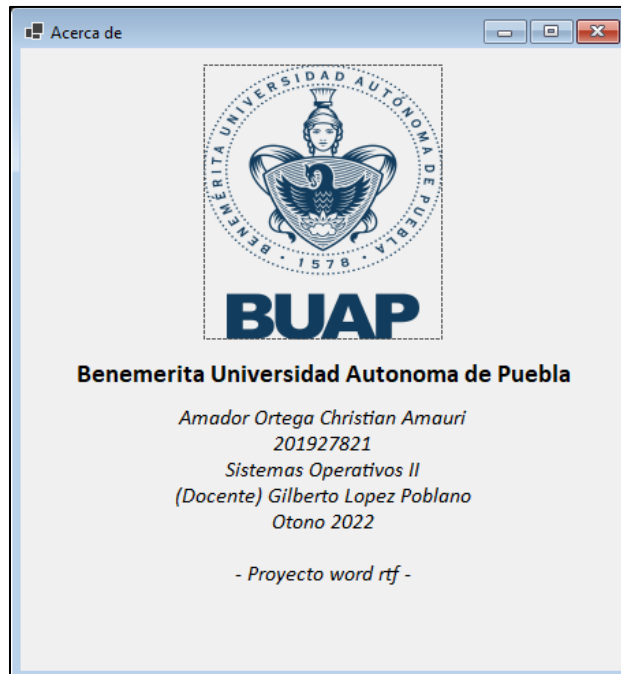
- Diseños -



#1 **Form1** (Form main)

Contiene los botones y las opciones con las que editaremos nuestros textos, tales como; Alinear a la izquierda, centro o derecha. Abrir archivo, Guardar archivo. establecer letra como: negrita, cursiva, subrayada. Cambiar el color y tamaño de la fuente. E insertar imágenes. Todas ellas corresponden a las funciones básicas de todo editor de textos / documentos.





#2 **Form2** (Acerca de)

Contiene información sobre el autor del código (no posee funcionalidad extra). Es invocada por la opción "Acerca de" en la opción "Ver" del menú strip de la ventana principal.

- Scripts -

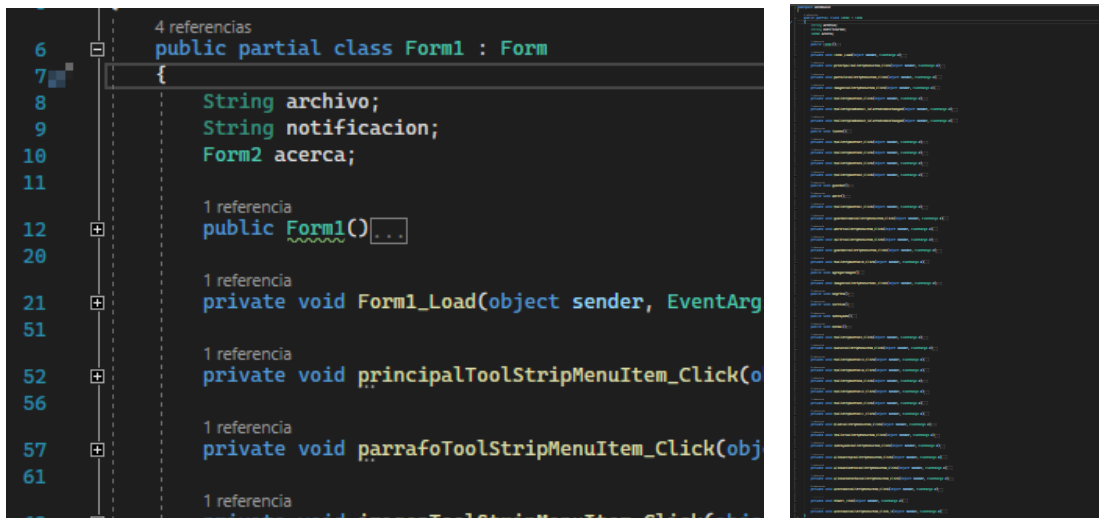
#1 *Form1.cs*

La estructura inicial de este script consta de:

- 2 librerías
- namespace definido como: wordNuevo
- public partial class Form1 : Form

```
1  using System.Drawing.Text;
2  using System.Text.RegularExpressions;
3
4  namespace wordNuevo
5  {
6      4 referencias
7      public partial class Form1 : Form
8      {
```

Dentro de Form1 : Form. se encuentran los siguientes componentes:



```
6  public partial class Form1 : Form
7  {
8      String archivo;
9      String notificacion;
10     Form2 acerca;
11
12     1 referencia
13     public Form1() ...
14
15     1 referencia
16     private void Form1_Load(object sender, EventArgs e)
17
18     1 referencia
19     private void principalToolStripMenuItem_Click(object sender, EventArgs e)
20
21     1 referencia
22     private void parrafoToolStripMenuItem_Click(object sender, EventArgs e)
23
24     1 referencia
25     private void imagenToolStripMenuItem_Click(object sender, EventArgs e)
26
27     43 referencias
28     private void ...
29
30     43 referencias
31     private void ...
32
33     43 referencias
34     private void ...
35
36     43 referencias
37     private void ...
38
39     43 referencias
40     private void ...
41
42     43 referencias
43     private void ...
44
45     43 referencias
46     private void ...
47
48     43 referencias
49     private void ...
50
51     43 referencias
52     private void ...
53
54     43 referencias
55     private void ...
56
57     43 referencias
58     private void ...
59
60     43 referencias
61     private void ...
62
63     43 referencias
64     private void ...
65
66     43 referencias
67     private void ...
68
69     43 referencias
70     private void ...
71
72     43 referencias
73     private void ...
74
75     43 referencias
76     private void ...
77
78     43 referencias
79     private void ...
80
81     43 referencias
82     private void ...
83
84     43 referencias
85     private void ...
86
87     43 referencias
88     private void ...
89
90     43 referencias
91     private void ...
92
93     43 referencias
94     private void ...
95
96     43 referencias
97     private void ...
98
99     43 referencias
100    private void ...
```

(de la imagen de la derecha, cada una de esas líneas es un método comprimido)

- 3 variables (tipo String y Form2)
- un constructor
- 43 métodos

Comencemos por las opciones de la opción “Archivo” del menu strip de la ventana principal:

El método `nuevoToolStripMenuItem_Click()` corresponde a la opción “Nuevo”

```
1 referencia
284 private void nuevoToolStripMenuItem_Click(object sender, EventArgs e)
285 {
286     richTextBox1.Clear();
287     archivo = "";
288     toolStripStatusLabel1.Text = "Creando nuevo archivo...";
289     notificacion = "El archivo se ha creado correctamente";
290     toolStripProgressBar1.Visible = true;
291     timer1.Start();
292 }
293
```

El método `abrirToolStripMenuItem_Click()` corresponde a la opción “abrir”. Manda a llamar al método `abrir()`, en donde se usa un `openFileDialog` para leer archivos con filtro .BUAP files:

```
2 referencias
132 public void abrir()
133 {
134     OpenFileDialog aArch = new OpenFileDialog();
135     aArch.DefaultExt = "*.buap";
136     aArch.Filter = "BUAP Files|*.buap";
137
138     if (aArch.ShowDialog() == DialogResult.OK)
139     {
140         archivo = aArch.FileName;
141         richTextBox1.LoadFile(aArch.FileName);
142         toolStripStatusLabel1.Text = "Abriendo...";
143         notificacion = "El archivo se ha abierto correctamente";
144         toolStripProgressBar1.Visible = true;
145         timer1.Start();
146     }
147 }
148
```

El método `guardarToolStripMenuItem_Click()` corresponde a la opción “Guardar”. está hecho para guardar de forma rápida sobre el archivo que se esté trabajando, pero si se está trabajando sobre un documento nuevo, se manda a llamar al método `guardar()`.

```
1 referencia
170 private void guardarToolStripMenuItem_Click(object sender, EventArgs e)
171 {
172     if (archivo == "")
173     {
174         guardar();
175     }
176     else
177     {
178         richTextBox1.SaveFile(archivo, RichTextBoxStreamType.RichText);
179         toolStripStatusLabel1.Text = "Guardando...";
180         notificacion = "El archivo se ha guardado correctamente";
181         toolStripProgressBar1.Visible = true;
182         timer1.Start();
183     }
184 }
185
```


El método `guardar()` funciona como cada método de guardado de archivos que hemos trabajado antes, Usa un filtro de formato de fichero para verificar que se guarden adecuadamente y mediante un `SaveFileDialog()` respaldado por un pequeño algoritmo de confirmación, realiza el proceso correspondiente:

```
113 3 referencias
114 public void guardar()
115 {
116     SaveFileDialog gArch = new SaveFileDialog();
117     gArch.DefaultExt = "*.buap";
118     gArch.Filter = "BUAP Files|*.buap";
119
120     if (gArch.ShowDialog() == DialogResult.OK)
121     {
122         archivo = gArch.FileName;
123         richTextBox1.SaveFile(gArch.FileName, RichTextBoxStreamType.RichText);
124         toolStripStatusLabel1.Text = "Guardando...";
125         notificacion = "El archivo se ha guardado correctamente";
126         toolStripProgressBar1.Visible = true;
127         timer1.Start();
128     }
129 }
130 }
```

El método `guardarComoToolStripMenuItem_Click()` corresponde a la opción “Guardar como”. Manda a llamar al método `guardar()` :

```
155 1 referencia
156 private void guardarComoToolStripMenuItem_Click(object sender, EventArgs e)
157 {
158     guardar();
}
```

El método `salirToolStripMenuItem_Click()` corresponde a la opción “Salir”. Este termina la ejecución de la aplicación:

```
165 1 referencia
166 private void salirToolStripMenuItem_Click(object sender, EventArgs e)
167 {
168     Application.Exit();
}
```

Sigamos con las opciones de la opción “Editar” del menú strip de la ventana principal. Estas son relativamente sencillas y parecidas, en algunos casos cada una manda a llamar a un sub-método que cumple su propósito (casi casi como meramente paso extra), de modo que por cada opción en este menú, tenemos dos métodos (el que llama al método real, y el que es llamado en primer lugar):

El método `blodToolStripMenuItem_Click()` manda a llamar al método `negrita()`, en donde se realiza el proceso real, el cual consiste simplemente en usar funciones predefinidas del lenguaje para darle a la fuente el efecto que deseamos:

```
342 1 referencia
343 private void blodToolStripMenuItem_Click(object sender, EventArgs e)
344 {
345     negrita();
}

218 2 referencias
219 public void negrita()
220 {
221     if (richTextBox1.SelectionFont != null)
222     {
223         if (richTextBox1.SelectionFont.Bold == true)
224         {
225             normal();
226         }
227         else
228         {
229             richTextBox1.SelectionFont = new Font(richTextBox1.SelectionFont.FontName,
230             richTextBox1.SelectionFont.Size, FontStyle.Bold);
231         }
232     }
233     richTextBox1.Focus();
}
```

Como dijimos, este concepto se repite para las opciones de cursiva, subrayada y agregar imagen:

```
347 1 referencia
348 private void italicToolStripMenuItem_Click(object sender, EventArgs e)
349 {
350     cursiva();
351 }

352 1 referencia
353 private void subrayadoToolStripMenuItem_Click(object sender, EventArgs e)
354 {
355     subrayado();
356 }

213 1 referencia
214 private void imagenToolStripMenuItem1_Click(object sender, EventArgs e)
215 {
216     agregarImagen();
217 }

192 2 referencias
193 public void agregarImagen()
194 {
195     OpenFileDialog iArch = new OpenFileDialog();
196     iArch.DefaultExt = "*.jpg";
197     iArch.Filter = "Image Files|*.jpg";
198
199     if (iArch.ShowDialog() == DialogResult.OK)
200     {
201         Bitmap imagen = new Bitmap(iArch.FileName);
202         Clipboard.SetDataObject(imagen);
203         DataFormats.Format formato = DataFormats.GetFormat(DataFormats.Bitmap);
204         richTextBox1.Paste(formato);
205         toolStripStatusLabel1.Text = "Cargando...";
206         notificacion = "La imagen se ha cargado correctamente";
207         toolStripProgressBar1.Visible = true;
208         timer1.Start();
209     }
210 }
211 }
```

Para las opciones de alinear a la derecha, centro o izquierda; se ejecuta el cambio en una simple línea de código, en un solo llamado:

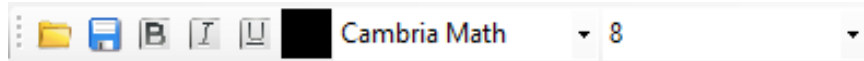
```
357 1 referencia
358 private void alinearIzqToolStripMenuItem_Click(object sender, EventArgs e)
359 {
360     richTextBox1.SelectionAlignment = HorizontalAlignment.Left;
361 }

362 1 referencia
363 private void alinearCentroToolStripMenuItem_Click(object sender, EventArgs e)
364 {
365     richTextBox1.SelectionAlignment = HorizontalAlignment.Center;
366 }

367 1 referencia
368 private void alinearDerechaToolStripMenuItem_Click(object sender, EventArgs e)
369 {
370     richTextBox1.SelectionAlignment = HorizontalAlignment.Right;
371 }
```

Finalmente en esta parte, las opciones de la opción “Ver”; La primeras tres opciones de esta opción corresponden a los métodos que establecen como visibles o NO visibles los tres conjuntos de íconos mostrados en la ventana principal. Esta acción se ejecuta en una sola línea, negando una vez el estado actual de visibilidad del conjunto de íconos:

- “Principal” corresponde a los íconos de color, tamaño y tipo de fuente:



- “Párrafo” corresponde a los íconos de alineado del texto:



- “Imagen” corresponde al ícono de imagen:



Métodos respectivos:

```
52 private void principalToolStripMenuItem_Click(object sender, EventArgs e)
53 {
54     toolStrip1.Visible = !toolStrip1.Visible;
55 }
56
57 1 referencia
58 private void parrafoToolStripMenuItem_Click(object sender, EventArgs e)
59 {
60     toolStrip2.Visible = !toolStrip2.Visible;
61 }
62 1 referencia
63 private void imagenToolStripMenuItem_Click(object sender, EventArgs e)
64 {
65     toolStrip3.Visible = !toolStrip3.Visible;
66 }
```

La opción `acercaDeToolStripMenuItem_Click_1()` “Acerca de” manda a llamar al método el cual crea un objeto de tipo `Form2`, el cual existe única y exclusivamente con el propósito de cubrir esta opción (mostrar información):

```
413 1 referencia
414 private void acercaDeToolStripMenuItem_Click_1(object sender, EventArgs e)
415 {
416     Form2 acerca = new Form2();
417     acerca.Show();
418 }
419
```

Ahora, para cada método de cada ícono en la ventana main, simplemente se manda a llamar el método correspondiente (otra vez), de ahí que el código parezca tan largo. En realidad no son tantos métodos como parecen, simplemente hay muchas formas de mandarlos a llamar...

```
1 referencia
329 private void toolStripButton5_Click(object sender, EventArgs e)
330 {
331     subrayado();
332 }
```

El método `Form1_Load()` es un método que se manda a llamar en el `Form.1Designer.cs`. Este sirve para agregar a la ventana principal los Items y características que esta usa (no hay mucho más que entender, esto es más técnico realmente):

```
21 1 referencia
22 private void Form1_Load(object sender, EventArgs e)
23 {
24     toolStripComboBox2.Items.Add(8);
25     toolStripComboBox2.Items.Add(9);
26     toolStripComboBox2.Items.Add(10);
27     toolStripComboBox2.Items.Add(11);
28     toolStripComboBox2.Items.Add(12);
29     toolStripComboBox2.Items.Add(14);
30     toolStripComboBox2.Items.Add(16);
31     toolStripComboBox2.Items.Add(18);
32     toolStripComboBox2.Items.Add(20);
33     toolStripComboBox2.Items.Add(22);
34     toolStripComboBox2.Items.Add(24);
35     toolStripComboBox2.Items.Add(26);
36     toolStripComboBox2.Items.Add(28);
37     toolStripComboBox2.Items.Add(36);
38     toolStripComboBox2.Items.Add(48);
39     toolStripComboBox2.Items.Add(72);
40
41     InstalledFontCollection ifc = new InstalledFontCollection();
42     FontFamily[] familia = ifc.Families;
43     for (int i = 0; i < ifc.Families.Length; i++)
44     {
45         toolStripComboBox1.Items.Add(ifc.Families[i].Name);
46     }
47
48     toolStripComboBox1.SelectedIndex = 41;
49     toolStripComboBox2.SelectedIndex = 0;
50 }
51
```

#2 *Form2.cs*

La estructura inicial de este script consta de:

- 9 librerías
- namespace definido como: wordNuevo
- public partial class Form2 : Form

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace wordNuevo
12 {
13     public partial class Form2 : Form
14     {
15         public Form2()
16         {
17             InitializeComponent();
18         }
19     }
20 }
21
```

Dentro de Form2 : Form. Simplemente se encuentra el constructor de la clase form2 (ventana de Acerca de) la cual es llamada cuando se presiona la opción “Acerca de” en la opción “Ver” del menú strip de la ventana principal.



Como comentario final, este proyecto es más didáctico que funcional (no debe ser tomado demasiado en serio). El resultado sigue estando muy pero muy por debajo de cualquier editor de texto o documentos. Si acaso podríamos hacer notas personales simples o algo por el estilo con esto. Y también debemos recordar que al estar construido enteramente con funciones definidas en el lenguaje, hacer una documentación detallada de cada parte del código se convertiría más bien en una guía del lenguaje. Esta documentación se reserva el derecho de mostrar únicamente detalles exteriores sobre la estructura de nuestro producto.