O evento *CheckStateChanged* apenas pode ser aplicado em caixas de verificação, já que os botões de opção não possuem um terceiro estado.

EXEMPLO

```
//Evento CheckedChanged
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
   MessagBox.Show("Alterou o valor da opção!");
}
//Evento CheckStateChanged
private void checkBox1_CheckStateChanged(object sender, EventArgs e)
{
   MessageBox.Show("Alterou a seleção!");
}
```

À semelhança dos controlos estudados anteriormente, estes objetos reconhecem os eventos *Click*, *DoubleClick*, *MouseMove*, *MouseHover*, *MouseEnter*, *MouseLeave*, *KeyDown*, *KeyUp*, *KeyPress*, *Resize*, *Move*, *Enter* e *Leave*.

6.12 EXERCÍCIOS RESOLVIDOS

Nesta secção são estudados, numa vertente teórico-prática, outros tipos de controlos, cujos métodos, propriedades e eventos não foram abordados nas secções anteriores.

6.12.1 SITES

Recorrendo a controlos do tipo *LinkLabel*, pretende-se embutir num formulário hiperligações para os seguintes *sites*:

- FCA (http://www.fca.pt); https://www.fca.pt
- Microsoft (http://www.microsoft.com); https://www.microsoft.com
- Google (http://www.google.com). https://www.google.com

Após o utilizador fazer clique sobre as hiperligações, os conteúdos deverão surgir embutidos num controlo do tipo *WebBrowser*. O formulário

deverá ser dividido em duas secções por intermédio de um *SplitContainer*. A secção esquerda conterá as hiperligações e a secção direita servirá para a apresentação das respetivas páginas.

RESOLUÇÃO

1) Crie um projeto seguindo o *template* **Windows Forms Application** (Aplicativo de Windows Forms);

2) Mantenha-se em modo de desenho com a janela de propriedades do formulário visível;

3) Faça com que o formulário surja maximizado quando é apresentado ao utilizador, alterando a propriedade *WindowState* para *Maximized*;

4) Divida o formulário em duas secções. Para isso, expanda a categoria **Containers** (Contêineres) presente na **Toolbox** (Caixa de Ferramentas) e arraste um controlo da categoria *SplitContainer* diretamente da caixa de controlos para o interior do formulário, de forma a obter dois painéis (Figura 6.53);

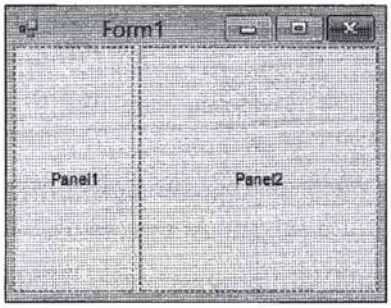

FIGURA 6.53 – Divisão de um formulário em dois painéis

Para alterar as dimensões dos painéis, faça clique com o botão direito do rato sobre uma das secções e escolha a opção **Select 'splitContainer1'** (Selecionar 'splitContainer1'). Depois, ajuste os dois objetos com o auxílio dos ponteiros de redimensionamento habituais.

5) Incorpore no painel esquerdo três *LinkLabels* para os *sites* FCA (*linkLabel1*), Microsoft (*linkLabel2*) e Google (*linkLabel3*);

6) Altere as respetivas propriedades *Text* até obter o aspeto apresentado na Figura 6.54;

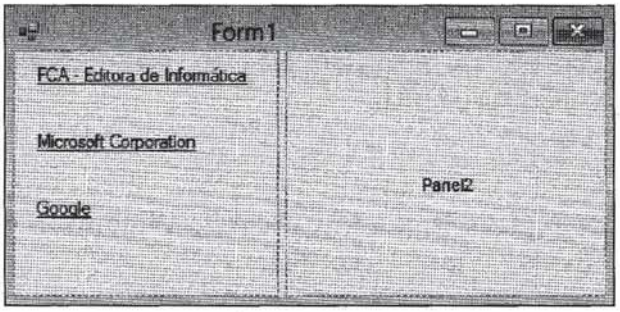

FIGURA 6.54 – Desenho de *LinkLabels*

Pode desenhar apenas um *LinkLabel*, selecioná-lo, copiá-lo para a área de transferência e colá-lo quantas vezes for necessário.

7) Para a apresentação das páginas *web*, inclua um controlo do tipo *WebBrowser* no painel direito (Figura 6.55);

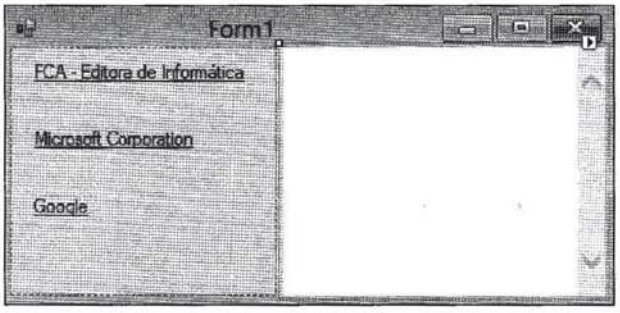

FIGURA 6.55 – Desenho de um controlo do tipo *WebBrowser*

8) Aceda ao módulo de programação do formulário e programe os eventos *LinkClicked* dos três *LinkLabels* da seguinte forma:

```
//Evento LinkClicked do primeiro link
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
   webBrowser1.Navigate("http://www.fca.pt");
}
//Evento LinkClicked do segundo link
private void linkLabel2_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
   webBrowser1.Navigate("http://www.microsoft.com");
}
```

```
//Evento LinkClicked do terceiro link
private void linkLabel3_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
   webBrowser1.Navigate("http://www.google.com");
}
```

9) Programe o evento *Load* do formulário com as instruções que se seguem:

```
private void Form1_Load(object sender, EventArgs e)
{
   int larguraForm;
   larguraForm = this.Width;
   splitContainer1.Panel1MinSize = 200;
   splitContainer1.Panel2MinSize = larguraForm - 200;
}
```

ANÁLISE

O evento *LinkClicked* ocorre sempre que o utilizador faz clique sobre a hiperligação. Para cada uma das hiperligações, recorreu-se ao método *Navigate* do objeto *WebBrowser* para atualizar o controlo de acordo com a página escolhida, por indicação do URL absoluto.

Quando o formulário surge no ecrã, é obtida a sua largura (valor da propriedade *Width*), cujo resultado é armazenado na variável numérica *larguraForm*. Para que o painel esquerdo não surja "cortado" foi definido um tamanho mínimo de 200 pixéis (deverá ajustar este valor em função da resolução do ecrã e dimensões dos controlos que utilizar). O painel direito fica com uma largura mínima igual à diferença entre a largura do formulário e a largura do painel esquerdo. Desta forma, consegue-se ocupar toda a área útil.

TESTE

Ao executar, o formulário aparece maximizado e não deverá mostrar qualquer página. Ao fazer-se clique sobre uma das hiperligações, o conteúdo do controlo *WebBrowser* deverá ser atualizado. A Figura 6.56 mostra o resultado do acesso a uma das páginas da Microsoft.

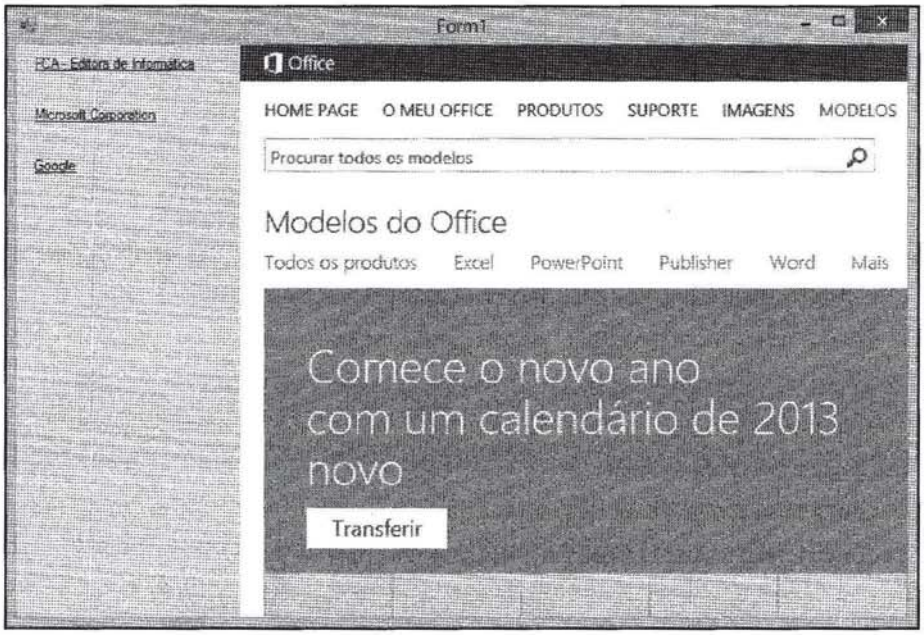

FIGURA 6.56 – Visualização de páginas *web*

Sirva-se das propriedades *LinkColor*, *VisitedLinkColor* e *ActiveLinkColor* para alterar as cores indicativas de quando a hiperligação ainda não tiver sido visitada, quando for visitada e quando for ativada, respetivamente.

6.12.2 FICHEIROS

Pretende-se criar uma aplicação capaz de aceder a uma determinada pasta, contendo um número indeterminado de ficheiros de imagem e mostrá-los (um a um) num controlo do tipo *PictureBox*. A escolha da imagem a ser visualizada é feita com a ajuda de um controlo do tipo *NumericUpDown*. Quando o utilizador posiciona o cursor do rato sobre a imagem apresentada, deverá surgir (em *tooltip*) o nome do ficheiro.

Os caminhos completos para todos os ficheiros serão guardados num *array* unidimensional.

RESOLUÇÃO

1) Crie um projeto do tipo **Windows Forms Application** (Aplicativo de Windows Forms);

2) Desenhe um controlo da categoria *PictureBox* e outro do tipo *NumericUpDown* como apresentado na Figura 6.57;

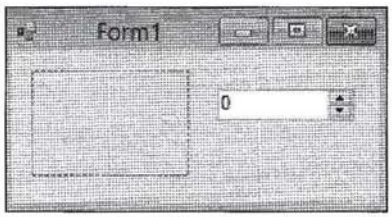

FIGURA 6.57 – Controlos *PictureBox* e *NumericUpDown*

3) Permita o autorredimensionamento das imagens, com a alteração da propriedade *BackgroundImageLayout* da *PictureBox* para *Stretch*;

4) Para apresentar informações adicionais aos utilizadores em estilo de dica, recorre-se a um controlo especial – o controlo *ToolTip*. Da **Toolbox** (Caixa de Ferramentas), arraste um objeto deste tipo até visualizá-lo na área inferior da estruturação do formulário;

Os controlos que não permitem uma interação direta utilizador/aplicação não são embutidos na área do formulário, mas sim à parte, numa secção inferior específica para agrupar esse conjunto de controlos especiais.

5) Por predefinição, a dica é apresentada sob a forma retangular. Para aplicar o estilo balão, selecione o controlo *ToolTip* e altere a sua propriedade *IsBalloon* para *true*;

6) Defina um título para a dica, alterando a propriedade *ToolTipTitle* para "Imagem";

7) É possível associar um ícone ao *ToolTip*. Escolha, a título de exemplo, o ícone de informação (valor *Info* na propriedade *ToolTipIcon*);

Pode ainda alterar as propriedades *BackColor* e *ForeColor* para definir outras cores de fundo e do tipo de letra usadas pelo controlo *ToolTip*.

8) Crie uma pasta com o nome **pictures** e inclua alguns ficheiros de imagem ao seu gosto;

Neste exemplo, foi criada a pasta **pictures** na raiz do sistema de ficheiros. Pode usar outro caminho para a pasta contendo as imagens, mas deverá alterar o código para que tudo funcione corretamente.

9) A *namespace System.IO* (*Input/Output*) define classes que implementam a manipulação do sistema de ficheiros do Windows, permitindo ainda obter diversas informações acerca dos ficheiros e pastas (caminho completo, tamanho, data de último acesso, etc.). Assim, proceda à importação da *System.IO* através de uma instrução `Imports`, como habitual:

```
using System.IO;
```

10) Programe o evento *ValueChanged* do controlo *NumericUpDown* com o conjunto de instruções que se seguem:

```
private void numericUpDown1_ValueChanged(object sender, EventArgs e)
{
   DirectoryInfo dirImagens = new DirectoryInfo("c:/pictures");
   FileInfo[] ficheiros;
   ficheiros = dirImagens.GetFiles();
   int numFicheiros;
   numFicheiros = ficheiros.Length;
   numericUpDown1.Minimum = 1;
   numericUpDown1.Maximum = numFicheiros;
   int numFichEscolhido;
   numFichEscolhido = (int) numericUpDown1.Value - 1;
   string caminhoFichEscolhido;
   caminhoFichEscolhido = ficheiros[numFichEscolhido].FullName;
   pictureBox1.BackgroundImage =
   Image.FromFile(caminhoFichEscolhido);
   toolTip1.SetToolTip(pictureBox1, caminhoFichEscolhido);
}
```

ANÁLISE

Sempre que ocorre mudança de valor (evento *ValueChanged*) deverá ser executada, na caixa *numericUpDown1*, uma série de instruções que façam com que a imagem escolhida apareça no ecrã.

A variável *dirImagens* aponta para um objeto *DirectoryInfo*, que faz referência a um diretório do sistema operativo. A construção desta classe é feita por especificação do caminho completo até à pasta **pictures**. A variável *ficheiros* (classe *FileInfo*) consiste num *array* de objetos que agrupa todos os ficheiros

existentes na pasta de imagens. O método *GetFiles* da classe *DirectoryInfo* retorna esse conjunto de ficheiros.

Como o número de ficheiros à partida é desconhecido, declarou-se o *array* sem limite.

A contagem de ficheiros é necessária para, posteriormente, se definir o limite mínimo (propriedade *Minimum*) e máximo (propriedade *Maximum*) para o controlo *numericUpDown1*. O número de ficheiros existentes é obtido à custa da propriedade *Length* do *array* de ficheiros. A variável *numFichEscolhido* foi declarada com o objetivo de se obter o índice de ficheiro a ser mostrado. Foi obtida através da propriedade *Value* do controlo *numericUpDown1* que retorna o valor escolhido pelo utilizador em determinado ponto de execução do programa; como as imagens se encontram num *array* e como a primeira imagem é a de índice 0, subtraiu-se uma unidade ao valor encontrado.

A variável *caminhoFichEscolhido* contém o caminho completo até ao ficheiro de imagem escolhido (recorreu-se à propriedade *FullName* do índice de imagem atual). Depois, definiu-se a imagem de fundo (propriedade *BackgroundImage*) do controlo *PictureBox*, através do método *FromFile*, como estudado anteriormente. Por fim, aplicou-se a dica através do método *SetToolTip*, por especificação do controlo responsável por mostrá-la e pelo texto associado (o caminho completo do ficheiro).

TESTE

Execute o formulário. Quando este surge no ecrã pela primeira vez, nenhuma imagem está disponível e o controlo *NumericUpDown* deverá conter 0. À medida que for alterando o número de imagem, os ficheiros vão sendo apresentados, bem como o *ToolTip* correspondente (Figura 6.58).

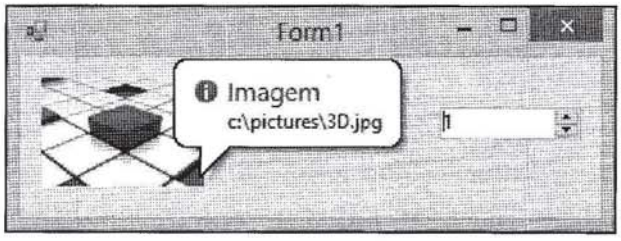

FIGURA 6.58 – Visualização de imagens com *tooltips* associados

O índice para cada imagem é atribuído seguindo uma ordenação alfabética.

6.12.3 DATAS

Um formulário deverá permitir a introdução de uma data a partir de duas origens distintas:

- Através do controlo *MonthCalendar*;
- Através de um controlo *MaskedTextBox* que aceite datas no formato #dd-mm-aaaa# (dois dígitos obrigatórios para o dia, dois dígitos obrigatórios para o mês e quatro dígitos obrigatórios para o ano, com o hífen como separador).

Obtida uma data correta, a mesma deverá ser apresentada por extenso num controlo do tipo *DateTimePicker*.

Neste exemplo em concreto, o controlo *DateTimePicker* não pode permitir edição.

RESOLUÇÃO

1) Crie um projeto do tipo **Windows Forms Application** (Aplicativo de Windows Forms);
2) Desenhe os controlos *MonthCalendar*, *MaskedTextBox* e *DateTimePicker* como demonstrado na Figura 6.59;

FIGURA 6.59 – Controlos *MonthCalendar*, *MaskedTextBox* e *DateTimePicker*

3) Uma vez que o controlo *DateTimePicker* deve ser bloqueado para edição, selecione-o e altere a sua propriedade *Enabled* para *false*;

A propriedade *ReadOnly* não existe em controlos do tipo *DateTimePicker*.

4) Defina uma máscara de introdução para o controlo *MaskedTextBox* de modo a que aceite apenas datas no formato pretendido. Para aplicar a máscara, recorra à propriedade *Mask* e escolha **Short date** da caixa de diálogo **Input Mask** (Figura 6.60);

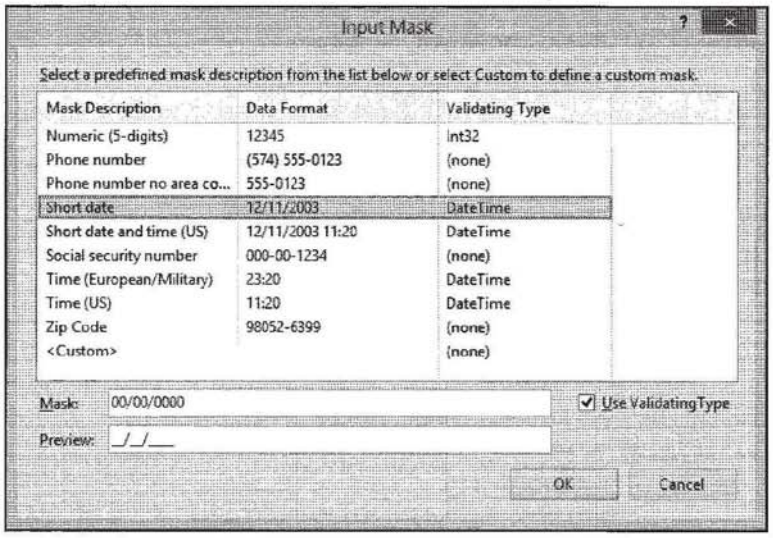

FIGURA 6.60 – Seleção de uma máscara de introdução predefinida

Consulte na secção A.4 do Anexo, no final deste livro, a simbologia utilizada na formação de máscaras de introdução personalizadas.

5) Confirme a criação da máscara em **OK**;

Pode alterar o caráter de posição através da propriedade *PromptChar*. Por predefinição, é usado o *underscore* (_).

6) O controlo *MonthCalendar* permite selecionar mais do que uma data. Como o objetivo é apenas a seleção de uma, deverá alterar a propriedade *MaxSelectionCount* de 7 (valor predefinido) para 1;

7) Programe o evento *DateChanged* do controlo *MonthCalendar*:

```
private void monthCalendar1_DateChanged(object sender,
DateRangeEventArgs e)
{
   dateTimePicker1.Value = monthCalendar1.SelectionStart;
}
```

8) Crie o procedimento de evento *TextChanged* da caixa de texto com validação (controlo *MaskedTextBox*) de acordo com as instruções que se seguem:

```
private void maskedTextBox1_TextChanged(object sender, EventArgs e)
{
   int dia, mes, ano;
   string textoData;
   DateTime data;
   if (maskedTextBox1.MaskCompleted == true)
   {
      textoData = maskedTextBox1.Text;
      dia = int.Parse(textoData.Substring(0, 2));
      mes = int.Parse(textoData.Substring(3, 2));
      ano = int.Parse(textoData.Substring(6,4));
      try
      {
          data = DateTime.Parse(dia + "-" + mes + "-"  +ano);
          dateTimePicker1.Value = data;
          monthCalendar1.SelectionStart = data;
      }
      catch
      {
         MessageBox.Show("Data incorreta!", "Erro!",
         MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
   }
}
```

ANÁLISE

O evento *DateChanged* ocorre sempre que o utilizador altera a data apresentada no controlo supracitado. É constituído apenas por uma instrução que permite obter a data selecionada (propriedade *SelectionStart*) e aplicá-la no controlo *dateTimePicker1* por redefinição da sua propriedade *Value*. Sempre que o utilizador altera o texto contido neste controlo, ocorre o evento *TextChanged*. Neste caso, recorreu-se à propriedade booleana *MaskCompleted* para determinar o momento exato em que todos os carateres obrigatórios se encontram formados no controlo *maskedTextBox1*. Após esta verificação, obtiveram-se as diferentes partes que constituem a data (dia, mês e ano) e tentou-se formar uma data válida. Se a data introduzida tiver sido válida, o controlo *dateTimePicker1* é atualizado; caso contrário, é apresentada uma mensagem de erro ao utilizador.

A verificação da data foi feita através de um *bloco try catch*. A utilização desta estrutura permite aos programadores gerirem através de código-fonte os erros que poderão ocorrer em tempo de exucução (por exemplo: se o utilizador escrever uma data incorreta, o programa não termina abruptamente e é dirigido para a cláusula *catch*).

TESTE

1) Execute o formulário. De início, deverá surgir a data atual nos controlos *monthCalendar1* e *dateTimePicker1* (a caixa de texto mostrará somente os carateres de introdução);
2) Selecione uma data diferente no controlo *MonthCalendar*. O controlo *dateTimePicker1* deverá ser atualizado com a nova data;
3) Forme uma data incorreta no controlo *maskedTextBox1* ;
4) Deverá aparecer a caixa de erro;
5) Forme uma data correta no controlo *maskedTextBox1* ;
6) Os controlos *dateTimePicker1* e *monthCalendar1* serão atualizados.

6.12.4 FORMATOS

Pretende-se criar um formulário que permita a alteração do tipo de letra, tamanho e estilo para o texto contido num controlo da categoria *RichTextBox*. Para isso, teremos três controlos do tipo *ComboBox* para cada tipo de formatação.

Os tipos de letra possíveis serão:

- Arial;
- Times New Roman;
- Verdana.

Os tamanhos a utilizar deverão ser:

- 10;
- 12;
- 14.

Os estilos disponíveis serão:

- Normal;
- Negrito;
- Itálico;
- Sublinhado.

O formato predefinido é Arial, tamanho 10, estilo normal, e deve ser aplicado após a seleção do texto a ser formatado e alteração de uma das opções disponíveis nos controlos *ComboBox*. Caso o utilizador não selecione qualquer porção de texto, este deverá surgir com as opções que estiverem selecionadas no momento.

RESOLUÇÃO

1) Crie um projeto do tipo **Windows Forms Application** (Aplicativo de Windows Forms);

2) Desenhe o controlo *RichTextBox*, as três *ComboBoxes* e as três *Labels* informativas;

3) Defina as respetivas propriedades *Text* dos rótulos como apresentado na Figura 6.61;

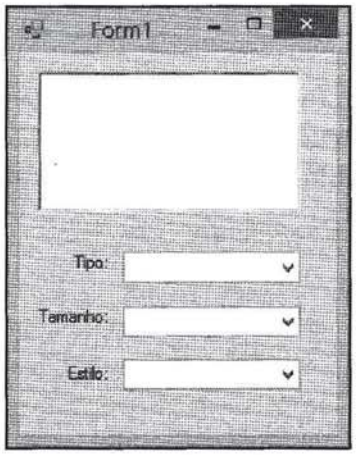

FIGURA 6.61 – Controlo *RichTextBox* associado a três controlos da categoria *ComboBox*

4) Altere o estilo (propriedade *DropDownStyle*) dos três controlos da categoria *ComboBox* para *DropDownList* (o utilizador apenas pode selecionar uma opção por caixa e não aplicar novos tipos de letra, tamanhos e estilos por edição de texto);

5) O evento *Load* do formulário é responsável pelo carregamento das caixas de combinação e pela aplicação do formato predefinido:

```
private void Form1_Load(object sender, EventArgs e)
{
    //Carregamento dos tipos de letra
    comboBox1.Items.Add("Arial");
```

```
    comboBox1.Items.Add("Times New Roman");
    comboBox1.Items.Add("Verdana");
    comboBox1.Text = "Arial"; //Tipo de letra predefinido
   //Carregamento dos tamanhos
    comboBox2.Items.Add("10");
    comboBox2.Items.Add("12");
    comboBox2.Items.Add("14");
    comboBox2.Text = "10"; //Tamanho predefinido
    //Carregamento dos estilos
    comboBox3.Items.Add("Normal");
    comboBox3.Items.Add("Negrito");
    comboBox3.Items.Add("Itálico");
    comboBox3.Items.Add("Sublinhado");
    comboBox3.Text = "Normal"; //Estilo predefinido
}
```

6) Como a aplicação de formatos será realizada mediante a alteração de valores em controlos diferentes, faz sentido criar-se um método de forma a evitar a repetição exaustiva de código:

```
void Formatar()
{
    //Verificação do tipo de letra selecionado
    string tipo;
    tipo = comboBox1.Text;
    //Verificação do tamanho selecionado
    int tamanho;
    tamanho = int.Parse(comboBox2.Text);
    //Verificação do estilo selecionado
    FontStyle estilo = new FontStyle();
    switch (comboBox3.Text)
    {
        case "Normal":
            estilo = FontStyle.Regular;
            break;
```

```
        case "Negrito":
            estilo = FontStyle.Bold;
            break;
        case "Itálico":
            estilo = FontStyle.Italic;
            break;
        case "Sublinhado":
            estilo = FontStyle.Underline;
            break;
    }
    //Aplicação do formato
    richTextBox1.SelectionFont = new Font(tipo, tamanho, estilo);
}
```

7) Os eventos *SelectionIndexChanged* dos três controlos da categoria *ComboBox* são exatamente iguais:

```
//Caixa para o tipo de letra
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (richTextBox1.SelectionLength > 0)
    {
        Formatar();
    }
}
//Caixa para o tamanho
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
    if (richTextBox1.SelectionLength > 0)
    {
        Formatar();
    }
}
//Caixa para o estilo
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
{
```

```
    if (richTextBox1.SelectionLength > 0)
    {
        Formatar();
    }
}
```

8) Por fim, programe o evento *TextChanged* da *RichTextBox*:

```
private void richTextBox1_TextChanged(object sender, EventArgs e)
{
    Formatar();
}
```

ANÁLISE

Como as caixas de combinação produzem valores do tipo *string*, recorreu-se ao método *Parse* para converter a cadeia de carateres obtida na *comboBox2* para um valor numérico inteiro correspondente ao tamanho do tipo de letra a ser aplicado. Foi também declarada a variável *estilo* como sendo do tipo *FontStyle*, que representa um dos estilos possíveis de aplicar (negrito, itálico, etc.); a conversão do texto proveniente da *comboBox3* foi feita à custa de um bloco *switch*. A aplicação de formatos ao texto atualmente selecionado é feita por redefinição da propriedade *SelectionFont*. Quando ocorre alteração de texto, deve ser aplicado o formato atual (ditado pelas três caixas de combinação). Por outro lado, quando uma porção de texto está selecionada e o utilizador altera um dos valores de qualquer uma das caixas de combinação, a alteração de formato também deve ser efetuada. Por esse motivo é que o evento *TextChanged* da *RichTextBox* efetua sempre uma chamada ao método *Formatar*, enquanto os diferentes eventos *SelectedIndexChanged* das caixas de combinação apenas aplicam novos formatos quando algo está selecionado (avaliação da propriedade *SelectionLength*).

TESTE

Execute o formulário e simule os dois modos de aplicação de formatos (com e sem seleção de texto).

As funcionalidades dos controlos *TreeView*, *ListView*, *ProgressBar* e *NotifyIcon* serão aprofundadas mais adiante neste livro.

6.13 EXERCÍCIOS PROPOSTOS

Resolva os exercícios que se seguem, implementando-os em C# através do *template* **Windows Forms Application** (Aplicativo de Windows Forms).

1) Considere o programa desenvolvido na secção 6.12.1. Altere-o de forma que quando o utilizador passar o cursor do rato sobre determinada hiperligação, a mesma apareça em fundo laranja (efeito *highlight*). Quando o utilizador mover o cursor do rato para outra zona, a cor de fundo do controlo deverá desaparecer;

2) Modifique o formulário criado no exercício anterior, substituindo os três controlos *LinkLabel* por rótulos normais (controlos *Label*);

O aspeto gráfico deverá permanecer exatamente igual (cores dos *links*, sublinhados e efeitos *highlight*).

3) Considere o programa desenvolvido na secção 6.12.2. O que acontece quando inclui um ficheiro que não é do tipo imagem, logo, incapaz de ser mostrado como uma imagem no controlo *PictureBox*? Simule o erro, copiando para a pasta **pictures** alguns ficheiros de outros tipos, tais como documentos do Office e de texto e altere o código para que o programa continue a apresentar imagens;

Utilize um bloco *try catch*.

4) Resolva o exercício anterior apresentando os nomes dos ficheiros em vez dos valores numéricos;

Substitua o controlo *NumericUpDown* por um do tipo *ListBox*.

5) Considere o programa desenvolvido na secção 6.12.3. Substitua o controlo *MaskedTextBox* por um controlo *TextBox* (caixa de texto simples), mas que implemente todas as funcionalidades do controlo substituído (à exceção da visualização do caráter de introdução). Os separadores (hífenes) deverão surgir automaticamente em função da posição atual do cursor do teclado e do número de dígitos atualmente formados na caixa;

Recorra ao evento *KeyPress* da *TextBox* para ir verificando (caráter a caráter) o tipo de informação que o utilizador vai introduzindo.