



Conteúdo

I - Web forms	3
Criar projeto C#, Windows Form App.....	3
Operação com números inteiros	4
Operações com textboxes.....	4
Exercício da Pizza – parte 1 (combobox e textbox)	4
Exercício da pizza – parte 2	5
Exercício da pizza – parte 3 (checkbox)	6
Exercício da pizza – parte 4 (radio button)	7
Operações com listbox – parte 1	8
Operações com listbox – parte 2	9
Comboboxes – manipulação de items e operação com strings	10
II - Formulário e controlos - Miniprojeto Shrek	11
III - Projeto “Nomes & Notas”, com listas	12
a) inicializar as listas	12
b) encontrar a maior avaliação	12
c) calcular a média	13
d) contar negativas	13
e) filtrar com combobox.....	13
f) filtrar também com radio button.....	14
g) remover os parêntesis e sexos, dos nomes da lista 3.....	15
IV - Projeto “Nomes & Notas”, com datagridview	15
a) gridview com 5 colunas	15
b) inicializar com 20 registos (descarregar anexo).....	16
c) o controlo menustrip (construção de menu e submenu)	16
d) contar as negativas, informar na lista.....	16
e) freguesia mais representada	17
f) encontrar melhor formando	17
g) gerar as avaliações aleatoriamente.....	18
h) salientar com cores, as avaliações negativas.....	19
V – Recapitulação: form com grid (lista de produtos)	19
a) problema proposto	19
b) resolução ponto 1: quantos produtos?.....	20
c) resolução ponto 2: preço mais elevado?	20
d) resolução ponto 3: contagem de produtos por categoria (com textbox)	20
d) resolução ponto 3: contagem de produtos por categoria (com combobox).....	21
e) resolução ponto 4: soma das existências	21
f) resolução ponto extra: contagem de produtos com filtro em radio buttons	22
OUTROS EXERCÍCIOS	24
Utilizando o try / catch.....	24
Validação de número inteiro – exemplo com int.TryParse	24
Pequenas operações de validação	26
Manipulação de combobox e operação com strings.....	27
Construtor e destrutor	28



Geração de outros controlos por codificação	29
Geração de listbox por codificação	32
Operações com strings	32
Exercício extra – botões de minicalculadora.....	34
Manipulação de listbox e array	35
SEGUNDA PARTE (COM BASE DE DADOS SQL SERVER)	36
Contexto e objetivo do problema proposto	36
Planeamento.....	36
Analisar a connection string.....	37
Pequenas consultas em SQL	39
A função Conecta ()	39
Ligar uma listbox à tabela de produtos	40
Adicionar um form novo.....	40
Definir qual o form que é executado	41
Ligar uma gridview à tabela de produtos no novo form	41
SQL – treinar consultas ativas	41
Layout de form pretendido para inserir, remover, alterar registo em tabela	42
Limpar e preencher a grid	42
Apagar um registo.....	42
Alterar um registo existente	42
Inserir um registo	43
Porquê usar a MessageBox.Show com a string SQL?	43
Contar registos da tabela (consulta SQL de agregação).....	43
Fechar aplicação.....	44
Invocar outro form	44
Inserir um produto (4 campos)	44
Estrutura da classe.....	46
Conclusão.....	46

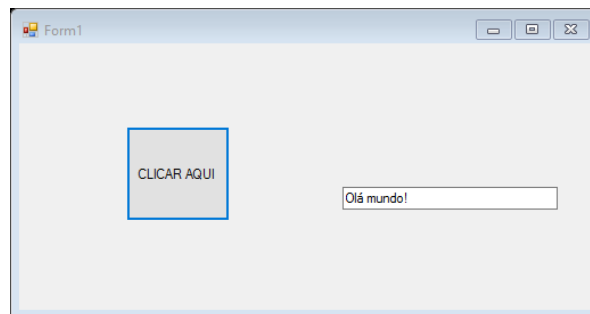
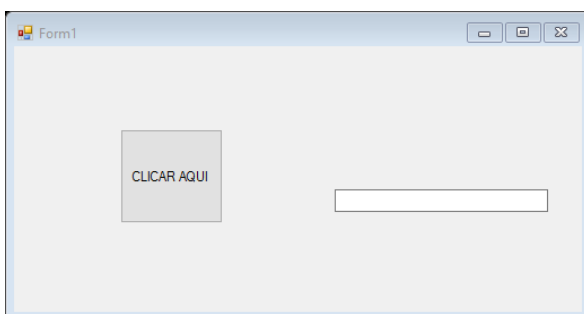
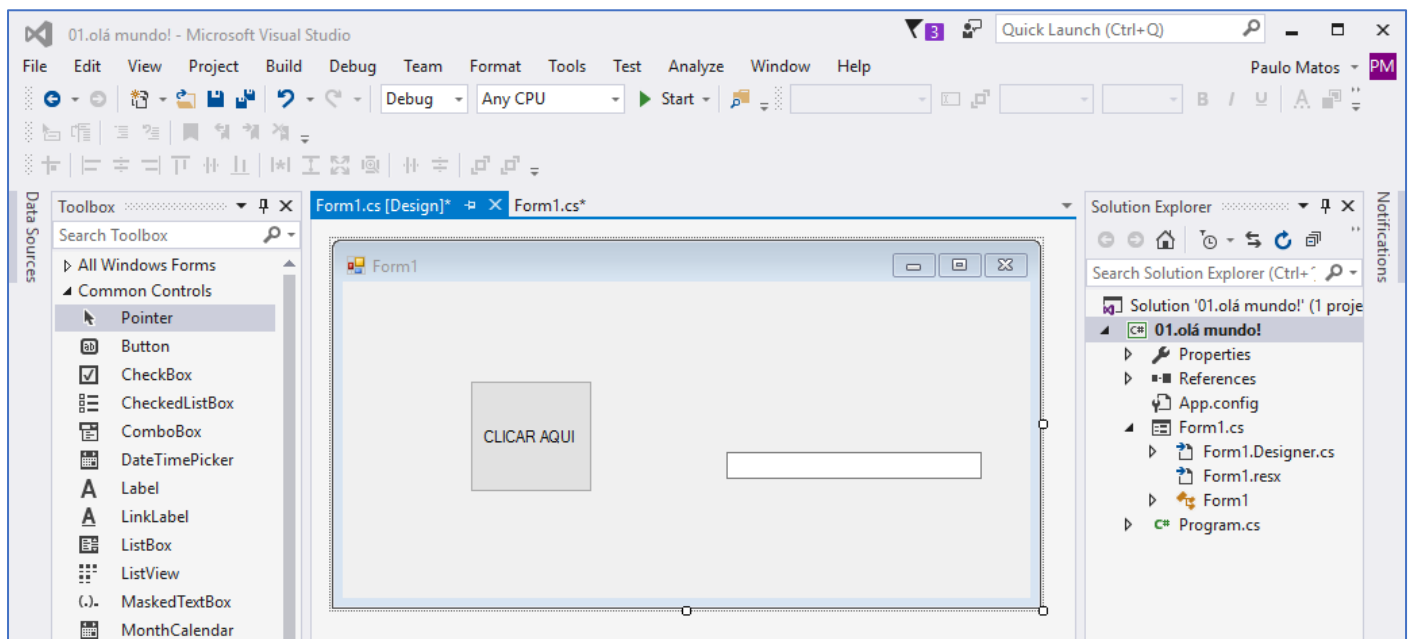
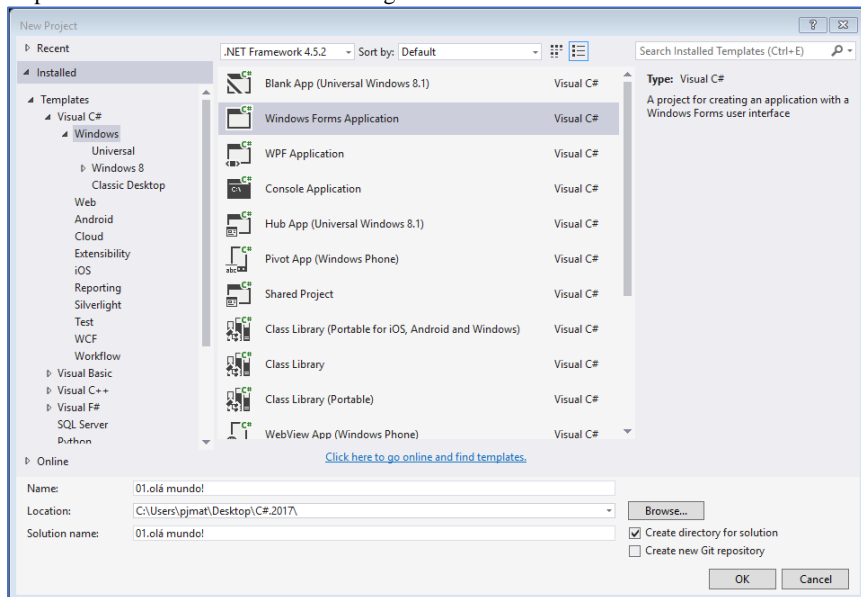


I - Web forms

Criar projeto C#, Windows Form App.

Adicionar um button e uma textbox.

Duplo-clicar no botão e escrever o código



```
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Text = "Olá mundo!";
}
```



Operação com números inteiros

Form com 3 textboxes e um botão para chamar o método.

```
public partial class Form1 : Form
{
    public Form1()
    {
        //inicialização das textboxes ao carregar o form
        InitializeComponent();
        tb1.Text = "0";
        tb2.Text = "0";
        tb3.Text = "";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        //código passo-a-passo
        int x, y, s;
        x = Convert.ToInt16(tb1.Text);
        y = Convert.ToInt16(tb2.Text);
        s = x + y;
        tb3.Text = Convert.ToString(s);
    }
}
```

Operações com textboxes

Fazer as contas das 3 textboxes azuis...

```
private void button1_Click(object sender, EventArgs e)
{
    int linha1, linha2;
    linha1 = Convert.ToInt32(qd1.Text) * Convert.ToInt32(pu1.Text);
    linha2 = Convert.ToInt32(qd2.Text) * Convert.ToInt32(pu2.Text);

    t11.Text = Convert.ToString(linha1);
    t12.Text = Convert.ToString(linha2);
    tf.Text = Convert.ToString(linha1 + linha2);
}
```

Exercício da Pizza – parte 1 (combobox e textbox)

Formulário da encomenda da pizza – parte 1

(neste exercício a estrutura de if's não é formalmente correta, mas perfeitamente aceitável)

```
private void button1_Click(object sender, EventArgs e)
{
    //declaração de variáveis:
```



```
string pizza_escolhida;
double preço=0;
int quantidade;
double valor;

//verificar qual é a pizza que foi escolhida e atribuir um preço:
pizza_escolhida = comboBox1.Text;
if (pizza_escolhida=="Napolitana") { preço = 10; }
if (pizza_escolhida == "Margarida") { preço = 8; }
if (pizza_escolhida == "Anchovas") { preço = 9; }

ctx_preço_pizza.Text = preço.ToString();

//buscar a quantidade e converter para inteiro:
quantidade = Convert.ToInt32(ctx_quantidade_pizza.Text);

valor = preço * quantidade;

//enviar valor da conta para o form:
ctx_valor_da_linha_da_pizza.Text = valor.ToString();
}
```

Exercício da pizza – parte 2

Pizza	Preço da pizza	Quantidade	
Napolitana	10	2	20

Bebida	Preço da bebida	Quantidade	
Cerveja	1,5	2	3

CALCULAR

```
//declaração de variáveis:
string pizza_escolhida;
string bebida_escolhida;
double preço=0;
int quantidade;
double valor;

//verificar qual é a pizza que foi escolhida e atribuir um preço:
pizza_escolhida = comboBox1.Text;
if (pizza_escolhida=="Napolitana") { preço = 10; }
. . .

//verificar qual é a bebida que foi escolhida e atribuir um preço:
bebida_escolhida = comboBox2.Text;
if (bebida_escolhida == "Coca Cola") { preço = 1.2; }
if (bebida_escolhida == "Cerveja") { preço = 1.5; }
if (bebida_escolhida == "Sumo") { preço = 1; }

ctx_preço_da_bebida.Text = preço.ToString();

//buscar a quantidade e converter para inteiro:
quantidade = Convert.ToInt32(ctx_quantidade_bebida.Text);
```



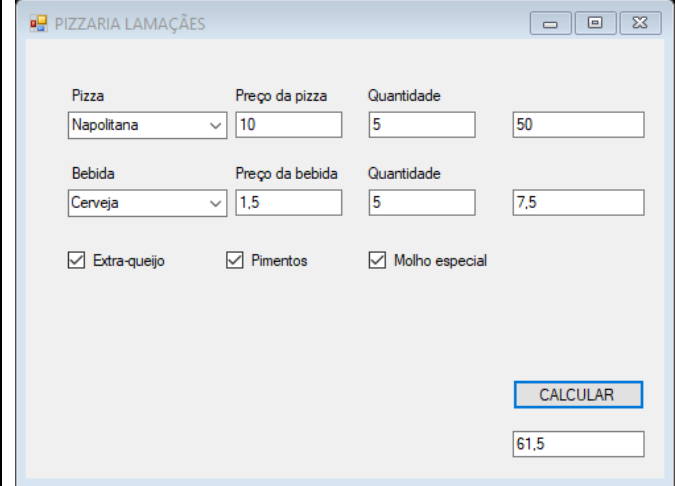
```
valor = preço * quantidade;
```

```
//enviar valor da conta para o form:
```

```
ctx_valor_da_linha_da_bebida.Text = valor.ToString();
```

Exercício da pizza – parte 3 (checkbox)

- inclusão das checkboxes e contas
- inicialização dos controlos no evento Form Load()



```
//declaração de variáveis:  
string pizza_escolhida = "";  
string bebida_escolhida = "";  
double preço_pizzas=0;  
double preço_bebidas = 0;  
int quantidade_pizzas=0;  
int quantidade_bebidas = 0;  
double valor_das_pizzas=0;  
double valor_das_bebidas=0;  
double valor_dos_extras = 0;  
double soma = 0;
```

Note que foram adicionadas variáveis e trocados alguns identificadores (nomes das variáveis).

Para apurar os extras:

```
if (chk_extra_queijo.Checked == true) valor_dos_extras = valor_dos_extras + 1; //adiciona um euro  
if (chk_pimentos.Checked == true) valor_dos_extras = valor_dos_extras + 2; //adiciona dois euros  
//a linha seguinte também está correta; porquê?  
if (chk_molho_especial.Checked) valor_dos_extras = valor_dos_extras + 1; //adiciona um euro
```

```
//finalmente, apurar as contas e enviar para o form:
```

```
soma = valor_das_bebidas + valor_das_pizzas + valor_dos_extras;  
ctx_VALOR_FINAL.Text = soma.ToString();
```

Para inicializar os controlos ao carregar o formulário:

```
private void Form1_Load(object sender, EventArgs e)  
{  
    //inicialização das textboxes; porquê fazer isto aqui?  
    ctx_preço_da_bebida.Text = "0";  
    ctx_preço_pizza.Text = "0";  
    ctx_quantidade_bebida.Text = "0";  
    ctx_quantidade_pizza.Text = "0";  
    ctx_valor_da_linha_da_bebida.Text = "0";  
    ctx_valor_da_linha_da_pizza.Text = "0";  
}
```

Estrutura do programa, neste momento (3 métodos):

```
namespace Pizza  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
        private void Form1_Load(object sender, EventArgs e)  
        {  
            . . . (código para inicializar mostrado acima)  
        }  
        private void button1_Click(object sender, EventArgs e)
```



```
{  
    . . .  
}  
}
```

Exercício da pizza – parte 4 (radio button)

- Inclusão dos radio buttons e apuramento de desconto (10% de desconto)
- Inicialização (valor pré-definido) das comboboxes

```
//finalmente, apurar as contas e enviar para o form:  
soma = valor_das_bebidas + valor_das_pizzas + valor_dos_extras;  
  
//para apurar o desconto (radio buttons):  
if (rbt_com.Checked) soma = soma - soma * 0.1;  
  
ctx_VALOR_FINAL.Text = soma.ToString();
```

A linha do radio button podia ser:

```
if (rbt_com.Checked == true) soma = soma - soma * 0.1;
```

O significado é o mesmo; o resultado será o mesmo.

Para inicializar as comboboxes (assim que abre o formulário):

```
private void Form1_Load(object sender, EventArgs e)  
{  
    //inicialização das textboxes; porquê fazer isto aqui?  
    ctx_preço_da_bebida.Text = "0";  
    . . .  
  
    comboBox1.SelectedItem = "Napolitana";  
    comboBox2.SelectedItem = "Cerveja";  
}
```



Operações com listBox – parte 1

Linhas de código eventualmente úteis

```
//elimina todos os nomes:
listBox1.Items.Clear();

listBox1.Items.Add("Carlos");

//remove o nome:
string s = txt_remove_nome.Text;
listBox1.Items.Remove(s);

//se nome existe, informa na textbox por baixo:
string s = txt_procura_nome.Text;
int i = listBox1.FindString(s);
//i fica com a posição na lista; atenção, começa em ZERO
//se não existir, i fica com -1:
if (i == -1) txt_resultado_sim_ou_nao.Text = "Não existe";
else txt_resultado_sim_ou_nao.Text = "Existe";
//mostra numa messagebox, a posição do nome na lista:
MessageBox.Show(i.ToString());
```




Operações com listbox – parte 2

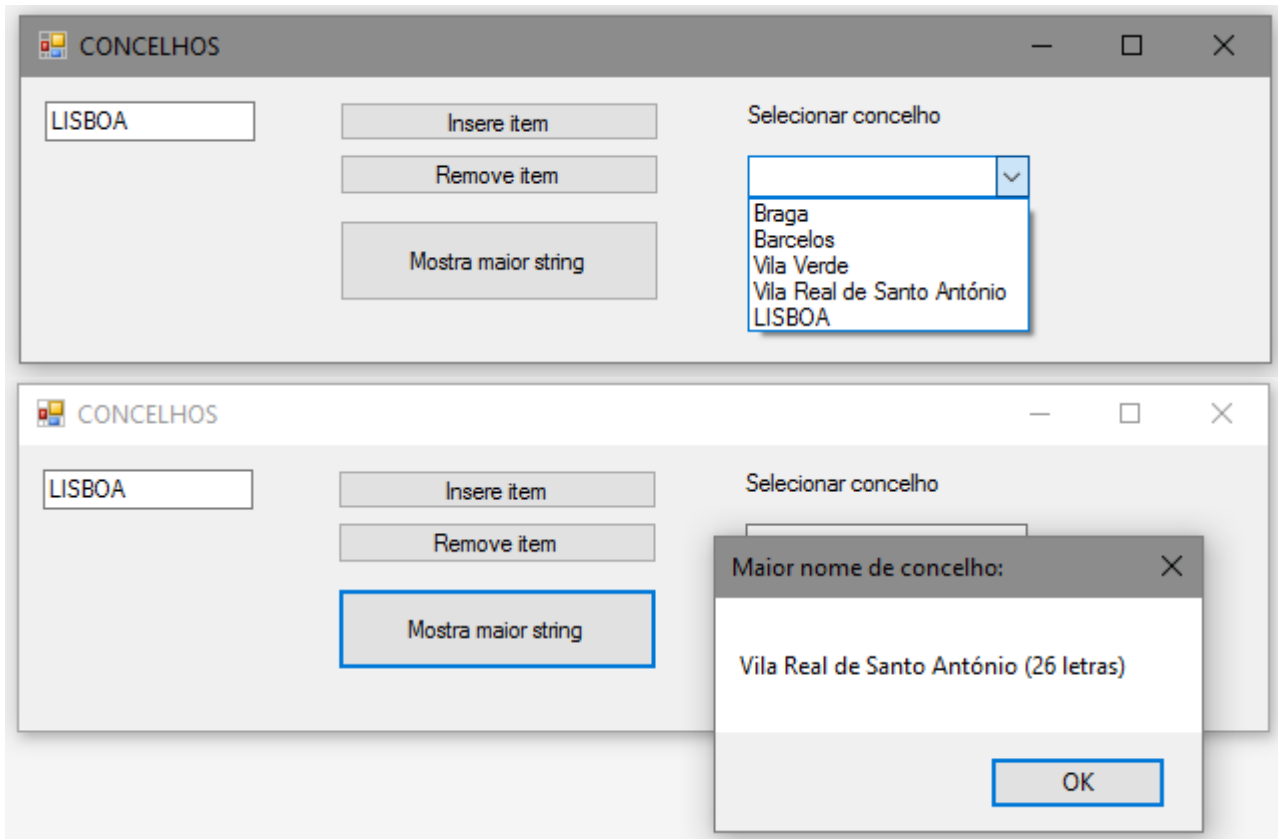
Linhas de código eventualmente úteis

```
string s = null;
//quantos items tem a lista?:
int contador = listBox1.Items.Count;//não leva parêntesis
int ultimo_dalista = contador - 1;//porque começa em zero
for (int i = 0; i <= ultimo_dalista; i++)
{
    //busca string na posição i da primeira lista:
    s = listBox1.Items[i].ToString();
    //envia essa string para a lista 2:
    listBox2.Items.Add(s);
}
```

```
string s = txt_nome_a_ser_trocado.Text;
int i = listBox1.FindString(s);
//i fica com a posição na lista; atenção, começa em ZERO
//se não existir, i fica com -1:
if (i == -1) MessageBox.Show ("O nome não existe na lista!");
else
{
    listBox1.Items[i] = txt_novo_nome.Text;
}
```



Comboboxes – manipulação de itens e operação com strings



Sugestão de linhas de código-ajuda:

Adicionar string à combobox:

```
dropconcelhos.Items.Add(s);
```

Eliminar string da combobox:

```
dropconcelhos.Items.Remove(s);
```

Buscar uma posição da combobox e colocar numa string:

```
s = dropconcelhos.GetItemText(dropconcelhos.Items[i]);
```

MessageBox:

```
strout = Convert.ToString(maiorstring + "(" + maiorstring.Length + " letras)");  
MessageBox.Show(strout, "Maior nome de concelho:");
```



II - Formulário e controlos - Miniprojeto Shrek

EXERCÍCIO

Form1

CÁLCULO DO VALOR DA AVENÇA

NOME: Shrek da Silva

ANO DE NASCIMENTO: 2001

VALOR BASE: 100

☒ Desconto de escalão? (10%)

☒ Desconto de sócio? (5%)

☒ Tem agravamento? (12 euros)

Reside no distrito? ☒ Sim ☐ Não

CONCELHO: Amares

CALCULAR 66

Elabore o formulário, tendo em conta as seguintes observações:

- não é pedido para fazer validações;
- o valor final obtido vem das seguintes operações:
 - valor base da avença = 100, neste exemplo
 - se ano de nascimento for 2000 ou superior, tem desconto de 10% (passa a 90)
 - se desconto de escalão ativo, desconta outros 10% sobre 100 (passa a 80)
 - se desconto de sócio ativo, desconta 5% (75)
 - se tem agravamento, acresce 12 (não é percentagem) (87)
 - se reside no distrito desconta 1% (86)
 - se o concelho for Amares, desconta 20% (66) (considere Braga, Amares e Guimarães, como concelhos para a combobox)
- para o exemplo mostrado, o valor final será 66

O código, aqui incompleto, para verificar e tratar os controlos, pode ser o seguinte:

```
double valorbase = 0;
double valorfinal = 0;
try
{
    int ano = Convert.ToInt16(textBox1.Text);
    valorbase = Convert.ToDouble(textBox3.Text);
    valorfinal = valorbase;

    if (ano > 1999) valorfinal = valorfinal - valorbase * 0.1;

    if (checkBox1.Checked == true) valorfinal = valorfinal - valorbase * 0.1;
```



```
if (checkBox2.Checked == true) valorfinal = valorfinal - valorbase * 0.05;
if (checkBox3.Checked == true) valorfinal = valorfinal + 12;

if (radioButton2.Checked == true) valorfinal = valorfinal - valorbase * 0.01;

if (comboBox2.Text == "Amares") valorfinal = valorfinal - valorbase * 0.2;

textBox4.Text = Convert.ToString(valorfinal);
}
catch (Exception)
{
    label_erros.Text = "Dados inválidos";
    textBox4.Text = "";
}
```

III - Projeto “Nomes & Notas”, com listas

Pretende-se elaborar formulário que permita manipular uma lista de nomes e avaliação de uma disciplina, utilizando controlos de um formulário.

a) inicializar as listas

```
private void button1_Click(object sender, EventArgs e)
{
    //inicializar lista com 4 nomes:
    lst_nomes.Items.Add("Ana");
    lst_nomes.Items.Add("Rui");
    lst_nomes.Items.Add("Abel");
    lst_nomes.Items.Add("Zacarias");

    //inicializar lista 2 com 4 notas:
    lst_notas.Items.Add("12");
    lst_notas.Items.Add("15");
    lst_notas.Items.Add("17");
    lst_notas.Items.Add("8");
}
```

b) encontrar a maior avaliação

```
int maior;
int posicao=0;
//começamos por afirmar: maior avaliação é a primeira:
maior = Convert.ToInt16(lst_notas.Items[0]);

int n_de_elementos = lst_notas.Items.Count;
//caso apareça outra mais elevada, troca-se:
for (int i = 0; i <= n_de_elementos-1; i++)
{
    if (Convert.ToInt16(lst_notas.Items[i])>maior)
    {
        maior = Convert.ToInt16(lst_notas.Items[i]);
        posicao = i;
    }
}
```



```
    }  
}  
MessageBox.Show(maior.ToString()); //mostra o valor maior  
MessageBox.Show(posicao.ToString()); //mostra em que posição da lista está  
string nome = Convert.ToString(lst_nomes.Items[posicao]);  
MessageBox.Show(nome); //mostra o nome que corresponde à maior avaliação
```

c) calcular a média

```
double media, soma=0;  
//percorrer a lista, somando os elementos  
for (int i = 0; i <= lst_notas.Items.Count-1; i++)  
{  
    soma = soma + Convert.ToDouble(lst_notas.Items[i]);  
}  
//2018-02-10: a linha seguinte tinha erro: lst_notas.Items.Count -1  
media = soma / lst_notas.Items.Count; //dividir pelo nº. de elementos  
txt_media.Text = media.ToString();
```

d) contar negativas

The screenshot shows a Windows application window titled 'Nomes & Avaliações'. It contains two list boxes: 'Nomes' with items 'Ana', 'Rui', 'Abel', and 'Zacarias'; and 'Avaliação de C#' with items '12', '8', '13', and '8'. There are four buttons: 'Inicializar (4 nomes e notas)', 'Limpar as 2 listas', 'Nome / nota mais elevada', and 'Calcular a média'. Below these buttons are two text boxes: 'Abel' and '10.25'. At the bottom, there is a button 'Contar as negativas' and a text box showing the value '2'.

```
int contador=0;  
//percorrer a lista, contando os elementos  
for (int i = 0; i <= (lst_notas.Items.Count - 1); i++)  
{  
    //se for negativa, incrementar o contador:  
    if (Convert.ToDouble(lst_notas.Items[i])<10) contador ++;  
}  
txt_n_negativas.Text = contador.ToString();
```

e) filtrar com combobox

Pretende-se: mostrar numa listbox (terceira listbox) os nomes dos aprovados ou reprovados, conforme escolha pela combo.

Primeiro: inicializar a combo com os valores “Aprovados” e “Reprovados”, recorrendo ao método Form_Load().

```
private void Form1_Load(object sender, EventArgs e)  
{  
    cbo_reprovados.Items.Add("Aprovados");  
    cbo_reprovados.Items.Add("Reprovados");  
}
```

Segundo: adicionar nova listbox e nomeá-la corretamente (ex: lst_filtrados).

Terceiro: associar o código ao evento “SelectedIndexChanged” da combo.

The screenshot shows the same application window as before, but with a third list box added on the right, titled 'Nomes & Avaliações'. This list box contains the names 'Ana' and 'Abel'. Above this list box is a combobox with the text 'Aprovados' and a dropdown arrow. The buttons and other elements remain the same.

Código do método:



```
//primeiro: limpar a lista 3:
lst_filtrados.Items.Clear();
//buscar o valor que está neste momento na combo:
string s = cbo_reprovados.SelectedItem.ToString();

if (s=="Reprovados")
{
    //percorrer a lista 2, com as notas:
    for (int i = 0; i < lst_notas.Items.Count; i++)
    {
        if (Convert.ToInt16(lst_notas.Items[i])<10)
        { //se for negativa (lista 1), escreve o nome (lista 2)
          //na lista 3
          lst_filtrados.Items.Add(lst_nomes.Items[i]);
        }
    }
}
if (s == "Aprovados")//considerar usar "else", em vez do if
{
    //percorrer a lista 2, com as notas:
    for (int i = 0; i < lst_notas.Items.Count; i++)
    {
        if (Convert.ToInt16(lst_notas.Items[i]) >= 10)
        { //se for positiva (lista 1), escreve o nome (lista 2)
          //na lista 3
          lst_filtrados.Items.Add(lst_nomes.Items[i]);
        }
    }
}
}
```

f) filtrar também com radio button

Pretende-se: mostrar os reprovados/aprovados, seleccionando também o sexo.
Exemplo: que formandos do sexo masculino estão reprovados?

Primeiro: inicializar com 8 formandos em vez de 4, especificando o sexo.

<pre>//inicializar lista com 8 nomes: lst_nomes.Items.Add("Ana (F)"); lst_nomes.Items.Add("Rui (M)"); lst_nomes.Items.Add("Rita (F)"); lst_nomes.Items.Add("Zacarias (M)"); lst_nomes.Items.Add("José (M)");</pre>	<div>Nomes & Avaliações</div> <table> <thead> <tr> <th>Nomes</th><th>Aval. de C#</th></tr> </thead> <tbody> <tr><td>Ana (F)</td><td>12</td></tr> <tr><td>Rui (M)</td><td>8</td></tr> <tr><td>Rita (F)</td><td>13</td></tr> <tr><td>Zacarias (M)</td><td>8</td></tr> <tr><td>José (M)</td><td>7</td></tr> <tr><td>Sofia (F)</td><td>10</td></tr> </tbody> </table>	Nomes	Aval. de C#	Ana (F)	12	Rui (M)	8	Rita (F)	13	Zacarias (M)	8	José (M)	7	Sofia (F)	10
Nomes	Aval. de C#														
Ana (F)	12														
Rui (M)	8														
Rita (F)	13														
Zacarias (M)	8														
José (M)	7														
Sofia (F)	10														

Segundo: adicionar 3 radio buttons e inicializar uma delas no form_load:

```
private void Form1_Load(object sender, EventArgs e)
{
    cbo_reprovados.Items.Add("Aprovados");
    cbo_reprovados.Items.Add("Reprovados");
    rbt_T.Checked = true;
}
```

Terceiro: escrever o código para fazer funcionara as escolhas do utilizador.

Nota: será mais complicado, porque é necessário extrair os "F" ou "M", ou seja, manipular strings.

Usando o mesmo código que já vem de trás, adiciona-se o novo código para implementar o novo pedido:

```
...
if (s=="Reprovados")
{
    //percorrer a lista 2, com as notas:
    for (int i = 0; i < lst_notas.Items.Count; i++)
    {
        //buscar cada nome à lista 1:
        string nome = lst_nomes.Items[i].ToString();
```



```
//saber em que posição está o parêntesis:
int p = nome.IndexOf("(");

//o sexo é o carácter seguinte ao parêntesis:
char sexo = nome[p+1];

if (Convert.ToInt16(lst_notas.Items[i])<10)
{
    //se for NEGATIVA (lista 1), escreve o nome (lista 2), na lista 3
    //se for feminino:
    if (rbt_F.Checked==true)
        if (sexo=='F')
            lst_filtrados.Items.Add(lst_nomes.Items[i]);
    if (rbt_M.Checked == true)
        if (sexo == 'M')
            lst_filtrados.Items.Add(lst_nomes.Items[i]);
    if (rbt_T.Checked == true)
        lst_filtrados.Items.Add(lst_nomes.Items[i]);
}
}
...(resto do código)
```

g) remover os parêntesis e sexos, dos nomes da lista 3

Pretende-se: remover dos nomes da lista 3, os parêntesis e o sexo.

Código a adicionar no final do método associado à combobox:

```
//antes de terminar, percorrer os nomes da lista 3 e remover os parêntesis e sexo:
for (int i = 0; i < lst_filtrados.Items.Count; i++)
{
    string nome_sem_sexo = lst_filtrados.Items[i].ToString();
    //em que posição está o parêntesis esquerdo?:
    int p = nome_sem_sexo.IndexOf("(");
    //a nova string fica com o que vai da posição zero até p:
    nome_sem_sexo = nome_sem_sexo.Substring(0, p);
    //guardar a nova string na lista 3, mas na mesma posição:
    lst_filtrados.Items[i] = nome_sem_sexo;
}
}
```

IV - Projeto “Nomes & Notas”, com datagridview

a) gridview com 5 colunas

Adicionar o controlo DataGridView, da toolbox (nomenclatura: dt_gr_view).



Nas propriedades, adicionar 5 colunas: Nome, Freguesia, Ano, Sexo, Avaliação.

Para adicionar uma linha à grid: (no evento Form_Load)

```
dt_gr_view.Rows.Add("Zeca", "Braga", "1999", "M", "13");
```

Resultado:

	Nome	Freguesia	Ano	Sexo	Avaliação
▶	Zeca	Braga	1999	M	13
✱					

b) inicializar com 20 registos (descarregar anexo)

No evento Form_Load, adicionar o código fornecido através do ficheiro “42-init_grid.txt”.

O form ficará:

	Nome	Freguesia	Ano	Sexo	Avaliação
▶	Ana Rita Cunha	Maximinos	F	1980	11
	Bela Costa Silva	Gualtar	F	1982	12
	Carlos Alberto Co...	Lomar	M	1981	8
	Carlos Serafin Fe...	Lomar	M	1980	8
	Daniel Bastos Go...	Sé	M	1980	14
	Diogo Silva Ferraz	Lamações	M	1980	18
	Elvira Gomes Pe...	Maximinos	F	1977	5
	Fernanda Maria ...	Adufeia	F	1977	18
	Fernando Gomes	Maximinos	M	1983	19
	Glennor Raposo	Sé	M	1959	13
	Hilda Formosa Sá	Calvinha	F	1980	11
	José Manuel Car...	Sé	M	1981	13
	José Alberto Gon...	Gualtar	M	1982	6
	Maria Silvéria Ba...	Sé	F	1975	17
	Arabela Bastos ...	Gualtar	F	1980	12
	Odílio Ferreira	Maximinos	M	1988	14
	Rui Vasco Santos	Maximinos	M	1987	9
	Silvestre Silva Tex...	Maximinos	M	1980	11
	Teodoro Remalho	Gualtar	M	1985	12

c) o controlo menustrip (construção de menu e submenu)

Para adicionar novos métodos, utilizar-se-á o controlo menustrip para substituir os botões.

form com GRIDVIEW

GridViewEstatísticasTratamento de textoFormatações

Inicializa

Limpa

	Freguesia	Ano	Sexo	Avaliação
	Maximinos	F	1980	11
Bela Costa Silva	Gualtar	F	1982	12

Para limpar a grid:

```
dt_gr_view.Rows.Clear();
```

Para inicializar a grid:

copiar o código de inicialização que já está no evento Form_Load.

d) contar as negativas, informar na lista

Pretende-se: percorrer a grid, contar as negativas (número de reprovados) e escrever o resultado numa listbox.

```
int contador=0, avaliacao;
int n_elementos_da_grid = dt_gr_view.Rows.Count;
//iterar a grid:
for (int i = 0; i < n_elementos_da_grid; i++)
{
    //a avaliação está na 5ª coluna, ou seja Cells[4]:
    avaliacao = Convert.ToInt16(dt_gr_view.Rows[i].Cells[4].Value);
    if (avaliacao < 10) contador++;
}
//escrever a contagem na listbox:
```




```
lst_resultados.Items.Clear();//limpar a listBox
lst_resultados.Items.Add("Nº de negativas:");
lst_resultados.Items.Add(contador.ToString());
```

Resultado:

e) freguesia mais representada

Pretende-se saber qual a freguesia que tem mais formandos.

Resultado:

Código utilizado:

```
int contador=0, contadorB = 0;
string freguesia="", freguesia_mais_representada="";
int n_elementos_da_grid = dt_gr_view.Rows.Count;
//iterar a grid:
for (int i = 0; i < n_elementos_da_grid; i++)
{
    //a freguesia está na 2ª coluna, ou seja Cells[1]:
    freguesia = dt_gr_view.Rows[i].Cells[1].Value.ToString();
    contadorB = 0;
    for (int j = 0; j < n_elementos_da_grid; j++)
    {
        //percorrer novamente a grid, contar esta freguesia, comparar a contagem:
        string f= dt_gr_view.Rows[j].Cells[1].Value.ToString();
        if (freguesia == f) contadorB++;
    }
    if (contadorB > contador)
    {
        //a freguesia mais representada já não é a mesma:
        contador = contadorB;
        freguesia_mais_representada = freguesia;
    }
}
```

f) encontrar melhor formando

Pretende-se encontrar o nome do formando com melhor avaliação:

- associar o método ao menu "Formatações\Melhor formando"
- informar na listBox
- pintar o fundo da célula com a cor verde



O resultado previsto seria:

Nome	Freguesia	Sexo	Ano	Avalia
Daniel Bastos Gomes	Sé	M	1980	14
Diogo Silva Ferraz	Lamações	M	1980	18
Elvira Gomes Pendes	Maximinos	F	1977	5
Fernanda Maria Silva	Adaúfe	F	1977	18
Fernando Gomes Barros	Maximinos	M	1983	19
Gilherme Alexandre Barros	Sé	M	1989	13
Hilda Fonseca Silva	Cabreira	F	1980	11
José Manuel Carvalho	Sé	M	1981	13
José Alberto Gomes	Gualtar	M	1982	6

Resultados:
Melhor formando:
Fernando Gomes Barros
19

O código usado poderia ser:

```
int posicao = 0; //posição do melhor formando na grid
//afirmação: o melhor formando é o que estiver na posição 0:
string melhor_formando = dt_gr_view.Rows[0].Cells[0].Value.ToString();
int melhor_nota = Convert.ToInt16(dt_gr_view.Rows[0].Cells[4].Value);

int n_elementos_da_grid = dt_gr_view.Rows.Count;
//iterar a grid:
for (int i = 0; i < n_elementos_da_grid; i++)
{
    //caso este formando seja, melhor, fazer a troca
    if (Convert.ToInt16(dt_gr_view.Rows[i].Cells[4].Value) > melhor_nota)
    {
        melhor_nota = Convert.ToInt16(dt_gr_view.Rows[i].Cells[4].Value);
        melhor_formando = dt_gr_view.Rows[i].Cells[0].Value.ToString();
        posicao = i;
    }
}
//informar, enviando para a listBox:
lst_resultados.Items.Clear(); //limpar a listBox
lst_resultados.Items.Add("Melhor formando:");
lst_resultados.Items.Add(melhor_formando.ToString());
lst_resultados.Items.Add(melhor_nota.ToString());
//e também, pintar a célula da grid a verde:
dt_gr_view.Rows[posicao].Cells[0].Style.BackColor = Color.Green;
```

g) gerar as avaliações aleatoriamente

Pretende-se: substituir as avaliações que estão na grid, por outras geradas aleatoriamente.

A função seguinte, que não recebe argumentos e devolve um número inteiro, gera um número entre 6 e 20.

```
private int geravalor()
{
    //devolve valor entre 6 e 20
    int avaliacao;
    Random r = new Random();

    avaliacao = r.Next(6, 21);
    if (avaliacao < 10 || avaliacao > 18) { avaliacao = r.Next(6, 21); }
    for (int k = 0; k < 10000000; k++)
    {
        //(se o computador ficar lento, diminuir o valor 10000000...
        //ou aumentar, se os números não forem suficientemente aleatórios)
        //diversão ao processador,
        //para melhorar a geração de aleatórios
    }
    return avaliacao;
}
```

Para atribuir uma nota aleatória à posição 10 da grid, a instrução seria:



```
dt_gr_view.Rows[10].Cells[4].Value = geravalor();
```

Para gerar as avaliações para toda a coluna, esta instrução ficaria dentro de um ciclo:

```
for (int i = 0; i < dt_gr_view.Rows.Count; i++)
{
    dt_gr_view.Rows[i].Cells[4].Value = geravalor();
}
```

h) salientar com cores, as avaliações negativas

Pretende-se: salientar as avaliações entre 7 e 10, ou seja, se forem 8 ou nove, com a cor laranja; abaixo de 8, a vermelho.

Carlos Alberto Costa	Lomar	M	1981	8
Carlos Serafim Ferreira	Lomar	M	1980	8
Daniel Bastos Gomes	Sé	M	1980	14
Diogo Silva Ferraz	Lamações	M	1980	18
Elvira Gomes Pendes	Maximinos	F	1977	5

As instruções de manipulação podem ser as seguintes:

```
dt_gr_view.Rows[0].Cells[0].Style.ForeColor = Color.Orange; //texto cor de laranja
dt_gr_view.Rows[0].Cells[0].Style.ForeColor = Color.Red; //texto vermelho
```

V – Recapitulação: form com grid (lista de produtos)

a) problema proposto

Uma certa ficha de produtos tem os seguintes campos: código de produto, nome, categoria a que pertence, preço, quantidade. Na imagem seguinte, vê-se um registo com esta estrutura:

#Prod.	Nome do produto	Categoria	Preço	Qde.
1	Chai	Bebidas	18	39

A imagem seguinte sugere o que se pede para fazer (formulário):

Pede-se para:

- ter uma lista de produtos numa gridview
- implementar um método que verifique quantos produtos há na grid (ponto 1, resposta: 77)
- um método que encontre o preço mais elevado (ponto 2, resposta: 263)
- que conte produtos de uma certa categoria (ponto 3, resposta: se for bebidas, há 12 produtos)
- que some as existências de todos os produtos (ponto 4, resposta: 3119)

O ficheiro [Produtos\(grid inicializada\).7z](#) pode ser descarregado da pasta online; contem um formulário com a gridview inicializada, ou seja, o método “`private void Form1_Load (...)`” inclui as linhas de código que inicializam a gridview com 77 registos, com a estrutura referida.

Implemente neste projeto os 4 métodos referidos.



A título de sugestão,

- comece por decidir se vai usar o controlo menustrip com 4 opções, ou então 4 botões
- atribua os nomes aos controlos de acordo com a nomenclatura que lhe for atrativa
- faça os exercícios começando pelos mais fáceis: 1 → 4 → 2 → 3
- não esqueça os backups...

b) resolução ponto 1: quantos produtos?

Neste caso não é necessário utilizar uma estrutura cíclica, porque se pode usar a propriedade “count” da grid:

```
int total_de_linhas_da_grid = dgv.Rows.Count;
//a grid tem uma linha em branco; subtrair 1:
int total_de_produtos = total_de_linhas_da_grid - 1;
//enviar a contagem para o form:
```

c) resolução ponto 2: preço mais elevado?

O código seguinte foi elaborado pelo formando Celso e responde com exatidão ao pedido.

```
// Para obter o número total de produtos é necessário subtrair o valor de um à contagem,
// devido à contagem incluir sempre a última linha em branco (no caso das DataGridViews)
int n_produtos = dgv.Rows.Count - 1;
// Inicialização da variável com o primeiro valor da lista de preços
int maior_preço = Convert.ToInt16(dgv.Rows[0].Cells[3].Value);

// Uma vez que se inicializou a variável com o primeiro valor da lista de preços,
// o ciclo for que percorre o resto dos valores dos preços pode ser inicializado
// a partir do segundo valor da lista (cujo índice é 1)
for (int i = 1; i < n_produtos; i++)
{
    // Compara o preço do produto na posição i da lista com o maior valor de
    // preço encontrado até ao momento. Se o valor de preço que se encontra na
    // posição i for maior do que o guardado na variável maior_preço, então
    // a variável maior_preço toma esse valor
    if (Convert.ToInt16(dgv.Rows[i].Cells[3].Value) > maior_preço)
    {
        maior_preço = Convert.ToInt16(dgv.Rows[i].Cells[3].Value);
    }
}

// No final do ciclo, coloca-se o resultado numa caixa de texto, fazendo a conversão
// do tipo de dados da variável maior_preço de inteiro para string.
txbx_MCaro.Text = maior_preço.ToString();
```

d) resolução ponto 3: contagem de produtos por categoria (com textbox)

(3) a)	Quantos produtos há desta categoria? (com textbox)	Qual a categoria?	Bebidas	12
--------	--	-------------------	---------	----

O código poderia ser:

```
int contador = 0;
for (int i = 0; i < dgv.Rows.Count - 1; i++)
{
    //se a categoria que está na grid for igual à que está na text:
    if (dgv.Rows[i].Cells[2].Value.ToString() == ctx_categoria.Text)
        contador++;
}
ctx_conta_por_categoria.Text = contador.ToString();
```

**d) resolução ponto 3: contagem de produtos por categoria (com combobox)**

Na imagem, a combobox foi previamente inicializada com as categorias. A forma mais simples de o fazer,

- é utilizar as propriedades da combobox (botão direito do rato ...),
- mas também poderia ser recorrendo ao método `Add()` (ex: `comboBox.Items.Add("Bebidas");`); por este método, a inicialização seria feita no evento `Load` associado ao `Form` (método `Form_Load()`);
- ou ainda, no mesmo método, percorrer a `grid`, detetar as diferentes categorias e preencher a combobox; o formando David implementou esta solução; no entanto, este mecanismo é mais sofisticado para o momento presente e não se trancreve aqui; opcionalmente, poderá descarregar o seu ficheiro ([Produtos\(David\).zip](#)), e ver por si.

O código (ou parte, escrito pelo formando David), associado ao botão da imagem, poderia ser:

```
//string que representa a categoria selecionada na comboBox
string cat = cbo_categoria.SelectedItem.ToString();

//contador, inicializado a zero, para contarmos o número de
//ocorrências da categoria na nossa gridview
int count = 0;

//Iteramos sobre a gridview, exceto a última linha (que está vazia)
for (int i = 0; i < dgv.RowCount - 1; i++)
{
    //string que representa a categoria do produto na linha com índice i
    string c = dgv.Rows[i].Cells[2].Value.ToString();

    //se a categoria do produto for igual
    //à categoria selecionada na comboBox...
    if (c == cat)
    {
        //...incrementamos o contador
        count++;
    }
}

//Apresentamos o valor de count, atribuindo o seu
//valor à propriedade Text da textBox tb_categoria
tb_categoria.Text = count.ToString();
```

Este código poderia ser associado também à combobox; assim, a contagem ocorrerá se:

- for clicado o botão, mas também,
- se for feita a escolha de categoria na combobox.

Os 2 métodos, com exatamente o mesmo código, seriam:

```
private void button3_Click(object sender, EventArgs e)...
```

```
private void cbo_categoria_SelectedIndexChanged(object sender, EventArgs e)...
```

e) resolução ponto 4: soma das existências

Para o quarto ponto, a formanda Marta escreveu o código seguinte, que permite calcular corretamente a soma solicitada.

```
//declaração de variáveis
int n_elementos_da_grid = dgv.Rows.Count;
int soma = 0;
```



```
//fazer iteração
for (int i = 0; i < n_elementos_da_grid; i++) //ou n_elementos_da_grid -1
{
    // soma de todos as quantidades existentes na coluna quantidades (4)
    soma = soma + Convert.ToInt32(dgv.Rows[i].Cells[4].Value);
}
//devolução da soma na caixa de texto
ctx_soma.Text = Convert.ToString(soma);
```

f) resolução ponto extra: contagem de produtos com filtro em radio buttons

Pretende-se: efetuar a contagem de produtos na grid, mas tendo em conta o radio button selecionado:

(1) Quantos produtos há? 5

☒ Quantidade a zero
☐ Quantidade não zero
☐ Todos

Há 5 produtos cuja quantidade está a zero, 72 com quantidade não nula, e 77 no total.

O código seguinte funciona, mas não tem em conta uma correta sequenciação de if's e else's. No entanto, a perda de desempenho é negligenciável, e a facilidade do algoritmo compensa; já sabe, claro, que isto é discutível...

```
int total_de_linhas_da_grid = dgv.Rows.Count;
//declaração e inicialização de 3 contadores, um para cada radio button:
int contador_zeros = 0, contador_nao_zeros = 0, contador_todos = 0;

for (int i = 0; i < total_de_linhas_da_grid - 1; i++)
{
    if (rb_zeros.Checked)//se for este que está "checkado", incrementar contador:
        if (Convert.ToInt16(dgv.Rows[i].Cells[4].Value) == 0) contador_zeros++;

    if (rb_nao_zeros.Checked)
        if (Convert.ToInt16(dgv.Rows[i].Cells[4].Value) != 0) contador_nao_zeros++;

    if (rb_todos.Checked) contador_todos++;
}
//para o que está "checkado", escrever o contador respetivo na textbox de saída:
if (rb_zeros.Checked) ctx_total_de_produtos.Text = contador_zeros.ToString();
if (rb_nao_zeros.Checked) ctx_total_de_produtos.Text = contador_nao_zeros.ToString();
if (rb_todos.Checked) ctx_total_de_produtos.Text = contador_todos.ToString();
```

Finalmente, um possível layout para o form completo seria:

Form1

#Prod.	Nome do produto	Categoria	Preço	Qde.
35	Steeleye Stout	Bebidas	18	20
36	Inlagd Sill	Pescados	19	112
37	Gravad lax	Pescados	26	11
38	Côte de Blaye	Bebidas	263	17
39	Chartreuse verte	Bebidas	18	69
40	Boston Crab Meat	Pescados	18	123
41	Jack's New England Clam...	Pescados	9	85
42	Singaporean Hokkien Frie...	Cereais	14	26
43	Ipoh Coffee	Bebidas	46	17
44	Gula Malacca	Condimentos	19	27
45	Rogede sild	Pescados	9	5
46	Spegesild	Pescados	12	95
47	Zaanse koeken	Confecções	9	36
48	Chocolate	Confecções	12	15
49	Maxilaku	Confecções	20	10
50	Valkoinen suklaa	Confecções	16	65
51	Manjimup Dried Apples	Pomar	53	20
52	Filo Mix	Cereais	7	38

(1) Quantos produtos há, tendo em conta o botão de rádio escolhido? 72

☐ Quantidade a zero
☒ Quantidade não zero
☐ Todos

(2) Qual o preço mais elevado? 263

(3) a) Quantos produtos há desta categoria? (com textbox) Bebidas 12

(3) b) Quantos produtos há desta categoria? (com combobox) Bebidas 12

(4) Quantos produtos há ao todo, em amazém? 3119





OUTROS EXERCÍCIOS

Utilizando o try / catch

```
using System;
using System.Windows.Forms;
namespace try_catch
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

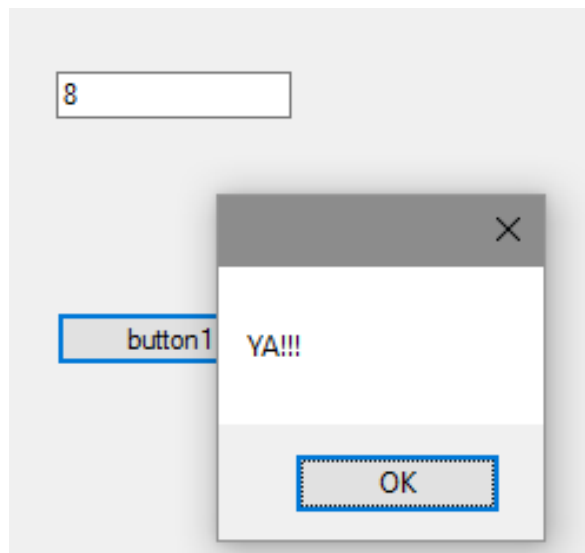
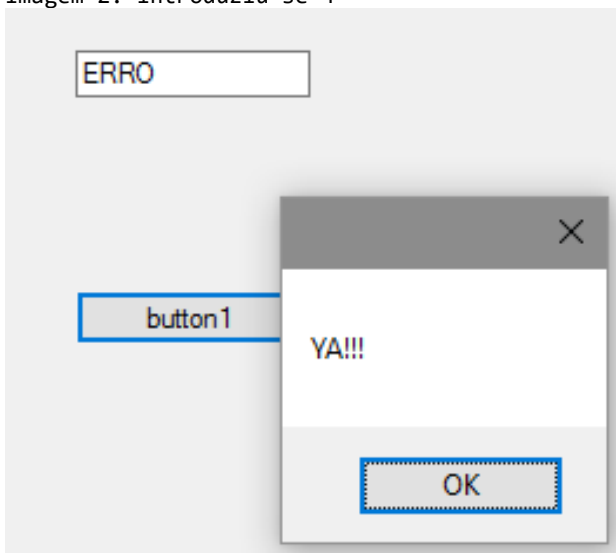
        private void button1_Click(object sender, EventArgs e)
        {
            int n;
            try
            {
                n = Convert.ToInt16(textBox1.Text);
                n = n * 2;
                textBox1.Text = Convert.ToString(n);
            }

            catch
            {
                textBox1.Text = "ERRO";
            }

            finally
            {
                MessageBox.Show("YA!!!");
            }
        }
    }
}
```

Imagem 1: introduziu-se dígito com letra ou seja, string; ex: 4K)

Imagem 2: introduziu-se 4



Validação de número inteiro – exemplo com int.TryParse



```
private void bt1_Click(object sender, EventArgs e)
{
    int n;
    string s;
    bool result;

    s = tbinput.Text;

    result = int.TryParse(s, out n);
    // verifica se é inteiro e devolve bool

    if (result) {
        tboutput.Text = Convert.ToString(n);
        //escreve no output
        tberro.Text = ""; //escreve no semáforo
    }
    else
    {
        tboutput.Text = "";
        tberro.Text = "Introduzir apenas inteiros";
    }
}
```

O preçário é uma listbox.

O novo texto abaixo do menu, é um rótulo.

Script do segundo botão de calcular, ou seja, para os extras:

```
private void button2_Click(object sender, EventArgs e)
{
```



```
int t, l1 = 0, l2 = 0, l3 = 0;
if (checkBox1.Checked) l1 = 1 * Convert.ToInt32(qe1.Text);
if (checkBox2.Checked) l2 = 2 * Convert.ToInt32(qe2.Text);
if (checkBox3.Checked) l3 = 1 * Convert.ToInt32(qe3.Text);

t = l1 + l2 + l3;
totalextras.Text = Convert.ToString(t);
}
```

Pequenas operações de validação

Neste exercício pretende-se exemplificar como fazer validações simples, ou seja, obrigar que os valores introduzidos no formulário estejam dentro de certos limites. Poderá, eventualmente, utilizar estes exemplos no exercício da Pizza, Shrek, outros.

- Nome não pode estar vazio
- Ano deve estar entre 1900 e 2016
- Telemóvel tem que ter tamanho 9
- Email tem que ter @
- Enquanto houver violações, a semáforo apresentará “Formulário inválido”

Problemas identificados:

Nome:	<input type="text"/>	Nome vazio
Ano de nascimento:	<input type="text" value="2900"/>	Fora do intervalo
Telemóvel:	<input type="text" value="123456"/>	Tamanho 9...
Email:	<input type="text" value="2@com"/>	Mínimo 8...?!

Semáforo geral
Formulário inválido

```
private void bt1_Click(object sender, EventArgs e)
{
    //RED=false, ou seja, há problemas com as validações no form

    bool VERMELHO = false; bool VERDE = true; bool semaforogeral = VERDE;
    int n; string s; bool result;

    // ----- validações na inputNOME:
    s = inputnome.Text;
    if (s.Length == 0) { semaforogeral = VERMELHO; semaforonome.Text = "Nome vazio"; }
    else { semaforonome.Text = "OK"; }
    // ----- validações na inputANO
    s = inputano.Text;
    if (s.Length == 0) { semaforogeral = VERMELHO; semaforoano.Text = "Ano vazio"; }
    if (s.Length != 0)
    {
        result = Int32.TryParse(s, out n);
        if (result)
        {

```



```
        if (n > 2016 || n < 1900)
        {
            semaforoano.Text = "Fora do intervalo";
            semaforogeral = VERMELHO;
        }
        else { semaforoano.Text = "OK"; }
    }
}

// ----- validações na inputTEL
s = inputtel.Text;
if (s.Length != 9) { semaforogeral = VERMELHO; semaforotel.Text = "Tamanho 9..."; }
else
{
    if (Int32.TryParse(s, out n)) { semaforotel.Text = "OK"; }
    else { semaforogeral = VERMELHO; semaforotel.Text = "Apenas dígitos"; }
}

// ----- validações na inputEMAIL
s = inputemail.Text;
if (s.Length < 8) { semaforogeral = VERMELHO; semaforoemail.Text = "Mínimo 8...?!"; }
else
{
    int posiarroba;
    posiarroba = s.IndexOf("@"); //a string tem @?
    if (posiarroba == -1) //não encontrou...
    {
        semaforogeral = VERMELHO;
        semaforoemail.Text = "Falta o @...";
    }
    else semaforoemail.Text = "OK";
}

// finalmente, verificar semáforo geral
if (semaforogeral == VERDE) { outputsemaforogeral.Text = "Formulário válido"; }
else { outputsemaforogeral.Text = "Formulário inválido"; }
```

Manipulação de combobox e operação com strings



FICHA DE TRABALHO

1. Descarregar o projeto compactado com o form mas sem código
2. Elaborar o código que implementa os botões do form

Nota: os 2 últimos métodos já têm as strings para fazer as colonizações.
Precisa de declarar variáveis e codificar o procedimento.

Construtor e destrutor

O programa mostrado tem a class “principal”, que declara o objeto, e a classe C.
A compilação do programa resulta na listagem mostrada na 3ª imagem.

Elabore o programa.

```
using System;
namespace meuPrograma
{
    class Programa
    {
        static void Main(string[] args)
        {
            C cobj = new C();
            cobj.lista();
        }
    }
}
```

```
class C
{
    public struct ficha...
    public ficha[] F = new ficha[21];
    public C(...) //constructor

    ~C(...) //destructor

    public void lista(...)
```



```
C:\WINDOWS\system32\cmd.exe

Ana Rita Cunha..... Maximinos..... F 1980
Bela Costa Silva..... Gualtar..... F 1982
Carlos Alberto Costa..... Lomar..... M 1981
Carlos Serafim Ferreira..... Lomar..... M 1980
Daniel Bastos Gomes..... Sé..... M 1980
Diogo Silva Ferraz..... Lamações..... M 1980
Elvira Gomes Pendes..... Maximinos..... F 1977
Fernanda Maria Silva..... Adaúfe..... F 1977
Fernando Gomes Barros..... Maximinos..... M 1983
Gilherme Alexandre Barros..... Sé..... M 1989
Hilda Fonseca Silva..... Cabreira..... F 1980
José Manuel Carvalho..... Sé..... M 1981
José Alberto Gomes..... Gualtar..... M 1982
Maria Silvéria Bastos..... Sé..... F 1975
Anabela Bastos Torres..... Gualtar..... F 1980
Otávio Ferreira..... Maximinos..... M 1988
Rui Vasco Santos..... Maximinos..... M 1987
Silvério Silva Teixeira..... Maximinos..... M 1980
Teodoro Armando Matos..... Gualtar..... M 1985
```

Geração de outros controlos por codificação

Criar Lista

Limpa Lista

Elimina Lista

Criar text

Remove

Criar combo

Remove Combo

Criar Lista

Limpa Lista

Elimina Lista

Criar text

Remove

Criar combo

Remove Combo

Inserir na lista

Ana Rita Cunha
Bela Costa Silva
Carlos Alberto Costa
Carlos Serafim Ferreira
Daniel Bastos Gomes
Diogo Silva Ferraz
Elvira Gomes Pendes
Fernanda Maria Silva
Fernando Gomes Barros
Gilherme Alexandre Barros
Hilda Fonseca Silva
José Manuel Carvalho
José Alberto Gomes
Maria Silvéria Bastos
Anabela Bastos Torres
Otávio Ferreira
Rui Vasco Santos
Silvério Silva Teixeira
Teodoro Armando Matos
Zacarias Alexandre Sampaio

Inserir na combo



```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        ListBox lista = new ListBox(); //Create an instance of the ListBox
        TextBox text = new TextBox();
        Button insere = new Button();
        ComboBox combocli = new ComboBox();
        Button insere2 = new Button();
        TextBox text2 = new TextBox();

        public Form1()
        {
            InitializeComponent();
        }
        // ===== listbox
        public void button1_Click(object sender, EventArgs e)
        {
            string[] F = new string[22];
            F[1] = "Ana Rita Cunha";
            F[2] = "Bela Costa Silva";
            F[3] = "Carlos Alberto Costa";
            F[4] = "Carlos Serafim Ferreira";
            F[5] = "Daniel Bastos Gomes";
            F[6] = "Diogo Silva Ferraz";
            F[7] = "Elvira Gomes Pendes";
            F[8] = "Fernanda Maria Silva";
            F[9] = "Fernando Gomes Barros";
            F[10] = "Gilherme Alexandre Barros";
            F[11] = "Hilda Fonseca Silva";
            F[12] = "José Manuel Carvalho";
            F[13] = "José Alberto Gomes";
            F[14] = "Maria Silvéria Bastos";
            F[15] = "Anabela Bastos Torres";
            F[16] = "Otávio Ferreira";
            F[17] = "Rui Vasco Santos";
            F[18] = "Silvério Silva Teixeira";
            F[19] = "Teodoro Armando Matos";
            F[20] = "Zacarias Alexandre Sampaio";

            lista.Size = new System.Drawing.Size(150, 300);
            //Set the size and location of the ListBox largura x altura

            lista.Location = new System.Drawing.Point(320, 50);
            //500 à margem esquerda, 50 ao topo do form

            Controls.Add(lista);
            // Add the ListBox to the form

            lista.Items.Clear();
            for (int x = 1; x < 21; x++)
            //copiar as strings para a lisbox
            {
                lista.Items.Add(F[x]);
            }
        }
        private void button2_Click(object sender, EventArgs e)
        {
            lista.Items.Clear();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            Controls.Remove(lista);
        }
    }
}
```



```
}
// ===== text
private void button4_Click(object sender, EventArgs e)
{
    // tratando da textbox...
    text.Size = new System.Drawing.Size(150, 40);
    text.Location = new System.Drawing.Point(150, 50);
    Controls.Add(text);
    // tratando do botão
    insere.Size = new System.Drawing.Size(120, 20);
    insere.Location = new System.Drawing.Point(150, 80);
    insere.Text = "Insere na lista";
    insere.Click += insere_Click; //evento associado
    // += operador neste contexto significa assinar um evento
    Controls.Add(insere);
}

private void insere_Click(object sender, EventArgs e)
{
    lista.Items.Add(text.Text);
    //copiou da text para a listbox
}

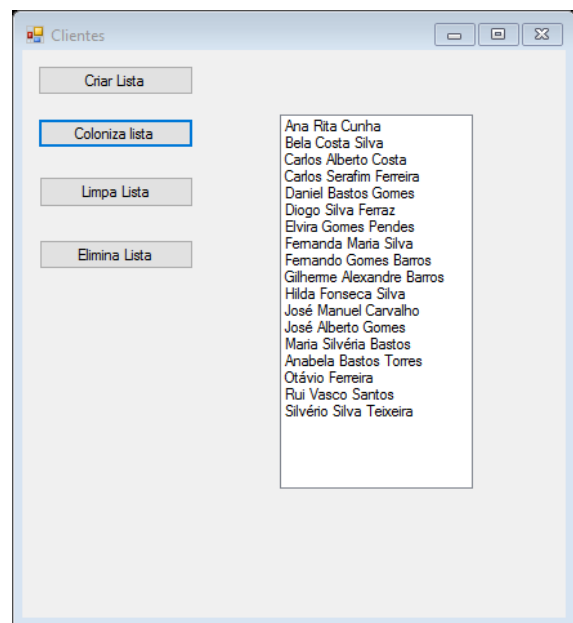
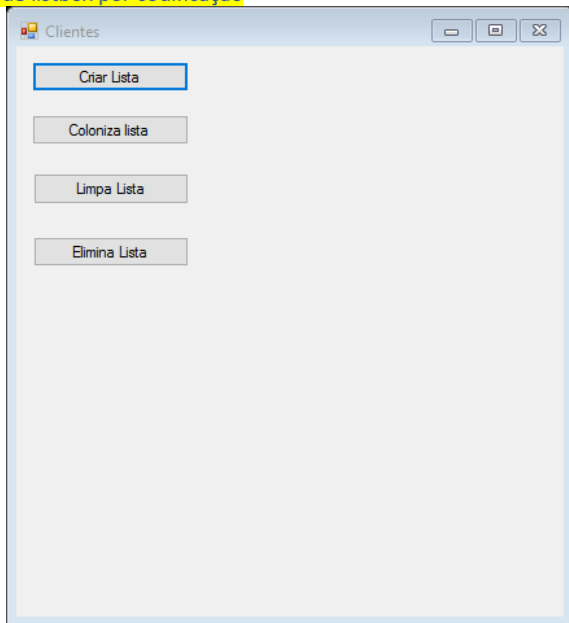
private void button5_Click(object sender, EventArgs e)
{
    Controls.Remove(text);
    Controls.Remove(insere);
}

// ===== combobox
private void button7_Click(object sender, EventArgs e)
{
    string[] F = new string[22];
    F[1] = "Ana Rita Cunha";
    F[2] = "Bela Costa Silva";
    F[3] = "Carlos Alberto Costa";
    F[4] = "Carlos Serafim Ferreira";
    F[5] = "Daniel Bastos Gomes";
    F[6] = "Diogo Silva Ferraz";
    F[7] = "Elvira Gomes Pendes";
    F[8] = "Fernanda Maria Silva";
    F[9] = "Fernando Gomes Barros";
    F[10] = "Gilherme Alexandre Barros";
    F[11] = "Hilda Fonseca Silva";
    F[12] = "José Manuel Carvalho";
    F[13] = "José Alberto Gomes";
    F[14] = "Maria Silvéria Bastos";
    F[15] = "Anabela Bastos Torres";
    F[16] = "Otávio Ferreira";
    F[17] = "Rui Vasco Santos";
    F[18] = "Silvério Silva Teixeira";
    F[19] = "Teodoro Armando Matos";
    F[20] = "Zacarias Alexandre Sampaio";
    // tratando da textbox...
    text2.Size = new System.Drawing.Size(150, 50);
    text2.Location = new System.Drawing.Point(500, 50);
    Controls.Add(text2);
    // tratando do botão
    insere2.Size = new System.Drawing.Size(120, 20);
    insere2.Location = new System.Drawing.Point(500, 90);
    insere2.Text = "Insere na combo";
    insere2.Click += insere2_Click; //evento associado
    // += operador neste contexto significa assinar um evento
    Controls.Add(insere2);
    combocli.Size = new System.Drawing.Size(150, 100);
    combocli.Location = new System.Drawing.Point(500, 120);
    Controls.Add(combocli);
    combocli.Items.Clear();
    for (int x = 1; x < 21; x++)
    //copiar as strings para a combo
    {
        combocli.Items.Add(F[x]);
    }
}
```



```
    }  
}  
private void insere2_Click(object sender, EventArgs e)  
{  
    combocli.Items.Add(text2.Text);  
    //copiou da text para a listBox  
}  
  
private void button6_Click(object sender, EventArgs e)  
{  
    Controls.Remove(combocli);  
    Controls.Remove(text2);  
    Controls.Remove(insere2);  
}  
}  
}
```

Geração de listBox por codificação



Linhas de código exemplo:

Declaração do objeto lista do tipo ListBox:

```
public partial class Form1 : Form  
{  
    //Create an instance of the ListBox:  
    ListBox lista = new ListBox();  
}
```

Criação e posicionamento do controlo lista no form:

```
//Set the size and location of the ListBox largura x altura:  
lista.Size = new System.Drawing.Size(150, 300);
```

```
//200 à margem esquerda, 50 ao topo do form  
lista.Location = new System.Drawing.Point(200, 50);
```

```
// Add the ListBox to the form  
Controls.Add(lista);
```

Eliminação do controlo (desaparece do form):

```
Controls.Remove(lista);
```

Operações com strings

NOTA:



Este projeto foi elaborado em modo consola (e não com Windows Form)

```
C:\WINDOWS\system32\cmd.exe
O cão e o gato são 2 bons amigos!

Comprimento:36
2º carácter:O
4º carácter:c
Upper: O CÃO E O GATO SÃO 2 BONS AMIGOS!
Minúsculas: o cão e o gato são 2 bons amigos!
Limpa espaços:O cão e o gato são 2 bons amigos!
Contem 'gato' ?Sim.
Substring:migos!
Replace: O cão e o tigre são 2 bons amigos!
Remove: O cão gato são 2 bons amigos!
Posição do !:34
Posição de 'gato':11
Primeiro a dps de 'gato':8

Press any key to continue . . .
```

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

namespace strings
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = " O cão e o gato são 2 bons amigos! ";

            Console.WriteLine(s);
            Console.WriteLine("\nComprimento:"); Console.Write(s.Length);
            Console.WriteLine("\n2º carácter:"); Console.Write(s[1]);
            Console.WriteLine("\n4º carácter:"); Console.Write(s[3]);
            Console.WriteLine("\nUpper:"); Console.Write(s.ToUpper());
            Console.WriteLine("\nMinúsculas:"); Console.Write(s.ToLower());
            Console.WriteLine("\nLimpa espaços:"); Console.Write(s.Trim());

            Console.WriteLine("\nContem 'gato' ?");
            if (s.Contains("gato")) Console.WriteLine("Sim."); else Console.WriteLine("Não");

            Console.WriteLine("\nSubstring:"); Console.Write(s.Substring(29));
            Console.WriteLine("\nReplace:"); Console.Write(s.Replace("gato", "tigre"));
            Console.WriteLine("\nRemove:"); Console.Write(s.Remove(6,4));
            Console.WriteLine("\nPosição do !:"); Console.Write(s.LastIndexOf('!'));
            Console.WriteLine("\nPosição de 'gato':"); Console.Write(s.LastIndexOf("gato"));

            Console.WriteLine("\nPrimeiro a dps de 'gato':");
            Console.WriteLine((s.Substring(s.LastIndexOf("2"))).LastIndexOf('a') );//incompleta
            Console.WriteLine("\n");
        }
    }
}
```

**Exercício extra – botões de minicalculadora**

Ex: digitar 100 --> clicar SQRT --> visor fica com 10

Ex: digitar 10 --> clicar + --> digitar 10 --> clicar = --> visor fica com 20

```
public Form1()
{
    InitializeComponent();
    visor.Text = "0";
    visor2.Text = "0"; //controlo oculto ...
}

...

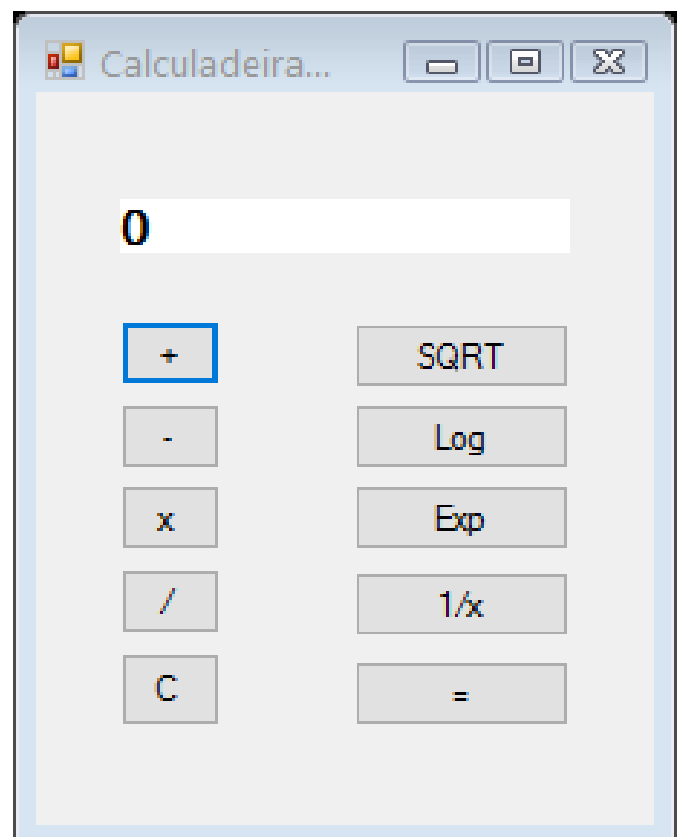
private void button6_Click(object sender, EventArgs e)
{
    double R, n;
    if (double.TryParse(visor.Text, out n))
    {
        R = Math.Round(Math.Sqrt(n), 2);
        //R = (Math.Sqrt(Convert.ToDouble(visor.Text)));
        visor.Text = Convert.ToString(R);
    }
}

...

private void button9_Click(object sender, EventArgs e)
{
    double R, n;
    if (double.TryParse(visor.Text, out n))
    {
        visor2.Text = Convert.ToString(n);
        R = 1/n;
        visor.Text = Convert.ToString(R);
    }
}

...

private void button10_Click(object sender,
EventArgs e)
{ //código exemplo para o "="
    double R, T, n;
    if (double.TryParse(visor.Text, out
n))
    {
        R = Convert.ToDouble(visor2.Text);
        R = R + n;
        visor.Text = Convert.ToString(R);
        visor2.Text = "0";
    }
}
```





Manipulação de listbox e array

The screenshot shows a Windows application window with a listbox containing 20 names. On the left, there are several buttons: "Adiciona string 'Xyz'", "Conta elementos", "Elimina primeiro", "Elimina último", "Add 4 nomes", "Elim. selecionado", and "Limpa todos". Below these buttons is a text input field containing "Bela Costa Silva" and a button labeled "Existe nome?". A small dialog box with the title "Existe" and an "OK" button is open over the "Existe nome?" button. At the bottom of the window, there are two more buttons: "Colonizar lista (inicializa vetor de strings e copia do vetor para a list manualmente, de vetor)" and "Colonizar lista (pelo método AddRange)".

Colonizar (declara vetor; inicializa vetor; copia do vetor p/ a lista:

```
string[] F = new string[21];  
F[1] = "Ana Rita Cunha";  
...  
F[20] = "Zacarias Alexandre Sampaio";  
listBox1.Items.Clear();  
for (int x = 1; x < 21; x++)  
    //copiar as strings para a lisbox  
    {  
        listBox1.Items.Add(F[x]);  
    }
```

Colonizar via AddRange:

```
string[] F = new string[21];  
F[0] = "---"; //porque o vetor tem a posição zero...  
//no caso anterior, podemos colocar o iterador a começar em 1; aqui não  
F[1] = "Ana Rita Cunha";  
...  
F[20] = "Zhacaryas Alexandre Sampaio";  
listBox1.Items.Clear();  
listBox1.Items.AddRange(F);
```

Remover a primeira posição da lista:

```
listBox1.Items.RemoveAt(0);
```

Mostrar o número de elementos da lista:

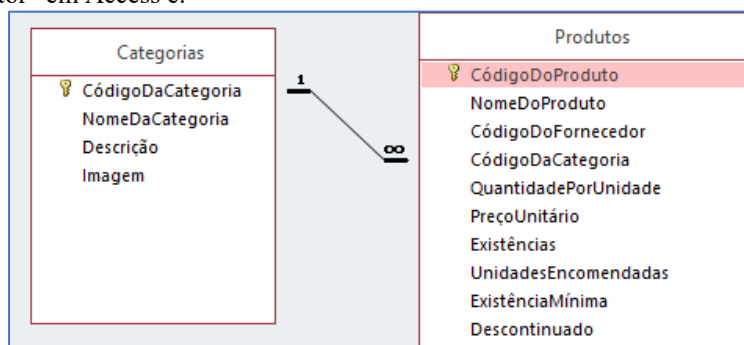
```
MessageBox.Show(Convert.ToString(listBox1.Items.Count));
```

**Remover o último da lista:**

```
int contador=listBox1.Items.Count;listBox1.Items.RemoveAt(contador-1);
```

SEGUNDA PARTE (COM BASE DE DADOS SQL SERVER)**Contexto e objetivo do problema proposto**

- Pretende-se elaborar uma pequena aplicação em C# que permita ligar a uma base de dados SQL Server
- a base de dados será criada e gerida através do SSOE (Sql Server Object Explorer, integrado no Visual Studio)
- a colonização (criação de “colónia” de dados) será feita com dados a partir do Excel
- o modelo de dados envolve 2 tabelas: categorias e produtos; o diagrama original, retirado da antiga base de dados “adamastor” em Access é:



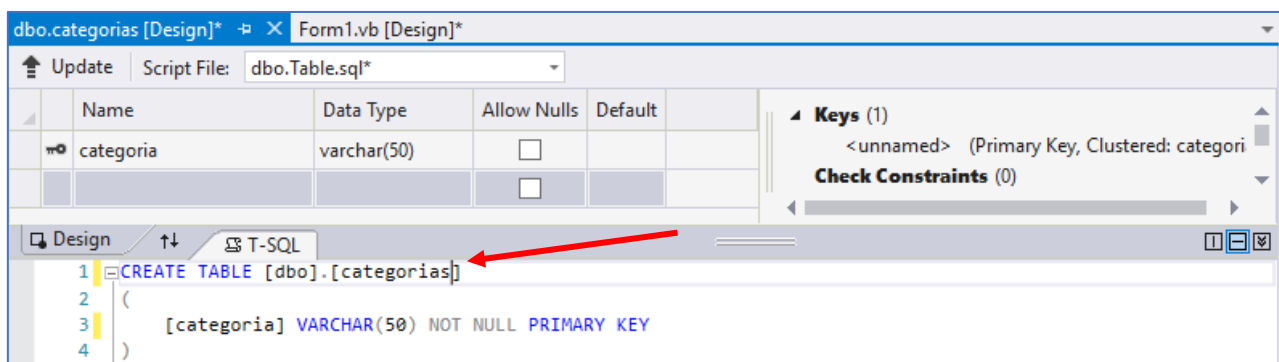
- para não sobrecarregar o trabalho, usar-se-á o seguinte conjunto de campos:
 - categorias: apenas o nome da categoria
 - produtos: código do produto | nome do produto | categoria | preço unitário
- a aplicação deverá permitir responder às seguintes tarefas
 - listar produtos, todos ou de uma certa categoria
 - contar produtos
 - encontrar o preço mais elevado
 - calcular a média dos preços dos produtos

Planeamento

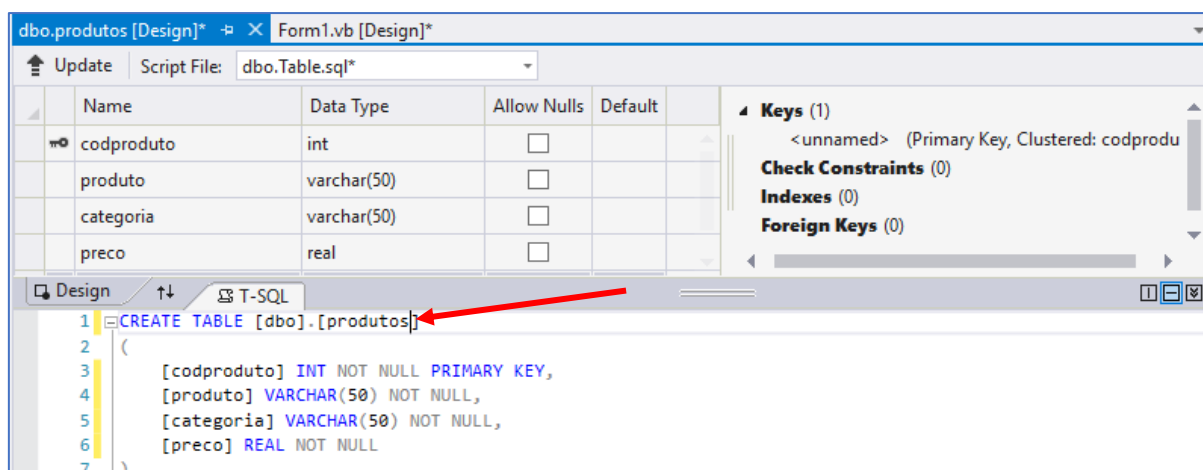
Uma forma de atacar o problema seria:

- definir
 - nome do projeto (Categorias-Produtos)
 - nome da pasta (Categorias-Produtos)
 - localização da pasta (c:\)
 - nome da base de dados (cats-prods-DB)
- criar a base de dados, usando o SSOE
- criar as 2 tabelas referidas (categorias e produtos)
- colonizar as 2 tabelas com dados a partir do Excel
- definir a string connection do projeto
- iniciar a construção do formulário

Criação da tabela de categorias:

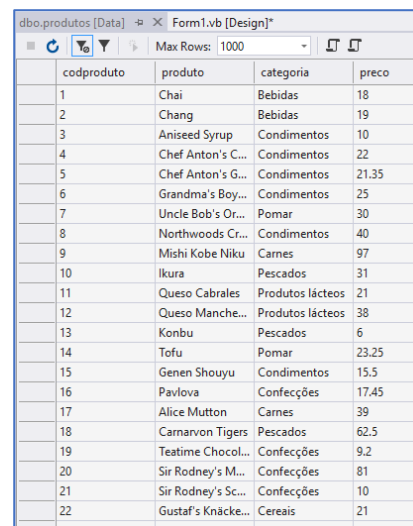
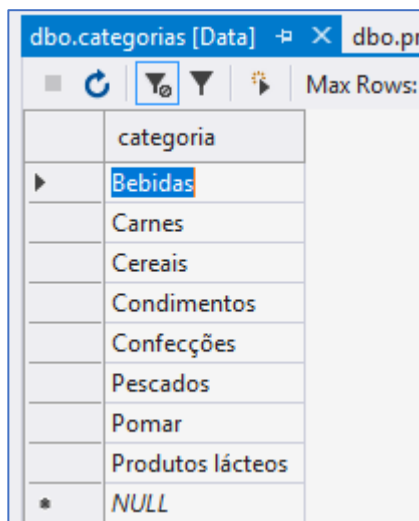
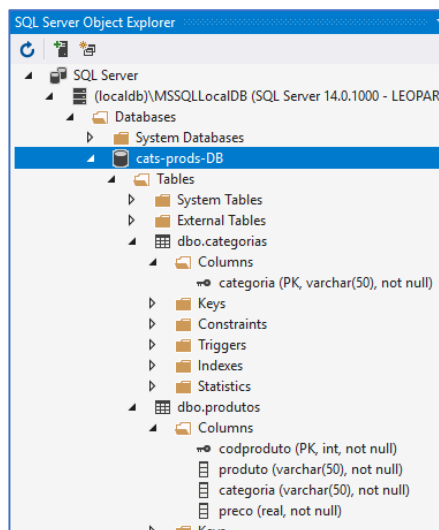


Criação da tabela de produtos:



Para colonizar uma tabela, bastará copiar os dados do ficheiro em Excel fornecido para a tabela pretendida. Opcionalmente, para ver como colonizar a tabela de produtos, pode ver um vídeo de 55 segundos (pasta online deste exercício).

Após a criação das tabelas e respetiva colonização, as 3 imagens seguintes deverão fazer sentido:



Analisar a connection string

No SSOE, nas propriedades da base de dados deste projeto, encontrará o seguinte (ou semelhante):

```
Connection string Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=cats-prods-DB;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite
```

Analizando a string connection, importa apenas entender o seguinte:

- a origem de dados é uma instância local ao seu computador (está a verde)
- a base de dados à qual se fará a ligação é a que foi criada anteriormente (a amarelo)



Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=cats-prods-DB;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False

De qualquer forma, o que interessa é “apontar” o projeto que vai ser desenvolvido para a base de dados pretendida, e isso pode ser feito definindo uma “variável global” na folha de código do form (classe Form1.cs)

O aspeto da classe referida seria o seguinte:

```
using System.Windows.Forms;
namespace Categorias_Produtos
{
    public partial class Form1 : Form
    {
        string SC = " Data Source= (localdb)\MSSQLLocalDB;Initial Cat..... RESTO DA STRING";

        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

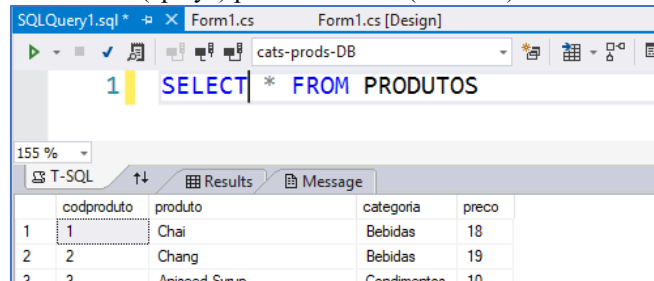
A execução do form não ainda não faz nada, mas as futuras operações que venham a ser implementadas com base de dados, serão especificamente na base de dados que foi criada inicialmente, porque a string connection foi “apontada” dessa forma. Com uma correta configuração da string connection poder-se-á apontar o projeto para uma base de dados num servidor de rede, ou até numa base de dados online.



Pequenas consultas em SQL

A linguagem SQL (Structured Query Language) permite ao utilizador escrever comandos de interrogação às tabelas, para obter informações, geralmente uma listagem, um total, uma contagem de registos, outros.

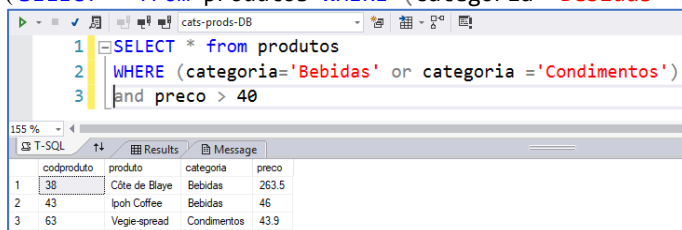
Para treinar algumas consultas, o botão direito do rato permite aceder à janela de novas consultas (new query). A imagem seguinte mostra isso; o botão verde (“play”) permite acionar (executar) a consulta:



Esta consulta mostrará 77 registos.

Nessa janela, coloque as seguintes interrogações à base de dados: (fazer, apagar, fazer a seguinte...)

1. listar todos os produtos (a da imagem acima) (`SELECT * FROM produtos`)
2. contar todos os produtos (`SELECT COUNT(*) FROM produtos`)
3. qual o preço mais elevado (`SELECT MAX(preco) FROM produtos`)
4. quantas categorias há? (`SELECT COUNT(*) FROM categorias`)
5. listar todas as bebidas (`SELECT * FROM produtos WHERE categoria = 'Bebidas'`)
6. qual a média de preços (`SELECT AVG(preco) FROM produtos`)
7. qual a média de preços das bebidas (`SELECT AVG(preco) FROM produtos WHERE categoria = 'Bebidas'`)
8. que produtos há nas categorias bebidas e condimentos, com preço acima de 40? (atenção aos parêntesis) (`SELECT * from produtos WHERE (categoria='Bebidas' or categoria ='Condimentos') and preco > 40`)



9. há produtos quem tenham o carácter y no nome? (`SELECT * from produtos WHERE produto LIKE '%y%'`)

A pergunta 8 poderá parecer manhosa, porque diz “bebidas e condimentos” mas na resposta está “categoria='Bebidas' or categoria = 'Condimentos'”; é mesmo assim; isto tem que ver com as pequenas diferenças entre a lógica da nossa língua natural e a álgebra de Boole que deverá ter estudado no 8º ano (Teoria de conjuntos, interseção, intervalos, pertence a, etc.).

Para a execução do resto do trabalho não precisa de entender todas as consultas; tente as seguintes: 1, 2, e 5.

A função Conecta ()

Esta função foi elaborada para facilitar a busca de listagens de dados às tabelas:

- usa a string connection definida no início, e que aponta para a base de dados **cats-prods-DB**
- assim, haverá acesso às 2 tabelas contidas nessa base de dados
- recebe pelo argumento o comando SQL pretendido (pode ser escrito diretamente no argumento da função ou atribuído a uma string, passando a string como argumento)
- se o comando SQL estiver correto, trará da base de dados a listagem pretendida sob a forma de uma tabela
- essa tabela poderá ser “despejada” para controlos do formulário: listboxes, comboboxes, gridviews, outros; para cada controlo será depois necessário afinar alguns parâmetros
- deverá ficar claro que o que se pretende é entender **como** utilizar esta ferramenta e não a ferramenta em si. Esta função pode ser utilizada noutros projetos, porque foi desenhada para ser genérica (depende apenas da string connection e da string SQL, portanto pode ser usada noutras bases de dados).

O código da função (pode ser copiado diretamente para o projeto) é:

```
DataTable Conecta(string str_sql)
```



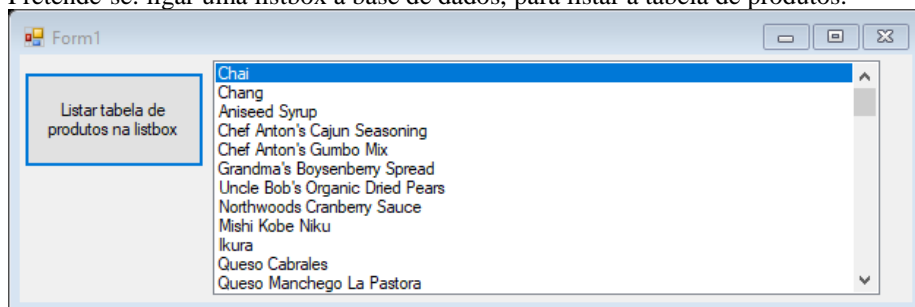
```
{  
    // usa a string de conexão à base de dados SC  
    // e recebe pelo argumento a string com o comando SQL;  
    // devolve uma tabela (datatable) com os registos provenientes da base de dados  
    SqlConnection C = new SqlConnection(SC); C.Open();  
    //criar comando SQL para extrair os dados pretendidos:  
    SqlCommand command = C.CreateCommand(); command.CommandText = str_sql;  
    //o comando SQL vem na string, via argumento da função  
    //trazer os dados da tabela especificada para uma "tabela" em memória:  
    SqlDataAdapter da = new SqlDataAdapter(command);  
    DataTable t = new DataTable();  
    da.Fill(t);  
    C.Close();  
    return t; //t leva os registos  
}
```

Também será necessário adicionar as 2 seguintes linhas:

```
using System.Data;  
using System.Data.SqlClient;  
using System.Windows.Forms; //esta linha já existe no projeto
```

Ligar uma listbox à tabela de produtos

Pretende-se: ligar uma listbox à base de dados, para listar a tabela de produtos.



```
private void button1_Click(object sender, System.EventArgs e)  
{  
    //Datatable é um tipo de dados, tal como int ou string;  
    //este tipo de dados permite trabalhar com tabelas:  
    DataTable tab_prods = Conecta("select * from produtos");  
    //a propriedade datasource da listbox recebe a tabela proveniente da função conecta:  
    listBox1.DataSource = tab_prods;  
    //a tabela recebida tem várias colunas; é preciso escolher qual a coluna a mostrar:  
    listBox1.DisplayMember = "produto";  
}
```

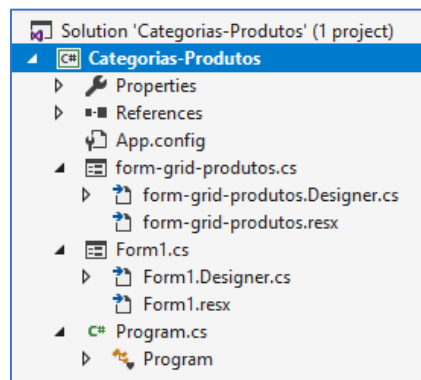
Adicionar um form novo

Para manter o form anterior inalterado, pode adicionar um novo form ao projeto.

Na janela Solution Explorer, nome do projeto, botão direito, Add → Windows Form. A designação do form é da responsabilidade do programador, o nome por omissão é neste caso Form2.cs.

O próximo form vai conter uma datagridview, assim, o form será designado "form-grid-produtos.cs".

Após a criação, o projeto passará a ter 2 forms, conforme a imagem.

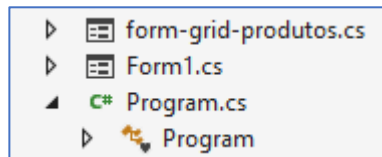




Definir qual o form que é executado

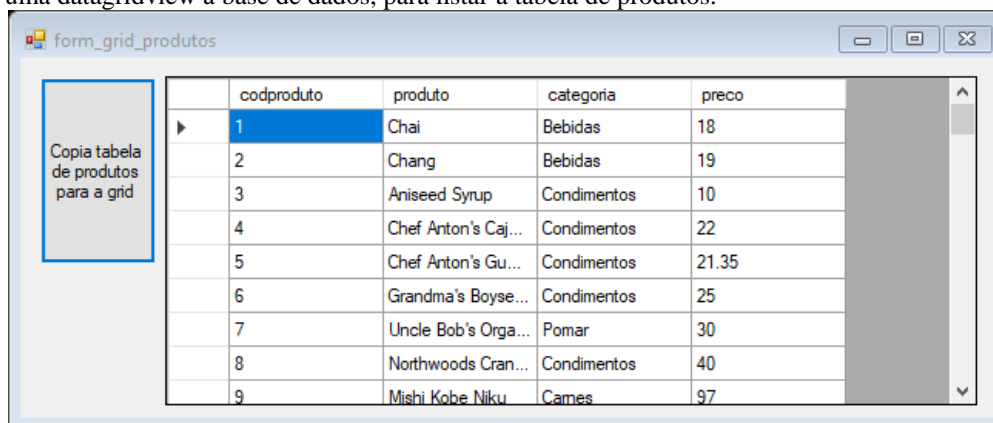
Para que o novo form seja o que é compilado e executado, podemos alterar o ficheiro “Program.cs”. Este ficheiro contém o método Main(), que “arranca” a aplicação; bastará trocar o form que vai ser invocado. No lugar de “Form1”, coloque o novo form, conforme a ilustração ao lado. O ficheiro Program.cs pode ser aberto na janela Solution Explorer.

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new form_grid_produtos());
}
```



Ligar uma gridview à tabela de produtos no novo form

Pretende-se: ligar uma datagridview à base de dados, para listar a tabela de produtos.



```
private void button1_Click(object sender, EventArgs e)
{
    //Datatable é um tipo de dados, tal como int ou string;
    //este tipo de dados permite trabalhar com tabelas:
    DataTable tab_prods = Conecta("select * from produtos");
    //a propriedade datasource da grid recebe a tabela proveniente da função conecta:
    dataGridView1.DataSource = tab_prods;
    //a tabela recebida tem várias colunas e é despejada na grid,
    //sem necessidade de qualquer configuração
}
```

SQL – treinar consultas ativas

Antes de prosseguir com o projeto, será necessário experimentar algumas consultas ativas, ou seja, que provocam alteração nas tabelas. Note-se que as consultas anteriores são todas passivas, ou seja, “buscam” dados mas não alteram nada.

Usar-se-á a base de dados cats-prods-DB, tabela categorias.
No SSQE, base de dados do projeto, botão direito, “New Query”...

1. Listar as categorias (passiva) - `select * from categorias`
2. Contar as categorias (passiva) - `select COUNT(*) from categorias`
3. Inserir a categoria “EXPLOSIVOS”
`INSERT INTO categorias values ('EXPLOSIVOS')`
4. Trocar nome da categoria de “EXPLOSIVOS” para “Doces”
`update categorias set categoria = 'Doces' Where categoria = 'EXPLOSIVOS'`
5. Apagar a categoria “Doces”
`delete from categorias where categoria='Doces'`

Estas consultas são suficientes para implementar um mecanismo de manipulação do conteúdo de tabela, usando C#.



Layout de form pretendido para inserir, remover, alterar registo em tabela

Elabore o formulário e implemente os métodos que limpam a grid e a inicializam. O objetivo seguinte é implementar a inserção, alteração e eliminação de registo.

Limpar e preencher a grid

Primeiro método:

```
//remover a conexão da origem de dados (limpa a grid):  
dataGridView1.DataSource = null;
```

Segundo método:

```
//preenche a grid com a tabela categorias, recorrendo à função Conecta  
dataGridView1.DataSource = Conecta("select * from categorias");
```

Apagar um registo

```
private void bt_delete_registo_Click(object sender, EventArgs e)  
{  
    string ssql = "";  
    //busca a categoria que está na textbox  
    ssql = "delete from categorias where categoria= '" + textBox1.Text + "'";  
  
    //neste caso, a função conecta retorna uma tabela vazia, mas  
    //executa na mesma a consulta pretendida  
    DataTable t;  
    t = Conecta(ssql);  
    //se a categoria existir, já não aparecerá ao refrescar a grid  
}
```

Alterar um registo existente

Para esta implementação, será necessário pedir 2 nomes de categoria ao utilizador (o antigo e o novo); será necessário adicionar outra textbox ao form.

Na imagem, o update (Bebidas → Bebidas BRANCAS).

```
private void bt_update_registo_Click(object sender, EventArgs e)
```



```
{  
    string ssql = "";  
    //busca a categoria que está nas textboxes: a existente e a nova  
    ssql = "update categorias set categoria = '" + textBox2.Text + "'" +  
        " where categoria = '" + textBox1.Text + "'";  
  
    //neste caso, a função conecta retorna uma tabela vazia, mas  
    //executa na mesma a consulta pretendida  
    DataTable t;  
    t = Conecta(ssql);  
}
```

Inserir um registo

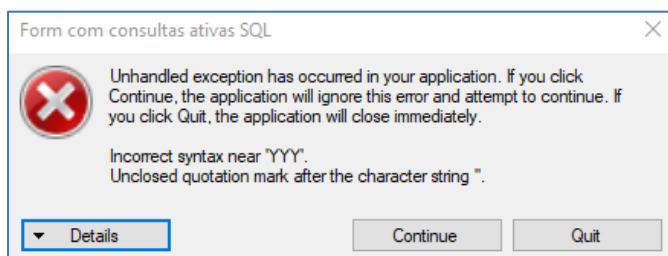
```
private void bt_insert_registo_Click(object sender, EventArgs e)  
{  
    string ssql = "";  
    //busca a categoria que está na textbox  
    ssql = "INSERT INTO categorias values ('" + textBox1.Text + "')";  
    //neste caso, a função conecta retorna uma tabela vazia, mas  
    //executa na mesma a consulta pretendida  
    DataTable t;  
    t = Conecta(ssql);  
}
```

Porquê usar a MessageBox.Show com a string SQL?

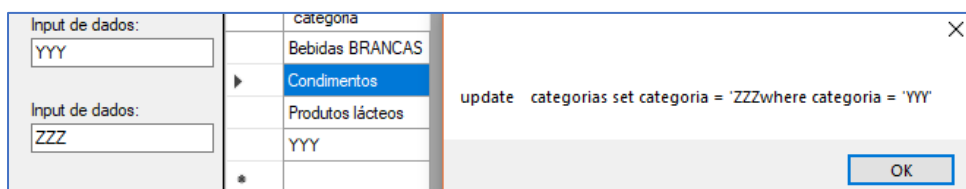
```
ssql = "update categorias set categoria = '" + textBox2.Text + "where categoria = '" + textBox1.Text + "'";
```

Analise a string mostrada acima.

A execução do form levou à geração de uma exceção:



Se a visualizar numa MessageBox, com a instrução "MessageBox.Show(ssql);" verá que é muito mais fácil ver o erro nesta janela do que no código. Neste caso, falta uma plica antes da cláusula (palavra) Where.



Na elaboração de consultas de inserção com várias colunas, a colocação das plicas e aspas torna-se trabalhosa. A utilização destas Message.Boxes, pode ser uma mais-valia.

Contar registos da tabela (consulta SQL de agregação)

Será necessário adicionar novo botão e textbox.



```
private void bt_contar_registos_Click(object sender, EventArgs e)
{
    string ssql = "select count (*) from categorias";
    DataTable t = Conecta(ssql);
    //a consulta é de agregação; dela resulta um valor e não uma tabela;
    //no entanto, um valor é um caso particular de uma tabela, ou seja,
    //é uma tabela com apenas uma linha e uma coluna; isso permite converter
    //t para um inteiro ou, neste caso, para uma string:
    textBox3.Text = Convert.ToString(t.Rows[0][0]);
}
```

Desta forma foi possível utilizar a função Conecta para 3 tipos de consultas:

- SELECT ..., consultas passivas que retornam uma listagem
- INSERT, UPDATE, DELETE, que são consultas ativas mas não retornam listagem
- SELECT COUNT, exemplo de consulta de agregação que retornam um valor

Fechar aplicação

Para o segundo botão, o código:

```
//fecha a aplicação; é desnecessário porque
//os forms podem ser fechados no canto superior direito...
Application.Exit();
```

Invocar outro form

Para o primeiro botão, e supondo que foi criado um formulário designado Form4 (classe Form4.cs):

```
//mecanismo de instanciação;
//f é um objeto da classe Form4
Form4 f = new Form4();
f.Show();
```

Alternativamente a instrução `f.ShowDialog();` abre o mesmo form, mas o foco fica “preso”, ou seja, enquanto não fechar o form que foi aberto, não pode voltar ao anterior.

Inserir um produto (4 campos)

Como exercício final, vamos inserir uma ficha de produto. O problema vai estar na string em sql, por causa da sequência de plicas, aspas e outros caracteres.

A estrutura da string SQL será semelhante ao seguinte:

```
string ssql = "insert into produtos values (' ', ' ', ' ', ' ')";
```



Inserir registo na tabela de produtos

999

PRODUTO NOVO

Pomar

44

A string seguinte tem a estrutura da anterior, apesar de não o parecer:

```
string ssq1 =  
    "insert into produtos values ('"+  
    Convert.ToInt32(txt1.Text)+ "','"+  
    txt2.Text+"', '"+  
    txt3.Text+"', '"+  
    Convert.ToDouble(txt4.Text) + "');" ;
```

Cada txt é uma textbox do form. De notar que a primeira precisa da conversão para inteiro, e a quarta para double, por causa da compatibilidade com as colunas da base de dados que vão receber estes valores.

Em qualquer caso, é bom utilizar a `MessageBox.Show` para ir visualizando a string, porque a probabilidade de correr bem à primeira é reduzida:

×

insert into produtos values ('999','PRODUTO NOVO','Pomar','44')

OK

Após a operação correr bem, podemos confirmar no SSQE que a inserção correu bem:

ID	Nome do Produto	Descrição	Preço
77	Original Frankfurter grüne Soße	Condimentos	13
999	PRODUTO NOVO	Pomar	44
*	NULL	NULL	NULL



Estrutura da classe

Se analisar a classe que suporta este form, veremos uma estrutura semelhante a:

```
public partial class form_SQL_insert_outros : Form
{
    // "variável pública" com string de conexão, para ligar à base de dados
    string SC = "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=cats-prod";

    // função que interpreta consultas em SQL e traz os resultados numa tabela
    DataTable Conecta(string str_sql) {...}

    // construtor: método que tem o mesmo nome da classe, executado ao abrir o form
    // e contém a inicialização dos controlos:
    public form_SQL_insert_outros() {...}

    // método associado ao botão...
    private void bt_limpa_grid_Click(object sender, EventArgs e) {...}
    // método associado ao botão...
    private void bt_atualiza_grid_Click(object sender, EventArgs e) {...}
    // método associado ao botão...
    private void bt_delete_registo_Click(object sender, EventArgs e) {...}
    // método associado ao botão...
    private void bt_update_registo_Click(object sender, EventArgs e) {...}
    // método associado ao botão...
    private void bt_insert_registo_Click(object sender, EventArgs e) {...}
    // método associado ao botão...
    private void bt_contar_registos_Click(object sender, EventArgs e) {...}
}
```

Conclusão

Ao olhar para a classe, consegue entender os termos método, evento, controlo, construtor? São palavras-chave associadas à Programação orientada ao objeto. Todo o código que escreveu está obrigatoriamente dentro de uma classe. Foi possível iniciar a escrita de código orientado ao objeto, mesmo sem ter consciência real desse facto. O menino que escreve a frase “Eu gosto de jogar à bola”, pode escrevê-la e entendê-la corretamente, sem ter consciência que as frases têm sujeito, predicado, complementos, etc. Claro que mais tarde... terá que estudar gramática para progredir. O tempo foi pouco...